Preface

In this book I have only made up a bunch
of other men's flowers, providing of my own
only the string that ties them together.
M. de Montaigne (1533–1592)
French essayist

Although it is hardly possible to keep up with advances in technology, it is reassuring to know that in science and engineering, development and innovation are possible through a solid understanding of basic principles. The theory of signals and systems is one of those fundamentals, and it will be the foundation of much research and development in engineering for years to come. Not only engineers will need to know about signals and systems—to some degree everybody will. The pervasiveness of computers, cell phones, digital recording, and digital communications will require it.

Learning as well as teaching signals and systems is complicated by the combination of mathematical abstraction and concrete engineering applications. Mathematical sophistication and maturity in engineering are needed. Thus, a course in signals and systems needs to be designed to nurture the students' interest in applications, but also to make them appreciate the significance of the mathematical tools. In writing this textbook, as in teaching this material for many years, the author has found it practical to follow Einstein's recommendation that "Everything should be made as simple as possible, but not simpler," and Melzak's [47] dictum that "It is downright sinful to teach the abstract before the concrete." The aim of this textbook is to serve the students' needs in learning signals and systems theory as well as to facilitate the teaching of the material for faculty by proposing an approach that the author has found effective in his own teaching.

We consider the use of MATLAB, an essential tool in the practice of engineering, of great significance in the learning process. It not only helps to illustrate the theoretical results but makes students aware of the computational issues that engineers face in implementing them. Some familiarity with MATLAB is beneficial but not required.

LEVEL

The material in this textbook is intended for courses in signals and systems at the junior level in electrical and computer engineering, but it could also be used in teaching this material to mechanical engineering and bioengineering students and it might be of interest to students in applied mathematics. The "student-friendly" nature of the text also makes it useful to practicing engineers interested in learning or reviewing the basic principles of signals and systems on their own. The material is organized so that students not only get a solid understanding of the theory—through analytic examples as well as software examples using MATLAB—and learn about applications, but also develop confidence and proficiency in the material by working on problems.

The organization of the material in the book follows the assumption that the student has been exposed to the theory of linear circuits, differential equations, and linear algebra, and that this material will be followed by courses in control, communications, or digital signal processing. The content is guided by the goal of nurturing the interest of students in applications, and of assisting them in becoming more sophisticated mathematically. In teaching signals and systems, the author has found that students typically lack basic skills in manipulating complex variables, in understanding differential equations, and are not yet comfortable with basic concepts in calculus. Introducing discrete-time signals and systems makes students face new concepts that were not explored in their calculus courses, such as summations, finite differences, and difference equations. This text attempts to fill the gap and nurture interest in the mathematical tools.

APPROACH

In writing this text, we have taken the following approach:

1. The material is divided into three parts: introduction, theory and applications of continuous-time signals and systems, and theory and applications of discrete-time signals and systems. To help students understand the connection between continuous- and discrete-time signals and systems, the connection between infinitesimal and finite calculus is made in the introduction part, together with a motivation as to why complex numbers and functions are used in the study of signals and systems. The treatment of continuous- and discrete-time signals and systems is then done separately in the next two parts; combining them is found to be confusing to students. Likewise, the author believes it is important for students to understand the connections and relevance of each of the transformations used in the analysis of signals and systems so that these transformations are seen as a progression rather than as disconnected methods. Thus, the author advocates the presentation of the Laplace analysis followed by the Fourier analysis, and the Z-transform followed by the discrete Fourier, and capping each of these topics with applications to communications, control, and filtering. The mathematical abstraction and the applications become more sophisticated as the material unfolds, taking advantage as needed of the background on circuits that students have.
2. An overview of the topics to be discussed in the book and how each connects with some basic mathematical concepts—needed in the rest of the book—is given in Chapter 0 (analogous to the ground floor of a building). The emphasis is in relating summations, differences, difference equations, and sequence of numbers with the calculus concepts that the students are familiar with, and in doing so providing a new interpretation to integrals, derivatives, differential equations, and functions of time. This chapter also links the theory of complex numbers and functions to vectors and to phasors learned in circuit theory. Because we strongly believe that the material in this chapter should be covered before beginning the discussion of signals and systems, it is not relegated to an appendix but placed at the front of the book where it cannot be ignored. A soft introduction to MATLAB is also provided in this chapter.
3. A great deal of effort has been put into making the text "student friendly." To make sure that the student does not miss some of the important issues presented in a section, we have inserted well-thought-out remarks—we want to minimize the common misunderstandings we have observed from our students in the past. Plenty of analytic examples with different levels of complexity are given to illustrate issues. Each chapter has a set of examples in MATLAB, illustrating topics presented in the text or special issues that the student should know. The MATLAB code is given so that students can learn by example from it. To help students follow the mathematical derivations, we provide extra steps whenever necessary and do not skip steps that are necessary in the understanding of a derivation. Summaries of important issues are boxed and concepts and terms are emphasized to help students grasp the main points and terminology.
4. Without any doubt, learning the material in signals and systems requires working analytical as well as computational problems. It is important to provide problems of different levels of complexity to exercise not only basic problem-solving skills, but to achieve a level of proficiency and mathematical sophistication. The problems at the end of the chapter are of different types, some to be done analytically, others using

MATLAB, and some both. The repetitive type of problem was avoided. Some of the problems explore issues not covered in the text but related to it. The MATLAB problems were designed so that a better understanding of the theoretical concepts is attained by the student working them out.

5. We feel two additional features would be beneficial to students. One is the inclusion of quotations and footnotes to present interesting ideas or historical comments, and the other is the inclusion of sidebars that attempt to teach historical or technical information that students should be aware of. The theory of signals and systems clearly connects with mathematics and a great number of mathematicians have contributed to it. Likewise, there is a large number of engineers who have contributed significantly to the development and application of signals and systems. All of them need to be recognized for their contributions, and we should learn from their experiences.
6. Finally, other features are: (1) the design of the index of the book so that it can be used by students to find definitions, symbols, and MATLAB functions used in the text; and (2) a list of references to the material.

CONTENT

The core of the material is presented in the second and third part of the book. The second part of the book covers the basics of continuous-time signals and systems and illustrates their application. Because the concepts of signals and systems are relatively new to students, we provide an extensive and complete presentation of these topics in Chapters 1 and 2. The presentation in Chapter 1 goes from a very general characterization of signals to very specific classes that will be used in the rest of the book. One of the aims is to familiarize students with continuous-time as well as discrete-time signals so as to avoid confusion in their processing later on—a common difficulty encountered by students. Chapter 1 initiates the representation of signals in terms of basic signals that will be easily processed later with the transform methods. Chapter 2 introduces the general concept of systems, in particular continuous-time systems. The concepts of linearity, time invariance, causality, and stability are introduced in this chapter, trying as much as possible to use the students' background in circuit theory. Using linearity and time invariance, the computation of the output of a continuous-time system using the convolution integral is introduced and illustrated with relatively simple examples. More complex examples are treated with the Laplace transform in the following chapter.

Chapter 3 covers the basics of the Laplace transform and its application in the analysis of continuous-time signals and systems. It introduces the student to the concept of poles and zeros, damping and frequency, and their connection with the signal as a function of time. This chapter emphasizes the solution of differential equations representing linear time-invariant (LTI) systems, paying special attention to transient solutions due to their importance in control, as well as to steady-state solutions due to their importance in filtering and in communications. The convolution integral is dealt with in time and using the Laplace transform to emphasize the operational power of the transform. The important concept of transfer function for LTI systems and the significance of its poles and zeros are studied in detail. Different approaches are considered in computing the inverse Laplace transform, including MATLAB methods.

Fourier analysis of continuous-time signals and systems is covered in detail in Chapters 4 and 5. The Fourier series analysis of periodic signals, covered in Chapter 4, is extended to the analysis of aperiodic signals resulting in the Fourier transform of Chapter 5. The Fourier transform is useful in representing both periodic and aperiodic signals. Special attention is given to the connection of these methods with the Laplace transform so that, whenever possible, known Laplace transforms can be used to compute the Fourier series coefficients and the Fourier transform—thus avoiding integration but using the concept of the region of convergence. The concept of frequency, the response of the system (connected to the location of poles and zeros of the transfer function), and the steady-state response are emphasized in these chapters.

The ordering of the presentation of the Laplace and the Fourier transformations (similar to the Z-transform and the Fourier representation of discrete-time signals) is significant for learning and teaching of the material.

Our approach of presenting first the Laplace transform and then the Fourier series and Fourier transform is justified by several reasons. For one, students coming into a signals and systems course have been familiarized with the Laplace transform in their previous circuits or differential equations courses, and will continue using it in control courses. So expertise in this topic is important and the learned material will stay with them longer. Another is that a common difficulty students have in applying the Fourier series and the Fourier transform is connected with the required integration. The Laplace transform can be used not only to sidestep the integration but to provide a more comprehensive understanding of the frequency representation. By asking students to consider the two-sided Laplace transform and the significance of its region of convergence, they will appreciate better the Fourier representation as a special case of Laplace's in many cases. More importantly, these transforms can be seen as a continuum rather than as different transforms. It also makes theoretical sense to deal with the Laplace representation of systems first to justify the existence of the steady-state solution considered in the Fourier representations, which would not exist unless stability of the system is guaranteed, and stability can only be tested using the Laplace transform. The paradigm of interest is the connection of transient and steady-state responses that must be understood by students before they can understand the connections between Fourier and Laplace analyses.

Chapter 6 presents applications of the Laplace and the Fourier transforms to control, communications, and filtering. The intent of the chapter is to motivate interest in these areas. The chapter illustrates the significance of the concepts of transfer function, response of systems, and stability in control, and of modulation in communications. An introduction to analog filtering is provided. Analytic as well as MATLAB examples illustrate different applications to control, communications, and filter design.

Using the sampling theory as a bridge, the third part of the book covers the theory and illustrates the application of discrete-time signals and systems. Chapter 7 presents the theory of sampling: the conditions under which the signal does not lose information in the sampling process and the recovery of the analog signal from the sampled signal. Once the basic concepts are given, the analog-to-digital and digital-to-analog converters are considered to provide a practical understanding of the conversion of analog-to-digital and digital-to-analog signals.

Discrete-time signals and systems are discussed in Chapter 8, while Chapter 9 introduces the Z-transform. Although the treatment of discrete-time signals and systems in Chapter 8 mirrors that of continuous-time signals and systems, special emphasis is given in this chapter to issues that are different in the two domains. Issues such as the discrete nature of the time, the periodicity of the discrete frequency, the possible lack of periodicity of discrete sinusoids, etc. are considered. Chapter 9 provides the basic theory of the Z-transform and how it relates to the Laplace transform. The material in this chapter bears similarity to the one on the Laplace transform in terms of operational solution of difference equations, transfer function, and the significance of poles and zeros.

Chapter 10 presents the Fourier analysis of discrete signals and systems. Given the accumulated experience of the students with continuous-time signals and systems, we build the discrete-time Fourier transform (DTFT) on the Z-transform and consider special cases where the Z-transform cannot be used. The discrete Fourier transform (DFT) is obtained from the Fourier series of discrete-time signals and sampling in frequency. The DFT will be of great significance in digital signal processing. The computation of the DFT of periodic and aperiodic discrete-time signals using the fast Fourier transform (FFT) is illustrated. The FFT is an efficient algorithm for computing the DFT, and some of the basics of this algorithm are discussed in Chapter 12.

Chapter 11 introduces students to discrete filtering, thus extending the analog filtering in Chapter 6. In this chapter we show how to use the theory of analog filters to design recursive discrete low-pass filters. Frequency transformations are then presented to show how to obtain different types of filters from low-pass prototype filters. The design of finite-impulse filters using the window method is considered next. Finally, the implementation of recursive and nonrecursive filters is shown using some basic techniques. By using MATLAB for the design of recursive and nonrecursive discrete filters, it is expected that students will be motivated to pursue on their own the use of more sophisticated filter designs.

Finally, Chapter 12 explores topics of interest in digital communications, computer control, and digital signal processing. The aim of this chapter is to provide a brief presentation of topics that students could pursue after the basic courses in signals and systems.

TEACHING USING THIS TEXT

The material in this text is intended for a two-term sequence in signals and systems: one on continuous-time signals and systems, followed by a term in discrete-time signals and systems with a lab component using MATLAB. These two courses would cover most of the chapters in the text with various degrees of depth, depending on the emphasis the faculty would like to give to the course. As indicated, Chapter 0 was written as a necessary introduction to the rest of the material, but does not need to be covered in great detail—students can refer to it as needed. Chapters 6 and 11 need to be considered together if the emphasis on applications is in filter design. The control, communications, and digital signal processing material in Chapters 6 and 12 can be used to motivate students toward those areas.

TO THE STUDENT

It is important for you to understand the features of this book, so you can take advantage of them to learn the material:

1. Refer as often as necessary to the material in Chapter 0 to review or to learn the mathematical background; to understand the overall structure of the material; or to review or learn MATLAB as it applies to signal processing.
2. As you will see, the complexity of the material grows as it develops. The material in part three has been written assuming good understanding of the material in the first two. See also the connection of the material with applications in your own areas of interest.
3. To help you learn the material, clear and concise results are emphasized by putting them in boxes. Justification of these results is then given, complemented with remarks regarding issues that need a bit more clarification, and illustrated with plenty of analytic and computational examples. Important terms are emphasized throughout the text. Tables provide a good summary of properties and formulas.
4. A heading is used in each of the problems at the end of the chapters, indicating how it relates to specific topics and if it requires to use MATLAB to solve it.
5. One of the objectives of this text is to help you learn MATLAB, as it applies to signal and systems, on your own. This is done by providing the soft introduction to MATLAB in Chapter 0, and then by showing examples using simple code in each of the chapters. You will notice that in the first two parts basic components of MATLAB (scripts, functions, plotting, etc.) are given in more detail than in part three. It is assumed you are very proficient by then to supply that on your own.
6. Finally, notice the footnotes, the vignettes, and the historical sidebars that have been included to provide a glance at the background in which the theory and practice of signals and systems have developed.

Acknowledgments

I would like to acknowledge with gratitude the support and efforts of many people who made the writing of this text possible. First, to my family—my wife Cathy, my children William, Camila, and Juan, and their own families—many thanks for their support and encouragement despite being deprived of my attention. To my academic mentor, Professor Eliahu I. Jury, a deep sense of gratitude for his teachings and for having inculcated in me the love for a scholarly career and for the theory and practice of signals and systems. Thanks to Professor William Stanchina, chair of the Department of Electrical and Computer Engineering at the University of Pittsburgh, for his encouragement and support that made it possible to dedicate time to the project. Sincere thanks to Seda Senay and Mircea Lupus, graduate students in my department. Their contribution to the painful editing and proofreading of the manuscript, and the generation of the solution manual (especially from Ms. Senay) are much appreciated. Equally, thanks to the publisher and its editors, in particular to Joe Hayton and Steve Merken, for their patience, advising, and help with the publishing issues. Thanks also to Sarah Binns for her help with the final editing of the manuscript. Equally, I would like to thank Professor James Rowland from the University of Kansas and the following reviewers for providing significant input and changes to the manuscript: Dimitrie Popescu, Old Dominion University; Hossein Hakim, Worcester Polytechnic Institute; Mark Budnik, Valparaiso University; Periasamy Rajan, Tennessee Tech University; and Mohamed Zohdy, Oakland University. Thanks to my colleagues Amro El-Jaroudi and Juan Manfredi for their early comments and suggestions.

Lastly, I feel indebted to the many students I have had in my courses in signals and systems over the years I have been teaching this material in the Department of Electrical and Computer Engineering at the University of Pittsburgh. Unknown to them, they contributed to my impetus to write a book that I felt would make the teaching of signals and systems more accessible and fun to future students in and outside the university.

RESOURCES THAT ACCOMPANY THIS BOOK

A companion website containing downloadable MATLAB code for the worked examples in the book is available at:

http://booksite.academicpress.com/chaparro

For instructors, a solutions manual and image bank containing electronic versions of figures from the book are available by registering at:

www.textbooks.elsevier.com

Also Available for Use with This Book – Elsevier Online Testing

Web-based testing and assessment feature that allows instructors to create online tests and assignments which automatically assess student responses and performance, providing them with immediate feedback. Elsevier's online testing includes a selection of algorithmic questions, giving instructors the ability to create virtually unlimited variations of the same problem. Contact your local sales representative for additional information, or visit *http://booksite.academicpress.com/chaparro/* to view a demo chapter.

PART 1

Introduction

This page intentionally left blank

CHAPTER 0

From the Ground Up!

In theory there is no difference
between theory and practice.
In practice there is.
Lawrence "Yogi" Berra, 1925
New York Yankees baseball player

This chapter provides an overview of the material in the book and highlights the mathematical background needed to understand the analysis of signals and systems. We consider a signal a function of time (or space if it is an image, or of time and space if it is a video signal), just like the voltages or currents encountered in circuits. A system is any device described by a mathematical model, just like the differential equations obtained for a circuit composed of resistors, capacitors, and inductors.

By means of practical applications, we illustrate in this chapter the importance of the theory of signals and systems and then proceed to connect some of the concepts of integro-differential Calculus with more concrete mathematics (from the computational point of view, i.e., using computers). A brief review of complex variables and their connection with the dynamics of systems follows. We end this chapter with a soft introduction to MATLAB, a widely used high-level computational tool for analysis and design.

Significantly, we have called this Chapter 0, because it is the ground floor for the rest of the material in the book. Not everything in this chapter has to be understood in a first reading, but we hope that as you go through the rest of the chapters in the book you will get to appreciate that the material in this chapter is the foundation of the book, and as such you should revisit it as often as needed.

0.1 SIGNALS AND SYSTEMS AND DIGITAL TECHNOLOGIES

In our modern world, signals of all kinds emanate from different types of devices—radios and TVs, cell phones, global positioning systems (GPSs), radars, and sonars. These systems allow us to communicate messages, to control processes, and to sense or measure signals. In the last 60 years, with the advent of the transistor, the digital computer, and the theoretical fundamentals of digital signal

Signals and Systems Using MATLAB®. DOI: 10.1016/B978-0-12-374716-7.00002-8
© 2011, Elsevier Inc. All rights reserved.

processing, the trend has been toward digital representation and processing of data, most of which are in analog form. Such a trend highlights the importance of learning how to represent signals in analog as well as in digital forms and how to model and design systems capable of dealing with different types of signals.

1948

The year 1948 is considered the birth year of technologies and theories responsible for the spectacular advances in communications, control, and biomedical engineering since then. In June 1948, Bell Telephone Laboratories announced the invention of the transistor. Later that month, a prototype computer built at Manchester University in the United Kingdom became the first operational stored-program computer. Also in that year, many fundamental theoretical results were published: Claude Shannon's mathematical theory of communications, Richard W. Hamming's theory on error-correcting codes, and Norbert Wiener's *Cybernetics* comparing biological systems with communication and control systems [51].

Digital signal processing advances have gone hand-in-hand with progress in electronics and computers. In 1965, Gordon Moore, one of the founders of Intel, envisioned that the number of transistors on a chip would double about every two years [35]. Intel, the largest chip manufacturer in the world, has kept that pace for 40 years. But at the same time, the speed of the central processing unit (CPU) chips in desktop personal computers has dramatically increased. Consider the well-known Pentium group of chips (the Pentium Pro and the Pentium I to IV) introduced in the 1990s [34]. Figure 0.1 shows the range of speeds of these chips at the time of their introduction into the market, as well as the number of transistors on each of these chips. In five years, 1995 to 2000, the speed increased by a factor of 10 while the number of transistors went from 5.5 million to 42 million.

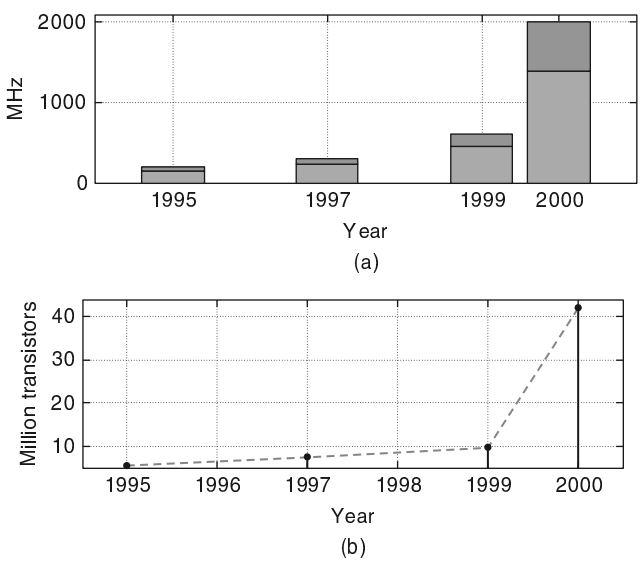


FIGURE 0.1
The Intel Pentium CPU chips. (a) Range of CPU speeds in MHz for the Pentium Pro (1995), Pentium II (1997), Pentium III (1999), and Pentium IV (2000). (b) Number of transistors (in millions) on each of the above chips. (Pentium data taken from [34].)

Advances in digital electronics and in computer engineering in the past 60 years have permitted the proliferation of digital technologies. Digital hardware and software process signals from cell phones, high-definition television (HDTV) receivers, radars, and sonars. The use of digital signal processors (DSPs) and more recently of field-programmable gate arrays (FPGAs) have been replacing the use of application-specific integrated circuits (ASICs) in industrial, medical, and military applications.

It is clear that digital technologies are here to stay. Today, digital transmission of voice, data, and video is common, and so is computer control. The abundance of algorithms for processing signals, and the pervasive presence of DSPs and FPGAs in thousands of applications make digital signal processing theory a necessary tool not only for engineers but for anybody who would be dealing with digital data; soon, that will be everybody! This book serves as an introduction to the theory of signals and systems—a necessary first step in the road toward understanding digital signal processing.

DSPs and FPGAs

A digital signal processor (DSP) is an optimized microprocessor used in real-time signal processing applications [67]. DSPs are typically embedded in larger systems (e.g., a desktop computer) handling general-purpose tasks. A DSP system typically consists of a processor, memory, analog-to-digital converters (ADCs), and digital-to-analog converters (DACs). The main difference with typical microprocessors is they are faster. A field-programmable gate array (FPGA) [77] is a semiconductor device containing programmable logic blocks that can be programmed to perform certain functions, and programmable interconnects. Although FPGAs are slower than their application-specific integrated circuits (ASICs) counterparts and use more power, their advantages include a shorter time to design and the ability to be reprogrammed.

0.2 EXAMPLES OF SIGNAL PROCESSING APPLICATIONS

The theory of signals and systems connects directly, among others, with communications, control, and biomedical engineering, and indirectly with mathematics and computer engineering. With the availability of digital technologies for processing signals, it is tempting to believe there is no need to understand their connection with analog technologies. It is precisely the opposite is illustrated by considering the following three interesting applications: the compact-disc (CD) player, software-defined radio and cognitive radio, and computer-controlled systems.

0.2.1 Compact-Disc Player

Compact discs [9] were first produced in Germany in 1982. Recorded voltage variations over time due to an acoustic sound is called an *analog signal* given its similarity with the differences in air pressure generated by the sound waves over time. Audio CDs and CD players illustrate best the conversion of a binary signal—unintelligible—into an intelligible analog signal. Moreover, the player is a very interesting control system.

To store an analog audio signal (e.g., voice or music) on a CD the signal must be first sampled and converted into a sequence of binary digits—a digital signal—by an ADC and then especially encoded to compress the information and to avoid errors when playing the CD. In the manufacturing of a CD,

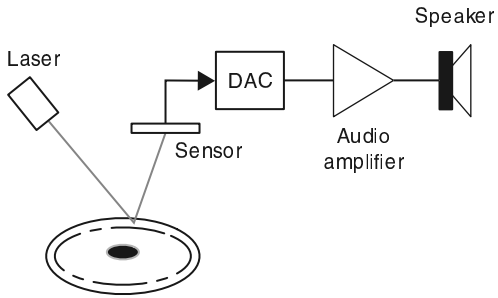


FIGURE 0.2
When playing a CD, the CD player follows the tracks in the disc, focusing a laser on them, as the CD is spun. The laser shines a light that is reflected by the pits and bumps put on the surface of the disc and corresponding to the coded digital signal from an acoustic signal. A sensor detects the reflected light and converts it into a digital signal, which is then converted into an analog signal by the DAC. When amplified and fed to the speakers such a signal sounds like the originally recorded acoustic signal.

pits and bumps corresponding to the ones and zeros from the quantization and encoding processes are impressed on the surface of the disc. Such pits and bumps will be detected by the CD player and converted back into an analog signal that approximates the original signal when the CD is played. The transformation into an analog signal uses a DAC.

As we will see in Chapter 7, an audio signal is sampled at a rate of about 44,000 samples/second (sec) (corresponding to a maximum frequency around 22 KHz for a typical audio signal) and each of these samples is represented by a certain number of bits (typically 8 bits/sample). The need for stereo sound requires that two channels be recorded. Overall, the number of bits representing the signal is very large and needs to be compressed and especially encoded. The resulting data, in the form of pits and bumps impressed on the CD surface, are put into a spiral track that goes from the inside to the outside of the disc.

Besides the binary-to-analog conversion, the CD player exemplifies a very interesting control system (see Figure 0.2). Indeed, the player must: (1) rotate the disc at different speeds depending on the location of the track within the CD being read, (2) focus a laser and a lens system to read the pits and bumps on the disc, and (3) move the laser to follow the track being read. To understand the exactness required, consider that the width of the track and the high of the bumps is typically less than a micrometer (10^{-6} meters or 3.937×10^{-5} inches) and a nanometer (10^{-9} meters or 3.937×10^{-8} inches), respectively.

0.2.2 Software-Defined Radio and Cognitive Radio

Software-defined radio and cognitive radio are important emerging technologies in wireless communications [43]. In software-defined radio (SDR), some of the radio functions typically implemented in hardware are converted into software [64]. By providing smart processing to SDRs, cognitive radio (CR) will provide the flexibility needed to more efficiently use the radio frequency spectrum and to make available new services to users. In the United States the Federal Communication Commission (FCC), and likewise in other parts of the world the corresponding agencies, allocates the bands for

different users of the radio spectrum (commercial radio and TV, amateur radio, police, etc.). Although most bands have been allocated, implying a scarcity of spectrum for new users, it has been found that locally at certain times of the day the allocated spectrum is not being fully utilized. Cognitive radio takes advantage of this.

Conventional radio systems are composed mostly of hardware, and as such cannot be easily reconfigured. The basic premise in SDR as a wireless communication system is its ability to reconfigure by changing the software used to implement functions typically done by hardware in a conventional radio. In an SDR transmitter, software is used to implement different types of modulation procedures, while ADCs and DACs are used to change from one type of signal to another. Antennas, audio amplifiers, and conventional radio hardware are used to process analog signals. Typically, an SDR receiver uses an ADC to change the analog signals from the antenna into digital signals that are processed using software on a general-purpose processor. See Figure 0.3.

Given the need for more efficient use of the radio spectrum, cognitive radio (CR) uses SDR technology while attempting to dynamically manage the radio spectrum. A cognitive radio monitors locally the radio spectrum to determine regions that are not occupied by their assigned users and transmits in those bands. If the primary user of a frequency band recommences transmission, the CR either moves to another frequency band, or stays in the same band but decreases its transmission power level or modulation scheme to avoid interference with the assigned user. Moreover, a CR will search

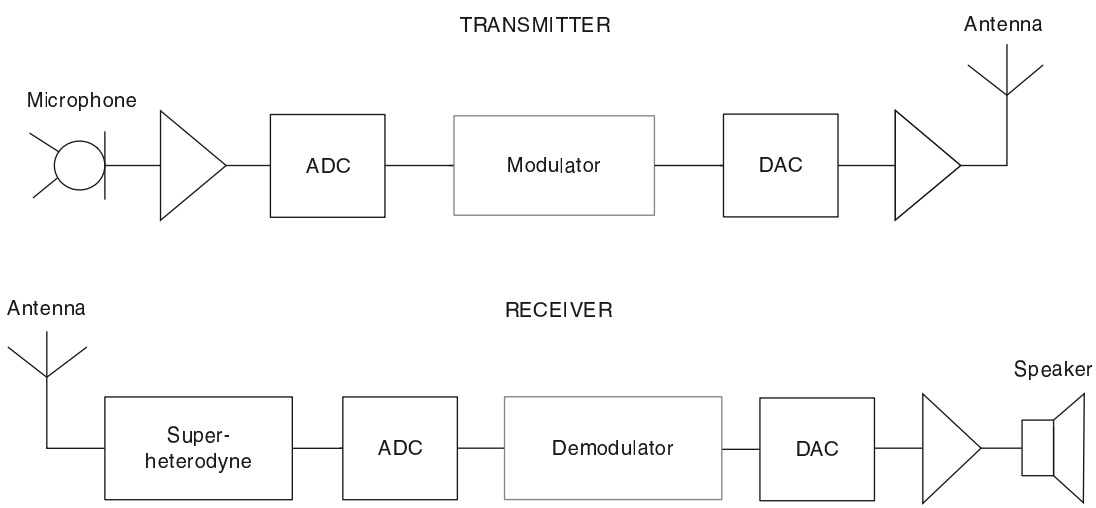


FIGURE 0.3

Schematics of a voice SDR mobile two-way radio. *Transmitter:* The voice signal is inputted by means of a microphone, amplified by an audio amplifier, converted into a digital signal by an ADC, and then modulated using software, before being converted into analog by an DAC, amplified, and sent as a radio frequency signal via an antenna. *Receiver:* The signal received by the antenna is processed by a superheterodyne front-end, converted into a digital signal by an ADC before being demodulated and converted into an analog signal by a DAC, amplified, and fed to a speaker. The modulator and demodulator blocks indicate software processing.

for network services that it can offer to its users. Thus, SDR and CR are bound to change the way we communicate and use network services.

0.2.3 Computer-Controlled Systems

The application of computer control ranges from controlling simple systems such as a heater (e.g., keeping a room temperature comfortable while reducing energy consumption) or cars (e.g., controlling their speed), to that of controlling rather sophisticated machines such as airplanes (e.g., providing automatic flight control) or chemical processes in very large systems such as oil refineries. A significant advantage of computer control is the flexibility computers provide—rather sophisticated control schemes can be implemented in software and adapted for different control modes.

Typically, control systems are feedback systems where the dynamic response of a system is changed to make it follow a desirable behavior. As indicated in Figure 0.4, the plant is a system, such as a heater, car, or airplane, or a chemical process in need of some control action so that its output (it is also possible for a system to have several outputs) follows a reference signal (or signals). For instance, one could think of a cruise-control system in a car that attempts to keep the speed of the car at a certain value by controlling the gas pedal mechanism. The control action will attempt to have the output of the system follow the desired response, despite the presence of disturbances either in the plant (e.g., errors in the model used for the plant) or in the sensor (e.g., measurement error). By comparing the reference signal with the output of the sensor, and using a control law implemented in the computer, a control action is generated to change the state of the plant and attain the desired output.

To use a computer in a control application it is necessary to transform analog signals into digital signals so that they can be inputted into the computer, while it is also necessary that the output of the computer be converted into an analog signal to drive an actuator (e.g., an electrical motor) to provide an action capable of changing the state of the plant. This can be done by means of ADCs and DACs. The sensor should also be able to act as a transducer whenever the output of the plant is

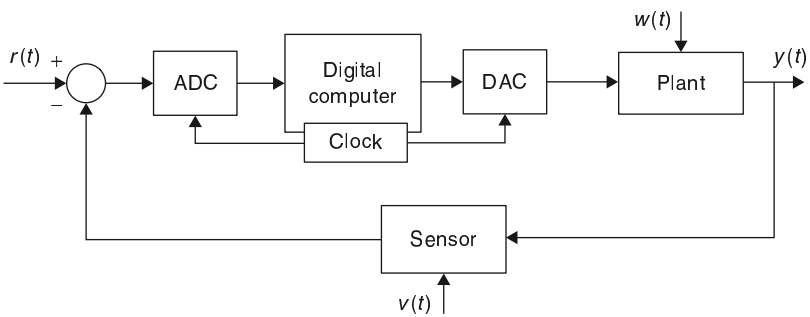


FIGURE 0.4

Computer-controlled system for an analog plant (e.g., cruise control for a car). The reference signal is $r(t)$ (e.g., desired speed) and the output is $y(t)$ (e.g., car speed). The analog signals are converted to digital signals by an ADC, while the digital signal from the computer is converted into an analog signal (an actuator is probably needed to control the car) by a DAC. The signals $w(t)$ and $v(t)$ are disturbances or noise in the plant and the sensor (e.g., electronic noise in the sensor and undesirable vibration in the car).

of a different type than the reference. Such would be the case, for instance, if the plant output is a temperature while the reference signal is a voltage.

0.3 ANALOG OR DISCRETE?

Infinitesimal calculus, or just plain *calculus*, deals with functions of one or more continuously changing variables. Based on the representation of these functions, the concepts of *derivative* and *integral* are developed to measure the rate of change of functions and the areas under the graphs of these functions, or their volumes. Differential equations are then introduced to characterize dynamic systems.

Finite calculus, on the other hand, deals with sequences. Thus, derivatives and integrals are replaced by differences and summations, while differential equations are replaced by difference equations. Finite calculus makes possible the computations of calculus by means of a combination of digital computers and numerical methods—thus, finite calculus becomes the more concrete mathematics.[1] Numerical methods applied to sequences permit us to approximate derivatives, integrals, and the solution of differential equations.

In engineering, as in many areas of science, the inputs and outputs of electrical, mechanical, chemical, and biological processes are measured as functions of time with amplitudes expressed in terms of voltage, current, torque, pressure, etc. These functions are called *analog or continuous-time signals*, and to process them with a computer they must be converted into binary sequences—or a string of ones and zeros that is understood by the computer. Such a conversion is done in a way as to preserve as much as possible the information contained in the original signal. Once in binary form, signals can be processed using algorithms (coded procedures understood by computers and designed to obtain certain desired information from the signals or to change them) in a computer or in a dedicated piece of hardware.

In a digital computer, differentiation and integration can be done only approximately, and the solution of differential equations requires a discretization process as we will illustrate later in this chapter. Not all signals are functions of a continuous parameter—there exist inherently discrete-time signals that can be represented as sequences, converted into binary form, and processed by computers. For these signals the finite calculus is the natural way of representing and processing them.

Analog or continuous-time signals are converted into binary sequences by means of an ADC, which, as we will see, compresses the data by converting the continuous-time signal into a discrete-time signal or a sequence of samples, each sample being represented by a string of ones and zeros giving a binary signal. Both time and signal amplitude are made discrete in this process. Likewise, digital signals can be transformed into analog signals by means of a DAC that uses the reverse process of the ADC. These converters are commercially available, and it is important to learn how they work so that digital representation of analog signals is obtained

[1]The use of *concrete*, rather than abstract, mathematics was coined by Graham, Knuth, and Patashnik in *Concrete Mathematics: A Foundation for Computer Science* [26]. Professor Donald Knuth from Stanford University is the the inventor of the Tex and Metafont typesetting systems that are the precursors of Latex, the document layout system in which the original manuscript of this book was done.

with minimal information loss. Chapters 1, 7, and 8 will provide the necessary information about continuous-time and discrete-time signals, and show how to convert one into the other and back. The sampling theory presented in Chapter 7 is the backbone of digital signal processing.

0.3.1 Continuous-Time and Discrete-Time Representations

There are significant differences between continuous-time and discrete-time signals as well as in their processing. A discrete-time signal is a sequence of measurements typically made at uniform times, while the analog signal depends continuously on time. Thus, a discrete-time signal $x[n]$ and the corresponding analog signal $x(t)$ are related by a sampling process:

$$x[n] = x(nT_s) = x(t)_{|t=nT_s} \tag{0.1}$$

That is, the signal $x[n]$ is obtained by sampling $x(t)$ at times $t = nT_s$, where n is an integer and T_s is the *sampling period* or the time between samples. This results in a sequence,

$$\{\cdots x(-T_s)\ \ x(0)\ \ x(T_s)\ \ x(2T_s)\cdots\}$$

according to the sampling times, or equivalently

$$\{\cdots x[-1]\ \ x[0]\ \ x[1]\ \ x[2]\cdots\}$$

according to the ordering of the samples (as referenced to time 0). This process is called *sampling* or *discretization* of an analog signal.

Clearly, by choosing a small value for T_s we could make the analog and the discrete-time signals look very similar—almost indistinguishable—which is good, but this is at the expense of memory space required to keep the numerous samples. If we make the value of T_s large, we improve the memory requirements, but at the risk of losing information contained in the original signal. For instance, consider a sinusoid obtained from a signal generator:

$$x(t) = 2\cos(2\pi t)$$

for $0 \le t \le 10$ sec. If we sample it every $T_{s1} = 0.1$ sec, the analog signal becomes the following sequence:

$$x_1[n] = x(t)\,|_{t=0.1n} = 2\cos(2\pi n/10) \quad 0 \le n \le 100$$

providing a very good approximation to the original signal. If, on the other hand, we let $T_{s2} = 1$ sec, then the discrete-time signal becomes

$$x_2[n] = x(t)\,|_{t=n} = 2\cos(2\pi n) = 2 \quad 0 \le n \le 10$$

See Figure 0.5. Although for T_{s2} the number of samples is considerably reduced, the representation of the original signal is very poor—it appears as if we had sampled a constant signal, and we have thus lost information! This indicates that it is necessary to come up with a way to choose T_s so that sampling provides not only a reasonable number of samples, but, more importantly, guarantees that the information in the analog and the discrete-time signals remains the same.

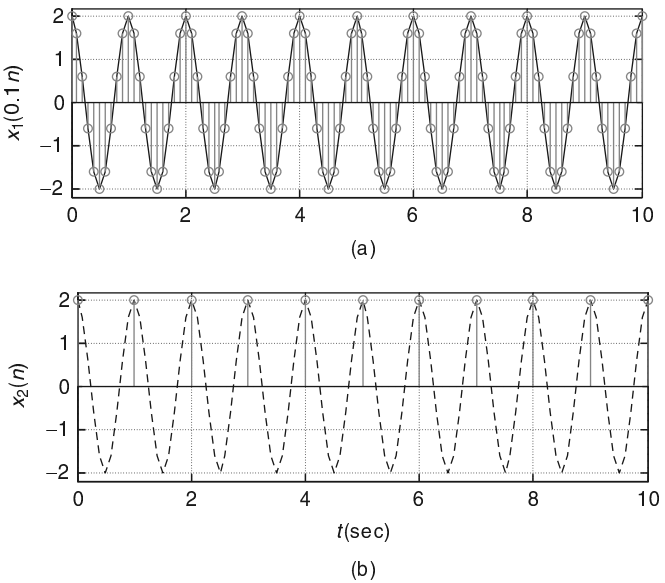


FIGURE 0.5
Sampling an analog sinusoid $x(t) = 2\cos(2\pi t), 0 \leq t \leq 10$, with two different sampling periods, (a) $T_{s1} = 0.1$ sec and (b) $T_{s2} = 1$ sec, giving $x_1(0.1n)$ and $x_2(n)$. The sinusoid is shown by dashed lines. Notice the similarity between the discrete-time signal and the analog signal when $T_{s1} = 0.1$ sec, while they are very different when $T_{s2} = 1$ sec, indicating loss of information.

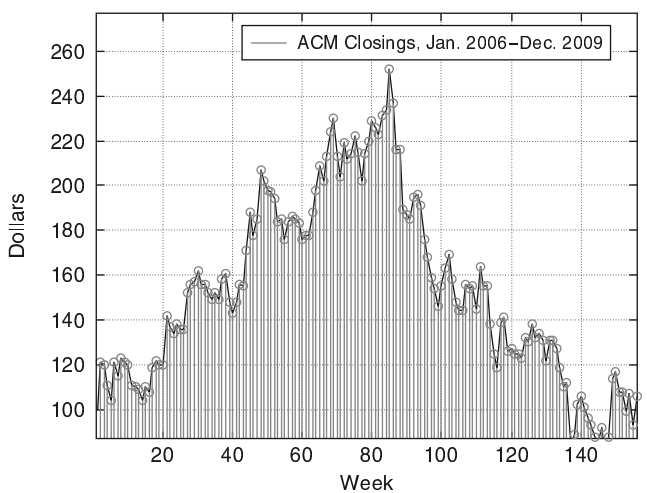


FIGURE 0.6
Weekly closings of ACM stock for 160 weeks in 2006 to 2009. ACM is the trading name of the stock of the imaginary company, ACME Inc., makers of everything you can imagine.

As indicated before, not all signals are analog; there are some that are naturally discrete. Figure 0.6 displays the weekly average of the stock price of a fictitious company, ACME. Thinking of it as a signal, it is naturally discrete-time as it does not come from the discretization of an analog signal.

We have shown in this section the significance of the sampling period T_s in the transformation of an analog signal into a discrete-time signal without losing information. Choosing the sampling period requires knowledge of the frequency content of the signal—this is an example of the relation between time and frequency to be presented in great detail in Chapters 4 and 5, where the Fourier representation of periodic and nonperiodic

signals is given. In Chapter 7, where we consider the problem of sampling, we will use this relation to determine appropriate values for the sampling period.

0.3.2 Derivatives and Finite Differences

Differentiation is an operation that is approximated in finite calculus. The derivative operator

$$D[x(t)] = \frac{dx(t)}{dt} = \lim_{h \to 0} \frac{x(t+h) - x(t)}{h} \tag{0.2}$$

measures the rate of change of an analog signal $x(t)$. In finite calculus the *forward finite-difference operator*

$$\Delta[x(nT_s)] = x((n+1)T_s) - x(nT_s) \tag{0.3}$$

measures the change in the signal from one sample to the next. If we let $x[n] = x(nT_s)$, for a known T_s, the forward finite-difference operator becomes a function of n:

$$\Delta[x[n]] = x[n+1] - x[n] \tag{0.4}$$

The forward finite-difference operator measures the difference between two consecutive samples: one in the future $x((n+1)T_s)$ and the other in the present $x(nT_s)$. (See Problem 0.4 for a definition of the *backward finite-difference operator*.) The symbols D and Δ are called operators as they operate on functions to give other functions. The derivative and the finite-difference operators are clearly not the same. In the limit, we have that

$$\frac{dx(t)}{dt} \Big|_{t=nT_s} = \lim_{T_s \to 0} \frac{\Delta[x(nT_s)]}{T_s} \tag{0.5}$$

Depending on the signal and the chosen value of T_s, the finite-difference operation can be a crude or an accurate approximation to the derivative multiplied by T_s.

Intuitively, if a signal does not change very fast with respect to time, the finite-difference approximates well the derivative for relatively large values of T_s, but if the signal changes very fast one needs very small values of T_s. The concept of frequency of a signal can help us understand this. We will learn that the frequency content of a signal depends on how fast the signal varies with time; thus a constant signal has zero frequency while a noisy signal that changes rapidly has high frequencies. Consider a constant signal $x_0(t) = 2$ having a derivative of zero (i.e., such a signal does not change at all with respect to time or it is a zero-frequency signal). If we convert this signal into a discrete-time signal using a sampling period $T_s = 1$ (or any other positive value), then $x_0[n] = 2$ and so

$$\Delta[x_0[n]] = 2 - 2 = 0$$

coincides with the derivative. Consider then a signal $x_1(t) = t$ with derivative 1 (this signal changes faster than $x(t)$ so it has frequencies larger than zero). If we sample it using $T_s = 1$, then $x_1[n] = n$ and the finite difference is

$$\Delta[x_1[n]] = \Delta[n] = (n+1) - n = 1$$

which again coincides with the derivative. Finally, we consider a signal that changes faster than $x(t)$ and $x_1(t)$ such as $x_2(t) = t^2$. Sampling $x_2(t)$ with $T_s = 1$, we have $x_2[n] = n^2$ and its forward finite difference is given by

$$\Delta[x_2[n]] = \Delta[n^2] = (n+1)^2 - n^2 = 2n + 1$$

which gives as an approximation to the derivative $\Delta[x_2[n]]/T_s = 2n + 1$. The derivative of $x_2(t)$ is $2t$, which at 0 equals 0, and at 1 equals 2. On the other hand, $\Delta[n^2]/T_s$ equals 1 and 3 at $n = 0$ and $n = 1$, respectively, which are different values from those of the derivative. Suppose $T_s = 0.01$, so that $x_2[n] = x_2(nT_s) = (0.01n)^2 = 0.0001n^2$. If we compute the difference for this signal we get

$$\Delta[x_2(0.01n)] = \Delta[(0.01n)^2] = (0.01n + 0.01)^2 - 0.0001n^2 = 10^{-4}(2n + 1)$$

which gives as an approximation to the derivative $\Delta[x_2(0.01n)]/T_s = 10^{-2}(2n + 1)$, or 0.01 when $n = 0$ and 0.03 when $n = 1$ which are a lot closer to the actual values of

$$\frac{dx_2(t)}{dt}|_{t=0.01n} = 2t|_{t=0.01n} = 0.02n$$

The error now is 0.01 for each case instead of 1 as in the case when $T_s = 1$. Thus, whenever the rate of change of the signal is faster, the difference gets closer to the derivative by making T_s smaller.

It becomes clear that the faster the signal changes, the smaller the sampling period T_s should be in order to get a better approximation of the signal and its derivative. As we will learn in Chapters 4 and 5 the frequency content of a signal depends on the signal variation over time. A constant signal has frequency zero, while a signal that changes very fast over time would have high frequencies. The higher the frequencies in a signal, the more samples would be needed to represent it with no loss of information, thus requiring that T_s be smaller.

0.3.3 Integrals and Summations

Integration is the opposite of differentiation. To see this, suppose $I(t)$ is the integration of a continuous signal $x(t)$ from some time t_0 to t ($t_0 < t$),

$$I(t) = \int_{t_0}^{t} x(\tau)d\tau \tag{0.6}$$

or the sum of the area under $x(t)$ from t_0 to t. Notice that the upper bound of the integral is t so the integrand depends on a dummy variable.[2] The derivative of $I(t)$ is

$$\frac{dI(t)}{dt} = \lim_{h\to 0} \frac{I(t) - I(t-h)}{h} = \lim_{h\to 0} \frac{1}{h} \int_{t-h}^{t} x(\tau)d\tau$$

$$\approx \lim_{h\to 0} \frac{x(t) + x(t-h)}{2} = x(t)$$

where the integral is approximated as the area of a trapezoid with sides $x(t)$ and $x(t-h)$ and height h. Thus, for a continuous signal $x(t)$,

$$\frac{d}{dt} \int_{t_0}^{t} x(\tau)d\tau = x(t) \tag{0.7}$$

or if using the derivative operator $D[.]$, then its inverse $D^{-1}[.]$ should be the integration operator. That is, the above equation can be written

$$D[D^{-1}[x(t)]] = x(t). \tag{0.8}$$

We will see in Chapter 3 a similar relation between the derivative and the integral. The Laplace transform operators s and $1/s$ (just like D and $1/D$) imply differentiation and integration in the time domain.

Computationally, integration is implemented by sums. Consider, for instance, the integral of $x(t) = t$ from 0 to 10, which we know is equal to

$$\int_0^{10} t \; dt = \frac{t^2}{2} \Big|_{t=0}^{10} = 50.$$

That is, the area of a triangle with a base of 10 and a height of 10. For $T_s = 1$, suppose we approximate the signal $x(t)$ by pulses $p[n]$ of width $T_s = 1$ and height $nT_s = n$, or pulses of area n for $n = 0, \ldots, 9$. This can be seen as a lower-bound approximation to the integral, as the total area of these pulses gives a result smaller than the integral. In fact, the sum of the areas of the pulses is given by

$$\sum_{n=0}^{9} p[n] = \sum_{n=0}^{9} n = 0 + 1 + 2 + \cdots 9 = 0.5 \left[\sum_{n=0}^{9} n + \sum_{k=9}^{0} k \right]$$

$$= 0.5 \left[\sum_{n=0}^{9} n + \sum_{n=0}^{9} (9-n) \right] = \frac{9}{2} \sum_{n=0}^{9} 1 = \frac{10 \times 9}{2} = 45$$

[2]The integral $I(t)$ is a function of t and as such the integrand needs to be expressed in terms of a so-called *dummy variable* τ that takes values from t_0 to t in the integration. It would be confusing to let the integration variable be t. The variable τ is called a *dummy variable* because it is not crucial to the integration; any other variable could be used with no effect on the integration.

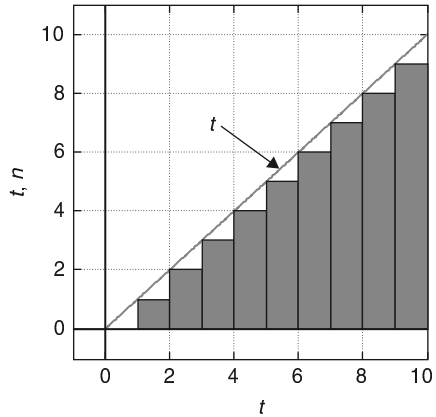


FIGURE 0.7
Approximation of area under $x(t) = t, t \geq 0$, 0 otherwise, by pulses of width 1 and height nT_s, where $T_s = 1$ and $n = 0, 1, \ldots$

The approximation of the area using $T_s = 1$ is very poor (see Figure 0.7). In the above, we used the fact that the sum is not changed whether we add the numbers from 0 to 9 or backwards from 9 to 0, and that doubling the sum and dividing by 2 would not change the final answer. The above sum can thus be generalized to

$$\begin{aligned}\sum_{n=0}^{N-1} n &= \frac{1}{2}\left[\sum_{n=0}^{N-1} n + \sum_{n=0}^{N-1}(N-1-n)\right] = \frac{1}{2}\sum_{n=0}^{N-1}(N-1) \\ &= \frac{N \times (N-1)}{2}\end{aligned} \tag{0.9}$$

a result that Gauss found out when he was a preschooler![3]

To improve the approximation of the integral we use $T_s = 10^{-3}$, which gives a discretized signal nT_s for $0 \leq nT_s < 10$ or $0 \leq n \leq (10/T_s) - 1$. The area of the pulses is nT_s^2 and the approximation to the integral is then

$$\begin{aligned}\sum_{n=0}^{10^4-1} p[n] &= \sum_{n=0}^{10^4-1} n10^{-6} \\ &= \frac{10^4 \times (10^4 - 1)}{10^6 \times 2} \\ &= 49.995\end{aligned}$$

[3]Carl Friedrich Gauss (1777–1855) was a German mathematician. He was seven years old when he amazed his teachers with his trick for adding the numbers from 1 to 100 [7]. Gauss is one of the most accomplished mathematicians of all times [2]. He is in a group of selected mathematicians and scientists whose pictures appear in the currency of a country. His picture was on the Mark, the previous currency of Germany [6].

which is a lot better result. In general, we have that the integral can be computed quite accurately using a very small value of T_s, indeed

$$\begin{aligned}\sum_{n=0}^{(10/T_s)-1} p[n] &= \sum_{n=0}^{(10/T_s)-1} nT_s^2 \\ &= T_s^2 \frac{(10/T_s) \times ((10/T_s) - 1)}{2} \\ &= \frac{10 \times (10 - T_s)}{2}\end{aligned}$$

which for very small values of T_s (so that $10 - T_s \approx 10$) gives $100/2 = 50$, as desired.

Derivatives and integrals take us into the processing of signals by systems. Once a mathematical model for a dynamic system is obtained, typically differential equations characterize the relation between the input and output variable or variables of the system. A significant subclass of systems (used as a valid approximation in some way to actual systems) is given by linear differential equations with constant coefficients. The solution of these equations can be efficiently found by means of the Laplace transform, which converts them into algebraic equations that are much easier to solve. The Laplace transform is covered in Chapter 3, in part to facilitate the analysis of analog signals and systems early in the learning process, but also so that it can be related to the Fourier theory of Chapters 4 and 5. Likewise for the analysis of discrete-time signals and systems we present in Chapter 9 the Z-transform, having analogous properties to those from the Laplace transform, before the Fourier analysis of those signals and systems.

0.3.4 Differential and Difference Equations

A differential equation characterizes the dynamics of a continuous-time system, or the way the system responds to inputs over time. There are different types of differential equations, corresponding to different systems. Most systems are characterized by nonlinear, time-dependent coefficient differential equations. The analytic solution of these equations is rather complicated. To simplify the analysis, these equations are locally approximated as linear constant-coefficient differential equations.

Solution of differential equations can be obtained by means of analog and digital computers. An electronic *analog computer* consists of operational amplifiers (op-amps), resistors, capacitors, voltage sources, and relays. Using the linearized model of the op-amps, resistors, and capacitors it is possible to realize integrators to solve a differential equation. Relays are used to set the initial conditions on the capacitors, and the voltage source gives the input signal. Although this arrangement permits the solution of differential equations, its drawback is the storage of the solution, which can be seen with an oscilloscope but is difficult to record. Hybrid computers were suggested as a solution—the analog computer is assisted by a digital component that stores the data. Both analog and hybrid computers have gone the way of the dinosaurs, and it is digital computers aided by numerical methods that are used now to solve differential equations.

Before going into the numerical solution provided by digital computers, let us consider why integrators are needed in the solution of differential equations. A first-order (the highest derivative present in the equation); linear (no nonlinear functions of the input or the output are present) with

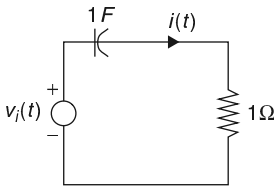


FIGURE 0.8
RC circuit.

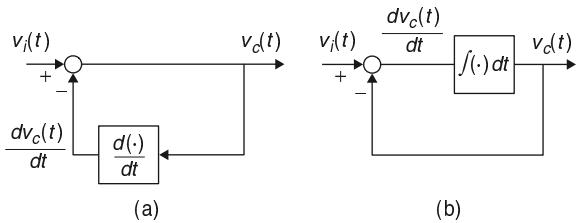


FIGURE 0.9
Realization of first-order differential equation using (a) a differentiator and (b) an integrator.

constant-coefficient differential equations obtained from a simple RC circuit (Figure 0.8) with a constant voltage source $v_i(t)$ as input and with resistor $R = 1\Omega$; and capacitor $C = 1$ F (with huge plates!) connected in series is given by

$$v_i(t) = v_c(t) + \frac{dv_c(t)}{dt} \tag{0.10}$$

with an initial voltage $v_c(0)$ across the capacitor.

Intuitively, in this circuit the capacitor starts with an initial charge of $v_c(0)$, and will continue charging until it reaches saturation, at which point no more charge will flow (the current across the resistor and the capacitor is zero). Therefore, the voltage across the capacitor is equal to the voltage source–that is, the capacitor is acting as an open circuit given that the source is constant.

Suppose, ideally, that we have available devices that can perform differentiation. There is then the tendency to propose that the differential equation (Eq. 0.10) be solved following the block diagram shown in Figure (0.9). Although nothing is wrong analytically, the problem with this approach is that in practice most signals are noisy (each device produces electronic noise) and the noise present in the signal may cause large derivative values given its rapidly changing amplitudes. Thus, the realization of the differential equation using differentiators is prone to being very noisy (i.e., not good). Instead of, as proposed years ago by Lord Kelvin,[4] using differentiators we need to smooth out the process by using integrators, so that the voltage across the capacitor $v_c(t)$ is obtained by integrating both sides of Equation (0.10). Assuming that the source is switched on at time $t = 0$ and that the capacitor has an initial voltage $v_c(0)$, using the inverse relation between derivatives and integrals gives

$$v_c(t) = \int_0^t [v_i(\tau) - v_c(\tau)]d\tau + v_c(0) \qquad t \geq 0 \tag{0.11}$$

[4]William Thomson, Lord Kelvin, proposed in 1876 the *differential analyzer*, a type of analog computer capable of solving differential equations of order 2 and higher. His brother James designed one of the first differential analyzers [78].

which is represented by the block diagram in Figure 0.9(b). Notice that the integrator also provides a way to include the initial condition, which in this case is the initial voltage across the capacitor, $v_c(0)$. Different from the accentuating the effect of differentiators on noise, integrators average the noise, thus reducing its effects.

Block diagrams like the ones shown in Figure 0.9 allow us to visualize the system much better, and are commonly used. Integrators can be efficiently implemented using operational amplifiers with resistors and capacitors.

How to Obtain Difference Equations

Let us then show how Equation (0.10) can be solved using integration and its approximation, resulting in a difference equation. Using Equation (0.11) at $t = t_1$ and $t = t_0$ for $t_1 > t_0$, we have that

$$v_c(t_1) - v_c(t_0) = \int_{t_0}^{t_1} v_i(\tau)d\tau - \int_{t_0}^{t_1} v_c(\tau)d\tau$$

If we let $t_1 - t_0 = \Delta t$ where $\Delta t \to 0$ (i.e., a very small time interval), the integrals can be seen as the area of small trapezoids of height Δt and bases $v_i(t_1)$ and $v_i(t_0)$ for the input source and $v_c(t_1)$ and $v_c(t_0)$ for the voltage across the capacitor (see Figure 0.10). Using the formula for the area of a trapezoid we get an approximation for the above integrals so that

$$v_c(t_1) - v_c(t_0) = [v_i(t_1) + v_i(t_0)]\frac{\Delta t}{2} - [v_c(t_1) + v_c(t_0)]\frac{\Delta t}{2}$$

from which we obtain

$$v_c(t_1)\left[1 + \frac{\Delta t}{2}\right] = [v_i(t_1) + v_i(t_0)]\frac{\Delta t}{2} + v_c(t_0)\left[1 - \frac{\Delta t}{2}\right]$$

Assuming $\Delta t = T$, we then let $t_1 = nT$ and $t_0 = (n-1)T$. The above equation can be written as

$$v_c(nT) = \frac{T}{2+T}[v_i(nT) + v_i((n-1)T)] + \frac{2-T}{2+T}v_c((n-1)T) \qquad n \geq 1 \tag{0.12}$$

and initial condition $v_c(0) = 0$. This is a first-order linear difference equation with constant coefficients approximating the differential equation characterizing the RC circuit. Letting the input

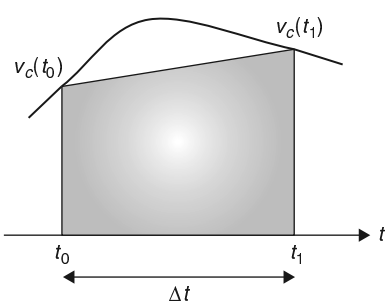


FIGURE 0.10
Approximation of area under the curve by a trapezoid.

be $v_i(t) = 1$ for $t \geq 0$, we have

$$v_c(nT) = \begin{cases} 0 & n = 0 \\ \frac{2T}{2+T} + \frac{2-T}{2+T} v_c((n-1)T) & n \geq 1 \end{cases} \tag{0.13}$$

The advantage of the difference equation is that it can be solved for increasing values of n using previously computed values of $v_c(nT)$, which is called a *recursive solution*. For instance, letting $T = 10^{-3}$, $v_i(t) = 1$, and defining $M = 2T/(2+T)$, $K = (2-T)/(2+T)$, we obtain

$$\begin{aligned}
&n = 0 \qquad v_c(0) = 0 \\
&n = 1 \qquad v_c(T) = M \\
&n = 2 \qquad v_c(2T) = M + KM = M(1+K) \\
&n = 3 \qquad v_c(3T) = M + K(M + KM) = M(1 + K + K^2) \\
&n = 4 \qquad v_c(4T) = M + KM(1 + K + K^2) = M(1 + K + K^2 + K^3) \\
&\ldots
\end{aligned}$$

The values are $M = 2T/(2+T) \approx T = 10^{-3}$, $K = (2-T)/(2+T) < 1$, and $1 - K = M$. The response increases from the zero initial condition to a constant value, which is the effect of the dc source—the capacitor eventually acts as an open circuit, so that the voltage across the capacitor equals that of the input. Extrapolating from the above results it seems that in the steady-state (i.e., when $nT \to \infty$) we have[5]

$$v_c(nT) = M \sum_{m=0}^{\infty} K^m = \frac{M}{1-K} = 1$$

Even though this is a very simple example, it clearly illustrates that very good approximations to the solution of differential equations can be obtained using numerical methods that are appropriate for implementation in digital computers.

The above example shows how to solve a differential equation using integration and approximation of the integrals to obtain a difference equation that a computer can easily solve. The integral approximation used above is the *trapezoidal rule* method, which is one among many numerical methods used to solve differential equations. Also we will see later that the above results in the *bilinear transformation*, which connects the Laplace s variable with the z variable of the Z-transform, and that will be used in Chapter 11 in the design of discrete filters.

[5]The infinite sum converges if $|K| < 1$, which is satisfied in this case. If we multiply the sum by $(1-K)$ we get

$$\begin{aligned}
(1-K) \sum_{m=0}^{\infty} K^m &= \sum_{m=0}^{\infty} K^m - \sum_{m=0}^{\infty} K^{m+1} \\
&= 1 + \sum_{m=1}^{\infty} K^m - \sum_{\ell=1}^{\infty} K^{\ell} = 1
\end{aligned}$$

where we changed the variable in the second equation to $\ell = m + 1$. This explains why the sum is equal to $1/(1-K)$.

0.4 COMPLEX OR REAL?

Most of the theory of signals and systems is based on functions of a complex variable. Clearly, signals are functions of a real variable corresponding to time or space (if the signal is two-dimensional, like an image) so why would one need complex numbers in processing signals? As we will see later, time-dependent signals can be characterized by means of frequency and damping. These two characteristics are given by complex variables such as $s = \sigma + j\Omega$ (where σ is the damping factor and Ω is the frequency) in the representation of analog signals in the Laplace transform, or $z = re^{j\omega}$ (where r is the damping factor and ω is the discrete frequency) in the representation of discrete-time signals in the Z-transform. Both of these transformations will be considered in detail in Chapters 3 and 9. The other reason for using complex variables is due to the response of systems to pure tones or sinusoids. We will see that such response is fundamental in the analysis and synthesis of signals and systems. We thus need a solid grasp of what is meant by complex variables and what a function of these is all about. In this section, complex variables will be connected to vectors and phasors (which are commonly used in the sinusoidal steady-state analysis of linear circuits).

0.4.1 Complex Numbers and Vectors

A complex number z represents any point (x, y) in a two-dimensional plane by $z = x + jy$, where $x = \mathcal{R}e[z]$ (real part of z) is the coordinate in the x axis and $y = \mathcal{I}m[z]$ (imaginary part of z) is the coordinate in the y axis. The symbol $j = \sqrt{-1}$ just indicates that z needs to have two components to represent a point in the two-dimensional plane. Interestingly, a vector $\vec{z}$ that emanates from the origin of the complex plane $(0, 0)$ to the point (x, y) with a length

$$|\vec{z}| = \sqrt{x^2 + y^2} = |z| \tag{0.14}$$

and an angle

$$\theta = \angle\vec{z} = \angle z \tag{0.15}$$

also represents the point (x, y) in the plane and has the same attributes as the complex number z. The couple (x, y) is therefore equally representable by the vector $\vec{z}$ or by a complex number z that can be written in a rectangular or in a polar form,

$$z = x + jy = |z|e^{j\theta} \tag{0.16}$$

where the magnitude $|z|$ and the phase θ are defined in Equations (0.14) and (0.15).

It is important to understand that a rectangular plane or a polar complex plane are identical despite the different representation of each point in the plane. Furthermore, when adding or subtracting complex numbers the rectangular form is the appropriate one, while when multiplying or dividing complex numbers the polar form is more advantageous. Thus, if complex numbers $z = x + jy = |z|e^{j\angle z}$ and $v = p + jq = |v|e^{j\angle v}$ are added analytically, we obtain

$$z + v = (x + p) + j(y + q)$$

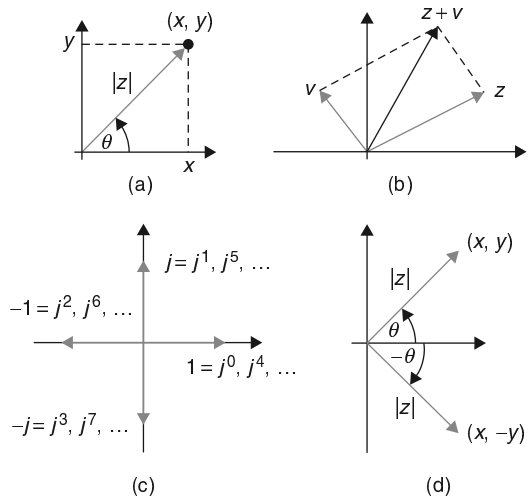


FIGURE 0.11
(a) Representation of a complex number z by a vector (b) addition of complex numbers z and v; (c) integer powers of j; and (d) complex conjugate.

Using their polar representations requires a geometric interpretation: the addition of vectors (see Figure 0.11). On the other hand, the multiplication of z and v is easily done using their polar forms as

$$zv = |z|e^{j\angle z}|v|e^{j\angle v} = |z||v|e^{j(\angle z+\angle v)}$$

but it requires more operations if done in the rectangular form—that is,

$$zv = (x + jy)(p + jq) = (xp - yq) + j(xq + yp)$$

It is even more difficult to obtain a geometric interpretation. Such an interpretation will be seen later on. Addition and subtraction as well as multiplication and division can thus be done more efficiently by choosing the rectangular and the polar representations, respectively. Moreover, the polar representation is also useful when finding powers of complex numbers. For the complex variable $z = |z|e^{\angle z}$, we have that

$$z^n = |z|^n e^{jn\angle z}$$

for n integer or rational. For instance, if $n = 10$, then $z^{10} = |z|^{10}e^{j10\angle z}$, and if $n = 3/2$, then $z^{1.5} = (\sqrt{|z|})^3 e^{j1.5\angle z}$. The powers of j are of special interest. Given that $j = \sqrt{-1}$ then, we have

$$j^n = (-1)^{n/2} = \begin{cases} (-1)^m & n = 2m, \quad n \text{ even} \\ (-1)^m j & n = 2m + 1, \quad n \text{ odd} \end{cases}$$

so that $j^0 = 1$, $j^1 = j$, $j^2 = -1$, $j^3 = -j$, and so on. Letting $j = 1e^{j\pi/2}$, we can see that the increasing powers of $j^n = 1e^{jn\pi/2}$ are vectors with angles of 0 when $n = 0$, $\pi/2$ when $n = 1$, π when $n = 2$, and $3\pi/2$ when $n = 3$. The angles repeat for the next four values, the four after that, and so on. See Figure 0.11.

One operation possible with complex numbers that is not possible with real numbers is *complex conjugation*. Given a complex number $z = x + jy = |z|e^{j\angle z}$ its complex conjugate is $z^* = x - jy = |z|e^{-j\angle z}$—that is, we negate the imaginary part of z or reflect its angle. This operation gives that

$$\begin{aligned}
&(i) \quad z + z^* = 2x \quad \text{or} \quad \mathcal{R}e[z] = 0.5[z + z^*] \\
&(ii) \quad z - z^* = 2jy \quad \text{or} \quad \mathcal{I}m[z] = 0.5[z - z^*] \\
&(iii) \quad zz^* = |z|^2 \quad \text{or} \quad |z| = \sqrt{zz^*} \\
&(iv) \quad \frac{z}{z^*} = e^{j2\angle z} \quad \text{or} \quad \angle z = -j0.5[\log(z) - \log(z^*)]
\end{aligned} \tag{0.17}$$

The complex conjugation provides a different approach to the division of complex numbers in rectangular form. This is done by making the denominator a positive real number by multiplying both numerator and denominator by the complex conjugate of the denominator. For instance,

$$z = \frac{1 + j1}{3 + j4} = \frac{(1 + j1)(3 - j4)}{(3 + j4)(3 - j4)} = \frac{7 - j}{9 + 16} = \frac{7 - j}{25}$$

Finally, the conversion of complex numbers from rectangular to polar needs to be done with care, especially when computing the angles. For instance, $z = 1 + j$ has a vector representing in the first quadrant of the complex plane, and its magnitude is $|z| = \sqrt{2}$ while the tangent of its angle θ is $\tan(\theta) = 1$ or $\theta = \pi/4$ radians. If $z = -1 + j$, the vector representing it is now in the second quadrant with the same magnitude as before, but its angle is now

$$\theta = \pi - \tan^{-1}(1) = 3\pi/4$$

That is, we find the angle with respect to the negative real axis and subtract it from π. Likewise, if $z = -1 - j$, the magnitude does not change but the phase is now

$$\theta = \pi + \tan^{-1}(1) = 5\pi/4$$

which can also be expressed as $-3\pi/4$. Finally, when $z = 1 - j$, the angle is $-\pi/4$ and the magnitude remains the same. The conversion from polar to rectangular form is much easier. Indeed, given a complex number in polar form $z = |z|e^{j\theta}$ its real part is $x = |z|\cos(\theta)$ (i.e., the projection of the vector corresponding to z onto the real axis) and the imaginary part is $y = |z|\sin(\theta)$, so that $z = x + jy$. For instance, $z = \sqrt{2}e^{3\pi/4}$ can be written as

$$z = \sqrt{2}\cos(3\pi/4) + j\sqrt{2}\sin(3\pi/4) = -1 + j$$

0.4.2 Functions of a Complex Variable

Just like real-valued functions, functions of a complex variable can be defined. For instance, the logarithm of a complex number can be written as

$$v = \log(z) = \log(|z|e^{j\theta}) = \log(|z|) + j\theta$$

by using the inverse connection between the exponential and the logarithmic functions. Of particular interest in the theory of signals and systems is the exponential of complex variable z defined as

$$v = e^z = e^{x+jy} = e^x e^{jy}$$

It is important to mention that complex variables as well as functions of complex variables are more general than real variables and real-valued functions. The above definition of the logarithmic function is valid when $z = x$, with x a real value, and also when $z = jy$, a purely imaginary value. Likewise, the exponential function for $z = x$ is a real-valued function.

Euler's Identity

One of the most famous equations of all times[6] is

$$1 + e^{j\pi} = 1 + e^{-j\pi} = 0$$

due to one of the most prolific mathematicians of all times, Leonard Euler.[7] The above equation can be easily understood by establishing Euler's identity, which connects the complex exponential and sinusoids:

$$e^{j\theta} = \cos(\theta) + j\sin(\theta) \tag{0.18}$$

One way to verify this identity is to consider the polar representation of the complex number $\cos(\theta) + j\sin(\theta)$, which has a unit magnitude since $\sqrt{\cos^2(\theta) + \sin^2(\theta)} = 1$ given the trigonometric identity $\cos^2(\theta) + \sin^2(\theta) = 1$. The angle of this complex number is

$$\psi = \tan^{-1}\left[\frac{\sin(\theta)}{\cos(\theta)}\right] = \theta$$

Thus, the complex number

$$\cos(\theta) + j\sin(\theta) = 1e^{j\theta}$$

which is Euler's identity. Now in the case where $\theta = \pm\pi$ the identity implies that $e^{\pm j\pi} = -1$, explaining the famous Euler's equation.

---

[6] A reader's poll done by *Mathematical Intelligencer* named Euler's identity the most beautiful equation in mathematics. Another poll by *Physics World* in 2004 named Euler's identity the greatest equation ever, together with Maxwell's equations. Paul Nahin's book *Dr. Euler's Fabulous Formula* (2006) is devoted to Euler's identity. It states that the identity sets "the gold standard for mathematical beauty" [73].

[7] Leonard Euler (1707–1783) was a Swiss mathematician and physicist, student of John Bernoulli, and advisor of Joseph Lagrange. We owe Euler the notation $f(x)$ for functions, e for the base of natural logs, $i = \sqrt{-1}$, π for pi, Σ for sum, the finite difference notation Δ, and many more!

The relation between the complex exponentials and the sinusoidal functions is of great importance in signals and systems analysis. Using Euler's identity the cosine can be expressed as

$$\cos(\theta) = \mathcal{R}e[e^{j\theta}] = \frac{e^{j\theta} + e^{-j\theta}}{2} \tag{0.19}$$

while the sine is given by

$$\sin(\theta) = \mathcal{I}m[e^{j\theta}] = \frac{e^{j\theta} - e^{-j\theta}}{2j} \tag{0.20}$$

Indeed, we have

$$e^{j\theta} = \cos(\theta) + j\sin(\theta)$$
$$e^{-j\theta} = \cos(\theta) - j\sin(\theta)$$

Adding them we get the above expression for the cosine, and subtracting the second from the first we get the given expression for the sine. The variable θ is in radians, or in the corresponding angle in degrees (recall that 2π radians equals 360 degrees).

These relations can be used to define the hyperbolic sinusoids as

$$\cos(j\alpha) = \frac{e^{-\alpha} + e^{\alpha}}{2} = \cosh(\alpha) \tag{0.21}$$

$$j\sin(j\alpha) = \frac{e^{-\alpha} - e^{\alpha}}{2} = -\sinh(\alpha) \tag{0.22}$$

from which the other hyperbolic functions are defined. Also, we obtain the following expression for the real-valued exponential:

$$e^{-\alpha} = \cosh(\alpha) - \sinh(\alpha) \tag{0.23}$$

Euler's identity can also be used to find different trigonometric identities. For instance,

$$\cos^2(\theta) = \left[\frac{e^{j\theta} + e^{-j\theta}}{2}\right]^2 = \frac{1}{4}[2 + e^{j2\theta} + e^{-j2\theta}] = \frac{1}{2} + \frac{1}{2}\cos(2\theta)$$

$$\sin^2(\theta) = 1 - \cos^2(\theta) = \frac{1}{2} - \frac{1}{2}\cos(2\theta)$$

$$\sin(\theta)\cos(\theta) = \frac{e^{j\theta} - e^{-j\theta}}{2j}\,\frac{e^{j\theta} + e^{-j\theta}}{2} = \frac{e^{j2\theta} - e^{-j2\theta}}{4j} = \frac{1}{2}\sin(2\theta)$$

0.4.3 Phasors and Sinusoidal Steady State

A sinusoid $x(t)$ is a periodic signal represented by

$$x(t) = A\cos(\Omega_0 t + \psi) \qquad -\infty < t < \infty \tag{0.24}$$

where A is the amplitude, $\Omega_0 = 2\pi f_0$ is the frequency in rad/sec, and ψ is the phase in radians. The signal $x(t)$ is defined for all values of t, and it repeats periodically with a period $T_0 = 1/f_0$ (sec), so

that f_0 is the frequency in cycles/sec or in Hertz (Hz) (in honor of H. R. Hertz[8]). Given that the units of Ω_0 is rad/sec, then $\Omega_0 t$ has as units (rad/sec) $\times$ (sec) = (rad), which coincides with the units of the phase ψ, and permits the computation of the cosine. If $\psi = 0$, then $x(t)$ is a cosine, and if $\psi = -\pi/2$, then $x(t)$ is a sine.

If one knows the frequency Ω_0 (rad/sec) in Equation (0.24), the cosine is characterized by its amplitude and phase. This permits us to define *phasors*[9] as complex numbers characterized by the amplitude and the phase of a cosine signal of a certain frequency Ω_0. That is, for a voltage signal $v(t) = A\cos(\Omega_0 t + \psi)$ the corresponding phasor is

$$V = Ae^{j\psi} = A\cos(\psi) + jA\sin(\psi) = A\angle\psi \tag{0.25}$$

and such that

$$v(t) = \mathcal{R}e[Ve^{j\Omega_0 t}] = \mathcal{R}e[Ae^{j(\Omega_0 t+\psi)}] = A\cos(\Omega_0 t + \psi) \tag{0.26}$$

One can thus think of the voltage signal $v(t)$ as the projection of the phasor V onto the real axis and turning counterclockwise at a rate of Ω_0 rad/sec. At time $t = 0$ the angle of the phasor is ψ. Clearly the phasor definition is true for only one frequency, in this case Ω_0, and it is always connected to a cosine function.

Interestingly enough, the angle ψ can be used to differentiate cosines and sines. For instance, when $\psi = 0$, the phasor V moving around at a rate of Ω_0 generates as a projection on the real axis the voltage signal $A\cos(\Omega_0 t)$, while when $\psi = -\pi/2$, the phasor V moving around again at a rate of Ω_0 generates a sinusoid $A\sin(\Omega_0 t) = A\cos(\Omega_0 t - \pi/2)$ as it is projected onto the real axis. This establishes the well-known fact that the sine lags the cosine by $\pi/2$ radians or 90 degrees, or that the cosine leads the sine by $\pi/2$ radians or 90 degrees. Thus, the generation and relation of sines and cosines can be easily obtained using the plot in Figure 0.12.

Phasors can be related to vectors. A current source, for instance,

$$i(t) = A\cos(\Omega_0 t) + B\sin(\Omega_0 t)$$

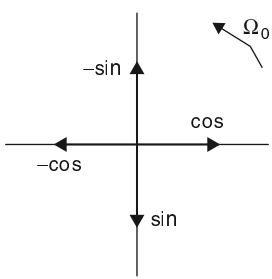


FIGURE 0.12
Generation of sinusoids from phasors of a frequency Ω_0.

[8] Heinrich Rudolf Hertz was a German physicist known for being the first to demonstrate the existence of electromagnetic radiation in 1888.

[9] In 1883, Charles Proteus Steinmetz (1885–1923), German-American mathematician and engineer, introduced the concept of phasors for alternating current analysis. In 1902, Steinmetz became a professor of electrophysics at Union College in Schenectady, New York.

can be expressed as

$$i(t) = C\cos(\Omega_0 t + \gamma)$$

where C and γ are to be determined (the sinusoidal components of $i(t)$ must depend on a unique frequency Ω_0; if that was not the case the concept of phasors would not apply). To obtain the equivalent representation, we first obtain the phasor corresponding to $A\cos(\Omega_0 t)$, which is $I_1 = Ae^{j0} = A$, and for $B\sin(\Omega_0 t)$ the corresponding phasor is $I_2 = Be^{-j\pi/2}$, so that

$$i(t) = \mathcal{R}e[(I_1 + I_2)e^{j\Omega_0 t}]$$

Thus, the problem has been transformed into the addition of two vectors I_1 and I_2, which gives a vector

$$I = \sqrt{A^2 + B^2}e^{-j\tan^{-1}(B/A)}$$

so that

$$\begin{aligned} i(t) &= \mathcal{R}e[Ie^{j\Omega_0 t}] \\ &= \mathcal{R}e[\sqrt{A^2 + B^2}\, e^{-j\tan^{-1}(B/A)}e^{j\Omega_0 t}] \\ &= \sqrt{A^2 + B^2}\cos(\Omega_0 t - \tan^{-1}(B/A)) \end{aligned}$$

Or, an equivalent source with amplitude $C = \sqrt{A^2 + B^2}$, phase $\gamma = -\tan^{-1}(B/A)$, and frequency Ω_0– that is, an equivalent phasor that generates $i(t)$ and has the magnitude C, the angle γ, and rotates at frequency Ω_0.

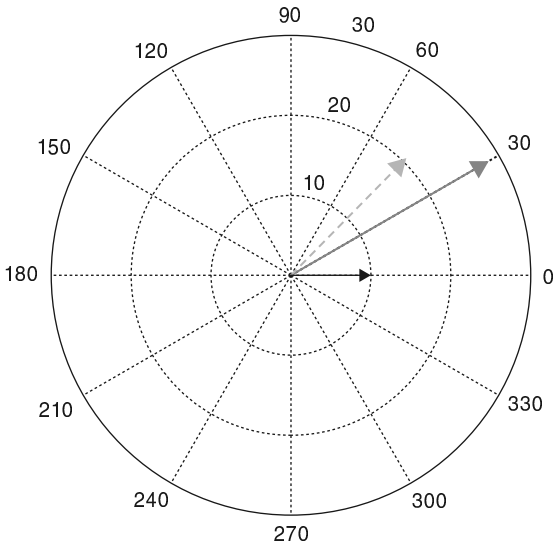
In Figure 0.13 we display the result of adding two phasors (frequency $f_0 = 20$ Hz) and the sinusoid that is generated by the phasor $I = I_1 + I_2 = 27.98e^{j30.4^o}$.

0.4.4 Phasor Connection

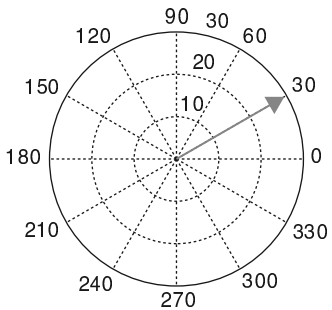
The fundamental property of a circuit made up of constant resistors, capacitors, and inductors is that its response to a sinusoid is also a sinusoid of the same frequency in steady state. The effect of the circuit on the input sinusoid is on its magnitude and phase and depends on the frequency of the input sinusoid. This is due to the linear and time-invariant nature of the circuit, and can be generalized to more complex continuous-time as well as discrete-time systems as we will see in Chapters 3, 4, 5, 9 and 10.

To illustrate the connection of phasors with dynamic systems consider a simple RC circuit ($R = 1\ \Omega$ and $C = 1$F). If the input to the circuit is a sinusoidal voltage source $v_i(t) = A\cos(\Omega_0 t)$ and the voltage across the capacitor $v_c(t)$ is the output of interest, the circuit can be easily represented by the first-order differential equation

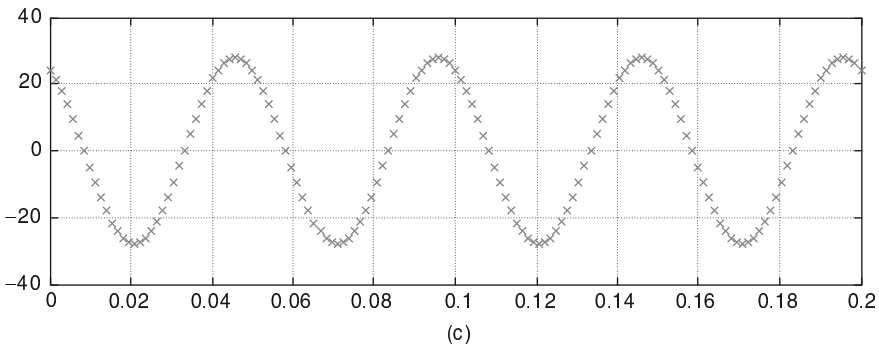
$$\frac{dv_c(t)}{dt} + v_c(t) = v_i(t)$$



(a)



(b)



(c)

FIGURE 0.13

(a) Sum of phasors $I_1 = 10e^{j0}$ (solid arrow) and $I_2 = 20e^{j\pi/4}$ (dashed arrow) with the result in blue; (c) sinusoid generated by phasor $I = I_1 + I_2$ (b).

Assume that the steady-state response of this circuit (i.e., $v_c(t)$ as $t \to \infty$) is also a sinusoid

$$v_c(t) = C\cos(\Omega_0 t + \psi)$$

of the same frequency as the input, with amplitude C and phase ψ to be determined. This response must satisfy the differential equation, or

$$\begin{aligned} v_i(t) &= \frac{dv_c(t)}{dt} + v_c(t) \\ A\cos(\Omega_0 t) &= -C\Omega_0 \sin(\Omega_0 t + \psi) + C\cos(\Omega_0 t + \psi) \\ &= C\Omega_0 \cos(\Omega_0 t + \psi + \pi/2) + C\cos(\Omega_0 t + \psi) \\ &= C\sqrt{1+\Omega_0^2}\cos(\Omega_0 t + \psi + \tan^{-1}(C\Omega_0/C)) \end{aligned}$$

Comparing the two sides of the above equation gives

$$\begin{aligned} C &= \frac{A}{\sqrt{1+\Omega_0^2}} \\ \psi &= -\tan^{-1}(\Omega_0) \end{aligned}$$

for a steady-state response

$$v_c(t) = \frac{A}{\sqrt{1+\Omega_0^2}}\cos(\Omega_0 t - \tan^{-1}(\Omega_0)).$$

Comparing the steady-state response $v_c(t)$ with the input sinusoid $v_i(t)$, we see that they both have the same frequency Ω_0, but the amplitude and phase of the input are changed by the circuit depending on the frequency Ω_0. Since at each frequency the circuit responds differently, obtaining the frequency response of the circuit will be useful not only in analysis but in the design of circuits.

The sinusoidal steady-state is obtained using phasors. Expressing the steady-state response of the circuit as

$$v_c(t) = \mathcal{R}e\left[V_c e^{j\Omega_0 t}\right]$$

where $V_c = Ce^{j\psi}$ is the corresponding phasor for $v_c(t)$, we find that

$$\frac{dv_c(t)}{dt} = \frac{d\mathcal{R}e[V_c e^{j\Omega_0 t}]}{dt} = \mathcal{R}e\left[V_c \frac{de^{j\Omega_0 t}}{dt}\right] = \mathcal{R}e\left[j\Omega_0 V_c e^{j\Omega_0 t}\right]$$

By replacing $v_c(t)$, $dv_c(t)/dt$, obtained above, and

$$v_i(t) = \mathcal{R}e\left[V_i e^{j\Omega_0 t}\right] \quad \text{where} \quad V_i = Ae^{j0}$$

in the differential equation, we obtain

$$\mathcal{R}e\left[V_c(1+j\Omega_0)e^{j\Omega_0 t}\right] = \mathcal{R}e\left[Ae^{j\Omega_0 t}\right]$$

so that

$$\begin{aligned} V_c &= \frac{A}{1+j\Omega_0} = \frac{A}{\sqrt{1+\Omega_0^2}} e^{-j\tan^{-1}(\Omega_0)} \\ &= Ce^{j\psi} \end{aligned}$$

and the sinusoidal steady-state response is

$$\begin{aligned} v_c(t) &= \mathcal{R}e\left[V_c e^{j\Omega_0 t}\right] \\ &= \frac{A}{\sqrt{1+\Omega_0^2}} \cos(\Omega_0 t - \tan^{-1}(\Omega_0)) \end{aligned}$$

which coincides with the response obtained above. The ratio of the output phasor V_c to the input phasor V_i,

$$\frac{V_c}{V_i} = \frac{1}{1+j\Omega_0}$$

gives the response of the circuit at frequency Ω_0. If the frequency of the input is a generic Ω, changing Ω_0 above for Ω gives the frequency response for all possible frequencies.

The concepts of *linearity* and *time invariance* will be used in both continuous-time as well as discrete-time systems, along with the Fourier representation of signals in terms of sinusoids or complex exponentials, to simplify the analysis and to allow the design of systems. Thus, transform methods such as Laplace and the Z-transform will be used to solve differential and difference equations in an algebraic setup. Fourier representations will provide the frequency perspective. This is a general approach for both continuous-time and discrete-time signals and systems. The introduction of the concept of the transfer function will provide tools for the analysis as well as the design of linear time-invariant systems. The design of analog and discrete filters is the most important application of these concepts. We will look into this topic in Chapters 5, 6, and 11.

0.5 SOFT INTRODUCTION TO MATLAB

MATLAB is a computing language based on vectorial computations.[10] In this section, we will introduce you to the use of MATLAB for numerical and symbolic computations.

[10] MATLAB stands for matrix laboratory. MatWorks, the developer of MATLAB, was founded in 1984 by Jack Little, Steve Bangert, and Cleve Moler. Moler, a math professor at the University of New Mexico, developed the first version of MATLAB in Fortran in the late 1970s. It only had 80 functions and no M-files or toolboxes. Little and Bangert reprogrammed it in C and added M-files, toolboxes, and more powerful graphics [49].

0.5.1 Numerical Computations

The following instructions are intended for users who have no background in MATLAB but are interested in using it in signal processing. Once you get the basic information on how to use the language you will be able to progress on your own.

1. Create a directory where you will put your work, and from where you will start MATLAB. This is important because when executing a program, MATLAB will look at the current directory, and if the file is not present in the current directory, and if it is not a MATLAB function, MATLAB gives an error indicating that it cannot find the desired program.
2. There are two types of programs in MATLAB: the script, which consists in a list of commands using MATLAB functions or your own functions, and the functions, which are programs that can be called with different inputs and provide the corresponding outputs. We will show examples of both.
3. Once you start MATLAB, you will see three windows: the command window, where you will type commands; the command history, which keeps a list of commands that have been used; and the workspace, where the variables used are kept.
4. Your first command on the command window should be to change to your data directory where you will keep your work. You can do this in the command window by using the command CD (change directory) followed by the desired directory. It is also important to use the command clear all and clf to clear all previous variables in memory and all figures.
5. Help is available in several forms in MATLAB. Just type helpwin, helpdesk, or demo to get started. If you know the name of the function, help will give you the necessary information on the particular function, and it will also give you information on help itself. Use help to find more about the functions used in this introduction to MATLAB.
6. To type your scripts or functions you can use the editor provided by MATLAB; simply type edit. You can also use any text editor to create scripts or functions, which need to be saved with the .m extension.

Creating Vectors and Matrices

Comments are preceded by percent, and to begin a script, as the following, it is always a good idea to clear all previous variables and all previous figures.

```
% matlab primer
clear all                % clear all variables
clf                      % clear all figures
% row and column vectors
x = [ 1 2 3 4]           % row vector
y = x'                   % column vector
```

The corresponding output is as follows (notice that there is no semicolon (;) at the end of the lines to stop MATLAB from providing an output when the above script is executed).

```
x =
    1    2    3    4
```

```
y =
     1
     2
     3
     4
```

To see the dimension of x and y variables, type

```
whos          % provides information on existing variables
```

to which MATLAB responds

```
Name      Size                Bytes  Class
x         1x4                    32  double array
y         4x1                    32  double array
Grand total is 8 elements using 64 bytes
```

Notice that a vector is thought of as a matrix; for instance, vector x is a matrix of one row and four columns. Another way to express the column vector y is the following, where each of the row terms is separated by a semicolon (;)

```
y = [1;2;3;4]     % another way to write a column
```

To give as before:

```
y =
     1
     2
     3
     4
```

MATLAB does not allow arguments of vectors or matrices to be zero or negative. For instance, if we want the first entry of the vector y we need to type

```
y(1)      % first entry of vector y
```

giving as output

```
ans =
        1
```

If we type

```
y(0)
```

it will give us an error, to which we get the following warning:

```
??? Subscript indices must either be real positive integers or logicals.
```

MATLAB also has a peculiar way to provide information in a vector, for instance:

```
y(1:3)          % first to third entry of column vector y
```

giving as expected the first to the third entries of the column vector *y*:

```
ans =
     1
     2
     3
```

The following will give the third to the first entry in the row vector *x* (notice the difference in the two outputs; as expected the values of *y* are given in a column, while the requested entries of *x* are given in a row).

```
x(3:-1:1)          % displays entries x(3) x(2) x(1)
```

Thus,

```
ans =
     3     2     1
```

Matrices are constructed as an concatenation of rows (or columns):

```
A = [ 1 2; 3 4; 5 6]    % matrix A with rows [1 2], [3 4] and [5 6]
```

```
A =
     1     2
     3     4
     5     6
```

To create a vector corresponding to a sequence of numbers (in this case integers) there are different approaches, as follows:

```
n = 0:10     % vector with entries 0 to 10 increased by 1
```

This approach gives the following as output:

```
n =
   Columns 1 through 10
      0     1     2     3     4     5     6     7     8     9
   Column 11
      10
```

which is the same as the command

```
n = [0:10]
```

If we wish the increment different from 1 (default value), then we indicate it as in the following:

```
n1 = 0:2:10  % vector with entries from 0 to 10 increased by 2
```

which gives

```
n1 =
     0     2     4     6     8    10
```

We can combine the above vectors into one as follows:

```
nn1 = [n n1]    % combination of vectors
```

to get

```
nn1 =
    Columns 1 through 10
      0    1    2    3    4    5    6    7    8    9
    Columns 11 through 17
     10    0    2    4    6    8   10
```

Vectorial Operations

MATLAB allows the conventional vectorial operations as well as facilitates others. For instance, if we wish to multiply by 3 every entry of the row vector x given above, the command

```
z = 3*x            % multiplication by a constant
```

would give

```
z =
    3    6    9   12
```

Besides the conventional multiplication of vectors with the correct dimensions, MATLAB allows two types of multiplications of one vector by another. The first one is where the entries of one vector are multiplied by the corresponding entries of the other. To effect this the two vectors should have the same dimension (i.e., both should be columns or rows with the same number of entries) and it is necessary to put a dot before the multiplication operator—that is, as shown here:

```
v = x.*x  % multiplication of entries of two vectors

v =
    1    4    9   16
```

The other type of multiplication is the conventional multiplication allowed in linear algebra. For instance, with that of a row vector by a column vector,

```
w = x*x'   % multiplication of x (row vector) by x'(column vector)

w = 30
```

the result is a constant—in this case, the length of the row vector should coincide with that of the column vector. If you multiply a column (say x') of dimension 4×1 by a row (say x) of dimension 1×4 (notice that the 1s coincide at the end of the first dimension and at the beginning of the second), the multiplication $z = x' * x$ results in a 4×4 matrix.

The solution of a set of linear equations is very simple in MATLAB. To guarantee that a unique solution exists, the determinant of the matrix should be computed before inverting the matrix. If the determinant is zero MATLAB will indicate the solution is not possible.

```
% Solution of linear set of equations Ax = b
A = [1 0 0; 2 2 0; 3 3 3];      % 3x3 matrix
t = det(A);                          % MATLAB function that calculates determinant
b = [2 2 2]';              % column vector
x = inv(A)*b;          % MATLAB function that inverts a matrix
```

The results of these operations are not given because of the semicolons at the end of the commands. The following script could be used to display them:

```
disp(' Ax = b')          % MATLAB function that displays the text in ' '
A
b
x
t
```

which gives

```
Ax = b
A =
     1     0     0
     2     2     0
     3     3     3
b =
     2
     2
     2
x =
      2.0000
     −1.0000
     −0.3333
t =
     6
```

Another way to solve this set of equations is

```
x = b'/A'
```

Try it!

MATLAB provides a fast way to obtain certain vectors/matrices; for instance,

```
% special vectors and matrices
x = ones(1, 10)    % row of ten 1s

x =
     1     1     1     1     1     1     1     1     1     1

A = ones(5, 5)     % matrix of 5 x 5 1s

A =
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1

x1 = [x zeros(1, 5)]    % vector with previous x and 5 0s
```

```
x1 =
  Columns 1 through 10
     1     1     1     1     1     1     1     1     1     1
  Columns 11 through 15
     0     0     0     0     0
```

```
A(2:5, 2:5) = zeros(4, 4)   % zeros in rows 2−5, columns 2−5
```

```
A =
     1     1     1     1     1
     1     0     0     0     0
     1     0     0     0     0
     1     0     0     0     0
     1     0     0     0     0
```

```
y = rand(1,10)       % row vector with 10 random values (uniformly
                                % distributed in [0,1]
```

```
y =
  Columns 1 through 6
    0.9501    0.2311    0.6068    0.4860    0.8913    0.7621
  Columns 7 through 10
    0.4565    0.0185    0.8214    0.4447
```

Notice that these values are between 0 and 1. When using the normal or Gaussian-distributed noise the values can be positive or negative reals.

```
y1 = randn(1,10)          % row vector with 10 random values
                                                % (Gaussian distribution)
```

```
y1 =
  Columns 1 through 6
    −0.4326   −1.6656    0.1253    0.2877   −1.1465    1.1909
  Columns 7 through 10
     1.1892   −0.0376    0.3273    0.1746
```

Using Built-In Functions and Creating Your Own

MATLAB provides a large number of built-in functions. The following script uses some of them.

```
% using built-in functions
t = 0:0.01:1;    % time vector from 0 to 1 with interval of 0.01
x = cos(2∗pi∗t/0.1);      % cos processes each of the entries in
     % vector t to get the corresponding value in vector x
% plotting the function x
figure(1)   % numbers the figure
plot(t, x)   % interpolated continuous plot
xlabel('t (sec)')   % label of x-axis
ylabel('x(t)')   % label of y-axis
```

```
 % let's hear it
sound(1000*x, 10000)
```

The results are given in Figure 0.14.

To learn about any of these functions use *help*. In particular, use *help* to learn about MATLAB routines for plotting *plot* and *stem*. Use *help sound* and *help waveplay* to learn about the sound routines available in MATLAB. Additional related functions are put at the end of these help files. Explore all of these and become aware of the capabilities of MATLAB. To illustrate the plotting and the sound routines, let us create a chirp that is a sinusoid for which the frequency is varying with time.

```
y = sin(2*pi*t.^2/.1);   % notice the dot in the squaring
                                  % t was defined before
sound(1000*y, 10000)  % to listen to the sinusoid
figure(2) % numbering of the figure
plot(t(1:100), y(1:100))  % plotting of 100 values of y
figure(3)
plot(t(1:100), x(1:100), 'k', t(1:100), y(1:100), 'r')  % plotting x and y on same plot
```

Let us hope you were able to hear the chirp, unless you thought it was your neighbor grunting. In this case, we plotted the first 100 values of t and y and let MATLAB choose the color for them. In the second plot we chose the colors: black (dashed lines) for x and blue (continuous line) for the second signal $y(t)$ (see Figure 0.15).

Other built-in functions are sin, tan, acos, asin, atan, atan2, log, log10, exp, etc. Find out what each does using help and obtain a listing of all the functions in the signal processing toolbox.

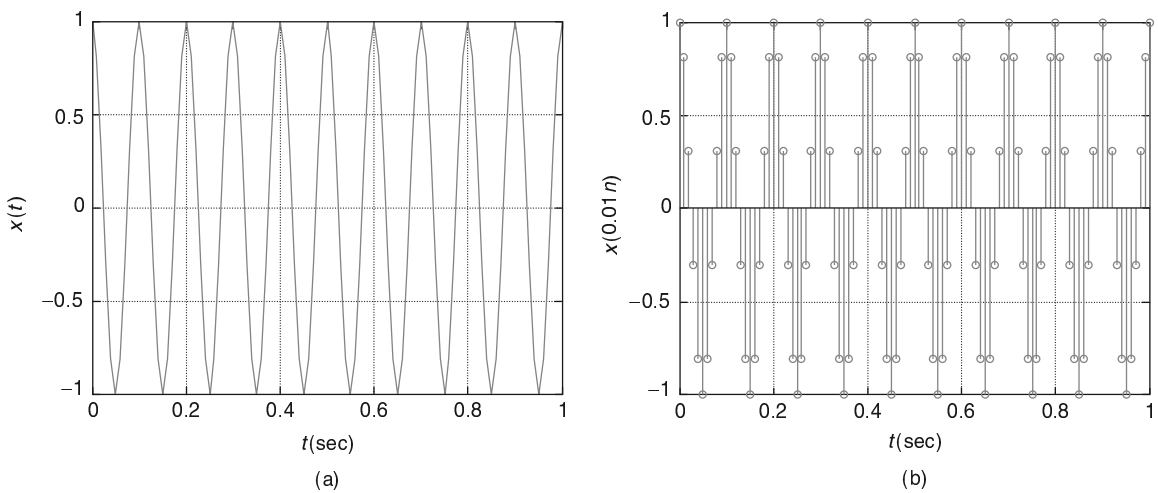


FIGURE 0.14
(a) Plotting of a sinusoid using *plot*, which gives a continuous plot, and (b) *stem*, which gives a discrete plot.

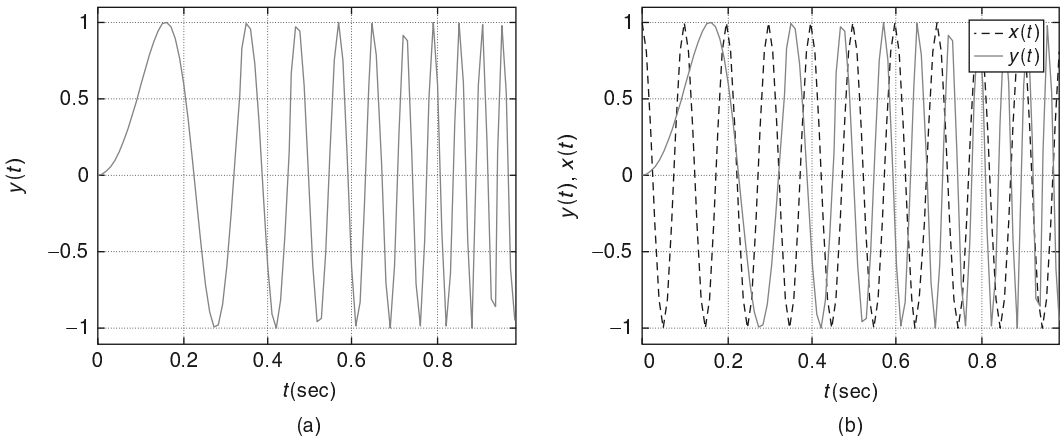


FIGURE 0.15
(a) Plotting chirp (MATLAB chooses color), (b) sinusoid and chirp (the sinusoid is plotted with dashed lines and the chirp with solid lines).

You do not need to define π, as it is already done in MATLAB. For complex numbers also you do not need to define the square root of -1, which for engineers is 'j' and for mathematicians 'i' (they have no current to worry about).

```
% pi and j
pi
j
i

ans =
    3.1416
ans =
          0 + 1.0000i
ans =
          0 + 1.0000i
```

Creating Your Own Functions

MATLAB has created a lot of functions to make our lives easier, and it allows us also to create—in the same way—our own. The following file is for a function f with an input of a scalar x and output of a scalar y related by a mathematical function:

```
function y = f(x)
y = x*exp(-sin(x))/(1 + x^2);
```

Functions cannot be executed on their own—they need to be part of a script. If you try to execute the above function MATLAB will give the following:

```
??? format compact;function y = f(x)
                   |
Error: A function declaration cannot appear within a script M-file.
```

A function is created using the word "function" and then defining the output (y), the name of the function (f), and the input of the function (x), followed by lines of code defining the function, which in this case is given by the second line. In our function the input and the output are scalars. If you want vectors as input/output you need to do the computation in vectorial form—more later.

Once the function is created and saved (the name of the function followed by the extension .m), MATLAB will include it as a possible function that can be executed within a script. If we wish to compute the value of the function for $x = 2$ ($f.m$ should be in the working directory) we proceed as follows:

$$y = f(2)$$

gives

$$y = 0.1611$$

To compute the value of the function for a vector as input, we compute for each of the values in the vector the corresponding output using a for loop as shown in the following.

```
x = 0:0.1:100;          % create an input vector x
N = length(x);          % find the length of x
y = zeros(1,N);         % initialize the output y to zeros
for n = 1:N,              % for the variable n from 1 to N, compute
    y(n) = f(x(n));         % the function
end
figure(3)
plot(x, y)
grid                          % put a grid on the figure
title('Function f(x)')
xlabel('x')
ylabel('y')
```

This is not very efficient. A general rule in MATLAB is: Loops are to be avoided, and vectorial computations are encouraged. The results are shown in Figure 0.16.

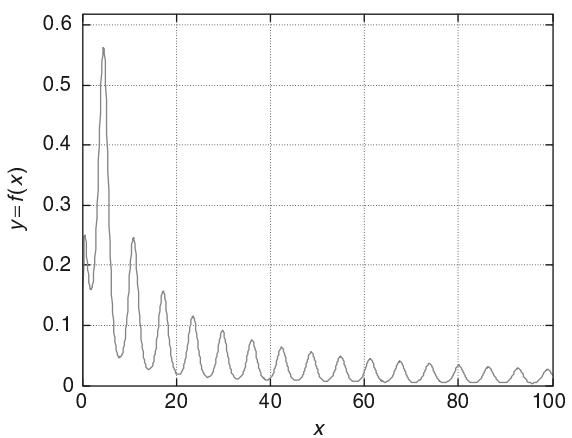


FIGURE 0.16
Result of using the function $f(.)$

The function working on a vector x, rather than one value, takes the following form (to make it different from the above function we let the denominator be $1 + x$ instead of $1 + x^2$):

```
function yy = ff(x)
% vectorial function
yy = x.*exp(-sin(x))./(1 + x);
```

Again, this function must be in the working directory. Notice that the computation of yy is done considering x a vector; the .* and ./ are indicative of this. Thus, this function will accept a vector x and will give as output a vector yy, computed as indicated in the last line. When we use a function, the names of the variables used in the script that calls the function do not need to coincide with the ones in the definition of the function. Consider the following script:

```
z = ff(x);     % x defined before,
                    % z instead of yy is the output of the function ff
figure(4)
plot(x, z); grid
title('Function ff(x)') % MATLAB function that puts title in plot
xlabel('x')  % MATLAB function to label x-axis
ylabel('z')   % MATLAB function to label y-axis
```

The difference between plot and stem is important. The function plot interpolates the vector to be plotted and so the plot appears continuous, while stem simply plots the entries of the vector, separating them uniformly. The input x and the output of the function are discrete time and we wish to plot them as such, so we use stem.

```
stem(x(1:30),  z(1:30))
grid
title('Function ff(x)')
xlabel('x')
ylabel('z')
```

The results are shown in Figure 0.17.

More on Plotting

There are situations where we want to plot several plots together. One can superpose two or more plots by using hold on and hold off. To put several figures in the same plot, we can use the function subplot. Suppose we wish to plot four figures in one plot and they could be arranged as two rows of two figures each. We do the following:

```
subplot(221)
plot(x, y)
subplot(222)
plot(x, z)
subplot(223)
stem(x, y)
subplot(224)
stem(x, z)
```

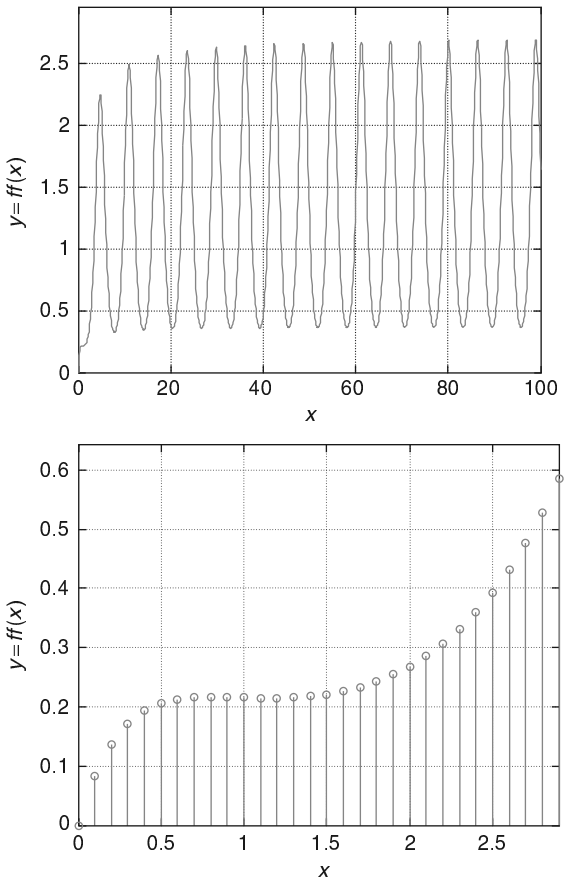


FIGURE 0.17
Results of using the function ff(.) (notice the difference in scale in the x axis).

In the subplot function the first two numbers indicate the number of rows and the number of columns, and the last digit refers to the order of the graph that is, 1, 2, 3, and 4 (see Figure 0.18).

There is also a way to control the values in the axis, by using the function (you guessed!) **axis**. This function is especially useful after we have a graph and want to improve its looks. For instance, suppose that the professor would like the above graphs to have the same scales in the y-axis (picky professor). You notice that there are two scales in the y-axis, one 0-0.8 and another 0-3. To have both with the same scale, we choose the one 0-3, and modify the above code to the following

```
subplot(221)
plot(x, y)
axis([0 100 0 3])
subplot(222)
plot(x, z)
axis([0 100 0 3])
subplot(223)
stem(x, y)
```

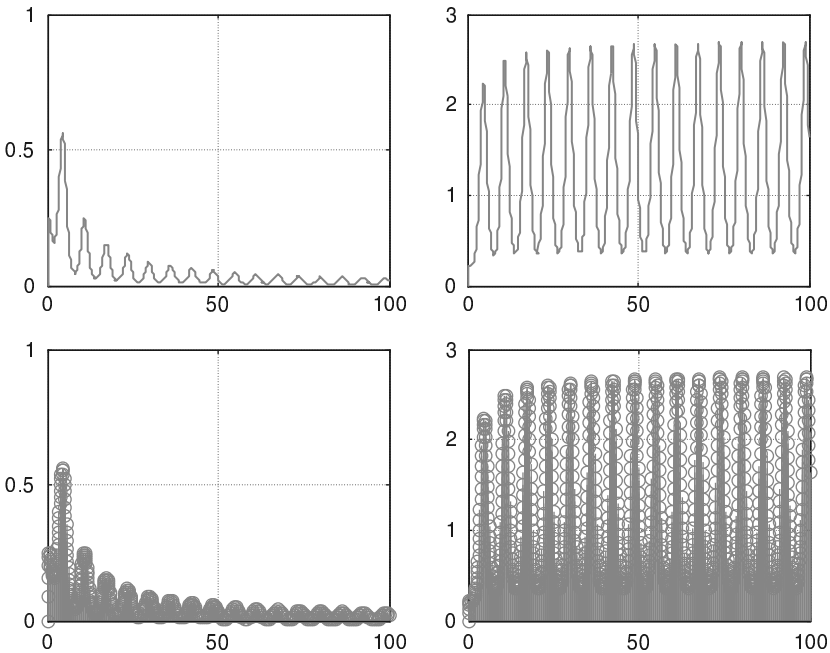


FIGURE 0.18
Plotting four figures in one.

```
axis([0 100 0 3])
subplot(224)
stem(x, z)
axis([0 100 0 3])
```

Saving and Loading Data

In many situations you would like to either save some data or load some data. The following is one way to do it. Suppose you want to build and save a table of sine values for angles between 0 and 360 degrees in intervals of 3 degrees. This can be done as follows:

```
x = 0:3:360;
y = sin(x*pi/180);  % sine computes the argument in radians
xy = [x' y'];  % vector with 2 columns one for x'
                    % and  another for y'
```

Let's now save these values in a file "sine.mat" by using the function save (use help save to learn more):

```
save sine.mat xy
```

To load the table, we use the function load with the name given to the saved table "sine" (the extension *.mat is not needed). The following script illustrates this:

```
clear all
load sine
whos
```

where we use whos to check its size:

```
Name      Size              Bytes  Class
xy        121x2              1936  double array
Grand total is 242 elements using 1936 bytes
```

This indicates that the array *xy* has 121 rows and 2 columns, the first colum corresponding to *x*, the degree values, and the second column corresponding to the sine values, *y*. Verify this and plot the values by using

```
x = xy(:, 1);
y = xy(:, 2);
stem(x, y)
```

Finally, MATLAB provides some data files for experimentation and you only need to load them. The following "train.mat" is the recording of a train whistle, sampled at the rate of F_s samples/sec, which accompanies the sampled signal $y(n)$ (see Figure 0.19).

```
clear all
load train
whos
```

```
Name          Size              Bytes  Class

Fs               1x1                8  double array
y            12880x1           103040  double array

Grand total is 12881 elements using 103048 bytes
```

```
sound(y, Fs)
plot(y)
```

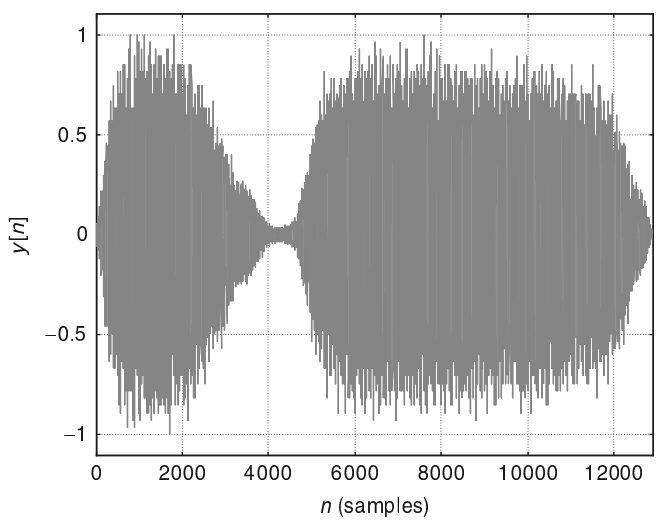


FIGURE 0.19
Train signal.

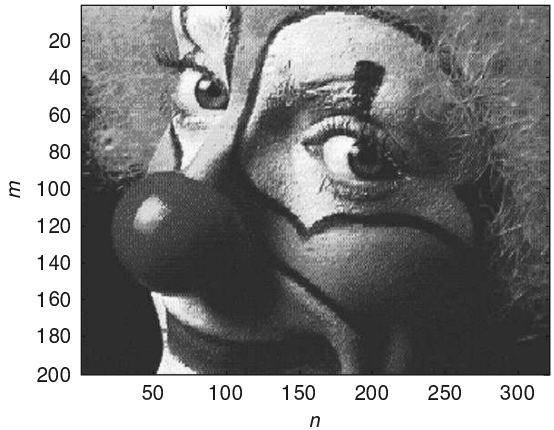


FIGURE 0.20
Clown in gray scale.

MATLAB also provides two-dimensional signals, or images, such as "clown.mat," a 200×320 pixels image.

```
clear all
load clown
whos
```

Name	Size	Bytes	Class
X	200x320	512000	double array
caption	2x1	4	char array
map	81x3	1944	double array

Grand total is 64245 elements using 513948 bytes

We can display this image in gray levels by using the following script (see Figure 0.20):

```
colormap('gray')
imagesc(X)
```

Or in color using

```
colormap('hot')
imagesc(X)
```

0.5.2 Symbolic Computations

We have considered the numerical capabilities of MATLAB, by which numerical data are transformed into numerical data. There will be many situations when we would like to do algebraic or calculus operations resulting in terms of variables rather than numerical data. For instance, we might want to find a formula to solve quadratic algebraic equations, to find a difficult integral, or to obtain the Laplace or the Fourier transform of a signal. For those cases MATLAB provides the Symbolic Math Toolbox, which uses the interface between MATLAB and MAPLE, a symbolic computing system. In this section, we provide you with an introduction to symbolic computations by means of examples, and hope to get you interested in learning more on your own.

Derivatives and Differences

The following script compares symbolic with numeric computations of the derivative of a chirp signal (a sinusoid with changing frequency) $y(t) = \cos(t^2)$, which is

$$z(t) = \frac{dy(t)}{dt} = -2t\sin(t^2)$$

```
clf; clear all
% symbolic
   syms t y z  % define the symbolic variables
   y = cos(t^2)  % chirp signal -- notice no . before ^ since t is no vector
   z = diff(y)  % derivative
   figure(1)
   subplot(211)
   ezplot(y, [0, 2*pi]);grid  % plotting for symbolic y between 0 and 2*pi
   hold on
   subplot(212)
   ezplot(z, [0, 2*pi]);grid
   hold on
%numeric
   Ts = 0.1;  % sampling period
   t1 = 0:Ts:2*pi;  % sampled time
   y1 = cos(t1.^2);  % sampled signal --notice difference with y above
   z1 = diff(y1)./diff(t1);  % difference  -- approximation to derivative
   figure(1)
   subplot(211)
   stem(t1, y1, 'r');axis([0 2*pi 1.1*min(y1) 1.1*max(y1)])
   subplot(212)
   stem(t1(1:length(y1) - 1), z1, 'r');axis([0 2*pi 1.1*min(z1) 1.1*max(z1)])
   legend('Derivative (black)','Difference (blue)')
   hold off
```

The symbolic function syms defines the symbolic variables (use help syms to learn more). The signal $y(t)$ is written differently than $y_1(t)$ in the numeric computation. Since t_1 is a vector, squaring it requires a dot before the symbol. That is not the case for t, which is not a vector but a variable. The results of using diff to compute the derivative of $y(t)$ is given in the same form as you would have obtained doing the derivative by hand—that is,

```
y = cos(t^2)
z = −2*t*sin(t^2)
```

The symbolic toolbox provides its own graphic routines (use help to learn about the different ez-routines). For plotting $y(t)$ and $z(t)$, we use the function ezplot, which plots the above two functions for $t \in [0, 2\pi]$ and titles the plots with these functions.

The numeric computations differ from the symbolic in that vectors are being processed, and we are obtaining an approximation to the derivative $z(t)$. We sample the signal with $T_s = 0.1$ and use again

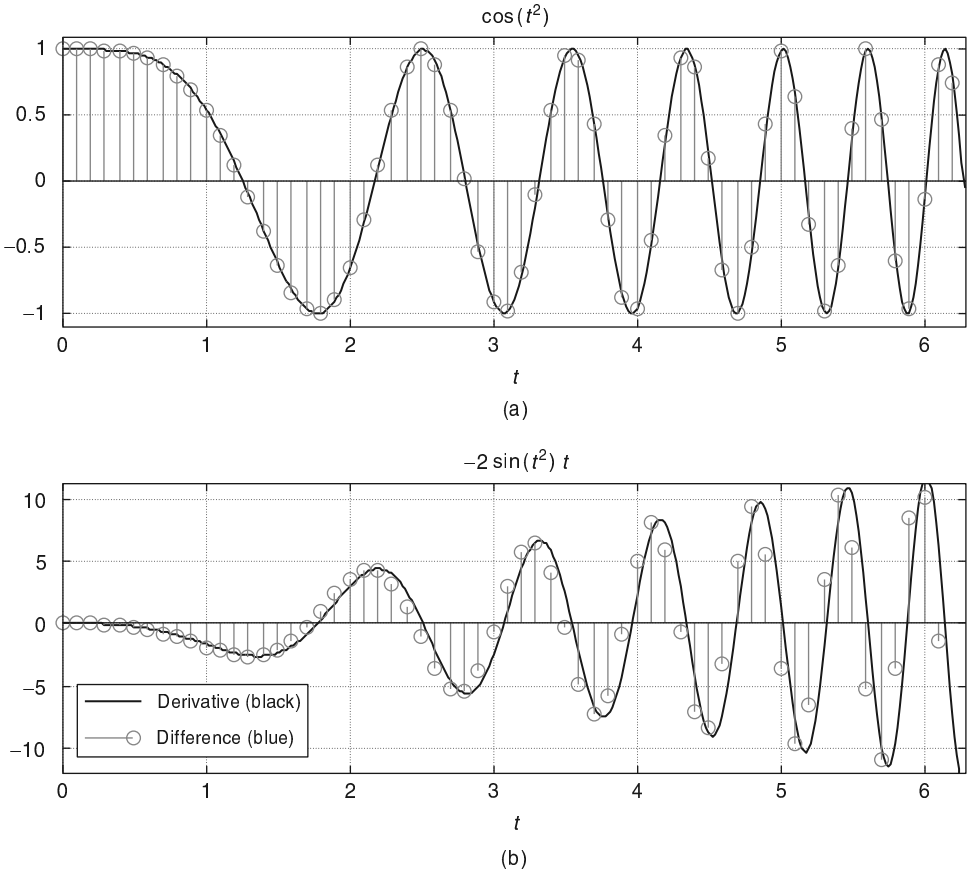


FIGURE 0.21
Symbolic and numeric computation of the derivative of the chirp $y(t) = \cos(t^2)$. (a) $y(t)$ and the sampled signal $y(nT_s)$, $T_s = 0.1$ sec. (b) Displays the exact derivative (continuous line) and the approximation of the derivative at samples nT_s. Better approximation to the derivative can be obtained by using a smaller value of T_s.

the function diff to approximate the derivative (the denominator *diff(t1)* is the same as T_s). Plotting the exact derivative (continuous line) with the approximated one (samples) using stem clarifies that the numeric computation is an approximation at nT_s values of time. See Figure 0.21.

The Sinc Function and Integration

The sinc function is very significant in the theory of signals and systems. It is defined as

$$y(t) = \frac{\sin \pi t}{\pi t} \qquad -\infty < t < \infty$$

It is symmetric with respect to the origin, and defined from $-\infty$ to ∞. The value of $y(0)$ can be found using L'Hôpital's rule. We will see later (Parseval's result in Chapter 5) that the integral of $y^2(t)$ is

equal to 1. In the following script we are combining numeric and symbolic computations to show this. First, after defining the variables, we use the symbolic function int to compute the integral of the squared sinc function, with respect to t, from 0 to integer values $1 \leq k \leq 10$. We then use the function subs to convert the symbolic results into a numerical array zz. The numeric part of the script defines a vector y to have the values of the sinc function for 100 time values equally spaced between $[-4, 4]$, obtained using the function linspace. We then use plot and stem to plot the sinc and the values of the integrals, which as seen in Figure 0.22 reach a value close to unity in less than 10 steps. Please use help to learn more about each of these functions.

```
clf; clear all
% symbolic
syms t z
for k = 1:10,
```

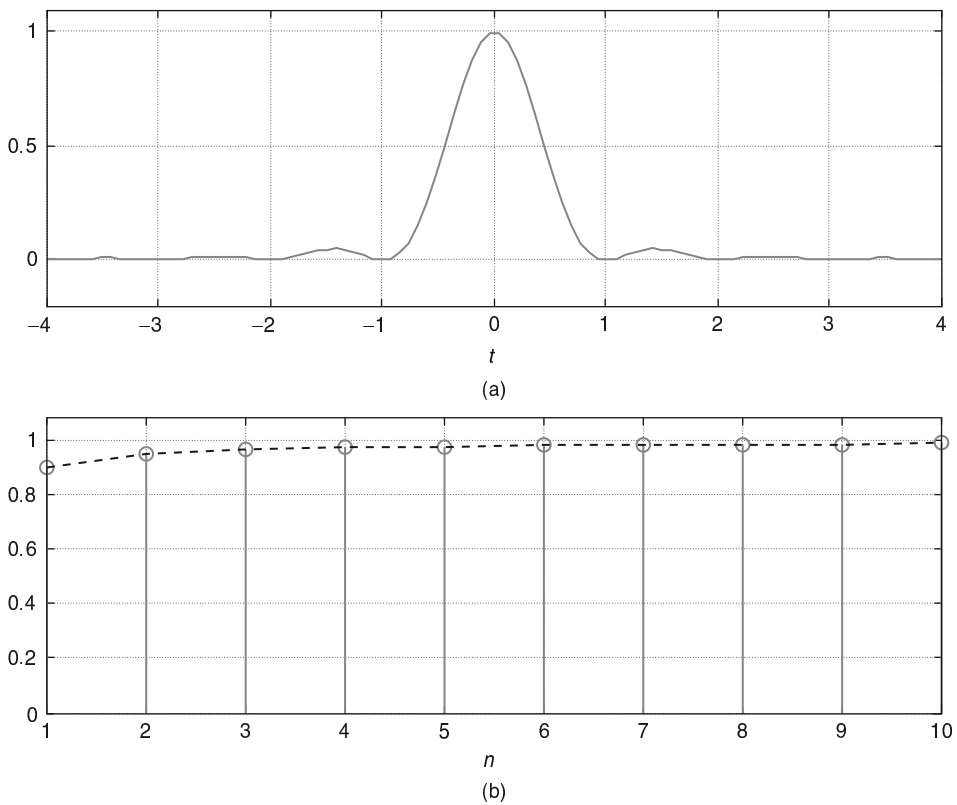


FIGURE 0.22

(a) Computation of the integral of the squared sinc function (b) Illustrates that the area under the curve of this function, or its integral, is unity. Using the symmetry of the function only the integral for $t \geq 0$ needs to be computed.

```
    z = int(sinc(t)^2, t, 0, k);    % integral of sinc^2 from 0 to k
    zz(k) = subs(2*z);          % substitution to numeric value zz
end
% numeric
t1 = linspace(-4, 4);       % 100 equally spaced points in [-4,4]
y = sinc(t1).^2;    % numeric definition of the squared sinc function
n = 1:10;
figure(1)
subplot(211)
plot(t1, y);grid;axis([-4 4 -0.2 1.1*max(y)]);title('y(t)=sinc^2(t)');
xlabel('t')
subplot(212)
stem(n(1:10), zz(1:10)); hold on
plot(n(1:10), zz(1:10), 'r');grid;title('∫ y(τ) dτ'); hold off
axis([1 10 0 1.1*max(zz)]); xlabel('n')
```

Figure 0.22 shows the squared sinc function and the values of the integral

$$2\int_0^k sinc^2(t)dt = 2\int_0^k \left[\frac{\sin(\pi t)}{\pi t}\right]^2 dt \qquad k = 1, \ldots, 10$$

which quickly reaches the final value of unity. In computing the integral from $(-\infty, \infty)$ we are using the symmetry of the function and thus the multiplication by 2.

Chebyshev Polynomials and Lissajous Figures

The Chebyshev polynomials are used in the design of filters. They can be obtained by plotting two cosine functions as they change with time t, one of fix frequency and the other with increasing frequency:

$$x(t) = \cos(2\pi t)$$
$$y(t) = \cos(2\pi kt) \qquad k = 1, \ldots, N$$

The $x(t)$ gives the x axis coordinate and $y(t)$ the y axis coordinate at each value of t. If we solve for t in the top equation, we get

$$t = \frac{1}{2\pi}\cos^{-1}(x(t))$$

which then replaced in the bottom equation gives

$$y(t) = \cos\left[k\cos^{-1}(x(t))\right] \qquad k = 1, \ldots, N$$

as an expression for the Chebyshev polynomials (we will see in Chapter 6 that these equations can be expressed as regular polynomials). Figure 0.23 shows the Chebyshev polynomials for $N = 4$. The following script is used to compute and plot these polynomials.

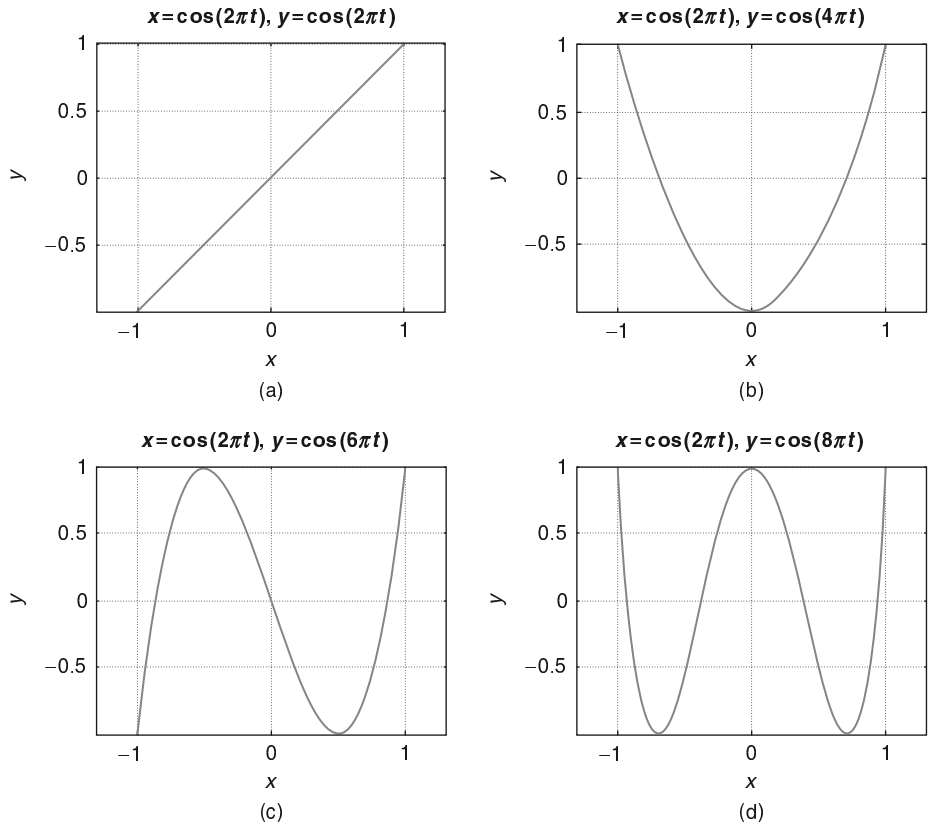


FIGURE 0.23
The Chebyshev polynomials for $n = 1, 2, 3, 4$. First (a) to fourth (d) polynomials. Notice that these polynomials are defined between $[-1, 1]$ in the x axis.

```
clear all;clf
syms x y t
x = cos(2*pi*t); theta=0;
figure(1)
for k = 1:4,
  y = cos(2*pi*k*t + theta);
  if k == 1, subplot(221)
    elseif k == 2, subplot(222)
    elseif k == 3, subplot(223)
    else subplot(224)
  end
  ezplot(x, y);grid;hold on
end
hold off
```

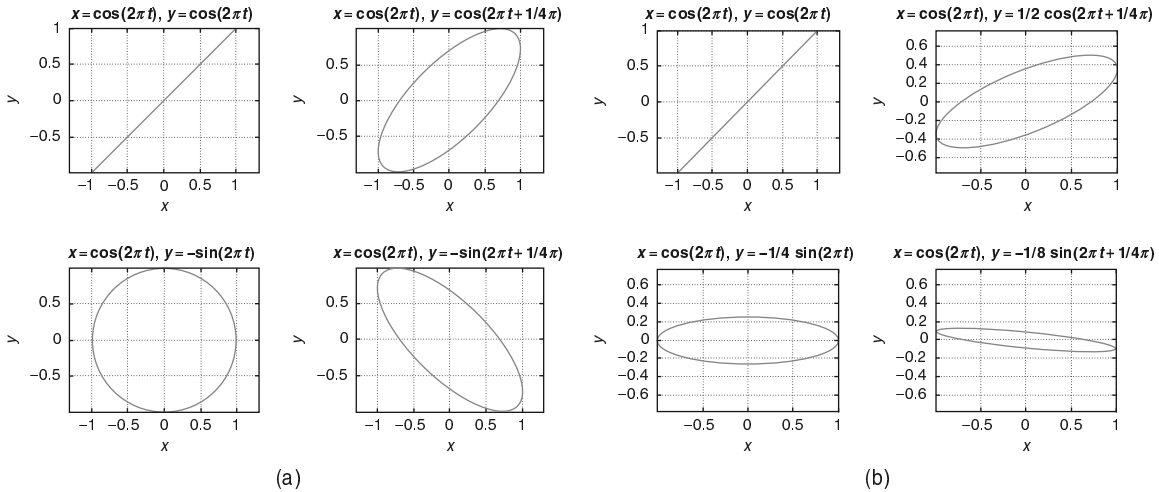


FIGURE 0.24

Lissajous figures: (a) (four left plots) case 1 input and output of same amplitude ($A = 1$) but phase differences of $0, \pi/4, \pi/2$, and $3\pi/4$; (b) (four right plots) case 2 input has unit amplitude but output has decreasing amplitudes and same phase differences as in case 1.

The Lissajous figures we consider next are a very useful extension of the above plotting of sinusoids in the x and y axes. These figures are used to determine the difference between a sinusoidal input and its corresponding sinusoidal steady state. In the case of linear systems, which we will formally define in Chapter 2, for a sinusoidal input the outputs of the system are also sinusoids of the same frequency, but they differ with the input in the amplitude and phase.

The differences in amplitude and phase can be measured using an oscilloscope for which we put the input in the horizontal sweep and the output in the vertical sweep, giving figures from which we can find the differences in amplitude and phase. Two situations are simulated in the following script, one where there is no change in amplitude but the phase changes from zero to $3\pi/4$, while in the other case the amplitude decreases as indicated and the phase changes in the same way as before. The plots, or Lissajous figures, indicate such changes. The difference between the maximum and the minimum of each of the figures in the x axis gives the amplitude of the input, while the difference between the maximum and the minimum in the y axis gives the amplitude of the output. The orientation of the ellipse provides the difference in phase with respect to that of the input. The following script is used to obtain the Lissajous figures in these cases. Figure 0.24 displays the results.

```
clear all;clf
syms x y t
x = cos(2*pi*t); % input of unit amplitude and frequency 2*pi
A = 1;figure(1)   % amplitude of output in case 1
for i = 1:2,
for k = 0:3,
```

```
    theta = k*pi/4;   % phase of output
    y = A^k*cos(2*pi*t + theta);
  if k == 0,subplot(221)
    elseif k == 1,subplot(222)
    elseif k == 2,subplot(223)
    else subplot(224)
  end
  ezplot(x, y);grid;hold on
end
A = 0.5; figure(2) % amplitude of output in case 2
end
```

Ramp, Unit-Step, and Impulse Responses

To close this introduction to symbolic computations we illustrate the response of a linear system represented by a differential equation,

$$\frac{d^2y(t)}{dt^2} + 5\frac{dy(t)}{dt} + 6y(t) = x(t)$$

where $y(t)$ is the output and $x(t)$ the input. The input is a constant $x(t) = 1$ for $t \geq 0$ and zero otherwise (MATLAB calls this function *heaviside,* but we will call it the *unit-step signal*). We then let the input be the derivative of $x(t)$, which is a signal that we will call *impulse,* and finally we let the input be the integral of $x(t)$, which is what we will call the *ramp* signal. The following script is used to find the responses, which are displayed in Figure 0.25.

```
clear all; clf
syms y t x z
% input a unit-step  (heaviside) response
y = dsolve('D2y + 5*Dy + 6*y = heaviside(t)','y(0) = 0','Dy(0) = 0','t');
x = diff(y); % impulse response
z = int(y); % ramp response
figure(1)
subplot(311)
ezplot(y, [0,5]);title('Unit-step response')
subplot(312)
ezplot(x, [0,5]);title('Impulse response')
subplot(313)
ezplot(z, [0,5]);title('Ramp response')
```

This example illustrates the intuitive appeal of linear systems. When the input is a constant value (or a unit-step signal or a heaviside signal) the output tries to follow the input after some initial inertia and it ends up being constant. The impulse signal (obtained as the derivative of the unit-step signal) is a signal of very short duration equivalent to shocking the system with a signal that disappears very fast, different from the unit-step signal that is like a dc source. Again the output tries to follow the input, eventually disappearing as t increases (no energy from the input!), and the ramp that is

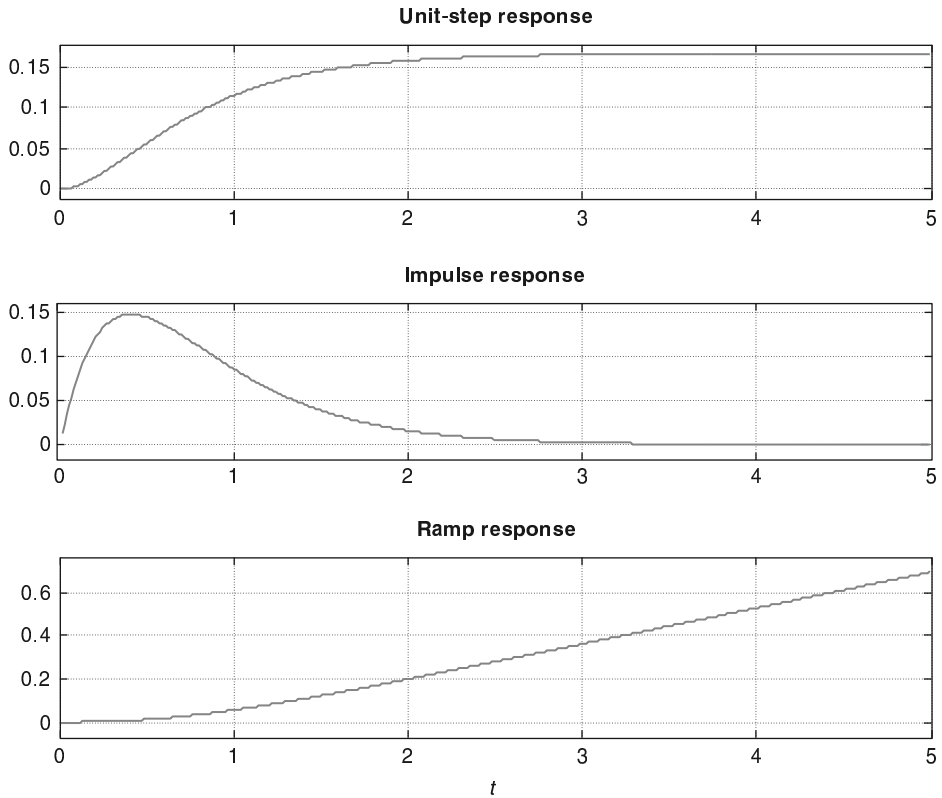


FIGURE 0.25
Response of a second order system represented by a differential equation for input of the unit-step signal, its derivative, or the impulse signal and the ramp signal that is the integral of the unit-step input.

the integral of the unit-step signal grows with time, providing more and more energy to the system as time increases, thus the response we obtained. The function dsolve solves differential equations explicitly given (D stands for the derivative operator, so D is the first derivative and D2 is the second derivative). A second-order system requires two initial conditions, the output and its derivative at $t = 0$.

We hope this introduction to MATLAB has provided you with the necessary background to understand the basic way MATLAB operates, and shown you how to continue increasing your knowledge of it. Your best source of information is the *help* command. Explore the different modules that MATLAB has and you will become quickly convinced that these modules provide a great number of computational tools for many areas of engineering and mathematics. Try it—you will like it! Tables 0.1 and 0.2 provide a listing of the numeric and symbolic variables and operations.

Table 0.1 Basic Numeric Matlab

Special variables	ans	Default name for result
	pi	π value
	inf, NaN	infinity, not-a-number error (e.g., $0/0$)
	i, j	$i = j = \sqrt{-1}$
	Function(s)	**Operation**
Mathematical	abs, angle	magnitude, angle of complex number
	acos, asine, atan	inverse cosine, sine, tangent
	acosh, asinh, atanh	inverse cosh, sinh, tanh
	cos, sin, tan	cosine, sine, tangent
	cosh, sinh, tanh	hyperbolic cosine, sine, tangent
	conj, imag, real	complex conjugate, imaginary, real parts
	exp, log, log10	exponential, natural and base 10 logarithms
Special operations	ceil, floor	round up, round down to integer
	fix, round	round toward zero, to nearest integer
	.*, ./	entry-by-entry multiplication, division
	.^	entry-by-entry power
	x', A'	transpose of vector x, matrix A
Array operations	x=*first*:*increment*:*last*	row vector x from *first* to *last* by *increment*
	x=linspace(first,last,n)	row vector x with n elements from *first* to *last*
	A=[x1;x2]	matrix A with rows $x1$, $x2$
	ones(N,M), zeros(N,M)	$N \times M$ ones and zeros arrays
	A(i,j)	(i, j) entry of matrix A
	A(i,:), A(:,j)	i row (j-column) and all columns (rows) of matrix A
	whos	display variables in workspace
	size(A)	(number rows, number of colums) of matrix A
	length(x)	number rows (colums) of vector x
Control flow	for, if, elseif	for loop, if, else-if loop
	while	while loop
	pause, pause(n)	pause and pause n seconds
Plotting	plot, stem	continuous, discrete plots
	figure	figure for plotting
	subplot	subplots
	hold on, hold off	hold plot on or off
	axis, grid	axis, grid of plots
	xlabel, ylabel, title, legend	labeling of axes, plots, and subplots
Saving and loading	save, load	saving and loading data
Information and managing	help	help
	clear, clf	clear variables from memory, clear figures
Operating system	cd, pwd	change directory, current working directory

Table 0.2 Basic Symbolic Matlab Functions

	Function	Operation
Calculus	diff	differentiate
	int	integrate
	limit	limit
	taylor	Taylor series
	symsum	summation
Simplification	simplify	simplify
	expand	expand
	factor	factor
	simple	find shortest form
	subs	symbolic substitution
Solving equations	solve	solve algebraic equations
	dsolve	solve differential equations
Transforms	fourier	Fourier transform
	ifourier	inverse Fourier transform
	laplace	Laplace transform
	ilaplace	inverse Laplace transform
	ztrans	Z-transform
	iztrans	inverse Z-transform
Symbolic operations	sym	create symbolic objects
	syms	create symbolic objects
	pretty	make pretty expression
Special functions	dirac	Dirac or delta function
	heaviside	unit-step function
Plotting	ezplot	function plotter
	ezpolar	polar coordinate plotter
	ezcontour	contour plotter
	ezsurf	surface plotter
	ezmesh	mesh (surface) plotter

PROBLEMS

For the problems requiring implementation in MATLAB, write scripts or functions to solve them numerically or symbolically. Label the axes of the plots, give a title, and use legend to identify different signals in a plot. To save space use subplot to put several plots into one. To do the problem numerically, sample analog signals with a small T_s.

0.1. Bits or bytes

Just to get an idea of the number of bits or bytes generated and processed by a digital system consider the following applications:

(a) A compact disc is capable of storing 75 minutes of "CD-quality" stereo (left and right channels are recorded) music. Calculate the number of bytes and the number of bits that are stored in the CD.
Hint: Find out what "CD quality" means in the binary representation of each sample, and what is the sampling rate your CD player uses.

(b) Find out what the vocoder in your cell phone is used for. Assume then that in attaining "telephone quality" you use a sampling rate of 10,000 samples/sec to achieve that type of voice quality. Each sample is represented by 8 bits. With this information, calculate the number of bits that your cell phone has to process every second that you talk. Why would you then need a vocoder?

(c) Find out whether text messaging is cheaper or more expensive than voice. Explain how text messaging works.

(d) Find out how an audio CD and an audio DVD compare. Find out why it is said that a vinyl long play record reproduces sounds much better. Are we going backwards with digital technology in music recording? Explain.

(e) To understand why video streaming in the Internet is many times of low quality, consider the amount of data that need to be processed by a video compressor every second. Assume the size of a video frame, in picture elements or pixels, is 352×240, and that an acceptable quality for the image is obtained by allocating 8 bits/pixel, and to avoid jerking effects we use 60 frames/sec.

- How many pixels would have to be processed every second?
- How many bits would be available for transmission every second?
- The above are raw data. Compression changes the whole picture (literally); find out what some of the compression methods are.

0.2. Sampling—MATLAB

Consider an analog signal $x(t) = 4\cos(2\pi t)$ defined for $-\infty < t < \infty$. For the following values of the sampling period T_s, generate a discrete-time signal $x[n] = x(nT_s) = x(t)|_{t=nT_s}$.

- $T_s = 0.1$ sec
- $T_s = 0.5$ sec
- $T_s = 1$ sec

Determine for which values of T_s the discrete-time signal has lost the information in the analog signal. Use MATLAB to plot the analog signal (use the plot function) and the resulting discrete-time signals (use the stem function). Superimpose the analog and the discrete-time signals for $0 \le t \le 3$; use subplot to plot the four figures as one figure. For plotting the analog signal use $T_s = 10^{-4}$. You also need to figure out how to label the different axes and have the same scales and units. In Chapter 7 on sampling we will show how to reconstruct sampled signals.

0.3. Derivative and finite difference—MATLAB

Let $y(t) = dx(t)/dt$, where $x(t)$ is the signal in Problem 0.2. Find $y(t)$ analytically and determine a value of T_s for which $\Delta[x(nT_s)]/T_s = y(nT_s)$ (consider $T_s = 0.01$ and $T_s = 0.1$). Use the MATLAB function diff or create your own to compute the finite difference. Plot the finite difference in the range [0,1] and compare it with the actual derivative $y(t)$ in that range. Explain your results for the given values of T_s.

0.4. Backward difference—MATLAB

Another definition for the finite difference is the backward difference:

$$\Delta[x(nT_s)] = x(nT_s) - x((n-1)T_s)$$

($\Delta[x(nT_s)]/T_s$ approximates the derivative of $x(t)$.)

(a) Indicate how this new definition connects with the finite difference defined earlier in this chapter.

(b) Solve Problem 0.3 with MATLAB using this new finite difference and compare your results with the ones obtained there.

(c) For the value of $T_s = 0.1$, use the average of the two finite differences to approximate the derivative of the analog signal $x(t)$. Compare this result with the previous ones. Provide an expression for calculating this new finite difference directly.

0.5. Differential and difference equations—MATLAB

Find the differential equation relating a current source $i_s(t) = \cos(\Omega_0 t)$ with the current $i_L(t)$ in an inductor, with inductance $L = 1\,\text{H}$, connected in parallel with a resistor of $R = 1\Omega$ (see Figure 0.26). Assume a zero initial current in the inductor.

(a) Obtain a discrete equation from the differential equation using the trapezoidal approximation of an integral.

(b) Create a MATLAB script to solve the difference equation for $T_s = 0.01$ and three frequencies for $i_s(t)$, $\Omega_0 = 0.005\pi, 0.05\pi$, and 0.5π. Plot the input current source $i_s(t)$ and the approximate solution $i_L(nT_s)$ in the same figure. Use the MATLAB function plot. Use the MATLAB function filter to solve the difference equation (use help to learn about filter).

(c) Solve the differential equation using symbolic MATLAB when the input frequency is $\Omega_0 = 0.5\pi$.

(d) Use phasors to find the amplitude of $i_L(t)$ when the input is $i_s(t)$ with the given three frequencies.

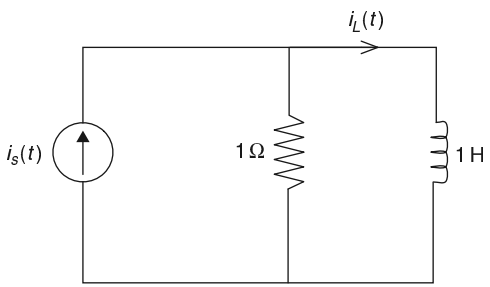


FIGURE 0.26
Problem 0.5. RL circuit: input $i_s(t)$ and output $i_L(t)$.

0.6. Sums and Gauss—MATLAB

Three rules in the computation of sums are

- Distributive law:

$$\sum_k ca_k = c\sum_k a_k$$

- Associative law:

$$\sum_k (a_k + b_k) = \sum_k a_k + \sum_k b_k$$

- Commutative law:

$$\sum_k a_k = \sum_{p(k)} a_{p(k)}$$

for any permutation $p(k)$ of the set of integers k in the summation.

(a) Explain why the above rules make sense when computing sums. To do that consider

$$\sum_k a_k = \sum_{k=0}^{2} a_k$$

and similarly for $\sum_k b_k$. Let c be a constant, and choose any permutation of the values [0,1,2] for instance [2,1,0] or [1,0,2].

(b) The trick that Gauss played when he was a preschooler can be explained by using the above rules. Suppose you want to find the sum of the integers from 0 to 10000 (Gauss did it for integers between 0 and 100 but he was then just a little boy, and we can do better!). That is, we want to find S where

$$S = \sum_{k=0}^{10000} k = 0 + 1 + 2 + \cdots + 10000$$

To do so, consider

$$2S = \sum_{k=0}^{10000} k + \sum_{k=10000}^{0} k$$

and apply the above rules to find S.

(c) Find the sum of an arithmetic progression

$$S = \sum_{k=0}^{N} (\alpha + \beta k)$$

for constants α and β, using the given three rules.

(d) Find out if MATLAB can do these sums symbolically (i.e., without having numerical values).

0.7. Integrals and sums—MATLAB

Suppose you wish to find the area under a signal using sums. You will need the following result found above:

$$\sum_{n=0}^{N} n = \frac{N(N+1)}{2}$$

(a) Consider first $x(t) = t, 0 \le t \le 1$, and zero otherwise. The area under this signal is 0.5. The integral can be approximated from above and below as

$$\sum_{n=1}^{N-1} (nT_s)T_s < \int_0^1 t dt < \sum_{n=1}^{N} (nT_s)T_s$$

where $NT_s = 1$ (i.e., we segment the interval [0,1] into N intervals of width T_s). Graphically show that the above equation makes sense by showing the right and left bounds as approximations for the area under $x(t)$.

(b) Let $T_s = 0.001$. Use the symbolic function symsum to compute the left and right bounds for the above integral. Find the average of these results and compare it with the actual value of the integral.

(c) Verify the symbolic results by finding the sums on the left and the right of the above inequality using the summation given at the beginning of the problem. You need to change the dummy variables.

(d) Write a similar MATLAB script to compute the area under the signal $\gamma(t) = t^2$ from $0 \leq t \leq 1$. Let $T_s = 0.001$. Compare the average of the lower and upper bounds to the value of the integral.

0.8. Integrals and sums—MATLAB

Although sums behave like integrals, because of the discrete nature of sums one needs to be careful with the upper and lower limits more than in the integral case. To illustrate this, consider the separation of an integral into two integrals and compare them with the separation of a sum into two sums. For the integral we have that

$$\int_0^1 tdt = \int_0^{0.5} tdt + \int_{0.5}^1 tdt$$

Show that this is true by computing the three integrals. Then consider the sum

$$S = \sum_{n=0}^{100} n$$

Find this sum and determine which of the following is equal to this sum:

$$S_1 = \sum_{n=0}^{50} n + \sum_{n=50}^{100} n$$

$$S_2 = \sum_{n=0}^{50} n + \sum_{n=51}^{100} n$$

Use symbolic MATLAB function symsum to verify your answers.

0.9. Sum of geometric series

The geometric series

$$S = \sum_{n=0}^{N-1} \alpha^n$$

will be used quite frequently in the next chapters, so let us look at some of its properties:

(a) Suppose $\alpha = 1$; what is S equal to?

(b) Suppose $\alpha \neq 1$; show that

$$S = \frac{1 - \alpha^N}{1 - \alpha}$$

This can be done by showing that $(1 - \alpha)S = (1 - \alpha^N)$. Why do you need the constraint that $\alpha \neq 1$? Would this sum exist if $\alpha > 1$? Explain.

(c) Give an expression of the above sum for all possible values of α.
(d) Suppose now that $N = \infty$; under what conditions will S exist? If it does, what would S be equal to? Explain.
(e) Suppose the derivative of S with respect to α is

$$S_1 = \frac{dS}{d\alpha} = \sum_{n=0}^{\infty} n\alpha^n$$

Obtain an expression to find S_1.

0.10. Exponentials—MATLAB

The exponential $x(t) = e^{at}$ for $t \geq 0$ and zero otherwise is a very common analog signal. Likewise, $\gamma[n] = \alpha^n$ for integers $n \geq 0$ and zero otherwise is a very common discrete-time signal. Let us see how they are related. Do the following using MATLAB:

(a) Let $a = -0.5$; plot $x(t)$.
(b) Let $a = -1$; plot the corresponding signal $x(t)$. Does this signal go to zero faster than the exponential for $a = -0.5$?
(c) Suppose we sample the signal $x(t)$ using $T_s = 1$; what would be $x(nT_s)$ and how can it be related to $\gamma(n)$ (i.e., what is the value of α that would make the two equal)?
(d) Suppose that a current $x(t) = e^{-0.5t}$ for $t \geq 0$ and zero otherwise is applied to a discharged capacitor of capacitance $C = 1$ F at $t = 0$. What would be the voltage in the capacitor at $t = 1$ sec?
(e) How would you obtain an approximate result to the above problem using a computer? Explain.

0.11. Algebra of complex numbers

Consider complex numbers $z = 1 + j1$, $w = -1 + j1$, $v = -1 - j1$, and $u = 1 - j1$.

(a) In the complex plane, indicate the point (x, y) that corresponds to z and then show a vector $\vec{z}$ that joins the point (x, y) to the origin. What is the magnitude and the angle corresponding to z or $\vec{z}$?
(b) Do the same for the complex numbers w, v, and u. Plot the four complex numbers and find their sum $z + w + v + u$ analytically and graphically.
(c) Find the ratios z/w, w/v, and u/z. Determine the real and imaginary parts of each, as well as their magnitudes and phases. Using the ratios find u/w.
(d) The phase of a complex number is only significant when the magnitude of the complex number is significant. Consider z and $\gamma = 10^{-16}z$; compare their magnitudes and phases. What would you say about the phase of γ?

0.12. Algebra of complex numbers

Consider a function of $z = 1 + j1$,

$$w = e^z$$

(a) Find $\log(w)$.
(b) Find the real and the imaginary parts of w.
(c) What is $w + w^*$, where w^* is the complex conjugate of w?
(d) Determine $|w|$, $\angle w$.
(e) What is $|\log(w)|^2$?
(f) Express $\cos(1)$ in terms of w using Euler's equation.

0.13. Euler's identity and trigonometric identities

Use Euler's identity to obtain an expression for $e^{j(\alpha+\beta)} = e^{j\alpha}e^{j\beta}$; obtain its real and imaginary components and show the following identities:

- $\cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$
- $\sin(\alpha + \beta) = \sin(\alpha)\cos(\beta) + \sin(\beta)\cos(\beta)$

Hint: Find real and imaginary parts of $e^{j\alpha}e^{j\beta}$ and of $e^{j(\alpha+\beta)}$.

0.14. Euler's identity and trigonometric identities

Use Euler's identity to find an expression for $\cos(\alpha)\cos(\beta)$, and from the relation between cosines and sines obtain an expression for $\sin(\alpha)\sin(\beta)$.

0.15. Algebra of complex numbers

(a) The complex conjugate of $z = x + jy$ is $z^* = x - jy$. Using these rectangular representations, show that

$$zz^* = x^2 + y^2$$

$$\frac{1}{z} = \frac{z^*}{zz^*}$$

(b) Show that it is easier to find the above results by using the polar representation $z = |z|e^{j\theta}$ of z where

$$|z| = \sqrt{x^2 + y^2}$$

is the magnitude of z and

$$\theta = \tan^{-1}\left(\frac{y}{x}\right)$$

is the angle or phase of z. Thus, whenever we are multiplying or dividing complex numbers the polar form is more appropriate.

(c) Whenever we are adding or subtracting complex numbers the rectangular representation is more appropriate. Show that for two complex numbers $z = x + jy$ and $w = v + jq$; then,

$$(z + w)^* = z^* + w^*$$

On the other hand, when showing that $(zw)^* = z^*w^*$ the polar form is more appropriate.

(d) If the above conclusions still do not convince you, consider then the case of multiplying two complex numbers:

$$z = r\cos(\theta) + jr\sin(\theta)$$

$$w = \rho\cos(\phi) + j\rho\sin(\phi)$$

Find the polar forms of z and w and then find zw by using the rectangular and then the polar forms and decide which is easier. As a bonus you should get the trigonometric identities for $\cos(\theta + \phi)$ and $\sin(\theta + \phi)$. What are they?

0.16. Vectors and complex numbers

Using the vectorial representation of complex numbers it is possible to get some interesting inequalities:

(a) Is it true that for a complex number $z = x + jy$:

$$|x| \leq |z|?$$

Show it geometrically by representing z as a vector.

(b) The so-called *triangle inequality* says that for any complex (or real) numbers z and v we have that

$$|z + v| \leq |z| + |v|$$

Show a geometric example that verifies this.

0.17. Complex functions of time—MATLAB

Consider the complex function $x(t) = (1 + jt)^2$ for $-\infty < t < \infty$.

(a) Find the real and the imaginary parts of $x(t)$ and carefully plot them with MATLAB. Try to make MATLAB plot $x(t)$ directly. What do you get? Does MATLAB warn you? Does it make sense?

(b) Compute the derivative $y(t) = dx(t)/dt$ and plot its real and imaginary parts. How do these relate to the real and the imaginary parts of $x(t)$?

(c) Compute the integral

$$\int_0^1 x(t)dt$$

(d) Would the following statement be true (remember * indicates complex conjugate)?

$$\left(\int_0^1 x(t)dt\right)^* = \int_0^1 x^*(t)dt$$

0.18. Euler's equation and orthogonality of sinusoids

Euler's equation,

$$e^{j\theta} = \cos(\theta) + j\sin(\theta)$$

is very useful not only in obtaining the rectangular and polar forms of complex numbers, but in many other respects as we will explore in this problem.

(a) Carefully plot $x[n] = e^{j\pi n}$ for $-\infty < n < \infty$. Is this a real or a complex signal?

(b) Suppose you want to find the trigonometric identity corresponding to

$$\sin(\alpha)\sin(\beta)$$

Use Euler's equation to express the sines in terms of exponentials, multiply the resulting exponentials, and use Euler's equation to regroup the expression in terms of sinusoids.

(c) As we will see later on, two periodic signals $x(t)$ and $y(t)$ of period T_0 are said to be orthogonal if the integral over a period T_0 is

$$\int_{T_0} x(t)y(t)dt = 0$$

For instance, consider $x(t) = \cos(\pi t)$ and $y(t) = \sin(\pi t)$. Check first that these functions repeat every $T_0 = 2$ (i.e., show that $x(t+2) = x(t)$ and that $y(t+2) = y(t)$). Thus, $T_0 = 2$ can be seen as their period. Then use the representation of a cosine in terms of complex exponentials,

$$\cos(\theta t) = \frac{e^{j\theta} + e^{-j\theta}}{2}$$

to express the integrand in terms of exponentials and calculate the integral.

0.19. Euler's equation and trigonometric expressions

Obtain using Euler's equation an expression for $\sin(\theta)$ in terms of exponentials and then

(a) Use it to obtain the trigonometric identity for $\sin^2(\theta)$.

(b) Compute the integral

$$\int_0^1 \sin^2(2\pi t)dt$$

0.20. De Moivre's theorem for roots

Consider the calculation of roots of an equation,

$$z^N = \alpha$$

where $N \geq 1$ is an integer and $\alpha = |\alpha|e^{j\phi}$ a nonzero complex number.

(a) First verify that there are exactly N roots of this equation and that they are given by

$$z_k = re^{j\theta_k}$$

where $r = |\alpha|^{1/N}$ and $\theta_k = (\phi + 2\pi k)/N$ for $k = 0, 1, \ldots, N-1$.

(b) Use the above result to find the roots of the following equations:

$$z^2 = 1$$
$$z^2 = -1$$
$$z^3 = 1$$
$$z^3 = -1$$

and plot them in a polar plane (i.e., indicating their magnitude and phase).

(c) Explain how the roots are distributed around a circle of radius r in the complex polar plane.

0.21. Natural log of complex numbers

Suppose you want to find the log of a complex number $z = |z|e^{j\theta}$. Its logarithm can be found to be

$$\log(z) = \log(|z|e^{j\theta}) = \log(|z|) + \log(e^{j\theta}) = \log(|z|) + j\theta$$

If z is negative it can be written as $z = |z|e^{j\pi}$ and we can find $\log(z)$ by using the above derivation. The log of any complex number can be obtained this way also.

(a) Justify each one of the steps in the above equation.

(b) Find

$$\log(-2)$$
$$\log(1 + j1)$$
$$\log(2e^{j\pi/4})$$

0.22. Hyperbolic sinusoids—MATLAB

In filter design you will be asked to use hyperbolic functions. In this problem we relate these functions to sinusoids and obtain a definition of these functions so that we can actually plot them.

(a) Consider computing the cosine of an imaginary number—that is, use

$$\cos(x) = \frac{e^{jx} + e^{-jx}}{2}$$

Let $x = j\theta$ and find $\cos(x)$. The resulting function is called the hyperbolic cosine or

$$\cos(j\theta) = \cosh(\theta)$$

(b) Consider then the computation of the hyperbolic sine $\sinh(\theta)$; how would you do it? Carefully plot it as a function of θ.

(c) Show that the hyperbolic cosine is always positive and bigger than 1 for all values of θ.
(d) Show that $\sinh(\theta) = -\sinh(-\theta)$.
(e) Write a MATLAB script to compute and plot these functions between -10 and 10.

0.23. Phasors!

A phasor can be thought of as a vector, representing a complex number, rotating around the polar plane at a certain frequency expressed in radians/sec. The projection of such a vector onto the real axis gives a cosine. This problem will show the algebra of phasors, which would help you with some of the trigonometric identities that are hard to remember.

(a) When you plot a sine signal $y(t) = A\sin(\Omega_0 t)$, you notice that it is a cosine $x(t) = A\cos(\Omega_0 t)$ shifted in time—that is,

$$y(t) = A\sin(\Omega_0 t) = A\cos(\Omega_0(t - \Delta_t)) = x(t - \Delta_t)$$

How much is this shift Δ_t? Better yet, what is $\Delta_\theta = \Omega_0 \Delta_t$ or the shift in phase? One thus only need to consider cosine functions with different phase shifts instead of sines and cosines.

(b) You should have found the answer above is $\Delta_\theta = \pi/2$ (if not, go back and try it and see if it works). Thus, the phasor that generates $x(t) = A\cos(\Omega_0 t)$ is Ae^{j0} so that $x(t) = \mathcal{R}e[Ae^{j0}e^{j\Omega_0 t}]$. The phasor corresponding to the sine $y(t)$ should then be $Ae^{-j\pi/2}$. Obtain an expression for $y(t)$ similar to the one for $x(t)$ in terms of this phasor.

(c) According to the above results, give the phasors corresponding to $-x(t) = -A\cos(\Omega_0 t)$ and $-y(t) = -\sin(\Omega_0 t)$. Plot the phasors that generate $\cos$, $\sin$, $-\cos$, and $-\sin$ for a given frequency. Do you see now how these functions are connected? How many radians do you need to shift in a positive or negative direction to get a sine from a cosine, etc.

(d) Suppose then you have the sum of two sinusoids, for instance $z(t) = x(t) + y(t)$, adding the corresponding phasors for $x(t)$ and $y(t)$ at some time (e.g., $t = 0$), which is just a sum of two vectors, you should get a vector and the corresponding phasor. Get the phasor for $z(t)$ and the expression for it in terms of a cosine.