Signal Processing for Neuroscientists

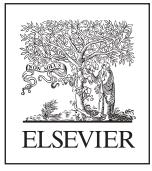
Second Edition

Wim van Drongelen

SIGNAL PROCESSING FOR NEUROSCIENTISTS

SECOND EDITION

WIM VAN DRONGELEN
Professor of Pediatrics, Neurology, and Computational Neuroscience,
Technical Director Pediatric Epilepsy Center Research Director Pediatric Epilepsy Program
Senior Fellow Computation Institute, The University of Chicago, Chicago, IL, USA



ACADEMIC PRESS
An imprint of Elsevier

Academic Press is an imprint of Elsevier
125 London Wall, London EC2Y 5AS, United Kingdom
525 B Street, Suite 1800, San Diego, CA 92101-4495, United States
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, United Kingdom

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the Library of Congress

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

ISBN: 978-0-12-810482-8

Publisher: Nikki Levy
Acquisition Editor: Natalie Farra
Editorial Project Manager: Kristi Anderson
Production Project Manager: Poulouse Joseph
Designer: Greg Harris

Typeset by TNQ Books and Journals

To Sebastiaan, Olivia, Simonne, Sofia, Micah, Solomon and Malachi

Preface to the Second Edition

This second edition of "Signal Processing for Neuroscientists" includes the partly revised and updated material of the reprinted first edition and its companion volume. At the request of my students, colleagues, and readers, I have extended several chapters and added new ones. As a result, this edition contains a few additional advanced topics (the imaging section of Chapters 7, 8, and 13), an introduction to differential equations and modeling dynamics (Chapters 9, 10, and 23), and an overview of modeling activity of single neurons and neuronal networks (Chapters 29 and 30). These last two chapters summarize approaches to modeling of neural activity with a flavor of a historical overview of modeling in neuroscience. Due to the review aspect of the modeling component, these chapters include a much longer reference list than the others. At multiple requests I have also included a series of exercises with each chapter. Answers to a subset of these exercises are available on http://booksite.elsevier.com/9780128104828/.

Overall, the end result of this second edition is fairly close to what I had initially hoped to produce for the first edition about a decade ago—but couldn't finish within the available time. However, although this text appears much later than I had envisioned, I now have the additional advantage of having taught the signal analysis and modeling topics multiple times to different groups of students, giving me a much better insight into how to present and explain the material. In addition, the current text, figures, and exercises were extensively evaluated and tested by students and readers. Three years ago, some of the lectures were made available on YouTube for a worldwide audience, and this triggered useful feedback and suggestions that were included in this edition. The overall goal of the text remains to present the mathematical background of different types of signal analysis and modeling in neuroscience, and to illustrate this with practical examples in the form of MATLAB® scripts that are available on http://booksite.elsevier.com/9780128104828/. Ultimately I hope this end result helps the reader when studying the material in this text.

I want to include a few suggestions for those who plan to use this text and/or its figures (http://booksite.elsevier.com/9780128104828/) in their lectures. Most chapters are based on material and examples that can be covered in a 1–1½ h lecture. However, depending on the background of the students, Chapters 9, 10, 15, 16, 21 and 22 may be

combined in single 1½ h lectures, while Chapters 7 and 13, as well as the overview in Chapter 30, are more suitable for multiple lectures of that length. In addition to the MATLAB® material, some chapters also include elements that can be used for lab sessions. A filter lab can be based on the RC-circuit described in Chapter 15. For setting up more advanced lab sessions, I describe membrane models using the integrate-and-fire circuit and the Nagumo equivalent circuit in Chapter 29 and Appendix 29.2.

It is difficult to thank everyone who contributed directly or indirectly to this book, but I will try (with an apology to those I forget to include). In addition to all I already acknowledged in the previous edition and its companion volume, I want to thank my collaborators at the University of Chicago and elsewhere: Drs. Michel J.A.M. van Putten, Stephan A. van Gils, Catherine A. Schevon, Andrew K. Tryba, Charles J. Marcuccilli, and Jan-Marino (Nino) Ramirez. I also thank Drs. Steven Small, Ahmed Shereen, and Justin Jureller for their critical input while writing the material on imaging. Special thanks to Dr. Jack D. Cowan for his inspiring guest lectures on the Wilson–Cowan equations, and who (in the many conversations we had) made me appreciate the population model approach in neuroscience. I also thank my Teaching Assistants Alex Sadovsky, Arup Sarma, Kyler Brown, Albert Wildeman, James Goodman, Graham Fetterman, and Joe Dechery for their useful suggestions. The feedback of the (under)graduate students, Mark Saddler, Tahra Eissa, Jyothsna Suresh, Albert Wildeman, and Jeremy Neuman, in my laboratory was very helpful. I also thank the people at Elsevier, Dr. Natalie Farra, Kristi Anderson, and Poulouse Joseph for their support and suggestions. Last but not least I thank Ingrid for her ongoing fabulous support!

Preface to the Companion Volume

This text is based on a course I teach at the University of Chicago for students in Computational Neuroscience. It is a continuation of the previously published text *Signal Processing for Neuroscientists: An Introduction to the Analysis of Physiological Signals* and includes some of the more advanced topics of linear and nonlinear systems analysis and multichannel analysis. In the following, it is assumed that the reader is familiar with the basic concepts that are covered in the introductory text and, to help the student, multiple references to the basics are included.

The popularity of signal processing in neuroscience is increasing, and with the current availability and development of computer hardware and software it may be anticipated that the current growth will continue. Because electrode fabrication has improved and measurement equipment is getting less expensive, electrophysiological measurements with large numbers of channels are now very common. In addition, neuroscience has entered the age of light, and fluorescence measurements are fully integrated into the researcher's toolkit. Because each image in a movie contains multiple pixels, these measurements are multichannel by nature. Furthermore, the availability of both generic and specialized software packages for data analysis has altered the neuroscientist's attitude toward some of the more complex analysis techniques. Interestingly, the increased accessibility of hardware and software may lead to a rediscovery of analysis procedures that were initially described decades ago. At the time when these procedures were developed, only few researchers had access to the required instrumentation, but now most scientists can access both the necessary equipment and modern computer hardware and software to perform complex experiments and analyses.

The considerations given above have provided a strong motivation for the development of this text, where we discuss several advanced techniques, rediscover methods to describe nonlinear systems, and examine the analysis of multichannel recordings. The first chapter describes two very specialized algorithms: Lomb's algorithm to analyze unevenly sampled data sets and the Hilbert transform to detect

instantaneous phase and amplitude of a signal. The remainder of the text can be divided into two main components: (1) modeling systems (Chapter 2) and the analysis of nonlinear systems with the Volterra and Wiener series (Chapters 3–5) and (2) the analysis of multichannel measurements using a statistical approach (Chapter 6) and examination of causal relationships (Chapter 7). Throughout this text, we adopt an informal approach to the development of algorithms and we include practical examples implemented in MATLAB®. (All the MATLAB® scripts used in this text can be obtained via http://www.elsevierdirect.com/companions/9780123849151.)

It is a pleasure to acknowledge those who have assisted (directly and indirectly) in the preparation of this text: Drs. V.L. Towle, P.S. Ulinski, D. Margoliash, H.C. Lee, M.H. Kohrman, P. Adret, and N. Hatsopoulos. I also thank the teaching assistants for their help in the course and in the development of the material in this text: thanks, Matt Green, Peter Kruskal, Chris Rishel, and Jared Ostmeyer. There is a strong coupling between my teaching efforts and research interests. Therefore, I am indebted to the Dr. Ralph and Marian Falk Medical Research Trust for supporting my research and to the graduate and undergraduate students in my laboratory: Jen Dwyer, Marc Benayoun, Amber Martell, Mukta Vaidya, and Valeriya Talovikova. They provided useful feedback, tested some of the algorithms, and collected several example data sets. Special thanks to the group of students in the 2010 winter class who helped me with reviewing this material: Matt Best, Kevin Brown, Jonathan Jui, Matt Kearney, Lane McIntosh, Jillian McKee, Leo Olmedo, Alex Rajan, Alex Sadovsky, Honi Sanders, Valeriya Talovikova, Kelsey Tupper, and Richard Williams. Their multiple suggestions and critical review helped to significantly improve the text and some of the figures. At Elsevier I want to thank Lisa Tickner, Clare Caruana, Lisa Jones, Mani Prabakaran, and Johannes Menzel for their help and advice. Last but not least, thanks to my wife Ingrid for everything and supporting the multiple vacation days used for writing.

Preface to the First Edition

This textbook is an introduction to signal processing primarily aimed at neuroscientists and biomedical engineers. The text was developed for a one-quarter course I teach for graduate and undergraduate students at the University of Chicago and the Illinois Institute of Technology. The purpose of the course is to introduce signal analysis to students with a reasonable but modest background in mathematics (including complex algebra, basic calculus, and introductory knowledge of differential equations) and a minimal background in neurophysiology, physics, and computer programming. To help the basic neuroscientist ease into the mathematics, the first chapters are developed in small steps, and many notes are added to support the explanations. Throughout the text, advanced concepts are introduced where needed, and in the cases where details would distract too much from the "big picture," further explanation is moved to an appendix. My goals are to provide students with the background required to understand the principles of commercially available analyses software, to allow them to construct their own analysis tools in an environment such as MATLAB®,[1] and to make more advanced engineering literature accessible. Most of the chapters are based on 90-min lectures that include demonstrations of MATLAB® scripts. Chapters 7 and 8 contain material from three to four lectures. Each chapter can be considered as a stand-alone unit. For students who need to refresh their memory on supporting topics, I include references to other chapters. The figures, equations, and appendices are also referenced independently by chapter number.

The CD that accompanies this text contains the MATLAB® scripts and several data files. These scripts were not developed to provide optimized algorithms but serve as examples of implementation of the signal processing task at hand. For ease of interpretation, all MATLAB® scripts are commented; comments starting with % provide structure and explanation of procedures and the meaning of variables. To gain practical experience in signal processing, I advise the student to actively explore the examples and scripts included and worry about algorithm optimization later. All scripts were developed to run in MATLAB® (Version 7) including the toolboxes for signal processing (Version 6), image processing (Version 5), and wavelets (Version 3). However, aside from those that use a digital filter, the Fourier slice theorem, or the wavemenu, most scripts will run without

[1]MATLAB® is a registered trademark of The MathWorks, Inc.

these toolboxes. If the student has access to an oscilloscope and function generator, the analog filter section (Chapter 10) can be used in a lab context. The components required to create the RC circuit can be obtained from any electronics store.

I want to thank Drs. V.L. Towle, P.S. Ulinski, D. Margoliash, H.C. Lee, and K.E. Hecox for their support and valuable suggestions. Michael Carroll was a great help as TA in the course. Michael also worked on the original text in Denglish, and I would like to thank him for all his help and for significantly improving the text. Also, I want to thank my students for their continuing enthusiasm, discussion, and useful suggestions. Special thanks to Jen Dwyer (student) for her suggestions on improving the text and explanations. Thanks to the people at Elsevier, Johannes Menzel (senior publishing editor), Carl M. Soares (project manager), and Phil Carpenter (developmental editor), for their feedback and help with the manuscript.

Finally, although she isn't very much interested in signal processing, I dedicate this book to my wife for her support: heel erg bedankt Ingrid.

CHAPTER

1

Introduction

1.1 OVERVIEW

Signal processing in neuroscience and neural engineering includes a wide variety of algorithms applied to measurements such as a one-dimensional time series or multidimensional data sets such as a series of images. Although analog circuitry is capable of performing many types of signal processing, the development of digital technology has greatly enhanced the access to and application of signal processing techniques. Generally, the goal of signal processing is to enhance signal components in noisy measurements or to transform measured data sets such that new features become visible. Other specific applications include characterization of a system by its input–output relationships, data compression, or prediction of future values of the signal.

This text will introduce the whole spectrum of signal analysis: from data acquisition (Chapter 2) to data processing; and from the mathematical background of the analysis to the implementation and application of processing algorithms. Overall, our approach to the mathematics will be informal, and we will therefore focus on a basic understanding of the methods and their interrelationships rather than detailed proofs or derivations. Generally, we will take an optimistic approach, assuming implicitly that our functions or signal epochs are linear, stationary, show finite energy, have existing integrals and derivatives, etc.

Noise plays an important role in signal processing in general; therefore we will discuss some of its major properties (Chapter 3). The core of this text will focus on what can be considered the "***golden trio***" in the signal processing field:

1. **Averaging** (Chapter 4);
2. **Fourier analysis** (Chapters 5–8);
3. **Filtering** (Chapters 15–19).

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00001-1

Most current techniques in signal processing have been developed with linear time-invariant (LTI) systems as the underlying signal generator or analysis module (Chapters 9–14). Because we are primarily interested in the nervous system, which is often more complicated than an LTI system, we will extend the basic topics with an introduction into the analysis of time series of neuronal activity (***spike trains***, Chapter 20), analysis of nonstationary behavior (***wavelet analysis***, Chapters 21 and 22), modeling and characterization of time series originating from ***nonlinear systems*** (Chapters 23–27), ***decomposition of multichannel data*** (Chapter 28), and finally how to apply this to ***neural systems*** (Chapters 29 and 30).

1.2 BIOMEDICAL SIGNALS

Due to the development of a vast array of electronic measurement equipment, a rich variety of biomedical signals exist, ranging from measurements of molecular activity in cell membranes to recordings of animal behavior. The first link in the biomedical measurement chain is typically a ***transducer*** or ***sensor***, which measures signals (such as a heart valve sound, blood pressure, or X-ray absorption) and makes these signals available in an electronic format. Biopotentials represent a large subset of such biomedical signals that can be directly measured electrically using an ***electrode*** pair. Some such electrical signals occur "spontaneously" (e.g., the electroencephalogram, EEG); others can be observed upon stimulation (e.g., evoked potentials, EPs).

1.3 BIOPOTENTIALS

Biopotentials originate within biological tissue as potential differences that occur between compartments. Generally, the compartments are separated by a (bio)membrane that maintains concentration gradients of certain ions via an active mechanism (e.g., the Na^+/K^+ pump). Hodgkin and Huxley (1952) were the first to model a biopotential (the action potential in the squid giant axon) with an electronic equivalent (Fig. 1.1). A combination of ordinary differential equations (ODEs) and a model describing the nonlinear behavior of ionic conductances in the axonal membrane generated an almost perfect description of their measurements. The physical laws used to derive the base ODE for the equivalent circuit are: ***Nernst***, ***Kirchhoff***, and ***Ohm's laws*** (Appendix 1.1). An example of how to derive the differential

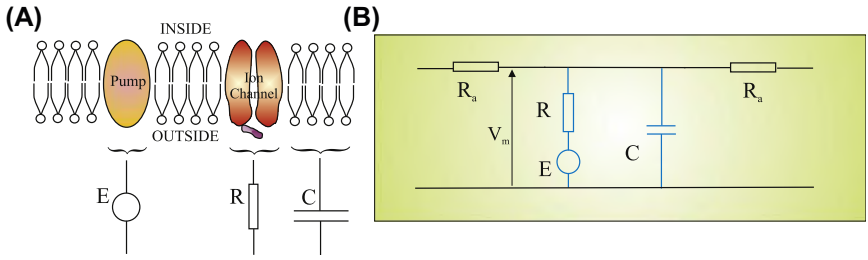


FIGURE 1.1 The origin of biopotentials. Simplified representation of the model described by Hodgkin and Huxley (1952). (A) The membrane consists of a double layer of phospholipids in which different structures are embedded. The ion-pumps maintain gradient differences for certain ion species, causing a potential difference (E). The elements of the biological membrane can be represented by passive electrical elements: a capacitor (C) for the phospholipid bilayer and a resistor (R) for the ion channels. (B) In this way, a segment of membrane can be modeled by a circuit including these elements coupled to other contiguous compartments via an axial resistance (R_a).

equation for a single ion channel in the membrane model is given in Chapter 13 (Fig. 13.2) and a more complete analysis of an extended version of the circuitry is described in Chapter 29 (Fig. 29.1).

1.4 EXAMPLES OF BIOMEDICAL SIGNALS

1.4.1 Electroencephalogram/Electrocorticogram and Evoked Potentials

The EEG represents overall brain activity recorded from pairs of electrodes on the scalp. In clinical neurophysiology the electrodes are placed according to an international standard (the 10–20 system or its extended version, the 10–10 system shown in Fig. 1.2A). In special cases, brain activity may also be directly measured via electrodes on the cortical surface (the electrocorticogram, Fig. 1.2B) or via depth electrodes implanted in the brain. Both EEG from the scalp and intracranial signals are evaluated for so-called foreground patterns (e.g., epileptic spikes) and ongoing background activity. This background activity is typically characterized by the power of the signal within different frequency bands:

delta rhythm (δ): 0–4 Hz,
theta rhythm (θ): 4–8 Hz,
alpha rhythm (α): 8–12 Hz,
beta rhythm (β): 12–30 Hz, and
gamma rhythm (γ): the higher EEG frequencies, usually 30–70 Hz.

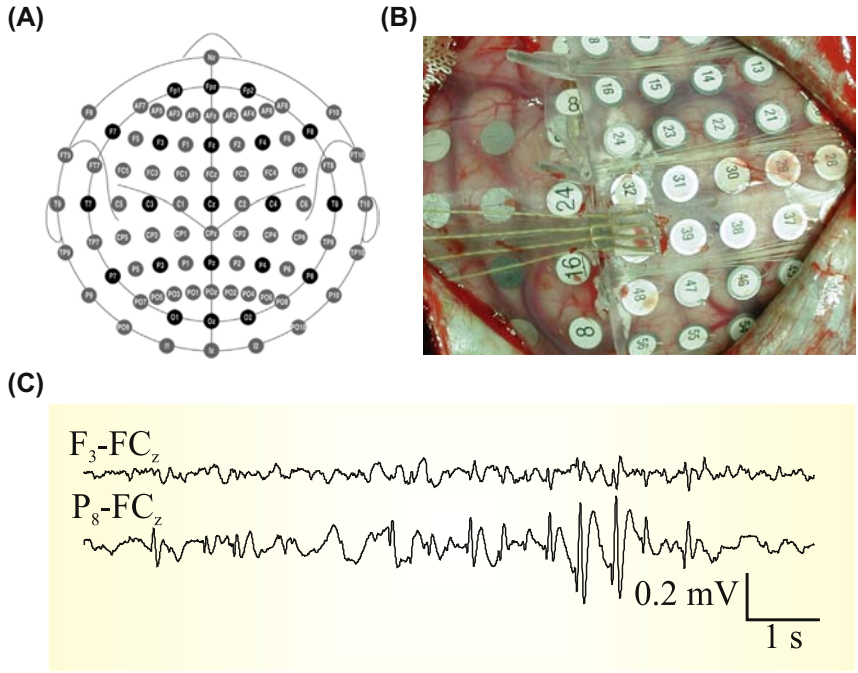


FIGURE 1.2 (A) An overview of the standard electroencephalogram (EEG) electrode positions. *Black circles* indicate positions of the original 10–20 system, and the additional positions of the 10–10 extension are indicated by *gray circles*. The diagram includes the standard electrode labels: Fp, prefrontal; F, frontal; C, central; P, parietal; O, occipital; and T, temporal (intermediate positions indicated as gray dots: AF, FC, CP, PO). Even numbers are on the right side (e.g., C_4) and odd numbers on the left side (e.g., C_3); larger numbers are farther from the midline. Midline electrodes are coded as z-zero positions (e.g., C_z). (B) An example of surgically placed cortical electrodes in a patient with epilepsy. In this application the electrode placement is determined by the location of the epileptic focus. (C) An example of two EEG traces recorded from the human scalp, including a burst of epileptiform activity with larger amplitudes on the posterior-right side (P_8-FC_z, representing the subtraction of the FC_z signal from the P_8 signal) as compared to the frontal-left side (F_3-FC_z). The signals represent scalp potential plotted versus time. The total epoch is 10 s. *(A) From Oostenveld, R., Praamstra, P., 2001. The five percent electrode system for high-resolution EEG and ERP measurements. Clin. Neurophysiol. 112, 713–719.*

Some of the higher EEG frequency components that are not routinely considered in clinical EEG review, are ω (60–120 Hz), ρ (120–500 Hz), and σ (500–1000 Hz). It should be noted that differences between the above EEG band names and specifications exist across different authors, and that the symbols ρ and σ are also used for different EEG phenomena to characterize sleep records.

Other common classes of neurophysiological signals used for clinical tests are auditory-, visual-, and somatosensory-evoked potentials

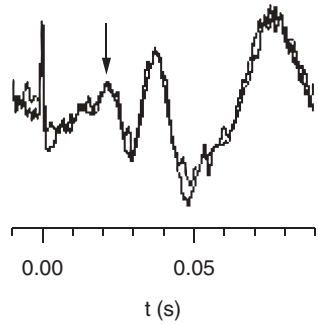


FIGURE 1.3 A somatosensory-evoked potential recorded from the human scalp as the average result of 500 electrical stimulations of the left radial nerve at the wrist. The stimulus artifact (at time 0.00) shows the time of stimulation. Two measurements are superimposed to show reproducibility. The *arrow* indicates the N20 peak at ~20 ms latency. *From Spiegel, J., Hansen, C., Baumgärtner, U., Hopf, H.C., Treede, R.-D., 2003. Sensitivity of laser-evoked potentials versus somatosensory evoked potentials in patients with multiple sclerosis. Clin. Neurophysiol. 114, 992–1002.*

(AEPs, VEPs, and SSEPs, respectively). These signals represent the brain's response to a standard stimulus such as a tone burst, click, light flash, change of a visual pattern, or an electrical pulse delivered to a nerve. When the brain responds to specific stimuli, the evoked electrical response is usually more than 10 times smaller than the ongoing EEG background activity. Signal averaging (Chapter 4) is commonly applied to make the brain's evoked activity visible. An example of an averaged SSEP is shown in Fig. 1.3. The averaging approach takes advantage of the fact that the response is time-locked with the stimulus, whereas the ongoing EEG background is not temporally related to the stimulus.

1.4.2 Electrocardiogram

The activity of the heart is associated with a highly organized muscle contraction preceded by a wave of electrical activity. Normally, one cycle of depolarization starts at the sino-atrial node and then moves as a wave through the atrium to the atrio-ventricular node, the bundle of His, and the remainder of the ventricles. This activation is followed by a repolarization phase. Due to the synchronization of the individual cellular activity, the electrical field generated by the heart is so strong that the electrocardiogram (ECG; though sometimes the German abbreviation "EKG " [elektrokardiogram] is used) can be measured from almost anywhere on the body. The ECG is usually characterized by several peaks, denoted P-QRS-T (Fig. 1.4B). The P-wave is associated with the activation of the atrium, the QRS-complex and the T-wave with

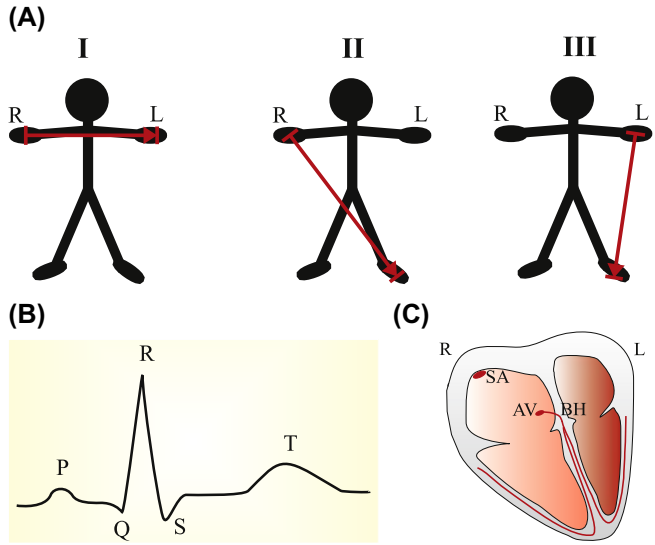


FIGURE 1.4 Einthoven's methods for recording the electrocardiogram (ECG) from the extremities. (A) The three directions (indicated as I, II, and III) capture different components of the ECG. R and L indicate right and left. (B) The normal ECG waveform is characterized by P, Q, R, S, and T peaks. (C) The electric activity starts at the top of the heart (sino-atrial [SA] node) and spreads down via the atrioventricular (AV) node and the bundle of His (BH).

ventricular depolarization and repolarization, respectively. In clinical measurements, the ECG signals are labeled with the positions on the body from which each signal is recorded. An example of Einthoven's I, II, and III positions as shown in Fig. 1.4A.

1.4.3 Action Potentials

The activity of single neurons can be recorded using microelectrodes with tip diameters around 1 μm. If both recording electrodes are outside the cell, one can record the extracellular currents associated with the action potentials. These so-called extracellular recordings of multiple neuronal action potentials are also referred to as spike trains. Alternately, if one electrode of the recording pair is inside the neuron, one can directly measure the membrane potential of that cell (Fig 1.5). Action potentials are obvious in these intracellular recordings as large stereotypical depolarizations in the membrane potential. In addition, intracellular recordings can reveal much smaller fluctuations in potential that are generated at synapses.

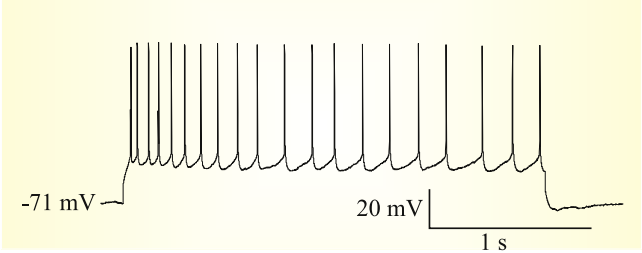


FIGURE 1.5 Action potentials from a neocortical neuron evoked by an intracellular current injection. The recording was performed using the patch clamp technique.

1.5 ANALOG-TO-DIGITAL CONVERSION

The nature of biomedical signals is analog: i.e., continuous both in amplitude and time. Modern data acquisition and analysis frequently depends on digital signal processing, and therefore the signal must be converted into a discrete representation. The time scale is made discrete by sampling the continuous wave at a given interval; the amplitude scale is made discrete by an analog-to-digital converter (***A/D converter*** or ***ADC***), which can be thought of as a truncation or rounding of a real-valued measurement to an integer representation.

An important characteristic of an ADC is its amplitude resolution, which is measured in bits. A simplified example with a 3-bit converter (giving $2^3 = 8$ levels) is shown in Fig. 1.6. Usually converters have at least an 8-bit range, producing $2^8 = 256$ levels. In most biomedical equipment a 16-bit range ($2^{16} = 65{,}536$ levels) or higher is considered state-of-the-art.

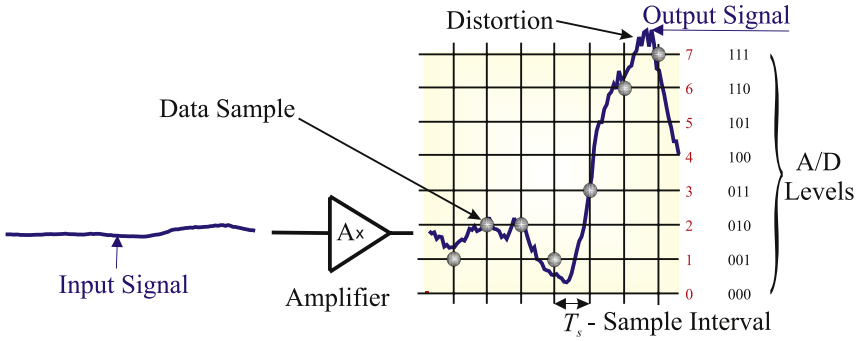


FIGURE 1.6 Analog-to-digital conversion (ADC). An example of an analog signal that is amplified A× and digitized showing seven samples (marked by the *dots*) taken at a regular sample interval T_s, and a 3-bit A/D conversion. There are $2^3 = 8$ levels (0–7) of conversion. The decimal (0–7) representation of the digitizer levels is in red, the 3-bit binary code (000–111) in black. Note that, in this example, the converter represents the amplified signal values as integer values based on the signal value rounded to the nearest discrete level.

As can be seen in Fig. 1.6, the resolution of the complete ADC process expressed in the potential step per digitizer unit (e.g., μV/bit) is not uniquely determined by the ADC, but also depends on the analog amplification. After the measurements are converted, the data can be stored in different formats: integer, real/float, or ASCII. It is common to refer to 8 bits as a byte and a combination of bytes (e.g., 4 bytes) as a word.

1.6 MOVING SIGNALS INTO THE MATLAB® ANALYSIS ENVIRONMENT

Throughout this course we will explore signal processing techniques with real signals. Therefore, it is critical to be able to move measurements into the analysis environment. Here we give two examples of reading recordings of neural activity into MATLAB®. To get an overview of file types that can be read directly into MATLAB®, you can search the MATLAB® documentation for "Data Import and Export." Most files

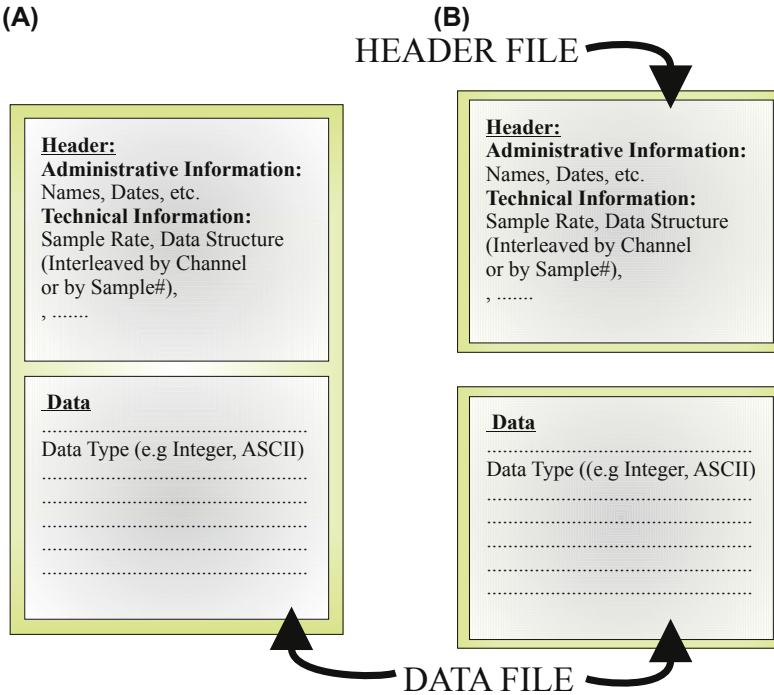


FIGURE 1.7 Data files. (A) An integrated file including both header information and data. Sometimes the header information is at the end of the file (tailer). (B) Separate header and data files.

recorded with biomedical equipment are not directly compatible with MATLAB® and must be edited and/or converted. Usually this conversion requires either a number of steps to reformat the file, or reading the file using the low-level fopen and fread commands. Since A/D converters typically generate integer values, most commercial data formats for measurement files consist of arrays of integer words. Such a file may contain some administrative information at the beginning (header) or end ("tailer"); in other cases, this type of measurement-related information is stored in a separate file (sometimes called a header file; see Fig. 1.7).

As an exercise we will move data from two example data sets into MATLAB®. The data sets and associated files and script can be downloaded from http://booksite.elsevier.com/9780128104828/; one set is an EEG recording (consisting of two files data.eeg and data.bni) and the other is a measurement of a neuron's membrane potential (Cell.dat). Like many biomedical signals these data sets were acquired using a proprietary acquisition system with integrated hardware and software tools. As we will see, this can complicate the process of importing data into our analysis environment.

The membrane potential recording (Cell.dat) can be directly read with AxoScope or any software package that includes the AxoScope reader (Axon Instruments, Inc.). If you have access to such a package, you can store a selection of the data in a text file format (*.atf). This file includes header information followed by the data itself (Fig. 1.7A). If you do not have access to the proprietary reader software, you can work with an output text file of AxoScope that is also available on the CD (Action_Potentials.atf). In order to be able to load this file (containing the single cell data) in MATLAB®, the header must be removed using a text editor (such as WordPad in a Windows operating system). The first few lines of the file as seen in WordPad are shown here:

```
ATF    1.0
7      4
"AcquisitionMode=Gap Free"
"Comment="
"YTop=10,100,10"
"YBottom=-10,-100,-10"
"SweepStartTimesMS=72839.700"
"SignalsExported=PBCint, neuron,current"
"Signals="    "PBCint"    "neuron"    "current"
"Time (s)"    "Trace #1 (V)"    "Trace #1 (mV)"    "Trace #1 (nA)"
72.8397       0.90332     -58.5938    0.00976563
72.84         0.898438    -58.5938    0
72.8403       0.90332     -58.7402    -0.00976563
....
```

After deleting the header information, the file contains only four columns of data.

```
72.8397        0.90332        -58.5938        0.00976563
72.84          0.898438       -58.5938        0
72.8403        0.90332        -58.7402        -0.00976563
72.8406        0.898438       -58.6914        0.00488281
72.8409        0.90332        -58.6426        -0.00488281
...
```

This can be stored as a text file (Action_Potentials.txt) containing the recorded data (without header information) before loading the file into MATLAB®. The MATLAB® command to access the data is `load Action_Potentials.txt -ascii`. The intracellular data are in the third column and can be made displayed with the command `plot(Action_Potentials(:,3))`. The obtained plot result should look similar to Fig. 1.5. The values in the graph are the raw measures of the membrane potential in mV. If you have a background in neurobiology, you may find these membrane potential values somewhat high; in fact these values must be corrected by subtracting 12 mV (the so-called liquid junction potential correction).

In contrast to the intracellular data, the example EEG measurement data we present below consist of a separate header file (data.bni) and data file (data.eeg), corresponding to the diagram in Fig. 1.7B. As shown in the figure, the header file is an ASCII text file, while the digitized measurements in the data file are stored in a 16-bit integer format. Since the data and header files are separate, MATLAB® can read the data without modification of the file itself, though importing this kind of binary data requires the use of lower-level commands (as we will show). Since EEG files contain records of a number of channels, sometimes over a long period of time, the files can be quite large and therefore unwieldy in MATLAB®. For this reason, it may be helpful to use a special review application to select smaller segments of data which can be saved in separate files and read into MATLAB® in more manageable chunks. In this example we do not have to find a subset of the recording because we have already preselected a 10-s EEG epoch. If you don't have access to special EEG reader software, you can see what the display would look like in the jpg files: data_montaged_filtered.jpg and data.jpg. These files show the display in an EEG reader application (EEGVue, Nicolet Biomedical Inc.) of the data.eeg file in a montaged and filtered version and in a raw data version, respectively.

***The following MATLAB® script shows the commands for loading the raw data from data.eeg*:**

```
% pr1_1.m
sr=400;                                  % Sample Rate
Nyq_freq=sr/2;                           % Nyquist Frequency
fneeg=input('Filename (with path and extension) :', 's');
t=input('How many seconds in total of EEG ? : ');
ch=input('How many channels of EEG ? : ');
le=t*sr;                                 % Length of the Recording
fid=fopen(fneeg, 'r', 'l');              % *) Open the file to read('r') and
                                         little-endian ('l')
EEG=fread(fid,[ch,le],'int16');          % Read Data -> EEG Matrix
fclose ('all');                          % Close all open Files
```

*) The little-endian byte ordering is only required in a few special cases, in straightforward PC to PC data transfer the 'l' option in the fopen statement can be omitted.

Executing the above commands in a MATLAB® command window or via the MATLAB® script that can be downloaded from http://booksite.elsevier.com/9780128104828/ (pr1_1.m) generates the following questions:

```
Filename (with path and extension) : data.eeg
How many seconds in total of EEG ? : 10
How many channels of EEG ? : 32
```

The answers to the questions are shown in bold. You can now plot some of the data you read into the matrix EEG with plot(-EEG(1,:)), plot(-EEG(16,:)), or plot(EEG(32,:)). The first two plot commands will display noisy EEG channels; the last trace is an ECG recording. The − (minus) signs in the first two plot commands are included in order to follow the EEG convention of showing negative deflections upward. To compare the MATLAB® figures of the EEG with the traces in the proprietary EEGVue software, see the jpeg file showing the raw data data.jpg. Alternatively you can quickly verify the integrity of your result by checking channel 32 for the occurrence of QRS complexes similar to the one shown in Fig. 1.4B.

Like the first few lines of header information in the single cell data file above, the first few lines of the separate EEG header file (data.bni) contain

similar housekeeping information. Again, this ASCII-formatted file can be opened with a text editor such as WordPad, revealing the following:

```
FileFormat = BNI-1
Filename = f:\anonymous_2f1177c5_2a99_11d5_a850_00e0293dab97\
data.bni
Comment =
PatientName = anonymous
PatientId = 1
.....
```

Usually the file formats of measurements in neuroscience are specific to the manufacturer or even the recording device. Therefore, each instrument may require its own, often nontrivial, custom procedure to export recordings across different software applications. One exception is the so-called European Data Format (EDF) that was developed as a generic file format of the type shown in Fig. 1.7A. It was initially introduced in the 1990s for storing sleep and EEG data and more recently updated to EDF^{+} to also include other modalities such as EKG, electromyography, and EP measurements (Kemp et al., 1992; Kemp and Olivan, 2003). Fortunately, several instrument manufacturers now support EDF as one of their file formats.

APPENDIX 1.1

This appendix provides a quick reference to some basic laws frequently used to analyze problems in neurobiology, and which are cited throughout this text (Fig. A1.1-1). Further explanation of these laws can be found in any basic physics textbook.

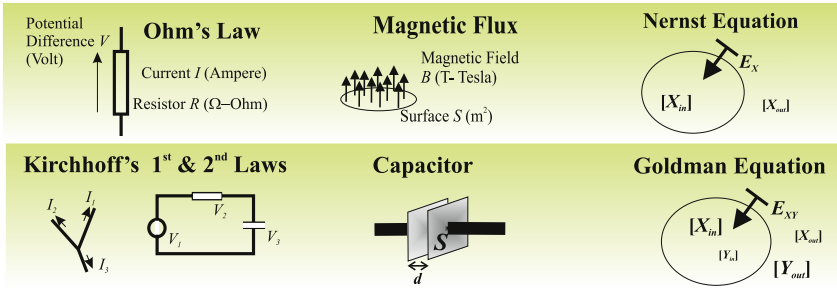


FIGURE A1.1-1 Overview of basic physics laws.

Ohm's Law: The potential difference V (V, or Volt) over a conductor with resistance R (Ω − Ohm) and current I (A, or Ampère) can be related by:

$$V = IR \tag{A1.1-1}$$

Kirchhoff's first Law: At a junction all currents add up to 0,

$$\sum_{i=1}^{N} I_i = 0 \tag{A1.1-2}$$

Kirchhoff's second law: In a circuit loop all potentials add up to 0,

$$\sum_{i=1}^{N} V_i = 0 \tag{A1.1-3}$$

Magnetic flux induces a potential difference:

$$V = -\frac{d\Phi_B}{dt} \tag{A1.1-4}$$

Φ_B = the magnetic flux (Wb—Weber) through a loop with surface area S (m^2) in a magnetic field of B (T—Tesla), i.e.: $\Phi_B = BS$.

Furthermore, the magnitude of the magnetic field B generated by a current I at a distance d (m—meter) is given by $B = \frac{\mu}{2\pi}\frac{I}{d}$ where μ = magnetic permeability (in a vacuum $\mu_0 = 4\pi \times 10^{-7}$).

Capacitance-related equations: The potential difference V between the two conductors of a capacitor is the quotient of charge Q (C—Coulomb) and capacitance C (F—Farad):

$$V = \frac{Q}{C} \quad \text{or} \quad Q = CV \tag{A1.1-5}$$

Current is the derivative of the charge Q:

$$i = \frac{dQ}{dt} \quad \text{and} \quad Q = \int i\, dt \tag{A1.1-6}$$

Capacitance C is proportional to the quotient of surface area S (m^2—square meter) of the conductors and their interdistance d:

$$C = \varepsilon \frac{S}{d} \tag{A1.1-7}$$

$\boldsymbol{\varepsilon}$ = dielectric constant of the medium in between the conductors ($\boldsymbol{\varepsilon} = 8.85 \times 10^{-12}$ for a vacuum).

***Nernst equation*:**

$$E_X = \frac{RT}{zF}\ln\left(\frac{[X_{out}]}{[X_{in}]}\right) \tag{A1.1-8}$$

The potential difference E_X created by a difference of concentrations of ion species X inside $[X_{in}]$ and outside $[X_{out}]$ the cell membrane. The constants R, T, and F are the gas constant, absolute temperature, and Avogadro's number, respectively. Parameter z denotes the charge of the ion, e.g., $+1$ for Na^+ or K^+, -1 for Cl^-, and $+2$ for Ca^{2+}.

***Goldman equation*:**

$$E_{XY} = \frac{RT}{F} \ln\left(\frac{p_X[X_{out}] + p_Y[Y_{out}]}{p_X[X_{in}] + p_Y[Y_{in}]}\right) \qquad \text{(A1.1-9)}$$

Similar to the Nernst equation, but here we consider the effect of multiple ion species, e.g., Na^+ and K^+ for X and Y. In this case, unlike the Nernst equation, the concentrations are weighted by the membrane permeability of the ions (e.g., p_{Na} and p_K for p_X and p_Y).

In both the Nernst and Goldman equations, at room temperature (25°C) RT/F ln(…) can be replaced by:

$$58 \text{ mV } \log_{10}(\ldots)$$

EXERCISES

1.1 A signal is amplified 1000 × and connected to an 8-bit ADC with an input range (note: input at the ADC) of ±5V
   a. What is the (dynamic) range at the amplifier input?
   b. What is the digitizer resolution at the amplifier input?
   c. How would you modify the setup in order to record an EEG signal with a 500 μV amplitude? (If possible, indicate different alternatives.)

1.2 We have two resistors R_1 and R_2 connected in series. R_2 is connected to ground (0V) and R_1 is hooked up to a battery of 9V. If $R_1 = 10\Omega$ and $R_2 = 10\Omega$,
   a. What is the current i running in the circuit?
   b. What is the potential in between R_1 and R_2?

1.3 We have a capacitor hooked up to the battery of 9V. The capacitance is 10 nF. Answer the following questions assuming the system reached equilibrium.
   a. What is the current in the circuit?
   b. What is the charge of the capacitor?

1.4 Consider a membrane with 140 mM Na^+ on one side and 14 mM Na^+ on the other. What is the membrane potential,
   a. if the membrane is impermeable for Na^+?
   b. if the membrane is permeable for Na^+?

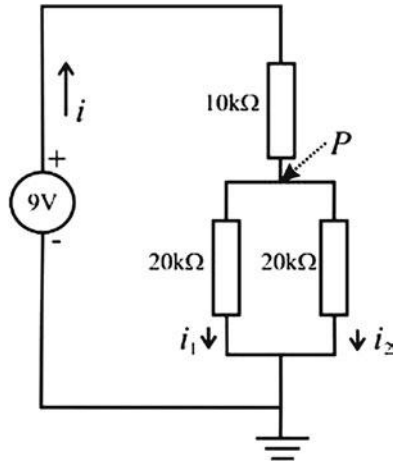


FIGURE E1.5 A three-resistor network connected to a 9-V battery.

1.5 For the circuit in Fig. E1.5, answer the following questions.
   a. What resistors are parallel?
   b. What resistors are in series?
   c. Use Ohm's law to compute total current i.
   d. Use Kirchhoff's first law to compute currents i_1 and i_2.
   e. Use Kirchhoff's second law to compute the potential at point P.
   [Recall that the equivalent resistor R for two parallel resistors R_1 and R_2 can be determined with $\frac{1}{R_1} + \frac{1}{R_2} = \frac{1}{R}$].

References

Hodgkin, A.L., Huxley, A.F., 1952. A quantitative description of membrane current and its application to conduction and excitation in the nerve. J. Physiol. 117, 500–544.
A seminal paper describing the Hodgkin and Huxley equations.

Kemp, B., Olivan, J., 2003. European data format 'plus' (EDF+), an EDF alike standard format for the exchange of physiological data. Clin. Neurophysiol. 114 (9), 1755–1761.

Kemp, B., Värri, A., Rosa, A.C., Nielsen, K.D., Gade, J., 1992. A simple format for exchange of digitized polygraphic recordings. Electroencephalogr. Clin. Neurophysiol. 82 (5), 391–393.

Oostenveld, R., Praamstra, P., 2001. The five percent electrode system for high-resolution EEG and ERP measurements. Clin. Neurophysiol. 112, 713–719.
Definition of the electrode placement in human EEG recording.

Spiegel, J., Hansen, C., Baumgärtner, U., Hopf, H.C., Treede, R.-D., 2003. Sensitivity of laser-evoked potentials versus somatosensory evoked potentials in patients with multiple sclerosis. Clin. Neurophysiol. 114, 992–1002.
An example of the application of evoked potentials in clinical electrophysiology.

CHAPTER

2

Data Acquisition

2.1 RATIONALE

Unless we use simulated signals, data acquisition necessarily precedes signal processing. In any recording setup, the devices that are interconnected and coupled to the biological process form a so-called measurement chain. In Chapter 1, we discussed the acquisition of a waveform via an amplifier and analog-to-digital converter (ADC) step (see Fig. 1.6). Here we elaborate on the process of data acquisition by looking at the role of the components in the measurement chain in more detail (Fig. 2.1).

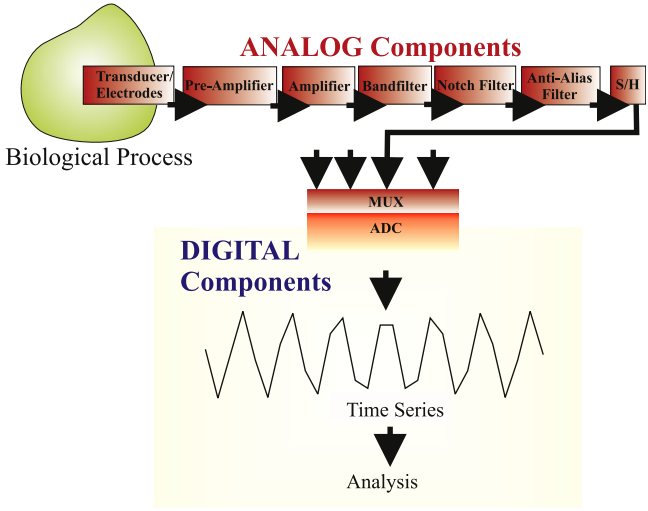


FIGURE 2.1 Diagram of a data acquisition setup, the measurement chain. The modules up to the analog-to-digital converter (ADC) are analog devices, while the remaining components are the digital devices. *MUX*, multiplexer; *S/H*, sample hold module.

In-depth knowledge of the measurement process is often critical for effective data analysis, because each type of data acquisition system is associated with specific artifacts and problems. Technically accurate measurement and proper treatment of artifacts are essential for data processing; these steps guide the selection of the processing strategies, the interpretation of results, and allow one to avoid the "***garbage in = garbage out***" trap that comes with every type of data analysis.

2.2 THE MEASUREMENT CHAIN

Most acquisition systems can be subdivided into analog and digital components (Fig. 2.1). The analog part of the measurement chain conditions the signal (through amplification, filtering, etc.) prior to the A/D conversion. Observing a biological process normally starts with the connection of a transducer or electrode pair to pick up a signal. Usually, the next stage in a measurement chain is amplification. In most cases the amplification takes place in two steps using a separate preamplifier and amplifier. After amplification, the signal is usually filtered to attenuate undesired frequency components. This can be done by passing the signal through a bandpass filter and/or by cutting out specific frequency components (using a band reject, or notch filter) such as a 60-Hz hum. A critical step is to attenuate frequencies that are too high to be digitized by the ADC. This operation is performed by the antialiasing filter. Finally, the sample-and-hold (S/H) circuit samples the analog signal and holds it to a constant value during the ADC process. The diagram in Fig. 2.1 represents a basic acquisition setup in which some functions can be interchanged, omitted, or moved into the digital domain; this will be discussed in Section 2.4.

The goal of the acquisition setup is to measure biological signals as "cleanly" (with as little noise) as possible, without significant interactions due to the measurement itself. For instance, if a bioelectrical response is to be measured, we want to ***establish the correct amplitude*** of the biopotential without ***influencing (i.e., stimulating or inhibiting) the system with current originating from the equipment***.

2.2.1 Analog Components

In the analog part of the measurement chain, one normally connects different instruments to obtain an analog signal with appropriate characteristics for the ADC (Fig. 2.1). As will be outlined in the following, when connecting equipment, one has to follow the rule of *low output impedance—high input impedance*. As Fig. 2.2 shows, any element in the

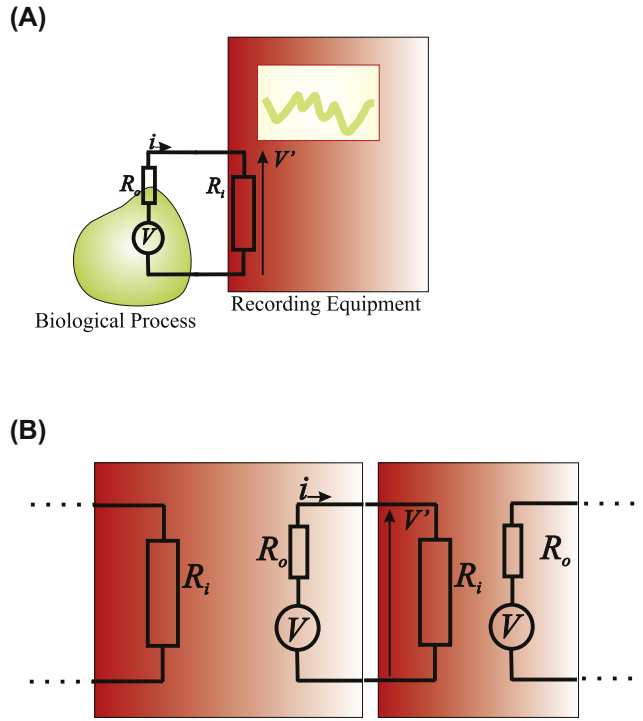


FIGURE 2.2 Equivalent circuit representation of elements in a measurement chain. (A) A simplified situation in which a biological process is directly coupled to an oscilloscope. (B) A generic diagram of coupling devices in a chain.

chain can be represented as a black box with an input and output resistance. The situation in Fig. 2.2A is a biological preparation generating a biopotential coupled via direct electrical contact to an oscilloscope screen displaying the measured signal. In this example, the biopotential (V) is associated with a current (i) that is (according to Ohm's law) determined by R_o (the output resistance) and R_i (the input resistance).

$$i = \frac{V}{R_i + R_o} \tag{2.1}$$

Ideally one would like to measure V without drawing any current (i) from the biological process itself. Because it is impossible to measure a potential without current, at best we can minimize the current drawn from our preparation at any given value of the biopotential (V); ***therefore considering Eq. (2.1) we may conclude that $R_i + R_o$ must be large to minimize current flow within the preparation from our instruments***.

The other concern is to obtain a reliable measurement reflecting the true biopotential. The oscilloscope in Fig. 2.2A cannot measure the exact value because the potential is attenuated over both the output and input resistors. The potential V' in the oscilloscope relates to the real potential V as:

$$V' = \frac{R_i}{R_i + R_o} V \tag{2.2}$$

V' is close to V if $R_i \gg R_o$ producing an attenuation factor that approaches 1.

The basic concepts in this example apply not only for the first step in the measurement chain, but also for any connection in a chain of instruments (Fig. 2.2B). Specifically, a high input resistance combined with a low output resistance ensures that:

1. no significant amount of current is drawn;
2. the measured value at the input represents the output of the previous stage.

Measurements of biopotentials are not trivial since the electrodes themselves constitute a significant resistance and capacitance (Fig. 2.3), usually indicated as electrode impedance. Electroencephalogram (EEG) electrodes on the skin have an impedance of about 5 kΩ (typically measured at 20–30 Hz); microelectrodes that are used in most basic electrophysiology studies have an impedance from several hundreds of kΩ up to several MΩ (measured at around 1 kHz). This isn't an ideal starting point; constraint (***1***) above will be easily satisfied (the electrodes by themselves usually have a high impedance which limits the current) but constraint (**2**) is a bit more difficult to meet. This problem can only be resolved by including a primary amplifier stage with an input-impedance that is extremely high, i.e., several orders of magnitude above the electrode's impedance. This is the main function of the preamplifier or head

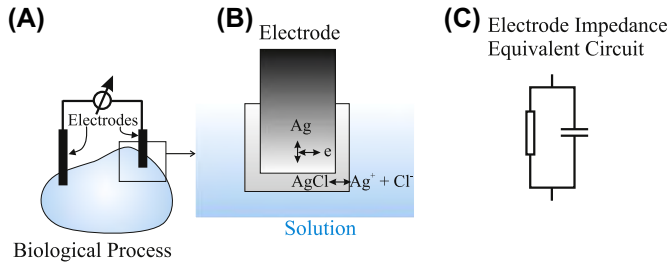


FIGURE 2.3 Components of typical biopotential measurement. (A) A setup with silver–silver chloride electrodes with (B) a detail of the chloride layer and (C) a simplified electronic equivalent circuit.

stage in measurement setups. For this reason these devices are sometimes referred to as impedance transformers: the input impedance is extremely high while the output impedance of the head stage is only several Ω.

In electrophysiology experiments, metal electrodes are often used to measure potentials from biological specimens which must be bathed in an ionic solution. A fundamental problem with such direct measurements of electricity in solutions is the interface between the metal and solution. This boundary generates an electrode potential that is material and solution specific. The electrode potential is usually not a problem when biopotentials are read from electrode pairs made of the same material. In cases where the metal and solutions are not the same for both electrodes, the DC offset generated at the electrode−solution interface can usually be corrected electronically in the recording equipment. Somewhat more problematically, the metal−fluid boundary can act as an impedance with a significant capacitive element (Fig. 2.3C). This capacitance may degrade the signal by blocking the low-frequency components. One widely used approach to mitigate this problem is to use a silver electrode with a silver chloride coating. This facilitates the transition from ionic (Ag^+ or Cl^-; Fig. 2.3B) to electronic (e; Fig. 2.3B) conduction, reducing the electrode capacitance at the solution interface, and consequently facilitating the recording of signals with low-frequency components.

The purpose of amplification in the analog domain is to increase the signal level to match the range of the ADC. Unfortunately, since amplifiers increase the level of both desirable and undesirable elements of signals, additional procedures are often required to reduce noise contamination. This is typically accomplished with analog filtering before, or digital filtering after, the ADC. With the exception of the antialiasing filter, the replacement of analog filters with digital filters is equivalent from a signal processing point of view. The purpose of the ***antialiasing filter*** in the analog part of the measurement chain is to prevent the system from creating erroneous signals at the ADC. This will be explained in the Sections 2.2.2 and 2.3.

So far we have considered the acquisition of a single channel of data. In real recording situations one is frequently interested in multiple channels. Recordings of clinical EEG typically vary between 20 and 32 channels, and electrocorticogram (ECoG) measurements often include more than 100 channels. These channels are usually digitized by a limited number of ADCs, with each ADC connected to a set of input channels via a multiplexer (MUX; Fig. 2.1), a high-speed switch that sequentially connects these channels to the ADC. Because each channel is digitized in turn, a small time lag between the channels may be introduced at conversion. In most cases with modern equipment, where the switching and conversion times are small, no compensation for these time shifts is necessary. However, with a relatively slow, multiplexed A/D converter, a so-called

sample-hold unit must be included in the measurement chain (Fig. 2.1). An array of these units can hold sampled values from several channels during the conversion process, thus preventing the converter from "chasing" a moving target and avoiding a time lag between data streams in a multichannel measurement.

2.2.2 Analog to Digital Conversion

ADC can be viewed as imposing a grid on a continuous signal (see Fig. 1.6). The signal becomes discrete both in ***amplitude*** and ***time***. It is obvious that the grid must be sufficiently fine and must cover the full extent of the signal to avoid a significant loss of information.

The discretization of the signal in the ***amplitude*** dimension is determined by the converter's input voltage range and the analog amplification of the signal input to it (Chapter 1, Fig. 1.6). As an example: we have a 12-bit converter with an input-range of 5V and an analog measurement chain with a preamplifier that amplifies 100× and a second-stage amplifier that amplifies 100×. The result is a total amplification of 10,000, translating into (5V ÷ 10,000 =) 500 μV ***range*** for the input of the acquisition system. The converter has 2^{12} steps (4096), resulting in a ***resolution*** at the input of (500 μV ÷ 4096 = 0.12 μV). It may seem that an ADC with a greater bit depth is better because it generates samples at a higher precision. However, sampling at this higher precision in the ADC may be inefficient because it requires a lot of memory to store the acquired data, without providing any additional information about the underlying biological process. In such a case all the effort is wasted on storing noise. Therefore, in real applications, there is a trade-off between resolution, range, and storage capacity.

At conversion, the amplitude of the analog signal is approximated by the discrete levels of the ADC. Depending on the type of converter this approximation may behave numerically as a truncation or as a round-off of the continuous-valued signal to an integer. In both cases one can consider the quantization as a source of noise in the measurement system, noise which is directly related to the resolution at the ADC (***quantization noise***, Chapter 3).

The continuous signal is also discretized (sampled) in ***time***. To obtain a reliable sampled representation of a continuous signal, the sample interval (T_s) or sample frequency ($F_s = 1/T_s$) must relate to the type of signal that is being recorded. In order to develop a mathematical description of sampling, we introduce the unit impulse (Dirac impulse) function δ.

The plots in Fig. 2.4A show how the unit step and unit impulse functions can be thought of as a ramp function and its derivative, respectively, in the limit as the ramp width τ approaches 0. In terms of the amplitude

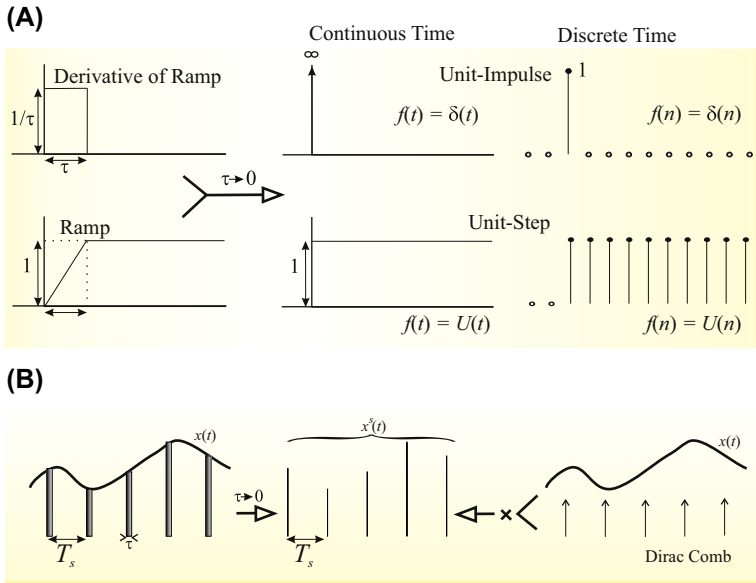


FIGURE 2.4 Graphical representation of the Dirac δ in continuous and discrete time. (A) The unit impulse (δ, top row) and unit step (U, bottom row) function. The unit impulse can be considered as the derivative of the unit step. The unit impulse can be considered a square wave with duration τ and amplitude $1/\tau$ in which $\tau \to 0$. Note also that in continuous time the amplitude of the unit impulse is ∞, whereas the amplitude is 1 in the discrete time version. Here both the impulse and step functions are derived from the ramp function, though other approaches exist (e.g., Chapter 20). (B) Sampling a continuous function $x(t)$ by multiplication with the Dirac comb generates discrete time function $x^s(t)$.

$\delta(0)$, the unit impulse (Dirac) function at 0, behaves a bit differently for the continuous (∞) and discrete time (1) versions. The unit step functions in discrete and continuous time both have amplitudes of 1.

The Dirac delta function in the integral and summation expressions in Table 2.1 can be used to sample a continuous function $x(t)$ at $t = 0$. If we

TABLE 2.1 Dirac Delta Function

Continuous Time	Discrete Time
$\delta(t) = 0 \quad \text{for } t \neq 0$	$\delta(n) = 0 \quad \text{for } n \neq 0$
$\int_{-\infty}^{\infty} \delta(t)dt = 1$	$\sum_{n=-\infty}^{\infty} \delta(n) = 1$

define the top-left function in Fig. 2.4A (a square wave with duration τ and amplitude $1/\tau$) as the approximation δ_τ for δ, we can state:

$$\int_{-\infty}^{\infty} x(t)\delta(t)dt = \lim_{\tau\to 0}\int_{-\infty}^{\infty} x(t)\delta_\tau(t)dt \tag{2.3}$$

Because $\delta_\tau(t) = 0$ outside the $0 \to \tau$ interval, we can change the upper and lower limits of the integration:

$$\lim_{\tau\to 0}\int_{-\infty}^{\infty} x(t)\delta_\tau(t)dt = \lim_{\tau\to 0}\int_{0}^{\tau} x(t)\delta_\tau(t)dt \tag{2.4}$$

Within these limits, $\delta_\tau(t) = \frac{1}{\tau}$; therefore we obtain:

$$\lim_{\tau\to 0}\int_{0}^{\tau} x(t)\delta_\tau(t)dt = \lim_{\tau\to 0}\int_{0}^{\tau} \frac{x(t)}{\tau}dt \tag{2.5}$$

If we now use $\tau \to 0$, so that $x(t)$ becomes $x(0)$, which can be considered a constant and not a function of t anymore, we can evaluate the integral:

$$\lim_{\tau\to 0}\int_{0}^{\tau} \frac{x(t)}{\tau}dt = \lim_{\tau\to 0} x(0)\underbrace{\int_{0}^{\tau}\frac{1}{\tau}dt}_{1} = x(0) \tag{2.6}$$

Because the integral evaluates to 1 and combining the result with our starting point in Eq. (2.3), we conclude:

$$x(0) = \int_{-\infty}^{\infty} x(t)\delta(t)dt \tag{2.7}$$

Here we assumed that the integral for the δ function remains 1 even as $\tau \to 0$. The reasoning we followed to obtain the result above isn't the most rigorous, but it makes it a plausible case for the integral in Eq. (2.7) evaluating to $x(0)$.

By using $\delta(t - \Delta)$ instead of $\delta(t)$, we obtain the value of a function at $t = \Delta$ instead of $x(0)$. If we now consider a function evaluated at arbitrary values of delay Δ, we obtain the so-called ***sifting property*** of the unit impulse function:

$$\boxed{x(\Delta) = \int_{-\infty}^{\infty} x(t)\delta(t-\Delta)dt} \tag{2.8}$$

Using this property, we can sift out specific values of a continuous function $x(t)$ at given values of t. As we will see in the remainder of this text, this property of the delta function is frequently used to evaluate integrals including the δ function.

The Dirac δ function is used to formalize the sampling of a continuous time function. We can depict this sampling procedure as a continuous time function $x(t)$ that is sampled over very short time intervals τ at regular intervals T_s, and that is considered zero in between the sampling times (Fig. 2.4B). Each of the gray rectangles at time instant nT_s in the left plot in Fig. 2.4B can be considered as an approximation of the Dirac delta $\delta_\tau(t - nT_s)$ that is weighted by the value of $x(t)$ at $t = nT_S$, i.e., each sample value at $t = nT_S$ equals $x(nT_s)\delta_\tau(t - nT_s)$. If we add all individual samples (sampling the whole function $x(t)$ at regular intervals separated by T_s), we get the ***sampled representation*** x^s, which can be written as: $\sum_{n=-\infty}^{\infty} x(nT_s)\delta_\tau(t - nT_s)$. If we subsequently let $\tau \to 0$, then the approximated delta function δ_τ approaches the true δ. Each impulse at $t = nT_S$ is weighted by $x(nT_s)$. The representation of the sampled function now looks like the middle panel in Fig. 2.4B, where the sampled function x^s is represented by very brief pulses of amplitude $x(nT_s)$ and zero in between these pulses. Following this reasoning we make it plausible to represent the sampled equivalent of continuous time function x as x^s:

$$x^s(nT_s) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \tag{2.9}$$

In the above equation we took the liberty of replacing $x(nT_s)$ with $x(t)$, i.e., we used the equality $x(nT_s)\delta(t - nT_s) = x(t)\delta(t - nT_s)$. This again is a plausible step because the delta function $\delta(t - nT_s)$ equals zero for all $t \neq nT_S$, so including values of $x(t)$ other than $t = nT_S$ does not affect the outcome of the product. The expression $\sum_{n=-\infty}^{\infty} \delta(t - nT_s)$ represents a series of Diracs at regular intervals and is often called the ***Dirac comb*** δ_{Ts} (Fig. 2.4B right panel). Because the sample interval T_s is usually a constant, it is often omitted, thereby indicating x^s as a function of n only. Finally, we obtain the commonly used representation of a sampled function as the product of a Dirac comb and the continuous time function (Fig. 2.4B):

$$\boxed{x^s(n) = x(t)\delta_{T_s}} \tag{2.10}$$

Again, the procedures we used above to introduce the properties of the Dirac functions in Eqs. (2.8) and (2.9) were more intuitive than mathematically rigorous; though the reasoning underlying these properties can be made rigorous using distribution theory, which is not further discussed in this text.

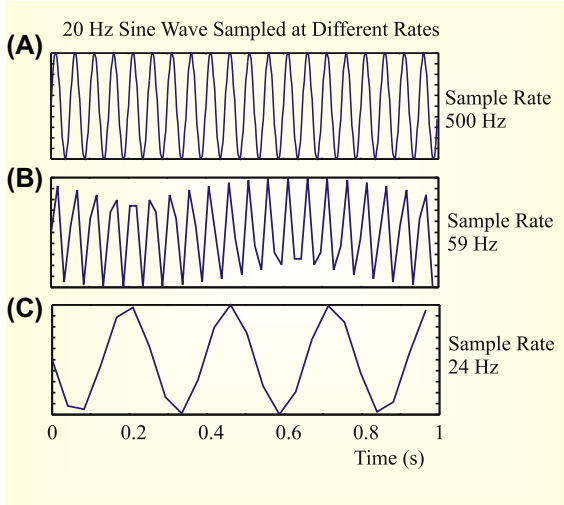


FIGURE 2.5 Sampling a 20-Hz sine wave at different rates $F_s = 1/T_s$. The effects shown in this figure can be further examined with the MATLAB® pr2_1.m script. *Fs*, sample rate; *Ts*, sample interval.

From time domain observation, it may be obvious that the sample rate at which one obtains $x^s(t)$ must be sufficient to represent the change in the continuous signal $x(t)$. Several examples are shown in Fig. 2.5. As illustrated schematically in the figure, it seems that sampling a 20-Hz sine wave at a rate of $2 \times 20 = 40$ Hz at least conserves the frequency content of the signal. If these samples were taken exactly at the peaks and valleys of the sine wave, the sampled wave would look like a 20-Hz triangular wave. If not sampled at the peaks and valleys, the waveform will have an even more severely distorted appearance.

The waves in Fig. 2.5 are examples created with pr2_1.m in MATLAB®.

```
% pr2_1.m
% Aliasing
% example signal
t=0:0.001:1;                    % 1 sec divided into ms steps
f=20;                           % Frequency in Hertz
signal=sin(2*pi*f*t);

% Simulate different sample rates and plot
figure
for skip=2:5:50;
   plot(t,signal,'r'); hold;        % The Original Signal
   plot(t(1:skip:1000),signal(1:skip:1000));
```

```
    tt=['Sine' num2str(f) ' Hz: space bar to continue: SAMPLE RATE = '
        num2str(1000/skip)];
    title(tt);
    drawnow
    pause;
    clf;
end;
```

If you need to refresh or practice your MATLAB® skills, do one of the introductory courses or see a text such as Ingle and Proakis (1997) or Wallisch et al. (2014). Running the preceding program shows the original waveform in red and the simulated sampled version in blue. Press "Enter" to see subsequent lower sample rates. The minimum sampling rate (in this example 40 Hz) is called the ***Nyquist sampling frequency*** or the ***Nyquist limit***. Thus, the sampling rate determines the highest frequency that can be represented by the sampled signal. This value (half the sample rate) is often indicated as the ***Nyquist frequency*** of the sampled signal.

In the example in Fig. 2.5 the highest frequency in the signal is 20 Hz, requiring a sample rate >40 Hz. The Nyquist limit is a real bare minimum to capture the 20-Hz frequency component and you can see in the figure that the wave morphology is already distorted at sample rates close to, but above, the Nyquist sampling frequency (e.g., 59 Hz in Fig. 2.5B). Clearly the signal is seriously misrepresented below the Nyquist limit (e.g., 24 Hz in Fig. 2.5C). This particular type of signal distortion is called ***aliasing***: the example in Fig. 2.5 shows a signal of ~4 Hz that is an alias of the real 20-Hz signal resulting from undersampling.

To remove the effect of aliasing in digitized signals, the analog measurement chain must remove/attenuate all frequencies above the Nyquist frequency by using a filter (***antialiasing filter***). To avoid distortion in the time domain (as seen in the example where the wave is digitized at 59 Hz) sampling at ~5 times the maximum frequency is not uncommon.

> *Note:* Aliasing is not a phenomenon that occurs only at the ADC, but at all instances where a signal is made discrete. It may also be observed when waves are represented on a screen or on a printout with a limited number of pixels. It is also not restricted to time series but also occurs when depicting images (2-D signals) in a discrete fashion.

2.3 SAMPLING AND NYQUIST FREQUENCY IN THE FREQUENCY DOMAIN

This section considers the Nyquist sampling theorem in the frequency domain. Unfortunately, this explanation in its simplest form requires a background in the Fourier transform and convolution, both topics that will be discussed later (see Chapters 5–7, 13). For readers who are not yet familiar with these topics, it is advised to skip this section and return to it later. In this section we approach sampling in the frequency domain somewhat intuitively, and focus on the general principles depicted in Fig. 2.6. A more formal treatment of the sampling problem can be found in Appendix 2.1.

When sampling a function $f(t)$, using the property of the δ function (Eq. 2.9), we multiply the continuous time function with a Dirac comb, a series of unit impulses with regular interval T_s:

$$\text{Sampled function}: \quad f(t)\sum_{n=-\infty}^{\infty}\delta(t-nT_s) \tag{2.11}$$

As will be discussed in Chapter 13, multiplication in the time domain is equivalent to a convolution $(\otimes)$ in the frequency domain, i.e.:

$$F(f)\otimes\Delta(f) \quad \text{with}: \quad F(f)\Leftrightarrow f(t) \quad \text{and} \quad \Delta(f)\Leftrightarrow\sum_{n=-\infty}^{\infty}\delta(t-nT_s) \tag{2.12}$$

The double arrow $\Leftrightarrow$ in Eq. (2.12) separates a Fourier transform pair: here the frequency domain is left of the arrow and the time domain equivalent is the expression on the right of $\Leftrightarrow$. We can use the sifting property to evaluate the Fourier transform integral of a single delta function:

$$\delta(t)\Leftrightarrow\int_{-\infty}^{\infty}\delta(t)e^{-2\pi ft}dt=e^{0}=1 \tag{2.13}$$

For the series of impulses (the Dirac comb) the transform $\Delta(f)$ is a more complex expression, according to the definition of the Fourier transform (Eq. 6.4, Chapter 6):

$$\Delta(f)=\int_{-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\delta(t-nT_s)e^{-2\pi ft}dt \tag{2.14}$$

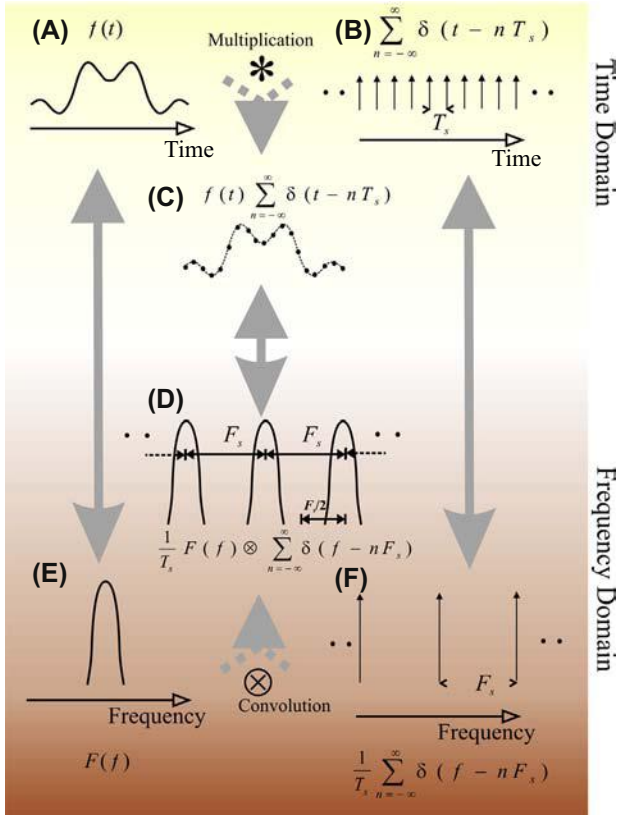


FIGURE 2.6 Fourier transform of a sampled function. Sampling a function $f(t)$ (A) in the time domain can be represented by a multiplication (*) of $f(t)$ with a train of δ functions with an interval T_s, as depicted in (B), resulting in a series of samples (C). The Fourier transform of the sampled version is a periodic function, as shown in (D). The Fourier transform of the sampled function can be obtained from the convolution ($\otimes$) of the Fourier transform $F(f)$ of $f(t)$ (shown in (E)) and the Fourier transform of the train of unit impulses with an interval $F_s = 1/T_s$, as shown in (F). From this diagram it can be appreciated that the width of $F(f)$ should fall within period F_s (i.e., the maximum value of the spectrum of the sampled signal must be less than $F_s/2$) in order to avoid overlap in the spectra (shown in Fig. 2.7). Further details can be found in Appendix 2.1.

Assuming that we can interchange the summation and integral operations, and using the sifting property again, the above expression evaluates to:

$$\sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(t - nT_s) e^{-2\pi ft} dt = \sum_{n=-\infty}^{\infty} e^{-2\pi fnT_s} \tag{2.15}$$

An essential difference between this expression and the Fourier transform of a single δ function, is the summation for n from $-\infty$ to ∞. Changing the sign of the exponent in Eq. (2.15) is equivalent to changing the order of the summation from $-\infty \rightarrow \infty$ to $\infty \rightarrow -\infty$. Therefore we may state:

$$\sum_{n=-\infty}^{\infty} e^{-2\pi fnT_s} = \sum_{n=-\infty}^{\infty} e^{2\pi fnT_s} \tag{2.16}$$

From Eq. (2.16) it can be established that the sign of the exponent in Eqs. (2.13) to (2.16) doesn't matter. Thinking about this a bit: taking into account the similarity between the Fourier transform and the inverse transform integrals (Eqs. 6.4 and 6.8 in Chapter 6), the main difference of the integral being the sign of the exponent, this indicates that the Fourier transform and the inverse Fourier transform of a Dirac comb must evaluate to a similar form. This leads to the conclusion that the (inverse) Fourier transform of a Dirac comb must be another Dirac comb. Given that in the ***time domain*** we have: $\sum_{n=-\infty}^{\infty} \delta(t - nT_s)$, its Fourier transform in the ***frequency domain*** must be proportional to $\sum_{n=-\infty}^{\infty} \delta(f - nF_s)$. In these expressions the sample frequency $F_s = 1/T_s$. If you feel that this "proof" is too informal, please consult Appendix 2.1 for a more thorough approach. You will find there that I am indeed ignoring a scaling factor equal to $1/T_s$ in the above expression (see Eq. (A2.1-7) in Appendix 2.1).

We will not worry about this scaling factor here; because for sample rate issues, we are interested in timing and not amplitude. For now we can establish the relationship between the Fourier transform $F(f)$ of a function $f(t)$ and the Fourier transform of its sampled version. Using the obtained result and Eq. (2.12), we find that the sampled version is proportional to:

$$F(f) \otimes \sum_{n=-\infty}^{\infty} \delta(f - nF_s) \tag{2.17}$$

This result is easiest interpreted by the graphical representation of convolution (Chapter 13 and Appendix 13.1) which is sliding the Dirac comb (Fig. 2.6F) along the Fourier transform $F(f)$ (Fig. 2.6E). At any point in this sliding procedure the impulses in the train sift the value in the Fourier transform $F(f)$. When $F(f)$ lies within the gaps between the individual δ functions, we obtain a periodic function as shown in Fig. 2.6D. This result illustrates the same relationship between sample frequency and highest frequency component in a signal as discussed before. In order for $F(f)$ to fall within the gaps of the δ function train, the highest frequency in signal $f(t)$ must be $<F_s/2$, the Nyquist frequency. If, on the contrary, $F(f)$ does not fall within the gaps of the δ function train there will be an overlap resulting in distortion due to an aliasing effect (Fig. 2.7).

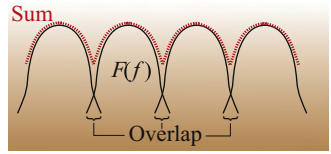


FIGURE 2.7 Equivalent of Fig. 2.6D in the case where the spectra $F(f)$ do not fit within the impulses in the impulse train. This will cause the sum of the individual contributions (red) to include overlap, resulting in an aliasing effect.

2.4 THE MOVE TO THE DIGITAL DOMAIN

Last, but not least, it must be noted that due to the digital revolution most of the functions performed by the analog components of the measurement chain (Fig. 2.1) become redundant or can be moved into the digital domain. With the development of high-resolution ADC, the range of the conversion process becomes large enough that little or no amplification is required in many cases. For example, a 32-bit ADC has a resolution of $2^{32} = 4.295 \times 10^9$ levels. If this is coupled to a 5V range, one can resolve amplitude differences at a 0.23-nV precision without any additional amplification. In addition, high-speed ADC and low-cost storage media allow one to sample so fast that the S/H function is no longer a requirement. The low cost of ADC circuits also allows you to use one converter per data channel, thus eliminating the need for an MUX. Furthermore, faster processors (central processing units, CPUs) and dedicated digital signal processing hardware allow implementation of real-time digital filters that can replace their analog equivalents.

From the above, one might almost conclude that by now we can simply connect an ADC to a biological process and start recording. This conclusion would be wrong since there are two fundamental issues that must be addressed in the analog domain. First, even if the nature of the process is electrical (not requiring a special transducer), there is the impedance conversion issue discussed previously (see Eqs. 2.1 and 2.2). Second, one must deal with the aliasing problem before the input to the ADC. Because most biological processes have a "natural" high-frequency limit, one could argue for omission of the antialiasing step at very high sample rates. Unfortunately, this would make one "blind" to high-frequency artifacts of nonbiological origin, and without subsequent down-sampling it would require huge amounts of storage.

APPENDIX 2.1

In this appendix we address the Fourier transform of a sampled function and investigate the relationship between this transform and the Fourier transform of the underlying continuous time function (see also Section 2.3). The following discussion is attached to this chapter because the topic of sampling logically belongs here. However, a reader who is not yet familiar with Fourier transform and convolution is advised to read this material after studying Chapters 5–8, 13.

We obtain the sampled discrete time function by multiplying the continuous time function with a train of impulses (Eq. 2.5). The Fourier transform of this product is the convolution of the Fourier transform of each factor in the product (Chapter 13), i.e., the continuous time function and the train of impulses. This approach is summarized in Fig. 2.6. In this appendix we will first determine the Fourier transform of the two individual factors; then we will determine the outcome of the convolution.

The transform of the continuous function $f(t)$ will be represented by $F(f)$. The Fourier transform $\Delta(f)$ of an infinite train of unit impulses (Dirac comb) is:

$$\Delta(f) = \int_{-\infty}^{\infty} \underbrace{\sum_{n=-\infty}^{\infty} \delta(t - nT_s)}_{\text{train of unit impulses}} e^{-j2\pi ft} dt \qquad \text{(A2.1-1)}$$

As shown in Section 2.3, we can evaluate this integral by exchanging the order of summation and integration, and by using the sifting property of the δ function for the value nT_s (see Eq. 2.8):

$$\Delta(f) = \sum_{n=-\infty}^{\infty} e^{-j2\pi fnT_s} = \sum_{n=-\infty}^{\infty} e^{j2\pi fnT_s} \qquad \text{(A2.1-2)}$$

Eq. (A2.1-2) shows that the exponent's sign can be changed because the summation goes from $-\infty$ to ∞. First we will consider the summation in Eq. (A2.1-2) as the limit of a summation for $\sum_{n=-N}^{N}$ with $N \to \infty$. Second, we use the Taylor series $\left(\frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots\right)$ of the exponential, i.e.:

$$\frac{1}{1 - e^{j2\pi fT_s}} = 1 + e^{j2\pi fT_s} + e^{2j2\pi fT_s} + e^{3j2\pi fT_s} + \cdots$$

to create and subtract the following two expressions:

$$\frac{e^{-j2\pi fNT_s}}{1-e^{j2\pi fT_s}} = e^{-j2\pi fNT_s} + e^{-j2\pi f(N-1)T_s} + e^{-j2\pi f(N-2)T_s} + \cdots$$

$$= \sum_{n=-N}^{\infty} e^{j2\pi fnT_s} \quad \text{for} \quad -N \to \infty \quad \text{range}$$

$$\frac{e^{j2\pi f(N+1)T_s}}{1-e^{j2\pi fT_s}} = e^{j2\pi f(N+1)T_s} + e^{j2\pi f(N+2)T_s} + e^{j2\pi f(N+3)T_s} + \cdots$$

$$= \sum_{n=N+1}^{\infty} e^{j2\pi fnT_s} \quad \text{for} \quad N+1 \to \infty \quad \text{range}$$

$$\frac{e^{-j2\pi fNT_s} - e^{j2\pi f(N+1)T_s}}{1-e^{j2\pi fT_s}} = \sum_{n=-N}^{N} e^{j2\pi fnT_s} \quad \text{for} \quad -N \to N \quad \text{range}$$

(A2.1-2)

Eq. (A2.1-3) is an expression similar to Eq. (A2.1-2), except for the range of summation from $-N$ to N instead of $-\infty \to \infty$. Subsequently, we multiply both the numerator and denominator in Eq. (A2.1-3) by $e^{-j2\pi fT_s/2}$ and use the Euler relationships $e^{jx} = \cos x + j \sin x$ and $e^{-jx} = \cos x - j \sin x$ to rewrite Eq. (A2.1-3) as follows:

$$= \frac{e^{-j2\pi f(N+1/2)T_s} - e^{j2\pi f(N+1/2)T_s}}{e^{-j2\pi fT_s/2} - e^{j2\pi fT_s/2}} = \frac{\sin[(N+1/2)2\pi fT_s]}{\sin[2\pi fT_s/2]}$$

$$= \frac{\sin[(N+1/2)2\pi fT_s]}{\pi f} \frac{\pi f}{\sin[2\pi fT_s/2]}$$

First we will show that the above expression is a periodic function with period $F_s = 1/T_s$. We substitute $f = f + F_s$ for $f + 1/T_s$ in $\frac{\sin[(N+1/2)2\pi fT_s]}{\sin[2\pi fT_s/2]}$ and obtain:

$$\frac{\sin[(N+1/2)2\pi(f+1/T_s)T_s]}{\sin[2\pi(f+1/T_s)T_s/2]} = \frac{\sin[(N+1/2)2\pi fT_s + (N+1/2)2\pi]}{\sin[2\pi fT_s/2 + \pi]}$$

Because a sine function is periodic over 2π, and N is an integer, we observe that both numerator and denominator are sine functions augmented by π; using $\sin(x+\pi) = -\sin(x)$, we then obtain:

$$= \frac{-\sin[(N+1/2)2\pi fT_s]}{-\sin[2\pi fT_s/2]} = \frac{\sin[(N+1/2)2\pi fT_s]}{\sin[2\pi fT_s/2]}$$

this is the same result as the expression we started with. Therefore, the expression is periodic for $1/T_s$.

Second, the expression must be taken to the limit for $N \to \infty$ in order to obtain the equivalent of Eq. (A2.1-2). First, we split the equation above into two factors. For $N \to \infty$ the first factor can be written as $\delta(f)$, i.e.:

$$\lim_{N\to\infty} \frac{\sin[(N+1/2)2\pi f T_s]}{\pi f} \frac{\pi f}{\sin[2\pi f T_s/2]} = \delta(f) \frac{\pi f}{\sin[2\pi f T_s/2]} \qquad \text{(A2.1-4)}$$

We already know that the expression in Eq. (A2.1-4) is periodic over an interval $F_s = 1/T_s$, therefore we can evaluate the behavior of Eq. (A2.1-4) between $-F_s/2$ and $F_s/2$. The δ function is 0 for all $f \neq 0$, therefore we must evaluate the second term in Eq. (A2.1-4) for $f \to 0$. Using l'Hôpital's rule (differentiate the numerator and denominator, and set f to zero) we find that the nonzero value between $-F_s/2$ and $F_s/2$, for $f = 0$ is:

$$\frac{\pi}{(2\pi T_s/2)\cos[2\pi f T_s/2]} = \frac{1}{T_s}$$

Combining this with Eq. (A2.1-4), we obtain:

$$\frac{1}{T_s}\delta(f) \qquad \text{(A2.1-5)}$$

This outcome determines the behavior in the period around 0, because the expression in Eq. (A2.1-5) is periodic with a period of $F_s = 1/T_s$, we may include this in the argument of the δ function and extend the result above to read:

$$\frac{1}{T_s}\sum_{n=-\infty}^{\infty} \delta(f - nF_s) \qquad \text{(A2.1-6)}$$

Combining Eqs. (A2.1-1) and (A2.1-6) we may state that:

$$\sum_{n=-\infty}^{\infty} \delta(t - nT_s) \Leftrightarrow \frac{1}{T_s}\sum_{n=-\infty}^{\infty} \delta(f - nF_s) \qquad \text{(A2.1-7)}$$

The expressions to the right and left of the $\Leftrightarrow$ in Eq. (A2.1-7) are the time and frequency domain representations of the train of impulses shown in Fig. 2.6B and F.

Finally we return to the original problem of the sampled version of continuous wave $f(t)$ and its Fourier transform $F(f)$. The Fourier transform of the sampled function is the convolution of the Fourier transforms of $f(t)$ with the transform of the train of impulses, i.e.:

$$F(f) \otimes \frac{1}{T_s}\sum_{n=-\infty}^{\infty} \delta(f - nF_s) = \frac{1}{T_s}\int_{-\infty}^{\infty} F(y) \sum_{n=-\infty}^{\infty} \delta(f - nF_s - y)dy$$

The expression after the equal sign is the convolution integral (Chapter 13). Assuming we can interchange the summation and integration:

$$\frac{1}{T_s}\sum_{n=-\infty}^{\infty}\int_{-\infty}^{\infty}F(y)\delta(f-nF_s-y)dy$$

The δ function is even (Appendix 5.1) and may be written as $\delta[y-(f-nF_S)]$. Using the sifting property of the δ function (Eq. 2.8) the integral above evaluates to $F(f-nF_s)$. Finally, we can relate the Fourier transforms of a continuous wave and its sampled version as follows:

$$f(t)\Leftrightarrow F(f) \quad \text{and}$$
$$f(t) \text{ sampled at rate } F_S=1/T_S\Leftrightarrow\frac{1}{T_s}\sum_{n=-\infty}^{\infty}F(f-nF_s) \tag{A2.1-8}$$

The relationship in Eq. (A2.1-8) is depicted in Fig. 2.6. Compare the continuous transform pair Fig. 2.6A and E with the sampled equivalent in Fig. 2.6C and D.

EXERCISES

2.1 Explain the role of a preamplifier.

2.2 What is the sifting property of the delta function?

2.3 Evaluate the following integrals

a. $\int_{-\infty}^{\infty}e^t\delta(t)dt$

b. $\int_{-\infty}^{\infty}\sin(2\pi\omega t)\delta(t)dt$

c. $\int_{-\infty}^{\infty}\cos(2\pi\omega t)\delta(t)dt$

d. $\int_{-\infty}^{\infty}at\delta(t)dt$

e. $\int_{-\infty}^{\infty}(at^4-t^3)\delta(t)dt$

2.4 Evaluate the same integrals in 2.3 after replacing $\delta(t)$ with $\delta(t-2)$.

2.5 What is aliasing?

References

Ingle, V.K., Proakis, J.G., 1997. Digital Signal Processing Using MATLAB V.4. PWS Publishing Company, Boston.

Wallisch, P., Lusignan, M.E., Benayoun, M.D., Baker, T.I., Dickey, A.S., Hatsopoulos, N.G., 2014. MATLAB for Neuroscientists: An Introduction to Scientific Computing in MATLAB, second ed. Academic Press, Amsterdam.
An excellent introduction to the use of MATLAB with many simple as well as advanced examples.

CHAPTER

3

Noise

3.1 INTRODUCTION

The noise components of a signal can have different origins. Sometimes noise is man-made, e.g., artifacts from switching instruments or 60-Hz hum originating from power lines. Other noise sources are random in nature, such as thermal noise originating from resistors in the measurement chain. Random noise is intrinsically unpredictable but it can be described by statistics. From a measurement point of view we can have noise that is introduced as a result of the measurement procedure itself, either producing ***systematic bias*** (e.g., measuring the appetite after dinner) or random ***measurement noise*** (e.g., thermal noise added by recording equipment). If we consider a measurement M as a function of the measured process x and some additive noise N, the i-th measurement can be defined as:

$$M_i = x_i + N_i \tag{3.1}$$

An example with $x_i = 0.8x_{i-1} + 3.5$ plus the noise contribution drawn from a random process is shown in Fig. 3.1A. This trace was produced by pr3_1.m.

Alternately, noise may be intrinsic to the process under investigation. This ***dynamical noise*** is not an independent additive term associated with the measurement, but instead interacts with the process itself. For example, temperature fluctuations during the measurement of cellular membrane potential not only add unwanted variations to the voltage reading, they physically influence the actual processes that determine the potential. If we consider appropriately small time steps, we can imagine the noise at one time step contributing to a change in the state at the next time step. Thus one way to represent dynamical noise D affecting process x is:

$$x_i = [0.8\, x_{i-1} + 3.5] + D_{i-1} \tag{3.2}$$

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00003-5

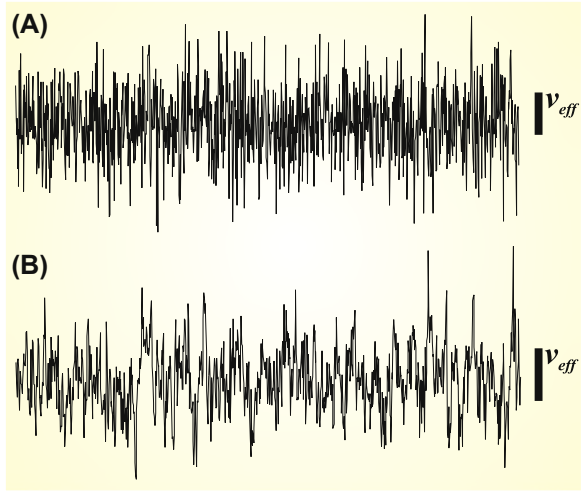


FIGURE 3.1 Time series including measurement noise (A) and a combination of dynamical and measurement noise (B).These examples were generated with MATLAB® scripts pr3_1 and pr3_2. The bars on the right size represent v_{eff} level for each signal (Eq. 3.14).

The process in Eq. (3.2) can be combined with a measurement function such as Eq. (3.1). Comparing the time series of such a process (Fig. 3.1B, generated by pr3_2.m) with the one generated by Eq. (3.1), you can see that the dynamical noise (due to the correlation between sequential values) creates slower trends when compared to the time series with only additive noise. It must be noted here that in many cases, a dynamic noise term is used to represent a random process simply because often we do not know all of the details necessary to accurately represent the entire range of complex interactions in a physiological system. In this sense, the random process compensates for our lack of detailed knowledge by giving us a statistical proxy for what we don't know about the system. As we will see in the discussion of nonlinear dynamics (Chapter 27), ***deterministic*** processes (processes in which the state is ***determined*** by the past) can produce signals with a random aspect, i.e., in some cases the difference between the behavior of a random number generator and a deterministic process can become fuzzy. These processes are similar to the bouncing balls in a lotto drawing; while the outcome is ultimately the result of completely deterministic physical laws, the exact result is entirely unpredictable.

Note: The process in Eq. (3.1) is deterministic, only its measurement is corrupted by noise. However, although the process in Eq. (3.2) includes a deterministic component, it is a so-called stochastic process because a noise component is part of the process itself.

3.2 NOISE STATISTICS

One common way to characterize a random process is by its ***probability density function (PDF)***, describing the probability $p(x)$ that particular values of $x(t)$ occur. For instance, if we create a function to describe the probability of each outcome of a fair roll of a single die, we would have the possible observations 1, 2, 3, 4, 5, 6. In this case, each of the six possible observations occurs with a probability $p(1)$, $p(2)$, … … , $p(6)$, each equal to 1/6. This would result in a PDF that is 1/6 for each of the values 1 through 6 and 0 for all other values. The PDF for the fair die is shown in Fig. 3.2A. This example can be extended to continuous variables, and such an example of a variable that ranges between 0 and 6 is shown in Fig. 3.2B. In this example all values within the range are equally likely to occur. Often this is not the case; the most well-known PDF is the normal distribution shown in Fig. 3.2C, reflecting a process

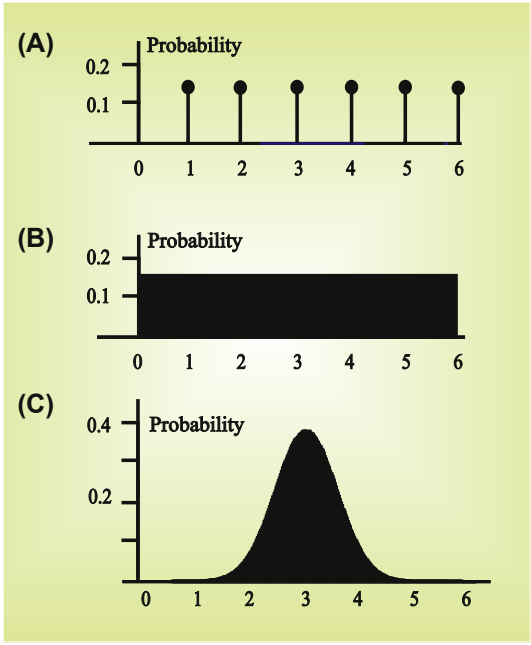


FIGURE 3.2 Probability density functions (PDF) of random processes. (A) The PDF of a die where each of the outcomes 1−6 is equally likely. (B) A similar uniform distribution for a continuous process. An example of such a process is quantization noise caused by analog-to-digital conversion (see Section 3.4(4)). (C) The normal distribution, where probabilities are not uniform across the domain. Values close to the mean are more likely to occur as compared to more extreme values. In this example the mean of the normal distribution is 3, while standard deviation and variance are both equal to 1.

where most values are close to the mean and extreme values (either positive or negative) are less likely to occur.

> *Note*: The correct name for the function describing the probability function of a discrete random variable is the ***probability mass function (PMF)***. In this text we use the term probability density function both in the case of discrete and continuous random variables.

In general, a PDF characterizes the probabilities of all possible outcomes of a random event. So, with multiple possible outcomes, the sum of their probabilities must equal 1, and probability values are therefore fractions <1. In the case of the single die the total is:

$$p(1)+p(2)+p(3)+p(4)+p(5)+p(6)=\sum_{i=1}^{6}p(i)=1,\ \text{ with }\ p(i)=1\div 6.$$

In the case of continuous random variables we replace the summation by an integral over the domain of x, which translates intuitively into the requirement that the area under the PDF must equal 1. In the case of a continuous uniform distribution as in Fig. 3.2B, we integrate over the domain 0 to 6, i.e., $\int_0^6 p(x)dx=1$. More generally, as in the example in Fig. 3.2C, we consider a domain from $-\infty$ to ∞:

$$\int_{-\infty}^{\infty} p(x)dx=1 \tag{3.3}$$

Two useful variations on the PDF can be derived directly from it: the ***cumulative*** $F(x)$ and ***survival*** $\mathscr{F}(x)$ functions are defined as:

$$F(x)=\int_{-\infty}^{x} p(y)dy \tag{3.4}$$

$$\mathscr{F}(x)=1-F(x)=\int_{x}^{\infty} p(y)dy \tag{3.5}$$

As can be inferred from the integration limits in Eqs. (3.4) and (3.5), the cumulative function $(-\infty, x)$ represents the probability that the random variable is $\leq x$, and the survival function (x, ∞) represents the probability that the random variable is $>x$.

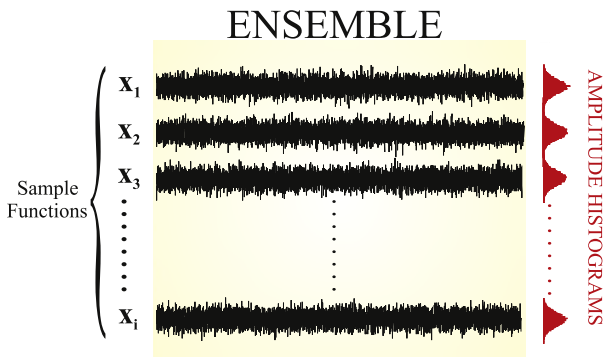


FIGURE 3.3 Observations of the random process characterized by the PDF shown in Fig. 3.2C. Sample functions are individual "samples" from the larger ensemble. For each trace the amplitude distribution histogram is shown on the side in red. To present amplitude in both the sample functions and histograms along the same axis, the orientation of the amplitude distribution histogram is rotated 90 degrees from that used in Fig. 3.2C (i.e., the vertical axis of this distribution corresponds to the range of amplitude values and the horizontal axis to the number of times this amplitude was present in the associated sample function).

If one observes a random process over time, one can obtain sample functions, series of measured values representing one instance of the random process (Fig. 3.3). A collection of these sample functions forms an ***ensemble***. The random process is called ***stationary*** if the distribution from which $x(t)$ originated doesn't change over time. In Fig. 3.3, the amplitude distribution is shown for each sample function. The similarity of these distributions makes the assumption of underlying stationarity a reasonable one. The process is ***ergodic*** if any of the particular sample functions is representative of the whole ensemble, thus allowing statistics to be obtained from averages over time. When applying signal processing techniques, the stationarity and ergodicity of signals are frequently (and often implicitly) assumed, and many techniques can be useful, even when these assumptions are not strictly met. Other, less stringent, definitions for both terms also exist (Appendix 3.1).

Two common parameters that are estimated from random processes are mean and variance. If a process is stationary and ergodic, one can characterize the distribution using any of the sample functions (Fig. 3.1), e.g., the ***estimate*** of the mean of x over an interval T is:

$$\widehat{x} = \frac{1}{T}\int_0^T x(t)dt \tag{3.6}$$

or for a discrete-valued signal over N points:

$$\widehat{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3.7}$$

Similarly one can compute the estimate of the variance from the time series ($\widehat{Var(x)}$):

$$\widehat{Var(x)} = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - \widehat{x}\right)^2 \tag{3.8}$$

To obtain a nonbiased estimate of the variance with small samples, $N-1$ instead of N is used in the denominator of the scaling term. In the above approach to estimating statistics from a sample of an ergodic process, a value close to the **true mean** $\boldsymbol{\mu}$ is obtained as the interval T extends toward infinity: i.e., $\mu = \lim_{T\to\infty} \frac{1}{T}\int_0^T x(t)dt$.

A different approach to obtaining the true mean and standard deviation is via the PDF of the observed variable x, using the **expectation** $\boldsymbol{E\{x\}}$:

$$E\{x\} = \int_{-\infty}^{\infty} x\, p(x)dx = \mu \tag{3.9}$$

Appendix 3.4 gives an example to make this equation plausible. In general one can use the expectation to obtain the n-th ***moment*** of the distribution:

$$E\{x^n\} = \int_{-\infty}^{\infty} x^n p(x)dx \tag{3.10}$$

or the ***n*-th central moment**:

$$E\{(x-\mu)^n\} = \int_{-\infty}^{\infty} (x-\mu)^n p(x)dx \tag{3.11}$$

The first moment is the ***mean*** ($\boldsymbol{\mu}$), the second central moment is the ***variance*** ($\boldsymbol{\sigma^2}$), and the square root of the variance is the ***standard deviation*** ($\boldsymbol{\sigma}$). The square root of the variance of the estimate of the mean is the ***standard error of the mean*** (***SEM***; see Chapter 4). The first central moment of a joint distribution of two variables x and y is the ***covariance***, i.e., $E\{(x-\mu_x)(y-\mu_y)\}$, with μ_x and μ_y the mean values of x and y.

Note: The Laplace and Fourier transforms of the PDFs are sometimes used to generate the moments of the distribution (Appendix 3.5).

3.3 SIGNAL-TO-NOISE RATIO

Generally, any (biomedical) measurement will necessarily be corrupted by some noise. Even if the process itself were noise-free, the measurement chain adds noise components because all analog instruments (amplifiers, analog filters) add, at the very least, a small amount of thermal noise (e.g., Eq. 3.1). If the noise component is sufficiently small compared to the signal component, one can still gather reasonable measurements of the signal. To quantify this ratio between signal and noise components, one can (in some cases) determine the amplitude or the power of each component and from those calculate a ***signal-to-noise ratio***. In discrete time series, the **power** can be measured as the mean squared amplitude $\left(\boldsymbol{ms},\ \frac{1}{N}\sum_{i=1}^{N} x_i^2\right)$ and the **amplitude** as the root of the mean squared amplitude $\left(\boldsymbol{rms},\ \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}\right)$. Analytical equivalents for continuous time series are $ms = \frac{1}{T}\int_0^T x(t)^2 dt$ and the *rms* is $\sqrt{\frac{1}{T}\int_0^T x(t)^2 dt}$. To establish the signal-to-noise ratio (*SNR*), one can use $\frac{ms(signal)}{ms(noise)}$ directly; however, it is more common to represent this ratio on a logarithmic decibel (dB) scale:

$$SNR = 10\log_{10}\frac{ms(signal)}{ms(noise)}\ \text{dB} \tag{3.12}$$

Alternatively, one may start from the *rms* values by substituting $ms = rms^2$ in Eq. (3.12):

$$SNR = 10\log_{10}\left[\frac{rms(signal)}{rms(noise)}\right]^2 = 20\log_{10}\frac{rms(signal)}{rms(noise)}\ \text{dB} \tag{3.13}$$

Note that the dB scale does not have a physical dimension; it is simply the logarithm of a ratio. The signal-to-noise ratio (without the log transform) is sometimes used as a **figure of merit** (*FOM*) by equipment manufacturers. If this ratio is close to 1, or even <1, signal processing can help to increase SNR in special cases.

In technical literature for analog devices, the noise level of $v(t)$ in an interval T is frequently indicated with v_{eff}, which equals the standard deviation of the signal, i.e.,:

$$v_{eff} = \sqrt{\frac{1}{T}\int_0^T (v-\overline{v})^2 dt} \tag{3.14}$$

with $\overline{v}$, the mean of $v(t)$ in the interval.

In the case of a sampled signal, the equivalent would be $\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\overline{x})^2}$, similar to the definition of rms above.

Note: To obtain a better-looking figure for the noise specification, most manufacturers present v_{eff} after it has been corrected for any amplification. For instance, if a 1000× amplifier has 1-mV effective noise, a v_{eff} of 1 μV *at the input* is reported.

For noise with a zero mean, v_{eff} is the square root of $E\{x^2\}$; in this case the difference between v_{eff} and *rms* disappears! It should further be noted that when observing a noise signal on a scope or chart writer, the ***amplitude of the noise band one observes is typically 4–5 times the*** v_{eff} (Fig. 3.1).

The effects of combined noise sources add up geometrically in the total result: the total v_{eff} of two ***independent*** noise sources 1 and 2 in series, such as the noise generated in two connected instruments in a measurement chain, can be found by:

$$v_{eff} = \sqrt{\left(v_{eff,1}^2 + v_{eff,2}^2\right)} \tag{3.15}$$

In MATLAB® you can verify this by creating two random time series (s1 and s2) and the total result (st) by typing the following in the command window:

```
s1 = randn(1000,1);
s2 = randn(1000,1);
st = s1 + s2;
```

You will find that the v_{eff}^2(variance) of st (vt) will be close to the sum of variances of s1 (v1) and s2 (v2); e.g., type:

```
v1 = (std(s1))^2
v2 = (std(s2))^2
vt = (std(st))^2
```

Due to the random aspect of the time series, the outcome of this little numerical experiment will be a bit different each time, but in each case you will find that vt ≈ v1 + v2.

3.4 NOISE SOURCES

In the measurement chain there are several sources of noise, and some of these sources can be extremely annoying for the experimenter. The following summarizes four major sources of noise in the measurement chain discussed in Chapter 2.

1. Thermal or Johnson noise originating from resistors in the circuitry. The value can be estimated by:

$$v_{eff}^2 = 4\,k\,T\,R\,\Delta f \tag{3.16}$$

$k = 1.38 \times 10^{-23}$, T absolute temperature (°K), R resistor value, and Δf bandwidth.

Problem

Calculate v_{eff} of the noise generated by a Giga seal ($10^9\ \Omega$) made between a patch clamp electrode and a neuron. Assume a temperature of 27°C and a recording bandwidth of 10 kHz.

Answer

Using Eq. (3.16) taking into account the conversion from °C into °K (by adding 273) we get:

$$v_{eff}^2 = 4 \times 1.38 \times 10^{-23} \times (27 + 273) \times 10^9 \times 10^4 = 1.6560 \times 10^{-7}\ \mathrm{V}^2$$

Taking the square root of the outcome we find $v_{eff} \approx$ 0.4 mV.

Usually thermal noise is associated with a particular application and it is rarely under direct control in a given setup. There are cases where designers have included cooling of the preamplifier (using a Peltier element as a cooling device) to reduce thermal noise from the input resistors. The usefulness of this approach is limited because the temperature factor in Eq. (3.14) is in °K, where a decrease of 10° only reduces v_{eff} by a few percent.

2. Finding sources of ***(a) electromagnetic*** or ***(b) electrostatic*** noise (usually hum from power lines) can be a frustrating exercise. Generally, noise caused by a fluctuating magnetic field is relatively small (<0.1 mV) and can be avoided by eliminating loops or twisting wires. Some of the basic physics required for this section is summarized in Appendix 1.1. The calculus-challenged reader can consult Appendix 3.2 for the derivatives used in the following examples.

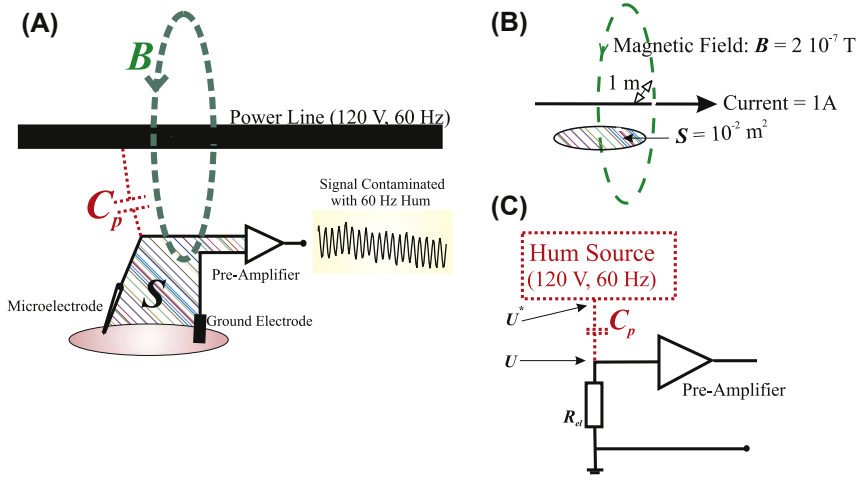


FIGURE 3.4 (A) Electromagnetic noise caused by a power line can be modeled by the effect of a magnetic flux through the surface S formed between the electrodes, and the capacitance C_p between the power line and the input of the preamplifier. (B) Simplified diagram of the magnetic effect in which a magnetic field of 2×10^{-7} T generated by a 1-A current passes through a surface S at 1 m distance. (C) Simplified diagram of the electrostatic effect.

a. ***Electromagnetic***. In this example we consider the effect of a magnetic field that is associated with a power line current (I) with an amplitude of 1 A, and line frequency of 60 Hz. Such a current generates a magnetic field (B) at 1 m distance (d) with amplitude (Fig. 3.4A and B):

$$B = \frac{\mu}{2\pi}\frac{I}{d} = 2\ 10^{-7} \quad \text{T(Tesla)} \tag{3.17}$$

using the magnetic permeability value for vacuum $\mu_0 = 4\pi \times 10^{-7}$. For a loop enclosing 10^{-2} m^2 and assuming (to simplify the example) that the magnetic field's orientation is perpendicular to the surface area S enclosed by the loop, this translates into a flux:

$$\Phi_B = B\,S = 2 \times 10^{-9} \sin(2\pi 60t) \quad \text{Wb (Weber)}$$

Calculating the amplitude of the potential difference in the loop (V) from the derivative of the flux (Appendices 1.1 and 3.2) generates:

$$V = \frac{d\Phi_B}{dt} = 2 \times 10^{-9} 2\pi\ 60 \cos(2\pi 60t) \approx \pm 0.75\ \mu\text{V}$$

To calculate the amplitude of the noise in the above equation, we only consider the extreme values (± 1) of $\cos(2\pi\ 60\ t)$. Thus, the v_{eff} of this sinusoidal signal (Appendix 3.3) is $\approx 0.53\ \mu V$.

b. ***Electrostatic***. The same power line producing the electromagnetic interference characterized above (Fig. 3.4) also has an electrostatic effect on the input circuitry of the preamplifier. We represent the AC power line as a hum source (U^*, Fig. 3.4C) of 120 V at 60 Hz close to the preamplifier input. The input is also connected to a 10 MΩ (1 MegaOhm $= 10^6\ \Omega$) resistance (R_{el}) representing the microelectrode. The conductors of the front end in this setup form a capacitance with conductors that carry the noise signal, the so-called parasitic capacitance. This parasitic capacitance C_p is typically very small, on the order of 10 fF (1 fF $= 10^{-15}$ F). The current i_c through C_p is the derivative of its charge (Appendix 1.1):

$$i_c = C_p \frac{d(U^* - U)}{dt} \quad \text{with :} \quad U^* = 120 \sin(2\pi 60 t) \tag{3.18}$$

Considering that $U^* >> U$, we can simplify the above to the following approximation:

$$i_c \approx C_p \frac{dU^*}{dt} \tag{3.19}$$

At the level of the preamplifier's input, the effect of current i_c on the input potential is:

$$U = i_c R_{el} \approx R_{el} C_p \frac{dU^*}{dt} \tag{3.20}$$

Here we only consider the effect of i_c on the measured potential U. Because we are interested in the noise component we can ignore any other sources at the preamplifier's input. The derivative (Appendix 3.2) in the above expression is:

$$2\pi 60 \times 120 \underbrace{\cos(2\pi 60 t)}_{\pm 1} \approx \pm\ 4.5\ 10^4$$

This outcome, multiplied by $R_{el}C_p = 10^{-7}$, results in a noise amplitude of ± 4.5 mV. The v_{eff} of this sinusoidal signal (Appendix 3.3) is therefore ≈ 3.2 mV.

As shown in the examples above, hum from an electrostatic noise source is usually much larger than the electromagnetic component. This

electrostatic noise must be eliminated by shielding or removing the source.

3. In addition to the noise added by passive components such as resistors, active elements also add noise. Therefore the application of low-noise amplifiers "early" in the chain (before major amplification steps) is desirable. Typically an active component will add 1–100 μV of noise.
4. The discretization error made at the ADC can also be considered a noise source, the so-called quantization noise. The level of this noise depends on the range and the resolution of the ADC. Assuming an ADC that truncates the sample values: all values between 0 and 1 become 0, values between 1 and 2 become 1, etc. This imprecision is exactly one unit (i.e., the precision) of the analog-to-digital converter and this applies for the whole range of the converter. The occurrence of truncation errors within the ADC precision can be depicted as a probability density distribution for the added noise. For the sake of this example let's use an A/D precision of q μV (1 μV $= 10^{-6}$ V), we will obtain a uniform distribution (as in Fig. 3.2B where $q = 6$) if we assume that the signal we sample is equally likely to occur anywhere within each of the units of the ADC. This is a fairly reasonable assumption since we sample a continuous signal and the ADC steps are relatively small. Knowing the PDF of the noise we can obtain the v_{eff}—that is, the standard deviation of the noise PDF (see Eq. 3.14), by calculating the square root of $E\{(x - \mu)^2\}$ (Eq. 3.11). First we obtain $E\{x\} = \mu$ using Eq. (3.9):

$$E\{x\} = \mu = \int_{-\infty}^{\infty} x\, p(x) dx \tag{3.21}$$

We can change the integration limits from $[-\infty, \infty]$ to $[0, q]$, because outside this domain $p(x) = 0$ and inside $p(x) = 1/q$:

$$\int_{-\infty}^{\infty} x\, p(x) dx = \int_{0}^{q} x \underbrace{p(x)}_{\frac{1}{q}} dx \tag{3.22}$$

Because $1/q$ is a constant and we are integrating with respect to x (Appendix 3.2) the above expression equates to:

$$= \frac{1}{q} \int_{0}^{q} x\, dx = \frac{1}{q} \left[\frac{1}{2} x^2 \right]_0^q = \frac{q}{2}\ \mu\text{V} \tag{3.23}$$

Of course we could have seen by inspection of the example of Fig. 3.2B where $q = 6$ that the mean $= q/2 = 3$. Subsequently we use $\mu = q/2$ and $p(x) = 1/q$ between 0 and q in Eq. (3.11):

$$E\left\{(x-\mu)^2\right\} = \int_{-\infty}^{\infty} \left(x - \underbrace{\mu}_{\frac{q}{2}}\right)^2 \underbrace{p(x)}_{\frac{1}{q}}\, dx = \int_0^q \left(x - \frac{q}{2}\right)^2 \frac{1}{q} dx \tag{3.24}$$

Because $1/q$ is a constant and using $(A-B)^2 = A^2 - 2AB + B^2$ we obtain:

$$= \frac{1}{q}\int_0^q \left(x^2 - qx + \frac{q^2}{4}\right) dx \tag{3.25}$$

Evaluating the integral (Appendix 3.2):

$$= \frac{1}{q}\left[\frac{1}{3}x^3 - \frac{1}{2}qx^2 + \frac{q^2}{4}x\right]_0^q = \frac{q^2}{12}\ \mu\text{V}^2 \tag{3.26}$$

The value v_{eff} is then the square root of this variance term, i.e., $v_{eff} = \sqrt{\frac{q^2}{12}}\,\mu\text{V}$.

In state-of-the-art electrophysiology equipment, quantization noise is a few microVolts (μV) or less. It is not uncommon to use at least a 12-bit converter. Taking into account a measurement chain with an amplification of 1000× and an analog ADC input range of 10 V (±5 V) we obtain an analog range of $10/1000 = 0.01$ V $= 10$ mV at the amplifier's input. This results in quantization noise values on the order of microvolts. In this example $q = \frac{10}{2^{12}} = 0.0024$ mV $= 2.4$ μV. This then results in $v_{eff} = \sqrt{\frac{2.4^2}{12}} \approx 0.7$ μV.

In the above example we evaluated the effect of a truncation at the conversion step. If we consider a converter that rounds instead of truncates, the noise characteristics are similar because the PDF (such as the one shown in Fig. 3.2B) only shifts to the left (zero mean). The shape of the distribution, its range, and (consequently) the standard deviation remain unaltered.

From the results shown in the examples above, it may be clear that with modern equipment, low noise recordings are indeed feasible. However, often the amplitude of the noise is comparable to the amplitude of different types of biopotentials (Fig. 3.5), indicating that strategies for

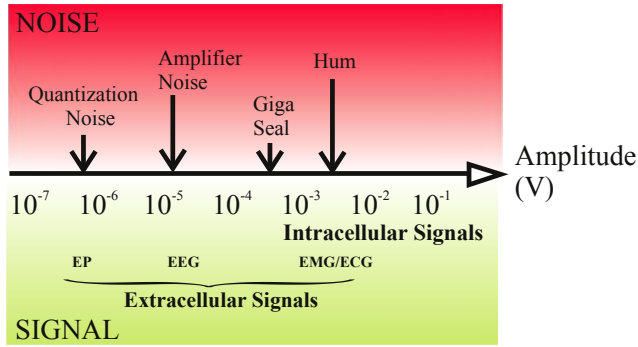


FIGURE 3.5 Overview of the amplitude of typical biopotentials and different types of noise.

noise reduction are required. Enemy #1 in any recording of biopotentials (or low-level transducer signals with similar amplitudes) is hum. As we will see, hum as a nonrandom noise source may even play a role in spoiling signal averaging results.

APPENDIX 3.1

1. Less strict definitions of stationarity and ergodicity exist. A random process with a mean that is time invariant and an autocorrelation function (Chapter 13) that is only dependent on time lag τ is called a ***wide sense stationary*** process. For ergodicity one may also use more relaxed definitions: e.g., a random stationary process is ***ergodic in the mean*** if at least the mean can be estimated with a time average of a sample function.
2. Because the sample functions from an ergodic process are statistically equivalent, ***an ergodic process is stationary*** and, although there are exceptions, a stationary process will usually also be ergodic. A somewhat trivial example of such an exception is sample function $x(t) = |Y \sin(2\pi(t + Y))|$, in which Y is selected randomly from the same PDF but selection occurs only once for each sample function.
3. A thorough discussion of stationarity and ergodicity is beyond the scope of this text, and for measured time series we will use the labels stationary and ergodic as a fancy way to state optimistically that we believe that our signal at hand allows us to estimate relevant statistics from time averages. In signal processing literature it is not uncommon to select sample epochs that seem stationary and representative of the signal as a whole, and to use this "Gestalt" as a reason to declare (explicitly or implicitly)

stationarity and ergodicity. Strict tests to provide proof of these assertions are not available because we are not old enough and don't live long enough to observe a process from $-\infty$ to ∞. Operational definitions for more reasonable time spans do exist, but in practice such tests are rarely used to justify stationarity and/or ergodicity assumptions.

APPENDIX 3.2

In this book it is assumed that the student is familiar with basic calculus. To refresh your knowledge of differentiation and integration, see Boas (1966), Jordan and Smith (1997), or any textbook on these mathematical techniques. This appendix provides a quick reference for those who need a reminder of the most common equations that are used throughout the text (Table A3.2-1).

Useful rules are:

1. The ***chain rule*** is used when differentiating a function $f(u)$ with $u = u(t)$:

$$\frac{df}{dt} = \frac{df}{du}\frac{du}{dt} \tag{A3.2-1}$$

For example the, derivative of $\sin(at)$ with $u = at$ is:

$$\left.\begin{aligned} \frac{df}{du} &= \frac{d\sin(u)}{du} = \cos(u) = \cos(at) \\ \frac{du}{dt} &= \frac{d(at)}{dt} = a \end{aligned}\right\} \rightarrow \frac{d[\sin(at)]}{dt} = \frac{df}{du}\frac{du}{dt} = a\cos(at)$$

TABLE A3.2-1 Derivatives and Integrals of Commonly Used Functions

***f*(function)**	**df/*dt* (Derivative)**	$\int f\,dt$ (Integral)
a (a constant)	0	$ax + C$
x^n **for** $n \neq -1$	nx^{n-1}	$\frac{1}{n+1}x^{n+1} + C$
x^{-1}	$-1x^{-2}$	$\ln(x) + C$
e^x	e^x	$e^x + C$
$\sin(x)$	$\cos(x)$	$-\cos(x) + C$
$\cos(x)$	$-\sin(x)$	$\sin(x) + C$

2. Differentiation and integration by parts for function $f = uv$.
   a. ***Differentiation*** (here we use the notation f' for the derivative):

$$\frac{df}{dt} = f' = (uv)' = uv' + u'v \tag{A3.2-2}$$

For instance, differentiate $f = 3x\,e^{ax}$. Using the above approach we have:

$$u = 3x \quad \text{and} \quad v = e^{ax}$$

Getting the differentials required:

$$u' = 3 \quad \text{and} \quad v' = a\,e^{ax}$$

Substituting this into Eq. (A3.2-2), we obtain the solution for the differential:

$$f' = uv' + u'v = 3ax\,e^{ax} + 3\,e^{ax} = 3\,e^{ax}(1 + ax)$$

   b. ***Integration***:

$$\int udv = uv - \int vdu \tag{A3.2-3}$$

We integrate the same function as above: $f = 3x\,e^{ax}$. Using integration by parts we have:

$$u = 3x \quad \text{and} \quad dv = e^{ax}dx$$

Getting the other expressions required:

$$du = 3dx \quad \text{and} \quad v = \frac{1}{a}e^{ax}$$

Substituting this into Eq. (A3.2-3):

$$\int 3x\,e^{ax}dx = uv - \int vdu = 3x\frac{1}{a}e^{ax} - \int \frac{1}{a}e^{ax}3\,dx$$

$$= \frac{3}{a}x\,e^{ax} - \frac{3}{a}\int e^{ax}dx = \frac{3}{a}x\,e^{ax} - \frac{3}{a}\left[\frac{1}{a}e^{ax}\right] + C = \frac{3}{a}e^{ax}\left[x - \frac{1}{a}\right] + C$$

As we can see, the above approach works well in this example because evaluation of $\int vdu$ is easier than the integral of $\int udv$.

APPENDIX 3.3

The v_{eff} of a sinusoidal signal with amplitude A can be calculated with Eq. (3.14). Consider a sine wave with frequency f (=$1/T$) for n periods, i.e., a time interval equal to nT:

$$v_{eff}^2 = \frac{1}{nT}\int_0^{nT} A^2 \sin^2(2\pi ft)\, dt \tag{A3.3-1}$$

In the above expression we assumed that the sine wave fluctuates around zero (mean = 0). Taking the constant A^2 out of the integration and using the trigonometric equality:

$$\sin^2\left(\frac{1}{2}\alpha\right) = \frac{1}{2} - \frac{1}{2}\cos(\alpha), \text{ we obtain :}$$

$$v_{eff}^2 = \frac{A^2}{nT}\int_0^{nT}\left(\frac{1}{2} - \frac{1}{2}\cos(4\pi ft)\right)dt \tag{A3.3-2}$$

Separating the terms in the integral:

$$v_{eff}^2 = \frac{A^2}{nT}\underbrace{\int_0^{nT}\left(\frac{1}{2}\right)dt}_{\left[\frac{t}{2}\right]_0^{nT}} - \frac{A^2}{nT}\underbrace{\int_0^{nT}\frac{1}{2}\cos(4\pi ft)dt}_{0} \tag{A3.3-3}$$

Evaluation of the first term in Eq. (A3.3-3) is the integration of a constant (1/2). Because the second term in Eq. (A3.3-3) is the integral of a cosine function over an integer number of periods, the net area enclosed by the wave is zero (see also Fig. 5.3B) and therefore this integral evaluates to zero. Eq. (A3.3-3) evaluates to:

$$v_{eff}^2 = \frac{A^2}{nT}\frac{nT}{2} = \frac{A^2}{2} \tag{A3.3-4}$$

The v_{eff} of a sine wave over a full number of periods is therefore equal to:

$$\sqrt{\frac{A^2}{2}} = \frac{A}{2}\sqrt{2} \approx 0.71A.$$

APPENDIX 3.4

In this example, we investigate the logic underpinning Eq. (3.9) that describes how to use the ***expectation*** formalism for the statistics of a distribution. Eq. (3.9) is repeated here for convenience: $E\{x\} = \int_{-\infty}^{\infty} x\, p(x)dx = \mu$.

We now consider a simplified scenario by adapting the formula for a discrete case. Suppose we flip a coin with possible outcomes $x_1 = H$and $x_2 = T$, with associated probabilities p_1 and p_2. Instead of heads and tails, we may use a numerical equivalent for the outcome, e.g., $x_1 = -1$ and $x_2 = 1$. If we now flip N times, or use N coins, we have:

$$E\{x\} = \mu = \frac{(Np_1)x_1 + (Np_2)x_2}{N} = \sum_{i=1}^{2} p_i x_i \qquad \text{(A3.4-1)}$$

In this expression, the summation $\sum_{i=1}^{2} p_i x_i$ is the discrete equivalent of the integral in Eq. (3.9).

Note that the meaning of the word ***expectation*** can be a bit confusing in this context! For example if we have a fair coin, p_1 and p_2 are both ½, and the ***expectation*** according to Eq. (A3.4-1) will be equal to zero. However, when we toss the coin, the outcome will be either $+1$ or -1 and never zero! Thus although the ***expectation*** is zero, we will never **expect** to see an outcome equal to zero.

APPENDIX 3.5 LAPLACE AND FOURIER TRANSFORMS OF PROBABILITY DENSITY FUNCTIONS

In this appendix we explore the use of Laplace and Fourier transforms to facilitate the determination of parameters that are associated with probability density functions (PDFs). If you are not (yet) familiar with the Laplace and Fourier transforms you can skip this part and readdress it later. The Laplace transform is frequently used in statistics to characterize combined processes with different probability density distributions, or to generate the moments of a PDF.

If T is a nonnegative random variable drawn from a PDF $f(t)$ with moments $E(T)$, $E(T^2)$, defined as:

$$E(T^n) = \int_0^{\infty} t^n f(t)dt \qquad \text{(A3.5-1)}$$

Note that the integration is from $0 \rightarrow \infty$, because T is nonnegative, i.e., $f(t) = 0$ for $t < 0$. The Laplace transform of $f(t)$ is:

$$F(s) = \int_0^\infty f(t)e^{-st}dt = E\left(e^{-sT}\right) \tag{A3.5-2}$$

The exponential can be written as a series:

$$F(s) = E\left(1 - \frac{sT}{1!} + \frac{s^2T^2}{2!} - \frac{s^3T^3}{3!} + \frac{s^4T^4}{2!}\ldots\ldots\right)$$

$$F(s) = E\left(\sum_{k=0}^{\infty}(-1)^k\frac{s^kT^k}{k!}\right) = \sum_{k=0}^{\infty}(-1)^k\frac{s^k}{k!}E\left(T^k\right) \tag{A3.5-3}$$

As can be seen in the last expression, the Laplace transform generates the moments $E(T^k)$. Sometimes it is easier to use this property to find the moments of a distribution than to explicitly evaluate the integral in Eq. (A3.4-1), for instance in the exponential distribution associated with a Poisson process (not to be confused with a Poisson distribution, see Chapter 20). The Poisson process $f(t) = \rho\ e^{-\rho t}$ has a Laplace transform $F(s) = \rho/(\rho + s)$. This Laplace transform can be presented as an infinite series:

$$F(s) = \frac{\rho}{\rho + s} = \sum_{k=0}^{\infty}(-1)^k\frac{s^k}{\rho^k} \tag{A3.5-4}$$

Comparing this series with the generic one, we can establish that for the Poisson process: $E(T^k) = k!/\rho^k$, indicating that the mean $E(T) = 1/\rho$ and the variance $\sigma^2 = E(T^2) - E(T)^2 = 2/\rho^2 - (1/\rho)^2 = 1/\rho^2$. Thus for the Poisson process PDF, the mean and standard deviation are both equal to $1/\rho$. A transform very similar to the Fourier transform of a PDF is also used for studying the propagation of noise through a system with a known transfer function. This transform of the PDF is called the ***characteristic function*** $\psi(\omega) = \int_{-\infty}^{\infty} f(t)\ e^{j\omega t}dt$ of the PDF, where the only difference from the Fourier transform is the sign of ω. The characteristic function can also be used to determine the moments of the PDF by taking the derivatives of $\psi(\omega)$: $\frac{d^n\psi(\omega)}{d\omega^n} = \int_{-\infty}^{\infty} j^n t^n f(t)\ e^{j\omega t}dt$. For $\omega = 0$:

$$\left[\int_{-\infty}^{\infty} j^n t^n f(t)\ e^{j\omega t}dt\right]_{\omega=0} = \int_{-\infty}^{\infty} j^n t^n f(t)dt = j^n E(T^n) \tag{A3.5-5}$$

This equation can sometimes make it easier to determine the moments of a distribution from a table of Fourier transforms. Again, we can use the example of the Poisson process $f(t) = \rho\, e^{-\rho t}$ with its characteristic function: $\rho/(\rho - j\omega)$. Note that we used the Fourier transform of the PDF and simply changed the sign of ω. To establish the first moment $n = 1$; the first derivative of this characteristic function is (remember that the derivative of a quotient u/v is $u'v - uv'/v^2$; here $u = \rho$ and $v = \rho - j\omega$):

$$\frac{d\,\psi(\omega)}{d\omega} = \left[\frac{-\rho(-j)}{(\rho - j\omega)^2}\right]_{\omega=0} = \frac{\rho\, j}{\rho^2} = \frac{j}{\rho} \tag{A3.5-6}$$

According to Eq. (A3.5-2) this expression equals $jE(T) \rightarrow$ the expected mean value $E(T) = 1/\rho$.

EXERCISES

3.1 Draw the PMF for flipping three coins simultaneously. We define one sample function (see Fig. 3.3) as flipping 100× and we repeat this every day for a week so that we have seven sample functions.
a. Is this a stationary process?
b. Is it an ergodic process?

3.2 We have a bag with 100 batteries. Although we are told they are 9 V batteries, we know that they are not all exactly 9 V; their precise values are drawn from a Gaussian distribution 8.5 ± 0.6 V (representing the mean ± standard deviation). We randomly pick a battery and record its potential for a minute at a sample rate equal to 10 samples/s. This recording is our sample function (see Fig. 3.3). We repeat the process of picking a battery at random and recording its potential 25 times, so that we get 25 sample functions.
a. Is this a stationary process?
b. Is it an ergodic process?

3.3 A continuous variable is uniformly distributed between 0 and 1.
a. Use Eq. (3.9) to compute the expected value (mean) of this distribution.
b. Use Eq. (3.11) to compute its variance.
Hint: Look into the example of quantization noise in Section 3.4

3.4 Amplification in the analog components of a measurement chain is 1000×. The final output of the analog components is connected to the input of a 12-bit ADC with a 5-V input range.

a. Depict the PDF of the quantization noise at the input of the measurement chain.
b. Characterize the first moment and the second central moment of the PDF.

3.5 What is the difference between v_{eff} and rms?

3.6 MATLAB® rand command
a. Show a graph of the estimated PDF of a noise array produced by MATLAB® command rand.
b. Derive the expressions for the mean and variance of this distribution.
c. Calculate the mean and variance of this distribution both by using these expressions and directly from the data using the mean and std commands.
d. Interpret the results.

References

Boas, M.L., 1966. Mathematical Methods in the Physical Sciences, second ed. John Wiley & Sons, New York.
A textbook on basic mathematical techniques.

Jordan, D.W., Smith, P., 1997. Mathematical Techniques. Oxford University Press, Oxford.
An overview of calculus, meeting requirements in engineering or physics. The book covers differentiation, integration, matrix/vector algebra, differential equations, transforms, discrete mathematics, and statistics.

CHAPTER

4

Signal Averaging

4.1 INTRODUCTION

Data analysis techniques are commonly subdivided into operations in the ***time domain (or spatial domain)*** and ***frequency domain***. In this chapter we discuss processing techniques applied in the time (spatial) domain with a strong emphasis on ***signal averaging***. Signal averaging is an important technique that allows estimation of low-amplitude signals that are buried in noise. The technique usually assumes that:

1. signal and noise are uncorrelated,
2. the timing of the signal is known,
3. a consistent signal component exists when performing repeated measurements, and that
4. the noise is truly random with zero mean.

In the real world all these assumptions may be violated to some degree; however, the averaging technique has proven sufficiently robust to survive minor violations of these four basic assumptions. A brief overview of other frequently used time domain techniques can be found in Section 4.8.

4.2 TIME-LOCKED SIGNALS

Averaging is applied to enhance a time-locked signal component in noisy measurements. One possible representation of such a signal is as measurement x consisting of a signal s and a noise component n, with the underlying assumption that the measurement can be repeated over N trials. In the case where each trial is digitized, the k-th sample-point in the j-th trial (Fig. 4.1) can be written as:

$$x_j(k) = s_j(k) + n_j(k) \tag{4.1}$$

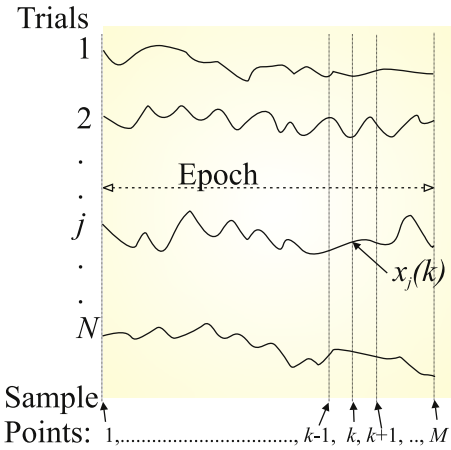


FIGURE 4.1 A set of N raw trials composed of a signal and significant noise component can be used to obtain an average with an enhanced signal-to-noise ratio. Each full epoch consists of a series of individual sample points $x_j(k)$, with $k = 1, 2, \ldots, k, \ldots, M$.

with k being the sample number ($k = 1, 2, \ldots, M$). The *rms* value of s can be several orders of magnitude smaller than that of n, meaning that the signal component may be invisible in the raw traces. After completion of N repeated measurements, we can compute an average measurement for each of the k sample indices:

$$x(k)_N = \frac{1}{N}\sum_{j=1}^{N} x_j(k) = \frac{1}{N}\sum_{j=1}^{N}\left[s_j(k) + n_j(k)\right] \tag{4.2}$$

The series of averaged points (for k from 1 to M) obtained from Eq. (4.2) constitutes the average signal of the whole epoch. In the following we explore some of the properties of signal averaging in a simulation.

The following MATLAB® routine pr4_1.m is a simulation of the averaging process.

```
% pr4_1
% averaging
clear

sz=256;
NOISE_TRIALS=randn(sz);              % a [sz × sz] matrix filled with noise
```

```
SZ=1:sz;                                % Create signal with a sine wave
SZ=SZ/(sz/2);                           % Divide the array SZ by sz/2
S=sin(2*pi*SZ);

for i=1:sz;                                     % create a noisy signal
   NOISE_TRIALS(i,:) = NOISE_TRIALS(i,:) + S;
end;

average=sum(NOISE_TRIALS)/sz;                   % create the average
odd_average=sum(NOISE_TRIALS(1:2:sz,:))/(sz/2);
even_average=sum(NOISE_TRIALS(2:2:sz,:))/(sz/2);
noise_estimate=(odd_average-even_average)/2;

figure
hold
plot(NOISE_TRIALS(1,:),'g')
plot(noise_estimate,'k')
plot(average,'r')
plot(S)
title('Average RED, Noise estimate BLACK; Single trial GREEN, Signal BLUE')
```

As shown in the simulation result depicted in Fig. 4.2, the averaging process described by Eq. (4.2) results in an estimate of the signal. As compared with the raw (signal + noise) trace in Fig. 4.2, the averaged noise component is reduced over 256 trials. When averaging real signals, the underlying component may not always be as clear as it is in the example provided in Fig. 4.2. In these cases the averages are often repeated in search of consistent components in two or three replicates (e.g., see the superimposed SEP waveforms in Fig. 1.4). The idea here is that it is unlikely that two or more consistent averaged results will be produced by chance alone. A specific way of obtaining replicates is to average all ***odd*** and all ***even*** trials in separate buffers (see the superimposed odd_average and even_average in Fig. 4.2). This has the advantage of allowing for comparison of the even and odd results from interleaved trials. An average of the odd and even averages (i.e., addition of the odd and even results divided by 2) generates the complete averaged result, while the difference of the two constitutes an estimate of the noise (see Section 4.4 for details on such a noise estimate).

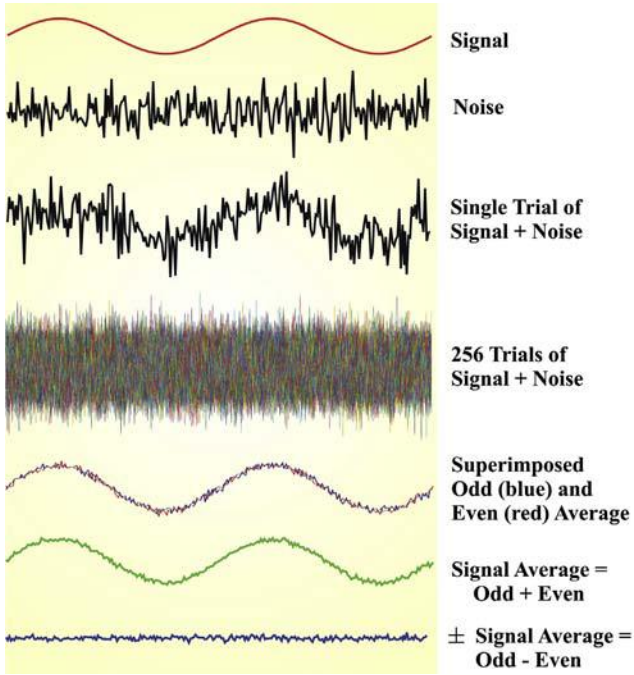


FIGURE 4.2 Signal averaging of a signal buried in noise (Signal + Noise). This example shows 256 **superimposed** trials of such a measurement and the average thereof. The average results of the odd and even trials are shown separately. The sum of all trials (Signal Average) shows the signal with a small component of residual noise. A ± signal average is shown as an estimate of residual noise. These example traces are generated with MATLAB® script pr4_1. An example of a recorded average of the brain's response to electrical nerve stimulation is depicted in Fig. 1.3.

4.3 SIGNAL AVERAGING AND RANDOM NOISE

The noise in Eq. (4.2) is a 0-mean random process: $\langle x(k)\rangle = \langle s(k)\rangle + 0$. Here, $< \ldots >$ denotes the average computed over a large number of trials ($N \rightarrow \infty$), equal to the true value of the enclosed variable. Therefore, the general idea of signal averaging is to reduce the noise term $\frac{1}{N}\sum_{j=1}^{N} n_j(k) \rightarrow 0$ for large N such that $x(k)_N \rightarrow s(k)_N$. Because in the real world $N << \infty$, we will not reach the ideal situation where the measurement x exactly equals the true signal s; there is a residual noise component that we can characterize by the variance of the estimate $\overline{x(k)_N}$. To simplify notation we will indicate $Var\left(\overline{x(k)_N}\right)$ as $Var(\overline{x})$. The square

root of $Var(\overline{x})$is the standard error of the mean (SEM). We can use Eq. (3.11) to estimate$Var(\overline{x})$:

$$Var(\overline{x}) = E\left\{(\overline{x} - \langle x\rangle)^2\right\} = E\left\{\overline{x}^2 - 2\overline{x}\langle x\rangle + \langle x\rangle^2\right\}$$

Taking into account that $\langle x\rangle$ represents the true average value of x (therefore $E\{\langle x\rangle\} = \langle x\rangle$ and $E\left\{\langle x\rangle^2\right\} = \langle x\rangle^2$), we may simplify:

$$E\left\{\overline{x}^2 - 2\overline{x}\langle x\rangle + \langle x\rangle^2\right\} = E\{\overline{x}^2\} - 2\langle x\rangle E\{\overline{x}\} + \langle x\rangle^2$$

Further, we note that the expected value of the average of x ($\overline{x}$) is equivalent to $\langle x\rangle$ (i.e., $E\{\overline{x}\} = \langle x\rangle$), the above expression can be simplified further leading to:

$$Var(\overline{x}) = E\{\overline{x}^2\} - \langle x\rangle^2 \tag{4.3}$$

Combining Eqs. (4.3) and (4.2), we obtain:

$$Var(\overline{x}) = E\left\{\left[\frac{1}{N}\sum_{i=1}^{N} x_i\right]^2\right\} - \langle x\rangle^2 = E\left\{\left[\frac{1}{N}\sum_{j=1}^{N} x_j\right]\left[\frac{1}{N}\sum_{i=1}^{N} x_i\right]\right\} - \langle x\rangle^2$$

$$= \frac{1}{N^2}\sum_{j=1}^{N}\sum_{i=1}^{N} E\{x_j x_i\} - \langle x\rangle^2 \tag{4.4}$$

The two summations in this expression represent all combinations of i and j, both going from 1 to N and therefore generating $N \times N$ combinations. This set of combinations can be separated into N terms for all $i = j$ and $N^2 - N = N(N-1)$ terms for $i \neq j$:

$$Var(\overline{x}) = \frac{1}{N^2}\overbrace{\underbrace{\sum_{j=1}^{N} E\left\{x_j^2\right\}}_{for\ \ i=j}}^{N\ \ terms} + \frac{1}{N^2}\overbrace{\underbrace{\sum_{j=1}^{N}\sum_{i=1}^{N} E\{x_j x_i\}}_{for\ \ i\neq j}}^{N(N-1)\ \ terms} - \langle x\rangle^2 \tag{4.5}$$

The separation in Eq. (4.5) is useful because the properties of the terms for $i = j$ and for $i \neq j$ differ significantly. As we will see in the following, by exploring this product $x_i x_j$, we will be able to further simplify this expression as well as clarify the mathematical assumptions underlying

the signal averaging technique. As these ***assumptions surface in the text they will be presented in bold and italic.*** Using Eq. (4.1) we can rewrite a single contribution to the summation of N terms with $i = j$ in Eq. (4.5) as:

$$E\left\{x_j^2\right\} = E\left\{[s_j + n_j]^2\right\} = E\left\{s_j^2\right\} + 2E\{s_j\}E\{n_j\} + E\left\{n_j^2\right\} \tag{4.6}$$

Assuming that the noise component ***n_j has zero mean and variance*** σ_n^2, i.e., $E\{n_j\} = 0$ and $E\left\{n_j^2\right\} = \sigma_n^2$:

$$E\left\{x_j^2\right\} = E\left\{s_j^2\right\} + \sigma_n^2 \tag{4.7}$$

The variance of the signal component (σ_s^2) is given by $\sigma_s^2 = E\left\{s_j^2\right\} - \langle s\rangle^2$, which we may substitute for the first term in Eq. (4.7), producing:

$$E\left\{x_j^2\right\} = \sigma_s^2 + \langle s\rangle^2 + \sigma_n^2 \tag{4.8}$$

Combining Eqs. (4.1) and (4.5), the expression for one of the $N(N-1)$ cross terms can be written as:

$$E\{x_j\, x_i\} = E\{[s_j + n_j][s_i + n_i]\} = E\{s_j s_i\} + E\{n_j n_i\} + E\{s_j n_i\} + E\{s_i n_j\} \tag{4.9}$$

If we assume that ***all noise terms and the signal are statistically independent within a given trial, and also across trials***, i.e., independent between trials i and j. Recall that s_i and s_j each include a signal component noise. Therefore, the first term in Eq. (4.9) becomes:

$$E\{s_j s_i\} = E\left\{\left[\langle s\rangle + n_{s_i}\right]\left[\langle s\rangle + n_{s_j}\right]\right\}$$

$$E\{s_j s_i\} = E\left\{\langle s\rangle^2 + \langle s\rangle n_{s_i} + \langle s\rangle n_{s_j} + n_{s_i} n_{s_j}\right\} = \langle s\rangle^2$$

Note that in this case, in contrast to the evaluation of $E\left\{s_j^2\right\}$ in Eqs. (4.7) and (4.8), all noise terms vanish. This result and the independence assumption allow us to rewrite all combined expectations in Eq. (4.9) as the product of the individual expectations:

$$\begin{aligned} E\{s_j s_i\} &= E\{s_j\}E\{s_i\} = \langle s\rangle \times \langle s\rangle = \langle s\rangle^2 \\ E\{n_j n_i\} &= E\{n_j\}E\{n_i\} = 0 \times 0 = 0 \\ E\{s_j n_i\} &= E\{s_j\}E\{n_i\} = \langle s\rangle \times 0 = 0 \\ E\{s_i n_j\} &= E\{s_i\}E\{n_j\} = \langle s\rangle \times 0 = 0 \end{aligned} \tag{4.10}$$

In the above we repeatedly use the property that the expectation of a product of two independent variables can be replaced by the product of the expectation of the individual variables (Appendix 4.1).

Substituting from Eq. (4.8) for the N $i = j$ terms and from Eqs. (4.9) and (4.10) for the $N(N-1)$ $i \neq j$ terms into Eq. (4.5), we obtain the following expression for the variance:

$$Var(\bar{x}) = \frac{1}{N^2}\left[N\left(\sigma_s^2 + \langle s\rangle^2 + \sigma_n^2\right) + \left(N^2 - N\right)\langle s\rangle^2\right] - \langle x\rangle^2 \tag{4.11}$$

Finally, again using the assumption that $\langle n\rangle = 0$, the true value of the measurement x is the averaged signal, i.e., $\langle x\rangle = \langle s\rangle$. This allows us to simplify Eq. (4.11):

$$Var(\bar{x}) = \frac{1}{N^2}\left[N\left(\sigma_s^2 + \langle s\rangle^2 + \sigma_n^2\right) + \left(N^2 - N\right)\langle s\rangle^2\right] - \langle s\rangle^2 \tag{4.12}$$

This expression simplifies to:

$$\boxed{Var(\bar{x}) = \frac{\sigma_s^2 + \sigma_n^2}{N}} \tag{4.13}$$

Eq. (4.13) quantifies the variance of the average ($\bar{x}$), showing that the estimate of the mean improves with an increasing number of repetitions N. In our example the variances σ_s^2, σ_n^2 are generated by two independent sources. In this case the compound effect of the two sources is obtained by adding the variances, similar to the combined effects of independent sources on v_{eff} in Eq. (3.15). The square root of the expression in Eq. (4.13) gives us the SEM; therefore we conclude that the noise in the average $\bar{x}$ decreases with a factor of $1/\sqrt{N}$.

4.4 NOISE ESTIMATES

The ultimate reason to perform signal averaging is to increase the signal-to-noise ratio (Chapter 3). The estimate of residual noise can easily be established in a theoretical example illustrated in the simulation in pr4_1, where all the components are known. In real measurements the noise and signal components are unknown and the averaged result is certain to contain both signal and residual noise (as in Fig. 4.2). In practical applications, there is a number of techniques one might use to estimate the residual noise in the averaged result, and the following sections discuss three of these techniques.

4.4.1 Prestimulus Noise

One might estimate the residual noise by using the prestimulus epoch if there is a reason to assume that the time-locked signal only occurs after the trigger for the signal average, such as an evoked potential. Without a clear indication of a poststimulus response, such as activity surrounding the occurrence of a spike in a spike-triggered average (Chapter 20), this approach obviously won't work. It should also be noted that the prestimulus epoch is not necessarily reliable in the case of a stimulus-evoked potential. Since this type of average is obtained by repetitive stimulation, the late component of the response to a stimulus can still be ongoing in the prestimulus epoch of the next stimulus. In this case, the noise estimate of the prestimulus average will include the effect of the late component of the response. One could attempt to mitigate this problem by using larger interstimulus intervals, removing the late/slow components by high-pass filtering (Chapters 15–18), or using an alternative noise estimator.

4.4.2 Bootstrap

Bootstrapping is another method that works well, especially in off-line average procedures where the signal average is produced with signals that were previously recorded and stored. In this case, a control-average can be produced by picking random triggers rather than the triggers used for producing the "true" average. By picking the trigger randomly, the time-locked aspect of the epochs producing the average is destroyed. This procedure will produce a control-average that still includes the effects of the time-locked signal. However, since the averages are not aligned with the real trigger, the time-locked signal will not be enhanced in this control-average. Obviously, this bootstrap method will work well if the power of the time-locked component is small with respect to the noise that embeds it. Fortunately, this is usually the case because it is a principal motivation to employ signal averaging in the first place!

One can produce a series of control-averages by using the bootstrap multiple times. In this case, you can compare the statistics of the control-averages with the "true" average and use this comparison to validate the "true" average result. The disadvantage of this method is that it requires multiple averages: one for the "true" average and several to produce the control-averages. In addition, the epochs used in the control-averages are not the same as the epochs in the "true" average.

4.4.3 ± Average

One efficient way of establishing the amount of residual noise using the same epochs as those in the "true" average is by using so-called ± averaging.

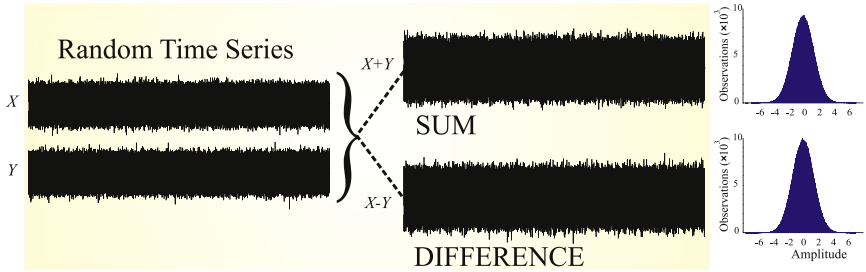


FIGURE 4.3 Random noise traces X and Y, their sum $(X + Y)$ and difference $(X - Y)$ waves. The two amplitude distributions on the right are similar for the sum and difference signals, suggesting that they can be characterized by the same PDF. For random noise, an addition or subtraction creates different time series (i.e., $X + Y \neq X - Y$), but does **not** create different statistical characteristics. This property of random noise is used when considering the $\pm$ signal average (Fig. 4.2 bottom trace) as the basis for estimating the rms of the residual noise in the averaged result.

This is a procedure in which measurements from every other trial are inverted prior to creating the averaged result. This technique removes any consistent signal component by alternating addition and subtraction. However, the residual noise is maintained in the end result (Fig. 4.3). The *rms* value of the noise component estimated from the $\pm$average is the same as that produced by the standard average because random noise samples from the inverted traces have the same distribution as those from noninverted trials. A demonstration (not a proof) is provided in the example in Fig. 4.3, where a pair of random signals X and Y are added and subtracted. The similarity of the amplitude distributions of $X + Y$ and $X - Y$ confirms that the sum and difference signals have the same statistical properties.

4.5 SIGNAL AVERAGING AND NONRANDOM NOISE

The result in the previous section depends heavily on a noise component being random, having zero mean, and being unrelated to the signal. A special case occurs when the noise is not random. This situation may affect the performance of the average and even make it impossible to apply the technique without a few critical adaptations. The most common example of such a situation is the presence of hum (50 or 60 Hz noise originating from the power lines, see Chapter 3 and Fig. 3.4). In typical physiological applications an average is obtained by repeating a standard stimulus of a certain sensory modality and recording a time-locked epoch of neural (of neuromuscular) responses at each repetition. Usually this series of stimuli is triggered at a given stimulus rate dictated by the

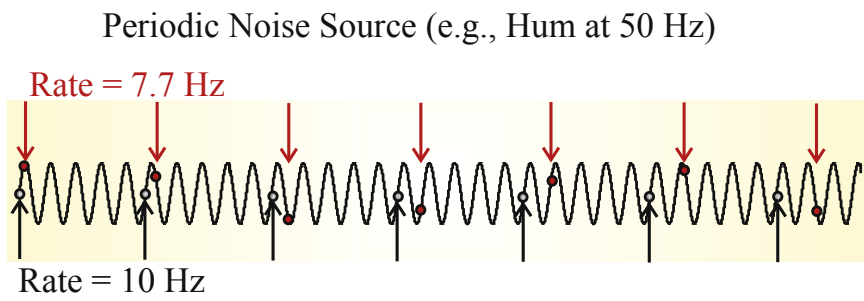


FIGURE 4.4 The stimulus-rate and a periodic component (e.g., a 50-Hz or 60-Hz hum artifact) in the unaveraged signal can produce an undesired effect in the average. An average produced with a 10-Hz rate will contain a large 50-Hz signal. In contrast, an average produced at a 7.7-Hz rate will not contain such a strong 50-Hz artifact. This difference is due to the fact that a rate of 10-Hz results in a stimulus onset that coincides with the same phase in the 50-Hz sinusoidal noise source (black dots), whereas the noninteger rate of 7.7 Hz produces a train of stimuli for which the relative phase of the noise source changes with each stimulus (red dots).

particular purpose of the experiment. It is critical to understand that in this scenario the time-locked components evoked by each stimulus will be enhanced in the average result, but also periodic components with a fixed relation to the stimulus rate (Fig. 4.4). For example, if one happens to stimulate at a rate of exactly 50 Hz, one enhances any minor 50 cycle noise in the signal. (The same example can be given for 60 Hz.) The situation is worse, because any stimulus rate *r* which divides evenly into 50 will have a tendency to enhance a small 50 cycle noise signal (for example the 10-Hz rate represented by the black dots in Fig. 4.4). This problem is often avoided by either ***randomizing the stimulus interval*** or by using a ***noninteger stimulus rate*** such as 3.1, 5.3, or 7.7 Hz (red in Fig. 4.4).

Although the above consideration with respect to periodic noise sources seems trivial, averaging at a poorly chosen rate is a common mistake. I have seen examples where expensive Faraday cages and power supplies were installed to reduce the effects of hum, while with normal averaging procedures, a simple change of the stimulus rate from 5.0 to 5.1 would have been much cheaper and, usually, more effective.

4.6 NOISE AS A FRIEND OF THE SIGNAL AVERAGER

It seems intuitive that a high-precision ADC combined with signal averaging equipment would contribute significantly to the precision of the end result, i.e., the average. In the following example it is shown that ADC precision isn't necessarily the most critical property in such a system and that noise can be helpful when measuring weak signals through

averaging. Noise is usually considered the "enemy," preventing one from measuring the signal reliably. Paradoxically, the averaging process, made to reduce noise, may (in some cases) work better if noise is present. As we will see in the following examples, this is especially true if the resolution of the ADC is low relative to the noise amplitude. Let's assume an extreme example of a 1-bit ADC, i.e., there are only two levels: 0 or 1. Every time the signal is >=0, the ADC assigns a value of 1; every time the signal is <0, the ADC assigns a 0. In this case a small deterministic signal without added noise cannot be averaged or even measured because it would result in the same uninformative series of 0s and 1s in each trial. If we now add noise to the signal, the probability of finding a 1 or a 0 sample is proportional to the signal's amplitude at the time of each sample. By adding the results of a number of trials, we now obtain a probabilistic representation of the signal that can be normalized by the number of trials to obtain an estimate of the signal ranging from 0 to 1.

We can use the individual traces from the simulation script pr4_1.m to explore this phenomenon. Let's take the elements in the matrix NOISE_TRIALS, which is used as the basis for the average, and replace each of the values with 0 if the element's value is <0 and with 1 otherwise. This mimics a 1-bit converter where only 0 or 1 can occur.

First run the script pr4_1 (!!) and then type in the following or use script pr4_3.m:

```
for k=1:sz;
   for m=1:sz;
      if (NOISE_TRIALS(k,m) < 0);      % Is the element < 0 ?
         NOISE_TRIALS(k,m)=0;          % if yes, the simulated ADC result=0
      else;
         NOISE_TRIALS(k,m)=1;          % if not, the simulated ADC result=1
      end;
   end;
end;

average2=sum(NOISE_TRIALS)/sz;
figure
plot(average2)                         % Signal between 0 and 1
```

The figure generated by the above commands/script shows a digitized representation of the signal on a scale from 0 to 1. In Fig. 4.5 we compare

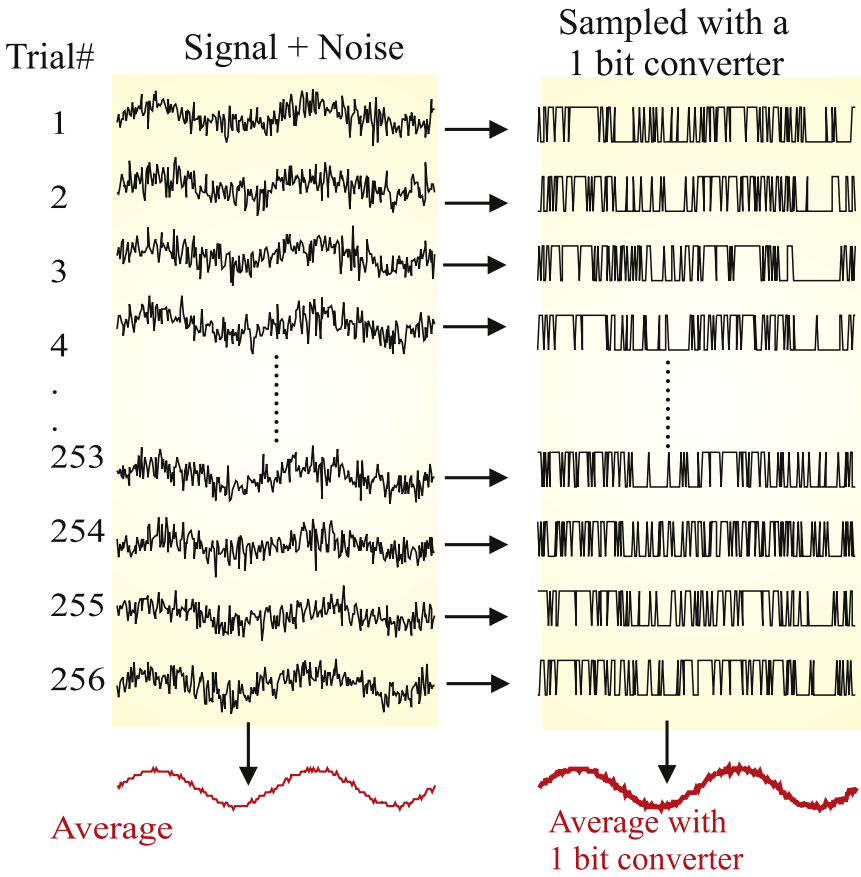


FIGURE 4.5 Signal averaging of the 256 traces generated by pr4_1.m is shown in the left column. The right column shows individual traces that were digitized with a 1-bit ADC using the MATLAB® commands in pr4_3. The averaged result of the traces in the right column is surprisingly close to the average obtained from the signals in the left column. It can be seen that the relative noise component of the 1-bit average is large compared to the standard result shown in the left column. Because the 1-bit converter only produces values between 0 and 1, all amplitudes are normalized to allow comparison.

the averaging result obtained in our original run of pr4_1 and the result obtained here by simulating a 1-bit converter.

The above example shows that reasonable averaging results can be obtained with a low-resolution ADC using the statistical properties of the noise component. This suggests that ADC resolution may not be a very critical component in the signal averaging technique. To explore this a bit further, let's compare two signal averagers that are identical with the

exception of the ADC precision: ***averager A*** has a ***4-bit resolution ADC*** and ***averager B*** a ***12-bit ADC***. ***Let us say we want to know the number of trials N required to obtain an averaged result with a signal-to-noise ratio of at least 3*** (according to Eq. 3.13 ≈9.5 dB) in both systems. Further, let's assume we have a ±15 V range at the ADC input and an amplification of 100,000×. In this example we consider an *rms* value for the signal of 5 μV and for the zero mean noise component of 50 μV. For simplicity we assume a consistent signal, i.e., the variance in the signal component is zero. The signal-to-noise ratio at the amplifier input of both (Eq. 3.13) is $20\log_{10}\left(\frac{5}{50}\right) = -20$ dB (our target is therefore a $9.5-(-20) = 29.5$ dB improvement in signal-to-noise ratio). At the amplifier output (= the ADC input) of both systems we have:

$$\begin{aligned} rms_{signal} \quad & 5\ \mu\text{V} \times 100,000 = 0.5\ \text{V} \\ rms_{noise} \quad & 50\ \mu\text{V} \times 100,000 = 5\ \text{V}. \end{aligned} \tag{4.14}$$

The quantization noise q_A and q_B in ***systems A and B*** are different due to the different resolution of their ADC components. At the output of the systems, the range of this added noise is:

$$\begin{aligned} &\text{Averager A:} \quad q_A = \pm 15\text{V}/2^4 = 30/16 \approx 1.88\ \text{V} \\ &\text{Averager B:} \quad q_B = \pm 15\text{V}/2^{12} = 30/4096 \approx 7.30\ \ 10^{-3}\ \text{V} \end{aligned} \tag{4.15}$$

The variances $\sigma^2_{q_A}$ and $\sigma^2_{q_B}$ associated with these quantization ranges (applying Eq. 3.26) are:

$$\begin{aligned} &\text{Averager A:} \quad \sigma^2_{q_A} = (30/16)^2 \Big/ 12\ \ \text{V}^2 \\ &\text{Averager B:} \quad \sigma^2_{q_B} = (30/4096)^2 \Big/ 12\ \ \text{V}^2 \end{aligned} \tag{4.16}$$

Combining the effect of the two noise sources in each system (using Eq. 3.15), we can determine the ***total noise*** at the input of the ADC as the combination of the original noise (Eq. 4.14; creating an ms of 5^2 V^2) and that produced by quantization:

$$\begin{aligned} &\text{Averager A:} \quad 5^2 + \sigma^2_{q_A} = 5^2 + (30/16)^2 \Big/ 12\ \ \text{V}^2 \\ &\text{Averager B:} \quad 5^2 + \sigma^2_{q_B} = 5^2 + (30/4096)^2 \Big/ 12\ \ \text{V}^2 \end{aligned} \tag{4.17}$$

According to Eq. (4.13) these noise figures will be attenuated by a factor N_A or N_B (number of trials in systems A and B) in the averaged result.

Using the signal-to-noise ratio $rms_{signal}/rms_{Total\ Noise}$ and including our target (a ratio of 3 or better), we get:

$$\begin{aligned}\text{Averager A:}\quad & \frac{0.5}{\sqrt{\dfrac{5^2+\sigma_{q_A}^2}{N_A}}} = \frac{0.5}{\sqrt{\dfrac{5^2+(30/16)^2/12}{N_A}}} \geq 3 \\ \text{Averager B:}\quad & \frac{0.5}{\sqrt{\dfrac{5^2+\sigma_{q_B}^2}{N_B}}} = \frac{0.5}{\sqrt{\dfrac{5^2+(30/4096)^2/12}{N_B}}} \geq 3\end{aligned} \tag{4.18}$$

Solving for the number of trials required in both systems to get this signal-to-noise target, we find that $N_A = 911$ and $N_B = 900$. From this example we conclude that in a high noise environment (i.e., with a high noise level relative to the quantization error q), the precision of the ADC doesn't influence the end result all that much; in our example a huge difference in precision (4 vs. 12 bit, which translates into a factor of 256) only resulted in a small difference in the number of trials required to reach the same signal-to-noise ratio (911 vs. 900, a factor of $\sim$1.01). The example also shows that in a given setup, improvement of the signal-to-noise ratio with averaging is best obtained by increasing the number of trials; from Eq. (4.18) we can determine that the signal-to-noise improvement is, as expected, proportional to $\sqrt{N}$.

4.7 EVOKED POTENTIALS

Evoked potentials (EPs) are frequently used in the context of clinical diagnosis; these signals are good examples of the application of signal averaging in physiology (Chapter 1). The most commonly measured evoked potentials are recorded with the EEG electrode placement (Fig. 1.2A) and represent neural activity in response to stimulation of the auditory, visual, or somatosensory system (AEP, VEP, or SEP, respectively). These examples represent activity associated with the primary perception process. More specialized evoked potentials also exist; these record the activity generated by subsequent or more complex tasks performed by the nervous system. One example is the so-called oddball paradigm, which consists of a set of frequent baseline stimuli, occasionally (usually at random) interrupted by a rare test stimulus. This paradigm usually evokes a centrally located positive wave at 300 ms latency in response to the rare stimulus (the P300). This peak is generally interpreted as representing a neural response to stimulus novelty.

An even more complex measurement is the contingent negative variation (CNV) paradigm; here the subject receives a warning stimulus (usually a short tone burst) that a second stimulus is imminent. When the second stimulus (usually a continuous tone or a series of light flashes) is presented the subject is required to turn it off with a button press. During the gap in between the first (warning) stimulus and the second stimulus, one can observe a centrofrontal negative wave. Relative to the ongoing EEG the CNV signal is weak and must be obtained by averaging; an example of individual trials and the associated average is shown in Fig. 4.6. Here it can be seen that the individual trials contain a significant amount of noise, whereas the average of only 32 trials clearly depicts the negative slope between the stimuli (note that negative is up in Fig. 4.6). The ± average provides an estimate for the residual noise in the averaged result. The original trials are included in the material that can be downloaded from http://booksite.elsevier.com/9780128104828/ (single_trials_CNV.mat).

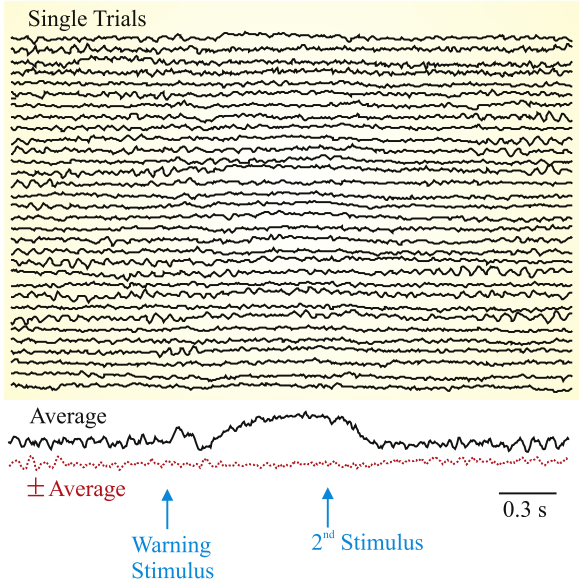


FIGURE 4.6 The contingent negative variation (CNV) measured from C_z (the apex of the scalp, Fig. 1.2A) is usually made visible in the average of individual trials in which a subject receives a warning stimulus that a second stimulation is imminent. The second stimulus must be turned off by a button press of the subject. The lower pair of traces shows the standard average revealing the underlying signal and the ±average as an estimate of the residual noise.

Typing the following MATLAB® commands will display the superimposed 32 original traces as well as the average of those trials.

```
clear
load single_trials_CNV
figure
plot(single)
hold
plot(sum(single')/32,'k+')
```

4.8 OVERVIEW OF COMMONLY APPLIED TIME DOMAIN ANALYSIS TECHNIQUES

1. *Power and related parameters*
   Biomedical applications often require some estimate of the overall strength of measured signals. For this purpose, the variance (σ^2, Chapter 3) of the signal or the mean of the sum of squares $\frac{1}{N}\sum_{n=1}^{N} x^2(n)$ are frequently used. Time series are also frequently demeaned (baseline corrected) before further analysis, making the mean of the sum of squares and the variance equivalent. Another variant is the *rms* (root mean square, Chapter 3).
   Hjorth (1970) describes the signal variance σ^2 as the ***activity index*** in EEG analysis. In the frequency domain, activity can be interpreted as the area under the curve of the power spectrum. To this metric he adds the standard deviations from the first and second derivatives of the time series, σ_d and σ_{dd}, respectively. On the basis of these parameters, Hjorth introduces ***mobility*** $\frac{\sigma_d}{\sigma}$ and ***complexity*** $\frac{\sigma_{dd}/\sigma_d}{\sigma_d/\sigma}$ parameters. In the frequency domain, mobility can be interpreted as the standard deviation of the power spectrum. The complexity metric quantifies the deviation from a pure sine wave as an increase from unity.
2. *Zero-Crossings*
   The 0-crossings in a demeaned signal can give an indication of the dominant frequency component in a signal. For example, if a signal is dominated by a 2-Hz sine wave, it will have four 0-crossings per second; i.e., the number of 0-crossings/s divided by 2 is the frequency of the dominant signal component. The lengths of epochs in between 0-crossings can also be used for ***interval analysis***. Note that there are two types of 0-crossings, from positive to negative and vice versa. Zero-crossings in the derivative of a time series can also be used to find local ***maxima*** and ***minima***.

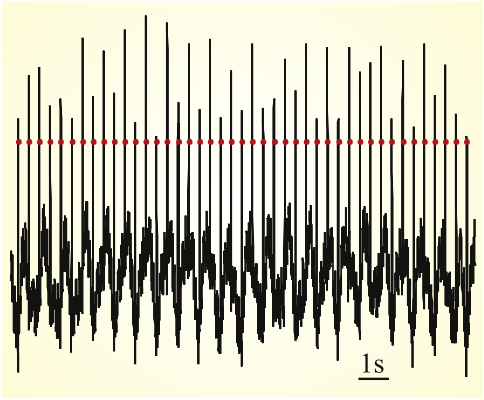


FIGURE 4.7 An ECG signal (see also Fig. 1.4) from a human neonate and the detected QRS complexes (red dots).

3. *Peak Detection*
   Various methods to detect peaks are used to locate extrema within time series. If the amplitudes between subsequent local maxima and minima are measured, we can determine the amplitude distribution of the time series. In the case of peak detection in signals consisting of a series of impulses, the peak detection procedures are used to calculate intervals between such events. This routine is frequently used to detect the events in signals containing spikes or in the ECG to detect the QRS complexes (Chapter 1, Fig. 1.5). An example of QRS complex detection in human neonates is shown in Fig. 4.7. The general approach in these algorithms consists of two stages: first pretreat the signal in order to remove artifacts and then detect extreme values above a set threshold.

   The following part of a MATLAB® script is an example of a peak detector used to create Fig. 4.7.
   NOTE that this is a part of a script! The whole script pr4_4.m and an associated data file (subecg) can be downloaded from http://booksite.elsevier.com/9780128104828/.

```
% 1. pre-process the data
[c,d]=butter(2,[15/fN 45/fN],'bandpass');          % 2nd order 15-45 Hz (XIII)
subecgFF=filtfilt(c,d,subecg-mean(subecg));        % use filtfilt to prevent
                                                      phase-shift

% 2. detect peaks
```

```
% In this routine we only look for nearest neighbours (three subsequent
      points)
% adding additional points will make the algorithm more robust
threshold=level*max(subecgFF);              % detection threshold
for i=2:length(subecgFF)-1;
  if (subecgFF(i)>threshold);               % check if the level is exceeded
    % is the point a relative maximum (Note the >= sign)?
    if((subecgFF(i)>=subecgFF(i-1))&(subecgFF(i)>=subecgFF(i+1)));
      % if yes, is it not a subsequent maximum in the same heartbeat
      if (i-i_prev > 50)
        D(n)=i;                             % Store the index in D
        i_prev=i;
        n=n+1;
      end;
    end;
  end;
end;
```

4. *Level and Window Detection*
   In some types of time series (such as in extracellular recordings of action potentials) one is interested in identifying epochs in which the signal is within a certain amplitude range. Analog- or digital-based window and level detectors are available to provide such data processing.
5. *Filtering (see Chapters 15–18)*
   The filters we will consider in later chapters are both analog and digital implementations. For analog filters we will focus on RC circuits, digital implementations will cover infinite impulse response (IIR) and finite impulse response (FIR) versions.
6. *Real Convolution (see Chapter 13)*
   Convolution plays an important role in relating input and output of linear time invariant systems.
7. *Cross-Correlation (see Chapter 13)*
   Cross-correlation is related to convolution and can be used to quantify the relationship between different signals or between different parts of the same signal (termed "auto-correlation").
8. *Template Matching*
   In some applications signal features are extracted by correlating a known template with a time series. Wavelet and scaling signals can be considered as a special type of template (see Chapters 21 and 22).
9. *Miscellaneous*
   In some cases the task at hand is highly specific: e.g., detection of epileptic spikes in the EEG. In these instances a specially

developed metric may provide a good solution. For example, in EEG spike detection, a "sharpness index" works reasonably well (Gotman and Gloor, 1976).

APPENDIX 4.1 EXPECTATION OF THE PRODUCT OF INDEPENDENT RANDOM VARIABLES

The development used to show that averaging reduces the noise component is based on the use of the expectation formalism, $E\{\ldots\}$ (see also Eq. 3.9). In this appendix we summarize a few of its properties that are frequently used.

Note that the expectation of a constant c (with distribution $\delta(x-c)$) is simply the constant itself: $E\{c\} = c$. Since $E\{\ldots\}$ is a linear operator, the sum of two random variables is the sum of the individual expectations: $E\{x+y\} = E\{x\} + E\{y\}$. The procedure we need to follow for the evaluation of the expectation of the product of two variables $E\{xy\}$ depends on whether they are independent or not. It is simply $E\{x\}E\{y\}$ if the joint distribution $p(x\,y)$ of x and y can be written as the product of two independent distributions, i.e., $f(x)g(y)$. In this case we may write

$$E\{x\,y\} = \int\int xy\,p(x,y)\,dxdy = \int x\,f(x)dx \int y\,g(y)dy = E\{x\}E\{y\} \tag{A4.1-1}$$

This property is repeatedly used in the derivation leading to Eq. (4.13) and multiple other cases, for example in the development for the Poisson–Wiener kernels in Appendix 26.1.

In the following we show a simulated example that for two independent random variables x and y we may indeed use $E\{x\,y\} = E\{x\}E\{y\}$. Let's create two normally distributed variables x and y with mean values of 1 and 3 and a variance of 1.

To do this, type in the MATLAB® command window:

```
x=randn(1,10000)+1;
y=randn(1,10000)+3;
```

If you want you can plot these variables to make sure they are really different, or even better you can plot the variables against each other with:

```
figure;
plot(x,y,'.')
```

and you will see a cloud of points around coordinates (1,3), reflecting the absence of a clear relationship between x_1 and x_2. Now recall that in an

ergodic process the mean values of x and y are $E\{x\}$, $E\{y\}$, so we can estimate the product $E\{x\} \times E\{y\}$ by typing:

```
mean(x)*mean(y)
```

The answer will be a value close to 3. The outcome will usually not be exactly equal to 3, because we simulate the procedure using data sets of random numbers with limited size.

Now we can compute $E\{x\,y\}$ by computing the mean of the product of x and y using:

```
X=x.*y;
sum(X)/length(X)
```

As we expect, the answer will again be a value close to 3.

You can repeat the procedure with another pair a and b that are **not** independent. For example:

```
a=randn(1,10000)+1;
b=a*2;
```

By following the same procedure we used above for x and y, you will now find that the plot of the variables against each other shows a clear nonrandom relationship, and that Eq. (A4.1-1) does not hold, i.e.,: $E\{ab\} \neq E\{a\} \times E\{b\}$.

EXERCISES

4.1 You want to apply the signal averaging technique. For reasons of efficiency you plan to monitor the progress of improving your signal-to-noise ratio (SNR) by estimating the SNR after each repetition (trial). You decide that you will try to accomplish this by plotting the ratio [rms(signal)/rms(residual noise)] versus trial number.

Note: From the average, you cannot compute the rms(signal) directly but only the rms (signal + residual noise). You may, however, obtain an estimate by assuming that the signal and noise are independent and that the mean values of the signal and noise are zero so that (see Eq. 3.15) ms(signal + residual noise) = ms(signal) + ms(residual noise). From this relationship you can estimate rms(signal) as the square root of ms(signal + residual noise), obtained from your average, minus ms(residual noise), obtained from your ±average.

An alternative practical solution to monitor SNR is to compute rms (signal + residual noise)/rms(residual noise).

Modify program pr4_1.m to evaluate averages at $n = 1, 2, \ldots, 1024$ repetitions.

a. Depict the average and ±average for $n = 1, 10, 100, 1000$.
b. Calculate the ms of the noise for each value of n.
c. Calculate a value for the SNR for each value of n.
d. Plot the values in (b) and (c) against n.
e. Relate your finding to Eq. (4.13).
f. Comment on the noise reduction in your simulated results by comparing it with the analytically obtained estimate of the SNR improvement as a function of the number of repetitions.

4.2 Now assume that in Exercise 4.1, the sine wave you averaged is not the signal but noise originating from another instrument. So, in your average, you want to get rid of both the sine wave as well as the random noise component.

a. Explain how you can accomplish this.
b. Write a Matlab® script or make modifications to pr4_1.m to show how this works.
c. Plot the ms of the noise versus n.
d. Relate your finding to Eq. (4.13).

4.3 A signal averager's output generates Gaussian noise with zero mean and variance σ_o^2. This output noise is added to any output and it is independent of the averager's input signal s, input noise n, sample point k (1, … , M), and number of trials N. (See also Fig. 4.1 and Eq. (4.1) for the definitions of these variables.)

The evoked signal s is identical across all trials (e.g., its variance is zero: $\sigma_s^2 = 0$). The input noise n is zero mean with variance σ_n^2. Derive an expression for the signal-to-noise ratio $G = SNR_{out}/SNR_{in} > 0$

4.4 Load ActionPotential.mat, a recording of action potentials, into MATLAB®.

a. Create a peak detection script that detects the action potentials.
b. Validate your detection by plotting the original signal and the detected peaks in a single graph (similar to Fig. 4.7).

4.5 Assume a measurement that includes a noise-free stimulus-evoked signal with a mean square (ms) value of 5 μV^2 and additive Gaussian noise with an ms value of 50 μV^2. Both signal and noise have zero mean.

a. What is the SNR of this measurement in dB?

b. How many repetitions N of this measurement are required to obtain a value for SNR >20 dB?
c. What is the ms in the ±average associated with your result in question b?
d. Check your answers in questions (a), (b), and (c) in a MATLAB® simulation employing a sine wave with an ms value of 5 arbitrary units (AU) contaminated with noise with an ms value of 50 AU (use the randn function to generate the noise).
(Hint: for the sinusoidal signal see Appendix 3.3.)
e. Why do we use the MATLAB® randn instead of rand function?

References

Gotman, J., Gloor, P., 1976. Automatic recognition and quantification of inter- ictal epileptic activity in the human scalp EEG. Electroencephalogr. Clin. NeuroPhysiol. 41, 513–529.
The first paper describing a successful automated time domain analysis to detect epileptic spike activity in clinical recordings. Although relatively simple, this method is still being used in clinical equipment today.

Hjorth, B., 1970. EEG analysis based on time domain properties. Electroencephalogr. Clin. Neurophysiol. 29, 306–310.
Introduction of activity, mobility, and complexity parameters in EEG analysis.

CHAPTER

5

Real and Complex Fourier Series

5.1 INTRODUCTION

In this chapter, the Fourier series in the real and complex form are introduced. First we develop the Fourier series as a technique to represent arbitrary periodic functions as a summation of sine and cosine waves. Subsequently we show that the complex version of the Fourier series is simply an alternative notation. At the end of this chapter we apply the Fourier series technique to decompose periodic functions into their cosine and sine components.

Because the underlying principle is to represent waveforms as a summation of periodic cosine and sine waves with different frequencies, one can interpret Fourier analysis as a technique for examining signals in the ***frequency domain***. At first sight, the term frequency domain may appear to be a novel or unusual concept. However, in daily language we do use frequency domain descriptions; for instance, we use a frequency domain specification to describe the power line source as a 120-V, 60-Hz signal. Also, the decomposition of signals into underlying frequency components is familiar to most; examples are the color spectrum obtained from decomposing white light with a prism (Fig. 5.1), or decomposing sound into pure tone components.

An example showing an approximation of a square wave created from the sum of five sine waves is shown in Fig. 5.2. The example shown in Fig. 5.2 can be reproduced with MATLAB® script pr5_1.m. This example illustrates the basis of spectral analysis: a time domain signal (i.e., the [almost] square wave) can be decomposed into five sine waves, each with a different frequency and amplitude. The graph depicting these frequency and amplitude values in Fig. 5.2 is a frequency domain representation of the (almost) square wave in the time domain. This task of deriving a frequency domain equivalent of a signal originally in the time or spatial domain is the topic of this chapter and Chapters 6–8. Here we introduce

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00005-9

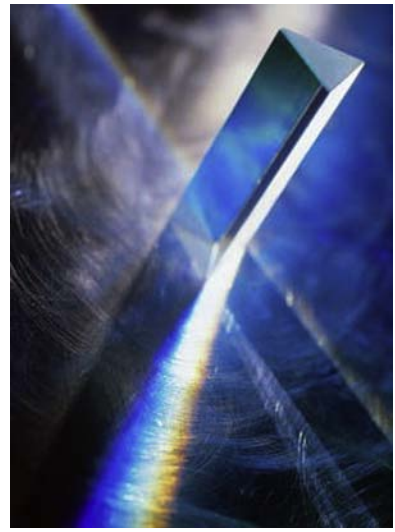


FIGURE 5.1 A prism performs spectral decomposition of white light in bands with different wavelengths that are perceived by us as different colors.

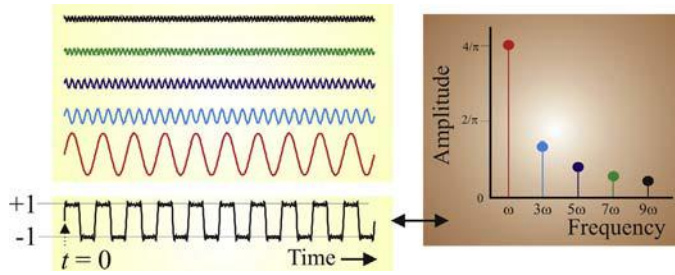


FIGURE 5.2 The sum of five sine waves approximates a square wave with amplitude ±1 (bottom trace). The amplitude of the sine waves decreases with frequency. The spectral content of the square wave is shown in a graph of amplitude versus frequency (right). The data can be obtained by running script pr5_1.m; the spectrum of a square wave is computed analytically in the second example in Section 5.4.

the Fourier series, and on the basis of this concept we introduce the continuous transform and its discrete version in Chapter 6. On the basis of the discrete Fourier transform we describe the development of algorithms to calculate the spectrum of a time series in Chapter 7. Further details of how to deal with unevenly sampled data or how to address some of the shortcomings of the standard procedures are presented in Chapter 8. The order in which we proceed from the Fourier series to spectral analysis and specific algorithms is depicted in Fig. 5.3. The end result of all this is that

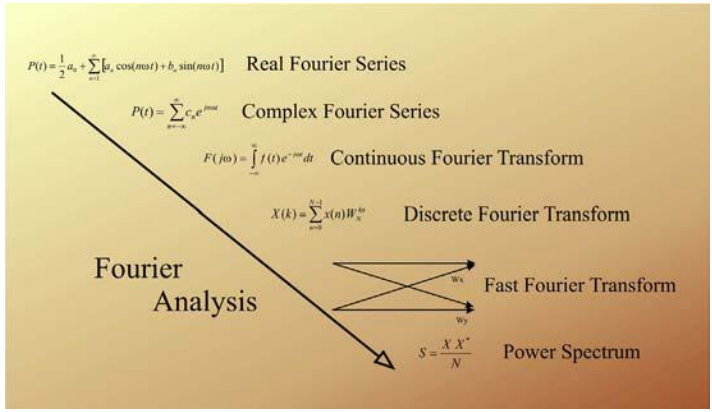


FIGURE 5.3 The relationship between different types of Fourier analysis. The real and complex Fourier series can represent a function as the sum of waves as shown in the example in Fig. 5.2. The continuous and discrete versions of the Fourier transform provide the basis for examining real-world signals in the frequency domain. The computational effort to obtain a Fourier transform is significantly reduced by using the fast Fourier transform (FFT) algorithm. The FFT result can subsequently be applied to compute spectral properties such as a power spectrum describing the power of the signal's different frequency components.

you will understand the underlying math of the Fourier transform technique, you will have an idea of when to apply this powerful analytical tool, and you will understand what happens "under the hood" when you type the command `fft` or `fft2` in MATLAB®.

5.2 THE FOURIER SERIES

The Fourier series provides a basis for analysis of signals in the frequency domain. In this section we show that a function $f(t)$ (such as the [almost] square wave in Fig. 5.2) with period T [i.e., $f(t) = f(t+T)$], frequency $f = 1/T$, and angular frequency ω defined as $\omega = 2\pi f$ can be represented by a series $P(t)$:

$$P(t) = \frac{1}{2}a_0 + a_1\cos(\omega t) + a_2\cos(2\omega t) + \cdots + b_1\sin(\omega t) + b_2\sin(2\omega t) + \cdots$$
$$= \frac{1}{2}a_0 + \sum_{n=1}^{\infty}[a_n\cos(n\omega t) + b_n\sin(n\omega t)] \tag{5.1}$$

with the first term $\frac{1}{2}a_0$ representing the ***DC* (*direct current*)** component, and the remaining sine and cosine waves weighted by the a_n and b_n

coefficients represent the ***AC* (*alternating current*)** components of the signal. The seemingly arbitrary choice of $\frac{1}{2}a_0$ allows us to obtain $2/T$ scaling for all coefficients (e.g., see Eqs. 5.6, 5.12, and 5.18). However, in some definitions of the Fourier series the first term is defined as a_0, leading to a difference of a factor of 2 in the end result shown in Eq. (5.6).

5.2.1 Minimization of the Difference Between $P(t)$ and $f(t)$

In the following sections we derive the expressions for the coefficients a_n and b_n in Eq. (5.1). Because it is easy to lose the big picture in the mathematical detail, we summarize the strategy in Fig. 5.4 and relegate some of the mathematical detail to Appendices 5.1 and 5.2. Examples of how to apply Fourier series analysis to time series are given in Section 5.4; the reader who is not (yet) interested in the derivations described in the following paragraphs can simply proceed to these examples and apply Eqs. (5.6), (5.12), and (5.18) to calculate the Fourier series coefficients.

Two strategies are commonly used to derive the equations for coefficients a_n and b_n. One method begins by multiplying the terms of the series in Eq. (5.1) with $\cos(n\omega t)$ or $\sin(n\omega t)$ with $n = 0, 1, 2, \ldots$ and

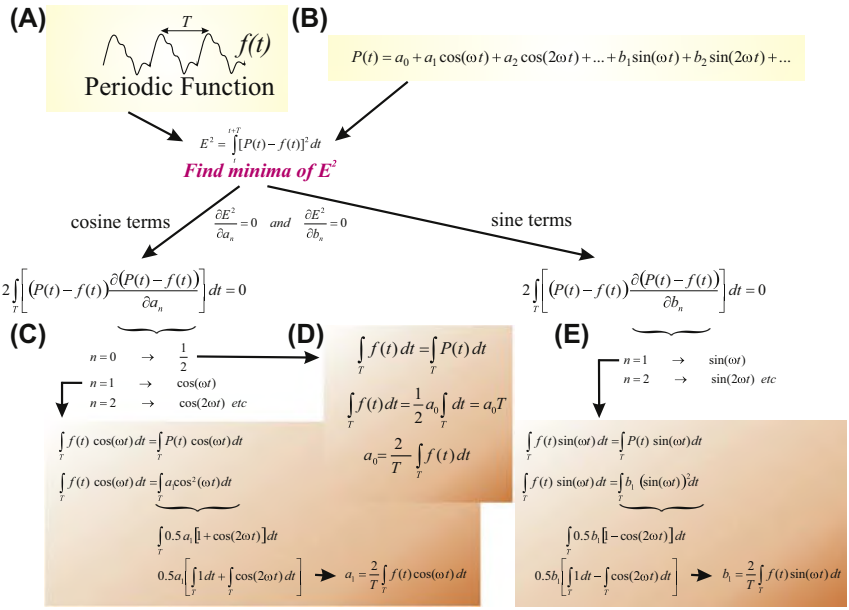


FIGURE 5.4 Overview of the real Fourier series representation of $f(t)$, a periodic function (A). (B) The real Fourier series $P(t)$. (C) and (D) Determination of coefficients a_0 and a_1 in $P(t)$. (E) The same as (C) for the b_1 coefficient (note that there is no b_0). Determination of a_n and b_n coefficients is similar to the procedure for a_1 and b_1.

integrating over a full period T associated with the lowest frequency ω. While it inevitably leads to the correct results, this approach is less intuitive, because it starts from Eqs. (5.8) and (5.14) directly without particular justification. The other method, which in any case leads to the same result, starts with an evaluation of the difference between the Fourier series approximation $P(t)$ and function $f(t)$ itself. The difference is considered the error of the approximation: i.e., the error E that is made by the approximation is $[P(t) - f(t)]$, which can be minimized by reducing E^2 over a full period T of the time series:

$$E^2 = \int_t^{t+T} [P(t) - f(t)]^2 dt \tag{5.2}$$

Assuming that the integral in Eq. (5.2) exists, we can find the minimum of the error function by: $\partial E^2/\partial a_n = 0$ and $\partial E^2/\partial b_n = 0$

Substitution of the expression in Eq. (5.2) into $\partial E^2/\partial a_n$ gives: $\partial\left[\int_t^{t+T} [P(t) - f(t)]^2 dt\right] \Big/ \partial a_n$. By reversing the order of differentiation and integration we obtain $\int_t^{t+T} \partial\left\{[P(t) - f(t)]^2 dt\right\} \Big/ \partial a_n$ which can be written as:

$$2\int_T \left[(P(t) - f(t))\frac{\partial(P(t) - f(t))}{\partial a_n}\right] dt = 0 \tag{5.3a}$$

The outcomes of the partial derivative expression $\partial(P(t) - f(t))/\partial a_n = \partial P(t)/\partial a_n$ for different a_n are summarized in Table 5.1.

TABLE 5.1 Evaluation of $\partial P(t)/\partial a_n$ for Different Values of n

Index	Derivative
$n = 0$	$\partial P(t)/\partial a_0 = \partial\left[\frac{1}{2}a_0 + a_1\cos(\omega t) + a_2\cos(2\omega t)\ldots + b_1\sin(\omega t) + \ldots\right] \Big/ \partial a_0 = 1/2$
$n = 1$	$\partial P(t)/\partial a_1 = \partial\left[\frac{1}{2}a_0 + a_1\cos(\omega t) + a_2\cos(2\omega t)\ldots + b_1\sin(\omega t) + \ldots\right] \Big/ \partial a_1 = \cos(\omega t)$
$n = 2$	$\partial P(t)/\partial a_2 = \partial\left[\frac{1}{2}a_0 + a_1\cos(\omega t) + a_2\cos(2\omega t)\ldots + b_1\sin(\omega t) + \ldots\right] \Big/ \partial a_2 = \cos(2\omega t)$
n	$\partial P(t)/\partial a_n = \partial\left[\frac{1}{2}a_0 + a_1\cos(\omega t) + a_2\cos(2\omega t)\ldots + b_1\sin(\omega t) + \ldots\right] \Big/ \partial a_n = \cos(n\omega t)$

Because for each partial derivative to a_n there is only one term in $P(t)$ containing a_n, the outcome is a single term for each value of n.

Notes:

1. The area enclosed by a periodic function is independent of the starting point, so integration over a full period is insensitive to the value of t. For example, we may integrate from $0 \rightarrow T$ or from $-T/2 \rightarrow T/2$ and obtain the same result. Therefore we change the notation from $\int_t^{t+T}$ to $\int_T$ in Eq. (5.3a); both notations indicate that the integration is taken over the entire period without a particular choice of integration limits.
2. In some of the textbook derivations of the Fourier series ωt is substituted by a variable x; the integration limits over a full period then become: $0 \rightarrow 2\pi$ rad or $-\pi \rightarrow \pi$ rad.
3. The partial derivatives in the above equations are ***not*** with respect to t but to a_n and b_n.

Because we evaluate $P(t)$ also as a function of a_n as well as of b_n, we should (strictly speaking) reflect that in the notation by using $\partial P(t, a_n, b_n)/\partial a_n$ and $\partial P(t, a_n, b_n)/\partial b_n$. In the text we simplify this cumbersome notation to $\partial P(t)/\partial a_n$ and $\partial P(t_n)/\partial b_n$.

Minimization of Eq. (5.2) for b_n:

$$2\int_T \left[(P(t) - f(t))\frac{\partial(P(t) - f(t))}{\partial b_n}\right]dt = 0 \tag{5.3b}$$

Again, the outcome of the partial derivative expression $\partial(P(t) - f(t))/\partial b_n = \partial P(t)/\partial b_n$ varies with the value of n (Table 5.2).

TABLE 5.2 Evaluation of $\partial P(t)/\partial b_n$ for Different Values of n

Index	Derivative
$n = 0$	Index does not exist for b coefficient
$n = 1$	$\partial P(t)/\partial b_1 = \partial\left[\frac{1}{2}a_0 + a_1\cos(\omega t) + a_2\cos(2\omega t)\ldots + b_1\sin(\omega t) + \ldots\right] \Big/ \partial b_1 = \sin(\omega t)$
$n = 2$	$\partial P(t)/\partial b_2 = \partial\left[\frac{1}{2}a_0 + a_1\cos(\omega t) + a_2\cos(2\omega t)\ldots + b_1\sin(\omega t) + \ldots\right] \Big/ \partial b_2 = \sin(2\omega t)$
n	$\partial P(t)/\partial b_n = \partial\left[\frac{1}{2}a_0 + a_1\cos(\omega t) + a_2\cos(2\omega t)\ldots + b_1\sin(\omega t) + \ldots\right] \Big/ \partial b_n = \sin(n\omega t)$

In the following sections we use the obtained results to derive expressions for the coefficients a_n and b_n. To simplify matters, we will frequently rely on two helpful properties: the fact that (1) the integral of a cosine or sine wave over one or more periods evaluates to zero, and (2) the orthogonal characteristics of the integrals at hand. The mathematical details of this approach can be found in Appendix 5.1.

5.2.1.1 *Coefficient* $\mathbf{a_0}$

Returning to the a_n coefficients: for $n = 0$ we found that the derivative associated with minimization evaluates to ½ (Table 5.1). Substitution of this result into Eq. (5.3a) gives us an expression for a_0:

$$2\int_T (P(t) - f(t))\tfrac{1}{2}dt = \int_T P(t)dt - \int_T f(t)dt = 0 \rightarrow \int_T f(t)dt = \int_T P(t)dt \tag{5.4}$$

$$\int_T f(t)dt = \int_T \left[\frac{1}{2}a_0 + a_1\cos(\omega t) + a_2\cos(2\omega t)\ldots + b_1\sin(\omega t) + \ldots\right]dt$$

$$= \int_T \frac{1}{2}a_0 dt + \int_T a_1\cos(\omega t)dt + \int_T a_2\cos(2\omega t)dt\ldots + \int_T b_1\sin(\omega t)dt + \ldots \tag{5.5}$$

In Eq. (5.5), the integrals of the cosine and sine terms (evaluated over T) equal zero (if this result isn't obvious, review Eq. (A5.1-3) in Appendix 5.1), leaving only the nonzero a_0 term:

$$\int_T f(t)\,dt = \frac{1}{2}a_0\int_T dt = \frac{1}{2}a_0 T \rightarrow \quad \boxed{a_0 = \frac{2}{T}\int_T f(t)\,dt} \tag{5.6}$$

Note: The factor of 2 in Eq. (5.6) originates from our choice to represent the first term in Eq. (5.1) as $\frac{1}{2}a_0$. The first term in the Fourier series is therefore $\frac{1}{2}a_0 = 1/T\int_T f(t)dt$, which is the mean of the function $f(t)$ in the interval T (see also Chapter 3, Eq. 3.6). In terms of electrical signals, this can also be thought of as the DC (direct current) component of $f(t)$.

5.2.1.2 Coefficients a_1 and a_n

For $n = 1$ we obtained $\cos(\omega t)$ for the partial derivative (Table 5.1); substituting this result into Eq. (5.3a):

$$2\int_T [(P(t) - f(t))\cos(\omega t)]dt = 2\int_T P(t)\cos(\omega t)dt - 2\int_T f(t)\cos(\omega t)dt = 0$$

$$\rightarrow \int_T f(t)\cos(\omega t)dt = \int_T P(t)\cos(\omega t)dt \tag{5.7}$$

Filling in the terms for the Fourier series $P(t)$:

$$\int_T f(t)\cos(\omega t)dt = \int_T \left[\frac{1}{2}a_0 + a_1\cos(\omega t) + a_2\cos(2\omega t)\ldots + b_1\sin(\omega t) + \ldots\right]\cos(\omega t)dt$$

$$= \int_T \frac{1}{2}a_0\cos(\omega t)dt + \int_T a_1(\cos(\omega t))^2 dt + \int_T a_2\cos(2\omega t)\cos(\omega t)dt\ldots + \int_T b_1\sin(\omega t)\cos(\omega t)dt + \ldots \tag{5.8}$$

From Eq. (5.8) we can solve for a_1 using the following logic:

1. The first term in the expression above, the integral of $\cos(\omega t)$ over a full period $\int_T \frac{1}{2}a_0\cos(\omega t)dt$ evaluates to zero. As mentioned above, this result is obtained because the area enclosed by a cosine function over a full period equals zero (Appendix 5.1, Equation (A5.1-3)).
2. The second term does not evaluate to zero, and we will address this term in Eq. (5.9).
3. All remaining terms in Eq. (5.8) are integrals over T that contain:
   a. the product of two cosine functions which evaluate to zero because of orthogonal behavior (Appendix 5.1, Equation (A5.1-9));
   b. or, sine × cosine products which evaluate to zero (Appendix 5.1, Equation (A5.1-9)).

Therefore, all the terms in Eq. (5.8) evaluate to zero, except $\int_T a_1(\cos(\omega t))^2 dt$, allowing us to simplify Eq. (5.7) to:

$$\int_T f(t)\cos(\omega t)dt = \int_T a_1(\cos(\omega t))^2 dt \tag{5.9}$$

We use the special trigonometric relationships summarized in Eq. (A5.1-6), with $A = \omega t$ and substitute the result in Eq. (5.9):

$$\int_T f(t)\cos(\omega t)dt = \int_T a_1(\cos(\omega t))^2 dt = \int_T \frac{1}{2}a_1[1+\cos(2\omega t)]dt$$

The part after the equal sign can be further simplified:

$$\frac{1}{2}a_1\left[\int_T dt + \int_T \cos(2\omega t)dt\right] = \frac{1}{2}[t]_0^T + 0 = \frac{T}{2}a_1 \tag{5.10}$$

Note that the second integral in between the square brackets in Eq. (5.10) is a cosine evaluated over two periods and therefore evaluates to zero (Appendix 5.1). Solving Eq. (5.10) for the coefficient:

$$a_1 = \frac{2}{T}\int_T f(t)\cos(\omega t)dt \tag{5.11}$$

The above procedure can be applied to find the other coefficients a_n. The integrals of the products $\cos(n\omega t) \times \cos(m\omega t)$ in the series all evaluate to zero with the exception of those in which $m = n$ (Appendix 5.1). The property that products of functions are zero unless they have the same coefficient is characteristic of ***orthogonal functions*** (Appendix 5.1). The integral of the products $\cos(n\omega t) \times \sin(m\omega t)$ all evaluate to zero also (Appendix 5.1). This leads to the general formula for a_n:

$$\boxed{a_n = \frac{2}{T}\int_T f(t)\cos(n\omega t)dt} \tag{5.12}$$

5.2.1.3 *Coefficients* $\mathbf{b_1}$ *and* $\mathbf{b_n}$

For $n = 1$ we obtained $\sin(\omega t)$ for the partial derivative (Table 5.2); substituting this result into Eq. (5.3b):

$$2\int_T [(P(t) - f(t))\sin(\omega t)]dt = 2\int_T P(t)\sin(\omega t)dt - 2\int_T f(t)\sin(\omega t)dt = 0$$

$$\rightarrow \int_T f(t)\sin(\omega t)dt = \int_T P(t)\sin(\omega t)dt \tag{5.13}$$

Substituting the terms for the Fourier series $P(t)$:

$$\begin{aligned}\int_T f(t)\sin(\omega t)dt &= \int_T \left[\frac{1}{2}a_0 + a_1\cos(\omega t) + a_2\cos(2\omega t)\ldots\right.\\ &\quad \left. + b_1\sin(\omega t) + \ldots\right]\sin(\omega t)dt\\ &= \int_T \frac{1}{2}a_0\sin(\omega t)dt + \int_T a_1\cos(\omega t)\sin(\omega t)dt\\ &\quad + \int_T a_2\cos(2\omega t)\sin(\omega t)dt\ldots + \int_T b_1(\sin(\omega t))^2dt + \ldots\end{aligned} \tag{5.14}$$

For the same reasons used in deriving the expression for a_1 above (***orthogonal function property,*** Appendix 5.1) all terms except the one with the $(\sin(\omega t))^2$ evaluate to zero. Therefore Eq. (5.14) simplifies to:

$$\int_T f(t)\sin(\omega t)dt = \int_T b_1(\sin(\omega t))^2dt \tag{5.15}$$

Again using the trigonometric identity (A5.1-6), with $A = \omega t$ and substituting the result in Eq. (5.15) we get:

$$\begin{aligned}\int_T b_1(\sin(\omega t))^2dt &= \int_T \frac{1}{2}b_1[1-\cos(2\omega t)]dt\\ &= \frac{1}{2}b_1\left[\int_T dt - \int_T \cos(2\omega t)dt\right] = \frac{T}{2}b_1\end{aligned} \tag{5.16}$$

Using the property from Eq. (A5.1-3), it can be seen that the second integral in Eq. (5.16) evaluates to zero. Solving Eq. (5.16) for the coefficient b_1:

$$b_1 = \frac{2}{T}\int_T f(t)\sin(\omega t)dt \tag{5.17}$$

And finally, applying the same procedure to solve for b_n:

$$\boxed{b_n = \frac{2}{T}\int_T f(t)\sin(n\omega t)dt} \tag{5.18}$$

5.3 THE COMPLEX FOURIER SERIES

The Fourier series of a periodic function is frequently presented in the complex form. In this section we first introduce the complex version of the Fourier series and we then show its equivalence to the real Fourier series derived above. The notation for the complex Fourier series is:

$$\boxed{P(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega t}} \tag{5.19}$$

The coefficients c_n in Eq. (5.19) are defined as:

$$\boxed{c_n = \frac{1}{T}\int_T f(t)e^{-jn\omega t}dt} \tag{5.20}$$

Just as in the real Fourier series formalism, the $\int_T \cdots$ in Eq. (5.20) indicates that the integral must be evaluated over a full period T, where it is not important what the starting point is, e.g., $-T/2 \to T/2$ or $0 \to T$. Note that as compared with the formulas for the coefficients in the real series (Eqs. 5.6, 5.12, and 5.18), the sine and cosine terms are replaced by a complex exponential. In addition, comparing Eqs. (5.19) and (5.1) we see that the summation is performed from $-\infty$ to ∞ instead of from 0 to ∞ as in the real series.

Note: The integral in Eq. (5.20) is finite if:

$$|c_n| = \left|\frac{1}{T}\int_T f(t)e^{-jn\omega t}dt\right| \leq \frac{1}{T}\int_T |f(t)||e^{-jn\omega t}|dt < \infty$$

Because $|e^{-jn\omega t}| = 1$ we may conclude that the integral must be finite if $\int_t^{t+T} |f(t)|dt < \infty$.

continued

This is the so-called weak ***Dirichlet*** condition that guarantees the existence of the Fourier series. A function such as $f(t) = 1/t$ over an interval from 0 to T would fail this criterion. The other (strong) Dirichlet conditions are that the frequencies included in $f(t)$ are finite (a finite number minima and maxima) and that the number of discontinuities (abrupt changes) are also finite.

Although these criteria play a role in the analysis of functions, for measured time series they are irrelevant because these signals are always bounded within the measurement range and limited in frequency content by the bandwidth of the recording equipment.

In the following we show that ***the real and complex Fourier series notations are equivalent***. The complex form of $P(t)$ in Eq. (5.19) can be rewritten as:

$$P(t) = c_0 + \sum_{n=1}^{\infty} c_n e^{jn\omega t} + \sum_{n=-\infty}^{-1} c_n e^{jn\omega t} \tag{5.21}$$

We can change the polarity of the summation in the third term of Eq. (5.21) from $-\infty \rightarrow -1$ to $1 \rightarrow \infty$. We correct this by changing the sign of n: in $c_n \rightarrow c_{-n}$ and in the exponent $e^{jn\omega t} \rightarrow e^{-jn\omega t}$. The result of this change is:

$$P(t) = c_0 + \sum_{n=1}^{\infty} c_n e^{jn\omega t} + \sum_{n=1}^{\infty} c_{-n} e^{-jn\omega t} \tag{5.22}$$

Subsequently we use Euler's relation $\left[e^{jx} = \cos(x) + j\sin(x)\right]$ to rewrite the coefficient c_n in Eq. (5.20):

$$c_n = \frac{1}{T}\int_T f(t)[\cos(n\omega t) - j\sin(n\omega t)]dt \tag{5.23}$$

The expression in Eq. (5.23) is a complex number. Because we can represent any complex number by its real (a_n) and imaginary (jb_n) parts, we may simplify:

$$\frac{1}{T}\int_T f(t)[\cos(n\omega t) - j\sin(n\omega t)]dt = \frac{1}{2}(a_n - jb_n) \tag{5.24}$$

with the factor ½ in Eq. (5.24) chosen for convenience, as will be shown in the text below. We used the factor of ½ and coefficients a_n, b_n for a reason. The reason is that we recognize the terms in Eq. (5.24) as being similar to the coefficients in the real Fourier series (see Eqs. 5.12 and 5.18): the first term $\frac{1}{T}\int_T f(t)\cos(n\omega t)dt = \frac{1}{2}a_n$, and the second term $\frac{1}{T}\int_T f(t)j\sin(n\omega t)dt = -j\frac{1}{2}b_n$. Note that the real ($\Re$) components of c_n and c_{-n} can be related as $\Re(c_{-n}) = -\Re(c_n) = -\frac{1}{2}a_n$ and the imaginary ($\Im$) components as $\Im(c_{-n}) = -\Im(c_n) = -\frac{1}{2}b_n$. Now recall that in the real Fourier series a_n is associated with the cosine expressions (with even symmetry, Appendix 5.2) and that b_n is associated with the sine expressions (with odd symmetry, Appendix 5.2). If we now optimistically assume that we will be successful in relating the complex and real versions of the Fourier series, we can use these odd and even properties to state: $\Re(c_{-n}) = \frac{1}{2}a_n$, i.e., we simply changed the sign of $\Re(c_{-n})$ because the negative and positive values of cosine terms are equal anyway. Of course we cannot do the same change of sign for $\Im(c_{-n})$ because the sine terms have odd symmetry. To summarize the above, we conclude that $a_n = a_{-n}$ and $b_n = -b_{-n}$. Using these properties of a_n and b_n we relate c_n and c_{-n} as follows:

$$c_n = \frac{1}{2}(a_n - jb_n) \quad \text{and} \quad c_{-n} = \frac{1}{2}(a_{-n} - jb_{-n}) = \frac{1}{2}(a_n + jb_n) \tag{5.25}$$

We substitute these results for c_n and c_{-n} in $P(t)$ in Eq. (5.22) and get:

$$P(t) = \frac{1}{2}(a_0 - jb_0) + \sum_{n=1}^{\infty}\frac{1}{2}(a_n - jb_n)e^{jn\omega t} + \sum_{n=1}^{\infty}\frac{1}{2}(a_n + jb_n)e^{-jn\omega t} \tag{5.26}$$

Here we can set the coefficient for the imaginary DC component to zero ($b_0 = 0$), because a DC term is even. Furthermore, using Euler's relation for the exponentials ($e^{jn\omega t}$ and $e^{-jn\omega t}$) in the above expression:

$$\begin{aligned} P(t) = \frac{1}{2}a_0 &+ \sum_{n=1}^{\infty}\frac{1}{2}\underbrace{(a_n - jb_n)(\cos(n\omega t) + j\sin(n\omega t))}_{I} \\ &+ \sum_{n=1}^{\infty}\frac{1}{2}\underbrace{(a_n + jb_n)(\cos(n\omega t) - j\sin(n\omega t))}_{II} \end{aligned} \tag{5.27}$$

Evaluating the results for parts I and II in Eq. (5.27):

$$\begin{array}{lcccccccc} I & \rightarrow & a_n\cos(n\omega t) & + & ja_n\sin(n\omega t) & - & jb_n\cos(n\omega t) & - & j^2b_n\sin(n\omega t) \\ II & \rightarrow & a_n\cos(n\omega t) & - & ja_n\sin(n\omega t) & + & jb_n\cos(n\omega t) & - & j^2b_n\sin(n\omega t) \\ I+II & \rightarrow & 2a_n\cos(n\omega t) & - & 0 & + & 0 & - & 2j^2b_n\sin(n\omega t) \end{array}$$

As can be seen in the addition of $I + II$ above, the complex terms in the products under both the Σ operations cancel (using $j^2 = -1$). Substituting this finding in Eq. (5.27), the final result becomes:

$$\boxed{P(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}[a_n\cos(n\omega t) + b_n\sin(n\omega t)]} \tag{5.28}$$

which is the same as the formula for a real Fourier series in Eq. (5.1), i.e., the complex series is equal to the real series described above.

EXAMPLES

In this section we apply Fourier series to analyze time domain signals. In the first example we apply the real series from Eq. (5.1) to describe a ***triangular wave*** with period T and amplitude A shown in Fig. 5.5. From inspection of the waveform we can see directly that:

1. a DC component is absent, therefore $a_0 = 0$, and
2. the time series is even (Appendix 5.2), i.e., the sinusoidal odd components in the Fourier series are absent, therefore $b_n = 0$.

From these observations we conclude that we only have to calculate the a_n coefficients of Eq. (5.1) to obtain the representation of the real Fourier series. We can calculate these coefficients by integration over a full period from $-T/2$ to $T/2$. In order to avoid trying to integrate over the discontinuity at $t = 0$, we can break the function up into two components where we observe (Fig. 5.5) that:

3. $A(t) = A + 4At/T$ for $-T/2 \le t \le 0$, and
4. $A(t) = A - 4At/T$ for $0 \le t \le T/2$

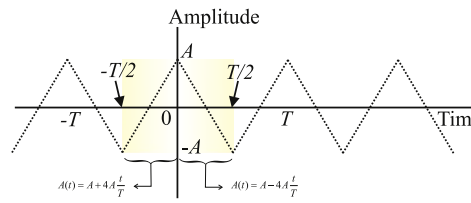


FIGURE 5.5 An even triangular waveform can be decomposed into a Fourier series that consists solely of cosine terms.

Applying Eq. (5.12) for a_n and integrating over the interval $-T/2$ to $T/2$ in these two domains produces:

$$a_n = \frac{2}{T}\int_{-T/2}^{0}\left(A + 4A\frac{t}{T}\right)\cos(n\omega t)dt + \frac{2}{T}\int_{0}^{T/2}\left(A - 4A\frac{t}{T}\right)\cos(n\omega t)dt \tag{5.29}$$

In this equation we can separate the terms for A and $4At/T$ and pull the constants A, 4, and T out of the integration:

$$\begin{aligned} a_n = &\underbrace{\frac{2A}{T}\int_{-T/2}^{0}\cos(n\omega t)dt + \frac{2A}{T}\int_{0}^{T/2}\cos(n\omega t)dt}_{I} \\ &\underbrace{+\frac{8A}{T^2}\int_{-T/2}^{0} t\cos(n\omega t)dt - \frac{8A}{T^2}\int_{0}^{T/2} t\cos(n\omega t)dt}_{II} \end{aligned} \tag{5.30}$$

Evaluating part I of Eq. (5.30):

$$\underbrace{\frac{2A}{T}\frac{1}{n\omega}[\sin(n\omega t)]_{-T/2}^{0} + \frac{2A}{T}\frac{1}{n\omega}[\sin(n\omega t)]_{0}^{T/2}}_{I} = 0 \tag{5.31}$$

The above result is zero because all terms with sin(0) are zero. Further, notice that $\omega = 2\pi f = 2\pi/T$; therefore the term with $\sin(n\omega T/2)$ is equal to $\sin(n\pi) = 0$, and the term with $\sin(n\omega(-T/2))$ is equal to $\sin(-n\pi) = 0$.

The integrals in part II in Eq. (5.30) can be evaluated by changing the variable from $t \rightarrow n\omega t$ and using integration by parts (Appendix 3.2):

$$\int t\cos(n\omega t)\,dt = \frac{1}{n^2\omega^2}\int \underbrace{(n\omega t)}_{u}\,\underbrace{\cos(n\omega t)\,d(n\omega t)}_{dv} \tag{5.32}$$

with: $\begin{cases} u = n\omega t \quad \rightarrow \quad du = d(n\omega t) \\ dv = \cos(n\omega t)\,d(n\omega t) \rightarrow v = \sin(n\omega t) \end{cases}$

Integrating by parts (Appendix 3.2): $\int \underbrace{(n\omega t)}_{u}\,\underbrace{\cos(n\omega t)\,d(n\omega t)}_{dv} =$

$$\underbrace{(n\omega t)\sin(n\omega t)}_{uv} - \int \underbrace{\sin(n\omega t)}_{v}\,\underbrace{d(n\omega t)}_{du} = (n\omega t)\sin(n\omega t) + \cos(n\omega t) + C$$

.

When applying the integration limits, the sine terms are again zero. When substituting our results in part *II* of Eq. (5.30) and taking into account that part *I* is zero, we obtain:

$$a_n = \frac{8A}{T^2}\frac{1}{n^2\omega^2}\left([\cos(n\omega t)]_{-T/2}^{0} - [\cos(n\omega t)]_{0}^{T/2}\right) \tag{5.33}$$

Using $\omega = 2\pi f = 2\pi/T$ to simplify this expression:

$$a_n = \frac{2A}{n^2\pi^2}\left(\left[\underbrace{\cos(0)}_{1} - \cos\underbrace{(-n\pi)}_{\substack{-1 \text{ for } n=odd \\ 1 \text{ for } n=even}}\right] - \left[\underbrace{\cos(n\pi)}_{\substack{-1 \text{ for } n=odd \\ 1 \text{ for } n=even}} - \underbrace{\cos(0)}_{1}\right]\right) \tag{5.34}$$

Depending on whether n is odd or even, the coefficients a_n are therefore $8\,A/(n\pi)^2$ or zero, respectively.

A second example is the ***square waveform*** we approximated with five sine waves in Fig. 5.2. From knowledge of this waveform we can conclude directly that:

1. a DC component is absent, therefore $a_0 = 0$, and
2. the time series is odd (Appendix 5.2), i.e., the cosine (even) components in the Fourier series are absent, therefore $a_n = 0$.

We thus conclude that we only have to calculate the b_n coefficients of Eq. (5.18) to obtain the expression for the real Fourier series. We can calculate these coefficients by integration over a full period from 0 to T. Again, splitting this period at the discontinuity:

3. $A(t) = A$ for $0 \le t \le T/2$, and
4. $A(t) = -A$ for $T/2 \le t \le T$

We use Eq. (5.18) for the calculation of coefficient b_n, integrating over period T in two steps:

$$\begin{aligned} b_n &= \frac{2}{T}\int_0^{T/2} A\sin(n\omega t)\,dt + \frac{2}{T}\int_{T/2}^{T} -A\sin(n\omega t)\,dt \\ &= \frac{2A}{T}\left[\underbrace{\int_0^{T/2}\sin(n\omega t)\,dt}_{-\frac{1}{n\omega}[\cos(n\omega t)]_0^{T/2}} - \underbrace{\int_{T/2}^{T}\sin(n\omega t)\,dt}_{-\frac{1}{n\omega}[\cos(n\omega t)]_{T/2}^{T}}\right] \end{aligned} \tag{5.35}$$

Using the results of this integration with $\omega = 2\pi f = 2\pi/T$, we simplify to:

$$b_n = \frac{2A}{T}\frac{1}{n\omega}\left([\cos(n\omega t)]_{T/2}^{T} - [\cos(n\omega t)]_{0}^{T/2}\right) =$$

$$\frac{A}{n\pi}\left(\left[\underbrace{\cos(2\pi n)}_{1} - \underbrace{\cos(\pi n)}_{\begin{matrix}-1\ for\ n=odd\\ 1\ \ for\ n=even\end{matrix}}\right] - \left[\underbrace{\cos(\pi n)}_{\begin{matrix}-1\ for\ n=odd\\ 1\ \ for\ n=even\end{matrix}} - \underbrace{\cos(0)}_{1}\right]\right) \tag{5.36}$$

from which we conclude that the coefficients b_n are $4A/n\pi$ or zero, respectively, for odd or even n. This result is in agreement with the MATLAB® script below that creates an approximation of a square wave with amplitude $A = 1$ (Fig. 5.2). In the example we only use $n = 1, 3, 5, 7, 9$ resulting in amplitudes of $4/\pi$, $4/(3\pi)$, $4/(5\pi)$, $4/(7\pi)$, and $4/(9\pi)$.

The following is a part of the MATLAB® program (pr5_1.m) that creates the harmonics, the amplitude coefficients are indicated in bold.

```
s1=(4/pi)*sin(2*pi*f*t);
% the (4/pi) factor is to get a total amplitude of 1
% define harmonics with odd frequency ratio.
s3= (4/pi)*(1/3) * sin(2*pi*3*f*t);
s5= (4/pi)*(1/5) * sin(2*pi*5*f*t);
s7= (4/pi)*(1/7) * sin(2*pi*7*f*t);
s9= (4/pi)*(1/9) * sin(2*pi*9*f*t);
```

Sinewaves s1−s9 correspond to the waves in the left panel of Fig. 5.2; amplitudes $4/\pi - 4/(9\pi)$ are depicted in the spectral plot in Fig. 5.2. If we extend the findings from the two examples above to the complex Fourier series, we can conclude that:

- In even functions only the a_n coefficients are required. In the complex series approach this translates into a series with real values only.
- With odd functions the b_n coefficients are required. If there is no DC component ($a_0 = 0$ in Eq. 5.1), this translates into a series with solely imaginary numbers.
- A function that has neither even nor odd properties can be composed of even and odd components (Appendix 5.2); this results in a Fourier series that includes both real and imaginary values.

Because of the close relationship between the Fourier transform, introduced in Chapter 6, and the complex Fourier series, these conclusions remain relevant for the transform as well.

APPENDIX 5.1

In general, functions f_{mn} that produce a nonzero value only if $m = n$ are called orthogonal functions and they play an important role in signal processing. In this chapter we derive the Fourier series using this property. More precisely, in this appendix we show that the coefficients a_n and b_n in Eqs. (5.6), (5.12), and (5.18) can be found because the following functions are orthogonal:

$$\int_T \sin(n\omega t)\sin(m\omega t)dt \qquad \int_T \cos(n\omega t)\cos(m\omega t)dt \tag{A5.1-1}$$

Another important integral that always (also if $m = n$) evaluates to zero is:

$$\int_T \sin(n\omega t)\cos(m\omega t)dt \tag{A5.1-2}$$

In the following we show that the underlying reason for these properties in Eqs. (A5.1-1) and (A5.1-2) is that the integral of a sine or cosine wave over an integral number of periods (NT) with $N = 1, 2, 3, \ldots$ evaluates to zero:

$$\int_T \cos(N\omega t) = 0 \text{ and } \int_T \sin(N\omega t) = 0 \tag{A5.1-3}$$

From the graphical presentation of a sine or cosine, this is clear without even performing the integration (Fig. A5.1-1); the areas enclosed by the waveform over one or more period(s) cancel since these functions are symmetric across the $y = 0$ axis.

An additional prerequisite to continue the derivation is the trigonometric functions that equate sine/cosine products to sums:

$$\begin{aligned}\cos(A)\cos(B) &= \tfrac{1}{2}[\cos(A-B)+\cos(A+B)]\\ \sin(A)\sin(B) &= \tfrac{1}{2}[\cos(A-B)-\cos(A+B)]\\ \cos(A)\sin(B) &= \tfrac{1}{2}[\sin(A-B)-\sin(A+B)]\end{aligned} \tag{A5.1-4}$$

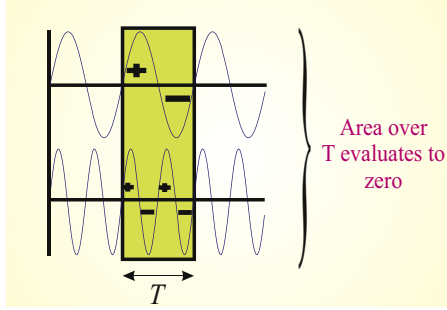


FIGURE A5.1-1 The areas enclosed by the positive and negative regions of a cosine or sine wave over at least one full period cancel; therefore the integral of these trigonometric functions over one or more periods evaluates to zero.

For $A \neq B$ all the above products generate two cosine or sine terms with a frequency $(A - B)$ or $(A + B)$. ***In Eq. (A5.1-1) these become functions of ω or harmonics of the base frequency ω***. For instance, if $A = 5\omega t$ and $B = 3\omega t$, the terms that are generated are the harmonics $2\omega t$ and $8\omega t$. As pointed out above (Eq. A5.1-3), integrals of these harmonics over a period of base frequency ω all evaluate to zero, e.g.,:

$$\int_T \frac{1}{2}[\cos(2\omega t) + \cos(8\omega t)]dt = \frac{1}{2}\int_T \cos(2\omega t)dt + \frac{1}{2}\int_T \cos(8\omega t)dt = 0 + 0 \tag{A5.1-5}$$

Special cases for products of trigonometric identities with $A = B$ can easily be derived from Eq. (A5.1-4):

$$\begin{aligned} \cos(A)\cos(A) &= ½[\cos(0) + \cos(2A)] \\ \sin(A)\sin(A) &= ½[\cos(0) - \cos(2A)] \\ \cos(A)\sin(A) &= ½[\sin(0) - \sin(2A)] \end{aligned} \tag{A5.1-6}$$

Using $\cos(0) = 1$ we can conclude that integrals of the first two equations over a period T evaluate to $T/2$, for instance for the top equation in (A5.1-6):

$$\int_T \frac{1}{2}[1 + \cos(2A)]dt = \frac{1}{2}\int_T 1dt + \frac{1}{2}\int_T \cos(2A)dt = \frac{1}{2}[t]_0^T + 0 = \frac{1}{2}T \tag{A5.1-7}$$

Using $\sin(0) = 0$, the last equation in (A5.1-6) evaluates to zero in all cases:

$$\int_T \frac{1}{2}[0 + \sin(2A)]dt = \frac{1}{2}\int_T \sin(2A)dt = 0 \tag{A5.1-8}$$

In ***summary*** we conclude from the above that the integral (over one or more periods) of sine × sine and cosine × cosine products are ***orthogonal*** functions, whereas the integral of a sine × cosine product always evaluates to zero:

$$\begin{aligned}
&\int_T \cos(n\omega t)\cos(m\omega t)dt = \begin{cases} T/2 & \text{for } m = n \\ 0 & \text{otherwise} \end{cases} \\
&\int_T \sin(n\omega t)\sin(m\omega t)dt = \begin{cases} T/2 & \text{for } m = n \\ 0 & \text{otherwise} \end{cases} \\
&\int_T \sin(n\omega t)\cos(m\omega t)dt = 0 \text{ for all } m \text{ and } n
\end{aligned} \tag{A5.1-9}$$

The properties in (A5.1-9) are used in the text to derive the expressions for the coefficients in the real Fourier series.

APPENDIX 5.2

In the development of the Fourier analysis the distinction between odd and even symmetric functions plays an important role. Here we show that odd and even functions can be used to describe any function $f(t)$.

As Fig. A5.2-1 shows, even functions are symmetrical around the vertical axis: $f(t) = f(-t)$. An odd function is symmetric by reflection across both axes: $f(t) = -f(-t)$. Examples of even and odd functions are $\cos(t)$ and $\sin(t)$, respectively. First we will show that any function $f(t)$ can be used to create an even or an odd function.

$$f_{even} = (f(t) + f(-t))/2. \tag{A5.2-1}$$

It is easily confirmed that this is an even function by substituting $-t$, from which it follows that $f_{even}(t) = f_{even}(-t)$. The odd component can be created as:

$$f_{odd} = (f(t) - f(-t))/2 \tag{A5.2-2}$$

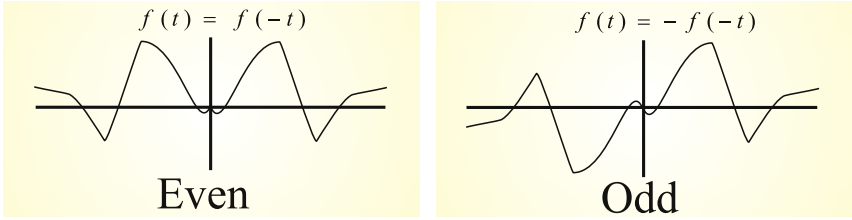


FIGURE A5.2-1 Example of an even and an odd function.

where substituting $-t$ reveals that $f_{odd}(t) = -f_{odd}(-t)$. Now we use the above results to show that the arbitrary selected function $f(t)$ can be considered as the sum of odd and even components:

$$f(t) = f_{even} + f_{odd} = \frac{f(t) + f(-t)}{2} + \frac{f(t) - f(-t)}{2} = \frac{2f(t)}{2} = f(t) \quad \text{(A5.2-3)}$$

EXERCISES

5.1 Fourier series
- a. Show that $\int_T \sin(n\omega t)\sin(m\omega t)dt$ is an orthogonal function.
- b. What do you need to change to make the function orthonormal?
- c. Show that $\int_T \sin(n\omega t)\cos(m\omega t)dt = 0$. Is this an orthogonal function?

5.2 Determine all coefficients for the real Fourier series of
- a. $4\cos(2\omega t)$
- b. $6\sin(2\omega t)$
- c. $4\cos(2\omega t) + 6\sin(2\omega t)$
- d. For all cases above, plot the coefficients versus ω

5.3 Repeat Exercise 5.2 for the complex Fourier series. (Hint: recall that Euler's relationship is useful to relate sine, cosine, and exponentials.)

5.4 The wave depicted in Fig. E5.4 is characterized by a period T and amplitude A.
- a. Does this wave show even or odd symmetry?
- b. Find the Fourier series of the depicted triangular wave $f(x)$. (Hint: Use a piecewise approach similar to the examples in Section 5.4 to solve this problem.)

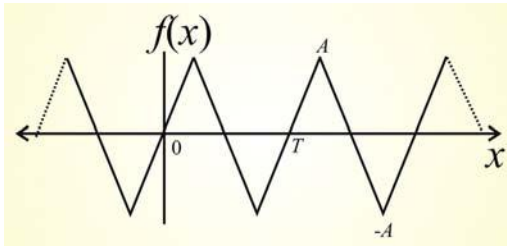


FIGURE E5.4 A "sine" triangle wave.

5.5 Determine coefficients a_n and b_n (for $n = 0 \rightarrow \infty$) in the real Fourier series and coefficients c_n (for $n = -\infty \rightarrow \infty$) in the complex Fourier series for the following functions.
a. $\cos(\omega_0 t)$
b. $\sin(\omega_0 t)$
(Hint: recall that Euler's relationship is useful to relate sine, cosine, and exponentials.)

5.6 Are the following functions even symmetric, odd symmetric, or neither? In case your answer is neither, determine the function's even and odd components.
a. $y = ax$
b. $y = ax^2$
c. $y = ax + 5$

5.7 Using the square waveform of the second example in Section 5.4,
a. modify MATLAB® script pr5_1.m to include sixth, seventh and eighth terms in the approximation of the square wave, and
b. plot the result.
c. How does your result differ from the square wave in Fig. 5.2?

CHAPTER

6

Continuous, Discrete, and Fast Fourier Transform

6.1 INTRODUCTION

In this chapter, the Fourier transform is related to the complex Fourier series presented in Chapter 5 (see overview in Fig. 5.3). The Fourier transform in continuous time (or space) is referred to as the continuous Fourier transform (CFT). In Section 6.3, we develop a discrete version of the Fourier transform (DFT) and explore an efficient algorithm for calculating it. This efficient algorithm is known as the fast Fourier transform (FFT). In the next chapter (Chapter 7) we show an example of the use of the CFT in the reconstruction of images and an application of the DFT for calculation of the power spectra of simulated and recorded signals.

6.2 THE FOURIER TRANSFORM

The Fourier transform is an operation that transforms data from the time (or spatial) domain into the frequency domain. In this section we demonstrate that the transform can be considered as the limiting case of the complex Fourier series.

In Fig. 6.1 we illustrate how one can create an arbitrary function $f(t)$ from a periodic signal $f_{T_0}(t)$ by increasing its period T_0 to ∞. This thought process is the basis on which the CFT is derived from the complex Fourier series. For clarity, we reiterate the expressions for the complex series and its coefficients that we derived in Chapter 5:

$$P(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega t}, \tag{6.1a}$$

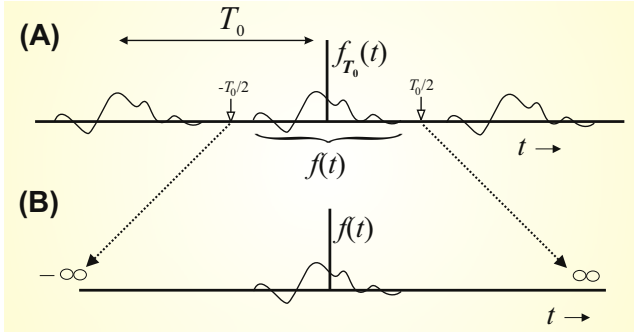


FIGURE 6.1 (A) Periodic function $f_{T_0}(t)$ and (B) function $f(t)$ derived from one cycle.

with the coefficients defined as:

$$c_n = \frac{1}{T}\int_T f(t)e^{-jn\omega t}dt \tag{6.1b}$$

The first step is to establish the coefficient c_n for the series $f_{T_0}(t)$ in Fig. 6.1A. Hereby we integrate over ***one cycle*** of the function $f_{T_0}(t)$, which equals $f(t)$ over the period from $-T_0/2$ to $T_0/2$:

$$c_n = \frac{1}{T_0}\int_{-T_0/2}^{T_0/2} f(t)e^{-jn\omega_0 t}dt \tag{6.2}$$

In Eq. (6.2) above, we include the period T_0, and the fundamental angular frequency ω_0. The relationship between these parameters is given by: $T_0 = \frac{1}{f_0} = \frac{2\pi}{\omega_0}$ (with f_0 the fundamental frequency in Hz). So far we are still simply applying the complex Fourier series to $f(t)$ as one cycle of $f_{T_0}(t)$. The second step is to stretch the period parameter T_0 to ∞ (which also means that the fundamental angular frequency $\omega_0 \rightarrow 0$), and to define c_n^* as:

$$c_n^* = \lim_{\substack{T_0 \rightarrow \infty \\ \omega_0 \rightarrow 0}} \int_{-T_0/2}^{T_0/2} f(t)e^{-jn\omega_0 t}dt \tag{6.3}$$

The coefficient c_n^* in Eq. (6.3) is very similar to the limit of c_n in Eq. (6.2) but ***note that we smuggled a 1/T_0 factor out of the expression!*** Stated a bit more formally we say that c_n^* is defined by Eq. (6.3), thereby avoiding a division by $T_0 \rightarrow \infty$. Because $\omega_0 \rightarrow 0$, we may consider the product $n\omega_0$ a continuous scale of the angular frequency ω $\left(\text{i.e., } \lim_{\omega_0 \rightarrow 0} n\omega_0 = \omega\right)$. Further,

representing the complex coefficients c_n^* in a function $F(j\omega)$, we may rewrite Eq. (6.3) as:

$$\boxed{F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt} \tag{6.4}$$

Eq. (6.4) is the definition of the continuous Fourier transform (CFT). In some texts $F(\omega)$ is used instead of $F(j\omega)$, subsuming the constant j into the right-hand side. Another common notation substitutes $\omega = 2\pi f$ resulting in: $F(f) = \int_{-\infty}^{\infty} f(t)e^{-j2\pi ft}dt$, which simply represents the results in units of Hz.

Given Eq. (6.4) we will now show that the inverse transform also follows from the complex Fourier series. Combining Eqs. (6.1b) and (6.4) and using $\omega = n\omega_0$, we have:

$$c_n = \frac{1}{T_0}F(jn\omega_0) \tag{6.5}$$

Note that the $1/T_0$ factor is reintroduced! The $1/T_0$ correction maintains the consistency of the transform with its inverse.

Using Eq. (6.1a) for the complex Fourier series:

$$\begin{aligned} f_{T_0}(t) &= \sum_{n=-\infty}^{\infty} \frac{1}{T_0}F(jn\omega_0)e^{jn\omega_0 t}, \quad \text{using} \quad \frac{1}{T_0} = \frac{\omega_0}{2\pi} \quad \rightarrow \\ f_{T_0}(t) &= \frac{1}{2\pi}\sum_{n=-\infty}^{\infty} F(jn\omega_0)e^{jn\omega_0 t}\omega_0 \end{aligned} \tag{6.6}$$

Now we let $T_0 \rightarrow \infty$ meaning that $\omega_0 \rightarrow 0$. If we change the notation $\omega_0 = \Delta\omega$ to use as a limiting variable, Eq. (6.6) becomes:

$$f(t) = \lim_{\substack{T_0 \rightarrow \infty \\ \omega_0 \rightarrow 0}} f_{T_0}(t) = \lim_{\Delta\omega \rightarrow 0} \frac{1}{2\pi}\sum_{n=-\infty}^{\infty} F(jn\Delta\omega)e^{jn\Delta\omega t}\Delta\omega \tag{6.7}$$

We can interpret the sum in Eq. (6.7) as calculating the area under the continuous function $F(j\omega)\,e^{j\omega t}$ using arbitrarily narrow slices in the limit. This interpretation generates the result for the inverse Fourier transform:

$$\boxed{f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(j\omega)e^{j\omega t}d\omega} \tag{6.8}$$

If $\omega = 2\pi f$ is substituted in Eq. (6.8) we get: $f(t) = \int_{-\infty}^{\infty} F(f)e^{j2\pi ft}df$.

Eqs. (6.4) and (6.8) form a consistent pair for the Fourier transform and the

inverse Fourier transform, respectively. With the exception of the $1/T_0$ factor, there is a direct correspondence between the transform and the complex series, whereby the transform can be considered as a series in the limit where $T_0 \to \infty$. Both the equations for the transform and its inverse apply for continuous time or space. Therefore this flavor of spectral analysis is called the CFT.

6.2.1 Examples of Continuous Fourier Transform Pairs

Eqs. (6.4) and (6.8) can be used to establish Fourier transform pairs; for complicated functions, it is common practice to use tables (e.g., Table 6.1) that summarize the pairs, or software tools to determine integrals. Here we focus on a few simple examples and associated interpretations relevant for signal analysis. First we consider the signal $\delta(t)$, known as the Dirac delta function; its Fourier transform is given by:

$$F(j\omega) = \int_{-\infty}^{\infty} \delta(t) e^{-j\omega t} dt = e^{-j\omega 0} = 1 \tag{6.9}$$

The above integral is evaluated using the sifting property of the delta function (Eq. 2.8). From a signal processing standpoint it is interesting to see that ***the time domain Dirac delta function corresponds to all frequencies in the frequency domain***.

We noticed when discussing the transforms that the equations for the Fourier transform and its inverse are fairly similar. Repeating Eqs. (6.4) and (6.8):

$$F(j\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt \Leftrightarrow f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(j\omega) e^{j\omega t} d\omega \tag{6.10}$$

Clearly the transform and its inverse are identical with the exception of the $1/2\pi$ scaling factor and the sign of the imaginary exponent. This means that one can derive the inverse transform from the transform and vice versa by correcting for the scaling and the sign of the variable over

TABLE 6.1 Examples of Fourier Transform Pairs

Time/Spatial Domain $f(t)$	Frequency Domain $F(\omega)$
$\delta(t)$	1
1	$2\pi\,\delta(\omega)$
$\cos(\omega_0 t)$	$\pi[\delta(\omega+\omega_0)+\delta(\omega-\omega_0)]$
$\sin(\omega_0 t)$	$j\pi[\delta(\omega+\omega_0)-\delta(\omega-\omega_0)]$

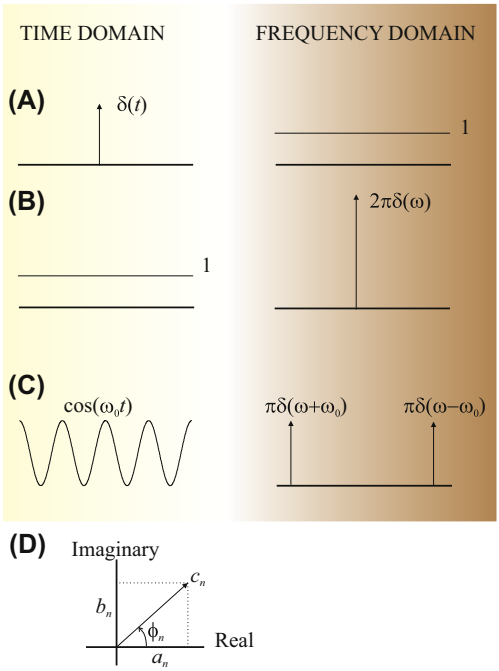


FIGURE 6.2 Common Fourier transform pairs. (A) A Dirac impulse function in the time domain is represented by all frequencies in the frequency domain. (B) This relationship can be reversed to show that a DC component in the time domain generates an impulse function at a frequency of zero. (C) A pure (cosine) wave shows peaks at $\pm\,\omega_0$ in the frequency domain. (D) In general a coefficient c_n being part of $F(j\omega)$ may contain both real (a_n) and imaginary (b_n) parts (represented here in a polar plot).

which one integrates. This is the so-called ***duality property*** which results in some interesting and useful parallels between time and frequency domain representations (Fig. 6.2).

Applying the above Fourier transform and inverse transform relationship to the Dirac impulse $\delta(t)$, one can conclude that the time domain equivalent for a delta function in the frequency domain $\delta(-\omega)$ must be the constant function $f(t) = 1/2\pi$. Because the scaling is a constant (not depending on ω) and $\delta(-\omega) = \delta(\omega)$, one can say that $1 \Leftrightarrow 2\pi\delta(\omega)$ forms a time domain−frequency domain pair; or in a different notation:

$$F(j\omega) = 2\pi\delta(\omega) = \int_{-\infty}^{\infty} 1e^{-j\omega t}dt \tag{6.11}$$

This outcome can be validated by evaluating the inverse Fourier transform $f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}[2\pi\delta(\omega)]e^{j\omega t}d\omega$. Using the sifting property of $\delta(\omega)$

it can be seen that this expression evaluates to 1. Interpreting this property from a signal processing point of view, this result indicates that an additive constant (1 in this case), or offset, representing ***a time domain DC component corresponds to a peak in the frequency domain at a frequency of zero***.

Another important example is the transform of a cosine function ($\cos\omega_0 t$). This function gives us insight into how a basic pure sinusoid transforms into the frequency (Fourier) domain. Intuitively, we would expect such a function to have a singular representation in the frequency domain. We attack this problem by expressing the cosine as the sum of two complex exponentials (using Euler's relation): $\cos\omega_0 t = \frac{1}{2}\left[e^{-j\omega_0 t} + e^{j\omega_0 t}\right]$. The transform of this function becomes:

$$\begin{aligned} F(j\omega) &= \frac{1}{2}\int_{-\infty}^{\infty}\left[e^{-j\omega_0 t} + e^{j\omega_0 t}\right]e^{-j\omega t}dt \\ &= \frac{1}{2}\left[\int_{-\infty}^{\infty} e^{-j(\omega+\omega_0)t}dt + \int_{-\infty}^{\infty} e^{-j(\omega-\omega_0)t}dt\right] \end{aligned} \tag{6.12}$$

Both integrals in this expression can be evaluated easily using the result for the exponential equation that we obtained above $\int_{-\infty}^{\infty} e^{-j\omega t}dt = 2\pi\delta(\omega)$, replacing ω with $\omega + \omega_0$ and $\omega - \omega_0$, respectively. Thus the Fourier transform of a cosine function (a real and even symmetric function) results in:

$$\cos(\omega_0 t) \Leftrightarrow \pi[\delta(\omega + \omega_0) + \delta(\omega - \omega_0)] \tag{6.13}$$

this is also real and even; that is, a delta function at an angular frequency of ω_0 and another at $-\omega_0$. While the impulse in the positive frequency domain is straightforward, the concept of negative frequency, originating from the complex series representation in Fourier analysis (Chapter 5, Section 5.3), defies common sense interpretations. Following a similar logic as that applied above it can be shown that a real and odd symmetric function results in an imaginary odd function:

$$\sin(\omega_0 t) \Leftrightarrow j\pi[\delta(\omega + \omega_0) - \delta(\omega - \omega_0)] \tag{6.14}$$

More commonly, functions that are not purely odd or even have both real and imaginary parts for each frequency component ω_n, i.e.,:

$$F(j\omega_n) = c_n = a_n + jb_n \tag{6.15}$$

That is, for each frequency component in the time domain, we obtain a complex number in the frequency domain. This number is proportional to the cosine and sine amplitudes.

In Chapter 7 we will show how to combine the real and imaginary parts into a metric representing the power for each frequency component in a signal.

6.3 DISCRETE FOURIER TRANSFORM AND THE FAST FOURIER TRANSFORM ALGORITHM

6.3.1 Relationship Between Continuous and Discrete Fourier Transform

The continuous and discrete Fourier transforms and their inverses are related but not identical. For the discrete pair we use a discrete time scale and a discrete frequency scale (Fig. 6.3). Because we want to apply the discrete transform to sampled real-world signals, both the time and frequency scales must also necessarily be finite. Furthermore, we can establish that both scales must be related. For example, in a signal that is observed over a 10-s interval T and sampled at an interval $\Delta t = 1$ ms (0.001 s), these parameters determine the range and precision of the discrete Fourier transform of that signal. It is intuitively clear that in a 10-s interval we cannot reliably distinguish frequencies below a precision of $\Delta f = 1/T = 1/10$ Hz and that the maximum frequency that fits within the sample interval is $1/\Delta t = 1/0.001 = 1000$ Hz. In angular frequency terms,

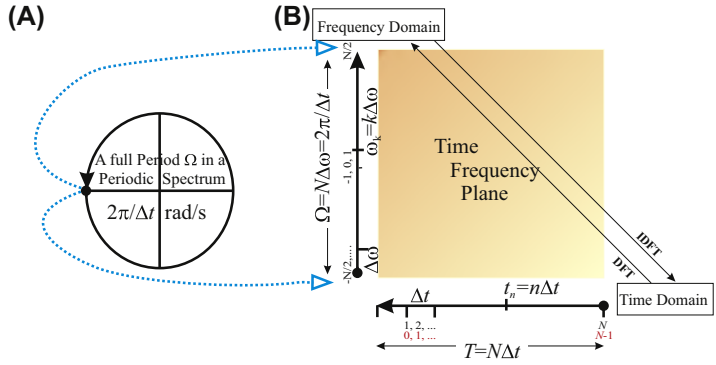


FIGURE 6.3 The time domain and frequency domain scales in the discrete Fourier transform. The Fourier spectrum is periodic, represented by a circular scale in (A). This circular frequency domain scale is mapped onto a line represented by the ordinate in (B). The abscissa in (B) is the time domain scale; note that on the frequency (vertical) axis, the point $-N/2$ is included and $N/2$ is not. Each sample in the time domain represents the preceding sample interval. Depending on which convention is used, the first sample in the time domain is either counted as the zeroth sample (indicated in red) or the first one (indicated in black).

the precision and maximum frequency translate into a step size of $\Delta\omega = 2\pi \times 1/10$ rad/s and a range of $\Omega = 2\pi \times 1000$ rad/s (Fig. 6.3).

Note: From earlier discussions about the Nyquist frequency (Chapter 2) we know that the highest frequency we can observe is actually $2\pi \times 500$ rad/s, ***half*** of Ω. This point will resurface in Chapter 7 when the actual spectra are introduced and we find that these spectra contain two symmetric halves.

The discrete approximation $F_a(j\omega)$ of the CFT $F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$ sampled over a finite interval including N samples is:

$$F_a(j\omega_k) = \sum_{n=1}^{N} f(t_n)\exp(-j\omega_k t_n)\Delta t \tag{6.16}$$

Discrete time signals are usually created by sampling a continuous time process; each sample thereby represents the signal immediately preceding analog-to-digital conversion. Using this approach, we have a sampled series representing N intervals of length Δt each (Fig. 6.3). For several reasons, it is common practice to use a range for n from 0 to $N-1$ instead of 1 to N. Furthermore, in Eq. (6.16) the time (t) and angular frequency (ω) are represented by discrete variables as indicated in Fig. 6.3. With $t_n = n\,\Delta t$ and $\omega_k = k\,\Delta\omega = k\,2\,\pi/N\Delta t$ (Fig. 6.3), we can write $F_a(j\omega)$ as:

$$F_a(j\omega_k) = \Delta t \sum_{n=0}^{N-1} f(t_n)e^{-j\frac{2\pi}{N}kn} = \Delta t \sum_{n=0}^{N-1} f(t_n)W_N^{kn} \tag{6.17}$$

where W_N^{kn} is a notational simplification of the exponential term. ***Smuggling Δt out of the above expression***, changing $f(t_n)$ to $x(n)$, and $F_a(j\omega_k)$ to $X(k)$ yields the standard definition for the DFT:

$$\boxed{X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn}} \tag{6.18}$$

Similarly, the inverse continuous Fourier transform (ICFT) can be approximated by:

$$f_a(t_n) = \frac{1}{2\pi} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} F_a(j\omega_k)e^{j\omega_k t_n}\Delta\omega \tag{6.19}$$

Note that the upper summation limit does not include $N/2$; due to the circular scale of ω, $-N/2$ and $N/2$ are the same (Fig. 6.3). Changing the

range of the summation from $-N/2 \rightarrow (N/2)-1$ into $0 \rightarrow N-1$ and $\Delta\omega = 2\pi/N\Delta t$ (see the axis in Fig. 6.3), Eq. (6.19) yields:

$$f_a(t_n) = \frac{1}{2\pi}\sum_{k=0}^{N-1} F_a(j\omega_k)e^{j\omega_k t_n}\frac{2\pi}{N\Delta t} = \frac{1}{N\Delta t}\sum_{k=0}^{N-1} F_a(j\omega_k)e^{j\omega_k t_n} \tag{6.20}$$

We now use $t_n = n\Delta t$ and $\omega_k = \frac{k2\pi}{N\Delta t}$ (see Fig. 6.3), ***smuggle Δt back***, change $f_a(t_n)$ to $x(n)$, and $F_a(j\omega_k)$ to $X(k)$, thereby obtaining the expression for the discrete inverse Fourier transform:

$$\boxed{x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j\frac{2\pi}{N}kn} = \frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-kn}} \tag{6.21}$$

To keep the transform $\Leftrightarrow$ inverse transform pair consistent, the division by Δt is corrected in the inverse discrete Fourier transform (IDFT), where the expression is multiplied by Δt.

6.3.2 The Twiddle Factor

The weighting factor introduced as W_N in the above formulae plays an important role in the practical development of DFT algorithms including the optimized one known as the FFT. The efficiency of this algorithm relies crucially on the fact that this factor, also known as the "twiddle" ***factor,*** is periodic.

6.3.3 Discrete Fourier Transform Versus a Base-2 Fast Fourier Transform

The basic idea used to optimize the DFT algorithm involves using the periodicity in the twiddle factor to combine terms and therefore reduce the number of computationally demanding multiplication steps required for a given number of samples (Cooley and Tukey, 1965). Specifically, the standard formulation of the DFT of a time series with N values requires N^2 multiplications for a time series with N points; whereas, the FFT requires only $N\log_2(N)$ multiplications.

For example, consider a four-point time series: $x(0)$, $x(1)$, $x(2)$, $x(3)$ and its DFT $X(0)$, $X(1)$, $X(2)$, $X(3)$. For $N = 4$ each of the X values is calculated with:

$$X(k) = \sum_{n=0}^{3} x(n)W_4^{kn} \tag{6.22}$$

If we were to perform the DFT directly from Eq. (6.22) we would have four multiplications for each $X(k)$; since we have $X(0)-X(3)$ this leads to a

total of $4 \times 4 = 16$ multiplications. However, since the expression in Eq. (6.22) is a summation, we can split the problem into odd and even components:

$$X(k) = \sum_{r=0}^{1} x(2r) W_4^{2rk} + \sum_{r=0}^{1} x(2r+1) W_4^{(2r+1)k} \tag{6.23}$$

The second twiddle factor can be separated into two factors:

$$W_4^{(2r+1)k} = W_4^{2rk} W_4^k \tag{6.24}$$

Expanding the summations for $X(0)-X(3)$ and combining terms we get:

$$\begin{aligned}
X(0) &= x(0)W_4^0 + x(2)W_4^0 + x(1)W_4^0 W_4^0 + x(3)W_4^0 W_4^0 \\
&= \left[x(0) + x(2)W_4^0\right] + W_4^0\left[x(1) + x(3)W_4^0\right] \\
X(1) &= x(0)W_4^0 + x(2)W_4^2 + x(1)W_4^1 W_4^0 + x(3)W_4^1 W_4^2 \\
&= \left[x(0) + x(2)W_4^2\right] + W_4^1\left[x(1) + x(3)W_4^2\right] \\
X(2) &= x(0)W_4^0 + x(2)W_4^4 + x(1)W_4^2 W_4^0 + x(3)W_4^2 W_4^4 \\
&= \left[x(0) + x(2)W_4^4\right] + W_4^2\left[x(1) + x(3)W_4^4\right] \\
X(3) &= x(0)W_4^0 + x(2)W_4^6 + x(1)W_4^3 W_4^0 + x(3)W_4^3 W_4^6 \\
&= \left[x(0) + x(2)W_4^6\right] + W_4^3\left[x(1) + x(3)W_4^6\right]
\end{aligned} \tag{6.25}$$

Note that with the exception of the first line in Eq. (6.25) we used $W_4^0 = 1$. In the first equation we kept W_4^0 in the expression solely to emphasize the similarity between all the expressions for $X(k)$. Further, we exploit the fact that the twiddle factors (W) represent periodic exponentials (Fig. 6.4B), i.e.,

$$\begin{aligned}
W_4^4 &= \exp\left\{j\left(-\frac{2\pi}{4}\right)4\right\} = 1 = W_4^0 \text{ and} \\
W_4^6 &= \exp\left\{j\left(-\frac{2\pi}{4}\right)6\right\} = -1 = W_4^2
\end{aligned} \tag{6.26}$$

In a MATLAB® script representing Eq. (6.25) may look like:

```
X(0+1)=(x(0+1)+x(2+1)*W0)+W0*(x(1+1)+x(3+1)*W0);
X(1+1)=(x(0+1)+x(2+1)*W2)+W1*(x(1+1)+x(3+1)*W2);
X(2+1)=(x(0+1)+x(2+1)*W0)+W2*(x(1+1)+x(3+1)*W0);
X(3+1)=(x(0+1)+x(2+1)*W2)+W3*(x(1+1)+x(3+1)*W2);
```

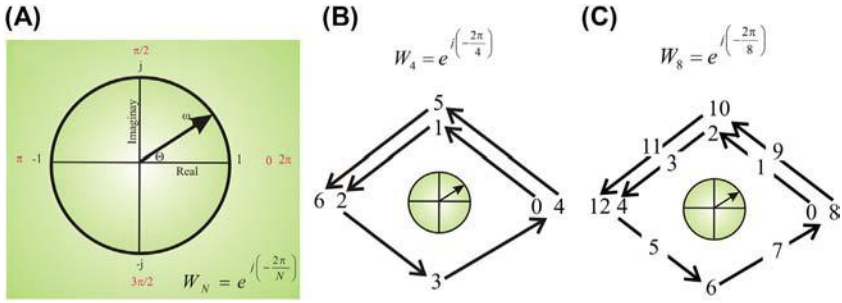


FIGURE 6.4 Periodicity of the twiddle factor W_N. (A) The values of Θ are indicated in red and the real and imaginary components in black. For instance $-1 + j0$ is associated with $\Theta = \pi$, $0 + j$ is associated with $\Theta = \pi/2$, etc. It can be seen that for $\Theta = 0$ or 2π, the values are identical $(1 + j0)$ due to the periodicity of W_N. In (B) and (C) concrete examples are provided for the periodicity of a four-point (W_4) and eight-point (W_8) algorithm. The numbers correspond to the powers of the twiddle factor (e.g., $0 \rightarrow W_4^0$; $1 \rightarrow W_4^1$; $2 \rightarrow W_4^2$, etc.): in case of $N = 4$, a cycle is completed in four steps; whereas for $N = 8$ the cycle is completed in eight steps. In the first case (B): $W_4^0 = W_4^4$, $W_4^1 = W_4^5$, and $W_4^2 = W_4^6$. In the second example (C): $W_8^0 = W_8^8$, $W_8^2 = W_8^{10}$, etc.

In this script, the time series is **x**, its transform is **X**, and the twiddle factors are **W0**−**W3.** Parenthetically, all indices are augmented with one because MATLAB® does not allow zero indices for vectors.

You may note that there are still 12 multiplications here, an improvement over the $4 \times 4 = 16$ multiplications for the brute force approach, but more than the expected $4\log_2(4) = 8$ multiplications. However, if we take advantage of the repeated multiplications in the above algorithm (i.e., x(2+1)*W0, x(2+1)*W2, x(3+1)*W0, and x(3+1)*W2) we end up with $12 - 4 = 8$ multiplications.

It is easier to see the algorithm flow in a diagram where the nodes are variables and the lines represent the operations on those variables (Fig. 6.5). In this example the input variables (left) are added in the output. In two of the cases, the input is multiplied by a twiddle factor (i.e., W_x and W_y in Fig. 6.5).

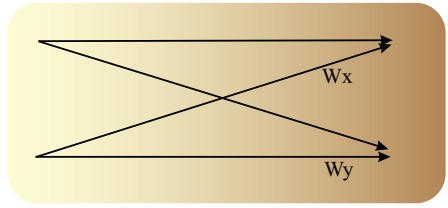


FIGURE 6.5 A flow diagram that forms the basis of the fast Fourier transform (FFT) algorithm. The base diagram is known as the **FFT butterfly.**

6.3.4 Examples

1. A four-point example

The following listing is an example of a four-point FFT MATLAB® Script (Fig. 6.6). To clearly show the specific features of the FFT algorithm, the program has not been optimized to avoid redundant operations.

```
% pr6_1.m
% A four point FFT
clear

x(0+1)=0;              % input time series x(n)
x(1+1)=1;              % Indices are augmented by 1
x(2+1)=1;              % MATLAB indices start at 1
x(3+1)=0;              % instead of 0

W4=exp(j*2*pi/4);      % the W4 twiddle factor
W0=W4^0;               % and the 0-3rd powers
W1=W4^1;
W2=W4^2;
W3=W4^3;

X(0+1)=(x(0+1)+x(2+1)*W0)+W0*(x(1+1)+x(3+1)*W0);
X(1+1)=(x(0+1)+x(2+1)*W2)+W1*(x(1+1)+x(3+1)*W2);
X(2+1)=(x(0+1)+x(2+1)*W0)+W2*(x(1+1)+x(3+1)*W0);
X(3+1)=(x(0+1)+x(2+1)*W2)+W3*(x(1+1)+x(3+1)*W2);

% Check with MATLAB fft command
fx=fft(x);

figure
hold
plot(X);
plot(fx,'r+');
```

2. An eight-point example

Evaluate the diagram in Fig. 6.7 with the **MATLAB® *script pr6_2.m***. While the signal vector indices seem rather arbitrarily ordered, a binary representation can make this indexing more straightforward; Table 6.2

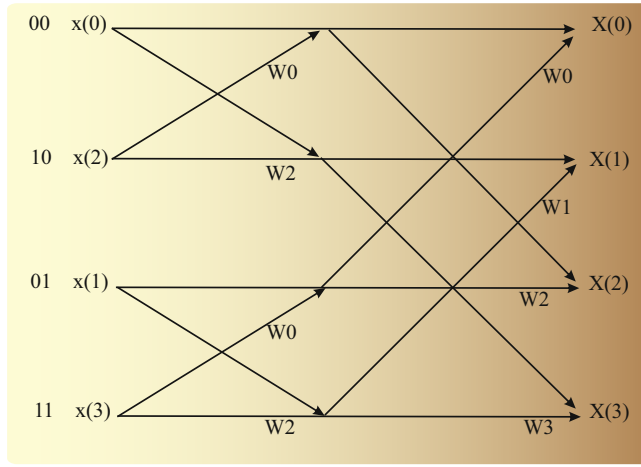


FIGURE 6.6 Diagram for a four-point fast Fourier transform. See the following for the diagram's implementation in MATLAB® script. Here we used $W_4^0 = W_4^4$ and $W_4^2 = W_4^6$ (Fig. 6.4) to optimize the algorithm. Note that we only indicate the superscript value of the twiddle factor in the diagram: $W_4^0 \rightarrow W0,\ W_4^2 \rightarrow W2,$ etc.

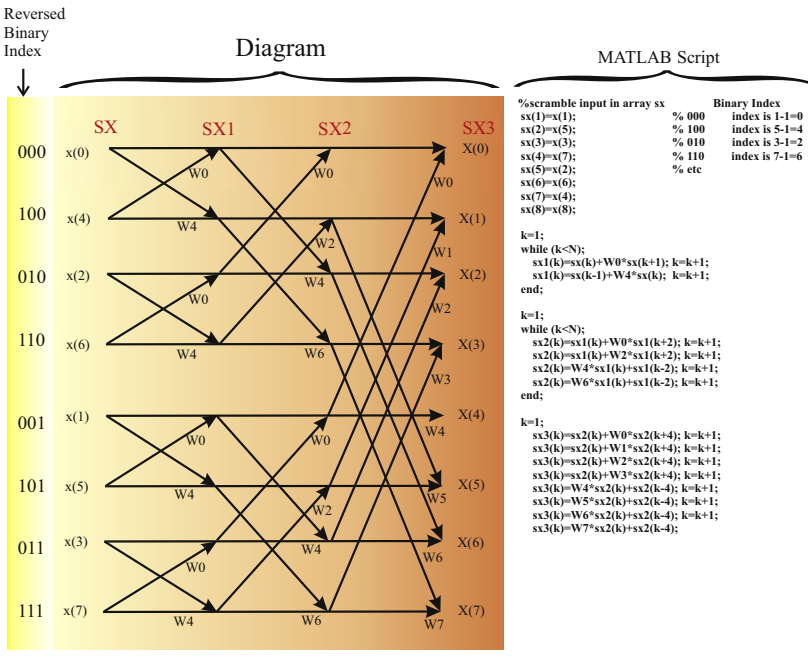


FIGURE 6.7 A diagram of an eight-point fast Fourier transform.

TABLE 6.2 A 3-Bit Binary Set of Numbers to Explain Scrambling in Fast Fourier Transform Input

Binary-Decimal	MATLAB® Index	Inverted Binary-Decimal	MATLAB® Index
000 = 0	1	000 = 0	1
001 = 1	2	100 = 4	5
010 = 2	3	010 = 2	3
011 = 3	4	110 = 6	7
100 = 4	5	001 = 1	2
101 = 5	6	101 = 5	6
110 = 6	7	011 = 3	4
111 = 7	8	111 = 7	8

provides an overview of binary numbers and how they relate to this index scrambling procedure.

In the example in Fig. 6.7, the input time series is $x(0)$, $x(1)$, …, $x(7)$. Note that the input of the algorithm is the lower case vector x, and the output is represented as capital X. First the input is ***scrambled*** to obtain the input to the FFT algorithm: i.e., $SX(0) = x(0)$, $SX(1) = x(4)$, …, $SX(7) = x(7)$. The scrambling process can be presented by reversing the index in binary format. In our time series we have eight data points ($x(0)$–$x(7)$); to represent indices from 0 to 7 we need 3 bits ($2^3 = 8$). We use the reverse binary values to scramble the input, for example, SX with index 1 (in binary 3 bit representation 001) becomes x with index 4 (in binary representation 100). An overview for all indices can be found in Table 6.2. Note that in the MATLAB® script example all indices are increased by one ($SX(0) \rightarrow SX(1)$ etc.) because MATLAB® cannot work with a zero index. After the input is scrambled, the rest of the diagram shows the flow of the calculations. For instance, working backward in the diagram from $X(3)$:

$$X(3) = SX2(3) + W3 \times SX2(7)$$

with:

$$SX2(3) = W6 \times SX1(3) + SX1(1) \text{ and } SX2(7) = W6 \times SX1(7) + SX1(5)$$

with:

$$SX1(1) = SX(0) + W4 \times SX(1) \ldots \text{ etc.}$$

As you can see the creation of the algorithm from the diagram is tedious but not difficult. The associated part of the MATLAB® script in Fig. 6.7 reflects the diagram with all indices increased by one. The purpose

of this example script is to make the FFT operation transparent; therefore this example script is neither optimized for efficiency nor particularly elegant (from a programmer's perspective).

EXERCISES

6.1. Show that when frequency f is used instead of angular frequency ω ($\omega = 2\pi f$), that $F(f) = \int_{-\infty}^{\infty} f(t)e^{-j2\pi ft}dt$ and $f(t) = \int_{-\infty}^{\infty} F(f)e^{j2\pi ft}df$ are equivalent to Eqs. (6.4) and (6.8).

6.2. Given Eqs. (6.4) and (6.8),
   a. Determine the Fourier transform of $\delta(t)$
   b. Determine the inverse transform of $\delta(\omega)$
   c. Interpret your findings.

6.3. Compute the CFT (show all the steps) of
(Hint: see Section 6.2.1. Examples of CFT pairs)
   a. $\exp(j\omega_0 t)$
   b. $\cos(\omega_0 t)$
   c. $\sin(\omega_0 t)$
   d. $\sin(\omega_0 t) + \cos(\omega_0 t)$
   e. $\cos(\omega_0 t + \phi)$
   f. Are the functions even, odd, neither, real, imaginary, complex?
   g. Are their CFTs even, odd, neither, real, imaginary, complex?

6.4. We sample an epoch of T seconds at a rate of s samples/second and we use DFT to compute a spectrum with a resolution of Δf Hz and a range of F Hz.
Complete the "—" in Table E6.4.

TABLE E6.4 DFT Time- and Frequency Domain Parameters

T (s)	s (samples/s)	Δf (Hz)	F (Hz)
10	10,000	—	—
—	—	1	50
—	—	0.5	10
5	—	0.2	—
5	100	—	—
—	1000	—	500
—	10,000	1	—

6.5. Using a sampled time series $x(n)$ with N samples we can determine the DFT $X(k)$ using the equation:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \text{ and } W_N^{kn} = \exp\left(-j\left[\frac{2\pi}{N}\right]kn\right)$$

a. Show that $X(0) = N\overline{x}$ with $\overline{x}$ = the mean of the time series
b. Explain why $X(k)$ is a periodic function
c. Determine the value of the period P for which $X(k) = X(k + P)$.

Reference

Cooley, J.W., Tukey, J.W., 1965. An algorithm for machine calculation of complex Fourier series. Math Comput. 19, 297–301.

CHAPTER

7

1-D and 2-D Fourier Transform Applications

7.1 SPECTRAL ANALYSIS

The raw complex-valued output of a fast Fourier transform (FFT) or discrete Fourier transform (DFT) is difficult to interpret directly. The most common approach is to present the ***power spectrum S*** of the given signal. Here the power for each frequency is plotted along the frequency axis (in Hz or in rad/s). This result is obtained by multiplying the FFT output X (Chapter 6, Eq. (6.18)) with its complex conjugate X^*. Representing the DFT or FFT output for a given frequency ω_k as $a_k + jb_k$ (similar to Eq. (6.15)), the power at ω_k is:

$$(a_k + jb_k)(a_k + jb_k)^* = (a_k + jb_k)(a_k - jb_k) = a_k^2 + b_k^2.$$

We used $j^2 = -1$ and the superscript * indicates the complex conjugate. Repeating this for all frequencies ω_k, the function defined over the whole spectrum S is:

$$S = \frac{XX^*}{N} \tag{7.1}$$

The power spectrum can be normalized by dividing by the number of data points N. This normalization will ensure that the energy (sum of squares) of the time series equals the sum of the elements in the power spectrum (Appendix 7.1).

An example of applying spectral analysis to a short (eight-point) time series is summarized in Table 7.1. The input time series is x consisting of eight ($n = 0$–7) real values; the output of the FFT is X, an array of eight complex values. It can be seen that the sum of squares in the time domain (column xx in Table 7.1) and the sum of the normalized power spectrum

TABLE 7.1 Example of an Eight-Point fft

N	x	xx	$R = \text{Real}(X)$	$I = \text{Imag}(X)$	XX^*	S	AS	ϕ
0	0.5	0.25	4	0	16	2	1	0
1	1.5	2.25	2.4142	−5.8284	39.799	4.9749	1.5772	−1.1781
2	2.5	6.25	−2	2	8	1	0.7071	2.3562
3	1.5	2.25	−0.4142	0.1716	0.201	0.0251	0.1121	2.7489
4	−0.5	0.25	0	0	0	0	0	0
5	−1.5	2.25	−0.4142	−0.1716	0.201	0.0251	0.1121	−2.7489
6	−0.5	0.25	−2	−2	8	1	0.7071	−2.3562
7	0.5	0.25	2.4142	5.8284	39.799	4.9749	1.5772	1.1781
Sum	4	14				14		

Array x is the input and X the output of the algorithm. The real and imaginary parts are indicated in the fourth and fifth columns. The nonnormalized power spectrum in the sixth column shows symmetry around $n = 4$ (not including the first value $n = 0$ representing the DC component). The real part of X is symmetric as well (corresponding to the even property of the cosine wave); whereas the imaginary part of X gains symmetry from the odd property of the sine wave. The sum of squares in the time domain (third column) and the normalized spectrum S (seventh column) is 14 in both cases. The spectrum is commonly depicted over half the number of points; here 0−3. All values in the table were obtained using the MATLAB® fft command.

(S in Table 7.1) both equate to 14. The first value of the power spectrum quantifies the DC component, the sum of the squares of all the values normalized by the number of values: $\frac{\left(\sum_{i=0}^{7} x_i\right)^2}{N} = \frac{(4)^2}{8} = 2$ (Table 7.1). Further note that the power spectrum is symmetric around the mid point ($n = 4$ in Table 7.1). Because of this symmetry it is common practice to show only the values corresponding to positive frequencies. Therefore, to preserve the relationship between power in the time and frequency domains, some authors may normalize this one-sided spectrum by $2/N$.

A related approach to displaying the results of spectral analysis is to show the ***amplitude spectrum AS*** (Table 7.1), the square root of the power spectrum. If one wants the amplitude in the spectrum to correspond with the amplitude of sinusoidal signals in the time domain, one must normalize by $2/N$:

$$AS = \frac{2}{N}\sqrt{XX^*} \tag{7.2}$$

A third commonly used presentation is the ***phase spectrum***. This depicts the phase versus frequency, where phase ϕ is calculated as:

$$\phi = \arctan\frac{I(X)}{R(X)} \tag{7.3}$$

with $I(X)$ and $R(X)$ denoting the imaginary and real parts of X, respectively. Unlike in the power and amplitude spectra no normalization is required for the phase.

The ***X-axis of the spectrum*** is frequency. As mentioned above, since the full spectrum contains redundant information (Table 7.1), typically only up to half of the spectrum is shown. Given a real-valued time series, we can establish that the power and amplitude spectra will be even functions; therefore one half of the function represents all information. For example, if a time series is sampled at a sample rate (SR) 1000 Hz and the FFT is calculated over $NP = 512$ (2^9) points of the time series, the frequency axis range covering half the spectrum ranges from 0 to $SR/(NP-1) \times (NP/2-1)$, in this example:

$$\begin{aligned} 0 &\rightarrow 1000/(512-1) \times (512/2-1)\ \text{Hz} = 499.02 \\ &\approx \text{Nyquist Frequency} = \text{SR}/2 \end{aligned} \tag{7.4}$$

Note that the frequency axis begins at 0 (representing the DC component in the time domain) therefore we subtract one from 512/2. The step size on the frequency axis is $1000/(512-1)$ Hz. Here we subtract one form 512 because the total sampled epoch contains $512-1$ intervals, resulting in a total epoch length of $(512-1) \times (1/1000)$. For a spectral plot against angular frequency, the values in Hz must be by multiplied by 2π for a scale in rad/s. **In the following examples, *NP* >> 1, and therefore we simplify our calculations by omitting to subtract one from *NP*/2 and *NP*.**

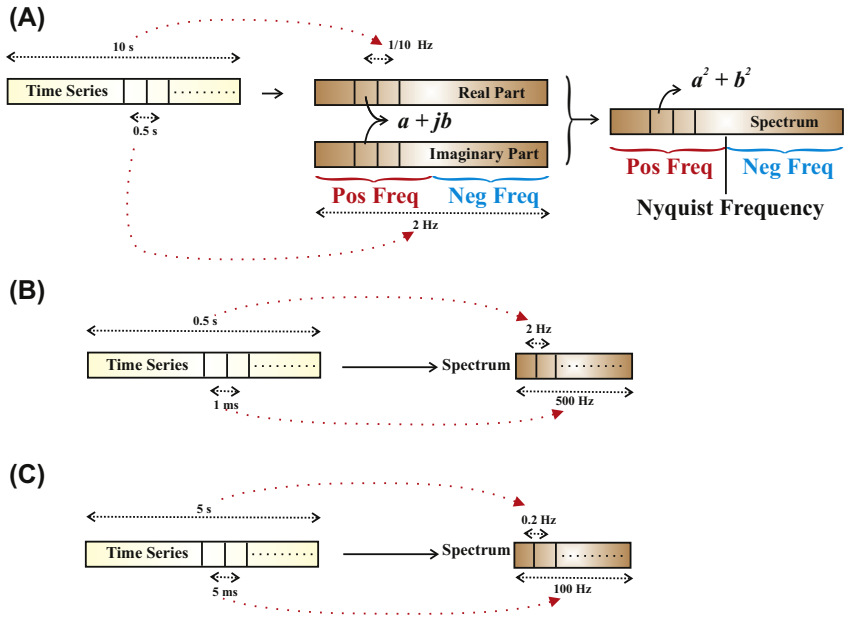


FIGURE 7.1 Overview of the relationships between the epoch length and sample rate of a time series with its precision and range in the associated spectrum. (A) An example of a time series sampled at an interval of ½ s for 10 s. The discrete Fourier transform consists of even (real) and odd (imaginary) parts in which the range (2 Hz) and resolution (1/10 Hz) are directly related to the time series. The spectrum resulting from these real and imaginary coefficients is even. The frequency scale can be represented as a full circle where $0 \rightarrow \pi$ can be considered positive frequencies and $\pi \rightarrow 2\pi$ as negative ones. Because the power spectrum is even, the part reflecting the negative frequencies is identical to the part containing positive frequencies, and therefore it is common practice to depict only the first half of the spectrum (up to the Nyquist frequency). Two more examples with different sample rates and durations are shown in (B) and (C). (B) An example of a 0.5-s epoch sampled at 1 kHz (1 ms sample interval) resulting in $1/0.5 = 2$ Hz precision and $1000/2 = 500$ Hz range. (C) An example of a 5-s epoch sampled at 200 Hz (5 ms sample interval) resulting in $1/5 = 0.2$ Hz precision and $200/2 = 100$ Hz range.

The above may seem overly technical, but the scaling makes perfect sense if we consider the following simple example. The signal's sample rate determines the highest frequency (Nyquist frequency, Chapter 2) that can be represented in the frequency domain. The epoch length determines the precision: e.g., a 500-point epoch sampled at 1000 Hz represents 0.5 s → the spectral resolution is the inverse of 0.5 s → 2 Hz. If we take a 1000-point epoch sampled at 200 Hz, the entire epoch is 5 s long, and the spectral resolution thus becomes 0.2 Hz (Fig. 7.1).

The signals in Fig. 7.2 illustrate the use of spectral analysis to detect periodic elements within a noisy signal. The time domain signal contains noise plus both 50- and 120-Hz sine waves. ***You can use the following MATLAB® script to recreate this example of spectral analysis.***

```
% pr7_1.m
% Spectrum

srate=1000;                          % sample rate
pt=512;                              % points (2^n) for the FFT
range=(pt/2);                        % range for the spectral plot

t=0:1/srate:0.6;                     % time axis
f=srate*(0:range)/pt;                % frequency axis
                                     % starts at 0!
x=sin(2*pi*50*t)+sin(2*pi*120*t);    % SIGNALS 50 and 120 Hz
y=x+randn(1,length(t));              % signal + noise in mV

figure                               % plot signal
plot(t,y)
title('Time Series')
xlabel('time (s)')
ylabel('Amplitude (mV)')

Y=fft(y,pt);                         % do a 512 pt FFT
Pyy=Y.*conj(Y)/pt;                   % Power spectrum

figure                               % Plot result
plot(f,Pyy(1:length(f)));
title('Powerspectrum')
xlabel('Frequency (Hz)')
ylabel('Power (mV2)')
```

A typical output of this script is shown in Fig. 7.2. Please note that your output may be slightly different from the graphs in Fig. 7.2, due to the random number generator randn used to simulate the noise component.

Spectral analysis is often used in electroencephalography (EEG) analysis to evaluate the classical EEG frequency bands (δ, θ, α, β; see Section 1.5.1). For this type of signal the frequency domain characteristics are relevant because of the clinical significance of the various rhythms (Chapter 1, Fig. 1.3). The EEG and its associated spectrum in Fig. 7.3 show a clear presence of the alpha rhythm, one of the most obvious components in the EEG in awake subjects with both eyes closed. The MATLAB® file "AlphaRhythm_5s.mat," containing 5 s of this time domain signal recorded from electrode position O_2 and sampled at a rate of 250 Hz, is included on the website, http://booksite.elsevier.com/9780128104828/.

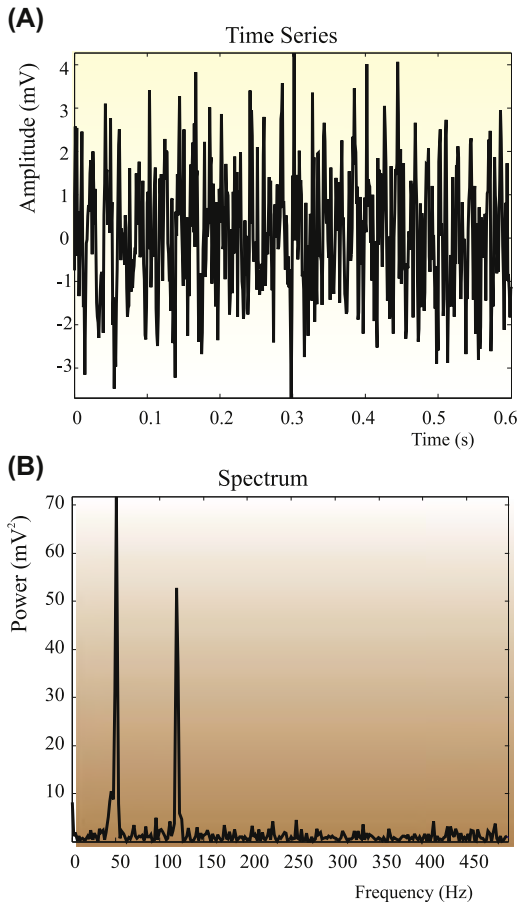


FIGURE 7.2 Output of the pr7_1.m script. (A) Time series. (B) Power spectrum. Note how the two sinusoidal signals at 50 and 120 Hz that are buried in noise (A) become clearly visible in the frequency domain (B).

7.1.1 Application of Data Windows

In the above examples we determined the spectrum from a finite epoch of data. Because we evaluate the signal over a limited time interval consisting of N samples, we are implicitly multiplying the theoretically infinite input signal of the FFT (or DFT) algorithm with a rectangular function (i.e., a rectangular data window). As we will see in Chapter 13, this multiplication in the time domain corresponds with a convolution in the frequency domain. Because the DFT/FFT is determined by default over a limited epoch of a time series, we can analyze the effect of such a limitation using the continuous Fourier transform (CFT). In the following example we consider the transform of a cosine wave truncated by a finite epoch length. The theoretical transform for

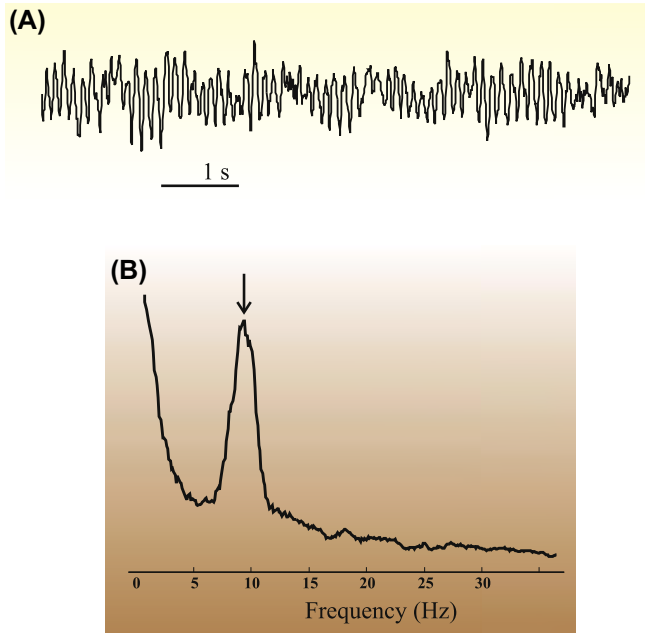


FIGURE 7.3 An example of spectral analysis of an electroencephalography trace recorded from position O_2 shown in (A). The trace includes strong oscillation in the alpha band. Accordingly, the power spectrum in (B) shows the clear presence of a component slightly below 10 Hz (*arrow*) representing this alpha rhythm. For clarity, the spectrum in (B) was smoothed in the frequency domain using a rectangular 1.5-Hz window.

the continuous wave (Fig. 6.2C, Section 6.2.1) is impulse functions at $\pm\omega_0$, i.e., the CFT pair is:

$$\cos(\omega_0 t) \Leftrightarrow \pi[\delta(\omega+\omega_0)+\delta(\omega-\omega_0)] \tag{7.5}$$

Using Eq. (6.4) we then obtain the Fourier transform $W_R(j\omega)$ of a rectangular window $w_R(t)$ function used to define our sampling epoch in time. The rectangular window is equal to one only for the duration of the epoch from $-T/2$ to $T/2$, and zero everywhere else. We use this fact that $w_R(t)=0$ for $|t|>T/2$ to change the integration limits in the Fourier transform integral; therefore the CFT for the window we use as the input for the DFT analysis is:

$$W_R(j\omega)=\int_{-\infty}^{\infty} w(t)e^{-j\omega t}dt=\int_{-T/2}^{T/2} 1e^{-j\omega t}dt=-\frac{1}{j\omega}\left[e^{-j\omega t}\right]_{-T/2}^{T/2}$$

$$=-\frac{1}{j\omega}\left[e^{-j\omega T/2}-e^{j\omega T/2}\right]=-\frac{1}{j\omega}[-2j\sin(\omega T/2)]=\frac{2\sin(\omega T/2)}{\omega}=\frac{\sin(\pi f T)}{\pi f} \tag{7.6}$$

In the last steps in Eq. (7.6) we used Euler's relationship [$e^{jx} = \cos(x) + j\sin(x)$], and $2\pi f$ was substituted for ω.

We can plot the power of this function against frequency for different widths of the window (Fig. 7.4). With increasing width (T), we observe that: (1) the amplitudes of the main peak and its associated ripples increase, and (2) the widths of these features decrease. The example in Fig. 7.4 shows that the spectrum of the window has a ripple at the frequency equal to the inverse of the duration of the window. When analyzing a pure wave at frequency f, this ripple effect results in the "leaking" of energy around the spectral peak f.

The leaking of energy to adjacent frequency bands is due to the fact that the multiplication of the time domain wave with a rectangular window is equivalent to a complex convolution in the frequency domain (Chapter 13). If you are not yet familiar with convolution you can skip this section and come back to it later. Combining Eqs. (7.5) and (7.6) we can evaluate the effect of a rectangular window in the time domain on the spectral analysis of a pure cosine wave. We may use complex convolution to describe the Fourier transform pair:

$$w_R(t) \times \cos(\omega_0 t) \Leftrightarrow \frac{1}{2\pi} \int_{-\infty}^{\infty} [W_R(j\omega - ju)]\pi[\delta(u + \omega_0) + \delta(u - \omega_0)]du \quad (7.7)$$

Using the sifting property of the Dirac delta function to evaluate the integral:

$$w_R(t) \times \cos(\omega_0 t) \Leftrightarrow \int_{-\infty}^{\infty} \left[\frac{\sin((\omega - u)T/2)}{\omega - u}\right][\delta(u + \omega_0) + \delta(u - \omega_0)]du$$

$$= \frac{\sin((\omega + \omega_0)T/2)}{\omega + \omega_0} + \frac{\sin((\omega - \omega_0)T/2)}{\omega - \omega_0} \quad (7.8)$$

Interpreting Eq. (7.8) we conclude that the spectrum of a truncated cosine wave with a frequency ω_0 produces a broadened peak surrounded by ripples (identical to the function in Fig. 7.4) at $\pm\omega_0$ in the frequency domain. This process of multiplication in the time domain and convolution in the frequency domain for the truncated cosine is summarized in Fig. 7.5.

More deliberately crafted time domain windowing functions (to replace the implicit rectangular window) are commonly applied to avoid the ugly ripple effects in the spectra. This reduction in ripples comes at a cost. For each sine/cosine wave, the window attenuates the amplitude of the spectral peak and increases its width. Several commonly applied data windows are summarized in Table 7.2. In MATLAB® these windows are

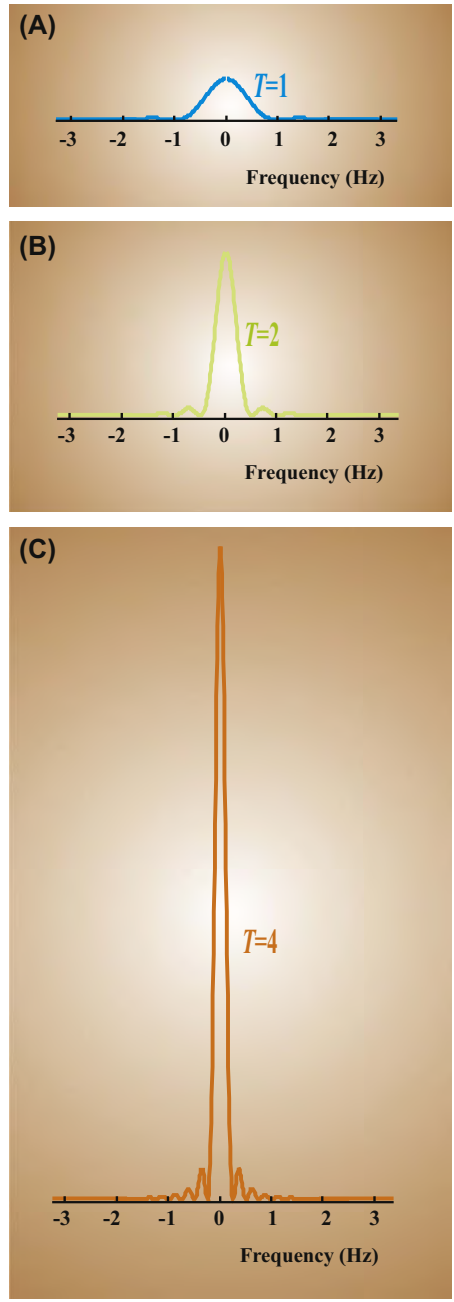


FIGURE 7.4 The **power spectrum** of rectangular windows of duration $T = 1$, 2, and 4. It can be seen that these spectra have ripples in the frequency domain that correspond to the inverse of their window duration, i.e., for $T = 4$ the ripples are at ¼ Hz, for $T = 2$ the ripples are at ½ Hz, and for $T = 1$ the ripples are at 1 Hz. This ripple effect causes the discrete spectrum of a pure wave such as $\cos(2\pi ft)$ or $\sin(2\pi ft)$ to show energy adjacent to the main peak at frequency f. The vertical calibration is in arbitrary units but is identical for all panels.

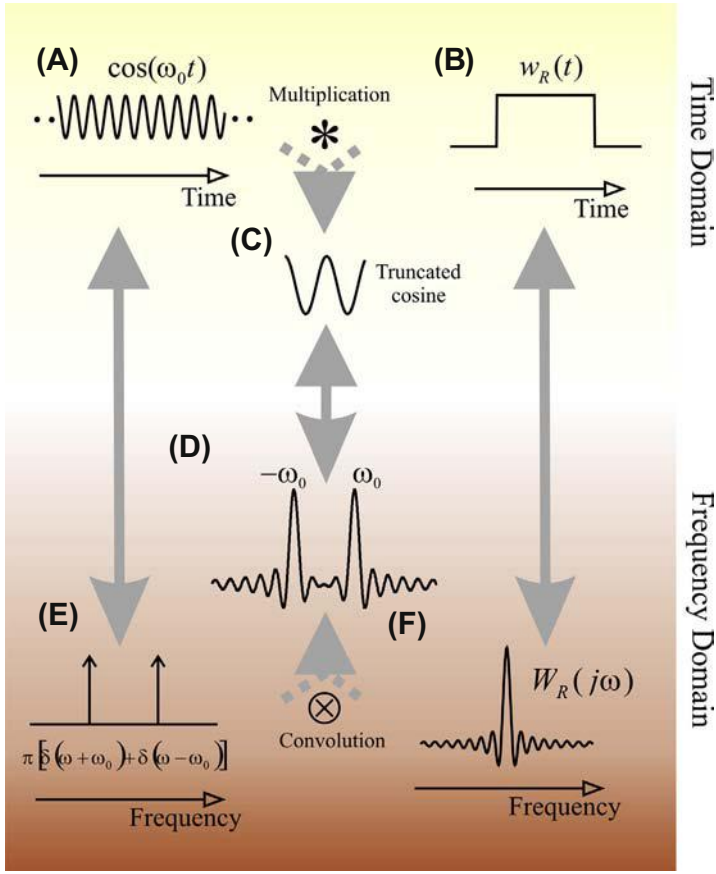


FIGURE 7.5 Overview of the Fourier transform of a truncated cosine wave. A theoretically infinite cosine wave (A) multiplied by a rectangular window (B) generates a truncated wave (C). The Fourier transform of the cosine and the window in the frequency domain are shown in (E) and (F), respectively. The transform of the truncated cosine is the convolution of its components, shown in (D).

included in the Signal Processing Toolbox. Examples of the bartlett, boxcar (rectangular), and hanning windows are depicted in Fig. 7.6.

Note: A window that transforms into a flat line in the frequency domain seems to be the simple solution to avoiding all the undesired effects we discussed above. This would result in a complex convolution result that does not distort the spectrum of the signal at hand. Unfortunately, such a flat line in the frequency domain does not transform into a window in the time domain.

TABLE 7.2 Overview of Commonly Used Data Window Functions

Data Window	Equation Window $w(t)$ for Epoch Size $-T \rightarrow T$
Bartlett (triangular, Fejér)	$w(t) = \begin{cases} 1 - \dfrac{\lvert t \rvert}{T} & \lvert t \rvert \le T \\ \text{else } 0 \end{cases}$
Hamming	$w(t) = \begin{cases} 0.54 + 0.46 \cos\left[\dfrac{\pi t}{T}\right] & \lvert t \rvert \le T \\ \text{else } 0 \end{cases}$
Hann (von Hann, hanning)	$w(t) = \begin{cases} 0.5 + 0.5 \cos\left[\dfrac{\pi t}{T}\right] & \lvert t \rvert \le T \\ \text{else } 0 \end{cases}$
Rectangular	$w(t) = \begin{cases} 1 & \lvert t \rvert \le T \\ \text{else } 0 \end{cases}$

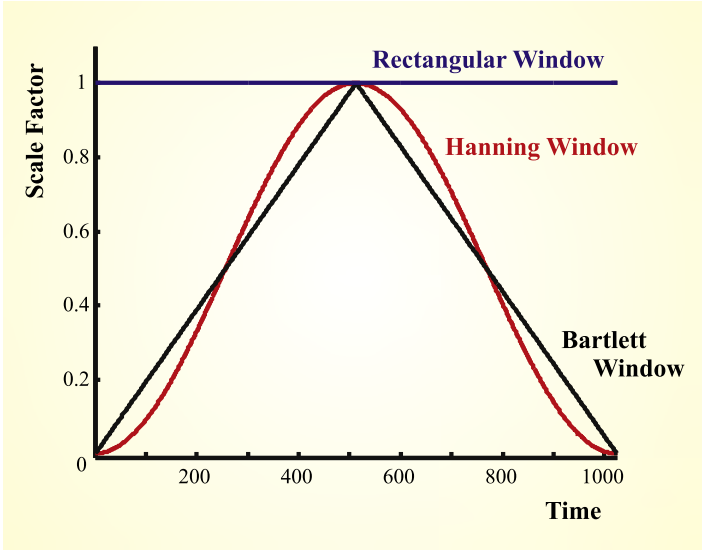


FIGURE 7.6 Examples of three window functions of 1024 points.

7.1.2 Spectral Analysis of Physiological Signals

Spectral analysis of signals composed of pure sine waves is theoretically straightforward. In physiological signals interpretation of spectra requires caution because these time series are rarely stationary and

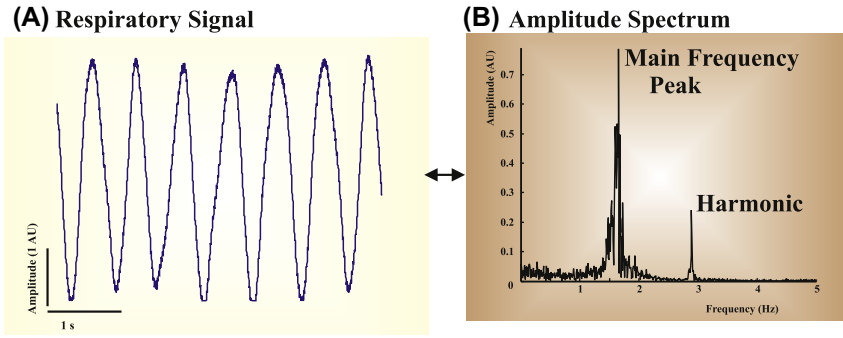


FIGURE 7.7 Frequency analysis of a respiratory signal from a human neonate. An epoch of the time domain signal is shown in (A) and the amplitude spectrum in (B). Clearly the main peak ~1.5 Hz shows the respiratory frequency, whereas the peak close to 3 Hz is a harmonic due to the imperfect sinusoidal signal. The respiration signal, sampled at 1 kHz, is available on http://booksite.elsevier.com/9780128104828/ (respiration.mat).

usually contain both nonperiodic and periodic components. Even when the DC component is removed, the spectra from physiological data may contain low-frequency components due to slow nonperiodic activity (e.g., trends) or periodic activity with a periodicity beyond the analysis window. Similarly, the high-frequency components may be contaminated by high-frequency nonperiodic processes (e.g., sudden events). Furthermore, the periodic activity in physiological signals is usually far from purely sinusoidal, leading to spectral components (so-called harmonics) at higher frequencies.

The take-home message from the above discussion is that not all peaks in a spectrum of physiological data directly correspond to actual physiological, periodic processes in the system at hand. Careful evaluation must be used to distinguish between real spectral peaks and irrelevant by-products. A somewhat trivial example showing the effect of a not exactly sinusoidal signal is the respiratory signal depicted in Fig. 7.7. Although the actual respiratory rhythm cycles at around 1.5 Hz, the spectrum shows a harmonic at ~3 Hz.

7.2 TWO-DIMENSIONAL FOURIER TRANSFORM APPLICATIONS IN IMAGING

The first aspect of application of Fourier transformation to imaging is that an image has two dimensions, unlike the one dimension of a single recording of neural activity. Therefore we need to extend the 1-D Fourier transform in Eq. (6.4) to a 2-D version. This can be done as follows. Let's start with an image $I(x,y)$ and create the Fourier transform with respect to x and associated spatial frequency u. Then we take that result and perform

a Fourier transform with respect to y and its associated spatial frequency v. These two steps give:

$$\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} I(x,y)e^{-j2\pi ux}dx\right]e^{-j2\pi vy}dy.$$

By combining the two integrals we get $F\{I(x,y)\} = \Im(u,v)$, the 2-D Fourier transform of $I(x,y)$:

$$F\{I(x,y)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)e^{-j(2\pi ux+2\pi vy)}\,dxdy. \tag{7.9}$$

By performing this transform, we obtain the 2-D spatial frequency domain that can be used for a variety of image processing applications. In the following, we will discuss several examples in optics and image reconstruction in medical imaging. Using a similar approach we can find the expression for the 2-D inverse Fourier transform. Starting from the 1-D inverse Fourier transform (Eq. (6.8)) with ω replaced by $2\pi f$, and using variables x, y and u, v as above, we get:

$$I(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Im(u,v)e^{j(2\pi ux+2\pi vy)}dudv \tag{7.10}$$

The MATLAB® commands to perform the discrete version 2-D Fourier transform and its inverse are fft2 and ifft2.

7.2.1 Fourier Transform by Lenses

In addition to the ray optics that most of us are familiar with, there is a significant part of optics known as Fourier optics. This part of optics is based on the Fourier transformation properties of optical devices such as a slit or a lens. The goal here is to present examples of the application of the 2-D Fourier transform. Therefore we won't go into the details governing the physics of Fourier optics here; such details can be found in Goodman (2004), or an introduction in Hecht (2016).

An intuitive example of the Fourier transform property of a lens is depicted in Fig. 7.8. Here it can be seen that a spatially uniform wavefront, i.e., a constant input, is transformed into a dot at the focal distance (f, Fig. 7.8A), and a point light source placed at the focal distance of the lens is transformed into a uniform wavefront (Fig. 7.8B). Now recall from the examples in Section 6.2.1 and Eq. (6.11) that the Fourier transform of a constant (or a DC electrical signal) is a weighted delta function δ, and vice versa. If we ignore the details of the weighting, this is exactly what the lens is showing in the examples in Fig. 7.8A and B.

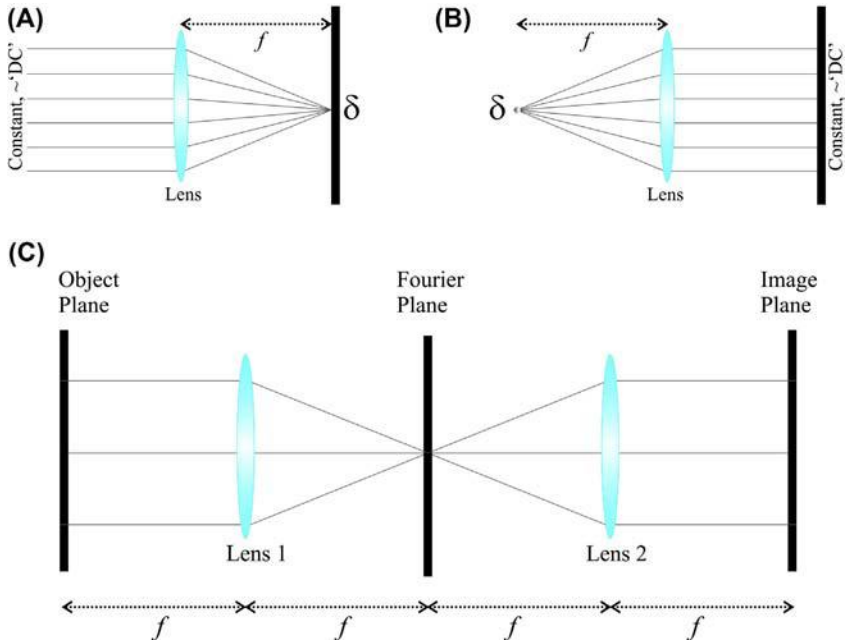


FIGURE 7.8 A lens as a Fourier transforming device. (A) A constant illumination is transformed into a single dot at the focal distance of the lens and (B) a point light source is transformed into a uniform, constant wavefront. (C) An example of a so-called 4*f* system by which an object is imaged via a pair of lenses, and where optical processing in the spatial frequency domain can be accomplished in the Fourier plane of this lens system. Note that the focal distances of the two lenses in the 4*f* system are the same in this example, but they don't have to be identical.

The Fourier transforming property of the lens can be employed to image objects and to optically process images. One well known and widely applied example is the so-called 4*f* system (Fig. 7.8C). In this example we have an object placed at one focal distance of the objective lens that creates a 2-D Fourier transform of the object in the Fourier plane at its focal distance. The Fourier plane is also placed at one focal distance to a second lens, the collector lens that performs the second Fourier transform. Due to the **duality property** of the Fourier transform and its inverse transform (Section 6.2.1), the collector lens creates an inverted image of the object in the image plane, again at its focal distance. The image is inverted because of the different signs in the exponentials of the Fourier transform and its inverse (see Eqs. (6.4) and (6.8)). To show this, let's define the coordinates of the object plane as x,y and the ones of the image plane as X,Y. If we now substitute $X = -x$ and $Y = -y$ (i.e., inverted coordinates) in the expression for the 2-D Fourier transform (Eq. 7.9), we

indeed obtain the correct expression for the inverse Fourier transform of the inverted object:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(-X,-Y)e^{j(2\pi uX+2\pi vY)}dXdY.$$

Since we deal with four focal distances in this procedure, this system is known as the 4*f* system. One important property of this setup is that we now can process the image in the frequency domain by placing masks in the Fourier plane. If we, for example, place a pinhole mask in the Fourier plane, we remove high spatial frequencies from the image that is produced, i.e., we attenuate edges and keep contrast information of the image. If, on the other hand, we place a mask blocking the center part in the Fourier plane, we remove the low spatial frequencies and keep the edge information; we now have an edge detector. An example of the latter procedure, albeit obtained via a digital filter procedure and not via direct optical processing, is depicted in Fig. 18.4.

Note that with respect to the duality property (Chapter 6), the discrete version of the Fourier transform (i.e., both the DFT and the FFT algorithms) behaves slightly different as compared to the CFT due to the scaling by $1/N$ in the discrete inverse transform case (compare Eqs. (7.9) and (7.10) versus (6.18) and (6.21)).

7.2.2 Tomography

In this section we apply the Fourier transform to a problem in tomography, used in medical imaging. We will approach tomography in a general fashion; the principles we discuss apply both to scanning emission and absorption profiles. Consider emission of activity and passive absorption as the same type of process. In the case of emission, each little voxel (or pixel in the 2-D case) emits its own contribution to the total that is measured outside the volume. The absorption model is slightly more complicated since each pixel instead contributes to attenuation across the area. We can use Beer's law to express the intensity of the output I_o of a beam as a function of input intensity I_i and attenuation caused by absorption a_k in N elements: $I_o = I_i e^{-\sum_{k=1}^{N} a_k}$. Using the property that the absorption law is exponential, we can use the logarithm of the absorption ratio A to obtain an additive effect for each element k, i.e.:

$$A = \ln\left[\frac{I_o}{I_i}\right] = \sum_{k=1}^{N} a_k$$

7.2.2.1 *Measured Absorption—Radon Transform*

In the following, we develop the ***Radon transform***, the ***Fourier slice theorem***, and ***filtered backprojection*** as each applies to CT image reconstruction. These techniques require reconstruction of a density function representing the internal structure of an object from sensor readings taken from outside that object. This is typically accomplished by calculating a series of 2-D density functions (or slices) through the object on a set of planes and reconstructing the 3-D image from those images. The task is to calculate the 2-D density function with readings from a sensor which typically rotates around the object in a given plane. The following derivations use Fourier analysis to relate a filtered version of this measured signal to the density function of the object within the measured region.

For ease of explanation we use polar coordinates to derive the theorem. Assume a source and detector moving at an angle θ with respect to the X-axis around an object enclosed in a circle with radius R (Fig. 7.9). The distance of the source–detector (SD) line from the origin is t, and the detector measures absorption (or emission) of all the points along the line. In polar coordinates, all points r,ϕ on the SD line relate to t as:

$$t = r\cos(\phi - \theta) \tag{7.11}$$

If the emitter/detector pair is moving at a constant speed, t represents time and the measurement at the detector becomes a time series.

Note that the value of θ varies between 0 and 180 degrees since measurements by the detector along a line at α degrees are the same as

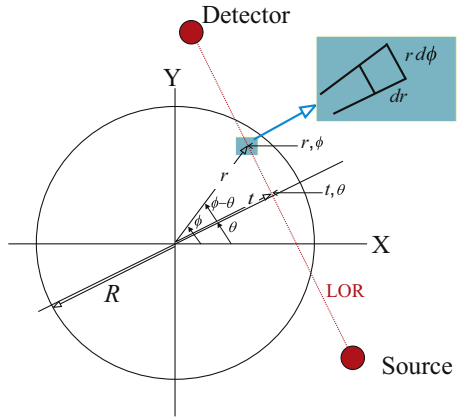


FIGURE 7.9 Diagram of a CT scan procedure with a source–detector setup scanning an object within a circle with radius *R*. *LOR*, line of response.

those made at $180 + \alpha$ degrees. The total absorption along SD is represented by $m(t,\theta)$ and is determined by the contributions of arbitrarily small surfaces $r\ dr\ d\phi$ (Fig. 7.9, inset). Denoting the absorption function inside the circle as $a(r,\phi)$, which corresponds to the mass to be scanned, we obtain a function $m(t,\theta)$ that represents the so-called Radon transform of $a(r,\phi)$:

$$m(t,\theta) = \iint_R a(r,\phi)\delta[t - r\cos(\phi - \theta)]rdrd\phi \tag{7.12}$$

Think of $\iint_R a(r,\phi)rdrd\phi$ as the total absorption of the whole object inside the circle with radius R. For a particular measurement $m(t,\theta)$ we are only interested in the contributions along the line of response (LOR, Fig. 7.8). We pull these out by adding a δ function that sifts for the values for ϕ and r on the LOR at a given t and θ. The delta function that accomplishes this must evaluate to zero within the LOR, i.e., the argument should be $t - r\cos(\phi - \theta) = 0$ and $\delta[t - r\cos(\phi - \theta)]$ in Eq. (7.12) accomplishes sifting the points on the LOR (see Chapter 2 for the sifting property). Using integration limits reflecting area R instead of $-\infty \rightarrow \infty$ is appropriate because: $a(r,\varphi) = 0$ for $r > R$.

The 1-D CFT of $m(t,\theta)$ in the spatial domain is:

$$M(z,\theta) = \int_{-\infty}^{\infty} m(t,\theta)e^{-j2\pi zt}dt \tag{7.13}$$

where z represents the spatial frequency domain. Substituting Eq. (7.12) in Eq. (7.13) and combining all terms related to t within the square brackets gives:

$$M(z,\theta) = \iint_R a(r,\phi)\left[\int_{-\infty}^{\infty} \delta[t - r\cos(\phi - \theta)]e^{-j2\pi zt}dt\right]rdrd\phi$$

When using the sifting property of the δ function for the integration over t, this expression evaluates to:

$$\boxed{M(z,\theta) = \iint_R a(r,\phi)e^{-j2\pi zr\cos(\phi-\theta)}rdrd\phi} \tag{7.14}$$

In the following section we will show that this expression is identical to the 2-D Fourier transform of the absorption function a.

7.2.2.2 The Absorption Function in the Spatial Frequency Domain

According to Eq. (7.9), the 2-D Fourier transform of the absorption function $a(x,y)$ in Cartesian coordinates is:

$$A(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} a(x,y) e^{-j2\pi(xu+yv)} dxdy \tag{7.15}$$

Changing u and v into polar coordinates z and θ gives:

$$A(z\cos\theta, z\sin\theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} a(x,y) e^{-j2\pi z(x\cos\theta + y\sin\theta)} dxdy \tag{7.16}$$

Similarly transforming x and y to polar coordinates r and ϕ gives:

$$A(z,\theta) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} a(r\cos\phi, r\sin\phi) e^{-j2\pi zr(\cos\phi\cos\theta + \sin\phi\sin\theta)} rdrd\phi \tag{7.17}$$

Using the trigonometric identity $\cos\phi\cos\theta + \sin\phi\sin\theta = \cos(\phi - \theta)$ and setting $a(r,\phi) = 0$ for all points outside the circle with radius R, Eq. (7.17) becomes:

$$\boxed{A(z,\theta) = \iint_R a(r,\phi) e^{-j2\pi zr\cos(\phi-\theta)} rdrd\phi} \tag{7.18}$$

Thus the 2-D Fourier transform of the absorption function a evaluates to the same expression as the 1-D transform of the measured Radon transform m (Eqs. 7.14 and 7.18), i.e.:

$$\boxed{A(z,\theta) = M(z,\theta)} \tag{7.19}$$

Eq. (7.19) is known as the ***Fourier slice theorem***.

7.2.2.3 The Inverse Transform

Using the same procedure we used in the previous section, we can relate the inverse transform pair expressed in Cartesian coordinates $a(x,y) \Leftrightarrow A(u,v)$ in polar coordinates: $a(r,\phi) \Leftrightarrow A(z,\theta)$. Using this procedure, the inverse transform of Eq. (7.18) returns $A(z,\theta)$ to the spatial domain:

$$a(r,\phi) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} A(z,\theta) e^{j2\pi zr\cos(\phi-\theta)} zdzd\theta \tag{7.20}$$

Using Eq. (7.19) and defining $G(z,\theta) = zM(z,\theta)$ Eq. (7.20) becomes:

$$a(r,\phi) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} G(z,\theta) e^{j2\pi zr\cos(\phi-\theta)} dz d\theta \tag{7.21}$$

The seemingly arbitrary multiplication of $M(z,\theta)$ with z in the frequency domain equates to convolution of $m(t,\theta)$ with a high-pass filter characteristic in the spatial domain (Chapter 17). The inverse transform of $G(z,\theta)$ is $g(t,\theta)$ can therefore be considered a high-pass filtered (differentiated) version of $m(t,\theta)$.

If we focus on the integration to dz in the above expression, the part $\int_{-\infty}^{\infty} G(z) e^{j2\pi zr\cos(\phi-\theta)} dz$ can be written as:

$$\int_{-\infty}^{\infty} G(z,\theta) e^{j2\pi zw} dz \quad \text{with} \quad w = r\cos(\phi-\theta) \tag{7.22}$$

Recognizing this as the inverse Fourier transform of $g(w,\theta)$, and changing the integration limits for θ to $0 \rightarrow 180$ degrees (or $0 \rightarrow \pi$ radian), Eq. (7.21) evaluates to:

$$\boxed{a(r,\phi) = \int_0^{\pi} g(w,\theta) d\theta = \int_0^{\pi} g(r\cos(\phi-\theta),\theta) d\theta} \tag{7.23}$$

Because the function $g(\ldots)$ is a filtered/differentiated version of $m(\ldots)$, Eq. (7.23) is the ***filtered backprojection equation***.

7.2.2.4 *Backprojection in Cartesian Coordinates*

For ease of use, we can transform Eq. (7.23) from polar to Cartesian coordinates. We use:

$$\cos\phi\cos\theta + \sin\phi\sin\theta = \cos(\phi-\theta)$$

Now Eq. (7.23) can be written as:

$$a(r,\phi) = \int_0^{\pi} g[r\cos(\phi)\cos(\theta) + r\sin(\phi)\sin(\theta),\theta] d\theta \tag{7.24}$$

With: $r = \sqrt{x^2+y^2}$, $x = r\cos(\phi)$, $y = r\sin(\phi)$ Eq. (7.24) becomes:

$$a(x,y) = \int_0^{\pi} g[x\cos\theta + y\sin\theta,\theta] d\theta \tag{7.25}$$

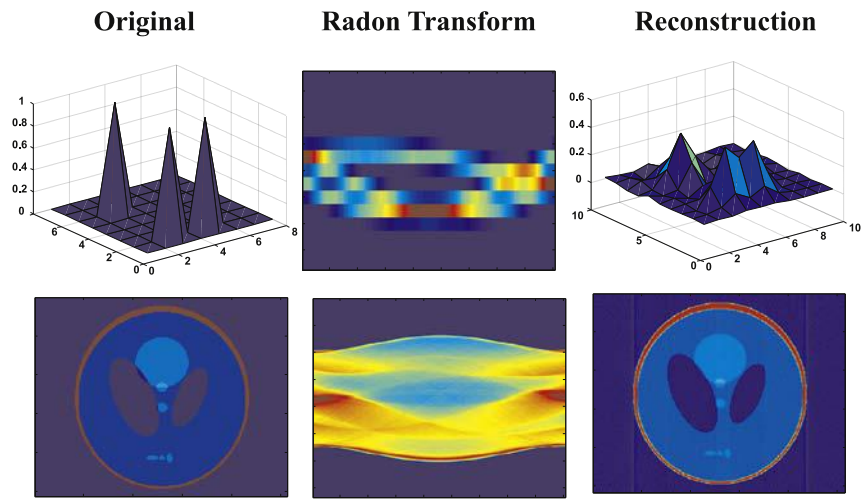


FIGURE 7.10 Examples of reconstruction of an image. The upper row represents the transform and reconstruction of an image (shown as a surface) composed of three pixels set to one in a field of zeros. The lower row is a similar example using the modified Shepp-Logan phantom available in MATLAB®'s image processing toolkit.

For a given θ, the original measurement of $m(\ldots)$ and its filtered version $g(\ldots)$ are ordered according to the variable t. Next, we relate t to x, y, and θ: $t = x\cos\theta + y\sin\theta$, i.e., the standard procedure to recalculate the new x-coordinate after a counterclockwise axis rotation of θ degrees.

The image processing toolbox in MATLAB® contains commands for the Radon transform and its inverse. The script pr7_2 uses these commands for the Shepp-Logan phantom **(Fig. 7.10)**.

```
% pr7_2
% im_phantom: example of an imaging app

clear;
P=phantom('Modified Shepp-Logan');
figure;                          % Depict P
imagesc(P);

% Create Projections
theta=0:1:180;                   % steps of theta
R=radon(P,theta);                % radon transform
figure; imagesc(R);              % show radon transform

figure;
```

```
for step=1:20;                  % loop to examine the resolution effect
   theta=0:step:180;
   Rtemp=R(:,theta+1);
   p=iradon(Rtemp,theta);       % perform filtered backprojection
                                        % NOTE: the iradon transform
                                % includes high pass filtering

   imagesc(p);                  % show result at each resolution 'step'
   drawnow;
   pause;                       % pause before proceeding to the next
end;
```

7.2.3 Magnetic Resonance Imaging

The magnetic resonance imaging (MRI) approach allows us to image anatomic structures inside the body, somewhat similar to the procedure we discussed for CT. Although, filtered backpropagation as described in the previous section, can in principle also be applied to nuclear magnetic resonance (NMR) signals, the procedures for image reconstruction in MRI are usually slightly different as compared to CT. In the following, the goal is to demonstrate and clarify the algorithms that are used to reconstruct an image from signals that originate from magnetic resonance of hydrogen nuclei in the tissue. A significant amount of quantum mechanics is required to describe the NMR phenomena underlying the MRI signal. Since this is nontrivial, many simplified explanations are available in textbooks and online versions. These explanations are sometimes simplified to a degree that they are even incorrect and misleading. Since an in depth description of quantum mechanics is outside the scope of this text on the principles of image creation in MRI, I will use a simplified classical physics representation of the NMR phenomenon. This representation avoids unnecessary detail, but it comes with the explicit warning to avoid using this approach beyond its purpose. Further detail and background can be found in Bushong (1996) and Weishaupt et al. (2006). For those interested in the details of the NMR physics, including the phenomena at the level of quantum mechanics, see Haacke et al. (1999).

7.2.3.1 Nuclear Magnetic Resonance

The spin of the nucleus of the hydrogen atom creates a weak magnetic dipole, and when placed in a strong magnet (B_0, Fig. 7.11A), a very small majority of the dipoles aligns in the direction of the field, creating a net magnetization M_0 in the z-direction, the direction of B_0 (Fig. 7.11B1). In

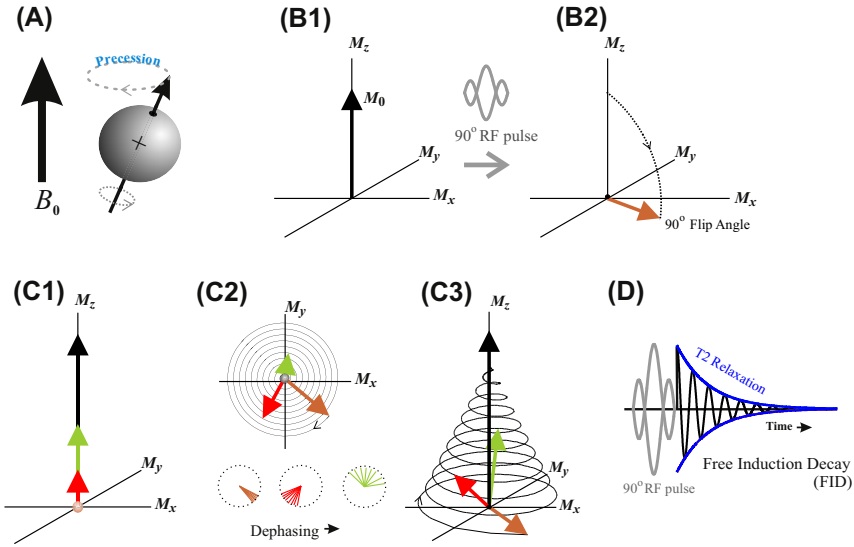


FIGURE 7.11 Simplified representation of the physics of the nuclear magnetic resonance phenomena underpinning magnetic resonance imaging. Proton spin is associated with a magnetic dipole. In an external magnetic field B_0, the magnetic dipole precesses at the so-called Larmor frequency, similar to a gyroscope or spinning toy (A). In the tissue, the magnetic dipoles create a net magnetization M_0 in the z-direction (B1). A radio frequency (RF) pulse at the Larmor frequency can excite the system and flip this net magnetization at an angle, in this example at 90 degrees, resulting in a measurable magnetization in the xy-plane (B2). After the RF offset, two independent types of relaxation occur. So-called T1 relaxation restores the magnetization in the z-direction (C1) and due to T2 relaxation, the xy-component decreases (C2, top diagram). The T2 relaxation is caused by a dephasing process, represented in the three bottom diagrams in panel (C2). A diagram of the combined relaxation processes is depicted in panel (C3). The signals in the z-direction cannot be measured because of the huge external magnetic field (B_0), but the xy-component can be recorded. The signal associated with this measurement is called the free induction decay, and an example is depicted in panel (D).

addition to the dipole orientation, there is a movement called ***precession***, similar to the wobble of a gyroscope or spinning top. This precession process occurs at the so-called ***Larmor frequency*** f_L, proportional to the field strength B_0 (Fig. 7.11A), that is:

$$f_L = \gamma B_0, \quad \text{with} \quad \gamma = 42.6\ \text{MHz/T}. \tag{7.26}$$

Using this relationship, we can determine that a field of 1.5 T (Tesla) creates a precession frequency of 63.9 MHz.

The small excess of dipoles that align with the external field (1/100,000 in a 3-T field at room temperature) is in a high-energy state, while the remainder is in a low-energy state. If a radio frequency pulse (RF pulse) at the Larmor frequency is emitted, we evoke a resonance phenomenon and

some protons in the low-energy state change their orientation relative to B_0, thereby going into the high-energy state. In addition, precession now occurs in phase. At the right amount of energy in the RF pulse, there is a net change of the magnetization angle from the M_z direction. In the example in Fig. 7.11B2 the angle changes to a direction in the xy-plane, indicated as a flip angle of 90 degrees. A magnetic component in the xy-plane is critical for the MRI application because magnetic components in the z-direction cannot be measured since they drown in the huge B_0 field while components perpendicular to the z-direction can be recorded.

The RF pulse excites the system and after termination of this pulse, the system will show relaxation: i.e., the situation as in Fig. 7.11B2 will return to a situation as depicted in Fig. 7.11B1. Two independent processes occur during relaxation: (1) the vertical component will gradually increase because vectors reorient from the high-energy state back into the low-energy one (Fig. 7.11C1), and (2) the horizontal component will disappear due to spin−spin interactions causing a dephasing of the components of the xy-magnetization (Fig. 7.11C2). The two processes are indicated as T1 and T2 relaxation, respectively. The T2 relaxation is faster than T1 relaxation. In practice, due to local inhomogeneity in the magnetic field, the dephasing process happens faster than the rate dictated by the T2 value; the combined dephasing due to both the T2 and the inhomogeneity is referred to as T2*.

7.2.3.2 From Nuclear Magnetic Resonance to Magnetic Resonance Imaging by Encoding the Spatial Coordinates

A critical property that allows us to image different types of tissue is that the relaxation rates are tissue-dependent. In images of the brain, this allows one to distinguish between the cerebrospinal fluid, gray and white matter. However, if the procedure of nuclear resonance is applied to a human, a signal can obviously be recorded, but it is not possible to determine where in the body the signal comes from! In order to accomplish that, the MRI device creates magnetic gradients and since the Larmor frequency depends on the magnetic field strength, these magnetic gradients determine the precession frequency in a location-specific manner.

To accomplish the spatial encoding of the signal, the imaging procedure consists of a sequence of events. First the system applies a magnetic field gradient in the z-direction. This means that according to Eq. (7.26), the precession frequency depends on and increases with z. Let's assume we want to image a slice of the head as indicated in Fig. 7.12A. Now, by adjusting the frequency of the RF pulse to the Larmor frequency in the desired slice, we can excite exactly that part of the head: i.e., the excitation and relaxation processes shown in Fig. 7.11B and C will only occur in that slice! So now we know at least the z coordinate where our signal is

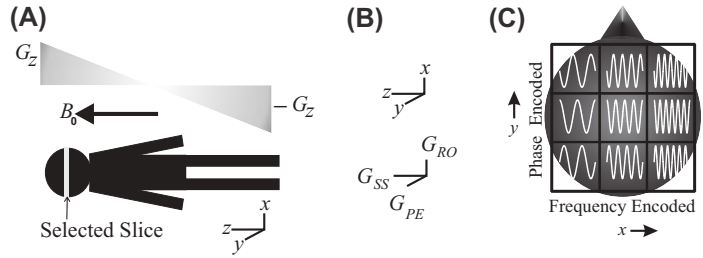


FIGURE 7.12 Gradients in the magnetic field strength create location-specific precession speeds. (A) A gradient in the z-direction determines the slice of tissue that will become excited at emission of an radio frequency signal. (B) Each direction is encoded using a gradient: G_{SS}, slice select; G_{PE}, phase encode; G_{RO}, frequency encode. (C) A very coarse and exaggerated diagram that shows how gradients associated with the x and y directions, within the selected slice, affect the frequency and phase of the signal in a location-specific manner: frequency increases from left to right, and phase from bottom to top.

generated, but within that slice we don't know yet the x, y coordinates of signal contributions. To resolve this, the system employs two additional gradients. First it briefly turns on a magnetic gradient in the y direction; this causes the protons in the anterior part of the head to precess faster than in the posterior part, and the faster ones get ahead in phase relative to the slower ones. After turning off the posterior–anterior gradient, the Larmor frequencies across the slice become uniform again, **but** now there is a phase difference that remains. Next, a left–right magnetic gradient in the x direction is turned on, and this causes the right side to precess faster than the left side. During the application of the left–right field, we can measure the remainder of the relaxation signal shown in Fig. 7.11D. To summarize, the result of this whole sequence is that: (1) by turning on a gradient for slice selection (G_{SS}, Fig. 7.12B) during the RF pulse, we only excite a slice of tissue around a specified value of z; (2) by using a brief epoch in which a phase encoding gradient is turned on (G_{PE}, Fig. 7.12B), signals are encoded by a phase difference in the posterior–anterior (y) direction; and (3) due to a frequency encoding gradient (G_{RO}, Fig. 7.12B), signals are encoded by a frequency difference in the left–right (x) direction.

The result of spatially encoding the signals in the selected slice is depicted in the example diagram in Fig. 7.12C; for clarity we show a very coarse resolution of 3×3 voxels and exaggerated differences between the signals. A diagram summarizing the sequence of events we described above is shown in Fig. 7.13. Finally, it should be noted that the directions of slice selection, phase, and frequency encoding are specific for this example, in practice they depend on several considerations, including the orientation of the desired image.

Thus now we know, in principle, what distinguishes the contributions from different voxels within the selected slice. Since the encoding of voxel

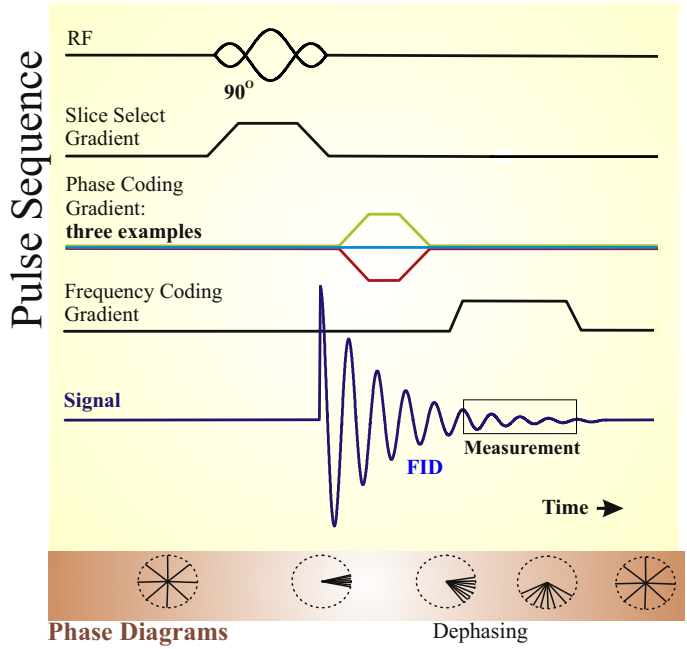


FIGURE 7.13 Diagram of a pulse sequence to perform slice selection, plus phase and frequency encoding within the selected slice. This sequence is repeated with different phase encoding gradients; here only three are shown. The bottom row shows the phase diagrams, similar to the diagrams in Fig. 7.11C2. Further explanation is provided in the text.

contributions occurs by both frequency and phase, it's probably not a surprise that we will use frequency analysis to tease the local signal contributions apart. This procedure of image reconstruction will be explained in more detail below in Section 7.2.3.4.

7.2.3.3 The Spin Echo Sequence

One problem associated with measuring a free induction decay (FID) is that, due to the T2* relaxation, its amplitude rapidly decreases. To mitigate this, a second RF pulse is often employed to rotate the rapidly dephasing proton spins by 180 degrees. The pulse sequence for this protocol, shown in Fig. 7.14, creates an echo with a much better signal-to-noise ratio than the signal we can measure from the FID. This sequence is known as the spin echo sequence.

The effect of rotating the vectors by 180 degrees can be explained by the following simplified analogy described in Bushong (1996). Compare the vectors that start to dephase at the 90 degrees RF offset with runners that take off at a start signal, some are running fast and some slow. After a while, at the 180 degrees RF signal, the vectors/runners turn around and

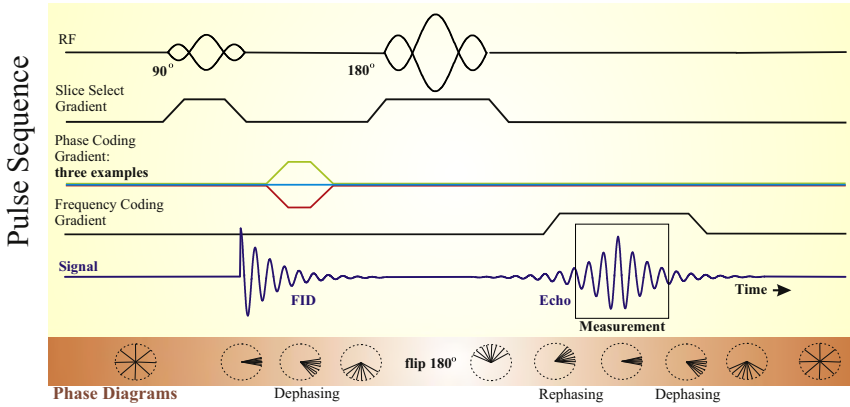


FIGURE 7.14 The spin echo sequence. A 90 degrees radio frequency (RF) signal has the usual free induction decay effect. When the initial pulse is followed by a 180 degrees RF pulse, an echo will occur. Further explanation is provided in the text.

if they keep running at the same pace, they will reach the start line simultaneously (rephasing, Fig. 7.14), and after passing the start line they will begin to dephase again. The phase diagrams in Fig. 7.14 represent this process of de- and rephasing as a result of the RF pulses. The clear advantage is that the echo can be measured at a favorable moment when the signal-to-noise ratio (SNR) is high. It should be noted here that this procedure of creating an echo does work because of the deterministic component of dephasing in the T2* process; the stochastic part of the dephasing process can, of course, not be turned into a rephasing one.

7.2.3.4 *Image Reconstruction*

An example of an MRI procedure including the measurement and image reconstruction, is shown in Fig. 7.15. In this case we assume that the sequence displayed in Fig. 7.14 is repeated across a range of phase coding gradients. Note that in the examples in Figs. 7.14 and 7.15 only three of the series of phase-encoding gradients (color coded green, blue, and red) are shown.

Let's assume we already have selected a slice, using the G_{ss} gradient, and that $I(x,y)$ represents the intensity of the image in that slice. In the example in Fig. 7.15, we simplify the signal contributions of the voxels in the selected slice by a 5×5 grid. The contribution of a single, small voxel to the measurement of the slice's echo, i.e., a local voxel signal $S(x,y)$ can be represented by an oscillation that is weighted by the local intensity:

$$S(x, y, t) = I(x, y)e^{jF}. \tag{7.27}$$

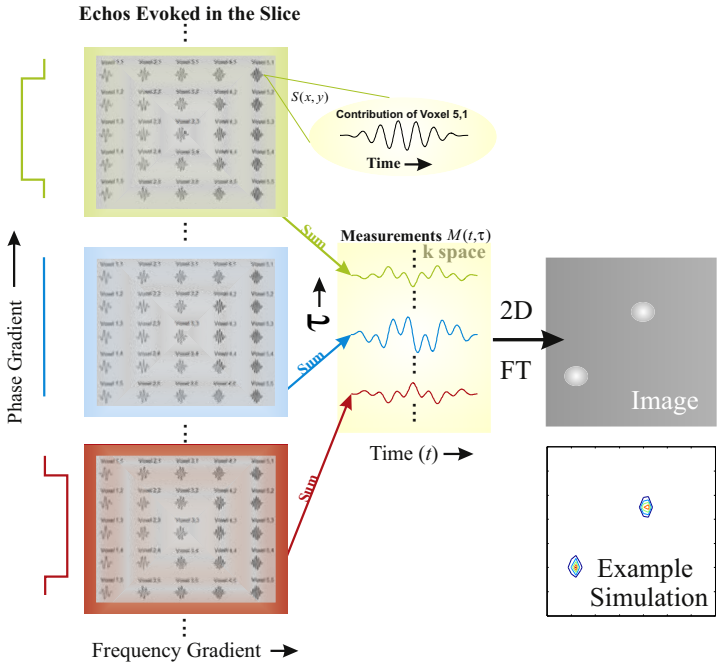


FIGURE 7.15 Diagram of a magnetic resonance imaging reconstruction procedure showing three out of 2^n measurements in k space and its 2-D Fourier transform generating the image. At each repetition τ of the pulse sequence, each voxel contributes its portion $S(x,y)$ to the measured signal $M(t,\tau)$. The example simulation was done with MATLAB® scripts pr7_3.m and pr7_4.m. Further details are provided in the text.

Here, F is a function of x and y, represented by a frequency- and phase-encoded component, respectively: $F = 2\pi f_x t + \Phi_y$, where the frequency f_x encodes x and the phase Φ_y encodes the y coordinate. We now introduce another variable τ to denote the subsequent measurements with increasing phase-encoding gradient (in Figs. 7.14 and 7.15, three such examples of changing G_{PE} are shown in red, blue, and green). The subsequent measurements across different phase-encoding gradients are placed in order, creating a data set that is called the k space. If we now increase Φ_y between each repetition of the pulse sequence in steps of ϕ_y, we get $\Phi_y = 2\pi\phi_y\tau$. Combining all this with Eq. (7.27), we get the following expression for the individual voxel contribution:

$$S(x, y, t, \tau) = I(x, y)e^{j\left(2\pi f_x t + 2\pi\phi_y\tau\right)}.$$

The procedure to measure so-called raw MRI data is to step through a series of values of G_{PE}, i.e., subsequent values of τ, and for each value of τ to measure the echo that the slice emits as a function of t, which we

represent by $M(t,\tau)$. This measurement includes the contributions of all the voxels. Thus, we integrate $S(x,y,t,\tau)$, i.e., the contributions over x and y:

$$M(t,\tau) = \int_x \int_y I(x,y) e^{j(2\pi f_x t + 2\pi \phi_y \tau)} dxdy. \tag{7.28}$$

Note that due to the encoding, we have that f_x and ϕ_y are proportional to G_{RO} (here G_x) and G_{PE} (here G_y), respectively. Since these gradients themselves are proportional to x and y, we have: $f_x = ax$ and $\phi_y = by$, with a and b constants. Next we change notation and use $u = ax$ and $v = by$, and rewrite Eq. (7.28):

$$M(t,\tau) = \frac{1}{ab} \int_u \int_v I(u,v) e^{j(2\pi ut + 2\pi v\tau)} dudv. \tag{7.29}$$

Here we recognize that $M(t,\tau)$ is a scaled version of the 2D ***inverse Fourier transform*** of the image, and that $u \Leftrightarrow \tau$ and $v \Leftrightarrow \tau$ are Fourier transform pairs (see Eq. 7.10). Thus $M(t,\tau) \Leftrightarrow I(x,y)$ is also a Fourier transform pair. Hence, by using the duality property (Chapter 6), we can reconstruct image $I(x,y)$ by computing the Fourier transform of $M(t,\tau)$, i.e., the k space.

Fig. 7.15 shows an example of an image reconstruction for a very simple case, an image with two highlights. This reconstruction was simulated with Matlab file pr7_4.m.

```
% pr7_4.m
% Simulation of two sources at xx1,yy1 and xx2,yy2

clear;
close all;

t=-.32:.01:.31;                 % time of k-space (series of 64 points)
h=hann(64);
k=zeros(32,64);                 % the measured signal in k-space

% location and amplitude of peaks in the image
xx1=10;
yy1=10;
xx2=40;
yy2=25;
amp=100;

for tau=1:32                    % the tau parm of k space
   pmax=(63.5/31)*tau-32.5;     % maximum phase
```

```
   for x=1:64;                    % x,y coordinates of the image
      for y=1:32;
         fx=(49/64)*x;            % the freq parm
         py=(pmax/32)*y;          % the phase parm
         sv=cos(2*pi*fx*t+py);    % individual voxel contribution
         % NOTE that we only usr the real component of the oscillator
         %      reflecting that we have one coil to pick up the signal

         % Inject a peak in the image at xx1,yy1
         if (x==xx1)&(y==yy1);
            sv=sv*amp;
         end;
         % Inject a peak in the image at xx2,yy2
         if (x==xx2)&(y==yy2);
            sv=sv*amp;
         end;

         sv=sv.*h';               % smooth with Hann
         k(tau,:)=k(tau,:)+sv;    % fill k space
      end;
   end;
end;
% k space K is filled
figure;
surf(k);
title('k space');

figure;
contour(k);
title('k space');

% compute the 2D fft
im=fft2(k);
figure;
contour(abs(im).^2);
axis([0 35 0 14]);
axis('square');
tmp=max(max(abs(im).^2))/10^9;
ttl=['coord: ' num2str(xx1) ',' num2str(yy1) ' and ' num2str(xx2) ','
num2str(yy2) ' amp: ' num2str(amp) ' max: ' num2str(tmp)];
title(ttl);
xlabel('x (unscaled)');
ylabel('y (unscaled)');
drawnow;
```

APPENDIX 7.1

Parseval's theorem states that the energy of a signal in the time domain equals the energy of the transformed signal in the frequency domain. Preservation of this equality is the underlying reason why the spectrum is normalized by $1/N$ in Eq. (7.1). In order to understand this normalization, we will first determine what the relationship is between the energy of a time series and its equivalent representation as a complex Fourier series. From this result we will subsequently develop the normalization for the DFT.

Consider three finite time series $f(t)$, $f_1(t)$, and $f_2(t)$ periodic over period $T = N\Delta t$, with $f(t) = f_1(t)\, f_2(t)$. For each of the time series we can generate the complex Fourier series (Eqs. 5.19 and 5.20), i.e.:

$$\begin{aligned} f_1(t) &= \sum_{n=-\infty}^{\infty} d_n e^{jn\omega t} \leftrightarrow d_n = \frac{1}{T}\int_T f_1(t) e^{-jn\omega t} dt \\ f_2(t) &= \sum_{n=-\infty}^{\infty} g_n e^{jn\omega t} \leftrightarrow g_n = \frac{1}{T}\int_T f_2(t) e^{-jn\omega t} dt \\ f(t) &= \sum_{n=-\infty}^{\infty} c_n e^{jn\omega t} \leftrightarrow c_n = \frac{1}{T}\int_T f(t) e^{-jn\omega t} dt \end{aligned} \tag{A7.1-1}$$

Using the relationship between the three signals, we can write c_n as:

$$c_n = \frac{1}{T}\int_T f_1(t) f_1(t) e^{-jn\omega t} dt \tag{A7.1-2}$$

Replacing f_1 by its Fourier series:

$$c_n = \frac{1}{T}\int_T \left(\sum_{m=-\infty}^{\infty} d_m e^{jm\omega t} \right) f_2(t) e^{-jn\omega t} dt \tag{A7.1-3}$$

and changing the order of the integral and summation operations results in:

$$c_n = \sum_{m=-\infty}^{\infty} d_m \left(\frac{1}{T}\int_T f_2(t) e^{-j(n-m)\omega t} dt \right) \tag{A7.1-4}$$

Using Eq. (A7.1-1), the terms in between the brackets can be set to g_{n-m}:

$$c_n = \frac{1}{T}\int_T f_1(t) f_2(t) e^{-jn\omega t} dt = \sum_{m=-\infty}^{\infty} d_m g_{n-m} \tag{A7.1-5}$$

For $n = 0$, we find that the following relationship must be satisfied:

$$\frac{1}{T}\int_T f_1(t)f_2(t)dt = \sum_{m=-\infty}^{\infty} d_m g_{-m} \tag{A7.1-6}$$

In order to evaluate the power of a single time series x we can substitute functions $x(t)$ and $x^*(t)$ for $f_1(t)$ and $f_2(t)$:

$$P = \frac{1}{T}\int_T x(t)x^*(t)dt \tag{A7.1-7}$$

The * in $x^*(t)$ indicates the complex conjugate of $x(t)$. The complex Fourier series and coefficients of $x(t)$ and $x^*(t)$ can be determined with:

$$\begin{aligned} x(t) &= \sum_{n=-\infty}^{\infty} h_n e^{jn\omega t} \;\leftrightarrow\; h_n = \frac{1}{T}\int_T x(t)e^{-jn\omega t}dt \\ x^*(t) &= \sum_{n=-\infty}^{\infty} y_n e^{jn\omega t} \;\leftrightarrow\; y_n = \frac{1}{T}\int_T x^*(t)e^{-jn\omega t}dt \end{aligned} \tag{A7.1-8}$$

The expression for y_n can be written as:

$$y_n = \left(\frac{1}{T}\int_T x(t)e^{jn\omega t}dt\right)^* = h^*_{-n} \tag{A7.1-9}$$

Combining Eqs. (A7.1-9), (A7.1-6), and (A7.1-7):

$$\begin{aligned} &\frac{1}{T}\int_T x(t)x^*(t)dt = \sum_{n=-\infty}^{\infty} h_n h_n^* \\ &\rightarrow \frac{1}{T}\int_T |x(t)|^2 dt = \sum_{n=-\infty}^{\infty} |h_n|^2 \end{aligned} \tag{A7.1-10}$$

Note that g_{-m} from Eq. (A7.1-6) is substituted by $h^*_{-(-n)} = h_n^*$ in Eq. (A7.1-10).

Finally, we translate the above relationship in Eq. (A7.1-10) for the complex Fourier series into the DFT of a signal with finite duration. The expression on the left-hand side in Eq. (A7.1-10) becomes $\frac{1}{N}\sum_{n=0}^{N-1} x^2(n)$. The expression to the right of the equal sign is proportional to the DFT, but must be corrected by a factor $1/T_0$ (Eqs. (6.4) and (6.5)) and Δt

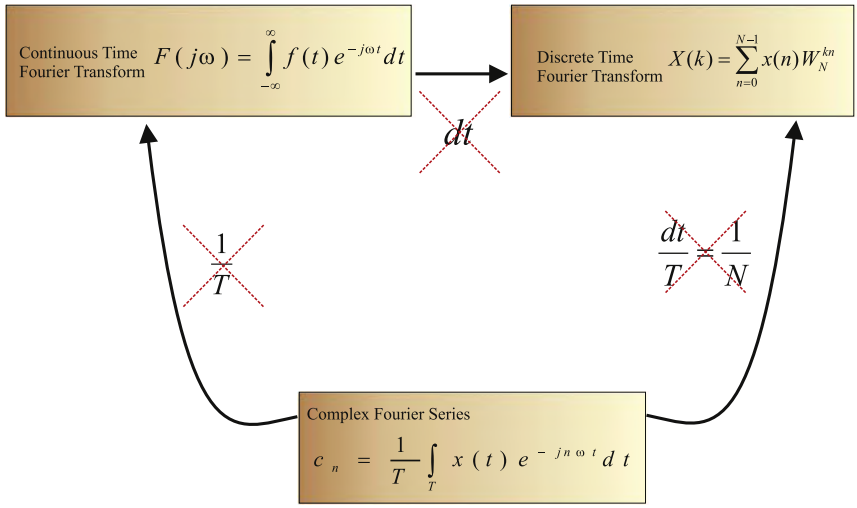


FIGURE A7.1-1 Overview of normalizations in the different flavors of the Fourier analysis. This diagram shows that the continuous Fourier transform relative to the Fourier series loses a factor $1/T$. The discrete Fourier transform loses a factor dt relative to the continuous time transform. Combining these two factors demonstrates that the discrete Fourier transform differs from the Fourier series by a factor $dt/T = 1/N$.

(Eqs. (6.18) and (6.21)), i.e., $\Delta t/T_0 = 1/N$ (see the diagram in Fig. A7.1-1). Using $X(k)$ to denote the DFT and taking into account this correction, the expression on the right-hand side becomes: $\sum_{k=0}^{N-1} \frac{1}{N} X(k) \frac{1}{N} X^*(k) \rightarrow \frac{1}{N^2} \sum_{k=0}^{N-1} |X(k)|^2$. Combining the above, we obtain Parseval's identity for the DFT:

$$\frac{1}{N} \sum_{n=0}^{N-1} x^2(n) = \frac{1}{N^2} \sum_{k=0}^{N-1} |X(k)|^2 \rightarrow \sum_{n=0}^{N-1} x^2(n) = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2 \qquad \text{(A7.1-11)}$$

In Eq. (A7.1-11) we see that the energy in the time series $x(n)$ can be related to the energy in the spectrum by the factor $1/N$. This is the underlying reason for the normalization of $X(k)\, X^*(k)$ by $1/N$ in Eq. (7.1). Of course, one might disagree with this approach and prefer the normalization that derives from the amplitude, i.e., normalize by $1/N^2$. Again if you only show half of the power spectrum, you must multiply the correction factor by 2, giving correction factors of $2/N$ or $2/N^2$.

EXERCISES

7.1 Evaluate the result from Exercise 6.3e [the CFT of $\cos(\omega_0 t + \phi)$].
   a. What is the difference between the CFTs of $\cos(\omega_0 t + \phi)$ and $\cos(\omega_0 t)$?
   b. Interpret this difference.
   c. Is the amplitude of the power spectrum different from that of $\cos(\omega_0 t)$?

7.2 Move EEG data (for example, a channel from the data.eeg file by using MATLAB® script pr1_1, see Chapter 1) to MATLAB®. Calculate and plot the power, amplitude, and phase spectra.

7.3 Create an array in MATLAB® using

```
T=1;
t=0:.1:T;
N=length(t);
y=cos(2*pi*1*t);
figure;plot(t,y);
```

   a. Compute the power spectrum of this signal and plot it.
   b. Compute and plot the power spectrum again after you recomputed the signal y for $T = 1.3$.
   c. Now multiply the signal y with a Hann window and compute the power spectrum again.
   d. Interpret your results.

7.4 Spectral analysis of sine waves.
   a. Find the CFT for $x(t) = \sin(\omega_0 t)$ by evaluating $\int_{-\infty}^{\infty} x(t)\exp(-j\omega t)dt$ and plot the result.
   b. Find the CFT for $y(t) = \cos(\omega_0 t) + \cos(5\omega_0 t)$ and plot the result.
   c. Use MATLAB® to compute an epoch of both $x(t)$ and $y(t)$ and to determine the amplitude spectrum (based on the fft command).
   d. Comment on your results obtained in a, b, and c.

7.5 Show in a similar manner as done for Eq. (7.9), that the inverse Fourier transform in Eq. (7.10) is correct. Why is there no scaling factor of 2π involved in this expression?

7.6 Use the MATLAB® file lena_double and the MATLAB® fft2 function to demonstrate the inversion effect of a *4f* lens system as shown in Fig. 7.8, governed by Eq. (7.10). You can load and depict the data by the following commands.

```
load lena_double;                % load the 512 x 512 matrix
lena=mat2gray(lena_double);      % convert the matrix to a gray
                                   scaled image

figure;
imshow(lena)                     % show the picture
title('Lena - Original')
```

(Note that with respect to the duality property the discrete FT (DFT and FFT) have a scaling difference by a factor of N.)

7.7 Medical imaging.

a. Use MATLAB® and the radon function to determine and plot the radon transform of a delta function that is NOT positioned in the origin (i.e., the middle) in a 128×128 digital image (e.g., try some locations such as position 64,96).
b. Reconstruct and plot your image from the radon transform using the iradon command.
c. Use Eq. (7.12) to explain your finding.
[Hint: Use the expression a delta function **not** located in the center, e.g., at a position $(r_0, \phi_0) = (0.5, 0)$, and substitute that for $a(r, \phi)$ in the equation and evaluate the integral.]
d. Simulate an MRI recording and image reconstruction of the same delta function. First construct the k-space, then do the image reconstruction (you may use a modification of pr7_4.m). Plot all your results.

References

Bushong, S.C., 1996. Magnetic Resonance Imaging Physical and Biological Principles, second ed. Mosby, St Louis, MO, USA.

Goodman, J., 2004. Introduction to Fourier Optics, third ed. Roberts and Co, Greenwood Village, CO, USA.

Haacke, E.M., Brown, R.W., Thompson, M.R., Venkatesan, R., 1999. Magnetic Resonance Imaging Physical Principles and Sequence Design. Wiley, New York, NY, USA.

Hecht, E., 2016. Optics, fifth ed. Addison-Wesley, Boston, MA, USA.

Weishaupt, D., Köchli, V.D., Marincek, B., 2006. How Does MRI Work? Springer, Berlin, Germany.

CHAPTER

8

Lomb's Algorithm and Multitaper Power Spectrum Estimation

8.1 OVERVIEW

This chapter describes two specialized algorithms to obtain power spectrum estimates. In Section 8.2 we introduce Lomb's algorithm (Lomb, 1976) to analyze unevenly sampled data sets that may have relevance when analyzing data with irregularly occurring events such as measurements of a series of heart beats or spike train recordings. In Section 8.3 we discuss the so-called multitaper approach, initially introduced by Thomson (1982), which is used to improve the estimation of the power spectrum by reducing spectral leakage across frequency bands and the noise component of the spectral estimate. The multitaper algorithm can be considered as an extended application of the data windows we discussed in Section 7.1.1. Examples of the use of Lomb's algorithm and the multitaper approach are demonstrated with MATLAB® scripts.

8.2 UNEVENLY SAMPLED DATA

In most measurements we have evenly sampled the data, e.g., the interval Δt between the sample points of the time series is constant, pixels in a picture have uniform interdistance, etc. Usually, this is an appropriate assumption, but there are examples where uneven sampling cannot be avoided. Spike trains (Chapter 20) or time series representing the heart rate (Van Drongelen et al., 2009) are examples; in these cases one may

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00008-4

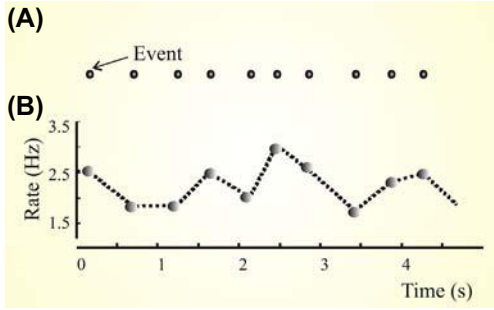


FIGURE 8.1 The QRS complexes in the ECG or extracellularly recorded spike trains can be considered as a series of events such as shown in panel (A). The rate of events can be depicted as the inverse of the interval between the events (B); here the inverse of the interval between each pair of events is plotted at the instant of the second event of the pair. The signal in panel (B) is unevenly sampled because the rate measure is available only at the occurrence of the events; the *dotted line* is a linear interpolation between these measures.

consider the spike or the heart beat to represent events that sample an underlying process that is invisible to the experimenter (Fig. 8.1A).

The heart rate signal is usually determined by measuring the intervals between peaks in the QRS complexes (see Chapters 1 and 4, Figs. 1.4 and 4.7). The inverse value of the interval between pairs of subsequent QRS complexes can be considered a measure of the instantaneous rate (Fig. 8.1B). This rate value can be positioned in a time series at the instant of either the first or second QRS complex of the pair, and because the heart beats do occur at slightly irregular intervals the time series is sampled unevenly. This example for the heart beat could be applied, in a similar fashion, to determine the firing rate associated with a spike train.

When a signal is unevenly sampled, many algorithms that are based on a fixed sample interval Δt, such as the discrete Fourier transform (DFT) or fast Fourier transform (FFT) algorithms (see Eqs. (6.16)−(6.21)), cannot be applied. In principle there are several solutions to this problem.

1. An evenly sampled time series can be constructed from the unevenly sampled one by using interpolation. In this approach the original signal is resampled at evenly spaced intervals. The interpolation technique (e.g., linear, cubic, spline) may vary with the application. In MATLAB® resampling may be accomplished with the interp1 command or any of the other related functions. After resampling the time series one can use standard Fourier analysis methods. The disadvantage is that the interpolation algorithm may introduce frequency components that are not related to the underlying process.

2. The measurements can be represented as the number of events in a binned trace; now our time series is a sequence of numbers, one number for each bin. Because the bins are equally spaced, standard DFT/FFT can be applied. In case of low-frequency activity, the bins must be relatively wide to avoid an overrepresentation of empty bins. The disadvantage is that wide bins represent a low sample rate and associated low Nyquist frequency.
3. The most elegant solution is to use Lomb's algorithm for estimating the spectrum. This algorithm is especially designed to deal with unevenly sampled time series directly, without the assumptions demanded by interpolation and resampling techniques (Lomb, 1976; Scargle, 1982; Press et al., 2007; Van Drongelen et al., 2009). The background and application of this algorithm will be further described in Sections 8.2.1 and 8.2.2.

8.2.1 Lomb's Algorithm

Lomb's idea is similar to the development of the Fourier series, namely to represent a signal by a sum of sinusoidal waves (Chapter 5). Lomb's procedure is to fit a demeaned time series x that **may be sampled unevenly** to a weighted pair of cosine and sine waves, the cosine is weighted by coefficient a and the sine by coefficient b. The fitting procedure is performed over N samples of $x(n)$ obtained at times t_n and repeated for each frequency f

$$\boxed{P(a,b,f,t_n) = a\cos(2\pi f t_n) + b\sin(2\pi f t_n)} \tag{8.1}$$

Coefficients a and b are unknown and must be obtained from the fitting procedure. For example, we can minimize the error $\varepsilon^2 = \sum_{n=0}^{N-1}[P - x(n)]^2$ for all samples n and each frequency f. To accomplish this, we follow the same procedure for developing the Fourier series (Chapter 5) and set the partial derivative for each coefficient to zero, i.e.,:

$$\partial\varepsilon^2/\partial a = 0 \tag{8.2a}$$

and

$$\partial\varepsilon^2/\partial b = 0. \tag{8.2b}$$

For the condition in Eq. (8.2a) we get:

$$\partial\varepsilon^2/\partial a = \sum_{n=0}^{N-1} 2\left[\underbrace{a\cos(2\pi f t_n) + b\sin(2\pi f t_n)}_{P} - x(n)\right]\underbrace{\cos(2\pi f t_n)}_{\partial[P-x(n)]/\partial a} = 0$$

This and a similar expression obtained from the condition in Eq. (8.2b) result in the following two equations:

$$\sum_{n=0}^{N-1} x(n)\cos(2\pi f t_n) = a\sum_{n=0}^{N-1}\cos^2(2\pi f t_n) + b\sum_{n=0}^{N-1}\cos(2\pi f t_n)\sin(2\pi f t_n) \quad (8.3a)$$

and

$$\sum_{n=0}^{N-1} x(n)\sin(2\pi f t_n) = a\sum_{n=0}^{N-1}\cos(2\pi f t_n)\sin(2\pi f t_n) + b\sum_{n=0}^{N-1}\sin^2(2\pi f t_n) \quad (8.3b)$$

Thus far the procedure is similar to the standard Fourier analysis described in Chapter 5. The special feature in Lomb's algorithm is that for each frequency the sample times t_n will now be shifted by an amount τ (Fig. 8.2). Thus, in Eqs. (8.3a) and (8.3b) t_n becomes $t_n - \tau$. The critical step is that, for each frequency f, we select τ so that the cosine-sine cross-terms disappear, that is:

$$\sum_{n=0}^{N-1}\cos(2\pi f(t_n-\tau))\sin(2\pi f(t_n-\tau)) = 0 \quad (8.4)$$

Using $\cos(A)\sin(B) = \frac{1}{2}[\sin(A-B) - \sin(A+B)]$, this can be simplified into:

$$\frac{1}{2}\left[\sum_{n=0}^{N-1}\underbrace{\sin(0)}_{0} - \sin(4\pi f(t_n-\tau))\right] = 0 \rightarrow \sum_{n=0}^{N-1}\sin(4\pi f(t_n-\tau)) = 0$$

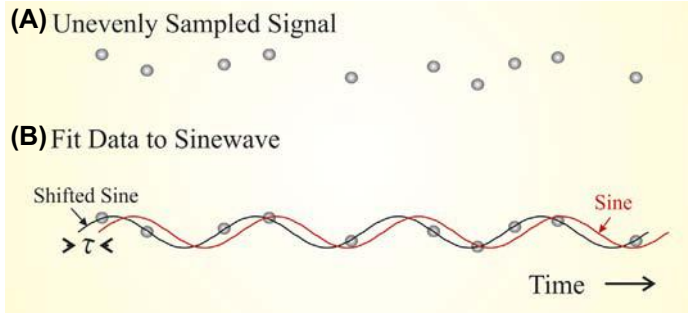


FIGURE 8.2 The Lomb algorithm fits sinusoidal signals to time series that may be unevenly sampled as in the example of panel (A). The fit procedure is optimized by shifting the sinusoidal signals by an amount τ (B).

To separate the expressions for t_n and τ, we use the trigonometric relationship:

$$\sin(A-B)=\sin(A)\cos(B)-\cos(A)\sin(B),$$

and we get

$$\sum_{n=0}^{N-1}\sin(4\pi f t_n)\cos(4\pi f\tau)-\sum_{n=0}^{N-1}\cos(4\pi f t_n)\sin(4\pi f\tau)$$
$$=\cos(4\pi f\tau)\sum_{n=0}^{N-1}\sin(4\pi f t_n)-\sin(4\pi f\tau)\sum_{n=0}^{N-1}\cos(4\pi f t_n)=0$$

This can be further simplified into:

$$\sin(4\pi f\tau)/\cos(4\pi f\tau)=\tan(4\pi f\tau)=\sum_{n=0}^{N-1}\sin(4\pi f t_n)\Big/\sum_{n=0}^{N-1}\cos(4\pi f t_n)$$

Hence, condition (8.4) is satisfied if:

$$\boxed{\tau=\tan^{-1}\left[\sum_{n=0}^{N-1}\sin(4\pi f t_n)\Big/\sum_{n=0}^{N-1}\cos(4\pi f t_n)\right]\Big/4\pi f} \tag{8.5}$$

The value of variable τ as a function of frequency f can be found with Eq. (8.5) and with $t_n \rightarrow (t_n-\tau)$ the cross-terms in Eq. (8.4) become zero. By applying the appropriate shift τ, we can determine the a and b coefficients for each frequency from the simplified expressions obtained from Eqs. (8.3a) and (8.3b) without the cross-terms:

$$\sum_{n=0}^{N-1}x(n)\cos(2\pi f(t_n-\tau))=a\sum_{n=0}^{N-1}\cos^2(2\pi f(t_n-\tau))$$
$$\rightarrow\boxed{a=\sum_{n=0}^{N-1}x(n)\cos(2\pi f(t_n-\tau))\Big/\sum_{n=0}^{N-1}\cos^2(2\pi f(t_n-\tau))} \tag{8.6a}$$

and

$$\sum_{n=0}^{N-1}x(n)\sin(2\pi f(t_n-\tau))=b\sum_{n=0}^{N-1}\sin^2(2\pi f(t_n-\tau))$$
$$\rightarrow\boxed{b=\sum_{n=0}^{N-1}x(n)\sin(2\pi f(t_n-\tau))\Big/\sum_{n=0}^{N-1}\sin^2(2\pi f(t_n-\tau))} \tag{8.6b}$$

Now we compute the sum of $P^2(a,b,f,t_n)$, that is the sum of squares of the sinusoidal signal in Eq. (8.1) for all t_n, to obtain an expression that is proportional with the power spectrum S of $x(n)$ as a function of f:

$$\begin{aligned} S(f,a,b) &= \sum_{n=0}^{N-1} P^2(a,b,f,t_n) \\ &= \sum_{n=0}^{N-1} \left[a^2 \cos^2(2\pi f(t_n-\tau)) + b^2 \sin^2(2\pi f(t_n-\tau)) + \underbrace{cross-terms}_{0} \right] \end{aligned} \tag{8.7}$$

Since we shift by τ, all *cross-terms* vanish and by substitution of the expressions for the a and b coefficients in Eq. (8.7) we get:

$$\begin{aligned} S(f) = &\frac{\left[\sum_{n=0}^{N-1} x(n)\cos(2\pi f(t_n-\tau))\right]^2}{\left[\sum_{n=0}^{N-1} \cos^2(2\pi f(t_n-\tau))\right]^2} \sum_{n=0}^{N-1} \cos^2(2\pi f(t_n-\tau)) \\ &+ \frac{\left[\sum_{n=0}^{N-1} x(n)\sin(2\pi f(t_n-\tau))\right]^2}{\left[\sum_{n=0}^{N-1} \sin^2(2\pi f(t_n-\tau))\right]^2} \sum_{n=0}^{N-1} \sin^2(2\pi f(t_n-\tau)) \end{aligned}$$

This can be further simplified into:

$$S(f) = \frac{\left[\sum_{n=0}^{N-1} x(n)\cos(2\pi f(t_n-\tau))\right]^2}{\sum_{n=0}^{N-1} \cos^2(2\pi f(t_n-\tau))} + \frac{\left[\sum_{n=0}^{N-1} x(n)\sin(2\pi f(t_n-\tau))\right]^2}{\sum_{n=0}^{N-1} \sin^2(2\pi f(t_n-\tau))} \tag{8.8}$$

The expression for the power spectrum in Eq. (8.8) is sometimes divided by 2 (to make it equal to the standard power spectrum based on the Fourier transform; see Appendix 8.1), or by $2\sigma^2$ (σ^2—variance of x) for the determination of statistical significance of spectral peaks. Some of the background for this normalization is described in Appendix 8.1, for

more details, see Scargle (1982). By applying the normalization we finally get:

$$S(f) = \frac{1}{2\sigma^2}\left\{\frac{\left[\sum_{n=0}^{N-1} x(n)\cos(2\pi f(t_n-\tau))\right]^2}{\sum_{n=0}^{N-1}\cos^2(2\pi f(t_n-\tau))} + \frac{\left[\sum_{n=0}^{N-1} x(n)\sin(2\pi f(t_n-\tau))\right]^2}{\sum_{n=0}^{N-1}\sin^2(2\pi f(t_n-\tau))}\right\} \tag{8.9}$$

From the derivation above, we can see that within Lomb's procedure unevenly sampled data are allowed (but not required). Note that in Eqs. (8.7) and (8.8) we did not compute power as the square of the cosine and sine coefficients, a and b, as we would do in the standard Fourier transform; this is because in Lomb's approach the sinusoidal signals are not required to have a complete period within the epoch determined by the samples $x(n)$. Because we do not have this requirement, the frequency f is essentially a continuous variable and the spectral estimate we obtain by this approach is therefore not limited by frequency resolution (in the DFT, the frequency resolution is determined by the total epoch of the sampled data) and range (in the DFT, the maximum frequency is determined by the Nyquist frequency). However, to avoid misinterpretation, it is common practice to limit the bandwidth of the Lomb spectrum to half the average sample rate or less. Similarly, the frequency resolution is usually not set smaller than the inverse of the signal's epoch.

8.2.2 A MATLAB® Example

To test Lomb's algorithm we apply it to the same example as we employed in pr7_1.m: i.e., two sinusoidal signals of 50 and 130 Hz plus a random noise component (Fig. 7.2A). In this example (pr8_1.m), we sample the signal with randomly distributed intervals (2000 points) and specify a frequency scale (f in the script) up to 500 Hz. Subsequently we use Eqs. (8.5), (8.8), and (8.9) to compute τ (tau in the script) and the unscaled and scaled versions of $S(f)$ (Pxx in the script) of input $x(n)$ (x in the script).

The following script (pr8_1.m) uses the Lomb algorithm to compute the spectrum from an unevenly sampled signal. The output of the script is a plot of the input (an unevenly sampled time domain) signal and its associated Lomb spectrum.

```
% pr8_1.m
% Application of Lomb Spectrum
clear;

t=rand(2000,1); t=sort(t);          % An array of 2000 random sample intervals
f=[1:500];                          % The desired frequency scale
% frequencies same as pr7_1.m
f1=50;
f2=130;
% data plus noise as in pr7_1.m
x=sin(2*pi*f1*t)+sin(2*pi*f2*t);
x=x+randn(length(t),1);
var=(std(x))^2;                     % The signal's variance

% Main Loop
for i=1:length(f)
   h1 = 4*pi*f(i)*t;
   % Eq. (8.5)
   tau = atan2(sum(sin(h1)),sum(cos(h1)))/(4*pi*f(i));
   h2 = 2*pi*f(i)*(t - tau);
   % Eq. (8.8)
   Pxx(i) = (sum(x.*cos(h2)).^2)/sum(cos(h2).^2) +
                                          (sum(x.*sin(h2)).^2)/sum(sin(h2).^2);
end;
% Normalize; Eq. (8.9)
Pxx=Pxx/(2*var);
% Plot the Results
figure;
subplot(2,1,1),plot(t,x,'.-')
title(' Irregularly Sampled Signal (USE ZOOM TO INSPECT UNEVEN
  SAMPLING)')
xlabel('Time (s)');ylabel('Amplitude')
subplot(2,1,2),plot(f,Pxx);
title(' Lomb Spectrum')
xlabel('Frequency (Hz)');ylabel('Normalized Power')
```

8.3 ERRORS IN THE PERIODOGRAM

The most commonly applied technique to obtain an estimated power spectrum $\widehat{S}(f)$ of a sampled time series $x(t)$ of finite length N, is to compute the Fourier transform $X(j2\pi f)$, followed by multiplication by its complex conjugate $X^*(j2\pi f)$ and scaling by the length of the finite observation, i.e., $\widehat{S}(f) = X(j2\pi f) \cdot X^*(j2\pi f)/N$ (see Chapter 7). The reliability of this estimated power spectrum, also called the periodogram, is significantly reduced both by:

1. the variance of the estimate $\widehat{S}(f)$ at each frequency f (i.e., the spectra computed with the above recipe usually look rather noisy), and
2. by leakage of energy across frequencies, creating a bias.

The leakage is due to the fact that we take a finite section of signal, which equates to multiplying the signal by a rectangular window (sometimes called a boxcar) (see three upper traces in Fig. 8.3). The Fourier transform of the rectangular window shows multiple side lobes (Fig. 8.4A) and this causes the leakage problem. Note that multiplication in the time domain is equivalent to a convolution in the frequency domain (Chapter 13). Hence the transform of a truncated signal is a convolution of the signal transform and the boxcar's Fourier transform. This procedure results in leakage of energy across the estimated power spectrum due to the side lobes associated with the window's spectrum.

A window is also called a **taper**, which is the term we will use in the following text when we present the so-called multitaper algorithm. This approach was specifically designed by Thomson (1982) to address both the noise and leakage problems outlined above.

8.3.1 Multitaper Power Spectrum Estimation

A strategy to reduce leakage in the frequency domain is to first multiply the signal in the time domain with a (nonrectangular) taper characterized by a Fourier transform with less energy in its side lobes. The approach of using a taper is explained in more detail in Section 7.1.1. Using a sampled signal $x(t)$ of length N and taper $a(t)$, we can estimate the signal's unscaled spectrum as:

$$\widehat{S}(f) = \left| \sum_{t=0}^{N-1} x(t)a(t)e^{-2\pi jft} \right|^2. \tag{8.10}$$

If taper $a(t)$ is not explicitly present in Eq. (8.10), or (equivalently) if it is defined as a rectangular window (in MATLAB® called a boxcar), the spectral estimate $\widehat{S}(f)$ is called the periodogram and, as explained above, one may expect that bias due to spectral leakage will be strong.

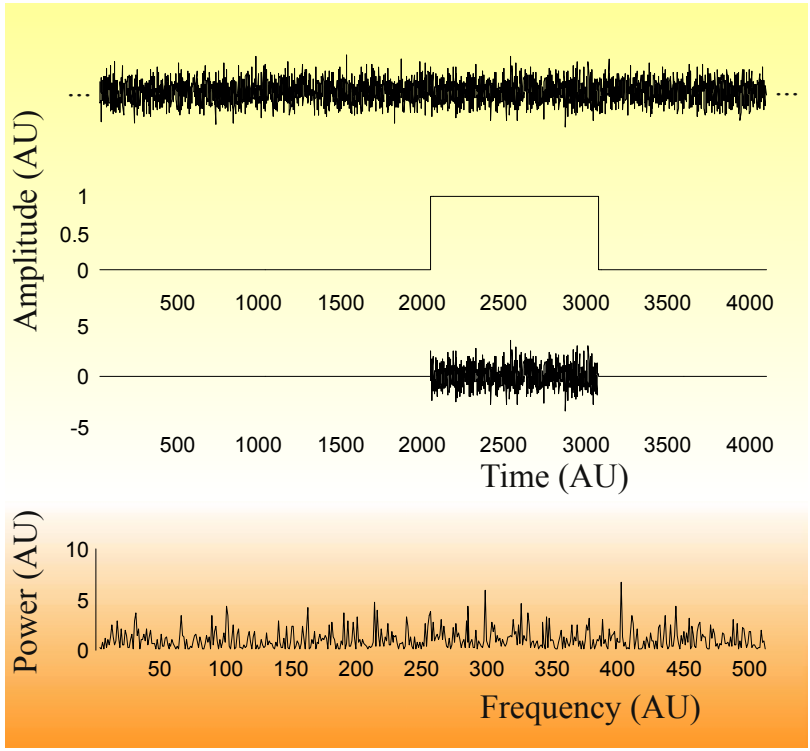


FIGURE 8.3 Power spectrum of a finite observation. Example of a Gaussian white noise signal (zero mean and **unit variance**; upper trace) and a rectangular/boxcar window (second trace). When signal and window are multiplied, the result (depicted in the third trace) mimics a finite observation of the signal with a length that is equal to the length of the rectangular window. The bottom trace shows the periodogram of the finite observation. **From the bottom graph it is obvious that the periodogram is very noisy due to the variance in this estimate.** The expected power is equal to the variance of the Gaussian distribution from which the samples in the time domain signal are drawn (**one in this example**). Leakage of energy across spectral bands, which is also present in this estimate (due to the applied boxcar window), cannot be directly observed in this plot but can be deduced from the spectrum of the boxcar itself (e.g., see Figs. 7.4 and 8.4) when convolved with the signal in the frequency domain; see text and Section 7.1.1 for further explanation.

Of course, the use of a taper affects the estimate by reducing leakage but it doesn't change the variance, i.e., the noise, of the estimate at each frequency. A common approach to reduce the variance is:

1. taking an average across several frequencies; and/or
2. computing an average spectrum from several time epochs.

Averaging across frequencies reduces spectral resolution and using multiple epochs is undesirable if the signal may be nonstationary. See

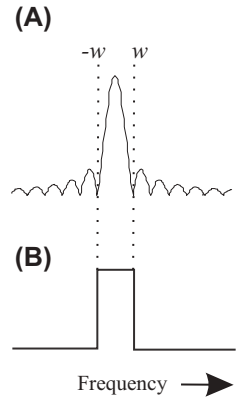


FIGURE 8.4 The spectrum of a rectangular window (boxcar) depicted in panel (A) has most energy concentrated in the desired bandwidth indicated between the *stippled vertical lines*, but also shows a lot of energy in the side lobes. The function in (B) is the spectrum of an ideal window where all energy is located within the interval between $-w$ and w. When a spectrum is convolved with the spectra of these windows, the one in (A) generates significant leakage and therefore bias, whereas the ideal one in (B) gives a bias-free estimate. As shown in Figs. 8.5 and 8.6, even the best optimized window isn't ideal as in panel (B), but definitely better than the one from the boxcar window depicted in panel (A).

Chapter 4 for a general overview of the noise reduction property of the average procedure.

In Section 8.3.2, we will first introduce the use of multiple tapers and the rationale behind their derivation. In Section 8.3.3, we depict an example of the application of this procedure by using a Gaussian autoregressive process with a known power spectrum. From this example we conclude that there is a need for optimization of the procedure, which is presented (without proof) in Section 8.3.4. Finally, in Section 8.3.5, we show a concrete example of the optimized multitaper spectral estimate in MATLAB®.

8.3.2 The Use of Tapers

The multitaper approach, first described in a seminal paper by Thomson (1982), improves the spectral estimate by addressing both leakage and variance in the estimate. The basic approach is simple: what is better than using a taper? … the answer is: you use multiple tapers. In this approach, every taper v_k out of a set of K tapers is a bit different and reduces leakage of energy across frequencies. In addition, in Thomson's approach, the tapers are orthogonal and they are used to provide K orthogonal samples of the data $x(t)$. These samples are used to create a set

of K spectral estimates $\widehat{S}_k(f)$ that can be used to compute an average $\overline{S}(f)$ with reduced variance:

$$\overline{S}(f) = \frac{1}{K}\sum_{k=1}^{K}\widehat{S}_k(f). \tag{8.11}$$

As explained in Chapter 4, the averaging procedure will reduce the variance proportional with K. The general idea for the application of a taper is as follows. Each taper $a(t)$ is associated with an estimated spectrum that is given by Eq. (8.10). In order to maintain correct values for total power, we assume the tapers are normalized such that $\sum_{t=0}^{N-1}|a(t)|^2 = 1$. Furthermore, the power spectrum of taper $a(t)$ is $|A(f)|^2$, and its spectral properties are important because they determine, via convolution, the estimate of the power spectrum of our windowed (tapered) time series:

$$\widehat{S}(f) = \int_{-1/2}^{1/2}|A(f')|^2\, S(f-f')df'. \tag{8.12}$$

Note in Eq. (8.12) that we use the property that multiplication in the time domain is associated with convolution in the frequency domain; see Section 13.3.3.

A desirable taper will have low-amplitude spectral values for all larger values of $|f-f'|$. In other words, these tapers will have low energy in the side lobes. This leads to estimates for $\widehat{S}(f)$ that will mainly and correctly consist of values close to f. In this case, we will have minimal leakage and thus bias in the spectral estimate. Examples of the spectral properties of the absence of a taper as well as an ideal taper are shown in Fig. 8.4.

The reasoning continues as follows. Suppose we want to estimate our spectrum with a resolution bandwidth W, which is necessarily set at a value in between the resolution of the spectrum $1/N$ and the maximum of the spectrum, i.e., the Nyquist frequency (see also Fig. 7.1). To simplify notation, we set the sample interval equal to unity so that the Nyquist

frequency is normalized at ½. The fraction λ of the taper's energy within the selected frequency band is given by:

$$\lambda(N,W) = \frac{\int_{-W}^{W} |A(f)|^2 df}{\int_{-1/2}^{1/2} |A(f)|^2 df} \tag{8.13}$$

The **basic idea** is that we want to find the tapers with minimal leakage by maximizing λ, i.e., the fraction of the energy within bandwidth W is maximized. To make a long story short, one maximizes λ by setting the derivative of the expression in Eq. (8.13) with respect to vector $a(t)$ equal to zero. This equates to solving the following, well-known matrix eigenvalue problem (see Appendix 8.2 for further details):

$$D \cdot a = \lambda a, \tag{8.14}$$

in which the $N \times N$ matrix D has components $D_{t,t'} = \frac{\sin(2\pi W(t-t'))}{\pi(t-t')}$. Note that this function has even symmetry. Therefore D is a symmetric matrix. The solution has N eigenvalues (λ_0, λ_1, …, λ_k, …, λ_{N-1}) and orthonormal eigenvectors (v_0, v_1, …, v_k, …, v_{N-1}) (recall that we normalize them to unity: $\sum_{t=0}^{N-1} |v_k(t)|^2 = 1$). The eigenvectors v_k are so-called **discrete prolate spheroidal sequences**, also called **Slepian sequences** (Slepian, 1978). (A prolate sphere is an elongated spherical object with the polar axis greater than the equatorial diameter.)

Note that matrix D can be considered as the unscaled covariance matrix of the inverse transform of a rectangular window between $-W$ and W in the frequency domain (Fig. 8.4B, Appendix 8.2). Therefore, the computation of eigenvalues and eigenvectors of D is the same as performing principal component analysis (PCA) on the inverse transform of the ideal spectral window (see Chapter 28 for an introduction to PCA). The PCA generates the desired set of orthogonal (uncorrelated) tapers. Since the dimension of D is $N \times N$, there are N eigenvalues λ_k and eigenvectors v_k. The first set of components capture most of the properties of this ideal window, whereas the subsequent components capture less, so the question is which components to include when using them as tapers. The first eigenvalue is very close to one and associated with a very good eigenvector, i.e., a taper that minimizes leakage. Since we use the discrete Fourier transform for obtaining the spectral estimate, the number of points P in bandwidth W is a multiple of the spectral resolution $1/N$ of the sample time series: thus $W = P/N$ and $P = NW$. As shown in Table 8.1, for choices of $N = 128$ and $NW = 4$, 3, or 2, the first $2NW - 1$ eigenvalues are all very close to unity. Therefore, as a first approach, the first $2NW - 1$

TABLE 8.1 Fractional Leakage of Eigentapers

	$P\pi$ Prolate		
	$P = 4$	$P = 3$	$P = 2$
λ_0	0.9999999998	0.999999885	0.999948125
λ_1	0.999999978	0.999992014	0.997764652
λ_2	0.999999008	0.999750480	0.962155175
λ_3	0.999972984	0.995477689	0.733922358
λ_4	0.999500363	0.951033908	0.287339619
λ_5	0.993525891	0.725208760	[a]
λ_6	0.943750573	0.307789684	[a]
λ_7	0.721233936	0.060764834	[a]
Boxcar	0.974748450	0.966410435	0.949939339

[a] $\lambda_k < 0.05$.
From Park, J., Lindberg, C.R., Vernon, F.L., 1987. Multitaper spectral analysis of high-frequency seismograms. J. Geophys. Res. 92, 12675–12684 (Table 1).

normalized eigenvectors could be considered good to use for reducing leakage and also (by averaging their results) for reducing the variance of the spectral estimate.

As you can imagine, there are a few alternatives for the average procedure of the tapered power spectra. One might simply average as in Eq. (8.11) (which is what we will discuss in Section 8.3.3), or one might improve the estimate by composing a weighted average (which gives better results as you will see in Sections 8.3.4 and 8.3.5).

8.3.3 An Example

Let's consider a concrete example of the use of the multitaper technique and analyze a signal that is generated using an example of an autoregressive process (AR) process $x(t)$ with four parameters, as described by Percival and Walden (1993) (their Eq. (46a)):

$$x(t) = ax(t-1) + bx(t-2) + cx(t-3) + dx(t-4) + \varepsilon(t) \tag{8.15}$$

$$\text{With constants}: \quad a = 2.7607, \quad b = -3.8106, \quad c = 2.6535, \quad d = -0.9238$$

Here $\varepsilon(t)$ is a Gaussian white noise (GWN) process with zero mean and unit variance. Since we use this series to evaluate spectral analysis, we want to compute its power spectrum using the approach described in Section 18.4. If we z-transform (Chapter 12) the AR expression: the

transform of $x(t)$ is $X(z)$, and since the transform of the GWN is $E(z) = 1$, we get:

$$X(z) = \frac{E(z)}{1 - az^{-1} - bz^{-2} - cz^{-3} - dz^{-4}} = \frac{1}{1 - az^{-1} - bz^{-2} - cz^{-3} - dz^{-4}}$$

The Fourier transform $X(j\omega)$ can now be obtained by substitution $z^{-1} = e^{-j\omega\Delta t}$ and the unscaled power spectrum by computing XX^*. We will use this theoretical power spectrum as our gold standard for comparison with our multitaper estimation.

The following is part of a MATLAB® routine (pr8_3.m) that computes and plots the spectrum in (dB) versus a frequency scale between 0 and 0.5 (as in Percival and Walden (1993) and in Figs. 8.5 and 8.6).

```
% constants of AR(4) in Eq. (8.15)
a=2.7607;
b=-3.8106;
c=2.6535;
d=-.9238;

dt=1;              % time step set to 1
w=0:.01:pi/dt;     % scale for frequency
f=w./(2*pi/dt);    % frequency scaled between 0 and 0.5 (for plot)
z=exp(-j*w*dt);    % z^(-1)

X=1./(1-a*z-b*z.^2-c*z.^3-d*z.^4);      % Fourier transform
PX=X.*conj(X);                          % Power
LPX=10*log10(PX);                       % Power in dB

figure;plot(f,LPX)      % Compare theoretical spectrum Figs. 8.5 and 8.6
```

The procedure we follow for this multitaper analysis is depicted in Figs. 8.5 and 8.6. In this example we demonstrate that the number of tapers used provides a compromise between reduction of leakage and variance. In this example $NW = 4$, thus we would consider up to seven $(2NW - 1)$ eigenvectors. Fig. 8.5 depicts the properties of the individual tapers (the eighth one is also shown), including their spectra and the effects of their application in the time and frequency domains. In Fig. 8.6, the effects of using these tapers in spectral averages are shown. As can be seen, the first and best taper reduces leakage almost to zero (almost no energy in the side lobes) but produces a single spectral estimate associated with rather large variance. Adding the result from a second taper

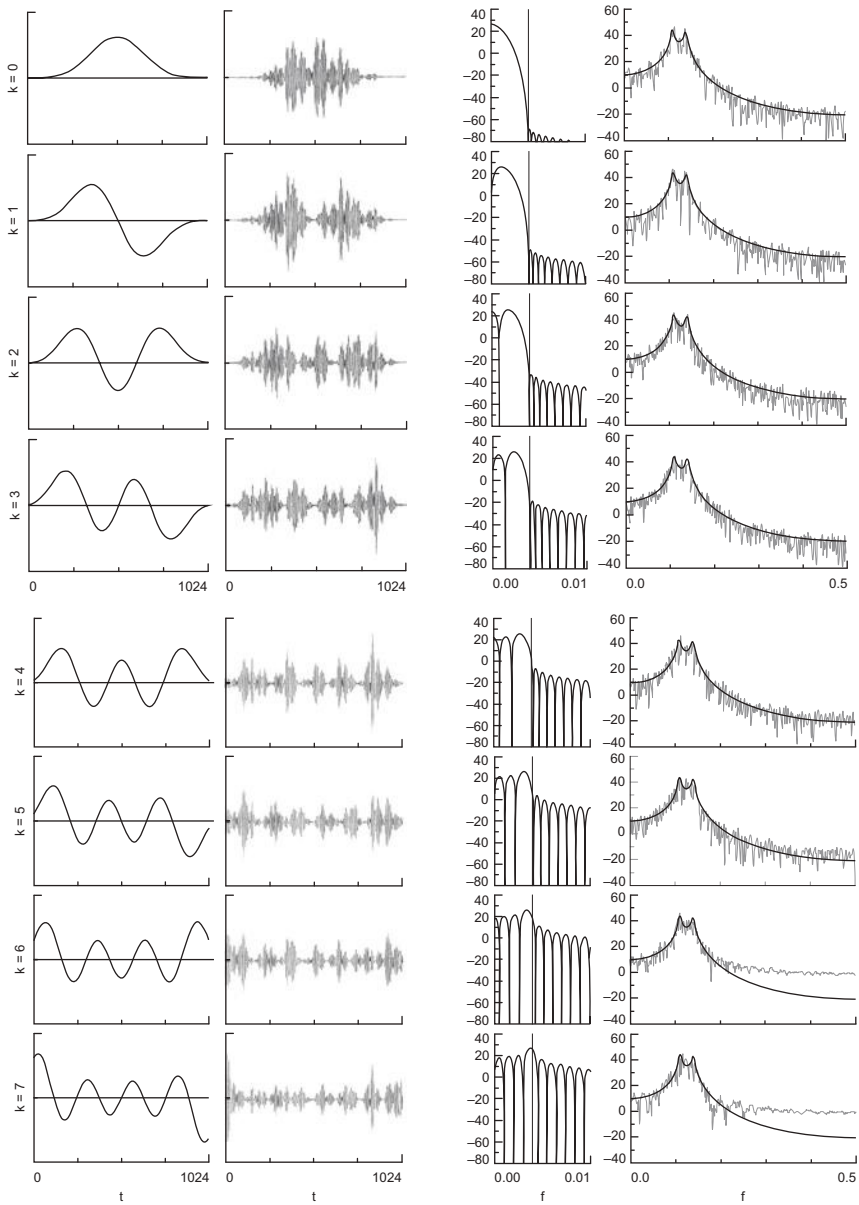


FIGURE 8.5 Example of the properties of the individual tapers ($NW = 4$). The left column in this set of plots shows eight discrete prolate spheroidal tapers. The $2NW$, eight tapers ($k = 1-8$) are depicted in subsequent rows. The panels in the second column are the products of the time series computed with Eq. (8.15) multiplied with these tapers. The third column shows the spectra of the tapers. The right column depicts the estimate of the spectra of the windowed time series; the *black line* is the theoretical spectrum and the *gray noisy lines* are the estimates obtained with the windowed signals. Obviously the energy of the side lobes and the associated leakage (especially in the range from 0.3 to 0.5) increases with k. *This figure is a combination of Figs. 336–339 in Percival, D.B., Walden, A.T., 1993. Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques. Cambridge University Press, Cambridge, UK.*

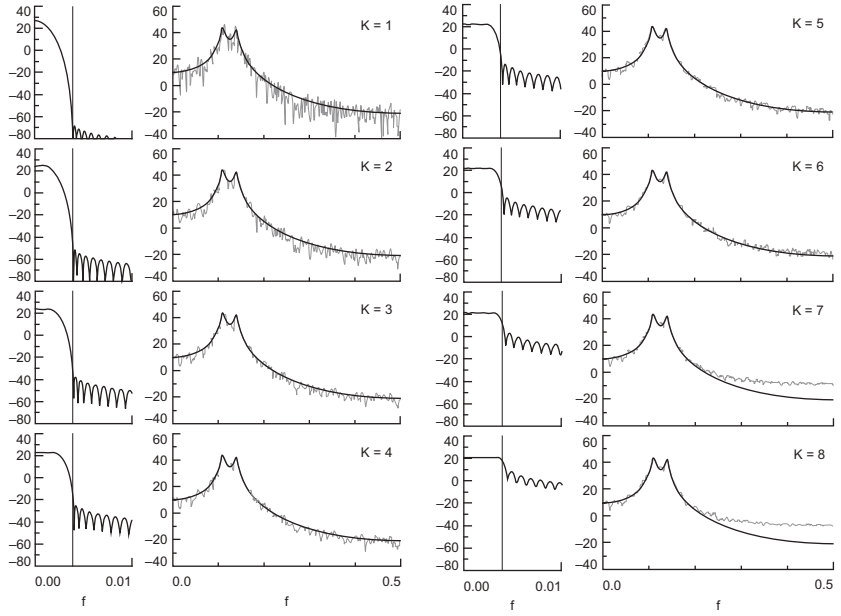


FIGURE 8.6 Averaged effects of the application of tapers. Spectral estimates of the time series of Eq. (8.15) by using the same eight tapers shown in Fig. 8.5. In this case the left and third columns show the spectra of the combined tapers. Note that the side lobes significantly increase with k. The second and right columns show the (averaged) multitaper estimates (*gray, noisy lines*) superimposed on the theoretical spectrum (*black lines*). With increasing k, the bias increases (due to the energy in the side lobes and associated leakage), especially in the range 0.3–0.5, but the variance (i.e. the noise in the spectrum) diminishes. In this example there seems to be a good compromise between reduced variance and leakage at $k = 5$. *This figure is a combination of Figs. 340 and 341 in Percival, D.B., Walden, A.T., 1993. Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques. Cambridge University Press, Cambridge, UK.*

adds a bit of side lobes (and thus leakage) but averaging it with the first one reduces variance because now the spectrum is averaged from two results. By including the result from each additional taper in the average estimate, there is both a bit of an increase of leakage and a reduction of variability due to the averaging procedure. In this example there seems to be a reasonable optimum at five tapers. At eight tapers the window contains so much energy in the side lobes that it causes a large amount of leakage and thus a strong bias in the spectral estimate. The take-home message of this example is that including the results obtained with the first $2NW - 1$ tapers in the averaged spectral estimate may still cause an unacceptable level of leakage! This problem is further discussed and addressed in Section 8.3.4.

8.3.4 Multitaper Spectrum With Optimized Weights

The example plots in Figs. 8.5 and 8.6 show that the use of $2NW - 1$ tapers, in this case seven, produces too much bias and that less tapers (in this case five tapers) give a better result. Using fewer tapers however, leads to a suboptimal reduction of the variance since we average over less spectra. One solution to this problem is to further optimize the weights of the tapers to minimize leakage (bias), so that all tapers can be used to reduce variance of the estimated spectrum. In short we want to investigate the use of a weighted average as an alternative to a simple average as in the previous example in Section 8.3.3. As described by multiple authors (e.g., Percival and Walden, 1993; Prieto et al., 2007), weighting of the result by each taper may be performed by their eigenvalues, or one might construct an optimized data adaptive procedure. The following presents these procedures without proof but with some background on how they are accomplished.

In the previous implementation of the multitaper approach all eigenvectors were normalized to unity. Associated with each eigenvector, we also have its eigenvalue. Without further proof, one could imagine that this eigenvalue could be employed as a weight when computing the averaged power spectrum estimate $\overline{S}(f)$. Eq. (8.11) is then modified by weighting the individual contributions of each tapered estimate $\widehat{S}_k(f)$ by its associated eigenvalue λ_k:

$$\overline{S}(f) = \frac{1}{K}\sum_{k=1}^{K}\frac{\widehat{S}_k(f)}{\lambda_k} \tag{8.16}$$

As it appears, the approach of weighting the estimates by their eigenvalues may still not give the best estimates. Using a regularization approach, Thomson (1982) developed a data adaptive method to weigh the individual spectral estimates that contribute to the average. This leads to the following expression for the spectral estimate:

$$\overline{S}(f) = \frac{\sum_{k=1}^{K} d_k^2 \widehat{S}_k(f)}{\sum_{k=1}^{K} d_k^2}, \tag{8.17a}$$

in which the weights d_k are frequency-dependent and computed by using:

$$d_k(f) = \frac{\sqrt{\lambda_k}S(f)}{\lambda_k S(f) - (1-\lambda_k)\sigma^2}, \tag{8.17b}$$

with σ^2 the variance of the time domain signal $x(t)$. However, the big unknown in the expression for d_k is the spectrum $S(f)$! Often the first two

eigenspectra $\widehat{S}_k(f)$ (for $k = 0$ and 1), provide an initial estimate for $S(f)$ and the adaptive weights are then found iteratively (for further details, see Thomson, 1982; Percival and Walden, 1993).

8.3.5 An Example in MATLAB®

The approach to compute a multitaper estimate of the spectrum is implemented in the MATLAB® pmtm command. In MATLAB® type: help pmtm to get the following description of the procedure.

```
pmtm   Power Spectral Density (PSD) estimate via the Thomson multitaper
  method (MTM).
  Pxx = pmtm(X) returns the PSD of a discrete-time signal vector X in
  the vector Pxx. Pxx is the distribution of power per unit frequency.
  The frequency is expressed in units of radians/sample. pmtm uses a
  default FFT length equal to the greater of 256 and the next power of
  2 greater than the length of X. The FFT length determines the length
  of Pxx.

  For real signals, pmtm returns the one-sided PSD by default; for
  complex signals, it returns the two-sided PSD. Note that a one-sided
  PSD contains the total power of the input signal.

  Pxx = pmtm(X,NW) specifies NW as the "time-bandwidth product" for the
  discrete prolate spheroidal sequences (or Slepian sequences) used as
  data windows. Typical choices for NW are 2, 5/2, 3, 7/2, or 4. If
  empty or omitted, NW defaults to 4. By default, pmtm drops the last
  taper because its corresponding eigenvalue is significantly smaller
  than 1. Therefore, The number of tapers used to form Pxx is 2*NW-1.

  Pxx = pmtm(X,NW,NFFT) specifies the FFT length used to calculate
  the PSD estimates. For real X, Pxx has length (NFFT/2+1) if NFFT is
  even, and (NFFT+1)/2 if NFFT is odd. For complex X, Pxx always has
  length NFFT. If empty, NFFT defaults to the greater of 256
  and the next power of 2 greater than the length of X.

  [Pxx,W] = pmtm(...) returns the vector of normalized angular
  frequencies, W, at which the PSD is estimated. W has units of
  radians/sample. For real signals, W spans the interval [0,Pi] when
  NFFT is even and [0,Pi) when NFFT is odd. For complex signals, W
  always spans the interval [0,2*Pi).

  [Pxx,W] = pmtm(X,NW,W) where W is a vector of normalized
  frequencies (with 2 or more elements) computes the PSD at
```

those frequencies using the Goertzel algorithm. In this case a two sided PSD is returned. The specified frequencies in W are rounded to the nearest DFT bin commensurate with the signal's resolution.

[Pxx,F] = pmtm(…,Fs) specifies a sampling frequency Fs in Hz and returns the power spectral density in units of power per Hz. F is a vector of frequencies, in Hz, at which the PSD is estimated. For real signals, F spans the interval [0,Fs/2] when NFFT is even and [0,Fs/2) when NFFT is odd. For complex signals, F always spans the interval [0,Fs). If Fs is empty, [], the sampling frequency defaults to 1 Hz.

[Pxx,F] = pmtm(X,NW,F,Fs) where F is a vector of frequencies in Hz (with 2 or more elements) computes the PSD at those frequencies using the Goertzel algorithm. In this case a two sided PSD is returned. The specified frequencies in F are rounded to the nearest DFT bin commensurate with the signal's resolution.

[Pxx,F] = pmtm(…,Fs,method) uses the algorithm specified in method for combining the individual spectral estimates:
'adapt' - Thomson's adaptive non-linear combination (default).
'unity' - linear combination with unity weights.
'eigen' - linear combination with eigenvalue weights.

[Pxx,Pxxc,F] = pmtm(…,Fs,method) returns the 95% confidence interval Pxxc for Pxx.

[Pxx,Pxxc,F] = pmtm(…,Fs,method,P) where P is a scalar between 0 and 1, returns the P*100% confidence interval for Pxx. Confidence intervals are computed using a chi-squared approach. Pxxc(:,1) is the lower bound of the confidence interval, Pxxc(:,2) is the upper bound. If left empty or omitted, P defaults to .95.

[Pxx,Pxxc,F] = pmtm(X,E,V,NFFT,Fs,method,P) is the PSD estimate, confidence interval, and frequency vector from the data tapers in E and their concentrations V. Type HELP DPSS for a description of the matrix E and the vector V. By default, pmtm drops the last eigenvector because its corresponding eigenvalue is significantly smaller than 1.

```
[Pxx,Pxxc,F] = pmtm(X,DPSS_PARAMS,NFFT,Fs,method,P) uses the cell
array DPSS_PARAMS containing the input arguments to DPSS (listed in
order, but excluding the first argument) to compute the data tapers.
For example, pmtm(x,{3.5,'trace'},512,1000) calculates the prolate
spheroidal sequences for NW=3.5, NFFT=512, and Fs=1000, and displays
the method that DPSS uses for this calculation. Type HELP DPSS for
other options.

[…] = pmtm(…,'DropLastTaper',DROPFLAG) specifies whether pmtm
should drop the last taper/eigenvector during the calculation. DROPFLAG
can be one of the following values: [ {true} | false ].
   true - the last taper/eigenvector is dropped
   false - the last taper/eigenvector is preserved

[…] = pmtm(…,'twosided') returns a two-sided PSD of a real signal
X. In this case, Pxx will have length NFFT and will be computed over
the interval [0,2*Pi) if Fs is not specified and over the interval
[0,Fs) if Fs is specified. Alternatively, the string 'twosided' can be
replaced with the string 'onesided' for a real signal X. This would
result in the default behavior.

The string input arguments may be placed in any position in the input
argument list after the second input argument, unless E and V are
specified, in which case the strings may be placed in any position
after the third input argument.

pmtm(…) with no output arguments plots the PSD in the current figure
window, with confidence intervals.

EXAMPLE:
   Fs = 1000;   t = 0:1/Fs:.3;
   x = cos(2*pi*t*200)+randn(size(t));    % A cosine of 200Hz plus noise
   pmtm(x,3.5,[],Fs);                     % Uses the default NFFT.
```

Fig. 8.7 depicts how the multitaper computation (using the default data adaptive method) applied to the AR(4) time series of Eq. (8.15) work in MATLAB®. It also shows how its result compares to the standard periodogram.

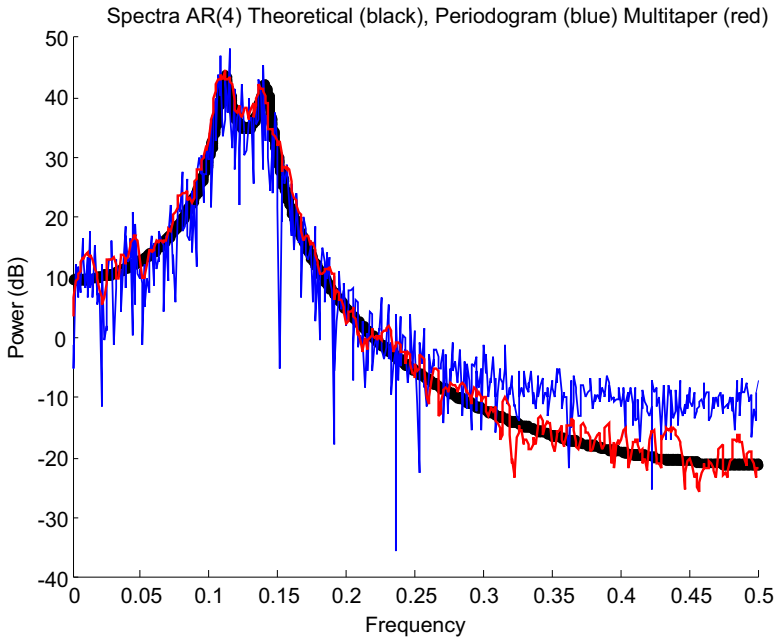


FIGURE 8.7 Comparison of the multitaper and boxcar methods. Comparison of the theoretical spectrum (black) of the AR(4) time series and the periodogram (blue) and the multitaper spectrum (red) using the pmtm command. Note that the multitaper spectrum, as compared to the periodogram, has less bias across the whole bandwidth. The variance at the lower frequencies is also less in the multitaper spectrum but not so much at the higher frequencies in this example. This figure was made with MATLAB® script pr8_3.m. Note that if you use this MATLAB® script, results can vary due to the random generation of the time series!

The results in Fig. 8.7 were obtained with the following MATLAB® script pr8_3.m

```
% pr8_3.m
% ARspectrum.m

clear;
close all;

% constants of AR(4) in Eq. (8.15)
a=2.7607;
b=-3.8106;
c=2.6535;
d=-.9238;
```

```
dt=1;                % time step set to 1
w=0:.001:pi/dt;      % scale for frequency
f=w./(2*pi/dt);      % frequency scaled between 0 and 0.5 (for plot)
z=exp(-j*w*dt);      % z^(-1)

X=1./(1-a*z-b*z.^2-c*z.^3-d*z.^4);      % Fourier transform
PX=X.*conj(X);                          % Power
LPX=10*log10(PX);                       % Power in dB

figure;plot(f,LPX)      % Compare theoretical spectrum Figs. 8.5 and 8.6
title('Spectrum AR(4)');
xlabel('Frequency');
ylabel('Power (dB)');

% create a time series
x(1:4)=randn(1,4);      % set initial values
for i=5:1028;           % create time series in loop
    x(i)=a*x(i-1)+b*x(i-2)+c*x(i-3)+d*x(i-4)+randn(1);
end;
x=x(5:1028);            % remove the initial values

figure;plot(x);
title('Instance of the AR(4) time series');
xlabel('Time');
ylabel('amplitude');

% Use the matlab pmtm function
NW=4;                       % set NW to 4 as in the example in Figs. 8.5 &
                            % 8.6
[Pxx,W] = pmtm(x,NW);
F=W./(2*pi/dt);             % frequency scaled between 0 and 0.5 (for plot)
LPxx=10*log10(Pxx*NW);      % compute in dB and multiply by NW to scale as
                            % LPX

% The standard Periodogram
Y=fft(x);
Pyy=Y.*conj(Y)/length(x);
LPyy=10*log10(Pyy);

% The following produces Fig. 8.7
% Note that results across trials may differ due to randomness
```

```
figure;hold;
plot(f,LPX,'k.')
plot(F,LPyy(1:length(F)))
plot(F,LPxx,'r')
title('Spectra AR(4) Theoretical(black), Periodogram(blue) Multitaper(red)');
xlabel('Frequency');
ylabel('Power (dB)');
```

APPENDIX 8.1

In the case of the standard power spectrum we have $S = XX^*/N$ (Eq. 7.1). The scaling by $1/N$ ensures that Parseval's conservation of energy theorem is satisfied. This theorem states that the sum of squares in the time domain and the sum of all elements in the power spectrum are equal (see Chapter 7: Table 7.1 and Appendix 7.1). In the case of Lomb's algorithm we compute the sum of squares for each frequency by using the expression in Eq. (8.8). Our expectation is therefore that Lomb's spectrum will also satisfy Parseval's theorem. However, there is a slight difference! In the standard Fourier transform we deal with negative frequencies and the positive and negative frequencies each contain half the energy. Basically, this is due to the fact that the Fourier transform is based on the complex Fourier series which includes negative frequencies. In contrast, if we compute the Lomb spectrum up to the Nyquist frequency only, we have all energy in the positive frequencies and therefore its values are twice as large as those in the standard power spectrum. An example for a single frequency is shown in Fig. A8.1-1. This figure is based on a

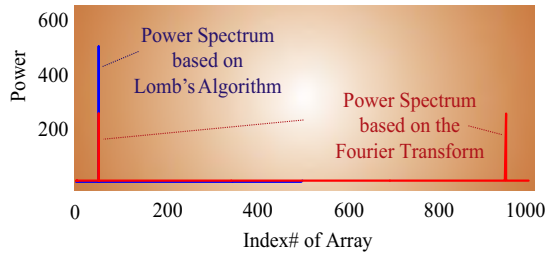


FIGURE A8.1-1 Spectral analysis of a 1-s epoch of a 50-Hz signal sampled at 1000 Hz. The graph depicts the superimposed results from a standard power spectrum (red) based on the Fourier transform and the power spectrum obtained with Lomb's algorithm (dark blue). Note that the total energy in both cases is identical. This figure can be created with pr8_2.m.

standard powers spectrum and Lomb spectrum computed for the same input, a sine wave of 50 Hz.

Thus, if we want the Lomb spectrum to have the same amplitudes as the standard power spectrum we need to divide by two. Further, if we want to normalize by the total power, we can divide by the variance σ^2. This normalization by $2\sigma^2$ is exactly the normalization commonly applied for Lomb's spectrum (see Eq. 8.9).

APPENDIX 8.2

Here we elaborate on the steps between Eqs. (8.13) and (8.14). For further details, see also Percival and Walden (1993).

For convenience we repeat Eq. (8.13):

$$\lambda(N,W) = \frac{\int_{-W}^{W} |A(f)|^2 df}{\int_{-1/2}^{1/2} |A(f)|^2 df} \tag{8.13}$$

We need to maximize this expression for λ. First we limit the signal in the time domain over the interval $[0, N-1]$ so that we can use the following expression for the discrete Fourier transform:

$$A(f) = \sum_{t=0}^{N-1} a(t) e^{-2\pi j f t}.$$

Using Parseval's theorem, we can rewrite the denominator in Eq. (8.13) as:

$$\sum_{t=0}^{N-1} a(t)^2. \tag{A8.2-1}$$

The numerator in Eq. (8.13) can also be written as (note the use of dummy variables t and t' and complex conjugate signs, *):

$$\left[\int_{-W}^{W} A^*(f) df\right]\left[\int_{-W}^{W} A(f) df\right] = \left[\int_{-W}^{W} \sum_{t=0}^{N-1} a^*(t) e^{2\pi j f t} df\right] \times \left[\int_{-W}^{W} \sum_{t'=0}^{N-1} a(t') e^{-2\pi j f t'} df\right]$$

Now we change the order of summation and integration and combine the integrals over f:

$$\left[\sum_{t=0}^{N-1} a^*(t) \int_{-W}^{W} e^{2\pi jft} df\right]\left[\sum_{t'=0}^{N-1} a(t') \int_{-W}^{W} e^{-2\pi jft'} df\right]$$
$$= \sum_{t=0}^{N-1}\sum_{t'=0}^{N-1} a^*(t) \underbrace{\left[\int_{-W}^{W} e^{2\pi j(t-t')} df\right]}_{\frac{1}{2\pi j(t-t')} e^{2\pi jf(t-t')}\Big|_{-W}^{W}} a(t') \tag{A8.2-2}$$

Note that in the above expression, each of the two integrals left of the equal sign is the inverse Fourier transform of a rectangular window in the frequency domain (this window ranges from $-W$ to W, e.g., Fig. 8.4B). The resulting integral in Eq. (A8.2-2) generates the covariance of this rectangular window:

$$\frac{1}{2\pi j(t-t')} e^{2\pi jf(t-t')}\Big|_{-W}^{W} = \frac{1}{2\pi j(t-t')} \underbrace{\left[e^{2\pi jW(t-t')} - e^{-2\pi jW(t-t')}\right]}_{2j \sin 2\pi W(t-t')}$$
$$= \frac{\sin 2\pi W(t-t')}{\pi(t-t')} \tag{A8.2-3}$$

Combining Eqs. (A8.2-1), (A8.2-2) and (A8.2-3) to rewrite Eq. (8.13), we get:

$$\lambda(N,W) = \frac{\sum_{t=0}^{N-1}\sum_{t'=0}^{N-1} a^*(t)\left[\frac{\sin 2\pi W(t-t')}{\pi(t-t')}\right] a(t')}{\sum_{t=0}^{N-1} a(t)^2}$$

This can be compactly rewritten in matrix/vector notation as:

$$\lambda = \frac{a \cdot D \cdot a}{a \cdot a}. \tag{A8.2-4}$$

Here we simplified the notation for λ and a, while matrix D is given by:

$$D(t,t') = \frac{\sin 2\pi W(t-t')}{\pi(t-t')}. \tag{A8.2-5}$$

In order to reduce leakage we need to maximize the expression for λ in Eq. (A8.2-4). We accomplish this by setting the derivative of this expression with respect to a equal to zero. If we simplify Eq. (A8.2-4) to the fraction u/v, the expression to solve is $(u'v - uv')/v^2 = 0$, which is the

same as solving $u'v - uv' = 0$. This can be rewritten as $u'/v' = u/v = \lambda$, or $u' - \lambda v' = 0$. Now we recall from Eq. (A8.2-4) that $u = a \cdot D \cdot a$ and that thus $u' = 2D \cdot a$. Note that this is the case because matrix D is symmetric and not a function of a. The denominator $v = a \cdot a$, and its derivative $v' = 2a$. Now we combine the expression for u' and v' with $u' - \lambda v' = 0$, we divide by 2 and get Eq. (8.14), the well-known eigenvalue expression:

$$D \cdot a - \lambda a = 0 \quad \text{or} \quad D \cdot a = \lambda a \tag{8.14}$$

EXERCISES

8.1 Use MATLAB® to create an unevenly sampled time series x with a 50-Hz sinusoidal signal + noise. The mean sample frequency is ~500 samples/s.
   a. Compute the Lomb spectrum of x.
   b. Now use the MATLAB® interp1 command and create an evenly sampled time series y from x.
   c. Compute the Lomb spectrum and standard Fourier-based spectrum of y.
   d. Interpret your findings of the spectral analyses.

8.2 Modify the pr8_3 script to investigate the effect of the "adapt," "unity," and "eigen" options in the MATLAB® pmtm command. What do these options mean and what can you conclude?

8.3 What is the variance of the averaged spectral estimate $\overline{S}(f)$ if we have K spectral estimates $\widehat{S}_k(f)$ each with a variance of σ^2 (see Eq. 8.11)?

References

Lomb, N.R., 1976. Least-squares frequency analysis of unequally spaced data. Astrophys. Space Sci. 39, 447–462.

Park, J., Lindberg, C.R., Vernon, F.L., 1987. Multitaper spectral analysis of high-frequency seismograms. J. Geophys. Res. 92, 12675–12684.

Percival, D.B., Walden, A.T., 1993. Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques. Cambridge University Press, Cambridge, UK.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 2007. Numerical Recipes in C, third ed. Cambridge University Press, Cambridge, MA.

Prieto, G.A., Parker, R.L., Thomson, D.J., Vernon, F.L., Graham, R.L., 2007. Reducing the bias of multitaper spectrum estimates. Geophys. J. Int. 171, 1269–1281.

Scargle, J.D., 1982. Studies in astronomical time series analysis. II. Statistical aspects of spectral analysis of unevenly spaced data. Astrophys. J. 263, 835–853.
Slepian, D., 1978. Prolate spheroidal wave functions, Fourier analysis, and uncertainty V: the discrete case. Bell Syst. Tech. J. 57, 1371–1429.
Thomson, D.J., 1982. Spectrum estimation and harmonic analysis. Proc. IEEE 70, 1055–1096.
Van Drongelen, W., Williams, A.L., Lasky, R.E., 2009. Spectral analysis of time series of events: effect of respiration on heart rate in neonates. Physiol. Meas. 30, 43–61.

CHAPTER

9

Differential Equations: Introduction

9.1 MODELING DYNAMICS

When modeling some aspect of a neural system, we can represent static variables with **algebraic expressions**, e.g., the concentration c of some chemical in the brain is 10 units, $c = 10$. Of course, we can make these expressions a bit more complicated; for instance, the concentration could also depend on some combination of other substances a_1 and a_2: e.g., $c = 5a_1 + a_2 + 10$. It is important to realize that the variables do not change with time or space. For that reason the value of this type of equation is limited to situations where we study properties that remain constant over the range and duration of our interest. In contrast, when we model the **dynamics** of a concentration of a brain substance over time or space, we have to include the rate of change of the variable(s) in the equation. For example, if the speed of change of concentration c is known to be five units per unit time, we could model this with: $v = 5$ and $c = v \times t$, where v is speed of change and t is time. So, if we assume an initial condition $c = 1$ at $t = 0$, we find that at time $t = 12$ the concentration is $c = 1 + 5 \times 12 = 61$. We can recast this scenario into a different notation. We start by setting the change of variable c to an expression for velocity: $v = \Delta c/\Delta t$, where Δc is the change in concentration c, and Δt is the time interval required for this change. For example, let us measure c at two instances in time $t1$ and $t2$. Let us say that, at these times, we measured concentrations $c1$ and $c2$, respectively. Now we can compute the average speed v of change in concentration as $v = (c2 - c1)/(t2 - t1)$; here we substituted $\Delta c = c2 - c1$ and $\Delta t = t2 - t1$. If we make the intervals over which we determine change infinitesimally small, and we change the notation for change of concentration and time $\Delta \rightarrow d$, we will obtain the

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00009-6

expression for the instantaneous speed, e.g., $v = dc/dt = 5$. This expression

$$\dot{c} = dc/dt = 5 \tag{9.1}$$

is a **first-order ordinary differential equation** (1st-order ODE). It is first order because the derivative, using the notation $\dot{c}$ or dc/dt is a first-order derivative and it is ordinary since there are no **partial differentials** such as $\partial c/\partial t$. Thus a differential equation is an expression that includes the rate of change of a variable rather than only its state—this enables us to quantify the dynamics (e.g., change over time) of a variable in a quantitative fashion.

Of course, if a state can change over time to quantify the velocity (rate of state change), the velocity itself can also change over time. For example, if you drive a car, your speed isn't constant because you can accelerate or decelerate. Similarly, the rate of change of some substances in the brain can also change over time: the speed of built up can accelerate or decelerate. The rate of change of a state was given by $v = dc/dt$. Similarly, we can use dv/dt for the rate of change in speed, i.e., for acceleration a. Now, combining $v = dc/dt$ and $a = dv/dt$, we find that we can also get the rate of change in the speed by taking the derivative from the state c twice! The notation for taking the derivative twice, i.e., a second-order derivative, is $a = d^2c/dt^2$. When such a second-order term is included in a differential equation, we have a second-order ODE. Suppose we find some equation $5 \times c + 6 \times v + 3 \times a = 0$, relating state of substance c, speed of change of c given by v, and rate of change of the speed as acceleration a, then we can rewrite this equation in the calculus notation as:

$$5c + 6dc/dt + 3d^2c/dt^2 = 0. \tag{9.2}$$

When written in this form it can be seen that this expression is a second-order ordinary differential equation (second-order ODE) because the highest derivative in the expression d^2c/dt^2 is a second-order one, and it is an ODE because the expression does not include any partial derivative.

The take-home message here is that an ODE is simply an equation that quantifies dynamics, i.e., the rate of change of some variable (e.g., with respect to time or space) rather than just its state. The following introduction in differential equations isn't an extensive treatment of the topic, but rather a review of frequently used strategies to solve them. The material in this chapter can be considered an introduction to the applications described in chapters on models and filters. We will discuss ODEs by using examples, and we won't worry too much about "technical details," such as the existence and uniqueness of solutions, but rather deal with these aspects as they occur. Readers familiar with ODEs can skip both this

chapter and Chapter 10. Readers that need more details on this topic are referred to Boas (1966), Jordan and Smith (1997), and Polking et al. (2006).

9.2 HOW TO FORMULATE AN ORDINARY DIFFERENTIAL EQUATION

The example above shows that differential equations may also include a term with the state of the variable itself, in this case $5c$. An intuitive example of an ODE that includes the state of the variable is the expression for the growth of a bacterial culture. Since each bacterium can divide to produce offspring, the rate of growth g in a bacterial population is proportional to its population size S, i.e., $g = kS$, here k is some value that reflects that proportionality. Since the growth g is the rate of change of S, i.e., dS/dt, we may also write

$$dS/dt = kS. \tag{9.3}$$

This expression is a first-order ODE, since the highest order of derivation is first order, i.e., dS/dt. Why is it advantageous to write $dS/dt = kS$ and not $g = kS$ (which seems much simpler)? The reason is that the former notation allows one to use the methods of calculus developed for solving differential equations.

Another example of the formulation of an ODE can be illustrated with a (bio)chemical reaction, e.g., a reversible reaction in which substances A and B produce substance C and vice versa:

$$A + B \underset{k_2}{\overset{k_1}{\rightleftharpoons}} C$$

Here, the symbols k_1 and k_2 represent the constants that govern the rate of the reactions, and A, B, and C denote the concentrations of these substances. Now suppose we are interested in finding the expression for the dynamics for compound C. In the reaction above we observe that production of C is proportional with A and B, and depends on rate constant k_1. In the reversed reaction we see that C's disappearance is proportional with itself and occurs at a rate governed by constant k_2. This leads to an expression for the rate of change for C that includes two terms, i.e., one for the production and one for the disappearance of C:

$$dC/dt = k_1AB - k_2C \tag{9.4}$$

This example shows how to develop the ODE from the chemical interaction. Although we won't go into the details of this example here,

this formalism can then be used to analyze the dynamics of the reaction. One side comment, it can easily be seen in Eq. (9.4) that the equilibrium state, $dC/dt = 0$ is determined by the rate constants and occurs at $k_1/k_2 = C/AB$. Note that this approach quantifies the average dynamics of a mixture with large amounts of molecules of types A, B, and C. It will not enable us to determine what occurs with individual molecules in the mixture.

Another alternative to formulate ODEs is to transform an expression from a symbolic domain (i.e., Laplace or Fourier transform domains). If you are not familiar with these transforms, you may skip the following part and return to it after reviewing Chapters 12 and 13. To illustrate the formulation of an ODE using a transformation of a symbolic equation, we consider a linear time-invariant system for which we know its unit impulse response: $h(t) = e^{-t}$ (the example we also discuss in Appendix 13.1). Now we can compute the system's response $y(t)$ to some input $x(t)$ as the convolution: $y(t) = h(t) \otimes x(t)$. In the transformed Laplace domain this convolution becomes a product: $Y(s) = H(s)X(s)$. Assuming that all system inputs and outputs are set to zero for $t < 0$, we find in Table A12.1-1 that the Laplace transform of e^{-t}, the unit impulse response, is $\frac{1}{s+1}$. Thus we may write the transformed convolution in this example as:

$$Y(s) = \frac{1}{s+1} X(s) \rightarrow sY(s) + Y(s) = X(s).$$

Now recall that $sY(s)$ is the Laplace transform of the derivative of $y(t)$. This allows us to inverse transform the expression from the symbolic s-domain into the time domain, giving us the ODE that governs the system's dynamics:

$$\dot{y}(t) + y(t) = x(t),$$

which is a linear first-order ODE with a so-called forcing term $x(t)$.

A similar example of transformation of a symbolic equation into a partial differential equation in the spatiotemporal domain is discussed in Chapter 30. In Section 30.5, we obtain a partial differential equation describing a neural field from an expression in the form of a spatiotemporal Fourier transformed convolution.

9.3 SOLVING FIRST- AND SECOND-ORDER ORDINARY DIFFERENTIAL EQUATIONS

Examples of first- and second-order ODEs were given in the introduction above. Here we will discuss how to solve this type of equation. In

this section we will discuss the type of ODE that is called a **linear homogeneous equation with constant coefficients**. Alternatively, one can use the term **unforced** instead of homogeneous. These ODEs have the form:

$$\begin{array}{l} \dot{S} + cS = 0 \quad \text{for the first order case, or} \\ \ddot{S} + b\dot{S} + cS = 0 \quad \text{for the second order case, with } b, c, \text{ constants} \end{array} \tag{9.5}$$

Here we used $\dot{S}$ and $\ddot{S}$ to denote first and second order time derivatives dS/dt and d^2S/dt^2, respectively.

As a start, let us go back to the example of bacterial growth in Eq. (9.3). Note that this ODE has indeed the form of a firstorder case in Eq. (9.5). This type of expression is relatively straightforward since we can **separate the variables** S and t. Using this property, the expression $dS/dt = kS$ can be rewritten as $dS/S = kdt$. The latter equation consists of two expressions left and right of the equal sign that each can be integrated:

$$\int dS/S = \int kdt. \tag{9.6}$$

Here it is fairly obvious that a limited range for S applies. Since we have S in the denominator, the equation is only valid for nonzero values of S, a reasonable condition since a bacterial population cannot grow if there are no bacteria. In addition, since the number of bacteria cannot be negative, we must have $S > 0$. Hence we can write the solution of the integrals in Eq. (9.6):

$$\ln(S) = kt + C \tag{9.7}$$

Taking the exponential at both sides of Eq. (9.7) gives us:

$$S(t) = e^{kt+C} = e^C e^{kt} = Ae^{kt}. \tag{9.8}$$

Here we explicitly wrote S as a function of t and replaced the constant e^C by A. We are almost done; the only task remaining is to determine constant A, and here we will use the so-called **initial condition** to find the value of A. Say that at time set to zero ($t = 0$) we found the value for S to be S_0. We can substitute this in Eq. (9.8) to find the value for the constant A. Recall that for $t = 0$ we have $e^{k0} = 1$, thus using the initial condition, we find: $S(0) = Ae^{k0} = A = S_0$. Now we plug this into Eq. (9.8) and obtain the solution to our first-order ODE:

$$S(t) = S_0 e^{kt}. \tag{9.9}$$

Instead of the initial condition, we could have used the value of S at any other point in time to determine A. Using the initial condition is just convenient because at $t = 0$ we have unity for the exponential in the

equation, i.e., $e^{k0} = 1$. For this reason, one might simply set $t = 0$ at the point in time that we have measured the value of S.

As we will see later, so-called two-dimensional dynamical systems satisfy a second-order ODE. Here, the term dimension should not necessarily be taken too literally. For example, the neuronal network activity equations formulated by Wilson and Cowan (1972), described in Chapter 30 are an example of a nonlinear version of such a two-dimensional system: in their model, the two dimensions are the activities of two neuronal populations, i.e., an excitatory and an inhibitory population. Here we will start with a general example of a linear second-order ODE:

$$a\ddot{S} + b\dot{S} + cS = 0.$$

Note that, without loss of generality, we can simplify this expression by dividing by a, i.e., making the coefficient of $\ddot{S}$ equal to 1. This is what we will do in the following, and we will simply write the generic expression for a second-order ODE as

$$\ddot{S} + b\dot{S} + cS = 0. \tag{9.10}$$

Note that this is according to the general expression for the second-order case in Eq. (9.5). Now let us assume there is a solution in the form of $S = e^{kt}$, a solution of the same form as the one we found for the first-order ODE. If we compute the derivatives $\dot{S}$ and $\ddot{S}$ of the assumed solution e^{kt}, we get ke^{kt} and k^2e^{kt}. Now plug these in the second-order ODE in Eq. (9.10):

$$k^2e^{kt} + bke^{kt} + ce^{kt} = 0.$$

This can be simplified to:

$$\left(k^2 + bk + c\right)e^{kt} = 0,$$

and since e^{kt} is not generally zero, we can satisfy this equation by solving the standard quadratic equation

$$k^2 + bk + c = 0, \tag{9.11}$$

which is called the **characteristic equation** of the ODE. The solution of this equation is:

$$k_{1,2} = \left(-b \pm \sqrt{b^2 - 4c}\right)\Big/2. \tag{9.12}$$

Here we have, depending on the value of the term under the square root, three possible scenarios:

1. two real solutions for k if $\left(b^2 - 4c\right) > 0$,
2. two complex solutions $\left(b^2 - 4c\right) < 0$, or
3. just one real solution $\left(b^2 - 4c\right) = 0$.

The first two cases (1) and (2) are fairly straightforward if we realize that any scaled superposition of solutions of the equation is a solution of a linear system (see also Chapter 13). Since we found two solutions $e^{k_1 t}$ and $e^{k_2 t}$, we find the general solution here as the sum of the two scaled solutions:

$$S = Ae^{k_1 t} + Be^{k_2 t}, \tag{9.13a}$$

with A and B scaling constants to be determined, often from known conditions at $t = 0$ and $t = \infty$ (see Exercise 9.2). Note that in condition (2), if the solutions for k_1 and k_2 are complex, the expression in Eq. (9.13a) represents an oscillation (this is straightforward to see if you recall Euler's formula: $e^{j\omega t} = \cos \omega t + j \sin \omega t$).

The third case $k_1 = k_2$ only provides one solution which, if applied to the same approach as above, would lead to $S = Ae^{k_1 t} + Be^{k_1 t} = Ce^{k_1 t}$, which is clearly not sufficient to describe two-dimensional dynamics. This dilemma is resolved by changing the constant into a function of time:

$$S = C(t)e^{k_1 t} \tag{9.13b}$$

In models of neuronal networks with explicit synaptic function, this type of expression is often referred to as the alpha function and is used to model the postsynaptic current (Chapter 30).

One important take-home message when comparing the solutions of the first- and second-order ODEs is that they each have a range of potential behaviors. The first-order ODE has a simple exponential $S_0 e^{kt}$ as its solution, and depending on the value of rate constant k, the value of the variable S can increase, decrease, or remain the same. The second-order ODE's solution allows for the same behavior, but there is additional behavior possible, namely oscillation. Although this directly follows from the solutions from these ODEs, it can be seen directly in the so-called phase space representation that we will introduce in Chapter 10.

9.4 ORDINARY DIFFERENTIAL EQUATIONS WITH A FORCING TERM

The ODEs in the previous section contain a number of terms, potentially including the dynamic variable and its derivative(s). Here we will extend this type of example to a case where there is an additional term that does not include the dynamic variable or its derivative. This results in an equation of the form:

$$\dot{S} + cS = f(t) \text{ for the first order case, or}$$
$$\ddot{S} + b\dot{S} + cS = f(t) \text{ for the second order case,}$$
$$\text{with } b, c, \text{ constants, and } f(t), \text{ forcing term}$$

In physical systems the additional term $f(t)$ often represents an external force and therefore it is called the **forcing term**. Due to this term, in contrast to the expressions discussed in the previous section, this type of ODE is an **inhomogeneous** equation.

To show the approach in solving an ODE with an additional term, we will discuss a simplified linear model for open and closed ion channels (see Fig. 1.1); in this case there is a finite amount of ion channels available and we use O and C as fractions of open and closed gates:

$$C \underset{h}{\overset{m}{\rightleftarrows}} O$$

Let us assume that, as in the famous Hodgkin and Huxley (1952) formalism, the tendency for gates to open or close depends on the membrane potential. Thus, at a given membrane potential E, the rate constants governing the opening and closing dynamics are m_E, h_E. Here, we simplify the notation by omitting the subscripts E. The dynamics for the fraction of open gates at the given membrane potential is governed by:

$$dO/dt = mC - hO. \tag{9.15}$$

Since O and C are fractions of the total number of available gates, we have $C = 1 - O$, this allows us to write Eq. (9.15) as $dO/dt = m(1 - O) - hO$, which can be simplified to:

$$dO/dt = \dot{O} = m - (m + h)O. \tag{9.16}$$

In this case our "forcing term" is not really originating from an external force, it is a constant (m) originating from the $1 - O$ term. The general approach in finding the solution in the case of a forcing term is to solve the equation without the forcing term first; this will give the intrinsic, unforced solution of the dynamics. Then one solves the full equation, usually by assuming that there is a **particular** solution in the same form as the forcing term. Because we assumed m being a constant in this example, we also assume that there is a solution in the form of a constant. After we found both solutions for the unforced and forced cases, we again apply the rule that any scaled superposition of solutions is a solution of a linear system (Chapter 13). Following this recipe, we combine the solution of the unforced ODE with the particular solution and obtain the **general solution** of the forced ODE. This solution will again contain constants that must be found from boundary values. Applying this approach, we

know that the solution of the unforced ODE is in the same form as Eq. (9.8), i.e.,

$$O(t) = Ae^{-(m+h)t}. \tag{9.17}$$

Since the forcing term in this example is a constant, we assume that the particular solution O_P is also a constant. Since the derivative of a constant vanishes, we know that $dO_P/dt = 0$. Plugging that into Eq. (9.16), we find that:

$$O_P(t) = \frac{m}{m+h}, \text{ or a scaled version of this result.} \tag{9.18}$$

The combination of Eqs. (9.17) and (9.18) leads to our general solution O_G:

$$O_G(t) = Ae^{-(m+h)t} + B\frac{m}{m+h}. \tag{9.19}$$

Now let us assume that at $t = 0$ all channels are closed, i.e., $O_G(0) = 0$. Further, let us assume that there is equilibrium at $t = \infty$, i.e., $\dot{O}_G = 0$ (note the dot!) at $t = \infty$. The latter condition means that the exponential term at $t = \infty$ vanishes (since m and h are both positive rate constants), and we can determine from Eq. (9.16) that the equilibrium state corresponds to $\frac{m}{m+h}$, i.e., we find that $B = 1$. Going back to the initial state at $t = 0$, we have the exponential term $e^{-(m+h)t} = 1$, thus we find that $A = -\frac{m}{m+h}$. Plugging the results for A and B into Eq. (9.19) gives:

$$O_G(t) = \frac{m}{m+h}\left(1 - e^{-(m+h)t}\right). \tag{9.20}$$

Now let's investigate a second-order case. For a second-order ODE with a forcing term, we follow exactly the same procedure as we did above for the first-order one. Since we established that the potential dynamics include oscillation, let's apply an oscillatory input to the dynamics in Eq. (9.10) with amplitude α and radial frequency ω_F, so that we get:

$$\ddot{S} + b\dot{S} + cS = \alpha \cos \omega_F t. \tag{9.21}$$

We have already determined the solution for the homogeneous equation (Eq. 9.13). Now, as we did for the first-order ODE, we assume there is a particular solution in the form of the forcing term. To make this straightforward, as you will notice below, we will replace the forcing term $\alpha \cos \omega_F t$ by the more general expression for an oscillation $\alpha e^{j\omega_F t}$, and we then assume that the particular solution is in the form $de^{j\omega_F t}$, where d is a constant. Thus now we have:

$$\ddot{S} + b\dot{S} + cS = \alpha e^{j\omega_F t}. \tag{9.22}$$

In this version, S is a complex variable. Strictly speaking it is a bit sloppy to use S for both the complex variable and its real part, but we will leave it so since there is no serious ambiguity in the following.

Note: We use here the imaginary exponential term $e^{j\omega_F t}$, representing a combination of sine and cosine waves—recall Euler's formula: $e^{j\omega_F t} = \cos \omega_F t + j \sin \omega_F t$. Alternatively, we could also employ $a \cos \omega_F t + b \sin \omega_F t$ for the particular solution and determine a and b by following the same procedure as below.

Differentiating the complex exponential expression for the particular solution with respect to time, we find $j\omega_F d e^{j\omega_F t}$ and $-\omega_F^2 d e^{j\omega_F t}$ for the first and second derivatives. We now plug this into the expression left of the equal sign of Eq. (9.22), and we get:

$$\ddot{S} + b\dot{S} + cS = -\omega_F^2 d e^{j\omega_F t} + j\omega_F b d e^{j\omega_F t} + c d e^{j\omega_F t} = \left[-\omega_F^2 + j\omega_F b + c \right] d e^{j\omega_F t}$$

Since this must be equal to the right-hand side of the expression in Eq. (9.22), we now have determined that $\left[-\omega_F^2 + j\omega_F b + c \right] d = \alpha$, which enables us to express how unknown constant d depends on the parameters of the ODE:

$$d = \frac{\alpha}{\left[-\omega_F^2 + j\omega_F b + c \right]}.$$

Obviously, d is a complex number. Thus the complex solution for Eq. (9.22) is:

$$S = \frac{\alpha}{\left[-\omega_F^2 + j\omega_F b + c \right]} e^{j\omega_F t}. \tag{9.23}$$

Recall that the cosine expression is the real component of the Euler formula. Thus, since the forcing term in Eq. (9.21) is a cosine, we only need the real part of the solution in the above expression. The real part can be obtained by writing the constant factor and the exponential in Eq. (9.23) in their real and imaginary components:

$$S = \left[\frac{\alpha\left(c - {\omega_F}^2\right)}{\left(c - {\omega_F}^2\right)^2 + b^2 {\omega_F}^2} - j \frac{\alpha b \omega_F}{\left(c - {\omega_F}^2\right)^2 + b^2 {\omega_F}^2} \right] [\cos \omega_F t + j \sin \omega_F t].$$

Now it is straightforward to determine the real part of S, i.e., the particular solution of Eq. (9.21):

$$S = \frac{\alpha\left(c - {\omega_F}^2\right)}{\left(c - {\omega_F}^2\right)^2 + b^2 {\omega_F}^2} \cos \omega_F t + \frac{\alpha b \omega_F}{\left(c - {\omega_F}^2\right)^2 + b^2 {\omega_F}^2} \sin \omega_F t. \tag{9.24}$$

Assuming we found a solution for the homogeneous equation as in Eq. (9.13a), our general solution becomes:

$$S = Ae^{k_1 t} + Be^{k_2 t} + \frac{\alpha\left(c - {\omega_F}^2\right)}{\left(c - {\omega_F}^2\right)^2 + b^2{\omega_F}^2}\cos\,\omega_F t + \frac{\alpha b\omega_F}{\left(c - {\omega_F}^2\right)^2 + b^2{\omega_F}^2}\sin\,\omega_F t. \tag{9.25}$$

The algebra may seem a bit cumbersome, but it is all straightforward algebra. You can check this, when using numbers for the parameters.

9.5 REPRESENTATION OF HIGHER-ORDER ORDINARY DIFFERENTIAL EQUATIONS AS A SET OF FIRST-ORDER ONES

The highest-order ODE we discussed thus far is the second-order one, such as in Eq. (9.10). Although one always makes a serious attempt to keep things as simple as possible, if we deal with real dynamical systems, higher-order ODEs are not exceptional. One extremely useful technique to solve these cases is that an nth order system can be decomposed into a set of n first-order ODEs. Here we repeat the second-order ODE of Eq. (9.10) for convenience:

$$\ddot{S} + b\dot{S} + cS = 0.$$

If we now define $S_1 = S$ and $S_2 = \dot{S} = \dot{S}_1$, we have $\dot{S}_2 = \ddot{S}_1$. Now we can rewrite the second-order ODE as two first-order differential equations:

$$\begin{aligned} \dot{S}_1 &= S_2 \\ \dot{S}_2 &= -cS_1 - bS_2, \end{aligned} \tag{9.26}$$

or if we use the boldface notation $\boldsymbol{S}$ to denote the vector $[\,S_1 \quad S_2\,]^T$, where T indicates the transpose of the row vector, we can simplify the two expressions into a single vector expression:

$$\begin{aligned} &\dot{\boldsymbol{S}} = \begin{bmatrix} 0 & 1 \\ -c & -b \end{bmatrix}\boldsymbol{S}, \text{ or for the general case,} \\ &\dot{\boldsymbol{S}} = A\boldsymbol{S}, \text{ using } A \text{ to denote a matrix.} \end{aligned} \tag{9.27}$$

Even for a second-order ODE this change of notation is attractive because we can now use straightforward linear algebra to obtain the solution. Analogous to what we did for solving the ODEs above, we assume there is a solution in the form

$$e^{\lambda t}\boldsymbol{V}. \tag{9.28}$$

In this expression λ is the rate of change and note that $\boldsymbol{V}$ is a vector. If we differentiate the expression for the solution, we get $\lambda e^{\lambda t}\boldsymbol{V}$. Now we substitute the assumed solution and its derivative in the generic form of Eq. (9.27):

$$\lambda e^{\lambda t}\boldsymbol{V} = e^{\lambda t}A\boldsymbol{V}.$$

Dividing by the (nonzero) exponential gives:

$$A\boldsymbol{V} = \lambda\boldsymbol{V}. \tag{9.29}$$

As you may recall, this is a well-known equation in linear algebra, where $\boldsymbol{V}$ is an eigenvector of matrix A and λ the associated eigenvalue. Following the standard procedure of linear algebra, the eigenvalues corresponding to Eq. (9.29) can be found by solving the **characteristic equation**:

$$|A - \lambda\boldsymbol{I}| = 0, \tag{9.30}$$

with $|\ldots|$ denoting the determinant and $\boldsymbol{I}$ the identity matrix (in this example a 2×2 matrix).

Let's consider an **example**.

We start using the matrix $A = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}$.

First let us determine the solutions for which $\dot{S}_1$ and $\dot{S}_2$ are zero. This gives the solutions $S_2 = -S_1/3$ and $S_2 = -S_1$, the so-called null-clines for S_1 and S_2, respectively.

Next, using Eq. (9.30), the eigenvalues can be found from $\begin{vmatrix} 1-\lambda & 3 \\ 2 & 2-\lambda \end{vmatrix} = 0$. If we expand this expression and factorize the result, we have $(\lambda - 4)(\lambda + 1) = 0$. This gives us the two solutions for the eigenvalues: 4 and -1. Finding the eigenvectors associated with these eigenvalues produces eigenvectors $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 3 \\ -2 \end{bmatrix}$ respectively (Fig. 9.1A). Thus the solution associated with the eigenvector $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ grows since it is governed by a positive exponential (i.e., a positive eigenvalue, $\lambda = 4$) and the other associated with $\begin{bmatrix} 3 \\ -2 \end{bmatrix}$ decays because the eigenvalue is negative ($\lambda = -1$).

We have used the example of the second order ODE: $\ddot{S} + b\dot{S} + cS = 0$ (Eq. 9.10) and rewrote this expression as $\dot{\boldsymbol{S}} = A\boldsymbol{S}$ (Eq. 9.27). Recall that matrix $A = \begin{bmatrix} 0 & 1 \\ -c & -b \end{bmatrix}$, and the solutions $k_{1,2}$ were found from the

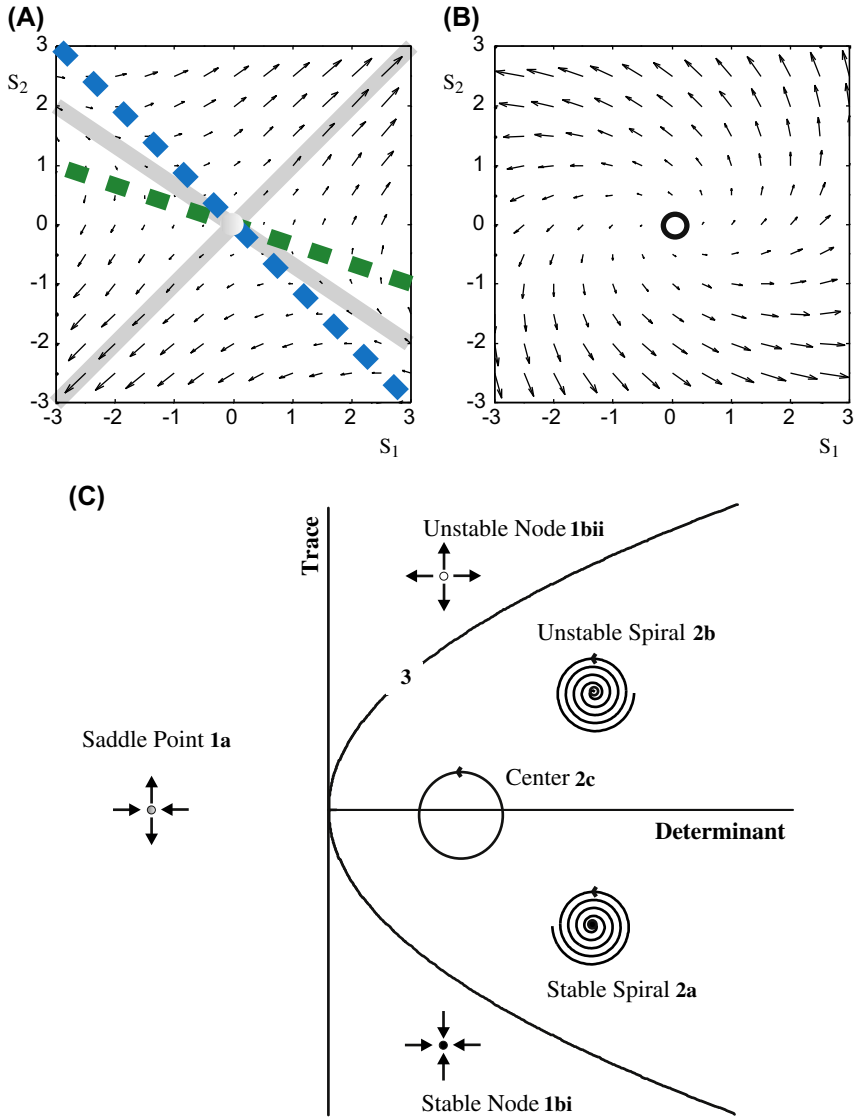


FIGURE 9.1 (A, B) Examples of phase plane plots for the second-order ordinary differential equation (ODE): $\ddot{S} + b\dot{S} + cS = 0$. Note that in both plots S is defined as S1 and $\dot{S}$ as S2. In panel A, we show a saddle point with one stable and one unstable manifold (the *gray lines*). The *colored lines* are the null clines for S1 (green) and S2 (blue). This case represents scenario 1a in Eq. (9.32). In panel B we show a scenario with unstable (growing) oscillatory behavior. This case represents scenario 2b in Eq. (9.32). (C) Diagram of the type and stability of the dynamics for the second-order ODE in Eq. (9.27) as a function of the values for the trace T (ordinate) and determinant D (abscissa) of matrix A. For negative values of the determinant we have saddle points. For a positive determinant, the value for the trace determines the solution's stability. The horizontally oriented parabola (governed by $T^2 - 4D$) separates nodes from spirals and centers. The labels 1a–3 correspond to the scenarios summarized in Eq. (9.32). On this parabola (label 3), the eigenvalues are identical in all directions, giving rise to so-called star (independent pair of eigenvectors) or degenerate (only one eigenvector) nodes.

characteristic equation (see also Eqs. 9.11 and 9.12): $k_{1,2} = \left(-b \pm \sqrt{b^2 - 4c}\right)/2$. Now note that the trace T and determinant D of matrix A are $-b$ and c, respectively. This allows us to write the expressions for the solutions for $\lambda_{1,2}$ as:

$$\lambda_{1,2} = \left(T \pm \sqrt{T^2 - 4D}\right)/2. \tag{9.31}$$

We can use Eq. (9.31) to show that trace and determinant for matrices A and $\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$ are the same, i.e.: $T = \lambda_1 + \lambda_2; \quad D = \lambda_1 \lambda_2$. Thus we have two solutions for the characteristic equation, i.e., the **eigenvalues** $\lambda_{1,2}$. Note that we adopt the commonly used symbol for the eigenvalue by substituting $\lambda_{1,2}$ for $k_{1,2}$. Since we have a linear differential equation, the solution to the ODE is a linear combination determined by the two eigenvalues (see Eq. 9.13a). Now that we have changed the notation for the second-order ODE, we can also change the notation for the scenarios listed in Eq. (9.12). If we now consider Eq. (9.31) and analyze the effects of T and D in addition to the value of the term under the square root, $T^2 - 4D$, we can further subdivide these scenarios and we get the following possibilities:

1. two real solutions for λ if
$$\left(T^2 - 4D\right) > 0, \tag{9.32}$$
   a. if $D < 0$, one solution is positive and one is negative,
   b. if $D > 0$, both solutions are either positive or negative,
      i. if $T < 0$ both solutions are negative,
      ii. if $T > 0$ both solutions are positive;
2. two complex conjugate solutions for λ if $\left(T^2 - 4D\right) < 0$,
   a. if $T < 0$, the real part of the solution is negative,
   b. if $T > 0$, the real part of the solution is positive,
   c. if $T = 0$, the solutions are imaginary;
3. just one real solution for λ if $\left(T^2 - 4D\right) = 0$,
   a. with two independent eigenvectors,
   b. with one eigenvector.

In the following we show how you can examine these alternative possibilities in MATLAB®. In this example we take the generic notation of a second-order ODE: $\dot{\boldsymbol{S}} = A\boldsymbol{S}$ (Eq. 9.27). Next, we use S1 and S2 as the components of the vector $\boldsymbol{S}$, and dS1 and dS2 for the components of its derivative $\dot{\boldsymbol{S}}$. In the MATLAB® command sequence below, the command meshgrid defines the range and precision with which we plot the result. The coefficients used to compute dS1 and dS2 below are example entries of matrix A, here we use $\begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}$. The quiver command plots the values of

derivatives dS1 and dS2 as a vector field in the S1−S2 plane—recall that this is the phase plane since $S_1 = S$ and $S_2 = \dot{S}$.

To obtain the plot in Fig. 9.1A type in the following at the MATLAB® command prompt (or run the script pr9_1.m).

```
[S1,S2] = meshgrid(-3:.5:3,-3:.5:3);
dS1=1*S1+3*S2;
dS2=2*S1+2*S2;
figure;
quiver(S1,S2,dS1,dS2,'k');
axis([-3 3 -3 3])
axis('square')
```

If we categorize this example using its matrix $A = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}$, compute the values for $T = 3$, $D = -4$ and $T^2 - 4D = 25$, using Eq. (9.32), we note that this represents **case 1a**: two real solutions for the characteristic equation, one positive and one negative. This is easily confirmed in the result depicted in Fig. 9.1A, we have a so-called saddle point with one stable manifold associated with a negative solution and one unstable one associated by the positive solution. Using the same MATLAB® procedure, it is easy to explore some other values for A, for example try $A = \begin{bmatrix} 2 & -4 \\ 2 & 3 \end{bmatrix}$ ***by replacing the lines for dS1 and dS2 in the above sequence of commands by the following ones:***

```
dS1=2*S1-4*S2;
dS2=2*S1+3*S2;
```

This will generate a result as depicted in Fig. 9.1B. Interestingly, in this example the vector arrangement in the S1−S2 plane follows a different pattern, i.e., spiral trajectories moving away from the origin. If we categorize this example using its matrix $A = \begin{bmatrix} 2 & -4 \\ 2 & 3 \end{bmatrix}$, compute the values for $T = 5$, $D = 14$ and $T^2 - 4D = -31$, and using Eq. (9.32), we note that this represents **case 2b**: two complex conjugate solutions with positive real values. When we relate this to the phase space plot in Fig 9.1B, the complex solutions govern the oscillatory behavior and the positive real values cause the oscillations to grow, away from the fixed point, a so-called unstable spiral, also called a spiral source.

The two results we obtained above and that are depicted in Fig. 9.1A and B are two examples out of the possible behaviors generated by a second-order ODE, i.e., the dynamics one may observe in a 2-D phase plane. Depending on the values we pick for matrix A, we can get any of the behaviors we summarized in Eq. (9.32). These examples show that the values for trace (T) and determinant (D) can be used to diagnose the dynamical behavior of the ODE. Fig. 9.1C depicts the scenarios listed in Eq. (9.32) in a diagram of T and D. Each of the numbers in Fig. 9.1C corresponds to the numbering in Eq. (9.32).

Using the same numbers for the different scenarios as we used in Eq. (9.32) and Fig. 9.1C, depending on T and D, we find the following dynamics:

(1a)	a saddle point,
(1bi)	a stable node,
(1bii)	an unstable node,
(2a)	a stable spiral, also called a spiral sink,
(2b)	an unstable spiral, also called a spiral source,
(2c)	a borderline case, the center
(3a)	borderline cases, a star node if $\lambda \neq 0$, or no dynamics if $\lambda = 0$
(3b)	borderline case, a degenerate node.

You can create a phase plane plot of each case by changing variables dS1 and dS2 in the above MATLAB® example. For instance, a star node (case 3a) can be observed when using:

```
dS1=1*S1+0*S2;
dS2=0*S1+1*S2;
```

Note that this case is located on the horizontally oriented parabola in Fig. 9.1C: since $T = 2$ and $D = 1$, thus the condition $T^2 - 4D = 0$ is satisfied.

A degenerate node (case 3b) can be observed when using:

```
dS1=1*S1+1*S2;
dS2=0*S1+1*S2;
```

Note that also in this case $T = 2$ and $D = 1$, however, there is an additional nonzero off-diagonal element in matrix A now.

A center (case 2c) can be observed when using:

```
dS1=0*S1+1*S2;
dS2=-1*S1+0*S2;
```

Note that in this case $T = 0$ and $D = 1$, satisfying the condition for the center depicted in Fig. 9.1C.

9.6 TRANSFORMS TO SOLVE ORDINARY DIFFERENTIAL EQUATIONS

Another, often convenient, way to solve ODEs is to use transforms: the Fourier, Laplace, or z transform. The idea here is to transform the ODE from the time or spatial domain into another domain and obtain the solution in the transformed domain. Next, the solution is inverse transformed into the original (time or spatial) domain, and voilà there we have the solution (see Fig. 12.1). For details and several examples of this approach see Chapters 12 and 16.

EXERCISES

9.1 Solve the ODE $dS/dt = k$, using an initial condition of four arbitrary units (AU), i.e., $S(0) = 4$ AU.

9.2 We have the characteristic equation $k^2 + 3k - 10 = 0$
- a. What is the corresponding ODE? (assume the dynamic variable is S)
- b. Solve the characteristic equation.
- c. What is the solution of the ODE?
- d. Determine the constants in c. by using the following conditions: at $t = 0$ we have $S = 10$ AU, and at $t = \infty$ we will have $S = 0$ AU.
- e. Write the ODE as two first-order ODEs
- f. In what part of the plot of Fig. 9.1C is the solution located? Explain what this means for the dynamics.

9.3 Take the resting potential values for m and h in the Hodgkin and Huxley formalism and plot Eq. (9.20). Repeat this for the values at a zero membrane potential. Interpret your result.

9.4 Find the particular solution of the inhomogeneous equation $\ddot{S} + 2\dot{S} - 3S = 5\cos 3\,t$ from scratch (i.e., don't use the Eq. (9.24) but show all the steps).
Use both the complex exponential and the real approach.
Answer: $-\frac{1}{3}\cos 3\,t + \frac{1}{6}\sin 3\,t$

References

Boas, M.L., 1966. Mathematical Methods in the Physical Sciences, second ed. John Wiley & Sons, New York.
A textbook on basic mathematical techniques.

Hodgkin, A.L., Huxley, A.F., 1952. A quantitative description of membrane current and its application to conduction and excitation in the nerve. J. Physiol. 117, 500–544.
A seminal paper describing the Hodgkin and Huxley equations.

Jordan, D.W., Smith, P., 1997. Mathematical Techniques. Oxford University Press, Oxford.
An overview of calculus, meeting requirements in engineering or physics. The book covers differentiation, integration, matrix/vector algebra, differential equations, transforms, discrete mathematics, and statistics.

Polking, J., Bogges, A., Arnold, D., 2006. Differential Equations With Boundary Value Problems. Pearson Prentice Hall, Upper Saddle River, NJ.
A textbook on basic techniques for solving differential equations.

Wilson, H.R., Cowan, J.D., 1972. Excitatory and inhibitory interactions in localized populations of model neurons. Biophys. J. 12, 1–24.
A seminal paper describing neuronal population dynamics.

CHAPTER

10

Differential Equations: Phase Space and Numerical Solutions

10.1 GRAPHICAL REPRESENTATION OF FLOW AND PHASE SPACE

In Chapter 9 we explored ODEs and how to solve them. In the examples we discussed in that chapter we were able to obtain solutions in a relatively straightforward manner. There are many types of ODEs however, where such a solution is not easily obtained, or using an analytical approach may even be impossible. The only approach in such a case is to compute a solution that satisfies the dynamics governed by the ODE. Especially in the lower-order cases, a good start to obtain insight into the dynamics determined by the ODE is to depict the flow of the dynamics graphically. If we deal with a first-order case, where the dynamics of variable S depends only on itself (i.e., $\dot{S} = S$), we can depict the direction of the flow $\dot{S}$ against the time axis (Fig. 10.1). The short lines in Fig. 10.1 show how the dynamics flow. Here it can be seen that there is an infinite amount of solutions that would satisfy the dynamics.

The following MATLAB® script, pr10_1.m was used to create Fig. 10.1

```
% pr10_1.m
% Solver Demo
% for dS/dt=kS

clear
close all
k=1;
Dt=.1;
figure;hold
```

```
for t=-3:.5:3;
   for S=-5:.25:10;
      DS=Dt*k*S;
      line([t-Dt t+Dt],[S-DS S+DS])
   end;
end;
```

In order to determine a single solution amongst all the possible ones, one needs to know at least one position. For example, if we know that $S = 1$ at $t = 0$, the initial condition indicated by the asterisk in Fig. 10.1, we can determine the solution that passes through that condition (green/red line). This shows graphically why expressions such as those in Eqs. (9.8), (9.13a) and (9.13b) that solve an ODE, require some boundary condition.

Another common procedure to explore dynamics graphically is the so-called phase space representation. Within this approach the dynamics of the variable(s) is shown as a vector field. In one-dimensional dynamics the phase space is a phase line, in a two-dimensional case the phase space is a phase plane. Obviously the phase space representation only works for analysis of low-dimensional dynamics because, above three dimensions, the whole phase space cannot be depicted. Using the same dynamics as in

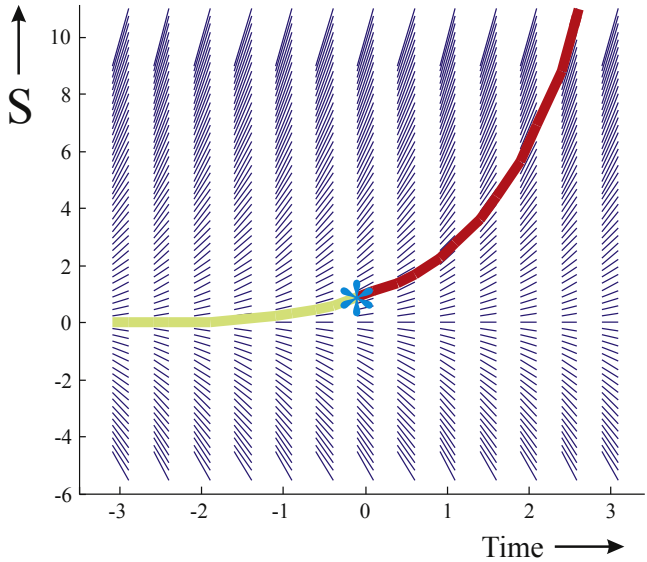


FIGURE 10.1 Dynamics of the ODE $\dot{S} = S$. The flow of S plotted against the time axis. The initial condition is depicted by the *blue asterisk* and the (hand-drawn) associated solution for negative and positive times by the *green and red lines*. The flow lines in this figure were obtained with MATLAB® script pr10_1.m.

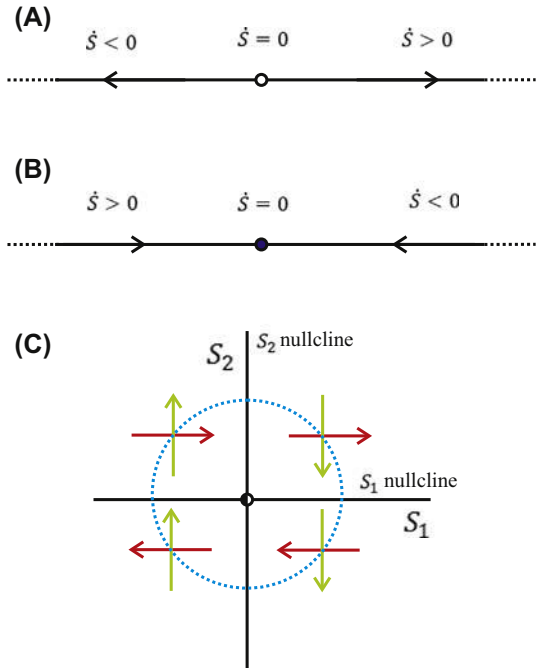


FIGURE 10.2 Phase space plots for a one- and two-dimensional case. The phase lines in panels (A) and (B) show the flow of $\dot{S} = S$ and $\dot{S} = -S$, respectively. The phase plane in panel (C) shows the flow for $\ddot{S} + S = 0$. Further explanation is given in the text.

Fig. 10.1, we can construct a phase line (Fig. 10.2A), where $\dot{S} = 0$ in the origin, while $\dot{S} > 0$ and $\dot{S} < 0$ right and left of the origin. This shows that there is an unstable equilibrium at the origin, $S = 0$ because as indicated by the arrows/vectors, any small perturbation away from the origin will lead to movement towards right or left on the line. This equilibrium can be compared to the unstable equilibrium of a marble sitting on the top of a hill; any small perturbation in the marble's position will cause it to roll downhill. If we redo the analysis for the case $\dot{S} = -S$, we get a fairly similar result (Fig. 10.2B). However, in this case the equilibrium at the origin is stable; any perturbation away from the origin results in a movement back to the origin. This is similar to a marble located at the bottom of a bowl where every displacement of the marble results in a movement back to the bottom of the bowl. It is common practice to indicate unstable fixed points with an open circle and stable ones with a filled circle (Fig. 10.2A and B).

Now let us move on to a two-dimensional case and use Eq. (9.10) $\ddot{S} + b\dot{S} + cS = 0$ again. If we simplify this further by setting $b = 0$ and $c = 1$. The physicist immediately recognizes the expression as one without a damping term, i.e., $b = 0$. We can decompose the second-order ODE into

a pair of really simple first-order ODEs (see also Chapter 9, Section 9.5, for a MATLAB® simulation in the phase plane):

$$\begin{aligned} \dot{S}_1 &= S_2 \\ \dot{S}_2 &= -S_1. \end{aligned} \tag{10.1}$$

Now we can depict the flow in the phase plane of S_1, S_2. The S_1 axis (where $S_2 = 0$) is precisely the border where $\dot{S}_1 = 0$ and where $\dot{S}_1 > 0$ and $\dot{S}_1 < 0$ above and below (red arrows, Fig. 10.2C). The line where $\dot{S}_1 = 0$ is also known as the S_1 **nullcline**; here it is a coincidence that the S_1 nullcline is also the S_1 axis. Similarly, in this example, the S_2 axis is also the S_2 **nullcline**. Only in this case, due to the minus sign, the direction of the flow of S_2 is positive for negative S_1 and vice versa (green arrows, Fig. 10.2C). Combining the flow for S_1, S_2 (red and green arrows, Fig. 10.2C), we obtain a clockwise movement depicted by the blue circle in Fig. 10.2C. Thus, in this case we observe oscillatory behavior associated with a so-called center.

Although the $\dot{S} - S$ relationship, as in Fig. 10.2 is commonly employed to visualize how dynamics evolve, there is an alternative method using the so-called potential. In this context, potential V is a function of location S. If we use an example in which the dynamics is governed by:

$$\dot{S} = f(S) = 25 - S^2, \tag{10.2}$$

the potential V is defined by:

$$\dot{S} = \frac{dS}{dt} = f(S) = -\frac{dV}{dS}. \tag{10.3}$$

From the above expressions we can derive two practical conclusions:

1. Applying the chain rule, we can write an expression for the change of the potential with respect to time: $\frac{dV}{dt} = \frac{dV}{dS}\frac{dS}{dt}$. Next, if we substitute the definition of the potential, $\frac{dS}{dt} = -\frac{dV}{dS}$, we get:

$$\frac{dV}{dt} = -\left(\frac{dV}{dS}\right)^2. \tag{10.4}$$

   Since $-\left(\frac{dV}{dS}\right)^2 \leq 0$, this finding shows that the potential must always decrease with time or remain zero.
2. Using the definition of $V(S)$, we see that the potential function can be computed by integration of $f(S)$ with respect to S. In our example, we get $V(S) = \frac{1}{3}S^3 - 25S + C$, where C is an integration constant.

For our 1-D example, we plotted both the potential function $V(S)$ and $\dot{S} = f(S)$ in Fig. 10.3. Now we can visualize the dynamics on the phase line by using our phase point approach indicated by the arrows on the phase line for S, or by the red particle that moves down on the potential

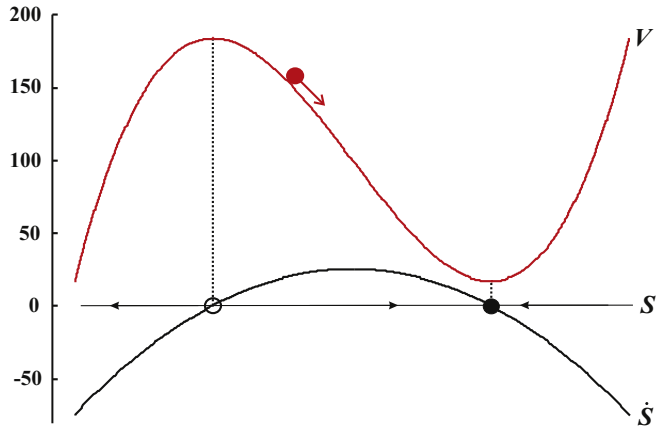


FIGURE 10.3 A plot of potential function V (red) and $\dot{S}$ (black) versus S for an example where the dynamics is governed by $\dot{S} = f(S) = 25 - S^2$. Both V and $\dot{S}$ can be used to evaluate the dynamics of the system on the phase line, and locate the stable and unstable fixed points indicated by the *filled and open dots*, respectively.

landscape. It should be noted here that we need to imagine that the potential landscape is covered with molasses, so that the movement of the red particle in Fig. 10.3 is extremely damped.

10.2 NUMERICAL SOLUTION OF AN ODE

In Chapter 9 we showed that in some cases an ODE can be solved analytically. Sometimes, this can be challenging or even impossible. In these cases, numerical solution of the dynamics is the answer. The most common approach for obtaining numerical solutions describing the dynamics of a system is to divide time into a **fixed time step**. The evolution of the system is then computed over this time step, allowing one to determine the state of the system at the end of the fixed time step. Then the numerical procedure is repeated for the next time step. The idea underlying this procedure can be visualized by the solution drawn in Fig. 10.1: one starts at some known condition (asterisk in Fig. 10.1) and follows the direction of the field (indicated by the lines) for a short time step, after reaching the end point of that time step, the procedure is repeated. In this case one might evolve the system in two directions: the past (green line, Fig. 10.1) and the future (red line, Fig. 10.1). Obviously, any numerical method will be associated with some error: one only approximates the dynamics when following the straight line over the time step, and (as in any numerical procedure) there will be rounding errors.

The first type of error depends on the size of the time step and especially in fast-evolving systems it may have a tendency to grow with time.

The simple fixed time step method is the so-called Euler's method. Let's go back to the example in Fig. 10.1 to demonstrate the principle: i.e., $\dot{S} = S$. Further, let's assume just as in Fig. 10.1 that the initial condition at $t = 0$, indicated by S_0 is 1. Using this initial value and using a fixed time step $\Delta t = 0.5$ s, we can compute that S after one time step denoted by S_1 is:

$$S_1 = S_0 + \dot{S}_0 \times \Delta t = 1 + 1 \times 0.5 = 1.5$$

We can repeat this procedure for the next time steps and we will get the following numerical outcomes for time steps 0–4 (i.e., $t = 0.0$, 0.5, 1.0, 1.5, 2.0):

$$S_0 = 1.0000$$
$$S_1 = 1.5000$$
$$S_2 = 2.2500$$
$$S_3 = 3.3750$$
$$S_4 = 5.0625$$

If we compute the values for $t = 0.0$, 0.5, 1.0, 1.5, 2.0 s directly from the analytical solution of the ODE, $S = e^t$, we would have gotten:

$$1.0000 \quad 1.6487 \quad 2.7183 \quad 4.4817 \quad 7.3891$$

Thus, the Euler's method generates an approximation that indeed becomes worse with time. In this example, our numerical approximation is systematically lower than the correct outcome, because we underestimate the slope of the graph governing the change across the time step, leading to an estimate below the correct value. Even worse, since we underestimate the outcome at the end of the time step, our mistake grows with each time step. One important improvement can be obtained by reduction of the time step, say using 0.01 s instead of 0.1 s (Exercise 10.2).

Another alternative to improve our prediction of the numerical approximation is to employ the improved Euler's method. Here we compute two slopes instead of one, and use the average of the two slopes to evolve the system over the time step. Applying this to the previous example we would compute for the first time step the slope at the start and the end of the time step.

One of the most commonly used numerical procedures is the Runge–Kutta method. As demonstrated in the MATLAB® script pr10_2.m, this algorithm employs a four-step approach to evaluate the dynamics over a fixed time step.

The following MATLAB® script, pr10_2.m summarizes the three different numerical procedures for the $\dot{S} = S$ dynamics.

```
% pr10_2.m
% Numerical Solution
% dS/dt=S

clear;
close all

dt=.5;
N=10;
t=0:dt:dt*N;
yA=exp(t);      % Analytical Solution, see Eq. (9.9)
S(1)=1;

% Euler's Method
for n=1:N
   s1=S(n);
   S(n+1)=S(n)+dt*s1;
end;
yE=S;           % Euler Solution

% Improved Euler's Method aka second-order Runge-Kutta
% employs two estimates of the slope s1 and s2
for n=1:N
   s1=S(n);
   s2=S(n)++dt*S(n);
   S(n+1)=S(n)+(dt/2)*(s1+s2);
end;
yiE=S;          % Improved Euler Solution

% Fourth-order Runge Kutta
% employs four estimates of the slope s1-s4
for n=1:N
   s1=S(n);
   s2=S(n)+(dt/2)*s1;
   s3=S(n)+(dt/2)*s2;
   s4=S(n)+dt*s3;
   S(n+1)=S(n)+(dt/6)*(s1+2*s2+2*s3+s4);
end;
yRK=S;          % Fourth-order Runge Kutta Solution

% Euler's Method @ 5 x smaller steps
dt=.5/5;
N=10*5;
```

```
ts=0:dt:dt*N;
for n=1:N
   s1=S(n);
   S(n+1)=S(n)+dt*s1;
end;
yEs=S;              % Euler Solution with smaller time step

% Plot Results
figure;hold;
plot(t,yA,'ko-')
plot(t,yE,'b.-')
plot(t,yiE,'r.-')
plot(t,yRK,'g.-')
plot(ts,yEs,'b*')

xlabel('time')
ylabel('S = exp (t)')
title('Black o-Analytical Outcome; Blue-Euler (* small steps); Red-Improved
Euler; Green-4th Order RK')

% Summarize and print the findings
Y=[yA;yE;yiE;yRK;yEs(1:5:N+1)]'
```

When again comparing the first four time steps across the different methods we can see the difference in accuracy (Table 10.1).

Without any special adaptation, numerical methods can applied to **nonlinear** ODEs that are difficult or impossible to solve. Let us look again at the example of a growth/birth process $\dot{S} = rS$, characterized by rate constant r. Now we add limiting resources and death to the dynamics of the population development. The result is that there is a target size, also

TABLE 10.1 Comparison of Numerical Solvers for $\dot{S} = S$

Time (s)	exp (*t*)	Euler's	Improved Euler's	Fourth-Order Runge–Kutta
0.0	1.0000	1.0000	1.0000	1.0000
0.5	1.6487	1.5000	1.6250	1.6484
1.0	2.7183	2.2500	2.6406	2.7173
1.5	4.4817	3.3750	4.2910	4.4794
2.0	7.3891	5.0625	6.9729	7.3840

known as **carrying capacity** K for the population. Below the target size, the population grows and above it the size decreases. This is described by the well-known nonlinear logistic equation, first constructed by Pierre-François Verhulst in the early 1800s (and rediscovered about a century later):

$$\dot{S} = rS\left(1 - \frac{S}{K}\right) \tag{10.5}$$

For convenience in the remainder of our analysis we will use $r = 1$. It can be seen in Eq. (10.5) that:

1. $\dot{S} = 0$ if $S = K$,
2. $\dot{S} > 0$ if $S < K$, and
3. $\dot{S} < 0$ if $S > K$.

Thus, at any size not equal to K, the system will change size to attain target size K.

The logistic equation is an example of a nonlinear ODE that can be solved analytically. We start by rewriting Eq. (10.5) as (note that we used $r = 1$):

$$\frac{KdS}{S(K-S)} = dt,$$

And we use partial fraction expansion to get:

$$\left(\frac{1}{S} + \frac{1}{K-S}\right)dS = dt.$$

Now we integrate the terms followed by some algebra:

$$\log|S| + \log|K-S| = t + C,$$

$$\log\left|\frac{S}{K-S}\right| = t + C,$$

$$\left|\frac{S}{K-S}\right| = e^{t+C} = e^t e^C.$$

Here e^C is a constant. If we now use an initial condition S_0 at $t = 0$, we find a value $\frac{S_0}{K-S_0}$ for the constant. Here and in the following we drop the absolute values since we allow the constant to be positive as well as negative. If we now solve for S and do a bit of algebra:

$$\frac{S}{K-S} = \frac{S_0}{K-S_0}e^t, \text{ and we find :}$$

$$S = \frac{KS_0}{S_0 + (K-S_0)e^{-t}}. \tag{10.6}$$

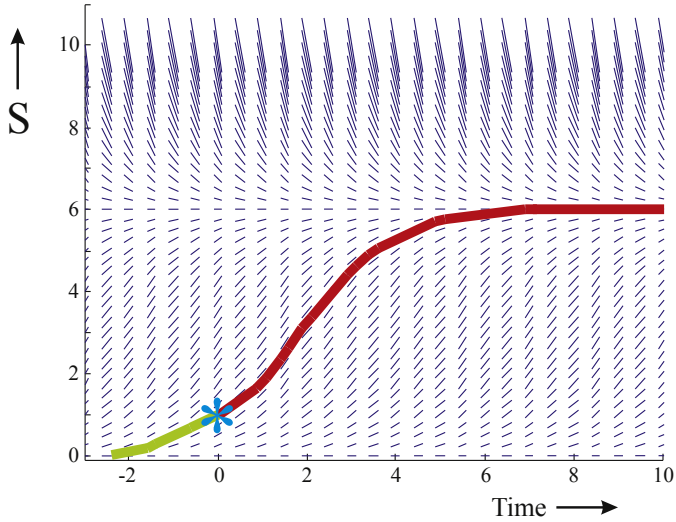


FIGURE 10.4 The logistic equation $\dot{S} = S\left(1 - \frac{S}{6}\right)$. The flow of S is plotted against the time axis. The initial condition is depicted by the *blue asterisk* and the (hand-drawn) associated solution for negative and positive times by the *green and red lines*. The flow lines in this figure were obtained with MATLAB® script pr10_3.m.

In Eq. (10.6) we can see that for $t \to \infty$ the size S becomes equal to the carrying capacity K. It is interesting that we did not have to go through all this algebra to find this result because we could directly see in the ODE (Eq. 10.5) that the ultimate equilibrium occurs at $S = K$. Another intuitive way to look at the dynamics governed by the logistic equation is to plot the directions in the same way we did in Fig. 10.1. The example depicted in Fig. 10.4 is computed for a carrying capacity of six. Using the same initial condition as in Fig. 10.1, we now see that the growth is not unlimited but stabilizes at $K = 6$.

10.3 PARTIAL DIFFERENTIAL EQUATIONS

Partial differential equations (PDEs) are used to describe the dynamics of a metric with respect to different variables. An obvious example is a description of spatiotemporal dynamics. For instance, a propagating brain wave is a potential field that changes with both time and location. The essence is that a PDE includes separate derivatives to describe dependence to time and location. Ordinary derivatives consider metrics that only depend on a single variable, e.g., with respect to time t or location x, reflected by notations dS/dt or dS/dx. In contrast, when metric

S depends on both time and location, the PDE employs partial derivatives. An example of a PDE is

$$\partial^2 S/\partial x^2 = K\,\partial S/\partial t. \tag{10.7}$$

This type of equation with S being a function of place x and time t plays a role in processes such as diffusion or conduction of heat. Sometimes these PDEs can be solved by transforming them into ODEs, e.g., if we assume that the dependence of S to location and time can be written as:

$$S(x,t) = u(x)v(t). \tag{10.8}$$

If this is possible, Eq. (10.7) can be replaced by two ODEs by using the following procedure. Using Eq. (10.8), we can define

$$\partial S/\partial t = u\,dv/dt, \quad \partial S/\partial x = v\,du/dt, \quad \text{and} \quad \partial^2 S/\partial x^2 = v\,d^2u/dx^2.$$

Plugging this into Eq. (10.7) gives:

$$v\,d^2u/dx^2 = K u\,dv/dt,$$

$$\frac{1}{u}d^2u/dx^2 = \frac{K}{v}dv/dt.$$

Note that in the last expression the part left of the equal sign is uniquely a function of location x, and the part right of the equal sign only depends on time t. Since they both have to be equal, say to a value k, we obtain two ODEs:

$$\frac{1}{u}d^2u/dx^2 = k \quad \text{or} \quad d^2u/dx^2 - ku = 0, \text{ and}$$

$$\frac{K}{v}dv/dt = k \quad \text{or} \quad dv/dt - \frac{k}{K}v = 0.$$

Following our assumption in Eq. (10.8), the product of the solutions to each of these ODEs is a solution to the PDE.

Just as when solving ODEs numerically, finding numerical solutions by simulation of the PDE is a good approach to examine the dynamics. An example is shown in Chapter 30 where we show spatiotemporal activity of a cortical area (Fig. 30.11). To obtain **numerical stability** in simulations of a PDE, it is important that the step sizes employed for both the temporal and spatial domains satisfy the so-called **von Neumann criterion** (e.g., Press et al., 2007). This criterion describes the required relationship between the spatial and time steps in the numerical solution:

$$\frac{2\Delta t}{K\Delta x^2} \leq 1 \tag{10.9}$$

Here, Δx and Δt are the spatial and time steps used for the numerical solution, and K is the constant from Eq. (10.7). If you forget about this and violate this criterion, you will be the first to know as your solutions will blow up.

EXERCISES

10.1 Create a MATLAB® script to solve the ODE of Exercise 9.2 and using the conditions as listed in 9.2d. Use the Euler and fourth-order Runge−Kutta methods.
Plot the numerical solutions you obtain and the analytical solution you obtained for Exercise 9.2 in a single figure.
Interpret your results.

10.2 Run the MATLAB® script pr10_2.m with different time steps and evaluate your results with respect to accuracy.

Reference

Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 2007. Numerical Recipes in C, third ed. Cambridge University Press, Cambridge, MA.

CHAPTER

11

Modeling

11.1 INTRODUCTION

Signal analysis is frequently used to characterize systems. The simplest approach for system identification is by using linear methods, an approach we will explore in Chapter 13. However, depending on the degree of nonlinearity of the system at hand, these linear methods may not always generate useful results. For example, in Chapter 27 we show how linear methods, such as cross-correlation, fail to characterize signals with a nonlinear component. To address this shortcoming, we have to use a nonlinear characterization such as those we discuss in Chapters 24–26, or by using metrics such as correlation dimension, the Lyapunov exponent, or Kolmogorov entropy to characterize the nonlinear properties (Chapter 27). Application of such advanced techniques is far from trivial. Therefore, the goal of this chapter is to introduce basics for modeling systems with an emphasis on techniques one can use to model and characterize nonlinear dynamics and their signals. In this context, this chapter will also provide an introduction to the application of Volterra series which will be described in more detail in Chapter 24. Nonlinear systems are also indicated as higher-order systems because they include operators beyond a first-order, linear one. Useful references on the characterization of nonlinear systems are the seminal text by Marmarelis and Marmarelis (1978) and the reprint edition of a text from the 1980s by Schetzen (2006). For more recent overviews, see Westwick and Kearney (2003) and Marmarelis (2004).

11.2 DIFFERENT TYPES OF MODELS

Before going into mathematical detail, it is useful to summarize some of the types of models that one may apply to characterize systems in

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00011-4

neuroscience. The first approach should always be the simplest; in this context that is a model for a linear system. For analysis of **linear** systems, we have first-order systems where output y depends linearly on input x. As an example of such a system, we can consider an attenuator or amplifier that multiplies input with a constant such as $y = 3x$. Expressions that characterize **nonlinear** systems will include higher-order terms and these systems, as we will see in Chapters 24–26, do not obey the scaling and superposition rules of linear models (to review these rules see Section 13.2). Examples of nonlinear higher-order systems are: $y = x^2$ (second-order system) and $y = 5 + x + 3x^3$ (third-order system).

The examples given in the above paragraph are **static** models (systems without memory), meaning that their output only depends on present input. In reality, and certainly in neuroscience, we usually must deal with **dynamical** models in which output depends on present and past input (not on future input); these systems are also called **causal**. Static models are represented by algebraic equations (as the ones in the previous paragraph), whereas dynamical systems are modeled by differential equations (for continuous time models) or difference equations (for discrete time models). General examples of linear dynamical systems with input x and output y are

$$\begin{aligned} &A_n\frac{d^n y(t)}{dt^n} + A_{n-1}\frac{d^{n-1}y(t)}{dt^{n-1}} + \cdots + A_0 y(t) \\ &\quad = B_m\frac{d^m x(t)}{dt^m} + B_{m-1}\frac{d^{m-1}x(t)}{dt^{m-1}} + \cdots + B_0 x(t) \end{aligned}$$

for continuous time systems and

$$\begin{aligned} &A_n y(k-n) + A_{n-1}y(k-n+1) + \cdots + A_0 y(k) \\ &\quad = B_m x(k-m) + B_{m-1}x(k-m+1) + \cdots + B_0 x(k) \end{aligned}$$

for discrete time systems (for the fundamentals of modeling dynamics see also Chapters 10, 13, and 23). If one of the terms in a differential or difference equation is of a higher order, we have a nonlinear dynamical system. For example $y - 4\left(\frac{dy}{dt}\right)^2 = 2x$ represents a second-order system.

Time invariance is a critical condition for the development of the convolution formalism. This property allows us to state that a system's response to identical stimuli at different points in time is always the same (provided that the system is in the same state of course). Just as we have linear time-invariant (LTI) systems, we also have nonlinear time-invariant systems which are usually abbreviated as NTI or NLTI systems.

Models of real systems can be generated according to two major methodological approaches. One might follow a **deductive** path and start from (physical) assumptions about the system and generate a **hypothetical model** to create predictions and then subsequently test these with measurements. These measurements can then be used to establish the parameters of the hypothetical model, and therefore this type of representation is often called a **parametric model**. An alternative to this procedure, the **inductive** path, is followed if one starts from the measurements of a system's input and output. This data-driven method uses (electrophysiological) recordings, rather than assumptions about the system, to mathematically relate input and output. Here, the system is considered as a **black box**, modeled by a mathematical relationship that transforms input into output. This type of model is often referred to as **nonparametric**. The method of induction is appropriate when dealing with complex systems that resist a reduction to a simpler parametric model. It can also be a starting point in which a system is first characterized as a black box and in subsequent steps pieces of the black box are replaced by parts with a physical meaning. In this combined approach, parts of the model may still be part of the black box, whereas other parts may be associated with a physical interpretation. In this case, the distinction between parametric and nonparametric models may becomes a bit fuzzy.

11.3 EXAMPLES OF PARAMETRIC AND NONPARAMETRIC MODELS

A **parametric** model usually has relatively few parameters. A simple example of a parametric model of a dynamical LTI system is the Ordinary Differential Equation (ODE) for a filter. For example, $x = RC\frac{dy}{dt} + y$ describes input x and output y of a simple RC circuit (Fig. 11.1A). The only parameters in this case are the values of resistor R and capacitor C in the equation. Subsequently, the value for these parameters can be determined experimentally from observing the system's behavior.

Note: For further analysis of this circuit, see Chapters 16 and 17. In Section 16.1 it is shown how RC can be obtained from the filter's unit step response (Eq. 16.8) and in Section 17.3, RC is determined from the -3 dB point of the frequency characteristic of the filter (Eq. 17.5).

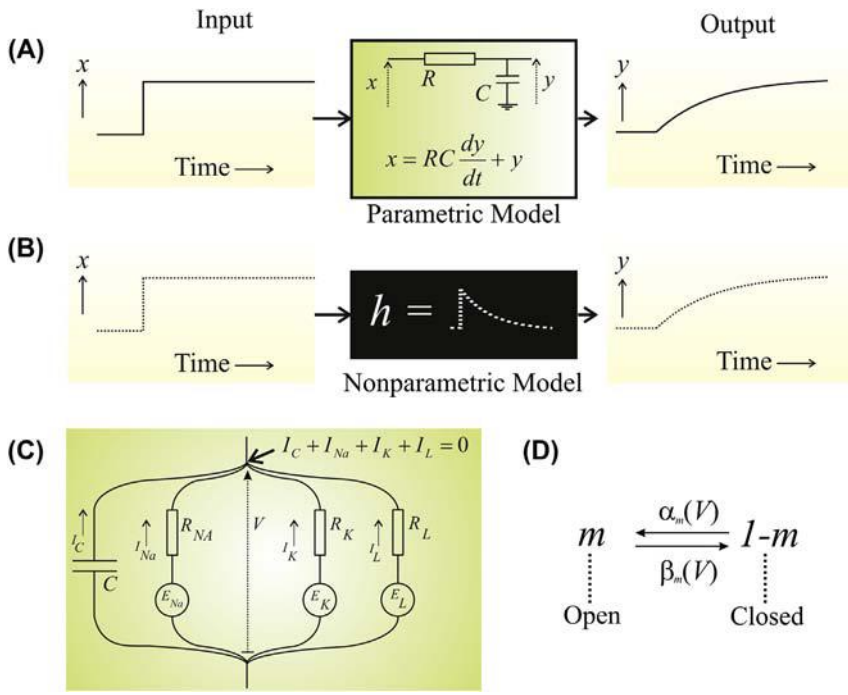


FIGURE 11.1 (A) Example of a parametric model of a dynamical linear system (a low-pass filter) and its input and output (x and y, respectively). (B) The *black box*, nonparametric equivalent of the same system is the *white curve* representing the (sampled) unit impulse response. Both models permit us to predict the output resulting from an arbitrary input such as the unit step function. The parametric model has two parameters (R and C) with a physical meaning. The nonparametric model consists of many parameters (the samples making up the unit impulse response) without a direct physical meaning. (C) Hodgkin and Huxley's electronic equivalent circuit for the biomembrane. The model consists of the membrane capacitance (C) and three parallel ion channels: one for sodium, one for potassium, and a leakage channel. According to Kirchhoff's first law the sum of all currents at the node (*arrow*) must be zero. (D) Model for gating variable m in the Hodgkin and Huxley formalism.

A very famous parametric model in neuroscience is the Hodgkin and Huxley (1952) formalism using four variables to describe the action potential generated in the squid's giant axon: V is the membrane potential and three other variables m, h, and n describe the membrane potential dependent characteristics of sodium and potassium conductivity. Initially in the 1950s, the proposed formalism was entirely hypothetical, later after the molecular basis for Na^+ and K^+ ion channels was elucidated, a physical interpretation of large parts of the model could be

made. The gating variable m determines the depolarization process caused by increased conductance of sodium ions (causing an influx of positively charged sodium ions) during the action potential generation. The variables h and n are slightly slower recovery variables representing inactivation of conductance for sodium ions (reduced Na^+ influx) and activation of potassium conductance (causing an outward flux of positively charged potassium ions).

Hodgkin and Huxley's model relates all these variables in an equivalent circuit of the excitable biomembrane (Fig. 11.1C) by setting the sum of all membrane currents equal to zero. This is Kirchhoff's first law (see Appendix 1.1) and by applying this law to the node indicated with the arrow in the membrane model in Fig. 11.1C we obtain:

$$\underbrace{C\frac{dV}{dt}}_{\substack{\text{Capacitive}\\ \text{Current}\\ =I_C}} + \underbrace{\frac{V-E_{Na}}{R_{Na}}}_{\substack{\text{Sodium}\\ \text{Current}\\ =I_{Na}}} + \underbrace{\frac{V-E_K}{R_K}}_{\substack{\text{Potassium}\\ \text{Current}\\ =I_K}} + \underbrace{I_L}_{\substack{\text{Leak}\\ \text{Current}}} = 0 \tag{11.1}$$

In this expression we have several parameters: C is the membrane capacitance; R_{NA} and R_K are the resistance values for sodium and potassium ions; E_{NA} and E_K are the equilibrium potentials for sodium and potassium ions computed with the Nernst equation (Appendix 1.1); and I_L is a constant leakage current attributed to Cl^- ions. The sodium and potassium currents are determined with Ohm's law (Appendix 1.1): each ion species experiences a potential drop equal to the difference between the membrane potential V and its equilibrium potential (e.g., for sodium: $V - E_{Na}$), this potential drop divided by the resistance is the ion current (e.g., for sodium the current is $\frac{V-E_{Na}}{R_{Na}}$). In addition to Eq. (11.1), Hodgkin and Huxley (1952) described the dynamics for R_{NA} and R_K by the nonlinear relationships $g_{Na} = \frac{1}{R_{Na}} = \overline{g_{Na}}m^3h$ and $g_K = \frac{1}{R_K} = \overline{g_K}n^4$ in which $\overline{g_{Na}}$ and $\overline{g_K}$ are the maximum conductivity values for sodium and potassium. Furthermore, the gating variable m is modeled by a reversible process between the open (m) and closed ($1 - m$) states (Fig. 11.1D), which can be represented by the following ODE:

$$\frac{dm}{dt} = \alpha_m(V)(1-m) - \beta_m(V)m \tag{11.2}$$

The rate parameters that govern this process α_m and β_m depend on the membrane potential V in a nonlinear fashion. The two other gating variables h and n follow the same formalism with membrane potential-dependent rate constants α_h, β_h, α_n, and β_n. Hodgkin and Huxley

determined these nonlinear relationships between the rate parameters and membrane potential from voltage clamp experiments.

Since the development of computer technology, the Hodgkin and Huxley formalism has been widely used in simulations of neural systems ranging from very detailed models of single neurons (e.g., De Schutter and Bower, 1994a,b) to large-scale networks of neocortex (e.g., Traub et al., 2005; van Drongelen et al., 2006).

Although Hodgkin and Huxley's model only contains four variables, it is still too complex to approach analytically. Several authors solved this problem by reducing the four-dimensional model into a two-dimensional one; the Fitzhugh–Nagumo model (Fitzhugh, 1961) is an example of such a reduction. In these models, the gating variable m of the Hodgkin and Huxley model is removed by considering sodium activation to be instantaneous; subsequently h and n are combined into a single recovery variable w. Fitzhugh used the following pair of coupled differential equations:

$$\frac{dV}{dt} = V(a - V)(V - 1) - w + I \quad \text{and} \quad \frac{dw}{dt} = bV - cw \tag{11.3}$$

in which a, b, and c are parameters; and I is a term representing injected current. The remaining two variables in these models are the membrane potential V and a single recovery variable w, generating a model that is amenable to mathematical analysis (for an excellent discussion of simplified versions of the Hodgkin and Huxley model, see Izhikevich, 2007).

Nonparametric models describe a system's input–output relationship, usually by using a large number of parameters, and these parameters do not necessarily have a physical interpretation. Generally speaking, a nonparametric model is generated from a procedure in which we relate a system's input $x(t)$ and output $y(t)$. Just as we can relate two variables with a function, we can link two time series with an **operator**. An example of such a nonparametric model would be the characterization of a LTI dynamical system with its (sampled) unit-impulse response (h, Fig. 11.1B). The operator in this case would be convolution because convolution of the input time series $x(t)$ with the system's unit impulse response $h(t)$ generates the system's output time series $y(t) : y(t) = h(t) \otimes x(t)$ (see Chapter 13). Although one might point out that such a nonparametric description doesn't necessarily provide direct insight into the system's components or the mechanisms underlying the system's operation, the curve of the unit impulse response permits us to predict the system's response to any input such as the unit step (Fig. 11.1B).

11.4 POLYNOMIALS

For static systems, both linear and nonlinear, one can use algebraic expressions to describe their input–output characteristic, and polynomials are often used for this purpose. Polynomials are sums of monomials, where monomials are expressions that consist of a constant multiplied by one or more variables; the exponent of the variable is its degree. For example, $z(t) = a\,x(t)^4 y(t)^3$ is a monomial with a constant (parameter) a and a degree of 4 for x and 3 for y. We can see that this expression represents a static process because at any time t, output z only depends on the present values of inputs x and y. It is important to note that although the relationship between z and x, y is nonlinear, the expression can be considered a linear function of the parameter a.

11.4.1 Describing Discrete Time Data Sets

Applying the above to the characterization of nonlinear systems, we could describe the relationship between input $x(t)$ and output $y(t)$ of a static nonlinearity (a nonlinear system without memory) with a polynomial such as the following power series:

$$y(t) = a_0 + a_1 x(t) + a_2 x(t)^2 + a_3 x(t)^3 + \cdots + a_i x(t)^i + \cdots = \sum_{i=0}^{\infty} a_i x(t)^i \tag{11.4}$$

In principle, power series are infinite, however in our applications they will always consist of a finite number of monomials. The fact that **Eq. (11.4) is linear with respect to its parameters a_i** can be used to fit the series by using a technique called least squares minimization. Using this approach of fitting polynomials to recorded data sets is often called regression analysis. This procedure works as follows. Suppose we have two sets of N measurements: a system's input x_n and associated output y_n. If we now model our system as a second-order static system, we can truncate the expression in Eq. (11.4) above the second power and estimate the output y as $a_0 + a_1 x_n + a_2 x_n^2$. Subsequently we can define the error of our estimate ε^2 as:

$$\varepsilon^2 = \sum_{n=1}^{N} \left[y_n - \left(a_0 + a_1 x_n + a_2 x_n^2 \right) \right]^2 \tag{11.5}$$

By following the same approach we used to find the coefficients in Lomb's algorithm (Chapter 8), we find the minimum associated with the

best choice for parameters a_0, a_1, and a_2 by setting the partial derivatives of ε^2 to these three parameters to a_0, a_1, and a_2 to zero:

$$\frac{\partial \varepsilon^2}{\partial a_i} = \sum_{n=1}^{N} 2\left[y_n - \left(a_0 + a_1 x_n + a_2 x_n^2\right)\right] \frac{\partial\left[y_n - \left(a_0 + a_1 x_n + a_2 x_n^2\right)\right]}{\partial a_i} = 0$$
$$\text{for } i = 0, 1, 2 \tag{11.6a}$$

and we get what are called the **normal equations**:

$$\begin{aligned}
\frac{\partial \varepsilon^2}{\partial a_0} &= -2\sum_{n=1}^{N}\left[y_n - a_0 - a_1 x_n - a_2 x_n^2\right] = 0 \\
&\rightarrow a_0 \underbrace{N}_{\sum_{n=1}^{N} 1} + a_1 \sum_{n=1}^{N} x_n + a_2 \sum_{n=1}^{N} x_n^2 = \sum_{n=1}^{N} y_n \\
\frac{\partial \varepsilon^2}{\partial a_1} &= -2\sum_{n=1}^{N}\left[y_n - a_0 - a_1 x_n - a_2 x_n^2\right] x_n = 0 \\
&\rightarrow a_0 \sum_{n=1}^{N} x_n + a_1 \sum_{n=1}^{N} x_n^2 + a_2 \sum_{n=1}^{N} x_n^3 = \sum_{n=1}^{N} y_n x_n \\
\frac{\partial \varepsilon^2}{\partial a_2} &= -2\sum_{n=1}^{N}\left[y_n - a_0 - a_1 x_n - a_2 x_n^2\right] x_n^2 = 0 \\
&\rightarrow a_0 \sum_{n=1}^{N} x_n^2 + a_1 \sum_{n=1}^{N} x_n^3 + a_2 \sum_{n=1}^{N} x_n^4 = \sum_{n=1}^{N} y_n x_n^2
\end{aligned} \tag{11.6b}$$

Note that in Eq. (11.6b) all summation (Σ) expressions are numbers that can be computed from the observations; therefore there are three linear equations with three unknown parameters a_0, a_1, and a_2 to compute. This should be no problem if the set of equations behaves well. Note that if we had truncated Eq. (11.4) at a_1, the normal equations that we would have obtained would have been the well-known equations for linear regression.

It is a bit tedious to solve the three equations in Eq. (11.6b), therefore one prefers to solve the coefficients by using the matrix notation for the three equations $XA = Y$:

$$\underbrace{\begin{bmatrix} N & \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} x_n^2 \\ \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} x_n^2 & \sum_{n=1}^{N} x_n^3 \\ \sum_{n=1}^{N} x_n^2 & \sum_{n=1}^{N} x_n^3 & \sum_{n=1}^{N} x_n^4 \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} \sum_{n=1}^{N} y_n \\ \sum_{n=1}^{N} y_n x_n \\ \sum_{n=1}^{N} y_n x_n^2 \end{bmatrix}}_{Y} \tag{11.6c}$$

The coefficients can be found by solving Eq. (11.6c): i.e., $A = X^{-1}Y$. In MATLAB® we can use the \ operator to obtain this result: `A = X\Y`. An example for approximating an exponential function $y = e^x$ is given in `pr11_1.m`.

11.4.2 Describing Analytic Functions

The previous example works if one has discrete time data such as a set of digitized recordings of a system's input and output. In other applications, one might deal with a parametric model and consequently have access to analytic functions that describe some nonlinear system under investigation (recall that an **analytic function** is smooth and differentiable). In this case, the so-called Maclaurin or Taylor series approaches, which are explained in Sections 11.4.2.1 and 11.4.2.2, may be applied to convert the function into a power series. Such a power series approach can also be helpful for creating a linear approximation of a nonlinear function in the neighborhood of a point of interest. Because linear relationships are easier to analyze than nonlinear ones, this technique of linearization of nonlinear functions can help us understand the behavior of complex nonlinear processes.

Like the polynomials discussed in the previous section, the Maclaurin and Taylor series describe static systems; to describe dynamical systems, we can use the Volterra series discussed in Chapter 24. In Section 11.5, we show that the Taylor series can be considered the static version of a Volterra series.

11.4.2.1 Maclaurin Series

A famous power series describing a function about the origin is the **Maclaurin** series. Let us consider an example with the exponential

function $f(t) = e^t$ and use the power series approach in Eq. (11.4) to represent this function:

$$f(t) = e^t = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + \ldots + a_i t^i + \ldots = \sum_{i=0}^{\infty} a_i t^i \tag{11.7}$$

The task at hand is to determine the values of the coefficients a_i for function e^t. We can use the following approach to perform this task. First we determine the derivatives of f.

$$\begin{aligned}
f(t) &= e^t = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + \ldots + a_i t^i + \ldots \\
\frac{df(t)}{dt} &= e^t = a_1 + 2a_2 t + 3a_3 t^2 + \ldots + i a_i t^{i-1} + \ldots \\
\frac{d^2 f(t)}{dt^2} &= e^t = 2a_2 + (2 \times 3) a_3 t + \ldots + (i \times (i-1)) a_i t^{i-2} + \ldots \\
\frac{d^3 f(t)}{dt^3} &= e^t = (2 \times 3) a_3 + \ldots + (i \times (i-1) \times (i-2)) a_i t^{i-3} + \ldots
\end{aligned} \tag{11.8}$$

and so forth.

The second step is to consider $f(t) = e^t$ about the origin; i.e., t becomes 0, and Eq. (11.8) simplify to:

$$\begin{aligned}
f(0) &= e^0 = 1 = \left[a_0 + a_1 t + a_2 t^2 + a_3 t^3 + \ldots + a_i t^i + \ldots\right]_{t=0} = a_0 \\
\frac{df(0)}{dt} &= e^0 = 1 = \left[a_1 + 2a_2 t + 3a_3 t^2 + \ldots + i a_i t^{i-1} + \ldots\right]_{t=0} = a_1 \\
\frac{d^2 f(0)}{dt^2} &= e^0 = 1 = \left[2a_2 + (2 \times 3) a_3 t + \ldots + (i \times (i-1)) a_i t^{i-2} + \ldots\right]_{t=0} = 2a_2 \\
\frac{d^3 f(0)}{dt^3} &= e^0 = 1 = \left[(2 \times 3) a_3 + \ldots + (i \times (i-1) \times (i-2)) a_i t^{i-3} + \ldots\right]_{t=0} \\
&= (2 \times 3) a_3
\end{aligned} \tag{11.9}$$

and so forth.

With the results obtained in Eq. (11.9), we can see that for the function $f(t) = e^t$, the values for the coefficients a_i are

$$a_i = \frac{1}{i!} \tag{11.10}$$

Combining this result in (11.10) with Eq. (11.7), we have found the well-known power series of an exponential function:

$$\boxed{f(t) = e^t = 1 + \frac{1}{1!}t + \frac{1}{2!}t^2 + \frac{1}{3!}t^3 + \ldots + \frac{1}{i!}t^i + \ldots = \sum_{i=0}^{\infty} \frac{1}{i!}t^i} \tag{11.11}$$

In the last expression $\sum_{i=0}^{\infty} \frac{1}{i!}t^i$, we use the definition $0! \equiv 1$. Note that by using this approach, we only include values of t, there are no previous or future values $(t \pm \tau)$ included in the power series; therefore this approach is **static** (or memoryless). An example of this approximation is implemented in MATLAB® script pr11_1.m.

In the example above we used the exponential $\exp(t)$ for $f(t)$; if we consider the development of Eq. (11.4) for any function that can be differentiated, we get:

$$\begin{aligned}
f(0) &= \left[a_0 + a_1 t + a_2 t^2 + a_3 t^3 + \ldots + a_i t^i + \ldots\right]_{t=0} = a_0 \rightarrow a_0 = f(0) \\
\frac{df(0)}{dt} &= \left[a_1 + 2a_2 t + 3a_3 t^2 + \ldots + i a_i t^{i-1} + \ldots\right]_{t=0} = a_1 \rightarrow a_1 = f'(0) \\
\frac{d^2 f(0)}{dt^2} &= \left[2a_2 + (2 \times 3)a_3 t + \ldots + (i \times (i-1))a_i t^{i-2} + \ldots\right]_{t=0} = 2a_2 \\
&\rightarrow a_2 = \frac{f''(0)}{2} \\
\frac{d^3 f(0)}{dt^3} &= \left[(2 \times 3)a_3 + \ldots + (i \times (i-1) \times (i-2))a_i t^{i-3} + \ldots\right]_{t=0} = (2 \times 3)a_3 \\
&\rightarrow a_3 = \frac{f'''(0)}{(2 \times 3)}
\end{aligned} \tag{11.12}$$

and so forth.

Here the notations $f'(0), f''(0), f'''(0), \ldots$ are not functions but represent the numbers computed as the value of the first, second, third, … derivatives of f at $t = 0$. This more general notation finally generates the expression for the so-called Maclaurin series of $f(t)$:

$$\boxed{f(t) = f(0) + \frac{1}{1!}tf'(0) + \frac{1}{2!}t^2 f''(0) + \frac{1}{3!}t^3 f'''(0) + \ldots} \tag{11.13}$$

11.4.2.2 Taylor Series

In the above example, we developed the power series for a function about the origin. The development of the Taylor series follows a similar

approach but now about any point α. For a power series of power N this becomes:

$$\begin{aligned} f(t) &= a_0 + a_1(t-\alpha) + a_2(t-\alpha)^2 + a_3(t-\alpha)^3 + \ldots + a_i(t-\alpha)^i + \ldots \\ &= \sum_{i=0}^{\infty} a_i(t-\alpha)^i \end{aligned} \tag{11.14}$$

We will now use a similar approach for the development of this series about α as we used in the Maclaurin series about the origin. Only in this case we set $t = \alpha$ so that all terms in Eq. (11.14) with $(t-\alpha)^i$ vanish. By following this procedure we get

$$\begin{aligned} f(\alpha) &= \Big[a_0 + a_1(t-\alpha) + a_2(t-\alpha)^2 + a_3(t-\alpha)^3 + \ldots + a_i(t-\alpha)^i + \ldots\Big]_{t=a} = a_0 \\ &\rightarrow a_0 = f(\alpha) \\ \frac{df(\alpha)}{dt} &= \Big[a_1 + 2a_2(t-\alpha) + 3a_3(t-\alpha)^2 + \ldots + ia_i(t-\alpha)^{i-1} + \ldots\Big]_{t=a} = a_1 \\ &\rightarrow a_1 = f'(\alpha) \\ \frac{d^2f(\alpha)}{dt^2} &= \Big[2a_2 + (2\times 3)a_3(t-\alpha) + \ldots + (i\times(i-1))a_i(t-\alpha)^{i-2} + \ldots\Big]_{t=a} = 2a_2 \\ &\rightarrow a_2 = \frac{f''(\alpha)}{2} \\ \frac{d^3f(\alpha)}{dt^3} &= \Big[(2\times 3)a_3 + \ldots + (i\times(i-1)\times(i-2))a_i(t-\alpha)^{i-3} + \ldots\Big]_{t=0} = (2\times 3)a_3 \\ &\rightarrow a_3 = \frac{f'''(\alpha)}{(2\times 3)} \end{aligned} \tag{11.15}$$

and so forth.

Similar to the notation used in the previous section, the notations $f'(\alpha), f''(\alpha), f'''(\alpha), \ldots$ are not functions but represent the numbers computed as the value of the first, second, third, … derivatives of f at $t = \alpha$. Substituting the findings in Eq. (11.15) into Eq. (11.14) we obtain the so-called Taylor series about $t = \alpha$:

$$\boxed{f(t) = f(\alpha) + \frac{1}{1!}(t-\alpha)f'(\alpha) + \frac{1}{2!}(t-\alpha)^2 f''(\alpha) + \frac{1}{3!}(t-\alpha)^3 f'''(\alpha) + \ldots} \tag{11.16}$$

Comparing Eqs. (11.13) and (11.16), we can establish that the Maclaurin series is the same as a Taylor series about the origin (that is $\alpha = 0$). This approach can be extended to higher dimensions, that is, for systems with multiple inputs; see Appendix 11.1 for examples of the two-dimensional case. It must be noted that in many texts the distinction between Maclaurin and Taylor series isn't always made and it is not uncommon to use the term Taylor series for both, a habit we will adopt in the following.

The number of terms in a Taylor series may be infinite. However, if we evaluate a system close to an equilibrium at the origin or α, the value of t or $(t - \alpha)$ is a small number $<<1$; this allows one to ignore higher-order terms in the power series t^n or $(t - \alpha)^n$ because they become progressively small. Thus, in general we can approximate any function close to α with a linear expression obtained from a Taylor series in which higher-order terms are ignored $f(t) \approx f(\alpha) + (t - \alpha)\, f'(\alpha)$; or, in the case where we evaluate the expression about the origin, we obtain the approximation $f(t) \approx f(0) + t f'(0)$. This technique of **linearizing a nonlinear function** plays an important role in the analysis of nonlinear systems. A system's behavior in a restricted part of its domain can be understood and approximated by a linear version of its characteristic. Sometimes a system is so complex that a piecewise approximation with linear functions is the best option for its analysis. For example, if we wanted to evaluate $\sin(t)$ close to the origin, we can apply Eq. (11.13), ignore the higher-order terms because they become negligibly small for small fluctuations about zero, and we find that $\sin(t) \approx t$. In general, such an approach may be useful if one studies a system close to equilibrium. For example, if one examines a neuron's subthreshold behavior, one must describe the membrane potential close to the resting potential; in this case it makes sense to linearize the nonlinear equations that govern the cell's electrical activity around the resting potential. Examples of a linearization procedure applied to nonlinear neuronal and network models can be found in Chapters 29 and 30.

When fitting a truncated power series to an analytic function, one could truncate the Taylor series at the desired order. However, due to the error introduced by truncation, one may actually obtain a better fit by a linear regression approach. An example is if one wants to approximate e^t with a second-order power function over a limited interval. The truncated Taylor series (see Eq. 11.11) is $1 + t + 0.5t^2$, but with a regression approach over the interval $[-1,1]$ one obtains a better fit with $0.9963 + 1.1037t + 0.5368t^2$. This can be seen by running MATLAB® script pr11_1 in which the original exponential function (red), the Taylor series (blue), and the regression result (black) are superimposed. The regression approach for obtaining a power series approximation is also a

valid solution if the Taylor series cannot be applied as in the case of a function that is nonanalytic, such as $y = |x|$ (no [unique] derivative at $x = 0$).

11.5 NONLINEAR SYSTEMS WITH MEMORY

In the above examples, the output $y(t)$ of the nonlinear systems could be described with a polynomial of $x(t)$ because there was a direct relationship between x and y; i.e., in these examples there was no memory in the system. Just as for the linear variant, nonlinear systems with memory also exist; and in this case, we must describe how output $y(t)$ depends on both present and past input: $x(t - \tau)$ with $\tau \geq 0$.

In Chapter 24, we will consider the so-called Volterra series for the characterization of dynamical nonlinear systems (that is nonlinear systems that do include a memory). With the Taylor series, for example, we link output value $y = f(x)$ to input value x as:

$$y = f(\alpha) + \frac{1}{1!}(x - \alpha)f'(\alpha) + \frac{1}{2!}(x - \alpha)^2 f''(\alpha) + \frac{1}{3!}(x - \alpha)^3 f'''(\alpha) + \ldots \tag{11.17}$$

In the example below, we will approximate a nonlinearity with a series truncated at the second-order:

$$y(t) = a_0 + a_1 x(t) + a_2 x(t)^2 \tag{11.18}$$

We can now generalize this procedure in which we related two **values** x and y into a slightly altered procedure in which we will relate a pair of **time series** $x(t)$ and $y(t)$. Just as we can relate two values x and y with a **function** f:

$$y = f(x), \tag{11.19a}$$

we can link two time series $x(t)$ and $y(t)$ with an **operator** F:

$$y(t) = F\{x(t)\} \tag{11.19b}$$

Note: In some texts on Volterra series F will be called a **functional**. Because F connects two functions $x(t)$ and $y(t)$, it is better to use the term operator because, strictly speaking, a **functional** maps a function onto a value, whereas an **operator** maps one function to another function.

A Volterra series can perform such an operation:

$$y(t) = \underbrace{h_0}_{\text{0th order term}} + \underbrace{\int_{-\infty}^{\infty} h_1(\tau_1)x(t-\tau_1)d\tau_1}_{\text{1st order term}} + \underbrace{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2}_{\text{2nd order term}} + \ldots$$
$$\underbrace{\ldots + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\ldots\int_{-\infty}^{\infty} h_n(\tau_1,\tau_2,\ldots,\tau_n)x(t-\tau_1)x(t-\tau_2)\ldots x(t-\tau_n)d\tau_1 d\tau_2\ldots d\tau_n}_{\text{nth order term}} \tag{11.20}$$

In the Volterra series (Eq. 11.20), input function $x(t)$ determines output function $y(t)$. The expression is analogous to the Taylor series except that the differentials of the Taylor series are replaced by integrals. The symbols h_0, h_1, h_2, h_n represent the so-called Volterra kernels. Note that the first-order component $\int_{-\infty}^{\infty} h_1(\tau_1)x(t-\tau_1)d\tau_1$ in the Volterra series is the convolution integral (Chapter 13). The higher-order components in Eq. (11.20) are convolution-like integrals. The representation with Volterra series is a nonparametric one, i.e., it allows one to predict system output when input is known but it does not necessarily provide insight into the system's components or underlying mechanisms. In the following we will examine examples of the relationship between Volterra and Taylor series. See also Chapter 24 for further details on the Volterra series.

In spite of the similarities between the Taylor series in Eq. (11.17) and the Volterra series in Eq. (11.20) discussed above, it may not be immediately obvious to you that they are related. Therefore, we will discuss the similarities for a simple dynamical nonlinear system which we will then subsequently transform into a static nonlinear one. Let us consider a dynamical second-order system that consists of a cascade of a dynamical linear component and a static nonlinear module (Fig. 11.2A). Such a cascade approach with the dynamics in the linear component combined with static nonlinearities is frequently applied in dynamical nonlinear system analysis. In this example, we have $h(t)$ as the linear component's unit impulse response and $a_0 + a_1 y + a_2 y^2$ (Eq. 11.18) to characterize the static second-order nonlinear component. From the diagram in Fig. 11.2A we can establish the output y of the linear module can be obtained from the convolution of the input x and the linear module's unit impulse response h:

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau \tag{11.21}$$

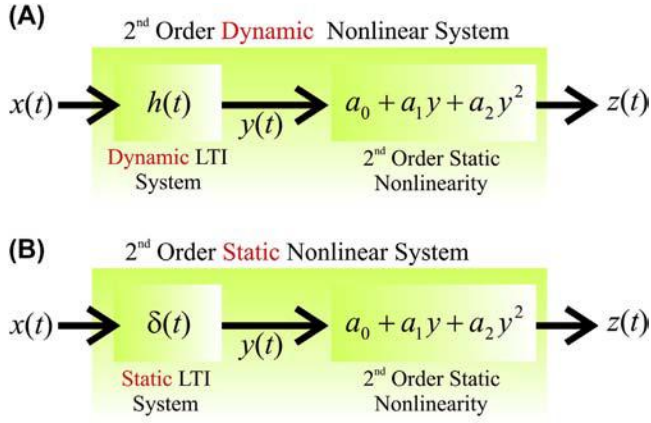


FIGURE 11.2 (A) Diagram of a second-order dynamical nonlinear system consisting of a cascade of a dynamical linear time invariant (LTI) system and a second-order nonlinearity. (B) A similar system for which the dynamical linear component is replaced by a static one.

The cascade's final output z can be obtained from the static nonlinearity characteristic by substituting the output of the linear component (Eq. 11.21) into the input of the static nonlinearity (Eq. 11.18):

$$z(t) = a_0 + a_1 \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau + a_2 \left[\int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau \right]^2 \tag{11.22}$$

This can be rewritten as:

$$z(t) = a_0 + a_1 \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau + a_2 \underbrace{\left[\left(\int_{-\infty}^{\infty} h(\tau_1)x(t-\tau_1)d\tau_1 \right) \left(\int_{-\infty}^{\infty} h(\tau_2)x(t-\tau_2)d\tau_2 \right) \right]}_{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(\tau_1)h(\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2} \tag{11.23}$$

This expression can be rearranged in the form of the Volterra series shown in Eq. (11.20):

$$z(t) = \overbrace{\underbrace{a_0}_{h_0}}^{\text{0th order term}} + \overbrace{\int_{-\infty}^{\infty} \underbrace{a_1 h(\tau)}_{h_1(\tau)}\, x(t-\tau)d\tau}^{\text{1st order term}} + \overbrace{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \underbrace{a_2 h(\tau_1)h(\tau_2)}_{h_2(\tau_1,\tau_2)}\, x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2}^{\text{2nd order term}} \tag{11.24}$$

Eq. (11.24) shows that the system in Fig. 11.2A can be characterized by a Volterra series for a second-order system with Volterra kernels h_0, h_1, and h_2.

To demonstrate that the Taylor series is the static equivalent of the Volterra series, we show the equivalence of Eq. (11.24) to the power series in Eq. (11.18). To accomplish this, we consider the case where our dynamical component in the cascade becomes static; the linear component is now replaced by the static function $y(t) = x(t)$. In other words the linear module's unit impulse response is the unit impulse δ itself, indicating that for this linear component output equals input (Fig. 11.2B). Therefore we can substitute $\delta(t)$ for $h(t)$ in Eq. (11.24):

$$\begin{aligned} z(t) &= a_0 + a_1 \underbrace{\int_{-\infty}^{\infty} \delta(\tau)x(t-\tau)d\tau}_{x(t)} + a_2 \underbrace{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \delta(\tau_1)\delta(\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2}_{x(t)x(t)=x(t)^2} \\ &= a_0 + a_1 x(t) + a_2 x(t)^2 \end{aligned} \tag{11.25}$$

The integrals in Eq. (11.25) are evaluated using the sifting property. Recall that $\int_{-\infty}^{\infty} x(\tau)\delta(\tau)d\tau = x(0)$, and accordingly $\int_{-\infty}^{\infty} x(t-\tau)\delta(\tau)d\tau = x(t)$.

Thus, in the static case, we can use the Volterra series to recover the original expression $z(t) = a_0 + a_1 x(t) + a_2 x(t)^2$, the power series in Eq. (11.18).

APPENDIX 11.1

Taylor Series for a Two-Dimensional Function

We can extend the above result for the Taylor series (Eq. 11.16) to a function $f(\tau, \sigma)$ of two variables τ and σ. In the case where we can subdivide the function into two separate ones (for example $f(\tau, \sigma) = f(\tau) + f(\sigma)$ or $f(\tau, \sigma) = f(\tau)f(\sigma)$), we can compute the Taylor series for each function $f(\tau)$ and $f(\sigma)$ and add or multiply the individual series to obtain the expression for $f(\tau, \sigma)$. Such an approach would, for example, work if $f(\tau, \sigma) = e^{\tau} \sin(\sigma)$.

Alternatively one can approach the development of a 2-D Taylor series more generally, and consider f around point α, β.

$$\begin{aligned} f(\tau, \sigma) = {} & a_{00} + a_{10}(\tau - \alpha) + a_{01}(\sigma - \beta) + a_{20}(\tau - \alpha)^2 + a_{11}(\tau - \alpha)(\sigma - \beta) \\ & + a_{02}(\sigma - \beta)^2 + a_{30}(\tau - \alpha)^3 + a_{21}(\tau - \alpha)^2(\sigma - \beta) \\ & + a_{12}(\tau - \alpha)(\sigma - \beta)^2 + a_{03}(\sigma - \beta)^3 + a_{40}(\tau - \alpha)^4 + \ldots \end{aligned} \tag{A11.1-1}$$

Using a similar approach as the one for the single-variable Taylor series, we set τ and σ to α and β and find $f(\alpha,\beta) = a_{00}$. To find the other coefficients we use partial differentiation for f at point α, β:

$$\frac{\partial f(\alpha, \beta)}{\partial \tau} = a_{10}, \quad \frac{\partial f(\alpha, \beta)}{\partial \sigma} = a_{01}, \quad \frac{\partial^2 f(\alpha, \beta)}{\partial \tau^2} = 2a_{20}, \quad \frac{\partial^2 f(\alpha, \beta)}{\partial \tau \partial \sigma} = a_{11},$$
$$\frac{\partial^2 f(\alpha, \beta)}{\partial \sigma^2} = 2a_{02}, \ \ldots \tag{A11.1-2}$$

This technique can be used to obtain the full power series of f. In most applications we are usually interested in the linear approximation of the two-dimensional series:

$$\boxed{f(\tau, \sigma) \approx f(\alpha, \beta) + \frac{\partial f(\alpha, \beta)}{\partial \tau}(\tau - \alpha) + \frac{\partial f(\alpha, \beta)}{\partial \sigma}(\sigma - \beta)} \tag{A11.1-3a}$$

All higher-order nonlinear terms are often not considered because we assume that we only look into f around point α, β; therefore $\tau - \alpha$ and $\sigma - \beta$ are very small numbers, and higher powers of these small contributions are even smaller; that is f in the neighborhood of point α, β can be approximated with the linear terms in Eq. (A11.1-3a). In many cases, especially in physics literature, you may encounter an alternative notation for the linear approximation of a nonlinear system. The small fluctuations $\tau - \alpha$ and $\sigma - \beta$ around α, β are indicated as perturbations $\delta\tau$ and $\delta\sigma$, and

the notation for $f(\alpha,\beta)$, $\frac{\partial f(\alpha,\beta)}{\partial \tau}$, and $\frac{\partial f(\alpha,\beta)}{\partial \sigma}$ is changed to $[f]_{\alpha,\beta}$, $\left[\frac{\partial f}{\partial \tau}\right]_{\alpha,\beta}$, and $\left[\frac{\partial f}{\partial \sigma}\right]_{\alpha,\beta}$:

$$\boxed{f(\tau,\sigma) \approx [f]_{\alpha,\beta} + \left[\frac{\partial f}{\partial \tau}\right]_{\alpha,\beta} \delta\tau + \left[\frac{\partial f}{\partial \sigma}\right]_{\alpha,\beta} \delta\sigma} \qquad \text{(A11.1-3b)}$$

Again, recall that in this notation $[f]_{\alpha,\beta}$, $\left[\frac{\partial f}{\partial \tau}\right]_{\alpha,\beta}$, and $\left[\frac{\partial f}{\partial \sigma}\right]_{\alpha,\beta}$ represent the coefficients in the equation. They are numbers and not functions, because these are the function and its derivatives evaluated at point α, β. An example of an application linearizing the nonlinear Hodgkin and Huxley equations can be found in Chapter 29 and in Koch (1999).

EXERCISES

11.1 Use the power series approach to find Euler's relation: $e^{-jt} = \cos(t) - j\sin(t)$.
Hint: determine a Maclaurin series for $\sin(t)$, $\cos(t)$, and $\exp(-jt)$.

11.2 Create a power series for $y = |x|$ in the interval $[-1, 1]$ by using the regression approach.

References

De Schutter, E., Bower, J.M., 1994a. An active membrane model of the cerebellar Purkinje cell. I. Simulation of current clamps in slice. J. Neurophysiol. 71, 375–400.

De Schutter, E., Bower, J.M., 1994b. An active membrane model of the cerebellar Purkinje cell. II. Simulation of synaptic responses. J. Neurophysiol. 71, 401–419.

Fitzhugh, R.A., 1961. Impulses and physiological states in theoretical models of nerve membrane. Biophys. J. 1, 445–466.

Hodgkin, A.L., Huxley, A.F., 1952. A quantitative description of membrane current and its application to conduction and excitation in the nerve. J. Physiol. 117, 500–544.

Izhikevich, E.M., 2007. Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting. MIT Press, Cambridge, MA.

Koch, C., 1999. Biophysics of Computation: Information Processing in Single Neurons. Oxford University Press, New York.

Marmarelis, P.Z., Marmarelis, V.Z., 1978. Analysis of Physiological Systems: The White Noise Approach. Plenum Press, New York.

Marmarelis, V.Z., 2004. Nonlinear Dynamic Modeling of Physiological Systems. IEEE Press, John Wiley & Sons Inc., Hoboken, NJ.

Schetzen, M., 2006. The Volterra & Wiener Theories of Nonlinear Systems, second reprint ed. Krieger Publishing Company, Malabar, FL.

Traub, R.D., Contreras, D., Cunningham, M.O., Murray, H., LeBeau, F.E.N., Roopun, A., et al., 2005. Single-column thalamocortical network model exhibiting gamma oscillations, sleep spindles, and epileptogenic bursts. J. Neurophysiol. 93, 2194–2232.

van Drongelen, W., Koch, H., Elsen, F.P., Lee, H.C., Mrejeru, A., Doren, E., et al., 2006. The role of persistent sodium current in bursting activity of mouse neocortical networks in vitro. J. Neurophysiol. 96, 2564–2577.

Westwick, D.T., Kearney, R.E., 2003. Identification of Nonlinear Physiological Systems. IEEE Press, John Wiley & Sons Inc., Hoboken, NJ.

CHAPTER

12

Laplace and z-Transform

12.1 INTRODUCTION

In this chapter we briefly summarize the use of the ***Laplace transform*** and the closely related ***z-transform***. The former is used in the analysis of continuous time systems, while the latter is the equivalent for discrete time (sampled) data sets. Both transforms are related to the Fourier transform, therefore (for those not familiar with spectral analysis) it is recommended to review Chapters 5–7 before proceeding with this chapter. The goal is to use the Laplace and z-transforms to analyze the input–output relationship of linear systems, which we will need specifically for the subsequent chapters which cover the application of analog and digital filters. The starting point for the mathematical description of these linear time-invariant (LTI) systems is their associated differential and difference equations that govern their input–output relationships (Chapters 11 and 13, Eqs. 13.1a and 13.1.b). It is strongly recommended to review Section 13.2 if you are not familiar with the LTI terminology.

12.2 THE USE OF TRANSFORMS TO SOLVE ORDINARY DIFFERENTIAL EQUATIONS

Solving ordinary differential equations (ODEs) and using convolution to analyze LTI systems can be mathematically complicated. In many cases, this task can be simplified considerably by transforming the problem into another domain (Fig. 12.1) where many operations can be performed algebraically. In Chapter 13 we will show, for example, that (complicated) convolution and correlation integrals in the time domain are equivalent to (simpler) multiplications in the frequency domain. Because the difference between the Fourier transform on one hand, and the Laplace and z-transforms on the other, is merely a change from the imaginary variable

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00012-6

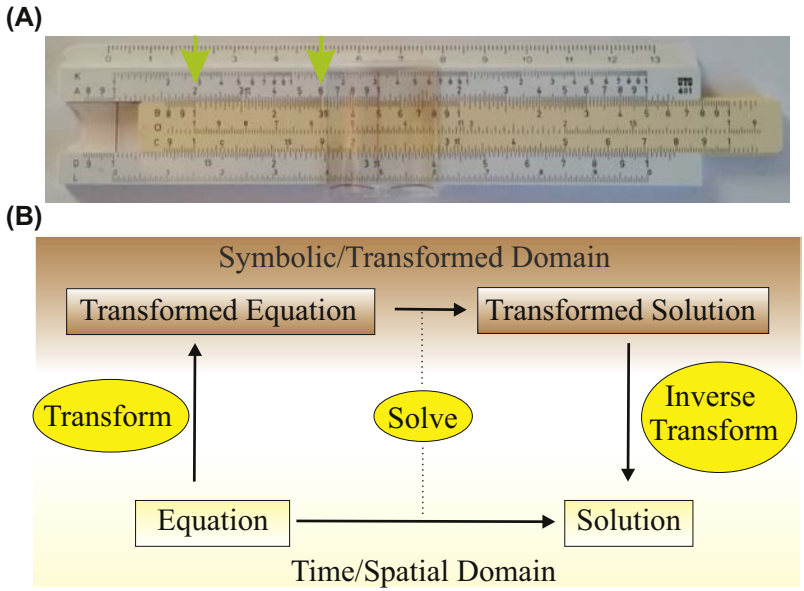


FIGURE 12.1 (A) The slide ruler uses logarithmic scales to do multiplications. In this example the *green arrows* show the procedure to find that $2 \times 3 = 6$. (B) Transforms and solving equations. Equations in the time/spatial domain are transformed into the symbolic domain and solved. Next, this solution is inverse transformed into the time/spatial domain.

$j\omega$ to a complex variable s or z, ***we can extend the frequency domain results for convolution and correlation into the s- and z-domains***.

The idea of using a ***transformation*** is to make use of properties that make a problem easier to solve in the transformed domain. Solving a multiplication problem by a transformation to logarithms is an example of such a procedure. The transformation allows substitution of addition for multiplication. For example $3.56 \times 4.18 = 14.8808$ can be calculated directly with multiplication. On the other hand, if one could use a table for $\log_{10}$ values we could find $\log_{10}(3.56) = 0.5514$ and $\log_{10}(4.18) = 0.6212$, and calculate $\log_{10}(3.56) + \log_{10}(4.18) = 1.1726$; the answer can then be obtained by looking up the inverse transformation of the resulting value in the table (i.e., $10^{1.1726} = 14.8808$). This example illustrates that we replace a multiplication by a (simpler) addition under the assumption that we can efficiently make use of a table of log-transforms and their inverses. An application of this strategy is the old-fashioned slide ruler (Fig. 12.1A). Here we slide two logarithmic scales relative to each other and because both scales are logarithmic, the additive effect allows one to perform multiplication. The left green arrow in Fig. 12.1A shows how the sliding scale is set to the value of 2 of the upper fixed scale, and at the right green arrow we can see how to establish that $2 \times 3 = 6$.

In a similar fashion as the log-transform, a solution of an ODE can be found by using the Laplace transform or the Fourier transform of the equation, while the z-transform can be used for the solution of difference equations. As in the log-transform example above, the ***rationale*** for the discussed approach (summarized in Fig. 12.1B) is that for some types of problems the solution in the transformed domain (plus the steps involved in finding both the transformation and inverse transformation from a table), is more easily calculated than with a direct approach of finding a solution in the time or spatial domain. Since deriving the transforms of arbitrary functions analytically is not often straightforward, a critical element in the relative ease of finding solutions in alternate domains is the existence of tables of Fourier, Laplace, and z-transform pairs. A few examples are summarized in the tables in Appendix 12.1. Extended versions of these tables can be readily found in general textbooks (e.g., Northrop, 2003; Hsu, 1995), while even more extended versions of such tables are available in specialized publications (e.g., Abramowitz and Stegun, 1975), or from specialized websites. Alternately, software packages such as Mathematica and the Symbolic Math Toolbox in MATLAB® can calculate transforms and their inverses.

12.3 THE LAPLACE TRANSFORM

In Chapter 6 the Fourier transform is defined as:

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt \tag{12.1}$$

The Laplace transform is similar:

$$F(s) = \int_{0}^{\infty} f(t)e^{-st}dt \tag{12.2}$$

Here the imaginary variable $j\omega$ is replace by complex variable s with both a real (σ) and imaginary ($j\omega$) part: $s = \sigma + j\omega$. Here we show the one-sided Laplace transform with the integration limits from $0 \rightarrow \infty$. We focus here on the one-sided Laplace transform because we commonly deal with causal systems which we begin to study while the system is in rest at some convenient point in time defined as $t = 0$. At this point in time we also may start to perturb the system with an input signal, but we do not need to worry about the system for $t < 0$, hence the integration limits $0 \rightarrow \infty$ in Eq. (12.2). A two-sided Laplace transform (with integration $-\infty \rightarrow \infty$) does exist, but because of its limited application in our context, it won't be discussed in this text. There are several reasons why

transformed ODEs are simpler to solve than their untransformed counterparts. Primarily the evaluation of convolution and cross-correlation integrals is replaced by multiplication. In addition, dealing with differentiation (and integration) is also fairly straightforward in the transformed domain. For example, the Laplace transform $L[\ldots]$ of the derivative of $f(t)$ is:

$$L\left[\frac{f(t)}{dt}\right] = \int_0^\infty \frac{f(t)}{dt}\, e^{-st} dt = \left[f(t)e^{-st}\right]_0^\infty + \int_0^\infty sf(t)e^{-st} dt = -f(0) + sF(s) \tag{12.3}$$

Notes:

1. The above integral was solved using integration by parts (Appendix 3.2): $\int udv = uv - \int vdu$, with $u = e^{-st}$ and $dv = f'(t)$).
2. Note that in Eq. (12.3) the Laplace transform is symbolized by ***operator*** $L[\ldots]$.

Using the result for the derivative in Eq. (12.3) we can apply the Laplace transform to solving ODEs, for example the general formalism describing an LTI system's input−output relationship (Chapter 13, Eq. 13.1a). Using the typical notation, we define $Y(s)$ as the transform of the system output $y(t)$ and $X(s)$ as the transform of input $x(t)$, and we further conveniently assume that all initial conditions $x(0),\ x'(0), \ldots, y(0),\ y'(0), \ldots$ etc. are zero. This allows us to transform the terms with $y(t)$ and $x(t)$ and their derivatives from Eq. (13.1a) to:

$$\begin{aligned} &x(t) \Leftrightarrow X(s), \\ &x'(t) \Leftrightarrow sX(s), \\ &x''(t) \Leftrightarrow s^2X(s), \text{ etc} \end{aligned}$$

and

$$\begin{aligned} &y(t) \Leftrightarrow Y(s), \\ &y'(t) \Leftrightarrow sY(s), \\ &y''(t) \Leftrightarrow s^2Y(s), \text{ etc} \end{aligned}$$

This results in:

$$\left[A_n s^n + A_{n-1}s^{n-1} + \cdots + A_0\right]Y(s) = \left[B_m s^m + B_{m-1}s^{m-1} + \cdots + B_0\right]X(s) \tag{12.4}$$

where it should be noted that, with zero initial conditions, higher-order derivatives are simplified to the complex variable s raised to higher powers. Thus, the ratio between the output and input of the system can be represented in the Laplace domain by:

$$H(s) = \frac{Y(s)}{X(s)} = \frac{B_m s^m + B_{m-1} s^{m-1} + \cdots + B_0}{A_n s^n + A_{n-1} s^{n-1} + \cdots + A_0} \tag{12.5}$$

The function $H(s)$ is the Laplace transform of $h(t)$, the ***unit impulse response*** of the LTI system. $H(s)$ is also called the ***transfer function*** of the LTI system.

12.4 EXAMPLES OF THE LAPLACE TRANSFORM

12.4.1 The Transform of a Few Commonly Used Functions

The Laplace transform of the unit impulse function can be obtained by using the sifting property. Here it is important to assume that the domain of the impulse function includes zero as part of the integration limits of the one-sided Laplace transform. In some texts this is specifically stressed by indicating the integration as $\int_{0^-}^{\infty}$; in the following we will not use this 0^- notation explicitly. The Laplace transform of the unit impulse evaluates to:

$$L[\delta(t)] = \int_0^{\infty} \delta(t) e^{-st} dt = e^{-0} = 1 \tag{12.6}$$

The Laplace transform of the unit step function $U(t)$:

$$L[U(t)] = \int_0^{\infty} 1 e^{-st} dt = \left[-\frac{1}{s} e^{-st} \right]_0^{\infty} = -\frac{1}{s}[0 - 1] = \frac{1}{s} \tag{12.7}$$

This result should not be too surprising considering the relationship we found between the Laplace transform of a function and its derivative in Eq. (12.3). The unit impulse can be considered the derivative of the unit step (Chapter 2) and in the Laplace domain they indeed differ by a factor s.

Some particularly important functions for analysis of linear systems are exponentials, sine and cosine waves. Let's consider the exponential function e^{at} in which a can be a positive, negative, real, or complex number. Further, we will only consider the exponential for $t \geq 0$ (formally

this can be thought of as multiplying the exponential by the unit step function: $U(t)e^{at}$). The Laplace transform is:

$$\begin{aligned} L[e^{at}] &= \int_0^\infty e^{at}e^{-st}dt = \int_0^\infty e^{-(s-a)t}dt\left[-\frac{1}{s-a}e^{-(s-a)t}\right]_0^\infty \\ &= -\frac{1}{s-a}[0-1] = \frac{1}{s-a} \end{aligned} \tag{12.8}$$

As usual we did not worry about the convergence of the integral here and conveniently assumed that the exponential $e^{-(s-a)t}$ at infinity is zero. More on the issue of convergence of integrals and existence of Laplace and z-transforms can be found in Appendix 12.2.

Sine and cosine waves can be expressed as exponential expressions using Euler's formula $\left[e^{j\omega t} = \cos(\omega t) + j\sin(\omega t)\right]$ and are easily solved using the result found in Eq. (12.8). For instance: $\sin(\omega t) = \frac{1}{2j}\left(e^{j\omega t} - e^{-j\omega t}\right)$ results in the following expression for the Laplace transform:

$$\begin{aligned} L[\sin(\omega t)] &= \frac{1}{2j}\int_0^\infty \left[\left(e^{j\omega t} - e^{-j\omega t}\right)\right]e^{-st}dt = \frac{1}{2j}\left[\int_0^\infty e^{j\omega t}e^{-st}dt - \int_0^\infty e^{-j\omega t}e^{-st}dt\right] \\ &= \frac{1}{2j}\left[\frac{1}{s-j\omega} - \frac{1}{s+j\omega}\right] = \frac{1}{2j}\left[\frac{2j\omega}{s^2-(j\omega)^2}\right] = \frac{\omega}{s^2+\omega^2} \end{aligned} \tag{12.9}$$

where $j^2 = -1$. Using the same approach for the Laplace transform of a cosine wave, we obtain:

$$\begin{aligned} L[\cos(\omega t)] &= \frac{1}{2}\int_0^\infty \left[\left(e^{j\omega t} + e^{-j\omega t}\right)\right]e^{-st}dt = \frac{1}{2}\left[\int_0^\infty e^{j\omega t}e^{-st}dt + \int_0^\infty e^{-j\omega t}e^{-st}dt\right] \\ &= \frac{1}{2}\left[\frac{1}{s-j\omega} + \frac{1}{s+j\omega}\right] = \frac{1}{2}\left[\frac{2s}{s^2-(j\omega)^2}\right] = \frac{s}{s^2+\omega^2} \end{aligned} \tag{12.10}$$

12.4.2 The Inverse Laplace Transform

The inverse $f(t)$ of the Laplace transform $F(s)$ can be obtained from the evaluation of a complex integral:

$$f(t) = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} F(s) e^{st} ds \tag{12.11}$$

Unlike the inverse Fourier transform, the inverse Laplace transform in Eq. (12.11) is rarely used explicitly. Instead, the most common procedure to find the inverse Laplace transform of an expression is a two-step approach (Appendix 12.3):

1. Apply partial fraction expansion to separate the expression into a sum of basic components.
2. Use a lookup table to find the inverse transforms for each basic component.

Examples of this two-stage approach can be found in Section 12.4.3. A direct application of Eq. (12.11) to obtain inverse Laplace transforms is not covered further in this text.

12.4.3 Application to Solving Ordinary Differential Equations

In the following example we will apply the Laplace transform technique to the simplified ***ion channel model*** we introduce in Chapter 13 (Fig. 13.2). The ODE describing this system (see the legend to Fig. 13.2) is:

$$y + RC\frac{dy}{dt} = x \tag{12.12}$$

where R and C are constants corresponding to the membrane resistance and capacitance, respectively. If we probe this system using a unit impulse δ as input x, the output y is the so-called system's unit impulse response h. Transforming each term of the equation into the Laplace domain gives:

$$\begin{aligned} x &= \delta(t) \Leftrightarrow 1 \\ y &= h(t) \Leftrightarrow H(s) \\ \frac{dy}{dt} &= \frac{dh}{dt} \Leftrightarrow sH(s) \end{aligned} \tag{12.13}$$

The $\Leftrightarrow$ symbol indicates the Laplace transform pair.

Note: To represent the differential above we applied Eq. (12.3) and assumed that $h(0) = 0$. Remember that in this case we are really taking the value at 0^-, so we may assume that at the onset of time t the system's output is zero.

Substitution of Eq. (12.13) into Eq. (12.12), and solving for $H(s)$, gives us the transformed ODE:

$$H(s) + RCsH(s) = 1 \rightarrow H(s) = \frac{1}{RC}\frac{1}{s + \frac{1}{RC}} \tag{12.14}$$

Notice that the right-hand side is equivalent to Eq. (12.8) with $a = -\frac{1}{RC}$. This allows us to obtain the inverse transform easily without having to deal with the analytically evaluating inverse transform integral (12.11). Thus the output in the time domain, the impulse response for $t \geq 0$, is:

$$y(t) = h(t) = \frac{1}{RC}e^{-\frac{t}{RC}} \tag{12.15}$$

Notice that arriving at expression (12.15), we simply moved the constant $1/RC$ over from the Laplace domain into the time domain, just as we would treat a constant when evaluating an integral equation. In this example with the unit impulse at the input, finding the inverse was really simple; had we instead chosen the step function $U(t)$ as the input we would have obtained:

$$\begin{aligned} x = U(t) &\Leftrightarrow \frac{1}{s} \\ y &\Leftrightarrow F(s) \\ \frac{dy}{dt} &\Leftrightarrow sF(s) \end{aligned} \tag{12.16}$$

Substitution of these terms in the ODE (Eq. 12.12) gives:

$$F(s) + RCsF(s) = \frac{1}{s} \rightarrow F(s) = \frac{1}{RC}\frac{1}{\left(s + \frac{1}{RC}\right)s} \tag{12.17}$$

Here finding the inverse is slightly more difficult because the denominator of the second factor is a polynomial in s: $(s + 1/RC)s$. This form, where the denominator is a polynomial, is very common, because the general form of the ODE describing an LTI system results in a quotient

of two polynomials (the form shown in Eq. 12.20). As is often the case **partial fraction expansion** must be used to decompose Eq. (12.17) in simpler terms that can be inverse-transformed more readily. Consult Appendix 12.3 if you need to refresh your mathematical skills in partial fraction expansion. Following partial fraction expansion we find that the Laplace transform pair associated with Eq. (12.17) is:

$$F(s) = \frac{1}{RC}\frac{1}{\left(s+\frac{1}{RC}\right)s} = \frac{1}{s} - \frac{1}{s+\frac{1}{RC}} \Leftrightarrow y(t) = 1 - e^{-\frac{t}{RC}}, \quad \text{for } t \geq 0 \tag{12.18}$$

In the chapters on linear filters (LTI systems), the Laplace transform technique is used to solve input−output relationships. Because the filters we consider can be characterized by the general equation for an LTI system, the Laplace transform associated with the differential Eq. (13.1a), can be used to analyze these systems. If we define $X(s)$ and $Y(s)$ as the transforms of the input $x(t)$ and output $y(t)$, respectively, we can transform Eq. (13.1a) into the Laplace domain:

$$\begin{aligned} &A_n s^n Y(s) + A_{n-1}s^{n-1}Y(s) + \cdots + A_0 Y(s) \\ &\quad = B_m s^m X(s) + B_{m-1}s^{m-1}X(s) + \cdots + B_0 X(s) \end{aligned} \tag{12.19}$$

As in Eq. (12.4), here we have also assumed for convenience that all initial values for $x(t)$, $y(t)$, and their derivatives are zero. Further, we assume the input $x(t)$ and its Laplace transform $X(s)$ are known; thus the expression for output $Y(s)$ results in the quotient of two polynomials in which the order *n of* the denominator is typically greater than the order of the numerator m:

$$Y(s) = \frac{B_m s^m X(s) + B_{m-1}s^{m-1}X(s) + \cdots + B_0 X(s)}{A_n s^n + A_{n-1}s^{n-1} + \cdots + A_0} \tag{12.20}$$

As mentioned above, the common approach to finding the inverse of the transformed output $Y(s)$ is to use two steps: partial fraction expansion, followed by looking up the inverse transforms of the individual terms. As demonstrated in Appendix 12.3, we then find $y(t)$ as the combined result of the inverse transforms of the individual terms. The application of this technique will become clear from the examples in the following chapters.

12.5 THE Z-TRANSFORM

In the following text we introduce the z-transform as the equivalent of the Laplace transform for discrete time. Subsequently it will be shown how this procedure can be useful for analyzing difference equations.

12.5.1 The Effect of Delay on the Laplace Transform

In Fig. 12.2 we consider a translated function $f(t)$. The Laplace transform of the function in Fig. 12.2 is $F(s) = \int_0^\infty f(t)e^{-st}dt$, and the Laplace transform of the right-shifted version can be formulated as:

$$L[f(t-\tau)] = \int_\tau^\infty f(t-\tau)e^{-st}dt \tag{12.21}$$

In the above, operator $L[\ldots]$ indicates the Laplace transform. Substituting $T = t - \tau$ (and consequently, $dt = dT$), we get:

$$L[f(t-\tau)] = \int_0^\infty f(T)e^{-s(T+\tau)}dT = e^{-s\tau}\int_0^\infty f(T)e^{-sT}dT = e^{-s\tau}F(s) \tag{12.22}$$

i.e., in general ***a delay τ in the time domain is transformed into the s-domain as a multiplication factor exp(−sτ)***. This result is critical for understanding the z-transform introduced in Section 12.5.2.

12.5.2 Complex Variable z

The z-transform can be considered the equivalent of the Laplace transform for discrete time (sampled) signals. Whereas continuous systems are described by differential equations and analyzed with Laplace (or Fourier) techniques, the equations that relate to discrete signals are difference equations, such as Eq. (13.1b). This type of equation includes terms such as $x(n)$, $x(n-1)$, $h(n-p)$, where n is an integer time index. The complex variable z can be considered as the delay operator $exp(s\tau)$.

Consider a discrete-time/sampled time series:

$$x(n) = x(0)\delta(t) + x(1)\delta(t-\tau) + x(2)\delta(t-2\tau) + \cdots + x(n)\delta(t-n\tau) + \cdots,$$

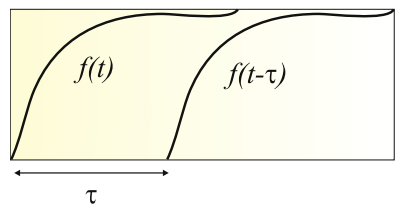


FIGURE 12.2 Function $f(t)$ and a delayed version $f(t-\tau)$.

and the Laplace transform of this series:

$$L[x(n)] = x(0) + x(1)e^{-s\tau} + x(2)e^{-2s\tau} + \cdots + x(n)e^{-ns\tau} + \cdots \tag{12.23}$$

By using the following definition:

$$e^{s\tau} \equiv z \quad \text{or} \quad e^{-s\tau} \equiv z^{-1}, \tag{12.24}$$

we can rewrite Eq. (12.23) as:

$$X(z) = x(0) + x(1)z^{-1} + x(2)z^{-2} + \cdots + x(n)z^{-n} + \cdots \tag{12.25}$$

Note that in the above equation, the constant time difference τ is not explicitly included anymore.

Difference equations describing discrete time systems, such as Eq. (13.1b) usually contain operations such as:

$$\ldots \quad x(n) - x(n-1) \quad \ldots, \tag{12.26}$$

In this example the notation $(n-1)$ is shorthand for a shift of the ***whole*** time series. To perform this shift on time series x, we can shift $x(n)$ one position to the right in order to obtain $x(n-1)$ (Fig. 12.3). In the z-domain, we can now define the expressions in terms of $X(z)$ and z:

$$\begin{array}{lccl} \text{original time series :} & x(n) & \Leftrightarrow & X(z) \\ \text{time series shifted by 1 sample :} & x(n-1) & \Leftrightarrow & z^{-1}X(z) \\ & \cdots & & \cdots \\ & \ldots & & \ldots \\ \text{time series shifted by } a \text{ samples :} & x(n-a) & \Leftrightarrow & z^{-a}X(z) \end{array} \tag{12.27}$$

Using Eq. (12.27), the subtraction in Eq. (12.26) transformed into the z-domain becomes:

$$\ldots \quad X(z) - z^{-1}X(z) \quad \ldots \tag{12.28}$$

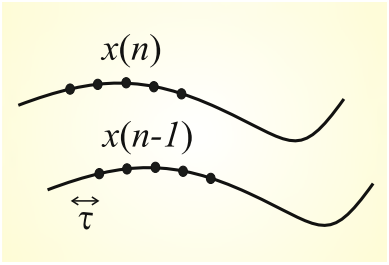


FIGURE 12.3 A right shift of a sampled time series is equivalent to a delay by one sample interval τ.

This procedure is of ***general importance*** because any difference equation describing discrete time LTI systems such as is shown in Eq. (13.1b) can be transformed into the z-domain following the same principle as that illustrated in the example above.

Note on the Lag Operator.
If you have consulted time series analysis in economics texts, you have probably encountered the lag operator (e.g., Hamilton, 1994). This operator symbolized by **L**, not to be confused with the Laplace-transform-operator $L[\ldots]$ as shown in Eq. (12.3), is similar to the z-transform. The difference is that in most signal processing and engineering texts z denotes a variable (Eq. 12.24), while **L** is always considered an operator.

12.6 THE Z-TRANSFORM AND ITS INVERSE

In Eq. (12.27) we introduced the z-transform of $x(n)$ as $X(z)$ without an explicit definition of how to derive $X(z)$ analytically (such as the definitions of the Fourier and Laplace transforms in Eqs. (12.1) and (12.2)). An alternative approach to introducing the z-transform is to create a discrete version of the Laplace transform in Eq. (12.2). Here the integral is replaced by a summation, similar to the step made in going from the continuous Fourier transform to the discrete Fourier transform. Using the same definition for z as above (Eq. 12.24, with t substituted for τ), we can transform a discrete time series $x(n)$ into the z-domain transform $X(z)$:

$$X(z) = \sum_{n=0}^{\infty} x(n) z^{-n} \tag{12.29}$$

The inverse transform is a (counterclockwise) contour integration:

$$x(n) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz \tag{12.30}$$

which we present here for completeness; in the remainder of this text we will not use this contour integral further. Rather, to obtain the inverse transform from the z-domain we will follow the same approach as for the inverse Laplace transform (Section 12.4.2): first we perform partial

fraction expansion (if needed), followed by looking up the inverse transform of each term in a table of z-transforms (see the example in Appendix 12.3).

12.7 EXAMPLE OF THE Z-TRANSFORM

As an example of the z-transform let us consider the algorithm for discrete differentiation of a time series $x(n)$ sampled with an interval Δ. Assume a signal differentiator system that outputs time series y, with y being the single time step differential of the input x. This differential can be ***approximated*** by taking the difference between subsequent samples:

$$y(n) = \frac{x(n) - x(n-1)}{\Delta} \tag{12.31}$$

Using $X(z)$ and $Y(z)$ as the z-transforms of $x(n)$ and $y(n)$, respectively, the z-transform of this difference equation becomes:

$$Y(z) = \frac{X(z) - X(z)z^{-1}}{\Delta} = \frac{X(z)\left(1 - z^{-1}\right)}{\Delta} \tag{12.32}$$

The transfer function of our differentiator can now be determined:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 - z^{-1}}{\Delta} = \frac{z - 1}{z\Delta} \tag{12.33}$$

If we set the interval Δ to one, the differentiator's transfer function becomes:

$$H(z) = \frac{z - 1}{z} = 1 - z^{-1} \tag{12.34}$$

Because we know the transfer function $H(z)$ of the differentiator, we can multiply the z-transform of its input with $H(z)$ to obtain the z-transformed output $Y(z)$: i.e., $Y(z) = X(z) \times H(z)$. Assuming an input a^n for $t \geq 0$ (i.e., $U(n)\, a^n$) with its z-transform: $z/(z - a)$ (see Appendix 12.1, Table A12.1-1B), we get the z-transform of the output as: $(z - 1)/(z - a)$. To find the inverse of this z-domain function, a similar approach as with the Laplace transform is used: separate the expression into basic terms (usually by partial fraction expansion), followed by looking up the solution for each component term in a table. An illustration of this procedure for the inverse transform of $(z - 1)/(z - a)$ with $a = ¼$ is given in Appendix 12.3. Further use and examples of the z-transform can be found in the following chapters, in which digital filters are introduced (e.g., Chapters 16–18).

APPENDIX 12.1

Laplace and z-Transform Tables

The following tables summarize a few Laplace and z-transform pairs, similar to the Fourier transform pairs in Table 6.1 (Chapter 6). In the tables below we multiply the time domain functions with the unit step function U to stress that we are dealing with one-sided transforms in which it is assumed that $x = 0$ for t or $n < 0$.

TABLE A12.1-1 Frequently Used Laplace and z-Transform Pairs

A. LAPLACE TRANSFORM PAIRS

$x(t)$	$X(s)$
$\delta(t)$	1
$U(t)$	$\frac{1}{s}$
$U(t)e^{at}$	$\frac{1}{s-a}$
$U(t)\sin(\omega t)$	$\frac{\omega}{s^2+\omega^2}$
$U(t)\cos(\omega t)$	$\frac{s}{s^2+\omega^2}$

B. z-TRANSFORM PAIRS

$x(n)$	$X(z)$
$\delta(n)$	1
$U(n)$	$\frac{z}{z-1} = \frac{1}{1-z^{-1}}$
$U(n)a^n$	$\frac{z}{z-a} = \frac{1}{1-az^{-1}}$
$U(n)\sin(\omega n)$	$\frac{[\sin(\omega)]z}{z^2-2[\cos(\omega)]z+1}$
$U(n)\cos(\omega n)$	$\frac{z^2-[\cos(\omega)]z}{z^2-2[\cos(\omega)]z+1}$

APPENDIX 12.2

Region of Convergence

Throughout the text we have generally used an optimistic approach with respect to the existence of transforms and convergence of integrals. Because we apply both Laplace and z-transforms as a tool for solving

equations and we use tables to find the transforms and their inverses, we usually do not worry about the domain of existence. For the interested reader we summarize a few comments about the existence of the Laplace and z-transform expressions in this appendix. In order for the transforms to exist, the associated integral/summation must be finite, similar to the Dirichlet conditions for the Fourier transform (Chapter 5, Section 5.3). Especially for functions representing a power such as e^{at} and a^n, the risk of the expressions exploding towards large values for t or n is clearly present. For example in the integral in Eq. (12.8), evaluating the Laplace transform of an exponential function, the term $e^{-(s-a)\infty}$ is zero only if the real part of $(s-a) > 0$. In this case the integral exists, i.e., it evaluates to a finite value. In the case where a in e^{at} is a real number, the condition for the existence of the Laplace transform for e^{at} can be formulated as: $\mathrm{Re}(s) > a$ (Re symbolizing the real part σ of s); the graphical representation of this area in the s-plane is shown in Fig. A12.2-1A. The area satisfying the existence condition for the Laplace transform is called the ***region of convergence* (*ROC*)**. The example of the ROC in Fig. A12.2-1A is clearly only relevant for the exponential function e^{at}; other functions will have a different ROC.

Just as in the Laplace transform one can determine a region where the result of the summation in Eq. (12.29) is finite. This area is the ROC for the z-transform (Fig. A12.2-1B) and as in the Laplace transform, it depends on the function at hand, i.e., $x(n)$. Considering an example where $x(n) = a^n$, we can define the z-transform using Eq. (12.29) as:

$$X(z) = \sum_{n=0}^{\infty} a^n z^{-n} = \sum_{n=0}^{\infty} \left(az^{-1}\right)^n \qquad \text{(A12.2-1)}$$

The summation in Eq. (A12.2-1) is a series that converges only if $|az^{-1}| < 1 \rightarrow |z| > |a|$.

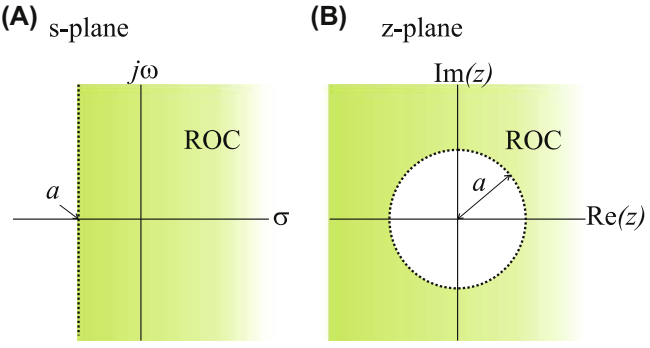


FIGURE A12.2-1 Examples of the region of convergence (ROC) (A) of the Laplace transform of e^{at}, and (B) of the z-transform of a^n.

Note: The convergence statement $|az^{-1}| < 1$ is provided without proof, but it isn't completely counterintuitive since the power of a fraction smaller than one becomes very small for large powers n, whereas the power of a number larger than one grows increasingly large for increasing n.

The z-plane consists (as the s-plane) of real and imaginary components, and z (being a complex exponential, Eq. 12.24) is usually defined in polar coordinates; an example of the ROC for a^n is shown in Fig. A12.2-1B.

APPENDIX 12.3

Partial Fraction Expansion

In finding the ***inverses of both Laplace and z-transforms*** it is often necessary to apply partial fraction expansion. This procedure is based on the fact that a rational function (the quotient of two polynomials), where the order of the numerator is lower than the denominator, can be decomposed into a summation of lower-order terms. The partial fraction expansion will be reviewed in this appendix without further proof.

To illustrate the principle we show the example in Eq. (12.17)

$$F(s) = \frac{1}{RC}\frac{1}{\left(s+\frac{1}{RC}\right)s} \tag{A12.3-1}$$

To focus on the expansion, initially we ignore the constant factor $1/RC$ and focus on a function in the form $\frac{1}{\left(s+\frac{1}{RC}\right)s} = \frac{1}{(s-a)s}$, defining $a = -1/RC$ (note the minus sign) for a simpler notation.

Since the order of the numerator is lower than the denominator, according to the algebra underlying the partial fraction expansion technique, we may state:

$$\frac{1}{(s-a)s} = \frac{A}{s-a} + \frac{B}{s} \tag{A12.3-2}$$

Step 1 is to solve for A by multiplying through by the denominator of the first term on the right-hand side $(s-a)$ and then setting $s = a$ in order to nullify the B coefficient:

$$\frac{1}{s} = A + \frac{B(s-a)}{s} \quad \& \quad s = a \rightarrow A = \frac{1}{s} = \frac{1}{a} \tag{A12.3-3a}$$

Step 2 is to solve for B by multiplying through by the denominator of the second term s and then setting $s = 0$ to eliminate the A term:

$$\frac{1}{s-a} = \frac{As}{s-a} + B \;\;\&\;\; s = 0 \rightarrow B = \frac{1}{s-a} = -\frac{1}{a} \qquad \text{(A12.3-3b)}$$

You probably noticed that in steps 1 and 2 we first multiply the entire equation with expressions from the denominator of the separate terms and then conveniently chose a value that makes that expression zero; a strange trick because it "feels" like division by zero, but it works!

Combining our results with the original expression in Eq. (A12.3-2) we get:

$$\frac{1}{(s-a)s} = \frac{1}{a}\frac{1}{(s-a)} - \frac{1}{a}\frac{1}{s}$$

Substituting $a = -1/RC$ we get:

$$-\frac{RC}{\left(s+\frac{1}{RC}\right)} + \frac{RC}{s}$$

Finally we substitute our result into Eq. (A12.3-1):

$$F(s) = \frac{1}{RC}\frac{1}{\left(s+\frac{1}{RC}\right)s} = \frac{1}{RC}\left[-\frac{RC}{\left(s+\frac{1}{RC}\right)} + \frac{RC}{s}\right] = \frac{1}{s} - \frac{1}{s+\frac{1}{RC}} \qquad \text{(A12.3-4)}$$

The inverse transform of both terms can be obtained easily by inverting the results we obtained in Eqs. (12.7) and (12.8):

$$y(t) = U(t) - U(t)e^{-\frac{t}{RC}} \quad \text{or} \quad y(t) = 1 - e^{-\frac{t}{RC}} \quad \text{for } t \geq 0 \qquad \text{(A12.3-5)}$$

Similar procedures are also commonly applied to find the inverses of the z-transform and the Fourier transform. There is one important condition that must be satisfied for this trick to work. The order of the numerator must be smaller than the order of the denominator! If this is not the case there are two procedures that can be followed:

1. use polynomial division, or
2. divide the entire expression temporarily by s and correct for this division later.

For example, the inverse of $\frac{s-a}{s+a}$ cannot be determined directly because the order of both the numerator and denominator are the same. Using ***the division approach***:

$$\begin{array}{r|l} s+a & 1 \\ & s-a \\ & \underline{s+a} \\ & -2a \end{array} \quad \rightarrow \quad \frac{s-a}{s+a} = 1 - \frac{2a}{s+a} \tag{A12.3-6}$$

These two terms can be easily transformed using the results obtained earlier (note that a = a constant!) in Eqs. (12.6) and (12.8):

$$\frac{s-a}{s+a} = 1 - \frac{2a}{s+a} \Leftrightarrow \delta(t) - 2ae^{-at} \quad \text{for } t \geq 0 \tag{A12.3-7}$$

In the above example it was fairly easy to divide and find the inverse transform, the alternative is to follow the other approach and ***divide by s or z*** (in the case of the z-transform) and correct for this slight-of-hand later. For example, the inverse transform of $\frac{z-1}{z-¼}$ can be found by dividing through by z:

$$\frac{z-1}{z-¼} \underset{\times\frac{1}{z}}{\rightarrow} \frac{z-1}{z(z-¼)} = \frac{A}{z} + \frac{B}{z-¼} \tag{A12.3-8}$$

producing an expression that meets the order condition, and thus allows us to use the same approach for the partial fraction expansion followed in Eq. (A12.3-3).

Step 1 is to multiply by the denominator of the first term z and then set $z = 0$:

$$\frac{z-1}{z-¼} = A + \frac{Bz}{z-¼} \quad \& \quad z = 0 \rightarrow A = \frac{-1}{-¼} = 4 \tag{A12.3-9a}$$

Step 2 is to similarly multiply by the denominator of the second term $(z - ¼)$ and then set $z = ¼$:

$$\frac{z-1}{z} = \frac{A(z-¼)}{z} + B \quad \& \quad z = ¼ \rightarrow B = \frac{¼-1}{¼} = -3 \tag{A12.3-9b}$$

Substituting our findings in (A12.3-9) back into Eq. (A12.3-8) and correcting the result by multiplying it by z:

$$\frac{z-1}{z-¼} \underset{\times\frac{1}{z}}{\rightarrow} \frac{z-1}{z(z-¼)} = 4\frac{1}{z} - 3\frac{1}{z-¼} \underset{\times z}{\rightarrow} 4 - 3\frac{z}{z-¼} \tag{A12.3-10}$$

The form of the expression in Eq. (A12.3-10) can be readily found in standard tables of the z-transform:

$$4 - 3\frac{z}{z - \frac{1}{4}} = 4 - 3\frac{1}{1 - \frac{1}{4}z^{-1}} \Leftrightarrow 4\delta(n) - 3(\tfrac{1}{4})^n U(n) \qquad \text{(A12.3-11)}$$

Here $\delta(n)$ and $U(n)$ are the discrete time versions of the unit impulse and unit step.

EXERCISES

(For the following problems you may use tables of Laplace transforms, z-transforms, and their inverses such as in Appendix 12.1.)

12.1 A causal continuous time LTI system, with input x and output y, is described by the following differential equation:

$$\frac{dy(t)}{dt} + 3y(t) = x(t)$$

a. Find the system's transfer function $H(s)$.
b. Find the system's unit impulse response $h(t)$.
c. Find the output if $x(t) = U(t)[1 + 2e^{-t}]$, with $U(t) =$ Unit step function.

12.2 We have a causal discrete time LTI system; input x and output y are related by:

$$y(n) - \frac{1}{3}y(n-1) = x(n) \quad \text{for } n > 0$$

a. Find the system's transfer function $H(z)$.
b. Use your result above to find the system's output if x is a unit step.

12.3 Use the Laplace transform technique to solve the following equation (in which x and y are functions of time t)

$$\frac{d^2y}{dt^2} + 5\frac{dy}{dt} + 6y = x$$

Given that: $y(0) = 2$, $\left.\frac{dy}{dt}\right|_{t=0} = \dot{y}(0) = 1$, and $x = e^{-t}U(t)$ [$U(t) =$ unit step function].

Further you may assume that: $y(0^-) = y(0)$ and $\dot{y}(0^-) = \dot{y}(0)$.
a. Solve y.
b. Use MATLAB® to plot y versus t.

12.4 Use the z-transform technique to solve the following problem. A discrete LTI system has the following response to a unit step function $U(n)$ at its input.

$$y(n) = 2\left(\frac{1}{3}\right)^n U(n)$$

a. Find the unit impulse response $h(n)$ of this LTI system.
b. What is the system's output when the input is $\left(\frac{1}{2}\right)^n U(n)$.
c. Use MATLAB® to plot y, h, and the output obtained in b versus n.

References

Abramowitz, M., Stegun, I.A., 1975. Handbook of Mathematical Functions. Academic Press, New York.
A table of mathematical functions including a part on Laplace transforms. This book can be downloaded from several websites.

Hamilton, J.D., 1994. Time Series Analysis. Princeton University Press, Princeton, NJ.
An excellent book providing a large overview of analysis of time series and their underlying models. A disadvantage for neuroscientists is that the examples are from economy and finance.

Hsu, H.P., 1995. Signals and Systems. Schaum's Outline Series. McGraw-Hill, New York.
A compilation of examples and exercises relevant for signal analysis.

Northrop, R.B., 2003. Signals and Systems Analysis in Biomedical Engineering. CRC Press, Boca Raton, FL.
An excellent introduction to signal processing and linear systems including a review of basic techniques for solving ordinary differential equations, matrix algebra, and transformations. Some examples of nonlinear systems analysis are included.

CHAPTER

13

LTI Systems: Convolution, Correlation, Coherence, and the Hilbert Transform

13.1 INTRODUCTION

In this chapter we present four important signal processing techniques that are based on linear time-invariant (LTI) systems:

- convolution,
- cross-correlation,
- coherence, and
- the Hilbert transform.

The convolution operation allows us to relate an LTI system's input and output in the time domain. A related technique is calculation of cross-correlation between two different signals or between a signal and itself (called autocorrelation). Coherence is a related type of analysis used to correlate components in the frequency domain. The latter has the advantage that frequency-specific correlations can be determined, whereas cross-correlation in the time domain mainly reflects large-amplitude components. The Hilbert transform provides a very unique approach to determine the instantaneous phase and amplitude of a signal.

We will show that techniques in the time domain have equivalents in the frequency domain and vice versa. For instance, convolution in the time domain is equivalent to multiplication in the frequency domain, and multiplication in the frequency domain corresponds to complex convolution in the time domain. The techniques to describe linear systems can be applied to characterize (the linear aspects of) physiological signals and will be applied in later chapters to develop analog and digital filters

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00013-8

(Chapters 15–18). It is important to realize that the techniques described in this chapter reflect linear relationships and therefore they generally fail when strong nonlinear interactions are involved in a system's dynamics (e.g., Chapter 27).

13.2 LINEAR TIME-INVARIANT SYSTEM

The basic idea of a linear system is that it can be fully characterized by knowledge of its response r to a basic, simple input s (stimulus). If a suitable function is chosen for s, any arbitrary stimulus S can be decomposed into a set of these simple inputs (S = sum of several stimuli s). The defining feature of a linear system is that the compound response R associated with stimulus S is simply the sum of all responses to the set of simple ones: i.e., the system's total reaction R is equal to the sum of the individual responses r to the stimuli s (thus R = sum of all responses r). From the engineering perspective the most basic element of any signal is the unit impulse and therefore the most basic response of LTI systems is the unit impulse response (UIR). From this UIR all behavior of the linear system can be derived. The procedure for deriving this behavior is called convolution.

The systems considered in the remainder of this chapter are called LTI. A system is linear if its output y is linearly related to its input x. Linearity implies that the output to a ***scaled*** version of the input $A \times x$ is equal to $A \times y$. Similarly, if input x_1 generates output y_1 and input x_2 generates y_2, the system's response to the combined input $x_1 + x_2$ is simply $y_1 + y_2$. This property (related to scaling) is called ***superposition***. The ***time-invariant*** part of the LTI system indicates that the system's response does not depend on time, i.e., at different points in time (given the same initial state of the system) such a system's response y to input x is identical: i.e., if $x(t) \rightarrow y(t)$ then $x(t - \tau) \rightarrow y(t - \tau)$. Details of how these scaling and time-invariant properties lead to the fourth property of an LTI system, ***convolution***, is further explained in Section 13.3. A system is considered nonlinear if it violates either of the two properties: scaling or superposition.

In addition to the LTI constraint, we usually deal with ***causal*** systems: i.e., the output is related to previous and/or current input only.

> *Note*: If a system only reacts to current input it is ***memoryless*** or ***static***. If a system's response is (also) determined on a previous or future input, it is ***dynamic***. In reality we usually deal with causal systems whose output depends on previous, but not future, input.

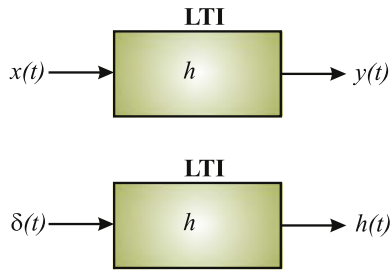


FIGURE 13.1 Linear time-invariant (LTI) system. Input–output relationship. The system's weighting function h and the impulse-response.

In Fig. 13.1, we represent a causal LTI system and show the response of this system to an arbitrary input function and to a unit impulse. The relationship between input $x(t)$ and output $y(t)$ can be described by a (set of) ordinary differential equations (ODEs). A special case of an input–output relationship shown in Fig. 13.1 is the system's response h to a unit impulse δ; as we will demonstrate in Section 13.3, the LTI system's ***weighting function*** or ***impulse response function h*** can be used to link any input to its associated output.

Two examples of LTI systems are shown in Fig. 13.2. The first example is a simple resistor network which attenuates the input x. The other example is a simplified electrical equivalent circuit for a membrane ion channel. The channel is modeled as a battery representing the equilibrium potential of the ion x in series with the channel's conductance $g = 1/R$. The ionic current charges the membrane capacitor C and therefore affects membrane potential y. In the analysis of these systems the relationship between input and output is described mathematically (see input–output in the legend of Fig. 13.2). These types of input–output relationships can all be generalized as:

$$A_n \frac{d^n y(t)}{dt^n} + A_{n-1} \frac{d^{n-1} y(t)}{dt^{n-1}} + \cdots + A_0 y(t) = B_m \frac{d^m x(t)}{dt^m} + B_{m-1} \frac{d^{m-1} x(t)}{dt^{m-1}} + \cdots + B_0 x(t) \tag{13.1a}$$

for continuous time systems and:

$$A_n y(k-n) + A_{n-1} y(k-n+1) + \cdots + A_0 y(k) = B_m x(k-m) + B_{m-1} x(k-m+1) + \cdots + B_0 x(k) \tag{13.1b}$$

for a discrete time system.

These equations link output with input in a generic fashion. In both Eqs. (13.1a and b), usually $n > m$.

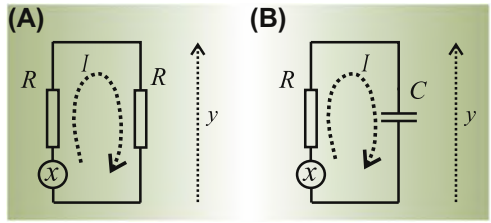


FIGURE 13.2 Two examples of input–output relationships. (A) A voltage divider consisting of two equal resistors. The potential at x is considered the input and the potential y across the second resistor is defined as the output. According to Kirchhoff's second law the potentials in the loop must equal zero; in other words, the potential of x equals the potential drop over both resistors. The drop over the right resistor is equal to the output y. Because there are no branches, the current (I) is equal throughout the loop (Kirchhoff's first law). (B) A similar situation where the resistor is replaced by a capacitor can be considered as a simplified passive membrane model. Upon closing switch S, the ion channel with equilibrium potential x and conductivity $g = 1/R$ discharges over the membrane capacitance C causing a change in the membrane potential y. The following table summarizes the circuit's analysis and the input–output relationship.

	Circuit (A)		**Circuit (B)**
Kirchhoff's second law:	$x = IR + y$	Kirchhoff's second law:	$x = IR + y$
Kirchhoff first and Ohm's laws:	$I = \frac{y}{R}$	Kirchhoff first law and capacitor:	$I = C\frac{dy}{dt}$
Input–Output:	$y = ½x$	**Input–Output:**	$y + RC\frac{dy}{dt} = x$

13.3 CONVOLUTION

13.3.1 Time Domain

13.3.1.1 Continuous Time

A key component in the analysis of linear systems is to relate input and output. The ***unit impulse response h(t)*** formalizes this relationship, and can be considered the system's ***weighting function***. Furthermore, a mathematical operation defined as ***convolution*** determines the output of the LTI with a known unit impulse response for any given input. The general idea is that any input function can be decomposed in a sequence of weighted impulses. Here we follow the procedure developed in Chapter 2 and present an arbitrary input function as a series of unit impulses. More specifically, we use the sifting property from Eq. (2.8) to represent input $x(t)$ as:

$$x(t) = \int_{-\infty}^{\infty} x(\tau)\delta(t-\tau)d\tau \tag{13.2}$$

Note that in Eq. (13.2) we have changed the names of variables relative to those used in Eq. (2.8). The variable t is substituted for Δ from Eq. (2.8), and τ is substituted for t. Further, we used the fact that δ is an even symmetric function: $\delta(t-\tau) = \delta(\tau - t)$.

> *Note*: This change in notation from $(\tau - t)$ in Eq. (2.8) to $(t - \tau)$ in Eq. (13.2) is introduced here so that our expressions are in agreement with the notation that is commonly used for the convolution formalism.

Now by writing the LTI system's input as a set of weighted unit impulses, we can determine the output of the system. The system's response to a weighted impulse $x(\tau) \times \delta(t)$ is equal to $x(\tau) \times h(t)$ (***scaling*** by $x(\tau)$); the system's response to $\delta(t-\tau)$ is equal to $h(t-\tau)$ (***time invariance***). Combining both the scaling and time invariance, the response to a single weighted impulse shifted in time can be characterized as:

$$x(\tau)\delta(t-\tau) \rightarrow x(\tau)h(t-\tau) \tag{13.3}$$

Finally we can relate the system's response $y(t)$ to the input $x(t)$ as the sum, taken to the continuous integral limit, of all responses to the weighted impulses (***superposition***):

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \tag{13.4}$$

In a graphical representation of Eq. (13.4) for each value of $y(t)$ one can consider the product $x(\tau)\,h(t-\tau)$ as the overlap of the two functions $x(\tau)$ and $h(-\tau)$ shifted by an amount equal to t (Appendix 13.1). The convolution integral in Eq. (13.4) is easier to interpret if one realizes that the time scale of the input x is represented by τ and that of the output y (or h if we consider the impulse response) by t. In reality the output y at time t does not depend on the whole input signal x with τ ranging from $-\infty$ to ∞.

- First we do not know the system's input at $\tau = -\infty$ (since we are not old enough). Therefore we usually bring a system into a rest state and we begin to perturb it with some input at a convenient point in time which we define as $\tau = 0$. All input that occurs at $-\infty < \tau < 0$ can therefore be considered zero.
- Second, real systems are usually causal and do not respond to future input at $\tau \rightarrow \infty$: i.e., the impulse response h at time t depends only on current and previous input ($\tau \leq t$), meaning that the entire input signal for $\tau > t$ is irrelevant for the response at time t.

Combining these two considerations, we can change the integration limits in Eq. (13.4) from $-\infty \rightarrow \infty$ to $0 \rightarrow t$:

$$y(t) = \int_0^t x(\tau)h(t-\tau)d\tau = x(t) \otimes h(t) \tag{13.5}$$

The $\otimes$ symbol which we will use throughout this text is often used to denote convolution. Convolution is commutative (Appendix 13.1), so we can also write $y(t)$ as the convolution of the system's impulse-response $h(t)$ with the input $x(t)$:

$$y(t) = h(t) \otimes x(t) = \int_0^t h(\tau)x(t-\tau)d\tau \tag{13.6}$$

An example of the convolution principle is shown in Fig. 13.3. The left column in Fig. 13.3 shows different combinations of weighted unit

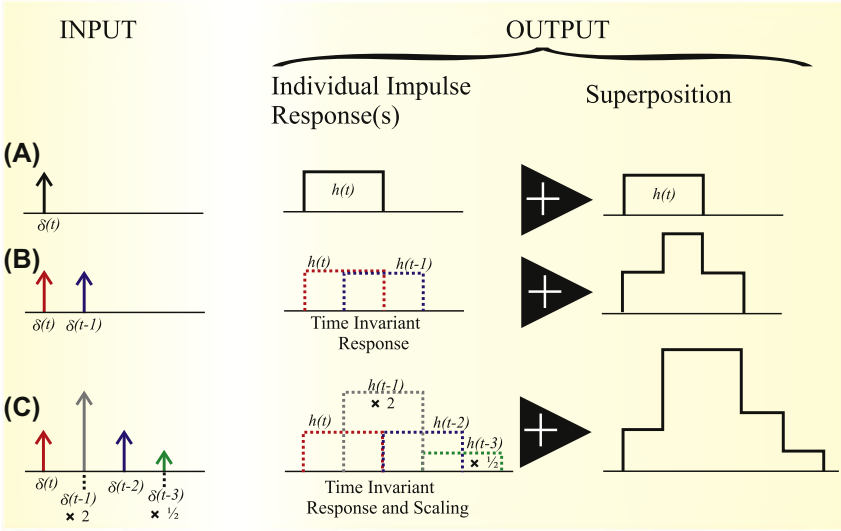


FIGURE 13.3 An example of the response of an linear time invariant system to impulse functions. The example in (A) shows the system's response $h(t)$ to a single δ function. To keep this example simple we have (arbitrarily) chosen a simple square pulse as the impulse response. (B) The sequence of two δ functions creates a compound response. This example shows that (1) the response to each δ function is identical and only shifted in time (time invariance), and (2) that the sum of these two responses $h(t)$ and $h(t-1)$ is the system's total response (superposition) to the combined input. (C) A sequence of four δ functions with different weights shows the same time invariance but also the scaling property: i.e., $\delta(t-1) \times 2$ generates a response $h(t-1) \times 2$ and $\delta(t-3) \times \frac{1}{2}$ generates a response $h(t-3) \times \frac{1}{2}$. The system's response to the whole sequence is the superposition of all individual reactions. Note that scaling is not a separate property; it can be derived directly from superposition.

impulse functions. The second column depicts the individual UIRs resulting from each of these input impulses. For instance, the $\delta(t-1)$ input generates $h(t-1)$ as output; the $\delta(t-1)\times 2$ input generates $h(t-1)\times 2$ as output, etc. Finally, the last column in Fig. 13.3 shows the superposition of the individual responses, corresponding to the convolution of the input with the impulse response function.

13.3.1.2 Discrete Time

The example in Fig. 13.3 shows how one could interpret convolution for discrete events in time (δ functions). Applying the same logic explicitly in discrete time, a system's response can be interpreted in the same manner. Let's consider an example of an LTI system in discrete time using n to index time:

$$y(n) = 0.25x(n) + 0.5x(n-1) + 0.25x(n-2) \tag{13.7}$$

The system's response to a discrete unit impulse would be:

$$\begin{aligned} n &= 0 \quad y(0) = 0.25 \\ n &= 1 \quad y(1) = 0.5 \\ n &= 2 \quad y(2) = 0.25 \\ n &> 2 \quad y(n) = 0 \end{aligned} \tag{13.8}$$

Note that the impulse response in Eq. (13.8) reproduces the weighting coefficients for $x(n)$, $x(n-1)$, and $x(n-2)$ in Eq. (13.7). Knowing the UIR, we can use the procedure depicted in Fig. 13.3 to determine output resulting from an input consisting of multiple impulses.

An example of the convolution procedure, also outlined in Appendix 13.1, can be examined with the following MATLAB® script.

```
% pr13_1.m
% Discrete Convolution

d=1;                              % unit impulse
h=[.25 .5 .25];                   % impulse-response
i=[20 20 20 12 40 20 20];         % input
ii=[20 20 40 12 20 20 20];        % reversed input
x=0:10;                           % x -axis
```

```
% Plot Routines
%-------------------------------
figure
subplot(6,1,1),stem(x(1:length(d)),d)
axis([0 7 0 1.5]);
title(' Unit Impulse')
axis('off')

subplot(6,1,2),stem(x(1:length(h)),h)
axis([0 7 0 1.5]);
title('Impulse-response y=.25x(n)+.5x(n-1)+.25x(n-2)')

subplot(6,1,3),stem(x(1:length(i)),i)
axis([0 7 0 50]);
title(' LTI Input')

subplot(6,1,4),stem(x(1:length(ii)-1),ii(2:length(ii)))
axis([0 7 0 50]);
title(' Reversed LTI Input @ n=5')

subplot(6,1,5),stem(x(1:length(h)),h)
axis([0 7 0 1.5]);
title(' Impulse-response Again')
axis('off');

r5=.25*20+.5*40+.25*12;
subplot(6,1,6),stem(x(1:length(r5)),r5)
axis([0 7 0 50]);
title('Response @n=5 = .25*20+.5*40+.25*12 = 28')
xlabel('Sample #')
```

13.3.2 Frequency Domain

Convolution in the time domain can be a difficult operation, requiring evaluation of the integral in Eq. (13.5) or (13.6); fortunately, in the s- or ω-domain convolution of functions can be simplified to a ***multiplication*** of their transformed versions, i.e.,:

$$x_1(t) \otimes x_2(t) \Leftrightarrow X_1(\omega)X_2(\omega)$$

$$\text{with :} \quad x_1(t) \Leftrightarrow X_1(\omega) \quad \text{and} \quad x_2(t) \Leftrightarrow X_2(\omega)$$

$$\text{i.e.,} \quad F\{x_1(t) \otimes x_2(t)\} = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} x_1(\tau) x_2(t-\tau) d\tau \right] e^{-j\omega t} dt \tag{13.9}$$

$F\{\ldots\}$ denotes the Fourier transform. Now, changing the order of integration:

$$F\{x_1(t) \otimes x_2(t)\} = \int_{-\infty}^{\infty} x_1(\tau) \left[\int_{-\infty}^{\infty} x_2(t-\tau) e^{-j\omega t} dt \right] d\tau \tag{13.10}$$

The expression within the brackets is the Fourier transform of function x_2 shifted by an interval τ. Using $T = t - \tau$ ($\rightarrow t = T + \tau$ and $dt = dT$), this expression can be rewritten as:

$$\int_{-\infty}^{\infty} x_2(t-\tau) e^{-j\omega t} dt = \int_{-\infty}^{\infty} x_2(T) e^{-j\omega(T+\tau)} dT = e^{-j\omega\tau} \underbrace{\int_{-\infty}^{\infty} x_2(T) e^{-j\omega T} dT}_{X_2(\omega)}$$

$$= X_2(\omega) e^{-j\omega\tau}$$

Substituting this result in Eq. (13.10) gives:

$$F\{x_1(t) \otimes x_2(t)\} = X_2(\omega) \int_{-\infty}^{\infty} x_1(\tau) e^{-j\omega\tau} d\tau = X_1(\omega) X_2(\omega) \tag{13.11}$$

Expressing Eq. (13.11) in English: the Fourier transform of the convolution of x_1 and x_2 (left-hand side) equals the product of the transforms X_1 and X_2 (right-hand side).

13.3.3 Complex Convolution

The Fourier transform of a product of two functions x and y in the time domain is $\int_{-\infty}^{\infty} x(t) y(t) e^{-j\omega t} dt$. Defining the Fourier transforms for x and y as X and Y, we can substitute the inverse of the Fourier transform $\frac{1}{2\pi} \int_{-\infty}^{\infty} X(\lambda) e^{j\lambda t} d\lambda$ for x and obtain:

$\int_{-\infty}^{\infty} x(t)y(t)e^{-j\omega t}dt = \int_{-\infty}^{\infty}\left(\frac{1}{2\pi}\int_{-\infty}^{\infty} X(\lambda)e^{j\lambda t}d\lambda\right)y(t)e^{-j\omega t}dt$. Changing the order of integration we can write:

$$= \int_{-\infty}^{\infty}\left(\frac{1}{2\pi}\int_{-\infty}^{\infty} X(\lambda)e^{j\lambda t}d\lambda\right)y(t)e^{-j\omega t}dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\lambda)\left(\int_{-\infty}^{\infty} y(t)e^{-j\omega t}e^{j\lambda t}dt\right)d\lambda$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\lambda)\underbrace{\left(\int_{-\infty}^{\infty} y(t)e^{-j(\omega-\lambda)t}dt\right)}_{Y(\omega-\lambda)}d\lambda = \frac{1}{2\pi}\underbrace{\int_{-\infty}^{\infty} X(\lambda)Y(\omega-\lambda)d\lambda}_{X(\omega)\otimes Y(\omega)}$$

$$= \frac{1}{2\pi}X(\omega)\otimes Y(\omega) \tag{13.12}$$

The expression $\frac{1}{2\pi}X(\omega)\otimes Y(\omega)$ is called the complex convolution, which is the frequency domain equivalent of the product of two functions in the time domain. We have seen this principle applied in Chapters 2 and 7 when evaluating the effects of sampling and truncation of continuous functions (Sections 2.3 and 7.1.1, Figs. 2.6 and 7.5).

13.3.4 Summary of Linear Time-Invariant System's Analysis

Based on the properties described for the LTI system and the material we covered on the Laplace and z-transforms in Chapter 12, we can summarize the different types of analyzing the system's behavior. We need the UIR to explain the system's input−output relationship in the time or spatial domain. In this case we employ the convolution integral (Eq. 13.5) to compute the output from UIR and input. If we transform the input−output relationship into the frequency domain by using the Fourier transform, we use the approach described in Section 13.3.2, i.e., we use the product of the system's frequency response and the Fourier transform of the input to get the Fourier transform of the output. A similar approach can be used in the Laplace domain as we described the examples in Chapter 12. Here we multiply the Laplace transform of the unit impulse, called the transfer function, with the Laplace transform of the input to obtain the Laplace transform of the corresponding output (see Eq. 12.5 and the example in Appendix 12.3). A summary of the relationships between input, LTI system characteristic, and output is given in Table 13.1.

TABLE 13.1 Summary of the Input (x)—Output (y) Relationship for a Linear Time-Invariant System

Domain	Function	Operation
Time/spatial	UIR	Convolution
	$h(t)$, $h(n)$, UIR	$y = x \otimes h$
Frequency/Fourier	Frequency response	Multiplication
	$H(j\omega)$, $H(k)$	$Y(j\omega) = X(j\omega)\, H(j\omega)$
		$Y(k) = X(k)\, H(k)$
Laplace/symbolic	Transfer function	Multiplication
	$H(s)$, $H(z)$	$Y(s) = X(s)\, H(s)$
		$Y(z) = X(z)\, H(z)$

UIR, the unit impulse response h; $\otimes$, denotes the convolution procedure; t and n indicate continuous and discrete time (or space); H, X, and Y denote the transforms of h, x, and y, respectively.

13.4 AUTOCORRELATION AND CROSS-CORRELATION

13.4.1 Time Domain

13.4.1.1 Continuous Time

Correlation between two time series or between a single time series and itself is used to find dependency between samples and neighboring samples. One could correlate, for instance, a time series with itself by plotting x_n versus x_n; it will be no surprise that this would result in a normalized correlation equal to 1. Formally the autocorrelation R_{xx} of a process x is defined as:

$$R_{xx}(t_1, t_2) = E\{x(t_1)x(t_2)\} \tag{13.13}$$

Here the times t_1 and t_2 are arbitrary moments in time, and the autocorrelation demonstrates how a process is correlated with itself at these two different times. If the process is stationary, the underlying distribution is invariant over time and the autocorrelation therefore only depends on the offset $\tau = t_2 - t_1$:

$$R_{xx}(\tau) = E\{x(t)x(t+\tau)\} \tag{13.14}$$

Further if we have an ergodic process (as defined in Chapter 3) we may use a time average to define an autocorrelation function over the domain τ indicating a range of temporal offsets:

$$R_{xx}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_0^T x(t)x(t+\tau)dt \text{ or } R_{xx}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t)x(t+\tau)dt \tag{13.15}$$

In some cases, where the process at hand is not ergodic or if ergodicity is in doubt, one may use the term "time autocorrelation function" for the expression in Eq. (13.15). These functions can be normalized to a range between -1 and 1 by dividing the end result by the variance of the process.

Applying a similar approach as in the autocorrelation above, the cross-correlation R_{xy} between two time series x and y can be defined:

$$R_{xy}(t_1, t_2) = E\{x(t_1)y(t_2)\} \tag{13.16}$$

If the processes are stationary, the underlying distributions are invariant over time and only the difference $\tau = t_2 - t_1$ is relevant:

$$R_{xy}(\tau) = E\{x(t)y(t+\tau)\} \tag{13.17}$$

Assuming ergodicity we can use a time average such that:

$$R_{xy}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_0^T x(t)y(t+\tau)dt \text{ or } R_{xy}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t)y(t+\tau)dt \tag{13.18}$$

Note that the correlation function as defined above may include direct current (DC) components if present in the component functions, if this component (the mean μ) is removed we obtain the ***covariance function***, i.e.:

$$C_{xx}(t_1, t_2) = E\{[x(t_1) - \mu][x(t_2) - \mu]\} \tag{13.19}$$

As with the Fourier transform (Chapter 6) in Eq. (6.4) where we defined the transform in the limit of c_n with the period $T \to \infty$, we can define the correlation integral using Eqs. (13.15) and (13.18) as a starting point. In this definition (just as in Eq. 6.4), we remove the $1/T$ factor and obtain:

$$z(\tau) = \int_{-\infty}^{\infty} x(t)y(t+\tau)dt \tag{13.20}$$

In the case where $y = x$ in the above integral, $z(\tau)$ represents the autocorrelation function. If the signals are demeaned, the integral in Eq. (13.20) is the covariance function.

13.4.1.2 Discrete Time

For a sampled time series x of a stationary and ergodic process, we can define the autocorrelation function R_{xx} in a similar fashion as in continuous time:

$$R_{xx}(n_1, n_2) = E\{x(n_1)x(n_2)\} \rightarrow R_{xx}(m) = E\{x(n)x(n+m)\} \tag{13.21}$$

Here the indices n_1, n_2, n, and m indicate samples in the time series. If we replace this expression with a time average:

$$R_{xx}(m) = E\{x(n)x(n+m)\} = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{n=N} x(n)x(n+m) \tag{13.22}$$

Similarly for the cross-correlation function in discrete time we obtain:

$$R_{xy}(m) = E\{x(n)y(n+m)\} = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{n=N} x(n)y(n+m) \tag{13.23}$$

In real signals we can maximize the epoch length from $-N$ to N in order to increase the accuracy of our correlation estimate, but it will (of course) always be a finite interval.

13.4.1.3 Example

We use Eq. (13.7) to generate a time series and estimate the autocorrelation function of y given that x is a zero mean random variable with zero mean and a variance equal to one. For this example we modify the equation with the original numerical values replaced by coefficients a, b, and c:

$$y(n) = ax(n) + bx(n-1) + cx(n-2) \tag{13.24}$$

Because we know the underlying generator (Eq. 13.24) and the probability function that characterizes the nature of the input x, we can use $E\{y(n)y(n+m)\}$ (Eq. 13.21) to analytically determine the autocorrelation function of the time series for different temporal lags.

For lag $m = 0$:

$$E\{y(n)y(n)\} = E\left\{(ax(n) + bx(n-1) + cx(n-2))^2\right\} \tag{13.25}$$

In the evaluation of the above expression, the expectation $E\{x(n)x(m)\} = 0$ (because input x is a zero mean random variable) for all $n \neq m$. However, for equal indices the expectation evaluates to the variance of the random input $E\{x(n)x(n)\} = \sigma^2$. Therefore Eq. (13.25) evaluates to:

$$E\left\{a^2x(n)^2 + b^2x(n-1)^2 + c^2x(n-2)^2\right\} = \left(a^2 + b^2 + c^2\right)\sigma^2 \tag{13.26}$$

For $m = 1$:

$$\begin{aligned} E\{y(n)y(n+1)\} = E\{&(ax(n) + bx(n-1) + cx(n-2))(ax(n+1) \\ &+ bx(n) + cx(n-1))\} \end{aligned} \tag{13.27}$$

Using the same properties as above (where $E\{x(n)x(m)\} = 0$ and $E\{x(n)x(n)\} = \sigma^2$), we can simplify Eq. (13.27) to:

$$(ab + bc)\sigma^2 \tag{13.28}$$

For $m = 2$:

$$\begin{aligned} E\{y(n)y(n+2)\} = E\{&(ax(n) + bx(n-1) + cx(n-2))(ax(n+2) \\ &+ bx(n+1) + cx(n))\} \end{aligned} \tag{13.29}$$

this simplifies to:

$$ac\sigma^2 \tag{13.30}$$

For all $m > 2$:

$E\{y(n)y(n+m)\}$ evaluates to zero. For instance at $m = 3$ one obtains:

$$\begin{aligned} E\{y(n)y(n+3)\} = E\{&(ax(n) + bx(n-1) + cx(n-2))(ax(n+3) \\ &+ bx(n+2) + cx(n+1))\} = 0 \end{aligned} \tag{13.31}$$

For the particular values we used in Eq. (13.7) ($a = 0.25$, $b = 0.5$, and $c = 0.25$) and the random process x with zero mean and unit variance, we obtain the autocorrelation values in Table 13.2 (second column). It is common to normalize the autocorrelation to reflect a value of one at zero lag, thereby preserving the mathematical relationship with nontime series statistics where the correlation of a data set with itself is necessarily unitary (third column in Table 13.2).

The outcome of the expectations summarized in Table 13.2 can be validated numerically against a time series produced using Eq. (13.7) with a random input. It must be taken into account that this approach will give only estimates of the expected values in Table 13.2 based on the particular output of the random number generator.

TABLE 13.2 Autocorrelation of $y(n) = 0.25x(n) + 0.5x(n-1) + 0.25x(n-2)$ for Different Lags m

Lag	$E\{y(n)\,y(n+m)\}$	Normalized: Divide by $E\{y(n)^2\}$
$m = 0$	6/16 Eq. (13.26)	1.00
$m = 1$	4/16 Eq. (13.28)	0.67
$m = 2$	1/16 Eq. (13.30)	0.17
$m > 2$	0 Eq. (13.31)	0.00

The following script is a MATLAB® routine to validate these analytically obtained autocorrelation values using a random input to generate time series y.

```
% pr13_2.m
% autocorrelation
clear;
le=10000;
x=randn(le,1);                  % input

y(1)=0.25*x(1);
y(2)=0.25*x(2)+0.5*x(1);

for i=3:le;
   y(i)=0.25*x(i)+0.5*x(i-1)+0.25*x(i-2);
end;

tau=-(le-1):(le-1);
c=xcov(y,'coef');          % normalized
                           % autocorrelation
figure;
stem(tau,c);
title('Autocorrelation ');
xlabel('Lag');
ylabel('Correlation (0-1)');
axis([-10 10 -.1 1.1]);
```

13.4.2 Frequency Domain

A similar approach as that discussed for convolution in Section 13.3.2 can be used to relate ***auto- and cross-correlation*** to the power spectrum. In this approach the correlation function can be denoted:

$$z(t) = \int_{-\infty}^{\infty} x(\tau)h(t+\tau)d\tau \tag{13.32}$$

Note that symbols τ and t for the delay and time variables are interchanged with respect to Eq. (13.20). By doing this, we can express the Fourier transform of $z(t)$ in a manner similar to the procedure used for convolution in the explanation in Section 13.3.2, i.e.,

$$\int_{-\infty}^{\infty} z(t)e^{-j\omega t}dt = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} x(\tau)h(t+\tau)d\tau \right] e^{-j\omega t}dt \tag{13.33}$$

Assuming again that we may change the order of integration:

$$Z(\omega) = \int_{-\infty}^{\infty} x(\tau) \left[\int_{-\infty}^{\infty} h(t+\tau)e^{-j\omega t}dt \right] d\tau \tag{13.34}$$

The term in between the brackets is the Fourier transform of function h with a time shift τ. It can be shown that such a shift in the time domain corresponds to multiplication by a complex exponential in the frequency domain, i.e.,

$$\int_{-\infty}^{\infty} h(t+\tau)e^{-j\omega t}dt = H(\omega)e^{j\omega\tau} \tag{13.35}$$

Substitution into the equation for $Z(\omega)$ gives:

$$Z(\omega) = \int_{-\infty}^{\infty} x(\tau)H(\omega)e^{j\omega\tau}d\tau = H(\omega)\int_{-\infty}^{\infty} x(\tau)e^{j\omega\tau}d\tau \tag{13.36}$$

The integral can be decomposed using the Euler's formula as:

$$\int_{-\infty}^{\infty} x(\tau)e^{j\omega\tau}d\tau = \int_{-\infty}^{\infty} x(\tau)\cos(\omega\tau)d\tau + j\int_{-\infty}^{\infty} x(\tau)\sin(\omega\tau)d\tau \tag{13.37}$$

while the Fourier transform of $x(\tau)$ is given by:

$$X(\omega) = \int_{-\infty}^{\infty} x(\tau)e^{-j\omega\tau}d\tau = \int_{-\infty}^{\infty} x(\tau)\cos(\omega\tau)d\tau - j\int_{-\infty}^{\infty} x(\tau)\sin(\omega\tau)d\tau \quad (13.38)$$

Comparing the two equations above, one can see that the two expressions are complex conjugates, therefore $\int_{-\infty}^{\infty} x(\tau)e^{j\omega\tau}d\tau = X^*(\omega)$. Using this in the equation for $Z(\omega)$ one obtains:

$$Z(\omega) = H(\omega)X^*(\omega) \quad (13.39)$$

Eq. (13.39) finally shows that cross-correlation in the time domain equates to a multiplication of the transform of one function with the complex conjugate of the transform of the other function. If H and X are the same function, Z is the frequency transform of autocorrelation function. Also note that the product of the Fourier transform with its complex conjugate is also the definition of the ***power spectrum*** (the unscaled version, see Chapters 5 and 6). This property is also known as the Wiener–Khinchin theorem: i.e., the power spectrum of a function and its autocorrelation are related by the Fourier transform. The power spectrum of x is by definition a real valued function (i.e., XX^*). The autocorrelation function of x, being the inverse transform of the power spectrum, is therefore an even function (to review these relationships, see examples and concluding remarks in Chapter 5, Section 5.4). A MATLAB® demonstration of the Wiener–Khinchin theorem can be found in pr13_3.

The equivalence of cross-correlation in the frequency domain is an important property that will be used in the evaluation of LTI systems such as linear filters. As an application of this technique, we will show in Chapter 17 that the autocorrelation and cross-correlation can be used to determine a filter's weighting function. In Chapter 20 we apply the correlation technique to spike trains, and an example of the use of cross-correlation to describe functional connectivity in spiking networks can be found in Suresh et al. (2016).

Note that because the Fourier transform X of a real even signal is also real (without an imaginary component) and even, its complex conjugate X^* equals X. Therefore convolution (Eq. 13.11) and correlation (Eq. 13.39) are identical for time series that are real and even.

13.5 COHERENCE

The ***coherence*** C between two signals x and y is defined as the ***cross-spectrum*** S_{xy} normalized by the power spectra S_{xx} and S_{yy}. To make the coherence a dimensionless number between 0 and 1, S_{xy} is squared, i.e.,:

$$C(\omega) = \frac{\left|S_{xy}(\omega)\right|^2}{S_{xx}(\omega)S_{yy}(\omega)} \tag{13.40}$$

In many applications, the square root of the above expression is used as the amplitude coherence. Note that S_{xy} in the numerator of Eq. (13.40) will usually be a complex function, whereas S_{xx} and S_{yy} are both real functions. Because we want a real-valued function to express correlation at specific frequencies, we take the magnitude $|S_{xy}|$ of the complex series. If we calculate the normalized cross-spectrum as a complex number for a single frequency and a single trial, the outcome always has magnitude 1 and phase angle ϕ. For instance, if we define:

$$X(\omega) = a + bj,$$

and

$$Y(\omega) = c + dj,$$

we can write the expressions in Eq. (13.40) as:

$$\begin{aligned}
&S_{xx}(\omega) = X(\omega)X^*(\omega) = (a+bj)(a-bj) = a^2 + b^2,\\
&S_{yy}(\omega) = Y(\omega)Y^*(\omega) = (c+dj)(c-dj) = c^2 + d^2, \text{ and}\\
&|S_{xy}(\omega)|^2 = |X(\omega)Y^*(\omega)|^2 = |(a+bj)(c-dj)|^2 = |(ac+bd) - j(ad-bc)|^2
\end{aligned} \tag{13.41}$$

Because S_{xy} is a complex number, the magnitude squared is the sum of the squares of the real and imaginary parts; in this case:

$$= (ac+bd)^2 + (ad-bc)^2 = a^2c^2 + b^2d^2 + a^2d^2 + b^2c^2 \tag{13.42}$$

Substituting the results in Eqs. (13.41) and (13.42) into Eq. (13.40) shows that computation of the coherence of an individual epoch always results in one:

$$C(\omega) = \frac{a^2c^2 + b^2d^2 + a^2d^2 + b^2c^2}{(a^2+b^2)(c^2+d^2)} = 1. \tag{13.43}$$

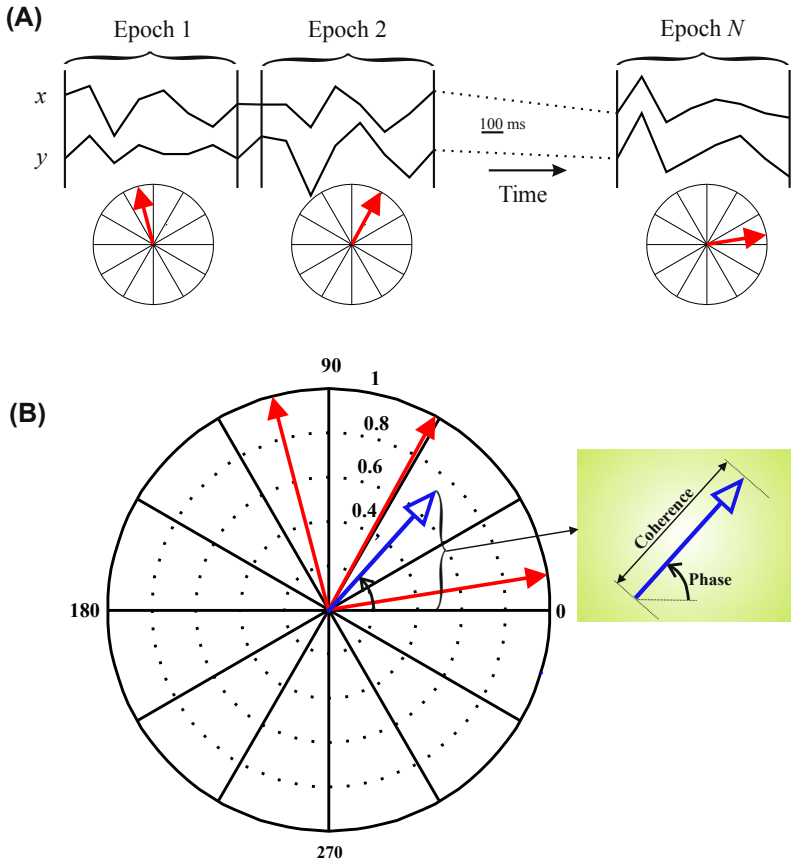


FIGURE 13.4 Example of computing coherence. (A) Two time series x and y are segmented into epochs of equal length. For each epoch the normalized cross-spectrum is determined. The resulting normalized cross-spectral values for a frequency of 2.5 Hz are depicted as a *red arrow* on the unit circle under each of the epochs. (B) The complex numbers indicated by red vectors in the complex plane representing the different values for the normalized cross-spectrum obtained from the three samples are now shown in the same plot. The *blue arrow* represents the average of these three results (so here we simplified the example by assuming that we only have three epochs rather than N). The magnitude of this average is the amplitude-coherence (often referred to as simply coherence), the phase is the phase-coherence. From this diagram it can be appreciated that phase coherence only has a meaning if the amplitude has a significant value. The data were generated by script pr13_4.

This property of giving unity values can be observed in Fig. 13.4A, where the normalized value obtained from one pair of signals x and y is plotted for Epochs 1, 2, and N. In practice the coherence is estimated by averaging over several epochs as well as frequency bands: i.e., the

quantity S_{xy} is determined by averaging over n epochs, indicated by $\langle ... \rangle_n$ in Eq. (13.44):

$$C(\omega) = \frac{\left| \langle S_{xy}(\omega) \rangle_n \right|^2}{\langle S_{xx}(\omega) \rangle_n \langle S_{yy}(\omega) \rangle_n} \tag{13.44}$$

Note that the averaging of cross-spectrum S_{xy} occurs before the absolute value is taken. A common beginner's mistake is to average the absolute value in Eq. (13.43); in this case the outcome is always one!

Thus, when we determine $C(\omega)$ for a single frequency ω over different samples out of an ensemble, we obtain several vectors on the unit circle, typically with different phase angles for each sample (Fig. 13.4A). Next the average over the epochs is computed following Eq. (13.44)—this procedure is depicted in Fig. 13.4B. The magnitude of the sum of the individual vectors (the blue arrow in Fig. 13.4B) indicates the degree of coherence, and the resulting phase angle is the ***phase coherence***. It must be noted here that phase coherence must always we judged in conjunction with the magnitude of the vector; if, for example, the sum of the individual vectors is close to zero, indicating a low level of coherence, the associated phase angle has no real meaning.

***An example of how to determine the coherence can be found in MATLAB® file pr13_4.m. Here we calculate the coherence both explicitly from the spectral components and with the standard MATLAB® routine* mscohere.**

```
% pr13_4
% CoherenceStudy

clear;
N=8;
SampleRate=10;
t=[0 .1 .2 .3 .4 .5 .6 .7];

% Three Replications of Two Signals x and y
x1=[3 5 -6 2 4 -1 -4 1];
x2=[1 1 -4 5 1 -5 -1 4];
x3=[-1 7 -3 0 2 1 -1 -2];
```

```
y1=[-1 4 -2 2 0 0 2 -1];
y2=[4 3 -9 2 7 0 -5 1];
y3=[-1 9 -4 -1 2 4 -1 -5];
f=SampleRate*(0:N/2)/N;                % Frequency Axis

% Signals Combined
X=[x1 x2 x3];
Y=[y1 y2 y3];
T=[0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2 2.1 2.2 2.3];

overlap=0;
cxy=mscohere(X,Y,boxcar(N),overlap,N);   % direct MATLAB®
                                         % command for
                                            coherence
                                         % boxcar(8) is a
                                         % Rectangular Window

% FFTs
fx1=fft(x1);
fx2=fft(x2);
fx3=fft(x3);

fy1=fft(y1);
fy2=fft(y2);
fy3=fft(y3);

%Power and Cross Spectra individual trials
Px1x1=fx1.*conj(fx1)/N;
Px2x2=fx2.*conj(fx2)/N;
Px3x3=fx3.*conj(fx3)/N;
MeanPx=mean([Px1x1', Px2x2', Px3x3']');        % Average the Trials

Py1y1=fy1.*conj(fy1)/N;
Py2y2=fy2.*conj(fy2)/N;
Py3y3=fy3.*conj(fy3)/N;
MeanPy=mean([Py1y1', Py2y2', Py3y3']');        % Average the Trials

Px1y1=fx1.*conj(fy1)/N;
Px2y2=fx2.*conj(fy2)/N;
Px3y3=fx3.*conj(fy3)/N;
MeanPxy=mean([Px1y1', Px2y2', Px3y3']');       % Average the Trials

% Calculate the Coherence, the abs command is to get
% the Magnitude of the Complex values in MeanPxy
```

```
C=(abs(MeanPxy).^2)./(MeanPx.*MeanPy);

% Plot the Results
figure
plot(f,C(1:5),'k');
hold;
plot(f,cxy,'r*');
title(' Coherence Study red* MATLAB routine')
xlabel(' Frequency (Hz)')
ylabel(' Coherence')

figure;
for phi=0:2*pi;polar(phi,1);end;            % Unit Circle
hold;

% Plot the individual points for the second frequency 2.5 Hz
plot(Px1y1(3)/sqrt((Px1x1(3)*Py1y1(3))),'r*')
plot(Px2y2(3)/sqrt((Px2x2(3)*Py2y2(3))),'r*')
plot(Px3y3(3)/sqrt((Px3x3(3)*Py3y3(3))),'r*')

% Plot the average
plot(MeanPxy(3)/sqrt((MeanPx(3)*MeanPy(3))),'k*');
title(' For individual Frequencies (e.g. here 2.5 Hz) all Three Points
(red) are on the Unit Circle, The Average black =<1')
```

13.5.1 Interpretation of the Coherence Values

Eq. (13.44) shows that a coherence estimate is an average obtained from cross-spectral and spectral data. Therefore this coherence estimate, like any average, is associated with a confidence interval ε that depends on the number of observations N included in the average. In addition, it can be shown that the confidence interval depends on the level of coherence C itself (see pp. 333–336 in Bendat and Piersol, 2000):

$$\varepsilon = \frac{\sqrt{2}(1-C)}{\sqrt{CN}} \tag{13.45}$$

For example, if we used 100 observations to estimate the coherence between two signals at ½, the 95% confidence interval becomes $\varepsilon = \frac{\sqrt{2}(1-½)}{\sqrt{½\times 100}} = 0.1$ so that the true value of C lies between 0.4 and 0.6.

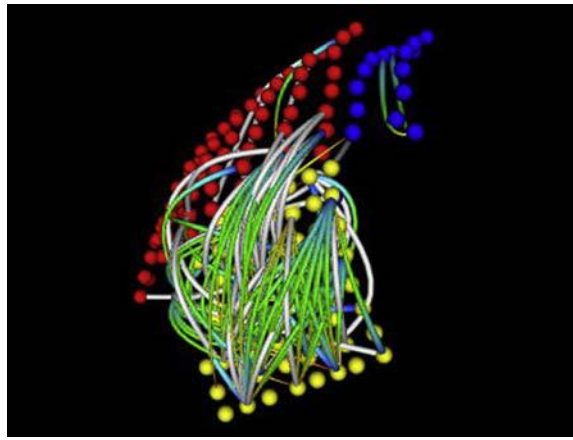


FIGURE 13.5 An example of coherence calculations associated with subdural electrode arrays implanted over the frontal cortex (red and blue, 1 cm spacing) and temporal cortex (yellow, 5 mm spacing) of a patient with medically intractable epilepsy. The *colored pipes* indicate pairs of electrodes with unusually high coherence between them. *White pipes* are not associated with a phase shift. *Green pipes* indicate a phase delay at the *blue end* of the pipe. These data were obtained as part of the surgical evaluation of the patient, who received a temporal lobectomy for treatment of their seizures. *From V.L. Towle with permission.*

13.5.2 Application of Coherence to an Electroencephalogram

An important hypothesis in neuroscience is that connectivity in the brain can be analyzed by determining the temporal relationships between activity patterns in different brain regions. In studies where propagation of a well-defined temporal feature plays a significant role (such as propagating epileptic spikes), the preferred method is cross-correlation. However, if the relationship is based on similarity between background activity at different locations, the coherence metric is frequently applied (e.g., Towle et al., 1999). An example of a pattern of coherence across brain regions for a frequency band of 0.5–4.0 Hz is shown in Fig. 13.5. Each dot represents the position of a cortical electrode, and the width of the interconnecting pipes denotes the level of coherence between the signals generated at those electrodes.

13.6 THE HILBERT TRANSFORM

One of the current frontiers in neuroscience is marked by our lack of understanding of neuronal network function. A first step in unraveling network activities is to record from multiple neurons and/or networks simultaneously. A question that often arises in this context is which signals lead or lag; the underlying thought here is that the signals that

lead cause the signals that lag. Although this approach is not full proof, since one can only make reasonable inferences about causality if all connections between and activities of the neuronal elements are established, it is a first step in analyzing network function. Multiple techniques to measure lead and lag can be used. The simplest ones are cross-correlation and coherence. A rather direct method to examine lead and lag is to determine the phase of simultaneously recorded signals. If the phase difference between two signals is not too big, one considers signal 1 to lead signal 2 if the phase of signal 1 (ϕ_1) is less than the phase of signal 2 (ϕ_2): $\phi_1 < \phi_2$. Of course, this procedure should be considered as a heuristic approach to describe a causal sequence between the components in the network activity since there is no guarantee that a phase difference reflects a causal relationship between neural element 1 (generating signal 1) and neural element 2 (generating signal 2). In this example one could easily imagine alternatives where neural elements 1 and 2 are both connected to a source causing both signals or where element 2 is connected to element 1 via a significant number of relays; in both alternatives the condition $\phi_1 < \phi_2$ might be satisfied without a causal relationship from element 1 to element 2. A frequently used technique to compute a signal's phase is the Hilbert transform which will be described in the remainder of this chapter. An alternative approach to study causality in multichannel data is discussed in Chapter 14.

The Hilbert transform is a tool to determine the amplitude and instantaneous phase of a signal. Here, we first define the transform before discussing the underlying mathematics. An easy way of introducing the application of the Hilbert transform is by considering Euler's formula multiplied with a constant A:

$$Ae^{j\omega t} = A[\cos(\omega t) + j\sin(\omega t)] = A\cos(\omega t) + jA\sin(\omega t) \tag{13.46}$$

In this example we consider the first term in Eq. (13.46), $f(t) = A\cos(\omega t)$ as the **signal** under investigation. This signal is ideal to demonstrate the Hilbert transform application because, in this example, we can see that the amplitude of $f(t)$ is A, and its instantaneous phase ϕ is ωt. The terminology for the Hilbert transform is as follows: the imaginary component, the second term, in Eq. (13.46) $\widetilde{f}(t) = A\sin(\omega t)$ is defined as the **Hilbert transform** of $f(t)$ (we will discuss further details in Sections 13.6.1 and 13.6.2), and the sum of both the signal and its Hilbert transform multiplied by j generates a complex signal:

$$f_a(t) = Ae^{j\omega t} = A\cos(\omega t) + jA\sin(\omega t) = f(t) + j\widetilde{f}(t)$$

in which $f_a(t)$ is defined as the **analytic signal**.

To summarize, the real part of the analytic signal is the signal under investigation $f(t)$ and its imaginary component is the Hilbert transform

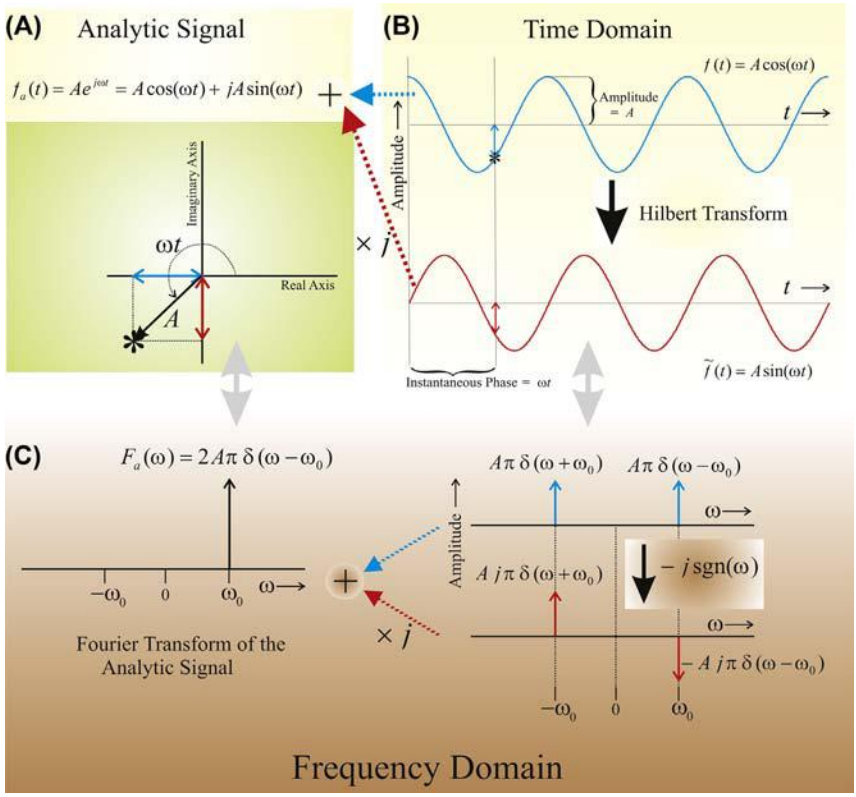


FIGURE 13.6 The signal amplitude A and instantaneous phase ωt of point * of the cosine function (panel (B) $f(t)$, blue) can be determined with the so-called analytic signal (green panel (A)). The analytic signal consists of a real part equal to the signal under investigation (the cosine) and an imaginary component (the sine). The imaginary component (red) is the Hilbert transform $\tilde{f}(t)$ of the signal $f(t)$. The frequency domain equivalents of the cosine wave, the sine wave, the Hilbert transform procedure, and the analytic signal are shown in panel (C). See text for further explanation.

$\tilde{f}(t)$ of the signal. The analysis procedure is summarized in Fig. 13.6. As can be seen in Fig. 13.6A and B, we can use the analytic signal $Ae^{j\omega t}$ to determine amplitude A and instantaneous phase ωt of any point such as that indicated with *. The amplitude is:

$$A = \sqrt{\text{real component}^2 + \text{imaginay component}^2}$$

and the phase is:

$$\phi = \tan^{-1}\left[\frac{\text{imaginary component}}{\text{real component}}\right].$$

Again, in this example we didn't need the analytic signal to determine phase and amplitude for our simple cosine signal, but our finding may be generalized to other signals where such a determination is less trivial.

13.6.1 The Hilbert Transform in the Frequency Domain

As can be seen in the example depicted in Fig. 13.6, the Hilbert transform can be considered as a phase shift operation on $f(t)$ to generate $\tilde{f}(t)$. In our example the signal $\cos(\omega t)$is shifted by $-\pi/2$ rad (or -90 degrees) to generate its Hilbert transform $\cos(\omega t - \pi/2) = \sin(\omega t)$. We may generalize this property, and define a Hilbert transformer as a phase shifting (LTI) system generating the Hilbert transform of its input (Fig. 13.7A). The generalization of the property associated with the cosine is not such a far stretch if you recall that—with the real Fourier series—any periodic signal can be written as the sum of sinusoidal signals (that is the cosine and sine waves in Eq. 5.1) and that our results above can be applied to each of these sinusoidal components.

To further define the shifting by the Hilbert transformer in Fig. 13.7A, we begin to explore this operation in the frequency domain, because here

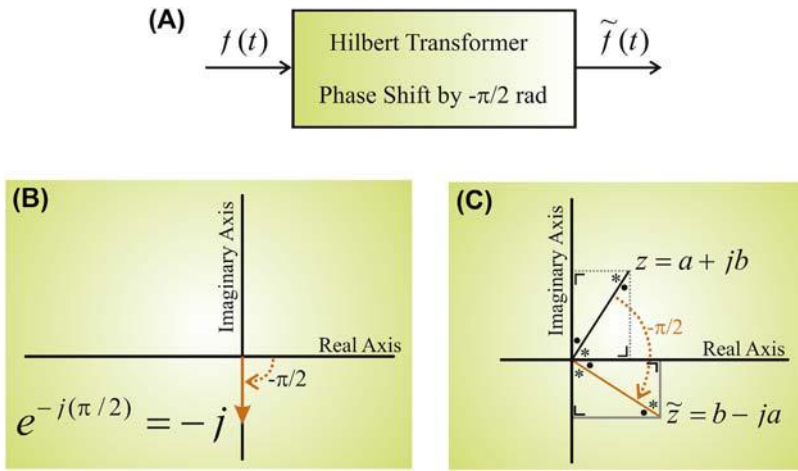


FIGURE 13.7 (A) The Hilbert transform can be represented as the operation of an linear time-invariant system (the Hilbert transformer). Input $f(t)$ is transformed into $\tilde{f}(t)$ by shifting it by $-\pi/2$ rad (-90 degrees). (B) The Hilbert transform operation in the frequency domain can be represented as a multiplication with $e^{-j(\pi/2)} = -j$ (*orange arrow*). (C) Example of the Hilbert transform in the frequency domain, that is multiplication of a complex number $z = a + jb$ with $-j$. The result is $\tilde{z} = b - ja$. As can be seen the result is a -90 degree rotation. Note that in this panel 90 degree angles are indicated by ∟ and that the angles indicated by • and * add up to 90 degrees.

the procedure of shifting the phase of a signal by $-\pi/2$ rad is relatively easy to define as a multiplication by $e^{-j(\pi/2)} = -j$ (Fig. 13.7B). If this is not obvious to you, consider the effect of this multiplication for any complex number $z = e^{j\phi}$ (representing phase ϕ) which can also be written as the sum of its real and imaginary parts $z = a + jb$. Multiplication by $-j$ gives its Hilbert transform $\tilde{z} = -j(a + jb) = b - ja$, indeed corresponding to a -90 degree rotation of z (see Fig. 13.7C). Although the multiplication with $-j$ is correct for the positive frequencies, a -90 degree shift for the negative frequencies in the Fourier transform (due to the negative values of ω) corresponds to multiplication with $e^{j(\pi/2)} = j$. Therefore the operation of the Hilbert transform in the frequency domain can be summarized as:

$$\boxed{\text{multiplication by } \; -j\text{sgn}(\omega)} \tag{13.47}$$

Here we use the so-called signum function sgn (Fig. A13.2-1) defined as:

$$\text{sgn}(\omega)\begin{cases} -1 & \text{for} \quad \omega < 0 \\ 0 & \text{for} \quad \omega = 0 \\ 1 & \text{for} \quad \omega > 0 \end{cases} \tag{13.48}$$

For further details on this function, see Appendix 13.2.

Let us go back to our phase-shifting system depicted in Fig. 13.7A. We now define its UIR as $h(t)$ and its associated frequency response as $H(\omega)$. Within this approach, the Hilbert transform is the convolution of input $f(t)$ with $h(t)$. Using our knowledge about convolution (Section 13.3.2), we can also represent the Hilbert transform in the frequency domain as the product of $F(\omega)$—the Fourier transform of $f(t)$—and $H(\omega)$. This is very convenient because we just determined above that the Hilbert transform in the frequency domain corresponds to a multiplication with $-j\,\text{sgn}(\omega)$. To summarize, we now have the following three Fourier transform pairs:

$$\begin{aligned} &\text{System's Input} \Leftrightarrow \text{Fourier Transform}: && f(t) \Leftrightarrow F(\omega) \\ &\text{System's Unit Impulse Response} \Leftrightarrow \text{Fourier Transform}: && h(t) \Leftrightarrow H(\omega) \\ &\text{Hilbert Transform} \Leftrightarrow \text{Fourier Transform}: && f(t) \otimes h(t) \Leftrightarrow F(\omega)H(\omega) \end{aligned} \tag{13.49}$$

Using these relationships we may state that the Fourier transform of the UIR of the Hilbert transformer is:

$$\boxed{H(\omega) = -j\text{sgn}(\omega)} \tag{13.50}$$

We can use the expression we found for $H(\omega)$ to examine the above example of Euler's equation $Ae^{j\omega_0 t} = \overbrace{\underbrace{A\cos(\omega_0 t)}_{\text{signal}} + \underbrace{j\ A\sin(\omega_0 t)}_{\text{Hilbert Transform}}}^{\text{Analytical Signal}}$ in the frequency domain. The Fourier transform of the cosine term (using Eq. 6.13) is:

$$A\pi[\delta(\omega+\omega_0)+\delta(\omega-\omega_0)] \tag{13.51}$$

The Fourier transform of the cosine's Hilbert transform is the product of the Fourier transform of the signal—the cosine—and $H(\omega)$, that is:

$$\begin{aligned}&\{A\pi[\delta(\omega+\omega_0)+\delta(\omega-\omega_0)]\}\{-j\text{sgn}(\omega)\}\\&A\pi[\delta(\omega+\omega_0)(-j\text{sgn}(\omega))+\delta(\omega-\omega_0)(-j\text{sgn}(\omega))]\end{aligned} \tag{13.52}$$

Because $\delta(\omega+\omega_0)$ is only nonzero for $\omega=-\omega_0$ and $\delta(\omega-\omega_0)$ is only nonzero for $\omega=\omega_0$ we may rewrite the $j\text{sgn}(\omega)$ factors in Eq. (13.52) and we get:

$$A\pi[\delta(\omega+\omega_0)(-j\text{sgn}(-\omega_0))+\delta(\omega-\omega_0)(-j\text{sgn}(\omega_0))]$$

Now we use the definition of sgn, $\text{sgn}(-\omega_0)=-1$ and $\text{sgn}(\omega_0)=1$ (Eq. 13.48) and we simplify to

$$A\pi[\delta(\omega+\omega_0)(j)+\delta(\omega-\omega_0)(-j)] = Aj\pi[\delta(\omega+\omega_0)-\delta(\omega-\omega_0)] \tag{13.53}$$

As expected, Eq. (13.53) is the Fourier transform of $A\text{sin}(\omega t)$ (see Eq. 6.14) which is indeed the Hilbert transform $\widetilde{f}(t)$ of $f(t)=A\text{cos}(\omega t)$.

Combining the above results, we can find the Fourier transform of the analytic signal $f_a(t)=A\cos(\omega t)+jA\sin(\omega t)=f(t)+j\widetilde{f}(t)$. If we define the following pairs:

$$\begin{aligned}f_a(t) &\Leftrightarrow F_a(\omega)\\ f(t) &\Leftrightarrow F(\omega)\\ \widetilde{f}(t) &\Leftrightarrow \widetilde{F}(\omega)\end{aligned}\quad ,$$

the above expressions can be combined in the following Fourier transform pair:

$$f_a(t)=f(t)+j\widetilde{f}(t) \Leftrightarrow F_a(\omega)=F(\omega)+j\widetilde{F}(\omega)$$

In the above equation we substitute the expressions for $F(\omega)$ from Eq. (13.51) and $\widetilde{F}(\omega)$ from Eq. (13.53) and get:

$$F_a(\omega) = F(\omega) + j\widetilde{F}(\omega)$$

$$= \overbrace{\underbrace{A\pi[\delta(\omega+\omega_0)+\delta(\omega-\omega_0)]}_{\text{Fourier Transform of Signal}} + j\underbrace{\{Aj\pi[\delta(\omega+\omega_0)-\delta(\omega-\omega_0)]\}}_{\text{Fourier Transform of Hilbert Transform}}}^{\text{Fourier Transform of Analytical Signal}}$$

With a bit of algebra we obtain:

$$\boxed{F_a(\omega) = F(\omega) + j\widetilde{F}(\omega) = 2\pi A\delta(\omega-\omega_0)} \tag{13.54}$$

This interesting finding shows that the Fourier transform of the analytic signal has zero energy at negative frequencies and only a peak at $+\omega_0$. The peak's amplitude at $+\omega_0$ is double the size of the corresponding peak in $F(\omega)$. Also, this finding may be generalized, that is "**the Fourier transform of the analytic signal $F_a(\omega)$ has no energy at negative frequencies $-\omega_0$, it only has energy at positive frequencies $+\omega_0$ and its amplitude is double the amplitude $+\omega_0$ at in $F(\omega)$.**"

13.6.2 The Hilbert Transform in the Time Domain

From the frequency response presented in Eq. (13.50) and the relationship between convolution in the time and frequency domains (Section 13.3.2), we know that the UIR $h(t)$ of the Hilbert transformer (Fig. 13.7A) is the inverse Fourier transform of $-j\,\text{sgn}(\omega)$. You can find details of $\text{sgn}(t)$ and its Fourier transform in Appendix 13.2; using the signum's Fourier transform, we can apply the duality property (Section 6.2.1) to determine the inverse Fourier transform for $-j\text{sgn}(\omega)$. For convenience we repeat the duality property:

$$\text{if } f(t) \Leftrightarrow F(\omega), \quad \text{then } F(t) \Leftrightarrow 2\pi f(-\omega) \tag{13.55a}$$

Applying this to our signum function (see also Appendix 13.2), we can define the inverse Fourier transform of $\text{sgn}(\omega)$:

$$\text{sgn}(t) \Leftrightarrow \frac{2}{j\omega}, \quad \text{therefore } \frac{2}{jt} \Leftrightarrow 2\pi\underbrace{\text{sgn}(-\omega)}_{-\text{sgn}(\omega)} \tag{13.55b}$$

Note that we can substitute $-\text{sgn}(\omega) = \text{sgn}(-\omega)$ because the signum function (Fig. A13.2-1) has odd symmetry (see Chapter 5, Appendix 5.2).

Using the result from applying the duality property in Eq. (13.55b), we can determine the inverse Fourier transform for the frequency response of the Hilbert transformer $H(\omega) = -j\text{sgn}(\omega)$ and find the corresponding UIR $h(t)$. Because 2π and j are both constants we can multiply both sides with j and divide by 2π; this generates the following Fourier transform pair:

$$\boxed{h(t) = \frac{1}{\pi t} \Leftrightarrow H(j\omega) = -j\text{sgn}(\omega)} \tag{13.56}$$

In Eq. (13.50) we found that the frequency response of the Hilbert transformer is $-j\,\text{sgn}(\omega)$. Because we know that multiplication in the frequency domain is equivalent to convolution in the time domain, we can use the result in Eq. (13.56) to define the Hilbert transform $\widetilde{f}(t)$ of signal $f(t)$ in both the time and frequency domains. We define the following Fourier transforms pairs.

$$\begin{aligned} \text{the input:} \quad & f(t) \Leftrightarrow F(\omega), \\ \text{the Hilbert transform of the input:} \quad & \widetilde{f}(t) \Leftrightarrow \widetilde{F}(\omega), \\ \text{and the unit impulse response of the Hilbert transformer:} \quad & h(t) \Leftrightarrow H(\omega). \end{aligned}$$

Using the above and Eq. (13.56), the Hilbert transform and its frequency domain equivalent are:

$$\widetilde{f}(t) = f(t) \otimes \underbrace{h(t)}_{\frac{1}{\pi t}} = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{f(t)}{t-\tau} d\tau \Leftrightarrow \widetilde{F}(\omega) = F(\omega)H(\omega) \tag{13.57}$$

There is a problem with our finding for the Hilbert transform expression in Eq. (13.57), which is that there is a pole for $\frac{f(t)}{t-\tau}$ within the integration limits at $t = \tau$. The solution to this problem is to define the Hilbert transform as:

$$\widetilde{f}(t) = \frac{1}{\pi} CPV \int_{-\infty}^{\infty} \frac{f(t)}{t-\tau} d\tau \tag{13.58}$$

in which *CPV* indicates the Cauchy principal value of the integral. The *CPV* is a mathematical tool to evaluate integrals that include poles within the integration limits. I give an example of such an application in Appendix 13.3, for those with further interest in the *CPV* procedure I refer to a general mathematics text such as Boas (1966).

13.6.3 Examples

The Hilbert transform is available in MATLAB® via the hilbert command. **Note** that this command produces the analytic signal $f(t) + j\tilde{f}(t)$ and not the Hilbert transform itself: that is, the Hilbert transform is the imaginary component of the output.

You can evaluate the example from Eq. (13.46) by computing the Hilbert transform for the cosine and plot the amplitude and phase. Type the following in the MATLAB® command window (or use pr13_5.m):

```
step=0.00001;                          % step size = 1/sample rate
t=0:step:1;                            % timebase
x=cos(2*pi*4*t);                       % 4 Hz signal
xa=hilbert(x);                         % compute the analytic signal
Amplitude=abs(xa);                     % amplitude of the signal
Phase=atan2(imag(xa),real(xa));        % instantaneous phase
Ohmega=diff(Phase)/(2*pi*step);        % instantaneous frequency
                                         in Hz

figure;plot(t,x,'k');hold;
plot(t,Amplitude,'r');
plot(t,Phase,'g');
plot(t(1:length(t)-1),Ohmega,'m.')
axis([0 1 -5 5])
```

You will obtain a graph of a 4-Hz cosine function with an indication of its amplitude (a constant) in red, its instantaneous phase in green (note that we use the atan2 MATLAB® command in the above example because we want to obtain phase angles between $-\pi$ and $+\pi$), and the frequency as the derivative of the phase in magenta.

You can now check the frequency characteristics we discussed above by computing the Fourier transforms and plotting these in the same graph.

```
X=fft(x);          % Fourier transform of the signal
XA=fft(xa);        % Fourier transform of the analytic signal
figure;plot(abs(X),'k');hold;plot(abs(XA))
```

If you use the **zoom** function of the graph to study the peaks in the plot you will see that the peaks for the positive frequencies (far left part of the graph) show a difference of a factor two between the Fourier transform of

the analytic signal and the Fourier transform of the signal. The negative component (in the discrete version of the Fourier transform this is on the far right part of the graph) shows only a peak in the Fourier transform of the signal. Both observations are as expected, from the theoretical considerations in Section 13.6.1.

Another property to check out is the phase shift between the signal and its Hilbert transform. This can be accomplished by typing the following lines.

```
figure; hold;
plot(t,imag(xa),'r');     % the imaginary part of the analytic signal =
                          % the Hilbert transform
plot(t,x,'k.')            % the signal
plot(t,real(xa),'y')      % real part of the analytic signal = signal
```

Now you will get a figure with the signal (4-Hz cosine wave) in both black (thick line) and yellow (thin line); the Hilbert transform (the 4-Hz sine wave) is plotted in red.

Finally we will apply these techniques to an example in which we have two neural signals, one signal generated by a single neuron and one signal generated by the network in which the neuron is embedded. Our question here is how the phases of these two signals relate. First, to study the signal's low-frequency component that is representative for the spike firing rate, the raw extracellular trace is sent through a low-pass filter. Note that the Hilbert transform technique of using the analytic signal to find the instantaneous phase requires a signal composed of a narrow band of frequencies (see Pikovsky et al. (2001) for more details). Next, for the cellular activity, we compose the raster plot of the intracellularly recorded spike times and send it through the same low-pass filter we used for the extracellular signal. We now have two signals representing the low-pass filtered spiking behavior of the cell and network (see Fig. 13.8A(iii)). We use the Hilbert transform technique to find the instantaneous phase of each signal (Fig. 13.8B(i)). For our case, we are interested in how the phases of the cellular and network signals are related. To find this relationship, we calculate the difference between the two instantaneous phase signals at each point in time and then use this information to generate a histogram (see Fig. 13.8B(ii) and (iii)). This method has been used to compare how the phases of cellular and network signals are related for different types of cellular behavior (Martell et al., 2008).

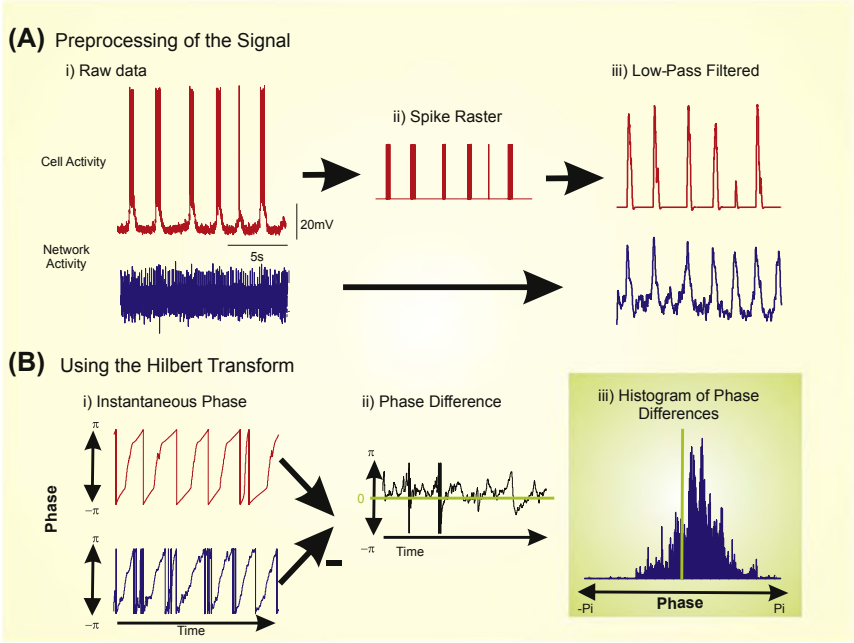


FIGURE 13.8 (A) Processing of a cellular and network activity signal into a low-frequency index of spiking activity (see text for details). (B) The low-pass filtered signals of A were transformed using the analytic signal technique to find the instantaneous phase over time (i). The relationship between the two signals was investigated by finding the difference between the phases over time (ii) and plotting these phase differences in a histogram (iii). *From A. Martell with permission.*

APPENDIX 13.1

Here we consider an example of the application of convolution to relate the input and output of an LTI system by using its unit impulse response. In the example of Fig. A13.1-1 we use input $x(t) = U(t)$, the unit step function, and unit impulse response function $h(t) = e^{-t}$. The output $y(t) = 1 - e^{-t}$ can be obtained by ***convolution***: $x(t) \otimes h(t)$ or $h(t) \otimes x(t)$.

The convolution depicted in Fig. A13.1-1B can be obtained by using Eq. (13.5):

$$y(t) = \int_0^t U(\tau)h(t-\tau)d\tau = \int_0^t e^{-(t-\tau)}\,d\tau = e^{-t}\int_0^t e^{\tau}d\tau = e^{-t}[e^{\tau}]_0^t$$
$$= e^{-t}(e^t - 1) = 1 - e^{-t} \qquad \text{(A13.1-1)}$$

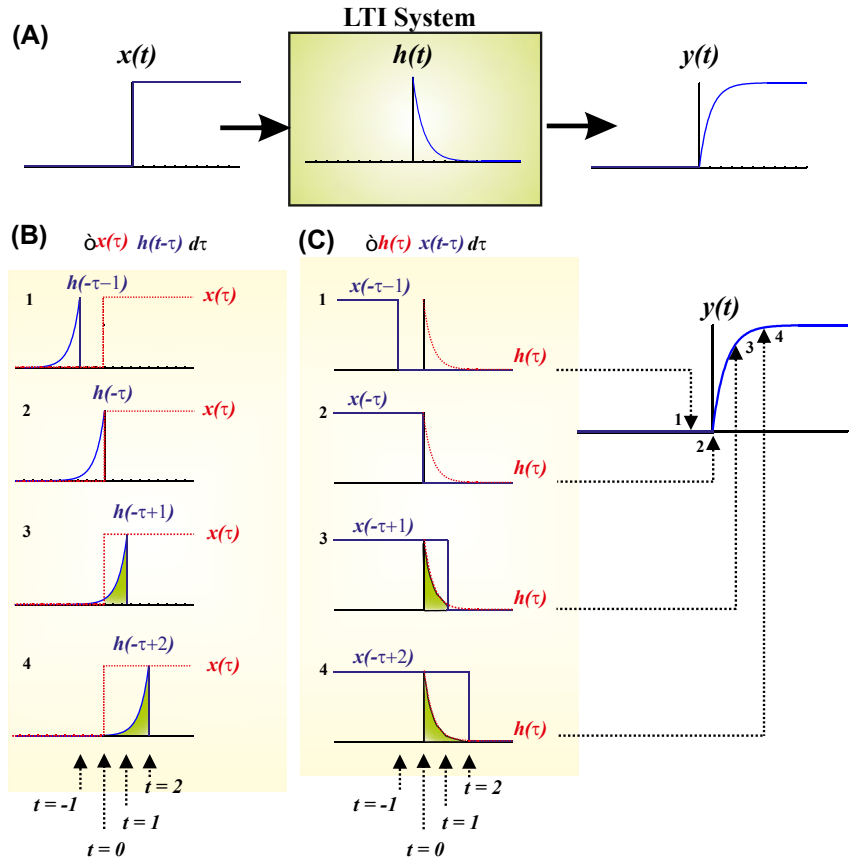


FIGURE A13.1-1 Graphical representation of the convolution process used to relate input and output functions of a linear time-invariant system. In this example, the input $x(t)$ is the unit step $U(t)$, the unit impulse response function $h(t)$ is e^{-t} for $t \geq 0$, and the output $y(t)$ is $1 - e^{-t}$ for $t \geq 0$. The convolution operation shifts the inverted impulse function along the input, and the area under the combined functions at time t is the output $y(t)$. It can be seen in (B) that for $t < 0$ there is no overlap and the output y is therefore zero. For $t = 1$ and $t = 2$ there is overlap and the area is indicated in green. This example also shows that for $t = 1$ integration limits can be established between $\tau = 0$ and $\tau = 1$, and for $t = 2$ the limits move from $0 \rightarrow 2$; more generally the integration limits of the convolution integral required to determine y at time t are $0 \rightarrow t$. Comparing (B) and (C) shows that convolution is commutative.

Using Eq. (13.6) for the convolution depicted in Fig. A13.1-1C and applying the commutative property, we obtain the same result:

$$y(t) = \int_0^t h(\tau)U(t-\tau)d\tau = \int_0^t e^{-\tau}d\tau = \left[-e^{-\tau}\right]_0^t = 1 - e^{-t} \qquad \text{(A13.1-2)}$$

APPENDIX 13.2

This appendix describes the signum function sgn(t), its derivative, and Fourier transform. The signum function is 1 for positive t and -1 for negative t (Fig. A13.2-1).

Similar to the derivative of the unit step function $U(t)$ (Section 2.2.2, Fig. 2.4A), the derivative of this function is only nonzero at $t = 0$. The only difference is that for sgn(t) the function goes up by 2 units instead of 1 unit in $U(t)$. Since the derivative of the unit step is $\delta(t)$, the derivative of the signum function yields an impulse strength 2, that is:

$$\frac{d[\text{sgn}(t)]}{dt} = 2\delta(t) \tag{A13.2-1}$$

The Fourier transform of the derivative of a function is equal to the Fourier transform of that function multiplied by $j\omega$. This property is similar to the relationship of the Laplace transform of a derivative of a function and the Laplace transform of the function itself; see Section 12.3 (Eq. 12.3). If we now use this property and define the Fourier transform of sgn(t) as $S(\omega)$ we can apply the Fourier transform to both sides of Eq. (A13.2-1):

$$j\omega S(\omega) = 2 \tag{A13.2-2}$$

Recall that the Fourier transform of the unit impulse is 1 (see Section 6.2.1, Eq. 6.9). Therefore, the Fourier transform pair for the signum function is:

$$\text{sgn}(t) \Leftrightarrow S(\omega) = \frac{2}{j\omega} \tag{A13.2-3}$$

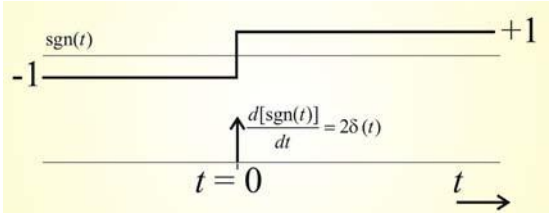


FIGURE A13.2-1 The signum function and its derivative, the unit impulse function with an amplitude of two.

APPENDIX 13.3

In Eq. (13.58) we use the Cauchy principal value *CPV*. This technique is used to approach integration of a function that includes a pole within the integration limits. We will not go into the mathematical details (for more on this subject please see a mathematics textbook such as Boas, 1966) but we will give an example to show the principle. For example, consider the integral $\int_{-d}^{d} \frac{1}{x} dx$. The function $1/x$ in this integral has a pole (is unbounded) at $x = 0$. The Cauchy principal value technique approximates the integral as the sum of two separate integrals:

$$\int_{-d}^{d} \frac{1}{x} dx \approx \int_{-d}^{-\varepsilon} \frac{1}{x} dx + \int_{\varepsilon}^{d} \frac{1}{x} dx$$

where ε is a small value approaching zero (via positive values of ε) at the same speed. In this case the two integrals cancel and approach $\int_{-d}^{d} \frac{1}{x} dx$. Our final result can be summarized as:

$$CPV \int_{-d}^{d} \frac{1}{x} dx = \lim_{\varepsilon \to 0^+} \left[\int_{-d}^{-\varepsilon} \frac{1}{x} dx + \int_{\varepsilon}^{d} \frac{1}{x} dx \right] = 0$$

Here the Cauchy principal value is indicated by *CPV*; in other texts you may also find *PV* or *P*.

EXERCISES

13.1 Describe the relationship between cross-correlation and coherence.

13.2 In MATLAB®, generate two time series (10,000 points each) with functions:

$$y_1(n) = x(n) + x(n-1) \text{ and}$$
$$y_2(n) = 0.5x(n) + x(n-1) + 3x(n-2)$$

Use the MATLAB® command randn to generate input x.
a. Compute the mean and variance of x, y_1, and y_2.
b. Compute the cross-correlation between y_1 and y_2 (use the xcov command) and plot the result to show the correlation function for lags ranging from −6 to 6.

c. Calculate the cross-correlation between y_1 and y_2 based on $E\{y_1(n)\, y_2(n+m)\}$, with m being the lag (use the equations for y_1 and y_2). Plot the results you obtain in (c) in the same graph as the results of (b).
d. Interpret the results obtained in (b) and (c).

13.3 Consider a discrete time LTI system with input $x(n)$ and output $y(n)$. It responds to a unit impulse input, i.e., $x(1) = 1$ and 0 (zero) for all other n. Its response is $y(1) = 0.7$, $y(2) = 0.3$, and 0 (zero) for all other n.
[Note that following MATLAB® convention n starts at 1, and not at 0.]
a. What is the difference equation governing the system? [Hint: See Section 13.3.1.2 and pr13_1.]
b. Compute the theoretical (expected) autocorrelations of x and y and the theoretical (expected) cross-correlation between x and y. [Hint: See Section 13.4.1.3.]
c. Simulate the system in MATLAB®; now use a random input x and compute the power spectrum of y.
d. Use the above simulation to also compute the autocorrelations of x and y, and the cross-correlation between x and y.
e. Use the above simulation to also compute the coherence between x and y.
[Hint: for the simulations you may use pr13_2 and pr13_4 as starting point.]

13.4 Review the text in Section 13.5 and answer the following questions.
a. What is the coherence value for a single epoch?
b. What is the result of averaging the coherence values across multiple epochs?
c. Why is the result in (b) different from the individual coherence values determined across the same epochs?
d. Which answer is the correct coherence value, the result in (b) or in (c)?
[Explain why.]

13.5 Given the signal $f(t) = A_1\cos(\omega_1 t) + A_2\cos(\omega_2 t)$,
a. What is the expression for its Hilbert transform $\widetilde{f}(t)$?
b. What is the expression for its associated analytic signal $f_a(t)$?
c. Is the Hilbert transformation $f(t)$ to $\widetilde{f}(t)$ a linear one?
[Explain your answer using superposition and scaling properties.]

d. Is the transformation $f(t)$ to $f_a(t)$ a linear one? [Explain your answer using superposition and scaling properties]
e. Are the transformations in (c) and (d) time-invariant? [Explain why.]

References

Bendat, J.S., Piersol, A.G., 2000. Random Data Analysis and Measurement Procedures. John Wiley & Sons, New York.

Boas, M.L., 1966. — Mathematical Methods in the Physical Sciences, second ed. John Whiley & Sons.
An excellent textbook on basic mathematical techniques.

Martell, A., Lee, H., Ramirez, J.M., Van Drongelen, W., 2008. Phase and frequency synchronization analysis of NMDA-induced network oscillation. In: P142, CNS 2008 Annual Meeting. http://www.biomedcentral.com/content/pdf/1471-2202-9-s1-p142.pdf.

Pikovsky, A., Rosenblum, M., Kurths, J., 2001. Synchronization: A Universal Concept in Nonlinear Sciences. Cambridge University Press, Cambridge, UK.

Suresh, J., Radojicic, M., Pesce, L., Bhansali, A., Wang, J., Tryba, A.K., Marks, J.D., van Drongelen, W., 2016. Network burst activity in hippocampal neuronal cultures: the role of synaptic and intrinsic currents. J. Neurophysiol. 115, 3073–3089.

Towle, V.L., Carder, R.K., Khorashani, L., Lindberg, D., 1999. Electrocortico-graphic coherence patterns. J. Clin. Neurophysiol. 16, 528–547.

CHAPTER

14

Causality

14.1 INTRODUCTION

It is common practice to decompose multichannel data into its components (e.g., Chapters 7 and 28) to detect structure in complex data sets. Another question that is often posed concerns the causal structure between channels or components: that is, does one channel generate another? In neuroscience the underlying question is: does one area in the brain activate others? A typical example is when an epileptologist examines multichannel recordings of brain electrical activity and attempts to find the source (focus) from where epileptic seizures originate. Often this task is accomplished by finding the signals that lead or lag; the leading signals are then considered as causing the lagging ones. Cross-correlation or nonlinear equivalents (such as mutual information) can be used to formalize and quantify timing differences between signal pairs in multichannel data sets (Chapter 13).

We have to start pessimistically by pointing out that translation from lead-lag to causality is not strictly possible—the example in Fig. 14.1 demonstrates this. If we record from areas A and B in Fig. 14.1A, our method of interpreting lead-lag as a causal relationship A → B is correct. However, if we measure signals from A, B, and C (Fig. 14.1B), we conclude that A → B, A → C, and B → C. We are only partly correct: the two former relationships are correctly inferred but the latter is not. It would get even worse if we hadn't recorded from A in this example: then we only find B → C and we would be 100% incorrect. So equating lead-lag with causality/connectivity can be incorrect. Having said this, in many studies in neuroscience this is (conveniently) ignored and timing in signals is frequently used as an argument for connectivity. Often, authors use terms such as "functional connectivity" or "synaptic flow" to (implicitly) indicate the caveats above. Because we often know the typical conduction velocity and delays caused by synaptic transmission, we can

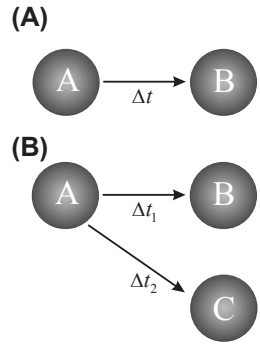


FIGURE 14.1 Example of areas in the brain—A, B, C—and their connections indicated by *arrows*. The delays, due to conduction velocity in the connections, between these areas are Δt, Δt_1, Δt_2 with $\Delta t_2 > \Delta t_1$. (A) Due to the delay, activity in area B lags relative to the activity in A. (B) Both B and C lag relative to A but since $\Delta t_2 > \Delta t_1$, activity in C also lags relative to B.

(at least) recognize unrealistic delays. For example, if we know that areas B and C in Fig. 14.1B are 10 cm apart and that conduction velocities of the fibers between B and C are ~1 m/s, we can expect delays ~100 ms plus a few milliseconds for each synapse involved. Now suppose that in this example the delay between B and C ($\Delta t_2 - \Delta t_1$) is ~5 ms, such a value is far below the expected delay of over 100 ms, which is an indication that direct connectivity doesn't play a role in the observed lead-lag between B and C.

14.2 GRANGER CAUSALITY

In the 1950s, Norbert Wiener proposed that one signal causes another if your prediction of the latter signal improves by including knowledge of the former. About a decade later, Clive Granger (1969) formalized this concept for linear regression of stochastic processes. To explain the principle we consider an example of two signals x and y, and we suppose that we can characterize them with the following autoregressive (AR) models

$$x_n = a_1 x_{n-1} + a_2 x_{n-2} + \ldots + a_i x_{n-i} + \ldots + b_1 y_{n-1} + \ldots + b_i y_{n-i} + \ldots + e_{x_n} \tag{14.1a}$$

$$y_n = c_1 y_{n-1} + c_2 y_{n-2} + \ldots + c_i y_{n-i} + \ldots + d_1 x_{n-1} + \ldots + d_i x_{n-i} + \ldots + e_{y_n} \tag{14.1b}$$

Here $a_{\ldots}$, $b_{\ldots}$, $c_{\ldots}$, and $d_{\ldots}$ are the model coefficients and e_{x_n} and e_{y_n} are error terms. If the variance of error e_{x_n} is reduced by including any value

of y, y_{n-i} (that is $b_i \neq 0$), then we say that y causes x. Similarly, if the variance of error e_{y_n} is reduced by including any value of x, x_{n-i} (that is $d_i \neq 0$), then we say that x causes y. In this example, if $b_i \neq 0$ and/or $d_i \neq 0$, we may use the term Granger causality to describe the relationship between x and y; we add the qualification "Granger" to indicate that we may not be dealing with a true causal relationship. Similar to the examples given above and in Chapter 13 (where we used the Hilbert transform), there may be alternative explanations for why x helps to predict y or vice versa.

14.3 DIRECTED TRANSFER FUNCTION

In the following we will transform the Granger causality into the z-domain and frequency domain. This will generate the so-called directed transfer function (DTF), first described by Kamiński and Blinowska (1991). The explanation here will be informal; a formal proof of the relationship between Granger causality and DTF is presented in Kamiński et al. (2001).

14.3.1 Autoregression in the Frequency Domain

To link time and frequency domains, we start from a one-dimensional AR model and its frequency domain presentation. Next, we extend the approach to a multidimensional model and define the DTF. Because we want to follow the derivation by Kamiński and Blinowska (1991), we start our explanation in the time domain and transform our findings into the frequency domain via the z-transform.

14.3.1.1 One-Dimensional Example

A one-channel model of order p can be represented by:

$$x_n = a_1 x_{n-1} + a_2 x_{n-2} + \ldots + a_i x_{n-i} + \ldots + a_p x_{n-p} + e_n \tag{14.2a}$$

Here x_n are the signal samples, a_i are the AR model's coefficients, and e_n is the error term, which we represent by zero mean Gaussian white noise (GWN).

Eq. (14.2a) can be rewritten in the form

$$x_n - a_1 x_{n-1} - a_2 x_{n-2} - \ldots - a_i x_{n-i} - \ldots - a_p x_{n-p} = e_n \tag{14.2b}$$

If we model this expression by a discrete linear time-invariant system (Chapter 13), characterized by Eq. (13.1b), we may interpret this as the equation for a system with input e_n and output x_n (Fig. 14.2A).

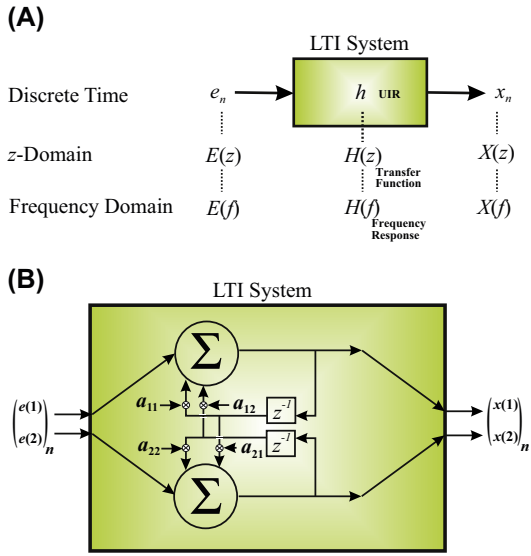


FIGURE 14.2 (A) A linear time invariant system with unit impulse response (UIR) h, representing the relationship between e_n and x_n in Eq. (14.2b). The signals in the discrete time domain can be transformed into the z-domain and into the frequency domain, and the UIR transformed into the transfer function and frequency response in Eqs. (14.2d and e). This is an example of a single-input and single-output system. (B) An example of a multi-input multioutput system. At time step n, the outputs $x(1)$ and $x(2)$ depend on inputs $e(1)$ and $e(2)$ and the past output at time step $n-1$. The latter is symbolized by the z^{-1} blocks. The strength of this dependence on past output is governed by a_{11}, a_{12}, a_{21}, and a_{22}.

Because we deal with a discrete time system, we can transform from the time-domain into the z-domain (to review the z-transform, see Chapter 12). We define the following transform pairs:

$$x_n \Leftrightarrow X(z)$$
$$e_n \Leftrightarrow E(z)$$

Applying the z-transform to Eq. (14.2b) we get:

$$X(z) - a_1 z^{-1} X(z) - a_2 z^{-2} X(z) - \ldots - a_i z^{-i} X(z) - \ldots - a_p z^{-p} X(z) = E(z) \tag{14.2c}$$

$$\rightarrow X(z)\left[1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_i z^{-i} - \ldots - a_p z^{-p}\right] = E(z)$$

$$\rightarrow X(z) = \underbrace{\left[\frac{1}{1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_i z^{-i} - \ldots - a_p z^{-p}}\right]}_{H(z)} E(z) = H(z)E(z) \tag{14.2d}$$

In the above (Eq. 14.2d), we defined $H(z)$ as the transfer function of the system with input noise e_n and output signal x_n.

Recall that the z-transform is the Laplace transform applied to discrete time series (Chapter 12); the complex variable z for a time series sampled with interval Δ is defined as $z = e^{s\Delta}$, where s is the complex variable of the Laplace transform. To get from the z-domain to the frequency domain, we now use the imaginary part of exponent $s = \sigma + j\omega = \sigma + j2\pi f$ of variable z, that is

$$z^{-1} = e^{-s\Delta} = e^{-(\sigma + j2\pi f)\Delta} \rightarrow z^{-1} \text{ substituted by } e^{-j2\pi f\Delta},$$

Using this, we can rewrite Eq. (14.2d) as a function of frequency f:

$$X(f) = H(f)E(f) \tag{14.2e}$$

Now we have an expression in the frequency domain, where $X(f)$ is the Fourier transform of signal x_n, $E(f)$ is the Fourier transform of the noise input e_n, and $H(f)$ is the frequency response characterizing the system with input e_n and output x_n. Since $X(f)$ is the discrete Fourier transform of x_n, the unscaled power spectrum S for x_n (Chapter 7) is:

$$S(f) = X(f)X(f)^* \tag{14.3a}$$

The * indicates the complex conjugate. Eq. (14.3a) combined with (14.2e) gives:

$$S(f) = [H(f)E(f)][H(f)E(f)]^*$$

Since this expression is the product of four complex numbers (for each frequency), we may remove the brackets and rearrange the terms:

$$S(f) = H(f)E(f)E(f)^*H(f)^* \tag{14.3b}$$

If the input process e_n is zero-mean, GWN, its unscaled power spectrum $E(f)E(f)^* = N\sigma^2$, where N is the number of measurements of x_n, and σ^2 is the variance of the noise process.

Note that the equality $E(f)E(f)^* = N\sigma^2$ is directly related to Parseval's theorem (Appendix 7.1), stating that the energy of a signal e in the time domain equals the energy of its power spectrum: $\sum \frac{EE^*}{N} = \sum e^2$. If the signal has zero mean we may use the following (biased) expression for variance: $\sigma^2 = \frac{1}{N}\sum e^2$. Combining the two latter expressions, we get: $\sum \frac{EE^*}{N} = N\sigma^2$. Now, if signal e is GWN, the power is equally distributed across the N bins of its spectrum, i.e., the power in each bin (the power for each frequency f) of the normalized spectrum is $N\sigma^2/N = \sigma^2$. Finally, for the non-normalized spectrum EE^* (instead of the normalized one EE^*/N) we find that the value for each frequency bin is $N \times \sigma^2 : E(f)E(f)^* = N\sigma^2$.

If we substitute this result in Eq. (14.3b), we get:

$$S(f) = N\sigma^2[H(f)H(f)^*] = N\sigma^2|H(f)|^2 \tag{14.3c}$$

Thus in this case the spectrum $S(f)$ is proportional with the power of the frequency response $|H(f)|^2 = [H(f)H(f)^*]$. Thus in the frequency domain $H(f)$ relates input with output, that is the input noise with the signal.

14.3.1.2 *Multidimensional Example*

The next step is to generalize the above from a one-channel p-th-order AR process to a k-channel one. As a first step let us consider a three-channel data set with a p-th-order AR process. For this system, the equivalent for Eq. (14.2a) becomes

$$\underbrace{\begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_n}_{\overrightarrow{x}_n} = \underbrace{\begin{pmatrix} a_{11}a_{12}a_{13} \\ a_{21}a_{22}a_{23} \\ a_{31}a_{32}a_{33} \end{pmatrix}_1}_{A_1} \underbrace{\begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_{n-1}}_{\overrightarrow{x}_{n-1}} + \underbrace{\begin{pmatrix} a_{11}a_{12}a_{13} \\ a_{21}a_{22}a_{23} \\ a_{31}a_{32}a_{33} \end{pmatrix}_2}_{A_2} \underbrace{\begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_{n-2}}_{\overrightarrow{x}_{n-2}} + \ldots$$

$$\ldots + \underbrace{\begin{pmatrix} a_{11}a_{12}a_{13} \\ a_{21}a_{22}a_{23} \\ a_{31}a_{32}a_{33} \end{pmatrix}_i}_{A_i} \underbrace{\begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_{n-i}}_{\overrightarrow{x}_{n-i}} + \ldots + \underbrace{\begin{pmatrix} a_{11}a_{12}a_{13} \\ a_{21}a_{22}a_{23} \\ a_{31}a_{32}a_{33} \end{pmatrix}_p}_{A_p} \underbrace{\begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_{n-p}}_{\overrightarrow{x}_{n-p}} + \underbrace{\begin{pmatrix} e(1) \\ e(2) \\ e(3) \end{pmatrix}_n}_{\overrightarrow{e}_n}$$

or

$$\overrightarrow{x}_n = A_1\overrightarrow{x}_{n-1} + A_2\overrightarrow{x}_{n-2} + \ldots + A_i\overrightarrow{x}_{n-i} + \ldots + A_p\overrightarrow{x}_{n-p} + \overrightarrow{e}_n \tag{14.4}$$

Here vector $\vec{x}_n = \begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_n$ are the samples for the three channels $x(1) - x(3)$, the 3×3 matrix $A_i = \begin{pmatrix} a_{11}a_{12}a_{13} \\ a_{21}a_{22}a_{23} \\ a_{31}a_{32}a_{33} \end{pmatrix}_i$ are AR model's coefficients, and vector $\vec{e}_n = \begin{pmatrix} e(1) \\ e(2) \\ e(3) \end{pmatrix}_n$ are the errors for each channel. Note that we use arrows to indicate that x and e are vectors instead of scalars now. A useful feature is that the **AR coefficients A_i relate the x values across time and, more importantly (in this context), across channels**. For instance, for the term

$$A_2 = \begin{pmatrix} a_{11}a_{12}a_{13} \\ a_{21}a_{22}a_{23} \\ a_{31}a_{32}a_{33} \end{pmatrix}_2 \begin{pmatrix} x(1) \\ x(2) \\ x(3) \end{pmatrix}_{n-2},$$

we have:

$(a_{11})_2 \times x(1)_{n-2}$ relating past values of $x(1)$ to the present value of $x(1)$—that is, the value of $x(1)$ at sample $n - 2$ contributes to the value of $x(1)$ at sample n;
$(a_{12})_2 \times x(2)_{n-2}$ relating past values of $x(2)$ to the present value of $x(1)$—that is, the value of $x(2)$ at sample $n - 2$ contributes to the value of $x(1)$ at sample n;
$(a_{13})_2 \times x(3)_{n-2}$ relating past values of $x(3)$ to the present value of $x(1)$—that is, the value of $x(3)$ at sample $n - 2$ contributes to the value of $x(1)$ at sample n;
$(a_{21})_2 \times x(1)_{n-2}$ relating past values of $x(1)$ at $n - 2$ to the present value of $x(2)$ at sample n; etc.

Following the definition of Granger causality as discussed in Section 14.2, we can use this expression to apply the principle of Granger causality: i.e., if one of the coefficients $a_{ij} \neq 0$ (with $i \neq j$), there is a causal relationship between the channels i and j, such that: $j \rightarrow i$. For example, if $(a_{12})_2 \neq 0$, we found $x(2) \rightarrow x(1)$, reflecting a causal relationship between channels $2 \rightarrow 1$.

Following our approach for Eq. (14.2a), Eq. (14.4) can be rewritten in the form

$$\vec{x}_n - A_1\vec{x}_{n-1} - A_2\vec{x}_{n-2} - \ldots - A_i\vec{x}_{n-i} - \ldots - A_p\vec{x}_{n-p} = \vec{e}_n \quad (14.5a)$$

We now repeat the same procedure as we employed for the one-dimensional case, and get to the frequency domain via the z-transform. First we define the following transform pairs:

$$\vec{x}_n \Leftrightarrow \vec{X}(z)$$
$$\vec{e}_n \Leftrightarrow \vec{E}(z)$$

The z-transform of Eq. (14.5a) is:

$$\overrightarrow{X}(z) - A_1 z^{-1}\vec{X}(z) - A_2 z^{-2}\vec{X}(z) - \ldots - A_i z^{-i}\vec{X}(z) - \ldots - A_p z^{-p}\vec{X}(z) = \vec{E}(z) \tag{14.5b}$$

$$\rightarrow \vec{X}(z)\left[I - A_1 z^{-1} - A_2 z^{-2} - \ldots - A_i z^{-i} - \ldots - A_p z^{-p}\right] = \vec{E}(z)$$

$$\rightarrow \vec{X}(z) = \underbrace{\left[\frac{I}{I - A_1 z^{-1} - A_2 z^{-2} - \ldots - A_i z^{-i} - \ldots - A_p z^{-p}}\right]}_{H(z)}$$

$$\vec{E}(z) = H(z)\vec{E}(z) \tag{14.5c}$$

In the above, I is the identity matrix and $H(z)$ is defined as the transfer function matrix between the input noise $\vec{E}(z) = \begin{pmatrix} E(1) \\ E(2) \\ E(3) \end{pmatrix}_z$ and output signal $\vec{X}(z) = \begin{pmatrix} X(1) \\ X(2) \\ X(3) \end{pmatrix}_z$. Similar to the procedure we followed in the one-dimensional example above, we now use the imaginary part of exponent $s = \sigma + j\omega = \sigma + j2\pi f$ of variable z to get to the frequency domain, that is:

$$z^{-1} = e^{-s\Delta} = e^{-(\sigma + j2\pi f)\Delta} \rightarrow z^{-1} \text{ substituted by } e^{-j2\pi f\Delta},$$

and rewrite Eq. (14.5c) as a function of frequency f:

$$\vec{X}(f) = H(f)\vec{E}(f) \tag{14.5d}$$

The unscaled spectrum S for $\vec{x}_n$ is:

$$S(f) = \vec{X}(f)\vec{X}(f)^* \tag{14.6a}$$

Note that for each frequency f in Eq. (14.6a), S is a 3×3 matrix. The auto and cross-spectra are computed for each frequency f, and * indicates **both** the complex conjugate and the transpose (note that in MATLAB®

both these operations are accomplished by the superscript $'$ after a vector or matrix variable, e.g., X'). Eq. (14.6a) combined with (14.5d) gives:

$$S(f) = \left[H(f)\overrightarrow{E}(f)\right]\left[H(f)\overrightarrow{E}(f)\right]^* = H(f)\overrightarrow{E}(f)\overrightarrow{E}(f)^*H(f)^* \qquad (14.6b)$$

In the above, we used the identity $\left[H(f)\overrightarrow{E}(f)\right]^* = \overrightarrow{E}(f)^*H(f)^*$. If the input noise processes $\overrightarrow{e}_n$ are independent GWN with zero mean and variance σ^2, we get $\overrightarrow{E}(f)\overrightarrow{E}(f)^* = N\sigma^2 I$, where N is the number of measurements of $\overrightarrow{x}_n$ and I is the identity matrix. This generates:

$$S(f) = N\sigma^2[H(f)H(f)^*] \qquad (14.6c)$$

Thus, the spectrum $S(f)$ is proportional with the power of the frequency response $H(f)H(f)^*$. Note that in the case of asymmetric causal relationships between channels, the matrix $H(f)$ is asymmetric, reflecting the asymmetric aspect of the connectivity.

14.3.1.3 The Directed Transfer Function

Unlike in Eq. (14.3c) in the one-dimensional example above, H in the multichannel version in Eq. (14.6c) is not a single value but a matrix for each frequency f (in the above example a 3×3 matrix). **Each element H_{ij} in H represents a transfer value between channels j and i.**

> To avoid confusion, please note that in this section we follow convention often used in matrix algebra and define i and j as subscripts to indicate a matrix element's row and column, respectively. Because the matrix element represents the channel interaction, i and j also denote channels. In contrast, j indicates the imaginary component in equations involving complex numbers.

Again we can see the relationship between this approach and Granger causality: if the transfer value between j and i is nonzero, there is an input–output (causal) relationship. Kamiński and Blinowska (1991) use a normalized version of H, which they call the DTF, to study interrelationships between channels in their data sets. They normalize each component H_{ij} by dividing it by the sum of all elements of H in the same row of the H matrix. Because H_{im} represents the effect of channel m on channel i, you can—in a K channel recording—normalize by division by all the contributions to channel i: $\sum_{m=1}^{K} H_{im}$. Therefore, the normalized version of transfer function element H_{ij} becomes $\frac{H_{ij}}{\sum_{m=1}^{K} H_{im}}$. Because H_{ij} is usually a complex number, this ratio is further simplified by using the

squared magnitude of its contributions, generating the commonly used definition of the DTF, γ_{ij}:

$$\gamma_{ij} = \frac{|H_{ij}|^2}{\sum_{m=1}^{K} |H_{im}|^2} \tag{14.7}$$

Thus the DTF is scaled between zero and one. The value of γ_{ij} can be determined from transfer matrix H, which can be determined using the matrix of AR coefficients A (Eq. 14.5c).

If we consider a simple case of two signals with an A matrix reflecting interactions between these signals at a one unit delay:

$$A = \begin{bmatrix} 0.0 & 0.3 \\ 0.8 & 0.0 \end{bmatrix},$$

and we compute z^{-1} for one delay as $z^{-1} = \exp(-j) = 0.5403 - 0.8415j$, thereby assuming for the sake of our example that $2\pi f\Delta = 1$. Following the definition in Eq. (14.5c), we compute $|H|$ using the MATLAB® command abs(inv(eye(2) – A.*exp(–j))). This produces the following (rounded) result for the magnitude of the frequency response $|H|$:

$$|H| = \begin{bmatrix} 0.89 & 0.27 \\ 0.71 & 0.89 \end{bmatrix}.$$

Now, we use Eq. (14.7) to compute the effect $2 \rightarrow 1$ as: $0.27^2/(0.89^2 + 0.27^2) = 0.08$ and the effect $1 \rightarrow 2$ as: $0.71^2/(0.71^2 + 0.89^2) = 0.39$. It can be seen that, although these numbers are not simply related to the numbers in A, these values do reflect the causal effects between the channels. A slight modification to the above example would be to remove the causal effect $2 \rightarrow 1$, resulting in the A matrix:

$$A = \begin{bmatrix} 0.0 & 0.0 \\ 0.8 & 0.0 \end{bmatrix}.$$

This generates the following magnitude of the frequency response:

$$|H| = \begin{bmatrix} 1.00 & 0.00 \\ 0.80 & 1.00 \end{bmatrix},$$

again reflecting the existence of a causal effect $1 \rightarrow 2$ but the absence of a causal effect $2 \rightarrow 1$. Also note that, because $H = (I - A)^{-1}$, the diagonal elements of the transfer function and frequency response represent both the direct effect of the noise input as well as a delayed effect of the channel. For instance, in this particular example, diagonal element h_{11} represents the effect $e(1)_n \rightarrow x(1)_n$ and the effect $x(1)_{n-1} \rightarrow x(1)_n$

(Fig. 14.2B). This is not a major problem since, for the causal analysis, the cross-channel relationships are of primary interest.

14.3.2 MATLAB® Examples

Having obtained the above results, it is appropriate to start thinking about an algorithm to determine γ_{ij} from measured data (for example, a multichannel electroencephalogram [EEG] recording). One practical approach to find the transfer matrix H in a measured data set, is to fit an AR model to the data and determine matrix A of the AR coefficients (Section 14.3.1.2). The inverse of A gives H (Eq. 14.5c). This is a parametric approach, because the basis is to fit parameters of an AR model to the data and the DTF is derived from there. Fitting a multichannel AR model to a data set is an "art" in itself, and is beyond the scope of this chapter. If you want to play with fitting models, there are several websites with MATLAB® scripts available. One example is the arfit toolbox from http://climate-dynamics.org/software/#arfit.

If you want to evaluate this arfit code you can create your AR model and evaluate the performance of the estimator's output with the known coefficients in your model. The following script, pr14_1.m, is an example of this procedure. (Note that this works only if you download and install the arfit toolbox! Recall to include the arfit directory in the path by using "Set Path …" in the "File" menu, or install the arfit scripts in your current directory.)

```
% pr14_1.m
% A test for identifying coefficients from a time series

% The program prints the coefficients we use (a and b)
% plus their estimates (vector A) obtained with the arfit function
['If needed INSTALL ARfit toolbox: http://climate-dynamics.org/
software/#arfit']
% installed in path C:\Program Files\MATLAB71\toolbox\ar_tools

clear;
close all;

% Set coefficients a and b
a=0.95;
b=-.55;
% Parameters & initial values for the time series
N=1000;
e=randn(1,N+3);                    % GWN input
x(1)=0;e(1)=0;
x(2)=0;e(2)=0;
```

```
% create the time series using autoregression
for i=3:N+3
   x(i)=a*x(i-1)+b*x(i-2)+e(i);
end;
% Remove the 1st 2nd zero-valued-points from x
x=x(3:N+3);
% Make e the same length
e=e(3:N+3);
% normalize x & e
x=x-mean(x);
x=x/std(x)^2;
e=e-mean(e);
e=e/std(e)^2;

% ARfit toolbox
[w,A,C,SBC,FPE,th]=arfit(x',0,4);     % arfit function
% Show the results
('coefficients in ')
[a b]
('estimated coefficients using arfit')
A
```

When you run the above script you will find that the arfit routine finds reasonable estimates for the coefficients a and b (i.e., your estimates for a and b will be close to 0.95 and −0.55, respectively).

Let us simulate an example in MATLAB®. First we create three signals: signal-1 to signal-3, grouped in a matrix X. Within this group of signals, we create causal relationships between signal-1 and the two other signals (Fig. 14.3A): i.e., X(1,:) → X(2,:) and X(1,:) → X(3,:).

Use the MATLAB® script pr14_2.m to simulate the signals with their causal relationship and analyze this causal relationship within the group of signals using the DTF.

```
%pr14_2
clear;
close all

sr=1000;
dt=1/sr;
Nyq=sr/2;
T=10;
F=0:1/T:Nyq;
% the A matrices for the autoregressive process
```

```
A1=[.5 0 0;0 .8 0;0 0 .7];      % x1, x2, and x3 depend on their own
                                   k-1 values
A2=[-.8 0 0;0 0 0;0 0 0];       % only x1 depends on its own k-2 value
A4=[-.2 0 0;.4 0 0;0 0 0];      % x1 and x2 both depend on x1 k-4 value
A7=[0 0 0;.8 0 0;.4 0 0];       % x2 and x3 both depend on x1 k-7 value
A13=[0 0 0;0 0 0;.9 0 0];       % x3 depends on x1 k-13 value
I=eye(3);
% time series
X=zeros(3,length(F));
% loop to simulate the time series note it starts at 14 since the max
   delay
% is 13 [as given by A13]
for k=14:length(F);
   E=randn(3,1);
   X(:,k)=A1*X(:,k-1)+A2*X(:,k-2)+A4*X(:,k-4)+A7*X(:,k-7)+
   A13*X(:,k-13)+E;
end;

k=0;
for f=0:1/T:Nyq;                % LOOP to compute H
   k=k+1;
   % compute the z values for each delay and use the imaginary part
   % j*2*pi*f*dt to obtain the values for the frequency response
   z1=exp(-j*2*pi*f*dt);
   z2=z1^2;
   z4=z1^4;
   z7=z1^7;
   z13=z1^13;
   H(:,:,k)=inv(I-A1*z1-A2*z2-A4*z4-A7*z7-A13*z13);      % Eq. (14.5c)
   H2(:,:,k)=abs(H(:,:,k)).^2;
end;

% Loops to plot the signals & compute and plot DTF
figure;hold on;
xmax=max(F);
% first plot the three signals
subplot(4,3,1);
plot(X(1,:),'k');hold;axis([0 length(F) min(X(1,:)) max(X(1,:))]);
t=['signal-1'];title(t);
axis('off');
subplot(4,3,2);
plot(X(2,:),'k');hold;axis([0 length(F) min(X(2,:)) max(X(2,:))]);
t=['signal-2'];title(t);
```

```
axis('off');
subplot(4,3,3);
plot(X(3,:),'k');hold;axis([0 length(F) min(X(3,:)) max(X(3,:))]);
t=['signal-3'];title(t);
axis('off');
% Next compute and plot the Spectra and DTFs
for n=1:3
  for m=1:3;
    if n==m;
      ttl=['Spectrum ' num2str(m)];
      % put the plot variable [the Spectrum] in a temporary array
      for k=1:5001;tmp(k)=H2(n,m,k);end;
      ymax=max(1.*tmp);
    else
      ttl=[' ' num2str(m) '->' num2str(n)];
      DTF(n,m,:)=H2(n,m,:)./sum(H2(n,:,:));          % Eq. (14.7)
      % put the plot variable [the DTF] in a temporary array
      for k=1:5001;tmp(k)=DTF(n,m,k);end;
      ymax=1.;
    end;
    subplot(4,3,n*3+m);
    plot(F,tmp,'k');
    axis([0 xmax 0 ymax]);
    title(ttl);
  end;
end;
```

The result of running the above scripts is a plot as depicted in Fig. 14.3B. The top row of plots in Fig. 14.3B shows the three signals in the time domain and the bottom part (3×3 matrix of panels) shows the spectral plus "causal" information. The diagonal panels are the spectra of each of the three channels and the off-diagonal plots show the DTF. It can be seen that, as expected, the plots in the first column show the presence of energy in the DTF, reflecting the causal relationships between signal-1 and the other two signals (signal-2 and signal-3).

14.4 APPLICATIONS OF CAUSAL ANALYSIS

Multichannel recording of brain electrical activity from the scalp (EEG) or the cortex (electrocorticogram, ECoG) is the clinical basis for the

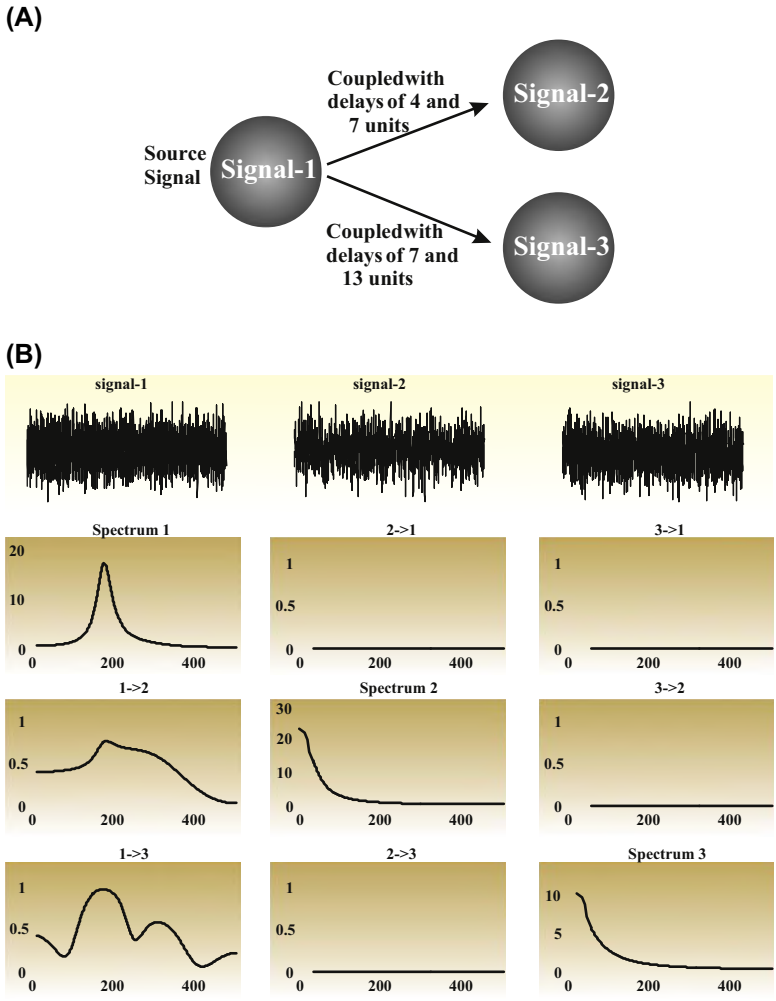


FIGURE 14.3 Test of the directed transfer function (DTF) algorithm on simulated data. (A) Diagram showing how three signals—signal-1, signal-2, and signal-3—relate to each other. The source signal-1 is coupled with delays to the other two electrode signals, signal-2 and signal-3. (B) The three top panels titled signal-1, signal-2, and signal-3 show 10-s epochs of the signals in the time domain and the bottom 3 × 3 panels are the result of the DTF analysis. The diagonal panels in the 3 × 3 arrangement show the power spectra of each signal and the off-diagonal panels show the DTF according to Eq. (14.7) (frequency scales in Hz). It is clear that the plots in the first column in the 3 × 3 arrangement contain energy, confirming that signal-1 is a source for signal-2 and signal-3. Both the amplitude and power in the plots are in arbitrary units, and the DTF is on a zero to one scale (Eq. 14.7). The graphs were prepared with MATLAB® script pr14_2.

evaluation of patients with epilepsy. These recordings can capture interictal spikes and epileptic seizures and they can be used to determine the temporal and spatial characteristics of these activity patterns. For surgical candidates, a precise localization of the region where seizures originate is highly significant because it determines the target for surgical resection. Therefore it is clinical practice to monitor surgical candidates for a period of several days. In such clinical recordings, even if we assume that the electrodes sufficiently cover the brain areas of interest, the determination of the origin of the ictal activity can be far from simple. First, the epileptologist must detect all seizures occurring in a large data set; second, within each seizure the origin of the epileptiform discharges must be determined. To determine the epileptic focus, the epileptologist will use multiple data sets reflecting brain structure and function (e.g., EEG, MRI, PET, etc.). The DTF analysis is a natural fit into this set of clinical data because it provides an indicator where activity may originate (Wilke et al., 2009). In this study, Wilke and coworkers describe how DTF can be combined with adaptive parameter estimation; this adaptive version of DTF allows time-variant coefficients of the AR model in order to deal with the nonstationarity of the EEG signal.

Finally, we discuss how several of the multichannel techniques can be employed to investigate brain activity. Gómez-Herrero et al. (2008) developed and applied a novel methodology based on multivariate AR modeling and independent component analysis (ICA) to determine the temporal activation of the intracerebral EEG sources as well as their approximate locations. First these authors used principal component analysis (PCA) to remove noise components (similar to our example in Chapter 28 with Lena's image in pr28_2.m) and ICA to identify the EEG sources (as in the example in Fig. 28.14). The direction of synaptic flow between these EEG sources is then estimated using DTF (as we did in the example shown in Fig. 14.3). The reliability of their approach is assessed with simulations and evaluated by analyzing the EEG alpha rhythm. Their results suggest that the major generation mechanism underlying EEG alpha oscillations consists of a strong bidirectional feedback between the thalamus and posterior neocortex. In conclusion, the study suggests that the combined application of PCA, ICA, and DTF can be a useful and noninvasive approach for studying directional coupling between neural populations.

EXERCISES

14.1 In the context of digital filters (Chapter 18), are the filter characteristics in Eq. (14.2d) and Eq. (14.5c) characteristic for infinite impulse response (IIR) or finite impulse response (FIR) filters?

14.2 Use the MATLAB® script pr14_2.m to examine the effect of coupling (different delays and strengths). Analyze several scenarios with strong and weak coupling, e.g.,:
a. Collect and summarize your results.
b. What is your conclusion?

14.3 Use the results you obtained in Exercise 14.2 and reanalyze the relationship between X(1,:), X(2,:), and X(3,:) using cross-correlation. You may use pr14_3.
See Chapter 13 for details on cross-correlation.
a. Collect and summarize your results.
b. Can you establish a causal relationship?
c. What is your conclusion?

14.4 Use coherence and phase coherence and reanalyze the relationship between X(1,:), X(2,:), and X(3,:) obtained in Exercise 14.2.
See Chapter 13 for details on coherence.
a. Collect and summarize your results.
b. Can you establish a causal relationship?
c. What is your conclusion?

14.5 Use the Hilbert transform to establish phase and reanalyze the relationship between X(1,:), X(2,:), and X(3,:) obtained in Exercise 14.2.
See Chapter 13 for details on the Hilbert transform.
a. Collect and summarize your results.
b. Can you establish a causal relationship?
c. What is your conclusion?

14.6 Compare your results of Exercises 14.2–14.5. What can you conclude?

14.7 In MATLAB®, modify pr14_2.m to create three signals where el1 → el2 → el3 (instead of the current scenario: el1 → el2 and el1 → el3). What can you conclude?

References

Gómez-Herrero, G., Atienza, M., Egiazarian, K., Cantero, J.L., 2008. Measuring directional coupling between EEG sources. Neuroimage 43, 497–508.

Granger, C.W.J., 1969. Investigating causal relations by econometric models and cross-spectral methods. Econometrica 37, 424–438.

Kamiński, M.J., Blinowska, K.J., 1991. A new method of the description of the information flow in the brain structures. Biol. Cybern. 65, 203.
Kamiński, M., Ding, M., Truccolo, W.A., Bressler, S.L., 2001. Evaluating causal relations in neural systems: Granger causality, directed transfer function and statistical assessment of significance. Biol. Cybern. 85, 145.
Wilke, C., Van Drongelen, W., Kohrman, M., He, B., 2009. Identification of epileptogenic foci from causal analysis of ECoG interictal spike activity. Clin. Neurophysiol. 120, 1449–1456.

CHAPTER

15

Introduction to Filters: The RC-Circuit

15.1 INTRODUCTION

In this chapter we introduce analog filters by exploring a simple RC-circuit consisting of a single resistor (R) and a single capacitor (C). Because most electrophysiology labs will have some basic electronic equipment (such as multimeters, oscilloscopes, etc.) available, it is recommended that the student uses this chapter as a guide for a practical exercise. If such equipment is not available, the chapter can be used as an introduction to electronic analog filters. First we will get acquainted with the basic behavior of a simple filter and in Chapter 16 we will worry about the mathematical analysis.

The purpose of most filters is clear-cut: to remove the part of a signal that is considered noise, and as we will see they usually do a great job. In a more general sense, the filters discussed in this text are good examples of linear time-invariant (LTI) systems, and the analysis techniques we apply to these filters can also be applied to characterize physiological systems. For instance, if we know the frequency response of a sensory cell and we assume (or establish empirically) that the cell behaves in a linear fashion, then we can model the cell as a filter and predict its response to any arbitrary input.

As with LTI systems in general, filters can be studied both in the ***time*** and ***frequency*** domain and they can be implemented using either ***analog*** or ***digital*** techniques. Analog filters can be analyzed with continuous-time mathematics, whereas the digital versions are described with discrete-time equations. If you want to read more about

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00015-1

filters, see Marven and Ewers (1996) for an introductory level text, or Chirlian (1994) for a more advanced text.

15.2 FILTER TYPES AND THEIR FREQUENCY DOMAIN CHARACTERISTICS

The most intuitive approach is to describe the operation of a filter in the frequency domain, where we can define the filter as an attenuator for certain undesirable frequency components while passing others (Fig. 15.1); in an ideal world, a filter would completely remove all noise components. Let's focus on the behavior of a filter to a single frequency,

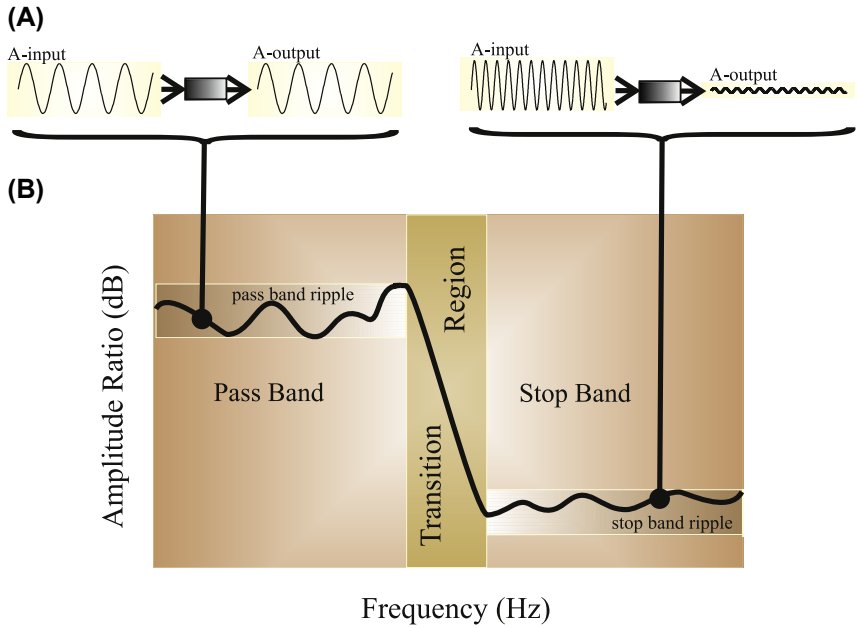


FIGURE 15.1 Filter characteristic determined by examining the sine wave input–output relationship. (A) The two examples show that sine waves in the so-called pass band can be unattenuated (left) or significantly reduced in amplitude (right) in the stop band. By determining the ratio between the amplitudes at the output (A-output) and the input (A-input) across a range of frequencies, one can construct the filter's frequency response/characteristic shown in (B). The filter characteristic expressed as amplitude ratio versus frequency of the filter input. The stop band denotes frequency regions in which amplitudes are attenuated, while the pass indicates the range of unattenuated frequencies. In real filters there is necessarily a transition region between these bands.

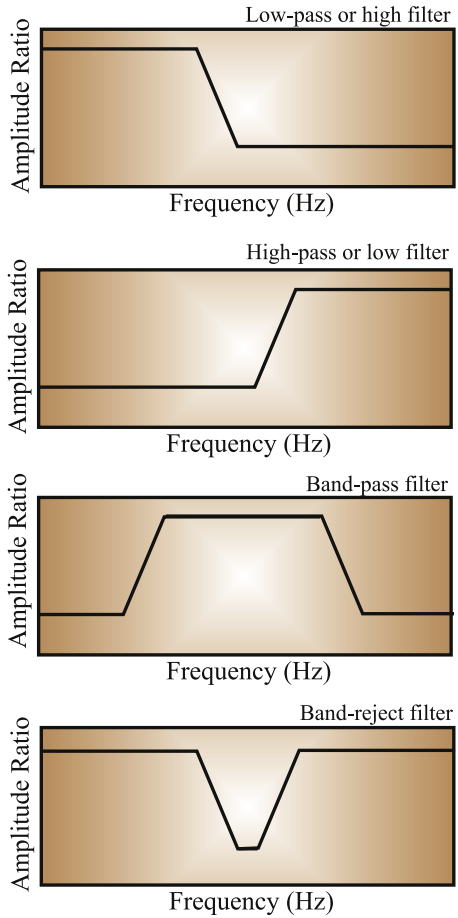


FIGURE 15.2 Filter types and their frequency characteristics. In these plots, the amplitude ratios for positive frequencies are shown. For digital filters it is not uncommon to also depict the values for the negative frequencies. Because the filter characteristic is an even function, there is no additional information provided in the plot for $\omega < 0$.

i.e., a pure sine wave. A central characteristic of any linear system is that its response to a sine wave input $A\sin(2\pi\omega t + \phi)$is also a sine wave in which the amplitude A and/or the phase ϕ may be altered. In Fig. 15.1A two examples are shown: a low-frequency sinusoid passes unattenuated while a higher-frequency sine wave is significantly reduced in amplitude at the filter output. In the examples in Fig. 15.1A the phase remains unaltered.

It is common practice to describe part of the frequency characteristic of a filter as the amplitude ratio between output and input for sine wave signals over a range of frequencies. In Fig. 15.1B the characteristic of the filter is divided into different bands: the pass band, the stop band, and the transition between these two bands. Ideally, the frequency components in the pass band would be unattenuated (i.e., a gain of $1\times$ or 0 dB; see Chapter 3, for a review of the dB scale), whereas the components in the stop band would be completely eliminated (i.e., a gain of $0\times$ or $-\infty$ dB). In addition, one would like a transition region of zero width. In the real world, gains in the pass band and stop band can deviate from 1 to 0, respectively. In addition, the amplitude ratio may show ripples, and the width of the transition region is necessarily >0 (Fig. 15.1).

Filters as described by their frequency response can be classified and combined into different types. The filter in Fig. 15.1 passes low frequencies and attenuates the high ones. This is referred to as a ***low-pass*** filter. The opposite type is the ***high-pass*** filter. A combination of low-pass and high-pass characteristics results in a ***band-pass*** filter and a system that attenuates a specific frequency band is a ***band-reject*** filter (Fig. 15.2).

15.3 RECIPE FOR AN EXPERIMENT WITH AN RC-CIRCUIT

The simplest analog electronic filter consists of a resistor (R) and a capacitor (C). A single so-called RC-circuit can either be a high-pass or low-pass filter, depending on how the components are connected. Examples of different diagrammatic representations of a low-pass filter, all denoting the same circuit, are shown in Fig. 15.3A–E. If the positions of the R and C are interchanged in the low-pass circuit in Fig. 15.3, we obtain a high-pass filter.

Prior to mathematical analysis, we will study the input–output relationship of RC-circuits with an experimental approach. We are interested in:

1. the ***transient response*** of the filter to a step function (a unit impulse would also be nice but is impossible to create) at the input, and
2. the ***steady-state response*** to a sinusoidal input, using a sine waves of different frequencies.

An example of a setup that includes a function generator to generate test signals and a dual-channel oscilloscope to simultaneously measure the input and output of a filter is shown in Fig. 15.4. Further basic

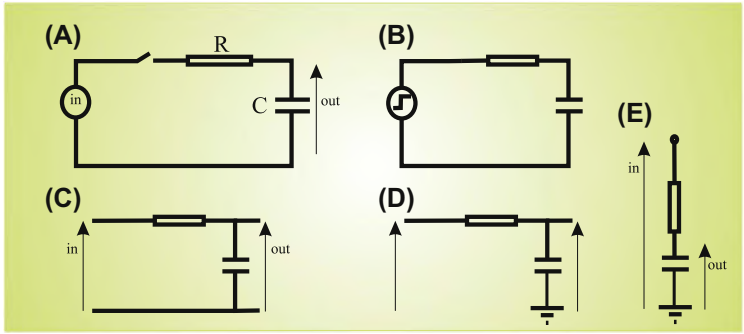


FIGURE 15.3 Different equivalent diagrams for an analog RC filter with low-pass characteristics. All diagrams represent a circuit with an R and a C component where the input signal is supplied over both the R and C components while the output is determined over the capacitor. The diagrams in (A) and (B) show most clearly that we deal with a closed circuit. The diagrams in (C), (D), and (E) (symbolizing the same circuit) are more frequently used in electronic diagrams and engineering texts. Note that this filter is the same circuitry as the simplified ion channel model introduced in Chapter 13.

requirements to make testing circuitry convenient are a breadboard for mounting the circuit and a simple multimeter.

In the first step an analog filter is created with a 10-kΩ resistor and a 3.3-μF capacitor. Because resistors are typically specified to different levels of precision (often allowing 5% variation from the indicated value), you can use the multimeter to determine the precise resistance value; without a multimeter you will have to believe the value indicated in the banded color code on the resister itself (e.g., brown-black-orange for 10 kΩ).

Note: A 10-kΩ resistor and a 3.3-μF capacitor would be the values I recommend for exploring a filter circuit; other values are also possible as long as we keep the resistance significantly lower than the input impedance of the oscilloscope (usually ∼1 MΩ), but higher than the function generator output (usually only several Ω); also for ease of measurement with standard equipment, the product of RC should be in the range of 5−100 ms.

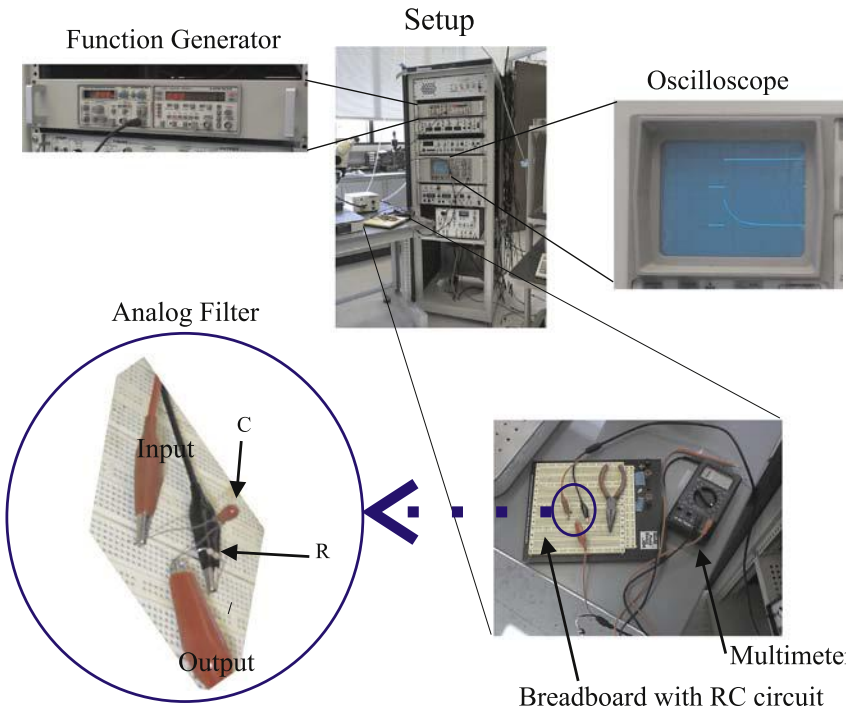


FIGURE 15.4 Setup used for analyzing analog filter circuits. A function generator is used to generate input signals (sine waves and step functions). The RC circuitry is built on a breadboard. The detail in the *blue circle* shows the R and C components plus the input (black = ground wire; red = signal wire) and output (red = signal wire) connections. The output of the function generator connects to both the first channel of the oscilloscope and the filter input. The output of the filter is connected to the second oscilloscope channel. Note that the (black) ground wire of the filter output is not connected because the filter's input and output are both measured simultaneously on the dual-channel oscilloscope, and the oscilloscope only needs to be connected to the ground signal once (in this case via the input wire). Two ground wires would result in a ground loop, which may add noise to a circuit's signal.

To build the test setup, construct an RC-filter on the breadboard, and then:

1. connect the output of the function generator to:
   a. the filter input, and
   b. the first input channel of the oscilloscope;
2. connect the filter's output to the second input channel of the oscilloscope.

Lab Assignment

After completing all connections we can start to characterize the circuit:

1. Determine the transient response: measure and sketch a detailed graph of the system's response to a voltage step of 1 V. In order to see that transient response clearly, you can set the frequency of a square wave of the signal generator to a very low value, and use the trigger or storage capability of the oscilloscope to examine the image.
2. Determine the steady-state response: Measure the system's output for sinusoidal inputs (0.2−1000 Hz). Since we will eventually present our data in a semilog plot, use a 1, 2, 5 sequence (i.e., 0.2, 0.5, 1, 2, 5, 10, etc.). You can also measure the phase difference between input and output signal by comparing zero-crossing times (although as compared to the amplitude ratio it is more difficult to measure this reliably).
3. Create a table and a graph of output amplitude (in dB) versus $\log_{10}$ of the frequency of each test sinusoid.
4. Now interchange the positions of R and C and redo steps 1−3.

Typical examples of a table and graphs for both filter types are shown in Table 15.1 and Fig. 15.5. In Chapter 16 we will analyze the data both in continuous time and in discrete-time models of this filter.

TABLE 15.1 Input−Output Ratios of an RC-Circuit for Sine Waves at Different Frequencies

Frequency (Hz)	Low-Pass AmpRatio	Low-Pass (dB)	High-Pass AmpRatio	High-Pass (dB)
0.3	1.00	0.00	0.05	−25.38
0.5	1.00	0.00	0.11	−19.36
1	0.96	−0.34	0.18	−14.67
2	0.92	−0.70	0.35	−9.21
5	0.69	−3.19	0.69	−3.19
10	0.46	−6.72	0.92	−0.70
20	0.27	−11.40	1.00	0.00
50	0.15	−16.26	1.00	0.00
100	0.06	−24.22	1.00	0.00
200	0.03	−30.24	1.00	0.00
500	0.01	−39.36	1.00	0.00
1000	0.01	−45.38	1.00	0.00

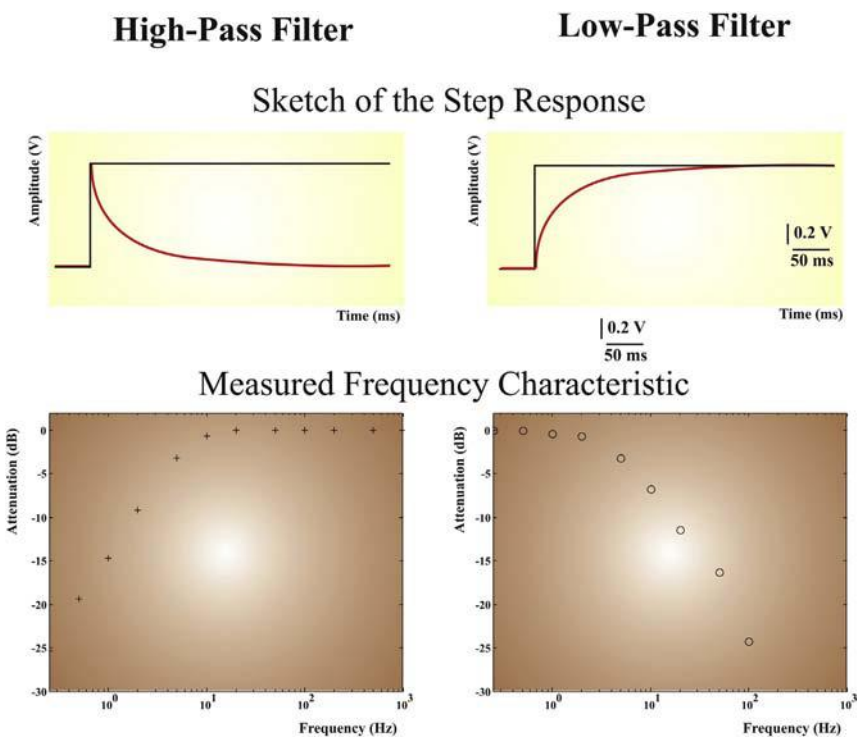


FIGURE 15.5 Typical result from measurements of an RC circuit either connected as a high-pass filter or a low-pass filter. The response to a 1-V step in the time domain is sketched in the two top plots. The ratio between input–output sine wave amplitudes is compiled in a table (e.g., Table 15.1), expressed in dB, and represented in a semilog plot (using MATLAB® command semilogx). These plots reflect the frequency characteristic of the particular circuit (compare these results with the plots for high-pass and low-pass filters in Fig. 15.2).

EXERCISE

15.1 Do the assignment described in Section 15.3. If you do not have access to a lab setup do the parts using the data from Table 15.1 to create a graph of output amplitude (in dB) versus $\log_{10}$ of the frequency of each test sinusoid.

References

Chirlian, P.M., 1994. Signals and Filters. Van Nostrand Reinhold, New York.

Marven, C., Ewers, G., 1996. A Simple Approach to Digital Signal Processing. John Wiley & Sons, New York, NY.

A simple introduction to data acquisition, filters, fast Fourier transform, and digital signal processing. This introduction requires minimal skills in mathematics.

CHAPTER

16

Filters: Analysis

16.1 INTRODUCTION

In Chapter 15, we experimentally determined filter properties in passive RC circuits. We observed that we could distinguish a pass band, a transition band, and a stop band in the frequency response of such a filter (e.g., Fig. 15.1). The filter's frequency characteristic can be used to define four basic types of filters: low-pass, high-pass, band-pass, and band-reject. In this chapter, we analyze the same RC filter with time domain and frequency domain techniques we introduced in previous chapters.

The gradually changing, frequency-dependent output observed from the RC circuit (in Chapter 15) demonstrates that this analog RC-filter response is far from that of an ideal filter, which would completely remove undesirable frequency components while leaving the components of interest unaltered (Fig. 16.1). Because ***analog*** filters are electronic circuits obeying the laws of physics, they behave in a ***causal*** fashion: i.e., the output cannot be determined by the input in the future, but must be determined by present and/or past input. Unfortunately, this makes it impossible to construct an analog filter with ideal characteristics because the inverse transform of the ideal profile (***a finite block of frequencies***, such as the one depicted in blue in Fig. 16.1) creates an impulse response ($\equiv$ the inverse Fourier transform of the frequency response) that violates causality because it includes values $\neq 0$ for $t < 0$ (Appendix 16.1).

Filter types that don't behave as causal linear time invariant systems do exist, but we will not consider these (more unusual) filter types in this chapter.

https://doi.org/10.1016/B978-0-12-810482-8.00016-3

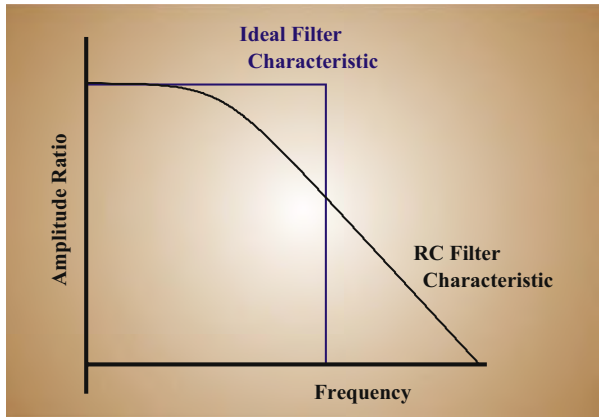


FIGURE 16.1 Low-pass filter frequency characteristic. An ideal characteristic (blue) would completely remove high frequencies while passing low-frequency components unaltered. In real filters, such as the RC-circuit (black), this ideal characteristic is compromised.

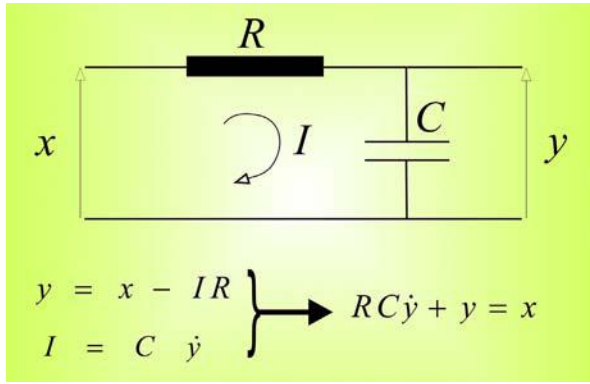


FIGURE 16.2 RC low-pass filter diagram and the associated ordinary differential equation. Current (I) passes through the resistor (R) and capacitor (C). High-frequency components of input x are attenuated in output y.

16.2 THE RC CIRCUIT

Fig. 16.2 shows a diagram and the associated ordinary differential equation (ODE) for the simple low-pass RC filter we explored in Chapter 15. An overview of the time and frequency domain properties associated with passive electronic components can be found in Appendix 16.2.

We can analyze this filter in several ways; all approaches generate an equivalent end result.

1. In ***continuous time*** analysis we can solve the ODE (Fig. 16.2) in several ways:
   a. ***Directly in the time domain (see also Chapter 9)***.
   Denoting y as the output and x as the input (Fig. 16.2), we can describe the RC circuit with the differential equation:

$$x = RC\frac{dy}{dt} + y \tag{16.1}$$

Setting the forcing term x to 0 to find the unforced solution we get:

$$\frac{dy}{dt} = -\frac{1}{RC}y \rightarrow \frac{dy}{y} = -\frac{dt}{RC} \rightarrow \ln(y) = -\frac{t}{RC} \rightarrow y = e^{-\frac{t}{RC}} \tag{16.2}$$

The above solution isn't the only one; any solution in the form $y = Ae^{-t/RC} + B$, with A and B as constants will work. In this case one usually solves for A and B by using the output values at $t = 0$ and large $t (t \rightarrow \infty)$. For $t \rightarrow \infty$, the first term $Ae^{-t/RC} \rightarrow 0$, hence B is the output at $t \rightarrow \infty$. For $t = 0$ the output is $A + B$. In most cases the output at ∞ is zero and the solution becomes $y = y_0 e^{-t/RC}$ with y_0 as output at $t = 0$.
Given an input x, a ***particular solution*** may be added to the ***unforced solution*** in order to obtain the ***general solution***. Often the choice of approaches for evaluating a particular solution depends on the forcing term (i.e., the input) x. For instance, if the input x is a sine wave with frequency f, we may find a particular solution of the form $A\sin(2\pi ft) + B\cos(2\pi ft)$; if x is an exponential function (e.g., $3e^{-2t}$) the particular solution of the same form (Ae^{-2t}) is expected.
   b. ***Directly in the frequency domain***.
   Using the formula for the impedance Z for a capacitor C as $Z = \frac{1}{j\omega C}$ together with Ohm's law we get:

$$x = i\left[R + \frac{1}{j\omega C}\right] \text{ and } y = i\frac{1}{j\omega C} \rightarrow x = j\omega Cy\left[R + \frac{1}{j\omega C}\right] \tag{16.3}$$

This results in an input−output relationship in the frequency domain; i.e., the frequency response:

$$\frac{y}{x} = \frac{1}{1 + j\omega RC} \tag{16.4}$$

c. ***Indirectly by using the Laplace or Fourier transform (see also Chapter 12)***.
Using the unit impulse as the input to our ODE/filter, i.e., $x = \delta$, we get the following transforms:

$\delta \Leftrightarrow 1$	for both the Laplace and Fourier transform;
$y \Leftrightarrow Y(s)$ or $Y(j\omega)$	for the Laplace and Fourier transform respectively;
$\frac{dy}{dt} \Leftrightarrow sY(s)$ or $j\omega Y(j\omega)$	for the Laplace and Fourier transform respectively.

The Laplace-transformed ODE is therefore:

$$1 = RCsY(s) + Y(s) \rightarrow Y(s) = H(s) = \frac{1}{1 + RCs} \tag{16.5}$$

In this case $Y(s)$ is the transfer function $H(s)$ because the input is the unit impulse δ. Using the Fourier transform instead of the Laplace transform we can determine the filter's frequency response:

$$Y(j\omega) = H(j\omega) = \frac{1}{1 + RCj\omega} \tag{16.6}$$

Using a table for Laplace transform pairs (Appendix 12.1), we can find the inverse transform for the transfer function (in the Laplace domain), generating the filter's unit impulse response function $h(t)$:

$$h(t) = (1/RC)e^{-t/RC} \text{ for } t \geq 0 \tag{16.7}$$

The inverse of the Fourier transform in Eq. (16.6) generates the same result. Note that we obtain an exponential function for $t \geq 0$ only where all output for $t < 0$ is supposed to be zero; this results in a single-sided Fourier transform pair that is equivalent to the single-sided Laplace transform used above.
Since we are dealing with a linear system and we know the RC-circuit's transfer function (Eq. 16.5), we can in principle determine the filter's

output $y(t)$ to an arbitrary input function $x(t)$. In the time domain this can be done using convolution:

$$y(t) = h(t) \otimes x(t) = x(t) \otimes h(t)$$

In the s-domain we can obtain the Laplace transform $Y(s)$ of time domain output $y(t)$ by multiplication of the transfer function (Eq. 16.5) with the Laplace transform of the input. For instance, if we want to determine the output caused by a step $U(t)$ at the input, we have the following transform pairs:

$$x(t) = U(t) \Leftrightarrow \frac{1}{s} \quad \text{(see Appendix 12.1)},$$

$$h(t) \Leftrightarrow \frac{1}{1 + RCs} \quad \text{(the transfer function)},$$

$$y(t) = h(t) \otimes x(t) \Leftrightarrow Y(s) = H(s)X(s) = \frac{1}{1 + RCs} \frac{1}{s} = \frac{1}{RC} \left[\frac{1}{s + 1/RC} \frac{1}{s} \right].$$

Using partial fraction expansion (see Appendix 12.3) and the table for Laplace transform pairs (Appendix 12.1), we find that the solution in the time domain is:

$$y(t) = 1 - e^{-t/RC} \quad \text{for } t \geq 0. \tag{16.8}$$

Here we determined the time domain response by finding the inverse transform of the solution in the s-domain. A graphical representation of the convolution procedure applied to the unit step and the exponential unit impulse response in the time domain is shown in Fig. A13.1-1. Not surprisingly, the outcomes of the direct convolution procedure and the Laplace transform method are the same.

2. In ***discrete time***, the ODE for the RC-circuit can be approximated with a difference equation. One technique to obtain an equivalent difference equation is by using a numerical approximation (such as the Euler technique) for the differential Eq. (16.1). Alternatively, if the sample interval is very small relative to the time constant (RC), one can approximate Eq. (16.1) in discrete time by (Appendix 16.3):

$$x(n) = RC \frac{y(n) - y(n-1)}{\Delta t} + y(n),$$

Simplifying notation by substituting $A = \frac{RC}{\Delta t}$ we obtain:

$$x(n) = y(n)[A + 1] - Ay(n-1) \rightarrow y(n) = \frac{x(n) + Ay(n-1)}{A + 1} \tag{16.9}$$

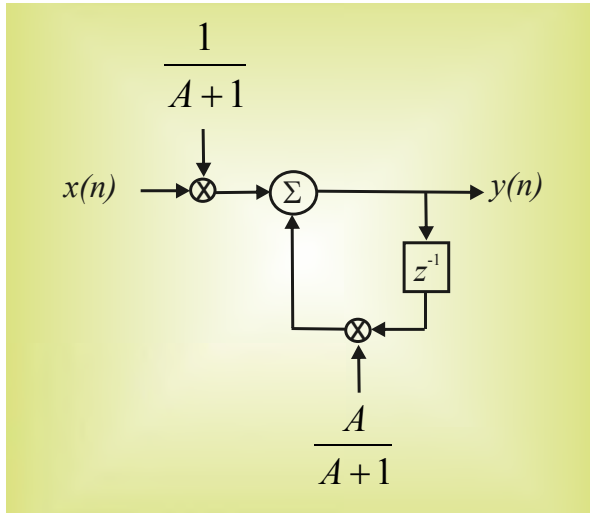


FIGURE 16.3 Discrete version of a continuous time analog low-pass RC filter. This block diagram depicts the algorithm for the difference Eq. (16.9).

The difference equation can be solved:

a. ***Numerically by direct calculation.***
A difference equation such as Eq. (16.9) where $y(n)$ is expressed as a function of a given input time series can easily be implemented in a MATLAB® script. Graphically, a block diagram can be used as the basis for such an implementation (e.g., Fig. 16.3). To mimic our experimentally obtained data in Chapter 15, type in the following parameters for the filter in the MATLAB® command window:

```
sr=400;
dt=1/sr;
R=10^4;
C=3.3e-6;
tau=R*C;
A=tau/dt;
t=0:dt:1;
x=ones(length(t),1);
y(1)=0;
```

Now, type in the following line representing the recursive algorithm of Eq. (16.8):

```
for n=2:length(t); y(n)=(A/(A+1))*y(n-1)+x(n)/(A+1);end;
```

You can study the outcome by plotting the results for the output and for the input in the same figure:

```
figure; hold;
plot(t,y,'r')
plot(t,x,'k')
axis([-0.1 1 0 1.1])
```

If you want, you can add axis labels and a title to the graph:

```
xlabel ('Time (s)');
ylabel ('Amplitude (V)')
title('Low pass filter response (red) to unit step input (black)');
```

The result of the plot is identical to the sketch of the filter response (shown in red in this example) to a unit step function (shown in black in this example). You can compare your finding with the example in Fig. 15.5.

b. ***Indirectly by using the z transform.***

The difference in Eq. (16.9) can be transformed into the z-domain:

$$x(n) \Leftrightarrow X(z)$$
$$y(n) \Leftrightarrow Y(z)$$
$$y(n-1) \Leftrightarrow z^{-1}Y(z)$$

Substituted in the difference equation we get:

$$X(z) = (A+1)Y(z) - Az^{-1}Y(z)$$

As in the s- or Fourier-domains, the transfer function $H(z)$ is a ratio of the output to the input:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{(A+1) - Az^{-1}} = \frac{1}{A+1}\frac{1}{1 - \frac{A}{A+1}z^{-1}} \tag{16.10}$$

Using the table for z-transform pairs in Appendix 12.1 we can determine that the inverse transform is:

$$h(n) = \frac{1}{A+1}\left[\frac{A}{A+1}\right]^n = \frac{A^n}{(A+1)^{n+1}} \tag{16.11}$$

This result can be used directly to simulate the impulse response for the discrete version of the low-pass filter.

16.3 THE EXPERIMENTAL DATA

The experiment described in Chapter 15 resulted in the measured response of the filter to step and sine wave inputs. In the analysis in this chapter we found that the unit step response of the filter can be represented by Eq. (16.8) and can be numerically calculated by using the MATLAB® commands described in Section 16.2. If you create the graphs using these commands, the fit between the theoretical and the measured step response in Fig. 15.5 (top-right plot) will be obvious.

The plot of the output/input amplitude ratio versus frequency in Fig. 15.5 (bottom-right plot) is the filter frequency response characteristic, which corresponds to Eq. (16.4) or (16.6). In these equations the output−input relationship is a complex-valued function (including a real and imaginary part) of frequency and the details of how to relate this complex function to measured data will be further discussed in Chapter 17; for now it is obvious that Eq. (16.4) and (16.6) both represent a function that decreases with frequency, which is consistent with a low-pass characteristic.

The conclusion that the frequency response of the filter is complex is directly related to the presence of the capacitance, which necessarily implies an imaginary value for the impedance (Fig. A16.2-1). In circuits where only resistors are involved, the impedance is real (i.e., equal to R, Fig. A16.2-1) and consequently the frequency response is also real. A real-valued frequency response (i.e., the imaginary component is zero; see also Chapter 17, Fig. 17.3) indicates that there is no change of phase (i.e., $\phi = 0$ in Fig. 17.3) between a sine wave at the output relative to the input. This principle can also be extended to a wider context, for instance in modeling experiments in slices, where the extracellular medium can be considered mainly resistive (capacitance can be neglected); in such a medium, the frequency response is real and no phase changes occur. On the other hand, in cases with transition layers between media such as membranes, membrane capacitance plays a critical role and phase changes in membrane current across such barriers may be significant.

APPENDIX 16.1

The Ideal Filter

An ideal filter characteristic passes a finite block of frequencies unaltered (let's say, up to a certain frequency ω_c) while completely removing frequencies outside the pass band from the signal (blue, Fig. 16.1). Because of the immediate transition between pass band and stop band, this filter is also called a brick–wall filter. Since the filter characteristic $H(j\omega)$ is an even function, it is typically only shown for $\omega > 0$.

If one calculates the inverse Fourier transform of the product of the filter characteristic $H(j\omega)$ (which is already in the frequency domain) and the Fourier transform of the unit impulse function $\delta(t)$ (i.e., 1, see Eq. 6.9), one obtains the unit impulse response $h(t)$:

To summarize:

$$1 \Leftrightarrow \delta(t)$$
$$H(j\omega)\begin{cases} 1 & \text{if } |\omega| \le \omega_c \\ 0 & \text{if } |\omega| > \omega_c \end{cases} \qquad \text{(A16.1-1)}$$

Because $H(j\omega)$ is 0 outside the $\pm\,\omega_c$ range we may change the integration limits in the inverse Fourier transform, i.e.,:

$$h(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(j\omega)e^{j\omega t}d\omega = \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} 1e^{j\omega t}d\omega = \frac{1}{2\pi jt}\left[e^{j\omega t}\right]_{-\omega_c}^{\omega_c}$$
$$= \frac{1}{2\pi jt}\left[e^{j\omega_c t} - e^{-j\omega_c t}\right] \qquad \text{(A16.1-2)}$$

Using Euler's relation, the above evaluates to the so-called sinc function:

$$h(t) = \frac{\sin(\omega_c t)}{\pi t} \qquad \text{(A16.1-3)}$$

In Fig. A16.1-1, it can be seen that $h(t)$ exists for $t < 0$, whereas the input $\delta(t)$ occurs only at $t = 0$; this indicates that such an ideal filter is a ***noncausal*** system. In the analog world where systems must behave causally, such a filter cannot be made, but only approximated.

In the digital world, there are other problems associated with implementing an ideal filter. In Fig. A16.1-1 it can be seen that there are oscillations in the impulse response $h(t)$ from $-\infty$ to ∞, and its frequency domain equivalent has an infinitely steep slope. In the first place neither of these properties can be represented in a real digital system with finite

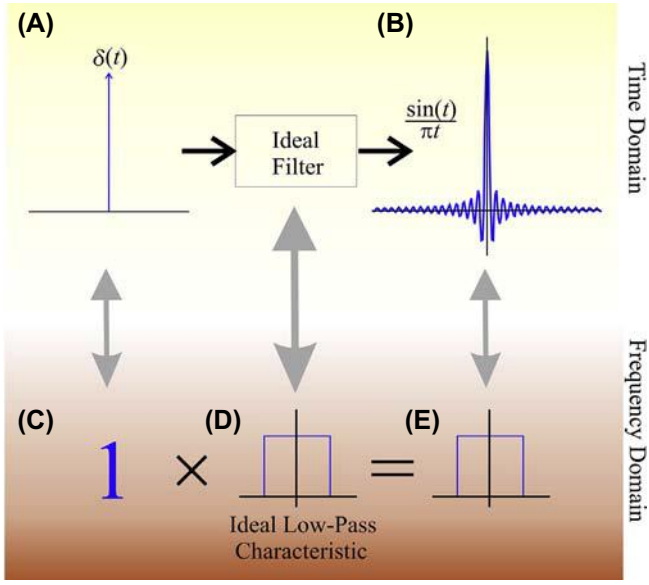


FIGURE A16.1-1 The ideal low-pass filter would completely remove high-frequency components and leave the low-frequency components unaltered. In the frequency domain this would correspond to a rectangular frequency response (D); **note that here the negative frequencies are also depicted**. In the frequency domain the output (E) is the product of input (C) and the frequency response (D). The time domain response of this filter (B) to a unit impulse (A) precludes the existence of such an ideal device because a nonzero component is present in the response at $t < 0$: i.e., there is a response before the input is given at $t = 0$; therefore the filter as a physical system cannot exist because it violates causality.

memory. Further, when an ideal filter is convolved with transients at the input (e.g., a square wave) this causes a ripple in its output. An example of a square wave approximated with a finite number of sine waves (i.e., a truncated spectrum) was shown in Chapter 5, Fig. 5.2. In this example, a ripple effect in the square wave approximation by five sinusoidal waves is clearly visible. This example mimics the effect of a simple truncation of the higher-frequency components of a square wave just as an ideal low-pass filter would do. Interestingly, while the ripple frequency increases with an increased number of component sine waves in the approximation, (strangely) the individual oscillations in the ripple have fixed amplitude ratios (first described by Gibbs in the 19th century). For an ideal filter with a square wave input (with zero-mean and an equal duty cycle), the first oscillation is an overshoot (see also Fig. 17.1, Chapter 17) with an amplitude that is 18% of the expected step amplitude. The MATLAB® script pr16_1.m simulates the effect of truncating the spectral content of a square wave.

APPENDIX 16.2

The resistor, inductor, and capacitor are the passive components used in electronic circuits for filtering. The symbols used for them in circuit diagrams and their properties in the time and frequency domains are summarized in Fig. A16.2-1.

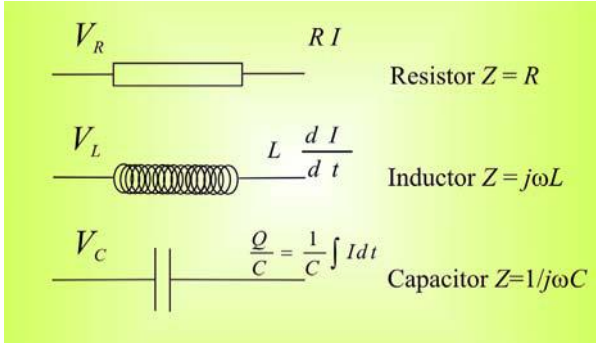


FIGURE A16.2-1 Electronic components, their relationships between current and potential in the time domain, and their representations of impedance in the frequency domain.

APPENDIX 16.3

The solutions to differential equations can be approximated numerically, e.g., by using the Euler method described in Chapter 10. Briefly, if $\dot{y} = f(y)$, and one wants to estimate a point y_n from the previous value y_{n-1} with an interval distance of Δt, one can use a linear approximation of the function at hand, and estimate the difference between y_n and y_{n-1} by the derivative at y_{n-1} multiplied by the distance, i.e.,:

$$y_n = y_{n-1} + \dot{y}_{n-1}dt = y_{n-1} + f(y_{n-1})\Delta t \qquad \text{(A16.3-1)}$$

Eq. (A16.3-1) is a difference equation that approximates the differential equation $\dot{y} = f(y)$.

Alternatively one can use knowledge about the solution of the differential equation to describe a difference equation. For instance, a difference equation that is equivalent to Eq. (16.1) is:

$$y_n = e^{-\Delta t/RC} y_{n-1} + \left(1 - e^{-\Delta t/RC}\right) x_n \qquad \text{(A16.3-2)}$$

Here we use the solution for Eq. (16.1) to relate the output at n with previous output and input. Using the unforced solution of Eq. (16.1) $Ae^{-t/RC} + B$, we can solve for A and B. We assume zero output for

$t \to \infty$, we set the initial value to $y_{n-1,}$ and the time difference between y_n and y_{n-1} to Δt; this results in $y_n = e^{-\Delta t/RC} y_{n-1}$, which is the first term in Eq. (A16.3-2). This term indicates that, for subsequent values of n, the output signal decays following an exponential with a time constant $\Delta t/RC$. If there is no input x, the second term in Eq. (A16.3-2) is 0, and this is the whole story. However, in the presence of input (i.e., a forcing term) we must add a particular solution to obtain the general one (see also Chapter 9). Let's assume that there is a constant input with amplitude x_n, and we know from our experiments that the low-pass filter will respond with a constant output (there will be no decay). This behavior leads to the second term in Eq. (A16.3-2) in which the correction factor $(1 - e^{-\Delta t/RC})$ for x_n is required to compensate for the leakage factor $e^{-\Delta t/RC}$ in the first term, thus maintaining the output y constant for constant input x.

Now we can show that Eq. (A16.3-2) can be approximated by Eq. (16.9) if $\Delta t \ll RC$, i.e., for small values of the exponent. Here we repeat Eq. (16.1) and the approximation in Eq. (16.9):

$$x = RC\frac{dy}{dt} + y \text{ and approximation } x(n) = RC\frac{y(n) - y(n-1)}{\Delta t} + y(n) \tag{A16.3-3}$$

We can rewrite the approximation as:

$$\begin{aligned} &RCy(n) + \Delta t y(n) = RCy(n-1) + \Delta t x(n) \\ &\to y(n) = \frac{RC}{RC + \Delta t} y(n-1) + \frac{\Delta t}{RC + \Delta t} x(n) \end{aligned} \tag{A16.3-4}$$

The correction factor for $y(n-1)$ can be written as:

$$\frac{1}{1 + \dfrac{\Delta t}{RC}} \approx \frac{1}{e^{\Delta t/RC}} = e^{-\Delta t/RC} \quad \text{if } \Delta t \ll RC \tag{A16.3-5}$$

Here we used the power expansion of the exponential $\left[e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots\right]$, where we set all higher-order terms to zero, i.e., $e^x = 1 + x$, with $x = \Delta t/RC$. Usually this is a reasonable thing to do since we sample relatively frequently so that Δt is small relative to the time constant. Similarly, we can write the factor for $x(n)$ in Eq. (A16.3-4) as:

$$1 - \frac{RC}{RC + \Delta t} = 1 - \frac{1}{1 + \dfrac{\Delta t}{RC}} \approx 1 - \frac{1}{e^{\Delta t/RC}} = 1 - e^{-\Delta t/RC} \quad \text{if } \Delta t \ll RC \tag{A16.3-6}$$

Combining Eqs. (A16.3-4)–(A16.3-6) we get the expression in Eq. (A16.3-2) again. It should be emphasized that the approximations in the Euler approach and the approximation in Eq. (16.9) are only suitable for smaller time intervals because the error in each step will be compounded in the following steps.

EXERCISES

16.1 Calculate the frequency response for the low-pass RC filter that was examined in Chapter 15.
- a. Show the steps of your calculation.
- b. Compare this with the data you measured in the filter demonstration session.
- c. Plot both the theoretical and measured data in a single semilog plot.

16.2 Compute and plot the low-pass RC filter's unit step response.

16.3 Fig. E16.1 represents a so-called finite impulse response filter with input $x(n)$ and output $y(n)$.
- a. Determine the expression that relates input and output of the filter in the discrete time domain.
- b. Determine the z-transform of this expression.
- c. Determine and sketch the unit impulse response of the filter.

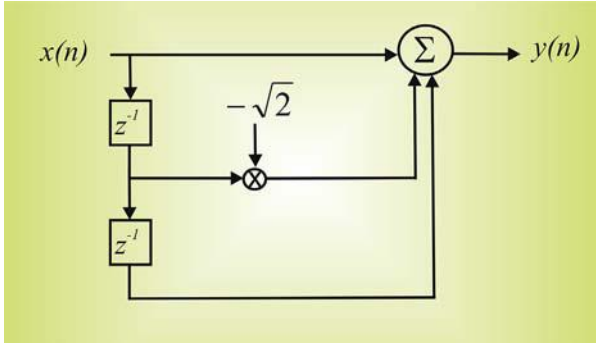


FIGURE E16.1 Diagram of a finite impulse response filter.

CHAPTER

17

Filters: Specification, Bode Plot, Nyquist Plot

17.1 INTRODUCTION: FILTERS AS LINEAR TIME-INVARIANT SYSTEMS

In this chapter we will continue to analyze filters while considering the RC filter, consisting of a resistor (R) and capacitor (C), presented in Chapters 15 and 16 as a linear time-invariant (LTI) system as described in Chapter 13. To fully characterize an LTI system we may specify any of the following properties:

1. the system's reaction to a unit impulse: the ***unit impulse response***, or
2. the Laplace or z-transform of the unit impulse response (***transfer function***), or
3. the Fourier transform of the unit impulse response (***frequency response***).

The unit impulse response is useful because ***convolution*** of the unit impulse response with the input provides the output. The transfer function is practical because, just as in the frequency domain, the convolution may be performed as a multiplication in the s- or z-domain (Chapter 13). The ***frequency response*** is of practical interest for the same reason but also because it relates immediately and intuitively to the filter's function and its specification into pass band, transition band, and stop band.

The frequency response $H(j\omega)$ of a filter (LTI system) can be obtained analytically by using the Fourier transform of the unit impulse response, or by deriving the solution in the frequency domain from knowledge of the system's components (Chapter 16, Section 16.2). In addition, one can determine the frequency response either from the transfer function or the z-transform of the impulse response. To convert from the Laplace transform $H(s)$ to frequency response $H(j\omega)$, one can often simply substitute $j\omega$ for s in the transfer function expression $H(s)$. In the case of the z-transform

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00017-5

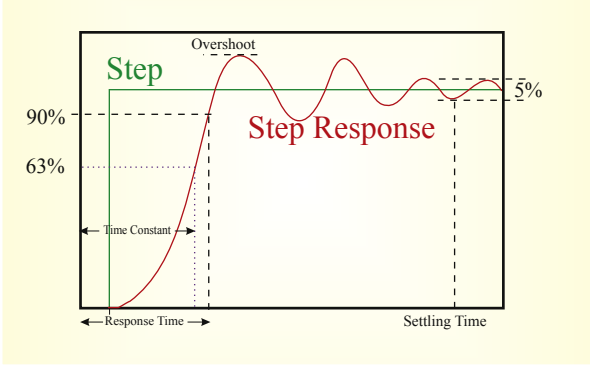


FIGURE 17.1 Example of a filter response to a unit step.

one can use the definition of the complex variable z (Chapter 12, Eq. (12.24)) to make the substitution $z = e^{j\omega\Delta t}$ in order to convert $H(z)$ into $H(j\omega)$.

Generally we can obtain the filter/system characteristic by determining the input−output relationship. Using a combined approach, we may study the filter by providing different types of input:

1. Transients such as the unit impulse δ (Dirac) or, in an experimental setting, the unit step U (Heaviside) function,
2. Continuous inputs studied during a steady state (SS):
   a. Sinusoidal inputs using a range of frequencies (as we demonstrated in the example for the RC circuit in Chapter 15).
   b. White noise representing all possible frequencies.

When we apply a transient such as the unit step to a filter's input, we may obtain a response as shown in Fig. 17.1. The filter's step response in Fig. 17.1 is typical for filters with a steep transition from pass band to stop band and it illustrates the typical overshoot followed by a ripple (Appendix 16.1). As we have already determined, in a passive filter with R and C components, the step response is smoother (Fig. 17.2A). The response to a transient is frequently characterized by the response time (Fig. 17.1) or the RC time (the so-called time constant $\tau = RC$, Figs. 17.1 and 17.2A). As will be shown in Section 17.2, the RC value characterizes the transient response of the filter but also relates to the frequency response characterization.

17.2 TIME DOMAIN RESPONSE

In the time domain, the dynamics of the low-pass filter's output is determined by the exponential $e^{-t/RC}$ (e.g., Eqs. 16.2 and 16.8). At the time

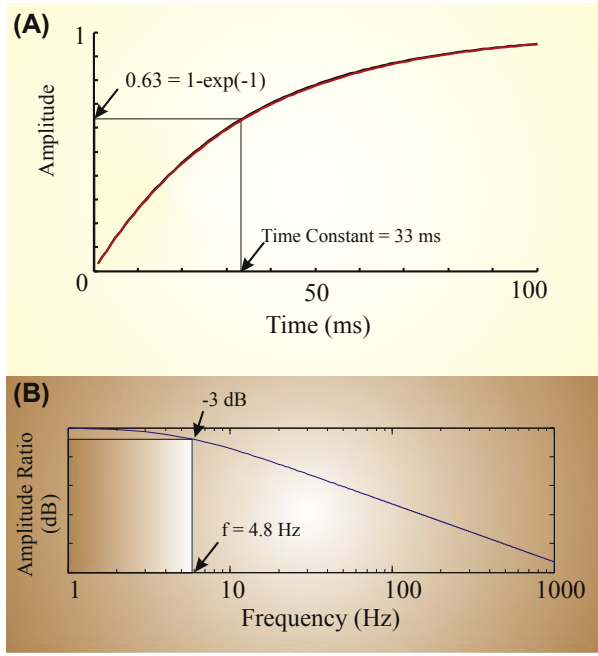


FIGURE 17.2 A low-pass filter's response to a unit step (A) and its frequency characteristic (B). In the time domain, the time constant τ (=RC) is determined to be: $[1 - \exp(-1)] = 0.6321$ of the final output. In the frequency domain the cut-off frequency at the −3-dB point is at $1/2\pi\tau$ Hz.

equal to the time constant $t = \tau = RC$, the value of the exponential is $e^{-1} \approx 0.37$. Thus, at $t = \tau$, depending on what direction the output goes (away from 0 or towards 0), the filter output is either at ~37% or ~63% of its final amplitude. In the case of the filter considered in Chapter 15 ($R = 10$ kΩ, $C = 3.3$ μF) we may find this at $\tau = RC = 33$ ms (Fig. 17.2A).

The following MATLAB® script simulates the response of a low-pass filter to a Unit Impulse and a Unit Step. Here we use the two different approaches (continuous time, discrete time) discussed in Chapter 16.

```
% pr17_1.m
% Filter Implementations for Impulse and Step response

clear
figure; hold;
% The basis is an analysis of a low-pass RC circuit
% we use R=10k and C=3.3uF
R=10e3;
```

```
C=3.3e-6;
RC=R*C;

%                                   UNIT IMPULSE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The analysis compares different approaches to obtain an
% impulse response

% COMPARED ARE:
% 1.    The analog continuous time approach using the Laplace transform
%       for the impulse response we obtain: 1=RCsH(s)+ H(s)
%       after the inverse transform this is: h(t)=(1/RC)*exp(-t/RC)
%       to compare with later discrete time approached we assume:
sample_rate=1000;
dt=1/sample_rate;
time=0.1;

i=1;
for t=dt:dt:time;
  yh(i)=(1/RC)*exp(-t/RC);
  i=i+1;
end;
plot(yh,'k')

% 2.    The difference equation mode
%       The difference equation: x(n*dt)=RC[(y(n*dt)-y(n*dt-1*dt))/dt] +
%       y(n*dt)
%       for the algorithm we set n*dt to n and obtain x(n)=[RC/dt]*[(y(n)-
%       y(n-1))]+y(n)

A=RC/dt;
x=zeros(1,100);x(1)=1/dt;   % the input is an impulse at t=0 we correct the
                              input
                            % with 1/dt
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% To be able to directly compare the analog and discrete impulse response,
% we have to correct the amplitude of the impulse. In case of a sampled signal
% we can assume the impulse to be of duration dt and amplitude 1/dt.
% Therefore either the input (i.e. the impulse) or the output (i.e. the impulse
% response) must be corrected for the amplitude % of 1/dt!!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
y_previous=0;

for n=1:100;
   y(n)=(A*y_previous+x(n))/(A+1);
   y_previous=y(n);
end;
plot(y,'r')

% 3.   The z-domain solution
%      Set the equation in (2) above to the z-domain
%                             X(z)=(A+1)Y(z)-(A/z)Y(z)
%                             Y(z)=1/[(A+1)-A/z]
%      Transformed: y(n)=A^n/(A+1)^(n+1)
for n=1:100;
   yz(n)=A^n/(A+1)^(n+1);
end;
yz=yz/dt;              % Because we calculated yz on the basis of the
                       % discrete impulse, we correct the output with 1/dt
plot(yz,'g')
title('Unit Impulse Response of a Low-Pass Filter')
xlabel('sample#')
ylabel('Amplitude')

%                                  UNIT STEP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure; hold;
% Compared are:
% 1. The analog continuous time approach using the Laplace transform
%    for the impulse response we obtain: 1=RCsH(s)+ H(s)
%    after the inverse transform this is: h(t)=(1/RC)*exp(-t/RC)
%    to compare with later discrete time approached we assume:
sample_rate=1000;
dt=1/sample_rate;
time=0.1;

i=1;
for t=dt:dt:time;
   yh(i)=1-exp(-t/RC);
   i=i+1;
end;
plot(yh,'k')
```

```
% 2.   The difference equation mode
%      The difference equation: x(n*dt)=RC[(y(n*dt)-y(n*dt-1*dt))/dt] +
       y(n*dt)
%      for the algorithm we set n*dt to n and obtain x(n)=[RC/dt]*
[(y(n)-y(n-1))]+y(n)
A=RC/dt;
x=ones(1,100);
y_previous=0;

for n=1:100;
   y(n)=(A*y_previous+x(n))/(A+1);
   y_previous=y(n);
end;
plot(y,'r')
title('Unit Step Response of a Low-Pass Filter')
xlabel('sample#')
ylabel('Amplitude')
```

Note that in the script above we corrected the unit impulse amplitude for the discrete time cases. For a sample interval dt, the amplitude correction is 1/dt; by applying this correction we obtain an impulse with unit area dt × 1/dt (see also Fig. 2.4A).

17.3 THE FREQUENCY CHARACTERISTIC

The calculated amplitude ratio of the frequency characteristic of a low-pass filter is depicted in Fig. 17.2B. The data in this figure are based on a filter with $\tau = 33$ ms.

In the frequency characteristic shown in Fig. 17.2B we can see that it would be difficult to objectively delimit the precise bands that define the filter specification (see Fig. 15.1). ***For this reason the transition from pass band to the transition band is conventionally (though arbitrarily) taken to be the so-called −3-dB point;*** this point corresponds with the frequency where the power of the output/input ratio is equal to ½. As we saw in Chapter 3, this attenuation of ½ in the power ratio can be expressed in decibels (Eq. 3.12):

$$10 \log_{10}\left(\frac{1}{2}\right) = 20 \log_{10}\left(\frac{1}{\sqrt{2}}\right) \approx -3 \text{ dB} \tag{17.1}$$

For the low-pass RC filter we studied in Chapters 15 and 16, the frequency response is (Eq. (16.4) or (16.6)):

$$Y(j\omega) = H(j\omega) = \frac{1}{1 + RCj\omega} \tag{17.2}$$

Eq. (17.2) includes a complex number that can be split into real and imaginary components by multiplying through by an appropriately chosen fraction equal to 1:

$$\begin{aligned} H(j\omega) &= \frac{1}{1 + RCj\omega} \times \frac{1 - RCj\omega}{1 - RCj\omega} = \frac{1 - RCj\omega}{1^2 + (RC\omega)^2} \\ &= \frac{1}{1 + (RC\omega)^2} - j\frac{RC\omega}{1 + (RC\omega)^2} \end{aligned} \tag{17.3}$$

The magnitude of $H(j\omega)$ (i.e., $|H(j\omega)|$) reflects the amplitude ratio between the filter output and input in the frequency domain. Defining a and jb as the real and imaginary parts of $H(j\omega)$ (Fig. 17.3), we can calculate the power ratio of output/input as the squared amplitude ratio, i.e.,:

$$|H(j\omega)|^2 = H(j\omega)H(j\omega)^* = (a + jb)(a - jb) = a^2 - (jb)^2 = a^2 + b^2$$

The * indicates the complex conjugate. Combining this with Eq. (17.3):

$$\begin{aligned} a^2 + b^2 &= \left[\frac{1}{1 + (RC\omega)^2}\right]^2 + \left[\frac{RC\omega}{1 + (RC\omega)^2}\right]^2 = \frac{1 + (RC\omega)^2}{\left[1 + (RC\omega)^2\right]^2} \\ &= \frac{1}{1 + (RC\omega)^2} \end{aligned} \tag{17.4}$$

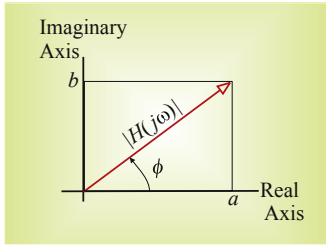


FIGURE 17.3 Argand diagram of a frequency response function (*red arrow*) at a given frequency ω. The frequency response is a complex-valued number $a + jb$, which can also be represented in polar coordinates by magnitude $|H(j\omega)|$ and phase ϕ.

Following the definition of the −3-dB point, the expression in Eq. (17.4) at the transition must equal ½: i.e., the angular frequency ω or the frequency f corresponding with this −3-dB transition is:

$$\begin{aligned}\frac{1}{1+(RC\omega)^2}=\frac{1}{2}\rightarrow 1+(RC\omega)^2=2\rightarrow RC\omega=1\\ \rightarrow \omega=2\pi f=\frac{1}{RC}\rightarrow f=\frac{1}{2\pi RC}=\frac{1}{2\pi\tau}\end{aligned} \tag{17.5}$$

Eq. (17.5) relates the value of the time constant ($\tau = RC$) of the transient response with the −3-dB point of the frequency characteristic of the RC filter. Remember again that the −3-dB point represents the frequency where the power is attenuated by a factor of 2 (Eq. 17.1), the amplitude is therefore attenuated by a factor of $\frac{1}{\sqrt{2}}$, i.e., at $\frac{1}{2}\sqrt{2}\approx 0.71$ of the input amplitude. If we define $\omega_{-3\text{dB}}=(RC)^{-1}$ and using Eq. (17.4), the power ratio $|H(j\omega)|^2$ can be written as:

$$|H(j\omega)|^2=\frac{1}{1+(\omega/\omega_{-3\text{dB}})^2} \tag{17.6a}$$

or using $\omega = 2\pi f$ and $\omega_{-3\text{dB}} = 2\pi f_{-3\text{dB}}$:

$$|H(j\omega)|^2=\frac{1}{1+(f/f_{-3\text{dB}})^2} \tag{17.6b}$$

This shows that the simple RC-circuit behaves as a ***Butterworth filter*** (see Chapter 18, Sections 18.5 and 18.6). Specifically, Eq. (17.6) represents a first-order roll-off filter that attenuates with a slope (roll-off) of ∼6 dB per octave. An octave is a doubling of frequency; using Eq. (17.6) it can be seen that, in the given low-pass filter setup, doubling of the frequency results in an increased attenuation of the output. Let's use an example with a cut-off frequency $f_{-3\text{dB}} = 10$ Hz and evaluate what happens to the attenuation factor at a series of 10, 20, 40, 80, 160 Hz, etc. Using these values in Eq. (17.6b) we get the following series of values for $|H(j\omega)|^2$: 1/2, 1/5, 1/17, 1/65, 1/257, etc. This series shows that (at the higher values for f) doubling of the frequency causes the power, $|H(j\omega)|^2$, to change with a ratio of ∼¼. This ratio corresponds with $10\times\log_{10}(¼)\approx -6$ dB, hence the 6-dB/octave roll-off characteristic. Note that this roll-off slope of an amplitude reduction of two (a power reduction of four) at doubling the frequency corresponds to an amplitude reduction of 10 for a frequency increase of 10×. Thus 6-dB/octave roll-off is the same as a slope of 20 dB/decade.

In the experimental evaluation in Chapter 15, we found that filters behave as linear systems, generating a sinusoidal output of the same frequency ω as any sinusoidal input. Generally, if one determines the response of a linear system (filter) to a sine wave $A\sin(\omega t + \phi)$, the only parameters that vary between output and input are the amplitude A and the phase ϕ. This aspect of linear systems is frequently summarized in a ***Bode plot*** or a ***Nyquist plot***. Representing the frequency response $H(j\omega)$ as a complex function $a + jb$ (with a and b representing the real and imaginary parts), we have:

$$\text{Gain} = \frac{A_{out}}{A_{in}} = |H(j\omega)| = \sqrt{a^2 + b^2}, \text{ and} \tag{17.7a}$$

$$\text{Phase} = \phi = \tan^{-1}\left(\frac{b}{a}\right) \tag{17.7b}$$

In Eq. (17.7a), gain may also represent attenuation (i.e., gain < 1); $\frac{A_{out}}{A_{in}}$ is the ratio between the amplitudes at the output and input; ϕ is the phase shift between output and input. We can also represent the complex number $H(j\omega)$ in polar form:

$$H(j\omega) = |H(j\omega)|e^{j\phi} \tag{17.8}$$

A diagram of the polar representation for a single frequency ω is shown in Fig. 17.3. The equations in Eq. (17.7) and the expression in Eq. (17.8) are essentially equivalent in the sense that they fully specify the frequency characteristic of the RC filter with a complex value (such as that depicted in Fig. 17.3) for each frequency ω.

Eq. (17.7) is the basis for the so-called Bode plot, where the frequency characteristic is represented by two separate plots. One of the plots describes the amplitude ratio between output and input $|H(j\omega)|$ versus frequency (as in Figs. 17.2B and 17.4A). A second plot describes the phase ϕ versus frequency (Fig. 17.4B). Usually the abscissa of a Bode plot is a $\log_{10}$ axis of frequency $f(=\omega/2\pi)$. Another representation of the same information is the Nyquist plot, which depicts the $H(j\omega)$ function (Eq. 17.8) in a polar plot (Fig. 17.4C). The advantage of the Nyquist plot over the Bode plot is that all information is contained in a single plot instead of two; the disadvantage is that the frequency axis isn't explicitly included. In most cases an arrow in the Nyquist plot (as in Fig. 17.4C) indicates the direction in which the frequency increases, allowing for a qualitative assessment of the frequency-related output−input relationship.

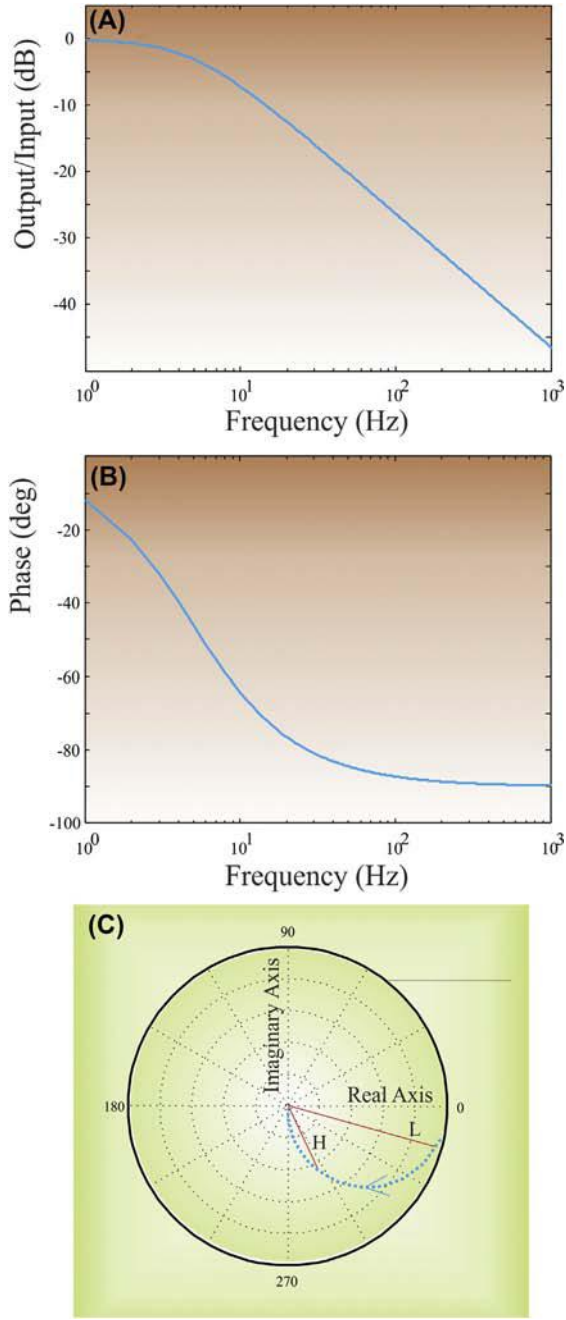


FIGURE 17.4 Filter characteristic of the RC-circuit low-pass filter. The Bode plot: (A) amplitude ratio and (B) the phase relationship between output and input. From the graphs in the Bode plot, the amplitude ratio and phase can be determined for each frequency. (C) An example of a Nyquist diagram that shows the output–input relationship (*blue dotted line*) of the same filter in a polar plot. The Nyquist diagram shows the frequency characteristic (such as that depicted in Fig. 17.3) as a complex-valued parametric function of frequency. In this type of plot specific frequency values cannot be determined; the *blue arrow* indicates the direction in which the frequency increases. In this example we indicate a low frequency (L) with an output/input ratio close to one and a small change in phase. The high frequency (H) is associated with a smaller ratio (it is a low-pass filter characteristic) and a more significant change in phase.

The following MATLAB® program can be used to produce the graphs shown in Fig. 17.4.

```
% pr17_2.m
% Bode_Nyquist.m
% Bode Plot and Nyquist Plot for a low pass filter

% Filter Components
R=10e3;
C=3.3e-6;

% Formula for amplitude (A) = 1/sqrt[1 + (RCw)^2] with w=2 x pi x f
for i=1:5000;
   f(i)=i;
   A(i)=1/(sqrt(1+(R*C*2*pi*f(i))^2));        % formula derived for the
                                                 absolute part
   H(i)=1/(1+R*C*2*pi*f(i)*j);                % frequency response
   rl=real(H(i));                             % real part of H
   im=abs(imag(H(i)));                        % magnitude of the imaginary
                                                 part of H
   PHI(i)=atan2(im,rl);                        % phase
end;

% for w=1/RC there is A=1/sqrt[1/2] ~ 0.7
% 20 x log10{[1/sqrt(2)]} ~ -3.0 (The -3dB point)

F_3db=1/(2*pi*R*C);       % Here we use frequency F (=w/(2 x pi) )

figure
subplot(3,1,1), semilogx(f,20*log10(A))
xlabel('Frequency(Hz)')
ylabel('Amplitude Ratio (dB)')
axis([0 1000 -50 0]);

t=['BODE PLOT Low Pass Filter: R = ' num2str(R) ' Ohm; C = ' num2str(C) ' F;
and -3dB frequency = ' num2str(F_3db) ' Hz'];
title(t)
subplot(3,1,2),semilogx(f,(PHI*360)/(2*pi))
xlabel('Frequency(Hz)')
ylabel('Phase (degrees)')
axis([0 1000 0 100]);

subplot(3,1,3),polar(PHI,A)
xlabel('Real')
ylabel('Imaginary')
title('Nyquist Plot')
```

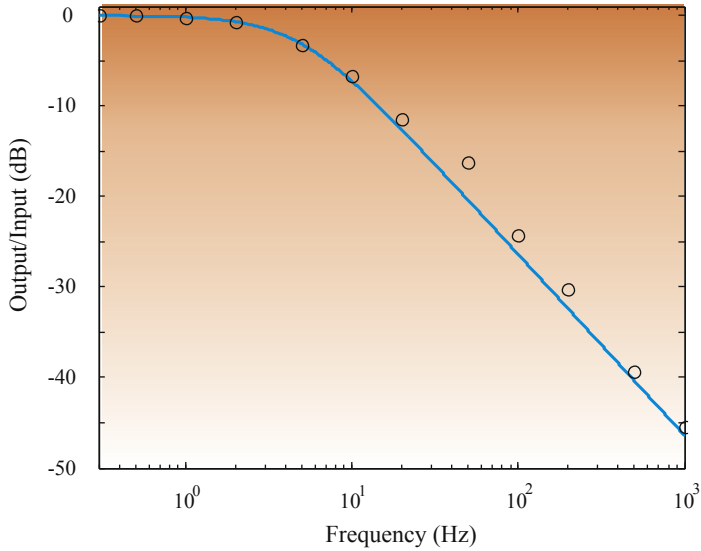


FIGURE 17.5 The amplitude ratio part of the Bode plot for a low-pass filter. The *open circles* are the measured values from the experiment described in Chapter 15, the *blue line* is the theoretically derived response $|H(j\omega)|$, Eq. (17.7a). The discrepancy between measured and calculated data is due to measurement error made by the author.

Finally we can relate our experimental findings we obtained in Chapter 15 in which we measured the response of the RC filter to sinusoidal inputs. The ratios of output amplitude to input amplitude we calculated there represent the output/input ratio graph $|H(j\omega)|$ of the Bode plot. These measurements can be compared with the theoretical expectation based on the known resistance and capacitance values. The MATLAB® program pr17_3.m shows our experimental results superimposed on the theoretical curve; you can use pr17_3.m to plug in your own recorded values; your results should look similar to Fig. 17.5.

17.4 NOISE AND THE FILTER FREQUENCY RESPONSE

In the previous sections we analyzed an RC filter's response to either a transient signal (such as δ or U) or a sine wave. The autospectrum (= power spectrum) of the response to white noise input can also be used to obtain the frequency characteristic. In the frequency domain, truly white noise represents all frequencies. Because the noise is random, subsequent samples are unrelated and its autocorrelation function is a delta function: i.e., correlation equal to 1 at zero lag,

and 0 elsewhere. The Fourier transform of this autocorrelation function represents the power spectrum (Chapter 13, Section 13.4.2), and the transform of a delta function is a constant (Eq. 6.9). Therefore a sufficiently long epoch of white noise provides all the frequencies to the filter's input, similar to feeding it sinusoidal signals for a range of frequencies as we discussed above. The output of the filter will be "colored" noise, meaning that frequency components will be attenuated in the transition and stop bands of the filter. At first sight the noise approach may seem a bit sloppy, but the results from this approach can easily be compared with other mathematical techniques: (1) the Fourier transform of the impulse response in continuous time systems or (2) by substituting $e^{j\omega t}$ for z in the z-transform of the impulse response in discrete time systems (see also Chapter 18, Section 18.4). You may run script pr17_4.m to compare the different techniques. Note that the result from the noise input technique may slightly vary each time that you run the script.

The following is a MATLAB® script that demonstrates the different techniques for obtaining the output/input ratio for a digital filter.

```
% pr17_4.m
% Two point Smoothing Filter's Frequency Response
% filter equation for the digital (FIR) filter:
%         y(n)=(x(n)+x(n-1))/2

clear;
wT=0:.1:2*pi;

% 1. Calculate the abs component of the fft of the impulse response
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The impulse response y for an impulse at time=0
% equals to pulse of .5 at time=0 and at time=T,
% i.e.
y=zeros(1,63);
y(1)=.5;y(2)=.5;

Y=fft(y);                                   % fft of the impulse response y
                                            % –> frequency response Y
figure
subplot(2,1,1),plot(wT,abs(Y))
title('Frequency Response = fft of the Impulse Response')
axis([0 max(wT) 0 max(abs(Y))]);
ylabel('Amplitude Ratio');
```

```
% 2. The second method is to use the z-transform and replace z by exp(jwT)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The z-transform of y(n)=(x(n)+x(n-1))/2 is:
%                    Y(z)=.5* X(z)*[1+1/z]
%                -->  H(z)=Y(z)/X(z)=.5 + .5*(1/z) = .5 + .5*exp(-jwT)
YY=(.5+.5*(exp(-j*wT)));
subplot(2,1,2),plot(wT,abs(YY))
title('Frequency Response = based on z-transform')
axis([0 max(wT) 0 max(abs(YY))]);
ylabel('Amplitude Ratio');
xlabel('Frequency (wT: Scale 0-2pi)');      % NOTE: Normally one would
                                               show 0-pi
                                            % with pi=the Nyquist
                                               frequency

% 3. The third method is to use white noise and compare the power spectra
     of in- and output
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Because white noise represents all frequencies at the input and the output
% shows what is transferred. The output over the input power spectra
% represent a frequency response estimate.

x=randn(10000,1);                           % create white noise
wT=1:length(x);wT=(wT/length(x))*2*pi;      % New Frequency Scale

for n=2:length(x);                          % Calculate the output of
                                               the 2-point
                                            % smoothing
  y(n)=(x(n)+x(n-1))/2;
end;

figure   % plot the input x
subplot(3,1,1),plot(x)
hold
subplot(3,1,1),plot(y,'k')
title('Input Noise (blue) and Output Noise (black)')
xlabel('sample #')
ylabel('amplitude')

X=fft(x);        % Calculate the power spectra
Y=fft(y);        % NOTE: The power spectrum is the
                 % fft of the autocorrelation
```

```
Px=X.*conj(X)/length(x);
Py=Y.*conj(Y)/length(y);

subplot(3,1,2), plot(wT,Px)
hold
subplot(3,1,2),plot(wT,Py,'k')
title('POWER SPECTRA Input Noise (blue) and Output Noise (black)')
xlabel('Frequency Scale (0-2pi)')
ylabel('power')

for i=1:length(x);
   % Strictly speaking abs(X(i)*conj(Y(i)))/(input variance * N) should be
   % sufficient; here we divide by (X(i)*conj(X(i)) to correct for
   % non whiteness (colored) of the Gaussian noise input
   h_square(i)=(abs(X(i)*conj(Y(i)))/(X(i)*conj(X(i))))^2;
end;
subplot(3,1,3),plot(wT,sqrt(h_square),'k')
title('Frequency Response = based on Input-Output white Noise')
xlabel('Frequency Scale(0-2pi)')
ylabel('Amplitude Ratio')
```

EXERCISES

17.1 We have a discrete time filter defined by

$$y(n) = a_0x(n) + a_1x(n-1) + a_2x(n-2)$$

a. Determine the filter's transfer function.
b. Draw a diagram as in Fig. E16.1 for this filter.
Now assume some numerical values for a_0—a_2 (e.g., $^1/_3$, $^1/_3$, $^1/_3$ and 1, −2, 1).
c. Determine the filter's frequency response (hint: use its transfer function, the z-transform, and substitute $z = \exp(-j\omega t)$ with $\omega = 2\pi f$ to obtain a function in the frequency domain).
d. Plot the filter's Bode and Nyquist plots (use MATLAB®).
e. Determine the filter's unit impulse response and its step response.

17.2 Calculate the frequency response for the RC high-pass filter data described in Chapter 15.
   a. Show the steps of your calculation.
   b. Compare your result with results you measured for Exercise 15.1 or the measured data in Table 15.1.
   c. Plot both the theoretical and measured data in a single semilog plot.

CHAPTER

18

Filters: Digital Filters

18.1 INTRODUCTION

With currently available fast processors and dedicated DSP hardware, most biomedical instruments perform at least some filter operations in the digital domain (Chapter 2). In principle this makes filtering more flexible; a different frequency response can be obtained with a simple change of parameters instead of requiring an alteration to the hardware. At first glance, it would seem that filtering in a ***digital*** world would allow arbitrary attenuation of undesired frequencies in the frequency domain representation of a signal. Unfortunately, there are limitations to this approach, since such manipulations in the frequency domain can introduce serious oscillations in the filter's response as well as unwanted transients in the time domain (Appendix 16.1).

18.2 INFINITE IMPULSE RESPONSE AND FINITE IMPULSE RESPONSE DIGITAL FILTERS

In our analysis of continuous-time LTI systems, we used a ***rational function*** to describe the input−output relationship in the time domain, the frequency domain and the s (Laplace) domain (Chapters 12 and 13). In discrete time, we can use the same approach for the sampled function using the z domain instead of the s domain, where we use time-delayed values instead of derivatives to characterize the evolution of the system. For a system with input $x(n)$ and output $y(n)$ with $n = 0, 1, 2,\ldots$:

$$\sum_{k=0}^{M} a_k y(n-k) = \sum_{k=0}^{N} b_k x(n-k) \tag{18.1}$$

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00018-7

with a_k and b_k as the parameters that determine the filter's characteristic. The z transform of the above can be used to find the transfer function:

$$Y(z)\sum_{k=0}^{M} a_k z^{-k} = X(z)\sum_{k=0}^{N} b_k z^{-k} \rightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} b_k z^{-k}}{\sum_{k=0}^{M} a_k z^{-k}} \tag{18.2}$$

In some texts, the numerator and denominator are divided by a_0 (the coefficient of $y(n)$); this results in the expression:

$$H(z) = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 + \sum_{k=1}^{M} a_k z^{-k}} \tag{18.3}$$

If a filter output depends on the previous output (i.e., $a_k \neq 0$ for $k \geq 1$), the response to an impulse at $n = 0$, never completely disappears, but continues to reverberate through the system indefinitely. Because the impulse response continues forever ($n \rightarrow \infty$), this type of algorithm represents a so-called infinite impulse response (***IIR***) filter. In the case where $a_k = 0$ for $k \geq 1$, the output only depends on a finite set of input terms. Thus the impulse response of this filter is finite: a finite impulse response (***FIR***) filter.

If we factor the polynomials in the numerator and denominator of Eq. (18.2), the rational function can also be fully characterized by a constant gain factor (K) plus the zeros of the numerator (z_k) and the zeros of the denominator, the so-called poles (p_k):

$$H(z) = \frac{Y(z)}{X(z)} = K\frac{(z^{-1} - z_1)(z^{-1} - z_2)\ldots(z^{-1} - z_N)}{(z^{-1} - p_1)(z^{-1} - p_2)\ldots(z^{-1} - p_M)} \tag{18.4}$$

It is easy to see that $H(z)$ is undefined at the poles, meaning an output/input ratio that explodes towards infinity. For a system to be stable it must not have any poles in the so-called region of convergence (ROC, Appendix 12.2). Since the IIR filter equation includes poles it is potentially unstable. In contrast, the FIR filters have no poles and are always stable (as long as the filter's input is stable of course).

18.3 AUTOREGRESSIVE, MOVING AVERAGE, AND ARMA FILTERS

An alternative classification of digital filters is based on the type of algorithm that is associated with the filter.

1. Autoregressive (***AR***) filters have a dependence on previous output, and therefore are characterized by an IIR. An example of such a (potentially unstable, depending on the coefficients) filter is:

$$y(n) = Ay(n-1) + By(n-2) \quad (A \text{ and } B \text{ are constants}) \tag{18.5}$$

2. Moving average (***MA***) filters only depend on the input and therefore have an FIR. An example of a moving average filter is:

$$y(n) = \frac{x(n) + x(n-1) + x(n-2)}{A} \quad (A \text{ is a constant}) \tag{18.6}$$

3. The combination of AR and MA is the ***ARMA*** filter that depends both on previous output and input. The ARMA filter has an IIR because previous output is involved. An example of such a filter type is:

$$y(n) = Ay(n-1) + Bx(n) + Cx(n-1) \quad (A,\ C \text{ and } B \text{ are constants}) \tag{18.7}$$

As can be seen in the above, the AR, MA, and ARMA classification overlaps with the IIR and FIR terminology.

18.4 FREQUENCY CHARACTERISTICS OF DIGITAL FILTERS

The steps to transform a digital filter representation from the discrete time domain to the z-domain were shown earlier (e.g., Eqs. 18.1 and 18.2). The z transform of the output/input ratio (the transfer function) is closely related to the system's frequency response. In a digital filter's transfer function, such as Eq. (18.2), the variable z represents $e^{s\tau}$ (Chapter 12, Section 12.5.2), where s is a complex variable with a real component σ and imaginary component $j\omega$ (Chapter 12, Section 12.3). For the frequency

response we are interested in the imaginary, frequency-related, part of the transfer function. Therefore we can determine the frequency response of a digital filter by substituting $e^{j\omega\tau}$ for z in its transfer function.

This procedure was followed to obtain the frequency response in the example illustrated in pr17_4.m. In the following, we analyze an example of a three-point smoothing (MA, FIR) filter in Eq. (18.6) with $A = 3$. The z-transform of the time domain equation is $Y(z) = \frac{X(z)(1+z^{-1}+z^{-2})}{3}$, generating a transfer function:

$$\frac{Y(z)}{X(z)} = H(z) = \frac{\left(1 + z^{-1} + z^{-2}\right)}{3} \tag{18.8}$$

Now we multiply the numerator and denominator by z, substitute $e^{j\omega\tau}$ for z, and use Euler's relation for $\cos(\omega\tau)$:

$$H(z) = \frac{\left(z + 1 + z^{-1}\right)}{3z} \rightarrow H(j\omega) = \frac{\left(e^{j\omega\tau} + e^{-j\omega\tau} + 1\right)}{3e^{j\omega\tau}} = \frac{e^{-j\omega\tau}}{3}\left[2\cos(\omega\tau) + 1\right] \tag{18.9}$$

Remember that τ can be considered as the sample interval. This means that $1/\tau$ is the sample rate, $1/(2\tau)$ is the Nyquist frequency for the filter in Hz, and π/τ is the Nyquist frequency in rad/s. From the complex function in Eq. (18.9) we can construct the Bode plot for values of ω ranging between 0 and π/τ rad/s. Use the following commands to calculate the expression in Eq. (18.9) and to plot the output/input amplitude ratio of the Bode plot in MATLAB®:

```
tau=1;                                  % sample interval
w=0:0.01:pi/tau;                        % rad Freq up to Nyquist
amp_ratio=abs((exp(-j*w*tau)/3).*(1+2*cos(w*tau)));
loglog(w,amp_ratio)                     % plot the result in log scales
```

If you prefer evaluating the result on a linear scale, you can use plot(w,amp_ratio) instead of the loglog command; the result you obtain using the plot command is shown in Fig. 18.1. From the plot that is generated by these commands, it easy to see that the three-point smoothing function behaves as a low-pass filter. Although this FIR filter is stable, the amplitude ratio of the frequency characteristic is far from ideal because there is a large side lobe above ($2\pi/3 \approx 2.1$ rad/s).

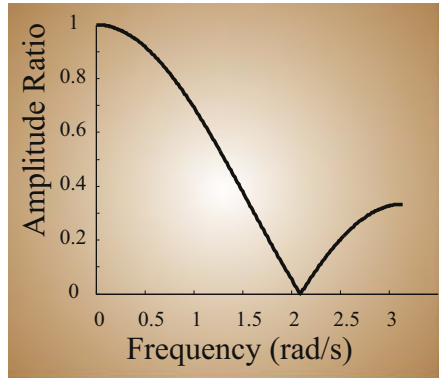


FIGURE 18.1 Frequency characteristic of a three-point smoothing filter. The amplitude ratio plotted against frequency.

18.5 MATLAB® IMPLEMENTATION

The commands discussed in the following paragraphs are included in the MATLAB® "Signal Processing" toolbox. It is important to note that unlike most textbooks, MATLAB®'s vector indices start at 1 and not at 0!

The ***filter*** command requires the a_k (A) and b_k (B) coefficients for the digital filter operation on input vector (e.g., X); the result is placed in another vector (e.g., Y). The following text shows the MATLAB® help information for the filter command (type: help filter).

```
FILTER One-dimensional digital filter.
   Y = FILTER(B,A,X) filters the data in vector X with the
   filter described by vectors A and B to create the filtered
   data Y.  The filter is a "Direct Form II Transposed"
   implementation of the standard difference equation:

   a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
                         - a(2)*y(n-1) - ... - a(na+1)*y(n-na)

   If a(1) is not equal to 1, FILTER normalizes the filter
   coefficients by a(1).

   FILTER always operates along the first non-singleton dimension,
   namely dimension 1 for column vectors and non-trivial matrices,
   and dimension 2 for row vectors.
```

```
[Y,Zf] = FILTER(B,A,X,Zi) gives access to initial and final
conditions, Zi and Zf, of the delays.  Zi is a vector of length
MAX(LENGTH(A),LENGTH(B))-1 or an array of such vectors, one
for
each column of X.

FILTER(B,A,X,[],DIM) or FILTER(B,A,X,Zi,DIM) operates along
the
dimension DIM.

See also FILTER2 and, in the Signal Processing Toolbox, FILTFILT.
```

Reprinted with permission of The MathWorks, Inc.

The vectors A and B contain the a_k and b_k coefficients that can be obtained directly or indirectly. For instance, if one wants to implement a filter:

$$y(n) - y(n-1) + 0.8y(n-2) = x(n) \tag{18.10}$$

The A and B coefficient vectors are:

$$\mathrm{B} = [1] \;\; and \;\; \mathrm{A} = [1, -1, 0.8]$$

However, in most cases you don't know the A and B coefficients explicitly, and you have to instead start from a filter specification in the frequency domain. For instance, we want to implement a bandpass filter that passes frequencies between 1 and 30 Hz. Suppose we are interested in implementing this by using a Butterworth filter (a special filter type, see also Section 18.6). We could do this the hard way by deriving the filter's transfer function and translating this into the discrete domain (Appendix 18.1). However, MATLAB® allows one to determine the coefficients more easily using the butter command (type: help butter). The following shows a part of the MATLAB® help text.

```
BUTTER Butterworth digital and analog filter design.
  [B,A] = BUTTER(N,Wn) designs an Nth order lowpass digital
  Butterworth filter and returns the filter coefficients in length
  N+1 vectors B (numerator) and A (denominator). The coefficients
  are listed in descending powers of z. The cutoff frequency
  Wn must be 0.0 < Wn < 1.0, with 1.0 corresponding to
  half the sample rate.

  If Wn is a two-element vector, Wn = [W1 W2], BUTTER returns an
  order 2N bandpass filter with passband  W1 < W < W2.
```

```
[B,A] = BUTTER(N,Wn,'high') designs a highpass filter.
[B,A] = BUTTER(N,Wn,'stop') is a bandstop filter if Wn = [W1 W2].

When used with three left-hand arguments, as in
[Z,P,K] = BUTTER(...), the zeros and poles are returned in
length N column vectors Z and P, and the gain in scalar K.

......
......
```

Reprinted with permission of The MathWorks, Inc.

Recall that the output of a digital filter depends on delayed values of its input and output (Eq. 18.1). Note that for a digital filter, its order is defined as the maximum delay used in the calculation of the filter's output. Therefore, as can be seen in the specification above, the order of a bandpass or bandstop filter is doubled as compared to a low- of high-pass filter. This order is therefore not necessarily the same as the roll-off order of 6 dB/octave we discussed in Chapter 17.

Suppose we sampled our data at 400 Hz → the Nyquist frequency of the signal is 200 Hz. This means our bandwidth parameters should be 1/200 to 30/200. The command: `[b,a] = butter(2, [1/200, 30/200])` produces the desired coefficients for a fourth-order filter. The coefficients can be used in the filter command in order to bandpass a signal sampled at 400 Hz between 1 and 30 Hz.

Similarly, `[b,a] = butter(6, [(60-5)/200,(60+5)/200],'stop')` produces a set of coefficients that attenuate a 60-Hz noise component (a 12th-order band reject filter between 55 and 65).

Another helpful feature in the Signal Processing Toolbox is the `freqz` command. This allows us to construct a ***Bode plot*** from the filter characteristic in the z-domain. The plot is made on the basis of the coefficients A and B:

```
freqz(b,a,100,400)
```

In the above command, we pass parameters for the precision of the calculation (in this case 100 points), and the sample frequency (400 in our example). The resulting Bode plot is shown in Fig. 18.2A.

The command `impz` shows the associated ***impulse response***, e.g., `impz(b,a,100)` shows the impulse response of the first 100 points (Fig. 18.2B).

The file hum.mat (available on the http://booksite.elsevier.com/9780128104828/; to load, type: `load hum`) contains an epoch of EEG

sampled at 256 Hz with a large 60-Hz component (after loading hum.mat, the data are stored in a variable called eeg). To attenuate this unwanted interference we can use a 60-Hz stop band filter (notch filter): [b,a] = butter(6, [(60-5)/128,(60+5)/128],'stop')

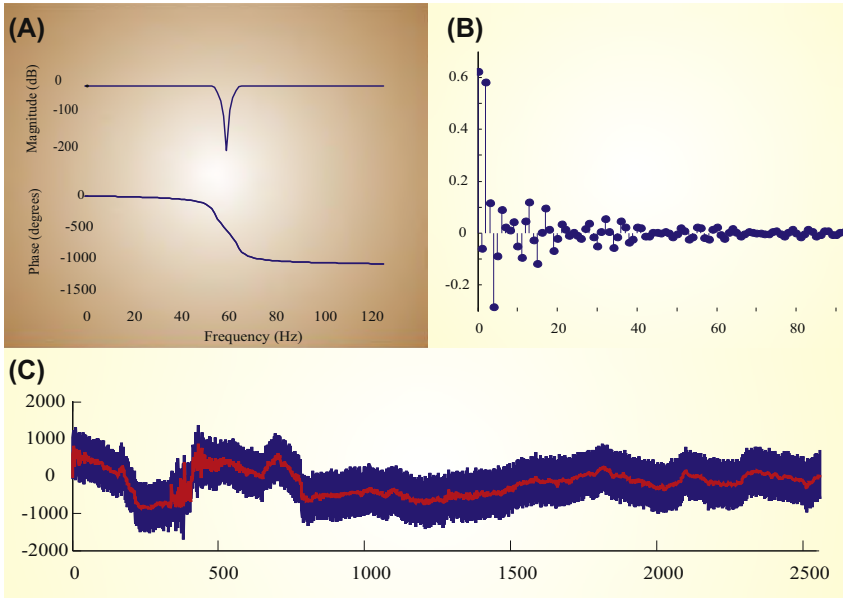


FIGURE 18.2 Example of a band-reject filter to remove 60-Hz hum. This type of filter is often called a notch filter. The graphs in (A) represent the Bode plot (the filter characteristic) and (B) shows the first part of the filter's impulse response. In this example, the full impulse response cannot be shown because this is an IIR type filter. (C) EEG with 60-Hz noise (blue) and the trace that was filtered using the notch filter (red) superimposed.

Now type in the following commands:

```
freqz(b,a,100,256)
figure
impz(b,a,100)
figure
plot(eeg)
hold

eegf=filter(b,a,eeg);
plot(eegf,'r')
```

The raw and filtered EEG data are shown in Fig. 18.2C.

Note: The success of a 60-Hz band reject filter should not be used as an excuse to record poor-quality data with lots of hum. First, ideal filters do not exist, therefore the attenuation is never complete. Second, 50/60-Hz notch filters have a tendency to produce oscillatory artifacts at discontinuities in the input signal.

18.6 FILTER TYPES

Thus far we have used the Butterworth filter as the basis for most of our analysis. As we saw in the Bode plot (Chapter 17), the characteristics of the Butterworth filter are not ideal; the transition band is fairly wide and the phase response is frequency-dependent (e.g., Fig. 17.4). Because the ideal filter cannot be made (Appendix 16.1), we always need to compromise in our approach to the ideal filter characteristic. This compromise may vary with each application. In some cases strong attenuation of noise is required, but phase response is not critical; in other cases, where we want to accurately measure delays, the phase response is critical. Not surprisingly, in the real world there is a tradeoff between a small transition band, also known as a steep roll-off, and a favorable (flat) phase response.

The different filter types realizing different compromises that are available in MATLAB® are summarized in Table 18.1. It can be seen that the Butterworth is a good compromise, realizing both a reasonable roll-off and phase response. The Butterworth filter's magnitude response $|H(j\omega)|$ is flat in the pass band and monotonic overall. The Bessel and Elliptic filter types are at the extreme ends of the tradeoff scale, realizing either a good phase response or a steep roll-off, respectively. Because Bessel filters are characterized by an almost constant delay for the frequencies across the passband, they preserve the wave shape of the filtered signal in the time domain. The increased roll-off of the Chebyshev and Elliptic filters comes

TABLE 18.1 Summary of Roll-Off and Phase Characteristics of Different Filter Types That Are Available in the MATLAB® Signal Processing Toolbox

Filter Type	MATLAB® Command	Roll-Off	Phase Response
Bessel	besself	−	++
Butterworth	butter	±	±
Chebyshev	cheby1, cheby2	+	−
Elliptic	ellip	++	−

at the cost of ripple in their magnitude response curves $|H(j\omega)|$. In MATLAB® there are cheby1 and cheby2 commands; the type I Chebyshev filter has a ripple in the pass band and a flat stop band, type II is the opposite with a flat pass band and ripple in the stop band. The Elliptic filter type has a magnitude response as shown in Fig. 15.1, i.e., ripple in both pass and stop bands.

18.7 FILTER BANK

In the previous text we considered filters with a single input and single output. In some applications, it is beneficial to look at the signal in a set of frequency bands. Instead of a single filter, one constructs a set of filters (a filter bank) with desired frequency responses. The filters in this bank can be applied in parallel to the same signal. This is the preferred approach if you want to explore the signal's frequency content or detect features associated with certain frequency components. As we will see, this approach is also the basis for the so-called spectrogram and scalogram representations of a time series (Chapter 22).

An interesting biomedical application of filter banks is the cochlear implant (Fig. 18.3). This instrument mimics the cochlea by separating the input (sound transduced into an electrical signal by a sensitive microphone) into separate spectral components. Physiologically it is known that the bottom part (base) of the cochlea is more sensitive to high-frequency components, whereas the top (apex) is more sensitive to low-frequency oscillations. The filter bank in the implant device mimics this normal cochlear operation and stimulates sensors connected to the auditory nerve in a pattern analogous to a normal cochlea. Of course, this procedure only works for patients whose auditory system downstream of the cochlea is intact.

18.8 FILTERS IN THE SPATIAL DOMAIN

At several instances in the text we have noted that our processing techniques on time series $x(t)$ can easily be translated into the spatial domain, such as with an intensity image conceived as a function of two spatial dimensions $I(x,y)$. Here we simply replace the time parameter t in our algorithm by a spatial variable x, y, or z; an example of such an application is described in Chapter 7, Section 7.2.

Spatial filters can be used to remove or enhance spatial frequency components. For instance, a high-pass filter can be used to enhance sudden transitions (edges) in the spatial domain while attenuating slow or gradual changes in the image. An example of such a procedure by

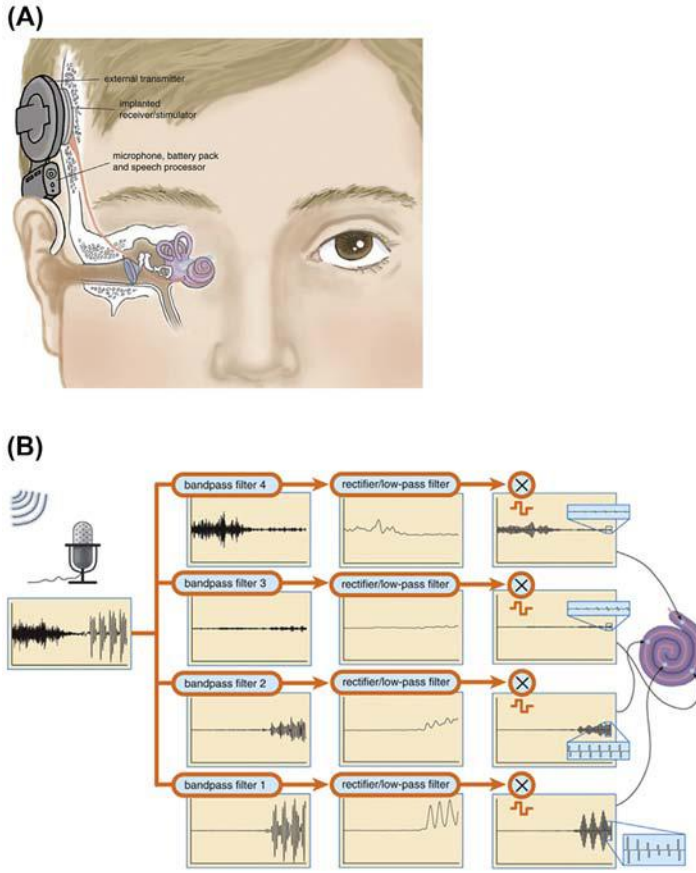


FIGURE 18.3 (A) Cochlear implants have five main components, only two of which are inside the body. A microphone above the ear senses sound waves which are directed to a small computer behind the ear. The computer transforms the signals into a coded stimulus that must be delivered to a set of electrodes that are implanted in the cochlea. These stimulus signals are (via an external transmitter) transmitted through the skin to an implanted receiver which converts them to electrical stimuli for the implanted electrodes. These stimuli excite the auditory nerve. (B) A diagram of how a filter bank is used to decompose a complex sound signal for the syllable "sa". Band-pass filters 1–4 (left column of panels) each pass a specific frequency component of the compound signal in the left panel. Filter 1 passes the lowest frequency components and filter 4 the highest ones. A set of rectifying low-pass filters (middle column of panels) subsequently creates an envelope for each of the frequency bands. This envelope signal is finally transformed into a train of biphasic pulses (right column of panels) that can be used to stimulate a specific location in the cochlea. The high-frequency components stimulate the base of the cochlea and the low frequencies stimulate nerve fibers more towards the apex. *With permission from: Dorman, M.F., Wilson B.S., 2004. The design and function of cochlear implants. Am. Sci. 92, 436–445.*

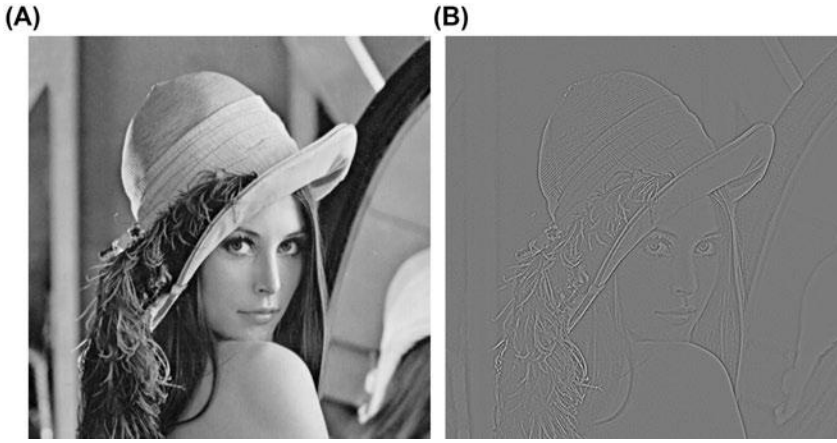


FIGURE 18.4 An example of a filter application in the spatial domain using a picture of Lena (A) as input for a 2-D Butterworth high-pass filter. Although this spatial filter isn't optimized for this application, by using the principle of high-pass filtering we can detect transitions (edges) in the spatial domain as shown in (B).

using a simple Butterworth filter is shown in Fig. 18.4, generated by MATLAB® script pr18_1.m. The image of Lena in Fig. 18.4A is commonly used to evaluate image processing algorithms because it contains a number of challenging properties that can be enhanced by signal processing techniques, and probably also because the image pleases many male image processing specialists.

The following is a part of pr18_1.m used to filter input image contained in matrix lena_double.

```
[b,a]=butter(1,100/256,'high');          % make a high-pass filter
                                           based on
                                         % a sample rate of 1 pixel and
                                         % Nyquist of 256 pixels
lenah=lena_double;

for k=1:512;
  lenah(k,:)=filtfilt(b,a,lenah(k,:));   % use filtfilt to prevent phase
                                           shift
end;
```

The part of the script shown above successively high-pass filters each horizontal line in the image, and therefore detects the vertical transitions (edges) in each row. Another part in pr18_1.m detects abrupt horizontal transitions, and the output of both filters can be added to show the edges

in the picture; such a result is shown in Fig. 18.4B. The detected edges can now be superimposed on the original picture in order to obtain an edge-enhanced image; you can run pr18_1.m to observe these effects. Applications such as the one shown with Lena's image can help you to enhance images but can also be used to detect regions of interest in optical imaging data sets such as microscopic images or movies. In Chapter 21 we describe how similar edge detection can be accomplished with a wavelet transform (Figs. 21.6 and 21.7).

APPENDIX 18.1

Comparison of the butter command in MATLAB® with the approximation from Fig. 16.3. Using the diagram in Fig. 16.3 and the description in Section 16.2, we get the following filter equation:

$$\underbrace{1}_{A(1)}\, y_n \underbrace{-\frac{(RC/\Delta t)}{(RC/\Delta t)+1}}_{A(2)}\, y_{n-1} = \underbrace{\frac{1}{(RC/\Delta t)+1}}_{B}\, x_n \qquad \text{(A18.1-1)}$$

Using $R = 10^4\ \Omega$, $C = 3.3\ \mu\text{F}$, and $\Delta t = 1/400$, we can calculate the filter coefficients (Eq. 18.1): $A = [\text{A(1)}\ \ \text{A(2)}] = [1.0000\ \ -0.9296]$ and $B = [0.0704]$. On the other hand, if we used the more precise Eq. (A16.3-2), discussed in Appendix 16.3 and repeated here in the same format as Eq. (A18.1-1) for convenience:

$$\underbrace{1}_{A(1)}\ y_n \underbrace{-e^{-\Delta t/RC}}_{A(2)}\, y_{n-1} = \underbrace{\left(1-e^{-\Delta t/RC}\right)}_{B}\, x_n \qquad \text{(A18.1-2)}$$

we get $A = [1.0000\ \ -0.9270]$ and $B = [0.0730]$. Using a first-order butter command for $f = 1/(2 \times \pi \times R \times C) = 4.8229$ Hz and a sample frequency of 400 Hz (Nyquist frequency: 200 Hz):

```
[B,A]=butter(1,4.8229/200)

B = 0.0365   0.0365

A =1   -0.9270
```

We obtain almost identical values with the difference being that B has two terms, each exactly half of the single term, i.e., 0.0730/2. This has the effect of a moving average of the input.

EXERCISES

18.1 Take the time series (y) generated by the program: pr7_1.m (see Fig. 7.2) and band-pass filter y between 45 and 55 Hz. Construct a second filter with a bandpass between 115 and 125 Hz. Depict the filtered time series versus the original in both the time domain and frequency domain (use MATLAB® functions to construct the filters).

18.2 A digital filter is characterized by the transfer function

$$H(z) = \frac{1}{\left(1 - \frac{1}{3}z^{-1}\right)\left(1 - \frac{1}{6}z^{-1}\right)}$$

a. Determine the difference equation for this filter.
b. Determine and sketch the filter's unit impulse response.

18.3 Load hum.mat, and do the following assignments:
a. Create a filter bank with the following filters: DC-4 Hz; 4–8 Hz; 8–12 Hz; 12–30 Hz; 30–100 Hz. You may use a fourth-order butterworth (i.e., $N = 2$ in the butter command).
b. Show the UIR and the Bode plot for each filter (see Fig. 18.2).
c. Apply these filters to the eeg signal (from the file hum.mat).
d. Compute and plot the amplitude spectra of the filter input and all five filter outputs.
e. Repeat (c) and (d) above after you filter the eeg with the 60-Hz band-reject filter first.
f. Interpret your results (see also Chapter 1 for EEG band definitions).

18.4 A DIY cortical implant design. The cochlear implant depicted in Fig. 18.3 consists of a filter bank, an envelope generator, and a stimulus generator. In this assignment we will mimic the design of such an implant. Start by loading some sound and play it, e.g.,:

```
load handel;
p = audioplayer(y, Fs);
play(p, [1 (get(p, 'SampleRate') * 8)]);
```

If you prefer a sound bit of another artist (including yourself if you play an instrument), go ahead and use it as long as it isn't much shorter than 5 s and sampled at a rate higher than 8 kHz.

a. Create a filter bank (as in Fig. 18.3). Say we use the following frequency bands: 50−100 Hz; 100−1000 Hz; 1000−2000 Hz; 2000−4000 Hz. You may use the fourth-order Butterworth characteristic to specify the filters (i.e., $N = 2$ in the butter command).
b. Depict the Bode plot for each filter.
c. Filter the music with each filter in the filter bank and depict input and the four outputs in a single figure. Also listen to the filtered results using the audioplayer and play commands. What do you hear?
d. Rectify (use abs in MATLAB®) and low-pass filter the filter outputs to create an envelope of the filter outputs. Show these outputs and their envelope in a single figure. You can use a first-order Butterworth as a low-pass filter and you can play with the cut-off frequency to create an envelope-like signal (similar to the envelopes shown in Fig. 18.3)
e. Create a pulse train for cochlear stimulation from each of the envelopes. Use a 1-V bipolar pulse (positive, negative) with a duration of 1 ms. Make a figure showing the input, filtered signal, envelope, and stimulus train for each filter in the filter bank.

Reference

Dorman, M.F., Wilson, B.S., 2004. The design and function of cochlear implants. Am. Sci. 92, 436−445.

CHAPTER

19

Kalman Filter

19.1 INTRODUCTION

Getting to understand a system can be quite a challenge. One approach is to create a model, an abstraction of the system. The idea is that this abstraction captures the system's important features. On the other hand, one might attempt to learn the system's behavior by measuring its "spontaneous" output or its input–output relationship. A pessimist would say that all models are wrong since they only represent part of the real thing, and all measurements are wrong too because they are also incomplete and noisy as well. Unfortunately, the pessimist is correct with this assessment. The Kalman filter approach represents a more optimistic view; this technique is to optimally combine the not so perfect information obtained via both modeling and measurements in order to gain the best possible knowledge about a system's state. This filter type is an example of a so-called least-square filter. As we will show in Section 19.3, this procedure separates signal and noise using a minimum square error fit. This principle for signal and noise separation was first introduced by Norbert Wiener in the late 1940s. Although Wiener's approach was novel and appealing, it was not easy to apply in practical engineering problems. More than a decade later, in 1960, Rudolf E. Kalman introduced the basis for an algorithm using the state space method, and his approach, although not initially recognized, provided a practical procedure to implement the novel least-square filtering method.

The basic idea is that we perform multiple (probably regular) measurements of a system's output and that we use these measurements and our knowledge of the system to reconstruct the state of the system. Practically, the Kalman filter process consists of the following steps:

1. We make an a priori estimate of the state of the system before a measurement is made.

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00019-9

2. Subsequently, after we have made the measurement, we compute a new a posteriori estimate by fusing the a priori prediction with this latest measurement.

In our examples we will assume trivial dynamics: i.e., our prediction of the next state is the current state. Of course, when your system dynamics is more complex, the prediction step 1 must reflect this complexity, but it wouldn't change step 2, i.e., the procedure of the assimilation/fusion of a prediction with a measurement.

For more in-depth treatment of the Kalman filter, see the texts of Brown and Hwang (1997), Maybeck (1982), and, especially for neuroscience applications, Schiff (2012). The website from Welch and Bishop listed in the References is also very useful.

19.2 INTRODUCTORY TERMINOLOGY

This section briefly summarizes the most important terminology required to develop the Kalman filter procedure in Section 19.3.

19.2.1 Normal or Gaussian Distribution

Although Kalman filter versions that deal with non-Gaussian noise processes exist, the noise components in the Kalman filter approach described in this chapter are Gaussian white noise terms with zero mean. Recall that the probability density function (PDF) of the normal or Gaussian distribution is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{19.1}$$

μ—mean (zero in our case), and σ—standard deviation.

19.2.2 Minimum Mean-Square Error

An optimal fit of a series of predictions with a series of observations can be achieved in several ways. Using the minimum mean-square error (MMSE) approach, one computes the difference between an estimate $\widehat{y}$ and a target value y. The square of this difference, $(\widehat{y} - y)^2$ gives an impression of how well the two fit. In case of a series of N estimates and target values, one gets a quantification of the error E by determining the sum of the squares of their differences $E = \sum_{i=1}^{N} (\widehat{y}_i - y_i)^2$, and a common technique is to minimize this expression with respect to the parameter

of interest. For example, if our model is $\widehat{y}_i = ax_i$, then a is the parameter we need to find. The minimum of E with respect to parameter a can be found by differentiating the expression for the sum of squares with respect to this parameter and setting this derivative equal to zero, i.e., $\partial E/\partial a = 0$. In Section 19.3, the Kalman filter is developed using the MMSE technique. For other examples of this minimization approach see Chapters 5 and 8.

19.2.3 Recursive Algorithm

Assume we measure the resting potential of a neuron based on a series of measurements: z_1, z_2, …, z_n. It seems reasonable to use the estimated mean $\widehat{m}_n$ of the n measurements as the estimate of the resting potential:

$$\widehat{m}_n = \frac{z_1 + z_2 + \cdots + z_n}{n}. \tag{19.2}$$

For several reasons, this equation isn't an optimal basis for the development of an algorithm. For each new measurement we need more memory: i.e., we need n memory locations for measurements $z_1 - z_n$. In addition, we need to complete all measurements before we have the estimate.

With a recursive approach (as is used in the Kalman filter algorithm) one estimates the resting potential after each measurement using updates for each measurement

$$\text{1st measurement: } \widehat{m}_1 = z_1 \tag{19.3}$$

$$\text{2nd measurement: } \widehat{m}_2 = \frac{1}{2}\widehat{m}_1 + \frac{1}{2}z_2 \tag{19.4}$$

$$\text{3rd measurement: } \widehat{m}_3 = \frac{2}{3}\widehat{m}_2 + \frac{1}{3}z_3 \tag{19.5}$$

and so on.

Now we update the estimate at each measurement and we only need memory for the previous estimate and the new measurement to make a new estimate.

19.2.4 Data Assimilation

Suppose we have two measurements, y_1 and y_2, and we would like to combine/fuse these observations into a single estimate y. Further, the uncertainty of the two measurements is given by their variance (= standard deviation squared), s_1 and s_2, respectively. If we follow the

fusion principle used in the Kalman approach, which is based on MMSE, we will get

$$y = \frac{s_2}{s_1 + s_2} y_1 + \frac{s_1}{s_1 + s_2} y_2 \tag{19.6}$$

for the new estimate, and its variance s can be obtained from

$$\frac{1}{s} = \frac{1}{s_1} + \frac{1}{s_2} \tag{19.7}$$

By following this procedure, we can obtain an improved estimate based on two separate ones (Fig. 19.1).

In the example in Fig. 19.1, we have two observations (black and blue) associated with two distributions with mean values of 5 and 10 and standard deviations of 1 and 3. Using Eqs. (19.6) and (19.7) we find that the best estimate is reflected by a distribution with a mean of 5.5 and a standard deviation of $\sqrt{0.9} \approx 0.95$ (red). From this example it is obvious that the measurement with less uncertainty (black curve) has more weight in the estimate, while the one with more uncertainty (blue curve) only contributes a little. In the context of the Kalman filter, the terms assimilation and blending are sometimes used instead of data fusion in order to describe the combination of estimate and measurement.

19.2.5 State Model

One way of characterizing the dynamics of a system is to describe how it evolves through a number of states. If we characterize state as a vector of

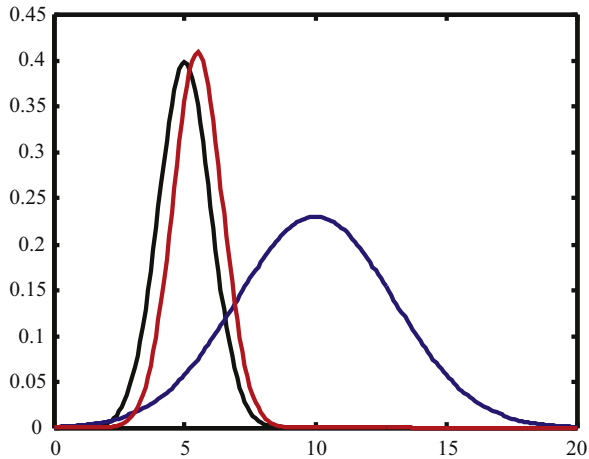


FIGURE 19.1 An example of the fusion procedure.

variables in a state space, we can describe the system's dynamics by its path through state space. A simple example is a moving object. If we take an example where there is no acceleration, its state can be described by the coordinates of its position $\vec{x}$ in three dimensions: i.e., the 3-D space is its state space. The dynamics can be seen by plotting position versus time, which also gives its velocity $d\vec{x}/dt$ (i.e., the time derivative of its position).

Let's take another example, a second-order ordinary differential equation (ODE)

$$\ddot{x}_1 + b\dot{x}_1 + cx_1 = 0 \tag{19.8}$$

Now we define

$$\dot{x}_1 = x_2, \text{ and rewrite the ODE as} \tag{19.9a}$$

$$\dot{x}_2 + bx_2 + cx_1 = 0 \tag{19.9b}$$

In this way we rewrote the single second-order ODE in Eq. (19.8) as two first-order ODEs: Eqs. (19.9a) and (19.9b) (see also Section 9.5). Now we can define a system state vector

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{19.10}$$

Note that, unlike the previous example, this vector is not the position of a physical object but a vector that defines the state of the system. The equations in (19.9a) and (19.9b) can now be compactly written as

$$\frac{d\vec{x}}{dt} = A\vec{x}, \text{ with } A = \begin{pmatrix} 0 & 1 \\ -c & -b \end{pmatrix} \tag{19.11}$$

Here, the dynamics is expressed as an operation of matrix A on state vector $\vec{x}$. This notation can be used as an alternative to any ODE.

19.2.6 Bayesian Analysis

In the Kalman filter approach we update the probability distribution function (i.e., the mean and variance) of the state x of a system based on measurement z, i.e., $p(x|z)$. Conditional probabilities and distributions are the topic of Bayesian statistics and therefore the Kalman approach is a form of Bayesian analysis.

19.3 DERIVATION OF A KALMAN FILTER FOR A SIMPLE CASE

In this section, we derive a simple version of a Kalman filter application following the procedure in Brown and Hwang (1997). They show a derivation for a vector observation, here we show the equivalent simplification for a scalar. All equations with a border are used in the example algorithm of the MATLAB® script pr19_1, and are also shown in the flowchart in Fig. 19.2.

We start with a simple process x, for example, a neuron's resting potential: i.e., its membrane potential in the absence of overt stimulation. We assume that the update rule for the state of the process is

$$x_{k+1} = x_k + w \tag{19.12}$$

The process measurement is z and is defined (as simple as possible) as

$$z_k = x_k + v \tag{19.13}$$

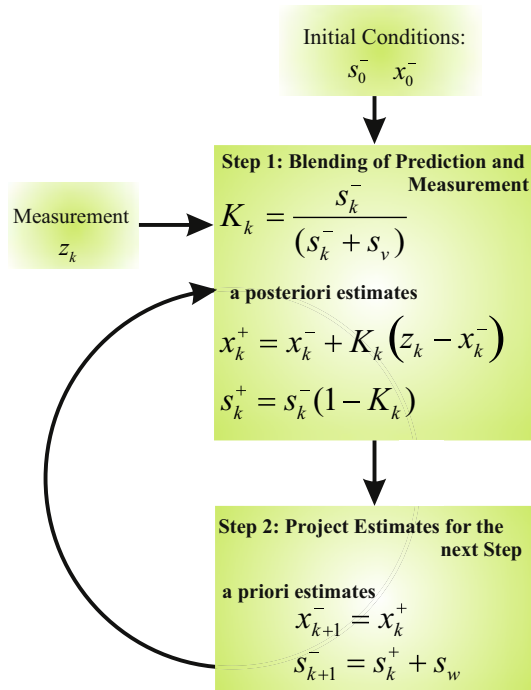


FIGURE 19.2 Flowchart of the Kalman filter procedure derived in this section and implemented in the MATLAB® script pr19_1.m. In this MATLAB® script, steps 1−2 are indicated. Step 1: Eqs. (19.19), (19.15), and (19.20); Step 2: Eqs. (19.21) and (19.22). The equations in the diagram are derived in the text.

In the update procedure, w and v represent Gaussian white noise variables that model the process and measurement noise, respectively. For the sake of this example, they are assumed stationary and ergodic (Chapter 3). The mean for both is zero and the associated variance values are s_w and s_v. The covariance of w and v is zero because they are independent.

Now assume we have an a priori estimate x_k^-. Note that in most texts authors include a "hat": i.e., $\hat{x}_k^-$ to indicate one is dealing with an estimate. To simplify notation, hats to denote estimates are omitted here. The a priori error we make with this estimate is $x_k - x_k^-$ and the variance s_k^- associated with this error is given by

$$s_k^- = E\left[\left(x_k - x_k^-\right)^2\right] \tag{19.14}$$

Now we perform a measurement z_k and assimilate this measurement with our a priori estimate to obtain an a posteriori estimate.

Note: If we integrate a noise term w (drawn from a zero mean normal distribution with variance σ^2) over time, i.e., $\int_{t_1}^{t_2} w\,dt$, we get $\sigma^2(t_2 - t_1)$. So, in the case of a large interval between measurements at t_1 and t_2, the uncertainty grows with a factor $(t_2 - t_1)$ and, consequently, the uncertainty s_{k+1}^- will become very large, and the prediction has almost no value in the fusion of prediction and measurement.

The a posteriori estimate x_k^+, determined by the fusion of the a priori prediction and the new measurement, is given by:

$$\boxed{x_k^+ = x_k^- + K_k\left(z_k - x_k^-\right)} \tag{19.15}$$

The factor K_k in Eq. (19.15) is the **blending factor** that determines to what extent the measurement and a priori estimate affect the new a posteriori estimate.

The a posteriori error we make with this estimate is $x_k - x_k^+$ and the variance s_k^+ associated with this error is given by

$$s_k^+ = E\left[\left(x_k - x_k^+\right)^2\right] \tag{19.16}$$

Substitution of Eq. (19.13) into Eq. (19.15) gives $x_k^+ = x_k^- + K_k\left(x_k + v - x_k^-\right)$. If we plug this result into Eq. (19.16), we get:

$$s_k^+ = E\left[\left(\left(x_k - x_k^-\right) - K_k\left(x_k + v - x_k^-\right)\right)^2\right] \tag{19.17}$$

We can expand this expression and take the expectations. Recall that $x_k - x_k^-$ is the a priori estimation error and uncorrelated with measurement noise v. The outcome of this is (in case this isn't obvious, details of these steps are summarized in Appendix 19.1):

$$s_k^+ = s_k^- - 2K_k s_k^- + K_k^2 s_k^- + K_k^2 s_v$$

This is usually written as:

$$s_k^+ = s_k^-(1 - K_k)^2 + K_k^2 s_v, \tag{19.18}$$

or

$$s_k^+ = s_k^- - 2K_k s_k^- + (s_k^- + s_v)K_k^2$$

To optimize the a posteriori estimate using optimal blending in Eq. (19.15), we must minimize the variance s_k^+ (= square of the estimated error) with respect to K_k, i.e., we differentiate the expression for s_k^+ with respect to K_k and set the result equal to zero:

$$-2s_k^- + 2(s_k^- + s_v)K_k = 0$$

This generates the expression for an optimal K_k,

$$\boxed{K_k = \frac{s_k^-}{(s_k^- + s_v)}.} \tag{19.19}$$

Note: At this point it is interesting to evaluate the result. Substituting the expression for K_k in Eq. (19.15), you can see what this optimized result means: $x_k^+ = x_k^- + \frac{s_k^-}{(s_k^- + s_v)}(z_k - x_k^-)$. If the a priori estimate is unreliable, indicated by its large variance $s_k^- \to \infty$, i.e., $K_k \to 1$, we will ignore the estimate and we believe more of the measurement, i.e., $x_k^+ \to z_k$. On the other hand, if the measurement is completely unreliable $s_v \to \infty$, i.e., $K_k \to 0$, we ignore the measurement and believe the a priori estimate: $x_k^+ \to x_k^-$.

Now we proceed and use the result in Eq. (19.19) to obtain an expression for s_v:

$$s_v = \frac{s_k^- - K_k s_k^-}{K_k}$$

Substitution of this result in Eq. (19.18) gives an expression to find the a posteriori error based on an optimized blending factor K_k:

$$s_k^+ = s_k^-(1-K_k)^2 + \frac{K_k^2 s_k^- - K_k^3 s_k^-}{K_k}$$

With a little bit of algebra we get

$$s_k^+ = s_k^- - 2K_k s_k^- + K_k^2 s_k^- + K_k s_k^- - K_k^2 s_k^-,$$

which we simplify to:

$$\boxed{s_k^+ = s_k^-(1-K_k)} \tag{19.20}$$

Here it should be noted that depending on how one simplifies Eq. (19.18), one might get different expressions. Some work better than others, depending on the situation. We use Eq. (19.20) here because of its simplicity.

Finally we need to use the information to make a projection towards the next time step occurring at t_{k+1} (Fig. 19.2, Step 2). This procedure depends on the model for the process we are measuring from! Since we employ a simple model in this example, Eq. (19.12), our a priori estimate of x is simply the a posteriori estimate of the previous time step:

$$\boxed{x_{k+1}^- = x_k^+} \tag{19.21}$$

Here we ignore the noise (w) term in Eq. (19.12) since the expectation of w is zero. In many applications this step can become more complicated or even an extremely complex procedure depending on the dynamics between t_k and t_{k+1}.

The a priori estimate of the variance at t_{k+1} is based on the error of the prediction in Eq. (19.21),

$$e_{k+1}^- = x_{k+1} - x_{k+1}^-.$$

If we substitute Eqs. (19.12) and (19.21) into this expression, we get $e_{k+1}^- = x_k + w - x_k^+$.

Since the a posteriori error at t_k is $e_k^+ = x_k - x_k^+$, we can plug this into the above expression and we get

$$e_{k+1}^- = e_k^+ + w$$

Using this expression for the a priori error, we see that the associated a priori variance at t_{k+1} depends on the a posteriori error at t_k and the variance of the noise process w:

$$s_{k+1}^{-} = E\left[\left(e_k^{+} + w\right)^2\right].$$

Because w and the error in the estimate are uncorrelated, we finally have

$$\boxed{s_{k+1}^{-} = s_k^{+} + s_w} \tag{19.22}$$

19.4 MATLAB® EXAMPLE

The example MATLAB® script pr19_1 (listed below) uses the above derivation to demonstrate how this could be applied to a recording of a membrane potential of a nonstimulated neuron. In this example we assume that the resting potential Vm = −70 mV and that the standard deviations of the process noise and the measurement noise are 1 and 2 mV, respectively. The program follows the flowchart in Fig. 19.2 and a typical result is depicted in Fig. 19.3. Due to the random aspect of the procedure, your result will vary slightly from the example shown here.

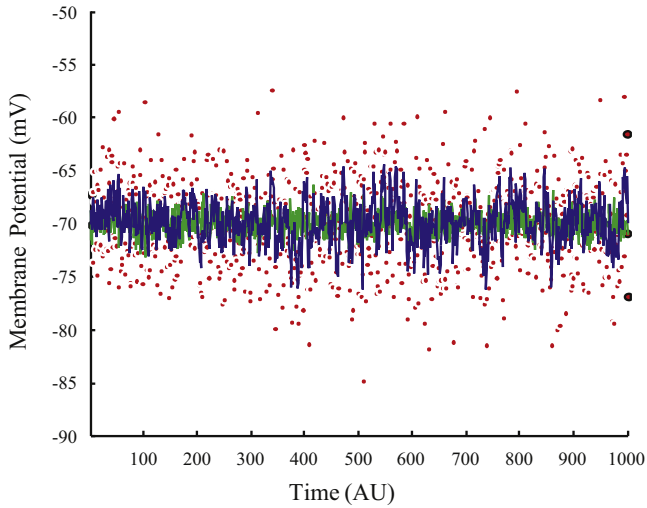


FIGURE 19.3 Example of the Kalman filter application created with the attached MATLAB® script pr19_1. The true values are *green*, the measurements are represented by the *red dots*, and the Kalman filter output is the *blue line*. Although the result is certainly not perfect, it can be seen that the Kalman filter output is much closer to the real values than the measurements.

MATLAB® Script pr19_1.m that performs Kalman filtering

```
% pr19_1
%
% Example using a Kalman filter to
% estimate the resting membrane potential of a neuron
% (Algorithm isn't optimized)

clear
close all

% Parameters
N=1000;      % number of measurements
Vm=-70;      % the membrane potential (Vm) value
sw=1^2;      % variance process noise
sv=2^2;      % measurement noise
% Initial Estimates
x_apriori(1)=Vm;      % first estimate of Vm
                      % This is the a priori measurement
s_apriori(1)=0;       % a priori estimate of the error; here variance =0
                      % i.e., we assume that first estimate is perfect

% create a simulated measurement vector
% -------------------------------------
for i=1:N;
   tru(i)=Vm+randn*sw;
   z(i)=tru(i)+randn*sv;
end;

% Perform the Kalman filter
% -------------------------
% Note: Malab indices start at 1, and not
%       at 0 as in most textbook derivations

% Equations are numbered as in CH 19
for i=1:N;
   % step 1: Blending of Prediction and Measurement
   % Compute blending factor K(i)
   K(i)=s_apriori(i)/(s_apriori(i)+sv);                    % Eq (19.19)
   % Update Estimate using the measurement
   % this is the a posteriori estimate
   x_aposteriori(i)=x_apriori(i)+K(i)*(z(i)-x_apriori(i)); % Eq (19.15)
   % Update the error estimate [Note that there
```

```
    % are several variants for this procedure;
    % here we use the simplest expression
    s_aposteriori(i)=s_apriori(i)*(1-K(i));                % Eq (19.20)
    % step 2: Project Estimates for the next Step
    x_apriori(i+1)=x_aposteriori(i);                       % Eq (19.21)
    s_apriori(i+1)=s_aposteriori(i)+sw;                    % Eq (19.22)
end;

% plot the results
figure;hold;
plot(z,'r.');
plot(tru,'g');
plot(x_aposteriori)
axis([1 N -100 10]);
xlabel ('Time (AU)')
ylabel ('Membrane Potential (mV)')
title ('Kalman Filter Application: true values (green); measurements
(red .); Kalman Filter Output (blue)')
```

19.5 USE OF THE KALMAN FILTER TO ESTIMATE MODEL PARAMETERS

The vector-matrix version of the Kalman filter can estimate a series of variables but can also be used to estimate model parameters. If a parameter of the model is a constant or changes very slowly so that it can be considered a constant, one treats and estimates this parameter as a constant with noise, just as we did with x_k in Eq. (19.12). The noise component allows the algorithm of the Kalman filter to find the best fit of the model parameter at hand.

APPENDIX 19.1 DETAILS OF THE STEPS BETWEEN EQS. (19.17) AND (19.18)

This appendix gives the details of the steps between Eqs. (19.17) and (19.18). For convenience, we repeat the equation.

$$s_k^+ = E\left[\left(\left(x_k - x_k^-\right) - K_k\left(x_k + v - x_k^-\right)\right)^2\right] \tag{19.17}$$

We can expand this expression and take the expectations. If we expand the squared function, we get three terms further evaluated in (1)−(3) below.

1. The first term can be rewritten using Eq. (19.14), i.e.,:

$$E\left[\left(x_k - x_k^-\right)^2\right] = s_k^-$$

2. The second term $E\left[-2\left(x_k - x_k^-\right)K_k\left(x_k + v - x_k^-\right)\right] = E\left[-2K_k\left\{\left(x_k - x_k^-\right)x_k + \left(x_k - x_k^-\right)v - \left(x_k - x_k^-\right)x_k^-\right\}\right]$

The second term between the curly brackets in the above expression, $\left(x_k - x_k^-\right)v$ will vanish when taking the expectation because $x_k - x_k^-$ is the a priori estimation error which is uncorrelated with measurement noise v. Thus, using Eq. (19.14) again, we can simplify

$$E\left[-2K_k\left\{\left(x_k - x_k^-\right)\left(x_k - x_k^-\right)\right\}\right] = -2K_k s_k^-$$

3. The third term is $E\left[K_k^2\left\{\left(x_k + v - x_k^-\right)\left(x_k + v - x_k^-\right)\right\}\right]$. First we focus on the expression between the curly brackets first and we get

$$x_k^2 + x_k v - x_k x_k^- + v x_k + v^2 - v x_k^- - x_k^- x_k - x_k^- v + x_k^{-2} =$$

$$2v\left(x_k - x_k^-\right) + x_k^2 - 2x_k x_k^- + x_k^{-2} + v^2 = 2v\left(x_k - x_k^-\right) + \left(x_k - x_k^-\right)^2 + v^2$$

When we take expectations, the first term of the simplified expression will vanish, so we get the following result:

$$E\left[K_k^2\left(x_k - x_k^-\right)^2 + K_k^2 v^2\right] = K_k^2 s_k^- + K_k^2 s_v$$

If we now collect all terms from (1)−(3), we can rewrite Eq. (19.17) as

$$s_k^+ = s_k^- - 2K_k s_k^- + K_k^2 s_k^- + K_k^2 s_v$$

This is usually written as:

$$s_k^+ = s_k^-\left(1 - K_k\right)^2 + K_k^2 s_v, \tag{19.18}$$

giving us the result in Eq. (19.18).

EXERCISES

19.1 Give the expression for estimate $\widehat{m}_n$ at the nth measurement, using a recursive approach as in Eqs. (19.3)–(19.5)

19.2 Why does (in the Kalman data fusion) adding a measurement always results in a decrease of the uncertainty (variance) of the estimate?

19.3 Write the following third-order ODE in the form of a state vector and matrix operator

$$\dddot{x}_1 + b\ddot{x}_1 + c\dot{x}_1 + dx_1 = 0.$$

Now repeat this for $\dddot{x}_1 + b\ddot{x}_1 + c\dot{x}_1 + dx_1 = e$.
Give the details of the procedure, i.e., define the state and matrix A (see Eqs. (19.8)–(19.11)).

19.4 We have a series of measurements y_n and system states x_n. We want to link both using a simple linear model: $y_n = a \cdot x_n$. Use the MMSE method to find an optimal estimate for a. Comment on how this result relates to linear regression.

19.5 Show that Eqs. (19.6) and (19.7) are the same as the Kalman approach to fusion of data.
Hint 1: Combine Eqs. (19.15) and (19.19); add $\frac{s_1}{s_1+s_2}y_1 - \frac{s_1}{s_1+s_2}y_1$ to the result and rearrange the terms as in Eq. (19.6).
Hint 2: Combine Eqs. (19.20) and (19.19); rearrange Eq. (19.7) as $s = \frac{s_1 s_2}{s_1+s_2}$ and compare the results.
Discuss the difference of combining variance (noise) between Eq. (19.7) and the combined effect of noise sources as in Eq. (3.15).

References

Brown, R.G., Hwang, P.Y.C., 1997. Introduction to Random Signals and Applied Kalman Filtering. John Wiley & Sons, New York, NY.

Maybeck, P.S., 1982. Stochastic Models, Estimation, and Control, vol. 141. Academic Press, New York.

Schiff, S.J., 2012. Neural Control Engineering: The Emerging Intersection Between Control Theory and Neuroscience. MIT Press, Cambridge, MA.

Welch, G., Bishop, G. An Introduction to the Kalman Filter. University of North Carolina. http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf.

CHAPTER

20

Spike Train Analysis

20.1 INTRODUCTION

In the world of neural signals, spike trains take a special position. Spike trains play a crucial role in communication between cells in the nervous system; one might argue that knowing all communication in a system is the same, or comes close to being the same, as knowing a system. We must, however, realize that this is an optimistic view, requiring integration of function across scales similar to (but more complex than) reconstructing the application that is running on a computer (e.g., a game or a word processor) from the signals on all its buses.

While time series of action potentials can be measured intracellularly, they are often recorded extracellularly and are then referred to as spike trains. The intracellular recordings show all aspects of the membrane potential fluctuations, whereas the extracellular spike trains mainly reflect the timing of the occurrence of an action potential. Fig. 20.1 shows how an action potential in a nerve fiber acts as a generator for extracellular current. This current can be measured by a biological amplifier resulting in a tri- or biphasic wave (spike) for each action potential. Although the spikes in spike trains have similar shapes, they are often slightly different. These differences in wave morphology are due to differences in relative position and impedance between electrodes and different neurons; spikes originating from different cells differ in amplitude and waveform.

20.1.1 Deterministic Versus Probabilistic Approach

A spike train may be considered as a list of the times t_i where spikes have occurred. If one considers the Hodgkin and Huxley equations (Hodgkin and Huxley, 1952) as the underlying principle for the generation of action potentials, one must come to the conclusion that the generation of a spike train $\{t_i\}$ is a fundamentally deterministic process. In

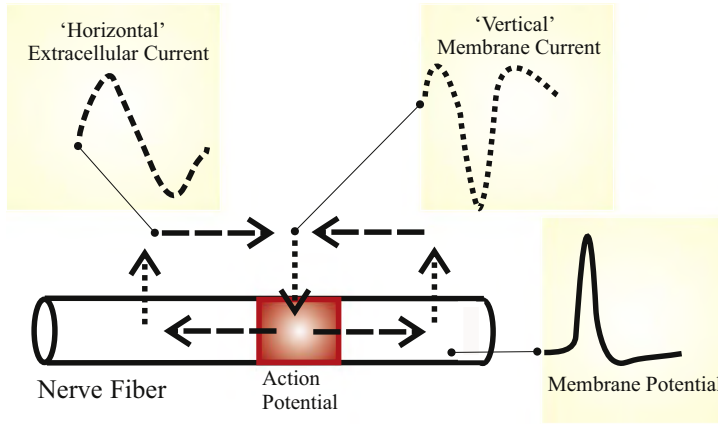


FIGURE 20.1 Schematic representation of intracellular, membrane, and extracellular currents associated with an action potential.

other words, the response to a given stimulus is reproducible and fully determined by the underlying equations. However, all experimental neurophysiologists know that neural responses to the same stimuli s in repeated trials are seldom completely identical. To deal with this variability, Rieke et al. (1999) have successfully applied a probabilistic approach to the analysis of spike train data. They relate the response to a stimulus in a probabilistic fashion with $P(\{t_i\}|s)$ denoting the probability of observing spike train $\{t_i\}$ given that stimulus s occurred. A unique aspect of their approach is that they do not only consider the response but also the stimulus to be drawn from a probability density function. Although this is a somewhat unusual experimental approach where the stimulus is determined by the investigator, with a little bit of imagination it is easy to see that this is analogous to what the brain must do to interpret incoming spike trains and link these to external stimuli. Looking at neural action potential activity from a probabilistic perspective allows the use of Bayes' rule to link the probability $P(\{t_i\}|s)$ of observing a response $\{t_i\}$ to a given stimulus s with the probability that stimulus s occurred when response $\{t_i\}$ is observed, $P(s|\{t_i\})$ (Appendix 20.1).

An example of the latter approach where one attempts to determine the stimulus based on a recorded spike train is shown in Fig. 20.2. In this example the stimulus signal occurring before each spike is averaged in order to find the underlying signal evoking the spike event. The assumption here is that the external stimulus evoking the spike is masked by a random (noise) component that can be reduced by averaging (Chapter 4). The more conventional approach where one determines spike activity evoked by a given stimulus is shown in Fig. 20.3.

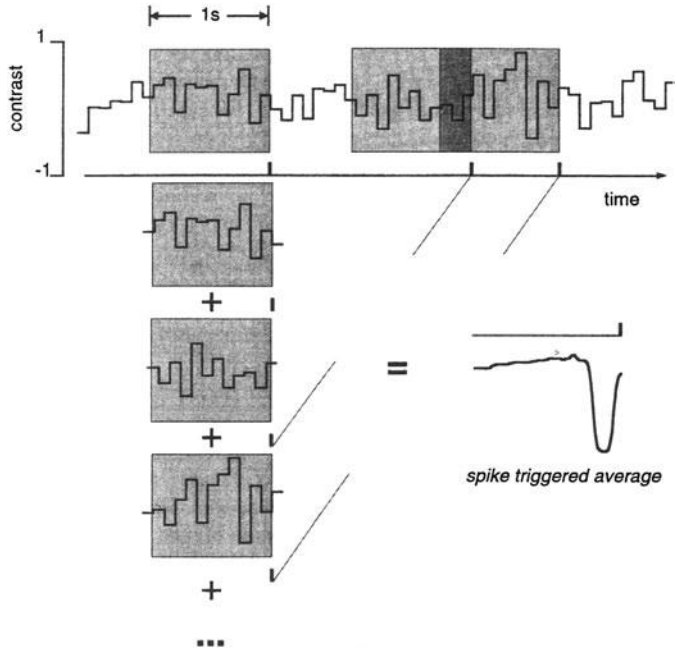


FIGURE 20.2 Interpretation of a spike train by averaging a prespike window of the stimulus. Top trace—stimulus; second trace shows a spike train with three spikes. The *gray boxes* represent the prespike windows that can be averaged to estimate a spike triggered average. *From Rieke, F., Warland, D., de Ruyter van Steveninck, R., Bialek, W., 1999. Spikes Exploring the Neural Code. MIT Press, Cambridge, MA.*

20.1.2 The δ Function

In terms of signals, one may think of a spike in a spike train as an all-or-nothing process. In a general sense, a train of action potentials is a series of events occurring in time; at any given time an event is either absent (off) or present (on). In many papers in which analysis of spike trains plays a role, the activity is therefore presented in so-called raster plots (e.g., the top panel in Fig. 20.3): a time axis with each spike represented by a dot (or a short vertical line) on this axis. Implicitly one has now reduced the action potential to an event on a timeline, with an event-duration of zero (the dot). Interestingly, this approach can also be used to derive a formal representation of a spike train. Considering an epoch on the timeline with an interval of size 1 and located between $-½$ and ½, we can define a spike count function f_s for this epoch such that:

$$\begin{aligned} f_s[\tau] &= 1 \text{ if } \quad -\frac{1}{2} \leq \tau \leq \frac{1}{2} \\ f_s[\tau] &= 0 \quad \text{otherwise} \end{aligned} \tag{20.1}$$

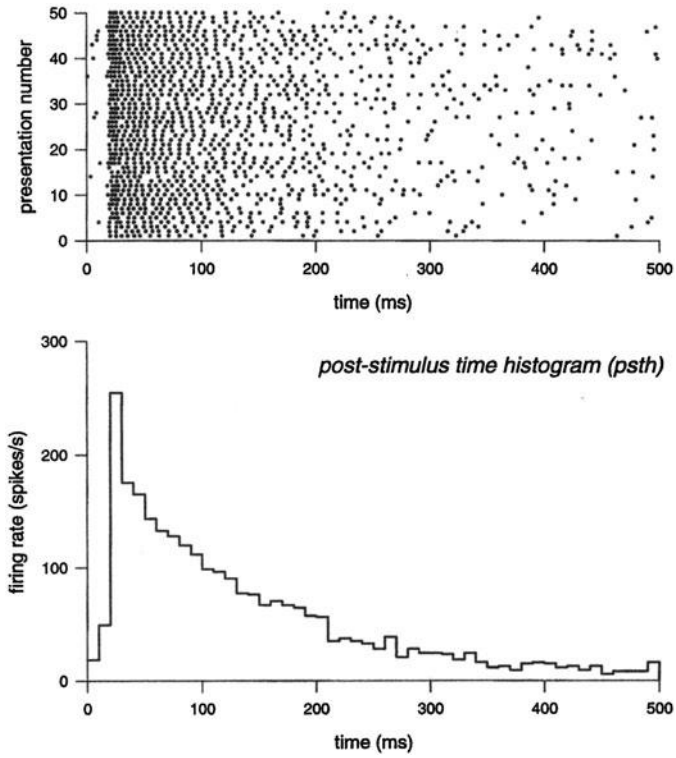


FIGURE 20.3 Top: Raster plots showing spikes in subsequent responses to the same stimulus. Each row is a single response plotted against time. Bottom: The rasterplot data are used to plot the average spike count $<N>$ in 10-ms bins. This is the so-called post-stimulus time histogram. *From Rieke, F., Warland, D., de Ruyter van Steveninck, R., Bialek, W., 1999. Spikes Exploring the Neural Code. MIT Press, Cambridge, MA.*

We can use this function to evaluate an epoch Δ around a particular time t_i by evaluating $f_s[(t-t_i)/\Delta]$. If there is a spike at t_i, the function generates a 1, so if we evaluate the sum of this function for a spike train including all times where a spike is found, we increment by 1 for each spike and obtain a spike count N:

$$N = \sum_i f_s[(t - t_i)/\Delta] \tag{14.2}$$

In real spike trains the neural response is typically variable and usually characterized by the average of a series of responses to an identical stimulus.

In the scenario shown in Fig. 20.3, the spike rate in each bin can be calculated as $<N>/\Delta$, where $<N>$ is the average count over the trials, just

as in the bottom panel in Fig. 20.3. The instantaneous rate $r(t)$ can be found by letting $\Delta \to 0$:

$$r(t) = \lim_{\Delta \to 0} \frac{\langle N \rangle}{\Delta} = \left\langle \lim_{\Delta \to 0} \frac{1}{\Delta} \sum_i f_s[(t - t_i)/\Delta] \right\rangle = \left\langle \sum_i \delta(t - t_i) \right\rangle \quad (20.3)$$

In Eq. (20.3), $\langle \cdot \rangle$ indicates the (vertical) average over each bin in the subsequent trials. The limiting case as $\Delta \to 0$ allows one to express the instantaneous rate as the average of the Dirac delta functions that sift out the spike timing in the responses (Chapter 2).

Notes:

1. The division of f_s, which according to Eq. (20.1) has an amplitude of 1, by a factor Δ (dimension of time) is explicitly included in the definition in Eq. (20.3). Therefore the expression for rate $r(t)$ formally has the dimension s^{-1}. In the introduction of the Dirac in Chapter 2 (see Fig. 2.4), δ didn't have the s^{-1} dimension because we included a dimensionless $1/\tau$ amplitude factor in the definition.
2. Because spikes actually have a finite duration, the derivation of the Dirac by letting $\Delta \to 0$ is not strictly appropriate for this application; in practice one tries to determine Δ so that the bin contains either one or no spikes. In spite of this difficulty, the δ function is often used to formally represent a spike in a spike train because it allows development of mathematical expressions for spike train analysis such as convolution, correlation, etc.

20.2 POISSON PROCESSES AND POISSON DISTRIBUTIONS

An important statistical model for understanding spike trains is the so-called Poisson process, which was explored most fruitfully in the context of a branch of statistics called renewal theory (for a great introduction to this field see Cox, 1962). The major challenge in renewal theory is to understand component failure and its associated statistics. In this sense there is a similarity between the component failure events and the occurrences of spikes in a spike train. The simplest model of the occurrence of an event is to assume a ***constant probability ρ of a component failing, given that is hasn't failed yet***. Imagine you are managing the light bulbs in a building where all lights are always on (broken bulbs are

replaced immediately), we may consider ρ to be the probability that a functional bulb will fail. In this analogy, the process leading to a spike event of a single unit can be compared to observing a new bulb failing in a single light fixture. Similarly, the process of resetting the membrane potential after the spike is analogous to replacement of a bulb that has failed. Because a bulb cannot fail twice (a broken bulb stays broken), the conditional probability (also called the age-specific failure rate) described by probability density function (PDF) $f(t)$ can be interpreted as the product of ρ and the survival function $\mathscr{F}(t) = 1 - F(t)$, where $F(t)$ is the cumulative distribution function (Chapter 3). It can be shown that the PDF for the time of occurrence of a failure given by $f(t) = \rho e^{-\rho t}$ (with $f(t) = 0$ for $t < 0$) satisfies the above condition:

$$\text{Survival } \mathscr{F}(t) = 1 - F(t) = \int_t^{\infty} f(x)dx = \left[-e^{-\rho t}\right]_t^{\infty} = 0 - \left[-e^{-\rho t}\right] = e^{-\rho t}$$

$$\rightarrow \qquad d\mathscr{F}(t)/dt = \frac{de^{-\rho t}}{dt} = -\rho e^{-\rho t}, \tag{20.4}$$

$$\text{also } d\mathscr{F}(t)/dt = d[1 - F(t)]/dt = -dF(t)/dt = -f(t) \tag{20.5}$$

Combining Eqs. (20.4) and (20.5) we get $f(t) = \rho e^{-\rho t}$, which is consistent with our initial assumption in that this PDF embodies a constant failure (event) probability for a component that hasn't previously failed. This process satisfying $f(t) = \rho e^{-\rho t}$ is the so-called Poisson process, and because this process doesn't have a specific aging component (i.e., given the absence of previous failure, there is a constant probability that a failure will occur), it can be classified as memoryless (Fig. 20.4). One

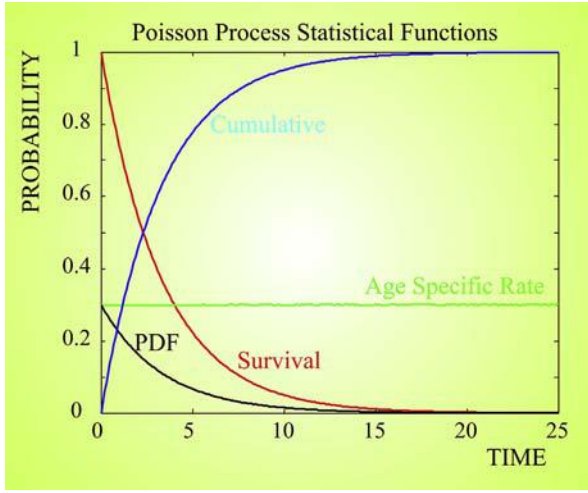


FIGURE 20.4 Overview of statistical functions associated with the Poisson process.

important statistical feature of the PDF of the ***Poisson process is that it is characterized by an equal mean and standard deviation*** (Appendix 20.2). Accordingly, in spike trains this property can be evaluated by calculating mean and standard deviation of the interspike intervals.

Using the approach in Eqs. (20.4) and (20.5) is equivalent to considering the spike train in continuous time. In Section 20.1.2 we considered the spike train in discrete time, i.e., as a set of events in a binned trace. Some of the bins will contain spikes, others will be empty. In the following we consider an epoch of stationary spike activity in n bins of duration Δt. Assuming that the bins are about the duration of a single action potential, we might get a sequence that looks as follows:

0	0	1	0	1	1	0	0	0	1	0	0

In this epoch we count four spikes and eight empty bins. From a distribution point of view this can be described by the binomial distribution, where the probability of a hit (event) occurring is p and the probability of a failure is $q = 1-p$. Here we have $p = \rho\Delta t + o(\Delta t)$, with $\rho\Delta t$ representing the probability that one event occurs in Δt and $o(\Delta t)$ for the probability that more than one event occurs in Δt. By selecting a suitably small value for Δt, we can ignore $o(\Delta t)$ because there won't be more than one spike per interval. This is a bit tricky because Δt cannot approach 0 either, because of the finite duration of a spike. If the occurrence of hits and failures is truly independent we could begin by saying that the probability of counting four spikes in the above sequence of 12 bins is p^4q^8. This probability then needs to be corrected for all the other arrangements that could also lead to a total count of four spikes, e.g.,:

1	1	1	1	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0	0
.....etc											

In this example for $n = 12$ trials we have $i = 4$ hits and $n-i = 8$ failures. The four hits can be arranged in $\frac{n!}{(n-i)!i!} = \frac{12!}{8!4!} = 12 \times 11 \times 10 \times 9/4 \times 3 \times 2 \times 1 = 495$ different ways over the 12 bins (an alternative more compact notation for $\frac{n!}{(n-i)!i!}$ is $\binom{n}{i}$; note that this is not a fraction). This reasoning underlies the derivation of the binomial distribution which links the

probability p of a single hit occurring in one trial to the probability $P(i)$ of encountering i hits in n trials as:

$$P(i) = \frac{n!}{(n-i)!i!} p^i (1-p)^{n-i} \tag{20.6}$$

In real spike trains the intervals can be considered very small (a few ms) compared to the complete time series length (usually 1 to several seconds), in which case with typical spike rates most intervals do not contain spikes. We can therefore consider the above probability function for very large n and small values of p. Suppose that the average number of hits in n observations is λ, then we could define p as λ/n. Using this to rewrite Eq. (20.6):

$$\begin{aligned}
P(i) &= \frac{n!}{(n-i)!i!}\left(\frac{\lambda}{n}\right)^i\left(1-\frac{\lambda}{n}\right)^{n-i} \\
&= \frac{n(n-1)(n-2)\ldots(n-i+1)}{i!}\left(\frac{\lambda}{n}\right)^i\left(1-\frac{\lambda}{n}\right)^{n-i} \\
&= \frac{n(n-1)(n-2)\ldots(n-i+1)}{\left(1-\frac{\lambda}{n}\right)^i i!}\left(\frac{\lambda}{n}\right)^i\left(1-\frac{\lambda}{n}\right)^{n} \\
&= \frac{n(n-1)(n-2)\ldots(n-i+1)}{n^i\left(1-\frac{\lambda}{n}\right)^i i!}\lambda^i\left(1-\frac{\lambda}{n}\right)^{n} \\
&= \frac{1(1-1/n)(1-2/n)\ldots(1-(i-1)/n)}{\left(1-\frac{\lambda}{n}\right)^i i!}\lambda^i\left(1-\frac{\lambda}{n}\right)^{n}
\end{aligned} \tag{20.7}$$

For large n, all terms in the first part that contain a division by n will approach 0, i.e.,:

$$\frac{1(1-1/n)(1-2/n)\ldots(1-(i-1)/n)}{\left(1-\frac{\lambda}{n}\right)^i i!}\lambda^i \rightarrow \frac{\lambda^i}{i!},$$

and the second term is a power series: $\left(1-\frac{\lambda}{n}\right)^n \rightarrow e^{-\lambda}$.

Combining the above results, we obtain the equation for the Poisson distribution:

$$P(i) = \frac{\lambda^i}{i!}e^{-\lambda} \quad \text{or} \quad P(i) = \frac{(\rho t)^i}{i!}e^{-\rho t} \tag{20.8}$$

Note that we used $\lambda = n\Delta t\rho = t\rho$ in the second version of Eq. (20.8). Eq. (20.8) represents a PDF in which the sum of probabilities of all possible outcomes $\sum_{i=1}^{\infty} \frac{\lambda^i}{i!} e^{-\lambda}$ equals 1. This can be seen when substituting the series $\sum_{i=1}^{\infty} \frac{\lambda^i}{i!}$ by exponential e^{λ}, i.e.,:

$$e^{\lambda} = 1 + \lambda + \frac{\lambda^2}{2!} + \frac{\lambda^3}{3!} + \ldots$$

Further we can show that the ***mean and variance of the Poisson distribution are both equal to*** $\boldsymbol{\lambda}$. The mean value can be obtained from $E\{i\} = \sum_{i=0}^{\infty} i \frac{\lambda^i}{i!} e^{-\lambda}$. Using $i\lambda^i = \lambda \frac{\partial}{\partial \lambda} \lambda^i$, taking the exponential and other non-i-related factors out of the summation, using the above power series for e^{λ}, and taking into account that $\partial e^{\lambda}/\partial \lambda = e^{\lambda}$ we get:

$$\begin{aligned} E\{i\} &= \sum_{i=0}^{\infty} i \frac{\lambda^i}{i!} e^{-\lambda} = e^{-\lambda} \sum_{i=0}^{\infty} \frac{1}{i!} \lambda \frac{\partial}{\partial \lambda} \lambda^i = e^{-\lambda} \lambda \frac{\partial}{\partial \lambda} \sum_{i=0}^{\infty} \frac{1}{i!} \lambda^i = e^{-\lambda} \lambda \frac{\partial}{\partial \lambda} e^{\lambda} \\ &= e^{-\lambda} \lambda e^{\lambda} = \lambda \end{aligned}$$

In a similar way by using the second derivative, one can show that the variance is equal to the mean $E\{i^2\} - E\{i\}^2 = \lambda$. For a Poisson distribution, the so-called ***Fano factor***, which is the ratio between variance/mean, is 1. The Fano factor is indeed $\sim$1 for short spike trains (order of second(s)), but for larger epochs the Fano factor usually becomes >1.

In Van Drongelen et al. (1978), the Poisson process was used to study sensitivity effects of the convergence of receptor cells in the olfactory system. In this system, $\sim$1000 peripheral sensory neurons project onto a single mitral cell in the olfactory bulb. At threshold levels of stimulation, the sensory neurons show probabilistic firing patterns, so by convention a particular unit's threshold is arbitrarily defined as the stimulus level that evokes a response in 50% of the presentations of that stimulus. Because the mitral cells receive input from 1000 neurons at the same time, we might predict that their threshold levels occur at significantly lower concentrations of odorant as compared to the peripheral threshold (i.e., the signal is amplified by the convergence of sensory neurons).

We can estimate the amplification effect if we analyze the above system in a simplified model. Assume absence of spontaneous activity and a sensory cell firing probability upon stimulation ρ. Under these assumptions, we can determine the probability that a stimulated sensory neuron fires at least once in observation interval T as:

$$1 - (\text{probability that the sensory neuron does NOT fire}) = 1 - e^{-\rho T}$$

Consider a single mitral cell observing 1000 of these sensory cells. Further assume that the mitral cell is rather sensitive so that it spikes upon a spike in any of its connected sensory cell population; therefore the probability that the mitral cell fires in the same interval T is:

$$1 - (\text{probability that NONE of the } 1000 \text{ sensory neurons fire})$$

If we consider the activity of the sensory neurons independent, which is not a bad assumption at low levels of odorant diffusing in the olfactory mucosa, we can express the probability that none of the 1000 sensors fires as the product of the individual probabilities $(e^{-\rho T})^{1000}$; therefore we can predict that the probability that the mitral cell fires is $1 - e^{-1000\rho T}$, much higher than the probability that a single sensory cell fires $1 - e^{-\rho T}$. Indeed, this prediction was experimentally confirmed (Duchamp-Viret et al., 1989).

20.3 ENTROPY AND INFORMATION

For most of us "information" is a familiar concept because sending and receiving messages, i.e., information exchange, is an important part of our daily life. However, for a study of information transfer in the nervous system we need a formal definition and preferably a metric quantifying information content. Shannon's communication system provides such a framework for analyzing the transmission of a message from a source to a destination (Shannon and Weaver, 1949). Considering a set of messages to be transmitted, Shannon's idea was to use the so-called entropy of this set to quantify its potential information content. It is important to realize that this entropy measure differs from what may be considered the everyday sense of information content. The entropy concept does not capture the information in a single message, but merely quantifies the potential information of the ensemble of messages. At first sight this may seem somewhat strange, but consider the example where we continuously transmit the same message. In this case, one might conclude that there is no need for information to be transmitted, or one might even argue that (from a data transmission standpoint) there is no information sent at all. Similarly, if a continuously received message is always the same, there is no reason to receive the message and one might state that in this case there is no information either. Therefore, when using Shannon's entropy concept, the context of the message in its ensemble and variability are critical for quantifying information content. As was mentioned in Section 20.1.1, in the analysis of spike trains it is not uncommon to consider both the stimulus and the neural response to be drawn from a probability density function (i.e., Rieke et al., 1999), thus making it possible that variability is associated with information. This idea will be explored further in the following paragraphs.

Let X be a message consisting of two statistically independent random variables X_1 and X_2. The probability of observing X_1 and X_2 is the product of the individual probabilities, i.e., $p[X_1] \times p[X_2]$. In other words, the uncertainty of X, or the information conveyed by revealing that message X has taken on the value X_1 and X_2 depends on the probability density functions of X_1 and X_2. Let's use the functional S to obtain the entropy associated with the observations X_1 and X_2, such that the information gained from observation X_1 is $S\{p[X_1]\}$ and the information associated with observation X_2 is $S\{p[X_2]\}$. If we now receive the information associated with both X_1 and X_2, the information gained from each should add to represent the combined information, i.e.,

$$S\{p[X]\} = S\{p[X_1, X_2]\} = S\{p[X_1] \times p[X_2]\} = S\{p[X_1]\} + S\{p[X_2]\} \quad (20.9)$$

Eq. (20.9) shows that the product $p[X_1] \times p[X_2]$ is converted into a sum of entropies. Therefore the entropy behaves as a logarithm of the distribution. Although this is not a proof of Shannon's approach, due to the conversion of a product of probabilities to a sum of entropies, the intuitive notion is that the entropy of a system is proportional with the logarithm of the number of its possible states. Let X be a discrete variable, representing a spike train response to a stimulus, taking a finite number of possible values $x_1, x_2, \ldots, x_N$. First, if we assume that each state is equally likely to occur (i.e., $p = 1/N$), the entropy is proportional to $\log(N)$ or $-\log(1/N)$ (note the $-$ sign). If the states are not equally probable, but occur with probabilities $p_1, p_2, \ldots, p_N$ (such that $\sum_{i=1}^{N} p_i = 1$), the entropy is proportional to $-\sum_{i=1}^{N} p_i \log p_i$ (note the $-$ sign). To obtain an entropy measure that can be expressed in bits, it is common practice to use a base 2 logarithm:

$$S = -\sum_{i=1}^{N} p_i \log_2 p_i \text{ bits} \quad (20.10)$$

The above discussion of the entropy-information quantification approach makes it clear that this information content can only be established if the PDF associated with a message is known.

In spite of the quantitative nature of Shannon's approach, it leaves quite a bit open to interpretation when applied to spike trains, a reflection of our ignorance of neural coding. For instance, if we record a particular response with N_1 spikes, do we consider the number N_1 as a number drawn from a PDF or not? If so, what PDF do we assume? Do we assume that the timing of each spike is important, or is it just the number N_1 that captures the essence of the message? It is difficult to answer these

questions without knowing how the nervous system processes this particular message. It is, however, critical to think about these issues since the answers to the above questions directly determine the assumed PDF from which our observation is drawn and therefore determine the entropy. Let's analyze an example where we observe N_1 spikes in a trace with N bins, similar to the examples we discussed in the previous section. In this example we determine a bin size that is small enough so that an individual bin either contains one or no spike. There are several possible approaches, e.g.,:

1. We have N bins, so we can support $N+1$ possible spike counts (a 0 spike result is also possible) (Fig. 20.5B). Assuming that each response is equally likely, the resulting entropy is $\log_2(N+1)$ bits. This example is mentioned to be complete, but is somewhat silly because its result depends on the number of bins (which we set arbitrarily). In addition, it is unlikely that all responses $0{-}N$ spikes are equally likely. On the other hand, there may be scenarios where, for a certain choice of interval, the number of spikes is the important message.
2. We have N_1 spikes, assuming that timing is important we have $N!/(N_1!N_0!)$ possible arrangements over the N bins, with $N_0 = N - N_1$. If we now assume that each arrangement is equally

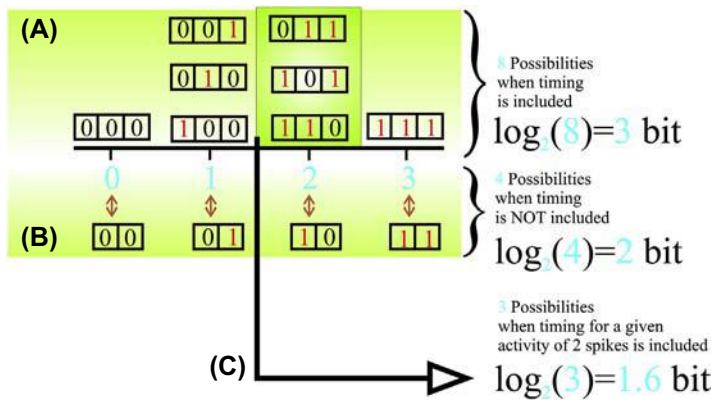


FIGURE 20.5 Simplified example of a spike train of three bins. The bin size is selected such that it can either contain one or no spikes. All eight possibilities having 0, 1, 2, or 3 spikes in the set of three bins are shown in (A). This results in an entropy value of 3 bits. This comes as no surprise since we have three bins that could contain 0 or 1. (B) If we consider the number of spikes, without paying attention to the timing, there are only four possibilities resulting in a 2-bit value for the entropy. Decimal values are shown in blue; binary values are black (0) or red (1). When taking the activity level as a given, for instance two spikes in a three-bin observation span, the timing of those spikes determines the entropy. In example (C) we have three possible arrangements, leading to a value of 1.6 bit.

likely we have an entropy of $\log_2[N!/(N_1!N_0!)]$. This example is a little less silly, but it still does not account for the fact that the observation of N_1 spikes varies, i.e., we consider the observation of N_1 spikes as deterministic. Within the given number of spikes, we allow different distributions over the available bins (i.e., we do consider the effect of timing). For instance, if we observe two spikes in a trace with three bins ($N_1 = 2$), we have $3!/(2!1!) = 3$ possibilities. Assuming these all equally likely, we have $3 \times \frac{1}{3}\log_2(3) \approx 1.6$ bits of total entropy (Fig. 20.5C).

3. The above naturally leads to the third example where the observed spike train is drawn from a PDF such as a Poisson or a binomial distribution. In this case the number of spikes is not fixed, and unlike in the example above where we observed two spikes, we also allow other observations (e.g., the number of spikes = 0, 1, 2, or 3 in Fig. 20.5A). In this example both the number of spikes and their timing are important.

In all the above examples it is clear that bin size affects the outcome of the entropy measure. Therefore one needs an approach to obtain an objective and a reasonable bin size for the particular problem at hand. A commonly used approach is to find a distribution and/or bin size that maximizes the entropy, considering at a minimum a bin equal to the duration of a spike plus the absolute refractory period. The guiding principle here is that this method determines the upper bound of the information of a measured spike train from a single neuron.

A few numerical examples of how one could calculate the entropy of a short spike train with three bins are shown in Fig. 20.5. We stay with the example of short bins where each may or may not contain a single spike, i.e., each bin can only contain a 1 or a 0. Therefore, if we select the first scenario described above, the three bins can code for a range of numbers from 0 to 3, and $S = \log_2(4) = 2$ bits. If instead we consider the third scenario described above, we assume a binomial distribution (Eq. 20.6) in which the occurrence of a 1 or 0 in each bin is equally likely ($p = .5$). We have the following possibilities (Fig. 20.5):

$$\text{Count} = 0 \;\rightarrow\; \frac{3!}{3!0!}\left(\frac{1}{2}\right)^0\left(\frac{1}{2}\right)^3 = \frac{1}{8} \quad 1 \text{ out of } 8 \text{ cases}$$

$$\text{Count} = 1 \;\rightarrow\; \frac{3!}{2!1!}\left(\frac{1}{2}\right)^1\left(\frac{1}{2}\right)^2 = \frac{3}{8} \quad 3 \text{ out of } 8 \text{ cases}$$

$$\text{Count} = 2 \;\rightarrow\; \frac{3!}{1!2!}\left(\frac{1}{2}\right)^2\left(\frac{1}{2}\right)^1 = \frac{3}{8} \quad 3 \text{ out of } 8 \text{ cases}$$

$$\text{Count} = 3 \;\rightarrow\; \frac{3!}{0!3!}\left(\frac{1}{2}\right)^3\left(\frac{1}{2}\right)^0 = \frac{1}{8} \quad 1 \text{ out of } 8 \text{ cases}$$

Note: In the above we used the definition $0! \equiv 1$.

We can see that in this case there are eight arrangements that are equally likely, using Eq. (20.10) the total entropy in this case is:

$$S = -\sum_{i=1}^{8} p_i \log_2 p_i = -8 \times \frac{1}{8} \log_2 \frac{1}{8} = \log_2 8 = 3 \text{ bits}$$

In other words, by revealing a single, particular sequence in this scenario, we provide total # of bits × the probability of that single scenario: $3 \times 1/8$ bits of information. These numerical examples are fairly artificial, but they illustrate the point that the context in which a message is considered is critical for the associated entropy value.

In studies of spike trains, the probability of spike occurrence is often estimated from the recorded time series. In addition, the bin size is usually selected to generate a maximum entropy value (e.g., Van Drongelen et al., 2003). In this study the spike activity of single cortical neurons was associated with bursting activity of the surrounding cortical network. The network bursts were used to align the spike trains, a similar procedure as one would follow when a stimulus is used as the trigger (e.g., Fig. 20.3). The aligned spike trains were binned to allow estimation of each spike train's entropy.

An illustration of the application of these entropy calculations is shown in Fig. 20.6. There are four spike train trials that are triggered (aligned) by a population burst; each trial is divided into nine bins resulting in a total of $4 \times 9 = 36$ bins. We can use our spike train statistics to estimate the probability associated with each observation. In all of the 36 bins we observe 14 bins with zero spikes (i.e., $p_0 = 14/36$); 16 bins with one spike (i.e., $p_1 = 16/36$); five bins with two spikes (i.e., $p_2 = 5/36$); and one bin with three spikes (i.e., $p_3 = 1/36$). These values can be used to estimate the total entropy H_t:

$$-(14/36) \times \log_2(14/36) - (16/36) \times \log_2(16/36) - (5/36) \times \log_2(5/36) - (1/36) \times \log_2(1/36) \approx 1.6 \text{ bits}$$

Normally one would correct the estimate for bias; here, in order to keep the example as simple as possible we omit the correction. Now if the spiking activity was only due to the population activity, the number of spikes across the vertical bins should be identical. In other words, the variability across the vertical bins represents activity that is not associated

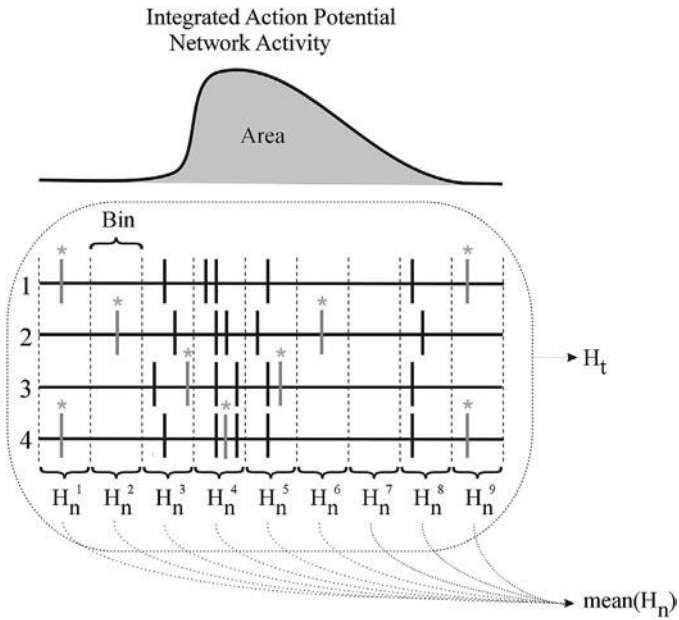


FIGURE 20.6 Integrated network activity-triggered average of a single neuron's spike train. In this theoretical example, the spikes marked with * are not burst-related. This is only for the sake of the example; in a real measurement the origin of the spikes is (of course) not known.

with the burst (the trigger event) and therefore can be considered as noise. Applying this to the example in Fig. 20.6 we get $(1/2) \times \log_2(1/2) + (1/2) \times \log_2(1/2)$ for the first column and $(3/4) \times \log_2(3/4) + (1/4) \times \log_2(1/4)$ for the second column, etc. Finally, the average noise entropy for all nine columns mean(H_n) can be subtracted from the total entropy in all traces (1.6 bits in this example) to estimate the information that is associated with the burst H_{burst}:

$$H_{burst} = H_t - \text{mean}(H_n) \tag{20.11}$$

20.4 THE AUTOCORRELATION FUNCTION

Due to the specific properties of the spike trains, applications of signal processing techniques such as correlation differ somewhat from the conventional approaches.

In Chapter 13, we defined autocorrelation R_{xx} of signal x as $R_{xx}(t_1,t_2) = E\{x(t_1)x(t_2)\}$. In the case of a spike train we don't have a continuous signal

as we did in Chapter 13, but we may use the instantaneous rate function $r(t)$ as a proxy. Consequently we can define the autocorrelation $R_{rr}(t_1, t_2)$ of r as:

$$R_{rr}(t_1, t_2) = E\{r(t_1)r(t_2)\} \tag{20.12}$$

Assuming we may use a time average (denoted by $\langle \cdot \rangle$), we get:

$$R_{rr}(t_1, t_2) = E\{r(t_1)r(t_2)\} = \langle r(t_1)r(t_2) \rangle \tag{20.13}$$

In Section 20.1.2 (Eq. 20.3) the rate function was defined as the spike average over a large number of epochs. This definition is practical because the average can be easily determined directly from experimental observations, such as in the example shown in Fig. 20.3. More theoretically one may relate the rate to the probability of the occurrence of a spike at a given time. This follows directly from the definition of the rate as the average occurrence of spikes across a large set of trials. Consequently the two extremes for this average are: (1) there is always a spike in the observed epochs, or (2) we never observe any action potential in these epochs. In these two extreme cases, the resulting average for the rate (in terms of spiking probability) is 1 or 0, respectively, and in all other cases the average value representing the instantaneous rate will be between 0 and 1. Note that in this example we wanted to estimate spike probability and therefore we did not divide by the bin width Δ. If we had we would have obtained values expressed in spikes/s instead of probabilities between 0 and 1: i.e., an alternative interpretation of the instantaneous rate (not weighed by Δ) is the probability of the occurrence of an action potential. Therefore the autocorrelation of a single spike train $R_{rr}(t_1, t_2)$ is proportional with the probability of observing spikes at t_1 and t_2:

$$p(\text{spike at } t_1 \ \& \ \text{spike at } t_2) = \underbrace{p(\text{spike at } t_1 | \text{spike at } t_2)}_{\text{conditional rate}} \times \underbrace{p(\text{spike at } t_2)}_{\sim r(t_2)} \tag{20.14}$$

The conditional rate, the first term in Eq. (20.14), is usually called the autocorrelation of the spike train. Just as we determined the probability of spiking after a stimulus by using a time average, it is intuitive to estimate the probability of a spike at t_1 in a similar fashion. In this case we use a time average of traces where spike occurrence instead of stimulus occurrence is used to align the traces. An example is shown in Fig. 20.7. Instead of shifting the time series by a small regular interval dt, the correlation function of the spike train is obtained from shifts that bring each subsequent occurrence of a spike to time 0 (top panel in Fig. 20.7); by following this procedure we satisfy $p(\text{spike at } t_2) = 1$ in Eq. (20.14). This process is repeated several times and then the average of the binned traces

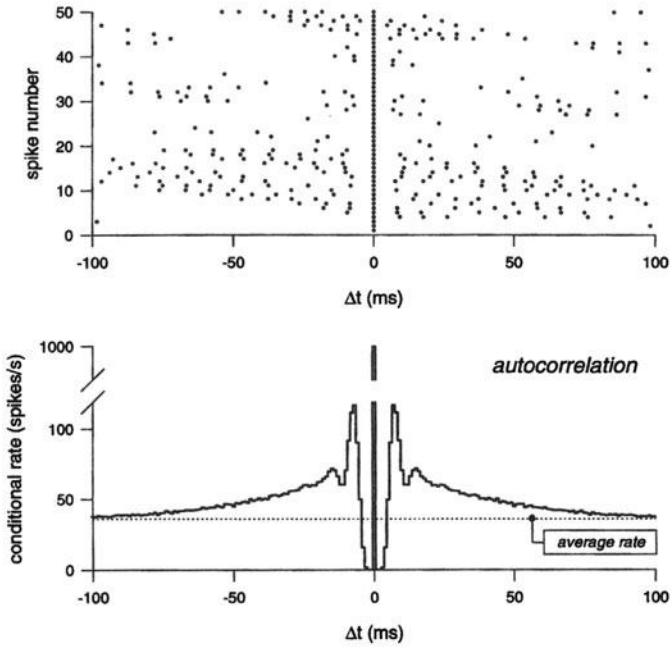


FIGURE 20.7 Autocorrelation of a spike train. Top: raster plot of spike activity, each row is a plot of spike occurrence that is aligned with the middle spike in the top row. Bottom: The average of the spikes in each bin as a function of delay. *From Rieke, F., Warland, D., de Ruyter van Steveninck, R., Bialek, W., 1999. Spikes Exploring the Neural Code. MIT Press, Cambridge, MA.*

is used as an estimate of the autocorrelation function (e.g., autocorrelation in the bottom panel in Fig. 20.7). As shown in this figure it is not uncommon to divide the outcome by the bin width in order to obtain a value in spikes/s. Also note that in the example in Fig. 20.7 each spike in the train is used in the average, i.e., the process is assumed to be stationary so that the autocorrelation only depends on the difference $\tau = t_2 - t_1$.

An alternative procedure for calculating the autocorrelation also follows from the definition of autocorrelation discussed in Chapter 13 and the representation of the spike train as a series of unit impulses. If the spike train represents a stationary process, the underlying distributions are invariant and only the difference $\tau = t_2 - t_1$ is relevant:

$$R_{rr}(\tau) = E\{r(t)r(t+\tau)\} \tag{20.15}$$

Assuming ergodicity we can use a time average:

$$R_{rr}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_0^T r(t)r(t+\tau)dt \tag{20.16}$$

Using the fact that T becomes very large, we can use the definition from Eq. (13.20):

$$R_{rr}(\tau) = \int_{-\infty}^{\infty} r(t)r(t+\tau)dt \tag{20.17}$$

So for each value of τ we correlate the spike train with itself. Since a single instantiation of a spike train can be represented by a series of Diracs: $\sum_{j=1}^{N} \delta(t - t_j)$, for each τ, this series of Diracs may be correlated with itself:

$$R_{rr}(\tau) = \int_{-\infty}^{\infty} \sum_{i=1}^{N} \delta(t - t_i) \sum_{j=1}^{N} \delta(t - t_j + \tau)dt \tag{20.18}$$

Assuming that we may interchange the integration and summation procedures:

$$R_{rr}(\tau) = \sum_{i=1}^{N} \sum_{j=1}^{N} \int_{-\infty}^{\infty} \delta(t - t_i)\delta(t - t_j + \tau)dt \tag{20.19}$$

Eq. (20.19) can be simplified by using the sifting property of the δ function.

In the following we illustrate a simple example of a spike train with three spikes at times $t_1 = 0$, $t_2 = 1$, $t_3 = 4$, and no other spike activity. Since we have two delta functions in the integral in Eq. (20.19), we can sift the first with the second delta function or sift the second with the first.

For the first spike:

$$i = 1 \;\&\; j = 1: \quad \boxed{\int_{-\infty}^{\infty} \delta(t - t_1)\delta(t - t_1 + \tau)dt = \delta(t_1 - t_1 + \tau) = \delta(\tau),}$$

$$\text{or :} \quad \int_{-\infty}^{\infty} \delta(t - t_1)\delta(t - t_1 + \tau)dt = \delta(t_1 - \tau - t_1) = \delta(-\tau).$$

$$i = 1 \;\&\; j = 2: \quad \boxed{\int_{-\infty}^{\infty} \delta(t - t_1)\delta(t - t_2 + \tau)dt = \delta(t_1 - t_2 + \tau) = \delta(\tau - 1);}$$

$$\text{or :} \quad \int_{-\infty}^{\infty} \delta(t - t_1)\delta(t - t_2 + \tau)dt = \delta(t_2 - \tau - t_1) = \delta(-\tau + 1).$$

$$i = 1 \;\&\; j = 3: \quad \boxed{\int_{-\infty}^{\infty} \delta(t - t_1)\delta(t - t_3 + \tau)dt = \delta(t_1 - t_3 + \tau) = \delta(\tau - 4);}$$

$$\text{or :} \qquad \int_{-\infty}^{\infty} \delta(t - t_1)\delta(t - t_3 + \tau)dt = \delta(t_3 - \tau - t_1) = \delta(-\tau + 4).$$

In the above it can be seen that the results for sifting one delta with the other (and vice versa) generates a pair of mirror shifts (i.e., τ and $-\tau$; $\tau - 1$ and $-\tau + 1$; $\tau - 4$ and $-\tau + 4$). Thus because the autocorrelation is an even function (e.g., $R_{rr}(\tau) = R_{rr}(-\tau)$, Appendix 5.2) we only need to consider sifting in one direction.

For the second spike (in the following we omit the equations for the mirror cases):

$$i = 2 \;\&\; j = 1: \quad \boxed{\int_{-\infty}^{\infty} \delta(t - t_2)\delta(t - t_1 + \tau)dt = \delta(t_2 - t_1 + \tau) = \delta(\tau + 1)}$$

$$i = 2 \;\&\; j = 2: \quad \boxed{\int_{-\infty}^{\infty} \delta(t - t_2)\delta(t - t_2 + \tau)dt = \delta(t_2 - t_2 + \tau) = \delta(\tau)}$$

$$i = 2 \;\&\; j = 3: \quad \boxed{\int_{-\infty}^{\infty} \delta(t - t_2)\delta(t - t_3 + \tau)dt = \delta(t_2 - t_3 + \tau) = \delta(\tau - 3)}$$

For the third spike:

$$i = 3 \;\&\; j = 1: \quad \boxed{\int_{-\infty}^{\infty} \delta(t - t_3)\delta(t - t_1 + \tau)dt = \delta(t_3 - t_1 + \tau) = \delta(\tau + 4)}$$

$$i = 3 \;\&\; j = 2: \quad \boxed{\int_{-\infty}^{\infty} \delta(t - t_3)\delta(t - t_2 + \tau)dt = \delta(t_3 - t_2 + \tau) = \delta(\tau + 3)}$$

$$i = 3 \;\&\; j = 3: \quad \boxed{\int_{-\infty}^{\infty} \delta(t - t_3)\delta(t - t_3 + \tau)dt = \delta(t_3 - t_3 + \tau) = \delta(\tau)}$$

Summing the results we obtained for i and j from 1 to 3 (the boxed equations above), Eq. (20.19) evaluates to:

$$R_{rr}(\tau) = \sum_{i=1}^{3}\sum_{j=1}^{3}\int_{-\infty}^{\infty}\delta(t-t_i)\delta(t-t_j+\tau)dt = \sum_{i=1}^{3}\sum_{j=1}^{3}\delta(t_i-t_j+\tau)$$
$$= [\delta(\tau+4)+\delta(\tau+3)+\delta(\tau+1)+3\times\delta(\tau)+\delta(\tau-1)+\delta(\tau-3)+\delta(\tau-4)] \tag{20.20}$$

In Fig. 20.8 we determine the autocorrelation for the train of three spikes; in panel A we obtain the autocorrelation by shifting the spike train for each spike relative to one of the spikes, followed by a summation (Σ in Fig. 20.8A, representing the nonscaled average). In this example we demonstrate this principle for each of the spikes at $t_1 = 0$, $t_2 = 1$, $t_3 = 4$ in black, red, and blue, respectively. This procedure is similar to the one shown in Fig. 20.7. In Fig. 20.8B we use a different approach and

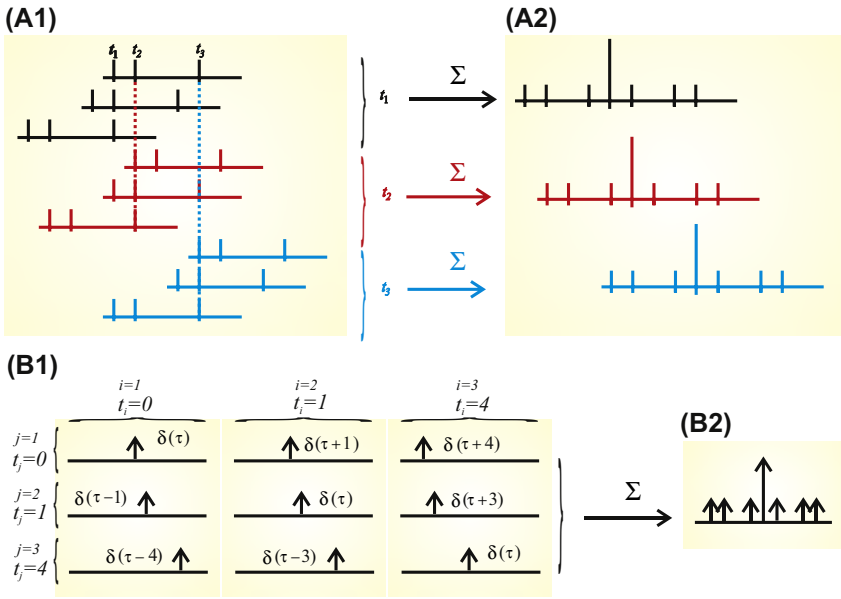


FIGURE 20.8 Example of a series of three spikes (A1) and the autocorrelation function (A2) based on the first spike (black), the second spike (red), and the third one (blue). The Σ symbol indicates the summation of the three traces of the associated color. In this example there are few spikes so that each event can be indicated individually. In real spike trains, the traces are binned and the number of spikes per bin is determined because there can be many closely spaced spikes (see Fig. 20.7). In addition, it is customary to scale the ordinate to represent spikes/s or the correlation coefficient. In panels (B1) and (B2), the correlation function based on Eqs. (20.19) and (20.20) is shown. Comparing the autocorrelation in (A2) and (B2) it can be seen that the different approaches generate the same result.

demonstrate the summation underlying the result in Eq. (20.20). It can be seen that both results in Fig. 20.8 are identical. In none of the examples are we worried about normalizing our result, but it can be seen that at the autocorrelation function at $\tau = 0$ always contains the summation of $N = 3$ spikes; therefore we obtain a scaled correlation equal to 1 at $\tau = 0$ by dividing the summed result by N.

20.5 CROSS-CORRELATION

Not surprisingly cross-correlation between two spike trains follows a similar procedure to that discussed above for autocorrelation. The two spike trains are shifted relative to each other and the spike-triggered average represents an estimate of the cross-correlation. Another interesting application of cross-correlation is between a spike train and a continuous signal such as the stimulus $s(t)$ evoking the spike train $\{t_i\}$. Examples of such a correlation are shown in Fig. 20.9. A cross-correlation

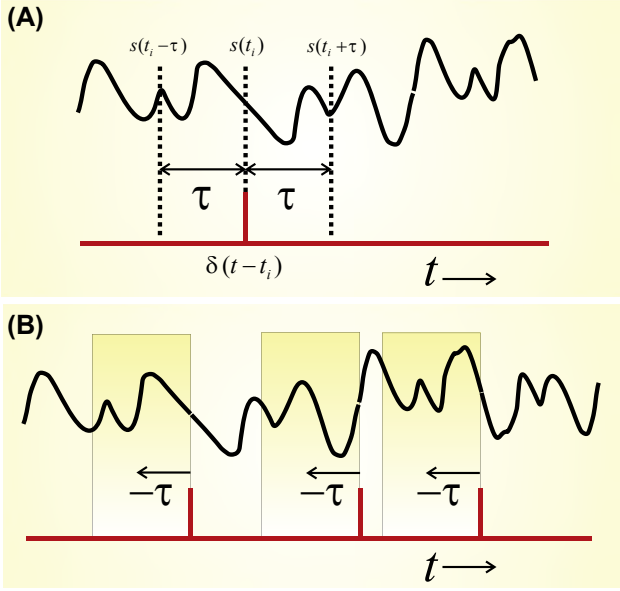


FIGURE 20.9 Cross-correlation between a spike train (red) and a continuous signal (black) representing the stimulus evoking the train. (A) The relationship between a single spike and its correlation. Because of the causal relationship between stimulus and spike, we are only interested in $s(t)$ preceding the spike, the so-called reverse-correlation function. (B) A spike train with three spikes and the associated preceding correlation windows. When the signal in these windows is averaged, we obtain the spike-triggered average (i.e., the estimated reverse correlation), as in the example shown in Fig. 20.2.

between a stimulus signal $s(t)$ and a train with a single spike at time t_i considered over an interval T can be defined as:

$$R(\tau)_{s(t),\{t_i\}} = \int_T s(t) \underbrace{\delta(t - t_i + \tau)}_{\delta(t-(t_i-\tau))} dt = s(t_i - \tau) \tag{20.21}$$

Unlike the autocorrelation, the cross-correlation of two different signals is not an even function. For $\tau < 0$, the correlation of the signal at $s(t_i - \tau)$ suggests that the spike predicts the stimulus, a fairly unrealistic assumption because it violates causality. In the following we consider only positive values of τ in $s(t_i - \tau)$ indicating that we are looking from the spike time t_i backward (reverse-correlation function). If we now consider a spike train with N spikes over interval T, we obtain the reverse-correlation function as:

$$R(\tau)_{s(t),\{t_i\}} = \int_T s(t) \left[\sum_{i=1}^{N} \delta(t - t_i + \tau) \right] dt \tag{20.22}$$

Interchange of the integration and summation gives:

$$R(\tau)_{s(t),\{t_i\}} = \sum_{i=1}^{N} \int_T s(t)\delta(t - t_i + \tau)dt = \sum_{i=1}^{N} s(t_i - \tau) \tag{20.23}$$

In other words, the cross-correlation can be estimated by the sum of the signal epochs preceding each evoked spike, as shown in Fig. 20.9. In the equations we didn't bother with normalizing the expression for the cross-correlation. A common normalization is to divide by N so that Eq. (20.23) becomes $\frac{1}{N} \sum_{i=1}^{N} s(t_i - \tau)$, a spike-triggered signal average of the stimulus preceding the spike event, such as in the example of Fig. 20.2. The use of reverse correlation is directly related to the fact that we are considering the stimulus signal and that the spiking neuron obeys causality. Of course, this reasoning would reverse if the signal $s(t)$ represented a movement and the spike train was from a motoneuron steering this movement.

Another case can occur if the continuous signal may include a stimulus component as well as a component evoked by the spike. This happens when a local field potential around a spike event is observed. Because this local field potential represents all ongoing activity in the surrounding tissue, it may include both pre- and postsynaptic signals that are associated with the spike. An example of this scenario is presented in Eissa et al. (2017).

APPENDIX 20.1

Bayes' Rule

In Rieke et al. (1999), both the spike trains and stimuli are considered as events drawn from a probability density function. The following explaining the application of Bayes' rule in the field of spike train analysis is based on their reasoning.

Assuming that both the spikes and the stimuli are probabilistic, we can define the joint probability P of observing a specific stimulus s and a specific spike train $\{t_i\}$ as $P(\{t_i\},s)$. Further, it seems reasonable to assume that the PDFs for spike trains and stimuli are linked such that when stimulus s occurs there is an associated probability of observing a specific spike train $\{t_i\}$; this can be defined as $P(\{t_i\}|s)$. This assumption characterizes the situation in which an experimenter provides a set of stimuli to a neuron while recording its spiking activity. If we now multiply the probability of observing $\{t_i\}$ given that s occurred with the probability that s indeed occurs $P(s)$, we obtain the probability to observe both s and $\{t_i\}$:

$$P(\{t_i\},s) = P(\{t_i\}|s) \times P(s) \tag{A20.1-1}$$

Eq. (A20.1-1) relates to the observation that an experimenter studying a neural response may make. Note that in Eq. (A20.1-1) the expressions $P(\ldots,\ldots)$ and $P(\ldots|\ldots)$ are defined as the probability of a combined event and the conditional probability, respectively. However, from the brain's standpoint we might reverse the reasoning we followed above. For example, the brain only "sees" spike trains $\{t_i\}$ coming in from the sensors and it must link these to a stimulus s. In other words the brain must evaluate the probability that s occurred given that $\{t_i\}$ is generated by the sensor; this probability is $P(s|\{t_i\})$. From the brain's point of view, the probability that both s and $\{t_i\}$ occur is $P(\{t_i\}|s)$ multiplied by the probability that we observe $\{t_i\}$:

$$P(\{t_i\},s) = P(\{s|\{t_i\}) \times P(\{t_i\}) \tag{A20.1-2}$$

Combining Eqs. (A20.1-1) and (A20.1-2), we have:

$$P(\{t_i\},s) = P(\{s|\{t_i\}) \times P(\{t_i\}) = P(\{t_i\}|s) \times P(s)$$

$$\rightarrow \boxed{P(\{s \mid \{t_i\}) = P(\{t_i\} \mid s) \times \frac{P(s)}{P(\{t_i\})}} \tag{A20.1-3}$$

Eq. (A20.1-3), linking both conditional probabilities, is Bayes' rule.

APPENDIX 20.2

Poisson Process

The mean μ and standard deviation σ of a Poisson process (PDF $f(t) = \rho e^{-\rho t}$, with $f(t) = 0$ for $t < 0$) are equal. This can be shown by using the-Laplace transform approach (see Appendix 3.4) or directly using Eqs. (3.9)−(3.11). For the mean we get (using the integration limits from $0 \rightarrow \infty$ because $f(t) = 0$ for $t < 0$):

$$E(t) = \int_0^\infty t\rho e^{-\rho t}dt = \left[-te^{-\rho t}\right]_0^\infty - \int_0^\infty -e^{-\rho t}dt = \left[-te^{-\rho t}\right]_0^\infty + \int_0^\infty e^{-\rho t}dt$$

$$= \left[-te^{-\rho t}\right]_0^\infty - \frac{1}{\rho}\left[e^{-\rho t}\right]_0^\infty = [0-0] - \left[0 - \frac{1}{\rho}\right] = \frac{1}{\rho} \tag{A20.2-1}$$

In the above we used integration by parts to evaluate the integral; it can be seen that $\left[-te^{-\rho t}\right]_0^\infty$ evaluates to zero because $\left[-\frac{t}{e^{\rho t}}\right] = 0$ for $t = 0$ and also for $t = \infty$. The latter can be seen replacing the exponential with a power series, or if you prefer, by using l'Hôpital's rule.

For the expectation of t^2:

$$E(t^2) = \int_0^\infty t^2\rho e^{-\rho t}dt = \left[-t^2e^{-\rho t}\right]_0^\infty - \int_0^\infty -2te^{-\rho t}dt = \left[-t^2e^{-\rho t}\right]_0^\infty + 2\int_0^\infty te^{-\rho t}dt$$

$$= \underbrace{\left[-t^2e^{-\rho t}\right]_0^\infty}_{0} + \frac{2}{\rho}\underbrace{\int_0^\infty t\rho e^{-\rho t}dt}_{E(t)=\frac{1}{\rho}} = \frac{2}{\rho^2} \tag{A20.2-2}$$

Similar to the approach we took in Eq. (A20.2-1), we used integration by parts to evaluate the integral. The expression $\left[-t^2e^{-\rho t}\right]_0^\infty$ evaluates to zero because $\left[-\frac{t^2}{e^{\rho t}}\right] = 0$ for $t = 0$ and after substituting a power series for

the exponential it can be seen that the expression is also 0 for $t = \infty$. The variance σ^2 of the Poisson process is:

$$\sigma^2 = E(t^2) - E(t)^2 = \frac{2}{\rho^2} - \frac{1}{\rho^2} = \frac{1}{\rho^2} \rightarrow \sigma = \frac{1}{\rho} \tag{A20.2-3}$$

Combining Eqs. (A20.2-1) and (A20.2-3), we find that the mean and standard deviation are both equal to $\frac{1}{\rho}$.

EXERCISES

20.1 Show that the mean and standard deviation of a Poisson process $f(t) = pe^{-pt}$ are equal by using the expectation for the moments of a distribution $\int_0^\infty t^n f(t)dt$, and compare the outcomes with those obtained with the Laplace transform of this PDF.

20.2 Show that the sum of all probabilities for the Poisson distribution is 1, i.e., show that $\sum_{i=0}^{\infty} \frac{\lambda^i}{i!} e^{-\lambda} = 1$

20.3 Consider the example of convergence between the sensory system and the olfactory bulb as described in Section 20.2. Create a MATLAB® script to compute and plot the threshold (defined as a 50% probability of firing) for convergence ratios between sensory cells and mitral cells ranging between 1 and 1000.

20.4 In MATLAB®, load the file MEA_Data.mat. It contains two traces of a broadband extracellular recording of spike activity in a cortical culture. Each channel is sampled at a rate of 25,000 samples/s, and the amplitude is scaled at 1 μV/unit.

a. Plot the spike data for both channels and calibrate the axes in s and μV.
b. Filter the data and detect the spikes in the trace and create a raster plot. Make sure you detect each spike only once! [Hint: if you want, use a variant of pr4_4.m to detect the spikes.]
c. For each channel, superimpose the spike data as you plotted in (a) and the raster you got in (b).
d. Considering your result in (c), how well did you detect the spikes?

e. Compute and plot the spike interval distribution.
f. Compute and plot the spike amplitude distribution.
g. Is this single-unit or multiunit activity?
h. Compute the Fano factor of the spike train.
i. Compute the two autocorrelations and cross-correlation for the two raster plots using the MATLAB® command xcorr.
j. Determine the spike-triggered average (STA) using the spikes in channel-1 as the trigger to average channel-2.
k. Relate the cross-correlation you found in (i) with the STA you obtained in (j). What can you conclude?
l. Can you mathematically explain your finding in (k)?

References

Cox, D.R., 1962. Renewal Theory. Methuen, London.
A classic introduction to the statistics of renewal theory. This approach has a high relevance for statistics of spike trains.

Duchamp-Viret, P., Duchamp, A., Vigouroux, M., 1989. Amplifying role of con-vergence in olfactory system a comparative study of receptor cell and second-order neuron sensitivities. J. Neurophysiol. 61, 1085–1094.

Eissa, T.L., Dijkstra, K., Brune, C., Emerson, R.G., van Putten, M.J.A.M., Goodman, R.R., McKhann, G.M., Schevon, C.A., van Drongelen, W., van Gils, S.A., 2017. Cross-scale effects of neural interactions during human neocortical seizure activity. PNAS 114, 10761–10766.

Hodgkin, A.L., Huxley, A.F., 1952. A quantitative description of membrane current and its application to conduction and excitation in the nerve. J. Physiol. 117, 500–544.
A seminal paper describing the Hodgkin and Huxley equations.

Rieke, F., Warland, D., de Ruyter van Steveninck, R., Bialek, W., 1999. Spikes Exploring the Neural Code. MIT Press, Cambridge, MA.
A unique book introducing spike train analysis from a probabilistic point of view.

Shannon, C.E., Weaver, W., 1949. The Mathematical Theory of Communication. University of Illinois Press, Urbana, IL.

Van Drongelen, W., Holley, A., Døving, K.B., 1978. Convergence in the olfactory system: quantitative aspects of odour sensitivity. J. Theor. Biol. 71, 39–48.
A modeling study in which neurons are represented as spike generators following a Poisson process. Based on this representation, the amplification effect of convergence of neural elements is estimated.

Van Drongelen, W., Koch, H., Marcuccilli, C., Peña, F., Ramirez, J.M., 2003. Syn-chrony levels during evoked seizure-like bursts in mouse neocortical slices. J. Neurophysiol. 90, 1571–1580.
Analysis of single- and multiunit recordings of neural activity. A basic application of the estimation of information content and entropy in spike trains.

CHAPTER

21

Wavelet Analysis: Time Domain Properties

21.1 INTRODUCTION

Although the mathematics for wavelet analysis have existed for about a century, most of its applications in signal processing, feature detection, and data compression have been developed over the past few decades. Wavelet analysis is very useful for analyzing physiological systems because, as opposed to most classical signal analysis approaches, it provides the means to detect and analyze nonstationarity in signals. The simplest wavelet is the Haar wavelet, first described in the early 1900s by Alfred Haar. A few other famous, more recent contributors to the field of wavelet analysis are Morlet, Mallat, and Daubechies. A great practical and simple introduction into wavelets is given by Walker (1999), and a thorough overview can be found in Mallat (1998).

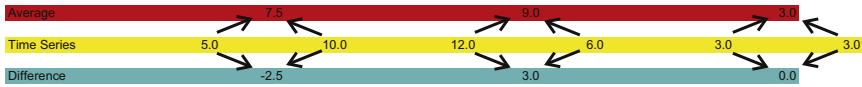
In this chapter we will introduce the techniques of wavelet analysis using the simplest example, the ***Haar*** wavelet. In the follow-up sections we will extend this to the application of the ***Daubechies*** wavelet. First we will explore the procedures used to obtain the wavelet transform and then discuss some of the mathematical details. Frequency domain properties of the wavelet transform are introduced in Chapter 22.

21.2 WAVELET TRANSFORM

The underlying principle of wavelet analysis is most easily explained by considering a sampled time series [5.0, 10.0, 12.0, 6.0, 3.0, 3.0, …]

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00021-7

which we want to examine for trends and fluctuations over subsequent pairs:



In this example, the average (red) of subsequent pairs is $[x(n-1)+x(n)]/2$, and the difference (blue) is $[x(n-1)-x(n)]/2$. In this process no information was lost because the original time series (yellow) can be reconstructed from a combination of the average and difference vectors: the first value (5.0) is the sum of the average and difference ($7.5-2.5=5.0$), the second point of the time series is ($7.5-(-2.5)=10.0$), etc. Because the time series contains the same information as the average and difference signals, we may represent it either in its original, raw form as [5.0, 10.0, 12.0, 6.0, 3.0, 3.0, …] or in a transformed fashion as a combination of the average and difference forms [7.5, 9.0, 3.0, …] [−2.5, 3.0, 0.0, …] As we will see in the following, aside from a factor of $\sqrt{2}$, application of the Haar wavelet transform is identical to calculating the average and difference as shown above.

21.2.1 Haar Wavelet and Scaling Signals

The level-1 Haar wavelet and the associated scaling signal are shown in Fig. 21.1. In this section we start with a level-1 wavelet, leaving an introduction to the meaning of different-level transforms and higher-level wavelets for Section 21.2.4. Both signals in Fig. 21.1 are square waves with amplitudes of $\frac{1}{\sqrt{2}}$, the wavelet is biphasic and the scaling signal is nonnegative. Let's consider the transform of an input signal of N samples. The first step is to define the level-1 Haar wavelet (W) and scaling signal (S) as vectors of length N:

$$W_1^1 = \left[\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, 0, 0, \ldots, 0\right] \tag{21.1}$$

and

$$S_1^1 = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0, \ldots, 0\right] \tag{21.2}$$

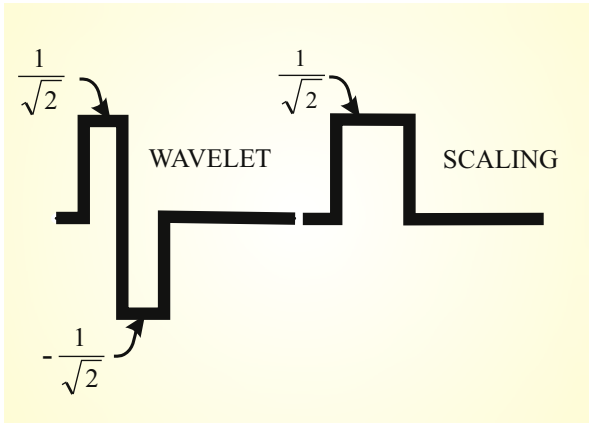


FIGURE 21.1 The level-1 Haar wavelet and scaling signal.

In W and S, the superscripts indicate that the signals are level-1 and subscripts indicate that both signals start at the first position in the vectors.

As in the numerical example above, we will use the scaling and wavelet signals to determine the ***trend*** (=weighted average) and the ***fluctuation*** (= weighted difference) in a time series. To demonstrate this, we start with a function G that is sampled N times at regular time intervals:

$$G = [g_1, g_2, g_3, \ldots, g_N] \tag{21.3}$$

The trend of the first two points can be obtained from:

$$t_1 = \frac{g_1 + g_2}{\sqrt{2}} = G \cdot S_1^1 \tag{21.4}$$

The second part of Eq. (21.4) shows that t_1 is the ***scalar product*** of vectors G and S_1^1. See Appendix 21.1 if you need to review the definition of the scalar product. Similarly, the fluctuation between the first two points is:

$$f_1 = \frac{g_1 - g_2}{\sqrt{2}} = G \cdot W_1^1 \tag{21.5}$$

Again, Eq. (21.4), the second part of Eq. (21.5) shows that f_1 is the scalar product of vectors G and W_1^1.

Notes:

- The reason for the weighting factor, i.e., division by the $\sqrt{2}$ instead of simply dividing by 2, is to preserve the energy content across the transformed variables. This will be further discussed in Section 21.2.3.
- Obtaining the trend as the sum of g_1 and g_2 weighted by $1\sqrt{2}$ effectively means that the average of the two data points is multiplied by $\sqrt{2}$ (i.e., $\frac{g_1+g_2}{2}\sqrt{2} = \frac{g_1+g_2}{\sqrt{2}}$). The same relationship exists between the fluctuation and the difference.

Continuing with this procedure, we shift wavelet and scaling signals by two positions:

$$W_2^1 = \left[0, 0, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}, \ldots, 0\right] \tag{21.6}$$

and

$$S_2^1 = \left[0, 0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \ldots, 0\right] \tag{21.7}$$

Note that the subscripts for S and W are now 2, reflecting the shift of the signals to the second pair of data points. We now repeat the process of calculating the trend and fluctuation. Using the scalar product notation, the weighted average of points 3 and 4 can be obtained from:

$$t_2 = G \cdot S_2^1 \tag{21.8}$$

The weighted difference between points 3 and 4:

$$f_2 = G \cdot W_2^1 \tag{21.9}$$

We continue to shift the wavelet and scaling signals in steps of 2 until the end of signal G. Because the length of G is N, we will obtain $N/2$ trend

values and $N/2$ fluctuation values, i.e., for all $m = 1, 2, 3, \ldots, N/2$ we obtain the following expression for the trend values:

$$t_m = \frac{g_{2m-1} + g_{2m}}{\sqrt{2}} = G \cdot S_m^1 \tag{21.10}$$

Similarly, the weighted difference between subsequent pairs of points is:

$$f_m = \frac{g_{2m-1} - g_{2m}}{\sqrt{2}} = G \cdot W_m^1 \tag{21.11}$$

We now group all the weighted averages and differences into two vectors:

$$a^1 = \left[t_1, t_2, \ldots\ldots\ldots, t_{N/2}\right] \tag{21.12}$$

$$\text{and } d^1 = \left[f_1, f_2, \ldots, f_{N/2}\right] \tag{21.13}$$

The superscripts of a and d indicate that we have used level-1 vectors. Again, the subscripts of the elements t and f indicate the position of the wavelet and scale signals within the original N data points.

21.2.2 Level-1 Haar Transform and the Inverse Haar Transform

The level-1 Haar transform can be defined from the above as a procedure to determine the trend and fluctuation components of a signal. In the example above the level-1 Haar transform of G is a^1 and d^1:

$$G \overset{H_1}{\mapsto} \left(a^1 \middle| d^1\right) \tag{21.14}$$

In the example time series [5.0, 10.0, 12.0, 6.0, 3.0, 3.0] we used earlier in Section 21.2, the level-1 Haar transform produces the two vectors: [7.5 √2, 9.0 √2, 3.0 √2], [-2.5 √2, 3.0 √2, 0.0]; a result very similar to the averages and differences calculated in Section 21.2. A graphical example of a 1-level Haar transform is shown in Fig. 21.2. The input signal (red) consists of a set of oscillations. The results of the level-1 Haar transform (black) consist of two main parts: the trend (1-512) and the fluctuation (513-1024).

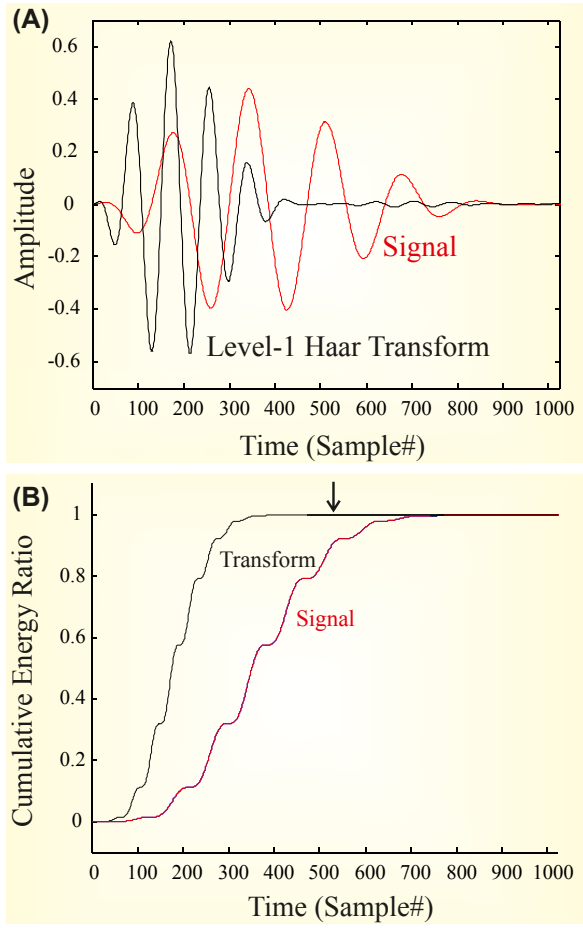


FIGURE 21.2 Example of a level-1 Haar transform. (A) The transform (black) of an input wave (red). (B) The cumulative energy shows that for the transformed wave the trend signal contains most of the energy; i.e., at point 512 (*arrow*) the ratio is ~1.

The first half of the transform result—produced with the scaling signal *S*—contains high amplitudes, while the second part of the transform—produced with the wavelet *W*—produces a low-amplitude signal. This seems a trivial observation, but it is a critical aspect of the analysis (for reasons that will soon become clear). The cumulative energy plot in Fig. 21.2B is further discussed in Section 21.2.3.

The MATLAB® Routine to produce Fig. 21.2A.

```
% pr21_1.m
% A level-1 Haar Wavelet Analysis

clear;

N=1024;                                    % # of points

for n=1:N;m=(n-1)/N;g(n)=20*m^2*(1-m)^4*cos(12*pi*m);end;
                                           % input signal

for m=1:N/2;
   a(m)=(g(2*m-1)+g(2*m))/sqrt(2);         % Use direct formulas
                                              for t and f

   d(m)=(g(2*m-1)-g(2*m))/sqrt(2);
end;

H1=[a d];                                  % The level-1 Haar transform


% plot results
figure
plot(g,'r');
hold
plot(H1,'k');
axis([0 1024 -0.7 0.7]);
xlabel ('Time (Sample#)')
ylabel ('Amplitude')
title(' Original Signal (red) and 1-Level Haar Transform (black)')
```

The **inverse Haar transform** starts from the a^1 and d^1 transformed vectors and allows us to recreate the original function G again. In the particular case of the Haar transform, the inverse procedure can be expressed in the form of a summation if we define A^1 and D^1 as:

$$A^1 = \left[t_1, t_1, t_2, t_2, \ldots, t_{N/2}, t_{N/2}\right] \Big/ \sqrt{2} \tag{21.15}$$

$$\text{and } D^1 = \left[f_1, -f_1, f_2, -f_2, \ldots, f_{N/2}, -f_{N/2}\right] \Big/ \sqrt{2} \tag{21.16}$$

Please note that the doubling of t_m and f_m are not typos and that each second f_m of the pair is associated with a minus sign. Also note that all terms in Eqs. (21.15) and (21.16) are divided by $\sqrt{2}$. This corrects for the fact that we multiplied the average and difference with $\sqrt{2}$ (see Eqs. 21.4 and 21.5 with the associated Notes). The inverse Haar transform is therefore simply the sum of both vectors, i.e.,:

$$G = A^1 + D^1 \tag{21.17}$$

Eqs. (21.15) and (21.16) can be put in a (more formal) vector form:

$$\begin{aligned} A^1 &= t_1 S_1^1 + t_2 S_2^1 + \ldots + t_{N/2} S_{N/2}^1 \\ &= \left(G \cdot S_1^1\right) S_1^1 + \left(G \cdot S_2^1\right) S_2^1 + \ldots + \left(G \cdot S_{N/2}^1\right) S_{N/2}^1 \end{aligned} \tag{21.18}$$

$$\begin{aligned} D^1 &= f_1 W_1^1 + f_2 W_2^1 + \ldots + f_{N/2} W_{N/2}^1 \\ &= \left(G \cdot W_1^1\right) W_1^1 + \left(G \cdot W_2^1\right) W_2^1 + \ldots + \left(G \cdot W_{N/2}^1\right) W_{N/2}^1 \end{aligned} \tag{21.19}$$

Note that each term in Eqs. (21.18) and (21.19) is a vector. In Eq. (21.18), $\left(G \cdot S_i^1\right)$ is a scalar product representing t_i; the factor $\left(G \cdot S_i^1\right)$ multiplied with vector S_i^1 produces a vector $[0, 0, \ldots, t_i, t_i, \ldots, 0]/\sqrt{2}$. The sum of these vectors for all i generates the expression for A^1 in Eq. (21.15). The same procedure can be followed for the wavelet $\left(G \cdot W_i^1\right)$ to obtain Eq. (21.19). The advantage of this notation is that it is easily extended from level-1 to level-k and can also be applied to inversion of other transforms besides the Haar.

21.2.3 Energy of the Level-1 Transform

The Haar transform looks fairly simple (a weighted average and weighted difference), the only apparent nuisance in this simple transform is the $\sqrt{2}$ factor that appears in the wavelet definition, the transform and the inverse transform. There is a reason for this $\sqrt{2}$ correction, namely the conservation of energy across domains. As with the Fourier transform, we would like to keep the energy content of the signal the same across the signal transformations and inverse transformations (Parseval's theorem, Appendix 7.1). For the level-1 Haar transform, the necessary correction factor is $\sqrt{2}$; though this normalization factor is necessarily different for higher levels of the Haar transform (discussed in Section 21.2.4) or different types of wavelet transforms.

Here, we define the energy of a sample as the square of the sampled value. This means that the energy of the first two samples of G is $g_1^2 + g_2^2$, the first elements derived in the transform from these samples are t_1 and f_1. Thus to preserve energy in this representation we want that:

$$g_1^2 + g_2^2 = t_1^2 + f_1^2 \tag{21.20}$$

We want this relationship to be true for all pairs and their associated trend and fluctuation values. We can use Eqs. (21.10) and (21.11) to show that the relationship in Eq. (21.20) is indeed correct for all m:

$$t_m^2 = \frac{g_{2m-1}^2 + g_{2m}^2 + 2g_{2m-1}g_{2m}}{2}$$

$$f_m^2 = \frac{g_{2m-1}^2 + g_{2m}^2 - 2g_{2m-1}g_{2m}}{2}$$

$$t_m^2 + f_m^2 = g_{2m-1}^2 + g_{2m}^2 +$$

As shown in Section 21.2.2 (Eq. 21.17), the inverse transform procedure exactly recreates the function G and therefore this procedure must also preserve the energy content.

Interestingly, if we look into the distribution of energy in the original and transformed signal in Fig. 21.2, most energy is transferred to the trend part of the transform and very little shows up in the fluctuation signal (e.g., $f_m \approx 0$). This distribution becomes clear if we look at the cumulative energy distribution of the squared signals. In MATLAB® you may use the command `cumsum` to calculate the cumulative sum of the elements in a vector. ***After running pr21_1.m***, type the following commands to evaluate the distribution of the energy.

```
figure;hold;
plot(cumsum(g.^2)/max(cumsum(g.^2)))
plot(cumsum(H1.^2)/max(cumsum(g.^2)),'r')
```

Align the obtained plot of the cumulative energy with the one obtained with the script (as in Fig. 21.2). Because both cumulative plots are normalized to the maximum power of the input signal `max(cumsum(g.^2))`, you can see that at around sample 512 (i.e., the transition point from the trend vector to the fluctuation values, arrow in Fig. 21.2B) close to 100% of the energy is already captured (you can use the `[x y]=ginput` command to read x and y values from the figure).

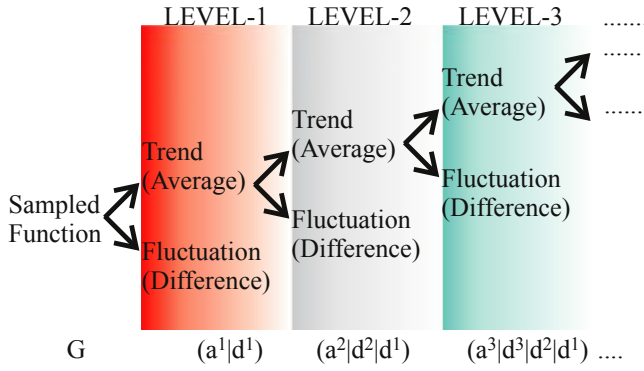


FIGURE 21.3 Higher-level Haar transforms of a sampled function *G*.

21.2.4 Multiresolution Analysis

The procedure we described above for the level-1 Haar transform can be repeated multiple times. By recursively applying the transform to the trend signal (the weighted average), we obtain higher-level transforms (Fig. 21.3). Note that we leave the fluctuation (= weighted difference) intact and continue to split the weighted average only!

> *Note*: This is only one possible approach. The so-called wavelet packet transform transforms **both** the trends and fluctuations. The wavelet packet transform (using the Haar scaling and wavelets) is called the Walsh transform.

Using the transform in Eq. (21.14) repeatedly we get:

$$G \overset{H_1}{\mapsto} (a^1 \quad d^1) \xrightarrow{H_1} (a^2 \quad d^2)$$

Combining these results after using the level-1 transform twice creates the level-2 transform:

$$G \overset{H_2}{\mapsto} (a^2 d^2 d^1)$$

Generally at the level-n transform we get:

$$G \overset{H_n}{\mapsto} \left(a^n d^n d^{n-1} \ldots d^1\right)$$

If we don't spend too much time thinking about possible optimization techniques to accomplish this procedure in fewer steps, we can simply use the algorithm from the level-1 Haar program multiple times to obtain the repeated action, the result of which is depicted in Fig. 21.4.

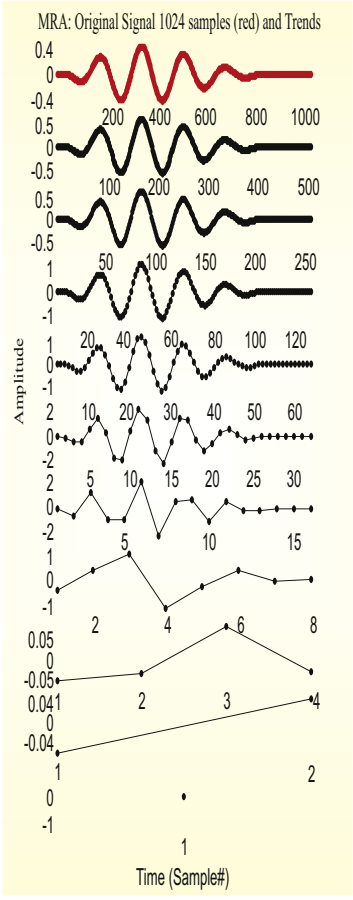


FIGURE 21.4 Multiresolution analysis (MRA) showing the trends (averages) of subsequent levels of the Haar transform using the procedure depicted in Fig. 21.3. The fluctuation signals are not shown here but can be obtained from MATLAB® script pr21_2.m.

The following listing is a snippet of a MATLAB® sample program performing MRA.

```
% pr21_2.m
% multihaar
% Multi Resolution Analysis MRA Haar Wavelet Analysis
% by a repeated level-1 transform

clear;

N=1024;                                          % # of points

for n=1:N;m=(n-1)/N;g(n)=20*m^2*(1-m)^4*cos(12*pi*m);end;
                                                 % input signal

for m=1:N/2;
   a1(m)=(g(2*m-1)+g(2*m))/sqrt(2);              % Use direct formulas for
                                                    t and f
                                                 % (See Eqs. 21.4 and 21.5)
   d1(m)=(g(2*m-1)-g(2*m))/sqrt(2);
end;

H1=[a1 d1];                                      % The 1-level Haar
                                                    transform

for m=1:N/4;
   a2(m)=(a1(2*m-1)+a1(2*m))/sqrt(2);            % Use direct formulas
                                                    for t and f
   d2(m)=(a1(2*m-1)-a1(2*m))/sqrt(2);
end;

H2=[a2 d2 d1];                                   % The 2-level Haar
                                                    transform

for m=1:N/8;
   a3(m)=(a2(2*m-1)+a2(2*m))/sqrt(2);            % Use direct formulas
                                                    for t and f
   d3(m)=(a2(2*m-1)-a2(2*m))/sqrt(2);
end;

H3=[a3 d3 d2 d1];                                % The 3-level Haar
                                                    transform

% ETC ETC complete the analysis up to a10 and d10 to get Fig. 21.4
```

The MRA result of pr22_2.m is shown in Fig. 21.4. The output of this script will also display graphs of the fluctuation signals. The input signal is the same as that in Fig. 21.2A and the first trend signal corresponds with the left half of the transformed signal in Fig. 21.2A.

The idea in multiresolution analysis is to repeatedly use the level-1 transforms, effectively leading to higher-level scaling and wavelet signals. For the sake of computational efficiency however, instead of applying the level-1 transform repeatedly, one can also formulate higher-level transforms directly. To generate these higher-level functions we start with a general form of the wavelet, which for the Haar wavelet can be represented in continuous time as a biphasic square wave:

$$W_H(t) = \begin{cases} 1 & \text{if} \quad 0 \le t < \dfrac{1}{2} \\ -1 & \text{if} \quad \dfrac{1}{2} \le t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{21.21}$$

The function W_H is the so-called mother wavelet. As we observed in the examples in Section 21.2.2, we obtain the wavelet transform of an input time series by ***translating*** the wavelet operation over the input. Similarly, to study the correlation at different scales, the mother wavelet is stretched (***dilated***). By using wavelets of ***different scales, we can produce different levels of the wavelet transform***.

The dilation k and translation n of the mother wavelet can be expressed by:

$$W(t)_n^k = \frac{1}{\sqrt{2^k}} W_H\left(\frac{t - 2^k n}{2^k}\right) \tag{21.22}$$

In the discrete time version, the ***support*** (set of time indices where the wavelet is non-zero) for the k-level wavelet is 2^k. For instance, the level-2 Haar wavelet is:

$$W_1^2 = \left[\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, 0, 0, \ldots, 0\right] \tag{21.23}$$

Compare the above wavelet with the level-1 Haar wavelet W_1^1 in Eq. (21.1). Note that the nonzero values change by a factor of $\frac{1}{\sqrt{2^k}}$, and the support increased from 2 to 4 points.

21.2.5 Application of Wavelets in Signal Compression and Denoising

Considering Fig. 21.4, where the features of the original waveform are preserved in fewer samples, it is not difficult to imagine that the energy compaction property of the Haar transform could be used for data compression purposes. The original signal in Fig. 21.4 (red) contains 1024 samples; each subsequent trend signal reduces the number of samples by a factor of 2. Of course, compression will at some point only be accomplished at the cost of detail in the signal (energy that is stored in the fluctuation parts of the transform). However, in the example in Fig. 21.4 it can be seen that the first steps, reducing 1024 to 256 points, seem to preserve the overall signal features rather well. Further compression beyond 256 points leads to severe loss of signal properties.

Alternatively, wavelet transforms may help in signal processing tasks such as the removal of noise. For example, if a signal is contaminated with high-frequency noise, the fluctuation part of the transform will mainly represent the noise component. Removal of the fluctuation signal, followed by an inverse transform, may be an efficient approach to "clean" up time series and pictures.

21.3 OTHER WAVELET FUNCTIONS

Currently a large set of different wavelets and wavelet analysis packages are available in signal processing. Depending on their purpose (e.g., signal compression, detection of transient phenomena in the time domain, quantifying instantaneous frequency components, etc.), they include real (e.g., Haar wavelet) and complex (e.g., Morlet wavelet), even symmetric (e.g., Mexican Hat wavelet) and odd symmetric (e.g., Haar wavelet) forms, etc. This rich set of types of varied wavelet and associated scaling signals is possible because they only have to satisfy a few fairly reasonable conditions (Appendix 21.2). It would be beyond the scope of this introduction to discuss different types of wavelets, but we want to consider at least one other well-known type, the Daubechies wavelet, in the following section. The purpose of introducing the Daub4 scaling signal and wavelet is to illustrate how different types of signals are optimized for different signal processing tasks.

21.3.1 Daubechies Wavelet

The Daub4 scaling signal and wavelet have a support of 4 points. The four values $Ds4(i)$ for the scaling signal for Daub4 are:

$$Ds4(1) = \frac{1+\sqrt{3}}{4\sqrt{2}} \quad Ds4(2) = \frac{3+\sqrt{3}}{4\sqrt{2}} \quad Ds4(3) = \frac{3-\sqrt{3}}{4\sqrt{2}} \quad Ds4(4) = \frac{1-\sqrt{3}}{4\sqrt{2}} \tag{21.24}$$

Using these values for the analysis of a time series with n samples, we get the following scaling signals:

$$\begin{aligned} DS4_1^1 &= [Ds4(1), Ds4(2), Ds4(3), Ds4(4), 0, 0, 0, \ldots, 0] \\ DS4_2^1 &= [0, 0, Ds4(1), Ds4(2), Ds4(3), Ds4(4), 0, \ldots, 0] \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ DS4_{(n/2)-1}^1 &= [0, 0, 0, \ldots, 0, Ds4(1), Ds4(2), Ds4(3), Ds4(4)] \\ DS4_{n/2}^1 &= [Ds4(3), Ds4(4), 0, 0, 0, \ldots, 0, Ds4(1), Ds4(2)] \end{aligned} \tag{21.25}$$

Note that the level-1 scaling signal translates in steps of two as does the Haar scaling signal. The difference is that in the last step ($DS4_{n/2}^1$ in Eq. 21.25), the coefficients **wrap around** to the beginning of the vector.

The associated Daub4 wavelet is defined by:

$$\begin{aligned} Dw4(1) &= \frac{1-\sqrt{3}}{4\sqrt{2}} \quad Dw4(2) = \frac{\sqrt{3}-3}{4\sqrt{2}} \\ Dw4(3) &= \frac{3+\sqrt{3}}{4\sqrt{2}} \quad Dw4(4) = \frac{-1-\sqrt{3}}{4\sqrt{2}} \end{aligned} \tag{21.26}$$

The level-1 translations are

$$\begin{aligned} DW4_1^1 &= [Dw4(1), Dw4(2), Dw4(3), Dw4(4), 0, 0, 0, \ldots, 0] \\ DW4_2^1 &= [0, 0, Dw4(1), Dw4(2), Dw4(3), Dw4(4), 0, \ldots, 0] \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ DW4_{(n/2)-1}^1 &= [0, 0, 0, \ldots\ldots\ldots, 0, Dw4(1), Dw4(2), Dw4(3), Dw4(4)] \\ DW4_{n/2}^1 &= [Dw4(3), Dw4(4), 0, 0, 0, \ldots\ldots\ldots, 0, Dw4(1), Dw4(2)] \end{aligned} \tag{21.27}$$

Note that the last one, $DW4_{n/2}^1$, also wraps around.

The example in Fig 21.5 shows the results of the level-1 Haar and Daubechies transform on two types of signal. In this example it can be seen that the Haar and Daub4 wavelets have different talents with respect to successfully compressing signals. The Daub4 transform compresses the oscillatory waveform in the upper panel of Fig. 21.5 rather well, i.e., there is not much energy left in the fluctuation signal. The square wave however, is more efficiently compressed by the Haar transform.

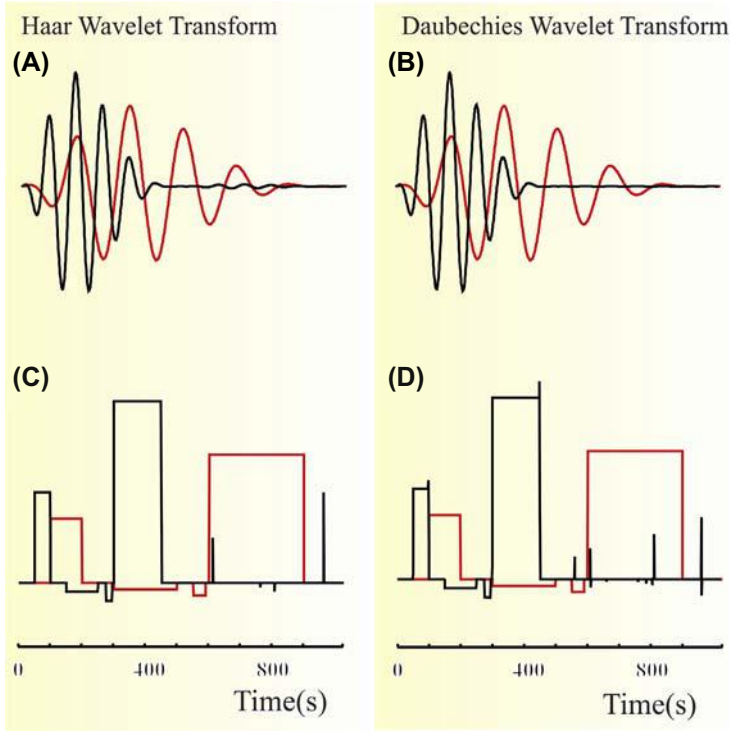


FIGURE 21.5 Level-1 wavelet transforms of an oscillatory signal and a signal with abrupt transients. Both waves were analyzed using the Haar and Daubechies (Daub4) wavelets. The red signal is the original, input, and the black traces represent the first average and difference signals (the same arrangement as in Fig. 21.2). It can be seen that compression of the oscillatory wave in (A) and (B) is done most efficiently by the Daubechies wavelet, i.e., the d^1 signal is almost 0 in the latter case. For the wave shown in (C) and (D), the Haar wavelet compresses better. The plots can be obtained with MATLAB® scripts haar1.m, haar2.m, daubechies1.m, and daubechies2.m.

Run daubechies1 in MATLAB® and compare the output of this program with daubechies2, haar1, and haar2 scripts.

```
% daubechies1
% Level-1 Daubechies Wavelet Analysis

clear;

% Define the Daub4 scaling (alpha) and wavelet (beta) coeff
alpha(1)=(1+sqrt(3))/(4*sqrt(2));
alpha(2)=(3+sqrt(3))/(4*sqrt(2));
```

```
alpha(3)=(3-sqrt(3))/(4*sqrt(2));
alpha(4)=(1-sqrt(3))/(4*sqrt(2));
beta(1)=alpha(4);
beta(2)=-alpha(3);
beta(3)=alpha(2);
beta(4)=-alpha(1);

% # of points
N=1024;

% input signal
for n=1:N;m=(n-1)/N;
   g(n)=20*m^2*(1-m)^4*cos(12*pi*m);
end;

% Ignore the wrap around at the end!!
for m=1:N/2-2;
   % Use direct formulas for t and f
   a(m)=(g(2*m-1)*alpha(1)+g(2*m)*alpha(2)+g(2*m+1)*alpha(3)+
   g(2*m+2)*alpha(4));
   d(m)=(g(2*m-1)*beta(1)+g(2*m)*beta(2)+g(2*m+1)*beta(3)+
   g(2*m+2)*beta(4));
end;

% The level-1 Daub4 transform
D1=[a d];

figure

plot(g,'r');
hold
plot(D1,'k');
axis([0 1024 -0.7 0.7]);
xlabel ('Time (Sample#)')
ylabel ('Amplitude')
title(' Original Signal (red) and Level-1 Daubechies Transform
  (black)')
```

Both the Haar and Daubechies wavelets can be applied at different levels and used to compress signals. The more closely the wavelet matches the input curve, the closer the difference signal is to zero, and the better the quality of the compression in the average signal. Better quality is judged by the level of energy of the original signal that is preserved by

the average or the (loss of) energy present in the fluctuation (e.g., Fig. 21.2B). Progressively higher levels of Daub wavelets are designed so that they fit higher-order polynomials. As a general rule one can apply a Daub$N$ wavelet transform to polynomials of the order $<N/2$. For instance, if the input signal over the support of the wavelet is largely linear, one uses a Daub4 wavelet; for a quadratic signal, one applies the Daub6 wavelet, etc. In other words, if the input function over the support of a j-level Daub$N$ wavelet is a polynomial of the order $<N/2$, then the difference signal (fluctuation) ≈ 0. For obvious reasons this feature plays an important role when compressing data sets.

21.4 2-D APPLICATION

Just as you apply a 1-D wavelet transform on a vector, you can also apply the same procedure to a 2-D matrix. Such applications are used in image compression and analysis. In this case the average/trend image can be considered a compressed version of the original data, while the difference/fluctuation image shows how successful compression was. Alternatively, similar to the filter procedure shown in Fig. 18.4 (Chapter 18), one can use the difference images as edge detectors. This property can also be used to enhance edges in images by multiplication of the difference signal of a transformed image with a factor > 1, followed by an inverse transform.

An example of a wavelet transform of an image is shown in Fig. 21.6. When transforming an image with the Haar wavelet, we follow the same procedure as we used in Eq. (21.14) for both the horizontal (rows) and vertical (columns) directions. We first perform a level-1, 1-D wavelet transform on all rows and create a new image where each row contains a trend and fluctuation component. On this new image we subsequently perform the same wavelet transform on each column. To summarize, the 2-D wavelet transform on an image consists of two sequential 1-D transforms on its rows and columns. We would obtain the same result by interchanging the order and perform the transform first on the columns followed by the rows.

In the end result of the level-1, 2-D transform we have four areas in the output image. One area containing the trend for both directions is the average image (a^1). A second area contains the trend in the horizontal direction and the fluctuation in the vertical direction (f^V). This procedure enhances change in the vertical direction and is insensitive to vertical edges (which is a change in the horizontal direction). Therefore f^V is a detector of oblique and horizontal edges. The third area (f^H) includes the trend in the vertical direction and the fluctuation in the horizontal direction. Because f^H evaluates the fluctuation along the rows of the image, it is insensitive to horizontal edges and it acts as a detector of vertical and oblique edges. Finally, a fourth area (f^D) includes fluctuations along both

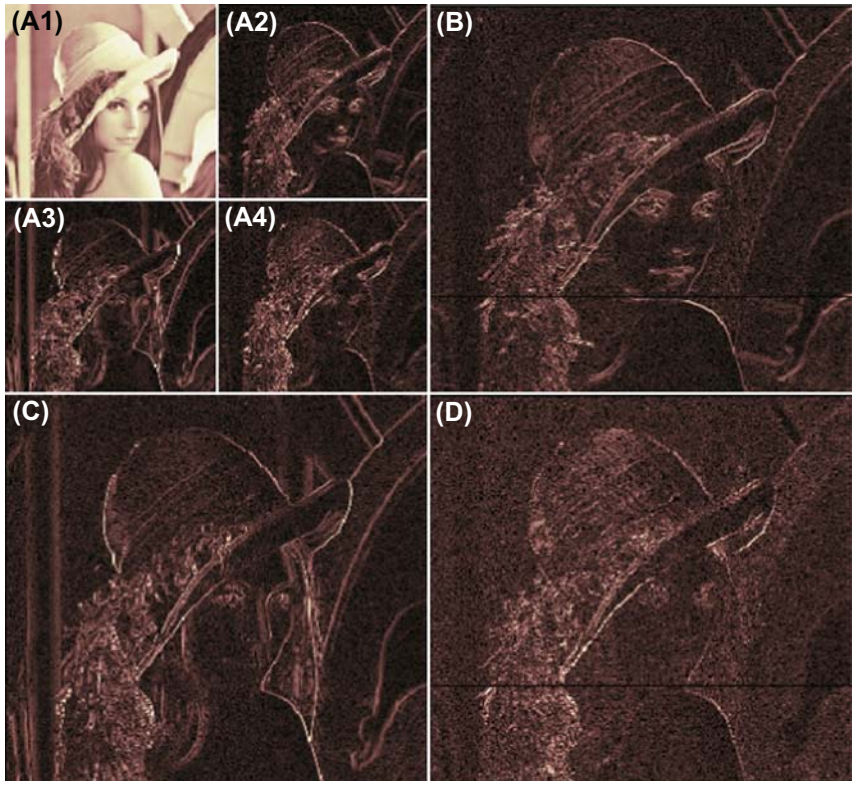


FIGURE 21.6 2-D Haar wavelet transform of an image. The top-left panel (A1) shows the trend (compressed) image and the fluctuations (edges) are shown in the remaining panels. For instance the top right panel (B) shows the level-1 fluctuations in the columns (*vertical lines*) and therefore includes horizontal and oblique edges; the bottom left panel (C) is the result of level-1 fluctuations along the rows (*horizontal lines*) and predominantly depicts the vertical and oblique edges. The bottom right panel (D) is a combination of both vertical and horizontal procedures and therefore mainly depicts the oblique edges. The panels (B), (C) and (D) represent the level-1 Haar transform fluctuation, while panels (A2), (A3), and (A4) represent the equivalent fluctuations for the level-2 transform. Accordingly, (A1) depicts the trend result of the level-2 transform. Note that, as compared to Fig. 21.7, the positions of f^H and f^V are switched.

horizontal and vertical directions, thereby removing horizontal and vertical edges, while the oblique edges remain. The result of the transform of image I can be arranged in four panels (please note that the position of a^1, f^H, f^V, and f^D in the four quadrants may differ between authors):

$$I \mapsto \left(\begin{array}{c|c} a^1 & f^H \\ \hline f^V & f^D \end{array} \right) \tag{21.28}$$

An example of a level-1 transform of an image is shown in Fig. 21.7. In this case we use a simple arrangement of horizontally, vertically, and diagonally oriented lines (Fig. 21.7, top left). The transforms in terms of

average and fluctuations are depicted in the bottom right panel in Fig. 21.7.

Just as with a one-dimensional time series, the procedure we followed to transform the original image I can be repeated on a^1 to obtain the level-2 transform. Repeating the same transform recursively leads to multi-resolution analysis applied to images. In this case the upper left quadrant occupied by a^1 is split again into four subpanels containing a^2 and the associated fluctuation signals. A concrete example of a level-2 Haar transform of Lena's image is shown in Fig. 21.6. Note that in this example, the positions of fluctuations f_H and f_V are switched as compared to the positions in Fig. 21.7 and Eq. (21.28).

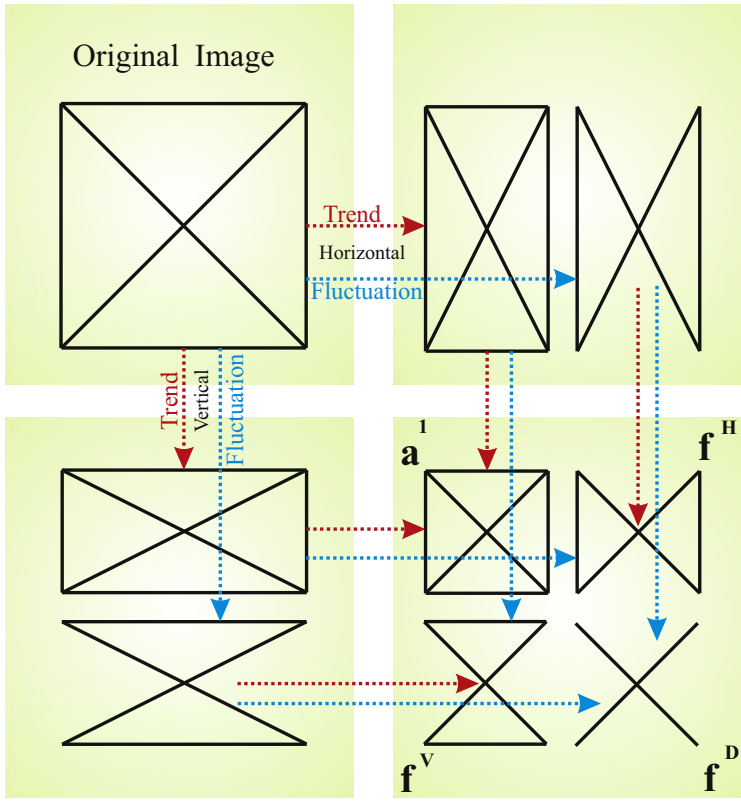


FIGURE 21.7 Example of a level-1 Haar wavelet transform on a set of vertical, horizontal, and diagonal lines (top left). The horizontal trend and fluctuation are depicted in the top right panel and the vertical trend and fluctuation in the bottom left panel. The trend and fluctuation of both the top right and bottom left panels create the (same) level-1 transform depicted in the bottom right panel. Here we obtain the trend (a^1), and the fluctuations for horizontal (f^H), vertical (f^V), and oblique (f^D) lines.

APPENDIX 21.1

The scalar product of two vectors $\vec{a}$ and $\vec{b}$ generates a single, scalar value. In this appendix we include arrows to indicate vectors; in the text of this chapter we simplify the notation and omit the arrows for the vectors. The scalar product of two vectors is also called the inner product or the dot product and it is usually denoted as $\vec{a} \cdot \vec{b}$ (note the dot). In case of two-dimensional vectors we can represent $\vec{a}$ by two elements $a_1 a_2$, and $\vec{b}$ by $b_1 b_2$. The scalar product $\vec{a} \cdot \vec{b}$ can now be written as

$$[a_1 a_2] \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = a_1 b_1 + a_2 b_2 \qquad \text{(A21.1-1)}$$

Note that in this notation, the first vector is a row vector and the second vector is a column vector. The inner product is obtained by multiplying each element in the row vector $\vec{a}$ by its corresponding element in the column vector $\vec{b}$, and then adding these results together. As you can see, the end result is a single value, hence the name scalar product, which is the procedure we use in Eqs. (21.4), (21.5), (21.8)–(21.11).

Alternatively, one can show that the scalar product of $\vec{a}$ and $\vec{b}$ can be obtained by

$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}|\cos\theta \qquad \text{(A21.1-2)}$$

Here $|\vec{a}|$ and $|\vec{b}|$ denote the magnitudes of the vectors $\vec{a}$ and $\vec{b}$, and θ the angle between them (Fig. A21.1).

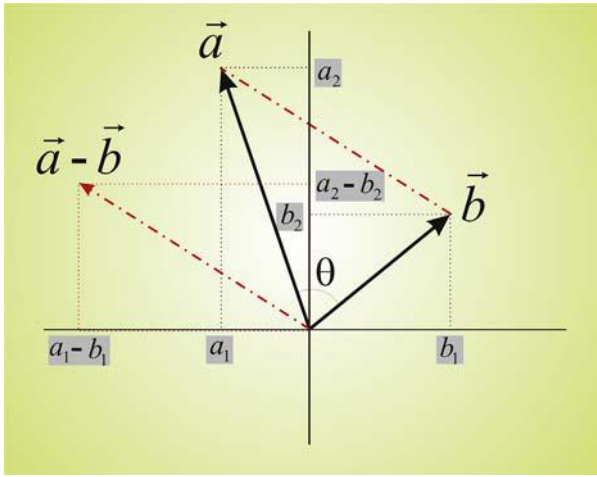


FIGURE A21.1 Vectors $\vec{a}$ and $\vec{b}$ and their angle θ in a plane (R^2). Their difference $\vec{a} - \vec{b}$ is indicated in red. Vector coordinates a_1, a_2, b_1, … are indicated in the *gray boxes*. Note that the line between $\vec{a}$ and $\vec{b}$ equals $\vec{a} - \vec{b}$.

We show that the expressions in Eqs. (A21.1-1) and (A21.1-2) are equivalent by applying the cosine rule to the triangle formed by $\vec{a}$, $\vec{b}$ and the line equal to $\vec{a} - \vec{b}$ in Fig. A21.1.

$$|\vec{a} - \vec{b}|^2 = |\vec{a}|^2 + |\vec{b}|^2 - 2|\vec{a}||\vec{b}|\cos\theta \qquad \text{(A21.1-3)}$$

APPENDIX 21.2

A wavelet basis function W, such as the one in Eq. (21.21), must satisfy a set of conditions. Two of these conditions relate to the time domain: the wavelet must have zero average $\int_{-\infty}^{\infty} W dt = 0$ and finite energy $\int_{-\infty}^{\infty} |W|^2 dt < \infty$. Usually one prefers an energy value for both scaling and wavelet signals normalized to 1 (i.e., $\int_{-\infty}^{\infty} |W|^2 dt = 1$). For example, the level-1 or level-2 Haar wavelets clearly satisfy these conditions. The averages are:

$$\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}} = 0 \text{ and } \frac{1}{2} + \frac{1}{2} - \frac{1}{2} - \frac{1}{2} = 0 \text{ respectively.}$$

The sum of squares are:

$$\left(\frac{1}{\sqrt{2}}\right)^2 + \left(\frac{1}{\sqrt{2}}\right)^2 = 1 \text{ and } \left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 = 1 \text{ respectively.}$$

Such a normalized value allows one also to interpret the energy function as a probability density function (PDF) for the process represented by the wavelet.

A third condition which is often included relates to the Fourier transform $W(\omega)$ of the wavelet. This condition requires that the following norm must be finite:

$$\int_{-\infty}^{\infty} \frac{|W(\omega)|^2}{|\omega|} d\omega < \infty$$

EXERCISES

21.1 Use Eqs. (21.24) and (21.26) to show that:
   a. the total energy of the Daub4 wavelet equals to one;
   b. the total energy of the Daub4 scaling signal equals to one; and
   c. that the wavelet signal's average is zero.
   (See also Appendix 21.2.)

21.2 Adapt pr21_2.m to perform Multi Resolution Analysis (MRA) using the Daub4 wavelet. Compare the results between the Haar and Daub4 MRA and interpret your result.

References

Mallat, S., 1998. A Wavelet Tour of Signal Processing. Academic Press, San Diego, CA.

Walker, J.S., 1999. A Primer on Wavelets and Their Scientific Applications. Chapman & Hall/CRC, Boca Raton, FL.
An excellent introduction into wavelet analysis including a lot of numerical and practical examples.

CHAPTER

22

Wavelet Analysis: Frequency Domain Properties

22.1 INTRODUCTION

In the discussion of digital filters (Chapter 18), we found that smoothing a signal by applying a window such as $y(n) = [x(n) + x(n-1)]/2$ in the time domain has its equivalent in the frequency domain; in this example the smoothing window has a low-pass filter characteristic. This is precisely what wavelet and scaling signals do, they provide a time domain window of a particular shape which is then translated over the signal and multiplied with the signal values (e.g., in Chapter 21 see Eqs. (21.4) and (21.5)). The spectral composition of scaling and wavelet signals is complementary. This becomes clear when comparing the level-1 Haar scaling signal and the level-1 Haar wavelet:

1. the Haar scaling signal produces the weighted average of the time domain signal which contains the lower-frequency components;
2. the Haar wavelet produces a weighted difference signal containing the higher-frequency fluctuations.

In this example, the scaling signal acts as a low-pass filter and the wavelet acts as a band pass filter which allows the higher-frequency components to persist in the fluctuation signal. As would be expected from our understanding of Fourier analysis, a comparison of different level wavelets demonstrates that smaller-scale wavelets (less dilated) have higher-frequency components than the large-scale ones (more dilated). The scaling signal and wavelet correlations with an input signal are therefore complementary, emphasizing the low- and high-frequency components, respectively.

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00022-9

22.2 THE CONTINUOUS WAVELET TRANSFORM

The frequency domain properties of wavelets can be used to design a filter bank (see also Chapter 18, Section 18.7), where each wavelet at progressively greater scales passes a narrow, lower band of frequencies from input signal x. The time domain procedure of the wavelet transform is to use wavelets w of different scales σ, move (translate) them over an interval τ along an input signal and correlate the wavelet with the input at each of these scales and translations (Fig. 22.1A); for each τ and σ the correlation c is given by:

$$c(\sigma,\tau) = \int_{-\infty}^{\infty} x(t)\frac{1}{\sqrt{\sigma}}w^*\left(\frac{t-\tau}{\sigma}\right)dt \tag{22.1}$$

The * superscript indicates the complex conjugate of the wavelet, in the ***case of a real wavelet (such as the Haar and Daubechies wavelets) the * can be omitted***. Eq. (22.1) represents the so-called continuous wavelet transform (CWT). Contrary to the usual $t + \tau$ (Eq. (13.20), we use $t - \tau$ in Eq. (22.1) to emphasize that we translate the wavelet from left to right (see Appendix 22.1 for more details). The scaling term $\frac{1}{\sqrt{\sigma}}$ is used to preserve signal energy across the transform (compare with Eq. 21.22 where the scale is 2^k). In this section we use the relationship between correlation and convolution in the time and frequency domains; please review Chapter 13 if you need to refresh your memory about these topics. To evaluate the

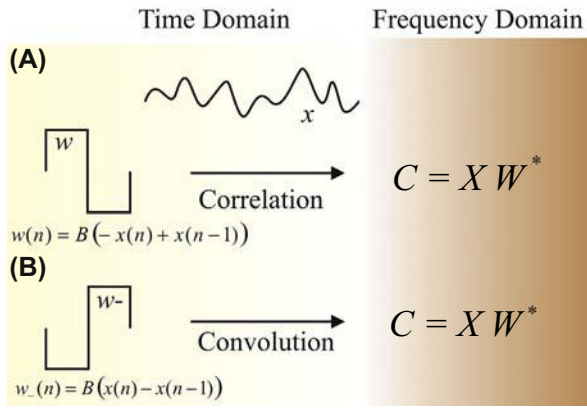


FIGURE 22.1 (A) Correlation of a signal x with a wavelet w is equivalent to convolution of x with the wavelet's reversed version w-, shown in (B). In both cases the frequency domain operation is the product of the Fourier transform of the input X with the complex conjugate of the wavelet W^*.

frequency domain properties associated with the time domain procedure in Eq. (22.1), we define the following Fourier transform pairs:

$$c(\sigma,\tau) \Leftrightarrow C; \quad x(t) \Leftrightarrow X; \quad \frac{1}{\sqrt{\sigma}} w\left(\frac{t}{\sigma}\right) \Leftrightarrow W$$

This allows us to write the equivalent of the correlation in the frequency domain as:

$$C = XW^*, \tag{22.2}$$

where superscript * indicates the complex conjugate (see also Eq. 13.39).

For reasons that will become clear below, it's important to note what happens if we reverse the wavelet signal in the time domain; its frequency representation will change accordingly:

1. for the (cosine) even terms nothing will change, but
2. the (sine) odd terms will change sign (Chapter 5, Appendix 5.2).

If you have difficulty following the above reasoning, review the examples and conclusion in Chapter 5, Section 5.4. Because the sine terms are the complex terms in the Fourier transform, this means that the Fourier transform of the reversed wavelet is the complex conjugate of the Fourier transform of the wavelet, i.e.:

$$\text{if } w^\sigma = \frac{1}{\sqrt{\sigma}} w\left(\frac{t}{\sigma}\right) \Leftrightarrow W \text{ then } w^\sigma_- = \frac{1}{\sqrt{\sigma}} w\left(-\frac{t}{\sigma}\right) \Leftrightarrow W^* \tag{22.3}$$

Note the ***minus sign in the second part of Eq. (22.3)***. From this we may conclude that the correlation of x with a wavelet at a given scale w^σ, is the same as the convolution with the reversed wavelet w^σ_-; we may conclude this because the Fourier transform pair for convolution with the reversed wavelet at scale σ is:

$$x \otimes w^\sigma_- \Leftrightarrow XW^* \tag{22.4}$$

From Eqs. (22.2) and (22.4) it can be concluded that ***convolution of the input x with a reversed wavelet*** w^σ_- results in the identical frequency domain expression as ***correlation c of the signal with the (nonreversed) wavelet*** w^σ, i.e.:

$$\begin{array}{ccc} x \otimes w^\sigma_- & \Leftrightarrow & XW^* = C \\ \| & & \\ c(\sigma,\tau) & \Longleftarrow & \end{array} \tag{22.5}$$

Compared with the procedure for the wavelet transform discussed in Chapter 21, we change the approach slightly by translating the scaling signal (s) and wavelet (w) over the input (x) signal with one-step increments, instead of jumping in steps of two points. We can formalize the Haar scaling signal and wavelet related operations as:

$$s(n) = \frac{1}{\sqrt{2}}(x(n) + x(n-1)), \text{ and} \tag{22.6a}$$

$$w(n) = \frac{1}{\sqrt{2}}(-x(n) + x(n-1)), \text{ respectively.} \tag{22.6b}$$

Here $s(n)$ is the output of the scaling signal procedure and $w(n)$ is the output of the wavelet operation, both of which can be considered FIR/MA filters (see Chapter 18 for this type of filter). Reversing wavelet w^s_- (so that we may use convolution) produces:

$$w_-(n) = \frac{1}{\sqrt{2}}(x(n) - x(n-1)) \tag{22.6c}$$

Since the scaling signal is an even symmetric function, $s_-(n) = s(n)$ and no reversal is necessary. This procedure is illustrated in Fig. 22.2. The scaling signal and reversed wavelet transfer functions in the z-domain (Chapter 12) $H_s(z)$ and $H_{w-}(z)$ are:

$$\begin{aligned} H_s(z) &= \frac{S(z)}{X(z)} = \frac{1}{\sqrt{2}}\left(1 + z^{-1}\right), \text{ and} \\ H_{w-}(z) &= \frac{W_-(z)}{X(z)} = \frac{1}{\sqrt{2}}\left(1 - z^{-1}\right), \text{ respectively.} \end{aligned} \tag{22.7}$$

As with any discrete linear operator, the frequency response can be obtained by substituting $z = \exp(j\omega T)$ with T being the sample interval. Alternatively, we can make this task very simple by using the MATLAB® freqz command to obtain the Bode plots for both the scaling signal and wavelets. We can find the parameters required for this command by writing the signals in the form of Eq. (18.2); for the scaling signal we have the transfer function $\frac{\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}z^{-1}}{1}$. Type in the MATLAB® command window:

```
b=[1/sqrt(2) 1/sqrt(2)];
a=[1];
freqz(b,a)
```

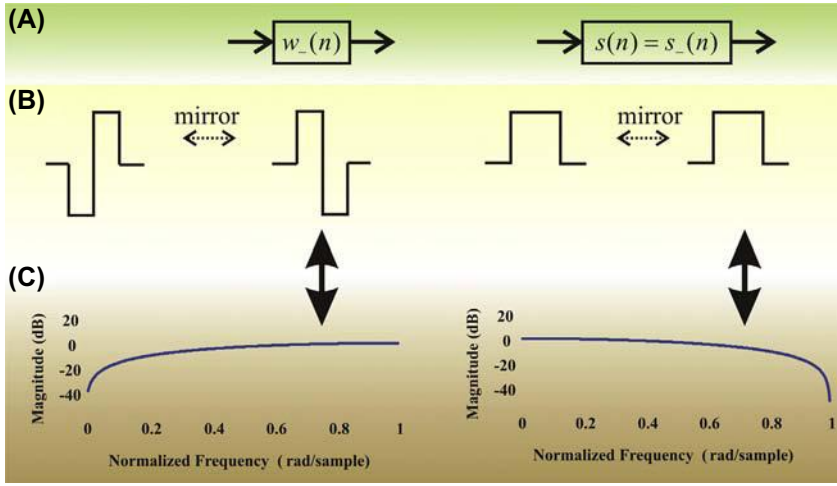


FIGURE 22.2 The time domain transforms using the Haar wavelet (left side) and Haar scaling signal (right side) can be represented as a high-pass and a low-pass filter, respectively. (A) depicts the linear time-invariant (LTI) system representation of the mirrored wavelet and scaling signals shown in (B). The unit impulse responses (UIRs) of these LTI systems are $w\_(n)$ and $s\_(n)$, respectively. Note that, since the Haar scaling signal shows even symmetry, $s\_(n)$ and $s(n)$ are identical. The frequency responses associated with the UIRs of the LTI systems are depicted in (C). The mirror operation in the time domain, shown in panel (B), allows us to convert the correlation procedure (Eq. 22.1) into a convolution. This enables us to consider the wavelet transform as a standard filter operation governed by convolution.

This will show the Bode plot for the Haar level-1 scaling signal, a low-pass filter. In order to evaluate the associated (reversed) wavelet coefficients we type:

```
bb=[1/sqrt(2) -1/sqrt(2)];
aa=[1];
freqz(bb,aa)
```

Note the "−" sign in bb.

This will produce a graph representative of a high-pass filter characteristic.

This specific finding can be generalized; scaling signals have a low-pass filter characteristic, while wavelets pass the higher-frequency components (Fig. 22.2C).

For even symmetric wavelets, the Fourier transform of the wavelet and its reversed version are necessarily identical; therefore the correlation and convolution of an even wavelet with an input signal yield identical results. An example for two scales of the Mexican hat wavelet (MHW) is shown in Fig. 22.3. Here we decompose a signal (Fig. 22.3B) that contains a low- and high-frequency component using the MHW at a large (Fig. 22.3A) and a small scale (Fig. 22.3C). In this example we show the effect of two wavelets only, the procedure where one uses a set of wavelets to filter the signal at different frequency bands is the CWT introduced in Eq. (22.1); examples of CWT are given in Sections 22.3 and 22.4.

22.3 TIME–FREQUENCY RESOLUTION

When performing spectral analysis on a sampled time series, the spectrum reveals frequency components in the input signal. Because the spectrum represents the whole time domain epoch, it is uncertain where exactly any particular frequency component is located in time. To increase resolution, one could reduce the size of the epoch of the input signal. This reduction, however, necessarily changes the resolution of the spectral analysis (Chapter 6, Fig. 6.3). To illustrate the time–frequency resolution of spectral analysis, let's consider a 10-s epoch sampled at 1000 Hz. This choice of parameters results in a spectrum with a resolution of 1/10 Hz up to the Nyquist frequency of 500 Hz. In this example, a spectral peak of a sinusoidal signal with a frequency of 30.06 Hz would be indicated by energy in the transform mainly between 30 and 30.1 Hz. We cannot determine where this frequency component occurs in time because the 30–30.1-Hz component might be present throughout the 10-s epoch, or could be localized in a burst somewhere within the 10-s epoch. We may conclude that the uncertainty of where this particular signal component occurs in time is 10 s and the uncertainty about its frequency value is between 30.0 and 30.1 Hz, i.e., 0.1 Hz. A reduction of the 10-s epoch to a 2-s window would give a 5× more precise (less uncertain) localization of the spectral components in time, because now the spectral components can be located somewhere within a 2-s window. However, this choice of a 2-s epoch results in a ½ Hz frequency domain resolution up to the 500 Hz Nyquist limit. In this case the energy of the 30.06-Hz component would appear in the spectrum mainly between 30 and 30.5 Hz, increasing the uncertainty about the frequency to 0.5 Hz. The above examples indicate that for the Fourier-based spectral analysis:

1. the size of the time domain epoch is proportional to the precision with which spectral components can be located in the time domain, i.e., time domain resolution of any of the spectral components equals the size of the selected epoch;

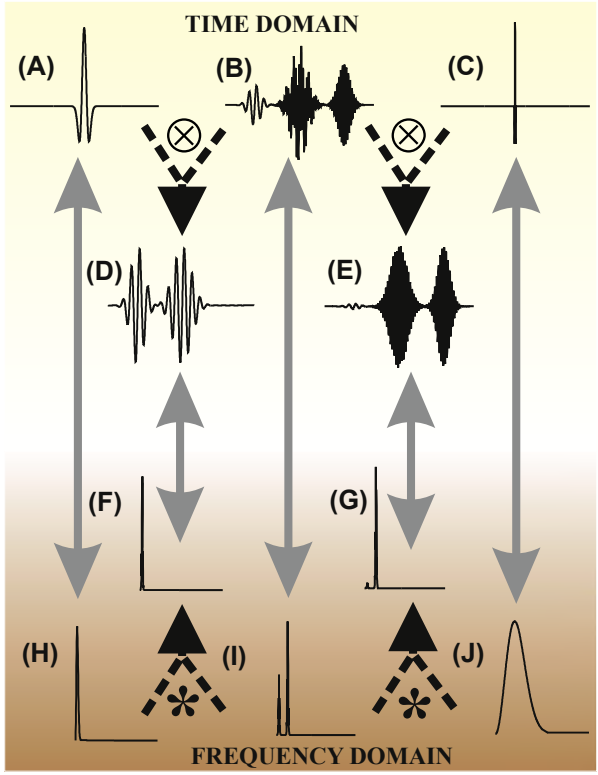


FIGURE 22.3 Example of using wavelet analysis as a filter bank. In this figure we show two scales of the Mexican hat wavelet (MHW), a higher scale in (A) and a lower one in (C). The transform using the higher-scale wavelet with the example signal in (B) generates a lower-frequency component (shown in D), whereas the transform with lower-scale wavelet produces the higher frequencies (shown in E). The above operations in the time domain can also be understood from the equivalent operations in the frequency domain. The frequency domain equivalent of the original signal in (B) is shown in (I). It can be seen in (I) that there are two frequency components. The higher-scale MHW in (A) has a transform containing low frequencies (shown in spectrum H), the transform of the other lower-scale wavelet in (C) is shifted to the higher frequencies (shown in spectrum J). When using the MHW transforms as the filter characteristic, it can be seen that in one case the lower frequencies are predominant (the resulting spectrum in F) and in the other case the higher frequencies predominate (the resulting spectrum in G). The spectra in (F) and (G) correspond with time domain signals (D) and (E), respectively. The gray arrows indicate the Fourier transform pairs. The $\otimes$ and * symbols represent convolution and multiplication, respectively.

Notes:

1. The vertical scaling is optimized in each panel (i.e., not the same between panels). In the frequency domain, the amplitude spectra (not the raw Fourier transforms) are shown.
2. Because the MHW is an even symmetric function, convolution and correlation with the input are equivalent.

2. the size of the time domain epoch is inversely proportional to the resolution in the frequency domain, i.e., the spectral resolution $= 1/\text{epoch}$.

Therefore, any choice of the epoch length is always associated with a compromise between time and frequency resolution; it is impossible to choose an epoch length that will accommodate both a high temporal and a high spectral resolution. A very high temporal resolution (small epoch) is always associated with a low spectral resolution and vice versa. A low frequency must be detected by the choice of a long epoch, which is OK because low-frequency components are spread over longer epochs. However, having made the choice for a longer epoch of perhaps several seconds, the higher-frequency components (such as the 30.06-Hz example above) can now be determined with high precision in the frequency domain but they cannot be precisely localized in time (e.g., the 30.06-Hz component with an intrinsic period of only $\sim$33 ms can only be localized within a 10- or 2-s window in the example above).

Compared to a single Fourier transform-based spectral analysis, continuous wavelet transforms have improved resolution of high-frequency components in the time domain. A low-scale wavelet correlates well with relatively high-frequency components, and the more the scale is stretched (a higher scale) the better the wavelet correlates with the lower-frequency components (Fig. 22.4). As in Eq. (21.22), the wavelet scale σ is often expressed as 2^k, and the frequency associated with a particular wavelet scale is proportional to the inverse of the scale. For every subsequent integer value of k, there is a factor 2 difference in the frequency; i.e., in musical terms there is an octave difference. All non-integer values between k and $k+1$, are steps (voices) within the octave. In some applications the scale is therefore indicated as $2^{n/v}$ with $n = 1, 2, 3, \ldots, N \times v$, v—number of voices, and N—the number of octaves.

An illustration of the uncertainty principle can also be seen in Fig. 22.3, where low- and high-frequency components are distributed in different periods of the input signal epoch (Fig. 22.3B). The power spectrum of the entire epoch (Fig. 22.3I) acknowledges the presence of these spectral components without indicating where in the epoch these occur. In contrast, the MHW transform indicates more precisely (with less uncertainty) where each component is located in time (Fig. 22.3D and E). In empirically measured time series, the spectral components are not known a priori, and the simple approach illustrated in Fig. 22.3 is not possible because one doesn't know in advance what scale(s) of wavelet to select. The solution to this problem is to explore a range of scales, similar to the Fourier transform or a filter bank where sets of frequencies are considered.

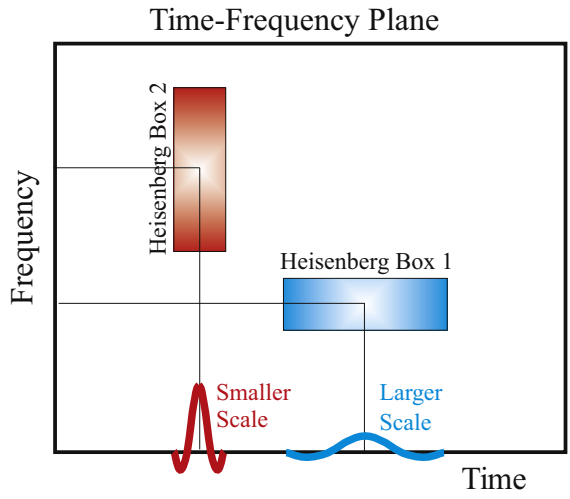


FIGURE 22.4 The time–frequency plane and Heisenberg uncertainty boxes for a low-frequency signal component poorly localized in time but with reasonable spectral resolution (box 1), and for a higher frequency tone with better time resolution but lower spectral resolution (box 2). This figure also demonstrates the features of the scalogram (Section 22.4) in which low-frequency components (longer period) are detected by larger-scale wavelets (blue) and higher-frequency components are detected by smaller-scale wavelets (red). This procedure creates a time resolution which is appropriate for each frequency: i.e., long epochs for slow oscillations and shorter ones for faster oscillations. In contrast, the classical Fourier transform-based spectrum has a fixed Heisenberg box for all frequencies determined by its epoch length (see an example for the 30.06-Hz component in Section 22.3).

In the following MATLAB® example (pr22_1) we use this approach, and calculate a continuous wavelet transform on a signal including three bursts: the first is a low-frequency burst, the last is a high-frequency burst, and the middle one is a combination of both low and high frequencies (same signal as in Fig. 22.3B). The CWT with the MHW shows where in time the energy of these frequencies is located with much greater precision than would be possible using a single Fourier transform.

```
% pr22_1.m
% cwt analysis (continuous wavelet transform) using
% CONVOLUTION and CORRELATION
% using a Mexican hat wavelet (MHW)

clear;
msg=('Pls. wait and MAXIMIZE COLOR PLOTS!')
   N=2048;                  % # of points
maxlag=N/2;                 % here maxlag is used to zoom in on the
                               correct part of C
```

```
C=zeros(128,2*N-1);                    % initialize convolution array
CC=zeros(128,2*maxlag+1);              % initialize correlation array

figure
   % Input signal with m from 0 - 1
for n=1:N;
   m=(n-1)/(N-1);
   tg(n)=m;
   g(n)=sin(40*pi*m)*exp(-100*pi*(m-0.2)^2)+(sin(40*pi*m)
+2*cos(160*pi*m))*exp(-50*pi*(m-0.5)^2)+2*sin(160*pi*m)*
exp(-100*pi*(m-0.8)^2);
end;

% Mexican Hat, a symmetrical real function
w=1/8;                  % NOTE: standard deviation parm w=1/8
index=1;

for k=0:128;            % Use 8 octaves and 16 voices 8 x 16 = 128
   s=2^(-k/16);         % 16 voices per octave
                        % Note that the scale decreases with k
   for n=1:N;
      % Mexican hat from -½ to ½
      m=(n-1)/(N-1)-1/2;            % time parameter
      if (k == 0)
         tmh(n)=m;                  % time axis for the plot
      end;
      mh(n)=2*pi*(1/sqrt(s*w))*(1-2*pi*(m/(s*w))^2)*exp(-pi*(m/
(s*w))^2);
   end;

   if (k == 0)                      % plot wavelet example

      subplot(2,1,1), plot(tmh,mh,'k'); axis('tight')
      ylabel ('Amplitude');
      title(' Mexican Hat');

   end;

   % save the inverted scales
   scale(index)=1/s;

   % Convolution of the wavelet and the signal
   C(index,:)=conv(g,mh);
```

```
    % Correlate the wavelet and the signal
    CC(index,:)=xcorr(g,mh,maxlag);

    index=index+1;

end;

% Plot the results
subplot(2,1,2), plot(tg,g,'r'); axis('tight')
xlabel ('Time ');
ylabel ('Amplitude');
title(' Original Signal containing 20 Hz and 80 Hz components(red)');

figure
pcolor(C(:,maxlag:5:2*N-maxlag).^2);
xlabel ('Time (Sample#/5)');
ylabel ('1/Scale#');
ttl=[' Convolution based Scalogram NOTE: Maxima of the CWT are
around the 1/scale #
            (70) and (38). Ratio = ' num2str(scale(70)/scale(38))];
title(ttl);

figure
pcolor(CC(:,1:5:2*maxlag+1).^2);
xlabel ('Time (Sample#/5)');
ylabel ('1/Scale#');
ttl=[' Correlation based Scalogram NOTE: Maxima of the CWT are
around the scale #
            (70) and (38). Ratio = ' num2str(scale(70)/scale(38))];
title(ttl);
```

Notes:

1. Because the MHW has even symmetry, the same result is obtained when using a cross-correlation between the wavelet and input signal instead of a convolution as in pr 22_1.m above.
2. In order to obtain the correct color display ***MAXIMIZE*** the figures generated by ***pr22_1.m***

22.4 MATLAB® WAVELET EXAMPLES

In MATLAB®, a special wavelet toolbox is available. Here is a short description of the wavemenu command, which launches the graphical user interface for the Wavelet Toolbox. Type: help wavemenu and the following description is displayed:

```
WAVEMENU Start the Wavelet Toolbox graphical user interface tools.
   WAVEMENU launches a menu for accessing the various.
   graphical tools provided in the Wavelet Toolbox.
```

Reprinted with permission of The MathWorks, Inc.

The wavemenu can be used for 1-D and 2-D wavelet transforms (Fig. 22.5). For 2-D analysis the Lena image (available on http://booksite.elsevier.com/9780128104828/, lena_double.mat) can be loaded. A level-2 Haar transform of Lena generates the example shown in Chapter 21 (Fig. 21.6). The eeg.mat file, also available on this website can be loaded to be used for 1-D analysis. The continuous wavelet 1-D generates a so-called scalogram, plotting the frequency components of the signal versus time. You can perform this analysis by selecting Continuous Wavelet 1-D in the menu; this will open a second window that will allow you to Load Signal in the File menu. This will open a dialog box that allows you to load input data such as the eeg.mat file. After loading the data, you can analyze the signal, select the wavelet, the range of scales, and the colormap.

Note: In the discussion of this topic we frequently use the term scale. Please keep in mind that ***scale*** (in the ***scalogram***) is associated with the width (dilation) of the wavelet signal and not with the ***scaling signal*** (*S*) introduced in Eq. (21.2) in Chapter 21.

So-called joint time–frequency analysis (JTFA; Northrop, 2003) is a broad class of techniques that generate representations of the spectral components from a time series. The most commonly applied procedure depicts the power of a set of frequency bands against time. In principle, these plots can be generated by displaying the graphs of spectral energy in a waterfall format, or by displaying spectral energy in a color-coded fashion (Fig. 22.6). Depending on whether the spectral parameters are determined with standard Fourier analysis or with a wavelet approach, the plot is called a ***spectrogram*** or ***scalogram***, respectively; an example of

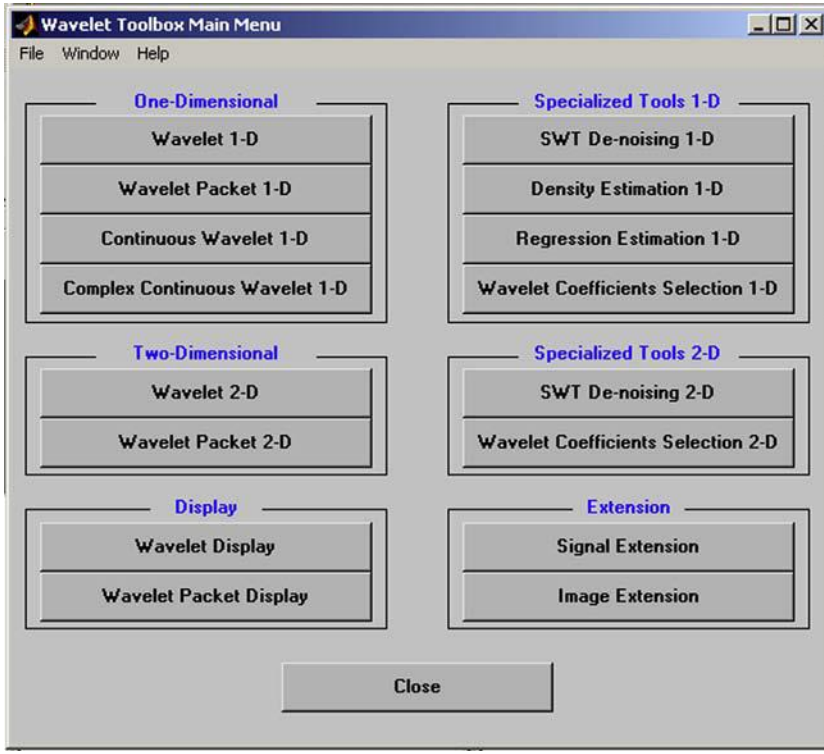


FIGURE 22.5 The MATLAB® menu for wavelet analysis. Figs. 22.6B and 21.6 and an multiresolution analysis, such as the one shown in Fig. 21.4, are a few examples of wavelet analyses that can be accomplished with the menu of this powerful toolbox. *Reprinted with permission of The MathWorks, Inc.*

both techniques applied to an electroencephalogram (EEG) epoch representing the onset of an epileptic seizure is shown in Fig. 22.6. The input signal is clearly nonstationary: the seizure onset presents a drastic transition in the character of the EEG that becomes clearly visible both in the spectrogram (Fig. 22.6A) and the scalogram (Fig. 22.6B).

In this example, the scalogram in Fig. 22.6B can be compared with the spectrogram in Fig. 22.6A. The spectrogram was created by applying a series of windowed Fourier transforms. Each transform is used to generate a power spectrum (or an amplitude spectrum) that then can be color-coded. Using this procedure, each spectrum is represented as a colored vertical bar. These bars then concatenated along a horizontal time scale. The scalogram in Fig. 22.6B is produced in the same way as in pr22_1.m. For each scale of the wavelet, a convolution between signal and wavelet is determined. This generates a filtered trace of the input signal.

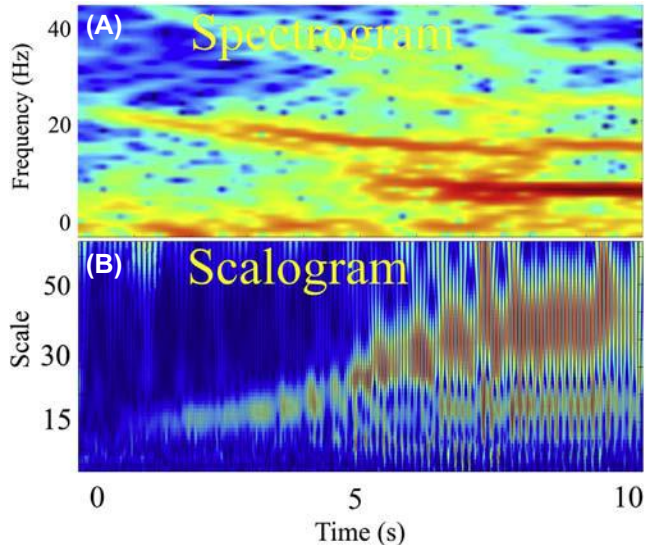


FIGURE 22.6 A spectrogram (A) and scalogram (B) of an electroencephalogram signal during the onset of an epileptic seizure.

For subsequent scales the filtered traces are stacked and the amplitude values in the matrix of stacked traces mapped onto a color code (see MATLAB® command pcolor).

By comparing the spectrogram and scalogram in Fig. 22.6 it can be seen that the time resolution of the scalogram is superior. Especially at the lower scales (corresponding to higher frequency) the contours are well defined in time, at higher scales spectral components are less well defined because they correspond with lower frequencies. In the spectrogram all frequency components are equally blurred in time, due to the uncertainty created by the epoch length (Section 22.3).

APPENDIX 22.1

Eq. (22.1) shows an unusual notation for the correlation. Here $t - \tau$ is used instead of $t + \tau$, and for the variable w its complex conjugate w^* is used. We repeat Eq. (22.1) for convenience

$$C(\sigma, \tau) = \int_{-\infty}^{\infty} x(t) \, \frac{1}{\sqrt{\sigma}} w^* \left(\frac{t - \tau}{\sigma} \right) dt$$

It is straightforward to show that this can be written in the same form as the correlation function we discussed earlier in Chapter 13 (Eq. 13.20) by substituting $T = t - \tau$ (therefore $t = T + \tau$ and $dt = dT$). We get

$$C(\sigma, \tau) = \int_{-\infty}^{\infty} x(T + \tau) \frac{1}{\sqrt{\sigma}} w^*\left(\frac{T}{\sigma}\right) dT \qquad \text{(A22.1-1)}$$

Eq. (A22.1-1) is an expression that is equivalent to Eq. (13.20).

Now we investigate the rationale for the use of the complex conjugate w^* in Eq. (22.1). Of course this is only relevant if wavelet w is complex, but it is still somewhat surprising. To investigate this, we transform C into the frequency domain.

$$C(\sigma, j\omega) = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} x(T + \tau) \frac{1}{\sqrt{\sigma}} w^*\left(\frac{T}{\sigma}\right) dT \right] e^{-j\omega\tau} \, d\tau \qquad \text{(A22.1-2)}$$

Now we interchange the integration operations and rearrange the expression (as we did in Section 13.4.2), and we get

$$C(\sigma, j\omega) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{\sigma}} w^*\left(\frac{T}{\sigma}\right) \left[\int_{-\infty}^{\infty} x(T + \tau) \, e^{-j\omega\tau} d\tau \right] dT \qquad \text{(A22.1-3)}$$

Subsequently we substitute $t = T + \tau$ (and $\tau = t - T$, $d\tau = dt$) in the term in between the […]. This term now becomes

$$\int_{-\infty}^{\infty} x(t) \, e^{-j\omega(t-T)} dt = e^{j\omega T} \int_{-\infty}^{\infty} x(t) \, e^{-j\omega t} dt = e^{j\omega T} X(j\omega) \qquad \text{(A22.1-4)}$$

When we substitute this result into Eq. (A22.1-3), we get

$$C(\sigma, j\omega) = X(j\omega) \int_{-\infty}^{\infty} \frac{1}{\sqrt{\sigma}} w^*\left(\frac{T}{\sigma}\right) e^{j\omega T} dT \qquad \text{(A22.1-5)}$$

Recall that the Fourier transform of the wavelet is defined as $\int_{-\infty}^{\infty} \frac{1}{\sqrt{\sigma}} w\left(\frac{T}{\sigma}\right) e^{-j\omega T} dT$. As you can see, since the sign of the imaginary terms is opposite in the integral in Eq. (A22.1-5), the integral represents the complex conjugate of the Fourier transform of the wavelet: $W(j\omega)^*$. Of course, if w is complex, its imaginary component must also have a

reversed sign for this to be true. This brings us back to the rationale for the w^* term in Eq. (22.1). In summary, this allows us to write the Fourier transform pair as

$$C(\sigma,\tau) \Leftrightarrow C(\sigma, j\omega) = X(j\omega)\, W(j\omega)^* \qquad \text{(A22.1-6)}$$

EXERCISES

22.1 Analyze mathematically why in the MHW example of a CWT it doesn't make a difference using either convolution or cross-correlation. See also MATLAB® script pr22_1.m.Would it make a difference for the Haar wavelet?

22.2 Analyze the files ActionPotential.mat and AlphaRhythm_5 seconds.mat with the MATLAB® wavemenu, using the Haar and the Daubechies4 wavelets, to:
a. perform an multiresolution analysis and
b. produce a scalogram
c. Interpret your findings.
(The files are available on http://booksite.elsevier.com/9780128104828/; use the 1-D Wavelet and Continuous Wavelet submenus.)

22.3 Analysis of the Daubechies wavelet.
a. Determine the z-transform for the Daubechies wavelet using the coefficients in Eq. (21.6), Chapter 21.
b. Use freqz to determine its frequency response

22.4 Show that the MHW is an inverted second derivative of a Gaussian (DOG) by:
a. Given that the Gaussian is $\frac{1}{\sigma\sqrt{2\pi}}\, e^{-t^2/2\sigma^2}$ show that the inverted second derivative is:

$$\Psi_{mh} = -d^2\left(\frac{1}{\sigma\sqrt{2\pi}}e^{-t^2/2\sigma^2}\right)\Big/ dt^2 = -\frac{1}{\sigma^3\sqrt{2\pi}}\left(t^2/\sigma^2 - 1\right)e^{-t^2/2\sigma^2} \qquad (1)$$

b. Show that MHW (Ψ_{mh}) satisfies the condition: $\int_{-\infty}^{\infty} \Psi_{mh} dt = 0$
c. Pretend you are one of these nerdy types and you want to normalize the MHW by applying the norm (N).
i. First show that $N = \int_{-\infty}^{\infty} \Psi_{mh}^2 dt < \infty$

ii. Then calculate the value for N, and use it to normalize the equation of the second derivative in Eq. (1) above. Such that the normalized MHW (Ψ_{mhN}) satisfies:

$$\int_{-\infty}^{\infty} \Psi_{mhN}^2 dt = 1, \text{ giving a normalized MHW:}$$

$$\frac{2}{\pi^{1/4}\sqrt{3\sigma}}\left(t^2/\sigma^2 - 1\right)e^{-t^2/2\sigma^2}$$

22.5 Use the expression for the normalized MHW: $\frac{2}{\pi^{1/4}\sqrt{3\sigma}}\left(t^2/\sigma^2 - 1\right)e^{-t^2/2\sigma^2}$
Determine its frequency response
(*Note*: for this problem, you may use online tables for transforms.)
a. analytically, and
b. numerically (using Matlab), and
c. plot the Bode plot for both results in (a) and (b).
d. Interpret your results.

22.6 Use MATLAB® to create a signal of a 10-Hz sinusoidal waveform with Gaussian white noise (noise characteristics: zero mean and unity variance). Use an epoch of 1 s and a sample rate of 1000 samples/s.
a. Analyze and plot this waveform using a level-3 Daubechies4 wavelet transform (you may use wavemenu in MATLAB®).
b. Interpret your result.

22.7 Use MATLAB® to compare and analyze the similarities and differences of the spectrogram and scalogram for the file eeg.mat (contains the variable eegf).

To produce the spectrogram:

```
load eeg
figure
spectrogram(eegf,250,0,128,250,'yaxis')
```

Note the parameters in spectrogram
input: eegf
window: 250 points
overlap: 0 points
fft window: 128 points
sample rate: 250 samples/s
'yaxis': time is on the x-axis and frequency along ordinate

To produce the scalogram:
Open wavemenu
Select Continuous Wavelet 1-D
Load the file eeg.mat
Select the db-4 wavelet and
Set the scale to 256
Set Color Map to hot.

Reference

Northrop, R.B., 2003. Signals and Systems Analysis in Biomedical Engineering. CRC Press, Boca Raton, FL.
An excellent introduction to signal processing and linear systems including a review of basic techniques for solving ODEs, matrix algebra, and transformations. Some examples of nonlinear systems analysis are included.

CHAPTER

23

Low-Dimensional Nonlinear Dynamics: Fixed Points, Limit Cycles, and Bifurcations

23.1 INTRODUCTION

In a linear system, responses are proportional to its input: e.g., when a system's input is doubled, so is its output. This simple scaling property is no longer valid when nonlinear dynamics is involved (Chapters 13 and 24). The nonlinearity may lead to strong disproportional effects, and may also cause counterintuitive changes in the behavior of a system. Another very important property is the existence of sudden qualitative changes in the dynamical behavior due to a parameter change of a nonlinear system. These sudden changes are known as bifurcations. A well-known property in neuroscience that can be modeled with a bifurcation is neuronal excitability: i.e., the sudden transition of the neuron from its resting/subthreshold state to firing action potentials. In this chapter we introduce the basics of bifurcations and illustrate their importance with examples based on neural excitability. Here, we employ parametric models of low-dimensional nonlinear systems to introduce the characteristic behavior of this type of system. The use of nonparametric models to describe and examine nonlinear systems and their signals is reviewed in Chapters 24–27. For excellent introductions into nonlinear dynamics in general, see Strogatz (1994) or Kaplan and Glass (1995), and for chaos, see Peitgen et al. (1992). For an overview of the applications of nonlinear dynamics and bifurcation theory to neuronal systems, see Izhikevich (2007).

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00023-0

23.2 NONLINEAR DYNAMICS IN CONTINUOUS AND DISCRETE TIME

One of the principal considerations for the analysis of nonlinear deterministic systems is whether they evolve in continuous time or discrete time. Although continuous time and discrete time dynamics share many properties, there are essential differences we will discuss in the following.

In **continuous time**, changes across time and state of the system are also continuous. Although abrupt changes in a system may occur, they still happen continuously (albeit very fast) in state space. For a 1-D deterministic system, its dynamics can be represented on a line. To envision the dynamics of such a system it is helpful to introduce the concept of a **phase point**. If we represent a system's dynamics by a fluid that flows in 1-D space (i.e., a line), a phase point can be imagined as a particle that is carried along by the flow. If we now represent the state of the 1-D system by such a phase point on a line, we find a rather limited repertoire of possibilities: the point either remains where it is, it moves to the right, or moves to the left (Fig. 23.1A). The point where the system's state does not change is called a **fixed point**. It is important to note that an oscillation on a straight phase line is not possible because it would violate the deterministic property that the state of a system unambiguously determines how it evolves. An oscillation would require an ambiguous evolution since a phase point would have to move right on one occasion and left on another (Fig. 23.1B).

One special solution to make oscillation possible in a 1-D model is to represent the phase of an oscillation on a circle (Fig. 23.1C). In this scenario one can model the phase of an oscillation using the position of a phase point on a circle. This creates a situation without any ambiguity since every point on the circle is associated with a unique rule for updating. However, since there is only one (circular) dimension, one can model the phase of the oscillation but one cannot also account for its amplitude. In order to model both the phase and amplitude of an oscillation one needs at least two dimensions, i.e., a phase plane (Fig. 23.1D). Now, a phase point can move in the phase plane following a cyclic motion for which both phase and amplitude can be accounted for—in this case we have a so-called **limit cycle**.

Also, the dynamics in the plane has its limitations because here the trajectories are not allowed to cross, as that would again create ambiguity with respect to the deterministic evolution of the system's state (Fig. 23.1E). More complex dynamics than rest, movement along a straight or curved line, and oscillation requires three or more dimensions. One example is the **chaotic** Lorentz attractor that requires a 3-D phase space—although it is depicted in a 2-D projection in Fig. 23.1F.

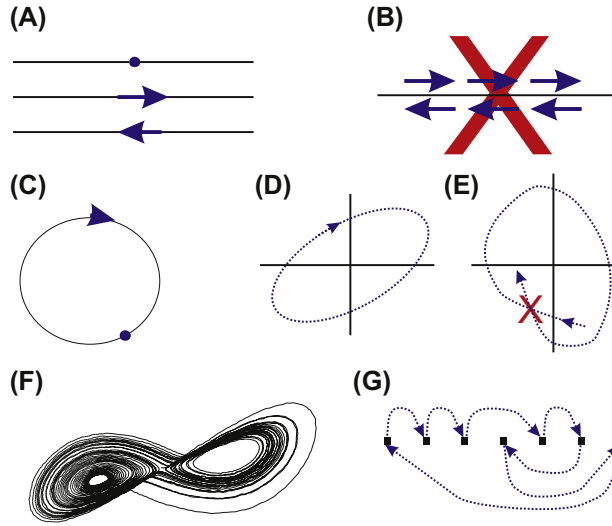


FIGURE 23.1 Overview of low-dimensional nonlinear dynamics. (A) On a line (1-D), a system can remain where it is (top), move to the right (middle), or left (bottom). (B) Oscillations are not possible in a continuous time 1-D case because it would violate determinism—i.e., a phase point at a given location on the line cannot move right on one occasion and move left at another time. (C) Oscillations in 1-D are possible on a *circle* without violating determinism. Modeling oscillation on a circle is limited because it can only account for change in phase and not amplitude. (D) Dynamics in 2-D can model oscillations while accounting for change in both phase and amplitude. (E) However, also in 2-D there are limitations: paths cannot cross because that would violate determinism. (F) In 3-D dynamics richer behavior is possible—here we show a 2-D projection of the 3-D trajectory of the chaotic Lorentz attractor. (G) While all dynamics depicted in panels (A)–(F) show behavior and limitations in continuous time, a nonlinear system evolving in discrete time may behave differently. Due to the discrete time, a 1-D system can exhibit oscillatory and even chaotic behavior. Here we show an oscillation involving seven discrete steps.

In **discrete time**, dynamics may evolve in a different fashion. This is the case because one may jump in the discrete phase space without violating the rule of determinism that a system's state is unambiguously determined by its past. For example, in the case of 1-D dynamics one has a discrete rather than a continuous phase line (Fig. 23.1G). Since the movement in state space (i.e., the line) is discrete, one can create oscillatory or even chaotic behavior without violating deterministic update rules. As you can see in the example in Fig. 23.1G, the state of the system can follow a complicated periodic pattern without any ambiguity in the evolution of the system's state. As we will discuss and show below, a simple 1-D discrete time version of the logistic equation can produce oscillatory behavior and even chaotic time series.

While continuous time dynamics is modeled with ordinary differential equations (ODEs; Chapters 9 and 10), discrete time dynamics is described by so-called difference equations. If it isn't trivial or even possible to solve the equations, one may employ graphical methods to investigate discrete time systems that are governed by a known difference equation. In this case we can use the so-called cobweb method to obtain insight into the dynamics. In the following we apply this method to determine the behavior of a system governed by the logistic equation $x_{i+1} = a\left(x_i - x_i^2\right)$.

Two examples of a cobweb presentation of the dynamics (for values of $a = 2.5$ and $a = 3.25$) are depicted in Fig. 23.2. The cobweb procedure starts with picking an initial condition (a value of 0.2 in Fig. 23.2). One then reads how this initial condition evolves to the next state using the graph using the vertical line labeled "1" in Fig. 23.2. This new value now determines the subsequent state following the rule determined by the plotted function, i.e., $x_{i+1} = a\left(x_i - x_i^2\right)$. In order to proceed, the new value is then positioned parallel to the abscissa by using the line $x_{i+1} = x_i$ and evolved to the next state, as indicated by the horizontal and vertical lines indicated by "2" and "3" in Fig. 23.2. This process is then repeated to find the next state again—in the example in Fig. 23.2A, the next step, indicated by "4", leads to a stable fixed point located on the line $x_{i+1} = x_i$. Following exactly the same procedure for a different parameter value for a, we find a stable oscillation (Fig. 23.2B). Cases requiring many iterations may create a web-like picture, hence the name cobweb method. Chaotic behavior governed by the logistic equation is discussed in Chapter 27.

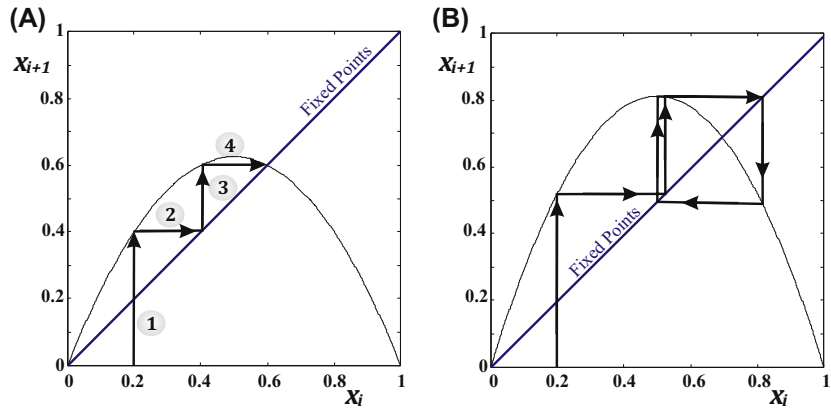


FIGURE 23.2 Illustration of the cobweb method to investigate the dynamics governed by a difference equation. In this example we used the logistic equation $x_{i+1} = a\left(x_i - x_i^2\right)$, with two different parameters. (A) An example for $a = 2.5$ leads to a stable fixed point, and in (B) for $a = 3.25$, we find a stable oscillation. See also Chapter 27, Fig. 27.1A and B for a plot of the numerical solutions for these scenarios.

As depicted in Fig. 23.1, oscillatory behavior can occur in ≥1-D dynamics in discrete time and in ≥2-D dynamics in continuous time. In the remainder of this chapter we will focus on systems that evolve in continuous time. Conditions for the presence of oscillatory behavior in a linear case were discussed and analyzed in Chapters 9 and 10. There it is shown that an imaginary solution to the second-order ODE results in an oscillation, also called a center (e.g., Fig. 10.2C). The essence of a center is that any perturbation that causes change in the center's radius will be either preserved for all time, or until another perturbation changes the radius again. In contrast, oscillation produced by a nonlinear system is called a **limit cycle**. Three types of limit cycle exist: stable, unstable, and half-stable. As a consequence, there is a critical difference between oscillatory behavior in linear systems and nonlinear systems. While a perturbation of the oscillation in a linear system results in a "new" center, the same perturbation in the nonlinear case either causes a drift back to the same limit cycle (the stable case), or it destroys the limit cycle behavior (the unstable case). Just as for fixed points, there is also a half-stable version of the limit cycle where trajectories in phase space move towards the limit cycle at one side of the cycle and away from it at the other side. Note that in a noisy (i.e., real) environment, limit cycle trajectories that are either unstable or half-stable will not be observed since a phase point will be perturbed so that it will move away from the unstable limit cycle trajectory, or ends up at the unstable side of a half-stable limit cycle with the same result. In addition, it should be noted that in real physical systems, an unstable limit cycle is usually surrounded by a stable one that ensures bounded behavior of the system.

Another point of interest is that there is always at least one fixed point within the area enclosed by the limit cycle's trajectory. That fixed point is stable if the limit cycle is unstable, and the fixed point is unstable if the limit cycle is stable. The presence of such a fixed point can be intuited as follows:

1. Trajectories must originate from somewhere if they approach the stable limit cycle from its inside, and an unstable fixed point acts like such a point of origin (also called the source).
2. In contrast, trajectories departing towards the inside of an unstable limit cycle cannot walk around infinitely in the limited space bounded by the limit cycle. Therefore they must end somewhere, and a stable fixed point can serve as such an end point (also called the sink).

As can be seen in Fig. 23.1 there is a fundamental difference between the conditions for the existence of **chaos** in continuous time and discrete time systems. In continuous time one needs at least a 3-D system, such as for the Lorentz attractor shown in Fig. 23.1F. Discrete time systems are not

subject to the same limitation. Paths can move back and forth on a discrete line (Fig. 23.1G), and also chaos can occur in a dynamical system governed by a 1-D nonlinear difference equation. Examples of existence chaos and the analysis of this type of behavior are further discussed in Chapter 27.

23.3 EFFECTS OF PARAMETER SELECTION ON LINEAR AND NONLINEAR SYSTEMS

One critical difference between linear and nonlinear systems is the occurrence of sudden changes in the behavior resulting from a change of its parameters. These sudden changes in nonlinear systems are the so-called **bifurcations**. For one-dimensional dynamics, the mechanism underlying a bifurcation can be envisioned in a combined plot of phase space and the associated dynamics. For example, for the linear case $\dot{S} = A - S$ and the nonlinear case $\dot{S} = A - S^2$, we can plot $\dot{S}$ as a function of S and obtain the so-called **phase portrait** (Fig. 23.3A–F). The change of the dynamics on the phase line at a given location of S, can be seen in the phase portrait, as described above. This change can be visualized by putting a phase point (our imaginary particle) on the phase line. For example, a phase point at the location $S = 50$ in Fig. 23.3A will move to the left until it reaches the fixed point at $S = 25$.

In both panels A and B in Fig. 23.3, we used parameter $A = 25$. Let us now change the value of parameter A in both the linear and nonlinear case (Fig. 23.3C–F). In the linear case the location of the fixed point changes, but the dynamics remains qualitatively the same: i.e., there is still a stable fixed point on the phase line. This fixed point is stable because small perturbations around the fixed point will drive the phase point back to the fixed point. The nonlinear example has two fixed points (Fig. 23.3B). The right fixed point is stable just as the one in Fig. 23.3A, while the left fixed point in Fig. 23.3B is unstable. At the unstable fixed point, a small perturbation will drive the phase point away from the fixed point.

In contrast to the example for the linear case, the change of the dynamics as a result of a parameter change is different for the nonlinear system. The two fixed points that were present in Fig. 23.3B are reduced to a single fixed point in Fig. 23.3D (where it is half-stable), and disappear in Fig. 23.3F! In this example, we changed the system's parameter, and as a result the system's behavior was changed qualitatively. In this case the system has undergone a **bifurcation** due to a change of its parameter A. Looking at the examples in Fig. 23.3, it is clear why one needs the nonlinear property to create such a bifurcation: when the relationship between S and $\dot{S}$ is a straight line, there will always be an intersection with the abscissa where $\dot{S} = 0$, i.e., a fixed point (Fig. 23.3A and C). One could

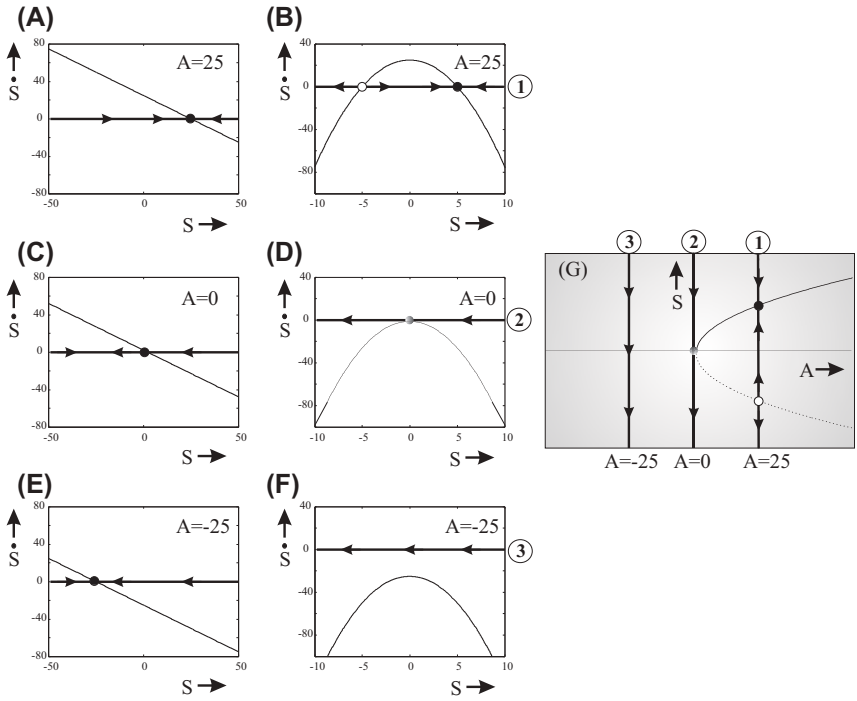


FIGURE 23.3 Phase space representation of the parameter dependent dynamics governed by a linear and nonlinear ODE. (A) $\dot{S} = A - S$ for $A = 25$ and (B) $\dot{S} = A - S^2$ for $A = 25$. (C) and (D) are the same ODEs, but for $A = 0$. (E) and (F) are the same ODEs, but for $A = -25$. The arrows on the phase line indicate the direction in which a phase point at that location would move. Stable fixed points are indicated by *solid dots*, unstable ones by a *circle*, and the half stable one in panel (D) with a half solid/half *open dot*. It can be seen that a change of A in the linear case affects the position of the stable fixed point (panels A, C, and E). In contrast, a change of A in the nonlinear case has a more dramatic effect: it determines the location of the fixed points but also the number of fixed points (panels B, D, and F). (G) The bifurcation diagram for the nonlinear case shown in panels (B), (D), and (F). This so-called bifurcation diagram is obtained by combining the dynamics on the phase line as a function of the value of the parameter A. In this example, the phase lines from panels (B), (D), and (F), denoted by 1, 2, and 3 are shown. The stable fixed points in the bifurcation diagram are located on the solid line while the unstable ones are on the dotted line. The location at $A = 0$, where the dynamics changes, is the so-called bifurcation point.

point out that the exception would be when the line moved to an orientation precisely parallel to the phase line; in that case there would not be an intersection, i.e., there is no fixed point. This observation is correct, but this would indicate that $\dot{S}$ is a constant and no longer a function of S. In contrast, if the relationship between S and $\dot{S}$ is nonlinear, the value of A and thus the position of the curve determines if there are intersections between that curve and the abscissa where $\dot{S} = 0$ (Fig. 23.3B and D).

If we determine the behavior of the nonlinear system for a range of parameter values, we can construct a so-called **bifurcation diagram** (Fig. 23.3G). In this diagram the parameter A becomes the independent variable, plotted on the abscissa, while the behavior on the S line is now the dependent variable, plotted on the ordinate. The stable fixed points on the bifurcation diagram are represented by a solid line, while the unstable ones are connected by a dotted line. The phase lines and arrows showing the dynamics as depicted in the example in Fig. 23.3G are usually omitted in bifurcation diagrams.

Although there is a wide variety of possible scenarios that can generate a bifurcation, the number types of bifurcation one can generate on a phase line by changing a single parameter (called the codimension) is limited. That is if one reduces the bifurcations to their prototypical/canonical form—these canonical forms are depicted in Fig. 23.4. In 1-D dynamical systems, we may observe three types of bifurcation: the saddle-node bifurcation, the transcritical bifurcation, and the pitchfork bifurcation. There are two subtypes of the pitchfork bifurcation: super- and subcritical.

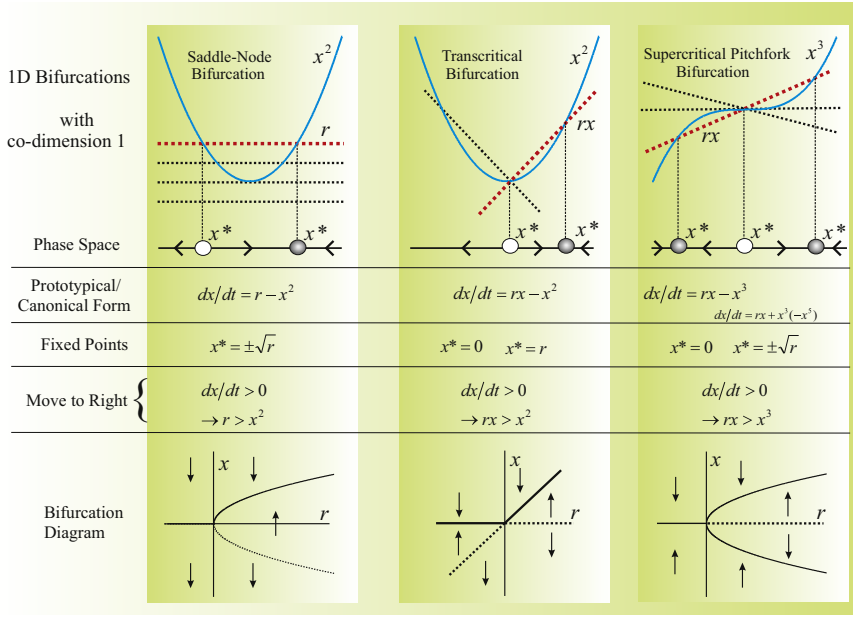


FIGURE 23.4 Overview of types of codimension-1 bifurcation on a phase line. (Left) The saddle-node bifurcation (also called fold, turning-point, or blue sky bifurcation). (Middle) The transcritical bifurcation. (Right) The supercritical pitchfork bifurcation (the equation for the subcritical pitchfork is shown in small font).

To illustrate how equations that differ from the expression in Fig. 23.4 can still show the same behavior, i.e., can be presented in their canonical form, we consider the dynamics governed by $\dot{x} = 1 + rx + x^2$. This expression is not the same as the equation we use for the saddle-node bifurcation in Fig. 23.4. Nonetheless we will now show that $\dot{x} = 1 + rx + x^2$ can exhibit saddle-node bifurcations. To facilitate the analysis, we separate the right-hand part of the expression into two parts: rx and $1 + x^2$. Next, we establish that fixed points (at $\dot{x} = 0$) must occur when $1 + x^2 = -rx$. Our phase point will move to the right when $\dot{x} > 0$, corresponding to $1 + x^2 > -rx$, and to the left when $\dot{x} < 0$, corresponding to $1 + x^2 < -rx$. To visualize this, we plotted both $1 + x^2$ and $-rx$ in Fig. 23.5A, and indicated the associated movements of the phase point on the x-axis. In the three panels in Fig. 23.5A, we summarize the dynamics for six values of r: -3, -2, -1, 1, 2, and 3. It is clear that with a change of both positive or negative values of the parameter r, the line $-rx$ rotates. While the line rotates, two fixed points collide and subsequently vanish—precisely the behavior we observed in the saddle-node bifurcation in Figs. 23.3 and 23.4. The bifurcation diagram reflecting this behavior is depicted in Fig. 23.5B.

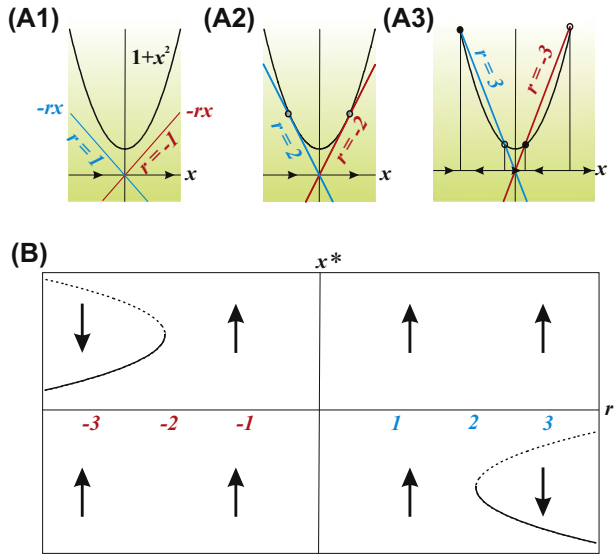


FIGURE 23.5 Saddle-node bifurcations in $\dot{x} = 1 + rx + x^2$. Panels in (A) plot two parts of the expression: $-rx$ and $1 + x^2$. The values of r vary across these panels. Fixed points (at $\dot{x} = 0$) must occur when $1 + x^2 = -rx$. *Arrows* on the x-axis depict the movement of the phase point. It moves to the right when $\dot{x} > 0$, corresponding to $1 + x^2 > -rx$, and to the left when $\dot{x} < 0$, corresponding to $1 + x^2 < -rx$. The bifurcation diagram is shown in (B).

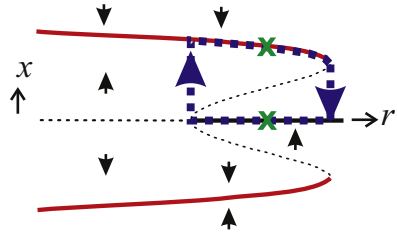


FIGURE 23.6 Bifurcation diagram of a subcritical pitchfork bifurcation. In this case there is a fifth-order term to ensure that there is a stable branch (indicated in red) if the system escapes the unstable equilibrium (*dashed black line*) of the pitchfork. This may lead to complicated behavior, so-called hysteresis, as outlined by the *blue dotted* trajectory. Accordingly, the system is multistable, i.e., a particular parameter value of r can be associated with different states (e.g., the green x's).

The subcritical pitchfork bifurcation is not explicitly shown in Fig. 23.4. Although one only needs a third-order nonlinear term to create this bifurcation, there is usually a fifth-order term added to the equation providing stability around the instable pitchfork branches. This effect of the fifth-order term is depicted as the red stable line in Fig. 23.6. The presence of such a fifth-order term leads to **hysteresis**. This is caused by a range of parameters (r in Fig. 23.6) where multiple stable states occur. Moving the values of r back and forth now leads to the behavior shown by the blue line in Fig. 23.6: i.e., depending on the history a particular value of r may result in different states (green x in Fig. 23.6)!

The presence of hysteresis in an experimental setting can be rather confusing because one may observe different responses to the same stimulus! In such a case it is important to control the state of the system before applying stimulation.

23.4 APPLICATION TO MODELING NEURAL EXCITABILITY

In the following we will employ low-dimensional dynamics to explore behavior in model neurons (see Chapters 29 and 30 for more details on modeling neuronal activity). Although 1-D dynamics is limited for modeling neuronal activity, with a few ad hoc additions it is feasible. Examples are described in the following.

23.4.1 One-Dimensional Dynamics: Excitability on a Line

Neuronal activity can be mimicked by one-dimensional models or their electronic equivalent in which the neuronal membrane is represented by its capacitance (C) and its resistance (R) (Fig. 1.1 and 23.7A). The only

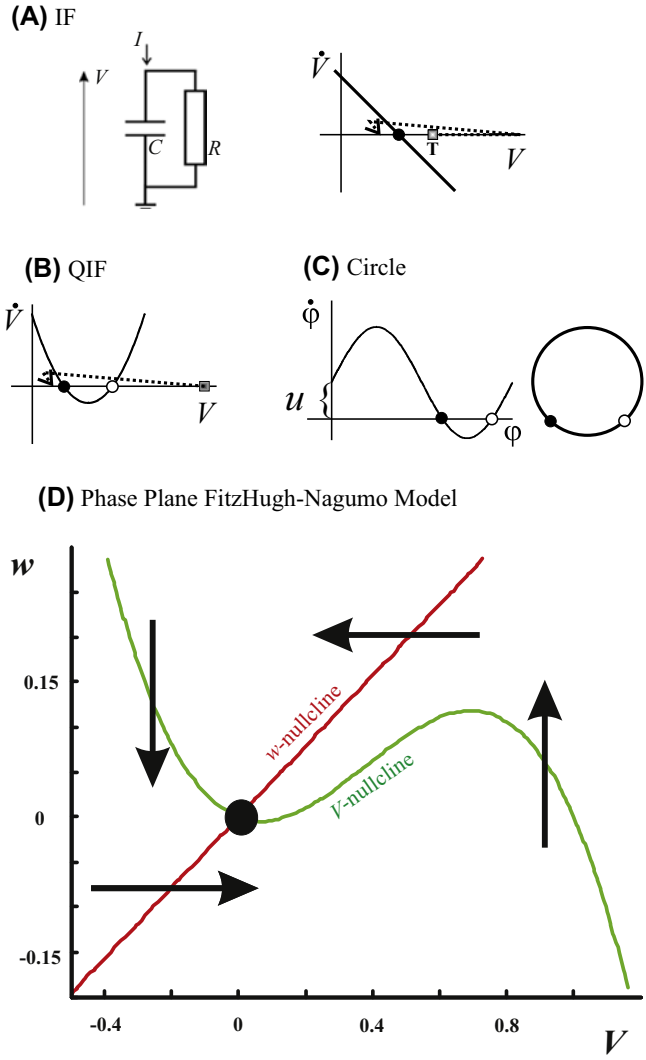


FIGURE 23.7 Neuronal dynamics modeled in one dimension. (A) Electronic equivalent circuit modeling the subthreshold behavior of an IF neuron: V represents the membrane potential and I the membrane current; resistor R and capacitor C model the membrane impedance. The graph is the phase portrait of the linear IF model, showing a stable fixed point (*black dot*) at the resting potential and a predetermined threshold (T). When this threshold is exceeded, an ad hoc spike (*dashed line*) is pasted into the IF's trajectory. (B) The QIF formalism is characterized by a stable fixed point (*black dot*) at the resting potential and an unstable fixed point (*circle*). The unstable fixed point acts as the threshold. When this threshold is exceeded, the membrane potential V will increase. A predetermined value of V serves as the maximum spike amplitude and if this value is reached, an ad hoc reset is performed (*dashed line*). (C) An example of 1-D dynamics on a *circle* that models the dynamics of phase φ governed by Eq. (23.5). The graph on the left depicts the phase portrait for the dynamics on the *circle* on the right part of panel (C). Stable and unstable fixed points are indicated by a black dot and open circle, respectively. (D) The phase plane (V−w) representation of the FitzHugh−Nagumo model for zero current injection. The nullclines for V (green) and w (red) are shown. For this set of parameters, their intersection creates a stable fixed point (Appendix 23.1). The *arrows* schematically indicate the flow in the plane.

adaptation to add excitable neuron-like behavior is to include an action potential (spike) generator. This mechanism creates a spike when a predefined threshold is reached. Such an RC-membrane model with spike generation is linear below the firing threshold. At the threshold the membrane potential is reset and (for visual reasons) a spike signal is pasted in the signal (the $V - \dot{V}$ plot in Fig. 23.7A). This type of model is known as the **Integrate-and-Fire neuron (IF-neuron)**, first described by Lapicque (1907). The circuit in Fig. 23.7A allows us to formulate the equations governing the subthreshold behavior (see Appendix 1.1 for a summary of equations governing electrical circuits). Injected current (I) splits into a part through C (I_C) and one through R (I_R):

$$I = I_C + I_R \tag{23.1}$$

Next, we use Ohm's law $I_R = \frac{V}{R}$ and the equation relating current and potential in the capacitor $I_C = C\frac{dV}{dt} = C\dot{V}$, substituting these into Eq. (23.1) we get

$$I = C\dot{V} + \frac{V}{R} \tag{23.2}$$

If we set both R and C to 1 we can simplify to:

$$\dot{V} = I - V \tag{23.3}$$

Eq. (23.2) can be used to determine the **linear subthreshold** behavior of the IF-neuron, including the time required for the membrane potential to increase from a reset to the threshold value. The MATLAB® routine pr23_1 simulates the behavior of an IF neuron using a simple RC circuit (R and C are both set to 1) and during the simulation loop it is checked if membrane potential V exceeds a threshold arbitrarily set at 0.25. If this threshold is exceeded, an action potential with an amplitude of 1 followed by a reset value of −0.1 is pasted in the model neuron's output signal.

Part of the MATLAB® script pr23_1 showing the paste procedure is listed below.

```
% test for threshold of 0.25 followed by pasting an action potential
if V(i+1)>=.25;
   V(i+1)=1;        % paste action potential: amplitude = 1
   i=i+1;           % update counter and
   T(i+1)=i*dt;     % time T
   V(i+1)=-.1;      % reset at -0.1 after the action potential
end;
```

An interesting modification of the IF-neuron is the **Quadratic Integrate-and-Fire neuron (QIF neuron)**. Here the second term in Eq. (23.3) is replace by a quadratic term (Latham et al., 2000):

$$\dot{V} = I + V^2 \tag{23.4}$$

Note that the subthreshold behavior of the QIF-neuron is codetermined by the quadratic term of V, therefore the behavior of a **QIF-neuron is not linear**. The attractive feature of this approach is that the quadratic parabola creates a stable fixed point to model the resting potential and an unstable fixed point that acts as a threshold above which V increases. The QIF model therefore captures the behavior of the neuron around the resting potential and also mimics the upstroke of the action potential. Unlike in the linear version of the IF-neuron, pasting in a spike isn't necessary in the QIF-neuron. However, since the membrane potential keeps increasing after it exceeds the instable fixed point, the QIF model requires an ad hoc (hard) reset when a predetermined spike amplitude is reached (Fig. 23.7B).

Part of the MATLAB® script pr23_2 showing the reset procedure is listed below. In addition to the reset, in order to ensure fixed amplitudes for the action potentials: if V exceeds the threshold value of 15 it is set to 15.

```
% test for predetermined spike amplitude of 15
if V(i+1)>=15
   V(i+1)=-1;          % reset to -1
   V(i)=15;            % set the AP amplitude to 15
end;
```

The MATLAB® routine pr23_3 illustrates a 1-D model with membrane depolarizing and leak currents. This model, described by Izhikevich (2007), is a 1-D reduction of the Hodgkin and Huxley equations (Hodgkin and Huxley, 1952). With a sufficient amount of injected current, the membrane characteristic of this model is capable of undergoing a saddle-node bifurcation (Fig. 23.4). Using this script, one can compare the behavior, depending on a number of initial conditions, with pr23_3(0); and pr23_3(60); the number in between brackets is the injected current. The shortcoming of this model is that it can only depolarize and if a repolarization is needed for a particular modeling approach, an ad hoc reset of the membrane potential is required, just as in the QIF model described above. See Izhikevich (2007) for further details of this model.

23.4.2 One-Dimensional Dynamics: Excitability on a Circle

If one changes the straight line into a circle, one can still use a 1-D approach while an oscillation now becomes possible (e.g., Fig. 23.1C). This approach can be explored with pr23_4. In this case the dynamic variable is the phase $0 \leq \varphi < 2\pi$. The dynamics is governed by:

$$\dot{\varphi} = u + I + \sin\varphi \tag{23.5}$$

The expression in Eq. (23.5) implies that this approach is limited in the sense that it can only be used to model the phase φ. The amplitude of the oscillation is not modeled but simply a function of the phase (Fig. 23.7C). In the example in pr23_4, the variable u determines the threshold and I represents the current input to the neuron.

The following loop is used in MATLAB® script pr23_4 to simulate the behavior of the model. The membrane potential v is the projection of the phase (phi) on the X-axis. Variables u and I are the same as in Eq. (23.5).

```
for i=1:n;                          % loop to simulate the output
   d_phi=(u+I+sin(phi(i)))*dt;
   phi(i+1)=phi(i)+d_phi;           % update phase (phi)
   v(i)=cos(phi(i));                % Membrane Potential=projection
   t(i+1)=t(i)+dt;                  of phi on X-axis
end;
```

Note: The type of oscillator in Eq. (23.5) can be employed to study networks of coupled oscillators, the so-called Kuramoto model (Breakspear et al., 2010; Chapter 30).

23.4.3 Two-Dimensional Dynamics to Model Neural Excitability

The formalism for the membrane potential model of the squid's giant axon proposed by Hodgkin and Huxley (1952) (HH) has been shown to be very powerful for simulation of a wide variety of neuronal systems. The original HH model is **nonlinear** and **four-dimensional** because it contains variables for membrane potential (V), potassium activation (n), sodium activation (m), and inactivation (h) (Chapter 29). Due to the nonlinearity and high-dimensionality of the model, it must be analyzed

with computational methods. To make the neuronal models accessible to mathematical analysis, attempts have been made to simplify the models.

A pioneer model in which the HH model is reduced to two dimensions is the **FitzHugh–Nagumo model** (FitzHugh, 1961; Nagumo et al., 1962):

$$\begin{aligned} \dot{V} &= I + V(a - V)(V - 1) - w \\ \dot{w} &= bV - cw \end{aligned} \tag{23.6}$$

Here a, b, and c are parameters, I is injected current, thus the remaining variables are membrane potential V and w responsible for resetting a depolarized membrane. The latter reset variable can be considered to represent potassium activation (n in HH), sodium inactivation (h in HH), or both. Sodium activation is ignored and replaced by its equilibrium value because Na^+ activation is much faster than its inactivation or the activation of K^+.

MATLAB® script pr23_5 simulates the behavior of the FitzHugh–Nagumo model with a series of input current pulses to demonstrate the nonlinear behavior (i.e., deviation from superposition). The second part of the program (not shown here) analyzes the behavior of this model in the phase plane.

```
% pr23_5
% FitzHugh-Nagumo Model Simulation
% Input current I consists of a set of impulses
% showing the nonlinear behavior (i.e. deviation
% from superposition) of the model

clear;
close all;
% parameters
dt=0.1;
ww(1)=0;VV(1)=.1;T(1)=0;
a=.139;b=.008;
c=2.54*.008;
% input current
I=zeros(1,10000);
I(1000)=1;I(2000)=1.5;
I(3000)=1.85;I(4000)=2;I(5000)=3.5;
I(8000)=2.5;I(8700)=3.5;
```

```
% loop to simulate the F-N behavior
for i=1:9999
   T(i+1)=i*dt;
   ww(i+1)=ww(i)+(b*VV(i)-c*ww(i))*dt;
   VV(i+1)=VV(i)+(VV(i)*(a-VV(i))*(VV(i)-1)-ww(i)+I(i))*dt;
end;

% plot the result
figure;hold;
subplot(311),plot(T,I,'r')
title('FitzHugh-Nagumo Model Simulation')
ylabel('Input Current')
subplot(312),plot(T,VV,'k');
xlabel('Time')
ylabel('Membrane Potential')
subplot(313),plot(T,ww,'k');
xlabel('Time')
ylabel('Recovery Variable')
```

Just as 1-D nonlinear dynamics can be examined on the phase line, 2-D cases can be examined in the phase plane. For instance, if we use Eq. (23.6) for the FitzHugh–Nagumo model, we can capture the system's dynamical properties in the V–w plane by depicting the flow at a number of locations (Fig. 23.7D) (use the second part of pr23_5 for a much more detailed diagram of the flow). Usually it is also efficient to determine the locations where flows for V and w are zero, the so-called nullclines or isoclines. The w nullcline, where $\dot{w} = 0$, can be determined from the second expression in Eq. (23.6) and is given by the relationship $w = (b/c)\,V$ (red line in Fig. 23.7D). Similarly, the V nullcline, where $\dot{V} = 0$, must be determined from the first expression in Eq. (23.6) and satisfies $w = I + V(a - V)(V - 1)$ (green line in Fig. 23.7D). The location where both nullclines intersect is a fixed point of the 2-D system (black dot in Fig. 23.7D). Identical to the 1-D case, fixed points in 2-D can be stable, unstable, or half stable (details on how to determine the stability in the phase plane are described in Appendix 23.1).

Different reductions of the HH formalism are described by Morris and Lecar (1981) and Izhikevich (2007). In these reduced models, similar to the FitzHugh–Nagumo model, depolarization is instantaneous (a Ca^{2+} current in Morris and Lecar, and a Na^{+} current in the model by Izhikevich), while K^{+} in both these models is responsible for a slower hyperpolarizing current. Therefore, both models are two-dimensional: membrane potential (V) and the K^{+} activation (n). In both models there is

also a leak current (I_L) included. The $I_{Na,p} + I_K$-model by Izhikevich (2007) can be explored with the MATLAB® script pr23_6 (see also Exercise 23.4).

This $I_{Na,p} + I_K$-model is governed by the following pair of differential equations:

$$\begin{aligned} C\dot{V} &= I_{\text{inject}} + g_L(V - E_L) + g_{Na}m_\infty(V)(V - E_{Na}) + g_K n(V - E_K) \\ \dot{n} &= [n_\infty(V) - n]/\tau(V) \end{aligned} \tag{23.7}$$

The symbols in Eq. (23.7) are the same as the ones commonly used in the HH model (for details, see Chapter 29, Eqs. 29.1–29.4). The two dynamic variables are V—membrane potential and n—the variable for the potassium conduction. The remaining parameters are as follows: C is membrane capacitance, E's are equilibrium potentials, g's are conductances, $m_\infty(V)$ and $n_\infty(V)$ are the membrane potential–dependent equilibrium values for m and n, $g_L(V - E_L)$ is Ohm's law for the leak current, $g_{Na}m_\infty(V)(V - E_{Na})$ is Ohm's law for the sodium current that is assumed instantaneous here, $g_K n(V - E_K)$ is Ohm's law for the potassium current, and $\tau(V)$ is the membrane potential–dependent time constant governing the dynamics of n.

The common features in the FitzHugh–Nagumo, Morris–Lecar, and $I_{Na,p} + I_K$ models are that they are nonlinear and two-dimensional. This limited number of dimensions makes these models accessible for mathematical analysis in the 2-D phase plane. Izhikevich (2007) describes the codimension-1 bifurcations that can be observed in these models. In spite of the limited dimensionality of these models, they can exhibit a rich variety of behaviors. Two examples of bifurcation in the $I_{Na,p} + I_K$-model are shown in Figs. 23.8 and 23.9, which were created with MATLAB® script pr23_6. Fig. 23.8 is the version with a high-threshold (threshold at −25 mV) and slow K^+ current. It shows the so-called Saddle-Node on Invariant Cycle (SNIC) bifurcation, a bifurcation that can describe the transition between resting and spiking states. The upper row in Fig. 23.8 shows the properties of the resting state where there is one stable equilibrium (the resting potential) located on a stable limit cycle. In the middle and bottom rows in Fig. 23.8, we simulate current injection and, in the bottom row, the fixed point vanishes while a stable limit cycle (i.e., spiking) appears.

Fig. 23.9 is the version with a low-threshold K^+ current (threshold at −45 mV). It shows the so-called supercritical Hopf bifurcation between resting and spiking states. The spiking state corresponds with a limit cycle which is drawn schematically in Fig. 23.9. The top row in Fig. 23.9 shows the rest state without injected current and a stable fixed point. With the injection of current a stable limit cycle appears and grows with increased current injection. Note that the frequency of the oscillation doesn't change much with injected current while the amplitude does. Also of interest is

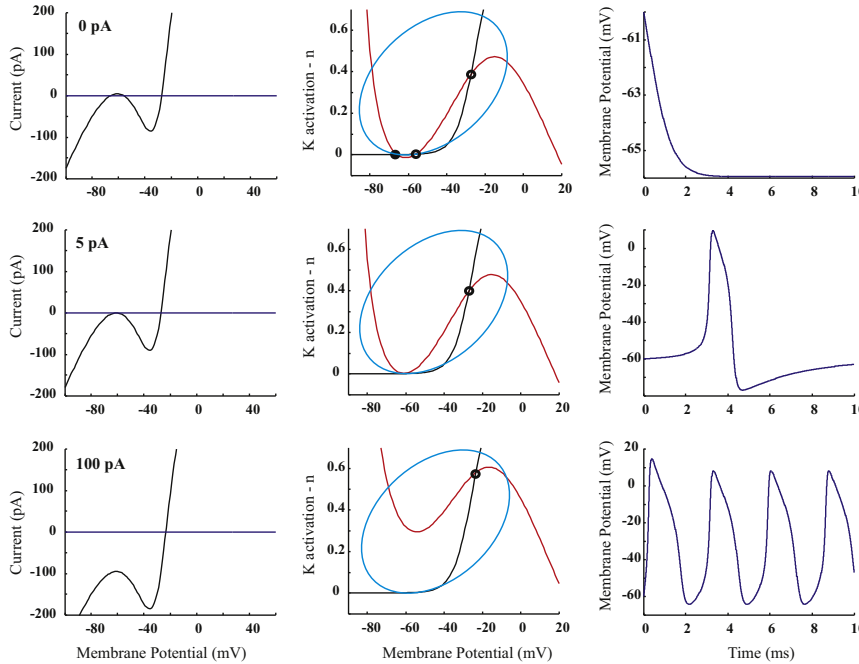


FIGURE 23.8 Integrator behavior at different values of current injection. Left column: $I-V$ relationship. Middle column: the null-clines (red—V-null and black—u-null). Right column: time series. From rest to spiking there is a saddle-node on invariant cycle bifurcation a little below an injection of 5 pA (middle row). The spike-amplitude is constant after passing the bifurcation. Spiking is initially slow but the firing frequency increases with distance from the bifurcation; therefore this bifurcation is also called the infinite-period bifurcation. This figure was created with pr23_6(InjectionCurrent,-25,-80); in which the injection current varied between 0 and 100 pA. *Notes: Black open* and *filled circles* are unstable and stable fixed points, respectively. Limit cycles (light blue) are drawn schematically and not based on the precise vector field!

the damped oscillation that occurs with small perturbations of the rest state (top-right panel in Fig. 23.9).

The examples in Figs. 23.8 and 23.9, based on the $I_{Na,p} + I_K$-model described by Izhikevich (2007), show that depending on the threshold for K^+ the stable resting potential loses stability via (1) a saddle-node (high threshold, Fig. 23.8) or (2) a Hopf bifurcation (low threshold, Fig. 23.9). Depending on other parameter values there are two subtypes of each type of bifurcation.

1. The saddle-node bifurcation can either be located on an invariant cycle (SNIC) if the rate of K^+ activation is low (as shown in Fig. 23.8), or off-the-cycle if the K^+ activation is fast. The off-cycle saddle-node bifurcation can be simulated with MATLAB® script pr23_7. Use the command pr23_7(Inject, -25,-80) and test with the

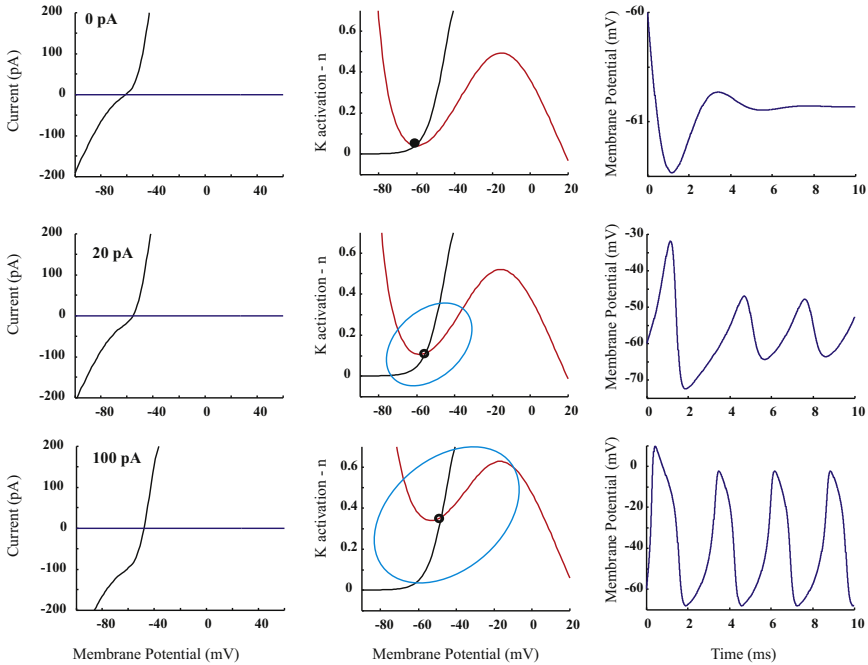


FIGURE 23.9 Resonator behavior at different values of current injection. Left column: $I-V$ relationship. Middle column: the null-clines (red—V-null and black—u-null). Right column: time series. From rest to spiking there is **supercritical Hopf bifurcation.** Note that the limit cycle amplitude grows with the amount of injected current, i.e., distance from the bifurcation but the oscillation's frequency only changes slightly with increased current injection. This figure was created with pr23_6(InjectionCurrent,-45,-78); in which the injection current was varied between 0 and 100 pA. *Notes: Black open* and *filled circles* are unstable and stable fixed points, respectively. Limit cycles (light blue) are drwawn schematically and not based on the precise vector field!

following values for Inject: 0, 4, 5, 7, 10 pA. This will show a bifurcation where the stable state disappears between injection values between 4 and 5 pA. The interesting phenomenon is that the excited state is associated with a DC shift of the membrane potential when the model neuron spikes.

2. The Hopf bifurcation can either be supercritical (shown in Fig. 23.9) if the sodium activation curve (m_∞) is not too steep, or subcritical if the sodium activation is steep. The subcritical Hopf bifurcation can be simulated with pr23_8. You can observe the bifurcation with the command pr23_8(Inject, -45,-78) and injection currents of 0, 30, 45, 50, 60 pA. In this subcritical case, the bifurcation occurs a little below 50 pA. Note that, close to the bifurcation value, this set of parameters is associated with strong subthreshold oscillations and that (unlike the behavior with the

TABLE 23.1 Overview of the Different Bifurcations From Rest to Firing in the $I_{Na,p} + I_K$-Model

High threshold K^+ activation	Slow K^+ activation	Saddle-node on invariant cycle (SNIC)
	Fast K^+ activation	Saddle-node
Low threshold K^+ activation	Steep Na^+ activation	Subcritical Hopf
	No steep Na^+ activation	Supercritical Hopf

supercritical Hopf scenario) the spike amplitude remains the same with increased current injection.

As compared to the dynamics associated with the bifurcations in Figs. 23.8 and 23.9 and both the off-the-cycle saddle-node and the subcritical Hopf bifurcations, there is another interesting difference: in the latter pair, the limit cycle coexists with a resting state (bistability), while in the examples in Figs. 23.8 and 23.9 there is only a single stable state available (monostability).

The model properties associated with the four different bifurcations we observe in the 2-D model are summarized in Table 23.1.

23.5 CODIMENSION-2 BIFURCATIONS

The saddle-node, transcritical, pitchfork, Hopf, SNIC bifurcations are all examples of codimension-1 bifurcations, which indicates that we varied a single parameter (e.g., r in Fig. 23.4) to observe the bifurcation. We can also consider higher codimension bifurcations such as the codimension-2 bifurcations in which two parameters are varied. In Chapter 30, we will consider such an example that can lead to sudden transitions in a network of model neurons using the so-called Ising spin model (Fig. 30.3).

APPENDIX 23.1

Stability Around a Fixed Point in the Phase Plane

The FitzHugh–Nagumo model is given in Eq. (23.6) and repeated here for convenience:

$$\begin{aligned} \dot{V} &= I + V(a - V)(V - 1) - w \\ \dot{w} &= bV - cw \end{aligned} \tag{23.6}$$

If we simplify the notation, we get:

$$\begin{aligned}\dot{V} &= f(V,w)\\ \dot{w} &= g(V,w)\end{aligned} \tag{A23.1-1}$$

At the fixed point(s) F in the phase plane we know that both $\dot{V}$ and $\dot{w}$ must be zero, thus:

$$\begin{aligned}f(V_F,w_F) &= 0\\ g(V_F,w_F) &= 0\end{aligned} \tag{A23.1-2}$$

Now we perturb the system slightly about the fixed point to see if the perturbation is followed by a return to the fixed point (when it is stable), or not (when it is unstable). We define these small perturbations in the V and w directions as x and y, respectively:

$$\begin{aligned}x &= V - V_F\\ y &= w - w_F\end{aligned} \tag{A23.1-3}$$

For the dynamics in the V-direction around the fixed point, we can determine from the first expression in Eq. (A23.1-3) that $\dot{x} = V$ because V_F is constant. Furthermore, by substitution of Eq. (A23.1-3) into the first expression in Eq. (A23.1-1) we find that $\dot{x} = f(x + V_F, y + w_F)$. Now we use a 2-D Taylor expansion (Appendix 11.1) where we ignore all higher-order terms because we only consider a small perturbation, and we get:

$$\dot{x} = \underbrace{f(V_F,w_F)}_{0} + x\left.\frac{\partial f(V,w)}{\partial V}\right|_F + y\left.\frac{\partial f(V,w)}{\partial w}\right|_F \tag{A23.1-4}$$

Because we are at a fixed point, the first term vanishes, so the linearized expression for the dynamics about the fixed point is:

$$\dot{x} = x\left.\frac{\partial f(V,w)}{\partial V}\right|_F + y\left.\frac{\partial f(V,w)}{\partial w}\right|_F \tag{A23.1-5a}$$

Similarly, we will get for $\dot{y}$:

$$\dot{y} = x\left.\frac{\partial g(V,w)}{\partial V}\right|_F + y\left.\frac{\partial g(V,w)}{\partial w}\right|_F \tag{A23.1-5b}$$

Thus to get the linearized version of the 2-D dynamics about the fixed point, we may write:

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = J\begin{pmatrix}x\\ y\end{pmatrix},$$

where the numerical values in the Jacobian matrix J are determined by:

$$J = \begin{bmatrix} \left.\frac{\partial f(V,w)}{\partial V}\right|_F & \left.\frac{\partial f(V,w)}{\partial w}\right|_F \\ \left.\frac{\partial g(V,w)}{\partial V}\right|_F & \left.\frac{\partial g(V,w)}{\partial w}\right|_F \end{bmatrix} \tag{A23.1-6}$$

Note that the matrix MUST be evaluated at the fixed point. In our example in Fig. 23.7D, there is no injected current I and the fixed point is located at the origin. Using Eq. (A23.1-6) and Eq. (23.6), this creates a linearized expression for the perturbation about the origin:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{bmatrix} -a & -1 \\ b & -c \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{A23.1-7}$$

This is a problem we can solve with the procedure described in Chapter 9, Section 9.5, Fig. 9.1C.

In our numerical example in pr23.5 we have $a = 0.139$, $b = 0.008$, and $c = 0.0203$. Using Eq. (A23.1-7), we have the following values for the Jacobian: trace $T = -a - c = -0.16$ and determinant $D = ac + b = 0.01$. We can see in Fig. 9.1C that this represents case 2a, a stable spiral. Using Eq. (9.31), repeated here for convenience, we find the solutions for the eigenvalues $\lambda_{1,2}$ as:

$$\lambda_{1,2} = \left(T \pm \sqrt{T^2 - 4D}\right) \Big/ 2. \quad (9.31)$$

Rounded to the second decimal, this gives the following solutions $\lambda_{1,2} = -0.08 \pm 0.07j$. This confirms that the fixed point is a stable spiral (also called a stable focus or a spiral sink): it is stable because the real part of the eigenvalues is negative, and oscillatory since the solutions are complex numbers. Thus the fixed point in the phase plane plot in the example in Fig. 23.7D is stable.

WARNING: Recall that we linearized the dynamics about the fixed point in Eq. (A23.1-7). If you apply this technique, do realize that this is **not always** justified. In borderline cases, this approach may not lead to a correct result. So what are borderline cases? These are the solutions that fall on the lines in Fig. 9.1C, where the trace T, or determinant D, or the term $\sqrt{T^2 - 4D}$ are zero. In these borderline cases, a good option is to simulate the dynamics about the fixed point.

EXERCISES

23.1 Analyze the following expressions and determine their bifurcation diagram (similar to the example in Fig. 23.5). Next classify the type of bifurcation according to Fig. 23.4.
a. $\dot{v} = rv + v^2$
b. $\dot{v} = rv - 4v^3$
c. $\dot{\varphi} = r + \sin(\varphi)$

23.2 Use MATLAB® pr23_4 to determine the input–output relationships of the phase model. Determine two relationships by using both the spike frequency and spike amplitude as the output metrics. Plot both findings in graphs of output versus input.

23.3 Modify MATLAB® script pr23_5 to examine the phase plane for nonzero input values of I. Use the procedure outlined in Appendix 23.1 to establish the stability of the fixed point for several values of I.

23.4 Use MATLAB® pr23_6 to examine input–output relationships of the $I_{Na,p} + I_K$-model. Determine the relationships using the parameters in Figs. 23.8 and 23.9 (i.e., low- and high-threshold values for the K^+-current).
a. For each case, determine two relationships by using both the spike frequency and spike amplitude as the output metrics.
b. Plot all findings in graphs of output versus input.

References

Breakspear, M., Heitmann, S., Daffertshofer, A., 2010. Generative models of cortical oscillations: neurobiological implications of the Kuramoto model. Front. Hum. Neurosci. 4, 190.

FitzHugh, R., 1961. Impulses and physiological states in theoretical models of nerve membrane. Biophys. J. 1, 445–466.

Hodgkin, A.L., Huxley, A.F., 1952. A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. 117, 500–544.

Izhikevich, E.M., 2007. Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting. MIT Press, Cambridge, MA.

Kaplan, D., Glass, L., 1995. Understanding Nonlinear Dynamics. Springer-Verlag, New York.
An excellent introduction into the application of nonlinear dynamics to analysis of time series. The work includes multiple examples and computer projects.

Lapicque, L., 1907. Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarization. J. Physiol. Pathol. Gen. 9, 620–635.

Latham, P.E., Richmond, B.J., Nelson, P.G., Nirenberg, S., 2000. Intrinsic dynamics in neuronal networks. I. Theory. J. Neurophysiol. 83, 808–827.

Morris, C., Lecar, H., 1981. Voltage oscillations in the barnacle giant muscle fiber. Biophys. J. 35, 193–213.
Nagumo, J., Arimoto, S., Yoshizawa, S., 1962. An active pulse transmission line simulating nerve axon. Proc. IRE 50, 2061–2070.
Peitgen, H.-O., Jürgens, H., Saupe, D., 1992. Chaos and Fractals New Frontiers of Science. Springer-Verlag, New York.
Strogatz, S.H., 1994. Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering. Westview Press, Cambridge, MA.

CHAPTER

24

Volterra Series

24.1 INTRODUCTION

Most physiological systems cannot be modeled successfully as linear systems. At best, a linear model can be considered an approximation of physiological activity in cases where the output of a physiological system behaves (almost) linearly over a limited range of the input. In the following, we extend the convolution integral that describes the behavior of linear devices to the convolution-like Volterra series, which can be used to represent nonlinear systems. Because the expressions for higher-order nonlinear terms require significant computational resources and become very complex to deal with, we will demonstrate the general principles for second-order systems. See Schetzen (2006) if you are interested in details of higher-order systems.

In a linear time-invariant (LTI) system, the convolution integral links output $y(t)$ and input $x(t)$ by means of its weighting function $h(t)$ (Fig. 24.1A) (Chapter 13):

$$y(t) = h(t) \otimes x(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau \tag{24.1}$$

Here $\otimes$ symbolizes the convolution operation and the system's weighting function $h(t)$ is its unit impulse response (UIR). This role of $h(t)$ can be verified by using a unit impulse $\delta(t)$ as the input. In this case we obtain (using the sifting property)

$$\int_{-\infty}^{\infty} h(\tau)\delta(t-\tau)d\tau = h(t) \tag{24.2}$$

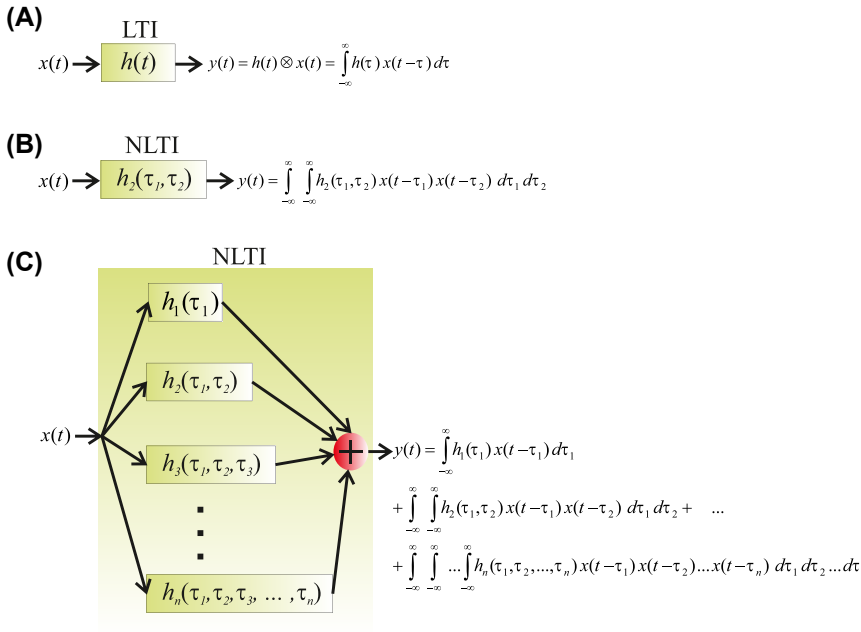


FIGURE 24.1 Example of linear time-invariant (LTI) and nonlinear time-invariant (NLTI) systems. (A) A linear system. (B) A second-order system and (C) a combined nth-order system. The output of the first-order components is determined by the convolution integral and output of higher-order components are obtained from convolution-like integrals. The output of the nth-order system is represented by a Volterra series consisting of the sum of the individual components, each determined by a convolution-like expression.

Note: In the following we will use the sifting property of the unit impulse repeatedly (for a discussion, see Section 2.2.2). The sifting property is defined as: $x(t) = \int_{-\infty}^{\infty} x(\tau)\delta(\tau - t)d\tau = \int_{-\infty}^{\infty} x(\tau)\delta(t - \tau)d\tau$. The unit impulse δ has properties of a function with even symmetry, therefore the evaluation of the integral above is the same for $\delta(t - \tau)$ and $\delta(\tau - t)$. You can also see that this must be the case since the outcome is $\delta(0)$ for $t = \tau$ in both cases, $\delta(t - \tau)$ and $\delta(\tau - t)$.

Such an LTI system shows superposition and scaling properties. For instance, if we introduce a **scaled delta function** $C\delta(t)$ (C, constant) at the input, we get a **scaled unit impulse response function** $Ch(t)$ at the output:

$$\int_{-\infty}^{\infty} h(\tau)C\delta(t - \tau)d\tau = C \int_{-\infty}^{\infty} h(\tau)\delta(t - \tau)d\tau = Ch(t) \qquad (24.3)$$

Now consider a system that is governed by an equation that is convolution-like (Fig. 24.1B):

$$y(t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1, \tau_2) x(t - \tau_1) x(t - \tau_2) d\tau_1 d\tau_2 \tag{24.4}$$

Unlike the convolution in Eq. (24.1), this system works on two copies of input $x(t)$ instead of only one. As we discussed in Section 11.5, such a system is an example of a so-called second-order Volterra system. Note that the double integral in Eq. (24.4) is identical to the last term in the expression in Eq. (11.24). If we determine the UIR for the system in Eq. (24.4), we get

$$h_2(t, t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1, \tau_2) \delta(t - \tau_1) \delta(t - \tau_2) d\tau_1 d\tau_2 \tag{24.5}$$

Here we applied the sifting property twice: once for each of the delays τ_1 and τ_2. The result $h_2(t,t)$ is the diagonal of kernel h_2.

Note: You can see that in a second-order Volterra system, the UIR $h_2(t,t)$ does not fully characterize the system (unlike the situation in an LTI system). Instead it only characterizes the two-dimensional function $h_2(\tau_1,\tau_2)$ along the diagonal $\tau_1 = \tau_2$ in the τ_1,τ_2-plane. As we will see in Section 24.3 we need sets of paired impulses to fully characterize h_2.

The system in Eq. (24.4) is nonlinear because scaling doesn't hold. For example, the response to a scaled delta function $C\delta(t)$ at the input is

$$\begin{aligned} &\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1, \tau_2) C\delta(t - \tau_1) C\delta(t - \tau_2) d\tau_1 d\tau_2 \\ &= C^2 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1, \tau_2) \delta(t - \tau_1) \delta(t - \tau_2) d\tau_1 d\tau_2 = C^2 h_2(t, t) \end{aligned} \tag{24.6}$$

By comparing Eqs. (24.3) and (24.6) we can see that in this system the UIR, $h_2(t,t)$ scales with C^2 instead of C. As we will show in Section 24.2.1, superposition doesn't hold for this system either, but showing that scaling

does not hold is sufficient to negate the linearity of the system. In the remainder of this chapter we will continue our introduction of Section 11.5 by studying the properties of the Volterra series and applying it for the characterization of higher-order systems.

24.2 VOLTERRA SERIES

The mathematician Vito Volterra used series of convolution-like expressions to define the input−output relationship of nonlinear time-invariant (NLTI) systems (Fig. 24.1C):

$$y(t) = \overbrace{\int_{-\infty}^{\infty} h_1(\tau_1)x(t-\tau_1)d\tau_1}^{\text{1st−order Term}} + \overbrace{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2}^{\text{2nd−order Term}} + \ldots$$
$$+ \underbrace{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\ldots\int_{-\infty}^{\infty} h_n(\tau_1,\tau_2,\ldots,\tau_n)x(t-\tau_1)x(t-\tau_2)\ldots x(t-\tau_n)d\tau_1 d\tau_2\ldots d\tau_n}_{n\text{th−order Term}} \tag{24.7}$$

The output $y(t)$ of an nth-order system depends on multiple copies of the input and is the sum of the first-, second-, …, nth-order convolution-like expressions. The functions h_1, h_2, …, h_n are called the first-, second-, …, nth-order **Volterra kernels**. In some texts, a 0th-order kernel (h_0) representing a DC term, or offset, is added to Eq. (24.7). Just as in an LTI system, $y(t)$ is the UIR if the input $x(t)$ is a unit impulse $\delta(t)$. In higher-order systems, the contribution of the nth-order Volterra kernel to the UIR is a so-called diagonal slice through the kernel: i.e., a section through the kernel with all delays τ_1, τ_2, …, τ_n equal. An example for a second-order system ($n = 2$) is shown in Eq. (24.5).

Note: We can refer to h_1 as the unit impulse response (UIR) only if we deal with a first order Volterra system without a DC term, i.e., a (linear) system where h_1 is the only term of $y(t)$. In all other cases, the UIR is determined by the contributions of all of the system's Volterra kernels and not just by h_1.

If we represent the first-, second-, …, nth-order convolution-like terms in Eq. (24.7) as H_1, H_2, …, H_n we get an alternative, simplified notation:

$$y(t) = H_1[x(t)] + H_2[x(t)] + \ldots + H_n[x(t)] \tag{24.8}$$

Eq. (24.8) can be generalized for an Nth-order system:

$$y(t) = \sum_{n=1}^{N} H_n[x(t)] \tag{24.9a}$$

In some cases a DC term $H_0[x(t)] = h_0$ (with h_0 being a constant) is added. This allows one to further generalize Eq. (24.8) for the NLTI system to:

$$y(t) = \sum_{n=0}^{N} H_n[x(t)] \tag{24.9b}$$

Just as with any series, we should consider the convergence of the Volterra series. In our case, we approach this by optimistically assuming that any system we consider will be associated with a converging series. We can afford this optimism because we will only apply the Volterra series to known, relatively low-order systems and because we would immediately notice if the output predicted by the Volterra series would poorly match the measured output of the system it represents.

Recall that we can consider the Volterra series' approximation of output as a Taylor series with the addition of memory (Section 11.5). The Taylor series links output with instantaneous input (no memory, Eq. 11.16) whereas the Volterra series includes a memory component in the convolution-like integrals. These integrals show that the output at time t not only depends on current input signal $x(t)$, but on multiple copies of the delayed input, represented by $x(t-\tau_1), x(t-\tau_2), \ldots, x(t-\tau_n)$ in the integrals in Eq. (24.7).

24.2.1 Combined Input to a Second-Order Volterra System

In general, the input–output relationship of a second-order Volterra system without lower-order components can be specified by Eq. (24.4). We also demonstrated above that a second-order Volterra system doesn't scale as an LTI system (compare Eqs. 24.3 and 24.6). We can next show that the **superposition property** of an LTI system does not hold in the second-order Volterra system either. To accomplish this, we will determine the system's response to the sum of two inputs $x(t) = x_1(t) + x_2(t)$ relative to

its responses to $x_1(t)$ and $x_2(t)$ individually. The response to the combined inputs is:

$$y(t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)[x_1(t-\tau_1)+x_2(t-\tau_1)][x_1(t-\tau_2)+x_2(t-\tau_2)]d\tau_1 d\tau_2 \tag{24.10}$$

In Eq. (24.10) we have the following four terms:

$$H_2[x_1(t)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x_1(t-\tau_1)x_1(t-\tau_2)d\tau_1 d\tau_2 \tag{24.11a}$$

$$H_2[x_2(t)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x_2(t-\tau_1)x_2(t-\tau_2)d\tau_1 d\tau_2 \tag{24.11b}$$

$$\left.\begin{aligned} H_2[x_1(t),x_2(t)] &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x_1(t-\tau_1)x_2(t-\tau_2)d\tau_1 d\tau_2 \\ H_2[x_2(t),x_1(t)] &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x_1(t-\tau_2)x_2(t-\tau_1)d\tau_1 d\tau_2 \end{aligned}\right\} \text{cross} - \text{terms} \tag{24.11c}$$

Note that Eqs. (24.11a) and (24.11b) represent the expressions for the system's response when its input would be x_1 and x_2, respectively. The two cross-terms in expression (24.11c) are determined by both x_1 and x_2, and can be considered equal because **the second-order kernel is symmetric**, that is, $h(\tau_1,\tau_2) = h(\tau_2,\tau_1)$.

Note on the symmetry of $h(\tau_1,\tau_2,\ldots,\tau_n)$.
Recall that the kernel h of a linear system is obtained from the system's response to a unit impulse. As we will see in the following section, h can be determined in higher-order systems from the responses to multiple unit impulses. Since kernel h can be obtained from responses to combinations of unit-impulses, the symmetry assumption makes sense. This is the

case because there is no reason to assume that a system would be able to distinguish (i.e., react differently) between unit-impulse-1 followed by unit-impulse-2 as compared to unit-impulse-2 followed by unit-impulse-1 (the unit impulses are indistinguishable because they are identical). For a formal explanation see Chapter 3 in Schetzen (2006). If you have problems following this reasoning, you may come back to it after studying the concrete example in pr24_1.m and Section 24.3.

Based on the symmetry, we can rewrite the second equation in (24.11c) as:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_2,\tau_1)x_1(t-\tau_2)x_2(t-\tau_1)d\tau_2 d\tau_1, \tag{24.11d}$$

If we now interchange the dummy variables τ_1 and τ_2 this becomes:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x_1(t-\tau_1)x_2(t-\tau_2)d\tau_1 d\tau_2. \tag{24.11e}$$

This result indicates that the two expressions in (24.11c) are equal so that we may combine the cross-terms into:

$$2H_2[x_1(t),x_2(t)] = 2\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x_1(t-\tau_1)x_2(t-\tau_2)d\tau_1 d\tau_2 \tag{24.11f}$$

By combining Eqs. (24.10) and (24.11a, b, and f), we get the following expression for the output $y(t)$ for the sum of the inputs $x_1(t)+x_2(t)$:

$$\begin{aligned} y(t) &= H_2[x(t)] = H_2[x_1(t)+x_2(t)] \\ &= H_2[x_1(t)] + H_2[x_2(t)] + 2H_2[x_1(t),x_2(t)] \end{aligned} \tag{24.12}$$

The cross-terms $2H_2[x_1(t),x_2(t)]$ in Eq. (24.11f) represent the deviation of the second-order Volterra system's response from the response to $x_1(t)+x_2(t)$ if superposition were to hold; that is, in the second-order Volterra system the total response $y(t)$ to $x_1(t)+x_2(t)$ is **not** equal to the sum (superposition) of the responses to $x_1(t)$ and $x_2(t)$ individually: $H_2[x_1(t)]+H_2[x_2(t)]$.

24.3 A SECOND-ORDER VOLTERRA SYSTEM

As we discussed in Chapter 11, we can create a dynamical nonlinear system by combining a dynamical linear system (L) and a static nonlinear (N) one (Figs. 11.2 and 24.2A); the type of system that emerges from this combination of L and N is often indicated as an LN cascade. In the example in Fig. 24.2A, the dynamical linear system is a simple low-pass filter consisting of a resistor (R) and capacitor (C) $\left(h_{RC} = \frac{1}{RC}\, e^{-t/RC}\right)$. If you need to refresh your basic knowledge about RC filters, see Chapters 15 and 16. The static nonlinear component in Fig. 24.2A relates an input to the square of the output: i.e., output $y(t)$ is the square of its input $f(t)$: $y(t) = f(t)^2$. From this relationship, the static nonlinear component is considered a squarer. Following the procedure described in Section 2.5, we can establish that this cascade is a second-order Volterra system with a second-order kernel:

$$h_2(\tau_1, \tau_2) = h_{RC}(\tau_1) h_{RC}(\tau_2) = \left(\frac{1}{RC}\, e^{-\tau_1/RC}\right)\left(\frac{1}{RC}\, e^{-\tau_2/RC}\right) = \left(\frac{1}{RC}\right)^2 e^{-(\tau_1+\tau_2)/RC} \tag{24.13}$$

The following MATLAB® code (the first part of pr24_1.m available on http://booksite.elsevier.com/9780128104828/) will create an image of the two-dimensional Volterra kernel (Fig. 24.3) based on the known structure of the nonlinear cascade (Eq. 24.13).

```
% Linear component is a low-pass RC circuit
% we use R=10k and C=3.3uF
R=10e3;
C=3.3e-6;
RC=R*C;

% Timing parameters
sample_rate=1000;
dt=1/sample_rate;
time=0.1;
A=RC/dt;
T=100;          % The setting for timing and the length of correlation
                % calculations for both Volterra and Wiener kernels

% Step 1. The analog continuous time approach using the square of
% the unit impulse response of the filter: h(t)=(1/RC)*exp(-t/RC)
```

```
% to compare with discrete time approach (in the following Steps) we
      assume
% the discrete time steps (dt) and interval (time)
j=1;
for tau1=0:dt:time;
   i=1;
   r1(j)=(1/RC)*exp(-tau1/RC);    % 1st-order response in the cascade

   for tau2=0:dt:time
      y(i,j)=((1/RC)*exp(-tau1/RC))*((1/RC)*exp(-tau2/RC));
                  % Output y is h2 (=2nd order Volterra kernel)
                  % which is the square of the filter response
      i=i+1;
   end;
   j=j+1;
end;

% plot the surface of h2
y=y*dt^2;       % scale for the sample value dt
figure; surf(y);
axis([0 T 0 T min(min(y)) max(max(y))])
view(100,50)
title('2nd order Volterra kernel (h2) of an LN cascade')
xlabel('tau1');ylabel('tau2');zlabel('h2');
```

Now we validate our (nonparametric) approach with the Volterra series by using a parametric model, the Wiener cascade depicted in Fig. 24.2A. So far we have assumed that the internal structure of the second-order system is known. In this example we study the LN cascade system in Fig. 24.2A but we pretend to only know that it is a second-order Volterra system and that we do **not** know that it is a cascade or what its components are. Now we can use Eq. (24.12) and the procedure described in steps 1−6 below to find the second-order kernel h_2. Finally we will compare this result with the one we can obtain analytically (Eq. 24.13), which is shown in Fig. 24.3.

1. Using the approach in the example in Section 24.2.1, we use a pair of unit impulses occurring at times T_1 and T_2: $\delta(t-T_1)$ and $\delta(t-T_2)$ as inputs $x_1(t)$ and $x_2(t)$, respectively.
2. We determine the system's response to each of the inputs individually: the response to $x_1(t) = \delta(t-T_1)$ is $H_2[\delta(t-T_1)]$ and the response to $x_2(t) = \delta(t-T_2)$ is $H_2[\delta(t-T_2)]$ (Fig. 24.2B).

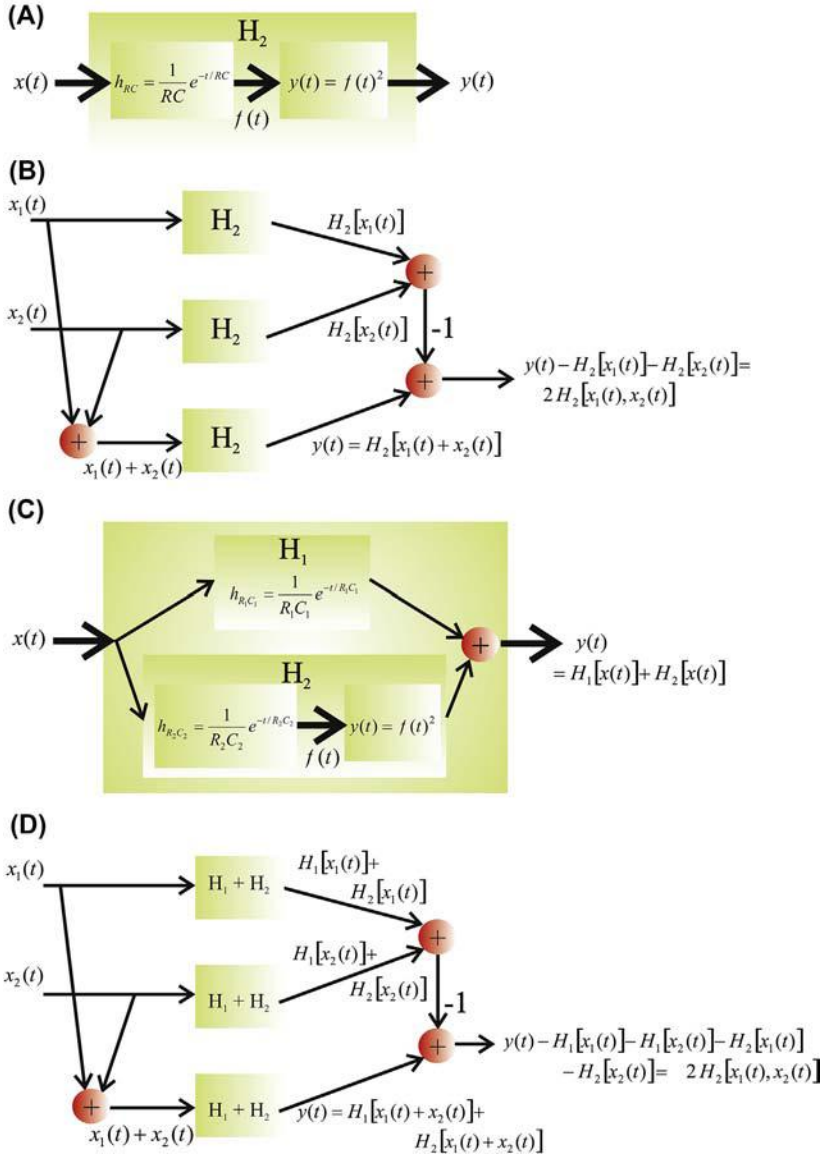


FIGURE 24.2 (A) Nonlinear system consisting of a cascade of a linear filter (a dynamical system) and a squarer (a static system). (B) Procedure to compute H_2 following Eq. (24.12). This procedure is one of the approaches used to determine kernel h_2 in pr24_1.m. (C) A general second-order Volterra system with a first- (H_1) and second-order (H_2) operator. (D) The same procedure as in panel (B) applied to the second-order system with a first-order component. This procedure is followed in script pr24_2.m.

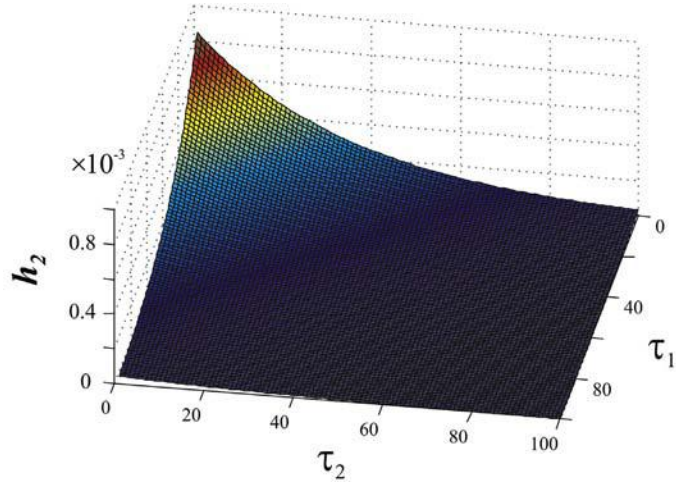


FIGURE 24.3 Example of a second-order Volterra kernel $h_2(\tau_1,\tau_2)$ determined by Eq. (24.13) in MATLAB® script pr24_1.m.

3. We determine the response to the sum of both impulses $x_1(t) + x_2(t) = \delta(t-T_1) + \delta(t-T_2)$ which is $y(t) = H_2[\delta(t-T_1) + \delta(t-T_2)]$ and according to Eq. (24.12):

$$y(t) = H_2[\delta(t-T_1)] + H_2[\delta(t-T_2)] + 2H_2[\delta(t-T_1), \delta(t-T_2)].$$

4. From the responses obtained in steps 2 and 3 above, we can solve for H_2:

$$H_2[\delta(t-T_1), \delta(t-T_2)] = \frac{y(t) - H_2[\delta(t-T_1)] - H_2[\delta(t-T_2)]}{2} \quad (24.14)$$

5. We use Eq. (24.11f) (divided by two) and substitute $\delta(t-T_1)$ for $x_1(t)$ and $\delta(t-T_2)$ for $x_2(t)$:

$$H_2[\delta(t-T_1), \delta(t-T_2)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)\delta(t-T_1-\tau_1)\delta(t-T_2-\tau_2)d\tau_1 d\tau_2$$

Using the sifting property twice, the double integral evaluates to:

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)\delta(t-T_1-\tau_1)\delta(t-T_2-\tau_2)d\tau_1 d\tau_2 = h_2(t-T_1, t-T_2) \quad (24.15)$$

This is the second-order Volterra kernel we are looking for.

6. To relate Eqs. (24.14) and (24.15) to the definition of the second-order kernel $h_2(\tau_1,\tau_2)$, we set $\tau_1 = t - T_1$ and $\tau_2 = t - T_2$. By using the common variable t, we can relate the delays by: $\tau_1 + T_1 = \tau_2 + T_2 \rightarrow \tau_2 = \tau_1 + T_1 - T_2$. In the τ_1–τ_2 plane, this represents a line at 45 degrees with an intercept at $T_1 - T_2$. Therefore the response obtained in Eq. (24.15) is a slice of the second-order kernel along the line $\tau_2 = \tau_1 + T_1 - T_2$.

Following this procedure, we can obtain the second-order kernel by repeatedly probing the system with pairs of unit impulses at different times T_1 and T_2. By varying the timing of the impulses, we can determine h_2 in both dimensions τ_1 and τ_2, i.e., we fill in the plane in Fig. 24.4.

24.3.1 Discrete Time Implementation

Now we are ready to implement the procedure described in steps 1–6 above for the cascade in Fig. 24.2A. A diagram of this procedure is shown in Fig. 24.2B. Because in this example we know the parameters of the LN cascade, we can compare the result we obtain following steps 1–6 with the earlier analytically derived result based on our knowledge of the system (depicted in Fig. 24.3).

Recall that for the discrete time solution in the MATLAB® file below it is assumed that the sample interval dt is much smaller than the time constant of the filter: that is, $RC/dt >> 1$ (see Section 16.2). If this assumption is violated too much, the approximation of the differential equation by the difference equation will be compromised.

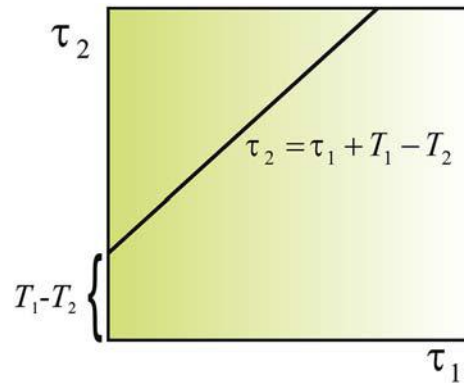


FIGURE 24.4 The τ_1–τ_2 plane and the section represented by $\tau_2 = \tau_1 + T_1 - T_2$. By varying the values for T_1 and T_2, the entire plane can be sampled.

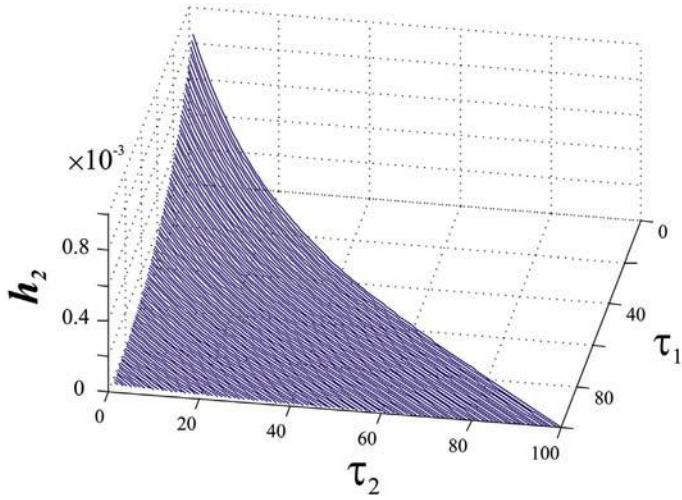


FIGURE 24.5 By following the procedures in the first and second parts in script pr24_1.m we can compare the second-order Volterra kernels obtained from the parametric LN cascade model (Fig. 24.3, based on Eq. 24.13) and that obtained using the nonparametric approach in which we determined h_2 by the procedure outlined in steps 1–6 (Section 24.3) and represented in Fig. 24.2B. The result of the latter procedure (obtained in the second part of pr24_1) is shown here. As can be seen by comparing this result with the earlier one in Fig. 24.3, both approaches agree. Because of the symmetry in h_2, only one half of the kernel is depicted.

Eq. (24.14) can be used to determine the second-order kernel of the system. The following MATLAB® code (the second part in pr24_1.m available on http://booksite.elsevier.com/9780128104828/) will create an image of the two-dimensional kernel shown in Fig. 24.5.

```
delay1=1;
for delay2=delay1:1:length(x);

   x1=zeros(1,100);x1(delay1)=1;          % unit impulse with delay 1
   x2=zeros(1,100);x2(delay2)=1;          % unit impulse with delay 2

% The summed input xs, containing two unit impulses
   if (delay1==delay2);
      xs=zeros(1,100);xs(delay1)=2;       % delays are equal
   else
      xs=zeros(1,100);xs(delay1)=1;xs(delay2)=1;
                                          % sum of two unit impulses if
                                             delays
                                          % are NOT equal
   end;
```

```
% Compute the system outputs to individual and combined unit
  impulses
   y1_previous=0;
   y2_previous=0;
   ys_previous=0;

   for n=1:length(x);
      % reponse to delay1
      y1(n)=(A*y1_previous+x1(n))/(A+1);   % the linear component
      y1_previous=y1(n);
      z1(n)=y1(n)^2;                       % the squarer
      % response to delay2
      y2(n)=(A*y2_previous+x2(n))/(A+1);   % the linear component
      y2_previous=y2(n);
      z2(n)=y2(n)^2;                       % the squarer
      % reponse to the sum of both delays
      ys(n)=(A*ys_previous+xs(n))/(A+1);   % the linear component
      ys_previous=ys(n);
      zs(n)=ys(n)^2;              % the squarer
   end;

   h=(zs-z1-z2)/2;                % A slice of the kernel h2
                                  % in the tau1-tau2 plane this is a line
                                  % at 45 degrees with intersection
                                  % delay1-delay2
   tau1=delay2:1:length(x);
   tau2=tau1+(delay1-delay2);
   h=h(delay2:length(h));

   plot3(tau1,tau2,h);

end;

axis([0 T 0 T])
view(100,50)
% Only half is shown because kernel h2 is symmetric
title('half of 2nd order Volterra kernel (h2) of an LN cascade')
xlabel('tau1');ylabel('tau2');zlabel('h2');
grid on
```

24.4 GENERAL SECOND-ORDER SYSTEM

The example of the cascade in Fig. 24.2A has a second order operator only. Generally a second-order system consists of both first- and second-order operators (assuming again that there is no H_0 component). Following the notation in Eq. (24.9a) with $N = 2$ we get:

$$y(t) = H_1[x(t)] + H_2[x(t)] \tag{24.16}$$

An example of such a system where the H_2 operator (representing an LN cascade) is extended with a first-order component H_1 is shown in Fig. 24.2C.

24.4.1 Determining the Second-Order Kernel

For determining h_2 in a system such as that described by Eq. (24.16), we can still use the procedure discussed in steps 1–6 (Section 24.3) and depicted in Fig. 24.2D. The method still works because superposition holds for the contribution of the first-order operator—that is, for input $x(t) = x_1(t) + x_2(t)$, the contribution of the first-order operator is simply the sum of the contributions for $x_1(t)$ and $x_2(t)$ separately

$$\begin{aligned}\int_{-\infty}^{\infty} h_1(\tau)x(t-\tau)d\tau &= \int_{-\infty}^{\infty} h_1(\tau)[x_1(t-\tau) + x_2(t-\tau)]d\tau \\ &= \int_{-\infty}^{\infty} h_1(\tau)x_1(t-\tau)d\tau + \int_{-\infty}^{\infty} h_1(\tau)x_2(t-\tau)d\tau\end{aligned} \tag{24.17a}$$

or in a more compact notation:

$$H_1[x(t)] = H_1[x_1(t) + x_2(t)] = H_1[x_1(t)] + H_2[x_2(t)] \tag{24.17b}$$

If we apply the same procedure (as shown in Fig. 24.2B) to a system that obeys $y(t) = H_1[x(t)] + H_2[x(t)]$ (for example, the system in Fig. 24.2C), the contribution of the first-order kernel will cancel because of the superposition property (Eq. 24.17). Just as in the previous example, the output will be $2H_2[x_1(t), x_2(t)]$, allowing us to find the second-order kernel by dividing the output of the procedure by 2 (see Eq. 24.14). In program pr24_2.m the procedure depicted in Fig. 24.2D is followed for the second-order system shown in Fig. 24.2C.

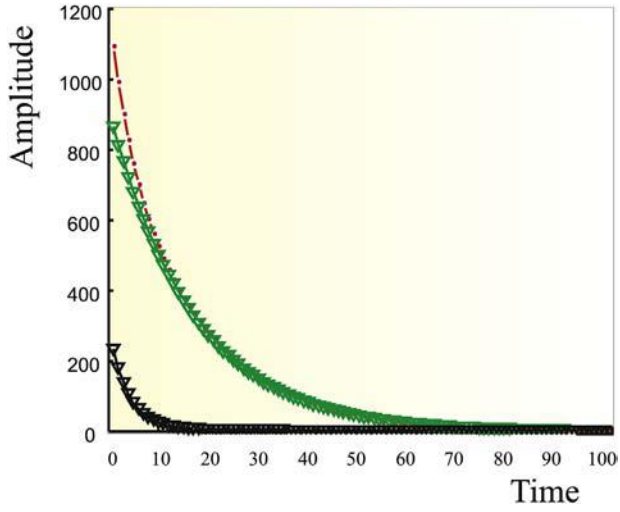


FIGURE 24.6 An example of a unit impulse response (UIR) (*red dots*, *upper curve*) of a second-order system such as in Fig. 24.2C. The response consists of a first-order component (*black triangles*, *lower curve*), a second-order part (*green triangles*, *middle curve*). This result was obtained with MATLAB® script pr24_2.m, albeit with different parameters than the version available on http://booksite.elsevier.com/9780128104828/.

24.4.2 Determining the First-Order Kernel

After we have determined the system's second-order kernel, we can also find its first-order kernel via the system's UIR. The UIR of the system in Fig. 24.2C will consist of a first- and second-order component (Fig. 24.6). Therefore, if we determine the system's UIR and subtract its second-order component, we have the first-order Volterra kernel h_1. The second-order component of the system's UIR is the slice through h_2 for $\tau_1 = \tau_2$ (i.e., the diagonal of the second-order kernel). This approach is now feasible, since we determined h_2 in the previous procedure. To summarize, we find h_1 by

$$h_1 = \text{UIR} - h_2(\tau_1, \tau_2) \quad \text{for} \quad \tau_1 = \tau_2 \tag{24.18}$$

24.5 SYSTEM TESTS FOR INTERNAL STRUCTURE

Nonlinear systems are usually complex and, to facilitate their characterization, one may attempt to simplify their structure by presenting them as a cascade of basic modules. As we have discussed in this chapter and in

Section 11.5, we often represent dynamical nonlinear systems with cascades of dynamical linear systems and static nonlinearities. In neuroscience, such cascades are frequently used to model neurons and their networks. For example, the integrate-and-fire neuronal model (e.g., Izhikevich, 2007) combines a linear low-pass filter (RC circuit) to mimic subthreshold integration of the biological membrane combined with a static nonlinearity that generates an action potential when the membrane potential exceeds a threshold. Models for neuronal networks also frequently use the cascade approach. For example, in a model to explain the electroencephalogram's alpha rhythm, Lopes da Silva et al. (1974) model synaptic function in the thalamocortical network with linear filters and a static nonlinearity to model action potential generation (see their Fig. 7). Examples of systems that are frequently used to represent nonlinear systems are depicted in Fig. 24.7; in this section we will discuss how these basic configurations may be recognized by examination of their Volterra kernels.

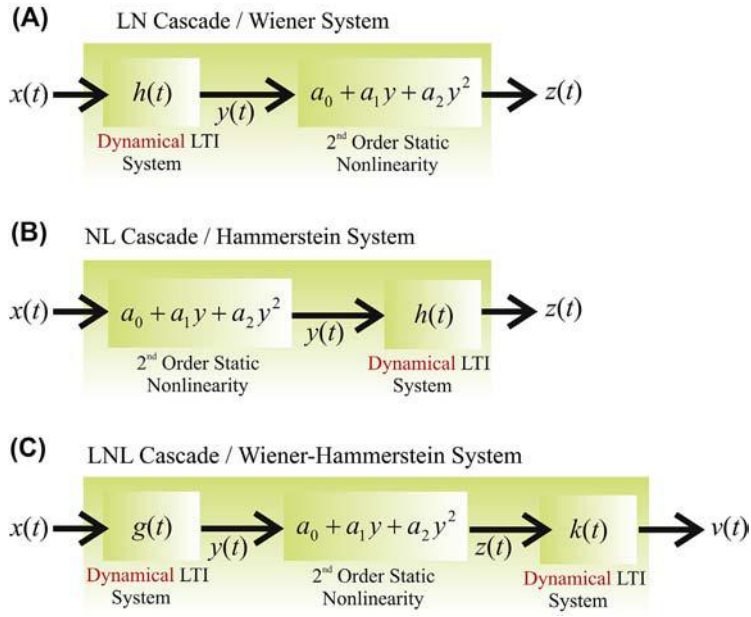


FIGURE 24.7 Frequently used cascade models to analyze nonlinear dynamical systems. (A) Cascade of a dynamical linear system followed by a static nonlinearity. (B) A similar cascade but as compared with (A) the order has changed: first the static nonlinearity followed by the linear component. (C) A static nonlinearity sandwiched in between two dynamical linear systems.

24.5.1 The LN Cascade

The LN cascade (linear system [L] followed by a nonlinear system [N], Fig. 24.7A), also called a Wiener system (not to be confused with the Wiener series we will discuss in Chapter 25), was also used in Section 11.5 when we demonstrated that the system's input–output relationship fits the Volterra series representation (Eq. 11.24). This result is repeated here:

$$z(t) = a_0 + \int_{-\infty}^{\infty} a_1 h(\tau) x(t-\tau) d\tau + \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} a_2 h(\tau_1) h(\tau_2) x(t-\tau_1) x(t-\tau_2) d\tau_1 d\tau_2$$

$$= h_0 + \int_{-\infty}^{\infty} h_1(\tau) x(t-\tau) d\tau + \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1, \tau_2) x(t-\tau_1) x(t-\tau_2) d\tau_1 d\tau_2 \tag{24.19}$$

From Eq. (24.19) we can see that the second-order Volterra kernel h_2 is related to the first-order kernel h_1. The first-order kernel is proportional with the UIR of the Wiener system's linear component: $h_1(\tau) = a_1 h(\tau)$, while the second-order kernel is given by $h_2(\tau_1, \tau_2) = a_2 h(\tau_1) h(\tau_2)$. If we keep one of the variables τ_1 or τ_2 constant, we obtain a section (slice) through the second-order kernel, which is also proportional with the linear component's UIR h. Let us keep τ_2 constant so that $h(\tau_2)$ is a constant value b, we then obtain the expression for a slice through the second-order kernel parallel to the τ_1 axis: $h_2(\tau_1) = b a_2 h(\tau_1)$. It is straightforward to show that the ratio between the first-order kernel and a slice (parallel to the τ_1 axis) of the second-order kernel is the constant $a_1/b a_2$. It is important to note here that this constant may be negative or positive; this should be taken into account when looking for proportionality! A similar result can be obtained for a slice parallel to the τ_2 axis when we hold τ_1 constant. It should be noted that this condition must be satisfied for a Wiener system but there are other configurations that may show the same property. Therefore, strictly speaking, failure of proportionality of first-order kernels and second-order slices can be used only to exclude the Wiener structure of a nonlinear system. Optimistically, one might say that if the condition is satisfied for a particular nonlinear system, we may use the Wiener structure to model the system.

24.5.2 The NL Cascade

The cascade shown in Fig. 24.7B, also called a Hammerstein system, is a cascade of a nonlinear static component (N) followed by a linear dynamic one (L). The output y of the first (nonlinear) component becomes the input

of the linear dynamical system. The output from this final dynamical component is then the system's output z:

$$z(t) = \int_{-\infty}^{\infty} h(\tau)y(t-\tau)d\tau = \int_{-\infty}^{\infty} h(\tau)\Big[a_0 + a_1x(t-\tau) + a_2x(t-\tau)^2\Big]d\tau \tag{24.20}$$

If we separate the three terms in Eq. (24.20) we can identify the three Volterra kernels h_0, h_1, and h_2.

$$\text{First term}: \underbrace{\int_{-\infty}^{\infty} h(\tau)a_0 d\tau}_{h_0} \tag{24.21a}$$

$$\text{Second term}: \int_{-\infty}^{\infty} \underbrace{h(\tau)a_1}_{h_1(\tau)}\, x(t-\tau)d\tau \tag{24.21b}$$

$$\begin{aligned}\text{Third term}: &\int_{-\infty}^{\infty} h(\tau)a_2x(t-\tau)^2d\tau \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \underbrace{h(\tau_1)a_2\delta(\tau_1-\tau_2)}_{h_2(\tau_1,\tau_2)}\, x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2\end{aligned} \tag{24.21c}$$

To obtain the Volterra formalism, we rewrote the single integral expression in Eq. (24.21c) as a double integral by separating the product $x(t-\tau)^2$ into $x(t-\tau_1)x(t-\tau_2)$. To make sure this product is only nonzero for $\tau_1 = \tau_2$, we added the $\delta(\tau_1 - \tau_2)$ function. The diagonal slice in the $\tau_1 - \tau_2$ plane of the Hammerstein's second-order kernel ($h(\tau)a_2$) is proportional to the UIR of the cascade's linear component ($h(\tau)$). It can be seen in Eq. (24.21b) that the first-order Volterra kernel ($h(\tau)a_1$) is also proportional to the linear component's impulse response. Consequently, the diagonal slice of the second-order kernel is proportional with the first-order kernel. Both characteristics discussed above (nonzero second-order kernel along the diagonal and its proportionality with the first-order kernel) may be used to test an unknown nonlinear system for an underlying Hammerstein structure.

24.5.3 The LNL Cascade

A combination of both the cascades discussed above is shown in Fig. 24.7C. Such a system is an LNL cascade, a nonlinear system (N) in between two linear systems (L), also called a Wiener–Hammerstein model. We obtain the output $z(t)$ of the static nonlinearity inside the cascade by following the same procedure we used to determine the Wiener system's output (Eq. 24.19):

$$\begin{aligned} z(t) &= a_0 + a_1 y(t) + a_2 y(t)^2 \\ &= a_0 + a_1 \int_{-\infty}^{\infty} g(\tau) x(t-\tau) d\tau + a_2 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(\tau_1) g(\tau_2) x(t-\tau_1) x(t-\tau_2) d\tau_1 d\tau_2 \end{aligned} \tag{24.22}$$

The LNL cascade's final output v is then the convolution of the expression above with the UIR k of the second linear system $v(t) = \int_{-\infty}^{\infty} k(\lambda) z(t-\lambda) d\lambda$ (we use λ here for the delay). Using $z(t)$ in Eq. (24.22) to determine $z(t-\lambda)$ gives:

$$\begin{aligned} v(t) = a_0 \int_{-\infty}^{\infty} k(\lambda) d\lambda + a_1 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k(\lambda) g(\tau) x(t-\tau-\lambda) d\tau d\lambda \ \ldots \\ + a_2 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k(\lambda) g(\tau_1) g(\tau_2) x(t-\tau_1-\lambda) x(t-\tau_2-\lambda) d\tau_1 d\tau_2 d\lambda \end{aligned} \tag{24.23}$$

To simplify, the first-order part of this expression can be rewritten using $\omega = \lambda + \tau$:

$$\begin{aligned} & a_1 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k(\lambda) g(\omega - \lambda) x(t-\omega) d\omega d\lambda \\ &= \int_{-\infty}^{\infty} \underbrace{\left[a_1 \int_{-\infty}^{\infty} k(\lambda) g(\omega - \lambda) d\lambda \right]}_{h_1(\omega)} x(t-\omega) d\omega \end{aligned} \tag{24.24a}$$

Similarly, using $v = \lambda + \tau_1$ and $\omega = \lambda + \tau_2$, the second-order part becomes:

$$a_2 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k(\lambda) g(v-\lambda) g(\omega-\lambda) x(t-v) x(t-\omega) dv d\omega d\lambda$$
$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \underbrace{\left[a_2 \int_{-\infty}^{\infty} k(\lambda) g(v-\lambda) g(\omega-\lambda) d\lambda \right]}_{h_2(v,\omega)} x(t-v) x(t-\omega) dv d\omega \tag{24.24b}$$

We can see that the second-order kernel is the integral expression in between the brackets: $a_2 \int_{-\infty}^{\infty} k(\lambda) g(v-\lambda) g(\omega-\lambda) d\lambda$. From this expression we can obtain the so-called second-order marginal kernel K_2^m (the sum of all kernel slices over one of the variables). By integrating this expression with respect to one of its variables, say v, we get:

$$K_2^m = a_2 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k(\lambda) g(v-\lambda) g(\omega-\lambda) d\lambda dv \tag{24.25a}$$

Now we make a change of timing variables again, $\xi = v - \lambda$, $d\xi = dv$, and rearrange the integral operation:

$$a_2 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k(\lambda) g(v-\lambda) g(\omega-\lambda) d\lambda dv$$
$$= a_2 \int_{-\infty}^{\infty} k(\lambda) \underbrace{\left[\int_{-\infty}^{\infty} g(\xi) d\xi \right]}_{A} g(\omega-\lambda) d\lambda \tag{24.25b}$$
$$= a_2 \underbrace{A}_{Part\ I} \underbrace{\left[\int_{-\infty}^{\infty} k(\lambda) g(\omega-\lambda) d\lambda \right]}_{Part\ II}$$

In the first step, we regrouped the integral operations and defined the outcome of the integral with respect to $d\xi$ as a number A. Subsequently we separated the expression into two parts. *Part I* is equal to A and *Part II* is proportional with the expression for the first-order kernel; this relationship can be seen by comparing *Part II* with the expression for h_1 in

Eq. (24.24a): $a_1 \int_{-\infty}^{\infty} k(\lambda)g(\omega - \lambda)d\lambda$. This latter term is simply *Part II* scaled by a_1. Of course, we would have obtained a similar outcome had we integrated the second-order kernel with respect to ω. This reasoning leads us to conclude that in an LNL sandwich, the marginal kernel K_2^m (the summation [integral] of all slices of the second-order kernel h_2 parallel to one of the axes) is proportional with the first-order kernel h_1. We can use the above proportionality between the marginal second-order kernel and the first-order one to test for a potential underlying sandwich structure of an unknown system. Because other types of cascade may show a similar property, this will allow us to exclude an LNL sandwich structure or to make it likely that we are dealing with one.

24.6 SINUSOIDAL SIGNALS

When we use a sinusoidal signal as the input to a linear system, we get a sinusoidal signal at its output. At the output, the amplitude of the sinusoidal signal may be amplified/attenuated and the waveform may have a changed phase, but the frequencies of the input and output of an LTI system are identical! We can use this property to completely characterize an LTI system, such as the RC filter, with a set of sinusoidal inputs (see Section 15.3). Since the frequency at the output doesn't change relative to the input frequency, we can describe the LTI system by depicting the change of amplitude and phase for each frequency with a Bode plot or a Nyquist plot (see Section 17.3, Fig. 17.4).

As you may have guessed, this simple relationship between input and output frequency is not valid for nonlinear systems. Let us investigate the response of the second-order nonlinear system introduced in Eq. (24.5) by feeding it a cosine with amplitude A and angular frequency ω_0. Further, let us use Euler's relationship $\left(e^{\pm j\varphi} = \cos\varphi \pm j\sin\varphi\right)$ to express the input in terms of two complex exponentials:

$$x(t) = A\cos\omega_0 t = \underbrace{\frac{A}{2}e^{j\omega_0 t}}_{x_1(t)} + \underbrace{\frac{A}{2}e^{-j\omega_0 t}}_{x_2(t)} \tag{24.26}$$

Note that the two components of input x (x_1 and x_2) are complex conjugates. Now we can treat this input as we did in Section 24.2.1 and repeat the result from Eq. (24.10) for the system's output y:

$$y(t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)[x_1(t-\tau_1) + x_2(t-\tau_1)][x_1(t-\tau_2) + x_2(t-\tau_2)]d\tau_1 d\tau_2 \tag{24.27}$$

In short notation we can write:

$$y(t) = H_2[x_1(t)] + H_2[x_2(t)] + H_2[x_1(t), x_2(t)] + H_2[x_2(t), x_1(t)] \quad (24.28)$$

The only difference between Eq. (24.28) and Eq. (24.12) we obtained in Section 24.2.1, is that we didn't use the symmetry property $H_2[x_1(t), x_2(t)] = H_2[x_2(t), x_1(t)]$. Let us then evaluate each of the four terms in Eq. (24.28). The first term is:

$$\begin{aligned} H_2[x_1(t)] &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1, \tau_2) x_1(t-\tau_1) x_1(t-\tau_2) d\tau_1 d\tau_2 \\ &= \left(\frac{A}{2}\right)^2 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1, \tau_2) e^{j\omega_0(t-\tau_1)} e^{j\omega_0(t-\tau_2)} d\tau_1 d\tau_2 \end{aligned} \quad (24.29)$$

Combining both exponential expressions we get:

$$\begin{aligned} &\left(\frac{A}{2}\right)^2 e^{j2\omega_0 t} \underbrace{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1, \tau_2) e^{-j\omega_0\tau_1} e^{-j\omega_0\tau_2} d\tau_1 d\tau_2}_{\Psi} \\ &= \left(\frac{A}{2}\right)^2 e^{j2\omega_0 t} \Psi(-j\omega_0, -j\omega_0) \end{aligned} \quad (24.30a)$$

Here we use the variable Ψ to symbolize the double integral, a complex function of ω_0.

> *Note*: Comparing the function Ψ above (symbolizing the double integral) with Eq. (6.4) and Eq. (7.9), it can be seen that the expression is the 2-D Fourier transform of the second-order kernel.

Similarly, substituting the exponential expression for x_2, the second term $H_2[x_2(t)]$ in Eq. (24.28) becomes:

$$\left(\frac{A}{2}\right)^2 e^{-j2\omega_0 t} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1, \tau_2) e^{j\omega_0\tau_1} e^{j\omega_0\tau_2} d\tau_1 d\tau_2 = \left(\frac{A}{2}\right)^2 e^{-j2\omega_0 t} \Psi(j\omega_0, j\omega_0) \quad (24.30b)$$

Note that both Eqs. (24.30a) and (24.30b) include an exponent in which the frequency is doubled ($2\omega_0$ instead of ω_0).

The third term in Eq. (24.28) is:

$$H_2[x_1(t), x_2(t)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x_1(t-\tau_1)x_2(t-\tau_2)d\tau_1 d\tau_2$$
$$= \left(\frac{A}{2}\right)^2 \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)e^{j\omega_0(t-\tau_1)}e^{-j\omega_0(t-\tau_2)}d\tau_1 d\tau_2 \quad (24.31)$$

Combining the exponentials in the expression above, we get:

$$\left(\frac{A}{2}\right)^2 \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)e^{-j\omega_0\tau_1}e^{j\omega_0\tau_2}d\tau_1 d\tau_2 = \left(\frac{A}{2}\right)^2 \Psi(-j\omega_0, j\omega_0) \quad (24.32a)$$

Using the same approach the fourth term becomes:

$$\left(\frac{A}{2}\right)^2 \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)e^{j\omega_0\tau_1}e^{-j\omega_0\tau_2}d\tau_1 d\tau_2 = \left(\frac{A}{2}\right)^2 \Psi(j\omega_0, -j\omega_0) \quad (24.32b)$$

Substituting the results for all four terms obtained in (24.30a,b) and (24.32a,b) into Eq. (24.28) we now have:

$$y(t) = \left[\left(\frac{A}{2}\right)^2 e^{j2\omega_0 t}\Psi(-j\omega_0, -j\omega_0) + \left(\frac{A}{2}\right)^2 e^{-j2\omega_0 t}\Psi(j\omega_0, j\omega_0)\right]$$
$$+ \left[\left(\frac{A}{2}\right)^2 \Psi(-j\omega_0, j\omega_0) + \left(\frac{A}{2}\right)^2 \Psi(j\omega_0, -j\omega_0)\right] \quad (24.33)$$

It can be seen that the first two terms and the second two terms (grouped by brackets) are the complex conjugates of each other. Therefore, we may conclude that the expression in Eq. (24.33) is real since the sum of two complex conjugates is real (the sum of imaginary numbers $a + jb$ and $a - jb$ is $2a$). Consequently we get the following result:

$$y(t) = 2\left(\frac{A}{2}\right)^2 \mathrm{Re}\left(e^{j2\omega_0 t}\Psi(-j\omega_0, -j\omega_0)\right) + 2\left(\frac{A}{2}\right)^2 \mathrm{Re}(\Psi(-j\omega_0, j\omega_0)) \quad (24.34)$$

In which Re(…) denotes the real component. Using Euler's relationship again, we can see that the output contains a sinusoid:

$$\begin{aligned} y(t) = 2\left(\frac{A}{2}\right)^2 \mathrm{Re}[(\cos 2\omega_0 t + j \sin 2\omega_0 t)\Psi(-j\omega_0, -j\omega_0)] \\ + 2\left(\frac{A}{2}\right)^2 \mathrm{Re}[\Psi(-j\omega_0, j\omega_0)] \end{aligned} \tag{24.35}$$

The output contains a constant (the second term in Eq. 24.35) and a sinusoid with a frequency $2\omega_0$ (the first term).

The expression in Eq. (24.35) is an important result for the analysis of higher-order systems: a certain frequency at the system's input may result in a higher-frequency component at the output. **When we digitize the output of a higher-order system as the result of some input signal it is important to estimate the highest frequency at the output to avoid aliasing** (Section 2.2.2). With a linear system, this problem does not occur; the highest frequency of the input is the highest frequency possible at the output. But with nonlinear systems, the maximum output frequency may be a multiple (as shown above, in a second-order system it is a factor of two, and in an nth-order system it is a factor of n) of the input's highest frequency value. A practical approach here is to first sample the system's output at a much higher sample rate than would be used routinely (one to a few orders of magnitude higher) and then compute a power spectrum to estimate the highest-frequency component. The outcome of this preliminary experiment can be used to establish an appropriate sample rate.

EXERCISES

24.1 Show that the response to $C\delta(t)$ of a third-order system obeying

$$\begin{aligned} y(t) &= H_3[x(t)] \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_3(\tau_1, \tau_2, \tau_3) x(t-\tau_1) x(t-\tau_2) x(t-\tau_3) d\tau_1 d\tau_2 d\tau_3 \end{aligned}$$

scales with C^3.

24.2 Analyze the deviation from superposition in this third-order Volterra system.

24.3 System specification.
a. Why does the UIR fully specify an LTI system and not an NLTI system?
b. What function does specify the NLTI system?

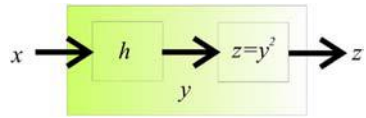


FIGURE E24.5 Diagram of a nonlinear system with input x and output z.

24.4 Show that an LTI system's response to a sine wave of frequency ω_0 is also a sine wave with the same frequency (follow the procedure used in Section 24.6).

24.5 Consider a nonlinear system consisting of a cascade of a linear filter and a nonlinear squarer, such as the one depicted in Fig. E24.5.
The input x and output y of the linear filter are characterized by the following difference equation:

$$y(n) = x(n) + 2x(n-1) + 4x(n-2) + 2x(n-3) + x(n-4) + 0.5x(n-5) + 0.25x(n-6) \quad (1)$$

The squarer is a static system, its input y and output z are determined by:

$$z = y^2 \quad (2)$$

Use MATLAB® to determine/compute and plot the following.
a. The UIR h of the filter in Fig. E24.5.
b. The second-order Volterra kernel of the cascade based on your knowledge of the system (that is Eqs. 1 and 2).
c. The second-order Volterra kernel of the cascade based on the deviation of superposition.
d. Interpret your findings.
(Suggestion: use a modification of an existing MATLAB® code.)

References

Izhikevich, E.M., 2007. Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting. MIT Press, Cambridge, MA.

Lopes da Silva, F.H., Hoeks, A., Smits, H., Zetterberg, L.H., 1974. Model of brain rhythmic activity: the alpha-rhythm of the thalamus. Kybernetik 15, 27–37.

Schetzen, M., 2006. The Volterra & Wiener Theories of Nonlinear Systems, second reprint ed. Krieger Publishing Company, Malabar, FL.

CHAPTER

25

Wiener Series

25.1 INTRODUCTION

Determining the Volterra kernels of an unknown system faces several practical problems: (1) the order of the system underlying the signal being investigated is usually unknown, and (2) the contributions of the individual components (of different order) of the Volterra series are not independent. The first problem is generally an issue if one wants to characterize a system with any type of series approximation, and the second problem can sometimes be resolved by the use of a series with orthogonal components. An example of the latter procedure is the development of the Fourier series; by having orthogonal terms, there are no dependencies between terms and one can determine the coefficients a_i and b_i of the Fourier series sequentially (Chapter 5). To address the dependence between components in a series approximation for nonlinear systems, **Norbert Wiener developed an approach where each component in his Volterra-like series is orthogonal to all lower-order ones**. Although within the Wiener series approach one cannot can not predetermine the order of the system being studied either (problem (1) above), the orthogonality between the terms in the series allows one to determine the Wiener kernels sequentially. Subsequently one can determine their contribution to the signal, and stop the kernel-estimation process at the order where the signal is sufficiently approximated. For practical reasons most studies limit their kernel estimates at either the second-order or (less often) at the third order. The third-order and higher-order kernels require significant computation and are difficult to depict.

In this chapter we will first discuss the main differences between the Wiener and Volterra series, after which we will derive the expressions for

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00025-4

the zeroth-, first-, and second-order Wiener kernels, and then finally we will discuss practical methods for determining Wiener kernels for simulated and recorded time series. Applications of these methods will be presented in the form of MATLAB® scripts. The last part of this chapter and Fig. 25.7 summarize the mathematical procedures we use to determine the Wiener series. For an extended background on this topic, see Marmarelis and Marmarelis (1978), the reprint of Schetzen's book (Schetzen, 2006), and for recent engineering-oriented overviews see Westwick and Kearney (2003) and Marmarelis (2004).

25.2 WIENER KERNELS

Similar to the Volterra series, the Wiener series characterizes the output z of a nonlinear system as the sum of a set of operators G_n dependent on kernels k_n and input x:

$$z(t) = \sum_{n=0}^{N} G_n[k_n; x(t)]$$

This equation is similar to those for the Volterra series (Eqs. 24.9a and 24.9b) but although there are many similarities between Volterra and Wiener series, there are also a few crucial differences that allow Wiener operators to be mutually independent. For clarity we first summarize the **three major differences** between the Volterra and Wiener series and then explain further details in the remainder of this chapter (e.g., the exact relationship between Volterra and Wiener kernels is discussed in Section 25.5).

1. Although a Volterra series usually does not include a zeroth-order term (see Chapter 24, Eq. 24.9), we may define such a term as a constant:

$$H_0[x(t)] = h_0 \tag{25.1a}$$

   In contrast, the Wiener series **always** includes a zeroth-order term. This term is defined as the average output (the DC component) equal to k_0:

$$\boxed{G_0[k_0; x(t)] = k_0} \tag{25.1b}$$

   In this equation, k_0 is the zeroth-order Wiener kernel. We use k_n for the Wiener kernels to distinguish them from the Volterra kernels h_n.
2. While individual Volterra operators are homogeneous (see, for example, Eq. 24.6 for a second-order one), that is $H_n[cx(t)] = c^n H_n[x(t)]$, the **Wiener operators are nonhomogeneous**—for

example, the first-order Wiener operator has a first-order and a derived zeroth-order component:

$$G_1[k_1; x(t)] = g_1\left[k_1, k_{0(1)}; x(t)\right] = K_1[x(t)] + K_{0(1)}[x(t)]$$
$$= \int_{-\infty}^{\infty} k_1(\tau_1)x(t-\tau_1)\, d\tau_1 + k_{0(1)} \tag{25.2}$$

The subscript 0(1) in $K_{0(1)}$ and $k_{0(1)}$ indicates that these are zeroth-order members of a first-order nonhomogeneous operator. Specifically, k_1 is the first-order **Wiener kernel** and $k_{0(1)}$ is the so-called **derived-Wiener-kernel** from operator G_1. In general, the Wiener kernels of the type $k_{n(m)}$ with $n < m$ are called derived-Wiener-kernels because, as we will see below, they must be derived from Wiener kernel k_m. The notation with the capital G indicates that the operator includes both the kernel and input, while the lowercase notation g differs by explicitly also indicating all of the derived-kernels.

The second-order Wiener operator is:

$$G_2[k_2; x(t)] = g_2\left[k_2, k_{1(2)}, k_{0(2)}; x(t)\right] = K_2[x(t)] + K_{1(2)}[x(t)] + K_{0(2)}[x(t)]$$
$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)\, d\tau_1\, d\tau_2 + \int_{-\infty}^{\infty} k_{1(2)}(\tau_1)x(t-\tau_1)\, d\tau_1 + k_{0(2)} \tag{25.3}$$

The subscript 0(2) and 1(2) indicate that these are zeroth-order and first-order members (derived-Wiener-kernels) of the second-order nonhomogeneous operator G_2, respectively. Kernel k_2 is the second-order Wiener kernel, while $k_{1(2)}$ and $k_{0(2)}$ are derived-Wiener-kernels from operator G_2. As we will demonstrate below, the rationale for using nonhomogeneous operators relates to their orthogonality.

3. In the case of a Wiener series we use a special input signal usually in the form of zero mean Gaussian White Noise (GWN) (alternative input signals are discussed in Section 25.7 and Chapter 26). Selection of a special input is critical because it allows us to create a series in which the **operators are orthogonal (uncorrelated) to the lower-order operators.** As we will see in Section 25.3, this property creates independence between the operators in the series, which will allow us to determine the Wiener kernels sequentially without having to worry about dependency issues.

The first-order Wiener operator is defined so that it is orthogonal to the zeroth-order Volterra operator:

$$E\left\{H_0[x(t)]g_1\left[k_1, k_{0(1)}; x(t)\right]\right\} = \left\langle H_0[x(t)]g_1\left[k_1, k_{0(1)}; x(t)\right]\right\rangle = 0 \quad (25.4a)$$

In the expression after the equal sign, $\langle \ldots \rangle$ indicates the time average.

Note: $\langle x(t) \rangle$ represents the time average of a signal $x(t)$ over a time interval T. This is an alternative notation for the integral notation: $\frac{1}{T}\int_0^T x(t)\, dt$.

Eq. (25.4a) indicates that we assumed ergodicity so that we may use a time average $\left\langle H_0[x(t)]g_1\left[k_1, k_{0(1)}; x(t)\right]\right\rangle$ to determine the expectation $E\{\ldots\}$ of the product of H_0 and g_1. If you need to review the concepts of expectation and time averages, see Section 3.2 and Appendix 3.1. Details about time averages for GWN are reviewed at the end of this chapter in Appendix 25.1.

Similarly the second-order Wiener operator is defined as orthogonal to zeroth- and first-order Volterra operators:

$$\left\langle H_0[x(t)]g_2\left[k_2, k_{1(2)}, k_{0(2)}; x(t)\right]\right\rangle = 0, \text{ and} \quad (25.4b)$$

$$\left\langle H_1[x(t)]\ g_2\left[k_2, k_{1(2)}, k_{0(2)};\ x(t)\right]\ \right\rangle = 0 \quad (25.4c)$$

To characterize any nonlinear system of order N, this approach is generalized for all Wiener operators; that is, for zero mean GWN input, operator $G_n[k_n; x(t)] = g_n\left[k_n, k_{n-1(n)}, \ldots, k_{0(n)}; x(t)\right]$ is defined such that it is orthogonal to ***any*** Volterra operator of a lower order:

$$\left\langle H_m[x(t)]g_n\left[k_n, k_{n-1(n)}, \ldots, k_{0(n)};\ x(t)\right]\ \right\rangle = 0 \quad \text{for } m < n \quad (25.4d)$$

In the following sections we will achieve orthogonality between the G operators and their lower-order Volterra operators by using the approach of the so-called Gram–Schmidt technique as illustrated for G_2 and H_0 in Fig. 25.1. This diagram illustrates why the Wiener kernel has associated lower-order derived-Wiener-kernels—the combined kernel and derived-kernel accomplishes the orthogonality between the Wiener operator and lower-order operators.

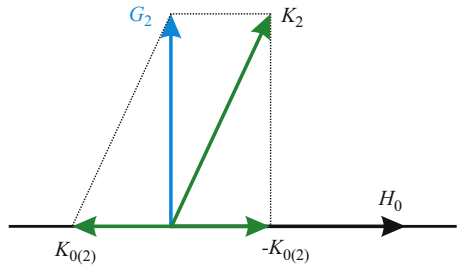


FIGURE 25.1 Illustration of the procedure to achieve orthogonality between the G operators and their lower-order Volterra operators. The approach of the so-called Gram–Schmidt technique applied to G_2 and H_0. Note that the second-order Wiener operator G_2 is the sum of K_2 and $K_{0(2)}$ (see also Eq. 25.13), the derived component $K_{0(2)}$ ensures orthogonality with H_0.

For further details of the Gram–Schmidt technique, see for example Arfken and Weber (2005). By defining the Wiener kernels according to this technique, we can determine the kernels of nonlinear systems from lower to higher order without knowledge of the system's components. This procedure is similar to approximating a function or signal with a Fourier series (Chapter 5) or a polynomial (Section 11.4). For each kernel (for each order) we can determine its contribution to the output and we can continue to add higher-order terms until we are satisfied with our approximation of the system at hand. The procedure for determining the Wiener kernels as sketched above and their independence from lower-order kernels is summarized in the diagram in Fig. 25.2. In the following sections we derive the expressions for the first- and second-order Wiener kernels. If you are interested in higher-order components see Schetzen (2006).

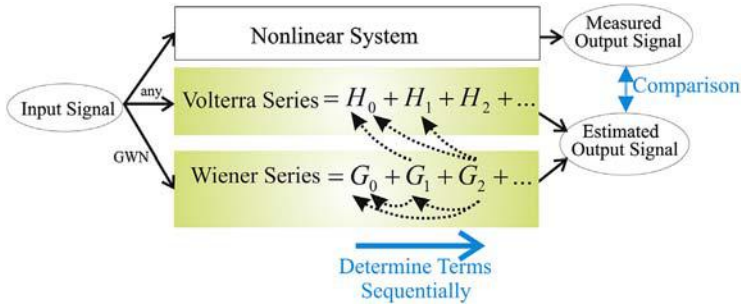


FIGURE 25.2 Diagram of the representation of a nonlinear system by Volterra and Wiener series. In contrast to the Volterra operators H_n, the operators G_n in the Wiener series are independent from lower-order operators, symbolized by the *stippled arrows*. This allows one to determine the Wiener operators and their kernels sequentially and compute their contribution to the estimated output. The comparison with the measured output signal can be used to determine at what order the output is sufficiently approximated. *GWN*, Gaussian white noise.

25.2.1 Derivation of the First-Order Wiener Operator

In the previous section we identified the zeroth-order kernel as the signal's DC component in Eq. (25.1b). Now we can use orthogonality defined in Eq. (25.4) to derive the Wiener kernels k_1 and k_2. Starting with the first-order kernel, we substitute Eqs. (25.1a) and (25.2) into Eq. (25.4a) and find that the following condition must be satisfied

$$\left\langle H_0[x(t)]g_1\left[k_1, k_{0(1)}; x(t)\right]\right\rangle = 0 \rightarrow$$

$$\left\langle h_0\left[\int_{-\infty}^{\infty} k_1(\tau_1)x(t-\tau_1)\,d\tau_1 + k_{0(1)}\right]\right\rangle = h_0\left[\int_{-\infty}^{\infty} k_1(\tau_1)\langle x(t-\tau_1)\rangle\,d\tau_1 + k_{0(1)}\right] = 0 \tag{25.5}$$

Note that in the expression after the equal sign we took all constants $\left(h_0,\ k_1(\tau_1),\ k_{0(1)}\right)$ out of the time average operation, such that only the (time-dependent) input time series x remains within the time average brackets $\langle\ldots\rangle$. **Now you will see how convenient it is to have zero mean GWN as input.** Because input x is zero mean GWN, the time average $\langle x(t-\tau_1)\rangle$ is zero. Therefore, the integral in Eq. (25.5) evaluates to zero and (since h_0 is not necessarily zero) we find that the orthogonality condition in Eq. (25.5) is satisfied when

$$\boxed{k_{0(1)} = 0} \tag{25.6}$$

Combining this result with Eq. (25.2), we find that the first-order Wiener operator is

$$\boxed{G_1[k_1; x(t)] = g_1[k_1; x(t)] = \int_{-\infty}^{\infty} k_1(\tau_1)x(t-\tau_1)\,d\tau_1} \tag{25.7}$$

Note that for a first-order system (without a DC component) k_1 is the unit impulse response (Chapter 13). Further, if the input x is zero mean GWN, the output of the first-order (linear) operator G_1 will also be zero mean GWN. Because $\langle x(t-\tau_1)\rangle = 0$, the expectation or time-average of G_1 is zero: that is, $E\{G_1\} = \langle G_1\rangle = 0$. Therefore G_1 is indeed orthogonal to any constant, such as zero-order operators G_0 and H_0.

25.2.2 Derivation of the Second-Order Wiener Operator

We can obtain the expression for the second-order operator g_2 using a procedure similar to the one we developed for the first-order one. Here

we must deal separately with the independence between the second-order Wiener operator and the two lower-order Volterra operators H_0 and H_1, respectively.

25.2.2.1 *Orthogonality Between* $\mathbf{H_0}$ *and* $\mathbf{g_2}$

Using Eqs. (25.1a), (25.3), and (25.4b) we get

$$\left\langle H_0[x(t)]g_2\left[k_2, k_{1(2)}, k_{0(2)}; x(t)\right]\right\rangle$$

$$= \left\langle h_0\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)\,d\tau_1\,d\tau_2 + \int_{-\infty}^{\infty} k_{1(2)}(\tau_1)x(t-\tau_1)\,d\tau_1 + k_{0(2)}\right]\right\rangle = 0$$

$$= h_0\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\langle x(t-\tau_1)x(t-\tau_2)\rangle\,d\tau_1\,d\tau_2 + \int_{-\infty}^{\infty} k_{1(2)}(\tau_1)\langle x(t-\tau_1)\rangle d\tau_1 + k_{0(2)}\right] = 0 \tag{25.8}$$

As we did in Eq. (25.5), we took the constants out of the time average $\langle\ldots\rangle$ such that only the time series x remains within it. Because the input is zero mean GWN with variance σ^2, the average $\langle x(t-\tau_1)\rangle = 0$ and the averaged product of both copies of input x is the autocorrelation R_{xx} (see Section 13.4.1):

$$\langle x(t-\tau_1)x(t-\tau_2)\rangle = R_{xx}(\tau_2-\tau_1) = \sigma^2\delta(\tau_2-\tau_1).$$

Again we can see how convenient the zero mean GWN input is: the time average $\langle x(t-\tau_1)\rangle$ vanishes and time average $\langle x(t-\tau_1)x(t-\tau_2)\rangle$ can be simplified to the expression for the autocorrelation (see also Appendix 25.1 for further details on averages of products of Gaussian variables.). Therefore Eq. (25.8) becomes

$$h_0\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\underbrace{\langle x(t-\tau_1)x(t-\tau_2)\rangle}_{R_{xx}(\tau_2-\tau_1)}\,d\tau_1\,d\tau_2 + h_0k_{0(2)}$$

$$= \sigma^2h_0\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\delta(\tau_2-\tau_1)\,d\tau_1\,d\tau_2 + h_0k_{0(2)}$$

The double integral on the right-hand side can be evaluated by using the sifting property while evaluating the integral for one of the time constants; here we integrate with respect to τ_2 and get

$$\sigma^2 h_0\int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\,d\tau_1 + h_0\,k_{0(2)} = 0 \;\rightarrow\; \boxed{k_{0(2)} = -\sigma^2\int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\,d\tau_1} \tag{25.9}$$

In this expression we can see that $k_{0(2)}$ is indeed a derived-Wiener-kernel because it is directly derived from Wiener kernel k_2.

25.2.2.2 *Orthogonality Between* $\mathbf{H_1}$ *and* $\mathbf{g_2}$

Subsequently we substitute an expression for the first-order Volterra operator (see Eq. 25.7) and Eq. (25.3) for the second-order Wiener operator in the orthogonality condition in Eq. (25.4c):

$$\left\langle H_1[x(t)] g_2\left[k_2, k_{1(2)}, k_{0(2)}; x(t)\right]\right\rangle \tag{25.10}$$

$$= \left\langle \left[\int_{-\infty}^{\infty} h_1(v)x(t-v)dv\right]\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)\,d\tau_1\,d\tau_2 + \int_{-\infty}^{\infty} k_{1(2)}(\tau_1)\,x(t-\tau_1)\,d\tau_1 + k_{0(2)}\right]\right\rangle$$

The above expression contains three terms. We will first show that the first and third terms always evaluate to zero if the input is zero mean GWN.

The **first** term

$$\left\langle \left[\int_{-\infty}^{\infty} h_1(v)x(t-v)\,dv\right]\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)\,d\tau_1\,d\tau_2\right]\right\rangle$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_1(v)k_2(\tau_1,\tau_2)\langle x(t-v)x(t-\tau_1)x(t-\tau_2)\rangle\,dv\,d\tau_1\,d\tau_2 = 0 \tag{25.11a}$$

evaluates to zero because of our choice of zero mean GWN as input and the odd product $\langle x(t-v)x(t-\tau_1)x(t-\tau_2)\rangle = 0$ (Appendix 25.1). **Again, taking advantage of our choice of GWN as the input.**

The **third** term in Eq. (25.10)

$$\left[\int_{-\infty}^{\infty} h_1(v)\langle x(t-v)\rangle\,dv\right] k_{0(2)} = 0 \tag{25.11b}$$

also evaluates to zero because $\langle x(t-v)\rangle = 0$.

The **second** term in Eq. (25.10) is:

$$\left\langle \left[\int_{-\infty}^{\infty} h_1(v)x(t-v)\,dv \right] \left[\int_{-\infty}^{\infty} k_{1(2)}(\tau_1)x(t-\tau_1)d\tau_1 \right] \right\rangle \tag{25.11c}$$
$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_1(v)k_{1(2)}(\tau_1)\langle x(t-v)x(t-\tau_1)\rangle\,dv\,d\tau_1$$

This second term is the only one that contains an even product of $x(t)$ and can be further evaluated using (again) the autocorrelation R_{xx} for the zero mean GWN with variance σ^2; that is, $\langle x(t-v)x(t-\tau_1)\rangle = R_{xx}(\tau_1 - v) = \sigma^2\delta(\tau_1 - v)$. This gives us

$$\sigma^2 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_1(v)k_{1(2)}(\tau_1)\delta(\tau_1 - v)\,dv\,d\tau_1 = \sigma^2 \int_{-\infty}^{\infty} h_1(\tau_1)k_{1(2)}(\tau_1)\,d\tau_1$$

In the above we evaluate the integral with respect to v by using the sifting property. Because the first and third terms already evaluate to zero, the second term must be zero in order to satisfy the orthogonality condition in Eq. (25.4c). We accomplish this by setting

$$k_{1(2)} = 0 \tag{25.12}$$

Substituting the results we obtained from the orthogonality conditions (in Eqs. 25.9 and 25.12) into Eq. (25.3) we find the second-order Wiener operator G_2,

$$\boxed{\begin{aligned} G_2[k_2;x(t)] &= g_2\left[k_2, k_{1(2)}, k_{0(2)}; x(t)\right] = K_2[x(t)] + K_{0(2)}[x(t)] \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)\,d\tau_1\,d\tau_2 - \underbrace{\sigma^2 \int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\,d\tau_1}_{k_{0(2)}} \end{aligned}} \tag{25.13}$$

Note that just as the expectation for G_1 is zero, the expected output of G_2 is also zero:

$$E\left\{ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)\,d\tau_1\,d\tau_2 - \sigma^2 \int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\,d\tau_1 \right\}$$
$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)E\{x(t-\tau_1)x(t-\tau_2)\}\,d\tau_1\,d\tau_2 - \sigma^2 \int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\,d\tau_1 \tag{25.14}$$

As for the time average of an even product of Gaussian variables, once again, the expectation is the autocorrelation: $E\{x(t-\tau_1)x(t-\tau_2)\} = \sigma^2\delta(\tau_1-\tau_2)$. Substituting this into Eq. (25.14) gives

$$\begin{aligned}&\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\sigma^2\delta(\tau_1-\tau_2)\,d\tau_1\,d\tau_2 - \sigma^2\int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\,d\tau_1\\ &= \sigma^2\int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\,d\tau_1 - \sigma^2\int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\,d\tau_1 = 0\end{aligned} \tag{25.15}$$

Here we evaluated the term with the double integral using the sifting property. Because the expectation is zero, G_2 (just as G_1) is orthogonal to any constant including G_0. Even without knowledge about our derivation above, using the same approach, it is straightforward to show that operator G_2 is designed to also be orthogonal to G_1. This orthogonality can be evaluated via the expectation of the product $E\{G_1G_2\}$, which contains odd products of the random input variable $x(t)$. **The odd products evaluate to zero (Appendix 25.1), causing the expectation to vanish.**

In the above we showed that the first-order Wiener operator is orthogonal to the zeroth-order one, and that the second-order operator is orthogonal to the first- and zeroth-order ones. We will not elaborate on this here, but, in general, the Wiener operators are constructed in a way that they are orthogonal to all lower-order ones. Higher-order Wiener kernels will not be derived here but the derivation follows a similar procedure as described for the zeroth–second-order kernels above. Details for these derivations can be found in Schetzen (2006).

25.3 DETERMINATION OF THE ZEROTH-, FIRST-, AND SECOND-ORDER WIENER KERNELS

Now that we know how the expressions for the terms in the Wiener series are developed, it is time to examine how we might determine the terms from measured and simulated data sets. Recall also that we can determine the kernels sequentially because of the orthogonality property. The best-known method to establish Wiener kernels from measurements is the cross-correlation method first described by Lee and Schetzen (1965). If we deal with a nonlinear system of order N, and we present a zero mean GWN x at its input, we obtain output z as the sum of the Wiener operators G_n:

$$z(t) = \sum_{n=0}^{N} G_n[k_n; x(t)] \tag{25.16}$$

25.3.1 Determination of the Zeroth-Order Wiener Kernel

As we extend our results for the first- and second-order operators to all higher-order ones, we find that the expectation of all Wiener operators G_n, except the zeroth-order operator G_0, is zero (see the last paragraphs in Sections 25.2.1 and 25.2.2.2). Therefore, assuming an ergodic process (allowing the use of time averages for estimating expectations), we find that the average of output signal z is

$$\boxed{\langle z(t)\rangle = \sum_{n=0}^{N}\langle G_n[k_n; x(t)]\rangle = G_0[k_0; x(t)] = k_0} \tag{25.17}$$

Thus the zeroth-order Wiener kernel is obtained from the mean output, i.e., the output's DC component.

25.3.2 Determination of the First-Order Wiener Kernel

Here we show that we can get the first-order Wiener kernel of a system from the cross-correlation between its input x and output z

$$\begin{aligned}\langle z(t)x(t-v_1)\rangle &= \langle G_0[k_0; x(t)]x(t-v_1)\rangle + \langle G_1[k_1; x(t)]x(t-v_1)\rangle \\ &+ \langle G_2[k_2; x(t)]x(t-v_1)\rangle + \cdots = \sum_{n=0}^{N}\langle G_n[k_n; x(t)]x(t-v_1)\rangle\end{aligned} \tag{25.18}$$

Recall that Wiener operators are defined to be orthogonal to lower-order Volterra operators (Fig. 25.2). This property may be generalized to **all** lower-order Volterra kernels. Since the delayed signal $x(t-v_1)$ can be presented as a first-order Volterra operator (Appendix 25.2), all Wiener operators G_n with $n \geq 2$ are orthogonal to $x(t-v_1)$ according to Eq. (25.4d). Let's check this property by examining the outcome for $\langle z(t)x(t-v_1)\rangle$ by determining the outcome for the individual operators G_0, G_1, G_2, …

Using Eq. (25.1b) for $\boldsymbol{n = 0}$:

$$\langle G_0[k_0; x(t)]x(t-v_1)\rangle = k_0 \underbrace{\langle x(t-v_1)\rangle}_{0} = 0 \tag{25.19a}$$

Using Eq. (25.2) for $n = 1$:

$$\langle G_1[k_1; x(t)]x(t-v_1)\rangle = \int_{-\infty}^{\infty} k_1(\tau_1)\underbrace{\langle x(t-\tau_1)x(t-v_1)\rangle}_{\sigma^2\delta(\tau_1-v_1)}\, d\tau_1$$

$$= \sigma^2 \int_{-\infty}^{\infty} k_1(\tau_1)\,\delta(\tau_1 - v_1)\, d\tau_1 = \sigma^2 k_1(v_1) \tag{25.19b}$$

For the second-order kernel we already know that $\langle G_2[k_2; x(t)]x(t-v_1)\rangle$ is zero because $x(t-v_1)$ can be considered a lower-order Volterra operator (Appendix 25.2). However, let's check the outcome of the cross-correlation anyway.

Using Eq. (25.3) for $n = 2$:

$$\begin{aligned}&\langle G_2[k_2; x(t)]x(t-v_1)\rangle \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\underbrace{\langle x(t-\tau_1)\,x(t-\tau_2)\;\; x(t-v_1)\rangle}_{0}\, d\tau_1\, d\tau_2 \\ &- \sigma^2 \int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\underbrace{\langle x(t-v_1)\rangle}_{0}\, d\tau_1 = 0\end{aligned} \tag{25.19c}$$

The above integrals evaluate to zero because they contain the time average of odd products of the GWN input x. We now state (without further checking) that the remaining averaged products ($\langle G_n[k_n; x(t)]\, x(t-v_1)\rangle$, $n \geq 3$) are zero by using the property of Eq. (25.4d). From the three expressions in Eqs. (25.19a)–(25.19c) we conclude that only the term for $n = 1$ is nonzero, therefore we can determine the first-order Wiener kernel by combining Eqs. (25.18) and (25.19b):

$$\langle z(t)x(t-v_1)\rangle = \sigma^2 k_1(v_1) \;\rightarrow\; \boxed{k_1(v_1) = \frac{1}{\sigma^2}\langle z(t)\,x(t-v_1)\rangle} \tag{25.20}$$

Thus the first-order Wiener kernel can be found by the cross-correlation between input x and output z weighted by the variance of the input.

25.3.3 Determination of the Second-Order Wiener Kernel

Using an analogous procedure for the higher-order Wiener kernels, we can find the second-order kernel by using a (second-order) cross-correlation between output z and now two copies of input x:

$$\begin{aligned}\langle z(t)x(t-v_1)x(t-v_2)\rangle &= \langle G_0[k_0;x(t)]x(t-v_1)x(t-v_2)\rangle \\ &+ \langle G_1[k_1;x(t)]x(t-v_1)x(t-v_2)\rangle + \langle G_2[k_2;x(t)]x(t-v_1)x(t-v_2)\rangle + \cdots \\ &= \sum_{n=0}^{N}\langle G_n[k_n;x(t)]x(t-v_1)x(t-v_2)\rangle\end{aligned} \tag{25.21}$$

Because $x(t-v_1)x(t-v_2)$ can be presented as a second-order Volterra operator (Appendix 25.2), all Wiener operators G_n with $n \geq 3$ are orthogonal to $x(t-v_1)x(t-v_2)$ according to Eq. (25.4d).

Using Eq. (25.1b) for $\boldsymbol{n=0}$:

$$\langle G_0[k_0;x(t)]\,x(t-v_1)x(t-v_2)\rangle = k_0\underbrace{\langle x(t-v_1)x(t-v_2)\rangle}_{\sigma^2\delta(v_1-v_2)} = k_0\,\sigma^2\delta(v_1-v_2) \tag{25.22a}$$

Using Eq. (25.7) for $\boldsymbol{n=1}$:

$$\langle G_1[k_1;x(t)]x(t-v_1)x(t-v_2)\rangle = \int_{-\infty}^{\infty} k_1(\tau_1)\underbrace{\langle x(t-\tau_1)x(t-v_1)x(t-v_2)\rangle}_{0}\,d\tau_1 = 0 \tag{25.22b}$$

Using Eq. (25.13) for $\boldsymbol{n=2}$:

$$\begin{aligned}&\langle G_2[k_2;x(t)]x(t-v_1)x(t-v_2)\rangle \\ &= \overbrace{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\underbrace{\langle x(t-\tau_1)\,x(t-\tau_2)\;x(t-v_1)x(t-v_2)\rangle}_{A}\,d\tau_1\,d\tau_2}^{I} \\ &- \overbrace{\sigma^2\int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\underbrace{\langle x(t-v_1)x(t-v_2)\rangle}_{\sigma^2\delta(v_1-v_2)}\,d\tau_1}^{II}\end{aligned} \tag{25.22c}$$

Using Wick's theorem (a theorem that relates higher-order moments to lower-order ones; Appendix 25.1), the average indicated by A in Eq. (25.22c) (fourth-order correlation) can be written as:

$$A = \langle x(t-\tau_1)x(t-\tau_2)x(t-v_1)x(t-v_2)\rangle = \underbrace{\langle x(t-\tau_1)x(t-\tau_2)\rangle}_{\sigma^2\delta(\tau_1-\tau_2)}\ \underbrace{\langle x(t-v_1)x(t-v_2)\rangle}_{\sigma^2\delta(v_1-v_2)}$$
$$+\underbrace{\langle x(t-\tau_1)x(t-v_1)\rangle}_{\sigma^2\delta(\tau_1-v_1)}\ \underbrace{\langle x(t-\tau_2)x(t-v_2)\rangle}_{\sigma^2\delta(\tau_2-v_2)} + \underbrace{\langle x(t-\tau_1)x(t-v_2)\rangle}_{\sigma^2\delta(\tau_1-v_2)}\ \underbrace{\langle x(t-\tau_2)x(t-v_1)\rangle}_{\sigma^2\delta(\tau_2-v_1)}$$

This allows us to separate Part I of the expression in Eq. (25.22c) into the following three terms:

1. $$\sigma^4\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\delta(\tau_1-\tau_2)\delta(v_1-v_2)\,d\tau_1\,d\tau_2 = \sigma^4\int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\delta(v_1-v_2)\,d\tau_1$$
2. $$\sigma^4\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\delta(\tau_1-v_1)\delta(\tau_2-v_2)\,d\tau_1\,d\tau_2 = \sigma^4 k_2(v_1,v_2)$$
3. $$\sigma^4\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} k_2(\tau_1,\tau_2)\delta(\tau_1-v_2)\delta(\tau_2-v_1)\,d\tau_1\,d\tau_2 = \sigma^4 k_2(v_2,v_1) = \sigma^4 k_2(v_1,v_2)$$

The three integrals above are evaluated using the sifting property. Furthermore, by using the same symmetry property of the Volterra kernels (Section 24.2.1), we have concluded that k_2 is symmetrical and that the terms in 2 and 3 above are identical.

Combining the results for parts 1–3 in I and the integral term II in Eq. (25.22c) we get

$$\langle G_2[k_2;x(t)]x(t-v_1)x(t-v_2)\rangle =$$
$$\overbrace{2\sigma^4 k_2(v_1,v_2)+\sigma^4\int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\delta(v_1-v_2)\,d\tau_1}^{I} \overbrace{-\sigma^2\int_{-\infty}^{\infty} k_2(\tau_1,\tau_1)\underbrace{\langle x(t-v_1)x(t-v_2)\rangle}_{\sigma^2\delta(v_1-v_2)}\,d\tau_1}^{II} \tag{25.22d}$$

The integral terms in the expression above cancel so that the final result becomes:

$$\langle G_2[k_2;x(t)]x(t-v_1)x(t-v_2)\rangle = 2\sigma^4 k_2(v_1,v_2) \tag{25.22e}$$

According to Eq. (25.4d) all Wiener operators for $n > 2$ are defined so that their contributions will be zero. This allows us to combine Eq. (25.21) with Eqs. (25.22a), (25.22b), and (25.22e):

$$\langle z(t)x(t-v_1)x(t-v_2)\rangle = k_0\sigma^2\delta(v_1-v_2) + 2\sigma^4 k_2(v_1,v_2) \tag{25.23}$$

Now we decide to ignore the case when $v_1 = v_2$ and assume that $v_1 \neq v_2$ so that $\delta(v_1 - v_2) = 0$. In this case the first term on the right-hand side of Eq. (25.23) evaluates to zero. Therefore, for the off-diagonal part ($v_1 \neq v_2$) of the second-order Wiener kernel we have:

$$\boxed{k_2(v_1,v_2) = \frac{1}{2\sigma^4}\langle z(t)x(t-v_1)x(t-v_2)\rangle \quad \text{for } v_1 \neq v_2} \tag{25.24}$$

The second-order Wiener kernel is the second-order cross-correlation between output and input weighted by $2\sigma^4$. Our trick to ignore the diagonally located terms may seem a bit strange but in practical applications, **the limitation imposed by $v_1 \neq v_2$ does not present a problem because we can compute k_2 for delays that are arbitrarily close to $v_1 = v_2$.**

25.4 IMPLEMENTATION OF THE CROSS-CORRELATION METHOD

In this section we present a practical application for finding Wiener kernels associated with a given nonlinear system (Fig. 25.3). MATLAB® implementations of this approach are in pr25_1.m and pr25_2.m. In principle we can use Eqs. (25.17), (25.20), and (25.24) to determine the Wiener kernels. However, since our input of random noise is necessarily finite, the subsequent kernels may not be exactly orthogonal. To mitigate the effects of this problem, it is common practice to determine the kernels from low to higher orders sequentially, while at each step subtracting the contribution of the lower-order kernels from the output (-1, $+$ operations in Fig. 25.3). For example, before computing k_1, it is common practice **to subtract k_0** (the DC component) from the output to obtain a zeroth-order residue v_0. This residue v_0 $(= z - k_0)$, instead of the output z, is then cross-correlated with the input to obtain the first-order kernel k_1 (recall Eq. 25.20),

$$\boxed{k_1(v_1) = \frac{1}{\sigma^2}\langle v_0(t)x(t-v_1)\rangle} \tag{25.25}$$

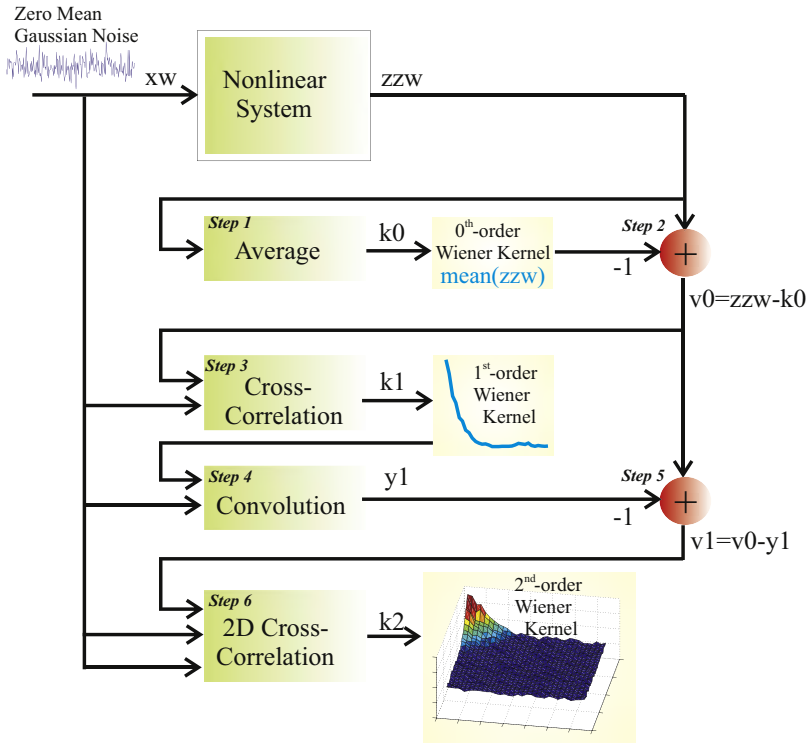


FIGURE 25.3 Diagram of the Lee–Schetzen cross-correlation method for obtaining the zeroth-, first-, and second-order Wiener kernels of a nonlinear system. Zero mean Gaussian white noise is used as the input (xw) of a nonlinear system. The average of the output (zzw) is used to estimate the zeroth-order kernel k0. The residue v0 (=zzw-k0) is then cross-correlated with the input in order to estimate the first-order kernel k1. Subsequently, the contribution y1 of the first-order kernel is determined by convolving it with the input. Finally, the residue v1 (=v0-y1) is correlated with two copies of the input (2-D cross-correlation) for the estimation of k2.

To estimate the first-order kernel's contribution (y_1) to the output, the first-order kernel k_1 is convolved with the input $x : y_1 = x \otimes k_1$. This first-order contribution y_1 is then subtracted from the zeroth-order residue v_0 to obtain the first-order residue v_1. The residue v_1 ($= z - k_0 - y_1$) is now cross-correlated with two copies of the input to estimate the second-order kernel k_2:

$$k_2(v_1, v_2) = \frac{1}{2\sigma^4} \langle v_1(t) x(t - v_1) x(t - v_2) \rangle \tag{25.26}$$

Note that, as in Eq. (25.24), the above expression is valid only for off-diagonal values with $v_1 \neq v_2$. This procedure is followed in the MATLAB® programs, and is depicted in Fig. 25.3. The input is variable xw, the output is zzw. The zeroth-and first-order residues are v0 and v1, respectively. The Wiener kernels are k0, k1, and k2.

An example of a MATLAB® implementation can be found in pr25_1.m and pr25_2.m. A snippet of the latter is shown here.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Estimation of the Wiener kernel estimation using
%%%%%%%%%% the Lee, Schetzen cross-correlation method
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% First create a set of input output using random noise
xw=randn(10000,1);              % create array with Gaussian white
                                  noise
xw=xw-mean(xw);
N=length(xw);
st=std(xw);

figure;subplot(2,1,1),plot(xcorr(xw),'k');
title('Autocorrelation of the Input Shows a Random Noise
 Characteristic');
subplot(2,1,2);hist(xw);
title('Amplitude Distribution of the Input --> Gaussian');

yw_previous1=0;
yw_previous2=0;
for n=1:length(xw);
   ywh1(n)=(A1*yw_previous1+xw(n))/(A1+1);   % the 1st order
                                                operator
   yw_previous1=ywh1(n);
   ywh2(n)=(A2*yw_previous2+xw(n))/(A2+1);   % the linear
                                                component of
                                                the 2nd
                                                order operator
   yw_previous2=ywh2(n);
   zzw(n)=ywh1(n)+ywh2(n)^2;                 % 1st order
                                                component +
                                                the squarer
end;
```

```
figure; hold;
plot(xw,'k');plot(zzw,'r')
title('Input (black) and Output (red) of a Wiener System')
xlabel('Time (ms)');ylabel('Amplitude')

%%%%%%%%%%%%%%%The Lee Schetzen Cross-correlation
 Method
% ------------------------------------------------------------------
% Step 1 (Fig. 25.3): Determine 0-order Wiener kernel
% ------------------------------------------------------------------
k0=mean(zzw)
y0=ones(1,length(xw))*k0;
% Step 2 (Fig. 25.3): Subtract k0 from the response to find residue v0
% ------------------------------------------------------------------
v0=zzw-k0;

% Step 3 (Fig. 25.3): Estimate k1 by 1st-order
%             cross-correlation of v0 and input
% ------------------------------------------------------------------

for i=0:T-1
   temp=0;
   for n=i+1:N
      temp=temp+v0(n)*xw(n-i);
   end;
   k1(i+1)=temp/(N*st^2);
end;

figure; plot(k1);
title(' 1st-order Wiener kernel')

% Step 4 (Fig. 25.3): Compute the output of the 1st-order
%             Wiener kernel using convolution
% ------------------------------------------------------------------

for n=1:N;
   temp=0;
   for i=0:min([n-1 T-1]);
      temp=temp+k1(i+1)*xw(n-i);
   end;
   y1(n)=temp;
end;
```

```
% Step 5 (Fig. 25.3): Compute the 1st-order residue
% ------------------------------------------------------------------
v1=v0-y1;

% Step 6 (Fig. 25.3): Estimate k2 by 2nd-order cross-correlation
%         of v1 with the input
% ------------------------------------------------------------------

for i=0:T-1
   for j=0:i

      temp=0;
      for n=i+1:N
         temp=temp+v1(n)*xw(n-i)*xw(n-j);
      end;

      k2(i+1,j+1)=temp/(2*N*st^4);
      k2(j+1,i+1)=k2(i+1,j+1);

   end;
end;

figure; surf(k2(1:T,1:T));

title('2nd-Order Wiener Kernel');
view(100,50);
```

The MATLAB® script pr25_2.m computes the Wiener kernels for a combined system such as the cascade discussed in Chapter 24 (Fig. 24.2C). In this example we use low-pass filters for the linear components and a squarer for the nonlinear one (Fig. 25.4).

In the example in Fig. 25.4, the Lee−Schetzen method is used to determine the Wiener kernels (Fig. 25.4C). Here the kernels are used to predict the output by convolving the input with the kernels and adding up the contributions from each kernel (Fig. 25.4D). It can be seen that the predicted and recorded output match very well; this can be further confirmed when we compute the percentage variance that is accounted for (VAF) as:

```
VAF=(1-(std(zzw-est)^2)/(std(zzw)^2))*100
```

Here zzw and est are the measured and estimated output, respectively, and std is the MATLAB® command to compute the standard deviation.

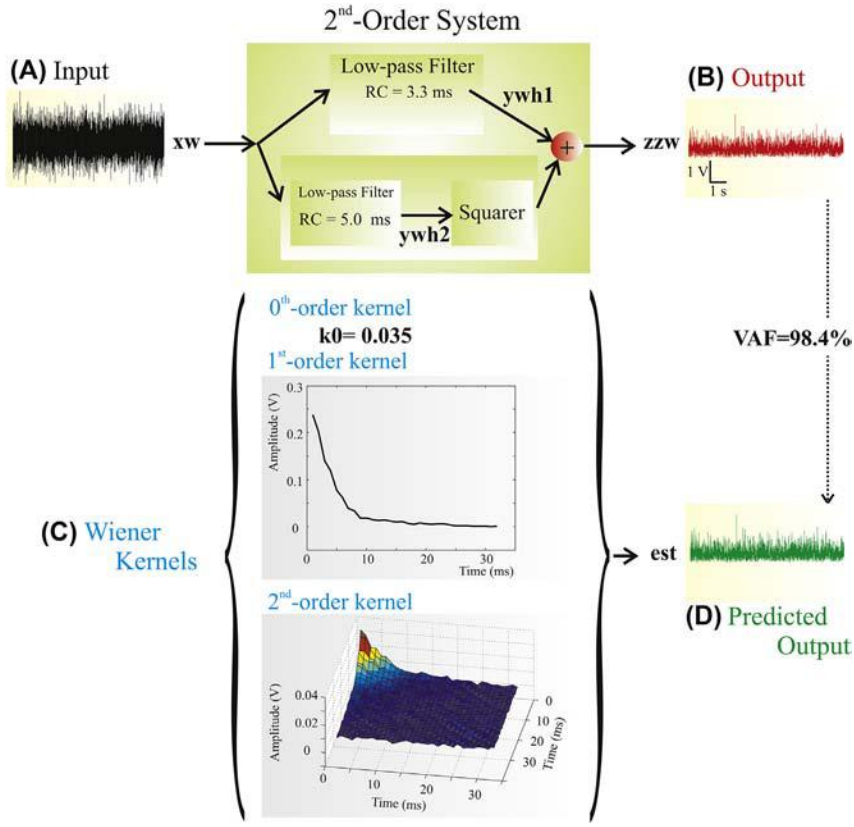


FIGURE 25.4 Wiener kernels of a second-order system similar to the one depicted in Fig. 24.2C; the example is computed with pr25_2.m. (A) The input signal xw is GWN. (B) The output signal is zzw. (C) Zeroth-, first-, and second-order Wiener kernels computed by the MATLAB® script using the procedure depicted in Fig. 25.3. (D) The predicted output est on the basis of the Wiener kernels approximates the measured output well: the variance accounted for VAF is 98.4%.

25.5 RELATION BETWEEN WIENER AND VOLTERRA KERNELS

To summarize the preceding sections, a principal problem with the Volterra series is the dependence between the convolution-like terms (operators) in the series. This dependence prevents us from determining each term separately; this problem is resolved by Wiener's approach. To achieve the independence between terms, Wiener modified the individual terms in the series (Wiener operators are nonhomogeneous) and adapted the input (zero mean GWN). Volterra operators H_n have Volterra kernels

(h_0, h_1, h_2 …), whereas Wiener operators G_n have Wiener kernels (k_0, k_1, k_2 …) as well as derived Wiener kernels ($k_{0(1)}$, $k_{0(2)}$, $k_{1(2)}$, …).

Both Wiener and Volterra kernels are equivalent in the sense that the Wiener kernels can be determined from the Volterra kernels and vice versa. In our examples above we considered the zeroth- to the second-order kernels; let us assume that we are looking into a second-order system so that these are the only kernels available (all higher-order kernels are zero). In this case we have the following kernel components: k_0, $k_{0(1)}$, $k_{0(2)}$, k_1, $k_{1(2)}$, and k_2. In this example the Volterra kernels h_0—h_2 are:

$$\begin{aligned} h_0 &= k_0 + k_{0(1)} + k_{0(2)} = k_0 + k_{0(2)} \\ h_1 &= k_1 + k_{1(2)} = k_1 \\ h_2 &= k_2 \end{aligned} \tag{25.27}$$

The above equations for h_0—h_2 simplify because $k_{0(1)}$ and $k_{1(2)}$ are zero (see Eqs. 25.6 and 25.12). So in a second-order system the relationship between the Wiener and Volterra kernels is fairly straightforward. Had we looked into a higher-order system, for example in a third-order system, we would add $k_{1(3)}$ to h_1 in Eq. (25.27). The expressions for h_0 and h_2 remain unaltered because the other derived third-order kernels $k_{0(3)}$ and $k_{2(3)}$ are zero (Schetzen, 2006). Again, the rationale for this redistribution of kernel components is to create independence between the operators in the series (the condition in Eq. 25.4d). For example, by moving the term $k_{0(2)}$ from the zeroth-order expression (h_0) to the second-order Wiener operator we satisfy the independence between the second-order Wiener operator and H_0 (Eq. 25.8). Considering the relationships in Eq. (25.27), it is unsurprising that a comparison between our findings for the Wiener kernels k_1 and k_2, obtained with pr25_2.m (depicted in Fig 25.4C), and the Volterra kernels h_1 and h_2 found in pr24_2.m from Chapter 24, reveals a close resemblance.

From Eq. (25.27) we can deduce that in order to obtain the Volterra kernels, we must know the system's order as well as all the Wiener kernels. In an experimental situation one usually doesn't know the system's order; at best one could estimate the order by establishing the number of Wiener kernels required to (sufficiently) approximate the system's output signal. In most experimental studies the Wiener kernels (up to the second or third order) and their contributions to the system's output are determined without any further attempt to identify the Volterra kernels.

25.6 ANALYZING SPIKING NEURONS STIMULATED WITH NOISE

When studying intracellular or extracellular recordings of a spiking neuron while stimulating it with noise, one might (of course) use the raw

output trace (including the action potentials) and relate this to the input as we have done previously (Fig. 25.3). However, instead of using the neuron's raw output signal, we can use alternative methods to represent the action potential activity. In the following discussion we assume that timing is the only relevant information associated with a neuronal spiking event. Methods that only consider spike timing can be applied to both intracellular and extracellular recordings of single cells. When dealing with high levels of spike activity it is common to represent the cell's output as the instantaneous spike rate (defined as the [interspike interval]$^{-1}$) plotted versus time; this procedure is shown in Fig. 8.1. Another frequently used technique is to bin the spike train and plot the number of spikes per bin against the time-stamp of the bin. However, if spike rates are low, both of these methods are impractical because we obtain time series that are either extremely unevenly sampled or too sparsely populated with values other than zeros and ones. In general, if one is only interested in the spike train, it seems reasonable to present the output time series of N spikes occurring at times t_i as a series of delta functions, thereby ignoring small subthreshold fluctuations of the neuronal response or noise in the recordings (Chapter 20).

With a little bit of work, the Schetzen correlation method can be adapted to analyze spiking neurons stimulated by GWN. An example for the auditory system was described by Recio-Spinoso et al. (2005). In this study, the auditory system is stimulated by auditory noise and the authors represent the neuron's output y (a spike train of N spikes) as a series of Diracs at times t_i:

$$y(t) = \sum_{i=1}^{N} \delta(t - t_i) \tag{25.28}$$

Following our result in Eq. (25.17), the **zeroth-order Wiener kernel** is the time average of the system's output:

$$k_0 = \langle y(t) \rangle = \left\langle \sum_{i=1}^{N} \delta(t - t_i) \right\rangle \tag{25.29a}$$

The time average $\langle \ldots \rangle$ can be written as an integral over the interval $[0, T]$, divided by epoch length $T\left(\frac{1}{T}\int_0^T \ldots\right)$:

$$\frac{1}{T}\int_0^T \sum_{i=1}^{N} \delta(t - t_i)dt = \frac{1}{T}\sum_{i=1}^{N}\int_0^T \delta(t - t_i)\, dt \tag{25.29b}$$

Here we interchanged the integration and summation operation. The timing t_i for each spike i is between 0 and T, consequently the Dirac $\delta(t - t_i)$ is located within the [0, T] integration interval and the integral $\int_0^T \delta(t - t_i)dt$ evaluates to 1 (see Section 2.2.2). Therefore, the expression in Eq. (25.29) simply counts the number N of action potentials divided by the time epoch T. Thus the zeroth-order Wiener kernel evaluates to N/T, which is the neuron's mean firing rate N_0:

$$\boxed{k_0 = \frac{N}{T} = N_0} \tag{25.30}$$

The **first-order Wiener kernel** is given by Eq. (25.20)

$$k_1(\tau_1) = \frac{1}{\sigma^2}\langle y(t)x(t-\tau_1)\rangle \tag{25.31}$$

If we rewrite the time average $\langle \ldots \rangle$ as an integral and substitute the output z in Eq. (25.20) with the spike time series y (given in Eq. 25.28), we get

$$\begin{aligned} k_1(\tau_1) &= \frac{1}{\sigma^2}\left[\overbrace{\frac{1}{T}\int_0^T \underbrace{\left(\sum_{i=1}^{N}\delta(t-t_i)\right)}_{\text{output}} \underbrace{x(t-\tau_1)}_{\text{input}}\, dt}^{\text{Time Average}}\right] \\ &= \frac{1}{\sigma^2}\left[\frac{1}{T}\int_0^T \left(\sum_{i=1}^{N}\delta(t-t_i)\, x(t-\tau_1)\right) dt\right] \end{aligned} \tag{25.32}$$

In the above we included input x in the summation. Now we again interchange the summation and integration operations

$$k_1(\tau_1) = \frac{1}{\sigma^2}\left[\frac{1}{T}\sum_{i=1}^{N}\underbrace{\int_0^T \delta(t-t_i)x(t-\tau_1)\, dt}_{x(t_i-\tau_1)}\right] = \frac{1}{\sigma^2}\underbrace{\frac{1}{T}N}_{N_0}\underbrace{\frac{1}{N}\sum_{i=1}^{N}x(t_i-\tau_1)}_{R_1(\tau_1)} \tag{25.33}$$

Here we evaluated the integral using the sifting property and multiplied the expression by N/N to allow substitution of $\boldsymbol{R_1(\tau_1)}$**, the reverse-correlation function** (see Section 20.5). The reverse-correlation function

is also known as the revcor, which can be determined by averaging the stimulus time course that precedes each spike (spike triggered average, see also Chapter 20). If we think of the zeroth-order kernel as the time average (mean firing rate) of the system's output, we can conceptualize the first-order Wiener kernel as the average stimulus value some time τ_1 before spike i occurs (that is $x(t_i - \tau_1)$). Simplifying notation, we finally get:

$$\boxed{k_1(\tau_1) = \frac{N_0}{\sigma^2} R_1(\tau_1)} \tag{25.34}$$

The **second-order Wiener kernel** on the other hand represents the mean of the product of two copies of the input x (at two times τ_1 and τ_2) before the occurrence of a spike. The second-order Wiener kernel as given by Eq. (25.24) becomes:

$$k_2(\tau_1, \tau_2) = \frac{1}{2\sigma^4} \langle y(t) x(t - \tau_1) x(t - \tau_2) \rangle \tag{25.35}$$

In Recio-Spinoso et al. (2005), the above equation is corrected by subtracting the zeroth-order kernel k_0 from the output. This makes sense for the following reasons. As discussed above, subtracting the contribution of lower-order kernels from the output is common practice (Fig. 25.3). In Eq. (25.32) we didn't correct the output for the first-order kernel estimate because theoretically its contribution should be independent from the zeroth order one—note that $k_{0(1)}$ is zero (Eq. 25.6). The validity of Eq. (25.6) for this case is also relatively simple to confirm: if we had included the effect of N_0 in Eq. (25.32), we would have gotten an additional term that would vanish, i.e., $\int_0^T N_0 x(t - \tau_1) dt = 0$, because for the GWN input we have $\langle x(t - \tau_1) \rangle = 0$. In contrast, we do correct for the DC (constant) term in the second-order estimate because a nonzero zeroth-order component $k_{0(2)}$ does exist (see Eq. 25.9). We will not correct y for the first-order contribution to k_2 because theoretically $k_{1(2)}$ is zero (Eq. 25.12). Therefore y in Eq. (25.35) can simply be corrected for the zeroth-order contribution N_0 by using the output y minus the zeroth-order kernel

$$y(t) - k_0 = \sum_{i=1}^{N} \delta(t - t_i) - N_0 \tag{25.36}$$

Again, here we do not subtract a contribution from the first-order component since that would vanish anyway. Accordingly, we get:

$$k_2(\tau_1, \tau_2) = \frac{1}{2\sigma^4} \left\langle \left[\sum_{i=1}^{N} \delta(t - t_i) - N_0 \right] x(t - \tau_1) x(t - \tau_2) \right\rangle \tag{25.37a}$$

Writing the time average in the integral notation, we get:

$$= \frac{1}{2\sigma^4}\left\{\frac{1}{T}\int_0^T \left[\sum_{i=1}^{N}\delta(t-t_i) - N_0\right] x(t-\tau_1)x(t-\tau_2)\,dt\right\} \tag{25.37b}$$

We can write the expression as two separate integral terms:

$$= \frac{1}{2\sigma^4}\left\{\frac{1}{T}\int_0^T \sum_{i=1}^{N}\delta(t-t_i)x(t-\tau_1)x(t-\tau_2)\,dt - \frac{1}{T}\int_0^T N_0 x(t-\tau_1)x(t-\tau_2)\,dt\right\} \tag{25.37c}$$

By changing the integration and summation order in the **first term** and applying the sifting property for the Dirac we get the following expression for the first term:

$$\frac{1}{2\sigma^4}\frac{1}{T}\sum_{i=1}^{N}\underbrace{\int_0^T \delta(t-t_i)x(t-\tau_1)x(t-\tau_2)\,dt}_{x(t_i-\tau_1)x(t_i-\tau_2)} = \frac{1}{2\sigma^4}\frac{1}{T}\sum_{i=1}^{N} x(t_i-\tau_1)x(t_i-\tau_2) \tag{25.38a}$$

As we did with the first-order kernel earlier, we can multiply by N/N to simplify notation by using the expression for the **second-order reverse correlation** $R_2(\tau_1,\tau_2) = \frac{1}{N}\sum_{i=1}^{N} x(t_i-\tau_1)x(t_i-\tau_2)$. Finally, the first term in Eq. (25.37c) simplifies to:

$$\frac{1}{2\sigma^4}\underbrace{\frac{1}{T}N}_{N_0}\underbrace{\frac{1}{N}\sum_{i=1}^{N} x(t_i-\tau_1)x(t_i-\tau_2)}_{R_2(\tau_1,\tau_2)} = \frac{N_0}{2\sigma^4}R_2(\tau_1,\tau_2) \tag{25.38b}$$

The **second term** in Eq. (25.37c)

$$= \frac{1}{2\sigma^4}\frac{1}{T}\int_0^T -N_0 x(t-\tau_1)x(t-\tau_2)\,dt = -\frac{N_0}{2\sigma^4}\underbrace{\frac{1}{T}\int_0^T x(t-\tau_1)x(t-\tau_2)\,dt}_{\phi(\tau_2-\tau_1)} = -\frac{N_0}{2\sigma^4}\phi(\tau_2-\tau_1) \tag{25.39}$$

The expression $\frac{1}{T}\int_0^T x(t-\tau_1)x(t-\tau_2)\,dt$ is the autocorrelation $\phi(\tau_2-\tau_1)$ of the input noise. Note that unlike the variable t_i (representing the spike times) in the expression for the reverse correlation R_2, the time variable t is continuous in ϕ. Combining the results for the first and second term we finally get

$$\boxed{k_2(\tau_1,\tau_2)=\frac{N_0}{2\sigma^4}\left[R_2(\tau_1,\tau_2)-\phi(\tau_2-\tau_1)\right]} \tag{25.40a}$$

The above approach was used by Recio-Spinoso et al. (2005) to determine the first- and second-order Wiener kernels of different types of auditory nerve fibers. An example of the second-order kernel for a so-called low-characteristic frequency nerve fiber is shown in Fig. 25.5. If the input is zero mean GWN we have $\phi(\tau_2-\tau_1)=\sigma^2\delta(\tau_2-\tau_1)$. Then if we decide to ignore values where $\tau_1=\tau_2$, as we did in Eq. (25.24), we get

$$\boxed{k_2(\tau_1,\tau_2)=\frac{N_0}{2\sigma^4}R_2(\tau_1,\tau_2)\quad \text{for } \tau_1\neq\tau_2} \tag{25.40b}$$

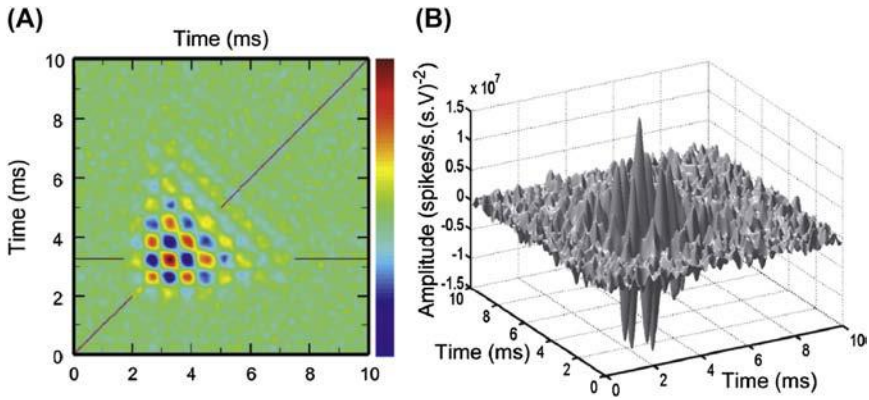


FIGURE 25.5 Example of a second-order kernel of an auditory low-characteristic frequency nerve fiber. (A) A 2-D color-coded presentation of k_2, and (B) the corresponding 3-D plot of k_2. *From Recio-Spinoso, A., Temchin, A.N., van Dijk, P., Fan, Y.-H., Rugero, M.A., 2005. Wiener-kernel analysis of responses to noise of chinchilla auditory-nerve fibers. J. Neurophysiol. 93, 3615–3634.*

25.7 NONWHITE GAUSSIAN INPUT

Zero mean GWN was selected as the input signal for the determination of the Wiener series. In real applications, however, this is not feasible because the bandwidth of the noise is limited. In some cases, the bandwidth of the noise at the input may be wide enough relative to the bandwidth that is relevant for the system under investigation that we may consider the noise as white. However, there are situations where such an assumption is not valid. In these cases the input noise is band limited (colored). The effect of using colored noise as input will be analyzed and discussed in the following paragraphs.

Recall that in Eq. (25.40a) we left the noise autocorrelation term $\varphi(\tau_2 - \tau_1)$ in the expression. In Eq. (25.40b), under the condition that the input is zero mean GWN, we ignored the correlation term because $\sigma^2\delta(\tau_2 - \tau_1)$ evaluates to zero for $\tau_1 \neq \tau_2$. In general, when we consider systems, the noise presented at the input may be zero mean and Gaussian, but nonwhite (Gaussian Colored Noise [GCN]). The term white indicates that all frequencies are equally present in the noise signal while in colored noise not all frequencies are equally present (i.e., we are dealing with filtered white noise). The filter effect has a direct consequence on the autocorrelation of the noise input (Fig. 25.6A and B). However, both colored and white noise may be Gaussian, a property that is related to their amplitude distribution (Fig. 25.6C and D).

In the following we assume that we have determined the zeroth-order kernel as the mean output and that we only deal with demeaned signals for input and output. Under this assumption, the cross-correlation ϕ_{xz} between GCN input x and output z can be developed similar to the procedure shown in Eqs. (25.18)–(25.20):

$$\begin{aligned}\phi_{xz}(v_1) &= \langle z(t)x(t - v_1)\rangle = \sum_{n=0}^{N}\langle G_n[k_n; x(t)]\, x(t - v_1)\rangle \\ &= \int_{-\infty}^{\infty} k_1(\tau_1)\langle x(t - \tau_1)x(t - v_1)\rangle\, d\tau_1 = \int_{-\infty}^{\infty} k_1(\tau_1)\phi_{xx}(\tau_1 - v_1)\, d\tau_1\end{aligned} \tag{25.41}$$

The above shows that the cross-correlation ϕ_{xz} is the convolution of the first-order kernel k_1 with the input autocorrelation ϕ_{xx}:

$$\phi_{xz} = k_1 \otimes \phi_{xx} \tag{25.42a}$$

Therefore k_1 can be obtained from the associated deconvolution. In the frequency domain, convolution and deconvolution can be simplified to

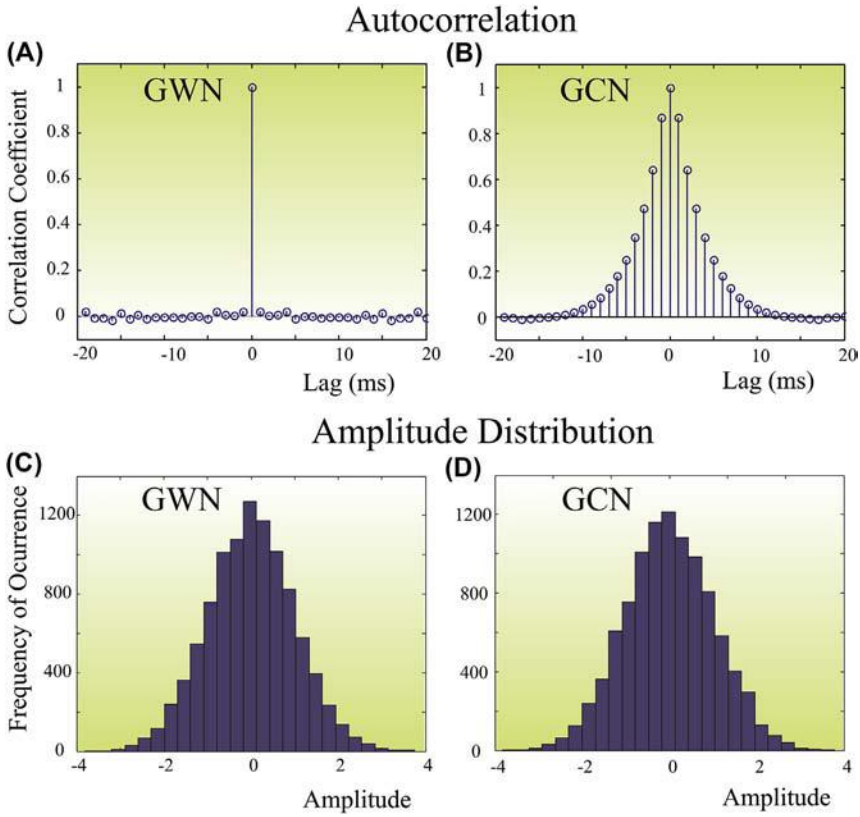


FIGURE 25.6 Autocorrelations (A and B) and amplitude distributions (C and D) of sampled Gaussian noise signals. The cases for GWN are depicted in (A) and (C). The same properties for colored (filtered) noise are shown in (B) and (D). *GCN*, Gaussian colored noise; *GWN*, Gaussian white noise.

multiplication and division, respectively (Section 13.3.2). The equivalent of Eq. (25.42a) in the frequency domain therefore is:

$$\Phi_{xz} = K_1 \Phi_{xx} \rightarrow K_1 = \Phi_{xz}/\Phi_{xx} \tag{25.42b}$$

Here Φ_{xz}, Φ_{xx}, and K_1 are the Fourier transforms of ϕ_{xz}, ϕ_{xx}, and k_1, respectively. Now recall that the cross- and autocorrelation in the frequency domain can also be expressed as products (Section 13.4.2) X^*Z and X^*X (where X and Z are the Fourier transforms of x and z, respectively, and * indicates the complex conjugate). Substituting these expressions for cross- and autocorrelation we get:

$$K_1 = X^*Z/X^*X \tag{25.42c}$$

In real applications we can use this expression to determine K_1 by averaging $\Phi_{xz}(X^*Z)$ and $\Phi_{xx}(X^*X)$ for each frequency f over a number of epochs:

$$K_1(f) = \langle X(f)^* Z(f) \rangle / \langle X(f)^* X(f) \rangle \tag{25.42d}$$

Here the angle brackets $\langle \ldots \rangle$ indicate the average procedure in the frequency domain. Note the similarities and differences between this expression and the one for coherence (Section 13.5). The inverse Fourier transform of K_1 in Eq. (25.42d) gives k_1 for a nonlinear system with GCN input. A similar development for the second-order kernel gives us:

$$K_2(f_1, f_2) = \frac{\langle X(f_1)^* X(f_2)^* Z(f_1 + f_2) \rangle}{2 \langle X(f_1)^* X(f_1) \rangle \langle X(f_2)^* X(f_2) \rangle} \tag{25.43}$$

and taking the inverse Fourier transform of the above expression then gives k_2.

25.8 SUMMARY

As we demonstrated in this chapter, the determination of the Wiener kernels can be obtained from input−output correlation. These scenarios for GWN input are depicted in Fig. 25.7, both for the case with continuous output (Fig. 25.7A and B) and for spike train outputs (Fig. 25.7C−F).

Panels (A) and (B) in Fig. 25.7 show first- and second-order correlation procedures: the multiplication of $z(t)x(t - \tau_1)$ and $z(t)x(t - \tau_1)x(t - \tau_2)$, respectively. Because the cross-correlations are determined by the integration of these products, one may envision moving the multiplications over the signal while summing (integrating) the resulting products. The delays τ, τ_1, τ_2 can be visualized by shifting input x relative to output z (Fig. 25.7A and B).

If the system's output z is a spike train, as shown in panels (C) and (D), the correlations required to compute the kernels are identical: that is, the input can be shifted relative to the output to obtain $x(t - \tau)$, $x(t - \tau_1)$, and $x(t - \tau_2)$. However, this procedure can also be depicted as reverse correlations of each spike at time t_i as shown in panels (E) and (F). Instead of shifting the input as we have just depicted, the reverse correlation procedure is shown here with left-pointing arrows. Note that this is just another way of representing the shifts τ, τ_1, τ_2, and that it is not essentially different from the visualization in panels (C) and (D). However, the fact that we only consider the products $z(t)x(t - \tau)$ and $z(t)x(t - \tau_1)\, x(t - \tau_2)$ at t_i is essentially different from the case when we have a system with continuous output (as depicted in panels (A) and (B)), and is caused by the fact that we model the spike train with a series of Diracs. In between the

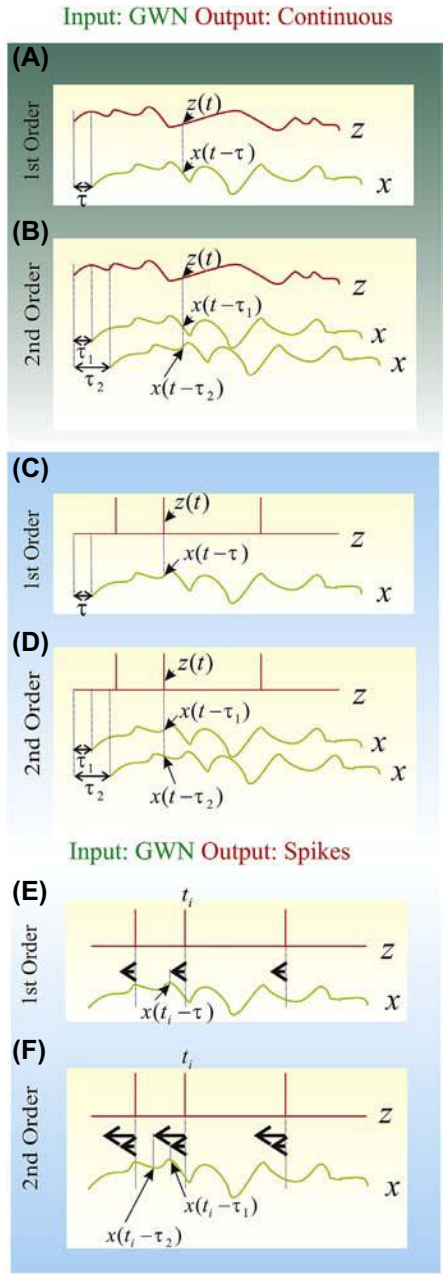


FIGURE 25.7 Summary diagrams of the characterization of a system with GWN input. Diagrams of the cross-correlation procedures for systems with a continuous output (panels A and B) or spike train output (panels C–F). Panels (A), (C), and (E) show the first-order case and panels (B), (D), and (F) represent the second-order procedure. Panels (C) and (E) depict two alternative visualizations for obtaining the first-order cross-correlation for systems with spiking output. In panel (C), the input is shifted by amount τ, whereas in panel (E), $x(t - \tau)$ at time $t = t_i$ is directly determined without shifting x (represented by the *left-pointing arrow*). For the spike output case, this procedure in (E) can be followed (as an alternative to the standard procedure in (C)) since the cross-correlation product is zero when there is no spike. The analogous alternatives for determining the second-order correlation are shown in panels (D) and (F). See text for further explanation. *GWN*, Gaussian white noise.

spikes (that is, in between the unit impulse functions), the output $z(t)$ is considered zero and the products $z(t)x(t-\tau)$ and $z(t)x(t-\tau_1)x(t-\tau_2)$ vanish.

From the examples in this chapter and in Chapter 24, it may be clear that computing the kernels in the series can be a demanding task computationally. Recently, Franz and Schölkopf (2006) described an alternative method to estimate Volterra and Wiener series. Details of their approach are beyond the scope of this text but the essence is to consider discrete systems only (which is not really a limitation if one wants to compute the series). In this case, the Volterra or Wiener series operators can be replaced by functions for which the parameters (the kernel parameters) can be estimated with regression techniques (see Section 11.4.1 for an example of a regression procedure). This approach is computationally more efficient than the Lee and Schetzen (1965) cross-correlation method (described here in Sections 25.3 and 25.4) and makes the estimation of high-order kernels feasible. An example of an application of this method to EEG is described in Barbero et al. (2009).

APPENDIX 25.1

Averages of Gaussian Random Variables

In this appendix we discuss averages of GWN variables because their properties are important for the development of the Wiener series approach (especially in demonstrating that the operators are orthogonal to lower-order operators). Because it is beyond the scope of this text to provide a detailed proof of all properties presented here, for further background see Appendix A of Schetzen (2006). The relationship between higher- and lower-order moments, which we will discuss below, is also known as Wick's theorem (see for example Zinn-Justin, 2002).

Let us consider ergodic and zero mean GWN represented by variable x: i.e., the expected value of x can be replaced by its time average, which is zero (zero mean):

$$E\{x(t-\tau)\} = \langle x(t-\tau)\rangle = 0 \tag{A25.1-1}$$

The product $\langle x(t-\tau_1)x(t-\tau_2)\rangle$ is equal to the autocorrelation and also to the autocovariance (because the noise is zero mean):

$$\langle x(t-\tau_1)x(t-\tau_2)\rangle = \sigma^2\delta(\tau_1-\tau_2) \tag{A25.1-2}$$

Because this may not be immediately apparent, let's define $t-\tau_1 = T$ and $\tau_2-\tau_1 = \tau$. We can now rewrite the autocorrelation in Eq. (A25.1-2) as $\langle x(T)x(T-\tau)\rangle$. In the case where $\tau = 0$ $(\tau_2 = \tau_1)$, we get the expression

$\langle x(T)x(T)\rangle = \left\langle x(T)^2 \right\rangle = E\left\{x(T)^2\right\}$. For GWN with zero mean, this is the definition of the variance σ^2 of signal x (see Section 3.2). Again, since we are dealing with GWN (which gives us a random signal x), two different samples of x are uncorrelated; that is, for $\tau \neq 0$ ($\tau_2 \neq \tau_1$), $x(T)$ is not correlated with $x(T-\tau)$. This means that:

$$\langle x(T)x(T-\tau)\rangle = E\{x(T)x(T-\tau)\} = 0 \quad \text{for } \tau \neq 0.$$

Combining the above findings for $\tau_2 = \tau_1$ and $\tau_2 \neq \tau_1$ we can use the expression in Eq. (A25.1-2) with the Dirac delta function. Let's look into an example in which we scale the correlation coefficient between ± 1. A scatterplot showing correlation for a GWN signal is shown in Fig. A25.1-1. The plot of the signal against itself with zero lag ($\tau = 0$) is depicted in Fig. A25.1-1A and obviously all points lie on the $y = x$ line, corresponding to a correlation coefficient of one. An example for a delay of $\tau = 1$ is shown in Fig. A25.1-1B; here the points are distributed in all directions corresponding to the absence of correlation (correlation coefficient of zero). This behavior is confirmed in a plot of the autocorrelation of GWN: we have a correlation coefficient of one for a lag τ of zero and a correlation coefficient of zero otherwise (see also Fig. 25.6A).

The findings from the paragraph above can be generalized to evaluate higher-order products between GWN signals (Schetzen, 2006). All averages of **odd** products evaluate to zero (e.g., $\langle x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)\rangle = 0$), while it can be shown that higher-order **even**

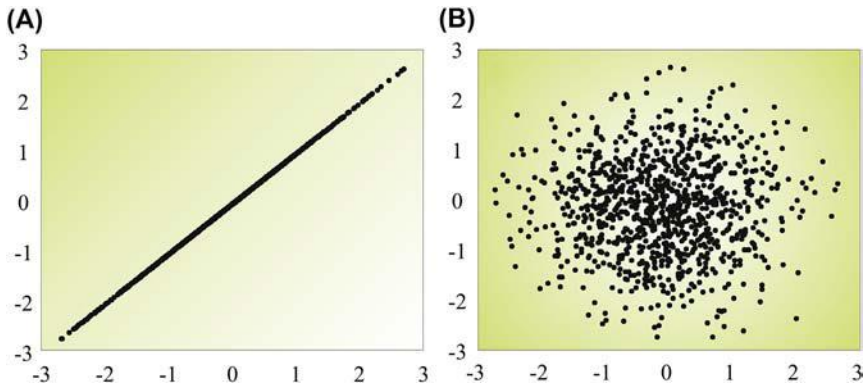


FIGURE A25.1-1 Correlation for $y(t)$, a digitized GWN signal of 1000 points. (A) A plot of $y(t)$ versus $y(t)$; (B) a plot of $y(t+1)$ versus $y(t)$.

products are equal to the sum of all distinct pair-wise products. For example:

$$\langle x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)x(t-\tau_4)\rangle = \langle x(t-\tau_1)x(t-\tau_2)\rangle\langle x(t-\tau_3)x(t-\tau_4)\rangle + \langle x(t-\tau_1)x(t-\tau_3)\rangle\langle x(t-\tau_2)x(t-\tau_4)\rangle + \langle x(t-\tau_1)x(t-\tau_4)\rangle\langle x(t-\tau_2)x(t-\tau_3)\rangle \quad \text{(A25.1-3)}$$

If you are interested in the formal proof of the above generalizations for the odd and even products, please see Appendix A in Schetzen (2006).

APPENDIX 25.2

Delay System as Volterra Operator

We used a specific delay operator earlier to create the Hilbert transform in Chapter 13. Here we will comment on delay operators in general. Creation of a delay v_1 in $x(t)$ is an operation by which we obtain $x(t-v_1)$; this operation can be considered a one-dimensional (1-D), first-order Volterra operator (Fig. A25.2-1A). Higher-dimensional 2-D and 3-D delay systems can be represented by second- and third-order Volterra systems (Fig. A25.2-1B and C), etc. The 1-D operator D_1 can be characterized by the notation

$$D_1[x(t)] = x(t-v_1) \quad \text{(A25.2-1)}$$

Because this is a first-order system, this operation can be represented by a convolution

$$D_1[x(t)] = x(t-v_1) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)\,d\tau \quad \text{(A25.2-2)}$$

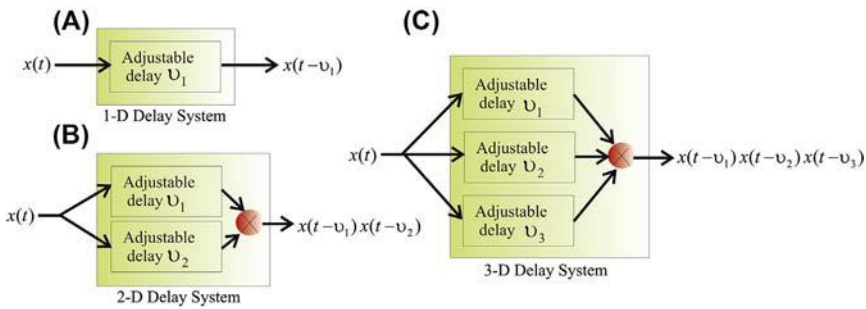


FIGURE A25.2-1 Examples of delay systems as Volterra operators. (A) 1-D delay system; (B) 2-D delay system; (C) 3-D delay system.

From Eq. (A25.2-2) we may conclude that the weighting function (the unit impulse response) of the 1-D system is $h(\tau) = \delta(\tau - \upsilon_1)$, thus resulting in:

$$x(t - \upsilon_1) = \int_{-\infty}^{\infty} \delta(\tau - \upsilon_1) x(t - \tau)\, d\tau \tag{A25.2-3}$$

Similarly, the delay operators for 2-D operator D_2 and 3-D operator D_3 can be defined as: $D_2[x(t)] = x(t - \upsilon_1)x(t - \upsilon_2)$ and $D_3[x(t)] = x(t - \upsilon_1)$ $x(t - \upsilon_2)x(t - \upsilon_3)$, respectively. In Fig. A25.2-1B and C we can see that each of the delays in the higher-dimensional system is a first-order operator. In the second-order (2-D) system the unit impulse responses are $\delta(\tau - \upsilon_1)$ and $\delta(\tau - \upsilon_2)$; in the third-order delay system, the unit impulse responses are $\delta(\tau - \upsilon_1)$, $\delta(\tau - \upsilon_2)$, and $\delta(\tau - \upsilon_3)$. Similar to Eq. (A3.2-3), these operations can be represented with the convolution-like integrals of the Volterra series (see Eq. 24.4), for example in the 2-D case:

$$x(t - \upsilon_1)x(t - \upsilon_1) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \underbrace{\delta(\tau_1 - \upsilon_1)\delta(\tau_2 - \upsilon_2)}_{\substack{\text{Second-order} \\ \text{Volterra kernel} \quad h_2(\upsilon_1,\, \upsilon_2)}} x(t - \tau_1)x(t - \tau_2)\, d\tau_1\, d\tau_2 \tag{A25.2-4}$$

Where the second-order Volterra kernel is:

$$h_2(\upsilon_1, \upsilon_2) = \delta(\tau_1 - \upsilon_1)\delta(\tau_2 - \upsilon_2) \tag{A25.2-5}$$

In the 3-D case the third-order Volterra kernel for a delay system is

$$h_3(\upsilon_1, \upsilon_2, \upsilon_3) = \delta(\tau_1 - \upsilon_1)\delta(\tau_2 - \upsilon_2)\delta(\tau_3 - \upsilon_3) \tag{A25.2-6}$$

Similarly we can extend this approach to an n-dimensional delay operator

$$h_n(\upsilon_1, \upsilon_2, \ldots, \upsilon_n) = \delta(\tau_1 - \upsilon_1)\delta(\tau_2 - \upsilon_2)\cdots\delta(\tau_n - \upsilon_n) \tag{A25.2-7}$$

EXERCISES

25.1 Show that operator G_2 is orthogonal to G_1, because the expectation of the product $E\{G_1G_2\}$ contains odd products of the random variable $x(t)$, which evaluate to zero (Appendix 25.1).

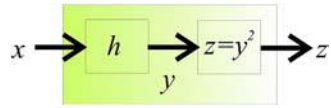


FIGURE E25.3 Diagram of a nonlinear system with input x and output z.

25.2 Create a combination of a linear system (L) and a nonlinear one (N) in an LN cascade similar to that in our examples but with a high-pass instead of a low-pass filter. Modify one of the MATLAB® scripts to determine the zeroth-, first-, and second-order Wiener kernels.

25.3 Consider a nonlinear system consisting of a cascade of a linear filter and a nonlinear squarer as the one depicted in Fig. E25.3. The input x and output y of the linear filter is characterized by the following difference equation:

$$y(n) = x(n) + 2x(n-1) + 4x(n-2) + 2x(n-3) + x(n-4) + 0.5x(n-5) + 0.25x(n-6) \tag{1}$$

The squarer is a static system, its input y and output z are determined by:

$$z = y^2 \tag{2}$$

Use MATLAB® to determine/compute and plot the following.

a. The unit impulse response h of the filter in Fig. E25.3. Create a zero mean GWN signal and determine/compute and plot the following.
b. The first-order Wiener kernel
c. The second-order Wiener kernel
d. Interpret your findings and compare your results with those of Chapter 24, Exercise 24.5
(Suggestion: use a modification of existing MATLAB® code.)

References

Arfken, G.B., Weber, H.J., 2005. Mathematical Methods for Physicists, sixth ed. Academic Press, Elsevier, Burlington, MA.

Barbero, A., Franz, M., Van Drongelen, W., Dorronsoro, J.R., Schölkopf, B., Grosse-Wentrup, M., 2009. Implicit Wiener series analysis of epileptic seizure recordings. Conf. Proc. IEEE Eng. Med. Biol. Soc. 1, 5304–5307.

Franz, M.O., Schölkopf, B., 2006. A unifying view of Wiener and Volterra theory and poly-nomial kernel regression. Neural Comput. 18, 3097–3118.
Lee, Y.W., Schetzen, M., 1965. Measurement of the kernels of a nonlinear system by cross-correlation. Int. J. Contr. 2, 237–254.
Marmarelis, P.Z., Marmarelis, V.Z., 1978. Analysis of Physiological Systems: The White Noise Approach. Plenum Press, New York.
Marmarelis, V.Z., 2004. Nonlinear Dynamic Modeling of Physiological Systems. IEEE Press, John Wiley & Sons Inc., Hoboken, NJ.
Recio-Spinoso, A., Temchin, A.N., van Dijk, P., Fan, Y.-H., Rugero, M.A., 2005. Wiener-kernel analysis of responses to noise of chinchilla auditory-nerve fibers. J. Neurophysiol. 93, 3615–3634.
Schetzen, M., 2006. The Volterra & Wiener Theories of Nonlinear Systems, second reprint ed. Krieger Publishing Company, Malabar, FL.
Westwick, D.T., Kearney, R.E., 2003. Identification of Nonlinear Physiological Systems. IEEE Press, John Wiley & Sons Inc., Hoboken, NJ.
Zinn-Justin, J., 2002. Quantum Field Theory and Critical Phenomena. Oxford University Press, New York.

CHAPTER

26

Poisson–Wiener Series

26.1 INTRODUCTION

In Chapter 25 we considered systems with continuous input signals. One such continuous input is Gaussian white noise (GWN), which allows us to create a series with orthogonal terms that can be estimated sequentially with the Lee–Schetzen cross-correlation method (also shown in Chapter 25). This approach can be adapted when the system's natural input consists of impulse trains such as a spike train. Identifying a system with an impulse train as input will be the topic of this chapter. We will elaborate on the approach that was described by Krausz (1975) and briefly summarized in Marmarelis (2004). Our task at hand is to develop a Wiener series-like approach that describes the input–output relationship of a nonlinear system when an impulse train is at its input. To create randomness at the input, we use an impulse sequence that follows a Poisson process (see Chapter 20, Section 20.2).

26.2 SYSTEMS WITH IMPULSE TRAIN INPUT

The approach is to create a set of operators that are orthogonal to all lower-order Volterra operators, which is analogous to the development of the Wiener series with a GWN input. We will call these operators Poisson–Wiener operators to distinguish our current development of operators (using impulses as input) from that of Chapter 25 (using GWN as input). For each order n, we will symbolize these Poisson–Wiener operators as P_n. Similar to the Wiener series we define the output z of a

nonlinear system as the sum of a set of these operators, each depending on kernel p_n and impulse train input x. For a system of order N we have:

$$z(t) = \sum_{n=0}^{N} P_n[p_n; x(t)]$$

This equation for the Poisson–Wiener series is similar to the ones for the Volterra and Wiener series, but as we will see there are important differences.

As we described in Chapter 25, the approach of the Wiener series works so well because of the specific characteristics of the GWN input signal: $\langle x(t-\tau_1)\rangle = 0$, $\langle x(t-\tau_1)x(t-\tau_2)\rangle = \sigma^2\delta(\tau_2-\tau_1)$, etc (Appendix 25.1). When the system's input changes to a series of impulses, these relationships no longer hold and we can no longer apply the equations we derived previously. In order to resolve this, we must start from scratch and first determine expressions for the averaged products $\langle x(t-\tau_1)\rangle$, $\langle x(t-\tau_1)x(t-\tau_2)\rangle$, … for the Poisson process. Subsequently we must use these new results to redo the Gram–Schmidt procedure for the derivation of our series' orthogonal terms, as was done in Section 25.2. Finally we must redevelop Lee–Schetzen's cross-correlation method in a similar fashion as the procedure described in Section 25.3.

A schematic overview of the procedures we develop in this chapter is depicted in Fig. 26.1. Similar to the properties of Wiener series, the output z of a nonlinear system can be described by a (Poisson–Wiener) series in which

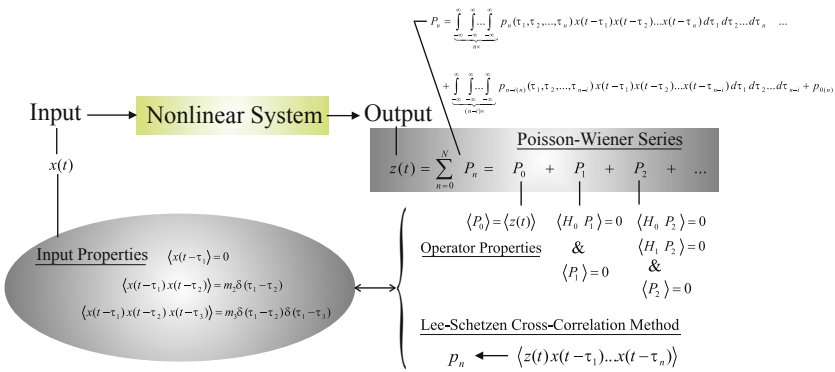


FIGURE 26.1 Diagram of the procedures used here to develop the Poisson–Wiener series, the properties of its operators, and the method to determine the kernels. Just as for the Wiener series, the input signal's properties play a crucial role in the development of the Poisson–Wiener approach.

1. operators P_n are heterogeneous (top-right in Fig. 26.1),
2. each operator is orthogonal to all lower-order Volterra operators,
3. except for P_0, the expectation (or time average) of all operators will vanish, and
4. except for p_0, the kernels can be determined from the cross-correlation of input and output (see also the Lee–Schetzen method introduced in Chapter 25).

In each of the above properties, it is important to know the expectation or time average for the input and its cross-products (see Input Properties in Fig. 26.1). Therefore we will first determine these time averages associated with the input in Section 26.2.1 before we derive the Poisson–Wiener kernels in Section 26.2.2 and adapt Lee–Schetzen's cross-correlation method for determining the kernels from recorded data in Section 26.3.

26.2.1 Product Averages for the Poisson Impulse Train

Let us use signal χ, a train of Diracs with amplitude A that follows a Poisson process with rate ρ (Fig. 26.2A). The first moment or mean μ of impulse train χ can be established by a time average over a sufficiently long interval T. In such an interval we expect to find $N = \rho T$ impulses in the input signal $\chi = \sum_{i=1}^{N=\rho T} A\delta(t - t_i)$. The time average of the input signal is $\langle \chi \rangle = \frac{1}{T} \int_0^T \sum_{i=1}^{N=\rho T} A\delta(t - t_i)dt$. Assuming we can interchange summation and integration we get:

$$\mu = \langle \chi \rangle = \frac{A}{T} \sum_{i=1}^{N=\rho T} \underbrace{\int_0^T \delta(t - t_i)dt}_{=1} = \frac{A}{T}\rho T = \rho A \tag{26.1a}$$

The integral in Eq. (26.1a) evaluates to one if the delta function is located within the interval T (that is $0 \leq t_i \leq T$). We could have done the computation of the mean in a simpler way because we know how many impulses we expect in epoch T and the amplitude of each impulse. The number of impulses (each with amplitude A) during this interval is ρT, resulting in the following expression for the first-moment:

$$\mu = \langle \chi \rangle = \frac{1}{T} \int_0^T \underbrace{\rho T}_{I} \underbrace{A\delta(t)}_{II} dt = \frac{1}{T}\rho T A = \rho A \tag{26.1b}$$

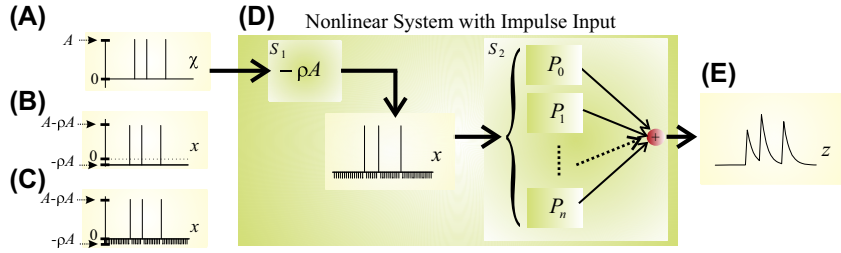


FIGURE 26.2 Impulse train inputs following a Poisson process can be used to identify nonlinear systems. A standard impulse train χ with amplitude A is shown in panel (A). A demeaned version of this time series x is depicted in (B). The signal in panel (C) is the same demeaned series x but now presented as a series of weighted unit impulses (each impulse is represented by a *vertical line*). The procedure depicted schematically in (D) shows the steps we use to identify a nonlinear system with such a train of impulses. First we pretend that the input is demeaned by part of the system (subsystem S_1) by subtracting ρA, the mean of χ. This demeaned series x is then used as input to subsystem S_2. We actually determine the operators P_n and kernels for S_2 instead of the whole system $S_1 + S_2$ but if we can characterize S_2 we have characterized the whole system since S_1 is a simple subtraction. Panel (E) depicts the output z of the system to the impulse input.

Part *I* in Eq. (26.1b) is the number of expected impulses over interval T and part *II* is the amplitude for each impulse. Unlike the first-moment for the GWN signal we used in Chapter 25, this result is **not** zero. The following step is therefore critical for the rest of our approach: **because the nonzero result for the first-moment would complicate matters, we create a new signal x which is the demeaned version of χ** (Fig. 26.2B):

$$\boxed{x(t) = \chi(t) - \rho A} \tag{26.1c}$$

We can check that this generates a zero first-moment for time series $x(t)$:

$$\langle x \rangle = \frac{1}{T}\int_0^T \left[\sum_{i=1}^{N=\rho T} A\delta(t-t_i) - \rho A\right] dt = \frac{1}{T}\left(\sum_{i=1}^{N=\rho T} \underbrace{\int_0^T A\delta(t-t_i)dt}_{A} - \int_0^T \rho A dt\right)$$

$$= \frac{1}{T}\left(\rho AT - [\rho At]_0^T\right) = \frac{1}{T}(\rho AT - \rho AT) = 0 \tag{26.1d}$$

Here we interchanged the integration and summation operations. Subsequently, we evaluate the integral with the delta function and find

that it is equal to the constant A if the delta function falls within epoch T. Alternatively, we can also approach the estimation of $\langle x \rangle$ a bit differently. As you can see in Fig. 26.2C, we can consider the demeaned signal as a series of Diracs (a sampled version of the signal) with amplitude $A - A\rho$ for each spike, and amplitude $-A\rho$ in between the spikes. Over interval T the number of spike occurrences is again ρT and the number of nonspike occurrences is $(1 - \rho)T$.

$$\begin{aligned} \langle x \rangle = \frac{1}{T} \int_0^T \underbrace{[\rho T}_{I} \underbrace{(A - A\rho)\delta(t)}_{II} + \underbrace{(1-\rho)T}_{III} \underbrace{(-A\rho)\delta(t)]}_{IV} dt = \\ \frac{1}{T} \int_0^T \underbrace{\left[\rho AT - \rho^2 AT - \rho AT + \rho^2 AT\right]}_{0} \delta(t) dt = 0 \end{aligned} \tag{26.1e}$$

Parts *I* and *III* above are the expected number of spiking and nonspiking events, respectively, and parts *II* and *IV* are their respective amplitudes. We will use the approach in Eq. (26.1e) to compute the higher-order products in the following. The bottom line is that by using the impulse series x as input, we have (just as for GWN) zero for the first-moment m_1

$$\boxed{m_1 = \langle x \rangle = 0} \tag{26.1f}$$

The next expression we must evaluate is the cross-correlation $\langle x(t - \tau_1)x(t - \tau_2) \rangle$. To start, we can look into the second-moment $\langle x^2 \rangle$ of the impulse train in Fig. 26.2C. As shown above in Eq. (26.1e), the number of events N is the event probability ρ times the interval T, and the nonevent probability equals $(1 - \rho)T$ (parts *I* and *III*, respectively). For the second-order moment, we will square the associated amplitudes (parts *II* and *IV*):

$$\langle x^2 \rangle = \frac{1}{T} \int_0^T \underbrace{\left[\rho T}_{I} \underbrace{(A - A\rho)^2 \delta(t)}_{II} + \underbrace{(1-\rho)T}_{III} \underbrace{(-A\rho)^2 \delta(t)\right]}_{IV} dt \tag{26.2a}$$

Note that by squaring the amplitudes we weight the unit impulse function $\delta(t)$ but we do not need to square the delta function itself. It is relatively simple to see why this isn't required. Imagine the input as the series of Dirac deltas weighted with different amplitudes shown in Fig. 26.2C. The sum of all amplitudes x divided by the epoch length T is the first-moment, the sum of all x^2 divided by T is the second-moment, the sum of all x^3 divided by T is the third-moment, and so on (see Section 3.2).

To sample the amplitudes of x, x^2, x^3, … we only have to weight a single Dirac with the desired amplitude (if you need to review the properties of the Dirac δ, see Section 2.2.2). Simplifying Eq. (26.2a) we get:

$$= \frac{1}{T}\int_0^T \underbrace{\left[\rho A^2 T - 2\rho^2 A^2 T + \rho^3 A^2 T + \rho^2 A^2 T - \rho^3 A^2 T\right]}_{\rho A^2 T - \rho^2 A^2 T} \delta(t)dt$$

$$= \frac{1}{T}\int_0^T T\rho A^2(1-\rho)\delta(t)dt = \frac{1}{T}\left[T\rho A^2(1-\rho)\right]$$

Finally, the expression for the second-moment m_2 becomes:

$$m_2 = \langle x^2 \rangle = \rho A^2(1-\rho) \tag{26.2b}$$

The next step is to determine the second-order cross-correlation using a time average of the product $x(t-\tau_1)x(t-\tau_2)$:

$$\langle x(t-\tau_1)x(t-\tau_2)\rangle = \frac{1}{T}\int_0^T \underbrace{\left[\rho T}_{I}\ \underbrace{(A-A\rho)^2\delta(t-\tau_1)\delta(t-\tau_2)}_{II} + \underbrace{(1-\rho)T}_{III} \times \underbrace{(-A\rho)^2\delta(t-\tau_1)\delta(t-\tau_2)\right]}_{IV} dt \tag{26.3a}$$

Parts I–IV are similar to the ones in the product Eq. (26.2a) of I and II, the first term in the integral in Eq. (26.3a) evaluates to

$$\frac{1}{T}\left[\rho T(A-A\rho)^2\right]\delta(\tau_1-\tau_2)$$

and the product of III and IV, the second term in Eq. (26.3a) becomes:

$$\frac{1}{T}\left[(1-\rho)TA^2\rho^2\right]\delta(\tau_1-\tau_2)$$

Combining the two terms above, we get the result for the second-order autocorrelation:

$$\boxed{\langle x(t-\tau_1)x(t-\tau_2)\rangle = \rho A^2(1-\rho)\delta(\tau_1-\tau_2) = m_2\delta(\tau_1-\tau_2)} \tag{26.3b}$$

This result is not unexpected since Eq. (26.3b) becomes the expression we derived for the second-moment m_2 (Eq. 26.2b) when we have the case

$\tau_1 = \tau_2$. Just as was the case for GWN, this expression will evaluate to zero otherwise.

For computing the third-moment m_3 we can use the same approach as in Eq. (26.2a):

$$\langle x^3 \rangle = \frac{1}{T} \int_0^T \underbrace{\Big[\rho T}_{I} \underbrace{(A - A\rho)^3 \delta(t)}_{II} + \underbrace{(1-\rho)T}_{III} \underbrace{(-A\rho)^3 \delta(t)\Big]}_{IV} dt \tag{26.4a}$$

If you do the algebra, you will find that this results in:

$$m_3 = \langle x^3 \rangle = \rho A^3 (1-\rho)(1-2\rho) \tag{26.4b}$$

The third-order cross-correlation is:

$$\begin{aligned} &\langle x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)\rangle \\ &= \frac{1}{T} \int_0^T \underbrace{\Big[\rho T}_{I} \underbrace{(A - A\rho)^3 \delta(t-\tau_1)\delta(t-\tau_2)\delta(t-\tau_3)}_{II} \\ &\quad + \underbrace{(1-\rho)T}_{III} \underbrace{(-A\rho)^3 \delta(t-\tau_1)\delta(t-\tau_2)\delta(t-\tau_3)\Big]}_{IV} dt \end{aligned} \tag{26.5a}$$

in which parts I–IV can be evaluated similarly to the ones in Eq. (26.3a). Accordingly, the result becomes:

$$\langle x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)\rangle = \rho A^3 (1-\rho)(1-2\rho)\delta(\tau_1-\tau_2)\delta(\tau_1-\tau_3).$$

Combined with Eq. (26.4b), we get

$$\boxed{\langle x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)\rangle = m_3 \delta(\tau_1-\tau_2)\delta(\tau_1-\tau_3)} \tag{26.5b}$$

for the third-order cross-correlation. As you can see, due to the presence of two Diracs, the third-order product is only nonzero for $\tau_1 = \tau_2 = \tau_3$.

In the above cases, things are relatively simple because we set the first-moment to zero by demeaning the input impulse train. This approach ensures that any product that contains $E\{x\}$ or $\langle x \rangle$ (the expectation or time average of x) vanishes (see Appendix 26.1). Appendix 26.1 explains that for the fourth-moment m_4 we have to deal with additional terms that include $E\{x^2\}$. If you are mainly interested in how we will next make Poisson–Wiener operators orthogonal, you can accept the results for m_4 and the fourth-order product below and skip Appendix 26.1. The expression for m_4 is obtained in the same manner as the lower-order moments above.

$$m_4 = \langle x^4 \rangle = \rho A^4 \Big[\rho(1-\rho)^2 + (1-\rho)(1-2\rho)^2\Big] \tag{26.6a}$$

The time averaged fourth-order cross-correlation critically depends on the values of the delays $\tau_1 - \tau_4$ in a piecewise manner (Appendix 26.1, Eq. A26-1.5):

$$\langle x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)x(t-\tau_4)\rangle \begin{cases} \tau_1=\tau_2=\tau_3=\tau_4: & m_4\delta(\tau_1-\tau_2)\delta(\tau_1-\tau_3)\delta(\tau_1-\tau_4) \\ \tau_1=\tau_2 \;\&\; \tau_3=\tau_4: & m_2^2\delta(\tau_1-\tau_2)\delta(\tau_3-\tau_4) \\ \tau_1=\tau_3 \;\&\; \tau_2=\tau_4: & m_2^2\delta(\tau_1-\tau_3)\delta(\tau_2-\tau_4) \\ \tau_1=\tau_4 \;\&\; \tau_2=\tau_3: & m_2^2\delta(\tau_1-\tau_4)\delta(\tau_2-\tau_3) \\ 0 \quad \text{otherwise} \end{cases} \tag{26.6b}$$

26.2.2 Orthogonal Terms of the Poisson–Wiener Series

In this section we use the same procedure (Gram–Schmidt orthogonalization, see Arfken and Weber, 2005) as in Chapter 25 to derive the orthogonal series that can characterize a nonlinear system given our impulse input. As depicted in Fig. 26.2D, the Poisson–Wiener series represents an output signal z consisting of the sum of operators P_n:

$$z(t) = P_0[p_0; x(t)] + P_1[p_1; x(t)] + P_2[p_2; x(t)] + \ldots + P_n[p_n; x(t)] \tag{26.7a}$$

in which the heterogeneous operator P_n is defined as:

$$\begin{aligned} P_n[p_n; x(t)] = & \underbrace{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\ldots\int_{-\infty}^{\infty}}_{n\times} p_n(\tau_1,\tau_2,\ldots,\tau_n)x(t-\tau_1)x(t-\tau_2) \\ & \ldots x(t-\tau_n)d\tau_1 d\tau_2\ldots d\tau_n \\ & + \underbrace{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\ldots\int_{-\infty}^{\infty}}_{(n-1)\times} p_{n-1(n)}(\tau_1,\tau_2,\ldots,\tau_{n-1})x(t-\tau_1)x(t-\tau_2) \\ & \ldots x(t-\tau_{n-1})d\tau_1 d\tau_2\ldots d\tau_{n-1} + \ldots \\ & + \underbrace{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\ldots\int_{-\infty}^{\infty}}_{(n-i)\times} p_{n-i(n)}(\tau_1,\tau_2,\ldots,\tau_{n-i})x(t-\tau_1)x(t-\tau_2) \\ & \ldots x(t-\tau_{n-i})d\tau_1 d\tau_2\ldots d\tau_{n-i} + p_{0(n)} \end{aligned} \tag{26.7b}$$

Here we have Poisson−Wiener kernel p_n and derived Poisson−Wiener kernels $p_{n-i(n)}$ $(i = 1, 2, \ldots, n)$. In Sections 26.2.2.1−26.2.2.3, we will derive the expressions for the Poisson−Wiener operators in a similar fashion we did for the Wiener series in Chapter 25.

26.2.2.1 The Zeroth-Order Poisson−Wiener Operator

Similar to the zeroth-order Wiener operator, we define the zeroth-order Poisson−Wiener operator P_0 as the output's DC component p_0:

$$\boxed{P_0[p_0; x(t)] = p_0} \tag{26.8}$$

In this equation, we use p_0 to symbolize the zeroth-order Poisson−Wiener kernel in order to distinguish it from the zeroth-order Volterra and Wiener kernels h_0 and k_0, respectively.

26.2.2.2 The First-Order Poisson−Wiener Operator

Now we use the orthogonality between Poisson−Wiener operators and lower-order Volterra operators to derive the expression for the first-order Poisson−Wiener kernel p_1. Similar to Eq. (25.5) from Chapter 25 we have:

$$\begin{aligned}\langle H_0[x(t)]P_1[p_1; x(t)]\rangle &= \left\langle h_0 \underbrace{\left[\int_{-\infty}^{\infty} p_1(\tau_1)x(t-\tau_1)d\tau_1 + p_{0(1)}\right]}_{P_1} \right\rangle = 0 \\ &= h_0 \underbrace{\left[\int_{-\infty}^{\infty} p_1(\tau_1)\langle x(t-\tau_1)\rangle d\tau_1 + p_{0(1)}\right]}_{\langle P_1\rangle} = 0\end{aligned} \tag{26.9}$$

The subscript 0(1) indicates that $p_{0(1)}$ is a derived kernel: a zeroth-order member of the first-order operator P_1. Note that we took all constants out of the time average operation, and only the (time-dependent) input time series x remains within the time average brackets $\langle \ldots \rangle$. Since input x is a demeaned impulse train following a Poisson process, we know that

$\langle x(t-\tau_1)\rangle = 0$ (see Eq. 26.1f). Consequently the integral evaluates to zero, and we therefore conclude that the orthogonality requirement demands that

$$\boxed{p_{0(1)} = 0} \tag{26.10}$$

Substituting this result in the general expression for our first-order Poisson–Wiener operator $P_1[p_1; x(t)] = \int_{-\infty}^{\infty} p_1(\tau_1)x(t-\tau_1)d\tau_1 + p_{0(1)}$, we obtain:

$$\boxed{P_1[p_1; x(t)] = \int_{-\infty}^{\infty} p_1(\tau_1)x(t-\tau_1)d\tau_1} \tag{26.11}$$

Note that this result is very similar to the first-order Wiener operator (Eq. 25.7). Furthermore, we see that $E\{P_1\} = \langle P_1\rangle = 0$: that is, the expectation or time-average of P_1, $\left\langle \int_{-\infty}^{\infty} p_1(\tau_1)x(t-\tau_1)\, d\tau_1 \right\rangle$, evaluates to zero because $\langle x(t-\tau_1)\rangle = 0$.

You can also see in Fig. 26.2D that this kernel is not the first-order kernel for our system but for the subsystem indicated by S_2 (the whole system is $S_1 + S_2$). Because we know that the other part, subsystem S_1, is a simple subtraction ($-\rho A$) we have effectively characterized the first-order component of the system under investigation.

26.2.2.3 The Second-Order Poisson–Wiener Operator

To establish the expression for the second-order operator we follow the same procedure as for the Wiener kernels: we demand both orthogonality between the second-order Poisson–Wiener operator and a zeroth-order Volterra operator plus orthogonality between the second-order operator and a first-order Volterra operator.

26.2.2.3.1 Orthogonality Between H_0 and P_2

Using the orthogonality condition we get:

$$\langle H_0[x(t)]P_2[p_2; x(t)]\rangle = 0$$

That is:

$$= \left\langle \underbrace{h_0}_{H_0} \underbrace{\left[\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2 + \int_{-\infty}^{\infty} p_{1(2)}(\tau_1)x(t-\tau_1)d\tau_1 + p_{0(2)} \right]}_{P_2} \right\rangle = 0$$

$$= h_0 \left[\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p_2(\tau_1,\tau_2)\langle x(t-\tau_1)x(t-\tau_2)\rangle d\tau_1 d\tau_2 + \int_{-\infty}^{\infty} p_{1(2)}(\tau_1)\langle x(t-\tau_1)\rangle d\tau_1 + p_{0(2)} \right] = 0 \tag{26.12}$$

Similar to the composition of the Wiener operator G_2, the components $p_{0(2)}$ and $p_{1(2)}$ are derived zeroth-order and first-order members of operator P_2. As we did in Eq. (26.9), we took all constants out of the time average $\langle ... \rangle$ and only kept the time series x within it. Again, because the input is a zero mean impulse train following a Poisson process, the right-hand term in the expression above is zero (since $\langle x(t-\tau_1)\rangle = 0$, Eq. 26.1f). The term on the left is dictated by the averaged product of both inputs $\langle x(t-\tau_1)x(t-\tau_2)\rangle$, which is given by Eq. (26.3b). Therefore the above expression becomes

$$h_0 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p_2(\tau_1,\tau_2)\langle x(t-\tau_1)x(t-\tau_2)\rangle d\tau_1 d\tau_2 + h_0 p_{0(2)}$$

$$= m_2 h_0 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p_2(\tau_1,\tau_2)\delta(\tau_1-\tau_2)d\tau_1 d\tau_2 + h_0 p_{0(2)} = 0$$

This equation can be evaluated by using the sifting property for one of the time constants; here we integrate with respect to τ_2 and get

$$m_2 h_0 \int_{-\infty}^{\infty} p_2(\tau_1,\tau_1)\, d\tau_1 + h_0\, p_{0(2)} = 0 \quad \rightarrow \quad \boxed{p_{0(2)} = -m_2 \int_{-\infty}^{\infty} p_2(\tau_1,\tau_1)\, d\tau_1} \tag{26.13}$$

As you can see, the kernel $p_{0(2)}$ is derived from p_2.

26.2.2.3.2 Orthogonality Between H_1 and P_2

To further express our second-order Poisson–Wiener operator, we will next demand orthogonality between second-order operator P_2 and first-order Volterra operator H_1. Similar to Eq. (25.10), we have

$$\langle H_1[x(t)]P_2[p_2; x(t)]\rangle = 0,$$

which can be written as:

$$\left\langle \left[\int_{-\infty}^{\infty} h_1(v)x(t-v)dv\right]\times \left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2 + \int_{-\infty}^{\infty} p_{1(2)}(\tau_1)x(t-\tau_1)d\tau_1 + p_{0(2)}\right]\right\rangle = 0 \tag{26.14}$$

Eq. (26.14) contains three terms that we will consider separately. The **first** term is:

$$\left\langle \left[\int_{-\infty}^{\infty} h_1(v)x(t-v)dv\right]\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2\right]\right\rangle$$
$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_1(v)p_2(\tau_1,\tau_2)\underbrace{\langle x(t-v)x(t-\tau_1)x(t-\tau_2)\rangle}_{m_3\delta(v-\tau_1)\delta(v-\tau_2)} dvd\tau_1 d\tau_2$$

In the Wiener series development, for systems with GWN input, the odd product $\langle x(t-v)x(t-\tau_1)x(t-\tau_2)\rangle = 0$. Here however, the odd product is given by Eq. (26.5b). This gives:

$$m_3\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_1(v)p_2(\tau_1,\tau_2)\delta(v-\tau_1)\delta(v-\tau_2)dvd\tau_1 d\tau_2$$
$$= m_3\int_{-\infty}^{\infty} h_1(v)p_2(v,v)dv \tag{26.15a}$$

The **second** term in Eq. (26.14):

$$\left\langle \left[\int_{-\infty}^{\infty} h_1(v)x(t-v)dv \right] \left[\int_{-\infty}^{\infty} p_{1(2)}(\tau_1)x(t-\tau_1)d\tau_1 \right] \right\rangle$$
$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_1(v)p_{1(2)}(\tau_1) \underbrace{\langle x(t-v)x(t-\tau_1)\rangle}_{m_2\delta(v-\tau_1)} dvd\tau_1$$

Using the expression for the second-order correlation in Eq. (26.3b) we can simplify to:

$$m_2 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_1(v)p_{1(2)}(\tau_1)\delta(v-\tau_1)dvd\tau_1 = m_2 \int_{-\infty}^{\infty} h_1(v)p_{1(2)}(v)dv \quad (26.15b)$$

Note that we used the sifting property of the Dirac to simplify the double integral.

Finally, the **third** term in Eq. (26.14) is:

$$\left\langle \left[\int_{-\infty}^{\infty} h_1(v)x(t-v)dv \right] p_{0(2)} \right\rangle = \left[\int_{-\infty}^{\infty} h_1(v)\langle x(t-v)\rangle dv \right] p_{0(2)} = 0 \quad (26.15c)$$

which evaluates to zero because $\langle x(t-v)\rangle = 0$ (Eq. 26.1f).

Substituting the results from Eqs. (26.15a–c) into Eq. (26.14), we have:

$$m_3 \int_{-\infty}^{\infty} h_1(v)p_2(v,v)dv + m_2 \int_{-\infty}^{\infty} h_1(v)p_{1(2)}(v)dv = 0$$

From which we may conclude that the derived first-order member of the second-order operator is

$$\boxed{p_{1(2)} = -\frac{m_3}{m_2}p_2(v,v)} \quad (26.16)$$

Again, you can see that the derived kernel $p_{1(2)}$ is indeed derived because it fully depends on p_2. Using the results in Eqs. (26.13) and (26.16),

we get the expressions for the second-order Poisson–Wiener operator in terms of the second-order Poisson–Wiener kernel p_2:

$$P_2[p_2;x(t)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2 \underbrace{-\frac{m_3}{m_2}\int_{-\infty}^{\infty} p_2(\tau,\tau)x(t-\tau)d\tau}_{+\int p_{1(2)}x(t-\tau)d\tau} \underbrace{-m_2\int_{-\infty}^{\infty} p_2(\tau,\tau)d\tau}_{+p_{0(2)}} \tag{26.17}$$

Note that the above result for the second-order Poisson–Wiener operator differs from the second-order Wiener operator (Eq. 25.13) (here $p_{1(2)}$ is nonzero). This difference is due to the fact that the autocorrelation results for a demeaned train of impulses following a Poisson process are different from a GWN signal (see Section 26.2.1 and compare Appendices 25.1 and 26.1). Using the expressions for $\langle x(t-\tau_1)\rangle$ and $\langle x(t-\tau_1)x(t-\tau_2)\rangle$, it is straightforward to show that $E\{P_2\} = \langle P_2\rangle = 0$.

26.3 DETERMINATION OF THE ZEROTH-, FIRST-, AND SECOND-ORDER POISSON–WIENER KERNELS

In this section we will compute the Poisson–Wiener kernels using the same cross-correlation method first described for the Wiener kernels (Lee and Schetzen, 1965). If we deal with a nonlinear system of order N, and we present a demeaned impulse train x following a Poisson process at its input, we obtain output z as the sum of the Poisson–Wiener operators (Fig. 26.2D):

$$z(t) = \sum_{n=0}^{N} P_n[p_n;x(t)] \tag{26.18}$$

In the following example we will describe how to determine the zeroth-, first-, and second-order Poisson–Wiener kernels.

26.3.1 Determination of the Zeroth-Order Poisson–Wiener Kernel

Similar to the Wiener operators, the expectation of all Poisson–Wiener operators P_n, except the zeroth-order operator P_0, is zero. Therefore,

assuming an ergodic process (time averages are allowed for estimating the expectations), we find the average of output signal z

$$\boxed{\langle z(t)\rangle = \sum_{n=0}^{N} \langle P_n[p_n; x(t)]\rangle = P_0[p_0; x(t)] = p_0} \tag{26.19}$$

Thus the zeroth-order Poisson–Wiener kernel is equal to the mean output $\langle z(t)\rangle$.

26.3.2 Determination of the First-Order Poisson–Wiener Kernel

Similar to the procedure for the Wiener kernels depicted in Fig. 25.3, the first-order Poisson–Wiener kernel of a system can be obtained from the cross-correlation between its input and output.

$$\begin{aligned}\langle z(t)x(t-\upsilon_1)\rangle &= \langle P_0[p_0; x(t)]x(t-\upsilon_1)\rangle + \langle P_1[p_1; x(t)]x(t-\upsilon_1)\rangle \\ &\quad + \langle P_2[p_2; x(t)]x(t-\upsilon_1)\rangle + \dots \\ &= \sum_{n=0}^{N} \langle P_n[p_n; x(t)]x(t-\upsilon_1)\rangle\end{aligned} \tag{26.20}$$

Recall that Poisson–Wiener kernels are defined to be orthogonal to lower-order Volterra kernels and recall that the delay operator $x(t-\upsilon_1)$ can be presented as a first-order Volterra operator (Appendix 25.2). Therefore, all Poisson–Wiener operators P_n with $n \geq 2$ are orthogonal to $x(t-\upsilon_1)$ and we only have to deal with operators of order $n = 0$ and 1.

For $\boldsymbol{n = 0}$:

$$\langle P_0[p_0; x(t)]x(t-\upsilon_1)\rangle = p_0 \underbrace{\langle x(t-\upsilon_1)\rangle}_{0} = 0 \tag{26.21a}$$

For $\boldsymbol{n = 1}$:

$$\begin{aligned}\langle P_1[p_1; x(t)]x(t-\upsilon_1)\rangle &= \int_{-\infty}^{\infty} p_1(\tau_1) \underbrace{\langle x(t-\tau_1)x(t-\upsilon_1)\rangle}_{m_2\delta(\tau_1-\upsilon_1)} d\tau_1 \\ &= m_2 \int_{-\infty}^{\infty} p_1(\tau_1)\delta(\tau_1-\upsilon_1)d\tau_1 = m_2 p_1(\upsilon_1)\end{aligned} \tag{26.21b}$$

Here we used Eq. (26.3b) to simplify $\langle x(t-\tau_1)x(t-v_1)\rangle$ and then used the sifting property of the Dirac to evaluate the above integral. From the results in Eq. (26.21b), we conclude that the only nonzero part in Eq. (26.20) is the term for $n=1$; therefore the first-order Poisson–Wiener kernel becomes

$$\langle z(t)x(t-v_1)\rangle = m_2 p_1(v_1) \rightarrow$$

$$\boxed{p_1(v_1) = \frac{1}{m_2}\langle z(t)x(t-v_1)\rangle} \tag{26.22a}$$

Therefore, the first-order Poisson–Wiener kernel is the cross-correlation between input and output weighted by the second-moment m_2 of the input.

We can use the properties of the Dirac to rewrite the cross-correlation expression, because the input is an impulse train. If we substitute the expression for the input in Eq. (26.22a) with a sum of Diracs and present the time average $\langle\ldots\rangle$ with an integral notation $\frac{1}{T}\int_0^T \ldots$, we get:

$$p_1(v_1) = \frac{1}{m_2}\underbrace{\frac{1}{T}\int_0^T z(t)\overbrace{\left[A\sum_{i=1}^{N=\rho T}\delta(t-t_i-v_1)-\rho A\right]}^{\text{input: } x(t-v_1)}dt}_{\text{Time Average}}$$

Assuming we may interchange the integration and summation and separating the terms for the impulse train (the Diracs) and the DC correction (ρA), this evaluates into two integral terms:

$$p_1(v_1) = \frac{A}{m_2}\frac{1}{T}\sum_{i=1}^{N=\rho T}\underbrace{\int_0^T z(t)\delta(t-t_i-v_1)dt}_{z(t_i+v_1)} - \frac{\rho A}{m_2}\underbrace{\frac{1}{T}\int_0^T z(t)dt}_{\langle z\rangle}$$

When using the sifting property it can be seen that the first term is a scaled average of $z(t_i+v_1)$ and may be rewritten as:

$$\frac{A}{m_2}\frac{\rho T}{T}\underbrace{\frac{1}{\rho T}\sum_{i=1}^{N=\rho T} z(t_i+v_1)}_{C_{zx}(v_1)} = \frac{\rho A}{m_2}C_{zx}(v_1) = \frac{\mu}{m_2}C_{zx}(v_1)$$

Note that we used the first-moment $\mu = \rho A$ of the original train of impulses χ here (Eq. 26.1a). Combining the above we get:

$$\boxed{p_1(\upsilon_1) = \frac{\mu}{m_2}[C_{zx}(\upsilon_1) - \langle z \rangle]} \tag{26.22b}$$

The average $\frac{1}{\rho T}\sum_{i=1}^{N=\rho T} z(t_i + \upsilon_1)$ is the cross-correlation $C_{zx}(\upsilon_1)$ between the input impulse train x and the system's output z. **Unlike the reverse-correlation we discussed in Section 20.5 and applied in Section 25.6, we deal with the forward-correlation here (see diagram in Fig. 26.5E)**. In the examples in Chapter 25, we used reversed correlation because the impulse train was the output caused by the input and we had to go back in time to reflect this causality. In this case the role is reversed; the impulse train is the input causing the output.

26.3.3 Determination of the Second-Order Poisson–Wiener Kernel

Using a procedure analogous to that developed for the Wiener kernel in Section 25.3.3, we find the second-order Poisson–Wiener kernel by using a second-order cross-correlation between output and input

$$\langle z(t)x(t-\upsilon_1)x(t-\upsilon_2)\rangle = \langle P_0[p_0; x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\rangle + \langle P_1[p_1; x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\rangle$$
$$+\langle P_2[p_2; x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\rangle + \ldots = \sum_{n=0}^{N}\langle P_n[p_n; x(t)]\, x(t-\upsilon_1)x(t-\upsilon_2)\rangle \tag{26.23}$$

Since $x(t-\upsilon_1)x(t-\upsilon_2)$ can be presented as a second-order Volterra operator (Appendix 25.2), all Poisson–Wiener operators P_n with $n \geq 3$ are orthogonal to $x(t-\upsilon_1)x(t-\upsilon_2)$ (because all Poisson–Wiener operators are orthogonal to lower-order Volterra operators). Furthermore, since we use a Poisson process as input, we will not allow impulses to coincide. Therefore, we neglect all results for equal delays $\upsilon_1 = \upsilon_2$ in the evaluation of Eq. (26.23). Taking into account the considerations above, we now analyze the second-order autocorrelation for $n = 0$, 1, 2 and $\upsilon_1 \neq \upsilon_2$.

For $\boldsymbol{n = 0}$:

$$\langle P_0[p_0; x(t)]x(t-\upsilon_1)x(t-\upsilon_2)\rangle = p_0 \underbrace{\langle x(t-\upsilon_1)x(t-\upsilon_2)\rangle}_{m_2\delta(\upsilon_1-\upsilon_2)} = m_2 p_0 \delta(\upsilon_1 - \upsilon_2) \tag{26.24a}$$

We can neglect this term because, due to the Dirac, it evaluates to zero for $v_1 \neq v_2$.

For $\boldsymbol{n = 1}$:

$$\langle P_1[p_1; x(t)]x(t-v_1)x(t-v_2)\rangle = \int_{-\infty}^{\infty} p_1(\tau_1)\underbrace{\langle x(t-\tau_1)x(t-v_1)x(t-v_2)\rangle}_{m_3\delta(\tau_1-v_1)\delta(\tau_1-v_2)}\, d\tau_1$$

$$= m_3 p_1(v_1)\delta(v_1 - v_2) \tag{26.24b}$$

Due to the Dirac, this expression also evaluates to zero for $v_1 \neq v_2$ and can therefore be ignored.

For $\boldsymbol{n = 2}$:

We compute $\langle P_2[p_2; x(t)]x(t-v_1)x(t-v_2)\rangle$ using Eq. (26.17) and we get:

$$\overbrace{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_2(\tau_1,\tau_2)\underbrace{\langle x(t-\tau_1)x(t-\tau_2)x(t-v_1)x(t-v_2)\rangle}_{\text{Eq. (26.6b)}}\, d\tau_1 d\tau_2}^{I}$$

$$\overbrace{-\frac{m_3}{m_2}\int_{-\infty}^{\infty} p_2(\tau_1,\tau_1)\underbrace{\langle x(t-\tau_1)x(t-v_1)x(t-v_2)\rangle}_{m_3\delta(\tau_1-v_1)\delta(\tau_1-v_2)}\, d\tau_1}^{II} \overbrace{- m_2\int_{-\infty}^{\infty} p_2(\tau_1,\tau_1)\underbrace{\langle x(t-v_1)x(t-v_2)\rangle}_{m_2\delta(v_1-v_2)}\, d\tau_1}^{III} \tag{26.24c}$$

Term I in Eq. (26.24c) is the most complex one and potentially consists of four terms (Eq. 26.6b). Given that we have four delays τ_1, τ_2, v_1, v_2 and taking into account the condition $v_1 \neq v_2$, there are only two combinations that remain to be considered: $\tau_1 = v_1$ & $\tau_2 = v_2$, and $\tau_1 = v_2$ & $\tau_2 = v_1$. The first term can now be rewritten as

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_2(\tau_1,\tau_2)\underbrace{\langle x(t-\tau_1)x(t-\tau_2)x(t-v_1)x(t-v_2)\rangle}_{m_2^2\delta(\tau_1-v_1)\delta(\tau_2-v_2)+m_2^2\delta(\tau_1-v_2)\delta(\tau_2-v_1)}\, d\tau_1 d\tau_2$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_2(\tau_1,\tau_2)\left[m_2^2\delta(\tau_1-v_1)\delta(\tau_2-v_2) + m_2^2\delta(\tau_1-v_2)\delta(\tau_2-v_1)\right]d\tau_1 d\tau_2$$

$$= \underbrace{m_2^2\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_2(\tau_1,\tau_2)\delta(\tau_1-v_1)\delta(\tau_2-v_2)d\tau_1 d\tau_2}_{p_2(v_1,v_2)} + \underbrace{m_2^2\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p_2(\tau_1,\tau_2)\delta(\tau_1-v_2)\delta(\tau_2-v_1)d\tau_1 d\tau_2}_{p_2(v_2,v_1)}$$

The double integral above can be evaluated by sifting for τ_1 and τ_2. Because we assume that the kernel is symmetric around its diagonal, we can use $p_2(v_1,v_2) = p_2(v_2,v_1)$ and the above evaluates to:

$$\boxed{2m_2^2 p_2(v_1, v_2)} \tag{26.24d}$$

Term *II* in Eq. (26.24c) can be written as

$$-\frac{m_3}{m_2}\int_{-\infty}^{\infty} p_2(\tau_1,\tau_1)m_3\delta(\tau_1 - v_1)\delta(\tau_1 - v_2)d\tau_1 = -\frac{m_3^2}{m_2}p_2(v_1, v_1)\delta(v_1 - v_2) \tag{26.24e}$$

Due to the delta function $\delta(v_1 - v_2)$, this part can be neglected since it is zero for $v_1 \neq v_2$.

Term *III* in Eq. (26.24c) evaluates to:

$$-m_2\int_{-\infty}^{\infty} p_2(\tau_1,\tau_1)m_2\delta(v_1 - v_2)d\tau_1 = -m_2^2\int_{-\infty}^{\infty} p_2(\tau_1,\tau_1)\delta(v_1 - v_2)d\tau_1 \tag{26.24f}$$

This term can also be ignored because it equals zero for $v_1 \neq v_2$.

To summarize Eq. (26.24), the only nonzero term for $v_1 \neq v_2$ is the result in Eq. (26.24d). Substituting this result into Eq. (26.23) we get an expression for our second-order Poisson–Wiener kernel p_2:

$$\langle z(t)x(t - v_1)x(t - v_2)\rangle = 2m_2^2 p_2(v_1, v_2)\rightarrow$$

$$\boxed{p_2(v_1, v_2) = \frac{1}{2m_2^2}\langle z(t)x(t - v_1)x(t - v_2)\rangle \quad \text{for } v_1 \neq v_2} \tag{26.25a}$$

Using the fact that the input x is a train of impulses, we can employ the same treatment as for Eq. (26.22a) and rewrite Eq. (26.25a) as:

$$p_2(v_1, v_2) = \frac{1}{2m_2^2}\frac{1}{T}\int_0^T z(t)\overbrace{\left[A\sum_{i=1}^{N=\rho T}\delta(t - t_i - v_1) - \rho A\right]}^{\text{first copy of the input: } x(t-v_1)}\overbrace{\left[A\sum_{j=1}^{N=\rho T}\delta(t - t_j - v_2) - \rho A\right]}^{\text{second copy of the input: } x(t-v_2)}dt$$

This expression generates four terms:

$$\text{I. } \frac{A^2}{2m_2^2}\frac{1}{T}\int_0^T z(t)\left[\sum_{i=1}^{N=\rho T}\delta(t - t_i - v_1)\right]\left[\sum_{j=1}^{N=\rho T}\delta(t - t_j - v_2)\right]dt$$

$$\text{II. } -\frac{\rho A^2}{2m_2^2}\frac{1}{T}\int_0^T z(t)\sum_{i=1}^{N=\rho T}\delta(t-t_i-v_1)dt = -\frac{\mu^2}{2m_2^2}\,C_{zx}(v_1), \text{ with:}$$

$$C_{zx}(v_1) = \frac{1}{\rho T}\sum_{i=1}^{N=\rho T} z(t_i+v_1) \text{ and } \mu = \rho A$$

$$\text{III. } -\frac{\rho A^2}{2m_2^2}\frac{1}{T}\int_0^T z(t)\sum_{j=1}^{N=\rho T}\delta(t-t_j-v_2)dt = -\frac{\mu^2}{2m_2^2}\,C_{zx}(v_2), \text{ with:}$$

$$C_{zx}(v_2) = \frac{1}{\rho T}\sum_{j=1}^{N=\rho T} z(t_j+v_2) \text{ and } \mu = \rho A$$

$$\text{IV. } \frac{\rho^2 A^2}{2m_2^2}\frac{1}{T}\int_0^T z(t)dt = \frac{\mu^2}{2m_2^2}\langle z\rangle$$

Terms II–IV were evaluated in a similar fashion as in the first-order case in Eq. (26.22). After changing the order of the integration and summations, term I above evaluates to:

$$\frac{A^2}{2m_2^2}\frac{1}{T}\sum_{i=1}^{N=\rho T}\sum_{j=1}^{N=\rho T}\underbrace{\int_0^T z(t)[\delta(t-t_i-v_1)][\delta(t-t_j-v_2)]\,dt}_{z(t_i+v_1)\delta(t_i-t_j+v_1-v_2)}$$

$$= \frac{A^2}{2m_2^2}\frac{\rho^2T^2}{T}\underbrace{\frac{1}{\rho^2T^2}\sum_{i=1}^{N=\rho T}\sum_{j=1}^{N=\rho T} z(t_i+v_1)\delta(t_i-t_j+v_1-v_2)}_{C_{zxx}(v_1,v_2)} = \frac{\mu^2}{2m_2^2}TC_{zxx}(v_1,v_2)$$

In the above expression we substituted μ for ρA (Eq. 26.1a); this is the first moment of the original impulse train χ. Combining the results from the four terms I–IV above, we have

$$\boxed{p_2(v_1,v_2) = \frac{\mu^2}{2m_2^2}\{TC_{zxx}(v_1,v_2) - [C_{zx}(v_1) + C_{zx}(v_2) - \langle z\rangle]\} \quad \text{for } v_1 \neq v_2} \tag{26.25b}$$

The second-order correlation $C_{zxx}(v_1,v_2)$ is the weighted average $\frac{1}{\rho^2T^2}\sum_{i=1}^{N=\rho T}\sum_{j=1}^{N=\rho T} z(t_i+v_1)$ under the condition set by the Dirac

$\delta(t_i - t_j + v_1 - v_2)$. This condition is equivalent to sampling the values of output signal z when $t_i - t_j + v_1 - v_2 = 0$. This indicates that:

1. the delay between the copies of the input is $\Delta = v_2 - v_1 = t_i - t_j$, which means that the delays under consideration for creating the averages are equal to the differences Δ between pulse times t_i, t_j;
2. there is a relationship between the individual delays given by $v_2 = t_i - t_j + v_1$, which represents a line in the v_1, v_2-plane at 45 degrees and with an intercept at $t_i - t_j$.

This conditional average is therefore a slice through $p_2(v_1,v_2)$ defined by this line. The delays we consider are strictly given by $t_i - t_j$ and the input to the averaging procedure is $z(t_i + v_1)$. A representation of C_{zxx} is shown in Fig. 26.5F. To keep Fig. 26.5F compatible with the symbols in the other panels in this figure, the delay v_1 is replaced by τ_1 in the diagram.

26.4 IMPLEMENTATION OF THE CROSS-CORRELATION METHOD

Because there is no standard command in MATLAB® to create a series of randomly occurring impulses following a Poisson process, we include an example function Poisson.m to create such an impulse train (for details see Appendix 26.2). In MATLAB® script pr26_1.m, we use this function to create the input (impulses with an amplitude of 2 units) to a nonlinear system consisting of a first-order component (a low-pass filter) and a second-order component (a low-pass filter amplifier with $5\times$ amplification plus a squarer), similar to the system in Fig. 24.2C.

Typical traces for input and output are shown in Fig. 26.3. By following the same steps depicted in Fig. 25.3 for the Wiener kernels, but now using Eqs. (26.19), (26.22b), and (26.25b), we find the Poisson–Wiener kernels for the system. Note that the cross-correlations are impulse-triggered averages in this case.

The following MATLAB® code is part of script* pr26_1.m *and shows the computation of the first-order cross-correlation and first-order kernel p1 according to Eq. (26.22b) (step 3 of the Lee–Schetzen method depicted in Fig. 25.3).

```
% Step 3. Create the 1st order average (see Fig. 25.3)
%-------------------------------------------------------------------
Czx=zeros(T,1);
for i=1:length(time)-10                % to avoid problems by ignoring
                                         last 10 impulses
```

```
   Czx=Czx+v0(time(i):time(i)+T-1)';
end;

% Now we scale Czx by the # of spikes (i.e. length(time) -10, which is
   the
% # of trials in the average. Using Equation (26.22b):
p1=(u1/m2)*((Czx/(length(time)-10))-mean(v0));   % Note that all
                                                    scaling
                                                    parameters
                                          % u1, m2, and mean(v0)
                                          % are at the ms - scale !
figure;
plot(p1)
title('1st order Poisson-Wiener kernel')
xlabel('Time (1 ms)')
ylabel('Amplitude')
```

The percentage of variance accounted for (VAF, see Section 25.4 for its definition) by the output from the Poisson–Wiener kernels in this example is typically in the high 90s. This VAF number is fairly optimistic because, as can be seen in the output trace in Fig. 26.3, a large number of points with a good match between output (dotted red line) and predicted

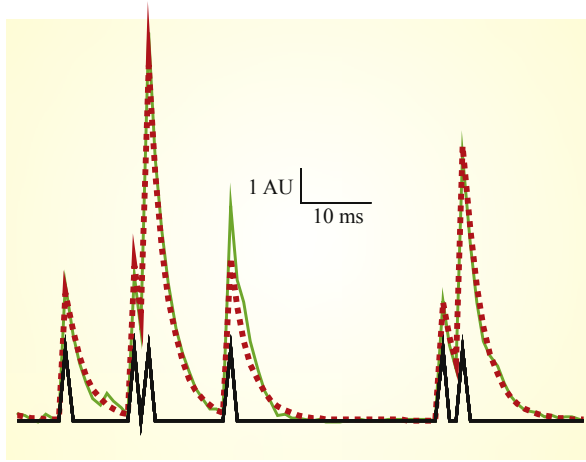


FIGURE 26.3 Example of input (pulses, *lower line* [black]) and output (*dotted line* [red]) traces. The (green) *line*, following the output closely, is the output contribution from the Poisson–Wiener kernels. The vertical scale is in Arbitrary Units (AU). The variance accounted for by the model output (VAF) in this example was 97.6%. All traces were generated by pr26_1.m.

output (green line) are zero or close to zero; the predicted output we mainly care about is (of course) the activity caused by the input (impulses) and not the rest state.

26.5 SPIKING OUTPUT

In Chapter 25, we considered continuous input to nonlinear systems with both continuous and spiking output. So far in this chapter, we have analyzed nonlinear systems with spike train input and continuous output. The possible cases one might encounter in neuroscience are summarized in Fig. 26.4. As you can see, the only case remaining for our discussion is a nonlinear system with both spike input and output (Fig. 26.4D). We can compute the Poisson–Wiener kernels by using the previously found expressions (Eqs. (26.19), (26.22b), and (26.25b)). In this case, kernel p_0 can be determined by the time average of the spike output. Just as in Eq. (25.30) in Chapter 25, p_0 evaluates to the mean firing rate of the output. In Eqs. (26.22b) and (26.25b) we can see that computing

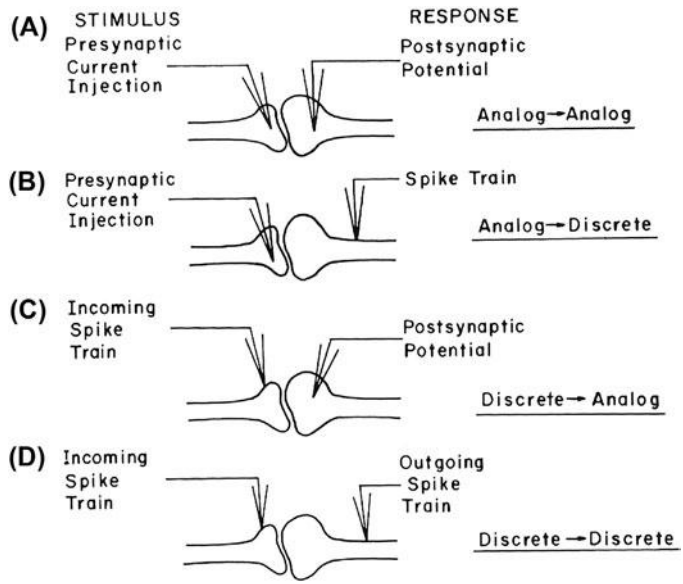


FIGURE 26.4 Types of signals one may encounter for a system's input and output in neuroscience. In this example a synapse is used to symbolize the four different possibilities (A–D). The incoming signal may be a Gaussian white noise signal (analog—presynaptic current injection) or a train of impulses following a Poisson process (discrete—incoming spike train). The output can be a postsynaptic potential (analog) or a spike train (discrete). *Fig. 11.1 from Marmarelis, P.Z., Marmarelis, V.Z., 1978. Analysis of Physiological Systems: The White Noise Approach, Plenum Press, New York, with kind permission of Springer Science and Business Media.*

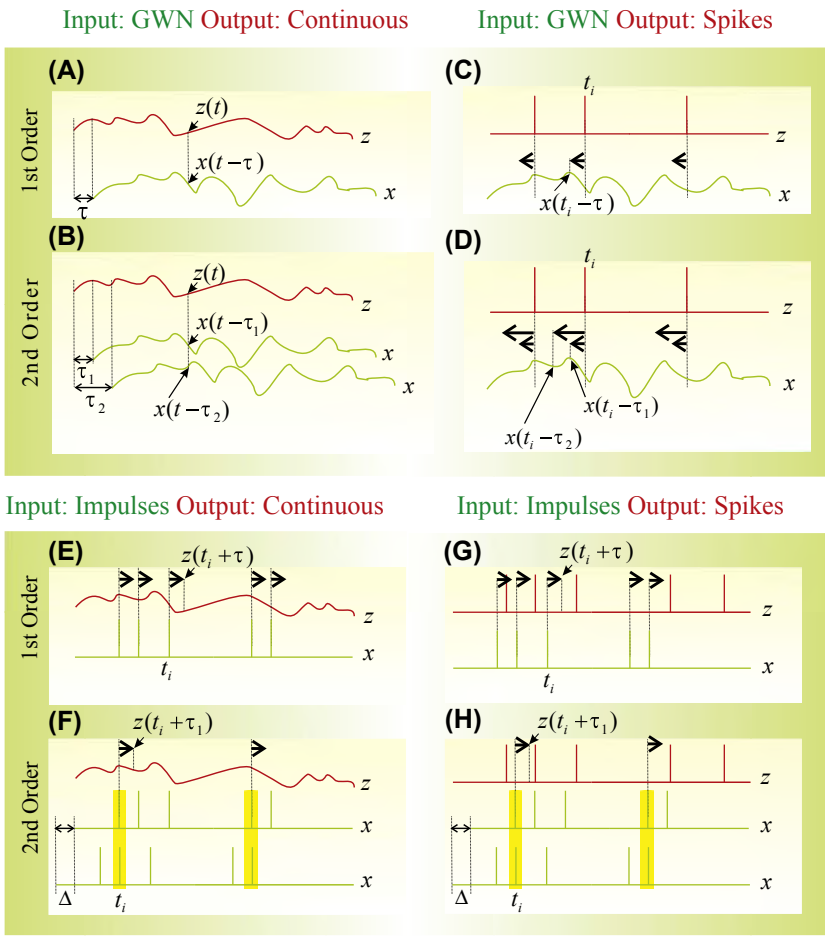


FIGURE 26.5 Summary of the procedures for determining first- and second-order cross-correlation for the different scenarios depicted in Fig. 26.4. In all panels, x (green) is the input and z (red) is the output. The panels for GWN input (A–D) are identical to Fig. 25.7A,B,E, and F. Panels (E–H) show the procedures for impulses as input. See text for further explanation.

first- and second-order kernels requires first- and second-order cross-correlations C_{zx} and C_{zxx} (in this case spike-triggered averages). The procedures for obtaining these cross-correlations are depicted in Fig. 26.5G and H. The first order cross-correlation is a spike-triggered average; we use the input spikes as the trigger (Fig. 26.5G). The second-order cross-correlation is triggered by coinciding spikes of two copies from the input, one of which is shifted by amount Δ (Fig. 26.5H). The procedures for obtaining these cross-correlation functions are very similar

to those discussed for a system with a spike input and continuous output; as you can see by comparing panels E with G and F with H in Fig. 26.5.

26.6 SUMMARY

The procedures for determining the first- and second-order cross-correlations for the four scenarios in Fig. 26.4 are summarized in Fig. 26.5. The part of this figure for GWN input is identical to the overview in Fig. 25.7. The panels for spike input show the procedures discussed in this chapter. In practice, the cross-correlations required for computation of the Poisson–Wiener kernels can all be obtained from spike-triggered averages (Fig. 26.5E–H). As such it is very similar to the procedure we followed for nonlinear systems with GWN input and spike output in Chapter 25 (Fig. 26.5C and D). The difference is that here we use the input spikes, instead of the output spike train, to trigger the average, hence we determine forward cross-correlation instead of reversed correlation. This reflects that the systems are considered causal (output is caused by input). Thus a system's output shows reversed correlation with the input (Fig. 26.5C and D) and its input is forward-correlated with its output (Fig. 26.5E–H). The procedures followed to obtain the cross-correlations for systems with both continuous input and output are depicted in Fig. 26.5A and B. Here the correlation products are not spike-triggered and the delays of the copies of the input are determined for each sample of the output $z(t)$ (Chapter 25).

APPENDIX 26.1

Expectation and Time Averages of Variables Following a Poisson Process

The results for time averages of GWN are well known and were briefly summarized in Appendix 25.1. For the application of impulse trains we use a different input signal, the Poisson process (Section 20.2). Products of variables following a Poisson process are important for determining the Poisson–Wiener kernels when impulse trains are used as input to a nonlinear system. A similar derivation was described by Krausz (1975) in his Appendix A.[1] Assuming that $x(t)$ follows a Poisson process, we can define the **first-moment** as the expectation of x: $E\{x\}$. Because the signal is

[1] Please note that the derivation by Krausz contains minor errors for the moments m_2 and m_3, leading to differences in the scaling of several of the derived expressions (as was also noted by Marmarelis (2004)).

ergodic, we may replace this with a time average $\langle x\rangle = \frac{1}{T}\int_0^T x(t)dt$ (see Section 3.2) if you need to review ergodicity and time averages. To simplify things further down the road, we start from a demeaned impulse train so that (see Eq. 26.1c):

$$\boxed{E\{x\} = \langle x\rangle = 0} \tag{A26.1-1}$$

The expectation of the **second-order product**, or autocorrelation, of variable x is $E\{x(t-\tau_1)x(t-\tau_2)\}$ (for autocorrelation, see Section 13.4). Note that the expression we use here is slightly different from Eq. (13.13): we substituted $t-\tau_1$ and $t-\tau_2$ for t_1 and t_2, respectively. Because x follows a Poisson process the factors $x(t-\tau_1)$ and $x(t-\tau_2)$ are independent if $\tau_1 \neq \tau_2$; in this case we may replace the expectation with two separate ones, that is:

$$E\{x(t-\tau_1)x(t-\tau_2)\} = E\{x(t-\tau_1)\}E\{x(t-\tau_2)\} = 0 \quad \text{for } \tau_1 \neq \tau_2$$

The above product evaluates to zero, because the first-moment of our impulse train is zero. The expression $E\{x(t-\tau_1)x(t-\tau_2)\}$ is only nonzero if $\tau_1 = \tau_2$ and (again) because x is ergodic we may apply a time average $\langle x(t-\tau_1)x(t-\tau_2)\rangle$. In Section 26.2.1 you can see that the final result for the expectation/time average of the second-order product becomes

$$\boxed{E\{x(t-\tau_1)x(t-\tau_2)\} = \langle x(t-\tau_1)x(t-\tau_2)\rangle = m_2\delta(\tau_1-\tau_2)} \tag{A26.1-2}$$

The expectation of the **third-order product** $E\{x(t-\tau_1)\ x(t-\tau_2)\ x(t-\tau_3)\}$ equals zero by independence if $\tau_1 \neq \tau_2 \neq \tau_3$, since in this case we can rewrite the expression as:

$$\begin{aligned} E\{x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)\} &= E\{x(t-\tau_1)\}E\{x(t-\tau_2)\}E\{x(t-\tau_3)\} \\ &= 0 \quad \text{for } \tau_1 \neq \tau_2 \neq \tau_3 \end{aligned}$$

If only one pair of τ's is equal, that is, $\tau_1 = \tau_2 \neq \tau_3$ or $\tau_1 \neq \tau_2 = \tau_3$ we can make the substitutions $\tau_1 = \tau_2$ or $\tau_2 = \tau_3$ and then separate the expectation into two factors:

$$\begin{aligned} &E\{x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)\} = E\{x(t-\tau_1)x(t-\tau_1)x(t-\tau_3)\} \\ &= E\Big\{x(t-\tau_1)^2x(t-\tau_3)\Big\} = E\Big\{x(t-\tau_1)^2\Big\}E\{x(t-\tau_3)\} = 0 \ \text{ for } \tau_1 \neq \tau_2 \neq \tau_3 \end{aligned}$$

and

$$\begin{aligned} E\{x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)\} &= E\Big\{x(t-\tau_2)^2\Big\}E\{x(t-\tau_1)\} \\ &= 0 \quad \text{for } \tau_1 \neq \tau_2 = \tau_3 \end{aligned}$$

In all of the above cases, the expressions evaluate to zero because $E\{x\} = 0$ and the only instance where the expectation of the third-order product is nonzero is for $\tau_1 = \tau_2 = \tau_3$. In this case (due to ergodicity), it may be replaced by $\langle x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)\rangle$ (see Eq. 26.5b). The final nonzero result is

$$\boxed{\begin{aligned} E\{x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)\} &= \langle x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)\rangle \\ &= m_3\delta(\tau_1-\tau_2)\delta(\tau_1-\tau_3) \end{aligned}} \quad \text{(A26.1-3)}$$

The expectation of the **fourth-order product** $E\{x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)x(t-\tau_4)\}$ is zero by independence if:

I. $\tau_1 \neq \tau_2 \neq \tau_3 \neq \tau_4$

and nonzero if all delays are equal:

II. $\tau_1 = \tau_2 = \tau_3 = \tau_4$

Using the time average approach we used in Section 26.2 we find the following for the fourth-moment:

$$m_4 = \langle x^4 \rangle = \frac{1}{T}\int_0^T \Big[\rho T(A - A\rho)^4\delta(t) + (1-\rho)T(-A\rho)^4\delta(t)\Big]dt$$

This can be written as:

$$m_4 = \langle x^4 \rangle = \rho A^4\big(1 - 4\rho + 6\rho^2 - 3\rho^3\big) = \rho A^4\Big[\rho(1-\rho)^2 + (1-\rho)(1-2\rho)^2\Big]$$

Including the condition $\tau_1 = \tau_2 = \tau_3 = \tau_4$, we find that the averaged product is:

$$\boxed{\langle x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)x(t-\tau_4)\rangle = m_4\delta(\tau_1-\tau_2)\delta(\tau_1-\tau_3)\delta(\tau_1-\tau_4)} \quad \text{(A26-1.4)}$$

in which the δ functions represent the condition that all delays must be equal for a nonzero result. Three alternatives with three equal delays are:

III. $\tau_1 \neq \tau_2 = \tau_3 = \tau_4$

IV. $\tau_1 = \tau_2 \neq \tau_3 = \tau_4$

V. $\tau_1 = \tau_2 = \tau_3 \neq \tau_4$

In all three cases **III**–**V**, the expectation of the fourth-order product evaluates to zero. For instance, in case **V** we have:

$$E\{x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)x(t-\tau_4)\} = E\Big\{x(t-\tau_1)^3 x(t-\tau_4)\Big\}$$
$$= \underbrace{E\Big\{x(t-\tau_1)^3\Big\}}_{m_3}\ \underbrace{E\{x(t-\tau_4)\}}_{0} = 0 \quad \text{for} \quad \underline{\tau_1=\tau_2=\tau_3\neq\tau_4}$$

Finally we have three cases in which delays are equal in pairs:

VI. $\tau_1=\tau_2$ and $\tau_3=\tau_4$

VII. $\tau_1=\tau_3$ and $\tau_2=\tau_4$

VIII. $\tau_1=\tau_4$ and $\tau_2=\tau_3$

These cases evaluate to a nonzero value. For instance, in case **VI** we get:

$$E\{x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)x(t-\tau_4)\} = E\Big\{x(t-\tau_1)^2 x(t-\tau_3)^2\Big\}$$
$$= \underbrace{E\Big\{x(t-\tau_1)^2\Big\}}_{m_2}\ \underbrace{E\Big\{x(t-\tau_3)^2\Big\}}_{m_2} = m_2^2 \text{ for } \underline{\tau_1=\tau_2} \ \&\ \underline{\tau_3=\tau_4}$$

If we represent the conditions $\tau_1=\tau_2$ and $\tau_3=\tau_4$ with Dirac delta functions, we get the final result for case **VI**:

$$m_2^2\delta(\tau_1-\tau_2)\delta(\tau_3-\tau_4)$$

To summarize the results for the expectation of the fourth-order product:

$$E\{x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)x(t-\tau_4)\}\begin{cases}\tau_1=\tau_2=\tau_3=\tau_4\colon m_4\delta(\tau_1-\tau_2)\delta(\tau_1-\tau_3)\delta(\tau_1-\tau_4)\\ \tau_1=\tau_2\ \&\ \tau_3=\tau_4\colon m_2^2\delta(\tau_1-\tau_2)\delta(\tau_3-\tau_4)\\ \tau_1=\tau_3\ \&\ \tau_2=\tau_4\colon m_2^2\delta(\tau_1-\tau_3)\delta(\tau_2-\tau_4)\\ \tau_1=\tau_4\ \&\ \tau_2=\tau_3\colon m_2^2\delta(\tau_1-\tau_4)\delta(\tau_2-\tau_3)\\ 0\quad\text{otherwise}\end{cases}$$

(A26.1-5)

In Section 26.3.3 we have to evaluate a case where we know that one pair of delays cannot be equal. Note that in such a case we have to combine from alternatives **VI**–**VIII**. For example, if $\tau_2 \neq \tau_3$, we have two possibilities for pair forming:

a. $\tau_1 = \tau_3$ and $\tau_3 = \tau_4$ in which pair τ_1,τ_2 is independent from pair τ_3,τ_4.
b. $\tau_1 = \tau_3$ and $\tau_2 = \tau_4$ in which pair τ_1,τ_3 is independent from pair τ_2,τ_4.

Now we can write the expectation for $\tau_2 \neq \tau_3$as the sum of (a) and (b):

$$E\{x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)x(t-\tau_4)\} \underset{\tau_2 \neq \tau_3}{=} E\left\{x(t-\tau_1)^2 x(t-\tau_3)^2\right\} + E\left\{x(t-\tau_1)^2 x(t-\tau_2)^2\right\}$$

$$= \overbrace{\underbrace{E\left\{x(t-\tau_1)^2\right\}}_{m_2}\ \underbrace{E\left\{x(t-\tau_3)^2\right\}}_{m_2}}^{\text{case a}} + \overbrace{\underbrace{E\left\{x(t-\tau_1)^2\right\}}_{m_2}\ \underbrace{E\left\{x(t-\tau_2)^2\right\}}_{m_2}}^{\text{case b}}$$

$$= m_2^2\delta(\tau_1-\tau_2)\delta(\tau_3-\tau_4) + m_2^2\delta(\tau_1-\tau_3)\delta(\tau_2-\tau_4) \tag{A26.1-6}$$

Throughout this chapter and this appendix we simply assume zero correlation for nonzero delays, e.g. Eq. (26.3b) and Eq. (A26.1-2). This property is frequently presented as valid for long epochs of the random pulse input χ, introduced in Section 26.2.1 and Fig. 26.2. In contrast, we employ this property for signal x, after input signal is corrected by subtracting its mean value in Eq. (26.1c). For instance, you may have noticed that in Eq. (26.3a) we ignore the effects between the pulses and the (corrected) nonzero values in between these pulses. However, as we show in the following, we are correct to apply this approach using zero autocorrelation for nonzero delays.

First, to simplify notation and without losing validity of our reasoning, we will set amplitude A equal to one. Comparing the autocorrelations $C_\chi(\tau)$ for the original input series of pulses $\chi = \sum_{i=1}^{N=\rho T} \delta(t-t_i)$ with $C_x(\tau)$ for the corrected signal $x = \sum_{i=1}^{N=\rho T} \delta(t-t_i) - \frac{N}{T}$, we find that:

$$C_x(\tau) = C_\chi(\tau) - \left(\frac{N}{T}\right)^2 = C_\chi(\tau) - \rho^2 \tag{A26.1-7}$$

Eq. (A26.1-7) indicates that a correction of $C_\chi(\tau)$, leads to subtraction of $-\rho^2 = -\left(\frac{N}{T}\right)^2$ in order to obtain $C_x(\tau)$. Although this may seem surprising, this term actually corrects for the nonzero values in $C_\chi(\tau)$ at $\tau \neq 0$. In fact, the autocorrelation functions of pulse series tends to have occasional positive values at $\tau \neq 0$ (e.g., Fig. 20.8), so in spike rasters $C_\chi(\tau)$ is not really zero for $\tau \neq 0$ but (on average) slightly positive. In the original spike rasters, the probability of two spikes randomly correlating at a given value of τ is proportional to the square of the pulse rate in χ: i.e. ρ^2. Thus $C_\chi(\tau) = \rho^2$ across the correlation epoch for $\tau \neq 0$, and subtracting ρ^2 from $C_\chi(\tau)$ creates the desired condition:$C_x(\tau) = 0$ at $\tau \neq 0$.

If you want to see a numerical example of this property, type the following in the MATLAB® command window.

```
clear;
ti=round(rand(1,100)*999);
X=zeros(1,1000);
X(ti)=1;
x=X-mean(X);
c=xcorr(x,500);
C=xcorr(X,500);
figure;hold;
stem(X);stem(x,'k')
figure;hold;
plot(C);plot(c,'k')
```

Here we use X and x for our variables χ and x above. This example shows that c, the autocorrelation of x(black trace) satisfies our assumption, $C_x(\tau) = 0$ at $\tau \neq 0$, reasonably well, much better than C representing $C_\chi(\tau)$(blue trace).

APPENDIX 26.2

Creating Impulse Trains Following a Poisson Process

For the generation of a series of random numbers following a Gaussian or uniform distribution, we use MATLAB® commands randn and rand, respectively. A standard MATLAB® command for generating a series of intervals according to a Poisson process does not exist. Therefore we will apply a Monte Carlo technique to create such an impulse train according to a Poisson process. Our target is to follow a Poisson process characterized by Probability Density Function (PDF) $\rho e^{-\rho x}$ (see Chapter 20). This works as follows. First we generate pairs of independent random numbers x,y with the MATLAB® rand command. Because the rand command generates numbers between 0 and 1, x is multiplied with the maximal epoch value we

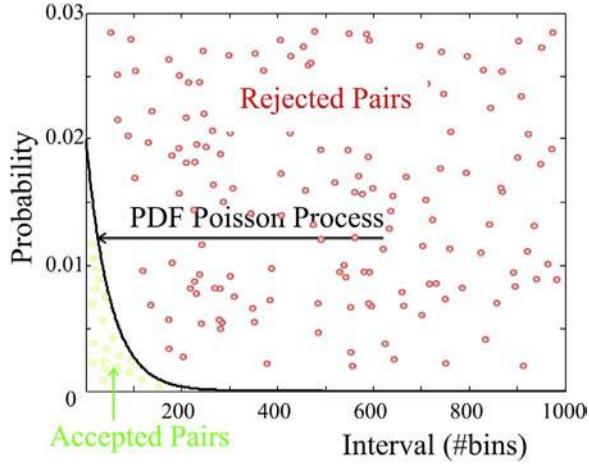


FIGURE A26.2-1 The Poisson process PDF can be used to create series of intervals obeying a Poisson process. Pairs of random uniformly distributed numbers x,y are generated: x is scaled between 0 and the maximum epoch length (1000 in this example) and y between 0 and 1. Each pair is then plotted in the X–Y plane. If $y < \rho e^{-\rho x}$ the point is accepted (green), otherwise it is rejected (red). If sufficient numbers are evaluated, the result is that epochs are retained according to the PDF describing the Poisson process.

want to consider, in order to rescale it between 0 and the maximum interval. Second, for each trial we compute $p = \rho e^{-\rho x}$ which is the probability p for interval x to occur according to the Poisson process. So far we will generate intervals x where all intervals have an equal probability because the MATLAB® rand command is uniformly distributed. The second random number y associated with the randomly generated interval will also be evenly distributed between 0 and 1. We now only include pairs x, y in our series if $y < p$ and discard all others (Fig. A26.2-1); by following this procedure, the accepted intervals x obey the Poisson process because the probability that they are retained is proportional with $\rho e^{-\rho x}$, which is the desired probability. This procedure can, of course, be used for other distributions as well; it is known as the accept–reject algorithm.

The following MATLAB® snippet of the function Poisson.m shows an implementation of the procedure to generate a series of intervals following a Poisson process. This function is applied in pr26_1.m. Note that this routine also avoids intervals that are smaller than one bin because we do not allow for superimposed impulses.

```
i=1;
while (i<len)
   x=rand;y=rand;          % two random numbers scaled 0-1
   x=x*epoch;              % the interval x is scaled 0-epoch
   p=rate*exp(-rate*x);    % the probability associated with the
                             interval
```

```
                    % using the Poisson process PDF
    if (y < p);     % Is the probability below the random # ?

      if x > 1;     % Avoid intervals that are too small (< 1 bin)
      series(i)=x;  % else the interval is included
      i=i+1;
      end;
    end;
end;
```

EXERCISES

26.1 Modify MATLAB® script pr26_1 to determine the Poisson–Wiener kernels for a model of a synapse (Chapter 30). Use the following model for the synaptic unit impulse response:

$$h(t) = \begin{cases} \alpha\beta t e^{-\beta t} & t \geq 0 \\ 0 \quad \text{otherwise} \end{cases}.$$

This corresponds to the following ordinary differential equation (ODE) you can use to simulate the response to the Poisson pulse train:

$$\ddot{y}(t) = \alpha\beta x(t) - 2\beta\dot{y}(t) - \beta^2 y(t),$$

with x and y representing the synaptic input (the Poisson impulse train) and output (the synaptic response), respectively (Scenario in Fig. 26.4C).

26.2. Modify the script you obtained in Exercise 26.1 by adding a squarer and determine the Poisson–Wiener kernels for the modified synapse output, i.e., y^2. Compare your results from Exercises 26.1 and 26.2, and interpret your findings.

References

Arfken, G.B., Weber, H.J., 2005. Mathematical Methods for Physicists, sixth ed. Academic Press, Elsevier, Burlington, MA.

Krausz, H.I., 1975. Identification of nonlinear systems using random impulse train inputs. Biol. Cybern. 19, 217–230.

Lee, Y.W., Schetzen, M., 1965. Measurement of the kernels of a nonlinear system by cross-correlation. Int. J. Control 2, 237–254.

Marmarelis, P.Z., Marmarelis, V.Z., 1978. Analysis of Physiological Systems: The White Noise Approach. Plenum Press, New York.

Marmarelis, V.Z., 2004. Nonlinear Dynamic Modeling of Physiological Systems. IEEE Press, John Wiley & Sons Inc., Hoboken, NJ.

CHAPTER

27

Nonlinear Techniques

27.1 INTRODUCTION

As was discussed in Chapter 13, most of the analytical tools we have introduced throughout the text are based on dynamical processes generated by linear time-invariant (LTI) systems. In general, the success of most of these time series analysis methods in physiology is surprising considering that physiological processes are known to include significant nonlinearities. The explanation for this relative success is perhaps due to cases in which physiological systems can be studied in a state where the linear behavior is most prominent, or where a limited range of a measured property is considered in which a linear process is a good approximation of the system's behavior. On the other hand, there are many examples where the inclusion of nonlinear dynamics is critical for understanding the physiology. The well-known Hodgkin and Huxley model (Hodgkin and Huxley, 1952) or higher-order kernels in the auditory response (e.g., Recio-Spinoso et al., 2005) are just a few examples. It is also very likely that our perception of the success of linear analysis is biased because many interrelationships in the nervous system remain undiscovered because the linear analysis techniques currently in use (e.g., correlation) fail even to detect them. In a general sense, there is a significant need for novel signal processing tools for studying nonlinear relationships in physiology as well as a critical necessity to evaluate the tools that have been developed over past decades.

The purpose of this chapter is to introduce a few of the nonlinear analysis tools that are available to analyze and describe biomedical signals. First, we explore some of the characteristics that distinguish linear from nonlinear systems by analyzing a few simple examples, including the so-called logistic equation. In a second step, we look into the failure and success of different techniques in characterizing and quantifying time series generated by nonlinear dynamical systems. An overview of metrics

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00027-8

that have been developed to characterize the nonlinear dynamics of time series is given in Section 27.5. Students with a greater interest in nonlinear systems are referred to Peitgen et al. (1992), Kaplan and Glass (1995), and Strogatz (1994); these authors provide excellent introductions to nonlinear dynamics and chaos theory with numerous practical examples.

27.2 NONLINEAR DETERMINISTIC PROCESSES

The purpose of many experiments is to find direct cause–effect relationships: so-called deterministic relationships in which the past uniquely determines the current state of a system. While some systems, such as a swinging pendulum, behave predictably, developments in stock markets or the weather don't seem to be so predictable. We might therefore conclude ***incorrectly*** that simple deterministic systems are predictable, whereas involvement of more complex processes puts that predictability at risk. In the following we show that even simple, deterministic processes can display surprisingly unpredictable behavior. For instance, a time series generated by a simple difference equation, such as the logistic equation:

$$x_i = a x_{i-1}(1 - x_{i-1}) \tag{27.1}$$

in which each point x_i depends only on the quadratic function of its previous value, can exhibit behavior ranging from stable, or oscillatory, to very erratic. Examples of time series x_i generated with different values of parameter a in Eq. (27.1) are shown in Fig. 27.1.

One way to understand this system is to investigate its convergence properties. In these examples it can be seen that the number of possible final states of the system critically depends on the value of a. For some values of this parameter the system converges to a single steady-state solution (Fig. 27.1A), while for other values the system generates anything from a handful to a seemingly infinite number of states (Fig. 27.1B and C). If we plot the final states generated by Eq. (27.1) against different values of a, we obtain the so-called final state diagram shown in Fig. 27.1D.

This diagram, which can be produced with MATLAB script pr27_1.m, shows:

1. the behavior of the logistic equation converges to a single value for $a < 3$ (e.g., Fig. 27.1A),
2. stable periodic behavior with two values occurs for $3 < a < 3.4495$ (e.g., Fig. 27.1B), and
3. a subsequently increasing number of final states with increasing values of a.

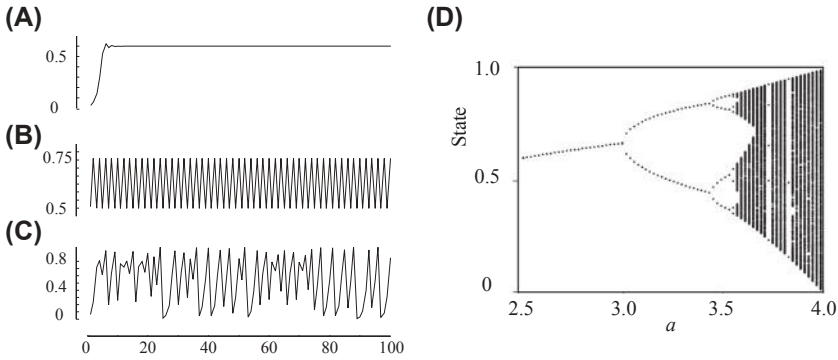


FIGURE 27.1 Characteristics of time series created with the logistic equation (Eq. 27.1). (A) The time series converges to a single value for $a = 2.50$. (B) For $a = 3.24$ there is oscillatory behavior between two states. (C) Chaos at $a = 4$. (D) One of the icons of chaos: the final state diagram showing the period-doubling route to chaos. In this graph final states are plotted against the value of a in the logistic equation. The logistic equation (a quadratic iterator) transitions to oscillatory behavior at the bifurcation $a = 3$. For $a > 3.569\ldots$, the system transitions to chaotic behavior. Feigenbaum (1983) discovered that the ratio of two successive ranges over which the period doubles, is a constant universally encountered in the period-doubling route to chaos (Feigenbaum's number: 4.6692…). The MATLAB script pr27_1.m can be used to create the final state diagram. *From Van Drongelen, W., Lee, H.C., Hecox, K.E., 2005. Seizure prediction in Epilepsy. In: He, B. (Ed.), Neural Engineering, Kluwer Academic/Plenum, NY, pp. 389–420.*

This characterization of the long-term behavior of the system, or the bifurcation diagram in Fig. 27.1D, shows a transition from stable to so-called chaotic behavior for a above values of 3.569… The transition pathway, from simple periodic behavior into an unpredictable regime shown in Fig. 27.1D is called the period-doubling route to a ***chaos*** (Fig. 27.1C). The logistic equation is not an exceptional case: many more examples of fairly simple systems showing complex behavior can be found. The seminal example, a simplified and deterministic model of a weather system consisting of a set of only three nonlinear differential equations, showed similarly, dramatic unpredictability (Lorenz, 1963). We can compare these unpredictable processes to rolling a die or drawing a numbered lotto ball; they all show random behavior that can be characterized by measuring the probabilities of the various outcomes. In principle, if one knew precisely all the positions and mechanical parameters of the elements in a lotto drawing, one would be able to calculate the end result. It is surprising that in spite of this "in-principle-predictability," randomness seems inherently associated with these types of deterministic processes. Interestingly, some very complex phenomena, such as tides, that depend on many other processes (position of the moon, the wind,

details in the coastline, etc.) can be fairly predictable. From the examples above, we therefore conclude:

1. that the level of complexity in a time series doesn't necessarily correspond with the level of complexity of the underlying process, and
2. that deterministic systems do not always show predictable behavior.

One might counter that the second conclusion simply represents lack of knowledge of the system and that one should be able to precisely compute the behavior of a system if the equations governing its dynamics (such as Eq. 27.1) are known. In principle this is correct, but there are serious practical problems with this approach. Usually there is a degree of unavoidable uncertainty that prevents us from knowing all aspects of the past and present states of a dynamical system. And, even if we do know all this, any knowledge, measurement, or computation of a system state is associated with a degree of precision which limits our exact knowledge of the initial and subsequent condition of an evolving process. Finally, it appears that in some systems with *nonlinear dynamics*, minute errors or perturbations (of the order of magnitude of a rounding error of a computer or even smaller) generate huge differences in the predicted outcomes even over short prediction windows. This difference can grow ***disproportionately*** towards the same order of magnitude as the predicted values: i.e., the evolution and outcome of certain types of processes may depend critically on initial conditions (see the example in Fig. 27.4D–F). This dependence is sometimes referred to as the "butterfly effect": as was pointed out by Lorenz, a perturbation as small as the flapping wings of a butterfly could influence the development of a tornado on another continent. Of course, sensitivity to perturbations also exists in linear systems. However, the error in a linearly evolving process grows ***proportionally*** with the predicted values.

27.3 LINEAR TECHNIQUES FAIL TO DESCRIBE NONLINEAR DYNAMICS

Linear techniques were designed to detect properties and/or relationships within or between time series generated by linear systems. Therefore, we may safely assume that these techniques perform well in signals with a strong linear component. In contrast, we may suspect that these techniques would fail in signal analysis when the data originate from a nonlinear dynamical system. To explore our expectations and suspicions, let's consider an example of the application of the autocorrelation function (Chapter 13) to three distinct time series:

a predominantly linear relationship:

$$x_i = ax_{i-1} + N_{i-1} \tag{27.2a}$$

a nonlinear relationship (logistic equation):

$$x_i = ax_{i-1}(1 - x_{i-1}) \tag{27.2b}$$

a completely random relationship:

$$x_i = N_{i-1}, \tag{27.2c}$$

with N being a random process with zero mean and a standard deviation of one.

In Fig. 27.2 we can observe a sample of each of these time series, their associated autocorrelation functions, and their so-called return plots. The return plot depicts the relationship between successive values of the time series x_{i+1} and x_i. As expected, the autocorrelation in Fig. 27.2A2 indicates

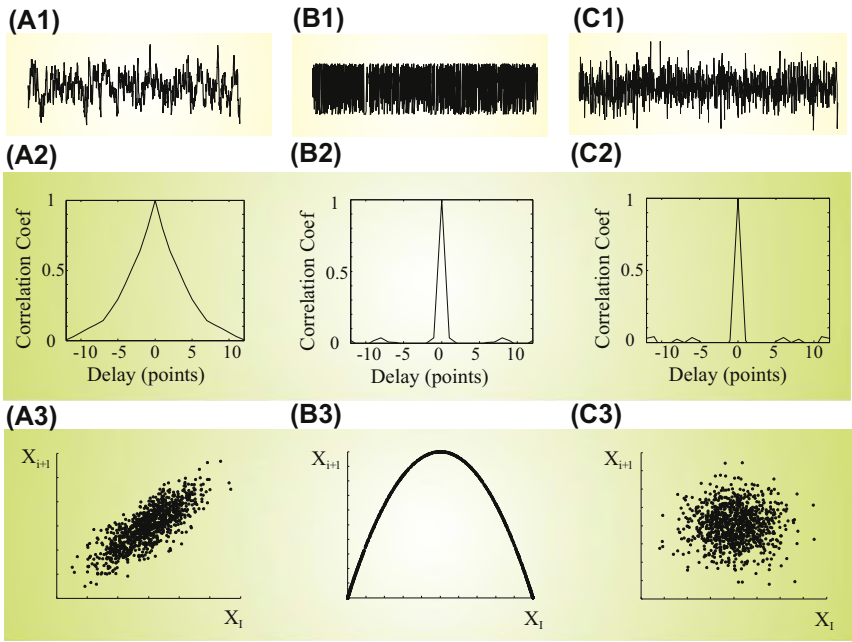


FIGURE 27.2 Three different time series generated with Eq. (27.2a–c). The waveform in (A1) is determined by a linear function; the signal in (B1) is from a nonlinear dynamical system (the logistic equation); (C1) is a random time series. The graphs in (A2)–(C2) show the corresponding autocorrelation functions. The scatter plots in (A3)–(C3) depict $x_{i+1} = f(x_i)$. *The graphs in this figure can be reproduced by MATLAB programs pr27_2.m to pr27_5.m.*

a relationship between subsequent points in the time series generated by Eq. (27.2a). While the autocorrelation reasonably detects no point-to-point relationship within the random series (since there is none to detect), it fails to show the relatively simple, and completely deterministic, connection between past and current values in the nonlinear logistic equation (Fig. 27.2B2 and C2). On the other hand, the return plots (Fig. 27.2A3,B3, and C3) clearly distinguish between the linear, quadratic, and random relationships. The important conclusion here is that in both the first two examples (Eqs. 27.2a and 27.2b) there are deterministic components, but only the first can be detected by linear analysis tools.

In case of the linear iterator in Eq. (27.2a) we added the random term N_{i-1} to perturb the system. Without this random term the time series reflecting the system is drawn to an equilibrium that ends all dynamics (compare the MATLAB scripts pr27_2.m and pr27_3.m); in this case the autocorrelation is similar to the one in Fig. 27.2A2 but the noise in the return plot is absent, resulting in points determined by equation $x_i = ax_{i-1}$ and most points being around (0, 0). In the quadratic relationship explored in MATLAB script pr27_4.m (Eq. 27.2b), the system's behavior remains dynamic and the addition of a significant noise term is not required and may even destroy the behavior.

An important characteristic of a linear system is that a sine wave input generates a sine wave output with the same frequency in which only the phase or amplitude may change. A nonlinear system, in contrast, may react very differently and alter the waveform and frequency of the output signal with respect to the input. This nonlinear property of a frequency change is also poorly detected by the correlation function. An example is shown in Fig. 27.3, summarizing a correlation study using a sinusoidal

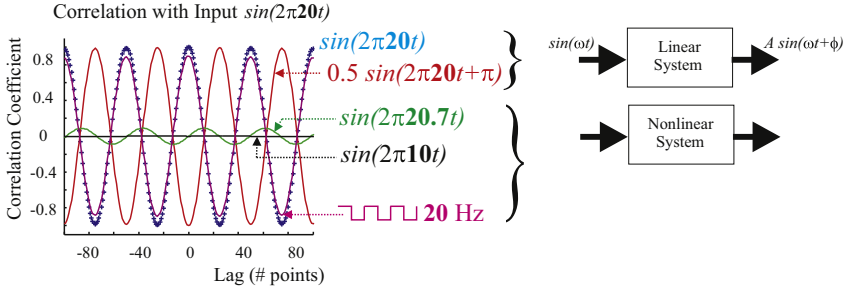


FIGURE 27.3 A correlation study between sine waves. The correlation between the input and output of a system shows a relationship if the frequency (here 20 Hz) remains unaltered. A minor (20.7 Hz) or larger (10 Hz) change in frequency as compared to 20 Hz causes correlation values to drop significantly. However, a rectangular pulse of 20 Hz, representing a seriously distorted sine wave, generates values close to the autocorrelation. The graphs in this figure can be obtained from MATLAB script pr27_6.m.

input of 20 Hz ($\sin(2\pi 20t)$). As expected, the autocorrelation (waveform indicated with blue + in Fig. 27.3) detects the correlation. Cross-correlation between sine waves of the same frequency of 20 Hz, but with different amplitude and phase also works well (red waveform in Fig. 27.3). However, as soon as we assume a nonlinear system that alters the frequency from 20 to 20.7 Hz (green waveform in Fig. 27.3) or even 10 Hz (black waveform in Fig. 27.3), the cross-correlation procedure applied to the time series does not detect a clear relationships.

27.4 EMBEDDING

In the previous section we confirmed our suspicion that linear analysis techniques may perform poorly when studying nonlinear dynamics. The return plots in Fig. 27.2 however, were fairly effective in showing the relationship within the different types of time series. In the return plot we depicted the relationship between two points delayed by a single sample point; this approach can be extended both to include more points and delays. Such a multidimensional version of the scatterplot is the conceptual basis for a powerful technique in the analysis of dynamical systems, the so-called ***embedding*** procedure. Embedding of a time series x_t (x_1, x_2, x_3, …, x_N) is done by creating a set of vectors X_i such that:

$$X_i = \left[x_i, x_{i+\Delta}, x_{i+2\Delta}, \ \ldots, \ x_{i+(m-1)\Delta}\right] \tag{27.3}$$

where Δ is the delay in the number of samples and m is the number of samples (dimension) of the vector. When embedding a time series we must choose the dimension m of X_i and the delay Δ, such that each vector X_i represents values that reveal the topological relationship between subsequent points in the time series. The number of samples in the embedded vector is usually chosen to be large enough to cover the dominant frequency in the time series, but m should not be so large that the first and last values in the epoch are practically unrelated. The evolution of the system can now be represented by the projection of the vectors X_i onto a trajectory through multidimensional space, often referred to as ***phase space*** or ***state space***. If the multidimensional evolution converges to a subspace within the phase space, this subspace is called the ***attractor*** of the system. The construction and characterization of system attractors plays a major role in the analysis of time series. As was proven mathematically, the attractor characterized by embedding a single variable (e.g., a single channel of EEG or ECoG) can characterize the potentially higher-dimensional system that generated the time series (Takens, 1981). Measures that are commonly used to describe the attractor in phase space are ***dimension***, ***entropy***, and ***Lyapunov exponents***. For the

dimension and entropy measures several "flavors" exist and a multitude of algorithms for each of these metrics have been developed over the past decades.

Examples of time series and a two-dimensional embedding are shown in Fig. 27.4. The upper time series (Fig. 27.4A) is an example of the swing of a pendulum and the associated plot shows a strict relationship between past and future points. The next example (Fig. 27.4B) shows a random time series where the embedded vector shows no specific relation between successive points. The example in Fig. 27.4C is from the logistic Eq. (27.1). Interestingly, from visual inspection the time series generated by the random process and the logistic iterator don't seem that different. However, by plotting x_t versus x_{t-1}, one can see that one time series shows a random relationship and the next has a fairly simple attractor characterized by the quadratic relationship from Eq. (27.1). The time series embedding in Fig. 27.4D is characterized by more complex relationships of a type often referred to as a strange attractor. This strange attractor represents a more intricate geometry than that of the curved line in the quadratic relationship, but is more confined in space than the random process, which covers the whole area of the plot. Both time series in Fig. 27.4D and E are examples of time series generated by the Henon map,

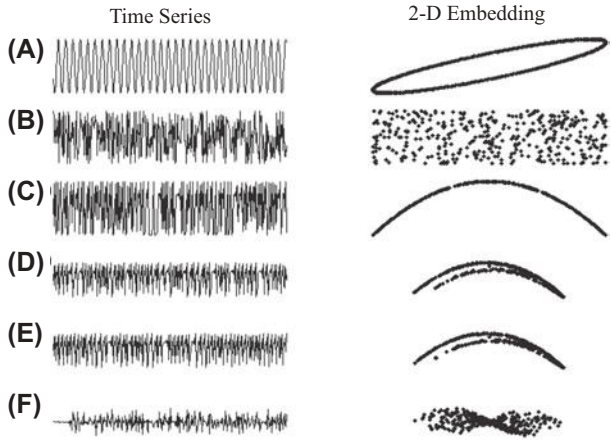


FIGURE 27.4 Examples of time series (left column) and embedding in two dimensions (right column). (A) Sinusoidal signal, (B) random signal, (C) time series determined by the logistic equation ($x_t = 4x_{t-1}[1\_x_{t-1}]$; $x_0 = 0.397$), (D, E) two examples of a Henon map ($x_t = y_{t-1} + 1 - ax^2{}_{t-1}$; $y_t = bx_{t-1}$, $a = 1.4$, $b = 0.3$). The initial conditions differ between (D) $x_0 = 0$; $y_0 = 0$ and (E) $x_0 = 10^{-5}$; $y_0 = 0$. (F) The difference between (D) and (E) shows that initially both time series develop along a similar path (difference → 0). However, after ~25 iterations the difference in initial condition causes a disproportionate difference in the values of the time series. This figure can be produced with MATLAB script pr27_7.m. *From Van Drongelen, W., Lee, H.C., Hecox, K.E., 2005. Seizure prediction in Epilepsy. In: He, B. (Ed.), Neural Engineering, Kluwer Academic/Plenum, NY, pp. 389–420.*

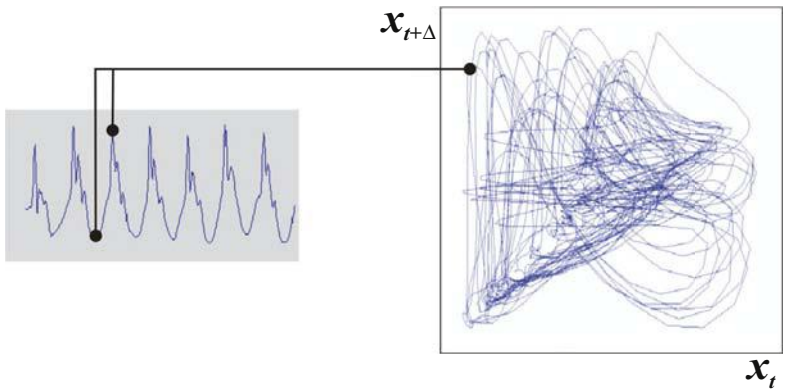


FIGURE 27.5 An example of embedding of an EEG signal during an epileptic seizure in two dimensions. Two points, $x_{t+\Delta}$ and x_t, of the time series are plotted as one single point in a two-dimensional state space diagram. By embedding all subsequent pairs in the same manner, a 2-D projection of the attractor is obtained. *From Van Drongelen, W., Lee, H.C., Hecox, K.E., 2005. Seizure prediction in Epilepsy. In: He, B. (Ed.), Neural Engineering, Kluwer Academic/ Plenum, NY, pp. 389–420.*

a classic chaotic iterator that defines the coevolution of two variables x_t and y_t. Both plots in Fig. 27.4D and E show x_t, but with only slightly different initial conditions: (0, 0) in Fig. 27.4D and (10^{-5}, 0) in Fig. 27.4E. The difference between the two closely related time series in Fig. 27.4D and E is shown in Fig. 27.4F, clarifying the sensitivity to a small perturbation (in this example 10^{-5}). Initially the difference between the two time series is small, but after 25 iterations the difference grows disproportionately until this error is of the same order of magnitude as the time series amplitudes themselves (Fig. 27.4D and E). This phenomenon illustrates that the point that even with knowledge of initial conditions to a precision of 10^{-5} ***chaotic processes are only weakly predictable***: i.e., the observed values may deviate considerably after only a few time steps.

An illustration of embedding of a measured EEG signal during an epileptic seizure is shown in Fig. 27.5. To demonstrate the principle of embedding, we show a two-dimensional depiction despite the fact that two dimensions are likely insufficient to capture the full dynamics of the EEG.

27.5 METRICS FOR CHARACTERIZING NONLINEAR PROCESSES

27.5.1 Attractor Dimension

Measures of dimensionality are used to characterize the geometry of an attractor in space. Several flavors of the dimension metric are currently in

use. An overview of the relationships between the different dimension measures (the so-called Renyi dimensions) would be beyond the scope of this chapter and can be found in Peitgen et al. (1992). Theoretically central among these measures is the **capacity dimension** *D_Cap* of an attractor, which can be estimated with a box-counting algorithm. This metric formalizes our observation that while random fluctuations fill state space, chaotic attractors are limited to a restricted subspace (see Fig. 27.4). The procedure estimates the space that is occupied by the attractor in terms of the number of hypercubes, or "boxes," $N(s)$ with size s in which points of the attractor are located (Fig. 27.6A):

$$D\_Cap = \lim_{s \to 0} \frac{\log_{10} N(s)}{\log_{10}(1/s)} \tag{27.4}$$

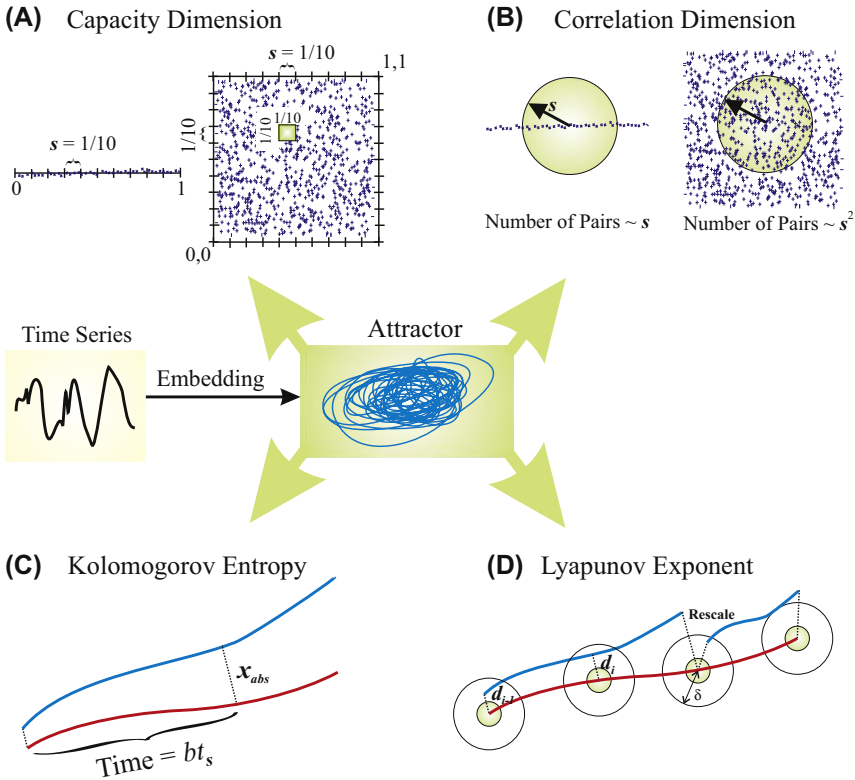


FIGURE 27.6 Simplified diagrams that reflect the algorithms to estimate measures that characterize an attractor. Two metrics (capacity (A) and correlation dimension (B)) reflect the distribution of the attractor in space; the other two measures (Kolmogorov entropy (C) and Lyapunov exponent (D)) quantify the divergence of initially close trajectories.

We will not provide a mathematical proof of this equation, but its rationale can easily be determined with a few examples (Fig. 27.6A). A single line segment of 1 m can be subdivided into 10 units ($N(s) = 10$) of 0.1 m (i.e., scale $s = 0.1$); this results in a dimension of 1, i.e.,:

$$\frac{\log_{10}10}{\log_{10}(1/0.1)} = 1$$

A surface of $1 \times 1\ \text{m}^2$ includes 100 boxes ($N(s) = 100$) of $0.1 \times 0.1\ \text{m}^2$ (i.e., scale $s = 0.1$); applying this to Eq. (27.4), the dimension is 2:

$$\frac{\log_{10}100}{\log_{10}(1/0.1)} = 2$$

Critically, this technique also works for different scales, for instance, a cube of $1 \times 1 \times 1\ \text{m}^3$, can be subdivided into 1000 small cubes of $0.1 \times 0.1 \times 0.1\ \text{m}^3$, and 1000,000 small cubes of $0.01 \times 0.01 \times 0.01\ \text{m}^3$, etc. In this example the number of small cubes versus the inverse of the size (s) scales as: $(1/s)^3$, the power being the capacity dimension of the cube:

$$\frac{\log_{10}1000}{\log_{10}(1/0.1)} = 3 \text{ and } \frac{\log_{10}1000,000}{\log_{10}(1/0.01)} = 3$$

For different sizes of s, the value of the number of boxes $N(s)$ scales according to a power law: $N(s) \propto (1/s)^{D\_Cap}$. Applying the same box counting and scaling procedure for more irregular structures, such as an attractor embedded in a hypercube can generate a ***noninteger value*** between 2 and 3 for the dimension. The smaller the size of the box in the counting procedure, the more precisely the area/volume/etc. covered by the attractor can be described. Unfortunately, a reliable small-box count necessarily requires an attractor that is known in great detail, i.e., many points are available to characterize the attractor's space. For measured time series, such large data sets are often not available. The use of larger boxes is easier to accomplish but reflects the attractor's dimension less precisely. For this reason, the capacity dimension isn't attractive for application to measured time series. Another measure that is related to *D_Cap* is the information dimension. This measure relates to the entropy, the distribution, and the local density of the attractor's points in space. In box-counting terms, one counts the number of boxes occupied in space and weights the box by the number of points it includes. Like capacity dimension, the computational burden of estimating the information dimension prevents it from being frequently used in experimental work.

The most popular dimension measure is the so-called **correlation dimension**. A metric derived from the so-called correlation integral is:

$$C(s) = \left[\frac{1}{N(N-1)}\right] \sum_{i \neq j} U(s - |X_i - X_j|) \tag{27.5}$$

With $U =$ Heaviside (unit step) function, and $N =$ the number of points. The term $|X_i - X_j|$ denotes the distance between the points in state space. The summation (Σ) and the Heaviside function count the vector pairs (X_i, X_j) with an interpoint distance smaller than the threshold s, because $U(\cdots)$ is one if this distance is smaller than s, and zero in all other cases, i.e.,:

$$U(s - |X_i - X_j|) \quad \begin{cases} 1 & \text{for } s - |X_i - X_j| > 0 \rightarrow |X_i - X_j| < s \\ 0 & \text{otherwise} \end{cases}$$

The value of $C(s)$ in Eq. (27.5) is a measure of the number of pairs of points (X_i, X_j) on the reconstructed attractor whose distance is smaller than a set distance (Fig. 27.6B). The expression in Eq. (27.5) is applied to discrete time data, therefore the correlation integral contains a summation rather than an integral. In the examples in Fig. 27.6B, it can be seen that a line structure within a circle of radius s creates a set of pairs satisfying $|X_i - X_j| < s$ that is proportional to s, while a 2-D distribution creates a number of pairs proportional with s^2. Generally, for a large number of points (N) and small distances (s), $C(s)$ scales according to a power law $C(s) \propto s^{D\_Cor}$, where D_Cor is the correlation dimension of the attractor.

27.5.2 Kolmogorov Entropy

Another metric that can characterize the dynamics of an attractor is the order-2 Kolmogorov entropy. Order-2 Kolmogorov entropy is a measure of the rate at which information about the state of a system is lost, and it can be estimated by examination of two initially close orbits in an attractor. The idea is that by selecting two neighboring points on an attractor, the evolution of one of the trajectories is informative about the other trajectory as long as they stay close. As soon as the trajectories diverge, the information about one trajectory relative to the other is lost. The time interval (t) required for the orbits to diverge beyond a set distance satisfies a distribution:

$$C(t) \propto e^{-E_K t} \tag{27.6}$$

where E_K is the Kolmogorov entropy. Schouten et al. (1994) described an efficient maximum-likelihood method of estimating E_K. Their method

assumes a time series of N points that is uniformly sampled at intervals of t_s; under these assumptions, Eq. (27.6) becomes:

$$C(b) = e^{-E_K t_s b} \tag{27.7}$$

where b represents the number of time steps required for pair separation beyond the set criterion. They then show that the maximum likelihood estimate of the Kolmogorov entropy E_K^{ML} (in bits per second) is:

$$E_K^{ML} = -\frac{1}{t_s}\left[\log_2\left(1 - \frac{1}{b_{avg}}\right)\right] \tag{27.8}$$

where b_{avg} is the average number of steps required for initially close pairs to diverge. The diagram in Fig. 27.6C shows the principle of determining Kolmogorov entropy from nearby trajectories in an attractor.

To collect the necessary b's in Eq. (27.8), methods of choosing nearby independent points as well as determining the divergence threshold are needed. Schouten et al. (1994) suggest estimating these from the data in the following way. First, the data are demeaned and divided by (normalized to) the average absolute deviation (x_{abs}) of the demeaned data:

$$x_{abs} = \frac{1}{N}\sum_{i=1}^{N} |x_i|,$$

where N is the number of sample points. The value x_{abs} is then used as an estimate of the divergence threshold; i.e., a value of 1 in the normalized data set. Second, the number of cycles in the time series is estimated as ½ of the number of zero crossings; this is used to calculate the number of samples/cycle m, which is used as the independence criterion. This criterion indicates that two points that are separated by at least m samples belong to a different cycle in the time series, and can therefore be considered independent. Therefore, the algorithm proceeds by selecting a pair of samples in the data at randomly chosen time steps i and j; if they are separated by at least m time steps ($|i - j| \geq m$), then they are considered to be independent, and therefore eligible for use in the following calculations. The largest of m absolute differences between pairs of values starting at i and j constitutes the ***maximum norm***, which is the distance metric used in this algorithm:

$$d = \max(|x_{i+k} - x_{j+k}|) \tag{27.9}$$

for $0 \leq k \leq m - 1$; if $d \leq 1$ (remember that this threshold of 1 corresponds to a threshold of x_{abs} in the nonnormalized data), the samples are considered nearby. Finally, having found a pair of randomly chosen, nearby, independent data points, the number of steps b needed for them to diverge (such that at least one pair exceeds the criterion, i.e.,: $|x_{i+m-1+b} - x_{j+m-1+b}| > 1$ can be added to the set used to calculate b_{avg}.

The above thresholds for determining independence and divergence work reasonably for many data sets, but we must stress that x_{abs} and m are heuristics that provide reasonable guidelines; they may yield better results for some data sets if modified by a factor of order unity.

27.5.3 Lyapunov Exponent

To begin with a trivial statement: an attractor wouldn't be an attractor if there wasn't attraction of trajectories into its space. On the other hand, an attractor wouldn't represent a chaotic process if neighboring trajectories within its space didn't diverge exponentially fast. The Lyapunov exponent describes speed of attraction (convergence) or divergence of trajectories in each dimension of the attractor. We indicate the exponent in the ith dimension as λ_i, describing the rate at which the distance between two initially close trajectories changes over time as an exponent: e^{λ_i}. A value of $\lambda_i > 0$ indicates there is divergence and $\lambda_i < 0$ indicates convergence in the ith dimension. In two dimensions, the sum of the two exponents determines how a surface in the ith and $(i+1)$th dimension evolves: $e^{\lambda_i}\ e^{\lambda_{i+1}} = e^{\lambda_i + \lambda_{i+1}}$. In three dimensions, three Lyapunov exponents describe the evolution of a cube, and the sum of all Lyapunov exponents indicate how a so-called hypercube evolves in a multidimensional attractor. In order to show divergence, and the chaotic signature of sensitivity to initial conditions, the largest Lyapunov exponent determined in an attractor of a chaotic process must be > 0. Therefore, the characterization of recorded signals by the Lyapunov exponent is usually focused on the largest exponent. The largest exponent describes the expansion along the principal axis (p_i) of the hypercube over a given time interval t. Formally, the exponent (λ_i) is calculated as:

$$\lambda_i = \lim_{t \to \infty} \frac{1}{t} \log_2 \left[\frac{p_i(t)}{p_i(0)} \right] \tag{27.10}$$

Wolf et al. (1985) developed an algorithm to estimate the largest Lyapunov exponent in a measured time series. The algorithm described by Wolf et al. (1985) was revised for application to electroencephalogram (EEG) and electrocorticogram (ECoG) time series by Iasemidis et al. (1990). The procedure is shown in Fig. 27.6D: the principle is to select a trajectory in the embedded time series and to determine a second point nearest to the starting point of this fiducial trajectory (red in Fig. 27.6D). The nearest point must not be too close because then the pair might be dynamically equivalent and only separated by some measurement noise (symbolized by the small green circle in Fig. 27.6D). Subsequently the trajectories from this and the starting point are followed for a fixed time

interval. The initial distance d_0 and the distance d_1 after the time interval are measured. If the distance d_1 is smaller than a preset value (the larger circle with radius δ in Fig. 27.6D), the procedure is repeated. Fig. 27.6D shows an example of two initially close trajectories (blue and red) and their start and end positions. If the distance between the end positions (d_1) grows larger than the preset value δ, an attempt is made to rescale the distance by searching for a new point closer to the reference trajectory. Because we want to determine the Lyapunov exponent in a given dimension, we must stay within that same dimension and the rescaling procedure must find a new neighboring point that satisfies this condition. In reality this is the critical component of the algorithm because in measured signals a nearby point in the same dimension may not be available, making the rescaling a challenge. This procedure of measuring the interdistance at the start and end of these trajectories is repeated k times to cover the measured attractor from t_0 to t_k, and the largest Lyapunov exponent ($\lambda_{\max}$) is calculated as:

$$\lambda_{\max} = \frac{1}{t_k - t_0} \sum_{i=1}^{k} \log_2 \left[\frac{d_i}{d_{i-1}} \right] \tag{27.11}$$

Both the value of the largest Lyapunov exponent and the Kolmogorov entropy describe how quickly nearby trajectories diverge and therefore relate directly to the predictability of the underlying process. For the Kolmogorov entropy estimation the interpoint distance is set and the time of divergence is measured, whereas for estimation of the largest Lyapunov exponent it is the other way around. For the Kolmogorov entropy estimation, close trajectories are selected randomly, while for the Lyapunov exponent the procedure covers the attractor sequentially (Fig. 27.6C and D). Large values of both measures indicate an important divergence of trajectories that are initially close. As in the example of the Henon map in Fig. 27.4D–F, small perturbations or inaccuracies in the initial state or in the calculation of subsequent values in a time series will create large differences after only a few iterations, thus limiting the potential for accurate prediction over a longer interval.

27.5.4 Surrogate Time Series

An important question when applying nonlinear time series analysis to recorded data is the nature of the underlying process. The algorithms for computing nonlinear metrics are constructed such that they will provide an estimate even when the underlying process is actually random or linear. Therefore, assigning a dimension value, Lyapunov exponent, etc. is

not a guarantee that the time series is actually generated by a nonlinear dynamical process. To determine whether a data set contains nonlinearities, several methods have been developed in which surrogate data sets are generated and compared against the measured data (e.g., Kaplan and Glass, 1995). If we want to test linearity versus nonlinearity, we can compute one of the nonlinear measures for both the measured time series and for a surrogate time series generated by some linear model of the system. A common approach is to estimate the linear model that generates the surrogate time series from the measured data itself. Subsequently, the values of the nonlinear measure obtained from the real data and a set of surrogate time series are compared. The null hypothesis is that the value of the computed nonlinear measure can be explained from the linear model, and if the null hypothesis is rejected, a nonlinear process may have generated the original data. The procedure to obtain surrogate data depends on the null hypothesis at hand. If the null hypothesis is that the data originate from a purely random process, a random shuffle of the measured data is sufficient to generate a surrogate time series. Another commonly applied null hypothesis is to assume that the underlying process is stationary, linear, and stochastic. A commonly applied technique to obtain surrogate time series satisfying this hypothesis is to compute the fast Fourier transform (FFT) followed by a randomization of the phase. The inverse FFT generates a surrogate time series representing linearly correlated noise with the same power spectrum and autocorrelation as the original signal but with the higher-order timing relationships destroyed. Methods of surrogate time series comparison provide a relatively robust technique for the task of making the presence of underlying nonlinearity plausible. Although nonlinearity is a prerequisite for the existence of chaos, similarly, objective tests to demonstrate an underlying chaotic process in measurements do not exist.

27.6 APPLICATION TO BRAIN ELECTRICAL ACTIVITY

Extraction of nonlinear metrics from brain activity reflected in EEG and ECoG (Chapter 1) time series has been used to anticipate or detect epileptic seizures or to describe sleep stages. Automated detection of sleep stages is based on the idea that the EEG rhythm during different stages of vigilance will be reflected in the metrics for dimensionality, entropy, etc. The assumption behind anticipating epileptic events is based on the hypothesis that such seizures are preceded by a so-called preictal state in which the processes leading to the ictal state (the seizure) take place. Although the use of nonlinear metrics in predicting epileptic events is still somewhat controversial, it does appear that at least in a number of

cases of preictal state can be detected prior to the clinical onset of the seizure. The epoch over which such detection occurs varies from seconds to hours prior to the clinical seizure. Further details of this topic may be found in a recent overview by van Drongelen et al. (2005).

EXERCISES

27.1 Use a MATLAB routine to compute 1024 points of a time series using $x_{n+1} = u{*}x_n{*}(1{-}x_n)$. For each step in the iteration the parameter u is dynamically defined as: $u = 0.9 + 3{*}\text{rand}(1)$. (Use the rand function and NOT randn)
- a. Compute mean, standard deviation.
- b. Comment on stationarity.
- c. Compute correlation for x_n and x_{n+1}.
- d. Compute and plot the autocorrelation function.
- e. Plot the return map (x_{n+1}, x_n).
- f. Plot a recurrence map (distances < 0.2).
- g. Interpret your findings.

27.2 Use the logistic equation, Eq. (27.1), to demonstrate the following properties (using MATLAB)
- a. Sensitivity to a (very) small change in initial condition (in the Chaotic Regime).
- b. Quantify how the average of 10 randomly selected paths of close neighboring initial points diverges in different regimes (i.e., check this for values of parameter $a < 3$, between 3 and 3.4495, and above this value). Plot the average divergence against the value of parameter a.
- c. Failure of autocorrelation in detecting a relationship.
- d. Success of the return plot in detecting a relationship.
- e. Interpret your findings.

References

Feigenbaum, M.J., 1983. Universal behavior in nonlinear systems. Phys. D 7, 16–39.

Hodgkin, A.L., Huxley, A.F., 1952. A quantitative description of membrane current and its application to conduction and excitation in the nerve. J. Physiol. 117, 500–544. *A seminal paper describing the Hodgkin and Huxley equations.*

Iasemidis, L.D., Sackellares, J.C., Zaveri, H.P., Williams, W.J., 1990. Phase space topography and the Lyapunov exponent of electrocorticograms in partial seizures. Brain Topogr. 2, 187–201.

Kaplan, D., Glass, L., 1995. Understanding Nonlinear Dynamics. Springer-Verlag, New York, NY.

An excellent introduction into the application of nonlinear dynamics to analysis of time series. The work includes multiple examples and computer projects.

Lorenz, E.N., 1963. — Deterministic non-periodic flow. J. Atmos. Sci. 20, 130—141.

Peitgen, H.-O., Jürgens, H., Saupe, D., 1992. Chaos and Fractals New Frontiers of Science. Springer-Verlag, New York.

An introduction into the field of nonlinear dynamics with a wealth of examples. Clear explanation of concepts such as self-similarity, dimensionality, entropy, and Lyapunov exponents are provided. Throughout the text a minimal background in mathematics is required.

Recio-Spinoso, A., Temchin, A.N., van Dijk, P., Fan, Y.-H., Rugero, M.A., 2005. Wiener-Kernel analysis of responses to noise of chinchilla auditory-nerve fibers. J. Neurophysiol. 93, 3615—3634.

Schouten, J.C., Takens, F., van den Bleek, C.M., 1994. Maximum-likelihood estimation of the entropy of an attractor. Phys. Rev. E 49, 126—129.

Strogatz, S.H., 1994. Nonlinear Dynamics and Chaos. Perseus Books, Cambridge, MA.

Takens, F., 1981. Detecting strange attractors in turbulence. In: Rand, D.A., Young, L.S. (Eds.), Lecture Notes in Mathematics, Dynamical Systems and Turbulence, vol. 898. Springer-Verlag, Berlin, pp. 366—381.

Van Drongelen, W., Lee, H.C., Hecox, K.E., 2005. Seizure prediction in Epilepsy. In: He, B. (Ed.), Neural Engineering. Kluwer Academic/Plenum, NY, pp. 389—420.

An introduction into the application of nonlinear dynamics for the prediction of seizure activity in EEG signals.

Wolf, A., Swift, J.B., Swinney, H.L., Vastano, J.A., 1985. Determining Lyapunov exponents from a time series. Phys. D 16, 285—317.

CHAPTER

28

Decomposition of Multichannel Data

28.1 INTRODUCTION

In previous chapters we have mainly focused on the analysis of single input–single output systems, single channel data, or single images. Even when we analyzed images we worked with one row or column of pixels at a time. At most we considered pairs of signals when we determined cross-correlation, coherence, or when we looked into input–output relationships. Although these techniques form an important basis for analysis in neuroscience research, current studies usually collect multiple channels of neural activity and/or movies.

Examples of commonly encountered multichannel data sets are 64-, 128-, or 256-channel ElectroCorticoGrams (ECoGs), a sequence of functional Magnetic Resonance Images (fMRI), recordings from microelectrode arrays with hundreds to thousands of channels, or movies made from neural tissue with voltage-sensitive dyes. In these examples we deal not just with two or three simultaneously recorded signals, but with potentially overwhelming numbers of channels representing both spatial and temporal components. In the ECoG each channel is recorded at a certain location and the signals evolve over time; in both the fMRI sequence and the movie, the signals can be represented as the intensity of each pixel as a function of time. Suppose we digitized the fMRI set with 128 samples in time and each image is 128×128 pixels, we now have a huge data set consisting of $128^3 = 2{,}097{,}152$ points. Say we sample 128 ECoG signals at a rate of 400 samples channel^{-1} second^{-1} and we record for 60 s, we now have a 1 min data set of $128 \times 400 \times 60 = 3{,}072{,}000$ data points. Examples of multichannel data sets of brain electrical activity are shown in Fig. 28.1.

Typical approaches to analyze multichannel data are data reduction, decomposition, or investigating the causal structure within the data. In the

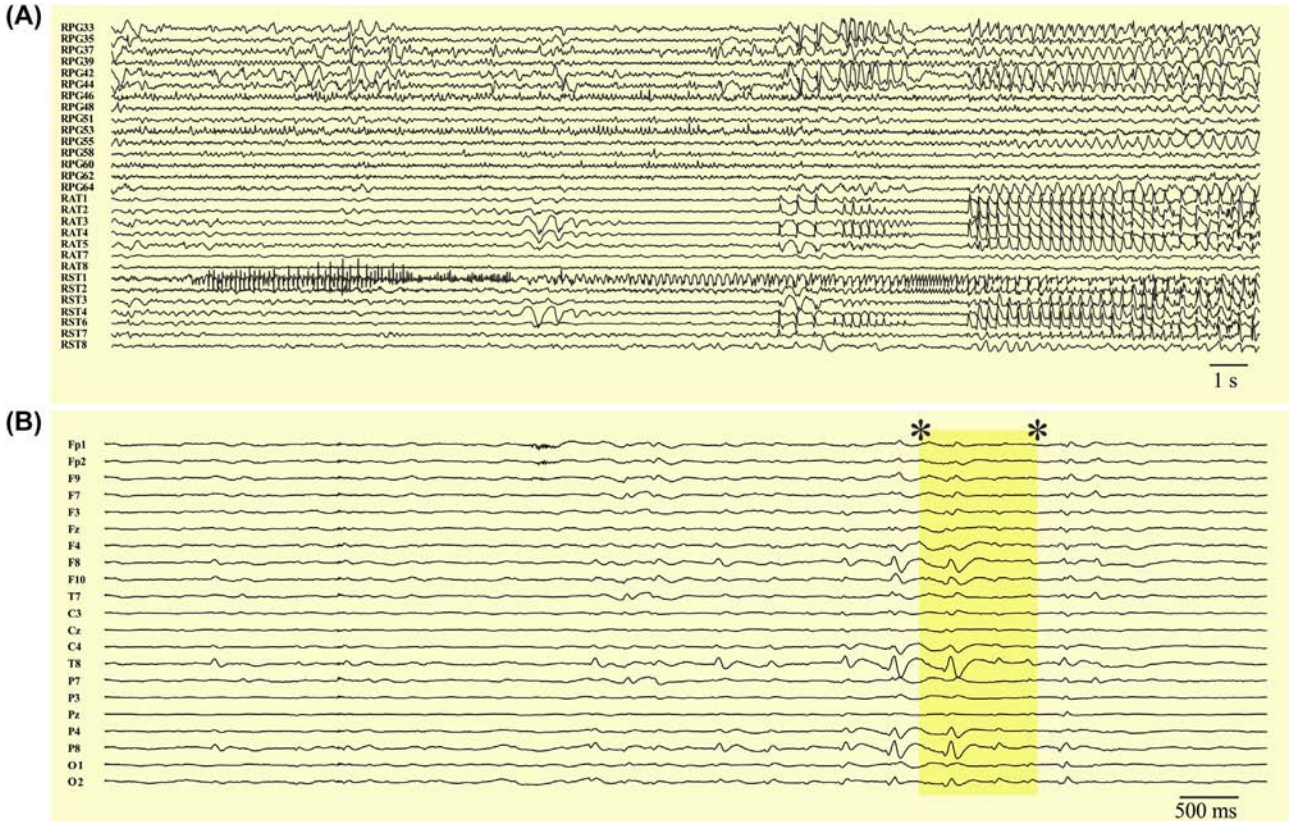


FIGURE 28.1 (A) A subset of an intracranial 128-channel recording from a patient with epilepsy. Only a limited number of data are shown: 15 channels of a right parietal grid (RPG), seven channels from a right anterior temporal strip (RAT), and seven from a right superior temporal strip (RST). The activity associated with a seizure onset starts with rapid oscillations in channel RST1 and eventually results in a spread of oscillatory activity in almost all channels. (B) A 21-channel scalp EEG recording from a different patient with epilepsy. The large-amplitude waveforms in this recording are epileptic spikes. The interval between the * (highlighted in yellow) is used for further analysis in Fig. 28.14.

case of data reduction, the idea is to find the signal and noise components, and the goal for decomposition is to find the basic signal components that are included in the multichannel data set. Of course, both these approaches are similar and can be related. Suppose we have a measurement of brain activity during a task. The activity associated with the task is signal and the remainder of the activities can be considered noise. If we can decompose our brain activity in these basic components we have effectively used decomposition as a tool for data reduction. Although the goal is to analyze multichannel data sets with large numbers of signals, we will demonstrate examples with reduced data sets with only a few channels. Using this approach, we will illustrate the principles that are valid in high-dimensional data space with 2-D or 3-D examples. Furthermore, throughout this chapter, we will demonstrate principles rather than formally prove them. First we will show the general principle of mixing and unmixing of signals (Section 28.2), then we will go into the details of specific strategies: principal component analysis (PCA) (Section 28.3) and independent component analysis (ICA) (Section 28.4). If your interests in multichannel data analysis go beyond the introduction here, see specialized texts on this topic (e.g., Stone, 2004; Bell and Sejnowski, 1995), on linear algebra (e.g., Jordan and Smith, 1997; Lay, 1997), and information theory (e.g., Shannon and Weaver, 1949; Cover and Thomas, 1991).

28.2 MIXING AND UNMIXING OF SIGNALS

The underlying model of decomposition is that the recordings we are looking at are mixtures of signals generated by several sources. For instance, our ear detects sounds from different sources simultaneously such as someone talking to us, the noise of a fan, and music from a radio. Another example is an ECoG electrode that picks up electrical activity from different parts of the brain. In other words, a set of measured signals Y_n consists of channels that are mixtures from a number of sources X_m. For now, we assume that the number of sources is equal to or smaller than the number of signals we record. In this case the problem of mixing and unmixing can be defined in a straightforward fashion.

Let us consider a concrete example with three sources X_1–X_3 and three sensors Y_1–Y_3 (Fig. 28.2). The sensors pick up the signal from each source but the signal is attenuated when traveling from source to sensor and the attenuation is proportional with distance. The level of attenuation in Fig. 28.2 is indicated by the width of the arrows: the signal from the source closest to the sensor attenuates least, only by a factor of 0.8. The other sources attenuate more, depending on distance by a factor of 0.6 or a factor of 0.2, respectively. If we look at sensor Y_1 we can see that it will pick up 0.8× the signal from source X_1 plus 0.6× the signal from source X_2 plus 0.2× the signal from source X_3. Note that we assumed that the mixing of the signals that originate from X_1, X_2, and X_3 is a linear process.

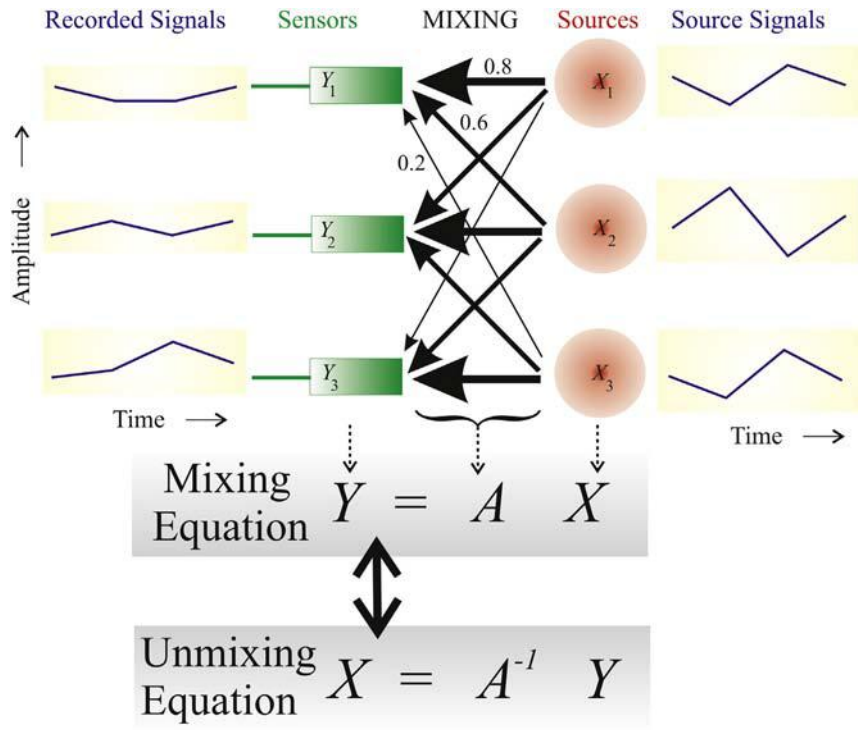


FIGURE 28.2 A mix from signals originating from a set of three sources X (X_1–X_3) is recorded by a set of three sensors Y (Y_1–Y_3). The signal from source to sensor is attenuated with distance. The amount of attenuation is symbolized by the width of the *arrows* (there are only three widths in this example: 0.8, 0.6, or 0.2). The mixing process can be represented by the matrix multiplication $Y = AX$ in which A is the mixing matrix. The unmixing process can be represented by $X = A^{-1}Y$ in which the unmixing matrix A^{-1} is the inverse of A.

Similar rules for the mixing process can be determined for the other sensors Y_2 and Y_3. Accordingly, the measurements at the three sensors in Fig. 28.2 can be summarized as:

$$Y_1 = 0.8X_1 + 0.6X_2 + 0.2X_3$$
$$Y_2 = 0.6X_1 + 0.8X_2 + 0.6X_3$$
$$Y_3 = 0.2X_1 + 0.6X_2 + 0.8X_3$$

As will be outlined in the following, because these are three equations with three unknowns X_1–X_3 (note that Y_1–Y_3 are known because they are measured), we can solve for the source signals. First we can put the three equations in matrix form:

$$\boxed{Y = AX} \tag{28.1a}$$

In which $Y = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix}$, $X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}$, and mixing matrix $A = \begin{bmatrix} 0.8 & 0.6 & 0.2 \\ 0.6 & 0.8 & 0.6 \\ 0.2 & 0.6 & 0.8 \end{bmatrix}$ represents the attenuation coefficients for the setup in Fig. 28.2. The fact that $A(i,j) = A(j,i)$, i.e., mixing matrix A is symmetric, is due to the symmetrical setup of the example presented in Fig. 28.2; this is not necessarily the case for any mixing matrix. Now suppose we have recorded four samples from the three sensors and we want to know the signals from the individual sources. Because we know that $Y = AX$ we can compute X by $X = A^{-1}Y$, where A^{-1} is the inverse of A (here we assume that the inverse of matrix A exists). If we take an example where the sources emit the following signals at times t_1 to t_4 (indicated as Source Signals in the right-hand-side panels in Fig. 28.2):

	t_1	t_2	t_3	t_4
X_1:	24.1667	1.6667	33.3333	17.5000
X_2:	−32.5000	0.0000	−55.0000	−22.5000
X_3:	20.8333	3.3333	41.6667	17.5000

Our sensors will pick up (in the following we only show two decimal precision):

$$Y = AX = \begin{bmatrix} 0.8 & 0.6 & 0.2 \\ 0.6 & 0.8 & 0.6 \\ 0.2 & 0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 24.17 & 1.67 & 33.33 & 17.50 \\ -32.50 & 0.00 & -55.00 & -22.50 \\ 20.83 & 3.33 & 41.67 & 17.50 \end{bmatrix}$$

$$= \begin{bmatrix} 4.00 & 2.00 & 2.00 & 4.00 \\ 1.00 & 3.00 & 1.00 & 3.00 \\ 2.00 & 3.00 & 7.00 & 4.00 \end{bmatrix}$$

So our measurement, indicated as Recorded Signals in the left-hand-side panels in Fig. 28.2, will be:

	t_1	t_2	t_3	t_4
Y_1:	4.00	2.00	2.00	4.00
Y_2:	1.00	3.00	1.00	3.00
Y_3:	2.00	3.00	7.00	4.00

Because we know mixing matrix A, we compute its inverse (if you want to check this example in MATLAB®, the inverse of a matrix can be obtained with the inv command) so that we can estimate the source activity $\widehat{X}$ from the measurements:

$$\boxed{\widehat{X} = A^{-1}Y} \tag{28.1b}$$

For our example that is:

$$\underbrace{\begin{bmatrix} 24.17 & 1.67 & 33.33 & 17.50 \\ -32.50 & -0.00 & -55.00 & -22.50 \\ 20.83 & 3.33 & 41.67 & 17.50 \end{bmatrix}}_{\widehat{X}} = \underbrace{\begin{bmatrix} 5.83 & -7.50 & 4.17 \\ -7.50 & 12.50 & -7.50 \\ 4.17 & -7.50 & 5.83 \end{bmatrix}}_{A^{-1}} \underbrace{\begin{bmatrix} 4.00 & 2.00 & 2.00 & 4.00 \\ 1.00 & 3.00 & 1.00 & 3.00 \\ 2.00 & 3.00 & 7.00 & 4.00 \end{bmatrix}}_{Y}$$

As you can see our estimate $\widehat{X}$ for X is very good (note that this example was computed at a four decimal precision while only two decimals are shown here). Although this example clarifies the mixing and unmixing process, it usually isn't helpful in practical applications. Even if we ignore the effects of noise that would be present in any real recording, these inverse problems are often ill-posed because the mixing matrix is unknown, and the number of sources outnumbers the number of sensors. In the remainder of this chapter we will focus on what one can do if the mixing matrix is unknown. In this case we want to separate the sources while we are "blind" for the mixing process, and therefore this procedure is called Blind Source Separation (BSS). We will specifically focus on two of these techniques: PCA and ICA.

28.3 PRINCIPAL COMPONENT ANALYSIS

In this section we introduce the concept of decomposing multichannel data into its principal components. With PCA of multidimensional measurements, one can find the directions in the multidimensional measurement space that are arranged according to the amount of variance in the measurements. In addition, we will see that these directions are mutually orthogonal, reflecting that the components extracted with the PCA procedure are uncorrelated. We will introduce the technique by analyzing a concrete 3-D example of four measurements S_1–S_4, each observation S_n

having three values or signals s_1, s_2, and s_3 (one for each of the three dimensions) arranged in a column vector:

$$S_1 = \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix} \quad S_2 = \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix} \quad S_3 = \begin{bmatrix} 2 \\ 1 \\ 7 \end{bmatrix} \quad S_4 = \begin{bmatrix} 4 \\ 3 \\ 4 \end{bmatrix}. \tag{28.2a}$$

The mean vector of these four observations, column vector M, contains the mean for each of the three signals m_1, m_2, and m_3:

$$M = \frac{1}{4}\{S_1 + S_2 + S_3 + S_4\} = \frac{1}{4}\left\{ \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \\ 7 \end{bmatrix} + \begin{bmatrix} 4 \\ 3 \\ 4 \end{bmatrix} \right\} \tag{28.2b}$$

$$= \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 4 \end{bmatrix}.$$

A 3-D plot of the observations and their mean is shown in Fig. 28.3A. If we now demean our four observations, that is we subtract M from S_1–S_4, and we arrange the demeaned observations in matrix B, we have:

$$B = \begin{bmatrix} 4-3 & 2-3 & 2-3 & 4-3 \\ 1-2 & 3-2 & 1-2 & 3-2 \\ 2-4 & 3-4 & 7-4 & 4-4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -2 & -1 & 3 & 0 \end{bmatrix}. \tag{28.3}$$

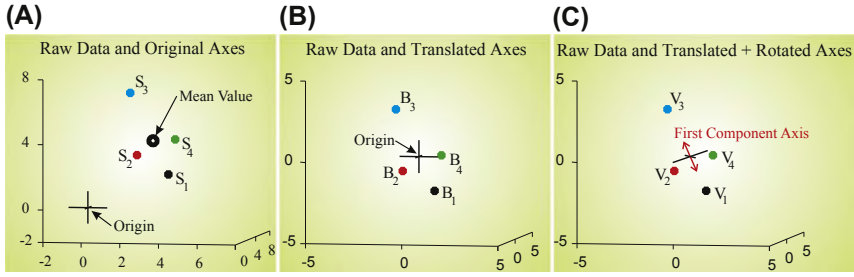


FIGURE 28.3 A numerical example of the application of principal component analysis to multivariate observations. (A) A 3-D plot of four observation vectors S_1–S_4 (Eq. 28.2a) and their mean M (Eq. 28.2b). (B) The same points, now indicated as B_1–B_4 because they are plotted against axes that are translated so that the mean M becomes the new origin. (C) Finally we plot the same points (now indicated as V_1–V_4) against axes that are also rotated to reflect the directions of the three principal components. The first principal component is indicated by the *double arrow* (red). This illustration was made with MATLAB® script pr28_1.m, and the numerical values can be found in Table 28.1.

In statistics, a data set from multichannel observations such as the concatenated matrix $S = [S_1\ S_2\ S_3\ S_4]$ or matrix B is called multivariate data. A scatterplot of the demeaned observations is shown in Fig. 28.3B. Note that the new mean value is now at the origin. From B, the covariance matrix C can now be computed as follows:

$$C = \frac{1}{N-1}BB^T = \frac{1}{3}\begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -2 & -1 & 3 & 0 \end{bmatrix}\begin{bmatrix} 1 & -1 & -2 \\ -1 & 1 & -1 \\ -1 & -1 & 3 \\ 1 & 1 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 1.33 & 0 & -1.33 \\ 0 & 1.33 & -0.67 \\ -1.33 & -0.67 & 4.67 \end{bmatrix}. \tag{28.4}$$

Superscript ^{T} indicates the transpose of matrix B [recall that in a transpose of a matrix, rows and columns are being interchanged: $B^T(i,j) = B(j,i)$]. Because we have three variables (s_1, s_2, s_3) in each observation, the covariance matrix is a 3×3 matrix and N is the number of observations; $N = 4$ in this example. Each diagonal value in C represents the variance of the b_1, b_2, b_3 values of observation vectors B_n. So the sum of the diagonal elements, the trace of C written as $tr(C)$ is the total variance. Each off-diagonal element is a covariance value, for example $C(2,3)$ is the covariance between b_2 and b_3. Of course $C(3,2)$ is the same value because it is the covariance between b_3 and b_2. Therefore a covariance matrix is a **symmetric** matrix (see the example in Eq. 28.4). An alternative way to establish the presence of symmetry for covariance matrices is to show that interchanging the rows and columns (that is transposing) of covariance matrix C results in the same matrix: that is $C = C^T$ because $C(i,j) = C(j,i)$. From Eq. (28.4), ignoring the normalization by $1/N - 1$, we can establish that C is proportional with BB^T. The transposing operation on C can thus be represented by $(BB^T)^T = B^{TT}B^T$. Because the transpose of a transposed matrix is the original matrix again (i.e., $B^{TT} = B$), we may simplify this outcome, that is $B^{TT}B^T = BB^T$, which shows that $(BB^T)^T = BB^T$. Thus the transpose of BB^T is BB^T again and therefore the covariance matrix C, which is proportional to BB^T, must be symmetric.

28.3.1 Finding Principal Components

If there is zero covariance or zero correlation between b_i and b_j, then $C(i,j)$ for $i \neq j$ is zero. It may be clear that analysis of multivariate data would be simpler when all signals are uncorrelated—that is a covariance matrix that is **diagonal** which means that all off-diagonal elements are zero—and **this is exactly the goal of the decomposition with PCA**.

Note that correlation (ρ_{xy}) between two variables x and y is a normalized version of the covariance ($Cov(x,y)$) between x and y, that is: $\rho_{xy} = Cov(x,y)/\sigma_x \sigma_y$, with σ_x and σ_y—the standard deviations for x and y. The effect of this normalization is that the correlation coefficient ρ_{xy} is scaled between -1 and 1.

To summarize the above in other words, the strategy of PCA is to manipulate our demeaned observations B_n $(b_1, b_2, b_3)_n$ for which correlations between b_i and b_j may exist, into transformed data V_n $(v_1, v_2, v_3)_n$ such that all correlations between v_i and v_j for $i \neq j$ vanish. Again, mathematically this means that the covariance matrix C of B may contain nonzero off-diagonal elements (see for example Eq. 28.4) and that the covariance matrix Σ of V must be a diagonal matrix (a matrix with zero off-diagonal elements). To illustrate this procedure, we will employ our numerical example in Fig. 28.3 and use the PCA approach to find the principal components.

Continuing the numerical example above, we will show that the 3×3 covariance matrix C in Eq. (28.4) can be diagonalized by applying a linear transformation. In the following, we will introduce and apply the PCA method first and justify the procedure later. To accomplish this, we first define a 3×3 matrix of orthogonal column vectors $U = [U_1\ U_2\ U_3]$ and a 3×3 diagonal matrix Σ with diagonal entries λ_1–λ_3, and use our demeaned observations in matrix B (Eq. 28.3). We can compute

$$CU = [\,CU_1 \quad CU_2 \quad CU_3\,], \tag{28.5a}$$

and

$$U\Sigma = [\,U_1 \quad U_2 \quad U_3\,]\begin{bmatrix}\lambda_1 & 0 & 0\\ 0 & \lambda_2 & 0\\ 0 & 0 & \lambda_3\end{bmatrix} = [\,\lambda_1 U_1 \quad \lambda_2 U_2 \quad \lambda_3 U_3\,]. \tag{28.5b}$$

Note that Σ is a diagonal matrix. Now we assume that C our covariance matrix is diagonizable and that U is invertible such that

$$C = U\Sigma U^{-1} \text{ and } \Sigma = U^{-1}CU \tag{28.5c}$$

If we right multiply the first expression in Eq. (28.5c) by U we get:

$$CU = U\Sigma. \tag{28.5d}$$

This result indicates that C is indeed diagonizable, then the expressions in Eqs. (28.5a) and (28.5b) must be equal. If we use this finding and equate the individual columns in the matrices in Eq. (28.5d), we get:

$$CU_1 = \lambda_1 U_1, \;\; CU_2 = \lambda_2 U_2, \;\; \text{and} \;\; CU_3 = \lambda_3 U_3. \tag{28.5e}$$

The result in Eq. (28.5e) shows that λ_1–λ_3 and U_1–U_3 must be the eigenvalues and corresponding eigenvectors of the covariance matrix C. See Chapter 9 if you need to review the concept of eigenvalues and eigenvectors; if you need more than a quick review see a text on linear algebra such as Jordan and Smith (1997) or Lay (1997).

In the context of our analysis, it is important to note that because covariance matrix C is a symmetric matrix, its eigenvectors are orthogonal. We can demonstrate this property of symmetric matrices by considering a simple 2-D case where we have two distinct eigenvalues (λ_1 and λ_2) with two corresponding eigenvectors (U_1 and U_2, both scaled to unit length). In order to show that these vectors are orthogonal we show that their scalar product equals zero.

> *Recall* that the inproduct (also called scalar product or dot product) of two vectors $\vec{a}$ and $\vec{b}$ is given by $abcos\varphi$, with a and b the length of the vectors and ϕ the angle between them. If the vectors are orthogonal, ϕ equals 90 degrees, and the outcome of their dot product is zero (See also Appendix 21.1).

We can show that the dot product $U_1 \cdot U_2 = 0$ by computing the following expression

$$\lambda_1 U_1 \cdot U_2 = (\lambda_1 U_1)^T U_2 = (CU_1)^T U_2 = U_1^T C^T U_2 \tag{28.6a}$$

Here we changed the vector dot product notation in vector notation $U_1 \cdot U_2 = U_1^T U_2$ (note the presence of the dot in the expression left from the equal sign), and we used the definition of the eigenvalue/eigenvector of C: $\lambda_1 U_1 = CU_1$ (for an introduction into eigenvalues and eigenvectors, review Chapter 9). We know that C is a covariance matrix that must be symmetric, therefore $C = C^T$. Using this property for symmetric matrices, we get

$$U_1^T C^T U_2 = U_1^T (CU_2) = U_1^T (\lambda_2 U_2) = \lambda_2 U_1^T U_2 = \lambda_2 U_1 \cdot U_2 \tag{28.6b}$$

Note the dot in the last expression. Combining Eqs. (28.6a) and (28.6b), we may conclude that for the symmetric covariance matrix

$$\lambda_1 U_1 \cdot U_2 = \lambda_2 U_1 \cdot U_2 \;\rightarrow\; (\lambda_1 - \lambda_2) U_1 \cdot U_2 = 0 \tag{28.6c}$$

Because we deal with two distinct eigenvalues we know that $(\lambda_1 - \lambda_2) \neq 0$ therefore the scalar product $U_1 \cdot U_2 = 0$, indicating that two eigenvectors of a symmetric matrix must be orthogonal (perpendicular):

$$U_1 \perp U_2. \tag{28.6d}$$

Thus the orthogonal eigenvectors of the covariance matrix can be used to create the matrix U for transforming the observed demeaned data. The above reasoning is based on distinct eigenvalues. Without further proof, it should be noted that in the case of a symmetric covariance matrix of experimental data, identical eigenvalues rarely occur in special cases, and even then they do not present a problem; Exercise 28.1 is an example.

Let us apply the results from the above paragraphs to our numerical example given in Eqs. (28.2)–(28.4). First we must find a 3×3 matrix of orthogonal eigenvectors vectors $U = [U_1 \;\; U_2 \;\; U_3]$ to transform the demeaned data, so that: $B = UV$. Matrix V contains the transformed vectors V_1–V_4. This means that for each demeaned observation B_n we want to identify an orthogonal change of variable V_n such that:

$$B_n = UV_n \rightarrow \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}_n = [\, U_1 \quad U_2 \quad U_3 \,] \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}_n \tag{28.7}$$

Recall that in the above U_1–U_3 are column vectors so that U is a 3×3 matrix $u_{i,j}$; that is: $b_1 = u_{1,1} \times v_1 + u_{1,2} \times v_2 + u_{1,3} \times v_3$ etc. Assuming again that U is invertible, we can write the relationship in Eq. (28.7) as $V_n = U^{-1} B_n$. Because U is an orthogonal matrix, its inverse is equal to its transpose (see a linear algebra text such as Lay (1997) if you need to review this), so we may write $U^{-1} B_n = U^T B_n$. The covariance matrix Σ for V is:

$$\begin{aligned} \Sigma &= \frac{1}{N-1} VV^T = \frac{1}{N-1} \left(U^T B\right)\left(U^T B\right)^T = \frac{1}{N-1} U^T B B^T U \\ &= U^T \underbrace{\frac{1}{N-1} BB^T}_{\text{C: Eq. (28.4)}} U = U^T C U \end{aligned} \tag{28.8a}$$

So the orthogonal matrix U can relate C to Σ:

$$\Sigma = U^T C U = U^{-1} C U. \tag{28.8b}$$

In the above we used again $U^{-1} = U^T$ to obtain a result for Σ that is the same as the second expression in Eq. (28.5c). Thus the covariance matrix for transformed observations V_n is the diagonal matrix Σ. **Because the off-diagonal elements (the covariance values) of Σ are zero, the transformed observations (v_1, v_2, and v_3) are uncorrelated**. The diagonal elements of Σ, eigenvalues λ_1–λ_3 are the variance values for the transformed observations v_1–v_3. Convention for PCA is that the eigenvalues and associated eigenvectors are sorted according to the eigenvalues (variance); the order of this sorting is from high to low variance.

Note: To summarize, the PCA application was developed using basic linear algebra. The following rules were especially important. The eigenvalues of a symmetric matrix C with real values are also real and its eigenvectors are orthogonal. The size of the eigenvectors can be scaled to unity, and these orthonormal vectors can be arranged in an orthogonal matrix U. The transpose of this matrix is equal to its inverse: $UU^T = I$.

28.3.2 A MATLAB® Example

In this section we show examples of PCA analysis using MATLAB®. Although there is a pca MATLAB® command in the Statistics and Machine Learning Toolbox, we will use the basic commands from linear algebra to illustrate the steps that are involved in the analysis. As we demonstrated in the example in the previous section, if we compute the eigenvalues and eigenvectors for covariance matrix C, we can transform our demeaned observations depicted in Fig. 28.3B. Computing eigenvalues and eigenvectors is straightforward but it can be a bit cumbersome. In MATLAB® this computation can be easily accomplished with the eig command, for example: [UU,SIGMA] = eig(C). In our example we obtain three eigenvectors which form a rotated set of axes relative to the translated axes in Fig. 28.3B because the eigenvectors are orthogonal, i.e., mutually perpendicular (Eq. 28.6d). If we arrange the eigenvectors according to the magnitude of their associated eigenvalues (variance) we get the first, second, and third principal components. In Fig. 28.3C the first component is indicated in red and the remaining two components in

black; in this example it is easy to see that the first component is in the direction of maximal variance. In our example the covariance matrix C and its eigenvectors and eigenvalues (sorted for the eigenvalues in **descending** order) are:

$$C = \begin{bmatrix} 1.3333 & 0 & -1.3333 \\ 0 & 1.3333 & -0.6667 \\ -1.3333 & -0.6667 & 4.6667 \end{bmatrix}$$

(See also Eq. (28.4)),

$$U = \begin{bmatrix} -0.3192 & 0.4472 & 0.8355 \\ -0.1596 & -0.8944 & 0.4178 \\ 0.9342 & 0 & 0.3568 \end{bmatrix}, \text{ and}$$

$$\Sigma = \begin{bmatrix} 5.2361 & 0 & 0 \\ 0 & 1.3333 & 0 \\ 0 & 0 & 0.7639 \end{bmatrix}$$

Note that, if you do this example in MATLAB®, the eig command sorts the eigenvalues from low to high, that is, in **ascending** order, which is contrary to convention for PCA. Therefore the order of the diagonal entries in SIGMA and Σ and the order of the associated eigenvectors (columns) in MATLAB® variable UU and U listed above are reversed.

Suppose we want to find the coordinates of our observations S_1–S_4 on the translated-and-rotated set of axes (Fig. 28.3C), i.e., the projections of the observations on the eigenvectors. Let us look into our numerical example for how this can be accomplished by computing the projection of the first observation on the first principal component. First, we take point B_1 (corresponding to a demeaned version of the first observation S_1 in Eq. (28.2a)), that is the first column of B in Eq. (28.3)

$$B_1 = \begin{bmatrix} 1 \\ -1 \\ -2 \end{bmatrix}.$$

The first eigenvector is the first column of matrix U

$$U_1 = \begin{bmatrix} -0.3192 \\ -0.1596 \\ 0.9342 \end{bmatrix}$$

The projection of the first point (black in Fig. 28.3) on this eigenvector can be determined by the scalar product of the two vectors:

$$B_1 \cdot U_1 = B_1^T U_1 = [1 \quad -1 \quad -2] \begin{bmatrix} -0.3192 \\ -0.1596 \\ 0.9342 \end{bmatrix} = -2.0279$$

The above can easily be checked in MATLAB® after running the example program pr28_1.m. Use B(:,1) and U(:,1) for B_1 and U_1, respectively; the scalar product can be computed with B(:,1)'*U(:,1) (note the ' for transposing B(:,1)). The outcome -2.0279 is the projection of the first point on the first eigenvector. The projection of the first point on the second and third eigenvectors will be scalar products $B_1 \cdot U_2$ and $B_1 \cdot U_3$ (note the dots). For the second point B_2 (red in Fig. 28.3) we can repeat the procedure: $B_2 \cdot U_1$, $B_2 \cdot U_2$, and $B_2 \cdot U_3$. The same, of course, for the third (blue, Fig. 28.3) and fourth (green, Fig. 28.3) points. We can compute all the scalar products V at once with the matrix multiplication $B^T U$. This will generate the coordinates of all four points on the three eigenvectors. The results for our numerical example are summarized in Table 28.1.

Note that in some texts the projection on the first eigenvector (row v_1 in Table 28.1) is indicated as the first principal component, the projection v_2 on the second eigenvector is then the second principal component, etc. To summarize, depending on the text, the principal components can be the eigenvectors U_1–U_3 or the projections of the observations on these vectors v_1–v_3, and in some texts the term principal component is used for both.

The variances in each direction, that is for each component v_1, v_2, and v_3 are easily calculated in MATLAB® after running the program pr28_1.m with the std command: std(V').^2. The outcome of this calculation is 5.2361, 1.3333, 0.7639, as expected, these values correspond to the eigenvalues in Σ. Because the origin of the axes in Fig. 28.3C is the same as in panel B, the mean of the components v_1–v_3 remains zero (mean(V')).

TABLE 28.1 Principal Component Analysis (PCA): Numerical Example

$S = [$	S_1	S_2	S_3	$S_4]$	**Original Observations**
s_1	4.0000	2.0000	2.0000	4.0000	Fig. 28.3A
s_2	1.0000	3.0000	1.0000	3.0000	
s_3	2.0000	3.0000	7.0000	4.0000	
$B = [$	B_1	B_2	B_3	$B_4]$	**Demeaned Observations**
b_1	1.0000	−1.0000	−1.0000	1.0000	Fig. 28.3B
b_2	−1.0000	1.0000	−1.0000	1.0000	
b_3	−2.0000	−1.0000	3.0000	0	
$V = [$	V_1	V_2	V_3	$V_4]$	**Projections on**
v_1	**−2.0279**	−0.7746	**3.2812**	−0.4787	Eigenvectors
v_2	1.3416	−1.3416	0.4472	−0.4472	Fig. 28.3C
v_3	−0.2959	−0.7746	−0.1829	1.2533	

Summary of PCA on four observations S_1−S_4. These data points are plotted in Fig. 28.3A. First the data are demeaned in B_1−B_4 so that a new set of axes with its origin in the point of gravity of all points is obtained (Fig, 28.3 B). Finally, the axes are rotated using the PCA (Fig. 28.3C). Note that the first component axis (red in Fig. 28.3C) indicates the direction of largest variance, easily appreciated when looking at the position of the first (V_1, black) and third (V_3, blue) point in Fig. 28.3C. For clarity, these extreme values for the first component v_1 are indicated in bold in the table (v_1 in vectors V_1 and V_3).

Further, we can test for zero covariance, that is the off diagonal entries of the covariance matrix (1/3)*V*V' must be zero. The result is:

5.2361	0.0000	−0.0000
0.0000	1.3333	0.0000
−0.0000	0.0000	0.7639

The outcome is as expected, the diagonal elements are again the variances for v_1−v_3 and all covariance values are zero.

As a final note you can see that the PCA would do a bad job distinguishing source signals from a mixture. Our S matrix was the same as the measured signals Y in the example of Fig. 28.2 in Section 28.2. **The temporal sequences in the decomposed results in V (Table 28.1) do not even come close to the source signals X in that example.**

28.3.3 Singular Value Decomposition

An alternative, often used, technique to compute the eigenvalues and eigenvectors of the covariance matrix is directly from the demeaned

observations using singular value decomposition. This technique is based on the fact that any rectangular matrix, such as the demeaned observation matrix B, can be decomposed as

$$B = U\Theta W^T \tag{28.9}$$

Note that this expression looks similar to the first expression in Eq. (28.5c). In Eq. (28.9) U and W are orthogonal matrices, and Θ is a matrix that includes a matrix Σ for which the diagonal entries are the so-called singular values $\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_r$. In our numerical example above with B being a 3×4 matrix (Eq. (28.3)), U is a 3×3 matrix of eigenvectors, W a 4×4 matrix of eigenvectors, and Θ the same size as B, a 3×4 matrix in which the first 3×3 diagonal entries are the singular values σ_1–σ_3. In this example

$$B = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -2 & -1 & 3 & 0 \end{bmatrix} \quad U = \begin{bmatrix} -0.3192 & 0.4472 & 0.8355 \\ -0.1596 & -0.8944 & 0.4178 \\ 0.9342 & -0.0000 & 0.3568 \end{bmatrix}$$

$$\Theta = \begin{bmatrix} 3.9634 & 0 & 0 & 0 \\ 0 & 2.0000 & 0 & 0 \\ 0 & 0 & 1.5139 & 0 \end{bmatrix}$$

$$W = \begin{bmatrix} -0.5117 & 0.6708 & -0.1954 & 0.5000 \\ -0.1954 & -0.6708 & -0.5117 & 0.5000 \\ 0.8279 & 0.2236 & -0.1208 & 0.5000 \\ -0.1208 & -0.2236 & 0.8279 & 0.5000 \end{bmatrix}$$

While the eigenvectors (columns of U) we find here correspond with those found for the covariance matrix above, you may be surprised that the singular values in Θ do not correspond with those in Σ above. This is because, unlike the eigenvalues of the covariance matrix, the singular values σ_i are the standard deviations and not the variance. Furthermore, the singular values are based on BB^T while the eigenvalues λ_i are based on the normalized version: BB^T divided by $1/(N-1)$. So if we compute $\Theta\Theta^T$ and divide by $N-1 = 3$ we get the same values as the diagonal entries in Σ:

$$\Sigma = \frac{\Theta\Theta^T}{N-1} = \frac{\Theta\Theta^T}{3} = \begin{bmatrix} 5.2361 & 0 & 0 \\ 0 & 1.3333 & 0 \\ 0 & 0 & 0.7639 \end{bmatrix}$$

This result is identical to the values we obtained for covariance matrix Σ we obtained earlier. If we use Eq. (28.9) to compute BB^T:

$$BB^T = (U\Theta W^T)(U\Theta W^T)^T = (U\Theta W^T)(W\Theta^T U^T) = U\Theta \underbrace{W^T W}_{I}\Theta^T U^T$$
$$= \underbrace{U\Theta\Theta^T U^T}_{\Sigma''} = U\Sigma'' U^T \tag{28.10}$$

In the above we used W^{TT}=W. Since W is orthogonal W^T=W^{-1} we may state $W^T W$=I with I—the identity matrix. Finally, because Θ has only nonzero diagonal entries, $\Theta\Theta^T = (N-1)\Sigma = \Sigma''$. Recalling that BB^T divided by $1/(N-1)$ is the covariance C, the outcome of Eq. (28.10) is, with the exception of the normalization $1/(N-1)$ (reflected by the use of Σ'' instead of Σ), the same as the left expression in Eq. (28.5c) which is restated here for convenience: $C = U\Sigma U^{-1}$ (recall that $U^T = U^{-1}$ because U is an orthogonal matrix).

We can use standard MATLAB® functions to compute the eigenvalues and eigenvectors from the covariance matrix using the* eig *command, or directly from the demeaned observations using singular value decomposition with the* svd *command. A part of* pr28_1.m *shows the use of these commands.

```
% Two Methods to Perform PCA using MATLAB® standard functions
   eig and svd
% 1. Eigenvalues and Eigenvectors (eig) of Covariance Matrix C
% [=(1/(N-1)*B*B']
[ei_vectors1,ei_values1]=eig(C)
[' NOTE that the eigenvalues above are sorted in ASCENDING order']

% 2. Singular Value Decomposition (svd) of DEMEANED Observa-
   tion Matrix B
[ei_vectors2,singular_values,vv]=svd(B)
[' NOTE that the eigenvalues above are sorted in DESCENDING
order']
   % IMPORTANT NOTE
   % singular_values is the sqrt of the eigenvalues of the non-
     normalized
   % covariance B*B' [i.e., sqrt(eig(B*B'))]
```

28.3.4 Applications of Principal Component Analysis

In neuroscience, PCA is used in a variety of applications that involve decomposition of compound signals in extracellular or optical measurements. For instance, in many software packages that are designed for sorting spikes in microelectrode measurements, PCA is used as the basis of the sorting operation (e.g., Prentice et al., 2011). However, PCA is not the only method for spike sorting and multiple alternative procedures analyzing raw spike data or features of the spike waves, using both supervised and unsupervised approaches, exist (e.g., Rossant et al., 2016). In the following example, we illustrate spike sorting by employing raw (simulated) waveform data as the basis for the PCA. The MATLAB® script pr28_2 is a simulation of the application of PCA to the sorting of spikes in extracellularly recorded multiunit activity. Here we illustrate the PCA approach by simulating two types of spike embedded in a noisy recording with a DC component. The superimposed demeaned spikes are depicted in Fig. 28.4A. We employ the MATLAB® svd command to determine the principal components and show that the first two principal components do an excellent job of separating the two types of spikes (Fig. 28.4B). This allows a user to distinguish different units that contribute to multiunit spike trains. However, it should be noted that spike sorting procedures applied to real data usually do not perform that well! There are multiple reasons for the reduced performance of spike sorting. In real measurements there is an unknown number of units that contribute to the multiunit activity, there is considerable noise, and usually there are more than two types of spike. More importantly, the "dogma" that each neuron consistently produces the same spike waveform when firing is not always true. For example, the amplitude and shape of the spikes from a single neuron may vary, especially at higher activity levels (Fig. 28.4C). Due to all these phenomena, separation of the clusters of spikes fired by different neurons in a multiunit record may not be well separated, rendering the spike sorting results rather arbitrary.

Another example of using PCA, in this case the separation of signal and noise components, is given in MATLAB® script pr28_3. Here we simulate a noisy imaging result by adding random noise to a picture of Lena. Next, we show that up to about the 15th principal components include much of the signal while the subsequent principal components increasingly represent the small details and noise of the image (we show up to component 30 in this example script).

28.4 INDEPENDENT COMPONENT ANALYSIS

In the previous section we introduced PCA, a technique to decompose multichannel data into uncorrelated components. If we use PCA to

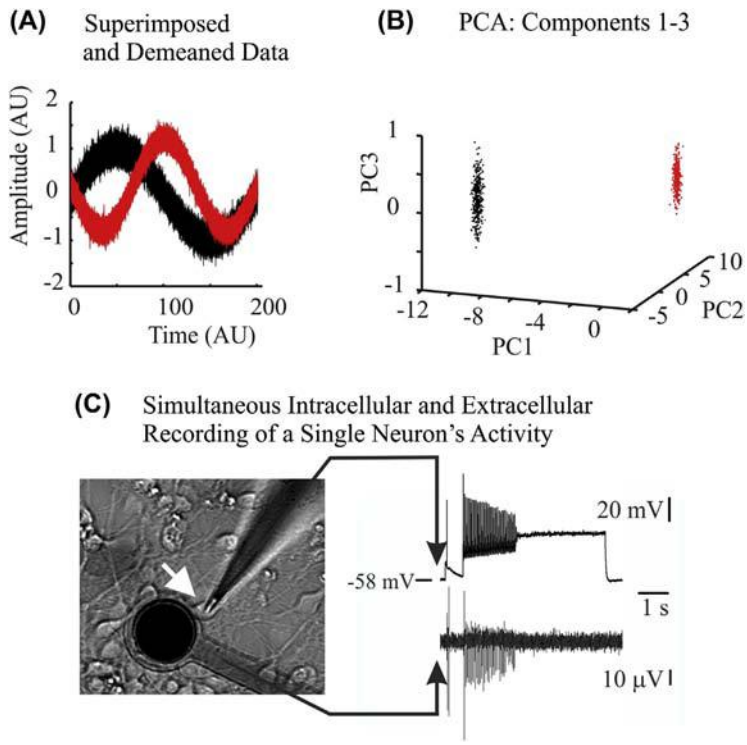
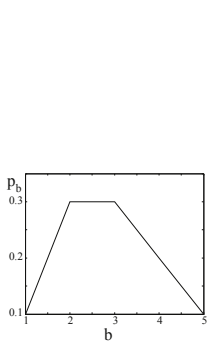
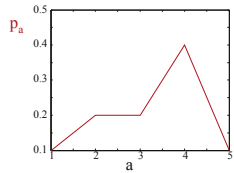


FIGURE 28.4 Application of principal component analysis (PCA) to spike sorting. (A) Superimposed demeaned noisy measurements of two types of synthesized spike waveforms. Both amplitude and time are in arbitrary units (AU). (B) The projection of the simulated spike measurements on the first three principal components (PC1, PC2 and PC3). It can be seen that the two types of waveform (*red and black dots*) are separated by the first two components while the third, vertical component (PC3) doesn't help in distinguishing the two types. Panels A and B are produced by MATLAB® script pr28_2. (C) An example of simultaneous intracellular (top trace) and extracellular (bottom trace) recordings of a cultured hippocampal pyramidal cell (*white arrow*) demonstrating how spike sorting may fail at higher levels of neuronal depolarization. The intracellular measurement shows the action potential activity around a depolarizing current injection. The variation in the amplitude in the burst of action potentials is reflected in the extracellularly recorded spike train. It is clear that any spike sorting that depends on consistent spike waveforms would fail in this case. *Experimental data in panel C from Dr. A.K. Tryba, with permission.*

decompose x and y into variables a and b, we showed that the covariance between decomposed variables a and b is zero and the covariance matrix is uniquely determined by their variance. **Independent Component Analysis (ICA) moves beyond the constraint of decorrelation and looks for components that are statistically independent.** When two signals a and b are statistically independent, they are each drawn from an





$p_b \backslash p_a$:	0.10	0.20	0.40	0.10
0.10	0.01	0.03	0.02	0.01
0.30	0.02	0.06	0.04	0.02
0.30	0.02	0.06	0.04	0.02
0.20	0.04	0.12	0.08	0.04
0.10	0.01	0.03	0.02	0.01

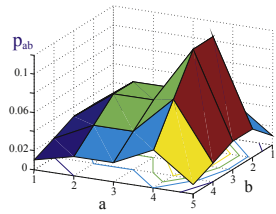


FIGURE 28.5 An example of a two-dimensional joint probability density function p_{ab} (in the green panel) and its marginal distributions p_a and p_b. The graphs show the individual, marginal distributions of a (top graph) and b (left graph). In the 3-D graph on the right, the joint probability is plotted on the vertical axis against the variables a and b.

independent probability density function (PDF), and the joint PDF of $[a\ b]$ is simply the product of the individual PDFs:

$$p_{ab}\big([ab]\big) = p_a(a)p_b(b) \tag{28.11}$$

The joint and individual PDFs are symbolized by p_{ab}, p_a, and p_b, respectively; an example is shown in Fig. 28.5. Suppose we have two processes a and b, with probabilities $p_a = [0.1\ 0.2\ 0.2\ 0.4\ 0.1]$ and $p_b = [0.1\ 0.3\ 0.3\ 0.2\ 0.1]$, then if they are independent, we may use Eq. (28.11) to compute joint probability p_{ab} (Fig. 28.5). Note that the probability functions in Fig. 28.5 all add up to 1 $\left(\sum_{i=1}^{5} p_{a_i} = 1,\ \ \sum_{i=1}^{5} p_{b_i} = 1,\ \text{ and }\ \sum_{i=1}^{5}\sum_{j=1}^{5} p_{a_i b_j} = 1\right)$.

Statistical independence between variables a and b means that all the moments and central moments of the distributions for a and b must also be independent:

$$E\{[a^p b^q]\} = E\{a^p\}E\{b^q\}, \tag{28.12a}$$

here $E\{\ldots\}$ denotes the expectation (see Chapter 3, Section 3.2, if you need to refresh your knowledge about expectation). If ***a* and *b* are demeaned**, the first central moment (i.e., exponents $p = 1$ and $q = 1$ in Eq. 28.12a) of the joint PDF $E\{a\ \ b\}$ is also known as covariance. If a and b are uncorrelated (as in the decomposed result from PCA), we have:

$$E\{[ab]\} = \underbrace{E\{a\}}_{0}\ \underbrace{E\{b\}}_{0} = 0 \tag{28.12b}$$

You can see that demanding that a and b are uncorrelated (Eq. 28.12b) is not as strong a condition as asking for statistical independence of a and b (Eq. 28.12a). There is an exception where PCA does generate statistically independent components: that is when the extracted signals are normally distributed. Normally distributed signals are determined by their first two moments; once these are known all higher-order moments are determined. **To summarize, two signals that are statistically independent are also uncorrelated; uncorrelated signals are not statistically independent except when the signals are normally distributed.**

In real cases, signal mixtures will have a tendency to be normally distributed due to the central limit theorem. To put it informally, the central limit theorem states that the sum or a mixture (= weighted sum) of multiple variables tends to be normally distributed also when individual components are not drawn from a normal distribution. An example of this theorem at work is shown in Fig. 28.6. In this example we study a mixture of variables that are each uniformly distributed (Fig. 28.6A). Interestingly, the mixture of only five such variables already shows a tendency towards a normal distribution (Fig. 28.6B). Nonetheless, it should be noted that the performance of PCA for the extraction of source contributions to a signal may be limited in the case that the data are not normally distributed. Because ICA is based on the statistical independence of the individual components, it is a much better technique capable of extracting sources that are not normally distributed. Looking at the distribution of observations from a uniform distribution (Fig. 28.6A), we observe that the points are scattered more or less evenly over a line in the one-dimensional case (Fig. 28.6C) or a plane in the two-dimensional case (Fig. 28.6E). In contrast, normally distributed (Gaussian) mixtures are concentrated around the mean value of the distribution (Fig. 28.6B,D,F).

28.4.1 Entropy of Sources and Mixtures

Recall that the entropy $S(X)$ of a random variable X (see Chapter 20, Section 20.3) depends on its probability distribution. It can be defined as the sum (in the case of a discrete variable) or the integral (in the case of a continuous variable) of the product $p(x)\log_2 \frac{1}{p(x)} = -p(x)\log_2 p(x)$ over all x:

$$S(X) = -\sum_{\text{All } x} p(x)\log_2 p(x) \tag{28.13}$$

In this case we defined S for a discrete variable and we use $\log_2$, a base 2 logarithm so that S is in bits.

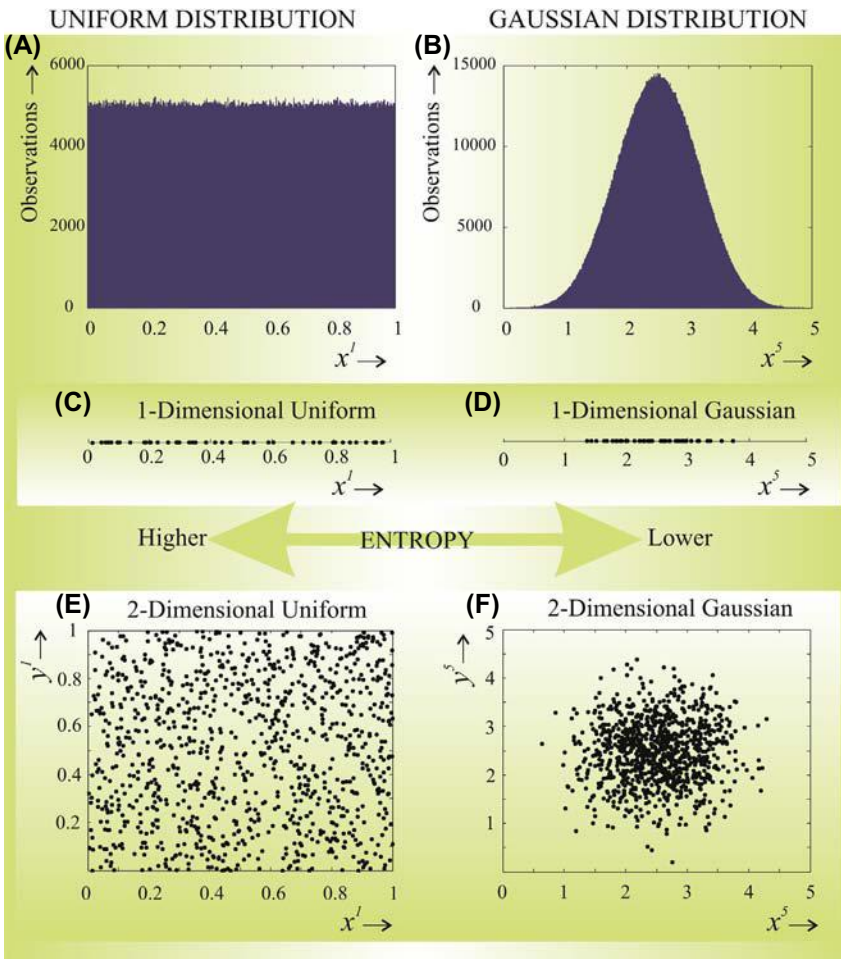


FIGURE 28.6 (A) Histogram of a variable x^1 that is uniformly distributed between 0 and 1. (B) The sum of only five of these uniformly distributed variables x^5 tends to be almost normally distributed. Panels (A) and (B) of this figure were made with script pr28_4.m. A series of one-dimensional observations from uniform (x^1) and (almost) Gaussian (x^5) distributions are shown in (C) and (D), respectively. The scatterplots in (E) and (F) are examples of a series of two-dimensional observations: two variables x^1 y^1 for the uniform case, and two variables x^5 y^5 for the Gaussian one. As expected, the uniform distribution results in a scatter of point throughout the plane, whereas the Gaussian case shows a concentration of plots around a center (the mean). Consequently, the entropy of the uniformly distributed points is higher than the entropy for the Gaussian distributed observations.

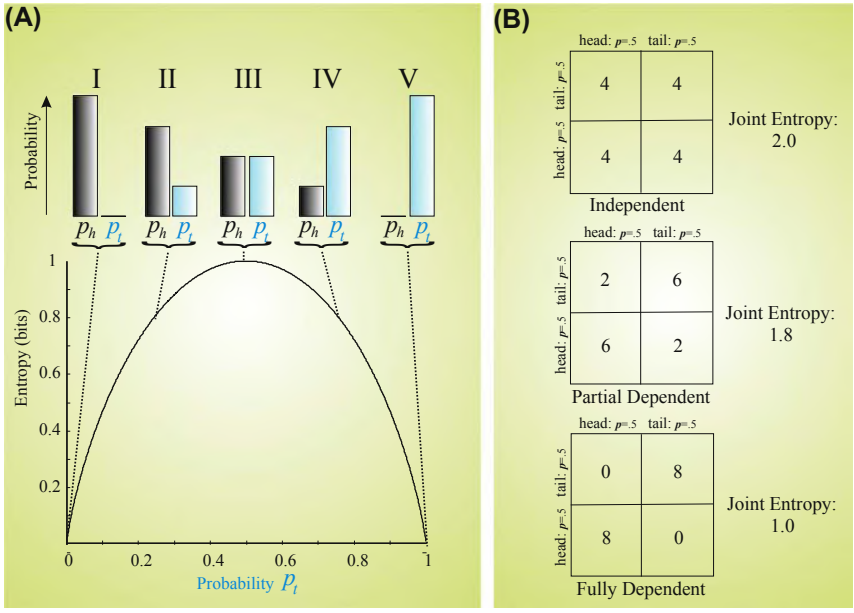


FIGURE 28.7 Statistics of a coin toss and entropy. (A) Five scenarios of probability distributions of heads (p_h) and tails (p_t). The graph depicts that each scenario is associated with a specific entropy value. (B) Statistics of coin tosses. Two coins are used for each observation and in the upper diagram the outcomes of each toss are completely independent. In the two lower diagrams there (magically) is some dependence between the two tosses in each observation. Either there is a full dependence (bottom diagram; the pair of outcomes in each observation are identical) or a partial dependence (middle diagram).

Let us consider a very simple case, a coin toss. If we have the usual situation, we have $p = ½$ for both heads and tail (scenario III, Fig. 28.7A) and the entropy according to Eq. (28.13) is

$$S(X) = -\left[\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right] = 1 \text{ bit}.$$

This is a reasonable result, because we have an outcome that fits in a single bit: either heads (1) or tails (0). Now suppose we have "faulty" (deterministic) coins that always land on one side, either heads or tail. In these scenarios (I and V, Fig. 28.7A) we have $p = 0$ for one outcome and $p = 1$ for the other; now the entropy is

$$S(X) = -[0\log_2 0 + 1\log_2 1] = 0 \text{ bit}.$$

Note that we define $\log_2 0 = 0$. Also this outcome seems reasonable since there is no surprise (information) with each outcome: it will always be heads in one scenario and always tails in the other. If our coin is biased and we get heads or tails in 75% of the cases (scenarios II and IV in Fig. 28.7A), we have probabilities $p = .75$ and $p = .25$ and the associated entropy is

$$S(X) = -\left[\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right] = 0.81 \text{ bit.}$$

Thus for every probability distribution we find a specific entropy (graph in Fig. 28.7A). The maximum entropy we find is when probability P is equal ($p = ½$) for heads and tails, that is when the probability distribution is uniform (Scenario III, Fig. 28.7A). Without further proof we state that the above result may be generalized to any probability distribution: **variables show maximum entropy when they are uniformly distributed** (Fig. 28.6).

In panels E and F in Fig. 28.6, we consider a two-dimensional distribution where each observation is represented by a dot in a plane. In this 2-D example we have a joint distribution just as the one for variables a and b shown in Fig. 28.5. To compute the entropy associated with such a joint probability distribution we follow the same approach as for the one-dimensional case: we summate $-p(x)\log_2 p(x)$ over the domain of x:

$$S(a,b) = -\sum_{i=1}^{5}\sum_{j=1}^{5} p_{a_i b_j}\log_2 p_{a_i b_j}$$

The entropy of the joint distribution (the table [green panel] in Fig. 28.5) is

$$-[0.01\log_2 0.01 + 0.03\log_2 0.03 + 0.03\log_2 0.03 + \ldots$$
$$\ldots + 0.03\log_2 0.03 + 0.02\log_2 0.02 + 0.01\log_2 0.01] = 4.29.$$

The entropy for the individual variables a and b can be obtained from the marginal distributions, for a (see marginal distribution [red] in Fig. 28.5) we find $S(a)$

$$-[0.1\log_2 0.1 + 0.2\log_2 0.2 + 0.2\log_2 0.2 + 0.4\log_2 0.4 + 0.1\log_2 0.1] = 2.12.$$

and for b (marginal distribution [black] in Fig. 28.5) we find $S(b)$

$$-[0.1\log_2 0.1 + 0.3\log_2 0.3 + 0.3\log_2 0.3 + 0.2\log_2 0.2 + 0.1\log_2 0.1] = 2.17.$$

Now we see that $S(a) + S(b) = 2.12 + 2.17 = 4.29$, which is equal to $S(a,b)$. This isn't so surprising because the probability distributions for a and b were independent such that $p_{ab}([ab]) = p_a(a)p_b(b)$. If our distribution in Fig. 28.5 had been uniform, we would have found different values for the entropies. In this case the five probabilities in p_a and p_b would be [0.2 0.2 0.2 0.2 0.2] and the joint distribution would also be uniform with all 25 probabilities equal to 0.04. The associated entropies would now be

$$S(a,b) = -25 \times 0.04 \times \log_2(0.04) = 4.64$$

$$S(a) \text{ and } S(b) \text{ are both } -5 \times 0.2 \times \log_2(0.2) = 2.32$$

In all cases the entropies are higher (because of the uniform distribution), but due to the independence of a and b, the relationship

$$S(a) + S(b) = S(a,b) \tag{28.14a}$$

still holds.

If there was a dependence between the two distributions of a and b we would have found a different result. Let us explore the effect of independence with an even simpler example and get back to our coin toss. Let us assume we toss two coins and in one case we have the usual situation where the tosses are independent (Fig. 28.7B upper diagram). However, in the other case there is a "magical" full dependence between the two coins, if one coin hits heads or tails the other coin does too (Fig. 28.7B lower diagram). With the two coins we have four alternative outcomes: head−head, head−tail, tail−head, tail−tail. Assuming we have equal probability for heads and tails, we get in the independent case that the probability for each outcome is ½ × ½ = ¼. In the dependent case however, the probabilities for head−tail and tail−head are zero because one coin will magically copy the outcome of the other (just for the sake of this example, don't worry how you would actually do such a thing). The probabilities for head−head and tail−tail are each ½. First we compute entropies S_1 and S_2 for each individual coin toss from the marginal distributions and we find

$$-2 \times 0.5 \times \log_2(0.5) = 1.$$

In the independent case (top diagram in Fig. 28.7B), we find for the joint entropy $S_{1,2}$

$$-4 \times 0.25 \times \log_2(0.25) = 2.$$

Here we see that just as in the case for Fig. 28.5, the sum of the individual entropies equals the joint entropy:

$$S_1 + S_2 = S_{1,2}.$$

Now we compute the joint entropies for the two other scenarios in Fig. 28.7B. In the fully dependent case (the bottom diagram in Fig. 28.7B), the joint entropy $S_{1,2}$ is

$$-[2 \times 0.5 \times \log_2(0.5) + 2 \times 0 \times \log_2(0)] = 1.$$

The case with some dependence between the two coins is shown in the middle diagram in Fig. 28.7B. Note that, due to the partial dependence, in most but not all cases the outcomes of the first and second coin toss are identical. This results in a joint entropy of

$$-[2 \times 0.125 \times \log_2(0.125) + 2 \times 0.375 \times \log_2(0.375)] = 1.8.$$

In both cases where there is (full or partial) dependence between the tosses of the coins we find that $S_1 + S_2 > S_{1,2}$, and the more dependence exists between the tosses, the larger the difference between $S_1 + S_2$ and $S_{1,2}$. Apparently we need to adapt Eq. (28.14a) when there is dependence between the two variables by including a term that reflects this dependence. This term is commonly indicated by mutual information (*MI*), which is an indication of the level of dependence between variables or, in other words, it quantifies the amount of information that variable 1 provides for variable 2. In case of the dependence between coin tosses, the outcome of one coin toss determines the outcome of the other; in the normal, fair tosses the outcome of one toss doesn't provide any information about the other since they are independent. Our findings are summarized in Table 28.2.

It can be seen in this example that the joint entropy and *MI* variables are indeed proportional with the level of dependence. Without further proof, we assume that we may generalize our findings and state that for any two random processes X and Y, we can compute the entropy for each of the individual processes $S(X)$ and $S(Y)$. The joint entropy $S(X,Y)$ is the sum of

TABLE 28.2 The Effect of Dependence on Joint Entropy and Mutual Information

	Independent (bit)	**Slightly Dependent (bit)**	**Fully Dependent (bit)**
Entropy Coin 1, S_1	1.0	1.0	1.0
Entropy Coin 2, S_2	1.0	1.0	1.0
Sum $S_1 + S_2$	2.0	2.0	2.0
Joint Entropy, $S_{1,2}$	2.0	1.8	1.0
Difference $(S_1 + S_2) - S_{1,2}$ (= Mutual Information)	0.0	0.2	1.0

the individual entropy values when X and Y are independent (Eq. 28.14a), otherwise we have

$$S(X,Y) = S(X) + S(Y) - MI(X,Y) \tag{28.14b}$$

in which $MI(X,Y)$ is the mutual information between X and Y. We could define joint entropy $S(X,Y)$ as the total information of the joint process X,Y. To summarize our findings in Table 28.2, it can be concluded from the above that for any given pair of processes X and Y, independence occurs at maximal joint entropy (or joint information) with minimal mutual information of the joint process. This is a basis for the ICA technique: independence of separated sources is evaluated by joint entropy (joint information) and mutual information. For the separation of independent sources, their joint information $S(X,Y)$ must be maximized, and therefore this ICA technique is also called **infomax**.

28.4.2 Using the Scalar Product to Find Independent Components

After we obtain the criteria for unmixing a mixture of signals (e.g., decorrelation, statistical independence, maximizing joint entropy), the procedure for separating components from mixtures in ICA and PCA is essentially the same as was outlined in Section 28.2: source signals are found from the product of the unmixing matrix and the recorded signals (Fig. 28.2). The unmixing matrix contains the vectors along which the components are extracted. The difference between ICA and PCA is the strategy for finding the directions of the vectors in the unmixing matrix. In PCA we found directions of maximal variance (Fig. 28.3C) while the components were decorrelated. For ICA we demand statistical independence.

To illustrate an ICA-type extraction procedure, let us consider a two-dimensional case: two sources x_1 and x_2 with a known mixing matrix A, creating two mixtures s_1 and s_2:

$$\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

A scatterplot representation of the sources and the mixtures is shown in Fig. 28.8; the sources are plotted in panel A and the resulting mixtures in panel B. In this example, we know the mixing matrix A:

$$A = \begin{bmatrix} 0.2 & 0.8 \\ 0.7 & 0.4 \end{bmatrix}$$

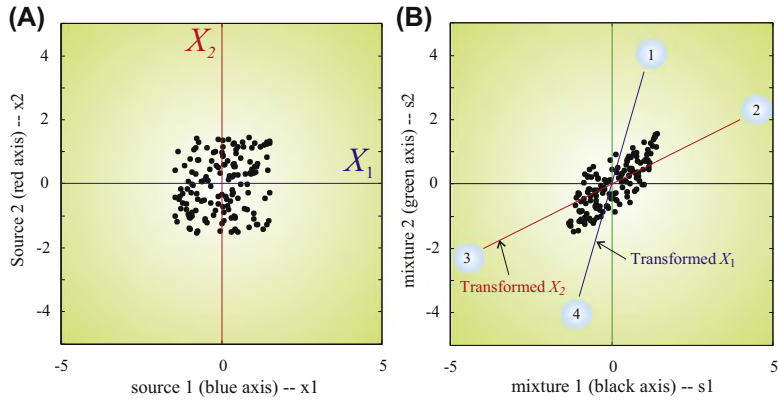


FIGURE 28.8 (A) Scatterplot of two source signals x_1 and x_2. (B) Scatterplot of two mixtures s_1 and s_2 that were created from the source signals. The transformed source axes (red and blue) are indicated in this mixture plot. Each transformed source axis is indicated at each end by numbers: X_1 by1−4 and X_2 by 2−3.

Using this information, we can determine the orientation of the original axes X_1 and X_2 from the source scatterplot (depicted in Fig. 28.8A) in the mixture scatterplot (shown in Fig. 28.8B). The first axis $X_1 = [1\ 0]$, so the transformed version of source axis X_1 in the scatterplot of the mixtures is

$$\underbrace{A}_{\text{Mixing Matrix}} \underbrace{\begin{bmatrix}1\\0\end{bmatrix}}_{X_1} = \begin{bmatrix}0.2 & 0.8\\0.7 & 0.4\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0.2\times 1 + 0.8\times 0\\0.7\times 1 + 0.4\times 0\end{bmatrix} = \begin{bmatrix}0.2\\0.7\end{bmatrix}$$

which is the first column of mixing matrix A. Similarly, the transformed second source axis X_2 in the mixture plot is the second column of A. The axes X_1 and X_2 from the scatterplot in Fig. 28.8A are also depicted, after transformation with mixing matrix A, in Fig. 28.8B. After this transformation X_1 becomes the axis 1−4 (blue) and X_2 becomes axis 2−3 (red).

In a real problem, we have only the measurement of the mixture! Thus the axes 1−4 and 2−3 are the ones to be determined! In the text below, we describe a strategy to accomplish this. For this explanation it helps to look at the plot of the mixtures in Fig. 28.8B and the orientation of axes and vector in Fig. 28.9A. First we establish that we know there are two sources, and that we have two mixtures. So, if we knew the orientation of axes X_1 and X_2, we could find the contribution of x_1 to the mixtures by excluding all contributions of x_2. Because the contributions of x_2 are in the direction of axis X_2, we can use the scalar product of all observation vectors of the mixtures (all points in Fig. 28.8B) and a vector $X_2^{\perp}$ perpendicular to axis X_2 (Fig. 28.9A). All components $X_2^{=}$ parallel to X_2 will cancel on the $X_2^{\perp}$ axis

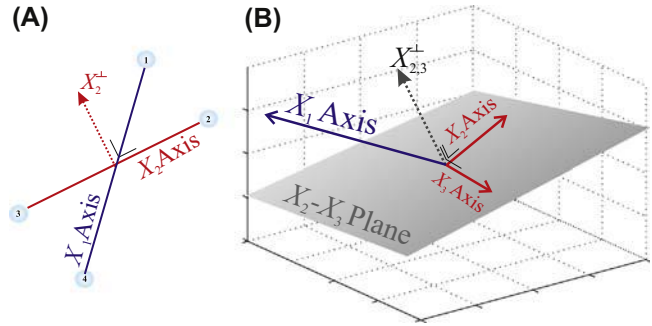


FIGURE 28.9 Panels (A) and (B) show the strategy for unmixing. In (A) we have a two-dimensional case: if we cancel all components in the direction of axis X_2 (by using the inner product of a vector $X_2^{\perp}$ perpendicular to X_2) the remainder must be a component of the X_1 axis. This approach can be extended to higher-dimensional cases (B): by cancelling components for X_2 and X_3 (by using the inner product of a vector $X_{2,3}^{\perp}$ perpendicular to X_2 and X_3), we keep the ones for X_1.

since the inner product $X_2^{\perp} \cdot X_2^{=} = 0$. Therefore the only component remaining in the scalar product of every observation $[s_1\ s_2]$ in the mixture plot with $X_2^{\perp}$ will be independent of x_2 and must be x_1.

Let us summarize the above approach in a few words. **All components independent of X_2, the axis for source x_2, can only be a component of x_1.** Further, we found that **we can use the inner product to remove x_2 components and only keep those independent from X_2.**

We can have a similar reasoning for mixtures from three or more sources. Let us consider a three-source three-mixture case (Fig. 28.9B). If we want to find the components for x_1, we need to remove the components for x_2 and x_3. So, if we construct a plane through the axes for x_2 and x_3, we can come up with $X_{2,3}^{\perp}$ (Fig. 28.9B). The inner product of observation $[s_1\ s_2\ s_3]$ with $X_{2,3}^{\perp}$ (perpendicular to the $X_2{-}X_3$ plane) removes all components associated with x_2 and x_3, and must therefore be the contribution of x_1. In a higher number of dimensions (that is with more sources and mixtures), we can always construct a hyperplane through an all-but-one selected axis (that is the axis of one selected source) and find a vector perpendicular to this hyperplane. This vector (analogous to $X_{2,3}^{\perp}$ in Fig. 28.9B) can then be used to remove the contributions from all directions embedded in the hyperplane (analogous to the $X_2{-}X_3$ plane in Fig. 28.9B) so that the remainder must be the contribution from the selected source.

28.4.3 A MATLAB® Example

In the previous sections we covered the background of the different aspects of the ICA procedure. Now we are ready to consider a brute force example of an ICA analysis (Fig. 28.10). We will do this by completing the following steps.

1. Mix two uniformly distributed sources x_1 and x_2 with a known mixing matrix A to create a pair of mixed signals s_1 and s_2 (e.g., Eq. (28.1a)). The sources can be plotted in source space along axes X_1 and X_2. Similarly, the mixtures can be plotted in the signal space along axes S_1 and S_2. Since we know A, we can also plot the transformed source axes AX_1 and AX_2 in the signal space.
2. Now pretend the sources and mixing matrix are unknown and apply ICA by using the following steps to recover the sources from the mixtures.
3. Test a series of pairs of candidate transformed source axes (i.e., candidates for AX_1 and AX_2) in the mixture/signal space. For example: use steps of 1 degree over a range from 0 to 360 degrees for each axis, creating 360^2 candidate pairs of axes.

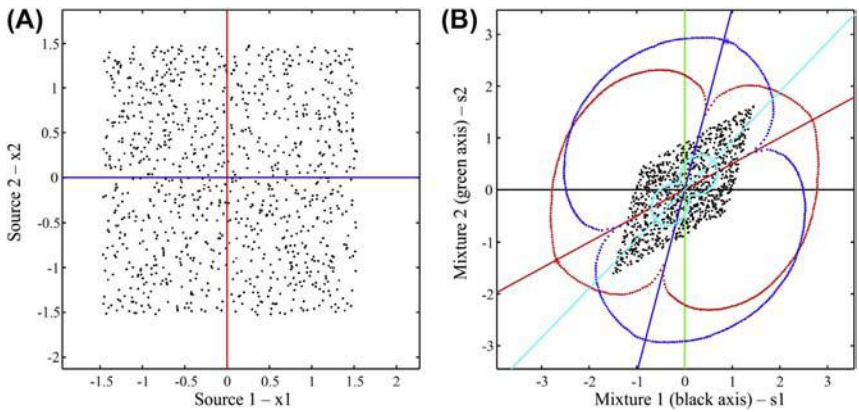


FIGURE 28.10 Example of the application of the independent component analysis (ICA) procedure. (A) Scatterplot of uniformly distributed source 1 (x_1—dark blue axis) and source 2 (x_2—red axis). (B) Scatterplot of the signals s_1 and s_2 made from mixtures of sources x_1 and x_2. In the mixture space, we repeatedly pick a pair of source axes and each time we compute the associated source values and their joint entropy. While following this procedure, by iteration, we determine the pair of axes with maximum joint entropy. The *dark blue* and *red dots* show the maximum joint entropy for that angle of source 1 (while the axis for maximizing source 2 is kept constant) and source 2 (while the axis for maximizing source 1 is kept constant), respectively. The dark blue and red axes in panel (B) are the best candidates for the transformed source axes. For comparison, the light-blue dots indicate the variance and the light-blue axis indicates the direction of the eigenvector associated with the largest eigenvalue (i.e., the first principal component). This figure was produced with pr28_5.

4. For each pair of candidate transformed source axes, establish the associated source components by determining the contribution perpendicular to each axis. For example, source-1 is all contributions perpendicular to the axis for source-2 (see Section 28.4.2).
5. Once the source components for each candidate pair of transformed source axes are determined, we compute their joint-entropy.
6. We collect all the joint-entropy values for the 360^2 cases and pick the one with the maximum value.
7. Finally, we evaluate the performance of our brute force approach by comparing the sources we used to produce the mixed signals in step 1 with the sources associated with the maximum joint-entropy we obtained in step 6.

This brute force, iterative procedure is followed in MATLAB® script pr28_5. The following is a snippet from this script showing the iteration loops. Each iteration loop goes through a range of angles: 0−2π rad. For each angle in the brute force search, the joint entropy (H) is determined using the entropy_2D function.

```
% ------------------------------------------------------------
% Now we pretend that we don't know the sources and mixing matrix
  and use
% a brute force approach to recover the sources from the mixtures
% Strategy:
% Rotate the candidate axes X1 and X2. Use these candidates to
  determine
% their associated sources and determine the joint entropy of the
  result
% ------------------------------------------------------------

H_max=0;                                % initialize maximum of the
                                          joint-entropy
phi_max1=0; phi_max2=0;                 % initialize the angles for
                                          axes X1 and X2
ct_phi1=0;                              % initialize counter 1
for phi1=0:2*pi/precision:2*pi;         % LOOP for rotating axis X1
   ct_phi1=ct_phi1+1;                   % update counter 1
   ct_phi2=0;                           % initialize counter 2
   for phi2=0:2*pi/precision:2*pi;      % LOOP for rotating axis X2
      ct_phi2=ct_phi2+1;                % update counter 2
      v1=[cos(phi1) sin(phi1)];         % unit vector parallel to X1
                                          with angle phi1
      v2=[cos(phi2) sin(phi2)];         % unit vector parallel to X2
                                          with angle phi2
      vv1=[cos(phi1+pi/2) sin(phi1+pi/2)];  % unit vector
                                              perpendicular to X1
```

```
    vv2=[cos(phi2+pi/2) sin(phi2+pi/2)];  % unit vector
                                             perpendicular to X2
    ic1=vv2*S;                            % unmix mixture S
    ic1=ic1-mean(ic1);sigma=std(ic1);     % demean & determine
                                             standard deviation
    ic2=vv1*S;                            % unmix mixture S
    ic2=ic2-mean(ic2);sigma=std(ic2);     % demean & determine
                                             standard deviation
    % Use 2D entropy estimate function entropy_2D to compute
       mutual
    % information (MI) and entropy (H) as a function of the
    % position of axes X1 (counter for phi1) and X2 (counter for phi2)
    [H(ct_phi1,ct_phi2), MI(ct_phi1,ct_phi2)]=entropy_2D(ic1,ic2);

    if H(ct_phi1,ct_phi2) > H_max;        % TEST: current H >
                                             current max of H?
        H_max=H(ct_phi1,ct_phi2);         % if so a new maximum
                                             for H is found
        imax=ct_phi1; jmax=ct_phi2;       % the indices for the
                                             new max are saved
        phi_max1=phi1;                    % and so are the other
                                             relevant data
        v_max1=v1;                        % the angles, the
                                             vectors &
                                             components

        ic1_max=ic1;
        phi_max2=phi2;
        v_max2=v2;
        ic2_max=ic2;
      end;
  end;
end;
```

28.4.4 What if Sources Are Not Uniformly Distributed?

For the ICA examples so far we have assumed that the sources were characterized by a uniform distribution (e.g., Fig. 28.6A,C,E) and we used entropy estimates to determine the level of independence (e.g., Fig. 28.7) of the separated candidate sources. So what should we do if we know that our sources are not uniformly distributed, such as a human voice or a chirp in a recording of a sound mixture? Such sources usually show a distribution with many values around zero (Fig. 28.11).

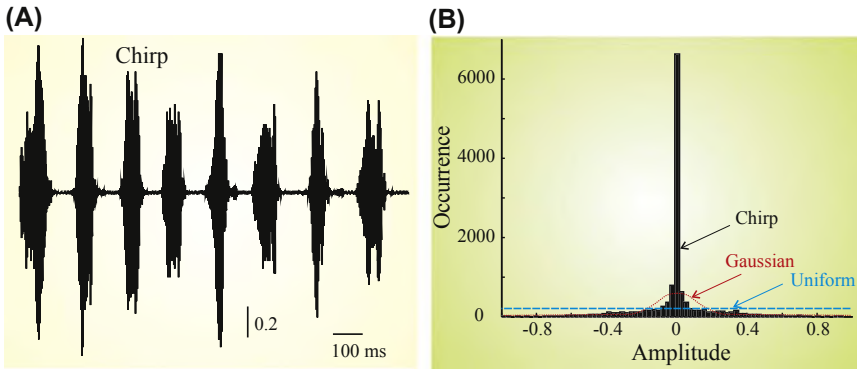


FIGURE 28.11 (A) Plot of a chirp (available in MATLAB® by load chirp). (B) Compared to a Gaussian (red) or uniform (blue) distributions, the histogram of the chirp signal shows a peaky distribution which is typical for many audio signals.

In such a case one could look for another function to maximize or minimize (instead of entropy or mutual information). In this example of a peaky distribution (Fig. 28.11B) one could maximize for peaky-ness of the distribution (kurtosis, a measure for how peaky a PDF is, might do the job in such a case). Alternatively, since we know that entropy is maximal if data are uniformly distributed, one could transform the nonuniform distribution into a uniform one and subsequently apply the same procedures that are available for the uniform distribution (Fig. 28.12). The example in Fig. 28.12 shows the transformation of a Gaussian distribution into a uniform one. The function used for this transformation is the cumulative probability density function (CDF). For a normally distributed variable x with zero mean, the CDF is $\frac{1}{2}\left[1+\mathrm{erf}\left(x/\sigma\sqrt{2}\right)\right]$, in which erf is the error function (available in MATLAB®) and σ is the standard deviation of x. If you think about this for a bit, it is plausible that for any PDF, the CDF is the optimum transformation to obtain a uniform distribution. The CDF will have the steepest slope where the probability is highest (and where you will therefore collect most observations) and a steeper slope will distribute the observations over a wider area (red dotted lines in Fig. 28.12A,B,C). In contrast, at low probabilities, where fewer observations occur, the slope of the CDF will be less steep and, consequently, the observations will be distributed over a smaller area. The overall effect of the transformation is thus to spread out observations more uniformly, exactly what we want for our purpose. After we transform our unmixing result into a uniform distribution we can apply exactly the same procedure we followed earlier.

An example of how to transform data using the CDF is demonstrated in MATLAB® script pr28_6. The implicit underlying thought of the procedure we follow in pr28_6 is that by transforming the unmixed sources

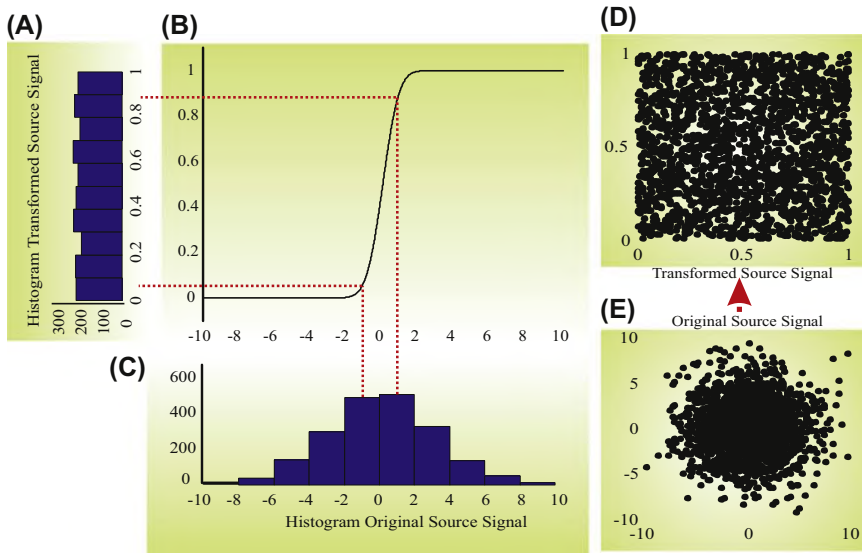


FIGURE 28.12 A uniform distribution (A) can be obtained from a nonuniform distribution by a transformation. This example shows a transformation of a histogram of observations drawn from a Gaussian distribution (C) using the cumulative probability density of the Gaussian distribution shown in (B). It can be seen that the majority of observations of the Gaussian distribution are located around zero in between the red lines. Following the *red lines* to panel (A), it can be seen that the transformation with the function in panel (B) distributes these points more evenly over a wider range. Accordingly, if such a transformation is applied to a two-dimensional scatterplot of a Gaussian variable (E), we get a scatterplot of uniformly distributed points (D). Part of this figure was produced with pr28_6.

into a uniform distribution we can apply the procedure to find the maximum joint entropy that we followed earlier while we do not affect the information content. This assumption may seem a bit of a stretch, but if we transform the data using an invertible function (such as our cumulative probability density function in Fig. 28.12B), we do not affect the mutual independence of the signals (see Stone, 2004). An example of the ICA procedure applied to a mixture of normally distributed data can be found in pr28_7.

> An invertible function is defined as a function that creates a unique new data point for each original data point and (because the function is invertible) this transformation can also be reversed. This means that if we have several independent data sets, they will remain independent after transformation with the invertible function into the other domain and vice versa.

28.4.5 Are There More Efficient Approaches Than the Brute Force Technique?

In the above iterative approach we looked into a two-dimensional case. We determined the source axes X_1 and X_2 with a precision of 1 degree, that is 360 computations for each axis. For the 2-D case this evaluates to $360^2 = 129{,}600$ iterations! That is, for each iteration, we compute candidate sources and we compute their mutual information. For more dimensions and/or higher precisions, the number of iterations grows rapidly, for example, if we wanted a ½ degree precision in a six-source case we have $720^6 \approx 1.4 \cdot 10^{17}$ iterations! As you can see, we need a more efficient procedure to find the best angles for the source axes, otherwise source extraction very rapidly becomes a computational nightmare. We used the property that the joint entropy of the sources is a function of their independence. Consequently we may assume that the joint entropy is a function of the orientation of the axes we use for the unmixing of the sources from their mixture. We can imagine this function as a landscape related to axes orientation, and since we are interested in the maximum joint-entropy, we should search for the peak in that landscape. If this landscape has a clearcut structure, we can use this to our benefit. Instead of iterating over all angles from 0 to 360 degrees, we now use the gradient in the landscape to locate a maximum. The procedure works as follows. First we pick a pair of random angles ϕ_1 and ϕ_2 for our pair of source axes X_1 and X_2 and compute joint-entropy H, then we pick another pair of angles at a small distance in the landscape, and compute H again. We determine for which of the two points H is largest and continue to evaluate a next point in that direction; we keep doing this until we cannot find a point with a larger value of H and conclude that we have reached a peak in the landscape. By doing this we use the slope in the landscape to climb towards a maximum, a procedure that is much faster than iteration and that scales much better when we increase the number of sources in each mixture and the number of measurements of mixtures. Examples of the application of this procedure for ICA are given in Stone (2004).

28.4.6 An Example of Independent Component Analysis Applied to Electroencephalogram Signals

The signals of brain electrical activity in Fig. 28.1 show recordings directly from the cortex (ECoG) and from the scalp (EEG). In the context of this chapter, it is fairly reasonable to assume that the signals generated at different locations separated by several millimeters in the brain will be statistically independent. Another way of saying the same thing is: the brain signals must carry a lot of information so sites that are relatively

remote must have low levels of mutual information. When recording directly from the cortex we can indeed observe this principle. When we use ICA to decompose an ECoG (Fig. 28.1A), our statistically independent components are almost identical to the recorded channels, indicating that the different sites on the cortex generate statistically independent signals. For the EEG (Fig. 28.1B) this phenomenon doesn't happen because the skull and scalp have a tendency to smear (mix) the contributions of the underlying sources and ICA may be a good tool to find individual sources in the signals.

The procedure that is commonly applied to EEG analysis is to find source signals that are temporally independent (because the EEG matrix has a temporal and spatial component, we could also look for components that are spatially independent). The underlying thought here is that source signals contribute to the signal at each EEG electrode. An example for two sources contributing to two EEG electrodes/channels is depicted in Fig. 28.13A. The EEG can therefore be considered as a linear mixture of the sources. Because the electrodes register the fields of the locally generated activity traveling at the speed of light, the delays for propagation between source and electrode are negligible. Our finding with the ECoG (that the ICA components resemble the original time series) shows that if sources are not too close, they can be considered independent. The simplified scenario in Fig. 28.13A shows how two electrodes Y_1 and Y_2 each record a different mixture from sources X_1 and X_2. Similar to the example in Fig. 28.2, we have that $Y = AX$ with A being the mixing matrix. The first column in mixing matrix A (a and b, in the example 0.8 and 0.4) indicate the coupling strength (proximity) of source X_1 to electrodes Y_1 and Y_2 (Fig. 28.13B). The second column in A (c and d, in the example 0.2 and 0.3) reflect the same coupling strength (proximity) of source X_2 to electrodes Y_1 and Y_2 (Fig. 28.13B). In this sense mixing matrix A contains spatial information because the values of its elements reflect the positions of sources and electrodes.

Now we can use ICA to estimate our source signals $\widehat{X}$ and the unmixing matrix $\widehat{A}^{-1}$ (Fig. 28.13C). Assuming that $\widehat{A}^{-1}$ is invertible, we can determine an estimate $\widehat{A}$ of the mixing matrix. Matrix $\widehat{A}$ contains the estimates for coupling strengths between each of the sources and the electrodes. These estimates $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{d}$ in Fig. 28.13C and D can now be used to depict the coupling between sources and electrodes. For Source 1 we find coupling strengths $\boldsymbol{a}$ and $\boldsymbol{b}$ (Fig. 28.13D1) for Electrode 1 and Electrode 2; for Source 2 we have strengths $\boldsymbol{c}$ and $\boldsymbol{d}$ (Fig. 28.13D2) for Electrode 1 and Electrode 2. As we will demonstrate in the following example, because usually the EEG recording includes multiple channels, it is common practice to show the coupling strength for each component (source) at each electrode in a color-coded fashion.

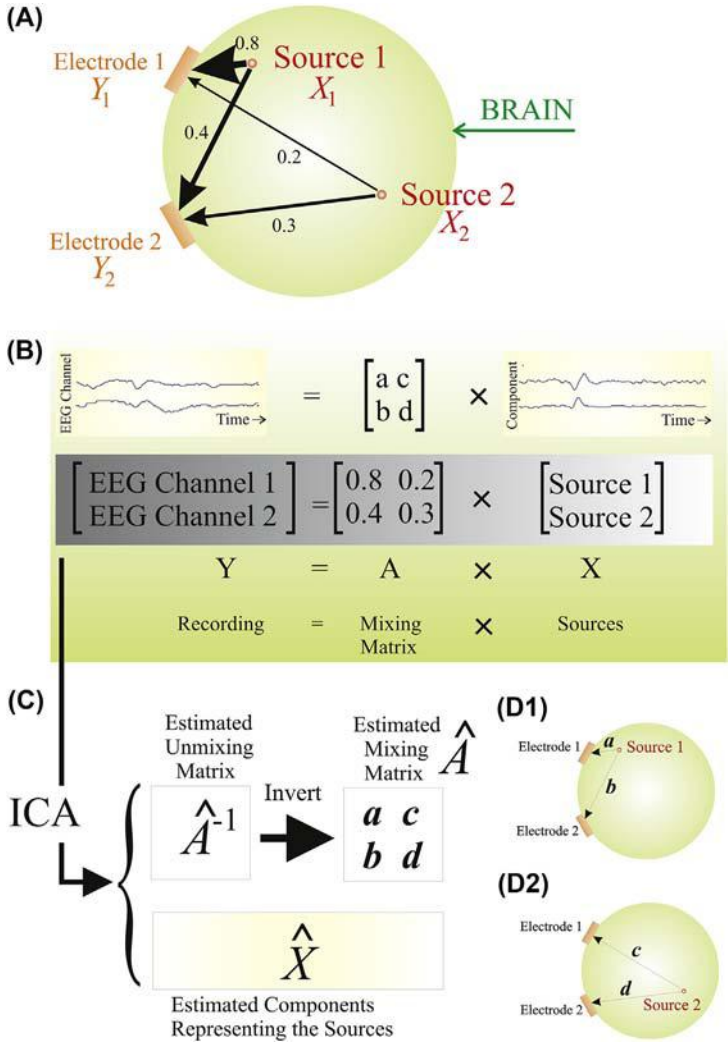


FIGURE 28.13 Independent component analysis (ICA) analysis in EEG: a two-source and two-channel example. (A) Electrodes Y_1 and Y_2 record channels 1 and 2, each containing a mixture of sources X_1 and X_2. In each mixture, the attenuation of the source signal is proportional with the distance between the source and electrode; symbolized by the arrows. Panel (B) shows the mathematics underlying the mixing process that can be represented by matrix multiplication $Y = AX$ with A being the mixing matrix (similar to the example in Fig. 28.2). The next step, depicted in (C), is to estimate the mixing matrix and source components with the ICA procedure. The estimated source activity can give an impression of the distribution of activities across the brain and the estimate of the mixing matrix can be used to determine the effect for each source on the electrodes (panel (D)).

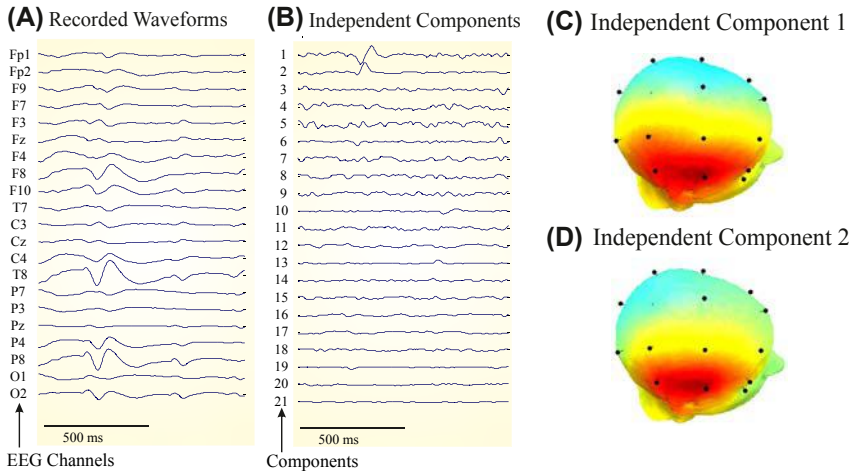


FIGURE 28.14 Part of the EEG recording shown in Fig. 28.1B is shown in panel (A); the 21 independent components are shown in (B). Here it can be seen that the epileptic spike waveforms are only represented in the first two independent components. Topographic maps of the scalp potential associated with these two components are shown in panels (C) and (D). These distributions are indicative for a source that is located right temporally. *This figure was prepared with eeglab software. This MATLAB®-based package can be downloaded from the following website: http://sccn.ucsd.edu/~scott/ica.html.*

A detail of the EEG recording in Fig. 28.1 (the epoch between the asterisks in Fig. 28.1B) is shown in in Fig. 28.14A. This EEG recording contains a high-amplitude epileptic spike. The ICA of this 21-channel record shows two components that seem associated with this spike signal (Components 1 and 2 in Fig. 28.14B). The topographic maps of both these independent sources show a right temporal location (Fig. 28.14C and D). Clinically it was confirmed that the epileptic focus was indeed located in the right temporal lobe in this patient. Although this confirmation is reassuring, it should be noted here that the brain area where epileptic spikes are generated and the focus where the epileptic seizures originate are not always the same location.

EXERCISES

28.1 As an example of a covariance matrix with identical eigenvalues, determine the principal components of a case of two uncorrelated variables with equal variance: kI (with k—constant and I—identity matrix)

28.2 Create a third waveform for the analysis in pr28_2.m and redo the analysis. What can you conclude?

28.3 Explain how PCA transforms observed data so that they become uncorrelated.
That is, show that the PCA procedure transforms the covariance-correlation matrix into a diagonal matrix where all off-diagonal elements are zero. Next, explain why this property translates into uncorrelated components.

References

Bell, A.J., Sejnowski, T.J., 1995. An information-maximization approach to blind separation and blind deconvolution. Neural Comput. 7, 1129–1159.

Cover, T.M., Thomas, J.A., 1991. Elements of Information Theory. John Wiley & Sons, New York.

Jordan, D.W., Smith, P., 1997. Mathematical Techniques. Oxford University Press, Oxford.

Lay, D.C., 1997. Linear Algebra and Its Applications. Addison-Wesley, New York.

Prentice, J.S., Homann, J., Simmons, K.D., Tkačik, G., Balasubramanian, V., Nelson, P.C., 2011. Fast, scalable, Bayesian spike identification for multi-electrode arrays. PLoS One 6, e19884.

Rossant, C., Kadir, S.N., Goodman, D.F., Schulman, J., Hunter, M.L., Saleem, A.B., Grosmark, A., Belluscio, M., Denfield, G.H., Ecker, A.S., Tolias, A.S., Solomon, S., Buzsáki, G., Carandini, M., Harris, K.D., 2016. Spike sorting for large, dense electrode arrays. Nat. Neurosci. 19, 634–641.

Shannon, C.E., Weaver, W., 1949. The Mathematical Theory of Communication. University of Illinois Press, Urbana, IL.

Stone, J.V., 2004. Independent Component Analysis: A Tutorial Introduction. MIT Press, Cambridge, MA.

CHAPTER

29

Modeling Neural Systems: Cellular Models*

29.1 INTRODUCTION

A fundamental and famous model in neuroscience was described by Hodgkin and Huxley (HH) in 1952. They modeled the dynamics of the membrane potential of the giant axon of squid, describing measurements of sodium, potassium, and leakage currents. Their approach can be represented by an equivalent electronic circuit of the axon's membrane in which a capacitor models the membrane's phospholipids and several voltage-dependent resistors represent its ion channels (see Appendix 1.1 if you need to review the basic laws for electrical circuits). Much later, after computer technology had become readily available, their formalism was widely employed and extended to include other types of ion-channels. In addition, it is used to create detailed cell models in which the cell is divided into a set of coupled compartments each represented by a separate membrane model. In many studies these computational models are embedded in networks (e.g., Traub et al., 2005; Lytton and Sejnowski, 1991; De Schutter and Bower, 1994; van Drongelen et al., 2005, 2006; Markam, 2006). In this approach, the individual nodes of the network and their connections are simulated with the purpose of finding the properties associated with emergent network activities such as the generation of synchronized bursts and oscillations. The advantage of these models is that they are close to the experimental domain, i.e., they may include recorded ion conductance values, morphology of observed cell types, and known intercellular connectivity. In spite of the many

*Parts of this chapter are based on previously published work by the author (van Drongelen, 2013).

Signal Processing for Neuroscientists, Second Edition
https://doi.org/10.1016/B978-0-12-810482-8.00029-1

parameters in these models, an overwhelming amount of detail is still missing. As a consequence, due to nonlinearities in neuronal function (e.g., membrane conductance dynamics, the coupling function between neurons), small inaccuracies in the model parameters may lead to large prediction errors of the model. In spite of this potential shortcoming, some models perform surprisingly well in predicting or mimicking experimental outcomes (e.g., McCormick and Huguenard, 1992).

A general problem with detailed models is that, because of their complexity, they cannot be analyzed mathematically, which may obstruct obtaining a deeper understanding of the process it models. Further model reduction is required before mathematical analysis can be employed. For example, the HH model is a four-dimensional nonlinear model (the dynamic variables are membrane potential V, and gating variables m, h, and n). Both Fitzhugh (1961) and Nagumo et al. (1962), inspired by the nonlinear van der Pol oscillator (van der Pol and van der Mark, 1927), reduced the HH formalism to a single pair of dynamic variables (membrane potential V and a recovery variable w) to gain insight into the dynamics of the excitable membrane. Later a similar simplified two-dimensional approach was published for the excitation in the barnacle giant muscle fiber by Morris and Lecar (1981).

A further simplification for simulation of nerve cell activity is the leaky integrate-and-fire (IF) model (Lapicque, 1907). The basic IF model describes a neuron's subthreshold behavior as a leaky integrator (a resistor and capacitor) and adds an ad hoc superthreshold spike when a preset threshold is exceeded. In between the spike events, the IF circuit is a one-dimensional linear model for the cell's subthreshold activity. There are additions to this model, for instance, resonance properties are included by making the model two-dimensional and a quadratic function has been proposed to add a nonlinear component that is capable of generating the spike upward deflection (Latham et al., 2000). More recently, Izhikevich (2007) extended and combined these approaches into what he defines as the simple model of choice (SMC). In this model there is no threshold for the spike generator, but there is an ad hoc downward deflection (i.e., a reset) to terminate the spike.

29.2 THE HODGKIN AND HUXLEY FORMALISM

The HH equations (Hodgkin and Huxley, 1952) describe the nonlinear dynamics of the electrical properties of an excitable membrane of the giant nerve fiber that innervates the muscles involved in water jet propulsion in the squid. The inside and outside of the nerve fiber are separated by a biomembrane (Chapter 1, Fig. 1.1A). The principal part of the biomembrane consists of a double layer of phospholipids that serves as an

insulator between the conductive media inside and outside of the fiber; this component can be modeled as a capacitor (C in Fig. 1.1A). Ion pumps in the membrane create gradients of different ion species, such as sodium and potassium, across the membrane, and ion channels allow passive leakage of the ions.

Because ions are charged, an ion pump creates a potential difference and can therefore be modeled as a potential source (E in Fig. 1.1A). For individual ion species, these potentials are known as Nernst potentials that can be computed by the Nernst equation: e.g., at 25°C, the Nernst potential for the positively charged sodium and potassium ions is determined from their intra- and extracellular concentrations ($[C_{in}]$, $[C_{out}]$, respectively) as: $58 \log_{10}([C_{out}]/[C_{in}])$ mV. An ion channel conducts one or more ion species and may be modeled as a resistor (R in Fig. 1.1A). The resistances of the sodium and potassium channels are not static but depend on the potential across the biomembrane. This membrane potential dependence is modeled by the HH equations using activation and inactivation processes that govern the degree of opening of the channel and consequently its conductance. It is of interest to note that HH proposed their formalism in the 1950s as an empirical model, and that molecular evidence for the existence of the different channels that are responsible for membrane potential-dependent conductance came decades later!

In the HH formalism, the potential difference between the fiber's inside and outside, the membrane potential (V) is related to activation and inactivation parameters of sodium and potassium ion channels (m, n, h). The differential equations that HH used to describe the dynamics are:

$$C\frac{dV}{dt} = -I_{\text{Leak}} - I_{\text{Na}} - I_{\text{K}} + I_{\text{Inject}}, \text{ or}$$

$$C\frac{dV}{dt} = -g_{\text{Leak}}(V - E_{\text{Leak}}) - \overline{g}_{\text{Na}} m^3 h (V - E_{\text{Na}}) - \overline{g}_{\text{K}} n^4 (V - E_{\text{K}}) + I_{\text{Inject}}. \tag{29.1}$$

The dynamics of parameters m, h, and n are governed by three first-order differential equations:

$$\frac{dm}{dt} = \frac{m_\infty(V) - m}{\tau_m(V)}, \tag{29.2}$$

$$\frac{dh}{dt} = \frac{h_\infty(V) - h}{\tau_h(V)}, \tag{29.3}$$

$$\frac{dn}{dt} = \frac{n_\infty(V) - n}{\tau_n(V)}, \tag{29.4}$$

Here, the variables τ_m, τ_h, and τ_n are the time constants of parameters m, h, and n.

Eq. (29.1) is based on Kirchhoff's first law stating that the sum of all membrane currents is equal to zero. An equivalent membrane circuit is depicted in Fig. 29.1. Using the equation $Q = CV$, with: Q—charge of the capacitor, C—its capacitance, and V—potential across the capacitor.

We determine that current I_C equals $C\frac{dV}{dt}$, the part to the left of the equal sign in Eq. (29.1).

Each ion current i_k is determined by Ohm's law as $g_k(V - E_k)$, with: g_k—conductance ($=1/R_k$) of ion species k, V—membrane potential, and E_k—equilibrium potential for ion species k.

Parameter g_k can be presented by the product of the maximum conductance $\overline{g}_k$ and activation and inactivation parameters. These ion currents represent the terms right of the equal sign in Eq. (29.1). Lastly, the expression in Eq. (29.1) includes a term for injected current, I_{Inject}.

Eqs. (29.2) and (29.3) describe the dynamics for the activation (m) and inactivation (h) parameters of the sodium channel. Eq. (29.4) describes the dynamics of the activation (n) of potassium. These activation and inactivation parameters are V-dependent, and these relationships are all nonlinear as well as the activation parameters associated with m and n, since HH determined experimentally that the model fits best if they used m^3 and n^4 in Eq. (29.1).

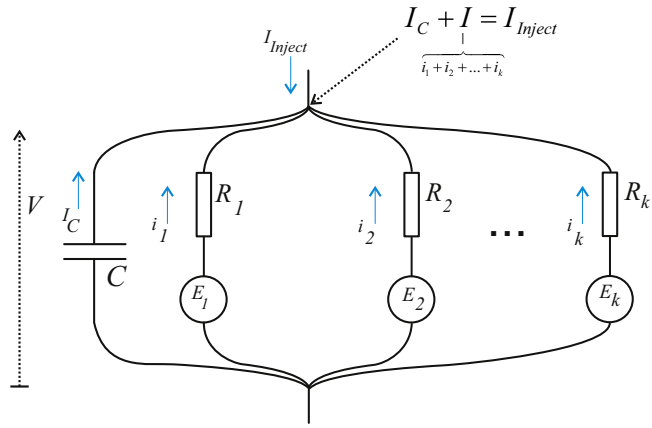


FIGURE 29.1 Membrane equivalent circuit with k ion channels. E_1, E_2, …, E_k are the Nernst potentials for the individual ion species; R_1, R_2, …, R_k and i_1, i_2, …, i_k represent the resistance of the different ion channels and their currents, respectively. I_{inject} is the current injected. The membrane capacitance and the associated current are C and I_C. V is the membrane potential. The equation $I_C + I = I_{\text{inject}}$ is Kirchhoff's law applied to the circuit. This circuit is an extension of the one shown in Fig. 1.1.

In the following we represent activation/inactivation variables generically by x. It can be seen in Eqs. (29.2)–(29.4) that the parameters x_∞ and τ_x depend on the membrane potential V. The dynamics of the parameters in Eqs. (29.2)–(29.4) were originally presented by HH in the form:

$$\frac{dx}{dt} = \alpha_x(1 - x) - \beta_x x,$$

where α_x and β_x are rate constants that are membrane potential (voltage) sensitive. This presentation can be linked to the one in Eqs. (29.2)–(29.4) by $\tau_x = \frac{1}{\alpha_x+\beta_x}$ and $x_\infty = \frac{\alpha_x}{\alpha_x+\beta_x}$. The voltage-dependent functions for α_x and β_x were determined by HH using the so-called voltage-clamp technique. This technique enables the experimenter to keep V constant, the holding potential, while measuring membrane current. By repeating this measurement for a range of holding potentials, HH determined the I–V relationship. This relationship was determined for sodium and potassium currents (a specific ion current can be determined by disabling other ion currents pharmacologically). In their model, HH used these I–V relationships to determine α_x and β_x. As can be seen in Eq. (29.1), the total membrane current is proportional to the time derivative of the membrane potential $dV/dt = \dot{V}$; therefore, the experimentally determined I–V relationship can be interpreted as the $\dot{V} - V$ phase space (Chapter 10) for membrane potential dynamics (e.g., Izhikevich, 2007), and an example is given in Martell et al. (2012).

MATLAB® routine pr29_1 simulates the behavior of a model neuron governed by the HH formalism. Note that the functions for initializing and updating the dynamic variables m, h, and n are required for this script. In the 1950s it was quite a task for HH to compute the shape of the action potential, but with our current access to computing resources it is simple to simulate their equations. However, to make this type of approach accessible for mathematical analysis, some simplification is required. As we will describe in the following, common strategies to simplify the 4-D HH model are:

1. linearization, and
2. reduction of the number of dimensions of the model.

29.2.1 Linearization of the Hodgkin and Huxley Equations

A linearized version of the HH equations is useful to examine small subthreshold behavior of the membrane potential around a resting or holding potential (say V^*). More mathematical details of the following can be found in Appendix 29.1. In order to linearize the nonlinear

equations we rewrite Eqs. (29.1)–(29.4) in a more compact form as $C\frac{dV}{dt} = f(V, m, h, n, I_{\text{Inject}})$ and $\frac{dx}{dt} = f(V, x)$, with x representing an activation/inactivation variable m, h, or n. Then we linearize about an equilibrium potential V^*. We define small variations in the parameters about this equilibrium as $\delta V, \delta m, \delta h, \delta n, \delta I$. Using this, we obtain a linearized version of Eq. (29.1):

$$C\frac{d\delta V}{dt} = \left(\frac{\partial f}{\partial V}\right)_* \delta V + \left(\frac{\partial f}{\partial m}\right)_* \delta m + \left(\frac{\partial f}{\partial h}\right)_* \delta h + \left(\frac{\partial f}{\partial n}\right)_* \delta n + \left(\frac{\partial f}{\partial I_{\text{inject}}}\right)_* \delta I_{\text{inject}}, \tag{29.5}$$

with: $f = f(V, m, h, n, I_{\text{Inject}})$, $\left(\frac{\partial f}{\partial V}\right)_* = -\left(g_{\text{Leak}} + \overline{g}_{\text{Na}} m^3 h + \overline{g}_{\text{K}} n^4\right)\big|_*$,

$\left(\frac{\partial f}{\partial m}\right)_* = -\overline{g}_{\text{Na}} 3m^2 h(V - E_{\text{Na}})\big|_*$, $\left(\frac{\partial f}{\partial h}\right)_* = -\overline{g}_{\text{Na}} m^3 (V - E_{\text{Na}})\big|_*$,

$\left(\frac{\partial f}{\partial n}\right)_* = -\overline{g}_{\text{K}} 4n^3 (V - E_{\text{K}})\big|_*$, $\left(\frac{\partial f}{\partial I_{\text{inject}}}\right)_* = 1$

For the gating parameters (Eqs. (29.2)–(29.4)), using $\frac{dx}{dt} = \frac{x_\infty(V) - x}{\tau_x(V)} = f(V, x)$, we get the linearized version:

$$\frac{d\delta x}{dt} = \left(\frac{1}{\tau_x(V)}\right)_* \left(\frac{dx_\infty(V)}{dV}\right)_* \delta V - \left(\frac{1}{\tau_x(V)}\right)_* \delta x \tag{29.6}$$

Note that all expressions $(\ldots)_*$ in this section must be evaluated at V^*. Therefore, each $(\ldots)_*$ represents a number and not an equation! In the literature it is common to further simplify the notation of Eqs. (29.5) and (29.6). For example, in Richardson et al. (2003) Eqs. (29.5) and (29.6) are simplified to (Appendix 29.1):

$$C\frac{dv}{dt} = -g_M v - \sum_{x=1}^{N} g_x w_x + \underbrace{\delta I_{\text{inject}}}_{I_{\text{app}}} \tag{29.7}$$

and

$$\tau_x(V)\frac{dw_x}{dt} = v - w_x. \tag{29.8}$$

29.2.1.1 *Linearization of the Potassium Channel*

Now we consider the potassium current and model small perturbations and show we can model this by using a two-branch electrical circuit

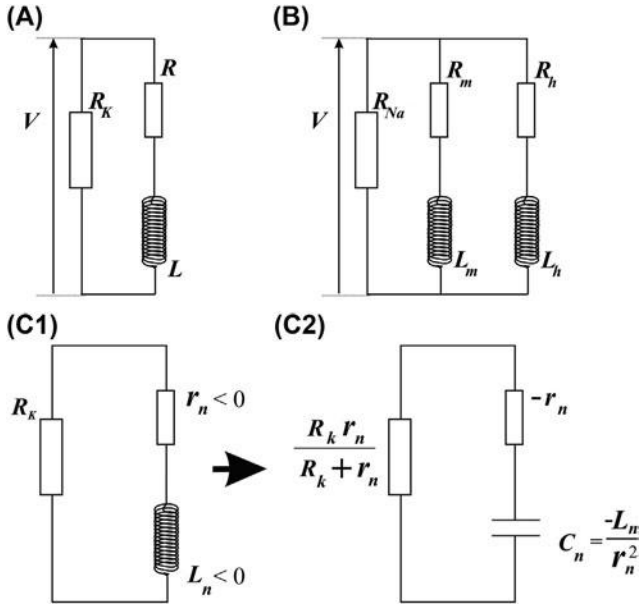


FIGURE 29.2 Diagrams of equivalent circuits representing ion channels. (A) A two-branch equivalent circuit for a linearized potassium channel. The potassium channel has properties similar to an inductor. A change in current flowing through an inductor creates a time-varying magnetic field inside the coil; this induces a potential that opposes the change in current that created it. In the potassium channel, a change of K^+ current is opposed by the change in membrane potential and the associated change in K^+ conductance caused by the current. (B) Equivalent circuit for linearized sodium activation and inactivation. (C) A part of the linearized membrane model for the sodium channel with negative values for both L_n and r_n (C1) and its equivalent RC circuit (C2).

with resistors R, R_K, and inductance L (Fig. 29.2A). A small perturbation in the circuit with the two parallel branches is governed by:

$$\delta V = \frac{\delta I}{\frac{1}{R_K} + \underbrace{\frac{1}{\left(R + L\frac{d}{dt}\right)}}_{Z_{RL}}} \tag{29.9}$$

Eq. (29.9) is just Ohm's law ($V = IZ$) applied to the circuit in Fig. 29.2A. Here Z, the inverse of $\frac{1}{R_K} + \frac{1}{\left(R+L\frac{d}{dt}\right)}$, represents the impedance of the circuit that links the fluctuations in potential and current (R and L are in series and they are parallel to R_K). Now if we consider a single ion

current, the K^+ current, we show that we can write this in the same form as Eq. (29.9). Experimentally, a single ion current can be measured by disabling all other currents pharmacologically.

First we restate, from Eq. (29.1), that the potassium current I_K is given by:

$$I_K = \overline{g}_K n^4 (V - E_K) \tag{29.10}$$

A small perturbation approximated by a linearization around potential V^* is

$$\delta I_K = \left(\frac{\partial I_K}{\partial V}\right)_* \delta V + \left(\frac{\partial I_K}{\partial n}\right)_* \delta n \tag{29.11}$$

with $\left(\frac{\partial I_K}{\partial V}\right)_* = \overline{g}_K n^4\big|_* = G_K^*$, and $\left(\frac{\partial I_K}{\partial n}\right)_* = 4\overline{g}_K n^3 (V - E_K)\big|_*$

We can now use Eq. (29.6), substituting n for x, to relate δn to δV

$$\delta n = \frac{\left(\frac{1}{\tau_n}\right)_* \left(\frac{dn_\infty}{dV}\right)_*}{\left(\frac{1}{\tau_n}\right)_* + \frac{d}{dt}} \delta V$$

Substituting this result into Eq. (29.11) gives

$$\delta I_K = G_K^* \delta V + \left(4\overline{g}_K n^3 (V - E_K)\right)_* \frac{\left(\frac{1}{\tau_n}\right)_* \left(\frac{dn_\infty}{dV}\right)_*}{\left(\frac{1}{\tau_n}\right)_* + \frac{d}{dt}} \delta V$$

This result can be written as

$$\delta V = \frac{\delta I_K}{\left(G_K^* + \left[\frac{A^*}{B^* + \frac{d}{dt}}\right]\right)} \tag{29.12}$$

with $A^* = \left(4\overline{g}_K n^3 (V - E_K)\right)_* \left(\frac{1}{\tau_n}\right)_* \left(\frac{dn_\infty}{dV}\right)_*$ and $B^* = \left(\frac{1}{\tau_n}\right)_*$.

Now we can directly relate the linearized potassium channel to the circuit in Fig. 29.2A. Comparing Eqs. (29.9) and (29.12), we can determine that both expressions are equivalent if $R_K = \frac{1}{G_K^*}$, $R = \frac{B^*}{A^*}$, and $L = \frac{1}{A^*}$.

Combining the above we find that

$$L = \frac{R}{B^*} = R\tau_n^* \quad \text{and} \quad R = \frac{B^*}{A^*} = \frac{1}{\left(4\overline{g}_K n^3 (V - E_K)\right)_* \left(\frac{dn_\infty}{dV}\right)_*}.$$

Thus we can successfully model small perturbations in the potassium channel with the equivalent circuit in Fig. 29.2A. Note that A^* is positive because $\frac{dn_\infty}{dV}$ and $V-E_K$ are both positive at physiological values of V^*. Therefore L is positive and (because B^* is also > 0) R is positive.

29.2.1.2 *Linearization of the Sodium Channel*

Now we apply the linearization to inward current I_{Na}:

$$I_{Na} = \overline{g}_{Na} m^3 h (V - E_{Na}) \tag{29.13}$$

A small perturbation approximated by a linearization around potential V^* is

$$\delta I_{Na} = \left(\frac{\partial I_{Na}}{\partial V}\right)_* \delta V + \left(\frac{\partial I_{Na}}{\partial m}\right)_* \delta m + \left(\frac{\partial I_{Na}}{\partial h}\right)_* \delta h \tag{29.14}$$

We simplify the notation

$$\delta I_{Na} = \left(G^*_{Na} + \left[\frac{A^*}{B^* + \frac{d}{dt}}\right] + \left[\frac{D^*}{E^* + \frac{d}{dt}}\right] \right) \delta V \tag{29.15}$$

using: $\left(\frac{\partial I_{Na}}{\partial V}\right)_* = \overline{g}_{Na} m^3 h\big|_* = G^*_{Na}, A^* = \left(3\overline{g}_{Na} m^2 h (V - E_{Na})\right)_* \left(\frac{1}{\tau_m}\right)_* \left(\frac{dm_\infty}{dV}\right)_*$,

$B^* = \left(\frac{1}{\tau_m}\right)_*$, $D^* = \left(\overline{g}_{Na} m^3 (V - E_{Na})\right)_* \left(\frac{1}{\tau_h}\right)_* \left(\frac{dh_\infty}{dV}\right)_*$, and $E^* = \left(\frac{1}{\tau_h}\right)_*$

From the three terms in Eq. (29.14), it is obvious that we need three branches in an equivalent electrical circuit for the sodium channel. One resistor R_{Na} and two branches each with a resistor and inductor, one for activation m and one for inactivation h (Fig. 29.2B). From the circuit properties in Fig. 29.2B we determine that a perturbation of the potential is governed by

$$\delta V = \frac{\delta I}{\frac{1}{R_{Na}} + \underbrace{\frac{1}{\left(R_m + L_m \frac{d}{dt}\right)}}_{Z^m_{RL}} + \underbrace{\frac{1}{\left(R_h + L_h \frac{d}{dt}\right)}}_{Z^h_{RL}}} \tag{29.16}$$

We find that the expressions in (29.15) and (29.16) are equal if: $R_{\mathrm{Na}} = \frac{1}{G^*_{\mathrm{Na}}}$, $R_m = \frac{B^*}{A^*}$, and $L_m = \frac{1}{A^*}$

with: $L_m = \frac{R_m}{B^*} = R_m \tau^*_m$, $R_m = \frac{B^*}{A^*} = \frac{1}{\left(3\bar{g}_{\mathrm{K}} m^2 h (V - E_{\mathrm{Na}})\right)_* \left(\frac{dm_\infty}{dV}\right)_*}$, $R_h = \frac{E^*}{D^*}$, and $L_h = \frac{1}{D^*}$.

Combining the above we find: $L_h = \frac{R_h}{E^*} = R_h \tau^*_h$ and $R_h = \frac{E^*}{D^*} = \frac{1}{\left(\bar{g}_{\mathrm{K}} m^3 (V - E_{\mathrm{Na}})\right)_* \left(\frac{dh_\infty}{dV}\right)_*}$.

So we can successfully model small perturbations in the sodium channel with the equivalent circuit in Fig. 29.2B. Note that in the expression above A^* is negative because $\frac{dm_\infty}{dV} > 0$ and $V - E_{\mathrm{Na}}$ is negative at physiological values of V^*. Therefore L_m is negative and (because B^* is > 0) R_m is also negative. In contrast, note that D^* is positive because both $\frac{dh_\infty}{dV}$ and $V - E_{\mathrm{Na}} < 0$ at V^*. Therefore L_h is positive and (because E^* is > 0) R_h is also positive. The model can easily be evaluated by simulation; however, building the model using hardware components cannot be done as simply because some of the conductance and inductance values are negative over the range we are interested in. To address this, Mauro (1970) and Koch (1999) proposed replacing the negative inductance by a positive capacitance and correcting for negative resistance in one of the other branches of the circuit. They showed that a branch with negative conductance and inductance parallel to a true (positive valued) resistor can be replaced with an RC circuit without changing the effective impedance of the two branches. The equivalent diagrams of the circuitry are shown in Fig. 29.2C. In the original circuit (Fig. 29.2C1) with the positive resistor R_{K}, negative resistance r_n, and negative inductor L_n we have impedance Z_o in the original circuit:

$$Z_o = \left(\frac{1}{R_{\mathrm{K}}} + \frac{1}{r_n + j\omega L_n}\right)^{-1} = \left(\frac{(R_{\mathrm{K}} + r_n) + j\omega L_n}{R_{\mathrm{K}} r_n + j\omega R_{\mathrm{K}} L_n}\right)^{-1}.$$

In the replacement circuit (Fig. 29.2C2) with the capacitor we have impedance Z_r:

$$Z_r = \left(\frac{R_{\mathrm{K}} + r_n}{R_{\mathrm{K}} r_n} + \frac{1}{\left(-r_n + \frac{1}{j\omega C_n}\right)}\right)^{-1}$$

$$= \left(\frac{-R_{\mathrm{K}} r_n + \frac{R_{\mathrm{K}}}{j\omega C_n} - r_n^2 + \frac{r_n}{j\omega C_n} + R_{\mathrm{K}} r_n}{\left(-R_{\mathrm{K}} r_n^2 + \frac{R_{\mathrm{K}} r_n}{j\omega C_n}\right)}\right)^{-1}$$

After substituting $C_n = \frac{-L_n}{r_n^2}$ and multiplying the numerator and denominator by $-j\omega L_n / r_n^2$ we get:

$$Z_r = \left(\frac{(R_K + r_n) + j\omega L_n}{R_K r_n + j\omega R_K L_n} \right)^{-1}$$

This result is identical to the original impedance Z_o; it will therefore also have the same frequency response: absolute value and phase (i.e., as shown in Chapter 17, the same Bode plot).

29.2.1.3 *Generalized Equivalent Circuit*

In the previous sections it was shown that the channels in a linearized version of the HH equations can be represented by equivalent circuits consisting of a network of resistors and inductors (Fig. 29.2). Even in the case where negative values occur for these components (due to positive feedback between sodium conductance and membrane depolarization), they can be represented by positive-valued alternative electronic components (Fig. 29.2C). When combining these findings, we simplify the equivalent circuit in Fig. 29.1 and consider the circuit with a resistor (R), capacitor (C) and inductor (L), the RCL network in Fig. 29.3 as a linearized representation of the membrane.

The inductor channel in the circuit represents the stabilizing effects of the outward current activation (e.g., potassium) and inward current inactivation (e.g., sodium). The resistor and capacitor represent membrane resistance and capacitance plus corrections for negative

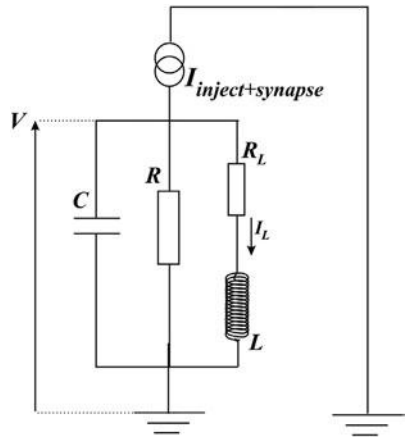


FIGURE 29.3 Electrical equivalent circuit of the linearized Hodgkin and Huxley equations.

impedances. Since we look at small perturbations around an equilibrium, which we conveniently set at zero, the driving forces of the difference between Nernst potential and resting potential can be ignored.

The pair of differential equations that govern the circuit in Fig. 29.3 are:

$$C\frac{dV}{dt}+\frac{V}{R}+I_L=I_{\text{inject+synapse}} \text{ and } L\frac{dI_L}{dt}=V-R_LI_L \tag{29.17}$$

We now simplify and nondimensionalize the system. First we substitute $w=R_LI_L$ and $I_0=I_{\text{inject+synapse}}$ and get $C\frac{dV}{dt}+\frac{V}{R}+\frac{w}{R_L}=I_0$ and $\frac{L}{R_L}\frac{dw}{dt}=V-w$. Then we change to a timescale τ such that $\tau=\frac{R_L}{L}t\rightarrow dt=\frac{L}{R_L}d\tau$; substituting this we get the following pair of ODEs:

$$C\frac{dV}{\left(\frac{L}{R_L}\right)d\tau}+\frac{V}{R}+\frac{w}{R_L}=I_0\rightarrow\frac{dV}{d\tau}+\left(\frac{L}{CR_L}\right)\frac{V}{R}+\left(\frac{L}{CR_L}\right)\frac{w}{R_L}$$

$$=\left(\frac{L}{CR_L}\right)I_0, \text{ and}$$

$$\frac{L}{R_L}\frac{dw}{\left(\frac{L}{R_L}\right)d\tau}=V-w\rightarrow\frac{dw}{d\tau}=V-w$$

Finally, using $V=w'+w$ and $V'=w''+w'$ (note that $w'=dw/d\tau$), we obtain the following second-order ODE:

$$w''+w'+\left(\frac{L}{CRR_L}\right)w'+\left(\frac{L}{CRR_L}\right)w+\left(\frac{L}{CR_L^2}\right)w=\left(\frac{L}{CR_L}\right)I_0\rightarrow$$

$$w''+\left[1+\underbrace{\left(\frac{L}{CRR_L}\right)}_{\alpha}\right]w'+\left[\underbrace{\left(\frac{L}{CRR_L}\right)+\left(\frac{L}{CR_L^2}\right)}_{\alpha+\beta}\right]w=\left(\frac{L}{CR_L}\right)I_0$$

This result for α and β is identical to the equations analyzed by Erchova et al. (2004) (their Equations (A16) and (A17)). An alternative route to create a single equation from the two original ones is by direct substitution while leaving dimensionality intact. If we differentiate the first expression in Eq. (29.17) with respect to time we have: $C\ddot{V}+\frac{\dot{V}}{R}+\dot{I}_L=\dot{I}_0$.

Here $\ddot{V}$, $\dot{V}$, and $\dot{I}$ indicate the second and first time derivatives of V and I. We can use the second expression in Eq. (29.17) to substitute $\dot{I}_L = \frac{V - R_L I_L}{L}$ where I_L as a function of V can be found from $C\dot{V} + \frac{V}{R} + I_L = I_0$. This yields:

$$\begin{aligned} C\ddot{V} + \frac{\dot{V}}{R} + \frac{V - R_L\left(I_0 - C\dot{V} - V/R\right)}{L} = \dot{I}_0 \rightarrow \\ C\ddot{V} + \underbrace{\left(\frac{1}{R} + \frac{R_L C}{L}\right)}_{\gamma}\dot{V} + \underbrace{\frac{1}{L}\left(1 + \frac{R_L}{R}\right)}_{\varepsilon}V = \frac{R_L}{L}I_0 + \dot{I}_0 \end{aligned} \tag{29.18}$$

This result is identical to the expression used by Erchova et al. (2004) (their Equation A1). Interestingly, this equation is also similar to the network equation by Jirsa and Haken (1997) in which the spatial derivative is set to zero (see Chapter 30). This similarity across the scale from neuron to network is seen frequently. In one example, investigators even use a cellular model for action potential generation to represent the activity of a population (Curto et al., 2009). Although Curto et al. (2009) don't explicitly describe this, in this case the inward and outward currents of the cell membrane can be considered to represent the network's excitation and inhibition, respectively.

29.2.1.4 The Frequency Response of the Linearized Model

In electrophysiology, cellular properties are often probed with hyperpolarizing and depolarizing current steps. The unit impulse response, the derivative of the unit step (Chapter 2), is the inverse Fourier transform or inverse Laplace transform of the system's frequency response or transfer function, respectively (Chapters 6 and 12). Since we deal with a linearized system with current I as input and membrane potential V as output, the frequency response is just the impedance Z (i.e., $Z(\omega) = V(\omega)/I(\omega)$, with ω − frequency in rad/s). We can use the Laplace transform of Eq. (29.18) to obtain the transfer function $H(s)$ of the linearized model:

$$H(s) = \frac{V(s)}{I(s)} = \frac{\left(R_L/L\right) + s}{\varepsilon + \gamma s + Cs^2}.$$

In this form we can use the MATLAB® *freqs* command to depict the Bode plot (Chapter 17) of the model as depicted in Fig. 29.4. The MATLAB® routine pr29_2 produces this plot.

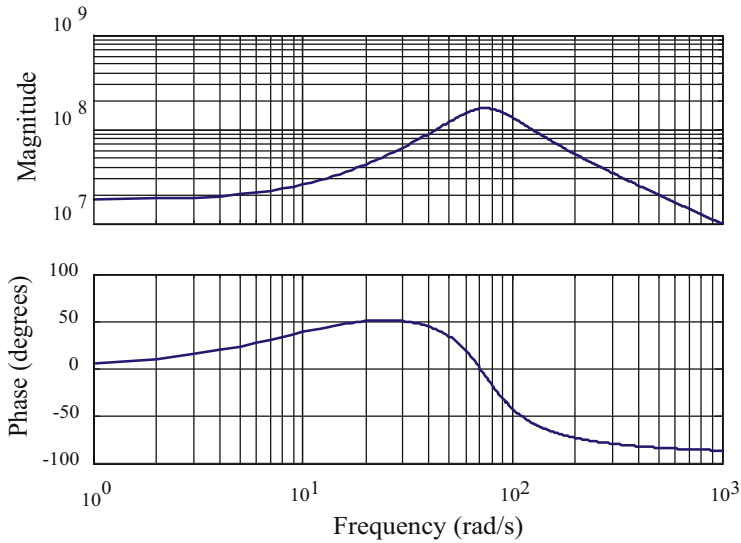


FIGURE 29.4 Bode plot of the linearized model depicted in Fig. 29.3. The parameters used to prepare the plots are $C = 100$ pF; $R = 200$ MΩ; $R_L = 20$ MΩ; $L = 2$ MH. Note in the upper panel that a resonance peak is present at 75 rad/s, which corresponds to a frequency of 11.9 Hz.

```
% pr29_2

clear;
close all;

C = 1e-10;
R = 2e8;
RL = 2e7;
L = 2e6;
gam=(1/R) + (RL*C/L);
eps=(1/L)*(1+(RL/R));
B = [1 RL/L];
A = [C gam eps];
w = 0:1000;
freqs(B,A,w)
```

An interesting observation is that the Bode plot in Fig. 29.4 shows a peak in the magnitude of the impedance, indicating that neurons and their networks display resonance (Hutcheon and Yarom, 2000; Brunel et al., 2003; Erchova et al., 2004; Dwyer et al., 2010, 2012). Here, with an appropriate and realistic choice of parameters, this resonance peak is located in the alpha frequency band of the EEG (at 75 rad/s, 11.9 Hz).

29.3 MODELS THAT CAN BE DERIVED FROM THE HODGKIN AND HUXLEY FORMALISM

The HH representation and its linearized version can be seen as a basis for many different neuronal models. Of course one could either (1) add complexity to the HH formalism or (2) further reduce it. The following sections present representative examples of both types.

29.3.1 Complex Cell Models

Using the HH formalism one might extend the sodium and potassium channels of the model neurons to also include different types of potassium channels (such as the A-current, a transient potassium current), persistent sodium channels, and different types of Ca^{2+} channels. In addition to the voltage-sensitive channels it is common practice to also include Ca^{2+}- and/or Na^{+}-sensitive channels. In addition to the addition of channels, the cell model can also be divided into many multiple coupled compartments (e.g., De Schutter and Bower, 1994). In this type of model, each compartment represents a small part of the cell that can be approximated by a compartment with a uniform membrane potential. The model can be further developed to also include synaptic input and experimental current injection. In that case the model presented in Fig. 29.1 can be extended to multiple compartments. A diagram of a multicompartment model for a part of a neuronal structure is depicted in Fig. 29.5. If we focus on compartment n in this multicompartment model and apply Kirchhoff's first law to the node indicated by the arrow in Fig. 29.5, we can modify Eq. (29.1) as follows:

$$C\frac{dV}{dt} + I_{\text{Leak}} + I_{V-\text{Sensitive}} + I_{\text{Synapse}} = I^{\text{total}}_{\text{Intercompartmental}} + I_{\text{Inject}} \quad (29.19)$$

The part of the intercompartmental current between Compartment n and Compartment $n+1$ in Fig. 29.5 can be determined by Ohm's law:

$$I_{\text{Intercompartmental}} = \frac{V_{n+1} - V_n}{R_{\text{axial}}},$$

$V_{n+1} - V_n$ is the potential difference between the compartments $n+1$ and n and R_{axial} is the resistance between them. Note that for Compartment n there is a second component to $I^{\text{total}}_{\text{Intercompartmental}}$, namely the intercompartmental current between Compartment n and Compartment $n-1$. Adding this component, and assuming that the axial resistance

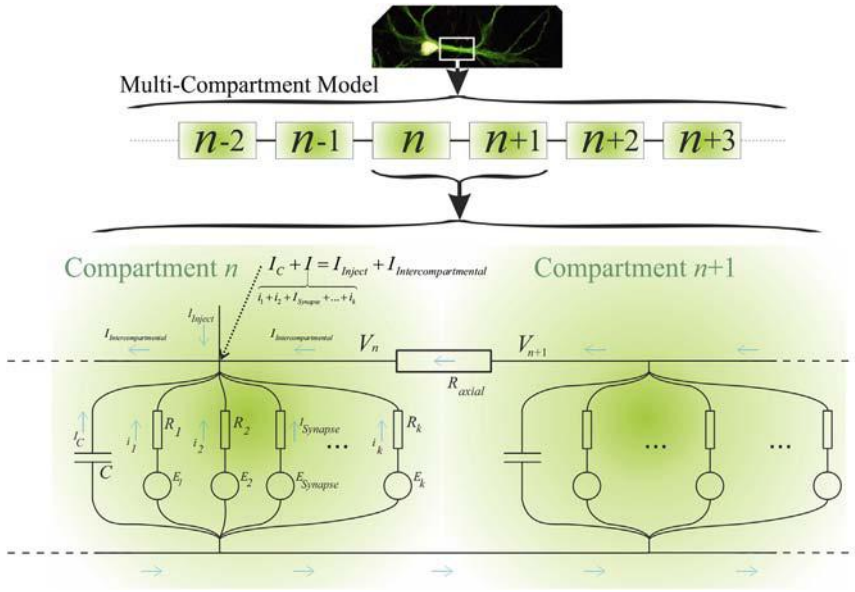


FIGURE 29.5 Multicompartment model of a neuronal structure and a detail showing compartments n and $n+1$. Each compartment consists of a unit governed by the HH formalism (as in Fig. 29.1), and they are connected via their internal, axial resistance R_{axial}. The intercompartmental current between compartments n and $n+1$ is determined by their potential difference and the axial resistance. *Top image from J. Suresh with permission.*

between compartments is the same, gives the **total** intercompartmental current for model compartment n in Fig. 29.5:

$$I^{\text{total}}_{\text{Intercompartmental}} = \frac{V_{n+1} - V_n}{R_{\text{axial}}} + \frac{V_n - V_{n-1}}{R_{\text{axial}}}$$

Also note that we do not include extracellular resistance in the model circuitry in Fig. 29.5—we implicitly lumped it into R_{axial}. In addition, the extracellular resistance between the compartments is small relative to the intracellular resistance and therefore it is often even conveniently omitted.

The rationale for the use of multiple equipotential compartments is to mimic the spatial separation of different functions: synaptic input, passive and active propagation of signals, and generation of action potentials. The more complex cell models are also frequently used to create networks in order to determine emergent behavior of neuronal populations (e.g., Lytton and Sejnowski, 1991; Traub and Llinas, 1979; Traub and Miles, 1991; Traub et al., 1994; van Drongelen et al., 2005, 2006). Special neural

simulation packages such as Genesis and Neuron have been developed to simulate such models, even on parallel machines (Bower and Beeman, 1998; Carnevale and Hines, 2006). An example in which many details are incorporated in the individual cell models is the Blue Brain Project (Markam, 2006). The advantage of these models is that they are based on experimental data enabling the modeler to link the model's behavior to measurements. This close relationship can also be a disadvantage of these complex models; due to the complexity the simulation results can be as difficult to interpret as real recorded data.

29.3.2 Simplified Cell Models

One can simplify the HH equations while preserving the nonlinearity, or one might use the linearization results described in Section 29.2.1. In fact, both these approaches have been employed. In one case, the four-dimensional HH model can be reduced to a lower-dimensional one and in the other case, the equivalent circuit of a linearized membrane model (Fig. 29.3) is coupled to an ad hoc spike generator. The advantage of preserving the nonlinearity is that the spike generation mechanism or a part thereof remains intact. However, mathematical analysis of such a model can be fairly complicated. The linearized cell models are more accessible for analysis but require an ad hoc spike generator (Table 29.1). Note that the simplified models summarized below have also been presented to illustrate low-dimensional nonlinear systems in Chapter 23. Therefore, I refer to Section 23.4 for additional details.

29.3.2.1 Fitzhugh–Nagumo and Morris and Lecar Models

The simplifications used by Fitzhugh (1961), Nagumo et al. (1962), and Morris and Lecar (1981), are all based on the observation that the cell has a relatively fast-activating inward current and a slower-activating outward current. This allows one to simplify Eqs. (29.1)–(29.4) from a four-dimensional model into a model with only two dynamic variables. In the Fitzhugh–Nagumo approach, the inward current is sodium, whereas in the Morris and Lecar model for a barnacle muscle cell, the inward current is calcium. For the following discussion it isn't critical which ion species carries the inward current, so I will focus on simplification of the HH formalism with a Na^+ current. In the original Eqs. (29.1)–(29.4) we have V, m, h, and n. The simplified version ignores the dynamics of the activation of the inward current m because it is so fast it is considered instantaneous. Furthermore, the effects of h and n are similar, i.e., they repolarize the membrane, so they can either be lumped or one of the two can be ignored so that only one repolarization parameter remains.

TABLE 29.1 Overview of the Dimensionality of Cellular Models

		Amplifying	Resonant			
	Membrane Potential (V)	**Inward Current Activation**	**Inward Current Inactivation**	**Outward Current Activation**	**Leak Current Capacitance (Not Time Dependent)**	**Physiological AP Mechanism**
Hodgkin and Huxley	1	(m) 2	(h) 3	(n) 4	Yes	Yes
Fitzhugh–Nagumo	1	Instantaneous (not time-dependent)	Generic recovery	(w) 2	Yes	Yes
Morris and Lecar (Ca instead of Na)	1	Instantaneous (not time-dependent)	No	(w) 2	Yes	Yes
IF	1	No	No	No	Yes	Generator
RF	1	No	No	Simplified (L) 2	Yes	Generator
QIF	1	Generic Amplification (Quadratic)	No	No	Yes	Reset
Izhikevich SMC	1	Generic Amplification (Quadratic)	No	Simplified (L) 2	Yes	Reset

AP, Action potential; L, inductor; IF, integrate and fire; QIF, quadratic integrate and fire; RF, resonate and fire; SMC, simple model of choice; V, m, h, n, parameters of the Hodgkin and Huxley Equations; w, recovery variable.

This generates a nonlinear dynamical system with two variables. Although a few variations exist, the original equations in Fitzhugh (1961) were

$$\begin{aligned}\dot{x} &= c\left(y + x - x^3/3 + z\right)\\ \dot{y} &= -(x - a + by)/c\end{aligned} \tag{29.20}$$

Here x, y, and z are membrane potential, recovery variable, and injected current, respectively; and a, b, and c are constants. Note the nonlinear, cubic relationship in the expression for $\dot{x}$. Also note that this form is similar, but not identical, to the equations we investigated in Chapter 23 (Eq. 23.6), repeated here for convenience:

$$\begin{aligned}\dot{V} &= I + V(a - V)(V - 1) - w\\ \dot{w} &= bV - cw\end{aligned}$$

Nagumo et al. (1962) employed an equivalent electronic circuit to simulate the membrane dynamics. They developed a circuit that is similar to the circuit we show in Fig. 29.3. However, in contrast to the model in Fig. 29.3, the version of Nagumo and coworkers is **not** a linearized version of the membrane currents. Instead, they included a tunnel diode to mimic the sodium current (TD in Fig. 29.6A). Note that the remaining two

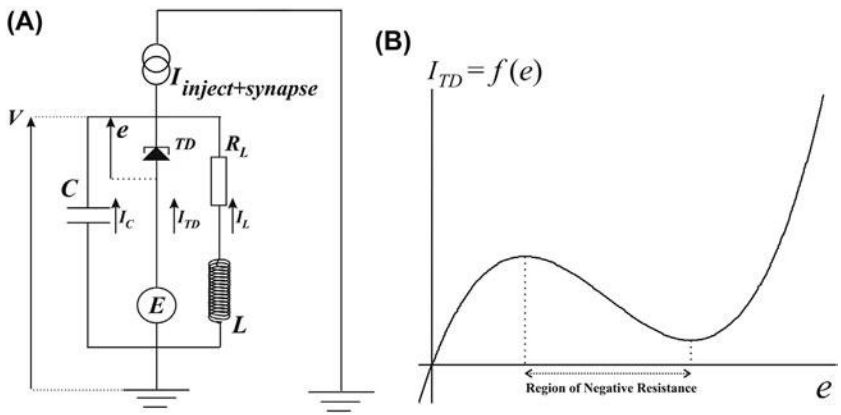


FIGURE 29.6 (A) The circuit developed and described by Nagumo et al. (1962). Note the similarity between this biomembrane equivalent circuit and the one we developed for the linearized version of membrane dynamics (Fig. 29.3). In this case the nonlinear sodium current is represented by the tunnel diode (TD). Note that the middle branch of this circuit includes the tunnel diode and a constant voltage source E that controls the level of e in order to adjust the current I_{TD}. In this model, V represents the membrane potential, and I_L is the recovery variable w in the FitzHugh–Nagumo model. Panel (B) is a plot of the I–V relationship (I_{TD} vs. e) of the tunnel diode that follows a third-order equation required for the FitzHugh–Nagumo formalism. Importantly, it includes a region of negative resistance to simulate the effect of the sodium current.

branches in the circuit in Fig. 29.6A are similar to those in Fig. 29.3: (1) a branch with C to model the membrane capacitance, and (2) a branch with R and L components to represent the recovery (e.g., the K^+ current). The tunnel diode possesses a relationship between current and potential (I_{TD} and e, Fig. 29.6B) that can be described by a third-order function (f, Fig. 29.6B). Applying Kirchhoff's laws to the circuit in Fig. 29.6A, we have $I = I_C + I_{TD} + I_L$, and in the middle branch of the circuit we have $V = E + e$. Now, we can express each of the currents in the three branches in Fig. 29.6A as follows: $I_C = C\dot{V}$, $I_{TD} = f(e)$, and we use the substitution $I_L = w$, where $L\dot{I}_L = V - I_L R$. Combining this, we have the following pair of ODEs:

$$\begin{aligned} \dot{V} &= \frac{1}{C}[I - f(e) - w] \\ \dot{w} &= \frac{1}{L}[V - Rw] \end{aligned} \tag{29.21}$$

Now the similarity to the FitzHugh–Nagumo formalism can be seen, realizing that $f(e)$ is a third-order function of V (since $V - E = e$ and E is a constant).

29.3.2.2 *Reduced Spiking Models*

From the linearization procedure we can appreciate that the neural membrane can be represented by the RCL circuit (Fig. 29.3). Of course, this representation only works for studying subthreshold behavior such as integration of inputs or resonance properties. This type of study explains the propensity of some neurons to resonate with specific input frequencies. If, as an additional simplification, the ion channel properties are completely ignored, we may even further simplify the membrane to an RC circuit; in this configuration there are no resonant properties. Both these simplifications, the RCL and RC circuits, ignore the nonlinear properties of the excitable membrane. In the so-called IF models the nonlinearity is again introduced, but in an ad hoc fashion by employing a firing threshold. When this threshold is exceeded, a spike is pasted into the signal. Although such an ad hoc firing process can be included in both RCL and RC circuits, it is commonly applied to the latter.

Historically, the RC-based IF model was introduced long before the HH formalism in 1907 by Lapicque (see also Chapter 23). Because of the leakage current in the resistor, it is also often called the leaky integrate-and-fire (LIF) neuron. An example of modeling the LIF neuron is shown in MATLAB® script pr29_3 and also in the script discussed in Chapter 23 (pr23_1). Furthermore, for details of simulating the RC circuit, see Chapter 16. Because the LIF neuron consists of electronic components, it can also be modeled in real-time using an electronic circuit: an R and C

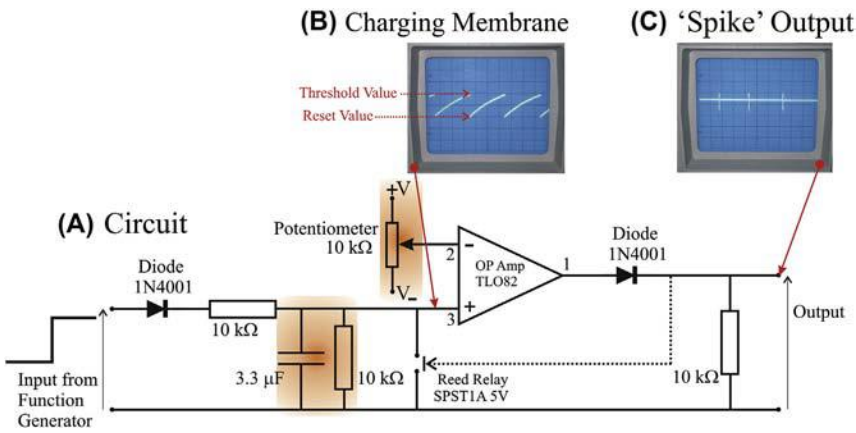


FIGURE 29.7 Panel A is an example of an electronic circuit of an IF neuron model using analog components. The membrane is modeled by the R and C components (orange), the threshold is implemented by the OP-Amp as a comparator (the threshold is the value at the potentiometer, orange). The reset function is performed by the Reed relay. The diodes and 10 kΩ resistors at the input and output are included to rectify the signals. This model is linear for the subthreshold part of the activity. Panel B shows a measurement of the subthreshold activity of the membrane including the reset at the threshold. Panel C depicts a measurement of the spike output.

to model membrane impedance, and a relay to reset the membrane potential at the threshold value (Fig. 29.7). In the example in Fig. 29.7, we employ the same values for R and C as in the filter in Chapter 15. A description of how to build this circuit is given in Appendix 29.2.

Much later a nonlinear variant of the LIF neuron was introduced where the $I-V$ relationship isn't linear as in Eq. (29.17), but quadratic. This is the so-called quadratic-integrate-and-file (QIF) model (e.g., Latham et al., 2000). You can examine this model in MATLAB® script described in Chapter 23 (pr23_2). Because of this quadratic relationship, the $I-V$ curve contains a stable and an unstable equilibrium, the spike's upstroke can be generated by the model, and only an ad hoc spike reset is required (see also Chapter 23, Fig. 23.7B).

Finally, one can combine the QIF scenario with the RCL circuit to produce a model that is capable of both resonance and spike upstroke generation. In contrast to the 1-D subthreshold behavior of the LIF and QIF models, we now have a 2-D subthreshold behavior combined with an ad hoc reset function. This model is defined as the SMC by Izhikevich and was shown to be able to produce a plethora of neuronal behavior (e.g., Izhikevich, 2007).

Table 29.1 summarizes the model dimensions and provides an overview of the spiking mechanisms of the models we evaluated in this chapter. The models vary in complexity: e.g., the HH model is four-dimensional and nonlinear, whereas the subthreshold component of the IF model is linear and one-dimensional. Of course, dimensions and nonlinear components are implicitly added when the ad hoc spike generator is included in the IF model, but many studies using this model will focus on its linear subthreshold behavior.

APPENDIX 29.1

In order to linearize the nonlinear equations we rewrite Eqs. (29.1)–(29.4) as $C\frac{dV}{dt} = f(V, m, h, n, I_{\text{Inject}})$ and $\frac{dx}{dt} = f(V, x)$, with x representing an activation/inactivation variable m, h, or n. Then we linearize about an equilibrium potential V^* (either resting potential with $I_{\text{inject}} = 0$, or a holding potential implying $I_{\text{inject}} \neq 0$). Small variations in the parameters about this equilibrium are $\delta V, \delta m, \delta h, \delta n, \delta I$. Using just the compact form of Eqs. (29.1)–(29.4), we can evaluate them for small changes: i.e., $\frac{d\delta V}{dt}$ and $\frac{d\delta x}{dt}$. For example, the effect due to perturbation δV is:

$$\delta f = C\frac{d\delta V}{dt} = \left(\frac{\partial f(V, m, h, n, I_{\text{Inject}})}{\partial V}\right)_* \delta V.$$

The expression in parentheses is evaluated at equilibrium *. Similarly, a perturbation δx yields an effect:

$$\delta f = C\frac{d\delta V}{dt} = \left(\frac{\partial f(V, m, h, n, I_{\text{Inject}})}{\partial x}\right)_* \delta x.$$

Combining the perturbation effects we obtain a linearized version of Eq. (29.1):

$$\begin{aligned} C\frac{d\delta V}{dt} = &\left(\frac{\partial f}{\partial V}\right)_* \delta V + \left(\frac{\partial f}{\partial m}\right)_* \delta m + \left(\frac{\partial f}{\partial h}\right)_* \delta h \\ &+ \left(\frac{\partial f}{\partial n}\right)_* \delta n + \left(\frac{\partial f}{\partial I_{\text{inject}}}\right)_* \delta I_{\text{inject}} \end{aligned} \tag{A29.1-1}$$

Here we simplified the notation by using f instead of $f(V,m,h,n,I_{\text{Inject}})$, further, using Eq. (29.1), we find:

$$\left(\frac{\partial f}{\partial V}\right)_* = -\left(g_{\text{Leak}} + \overline{g}_{\text{Na}} m^3 h + \overline{g}_{\text{K}} n^4\right)\Big|_*, \quad \left(\frac{\partial f}{\partial m}\right)_* = -\overline{g}_{\text{Na}} 3m^2 h(V - E_{\text{Na}})\Big|_*,$$
$$\left(\frac{\partial f}{\partial h}\right)_* = -\overline{g}_{\text{Na}} m^3 (V - E_{\text{Na}})\Big|_*, \left(\frac{\partial f}{\partial n}\right)_* = -\overline{g}_{\text{K}} 4n^3 (V - E_{\text{K}})\Big|_*, \left(\frac{\partial f}{\partial I_{\text{inject}}}\right)_* = 1$$

For the gating parameters, using $\frac{dx}{dt} = \frac{x_\infty(V) - x}{\tau_x(V)} = f(V,x)$, we get the linearized expression:

$$\frac{d\delta x}{dt} = \left(\frac{\partial f(V,x)}{\partial V}\right)_* \delta V + \left(\frac{\partial f(V,x)}{\partial x}\right)_* \delta x, \quad \text{with}$$

$$\left(\frac{\partial f(V,x)}{\partial V}\right)_* = \left.\frac{\tau_x(V)\left(\frac{dx_\infty(V)}{dV}\right) - \overbrace{(x_\infty(V) - x)}^{\delta x}\left(\frac{d\tau_x(V)}{dV}\right)}{\tau_x(V)^2}\right|_* \quad \text{and}$$

$$\left(\frac{\partial f(V,x)}{\partial x}\right)_* = -\left.\frac{1}{\tau_x(V)}\right|_*$$

Note that x_∞, τ_x are functions of V only, and consequently the partial differential is replaced by an ordinary one. Putting it all together and ignoring the nonlinear cross term $\left.\frac{\left(\frac{d\tau_x(V)}{dV}\right)}{\tau_x(V)^2}\right|_* \delta x \delta V$, the above equation simplifies to:

$$\frac{d\delta x}{dt} = \left(\frac{1}{\tau_x(V)}\right)_* \left(\frac{dx_\infty(V)}{dV}\right)_* \delta V - \left(\frac{1}{\tau_x(V)}\right)_* \delta x \qquad \text{(A29.1-2)}$$

In the literature it is common to simplify the notation of Eqs. (A29.1-1) and (A29.1-2).

For example, in Richardson et al. (2003), Eq. (A29.1-2) is simplified by multiplying both sides of the expression by $\left(\tau_x(V) \Big/ \frac{dx_\infty(V)}{dV}\right)_*$ and substituting $\nu = \delta V$ and $w_x = \delta x \Big/ \left(\frac{dx_\infty(V)}{dV}\right)_*$. This results in:

$$\tau_x(V)\frac{dw_x}{dt} = \nu - w_x.$$

In the same manner, the notation of Eq. (A29.1-1) can be simplified. The first term of the right-hand side of Eq. (A29.1-1) becomes:

$$\left(\frac{\partial f}{\partial V}\right)_* \delta V = \underbrace{-\left(g_{\text{Leak}} + \overline{g}_{\text{Na}} m^3 h + \overline{g}_{\text{K}} n^4\right)\big|_*}_{-g_M} \underbrace{\delta V}_{v} = -g_M v$$

Notation for each of the conductivity terms with m, h, or n can also be simplified. For example, the expression for n becomes:

$$\left(\frac{\partial f}{\partial n}\right)_* \delta n = -\overline{g}_{\text{K}} 4n^3 (V - E_{\text{K}})\big|_* \delta n = \underbrace{\left(-\overline{g}_{\text{K}} 4n^3 (V - E_{\text{K}})\big|_*\right)\left(\frac{dn_\infty(V)}{dV}\right)_*}_{-g_{\text{K}}} \times \left[\underbrace{\delta n \Big/ \left(\frac{dn_\infty(V)}{dV}\right)_*}_{w_n}\right] = -g_{\text{K}} w_n$$

In the above, we multiplied by $\left(\frac{dn_\infty(V)}{dV}\right)_* \Big/ \left(\frac{dn_\infty(V)}{dV}\right)_*$ to allow the change of the variable from δn to w_n. Similarly, we can simplify the expressions for m and h in $-g_{\text{Na}} w_m$ and $-g_{\text{Na}} w_h$. Consequently, Eq. (A29.1-1) can be simplified to (e.g., Richardson et al., 2003):

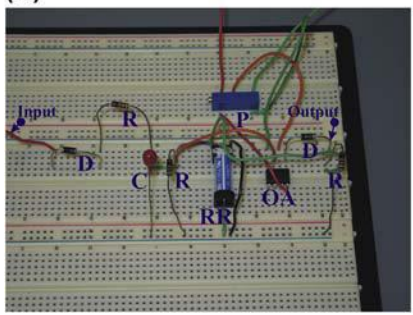
$$C\frac{dv}{dt} = -g_M v - \sum_{x=1}^{N} g_x w_x + \underbrace{\delta I_{\text{inject}}}_{I_{\text{app}}}$$

Here x is Na, K, and all other ion species included in the model.

APPENDIX 29.2 BUILDING THE INTEGRATE-AND-FIRE CIRCUIT

An example of the circuit we discussed in Fig. 29.7 is shown in Fig. A29.2-1. It is not uncommon that students without experience building circuitry can become somewhat overwhelmed by the IF model. The best approach is **not** to set up and test the complete circuit at once. If there is one mistake anywhere in the connections, it will be hard to find.

(A)



(B)

Components
D-Diode 1N4001 (276-1101)
R-Resistor 10 kΩ
C-Capacitor 3.3 μF
RR-Reed Relay SPST1A 5V (275-232)
P-Potentiometer 10 kΩ (271-0304)
OA-OP Amp TLO82 (276-1715)
Equipment
± 5V Power Supply
Function Generator
Dual Channel Oscilloscope

FIGURE A29.2-1 An example of the integrate-and-fire circuit in Fig. 29.7 built on a breadboard is shown in Panel A. The abbreviations of the components in this circuit are listed in panel B.

The preferred method is to create the circuitry in stages and test different parts separately. Test each part and if it doesn't perform as expected, review that part and correct it where needed.

The following is a recommended sequence to follow when building the circuit.

1. First build the part that serves as input to the op-amp plus input: the diode, the 2 R's, and the C. Then feed it an input from the function generator and determine if this is what you expect, i.e., a low-pass filter system that charges when input is provided.
2. Second, you connect the 10-kΩ potentiometer to the $+V$ and $-V$ and turn the screw until the variable potential (middle connector) is slightly positive but close to zero.
3. Third build the op-amp stage with the potentiometer hooked up to the inverting input and the diode connected to the output, but without connecting your RC circuit to the + input. Check if the op-amp can work as a trigger by feeding it an input from the function generator to the + input.
4. If this all works, hook up the resistor and relay to the diode at the amp's output and check if the relay is working (you can hear it click). Note that you don't need to hook up all of the relay's connectors for this test, only the part that steers the coil for switching.
5. Finally you can hook up the RC input and reed relay to the + input of the op-amp. For the best result, you may want to move the threshold to a slightly more positive value.

EXERCISES

29.1 Use MATLAB® to compute and plot the input–output relationship (input is defined as input current and output as the action potential frequency) of the following model neurons
   a. The Hodgkin Huxley model
   b. An IF model neuron
   c. What do you conclude comparing both relationships? [Explore the parameters and make sure you plot each relationship over a meaningful interval that shows the dynamics of the neuronal models. Although you may have to modify some of the parameters, you may use pr29_1 and pr29_3 as the basis to do this exercise.]

29.2 Examine the dynamics governed by Eq. (29.20).
   a. Use MATLAB® to simulate the equations by using the parameters that FitzHugh (1961) used in his original paper: $a = 0.7$, $b = 0.8$, $c = 3$, and $z = 0$ and $z = 0.4$.
   b. By linearizing about the fixed point, show that the scenario for $z = 0$, corresponds to a stable spiral.
   (Hint: Use the technique described in Appendix 23.1 and do realize that the fixed point is NOT at the origin in this version of the FitzHugh–Nagumo model.)
   c. What type of bifurcation do you observe between $z = 0$ and $z = 0.4$?
   d. Determine the input–output relationship of the FitzHugh–Nagumo model following the procedure in Exercise 29.1.

29.3 Use Appendix 29.2 to build the IF Model and determine the Input–output relationship of the model by providing step input with different amplitudes and measuring the output firing rate of the IF model. Adapt the MATLAB® script for Exercise 29.1b to simulate the findings for this circuit

References

Bower, J.M., Beeman, D., 1998. The Book of Genesis. Springer-Verlag, New York.

Brunel, N., Hakim, V., Richardson, M.J.E., 2003. Firing rate resonance in a generalized integrate-and-fire neuron with subthreshold resonance. Phys. Rev. E 67, 051916.

Carnevale, N.T., Hines, M.L., 2006. The Neuron Book. Cambridge University Press, Cambridge, UK.

Curto, C., Sakata, S., Marguet, S., Itskov, V., Harris, K.D., 2009. A simple model of cortical dynamics explains variability and state dependence of sensory responses in Urethane-anesthetized auditory cortex. J. Neurosci. 29, 10600–10612.

De Schutter, E., Bower, J.M., 1994. An active membrane model of the cerebellar Purkinje cell: II. Simulation of synaptic responses. J. Neurophysiol. 71, 401–419.
Dwyer, J., Lee, H., Martell, A., Stevens, R., Hereld, M., van Drongelen, W., 2010. Oscillation in a network model of neocortex. Neurocomputing 73, 1051–1056.
Dwyer, J., Lee, H., Martell, A., van Drongelen, W., 2012. Resonance in neocortical neurons and networks. Eur. J. Neurosci. 36, 3698–3708.
Erchova, I., Kreck, G., Heinemann, U., Herz, A.V.M., 2004. Dynamics of rat entorhinal cortex layer II and III cells: characteristics of membrane potential resonance at rest predict oscillation properties near threshold. J. Physiol. 560, 89–110.
FitzHugh, R., 1961. Impulses and physiological states in theoretical models of nerve membrane. Biophys. J. 1, 445–466.
Hodgkin, A.L., Huxley, A.F., 1952. A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. 117, 500–544.
Hutcheon, B., Yarom, Y., 2000. Resonance, oscillation and the intrinsic frequency preferences of neurons. Trends Neurosci. 23, 216–222.
Izhikevich, E.M., 2007. Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting. MIT Press, Cambridge, MA.
Jirsa, V.K., Haken, H., 1997. A derivation of a macroscopic field theory of the brain from the quasi-microscopic neural dynamics. Phys. D 99, 503–526.
Koch, C., 1999. Biophysics of Computation: Information Processing in Single Neurons. Oxford University Press, New York.
Lapicque, L., 1907. Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarization. J. Physiol. Pathol. Gen. 9, 620–635.
Latham, P.E., Richmond, B.J., Nelson, P.G., Nirenberg, S., 2000. Intrinsic dynamics in neuronal networks. I. Theory. J. Neurophysiol. 83, 808–827.
Lytton, W.W., Sejnowski, T.J., 1991. Simulations of cortical pyramidal neurons synchronized by inhibitory interneurons. J. Neurophysiol. 66, 1059–1079.
Markram, H., 2006. The blue brain project. Nat. Rev. Neurosci. 7, 153–160.
Martell, A.L., Ramirez, J.-M., Lasky, R.E., Dwyer, J.E., Kohrman, M., van Drongelen, W., 2012. The role of voltage-dependence of the NMDA receptor in cellular and network oscillation. Eur. J. Neurosci. 36, 2121–2136.
Mauro, A., Conti, F., Dodge, F., Schor, R., 1970. Subthreshold behavior and phenomenological impedance of the squid giant axon. J. Gen. Physiol. 55, 497–523.
McCormick, D.A., Huguenard, J.R., 1992. A model of the electrophysiological properties of thalamocortical relay neurons. J. Neurophysiol. 68, 1384–1400.
Morris, C., Lecar, H., 1981. Voltage oscillations in the barnacle giant muscle fiber. Biophys. J. 35, 193–213.
Nagumo, J., Arimoto, S., Yoshizawa, S., 1962. An active pulse transmission line simulating nerve axon. Proc. IRE 50, 2061–2070.
Richardson, M.J.E., Brunel, N., Hakim, V., 2003. From subthreshold to firing-rate resonance. J. Neurophysiol. 89, 2538–2554.
Traub, R.D., Jefferys, J.G., Miles, R., Whittington, M.A., Tóth, K., 1994. A branching dendritic model of a rodent CA3 pyramidal neurone. J. Physiol. 481, 79–95.
Traub, R.D., Llinas, R., 1979. Hippocampal pyramidal cells: significance of dendritic ionic conductances for neuronal function and epileptogenesis. J. Neurophysiol. 42, 476–496.
Traub, R.D., Miles, R., 1991. Neuronal Networks of the Hippocampus, vol. 777. Cambridge University Press, Cambridge.
Traub, R.D., Contreras, D., Cunningham, M.O., Murray, H., LeBeau, F.E., Roopun, A., Bibbig, A., Wilent, W.B., Higley, M.J., Whittington, M.A., 2005. Single-column thalamocortical network model exhibiting gamma oscillations, sleep spindles, and epileptogenic bursts. J. Neurophysiol. 93, 2194–2232.
van der Pol, B., van der Mark, J., 1927. Frequency demultiplication. Nature 120, 363–364.

van Drongelen, W., Lee, H.C., Koch, H., Hereld, M., Elsen, F., Chen, Z., Stevens, R.L., 2005. Emergent epileptiform activity in neural networks with weak excitatory synapses. IEEE Trans. Neur. Sys. Rehab. 13, 236–241.

van Drongelen, W., Koch, H., Elsen, F.P., Lee, H.C., Mrejeru, A., Doren, E., Marcuccilli, C.J., Hereld, M., Stevens, R.L., Ramirez, J.M., 2006. The role of persistent sodium current in bursting activity of mouse neocortical networks in vitro. J. Neurophysiol. 96, 2564–2577.

CHAPTER

30

Modeling Neural Systems: Network Models*

30.1 INTRODUCTION

When modeling neuronal networks, one can start from the network nodes, i.e., the neurons, or make direct assumptions at a higher organization level, i.e., the population activity. In the first case, the bottom-up approach, one must decide how to model the nodes. Simple on/off switches are the most abstract models of neurons in networks. The underlying hypothesis of this approach is that, for some of the aggregate behavior, the network properties themselves are of principal interest while the details of the cellular properties (beyond having active and inactive states) are irrelevant (McCulloch and Pitts, 1943; Little, 1974; Hopfield, 1982; Benayoun et al., 2010a; Wallace et al., 2011). Many network models that use the bottom-up approach include more complex representations for the nerve cells and their connections. These representations may vary from a simple integrate-and-fire (IF) model to compartmental models including biophysically realistic channels (e.g., van Vreeswijk et al., 1994; van Vreeswijk, 1996; Olmi et al., 2010; Tattini et al., 2012; Izhikevich and Edelman, 2008; Traub and Miles, 1991; Traub et al., 2005; Lytton and Sejnowski, 1991; De Schutter and Bower, 1994; van Drongelen et al., 2005, 2006, 2007b; Markam, 2006). In spite of the many parameters in these models, an overwhelming amount of detail is still missing. As a consequence, due to nonlinearities in neuronal function (e.g., membrane conductance dynamics, the coupling function between neurons), small inaccuracies in the model parameters may lead

*Parts of this chapter are based on previously published work by the author (van Drongelen, 2013).

to large prediction errors of the model. In spite of this potential shortcoming, some models perform surprisingly well in predicting or mimicking experimental outcomes (e.g., McCormick and Huguenard, 1992). Artificial networks can even be used to replicate human behavior and perform functions such as image recognition or driving a vehicle (e.g., Eliasmith et al., 2012; LeCun et al., 2015).

A different family of network models describes aggregate behavior of neuronal populations. Although the components of these models are derived from or inspired by the neuronal function, they do not explicitly consist of a network of individual agents. This approach was pioneered by Wilson and Cowan (1972, 1973). Their result is a nonlinear, two-dimensional model describing the dynamics of excitatory and inhibitory activity. The expressions for the population activity are based on assumptions about cellular statistics such as the distribution of neuronal excitability. The initial 1972 Wilson–Cowan model describes the time-dependent dynamics of the neural populations; whereas their 1973 paper adds a spatial component to this model. In the same decade, Nunez proposed models of cortical networks to explain and describe the electroencephalogram (EEG) representing compound activity of millions of nerve cells (Nunez, 1974, 1995; Nunez and Srinivasan, 2006a,b). This work focused on the synaptic activities in the cortical pyramidal cell population since this population is considered to be the major contributor to the mammalian EEG. To describe cortical activity, Nunez included inhibitory and excitatory populations and used linearized relationships to describe their interactions. Since the early 1970s, many models have been directly or indirectly inspired by the Wilson and Cowan approach and the models by Nunez. Examples are the models by Lopes da Silva et al. (1974) and van Rotterdam et al. (1982), the Jansen neural mass model (Jansen and Rit, 1995; Grimbert and Faugeras, 2006), and Freeman's models of the olfactory system (Freeman, 1987; Chang et al., 1998). More recent examples are the models by Wendling and coworkers (e.g., Wendling and Chauvel, 2008), Liley and Bojak (2005), van Albada and Robinson (2009), and Hindriks and van Putten (2012). A model developed by Jirsa and Haken (1997) produces a field equation for the network activity generated by a cortical surface and can be considered a synthesis of the different population model approaches. In part due to increased access to computer technology facilitating the analysis of nonlinear dynamics, the original Wilson–Cowan approach and its derived models have been "rediscovered" (Destexhe and Sejnowski, 2009). Due to the deterministic character of these models, they fail to model fluctuations. Stochastic/statistical approaches are capable of modeling higher-order moments that characterize the neural activity, such as synaptic input

and spike trains (e.g., Rieke et al., 1997; Rolls and Deco, 2010; Buice and Cowan, 2009).

Understanding the function of neuronal networks is a frontier in current neuroscience. Our lack of fundamental understanding of network function in the nervous system prevents us from linking neuronal activity to higher-level physiologic and pathologic behavior. Although compound activity of huge networks can be captured by the EEG or possibly by functional magnetic resonance imaging (fMRI), current experimental techniques in neuroscience cannot record the individual activity of cells forming large networks ($>10^6$ neurons) with sufficient resolution in the time domain (<1 ms). Due to the lack of such techniques, there is a real need for theoretical and computational models in neuroscience because they can provide a framework for the integration of experimental data. In addition, models provide an efficient tool for generating and testing series of hypotheses and guide experimental effort; to cite Abbott (2008), "A skillful theoretician can formulate, explore, and often reject models at a pace that no experimental program can match." Before reading the following material, depending on your background in mathematical modeling, I recommend you review previous chapters on modeling linear and nonlinear dynamics (especially Chapters 9–13 and 17) and single neuron models (Chapters 23 and 29).

30.2 NETWORKS OF INDIVIDUAL CELL MODELS

Network models containing concrete nodes come in a wide variety, ranging from nodes represented by an on/off switch to a detailed multicompartmental cell model with multiple ion channels. Similarly, the connections between the nodes can be modeled by relatively simple rules or they can be simulated by biophysically realistic synaptic ion channels. In general, the purpose of these models is to provide simulation results that can be compared to a result from mathematical analysis and/or an experimental measurement.

30.2.1 Neurons Represented by On/Off Switches

The first simple network model using coupled switches was described by McCulloch and Pitts (1943). Their approach was to represent network elements as gates in a logical circuit (e.g., AND, OR, etc.). Others have used a similar approach and also compared the network on/off node with the spins in a spin-glass lattice (e.g., Little, 1974). These so-called Ising spins can be up or down, indicated by $+1$ and -1, respectively

(Ising, 1925). Each spin S_i tends to align with its neighbors. However, at high temperatures this tendency to align vanishes because the probability of alignment decreases to ½. We model the cortical surface by a two-dimensional lattice of Ising spins. The overall state of this lattice can be characterized by its magnetization m; the average of the individual spins: $m = \frac{1}{N}\sum_{i=1}^{N} S_i$. Below a certain critical temperature T_c, the lattice will retain global magnetization if an external magnetic field h is applied. Above the critical temperature (also called the Curie temperature), thermal fluctuations will cause demagnetization. Although this is a mathematical framework in statistical mechanics, it has been used to represent neuronal networks where the spins are the neuronal nodes in the network; in these models the up and down states are often presented by 1 (active) and 0 (inactive), respectively. These networks with nodes S_i are typically simulated in discrete time t and follow the following update rules.

1. At each time step, the state of S_i is determined by the state of its set (S_k) of its connected neighbors, reflected by function $H_i = \sum_{j \in S_k} w_{j,i} S_j$, with $w_{j,i}$ representing a weighting function from node j to node i (in many models this weight is set to one).
2. The update of S_i now takes place in a probabilistic fashion; the probability P that S_i will be in the up state in the next time step $t+1$ is: $P[S_i(t+1) = 1] = \frac{1}{1+e^{-H_i/T}}$ the variable T is the temperature; it can be seen that $P = ½$ for $T \to \infty$ and that the update is a deterministic process ($P = 1$) at $T = 0$.

This update process can be considered as a stochastic variant of one of the rules described by McCulloch and Pitts (1943), where a neuron's output y is updated by the state of its input from neurons x_i in a deterministic fashion. A mean field approach for the analysis of the Ising spin model is described in Section 30.3.1.

A model similar to the Ising spin lattice was described by Hopfield (1982). He also considered switches interconnected in a network in which the strength of the connections between any neuron pair i and j is determined by the synaptic weight w_{ij} of their connection. No unit has a connection with itself, $w_{ii} = 0$. Unrealistically, he also assumed that the weight matrix is symmetric, $w_{ij} = w_{ji}$. The principal difference between Hopfield's approach and those described above is that the updates occur both randomly and asynchronously, i.e., for every time step a node is randomly selected and then an update rule is applied for that node only.

Once a node is selected, the update rule is similar to the deterministic one employed by McCulloch and Pitts (1943), i.e., the selected neuron is active if the sum of activities of its connected neurons exceeds a threshold.

In contrast to the probabilistic update of the model nodes, one can also construct node-based networks in which the update of nodes based on their surrounding follows a deterministic rule. In a simulation, we update each network node at subsequent time steps following such an update rule. For example, if we have a lattice of nodes where each node has eight neighbors, we could make the node active if the majority of the node itself plus its neighbors (i.e., nine nodes) are active: that is, if five or more nodes are active the node becomes active at the next time step, else it becomes inactive. This deterministic rule is often indicated as the "majority rules" update. This example of discrete time dynamics governing a network is often called a **cellular automaton**. The cellular automaton consists of cells arranged in a grid. The cells can be in one out of a number of possible states. A simple scenario is to implement a "majority rules" network and allow just two possible states, i.e., we simplify the dynamics so that nodes can only be inactive (0) or active (1) while the update of each node is governed by Boolean logic. At each time step, the state of the cell is updated according to a rule determined by the state of adjacent cells, i.e., the neighborhood. Although this setup seems rather simple and may look like some useless game, the cellular automaton plays an important role in modeling in science and engineering (Wolfram, 1986).

An example of a cellular automaton simulation is implemented in MATLAB® pr.30_1.m. It is quite interesting to see that this simulation, although it is started from a randomly picked initial state, converges very fast to a stable pattern. A specific stable pattern that emerges depends on the initial condition. In our simulation such a stable network pattern can usually be observed well within 35 time steps. In this example we examine a network arranged in a 2-D lattice (Fig. 30.1). The update rule is "the majority rules," which means that a node becomes active (1) if the majority of the node itself plus its neighbors are active and the node will become inactive (0) otherwise. Since, in this example, we don't want to deal with the effect of boundaries, we connect them by arranging the network in a torus (Fig. 30.1C). Thus, although the result of running pr30_1 is depicted in a square (Fig. 30.1D), there are no boundaries since the nodes depicted on the square's boundaries are connected. In this example, each simulation starts from a different randomly picked initial condition. Although the end result is different each time, it is remarkable how fast (usually in about 10 steps) the system is attracted to a stable state.

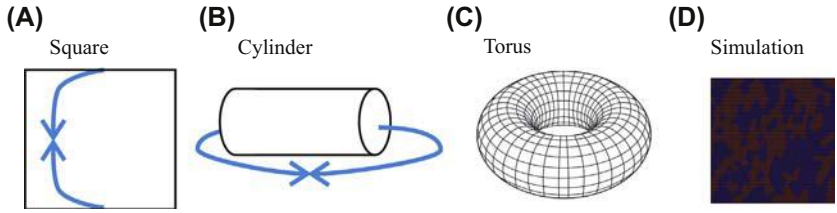


FIGURE 30.1 Example of a 2-D network of cellular automata. (A) Nodes are placed in a *square*. To avoid boundary effects, boundaries are connected (*blue arrows*). If the top and bottom nodes are connected, we obtain a cylinder (shown in B), and when the nodes of the right and left boundaries are connected we obtain a torus shown in (C). An example simulation showing the final state of a 100×100 node network governed by "the majority rules" is shown in (D). The simulation result was produced by MATLAB® script pr30_1.

30.2.2 Networks of Reduced Spiking Models

As described in Section 29.3.2.2, there is a variety of simplified models ranging from simple IF to a more complete simple model of choice (SMC) representations (Chapter 29, Table 29.1).The IF-model is especially frequently used for network simulations. One advantage of this model is that it is simply linear in between the spikes. By employing this approach, one can not only simulate the network but it is also mathematically tractable. Using the simplification in Fig. 29.3 and a further simplification by neglecting induction effects, we can simplify Eq. (29.17): $C\dot{V} = -V/R + I_{\text{inject+synapse}}$. By simplifying this notation, one can describe the network dynamics by membrane potential v_i, the activity level of each neuron, by

$$\dot{v}_i(t) = A - v_i(t) + E_i(t), \tag{30.1}$$

(e.g., van Vreeswijk et al., 1994; van Vreeswijk, 1996). The injected current is separated into A and $E(t)$. Parameter A represents a constant spontaneous drive and E is the time-dependent input. The presynaptic spike train can be represented by a train of unit impulses $\sum_n \delta(t - t_n)$, (Chapter 20). Each incoming impulse generates a postsynaptic signal that can be modeled by a so-called alpha function, i.e., for an impulse arriving at time t_n, its postsynaptic effect is given by $\sum_n \theta[t - t_n]$ $(t - t_n)\alpha^2 e^{-\alpha(t-t_n)}$. The Heaviside function θ is included because the alpha function doesn't have a physical meaning for negative arguments, $(t - t_n) < 0$. Combining these considerations and ignoring nonlinear effects between synaptic inputs, one can conclude that input E is the

weighted sum of all alpha functions resulting from the incoming spike train (e.g., Olmi et al., 2010; Tattini et al., 2012):

$$E_i(t) = \frac{1}{k_i} \sum_n W_{j,i}\theta[t - t_n](t - t_n)\alpha^2 e^{-\alpha(t-t_n)} \tag{30.2a}$$

Here we determine the input by using the synaptic weight matrix W, in which each entry is the connectivity strength of neuron j to neuron i. Scaling factor k_i represents the number of presynaptic inputs of neuron i. We now determine the Laplace transform of Eq. (30.2a) and rearrange terms $\left(s^2 + 2\alpha s + \alpha^2\right)E_i(t) = \frac{\alpha^2}{k_i} \sum_n W_{j,i}\, e^{-t_n s}$, which can also be inverse transformed into the time domain in the form of a differential equation:

$$\ddot{E}_i(t) + 2\alpha\dot{E}_i(t) + \alpha^2 E_i(t) = \frac{\alpha^2}{k_i} \sum_n W_{j,i}\, \delta(t - t_n) \tag{30.2b}$$

Since the first-order differential equation (Eq. 30.1) and the second-order one (Eq. 30.2b) are coupled, via input E, they create a three-dimensional system. If we consider the nonlinear spiking component, the system is nonlinear and capable of exhibiting limit cycle and chaotic behavior. To study the activity in the model, Olmi et al. (2010) rewrite the system as three first-order differential equations and integrate between pulses t_n and t_{n+1}. This creates an event-driven map that can be further investigated. Tattini et al. (2012) examine the propensity for chaotic behavior by evaluating the attractors characterizing the network activities and they show that both connectivity and network size are critical properties—unsurprisingly, the level of chaos, reflected by the maximum Lyapunov exponent, decreases with network size, although less so for sparsely connected ones.

In the above model, the nodes in the network are coupled via action potentials that are represented by a series of unit impulses: $\sum_n \delta(t - t_n)$. For this reason this type of network is often referred to as a **pulse-coupled-network**. Another frequently employed model is based on the 1-D phase model of neuronal activity we described in Chapter 23 (Eq. 23.5). Each network node is characterized by its phase φ_n. Now we assume that the interaction between nodes n and m is determined by their phase difference $\varphi_m - \varphi_n$, and a coupling constant K. These networks are therefore often referred to as **phase-coupled networks**. To illustrate this, we discuss the application of the so-called Kuramoto model in neuroscience (e.g., Breakspear et al., 2010). In a network where

node n is coupled to all N neighbors (all-to-all connectivity), we get the following formalism for the dynamics:

$$\dot{\varphi}_n = \omega_n + \frac{K}{N}\sum_{m=1}^{N}\sin(\varphi_m - \varphi_n) \tag{30.3a}$$

In this example, the coupling K is scaled by the network size N. The network may be heterogeneous because the oscillators are allowed different intrinsic frequencies ω_n. The average of the phase oscillators in the network is often used as the **order parameter**. For convenience, the oscillators are represented as $\exp(j\varphi_n)$, i.e., its phase represented on the unit circle in the Argand plane. The so-called order parameter z is the average over all oscillators:

$$z = r\exp(j\phi) = \frac{1}{N}\sum_{m=1}^{N}\exp(j\varphi_m)$$

Note that the magnitude of z (r) will be determined by the level of synchrony across the oscillators, and represents the coherence, while the phase (ϕ) of z (r) represents phase coherence (Chapter 13). Interestingly, we can use the order parameters to formulate the mean field effect on oscillator n. If we multiply z by $\exp(-j\varphi_n)$, we get:

$$r\exp[j(\phi - \varphi_n)] = \frac{1}{N}\sum_{m=1}^{N}\exp[j(\varphi_m - \varphi_n)]$$

The imaginary part of this expression (using Euler's formula) is:

$$r\sin(\phi - \varphi_n) = \frac{1}{N}\sum_{m=1}^{N}\sin(\varphi_m - \varphi_n)$$

This can be used to substitute for the average expression in Eq. (30.3a), giving:

$$\dot{\varphi}_n = \omega_n + Kr\sin(\phi - \varphi_n) \tag{30.3b}$$

Rather than the expression in Eq. (30.3a), where the effect of the network was obtained by taking into account all individual oscillators, Eq. (30.3b) shows the effect of all oscillators connected to oscillator n as the mean effect of the whole network (field), i.e., r and ϕ. As we will discuss in Section 30.3, such a mean field approach can simplify analysis. A simulation of this model with 10 coupled oscillators is provided in

MATLAB® script pr30_2. This script demonstrates the capability of the oscillators to synchronize.

Other simplified, albeit slightly more complicated, cell models have also been employed to investigate network activity. For example, Izhikevich and Edelman (2008) used the SMC model to create a large network while using connectivity based on the tracts determined by a diffusion tensor image of the brain.

30.2.3 Networks of Ion Channel Models Based on Biophysical Considerations

Using the Hodgkin and Huxley (HH) formalism (Hodgkin and Huxley, 1952) (Chapter 29), combined with approximation of the cell morphology by segments, one can create realistic representations of neurons. Addition of ligand-sensitive, synaptic channels allows one to create networks from these model neurons. Eq. (30.1) is then adapted to include currents associated with other voltage-sensitive channels, the synaptic currents, and intercompartmental currents. This procedure is explained in Chapter 29, see also Section 29.3.1 and Eq. (29.19) repeated here for convenience:

$$C\frac{dV}{dt} = -I_{\text{Leak}} - I_{V-\text{Sensitive}} - I_{\text{Synapse}} + I_{\text{Intercompartmental}} + I_{\text{Inject}}$$

Interestingly, by applying Kirchhoff's first law, it is apparent that intracellular intercompartmental currents must be equal in size and opposite in direction to the extracellular current between the compartments (e.g., Fig. 29.5), thereby providing a tool to model extracellular measurements. The dynamics of the voltage-sensitive channels is governed by expressions similar to Eqs. (29.2)−(29.4). The intercompartmental current, following Ohm's law, is determined by the potential difference between the isopotential compartments divided by the axial resistance between them. Traub and coworkers were amongst the first to develop this approach (e.g., Traub and Miles, 1991; Traub et al., 2005), and the blue brain project (Markam, 2006) is a famous recent attempt to determine properties that govern network activity at mesoscopic levels. The advantage of the approach is that the neurons in the network are modeled with realistic ion channels, which allows a direct comparison with experimental data. However, well-known problems with the approach are associated with the complexity and large number of parameters, many of which are not experimentally determined yet. This type of model is ill-posed and the complexity of the large parameter space can make interpretation of the results very

difficult. A combined approach with associated detailed and global models can be very useful to understand the principles underpinning the simulation results of the detailed models (van Drongelen et al., 2005, 2006, 2007b; Visser et al., 2010, 2012).

A detailed model of neocortex outlined in Fig. 30.2A shows the excitatory pyramidal cell populations and the inhibitory basket and chandelier cells. This network model generates an EEG signal based on the weighted sum of the neuronal currents. Surprisingly, during reduced excitatory coupling, the model starts to oscillate (top trace in Fig. 30.2A and B) (van Drongelen et al., 2005). The mechanism generating these oscillations is difficult to grasp when studying the individual rasters of the superficial and deep pyramidal neurons (gray panels in Fig. 30.2B). However, when lumping each of the superficial and deep pyramidal activities (Fig. 30.2B and C), it can be seen that the simulated EEG oscillation is associated with alternating oscillations in the two populations. The mechanism of the EEG oscillation can be much better understood if one considers population activity patterns while ignoring the details of the individual neural activity. If one considers individual neuron's activity as depicted in the rasters in Fig. 30.2B, one might not

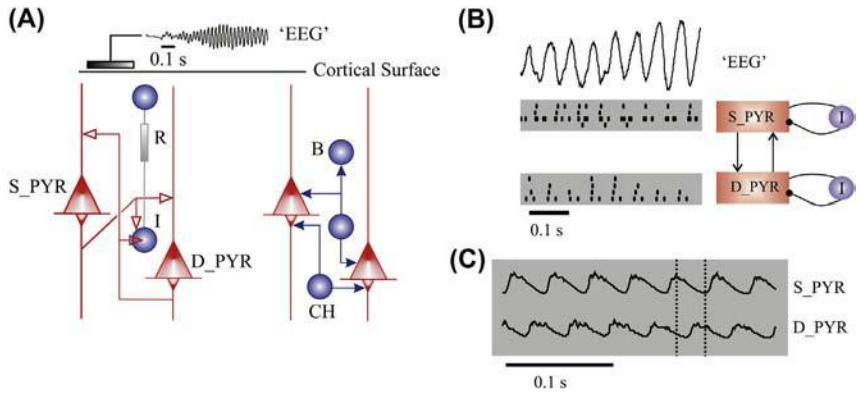


FIGURE 30.2 Model of neocortex. The detailed model (after van Drongelen et al., 2006) is outlined in diagram (A). The excitatory populations consist of superficial (S_PYR) and deep (D_PYR) pyramidal cells and the inhibitory neuron type (I) consists of populations of basket cells (B) and chandelier cells (CH). The weighted sum of all soma currents is used as the "EEG" signal (top, panel A). The excitatory connections of the microcircuitry are shown in the left part, and the inhibitory synapses in the right part of panel (A). Panel (B) depicts a detail of the EEG during seizure-like oscillation in the 'EEG' and the associated raster plots of the S_PYR and D_PYR populations. The simplified network version, after Visser et al. (2010), where only excitatory pyramidal cells and inhibitory neuronal populations are considered, is shown in the right part of panel (B). (C) The average membrane potentials of both pyramidal cell populations show that the activities of the deep S_PYR and D_PYR populations alternate during seizure-like oscillation.

see the relationship between neuron and network activity since only few neurons are active during each cycle; i.e., most neurons show cycle skipping (Wallace et al., 2011). Not only the oscillatory activity pattern itself but also the sudden transition between nonoscillatory and oscillatory network behavior can be understood from bifurcation analysis of a global model of two coupled excitatory–inhibitory populations as depicted in Fig. 30.2B (Visser et al., 2010, 2012).

30.3 MEAN FIELD NETWORK MODELS

The network models discussed in this section do not explicitly simulate each network node, but start directly by modeling populations of nerve cells. Some of the models explicitly use the mean field approach commonly employed in statistical mechanics, others create neural populations with functionality inspired by the single neuron function. The final result of these approaches can be fairly similar since they both usually contain excitatory and inhibitory components (the Ising spin model that represents excitatory neurons only is an exception). As I will point out repeatedly, this results in a strong similarity between the equations that result from the various network models and even between these and the equations describing single neurons.

30.3.1 The Ising Spin Model

Two- or three-dimensional models of the Ising spin lattice, where the spins represent the neurons, are not easy to solve. A mean field approach is the simplest approximation to describe such a system. As described in Section 30.2.1, each node in the lattice experiences the sum of the magnetic forces created by the other nodes and, if present, an external magnetic field. In the mean field approach we replace the internal field generated by the spins by its average value, similar to the procedure we followed for the Kuramoto model in Eq. (30.3). It is beyond the scope of this chapter to derive the mean field relationship, which can be found in many physics textbooks (e.g., Chapter 7 in McComb, 2004). Using the mean field expression, we can describe the equilibrium state of the lattice of spins—or network of model neurons—by the expression

$$(Jnm + h) - kT \tanh^{-1}(m) = 0, \tag{30.4}$$

With k—Boltzmann constant (in the following k is set to unity), T—temperature, J—magnetic coupling strength, n—number of neighbors, m—magnetization of the lattice, and h—an externally applied magnetic field. As shown in the following, Eq. (30.4) shows sudden transitions in

behavior according to the so-called cusp catastrophe (Thom, 1975). First we can determine the fixed points for the expression graphically by finding the intersections of both terms, $T\tanh^{-1}(m)$ and $Jnm + h$, as a function of m. When depicting these functions for different T and h, it can be seen that they intersect at either three or at one point(s). In case of the intersection at three points, the middle one is an unstable fixed point, the other two are stable ones; therefore the system is bistable in one area of the $T-h$ plane, and monostable elsewhere.

The transition between bi- and monostability occurs at the bifurcation where the straight line $Jnm + h$ intersects the sigmoid $T\tanh^{-1}(m)$ tangentially; this indicates that two conditions must be satisfied at the transition in point T_c, h^*:

$$T_c \tanh^{-1}(m) = Jnm + h^*, \text{ and since they must be tangent}$$

$$\frac{d}{dm}\left[T_c \tanh^{-1}(m)\right] = \frac{d}{dm}[Jnm + h^*] \rightarrow \frac{T_c}{1 - m^2} = Jn$$

From these two expressions, one can find the expression for T_c and h^* (as a function of m). The plot of these functions in the $T-h$ plane is the so-called stability diagram (Fig. 30.2A). These lines mark the border where the system transitions from mono- (a single fixed point) to bistability (three fixed points with the middle point unstable). Another common presentation is a bifurcation diagram; here we have a codimension-2 case with two independent variables T and h. Note that for $h = 0$, $m = 0$ is always a solution (fixed point) for Eq. (30.4). If we use $Jn = 1$, we have a stable fixed point at $m = 0$ for $T > 1$ and an unstable one plus two stable fixed points for $T < 1$. This is depicted in Fig. 30.3B; we can see that we have the characteristic of a supercritical pitchfork bifurcation (Fig. 23.4). This plot can be considered as cross-sections of a 3-D plot of

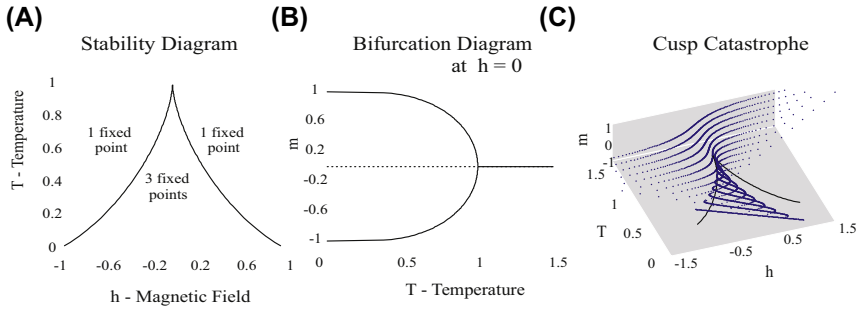


FIGURE 30.3 Properties of the mean field equation of the Ising model. (A) The stability diagram. (B) Bifurcation diagram. (C) The cusp catastrophe shows the dependence of magnetization m on both the temperature (T) and an externally applied magnetic field (h).

$m(h,T)$ shown in Fig. 30.3C; this surface depicts the so-called cusp catastrophe. The cusp catastrophe in the Ising model can be simulated with MATLAB® script pr30_3.

30.3.2 The Wilson and Cowan Model

The first model by Wilson and Cowan (1972) is exclusively temporal and describes the behavior of two neuronal populations—one excitatory and one inhibitory—embedded in neural tissue. It is much more realistic than using the Ising model that only considers the equivalent of excitation. In the Wilson–Cowan model, population properties are derived from assumed statistics of the neurons (Wilson and Cowan, 1972). Each population receives input from itself, the other population, and external sources. The input to a population affects the neurons of that population that are available (the inactive proportion of the population). The model is nonlinear because the incoming activity for the population is related to its output through a nonlinear sigmoid function S. In a second version of the model that will not be further discussed here, Wilson and Cowan (1973) extended their model to include the spatial component as well. Instead of formulating the model from underlying statistical properties, a coarse-grained version of the model can be directly derived from the diagram in Fig. 30.4. Two populations, the excitatory and inhibitory, are coupled with strengths c_1–c_4. The external population affects the excitatory and inhibitory networks with strengths P and Q. We denote excitatory activity by E and inhibitory activity by I, the excitatory population in Fig. 30.4 receives input that can be quantified as $c_1E - c_2I + P$. This input of the population is converted to an output expression via the sigmoid function $S_e\{\ldots\}$. In most models inspired by the Wilson–Cowan approach, such a sigmoid curve is employed to represent conversion from membrane potential to

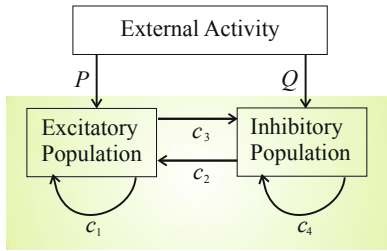


FIGURE 30.4 Diagram of the Wilson–Cowan model containing excitatory and inhibitory populations. The coupling strength between the populations is denoted with constants c_1–c_4 and external input to the populations is symbolized by P and Q.

spike rate and/or vice versa. In the Wilson and Cowan approach, the sigmoid curve represents the cumulative distribution of the probability that a neuron will fire at a given input; i.e., the distance between membrane potential and firing threshold. Further, assuming that the unavailable part of the population, i.e., the active + refractory portion is proportional (by a fraction r_e) to the activity level E then the receptive population (available portion of E) is $1 - r_eE$. In the original paper by Wilson and Cowan (1972) the authors use $k_e - r_eE$ instead, where k_e is a value slightly less than unity. The formalism for population activity must also describe spontaneous decay proportional to E, reflecting that a nonstimulated group of neurons converge to a base level of activity (zero, reflecting a low level of spontaneous activity in this model). Finally, all changes dE/dt are weighted by the time constant of the neuronal population τ_e. Putting this all together the coarse-grained equation for excitatory activity becomes:

$$\underbrace{\tau_e}_{\substack{\text{Excitatory}\\ \text{Time Constant}}} \frac{dE}{dt} = \underbrace{-E}_{\text{Decay}} + \underbrace{(k_e - r_eE)}_{\substack{\text{Available}\\ \text{Population}}} \overbrace{S_e\underbrace{\{c_1E - c_2I + P\}}_{\text{Input}}}^{\text{Sigmoid}} \tag{30.5}$$

A similar approach for the inhibitory population yields

$$\tau_i\frac{dI}{dt} = -I + (k_i - r_iI)S_i\{c_3E - c_4I + Q\} \tag{30.6}$$

In their 1972 paper, Wilson and Cowan derive these relationships from first principles followed by a coarse-graining procedure (their Eqs. 11 and 12). When looking into the two-dimensional nonlinear Wilson–Cowan model, there should be a variety of behaviors typical for 2-D nonlinear systems (Chapter 23), including fixed points, a spiral sink, and limit cycle (Fig. 30.5).

MATLAB® script pr30_4 shows the phase plane of the Wilson–Cowan equations with parameters associated with three fixed points. When comparing the phase planes in Fig. 30.5A and the result of pr30_4, note that the *E*-*I* axes are flipped.

30.3.3 Models Related to the Wilson–Cowan Approach

Several models that are used to simulate neural activity include properties that are similar to, or even directly inspired by, the Wilson–Cowan formalism. In the following sections, we present examples of this type of modeling approach.

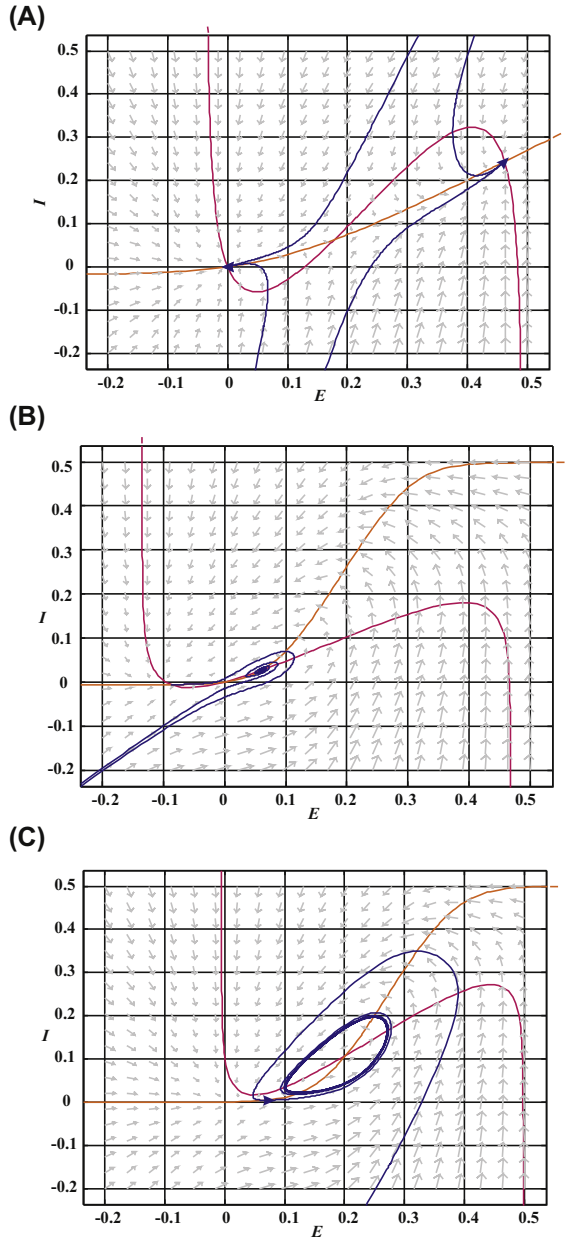


FIGURE 30.5 Phase plane representations of the Wilson–Cowan model showing fixed point, spiral sink, and limit-cycle behaviors. These plots correspond to the parameters in Figs. 4, 10, and 11 of the original 1972 paper. Note that in the top plot there is bistability: two stable fixed points, one in a low-activity (down) mode and one in a high-activity (up) state. Also note that in these examples, oscillations are obtained by flattening the E null-cline (magenta) and making the I null-cline (orange) steeper. Sample trajectories are depicted in blue. (A) Three fixed points, two of them are stable and one (in between) is unstable. The eigenvalues of these points from left to right are: -0.59, -1.13; 0.73, -1.6; -1.43, -2.88. (B) Two fixed points, one is clearly associated with a damped oscillation. The eigenvalues from left to right are: -0.34, -0.98; $-0.09 \pm 0.86j$. (C) The limit cycle with an unstable fixed point inside with eigenvalues $0.12 \pm 1.88j$.

An early example of the application of an artificial neural structure is the perceptron, first developed in the 1950s. In the context of the above framework of modeling neural activity, it is interesting to note that the artificial intelligence community, using neural nets, initially adopted the sigmoid relationship to convert input to output. More recently different nonlinear functions have proven to be more successful and this field is now indicated as deep learning rather than neural nets (LeCun et al., 2015).

In the above example of cellular automata, we simulated a network in which the connectivity strengths between the network nodes remained the same. Modification of connection strength, i.e., synaptic plasticity, is considered a critical feature of the learning process in biological neuronal networks. A famous rule governing plasticity was formulated by Donald Hebb (1949), and is often summarized by the phrase: "cells that fire together, wire together." Inspired by the biology, the modification of connection strength is also used in artificial neural-net implementations. The artificial neural networks play an important role in artificial intelligence applications. Their usefulness has been recently rediscovered (e.g., LeCun et al., 2015), and they are now employed in a variety of applications such as self-driving cars, smart phone apps, etc. The critical component of developing these neural networks is that they go through a learning process of adjusting the network connectivity. This learning process of adapting connection strength is called unsupervised if the network organizes itself, i.e., without the presence of a teacher. In contrast, a supervised learning method is obtained by observing the network output, and using its deviation from a predefined output to adapt connectivity strengths.

The early-developed artificial neural system was the so-called Perceptron. An example of a Perceptron with two inputs x_1 and x_2 and threshold θ is depicted in Fig. 30.6. In this diagram, the inputs are weighted by w_1 and w_2. If $x_1w_1 + x_2w_2$ exceeds θ, the unit fires. Note that, while the input to the Perceptron is a linear combination of its inputs, the threshold, effectively switching the unit's output on $(+1)$ or off (-1), represents a nonlinear function. Let us consider a simple example where the Perceptron performs an AND function: i.e., the Perceptron's inputs are either 0 or 1, its output is $+1$ (active, A) if the inputs are both 1, or -1 (in rest, R) otherwise. A summary of binary functions, including the AND, can be found in Table 30.1. Note that, in this case, we use -1 rather than 0 to symbolize the rest state of the Perceptron. Now, we will use our knowledge of the desired output to implement a supervised learning approach to create an AND unit. We should find A at input (1 1) and R for (0 0), (0 1), and (1 0). Our task is now to find values for weights w_1 and w_2 and threshold θ that are associated with the AND function.

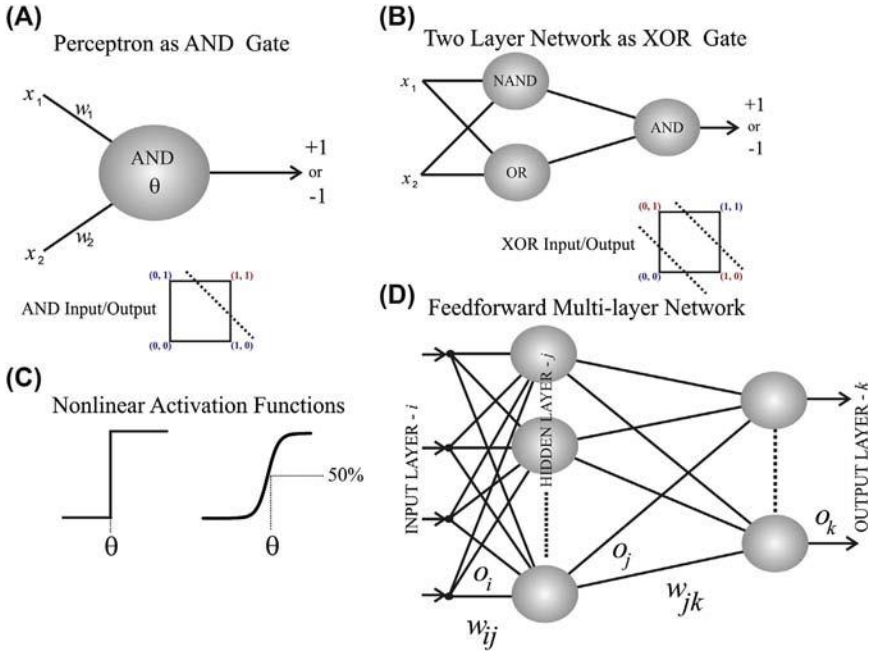


FIGURE 30.6 Neural networks and pattern recognition. (A) Diagram of a Perceptron performing an AND function. The rectangular area shows the input domain and the associated output: red corresponds to an output of 1, and blue to an output of −1. It can be seen that a single straight line separates the input domains associated with 1 and −1 outputs. (B) A single-layer network cannot perform an XOR operation, but a two-layer network can. The rectangular area shows the input domains associated with the two types of output. Note that in this case we need two straight lines to separate the domains for outputs 1 (coded red) and −1 (coded blue). (C) Although the neuron's input is a linear combination of its individual inputs, the output is governed by a nonlinear function such as a step (left) or sigmoid function (right) performing a threshold (θ) operation. (D) Feedforward multilayer networks can perform a multitude of tasks. See Appendix 30.2 for the weight adjustment procedure in the multilayer case.

TABLE 30.1 Output of Binary Functions of Inputs x_1 and x_2. Boolean Functions That Perform a Logical Operation on Logical Input(s), and Produce a Single Logical Output

x_1	x_2	AND	NAND	OR	NOR	XOR
0	0	0	1	0	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	0	1	0	0

We can simplify the learning task by treating the threshold parameter θ also as a weight. Since the input to the Perceptron is $x_1w_1 + x_2w_2 - \theta$, we can replace θ in this expression by x_3w_3 with $w_3 = -\theta$ and constant "fake" input $x_3 = 1$. Following this procedure, we can create an *extended input vector* that, in addition to the first two inputs and weights, also includes w_3 and x_3. Using this approach we can state that the AND function works correctly if the following four conditions are satisfied:

$$\begin{aligned}(0\,0\,1)\cdot(w_1\ w_2\ \ w_3)' &< 0\\ (0\,1\,1)\cdot(w_1\ w_2\ \ w_3)' &< 0,\\ (1\,0\,1)\cdot(w_1\ w_2\ \ w_3)' &< 0,\\ (1\,1\,1)\cdot(w_1\ w_2\ \ w_3)' &> 0,\end{aligned} \tag{30.7a}$$

where the dot denotes the vector in product and $'$ indicates the transpose, turning the row vector $(w_1\ w_2\ w_3)$ into a column vector. Note that the third "fake" input x_3 is indeed always 1.

If we now change the signs of the "1" values in the top three conditions, we get:

$$\begin{aligned}(\ 0\ \ 0\ -1\ )\cdot(\ w_1\ \ w_2\ \ w_3\ )' &> 0,\\ (\ 0\ -1\ -1\ )\cdot(\ w_1\ \ w_2\ \ w_3\ )' &> 0,\\ (-1\ \ 0\ -1\ )\cdot(\ w_1\ \ w_2\ \ w_3\ )' &> 0,\\ (\ 1\ \ \ 1\ \ \ 1\ )\cdot(\ w_1\ \ w_2\ \ w_3\ )' &> 0.\end{aligned} \tag{30.7b}$$

Now this set of conditions can be written in a single expression:

$$M\vec{w} > \vec{0}, \tag{30.7c}$$

with M the 3×4 matrix of all the 0s, -1s and 1s, with its rows representing all possible extended inputs $\vec{x}$; $\vec{w}$ is the 1×3 column vector of the three weights; $\vec{0}$ is a 1×4 column vector of zeros.

The MATLAB® script pr30_5 implements the AND function of the Perceptron using the so-called Widrow−Hoff routine for finding the weights (recall that the third weight is actually not a real weight, but the Perceptron's threshold). For this example the routine works as follows:

1. The initial weight vector is set by picking three random numbers (w = rand(3,1)).
2. We then randomly pick an extended input vector, i.e., one of the four rows in Eq. (30.7a).
3. Depending on the selected input, it sets the variable O to reflect the correct output of the AND gate: i.e., according to Table 30.1,

this must be 1 by extended input vector (1 1 1), and −1 otherwise. Note that we represent inactivity by −1 instead of 0.

4. Next the procedure follows an iterative routine to determine if the weight needs correction. This is determined as follows. The Perceptron's output is computed as the dot product (x*w) of its extended input (x) and weight vector (w). If this output is NOT the correct one, i.e., output is −1 when the input was (1 1 1) or +1 in the other cases, the weights are corrected. In contrast, if the output is correct, as stated in point 3 above, the weights are not altered.
5. If, according to the criteria in point 4, a weight correction is required for the weight at the kth iteration of the update, the following expression is employed for the correction:

$$\vec{w}_{k+1} = \vec{w}_k + \rho(O_k - \vec{x}_k \cdot \vec{w}_k)\vec{x}_k \tag{30.8}$$

Here, $\vec{w}_k$ is the weight vector; $\vec{x}_k$ is the input vector; O_k is the desired output for the selected input vector, and ρ is a gain factor for the output's error term $(O_k - \vec{x}_k \cdot \vec{w}_k)$ in the correction. The value of ρ may be decreased with iteration number k, here we get away with keeping it at ½.

The following part of the MATLAB® script* pr30_5 *implements the iterative procedure for weight change.

```
% now correct if the output O is incorrect (Eq. 30.8)
% ---------------------------------------------------
% First, check if the output x*w is less than or equal to zero while it
    should be 1.
if (x*w <= 0) & (O==1)              % perform test.
        E=O-x*w;                    % error term.
        w=w+rho*E*x';               % adapt weights if the test fails.
end;
% Second, check if output is larger than or equal to zero while it
    should be -1.
if (x*w >= 0) & (O==-1)             % perform test.
        E=O-x*w;                    % error term.
        w=w+rho*E*x';               % adapt weights if the test fails.
end;
```

6. Finally, at the end of each iteration, we check the performance of the updated Perceptron using the condition in Eq. (30.7c). We stop the iterations as soon as that condition is satisfied or, to avoid excessive computation time, if the number of iterations exceed 1000.

A typical output of this MATLAB® script is a graph showing the error plotted versus iteration number and the following text:

```
Weights: 0.44,755      1.4404     -1.4476.
Output for inputs (0 0),(0 1),(1 0),(1 1): 0 0 0 1.
Correct result was obtained in 31 iterations.
```

We should point out that the above example works really well since the function (AND) is simple. The input is limited to four alternative ones, and simple (0s or 1s), and the input domain associated with the outputs 1 and −1 can be separated by a single straight line (Fig. 30.6A). Our strategy fails if the input domain separation for the desired output requires more than a single straight line. Therefore, a slightly more complicated function, the eXclusive-OR (XOR, Table 30.1) fails miserably since one needs two straight lines to separate the domains (Fig. 30.6B) (see Exercise 30.2).

The simple network we presented here can, of course, be extended to perform more complex functionality, or process more complicated input. For example, the Perceptron we implemented here may be extended to create networks with more layers (Fig. 30.6B). Another modification would be a change of the type of nonlinearity, e.g., by replacing the threshold (i.e., a hard switch) by a sigmoid function (a soft switch) (Fig. 30.6C). In script pr30_5, the iterations simply stop when the neural-net's response is correct for all possible inputs, i.e., it satisfies Eq. (30.7c). This approach is not always feasible. For example, if the input repertoire consists of real numbers (rather than the 0 or 1 in our example), the number of inputs is not really limited. In addition, we may encounter scenarios where the neural-net's response only approximates the desired output—this would be the case when one uses a sigmoidal nonlinearity that approximates the desired outputs −1 or 1. In these cases one must find alternative criteria to end the iteration procedure for adapting the weights. Usually this is done by ending the iterations at a preset acceptable output error level, and/or by stopping the iterative procedure if no significant improvement is further obtained. In these cases it is common to employ two data sets: a training-set to train the network and establish its weights, and a separate test-set to validate the performance of the trained network. It is not uncommon that a network's performance is perfect on the training-set while it performs poorly on the test-set—in this case we have the common problem of a network that is overtrained.

By adding nodes and connections to the network, its capacity to classify complex inputs is increased. We saw that a single layer did a good job of implementing a Boolean function, such as AND, but was not capable performing an XOR operation—for that we needed an extra layer (Fig. 30.6A and B). Although the two layer nets are more powerful, they also have their limitations, and adding even more layers makes the network more powerful in pattern recognition tasks, for example when clusters of different classes of inputs cannot be separated by a simple pair of straight lines. An example of a more complex network is depicted in Fig. 30.6D. Here we have an input layer, a hidden layer, and an output layer. Although adding layers to the network increases its detection performance, a more complex design comes with the cost of having to learn more parameters! As we did in the example of the AND function above, we can use an error correction approach to develop a learning strategy with supervision. However, in this case we only know the desired output of the output layer, and we have no idea what output of the nodes in the hidden layer should be! Therefore, a strategy was developed that uses the knowledge of the desired output at the output layer to adjust the weights across the whole network, including those of the hidden layer. This strategy is based on finding the best solution for the weights using derivatives and the chain-rule. Details of this strategy to adjust the weights in more complex multilayer networks are discussed in Appendix 30.2.

The model described by Lopes da Silva et al. (1974) is a model of the thalamus and one of the first models in electrophysiology that is inspired by the Wilson−Cowan approach. A similar treatment of the mesoscopic model, with altered synaptic function, was given by van Rotterdam et al. (1982). The base model describes a thalamic network, however, the populations are identical to those in the cortical Wilson−Cowan model: one excitatory and one inhibitory cell group. Similar to the Wilson−Cowan approach they assume an invertible function f that translates membrane potential, $V_e(t)$ and $V_i(t)$ (i.e., the input) for the excitatory and inhibitory populations, into a rate of action potentials for each population $E(t)$ and $I(t)$ (i.e., the output). For the excitatory population we then get the following relationships: $E(t) = f_e[V_e(t)]$ and $V_e(t) = f_e^{-1}[E(t)]$. Here f_e and $f_e^{-1}$are the function and its inverse that translates membrane potential into spike rate and vice versa. Both the functions f_e and its equivalent for the inhibitory population f_i are basically the static nonlinearities in their model. Each nonlinear component is subsequently approached by a Taylor series about the mean value of E denoted as $\overline{E}$:

$$f_e^{-1}[E(t)] = a_{e0}\overline{E} + a_{e1}\left(E(t) - \overline{E}\right) + a_{e2}\left(E(t) - \overline{E}\right)^2 + \ldots. \tag{30.9}$$

Subsequently, Lopes da Silva et al. simplified the notation by considering the deviation from the mean only, $e(t) = E(t) - \overline{E}$. This is done for all variables in the model, and next they linearize about the mean by neglecting the second- and higher-order terms in the Taylor series so that the original equations can be approximated by $v_e(t) = f_e^{-1}[e(t)] = a_{e1}e(t)$ and $v_i(t) = f_i^{-1}[i(t)] = a_{i1}i(t)$. Now, after applying the linearization, the diagram in Fig. 30.7A can be considered as a set of coupled LTI systems and we can apply the analysis techniques we outlined in Chapters 12, 13, and 16–18. External input $P(t)$ or its demeaned version $p(t)$ perturbs the system (top left in Fig. 30.5A). For the excitatory component in the diagram in the Laplace domain we get $V_e(s) = H_e(s)P(s)$, with the following Laplace transform pairs: $V_e(s) \Leftrightarrow v_e(t)$, $H_e(s) \Leftrightarrow h_e(t)$, and $P(s) \Leftrightarrow p(t)$. Here h_e and H_e are the unit impulse response and transfer function of the excitatory function.

In a similar fashion we include the effect of the inhibition in the diagram in Fig. 30.7A, and get

$$V_e(s) = a_{e1}E(s) = H_e(s)P(s) - C_2I(s)H_i(s) \tag{30.10}$$

Following the same procedure for the linearized inhibitory component (bottom part in Fig. 30.7A) gives

$$V_i(s) = a_{i1}I(s) = C_1E(s)H_e(s) \tag{30.11}$$

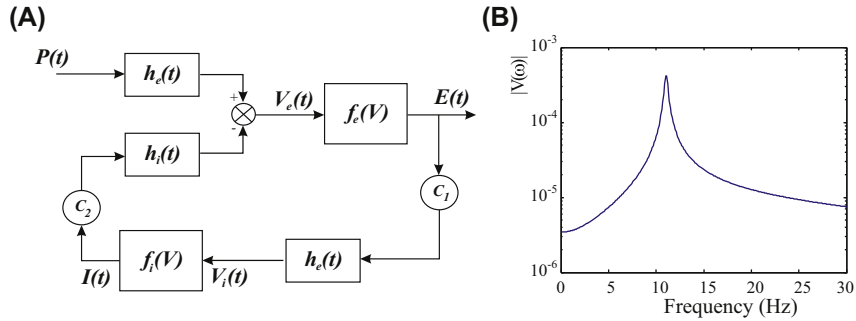


FIGURE 30.7 (A) Diagram of a network model described by Lopes da Silva et al. (1974). $P(t)$ is the external input; h_e and h_i are the population's linear unit impulse responses reflecting the synaptic function. Variables V_e and E are the population's mean membrane potential and firing rate, respectively; V_i and I are the same for the inhibitory population. Functions f_e and f_i are the nonlinear functions that convert population membrane potential into population firing rate. C_1 and C_2 are coupling constants. (B) Result for the spectrum following Eq. (30.13).

If we substitute the expression for $I(s)$ (obtained from Eq. 30.11) into Eq. (30.10) and use $E(s) = V_e(s)/a_{e1}$(from Eq. 30.10), we obtain the following expression for $V_e(s)$:

$$V_e(s) = \frac{H_e(s)P(s)}{1 + \dfrac{C_1 C_2 H_i(s) H_e(s)}{a_{i1} a_{e1}}} \tag{30.12}$$

Then the authors use a linear system to model synaptic input with unit impulse responses (UIRs) h_e and h_i. In Lopes da Silva et al. (1974) the synaptic UIR is a dual exponential $A\left[e^{-a_1 t} - e^{-a_2 t}\right]$ and in van Rotterdam et al. (1982) it is an alpha function Ate^{-at}; in both cases these functions are only considered for $t \geq 0$. The Laplace and Fourier transforms of the dual exponential and alpha function are, of course, slightly different. Here we follow the paper of Lopes da Silva et al. (1974) and use the dual exponential unit impulse response for both the excitatory and inhibitory populations. These functions and their Laplace transforms are:

$$h_e(t) = A\left[e^{-a_1 t} - e^{-a_2 t}\right] \Leftrightarrow H_e(s) = \frac{A(a_2 - a_1)}{(s + a_1)(s + a_2)}$$

$$h_i(t) = B\left[e^{-b_1 t} - e^{-b_2 t}\right] \Leftrightarrow H_i(s) = \frac{B(b_2 - b_1)}{(s + b_1)(s + b_2)}$$

We can substitute this into Eq. (30.12) and simplify:

$$V_e(s) = \frac{A(a_2 - a_1)(s + b_1)(s + b_2)}{(s + a_1)(s + a_2)(s + b_1)(s + b_2) + K} P(s), \tag{30.13}$$

with $K = \dfrac{C_1 C_2}{a_{i1} a_{e1}} (a_2 - a_1)(b_2 - b_1) AB$

We can substitute s by $j\omega$ in $V_e(s)$ to obtain an expression for the spectrum of v_e. The spectrum can be further quantified by assuming that the thalamic input $p(t)$ is white noise and therefore $P(\omega)$ is constant. The numerical values we use for Fig. 30.7B are the same as those in the example given in Lopes da Silva et al. (1974): $A = 1.65$ mV, $B = 32$ mV, $C_1 = 32$ cells, $C_2 = 3$ cells, $a_1 = 55\ \text{s}^{-1}$, $a_2 = 605\ \text{s}^{-1}$, $b_1 = 27.5\ \text{s}^{-1}$, $b_2 = 55\ \text{s}^{-1}$, $q_{e1} = 1/a_{e1}$, $q_{i1} = 1/a_{i1}$, and $q_{i1} q_{e1} = 4.55 \times 10^6$ (K $\approx$ 3.5e8). It can be seen that the spectrum peaks at around 10 Hz, the alpha rhythm frequently encountered in the EEG of subjects with eyes closed. The result depicted in Fig. 30.7B can be obtained with MATLAB® script pr30_6.

30.3.3.1 Neural Mass Models

The approach by Wilson and Cowan (1972) and Lopes da Silva et al. (1974) inspired many other models in neuroscience. Instead of considering the details of individual neurons, these models determine the activity of coupled populations of nerve cells, each population representing a **neural mass**. A very basic model, using two populations representing a coupled excitatory and inhibitory network is described by Dayan and Abbott (2001: pp. 265–270). The rate of change for each network is inversely proportional to its activity level, and proportional to the input it receives from itself, the other population, and an external input. This results in a set of two coupled differential equations. Although the model seems to be linear, their model is actually nonlinear because their population activities are restricted to positive values only. Therefore their model is capable of generating limit cycles.

Jansen's neural mass model (Jansen and Rit, 1995) is applied by Grimbert and Faugeras (2006) to create a cortical network unit (Fig. 30.8A). It is a temporal model of neuronal populations where each population is represented by a Wiener cascade (Chapters 24–26): a dynamic linear component, modeling the synaptic conversion process, and a nonlinear static component that represents the conversion of membrane potential to spike rate (Fig. 30.8B). As can be seen in Fig. 30.8B, input x_1 (input spike rate) is first convolved with the unit impulse response (UIR) h of the linear dynamic component, the result $x_2 = x_1 \otimes h$ (x_2—population membrane potential; $\otimes$— convolution) is subsequently converted with a static sigmoid to produce output $x_3 = \sigma(x_2)$ (output spike rate). The structure of the model by Grimbert and Faugeras (2006) is similar to the canonical cortical model of Douglas and Martin (1991) in the sense that it also includes two excitatory populations E, one inhibitory population I, and an external input P. The steps to derive the equations that govern the activities are straightforward and outlined in the following.

In **step 1**, we use the UIR to write a convolution expression. Again, the synapse is considered a linear system and, in this model, the synaptic impulse response function is defined as an alpha function (as in van Rotterdam et al., 1982):

$$h(t) = \begin{cases} \alpha\beta t e^{-\beta t} & t \geq 0 \\ 0 \quad \text{otherwise} \end{cases},$$

in which α and β are population-specific constants. The Laplace transform (Chapter 12) of the unit impulse response is:

$$H(s) = \frac{\alpha\beta}{(s+\beta)^2}.$$

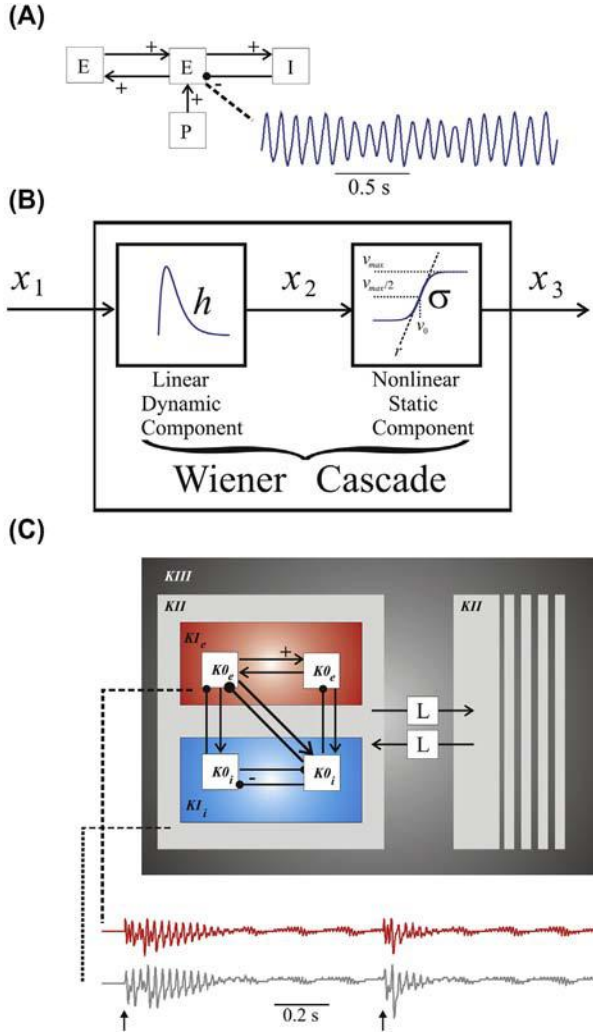


FIGURE 30.8 Neural mass models. (A) A diagram of the model of a cortical unit employed by Grimbert and Faugeras (2006) and a sample trace showing an oscillation similar to an EEG alpha rhythm. The trace was produced with pr30_7. (B) Representation of a Wiener cascade used to model a population of nerve cells. The Wiener cascade consists of a linear dynamical module and a static nonlinear one. (C) Schematic representation of the model described by Freeman (1987) and sample traces showing chaotic behavior of the impulse responses (impulse input – *arrows*). Sample traces were produced with pr30_9.

In the time ⇔ Laplace domain we may link the input $x(t) \Leftrightarrow$ transformed input $X(s)$ to the output $y(t) \Leftrightarrow$ transformed output $Y(s)$ of any linear system by the transform pair:

$$Y(s) = H(s)X(s) \Leftrightarrow y(t) = h(t) \otimes x(t).$$

In **step 2**, we write the convolution relationship as an ODE. Substituting the above expression for H in the Laplace transformed expression we get:

$$(s+\beta)^2\, Y(s) = \alpha\beta\, X(s).$$

This result can be rewritten as:

$$s^2Y(s) + 2\beta\, sY(s) + \beta^2 Y(s) = \alpha\beta\, X(s).$$

Recall that $sY(s)$ and $s^2Y(s)$ denote the Laplace transforms of the first and second derivatives of $y(t)$ in the time domain; using this, we can transform the above expression into a second-order ODE for the linear synaptic process: $\ddot{y}(t) = \alpha\beta\, x(t) - 2\beta\, \dot{y}(t) - \beta^2 y(t)$, which can be written as two first-order ODEs (Chapter 9):

$$\begin{aligned}\dot{y}(t) &= z(t)\\ \dot{z}(t) &= \alpha\beta x(t) - 2\beta z(t) - \beta^2 y(t)\end{aligned}$$

In **step 3**, we add the static nonlinearity employed for the membrane potential to pulse rate conversion, a sigmoid curve $\sigma(v)$ (Fig. 30.8B). In this study it is defined as a Boltzmann function with maximum rate $v_{\max}$, slope r, and a 50% level threshold v_0: $\sigma(v) = \dfrac{v_{\max}}{1+e^{-r(v-v_0)}}$. This completes the formalism for one population, a single Wiener cascade in Fig. 30.8B. The pair of ODEs plus the static nonlinearity can be defined for each of the three populations in the neural mass model (two excitatory and one inhibitory; Fig. 30.8A). This means we will have a pair of first-order differential equations per population, generating a total of six equations (Eq. (3) in Grimbert and Faugeras, 2006):

$$\begin{array}{ll}\dot{y}_0(t) = y_3(t) & \dot{y}_3(t) = Aa\sigma(y_1(t) - y_2(t)) - 2ay_3(t) - a^2y_0(t)\\ \dot{y}_1(t) = y_4(t) & \dot{y}_4(t) = Aa\{p(t) + C_2\sigma(C_1y_0)\} - 2ay_4(t) - a^2y_1(t)\\ \dot{y}_2(t) = y_5(t) & \dot{y}_5(t) = BbC_4\sigma(C_3y_0) - 2by_5(t) - b^2y_2(t)\end{array} \quad (30.14)$$

The parameters $y_0 - y_2$ denote the membrane potentials, and A, a, B, b are the constants in the population-specific alpha functions. Note that in these equations, some outputs y and sigmoid functions are weighted by connectivity strengths C_1–C_4. The focus in this model is on the variable y, the aggregate membrane potential of the main population, which is thought to be proportional with the local field potential. A simulation using the following parameters $A = 3.25$, $a = 100$, $B = 22$, $b = 50$, $C_1 = 135$, $C_2 = 0.8$*C_1, $C_3 = 0.25$*C_1, $C_4 = 0.25$*C_1, $v_{\max} = 5$, $r = 0.56$, and $v_0 = 6$, shows a signal that is similar to the EEG

alpha rhythm (Fig. 30.8A). This simulation is implemented in MATLAB® script pr30_7.

30.3.3.2 *Modeling the Olfactory System*

The models created by Freeman in the 1980s used a combined network modeling and experimental approach to study electrical activity in the rabbit olfactory system. The fundamental building blocks of his model are neural masses. The appeal of Freeman's approach is that the spatial component of the model is based on the anatomy of the olfactory system. In addition, modeling results are related to electrophysiological experiments. As compared to the previous models, a unique addition is that it contains latencies L1−L4, modeling the conduction delays occurring in between populations (Freeman, 1987). Differential equations without delays need initial values, whereas equations with delays require initial functions to determine the delayed response of the system. Equations with delays can exhibit behavior that cannot be displayed by equations without delay of the same order. It is a given that delays in the nervous system occur because of the finite conduction velocity of nerve fiber activity. In addition to Freeman's model, several others have explored the role of delays or included the delays in the formalism (e.g., Foss et al., 1996; Milton, 2000, 2012; Visser et al., 2010, 2012).

Freeman and coworkers define the following neuronal population levels (Fig. 30.8C):

1. ***K0*:** A subset of **noninteracting** neurons, either all excitatory ($K0_e$) or inhibitory ($K0_i$). Note that although the neurons within each *K0* population do not interact, interaction between *K0* populations does exist. Accordingly, the smallest unit in Fig. 30.8C is the *K0* subset. In each *K0* subset j, the wave variable v_j is governed by a linear second-order ODE:

$$ab\,F(v_j) = \ddot{v}_j + (a+b)\dot{v}_j + abv_j, \quad \text{with } a \text{ and } b \text{ constants.}$$

This is similar to the approach for modeling synaptic dynamics that is used in the population models of Lopes da Silva and Grimbert and Faugeras discussed above. For the output there is a sigmoid function to convert the membrane potential to pulse rate:

$$p_j = A - B\exp(C - Ce^{v_j}), \quad \text{with } A,\ B,\ \text{and } C \text{ constants}$$

2. ***KI***: Coupled *K0* populations create the *KI* network level. In Freeman's terminology, two $K0_e$ modules give a KI_e module and a pair of $K0_i$ gives one KI_i (Fig. 30.8C). For each population j, the coupling is governed by forcing term $F(v_j)$, a function

representing the incoming pulses originating from the connected populations. As shown above, the dynamics of the system's basic *K0* population is described by a second-order differential equation with a driving term (i.e., $F(v_j)$). If we Laplace transform the ODE while assuming that the driving force $F(v_j)$ is a unit impulse $\delta(t)$, we get:

$$ab = \left[s^2 + (a+b)s + ab\right]H(s), \quad \text{with } H(s) - \text{transfer function.}$$

The inverse Laplace transform of the transfer function, the UIR, is a dual exponential (also used in other models such as Lopes da Silva, 1974).

3. *KII*: This type of network arises when a KI_e and KI_i are coupled (Fig. 30.8C). The distance between *KII* populations can be large enough that conduction delays cannot be neglected. These delays are indicated by L in Fig. 30.8C.
4. *KIII*: When *KII* sets are interconnected, a *KIII* set is created. Note that there is just one such set in Fig. 30.8. In Freeman (1987) there are three coupled *KII* sets: the olfactory bulb (OB), the anterior olfactory nucleus (AON), and the prepyriform cortex (PC). The connections in this set are longer range so that lags (latencies) were incorporated. These latencies are included in his expressions of the driving force by a backward counter *J*. For example, let us consider the effect on one of the two populations of granular cells, *G*1 in the OB in Freeman (1987):

$$F(v_{G1}) = 1.5p_{M1} + 1.5p_{M2} - 1.8p_{G2} + (1/9)\sum_{J=14}^{22} p_{E1}(J) + (k/17)\sum_{J=22}^{38} p_C(J) \tag{30.15a}$$

The first three terms represent the effect of the incoming pulse rates from mitral cell populations *M*1 and *M*2 as well as the inhibitory effect from the other granular cell population *G*2 on the forcing term $F(v_{G1})$. The fourth term is the effect of a subpopulation from AON:

$$(1/9)\sum_{J=14}^{22} p_{E1}(J) \tag{30.15b}$$

This term represents the weighted sum of the AON population's pulse values P_{E1} in the past 14–22 steps. Note that the coefficient is scaled by the number of backward steps that are included in the

sum, i.e., nine. Note that Eq. (30.15a) also includes a fifth term that reflects the delayed input from cortical network PC (in this term k is a constant).

Simulations of Freeman's model were implemented in MATLAB® scripts pr30_8 and pr30_9, and the results are depicted in the bottom panel in Fig. 30.8C. To show that the system is initially quiet, we deliver a first stimulus a little after the onset of the simulation and to demonstrate that the response of the active intact network is slightly different, we deliver another stimulus at about the halfway point of the simulation (arrows in Fig. 30.8C).

30.4 SPATIOTEMPORAL MODEL FOR THE ELECTROENCEPHALOGRAM

The electrical signal that can be recorded from the scalp, the EEG, reflects the currents generated by the brain's nerve cells. If we ignore electrical signals from other more remote sources (e.g., muscle, heart) and nonbiological artifacts, we can assume that the EEG is the weighted sum of the activities of the underlying neural network. Due to volume conduction responsible for attenuating signals propagating across the tissue between source and electrode, the weights in this weighted sum are determined by the geometry and the tissue's conductivity properties. The EEG signal is used both in research and clinical settings. Because a single EEG signal includes the activity of millions of nerve cells, the relationship between smaller networks of the brain and the EEG signal is not necessarily a trivial one. Therefore this relationship has been the target of many modeling studies; the framework developed by physicist Paul Nunez is an example. Because it is generally thought that the slower synaptic potentials, especially those of the neocortical pyramidal cells, contribute most to the EEG signals on the surface of cortex and scalp, Nunez' model focused on describing synaptic activity across the cortex (e.g., Nunez, 1974, 1995). The spatiotemporal model of cortical activity is based on the schematic of the neocortex shown in Fig. 30.9. This diagram shows the effect of action potential firing activity g produced by the left cube on the synaptic activity h of the right cube. The synaptic activity has subscripts e and i for the excitatory and inhibitory synapses, respectively. Both activity variables g and h are a function of location: $\vec{r}_1$(left cube), $\vec{r}$ (right cube), and time t. The remaining variables in Fig. 30.9 are v for conduction velocity and $u(\vec{r},t)$ for external, subcortical input. Consequently the delay for action potentials arriving at the right cube that were generated in the left cube is $\frac{|\vec{r}-\vec{r}_1|}{v}$; therefore the action potential firing function from the left cube arriving

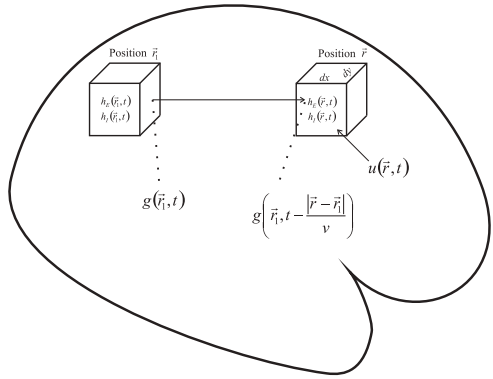


FIGURE 30.9 Diagram of the model presented by Nunez showing the interaction between cortical volume units. The output g of one volume at position $\vec{r}_1$ arrives at its target at position $\vec{r}$ with a delay determined by conduction velocity (v). Each volume unit is characterized by its synaptic activity, both for excitatory (h_e) and inhibitory synapses (h_i).

at the right cube is indicated as: $g\left(\vec{r}_1,\ t-\frac{|\vec{r}-\vec{r}_1|}{v}\right)$. Furthermore, the connectivity within the cortex is governed by functions R_E and R_I for the excitatory and inhibitory connections, respectively. These functions also depend on the locations $\vec{r}$, $\vec{r}_1$, and fiber system (with conduction velocity v), i.e., $R_E = R_E(\vec{r}, \vec{r}_1, v)$. The equation that describes the excitatory synaptic activity as a function of the input to the right cube can now be formulated as:

$$h_E(\vec{r},t) = u(\vec{r},t) + \int_0^\infty dv \int_S R_E(\vec{r}, \vec{r}_1, v) g\left(\vec{r}_1, t-\frac{|\vec{r}-\vec{r}_1|}{v}\right) d^2 r_1 \tag{30.16}$$

For the inhibitory synaptic activity a simpler expression can be used since there are only local inhibitory fibers all with similar conduction velocity, i.e., $R_I = R_I(\vec{r}, \vec{r}_1)$. Further, because of the local character of inhibition, one can neglect the delays for the inhibitory fibers: i.e., now the action potential firing function is simply $g(\vec{r}_1, t)$. For the inhibitory expression we get:

$$h_I(\vec{r},t) = u_0 + \int_S R_I(\vec{r}, \vec{r}_1) g(\vec{r}_1, t) d^2 r_1 \tag{30.17}$$

The external input u_0 represents subcortical input, which is assumed to be constant for a given state. Because the physiological basis for this

input isn't obvious, Nunez dropped this term in later versions of the model. In the following it isn't very important since the effect of u_0 vanishes when linearizing the expression. Note that both in Eqs. (30.16) and (30.17) the connectivity functions R_E and R_I and/or activity function g must contain a component that translates the action potential rate into a level of synaptic activity. In his development Nunez assumes this component to be linear by using a simple gain/attenuation factor. The same can be said for external input functions $u(\vec{r},t)$ and u_0.

Based on the comments above, it is clear that while the expressions in Eqs. (30.16) and (30.17) explicitly relate spike input to synaptic activity (pulse-to-wave conversion), there is also the effect of the synaptic activity on the spike output (wave-to-pulse conversion) to be considered. The nonlinear relationship for both synaptic activities h_E and h_I on action potential rate function g is commonly modeled by a sigmoid function (e.g., Nunez, 1995). Nunez linearizes this relationship about an assumed fixed state g_0. If we consider a small change of the action potential rate δg around state g_0 and relate this to small changes in the synaptic activities δh_E and δh_I, we get:

$$\delta g = \left(\frac{\partial g}{\partial h_E}\right)_{g=g_0} \delta h_E + \left(\frac{\partial g}{\partial h_I}\right)_{g=g_0} \delta h_I.$$

Defining:

$$H_E = \delta h_E \quad \text{and} \quad H_I = \delta h_I,$$

$$Q_E = \left(\frac{\partial g}{\partial h_E}\right)_{g=g_0}, \quad \text{and} \quad Q_I = -\left(\frac{\partial g}{\partial h_I}\right)_{g=g_0},$$

we can simplify the notation and get $\delta g = Q_E H_E - Q_I H_I$. If we now use Eq. (30.16) to write an expression for δh_E, we get:

$$\underbrace{\delta h_E(\vec{r},t)}_{H_E(\vec{r},t)} = \underbrace{\delta u(\vec{r},t)}_{U(\vec{r},t)} + \int_0^\infty dv \int_S R_E(\vec{r},\vec{r}_1,v) \times \underbrace{\delta g\left(\vec{r}_1, t - \frac{|\vec{r}-\vec{r}_1|}{v}\right)}_{Q_E H_E\left(\vec{r}_1, t-\frac{|\vec{r}-\vec{r}_1|}{v}\right) - Q_I H_I\left(\vec{r}_1, t-\frac{|\vec{r}-\vec{r}_1|}{v}\right)} d^2 r_1,$$

with the changes of notation indicated by the curly brackets, we obtain (Eq. 11.5 in Nunez, 1995):

$$H_E(\vec{r},t) = U(\vec{r},t) + \int_0^\infty dv \int_S R_E(\vec{r},\vec{r}_1,v) \left[Q_E H_E\left(\vec{r}_1, t - \frac{|\vec{r}-\vec{r}_1|}{v}\right) - Q_I H_I\left(\vec{r}_1, t - \frac{|\vec{r}-\vec{r}_1|}{v}\right)\right] d^2 r_1 \tag{30.18}$$

Following the same procedure for Eq. (30.17) gives (Nunez' Eq. 11.6):

$$H_I(\vec{r},t) = \int_S R_I(\vec{r},\vec{r}_1)[Q_E H_E(\vec{r}_1,t) - Q_I H_I(\vec{r}_1,t)] d^2 r_1 \tag{30.19}$$

One of the strong aspects of Nunez' model is that it acknowledges different conduction velocities, the weak part is that it linearizes the relationship between synaptic activity and action potential rate, and vice versa.

30.5 A FIELD EQUATION FOR THE ELECTROENCEPHALOGRAM

A summary of mesoscopic modeling is provided in Jirsa and Haken (1997). These authors integrate temporal and spatial aspects, the firing rate models first described by Wilson and Cowan (1972, 1973), and the synaptic fields described by Nunez (e.g., Nunez, 1974, 1995). Their approach is straightforward. First they develop expressions for the synaptic and action potential activities. Then they simplify by zooming out, determine the expression for the synaptic field, and write that expression in the form of a convolution operation. Finally, they take the Fourier transform of the convolution and transform this result back into the spatiotemporal domain as a partial differential equation for the synaptic field of the excitatory neurons. As did Nunez (Section 30.4), Jirsa and Haken assume that this field is a representative for the EEG.

30.5.1 Spatiotemporal Expression for the Synaptic Field

The excitatory (e) and inhibitory (i) populations each have pulse rate E and I ($E(\boldsymbol{x},t)$ and $I(\boldsymbol{x},t)$) similar to the approach by Wilson and Cowan (1972), and a wave expression $\psi(\psi_e(\boldsymbol{x},t)$ and $\psi_i(\boldsymbol{x},t))$ similar to the

development by Nunez. In the above, the spatial component $\boldsymbol{x}$ can be considered three-dimensional; in the remainder of the text, the spatial variable is reduced to one dimension, represented by x. In the following text, we follow the derivation of the macroscopic field equation as it is visualized in Fig. 30.10. In order to find the expression for the synaptic activity, we determine the synaptic component (top, Fig. 30.10), while including the spike rate generator (bottom, Fig. 30.10). The left part of Fig. 30.10 shows the details, whereas the right part shows the zoomed-out version representing a mesoscopic view where some detail is lost.

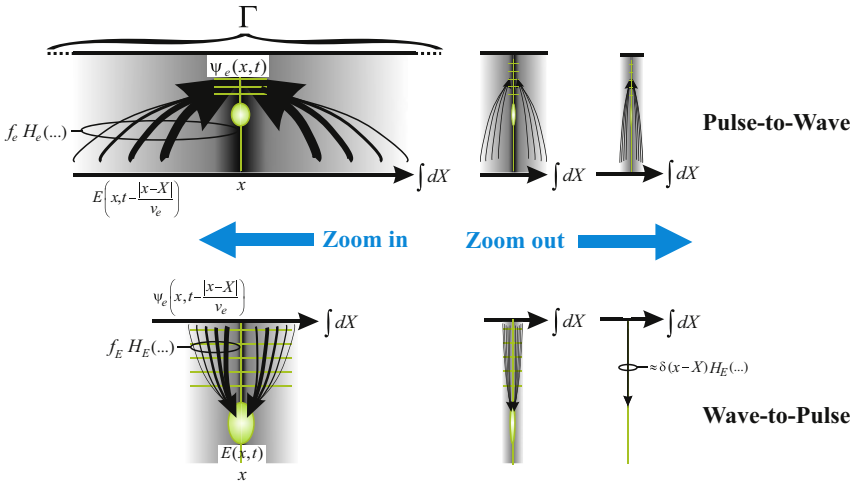


FIGURE 30.10 Diagrams of the model described by Jirsa and Haken (1997). They explicitly model the conversion between cellular output and the synaptic input (transformation from pulse-to-wave, upper panels) and the conversion between synaptic activity and spike rate (transformation from wave-to-pulse, bottom panels). Top-left panel shows how the excitatory synaptic field ψ_e (as a function of time t and place x) is generated by the action potential field E by integration over the one-dimensional cortical "area" Γ using integration variable X. The outcome (Eq. 30.20) depends on the distribution of connectivity f_e and the conversion operation H_e (Eq. 30.21). The other panels in the top row show the same picture in various degrees of zooming out. The bottom-left panel shows the generation of the action potential field E (as a function of time t and place x) by the excitatory synaptic field ψ_e (Eq. 30.22). The outcome depends on the distribution of connectivity f_E and the conversion operation H_E (Eq. 30.23). The other panels in the bottom row show the same in various degrees of zooming out. Note that the zoomed-out distribution function f_E approaches a delta function δ $(x-X)$. The depicted relationships only show the excitatory population; for the inhibitors, similar diagrams can be constructed.

30.5.1.1 Synaptic Component

The synaptic activityψ resulting from the incoming spike activity; for the excitatory population is:

$$\underbrace{\psi_e(x,t)}_{\substack{\text{synaptic activity excitatory population at location } x \text{ and time } t}} = \underbrace{\int_{\Gamma}}_{\substack{\text{Surface Area of the brain}}} dX \quad \underbrace{f_e(x,X)}_{\substack{\text{Distribution Function of Spatial Connectivity of axons(output) at } X \text{ to } x}} \quad \underbrace{H_e(x,X,t)}_{\substack{\text{Output of Conversion Pulse}\rightarrow\text{Wave}}} \tag{30.20}$$

For the inhibitory population a similar expression can be presented. In the equations for synaptic activity, the contribution from the pulse activity over the whole cortex is included: at location x, this contribution is the product of all converted cortical activity H multiplied by the connection distribution f for x. Synaptic activity ψ is then computed by integrating (using integration variable X) over cortical surface Γ (Fig. 30.10). The conversion function H_e determines how incoming action potentials originating from excitatory cells are converted into excitatory synaptic activity (excitatory postsynaptic potentials). This is represented by a sigmoid function S that can be linearized over most of the trajectory using the constant slope a_e:

$$H_e(x,X,t) \approx a_e E\left(X, t - \frac{|x-X|}{v_e}\right). \tag{30.21}$$

Note that the $E(\ldots)$ function describes all action potentials generated in the past that could have traveled to location x originating from location X (Fig. 30.10, top row) in a time interval $\frac{|x-X|}{v_e}$, with v_e being the conduction velocity. Those that have really arrived at x depend on the connectivity between x and X; this is described by function $f_e(x,X)$.

30.5.1.2 Spike Rate Component

The expression for the spike rate for the excitatory population E is similar to Eq. (30.20):

$$\underbrace{E(x,t)}_{\substack{\text{spike density excitatory population at location } x \text{ and time } t}} = \underbrace{\int_{\Gamma}}_{\substack{\text{Surface Area of the brain}}} dX \quad \underbrace{f_E(x,X)}_{\substack{\text{Distribution Function of Spatial Connectivity of dendrites(input) at } X \text{ to } x}} \quad \underbrace{H_E(x,X,t)}_{\substack{\text{Output of Conversion Wave}\rightarrow\text{Pulse}}} \tag{30.22}$$

Again, note the difference in the subscripts, such as in H_e and H_E. The output conversion H_E: wave-to-pulse becomes an expression that reflects the conversion from: (a) the excitatory synapses, (b) the inhibitory synapses, and (c) a term for the extracortical activity. The synaptic activities for the three wave sources are simply added and a (static) nonlinearity S_e converts it to pulse rate. Like in the expression in Eq. (30.21), there is a temporal delay term $|x - X|/v$ that corrects for conduction velocity. The expression for the wave-to-pulse conversion becomes

$$\underbrace{H_E(x, X, t)}_{\substack{\text{Conversion} \\ \text{Wave}\rightarrow\text{Pulse}}} = S_e\left(\underbrace{\psi_e\left(X, t - \frac{|x - X|}{v_e}\right)}_{\substack{\text{Contribution of the Excitatory Synaptic Activities from location } X \text{ at times } t-\text{delay}}} - \underbrace{\psi_i\left(X, t - \frac{|x - X|}{v_i}\right)}_{\substack{\text{Contribution of the Inhibitory Synaptic Activities}}} + \underbrace{p_e\left(X, t - \frac{|x - X|}{v}\right)}_{\substack{\text{Contribution of the External Synaptic Activities}}}\right) \tag{30.23}$$

The inclusion of a conduction delay is correct when assuming active propagation of synaptic input, for passive propagation (at the speed of light) such a delay isn't needed. For the remainder of the derivation by Jirsa and Haken (1997) this point is irrelevant because these delays will be neglected. The conversion function H_E describes the contributions from all cortical areas X to location x and the distribution function f_E included in Eq. (30.22) determines the strength of the connection.

30.5.2 Zooming Out

In the model, cortical connectivity is described by exponential distributions of the form $\frac{1}{2\sigma}e^{-\frac{|x|}{\sigma}}$, functions where the area under the curve evaluates to unity: $\int_{-\infty}^{\infty} \frac{1}{2\sigma}e^{-\frac{|x|}{\sigma}}\,dx = 1$. The advantage is that when we zoom out to a global overview of the cortex, these distributions can be replaced by delta functions. In the model, the distributions f_i, f_I, and f_E are represented by a delta function because they are local (see, for example, f_E in Fig. 30.10 bottom row). This procedure for f_E is depicted in Fig. 30.10, bottom row: $f_E(x, X) \approx \delta(X - x)$. Distribution f_e, describing the

excitatory connections over a longer range, remains in the form of the exponential expression (Fig. 30.10, top row).

30.5.2.1 The Expression for ψ_e

Now we use the above to formulate the expression for the synaptic activity because this variable is considered proportional to the EEG. In this approach we focus on solving for the excitatory component since this is considered the major contributor to the EEG signal. First we substitute the linearized version of H_E (Eq. (30.21) into ψ_e (Eq. 30.20):

$$\psi_e(x,t) = \int_\Gamma dX f_e(x,X) a_e E\left(X, t - \frac{|x-X|}{v_e}\right)$$

In this expression, we substitute Eq. (30.22) for $E(\ldots)$, while changing integration variable X to X_1, and we get $\psi_e(x,t) = \int_\Gamma dX f_e(x,X) a_e \int_\Gamma dX_1 f_E(X,X_1) H_E\left(X, X_1, t - \frac{|x-X|}{v_e}\right)$. Subsequently we substitute the expression for H_E from Eq. (30.23) and replace f_E with a delta function:

$$\psi_e(x,t) = \int_\Gamma dX f_e(x,X) a_e \int_\Gamma dX_1 \delta(X - X_1)$$

$$\times S_e \left(\begin{array}{l} \psi_e\left(X_1, t - \frac{|X-X_1|}{v_e} - \frac{|x-X|}{v_e}\right) - \psi_i\left(X_1, t - \frac{|X-X_1|}{v_e} - \frac{|x-X|}{v_e}\right) \\ \\ + p_e\left(X_1, t - \frac{|X-X_1|}{v_e} - \frac{|x-X|}{v_e}\right) \end{array} \right)$$

Now we apply the sifting property to evaluate the second integral and get:

$$\psi_e(x,t) = a_e \int_\Gamma dX f_e(x,X)$$

$$S_e \times \left(\psi_e\left(X, t - \frac{|x-X|}{v_e}\right) - \psi_i\left(X, t - \frac{|x-X|}{v_e}\right) + p_e\left(X, t - \frac{|x-X|}{v_e}\right) \right) \tag{30.24}$$

For the inhibitory population we follow a similar procedure and get:

$$\psi_i(x,t) = a_i \int_\Gamma dX f_i(x,X) \\ S_i \times \left(\psi_e\left(X, t - \frac{|x-X|}{v_i}\right) - \psi_i\left(X, t - \frac{|x-X|}{v_i}\right) + p_i\left(X, t - \frac{|x-X|}{v_i}\right)\right)$$

If we now evaluate the integral in this expression, while replacing the distribution function f_i with the delta function $\delta(X-x)$, so we can use the sifting property, we obtain the result for the inhibitory synaptic activity: $\psi_i(x,t) = a_i S_i(\psi_e(x,t) - \psi_i(x,t) + p_i(x,t))$. Subsequently we follow the procedure by Jirsa and Haken (1997) and only take the linear aspect of the S_i function, using a slope of α_i and finally solve for ψ_i:

$$\psi_i(x,t) = \frac{a_i\alpha_i}{1 + a_i\alpha_i}(\psi_e(x,t) + p_i(x,t)) \tag{30.25}$$

Substitution of Eq. (30.25) into (30.24) gives

$$\psi_e(x,t) = a_e \int_\Gamma dX f_e(x,X) S_e \left\{ \overbrace{\psi_e\left(X, t - \frac{|x-X|}{v_e}\right) - \frac{a_i\alpha_i}{1 + a_i\alpha_i}\psi_e\left(X, t - \frac{|x-X|}{v_e}\right)}^{\widetilde{\rho}\psi_e} \right. \\ \left. \underbrace{+ p_e\left(X, t - \frac{|x-X|}{v_e}\right) - \frac{a_i\alpha_i}{1 + a_i\alpha_i} p_i\left(X, t - \frac{|x-X|}{v_e}\right)}_{p\left(X, t - \frac{|x-X|}{v_e}\right)} \right\}$$

Using the simplifications indicated by the curly brackets above we have the expression for the excitatory synaptic activity:

$$\psi_e(x,t) = a_e \int_\Gamma dX f_e(x,X) S_e \left\{ \widetilde{\rho}\psi_e\left(X, t - \frac{|x-X|}{v_e}\right) + p\left(X, t - \frac{|x-X|}{v_e}\right) \right\} \tag{30.26}$$

30.5.3 Expression for ψ_e as a Convolution

The next step is to write the result for ψ_e in the form of a convolution. First, the notation for the integrand in Eq. (30.26) can be simplified:

$$f_e(x,X)\underbrace{a_e S_e\left\{\tilde{\rho}\psi_e\left(X,t-\frac{|x-X|}{v_e}\right)+p\left(X,t-\frac{|x-X|}{v_e}\right)\right\}}_{\rho\left(X,t-\frac{|x-X|}{v_e}\right)}$$
$$=f_e(x,X)\rho\left(X,t-\frac{|x-X|}{v_e}\right) \tag{30.27}$$

Because the excitatory connections cover a larger area its distribution function cannot be replaced by δ. Therefore we use the exponential distribution for f_e and get:

$$\frac{1}{2\sigma_e}e^{-\frac{|x-X|}{\sigma_e}}\rho\left(X,t-\frac{|x-X|}{v_e}\right).$$

Although it seems that we make things more complicated, we rewrite this as an integral:

$$\frac{1}{2\sigma_e}e^{-\frac{|x-X|}{\sigma_e}}\int_{-\infty}^{\infty}\rho(X,T)\delta\left(T-\left(t-\frac{|x-X|}{v_e}\right)\right)dT.$$

Now we substitute both this result and Eq. (30.27) in Eq. (30.26) and rearrange the order of the terms:

$$\psi_e(x,t)=\int_{\Gamma}\int_{-\infty}^{\infty}\underbrace{\delta\left(t-T-\frac{|x-X|}{v_e}\right)\frac{1}{2\sigma_e}e^{-\frac{|x-X|}{\sigma_e}}}_{G(x-X,t-T)}\rho(X,T)dXdT$$

As indicated with the horizontal bracket, the part $\frac{1}{2\sigma_e}e^{-\frac{|x-X|}{\sigma_e}}\delta\left(\left(t-\frac{|x-X|}{v_e}\right)-T\right)$ can be interpreted as a unit impulse response or Green's function G. Now we complete this step and write the expression in the form of a convolution with respect to space and time:

$$\psi_e(x,t)=\int_{\Gamma}\int_{-\infty}^{\infty}G(x-X,t-T)\rho(X,T)dXdT \tag{30.28}$$

30.5.4 From Convolution to the Field Equation

Now we take advantage of the convolution form of Eq. (30.28). With the spatiotemporal Fourier transforms for $\psi(x,t)$, $\rho(x,t)$, and $G(x,t)$, we can write the transform of the convolution as a product in the frequency domain. To summarize we have the following transform pairs:

$$\psi_e(x,t) \Leftrightarrow \psi_e(k,\omega)$$
$$\rho(x,t) \Leftrightarrow \rho(k,\omega)$$
$$G(x,t) \Leftrightarrow \frac{\omega_0^2 + j\omega_0\omega}{v_e^2k^2 + (\omega_0 + j\omega)^2}, \quad \text{with} \quad \omega_0 = v_e/\sigma_e.$$

Using $\psi_e(x,t) = G(x,t) \otimes \rho(x,t) \Leftrightarrow \psi_e(k,\omega) = G(k,\omega) \times \rho(k,\omega)$, we get:

$$\psi_e(k,\omega) = \frac{\omega_0^2 + j\omega_0\omega}{v_e^2k^2 + (\omega_0 + j\omega)^2}\rho(k,\omega) \tag{30.29}$$

Note that in the temporal Fourier transform $j\omega$ and $-\omega^2$ represent the transformed first and second derivatives in the time domain, the representatives for the first and second spatial derivatives are jk and $-k^2$. Transforming Eq. (30.29), we complete our last step and get the equation for the spatiotemporal excitatory synaptic activity:

$$\left(\omega_0^2 - v_e^2\frac{\partial^2}{\partial x^2}\right)\psi_e(x,t) + 2\omega_0\dot{\psi}_e(x,t) + \ddot{\psi}_e(x,t) = \left(\omega_0^2 + \omega_0\frac{\partial}{\partial t}\right)\rho(x,t) \tag{30.30}$$

An example of simulated activity is depicted in Fig. 30.11. In this example we show the model's response to a half sine wave pulse in the center of a cortical surface. As expected, the evoked activity is damped over both space and time. A simulation of the spatiotemporal behavior of cortical activity can be done with MATLAB® script pr30_10. Interestingly, the expression in Eq. (30.30) is also similar to earlier results obtained for linearized cellular and network models. If we ignore the spatial component in Eq. (30.30), we get a second-order ODE with a forcing term, similar to the result in Appendix 30.1 and Eq. (29.18). However, in this case a nonlinearity is included in the factor ρ of the forcing term.

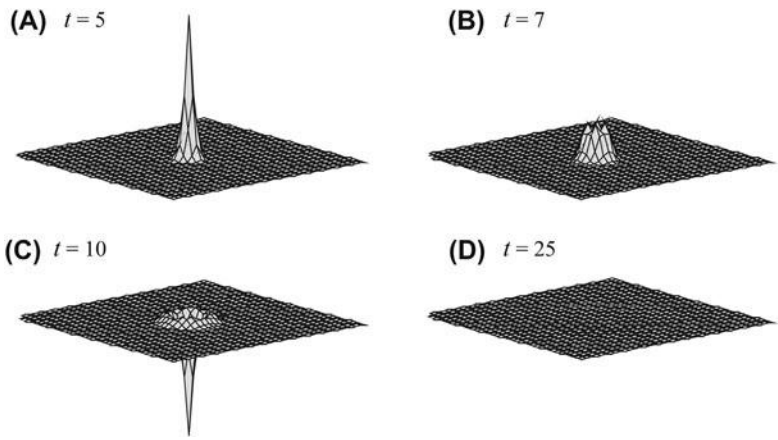
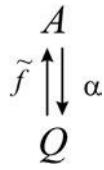


FIGURE 30.11 Simulation of Eq. (30.30) for times 5, 7, 10, and 25 showing the response of a cortical surface to a stimulus pulse in the middle of the surface. The pulse was half a period of a sine wave between times 0 and 10. Parameters used are $\omega_0 = 0.15$; $v_e = 1.52$; $\widetilde{\rho} = 0.9$; $a_e = 1.2$; $\alpha_e = 1.2$.

30.6 MODELS WITH A STOCHASTIC COMPONENT

With the exception of the spin model and Hopfield's approach (Section 30.2.1), all models we have discussed are deterministic. Most of the deterministic approaches can easily be modified to include a stochastic component (Rolls and Deco, 2010). Such a component may represent intrinsic noise in the membrane and/or the synaptic input originating from sources external to the modeled unit. In the 1960s and 1970s, membrane noise was investigated to quantify electrical channel properties (e.g., Verveen and DeFelice, 1974). Especially when modeling a network embedded in the brain, its input from the rest of the brain is generated by enormous networks and a stochastic approach to this input is appropriate and also significant because it affects the network function (e.g., Rudolph and Destexhe, 2001). A critical shortcoming of the deterministic models (e.g., the mean field models discussed in Section 30.3), that can be mitigated with a stochastic approach, is the absence of fluctuations and correlations of the neural activity, especially spike trains (e.g., Rodrigues and Tuckwell, 1996; Giorno et al., 1997). Moreover, a statistical approach is applied to spike trains to relate to information coding in networks in general (Rieke et al., 1997; Doya et al., 2007). Mathematically, stochastic modeling often results in a diffusion equation approach, where the models are frequently approximated by either the Fokker–Planck equation and/or the Langevin equation (e.g., see the Appendix in Rolls and Deco, 2010). Further discussion of these approaches is beyond the scope of this text.

Benayoun et al. (2010a) and Wallace et al. (2011) used a network of switches in which the update rule is both asynchronous and stochastic, known as the Gillespie's exact stochastic simulation algorithm (Gillespie, 1976, 1977). To illustrate this approach, I describe a simplified version



(30.31)

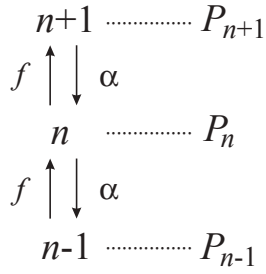
The active neurons A are created from an infinite pool of inactive ones Q at a rate $\tilde{f}$ and they become inactive again with a rate α. The rate function $\tilde{f}$ depends on the activity level in the network, however, for the sake of the example we won't worry about this "detail" here. Rate α is the intrinsic rate at which active cells become inactive (this also includes the refractory period). The deterministic kinetic rate equation for this process is:

$$\frac{dA}{dt} = \tilde{f}Q - \alpha A = f - \alpha A \tag{30.32}$$

We assume here that Q is available in large excess (i.e., an infinite pool) so that it can be considered constant. This equation can be solved both analytically and numerically.

The next step is to consider the so-called master equation for this process. In contrast to the deterministic approach above, we now use a stochastic approach and describe the probability that there are n active neurons (i.e., n cells in our infinite pool are in state A). We may visualize this by a line on which we see all the possible states for the number of active neurons, i.e.,: $0, 1, 2, 3, 4, 5, \ldots, n-1, n, n+1, n+2, n+3, \ldots, \infty$. If we update the system fast enough, we can safely assume that a state change only involves a single step.

Under this assumption, the probabilities describing the dynamics around state n are easily seen in the following diagram that depicts part of the line of possible states.



(30.33)

This leads to the master equation describing the dynamics of the probability P_n for the state with n active neurons

$$\frac{dP_n}{dt} = \alpha(n+1)P_{n+1} - \alpha n P_n + fP_{n-1} - fP_n \tag{30.34}$$

It should be noted that the notation here is simplified by wrapping several dependencies into f. To simplify this further, we can apply shift operators E^+ and E^- that operate on function $f(n)$ and increase or decrease the number n by 1, respectively (Van Kampen, 1992). This allows one to rewrite Eq. (30.34) as a function of Pn:

$$\frac{dP_n}{dt} = \left[\alpha E^+ n - \alpha n + fE^- - f\right]P_n \tag{30.35}$$

The problem is that in most cases the master equation is difficult to solve analytically or cannot be solved at all. Gillespie (1976, 1977) formulated a remarkable and exact algorithm to simulate chemical systems in a stochastic fashion. Within this approach, one starts from the system in a particular state at time t and determines the probability density function (PDF) for the next reaction. Briefly, if we consider $1, 2, \ldots, \mu, \ldots, M$ reactions; if one simulates such a system over time, we only need to know

1. when the next reaction (any of the M) will occur, and
2. what kind of reaction (μ) it will be.

In order to resolve these questions, Gillespie follows the system of reactions from any time t to $t + \tau + d\tau$. Accordingly, he formulates the joint probability $P(\tau, \mu)\, d\tau$ as the probability that none of the reactions occurs in the interval between t and $t + \tau$, and that reaction μ does occur in the interval between $t + \tau$ and $t + \tau + d\tau$ (Fig. 30.12A). It is straightforward to show that $P(\tau, \mu)\, d\tau$ is characterized by the product of both probabilities. Let's first determine the first component P_0 that no reaction occurs in the interval between t and $t + \tau$. Consider a time axis τ' set at 0 immediately after one of the reactions occurred. Consequently, the probability that none of the reactions occurs in $d\tau'$ is

$$1 - \sum_{\mu=1}^{M} a_\mu d\tau' \tag{30.36}$$

Thus one can define the probability of no reaction at $\tau' + d\tau'$ is the probability of no reaction at τ' multiplied by the expression in Eq. (30.36):

$$P_0(\tau' + d\tau') = P_0(\tau')\left[1 - \sum_{\mu=1}^{M} a_\mu d\tau'\right] \tag{30.37}$$

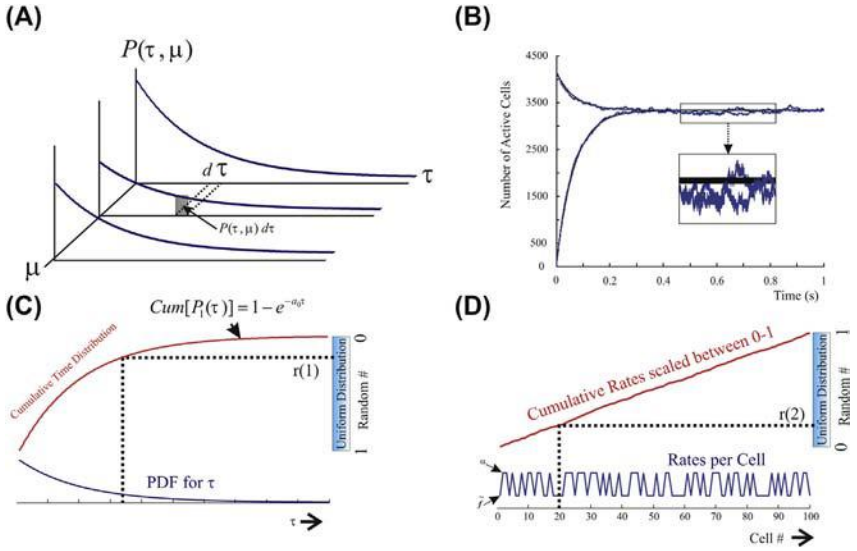


FIGURE 30.12 Gillespie's approach for Monte Carlo simulation of a master equation. (A) The joint PDF governing the reactions μ as a function of time τ. Note that the variable τ is continuous while μ is discrete. (B) Two examples of deterministic (*black lines*) and associated stochastic (*blue lines*) simulations superimposed. The two cases start from different initial values: a low and a high value. The inset is a detail of how the stochastic solutions fluctuate around the deterministic ones. (C) The procedure of selecting the time steps. Panel (D) depicts how to determine the cell to update at each time step.

which can be written as:

$$\frac{P_0(\tau' + d\tau') - P_0(\tau')}{P_0(\tau')} = -\sum_{\mu=1}^{M} a_\mu d\tau' = -a_0 d\tau' \tag{30.38}$$

where a_0 is the sum over all a_μ. Integration of Eq. (30.38) from 0 to τ results in

$$P_0(\tau) = e^{-a_0\tau} \tag{30.39}$$

Component (b) that reactions μ does occur in the interval between $t + \tau$ and $t + \tau + d\tau$ is given by $a_\mu d\tau$. Thus the probability $P(\tau, \mu)\, d\tau$ is the product of this result and Eq. (30.39), therefore

$$P(\tau, \mu)d\tau = a_\mu e^{-a_0\tau} d\tau \rightarrow P(\tau, \mu) = a_\mu e^{-a_0\tau} \tag{30.40}$$

For all values of μ (the reaction number) and τ (time steps > 0). Now we return to points **1** and **2** above, and state that

$$P(\tau, \mu) = \underbrace{P_1(\tau)}_{\substack{\text{When is the} \\ \text{next reaction?}}} \quad \underbrace{P_2(\mu|\tau)}_{\substack{\text{Given } \tau \text{ what} \\ \text{reaction is it?}}} \tag{30.41}$$

The first factor indicates the probability for the next reaction and it doesn't matter which one, therefore

$$P_1(\tau) = \sum_{\mu=1}^{M} P(\tau, \mu) = \sum_{\mu=1}^{M} a_\mu e^{-a_0\tau} = a_0 e^{-a_0\tau} \tag{30.42}$$

From substitution of Eqs. (30.40) and (30.42) in Eq. (30.41), we get

$$P_2(\mu|\tau) = \frac{P(\tau, \mu)}{\sum_{\mu=1}^{M} P(\tau, \mu)} = \frac{a_\mu}{a_0} \tag{30.43}$$

So now we have the probabilities for both τ and μ and the only question that remains is how we use this knowledge to implement a simulation. Usually the particular probability functions in Eq. (30.43) are not readily available in a simulation environment. But computers do have a random number generator based on a uniform distribution between 0 and 1. In a simulation, the uniform distribution can be used to generate values taken from any arbitrary PDF by using the uniform distribution to pick random points on the cumulative distribution of the target PDF. The procedures for τ and μ are shown in Fig. 30.12C and D. Because τ is a continuous variable we can compute the cumulative ($Cum[\ldots]$) function by integration of Eq. (30.42)

$$Cum[P_1(\tau)] = 1 - e^{-a_0\tau} \tag{30.44}$$

In order to determine both the values for τ and μ, we generate two random numbers $r(1)$ and $r(2)$, respectively. Now we can connect the random number $r(1)$ to the cumulative distribution of P_1 (Fig. 30.12C). However, since the cumulative function ranges between 0−1, we can also connect $r(1)$ directly to $\exp(-a_0\tau)$ (Fig. 30.12C; note that 0 and 1 are inverted in this case). If we now solve for τ, we get

$$r(1) = e^{-a_0\tau} \rightarrow \tau = \frac{1}{a_0} \log \frac{1}{r(1)} \tag{30.45}$$

Subsequently we create the cumulative distribution for the reactions and use $r(2)$ to update the system (Fig. 30.12D). This procedure of picking a time step and updating the system can be repeated for the entire simulation epoch. Examples of two simulations of our simplified system are depicted in Fig. 30.12B. For the sake of this

example (see MATLAB® script pr30_11), we update individual neurons at each time step. In homogeneous all-to-all networks, the simulation algorithm of the number of neurons active in each population may be simplified along the lines of Gillespie's original presentation for a well-mixed chemical system, since the downward and upward transition rates are identical for all neurons. This simplification will result in a faster computation since the time steps will increase (due to a smaller a_0) as compared to our approach in the example.

Because the master equation can be used to recover the deterministic rate equation, the stochastic approach is more general than the mean field models we discussed in Section 30.3. We recover the mean field equation by determining the mean number of active neurons n from the master equation via its definition (the discrete version of the Expectation, Chapter 3)

$$\langle n \rangle = \sum_{n=0}^{\infty} nP_n \tag{30.46}$$

In our case we assumed an infinite pool of nerve cells so there is no finite limitation to the number of active neurons. Therefore, the sum in Eq. (30.46) is taken from $0 \rightarrow \infty$. Other models have assumed a limited number of inactive and active nerve cells.

We now plug Eq. (30.46) into Eq. (30.34) (i.e., we multiply each term by n and sum over all n), and we get

$$\frac{d}{dt}\sum_{n=0}^{\infty} nP_n = \alpha \sum_{n=0}^{\infty} n(n+1)P_{n+1} - \alpha \sum_{n=0}^{\infty} n^2 P_n + f \sum_{n=0}^{\infty} nP_{n-1} - f \sum_{n=0}^{\infty} nP_n \tag{30.47}$$

This generates an expression for the time derivative of the mean (the term left of the equal sign).

We now substitute n' and n'' for $n-1$ and $n+1$, respectively, and we have

$$\begin{aligned} \frac{d}{dt}\sum_{n=0}^{\infty} nP_n = \alpha \sum_{n''=1}^{\infty} n''(n''-1)P_{n''} - \alpha \sum_{n=0}^{\infty} n^2 P_n \\ + f \sum_{n'=-1}^{\infty} (n'+1)P_{n'} - f \sum_{n=0}^{\infty} nP_n \end{aligned} \tag{30.48}$$

Note that we changed the range over which we sum in the first and third term. We now sum again from $0 \rightarrow \infty$. Note that we can make this change in the summation interval since P_{-1}=0 in the third term, and for

$n'' = 0$, $n''\,(n'' - 1)P\,n'' = 0$ in the first term. Furthermore, we expand the first and third terms, and get:

$$\frac{d}{dt}\sum_{n=0}^{\infty} nP_n = \alpha \sum_{n''=0}^{\infty} n''^2 P_{n''} - \alpha \sum_{n''=0}^{\infty} n'' P_{n''} - \alpha \sum_{n=0}^{\infty} n^2 P_n + f \sum_{n'=0}^{\infty} n' P_{n'} + f \sum_{n'=0}^{\infty} P_{n'} - f \sum_{n=0}^{\infty} nP_n$$

Terms 1 and 3, 4 and 6 cancel and terms 2 and 5 remain, i.e.:

$$\frac{d}{dt}\sum_{n=0}^{\infty} nP_n = -\alpha \sum_{n''=0}^{\infty} n'' P_{n''} + f \sum_{n'=0}^{\infty} P_{n'} \tag{30.49}$$

Using Eq. (30.46), and knowing that all probabilities must add up to one, we obtain an expression similar to the kinetic rate equation (Eq. 30.32)

$$\frac{d}{dt}\langle n\rangle = f - \alpha\langle n\rangle \tag{30.50}$$

Note that the mean of n is the variable A. Thus the mean number of active neurons in the master equation of the process produces the kinetic rate equation. Accordingly, the work by Benayoun et al. (2010a) and Wallace et al. (2011) showed that the mean activity generated by the stochastic model in which f depended on the network activity, corresponded to the mean field Wilson–Cowan model (Section 30.3.2). In addition, the simulations of their stochastic model made predictions about the fluctuations of the neural activity. These fluctuations followed avalanche statistics as observed in real neuronal networks or even in the scalp or intracranial EEG (Beggs and Plenz, 2003, 2004; Benayoun et al., 2010b).

A recently applied strategy to solve the master equation for a neuronal network model used the equation of motion of the generator function in operator notation

$$|G\rangle = \sum_{n=0}^{\infty} P_n |n\rangle \tag{30.51}$$

Here, we use the Dirac bra-ket notation (for a didactic overview of the notation see Shankar, 1994). The ket of G represents the probability

Pn for each state n represented by the ket of n in an infinite dimensional (Hilbert) space. The advantage of this approach is that the extensive mathematics developed for the operator notation can now be used to analyze neuronal networks (e.g., Buice and Cowan, 2009; Bressloff, 2009). Using Eq. (30.34), multiplying with the ket of n and summing over n, we get

$$\begin{aligned}\frac{d}{dt}\sum_{n=0}^{\infty}P_n|n\rangle = \alpha\sum_{n=0}^{\infty}(n+1)P_{n+1}|n\rangle - \alpha\sum_{n=0}^{\infty}nP_n|n\rangle \\ +f\sum_{n=0}^{\infty}P_{n-1}|n\rangle - f\sum_{n=0}^{\infty}P_n|n\rangle\end{aligned} \tag{30.52}$$

Now we use the equality in Eq. (30.51) and n' and n'' for $n-1$ and $n+1$, respectively, and rewrite the expression in Eq. (30.52)

$$\begin{aligned}\frac{d}{dt}|G\rangle = \alpha\sum_{n''=1}^{\infty}(n'')P_{n''}|n''-1\rangle - \alpha\sum_{n=0}^{\infty}nP_n|n\rangle \\ +f\sum_{n'=-1}^{\infty}P_{n'}|n'+1\rangle - f|G\rangle\end{aligned} \tag{30.53}$$

For the same reasons as we did above, we summate n' and n'' from 0 to ∞ and, in addition, we use the raising and lowering operators a^+ and a^-

$$a^+|n\rangle = |n+1\rangle \tag{30.54}$$

$$a^-|n\rangle = n|n-1\rangle \tag{30.55}$$

Note that the eigenvalues are 1 and n instead of the conventional $\sqrt{(n+1)}$ and $\sqrt{n}$.

Now we get

$$\frac{d}{dt}|G\rangle = \alpha\sum_{n''=0}^{\infty}P_{n''}a^-|n''\rangle - \alpha\sum_{n=0}^{\infty}P_na^+a^-|n\rangle + f\sum_{n'=0}^{\infty}P_{n'}a^+|n'\rangle - f|G\rangle \tag{30.56}$$

Using Eq. (30.51), we can simplify this to

$$\frac{d}{dt}|G\rangle = \alpha a^-|G\rangle - \alpha a^+a^-|G\rangle + fa^+|G\rangle - f|G\rangle \tag{30.57}$$

With a bit of algebra we may rewrite this, and recognize it as a linear equation of motion

$$\frac{d}{dt}|G\rangle = -\left(a^{+}-1\right)\left(\alpha a^{-}-f\right)|G\rangle \rightarrow \left|\dot{G}\right\rangle = -L|G\rangle \tag{30.58}$$

If we use the eigenfunctions $|j\rangle$ of L, to expand $|G\rangle$

$$|G\rangle = \sum_{j} G_j|j\rangle \tag{30.59}$$

For each eigenfunction there is a corresponding eigenvalue j of L, and therefore

$$\dot{G}_j = -jG_j \tag{30.60}$$

The solution of this differential equation is

$$G_j(t) = K_j e^{-jt} \tag{30.61}$$

Now we can retrieve the PDF by projecting G on bra-n, and using the expressions in Eqs. (30.59)–(30.61):

$$P_n(t) = \langle n|G(t)\rangle = \langle n|\sum_{j} G_j(t)|j\rangle = \sum_{j} K_j e^{-jt}\langle n|j\rangle \tag{30.62}$$

The Kj coefficients are determined from the initial distribution $P_n(0)$.

30.7 CONCLUDING REMARKS

In Chapter 29 and this chapter we have presented a variety of modeling approaches used in neuroscience while using the principles introduced in other parts of this text. In general, interest in and acceptance of modeling in neuroscience are growing (e.g., Abbott, 2008; Destexhe and Sejnowski, 2009). However, an important roadblock for the application of modeling is that, in part due to the complexity of the nervous system, there is little consensus amongst neuroscientists on what properties are to be considered critical in a theoretical approach. For this reason, traditionally, the neuroscientist prefers "realistic models" in which many experimental details are represented. Of course, in this sense, the skeptical neuroscientist is right that all models are wrong, i.e., no model fully captures reality, both by design and because some details are simply not known. To cite Rosenblueth and Wiener (1945), "the best material model for a cat is another, or preferably the

same cat." However, it may be obvious that such a "best material model" is not the most helpful tool for a theoretical study of specific aspects of cat behavior because it clearly contains too much detail. In many cases the choice of the properties in a model is a judgment (common sense) call. For example, if one models traffic-flow during rush hour, one probably wouldn't include some details such as the color of the cars, while including others such as the number of traffic lanes between suburbs and industrial zones. On the other hand, one would make the opposite assessment about these possible model parameters in a marketing model simulating consumer behavior for buying cars. Just like in physics, when modeling in biology, it is good to recall Einstein's dictum: "models should be as simple as possible, but not more so." (e.g., May, 2004). Due to simplifications, models do not include all possible parameters, but they can nonetheless be very useful because they allow us to make predictions and they may generate uncluttered insight into the critical components of the underlying mechanisms of the modeled process. The validation of having picked the right modeling parameters is obtained when the model appears to provide useful predictions: i.e., if the model provides correct predictions, one can conclude that the parameters of the model are apparently all you needed to include. Of course, all the above general modeling statements are also valid for modeling in neuroscience, whether we deal with models that describe the microscopic details of cellular function or ones that deal with the macroscopic networks of billions of nerve cells in the human brain.

Due to the necessary simplifications required in modeling, there are limitations in the models reviewed here; they neglect many aspects that play important roles. A critical part of the dynamics of neuronal networks is determined by synaptic plasticity, i.e., the change in network connection strength. As was also recognized by the artificial intelligence community, these changes may play a crucial role in the process of learning. In addition, they may also contribute to neurological diseases such as epilepsy. It was more than a half century ago that Hebb (1949) recognized this and postulated his law. Hebb's law describes how synaptic coupling strength depends on local activity. It is often stated in a simplified form as: "neurons that fire together wire together." Much later, an experimental approach has confirmed the existence of activity-dependent plasticity at the level of the coupling between nerve cells, i.e., the synapse (e.g., Bi and Poo, 2001; Woodin et al., 2003). These reports show that plastic changes depend on the timing of the action potential (spike) activity of the pre- and postsynaptic neurons and is therefore also called spike-time-dependent plasticity (STDP). STDP has been used in simulations to find appropriate synaptic strength (e.g., Izhikevich and Edelman, 2008), effectively as a parameter estimation

technique. It is fairly obvious that if connections between excitatory neurons that fire together also wire together (i.e., get strengthened) represents a positive feedback mechanism, it may lead to instability and possibly seizure-like hyperactivity (Beenhakker and Huguenard, 2009). Therefore, under normal physiological conditions, there must be additional homeostatic mechanisms counteracting this positive feedback. Although plasticity is an important aspect of network behavior, the neuroscience models reviewed here did not include it as one of the properties. Plasticity and many other aspects involved in nervous system function, such as the role of glia (e.g., Volman et al., 2012), remain to be investigated.

Finally, one important conclusion is that there is not "a best approach" when modeling neural function. For example, nonlinear models can help us to understand specifics such as sudden transitions, but linear ones can be more suitable to analyze properties such as subthreshold oscillations. Complex models can be studied with simulations, they are more complete since they can deal with more parameters than the more abstract models that are mathematically tractable. However, both have a place and they can be complementary. Analysis of simplified mathematical models can generate fundamental insight in spite of the fact that they lack detail. Simulation can be too complex to directly generate such fundamental insight into the interaction of the ongoing processes, but they have the advantage of being closer to reality. This property may appeal to the experimenter since it can produce data with a temporal-spatial resolution that isn't (yet) feasible experimentally. In this context, the ultimate goal in understanding the nervous system, in both physiological and pathological states, is to create a framework of experimental models, detailed simulations, and mathematical formalisms that allow us to understand and predict dynamics of single cells as well as network activities.

APPENDIX 30.1 LINEARIZATION OF THE WILSON–COWAN EQUATIONS

To study resonance phenomena we will consider the effect on E and I by small sinusoidal perturbations (e.g., evoked by electrical stimulation) about an equilibrium state. To accomplish this, we can linearize the Wilson–Cowan equations. Here, we will assume that both the excitatory and inhibitory populations receive the same stimulus current: $P = Q$.

Further, simplifying the sigmoid curves S_e and S_i to lines with slopes α and β, the linearized version of Eqs. (30.5) and (30.6) become:

$$\dot{E} = -\frac{E}{\tau_e} + \frac{(k_e - r_e E)}{\tau_e}\alpha(c_1 E - c_2 I + P) \text{ and}$$

$$\dot{I} = -\frac{I}{\tau_i} + \frac{(k_i - r_i I)}{\tau_i}\beta(c_3 E - c_4 I + P)$$

Rewriting these results and using $\alpha_\tau = \alpha/\tau_e$ and $\beta_\tau = \beta/\tau_i$, we get:

$$\dot{E} = f(E,I,P) = -\frac{E}{\tau_e} + k_e\alpha_\tau c_1 E - k_e\alpha_\tau c_2 I + k_e\alpha_\tau P - r_e\alpha_\tau c_1 E^2 + r_e\alpha_\tau c_2 EI - r_e\alpha_\tau EP \text{ and}$$

$$\dot{I} = g(E,I,P) = -\frac{I}{\tau_i} + k_i\beta_\tau c_3 E - k_i\beta_\tau c_4 I + k_i\beta_\tau P - r_i\beta_\tau c_3 EI + r_i\beta_\tau c_4 I^2 - r_i\beta_\tau IP$$

We now further linearize and consider small perturbations $\delta E, \delta I, \delta P$ about an equilibrium indicated by *:

$$\delta\dot{E} = \left(\frac{\partial f}{\partial E}\right)_* \delta E + \left(\frac{\partial f}{\partial I}\right)_* \delta I + \left(\frac{\partial f}{\partial P}\right)_* \delta P \text{ and}$$

$$\delta\,\dot{I} = \left(\frac{\partial g}{\partial E}\right)_* \delta E + \left(\frac{\partial g}{\partial I}\right)_* \delta I + \left(\frac{\partial g}{\partial P}\right)_* \delta P$$

The partial derivatives determined at equilibrium * are:

$$\varepsilon_E = \left(\frac{\partial f}{\partial E}\right)_* = \left(-\frac{1}{\tau_e} + k_e\alpha_\tau c_1 - 2r_e\alpha_\tau c_1 E + r_e\alpha_\tau c_2 I - r_e\alpha_\tau P\right)_*,$$

$$\varepsilon_I = \left(\frac{\partial f}{\partial I}\right)_* = (-k_e\alpha_\tau c_2 + r_e\alpha_\tau c_2 E)_*, \varepsilon_P = \left(\frac{\partial f}{\partial P}\right)_* = (k_e\alpha_\tau - r_e\alpha_\tau E)_*$$

$$\gamma_E = \left(\frac{\partial g}{\partial E}\right)_* = (k_i\beta_\tau c_3 - r_i\beta_\tau c_3 I)_*,$$

$$\gamma_I = \left(\frac{\partial g}{\partial I}\right)_* = \left(-\frac{1}{\tau_i} - k_i\beta_\tau c_4 - r_i\beta_\tau c_3 E + 2r_i\beta_\tau c_4 I - r_i\beta_\tau P\right)_*,$$

$$\gamma_P = \left(\frac{\partial g}{\partial P}\right)_* = (k_i\beta_\tau - r_i\beta_\tau I)_*$$

Combining these expressions and substituting e and i for δE and δI, and making the perturbation δP sinusoidal $p\sin\omega t$ (where amplitude p is a small value), we can simplify the notation to:

$$\dot{e} = \varepsilon_E e + \varepsilon_I i + \varepsilon_P p \sin \omega t \quad \text{and} \tag{A30.1-1}$$

$$\dot{i} = \gamma_E e + \gamma_I i + \gamma_P p \sin \omega t \tag{A30.1-2}$$

These two first-order differential equations can also be written as a single second-order one. Taking the derivative of Eq. (A30.1-1) and substituting the expression in (A30.1-2) for $\dot{i}$ gives:

$$\ddot{e} = \varepsilon_E \dot{e} + \varepsilon_I(\gamma_E e + \gamma_I i + \gamma_P p \sin \omega t) + \varepsilon_P \omega p \cos \omega t$$

Using Eq. (A30.1-1) we can substitute $i = (\dot{e} - \varepsilon_E e - \varepsilon_P p \sin \omega t)/\varepsilon_I$, and we get the standard expression for a second-order differential equation with a forcing term:

$$\ddot{e} - \underbrace{(\varepsilon_E + \gamma_I)}_{b}\,\dot{e} + \underbrace{(\varepsilon_E \gamma_I - \varepsilon_I \gamma_E)}_{c}\,e = \underbrace{\varepsilon_P p \frac{d \sin \omega t}{dt} + (\varepsilon_I \gamma_P - \varepsilon_P \gamma_I) p \sin \omega t}_{\text{forcing term}}$$

This equation is again of the same form as the equation that governs a parallel RCL circuit as shown in Fig. 29.3. The second-order ODE governing this RCL circuit is Eq. (29.18). This equation is also very similar to the network equation by Jirsa and Haken (1997) in which the spatial derivative is set to zero (Section 30.5.4). Thus the linearized network model is capable of resonance behavior, as described in Section 29.2.1.4. Since we deal with a linearized version of a nonlinear system, unstable solutions of the linearized case may relate to bifurcations in the full nonlinear version of the model.

APPENDIX 30.2 ADJUSTING WEIGHTS BY BACKPROPAGATION OF THE ERROR

In this appendix we discuss the extension of a neural-net to include hidden layers (Fig. 30.6D) and the strategy to employ supervised learning to adjust weights across the entire network, including those of a hidden layer. For more background on this procedure see, for example, Pao (1989). In this example, we use a sigmoid function rather than a step function to convert the node's input to its output (Fig. 30.6C). Let us

start to describe what happens at the output layer; here we have the output o_k at each node satisfying:

$$o_k = F\{i_k\} = F\left\{\sum_j w_{jk} o_j\right\} = \frac{1}{1 + e^{-\sum_j w_{jk} o_j + \theta_k}} \tag{A30.2-1}$$

The term $\sum_j w_{jk} o_j$ is the input i_k to the node, the weighted linear sum of the output from the hidden layer, and $F\{i_k\}$ is a nonlinear conversion of the input of the output layer to the output $o_k$of the output layer, i.e., the sigmoid function. Using the desired/target output of the node t_k, enables us to define the error E at the output as:

$$E = \frac{1}{2} \sum_k (t_k - o_k)^2 \tag{A30.2-2}$$

Since our goal is to minimize this error by changing the weights, we use a correction term proportional to the gradient of the error with respect to weight, i.e., the **derivative** $\frac{\partial E}{\partial w_{jk}}$ for updating the weight w_{jk}:

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} \tag{A30.2-3}$$

Here Δw_{jk} is the amount that the weight is updated and η is a constant, or a value that decreases with each iteration of the update procedure. Assuming that this procedure converges to a solution, the update process can be halted when Δw_{jk} is zero or almost zero. Using the **chain rule** twice, we may rewrite the partial differential in Eq. (A30.2-3) as an expression of the following three terms:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial i_k} \frac{\partial i_k}{\partial w_{jk}} \tag{A30.2-4}$$

Looking at the last of the three terms, and using the term $\sum_j w_{jk} o_j$ for input i_k, we have:

$$\frac{\partial i_k}{\partial w_{jk}} = \frac{\partial \left(\sum_j w_{jk} o_j \right)}{\partial w_{jk}} = o_j \tag{A30.2-5a}$$

The middle term of the three in Eq. (A30.2-4), while using Eq. (A30.2-1) becomes:

$$\frac{\partial o_k}{\partial i_k} = \frac{\partial F\{i_k\}}{\partial i_k} = F'_k \tag{A30.2-5b}$$

Using Eq. (A30.2-2), the first term of Eq. (A30.2-4) evaluates to:

$$\frac{\partial E}{\partial o_k} = -(t_k - o_k) \tag{A30.2-5c}$$

Now combining Eqs. (A30.2-3)–(A30.2-5), we have:

$$\Delta w_{jk} = -\eta \underbrace{\frac{\partial E}{\partial o_k}}_{-(t_k - o_k)} \underbrace{\frac{\partial o_k}{\partial i_k}}_{F'_k} \underbrace{\frac{\partial i_k}{\partial w_{jk}}}_{o_j} = \eta(t_k - o_k)F'_k o_j \tag{A30.2-6}$$

At each iteration of the weight update procedure, all of the terms in Eq. (A30.2-6) can be computed and employed to update the weights between the hidden layer and output layer. Similar reasoning can be applied for the weights between the input layer and hidden layer. This leads to a similar expression as shown in Eq. (A30.2-6), i.e.,:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial o_j} \underbrace{\frac{\partial o_j}{\partial i_j}}_{F'_j} \underbrace{\frac{\partial i_j}{\partial w_{ij}}}_{o_i} = \eta \left(-\frac{\partial E}{\partial o_j} \right) F'_j o_i \tag{A30.2-7}$$

The problem we encounter here is that the error term $\frac{\partial E}{\partial o_j}$ refers to the output error of the hidden layer! Since we don't know what the target output at the hidden layer is supposed to be, we have a problem… Now note that the error we are interested in is at the output of the output layer (and, again, **not** at the output of the hidden layer). Thus, in order to be able to evaluate the error/correction term $-\frac{\partial E}{\partial o_j}$, we will rewrite it in terms of the output error of the output layer (i.e., the error we care about). This allows us to use the error at the output layer for correction of the weights of the hidden layer. This procedure is called **backpropagation** of the error. Using the **chain rule** again we get:

$$-\frac{\partial E}{\partial o_j} = -\sum_k \frac{\partial E}{\partial i_k} \frac{\partial i_k}{\partial o_j} = \sum_k \left(-\frac{\partial E}{\partial i_k} \right) \frac{\partial \sum_n w_{nk} o_n}{\partial o_j} = \sum_k \left(-\frac{\partial E}{\partial i_k} \right) w_{jk} \tag{A30.2-8}$$

Next we use Eqs. (A30.2-5b) and (A30.2-5c) and rewrite the partial derivative $\frac{\partial E}{\partial i_k}$ as follows $\frac{\partial E}{\partial i_k} = \underbrace{\frac{\partial E}{\partial o_k}}_{-(t_k - o_k)} \underbrace{\frac{\partial o_k}{\partial i_k}}_{F'_k} = -(t_k - o_k)F'_k$. Substituting this into Eq. (A30.2-8) gives $-\frac{\partial E}{\partial o_j} = \sum_k (t_k - o_k)F'_k w_{jk}$. This allows us to write the update for the weight of the hidden layer Δw_{ij} in terms of the output errors and properties of the input and output layers:

$$\Delta w_{ij} = \eta \left(\sum_k (t_k - o_k)F'_k w_{jk} \right) F'_j o_i \tag{A30.2-9}$$

Eq. (A30.2-9) allows us to employ backpropagation of the error at the output, i.e., $\sum_k (t_k - o_k)$, to correct the weights w_{ij} of the hidden layer.

We have a special case when using a sigmoid (Fig. 30.6C) for all the nonlinear relationships, all with unity slope and thresholds at zero, e.g., Eq. (A30.2-1) now becomes:

$$o_k = F\{i_k\} = F\left\{ \sum_j w_{jk} o_j \right\} = \frac{1}{1 + e^{-\sum_j w_{jk} o_j}} \tag{A30.2-10}$$

Note that a special case is the solution of the ODE $\frac{do_k}{di_k} = o_k(1 - o_k)$, with $o_k(0) = 1/2$.

When applying this approach, the network is trained on the training set and subsequently tested on a test set. Within the training set the weights in the network are adjusted by presenting all patterns p to the network for a number of iterations, and updating the weights using the sum of weight changes across all patterns for each of the iterations:

$$\Delta w_{ij} = \sum_p \Delta_p w_{ij} \tag{A30.2-11}$$

EXERCISES

30.1 Modify pr30_4 to simulate spiral sink and limit cycle behavior of the Wilson–Cowan equations.

30.2 Use a modified version of pr30_5 to obtain evidence that the perceptron **cannot** be used for implementing an XOR

(eXclusive OR) function that is only active if one of its inputs is 1 while the other input is 0; i.e., (0 1) and (1 0) give a $+1$ at the output while (0 0) and (1 1) result in an output of -1.

30.3 In the model of Lopes da Silva et al. (1974), rederive Eq. (30.13) using the alpha function (Ate^{-at}) instead of the dual exponential function for the synaptic unit impulse response.

30.4 Modify pr30_11 to simulate a population with different size (N_cells). What effect do you observe?

References

Abbott, L.F., 2008. Theoretical neuroscience rising. Neuron 60, 489–495.

Beenhakker, M.P., Huguenard, J.R., 2009. Neurons that fire together also conspire together: is normal sleep circuitry hijacked to generate epilepsy? Neuron 62, 612–632.

Beggs, J.M., Plenz, D., 2003. Neuronal avalanches in neocortical circuits. J. Neurosci. 23, 11167–11177.

Beggs, J.M., Plenz, D., 2004. Neuronal avalanches are diverse and precise activity patterns that are stable for many hours in cortical slice cultures. J. Neurosci. 24, 5216–5229.

Benayoun, M., Cowan, J.D., van Drongelen, W., Wallace, E., 2010a. Avalanches in a stochastic model of spiking neurons. PLoS Comput. Biol. 6, e1000846.

Benayoun, B., Kohrman, M., Cowan, J., van Drongelen, W., 2010b. EEG, avalanches, and temporal correlations. J. Clin. Neurophysiol. 27, 458–464.

Bi, G., Poo, M., 2001. Synaptic modification by correlated activity: Hebb's postulate revisited. Annu. Rev. Neurosci. 24, 139–166.

Breakspear, M., Heitmann, S., Daffertshofer, A., 2010. Generative models of cortical oscillations: neurobiological implications of the Kuramoto model. Front. Hum. Neurosci. 4, 190.

Bressloff, P.C., 2009. Stochastic neural field theory and the system-size expansion. SIAM J. Appl. Math. 70, 1488–1521.

Buice, M.A., Cowan, J.D., 2009. Statistical mechanics of the neocortex. Prog. Biophys. Mol. Biol. 99, 53–86.

Chang, H.-J., Freeman, W.J., Burke, B.C., 1998. Optimization of olfactory model in software to give $1/f$ power spectra reveals numerical instabilities in solutions governed by aperiodic (chaotic) attractors. Neural Networks 11, 449–466.

Dayan, P., Abbott, L.F., 2001. Theoretical Neuroscience Computational and Mathematical Modeling of Neural Systems. MIT Press, Cambridge, MA.

De Schutter, E., Bower, J.M., 1994. An active membrane model of the cerebellar Purkinje cell: II. Simulation of synaptic responses. J. Neurophysiol. 71, 401–419.

Destexhe, A., Sejnowski, T.J., 2009. The Wilson–Cowan model, 36 years later. Biol. Cybern. 101, 1–2.

Douglas, R.J., Martin, K.A.C., 1991. A functional microcircuit for cat visual cortex. J. Physiol. 440, 735–769.

Doya, K., Ishii, S., Pouget, A., Rao, R.P.N., 2007. Bayesian Brain: Probabilistic Approaches to Neural Coding. MIT Press, Cambridge, MA.

Eliasmith, C., Stewart, T.C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., Rasmussen, D., 2012. A large-scale model of the functioning brain. Science 338, 1202–1205.

Foss, J., Longtin, A., Mensour, B., Milton, J., 1996. Multistability and delayed recurrent loops. Phys. Rev. Lett. 76, 708–711.

Freeman, W.J., 1987. Simulation of chaotic EEG patterns with a dynamic model of the olfactory system. Biol. Cybern. 56, 139–150.

Gillespie, D.T., 1976. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. J. Comput. Phys. 22, 403–434.
Gillespie, D., 1977. Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. 81, 2340–2361.
Giorno, V., Nobile, A.G., Ricciardi, L.M., 1997. Single neuron's activity: on certain problems of modeling and interpretation. Biosystems 40, 65–74.
Grimbert, F., Faugeras, O., 2006. Bifurcation analysis of Jansen's Neural Mass Model. Neural Comput. 18, 3052–3068.
Hebb, D.O., 1949. The Organization of Behavior. Wiley & Sons, New York.
Hindriks, R., van Putten, M.J., 2012. Meanfield modeling of propofol-induced changes in spontaneous EEG rhythms. Neuroimage 60, 2323–2334.
Hodgkin, A.L., Huxley, A.F., 1952. A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. 117, 500–544.
Hopfield, J.J., 1982. Neurons with graded responses have collective computational properties like those of two-state neurons. PNAS 1, 3088–3092.
Ising, E., 1925. Beitrag zur Theorie des Ferromagnetismus. Z. Physik. 31, 253–258.
Izhikevich, E.M., Edelman, G.M., 2008. Large-scale model of mammalian thalamocortical system. PNAS 105, 3593–3598.
Jansen, B.H., Rit, V.G., 1995. Electroencephalogram and visual evoked potential generation in a mathematical model of coupled cortical columns. Biol Cybern. 73, 357–366.
Jirsa, V.K., Haken, H., 1997. A derivation of a macroscopic field theory of the brain from the quasi-microscopic neural dynamics. Physica D 99, 503–526.
LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444.
Liley, D.T., Bojak, I., 2005. Understanding the transition to seizure by modeling the epileptiform activity of general anesthetic agents. J. Clin. Neurophysiol. 22, 300–313.
Little, W.A., 1974. The existence of persistent states in the brain. Math. Biosci. 19, 101–120.
Lopes da Silva, F.H., Hoeks, A., Smits, H., Zetterberg, L.H., 1974. Model of brain rhythmic activity. Biol. Cybern. 15, 27–37.
Lytton, W.W., Sejnowski, T.J., 1991. Simulations of cortical pyramidal neurons synchronized by inhibitory interneurons. J. Neurophysiol. 66, 1059–1079.
Markram, H., 2006. The blue brain project. Nat. Rev. Neurosci. 7, 153–160.
May, R.M., 2004. Uses and abuses of mathematics in biology. Science 303, 790–793.
McComb, W.D., 2004. Renormalization Methods. Clarendon Press, Oxford.
McCormick, D.A., Huguenard, J.R., 1992. A model of the electrophysiological properties of thalamocortical relay neurons. J. Neurophysiol. 68, 1384–1400.
McCulloch, W.S., Pitts, W.H., 1943. A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. 5, 115–133.
Milton, J., 2000. Epilepsy: multistability in a dynamic disease. In: Walleczek, J. (Ed.), Self-organized Biological Dynamics & Nonlinear Control. Cambridge University Press, New York, pp. 374–386.
Milton, J., 2012. Neuronal avalanches, epileptic quakes and other transient forms of neurodynamics. Eur. J. Neurosci. 36, 2156–2163.
Nunez, P.L., 1974. The brain wave equation: a model for the EEG. Math. Biosci. 21, 219–291.
Nunez, P.L., 1995. Neocortical Dynamics and Human EEG Rhythms. Oxford University Press, New York.
Nunez, P.L., Srinivasan, R., 2006a. A theoretical basis for standing and traveling brain waves measured with human EEG with implications for an integrated consciousness. Clin. Neurophysiol. 117, 2424–2435.
Nunez, P.L., Srinivasan, R., 2006b. Electrical Fields of the Brain, second ed. Oxford University Press, New York.
Olmi, S., Livi, R., Politi, A., Torcini, A., 2010. Collective oscillations in disordered neural networks. Phys. Rev. E 81, 046119.

Pao, Y.-H., 1989. Adaptive Pattern Recognition and Neural Networks. Addison-Wesley, Reading, MA.
Rieke, F., Warland, D., de de Ruyter van Steveninck, R., Bialek, W.W., 1997. Spikes: Exploring the Neural Code. MIT Press, Cambridge, MA.
Rodriguez, R., Tuckwell, H.C., 1996. Statistical properties of stochastic nonlinear dynamical models of single spiking neurons and neural networks. Phys. Rev. E 54, 5585–5590.
Rolls, E.T., Deco, G., 2010. The Noisy Brain: Stochastic Dynamics as a Principle of Brain Function. Oxford University Press, Oxford, UK.
Rosenblueth, A., Wiener, N., 1945. The role of models in science. In: Philosophy of Science, vol. 12. The University of Chicago Press, Chicago IL, pp. 316–321.
Rudolph, M., Destexhe, A., 2001. Correlation detection and resonance in neural systems with distributed noise sources. Phys. Rev. Lett. 86, 3662–3665.
Shankar, R., 1994. Principles of Quantum Mechanics. Springer, New York.
Tattini, L., Olmi, S., Torcini, A., 2012. Coherent periodic activity in excitatory Erdos-Renyi neural networks: the role of network connectivity. Chaos 22, 023133.
Thom, R., 1975. Structural Stability and Morphogenesis: An Outline of a General Theory of Models. WA Benjamin, Inc., Reading, MA.
Traub, R.D., Miles, R., 1991. Neuronal Networks of the Hippocampus, vol. 777. Cambridge University Press, Cambridge.
Traub, R.D., Contreras, D., Cunningham, M.O., Murray, H., LeBeau, F.E., Roopun, A., Bibbig, A., Wilent, W.B., Higley, M.J., Whittington, M.A., 2005. Single-column thalamocortical network model exhibiting gamma oscillations, sleep spindles, and epileptogenic bursts. J. Neurophysiol. 93, 2194–2232.
van Albada, S.J., Robinson, P.A., 2009. Mean-field modeling of the basal ganglia-thalamocortical system. I Firing rates in healthy and parkinsonian states. J. Theor. Biol. 257, 642–663.
van Drongelen, W., 2013. Modeling neural activity. Biomathematics, 871472.
van Drongelen, W., Lee, H.C., Koch, H., Hereld, M.H., Elsen, F., Chen, Z., Stevens, R.L., 2005. Emergent epileptiform activity in neural networks with weak excitatory synapses. IEEE Trans. Neur. Sys. Rehab. 13, 236–241.
van Drongelen, W., Koch, H., Elsen, F.P., Lee, H.C., Mrejeru, A., Doren, E., Marcuccilli, C.J., Hereld, M., Stevens, R.L., Ramirez, J.M., 2006. The role of persistent sodium current in bursting activity of mouse neocortical networks in vitro. J. Neurophysiol. 96, 2564–2577.
van Drongelen, W., Lee, H.C., Stevens, R.L., Hereld, M., 2007. Propagation of seizure-like activity in a model of neocortex. J. Clin. Neurophysiol. 24, 182–188.
Van Kampen, N.G., 1992. Stochastic Processes in Physics and Chemistry. Elsevier, Amsterdam.
van Rotterdam, A., Lopes da Silva, F.H., van den Ende, J., Viergever, M.A., Hermans, A.J., 1982. A model of the spatial-temporal characteristics of the alpha rhythm. Bull. Math. Biol. 44, 283–305.
van Vreeswijk, C.A., 1996. Partial synchronization of populations of pulse-coupled oscillators. Phys. Rev. E54, 5522.
van Vreeswijk, C.A., Abbott, L.F., Ermentrout, G.B., 1994. Inhibition, not excitation, synchronizes coupled neurons. J. Comp. Neurosci. 1, 303–313.
Verveen, A.A., DeFelice, L.J., 1974. Membrane noise. Prog. Biophys. Mol. Biol. 28, 189–265.
Visser, S., Meijer, H.G.E., Lee, H.C., van Drongelen, W., van Putten, M.J.A.M., van Gils, S.A., 2010. Comparing epileptiform behavior of meso-scale detailed models and population models of neocortex. J. Clin. Neurophysiol. 27, 471–478.
Visser, S., Meijer, H.G.E., van Putten, M.J.A.M., van Gils, S.A., 2012. Analysis of stability and bifurcations of fixed points and periodic solutions of a lumped model of neocortex with two delays. J. Math. Neurosci. 2, 8 (Epub ahead of print).

Volman, V., Bazhenov, M., Sejnowski, T.J., 2012. Computational models of neuron-astrocyte interaction in epilepsy. Front. Comput. Neurosci. 6, 1–10.
Wallace, E., Benayoun, M., van Drongelen, W., Cowan, J.D., 2011. Emergent oscillations in networks of stochastic spiking neurons. PLoS One 6 (5), e14804.
Wendling, F., Chauvel, P., 2008. Transition to ictal activity in temporal lobe epilepsy: insights from macroscopic models. In: Soltesz, I., Staley, K. (Eds.), Computational Neuroscience in Epilepsy. Elsevier, Amsterdam, pp. 356–387.
Wolfram, S. (Ed.), 1986. Theory and Application of Cellular Automata. World Scientific, Singapore.
Woodin, M.A., Ganguly, K., Poo, M.M., 2003. Coincident pre- and postsynaptic activity modifies GABAergic synapses by postsynaptic changes in Cl^- transporter activity. Neuron 39, 807–820.
Wilson, H.R., Cowan, J.D., 1972. Excitatory and inhibitory interactions in localized populations of model neurons. Biophys. J. 12, 1–24.
Wilson, H.R., Cowan, J.D., 1973. A mathematical theory of the functional dynamics of nervous tissue. Biol. Cybern. 13, 55–80.

Index

'*Note*: Page numbers followed by "f" indicate figures, "t" indicate tables and "b" indicate boxes.'

D

E

K

L

N

O

P

Q

R

Signal Processing for Neuroscientists

Second Edition

Wim van Drongelen

Signal Processing for Neuroscientists, Second Edition provides an introduction for those with a modest understanding of algebra, trigonometry, and calculus. Combining the previous edition (2006), its more advanced companion volume (2010), and a set of new related topics, it features eight new or extended chapters and a new set of exercises developed by the author over the past decade. The combination of signal processing and a new, strong modeling component results in a significant didactic resource for the neuroscientist, beginning with the fundamentals and moving to practical applications of numerical recipes and simulations of neural activity. This book presents a fundamental and uncluttered background for students of neuroscience, the nonspecialist scientist, or engineer. It is a starting point to develop applications and allows the reader to interpret and master the field of signal processing and modeling.

Key Features

- Includes an introduction to biomedical signals, noise characteristics, recording techniques, fundamentals of linear and nonlinear systems analysis, and multichannel analysis that were previously split into two separate volumes
- Features new chapters on modeling fundamentals, applications to modeling neuronal and network activity, and novel introductions to the Kalman filter, medical imaging, and multitaper power spectrum estimation
- Includes practical examples of algorithm development and implementation in MATLAB® and a new set of exercises with each chapter
- Features a companion website that hosts the MATLAB® scripts, data files, answers to selected exercises, figures, and (links to) video lectures

Neuroscience

ISBN 978-0-12-810482-8

9 780128 104828

ELSEVIER

ACADEMIC PRESS

An imprint of Elsevier

elsevier.com/books-and-journals