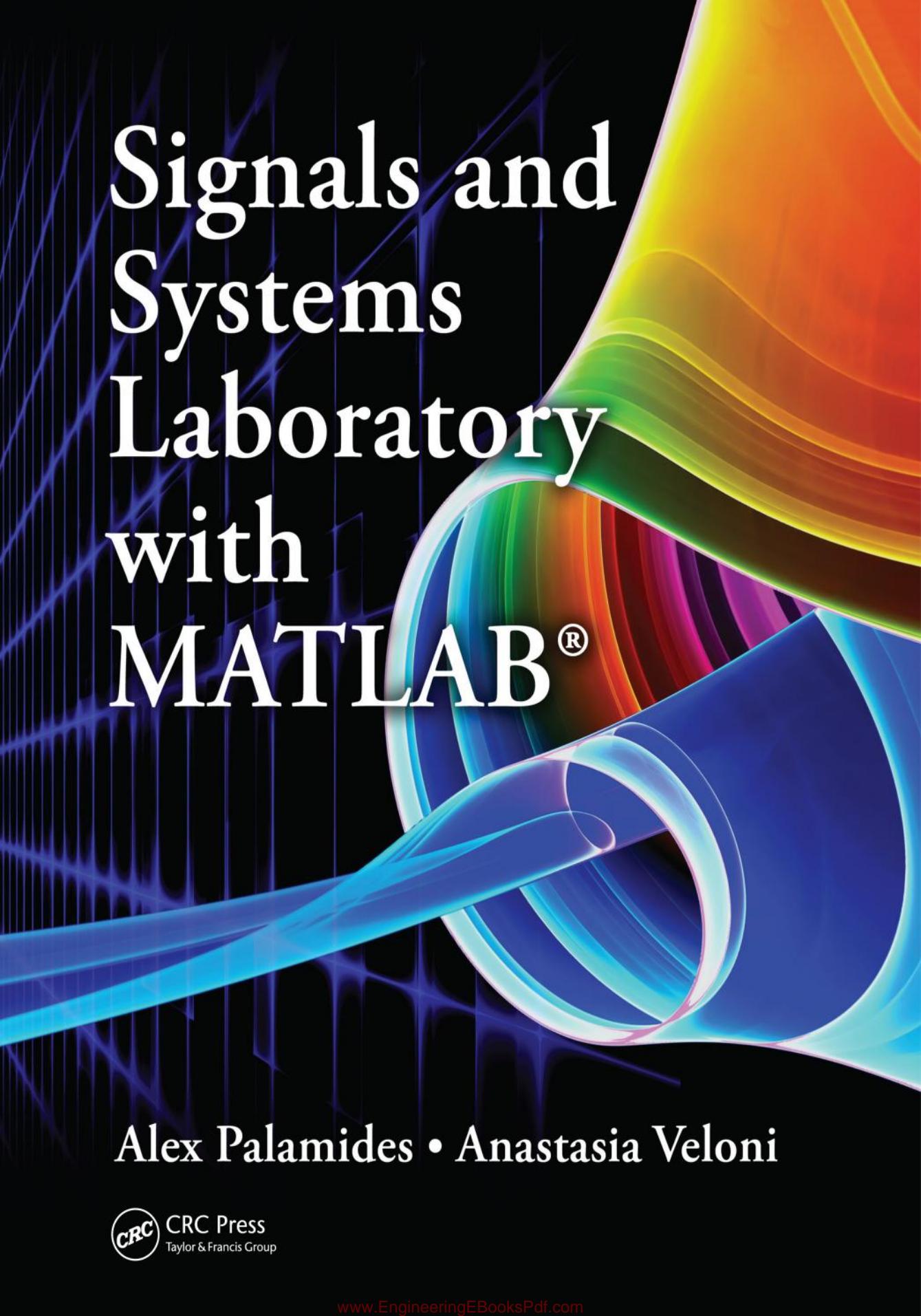


Signals and Systems Laboratory with MATLAB®



Alex Palamides • Anastasia Veloni



CRC Press
Taylor & Francis Group

**Signals and
Systems
Laboratory
with
MATLAB®**

This page intentionally left blank

Signals and Systems Laboratory with MATLAB®

Alex Palamides
Anastasia Veloni



CRC Press

Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

MATLAB® and Simulink® are trademarks of the MathWorks, Inc. and are used with permission. The MathWorks does not warrant the accuracy of the text or exercises in this book. This book's use or discussion of MATLAB® and Simulink® software or related products does not constitute endorsement or sponsorship by the MathWorks of a particular pedagogical approach or particular use of the MATLAB® and Simulink® software.

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

© 2011 by Taylor & Francis Group, LLC
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works
Version Date: 20110714

International Standard Book Number-13: 978-1-4398-9429-3 (eBook - PDF)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Visit the Taylor & Francis Web site at
<http://www.taylorandfrancis.com>

and the CRC Press Web site at
<http://www.crcpress.com>

To Zoe

A.P.

To John

A.V.

This page intentionally left blank

Contents

Preface.....	xiii
Authors	xvii

1. Introduction to MATLAB.....	1
1.1 What Is MATLAB?.....	1
1.2 Working Environment.....	1
1.3 Getting Started.....	2
1.3.1 Simple Arithmetic Operations.....	3
1.3.2 Comments	3
1.3.3 The Variable <i>ans</i>	3
1.3.4 Priority of Operations.....	3
1.3.5 Constants.....	4
1.3.6 Built-In Functions.....	5
1.3.7 Variables.....	6
1.3.8 Format.....	7
1.3.9 Help in MATLAB.....	7
1.4 Memory Management	8
1.4.1 Commands <i>save-load-exit-quit</i>	9
1.4.2 The Command <i>clear</i>	10
1.5 Commands <i>diary</i> and <i>clc</i>	10
1.6 Vectors	11
1.6.1 Row Vectors.....	11
1.6.2 Commands <i>length/size</i>	11
1.6.3 Addition/Subtraction	12
1.6.4 Multiplication, Division, and Power.....	13
1.6.5 Column Vectors.....	14
1.6.6 Dot Product of Two Vectors.....	14
1.6.7 Useful Commands	15
1.7 Matrices.....	16
1.7.1 Matrix Concatenation	17
1.7.2 Working with Matrices	17
1.7.3 Addition/Subtraction	19
1.7.4 Multiplication of Matrices.....	19
1.7.4.1 The Dot Product as a Special Case of Matrix Multiplication.....	21
1.7.5 Power of a Matrix	21
1.7.6 Inverse of a Matrix.....	22
1.7.7 Determinant of a Matrix	22
1.7.8 Division of Matrices.....	23
1.7.9 Transpose of a Matrix.....	24
1.7.10 Special Forms of Matrices.....	25
1.7.11 Useful Commands	26
1.8 Plotting with MATLAB.....	27
1.8.1 Plotting in Two Dimensions.....	27

1.8.2	The Fig File.....	29
1.8.3	The Command <code>linspace</code>	29
1.8.4	Plotting Several Functions in One Figure.....	30
1.8.5	Formatting a Figure	32
1.8.6	Plotting in Different Figures	34
1.8.7	Commands for Plotting.....	36
1.8.8	Plotting Discrete-Time Functions.....	38
1.8.9	Graph in Polar Coordinates.....	39
1.8.10	Piecewise Functions	39
1.8.11	Plotting in Three Dimensions.....	40
1.8.11.1	Plotting Curves in Three Dimensions	41
1.8.11.2	Plotting Surfaces in Three Dimensions	41
1.9	Complex Numbers.....	43
1.9.1	Useful Commands	43
1.9.2	Forms of Complex Numbers.....	44
1.9.3	Operations with Complex Numbers.....	45
1.9.4	Graph of Complex Numbers.....	46
1.10	M-Files	48
1.10.1	Scripts.....	48
1.10.2	Functions	51
1.11	Input/Output Commands	54
1.12	File Management.....	55
1.13	Logical/Relational Operators	57
1.14	Control Flow	58
1.15	Symbolic Variables.....	62
1.15.1	Differentiation of a Function	62
1.15.2	Integration of a Function	63
1.15.3	Summation of a Function	63
1.15.4	Rational Form	64
1.15.5	Solving Algebraic Equations	64
1.15.6	Solving Differential Equations	65
1.15.7	The Command <code>subs</code>	66
1.16	Polynomials.....	66
1.17	(Pseudo)Random Numbers	68
1.18	Solved Problems.....	69
1.19	Homework Problems.....	75
2.	Signals.....	77
2.1	Categorization by the Variable Type	77
2.1.1	Continuous-Time Signals.....	77
2.1.2	Discrete-Time Signals	78
2.1.3	Digital Signals.....	79
2.2	Basic Continuous-Time Signals.....	81
2.2.1	Sinusoidal Signals	81
2.2.2	Exponential Signals.....	82
2.2.3	Complex Exponential Signals.....	83
2.2.4	Unit Step Function	84
2.2.5	Unit Impulse or Dirac Delta Function	89

2.2.6	Ramp Function	93
2.2.7	Rectangular Pulse Function.....	96
2.3	Discrete-Time Signals	99
2.3.1	Unit Impulse Sequence.....	100
2.3.2	Unit Step Sequence	102
2.3.3	Real Exponential Sequence	104
2.3.4	Complex Exponential Sequence.....	105
2.3.5	Sinusoidal Sequence	109
2.4	Properties of Signals	111
2.4.1	Periodic Signals	111
2.4.1.1	Sum of Periodic Continuous-Time Signals	112
2.4.1.2	Construction of Periodic Signals	114
2.4.2	Causal Signals.....	118
2.4.3	Even and Odd Signals.....	119
2.4.4	Energy and Power Signals.....	121
2.4.5	Deterministic and Stochastic Signals.....	124
2.5	Transformations of the Time Variable for Continuous-Time Signals	126
2.5.1	Time Reversal or Reflection.....	126
2.5.2	Time Scaling.....	127
2.5.3	Time Shifting.....	129
2.6	Transformations of the Time Variable for Discrete-Time Signals.....	132
2.7	Solved Problems.....	135
2.8	Homework Problems.....	145
3.	Systems	147
3.1	Systems Classification.....	147
3.1.1	Classification according to the Number of Inputs and Outputs	147
3.1.2	Continuous-Time and Discrete-Time Signals	151
3.1.3	Deterministic and Stochastic Systems.....	151
3.2	Properties of Systems.....	151
3.2.1	Causal and Noncausal Systems	151
3.2.2	Static (Memoryless) and Dynamic (with Memory) Systems	152
3.2.3	Linear and Nonlinear Systems.....	155
3.2.4	Time-Invariant and Time-Variant Systems	158
3.2.5	Invertible and Non-Invertible Systems	165
3.2.5.1	Construction of the Inverse System.....	166
3.2.6	Stable and Unstable Systems.....	167
3.3	Solved Problems.....	168
3.4	Homework Problems.....	176
4.	Time Domain System Analysis	179
4.1	Impulse Response.....	179
4.2	Continuous-Time Convolution	179
4.2.1	Computation of Convolution	180
4.2.2	The Command conv	186
4.2.3	Deconvolution	188
4.2.4	Continuous-Time Convolution Examples	189
4.3	Convolution Properties	199

4.4	Interconnections of Systems	202
4.5	Stability	206
4.6	Discrete-Time Convolution	208
4.6.1	The Unit Impulse Sequence as Input to a System	208
4.6.2	Computation of Discrete-Time Convolution	211
4.6.3	Discrete-Time Convolution Examples	219
4.7	Systems Described by Difference Equations	223
4.8	Filters.....	224
4.8.1	The Command <code>filter</code>	224
4.8.2	Infinite Impulse Response Filters	228
4.8.3	Finite Impulse Response Filters	232
4.9	Stability Criterion for Discrete-Time Systems.....	234
4.10	Systems Described by Differential Equations.....	235
4.11	Step Response of a System	236
4.12	Solved Problems.....	237
4.13	Homework Problems.....	245
5.	Fourier Series.....	249
5.1	Orthogonality of Complex Exponential Signals.....	249
5.2	Complex Exponential Fourier Series	250
5.3	Trigonometric Fourier Series	253
5.4	Fourier Series in the Cosine with Phase Form.....	256
5.5	Plotting the Fourier Series Coefficients.....	258
5.6	Fourier Series of Complex Signals.....	263
5.7	Fourier Series of Periodic Signals	265
5.8	Line Spectra.....	270
5.9	Properties of Fourier Series.....	272
5.9.1	Linearity	272
5.9.2	Time Shifting.....	273
5.9.3	Time Reversal	275
5.9.4	Time Scaling.....	275
5.9.5	Signal Multiplication	276
5.10	Symmetry	277
5.10.1	Even Symmetry	277
5.10.2	Odd Symmetry	278
5.11	Parseval's Identity	280
5.12	Criterion for the Approximation of a Signal by a Fourier Series Expansion ...	281
5.13	Relationship between Complex Exponential and Trigonometric Fourier Series Coefficients.....	283
5.14	Solved Problems.....	285
5.15	Homework Problems.....	297
6.	Fourier Transform.....	301
6.1	Mathematical Definition.....	301
6.2	The Commands <code>fourier</code> and <code>ifourier</code>	302
6.3	Fourier Transform Pairs	304
6.4	Properties of Fourier Transform	305
6.5	Convolution in Time and Frequency	311

6.6	Symmetry of the Real and Imaginary Parts of Fourier Transform	312
6.7	Parseval's Theorem	313
6.8	Autocorrelation and Cross-Correlation.....	314
6.9	Solved Problems.....	318
6.10	Homework Problems.....	324
7.	Fourier Analysis of Discrete-Time Signals.....	327
7.1	Discrete-Time Fourier Transform	327
7.2	Properties of Discrete-Time Fourier Transform.....	329
7.3	Parseval's Theorem for Discrete-Time Fourier Transform.....	336
7.4	Discrete Fourier Transform.....	336
7.5	Properties of Discrete Fourier Transform	339
7.6	Inverse Discrete Fourier Transform.....	341
7.7	Circular Shift of a Sequence.....	342
7.7.1	Discrete Fourier Transform of a Circularly Shifted Sequence.....	346
7.8	Circular Convolution.....	347
7.8.1	Discrete Fourier Transform of Circular Convolution	351
7.8.2	Relationship between Linear and Circular Convolution	352
7.9	Fast Fourier Transform.....	353
7.10	Relationship between DFT and DTFT	357
7.11	Relationship between Fourier Transform and Discrete Fourier Transform ...	360
7.12	Linear Convolution Computation via Fast Fourier Transform.....	361
7.13	Solved Problems.....	362
7.14	Homework Problems.....	370
8.	Frequency Response.....	373
8.1	Continuous-Time Frequency Response	373
8.2	The Command <code>freqs</code>	376
8.2.1	The Command <code>invfreqs</code>	381
8.3	The Command <code>lsim</code>	383
8.4	System Response to Sinusoidal Input.....	384
8.5	Ideal Filters.....	389
8.6	Frequency Response of Discrete-Time Systems.....	394
8.7	The Command <code>freqz</code>	396
8.7.1	The Command <code>invfreqz</code>	397
8.8	System Response to Discrete-Time Sinusoidal Input	399
8.9	Moving Average Filter	399
8.10	Solved Problems.....	401
8.11	Homework Problems.....	411
9.	Laplace Transform	415
9.1	Mathematical Definition.....	415
9.2	Commands <code>laplace</code> and <code>ilaplace</code>	416
9.3	Region of Convergence	419
9.4	Laplace Transform Pairs	420
9.5	Laplace Transform Properties and Theorems	421
9.6	Partial Fraction Expansion of a Rational Function.....	425
9.6.1	The Command <code>residue</code>	429

9.7	Convolution in Time and in Complex Frequency.....	432
9.7.1	Convolution in the Time Domain.....	432
9.7.2	Convolution in the Complex Frequency Domain	433
9.8	Using the Laplace Transform to Solve Differential Equations	433
9.9	Solved Problems.....	436
9.10	Homework Problems.....	441
10.	<i>z</i>-Transform.....	443
10.1	Mathematical Definition.....	443
10.2	Commands <i>ztrans</i> and <i>iztrans</i>	444
10.3	Region of Convergence	446
10.4	<i>z</i> -Transform Pairs	446
10.5	Properties of <i>z</i> -Transform	447
10.6	Partial Fraction Expansion of a Rational Function.....	453
10.6.1	Commands <i>residue</i> and <i>residuez</i>	455
10.7	Using the <i>z</i> -Transform to Solve Difference Equations.....	457
10.8	Solved Problems.....	460
10.9	Homework Problems.....	467
11.	Transfer Function.....	471
11.1	Continuous-Time Systems	471
11.2	The <i>t f</i> Command	473
11.3	Stability of Continuous-Time Systems	475
11.4	Transfer Function in Zero/Pole/Gain Form.....	477
11.5	Interconnections of Systems	478
11.6	Continuous-Time System Response	481
11.7	Discrete-Time Systems.....	485
11.8	The Command <i>t f</i> for Discrete-Time Systems.....	486
11.9	Stability of Discrete-Time Systems	486
11.10	Discrete-Time System Response.....	489
11.10.1	Step Response	489
11.10.2	Impulse Response	491
11.10.3	The Command <i>d1sim</i>	493
11.11	Conversion between Continuous-Time and Discrete-Time Systems	494
11.12	Transfer Function and Frequency Response	495
11.13	Bode Plot.....	498
11.14	State-Space Representation	499
11.14.1	Construction of a State-Space Model	503
11.14.2	Discrete-Time State-Space Models.....	506
11.15	Solved Problems	508
11.16	Homework Problems.....	518
12.	Suggested Laboratory Exercises	523
12.1	Laboratory 1: Introduction to MATLAB.....	523
12.2	Laboratory 2: Signals	524
12.3	Laboratory 3: Systems	525
12.4	Laboratory 4: Time Domain System Analysis	525
12.5	Laboratory 5: Fourier Series	526
12.6	Laboratory 6: Fourier Transform	527

12.7 Laboratory 7: Fourier Analysis of Discrete-Time Systems.....	528
12.8 Laboratory 8: Frequency Response	528
12.9 Laboratory 9: Laplace Transform	529
12.10 Laboratory 10: z-Transform	530
12.11 Laboratory 11: Transfer Function	531
Appendix A: Signal Crossword.....	533
Appendix B: Notation	535
Bibliography.....	537
Index	539

This page intentionally left blank

Preface

The objective of this book is to provide a comprehensive and practical coverage of the concepts of signals and systems theory. As the title suggests, it is intended for use at the laboratory part of a signals and systems course or as a companion to a standard textbook. However, the coverage of the theoretical aspects is complete, which makes this book useful for engineers, scientists, and students who want to learn or refresh their knowledge on this subject, while learning at the same time how to apply computer methods to signals and systems analysis. The positive features of this book are as follows:

- The extensive use of MATLAB®. More than 5000 lines of code have been included, making it unique in the category of textbooks teaching signals and systems with the use of MATLAB. Every theoretical concept is accompanied by a corresponding MATLAB implementation. Thus, a student can broaden his or her understanding on the different aspects of this subject by learning ways to approach and resolve problems using software programs.
- The MATLAB code used in this book and other learning materials are available in a downloadable supplement from <http://www.crcpress.com/product/isbn/9781439830550>.
- The use of computers to quickly visualize solutions to problems discussed. More than 700 figures are provided in the book, allowing the reader instant access to simulation results without having to use his or her computer.
- More than 400 examples and solved problems are covered in the book, providing an in-depth perspective on problems encountered in signal processing and system analysis.
- The use of step-by-step examples, as data is presented in tabular format. The left part of each example provides the relevant MATLAB statements, the middle part presents the results of the executed statements, and the right part explains the statements and the corresponding results. This type of presentation has a high educational value as it enables the reader to understand the function of each code. Moreover, in the usual case, a problem is solved in several ways, improving the reader's general understanding of how to approach problems.
- It enables students to learn easily how to program in MATLAB while learning at the same time signals and systems concepts. The book was written under the assumption that the reader does not have any previous experience in MATLAB. Thus, a complete MATLAB tutorial covering all of the commands (and many more) used in the next chapters is included.
- Continuous-time and discrete-time signals and systems are treated in parallel fashion. In this way, the similarities and the differences between the two cases are pointed out effectively.
- Finally, eleven laboratory exercises are provided in order to help a lecturer easily organize a laboratory component for the course on signals and systems, as we believe that a MATLAB approach for this course is very effective and enables students to get hands-on experience in this field.

Overview of the Book

Chapter 1 provides an introduction to MATLAB. Starting from the very basics, that is, how to perform simple arithmetic operations, we proceed to vectors and matrices and introduce how to plot functions in two- and three-dimensional space. Next, we study complex numbers and learn how to program in MATLAB by using M-files (scripts and functions). Following this, the symbolic math toolbox is employed for computing integrals, derivatives, and summations of functions, and for solving algebraic and differential equations. The chapter ends with a discussion on polynomials and random numbers.

Chapter 2 introduces the concept of signals. Signals are classified according to their basic characteristics and properties. Elementary signals in continuous-time and in discrete-time systems are introduced. We discuss the periodicity of a signal and learn how to derive if the sum of signals is periodic. We also establish how a signal can be decomposed in even and odd parts and how to determine if a signal is of energy type or power type. Finally, we present the possible transformations of the time variable for continuous-time and discrete-time signals.

Chapter 3 deals with continuous-time and discrete-time systems. The main categories in which systems are classified are given and the basic properties of systems are analyzed. More specifically, we establish and demonstrate through examples the following properties: linearity, causality, memory, time invariance, stability, and invertibility. These are examined for various continuous-time and discrete-time systems.

Chapter 4 begins by establishing the concept of impulse response. We next introduce the convolution operation between two continuous-time signals, discuss the possible interconnections between systems, and present a stability criterion based on a system's impulse response. We also introduce convolution between two discrete time signals and state how a system can be described by a difference or a differential equation. Finally, we introduce finite impulse response and infinite impulse response filters and discuss ways to compute the response of these systems to various input signals.

Fourier series are described in Chapter 5. We first present the three forms of Fourier series and explain how a periodic signal can be expanded in Fourier series. Next, we discuss ways to plot Fourier series coefficients and introduce the main properties of Fourier series. Finally, we establish the approximation criterion of a signal by a Fourier series expansion.

Chapter 6 introduces the Fourier transform of continuous-time signals. The Fourier transform and the inverse Fourier transform of various signals are computed and the most common Fourier transform pairs are presented. Next, we discuss the main properties of Fourier transform and learn to compute the energy of a signal in the frequency domain. The chapter ends with a discussion on the autocorrelation of a signal and its relationship with the signal spectrum.

Chapter 7 covers the two types of Fourier transform that are applicable to discrete time signals: discrete time Fourier transform (DTFT) and discrete Fourier transform (DFT). The properties of these two transforms as well as the relationship between them and between the DFT and the continuous-time Fourier transform are established. The fast Fourier transform (FFT) algorithm is also described and its usefulness in real-time applications is proven by a speed test. Additionally, we introduce the circular shift of a sequence, the circular convolution between two sequences, and, finally, the relationship between linear convolution, circular convolution, and DFT.

Chapter 8 deals with the frequency response of continuous-time and discrete-time systems. We discuss ways to compute the system response to an arbitrary signal based

on its frequency response, and establish the relationship that computes the response of a system to a sinusoidal input signal. We next introduce ideal filters both in the time domain and in the frequency domain. The chapter ends with a discussion on the N -point moving average filter.

Chapter 9 covers the Laplace transform of a continuous-time signal. We discuss ways to compute the Laplace transform and the inverse Laplace transform of a signal as well as the region of convergence of Laplace transform. The most common Laplace transform pairs and properties are introduced. We also describe the procedure of expanding in partial fraction form a function that is written in rational form. Finally, we learn how Laplace transform can be used to compute the solution of a differential equation.

Chapter 10 deals with the z -transform of a discrete time signal. We discuss how to compute the z -transform of a sequence and the inverse z -transform of a function by presenting the most common z -transform pairs. Next, we introduce the properties of z -transform and describe how a function written in rational form can be expanded in partial fraction form. The chapter ends with a discussion on the use of z -transform in computing the solution of a difference equation.

In Chapter 11, we define the transfer function for continuous-time and discrete-time systems. The stability of a system is derived from the system transfer function while the possible interconnections between systems are described in terms of their transfer functions. We also learn how to compute and plot the step response, the impulse response, the response to an arbitrary input signal, and the system frequency response from the transfer function of the system. Finally, we introduce the state space representation for continuous-time and discrete-time systems.

Chapter 12 consists of the proposed laboratory exercises while an innovative revision exercise is given in Appendix A.

We hope that the readers will benefit from this book while at the same time enjoying their read, and we are looking forward to receiving questions, comments, and suggestions regarding our work at palamid@gmail.com.

For MATLAB® and Simulink® product information, please contact

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA, 01760-2098 USA
Tel: 508-647-7000
Fax: 508-647-7001
E-mail: info@mathworks.com
Web: www.mathworks.com

This page intentionally left blank

Authors

Dr. Alex Palamides is with the European Space Agency, European Space Research and Technology Centre, in Noordwijk, the Netherlands. He is the author/coauthor of several research publications in journals and conferences, and has authored another textbook. His research interests lie in the areas of signal processing, dynamic systems, telecommunications, and differential equations.

Prof. Anastasia Veloni is with the Technological Educational Institute of Piraeus, Department of Electronic and Computer Systems, in Athens, Greece. She has extensive educational experience in a variety of courses in the field of signals and systems. She is the author/coauthor of three other textbooks. Her research interests lie in the areas of signal processing, dynamic systems, and automatic control.

This page intentionally left blank

1

Introduction to MATLAB®

The aim of this chapter is to make the reader familiar with the programming tool MATLAB. No previous knowledge of MATLAB is assumed, hence the introduction is given in a very analytical way, starting from the very basics. Before starting our tour to the wonderful world of MATLAB let us note some conventions made during the writing of this book:

1. In some cases, a command or a result is written in more than one line. You should consider that the command is written in one line. In the usual case, the result also occupies a single line.
2. There are times that the results of a MATLAB command are given in a slightly different format (e.g., fewer spaces or less decimal digits) from the one that you will see in your computer. This is made to save space.

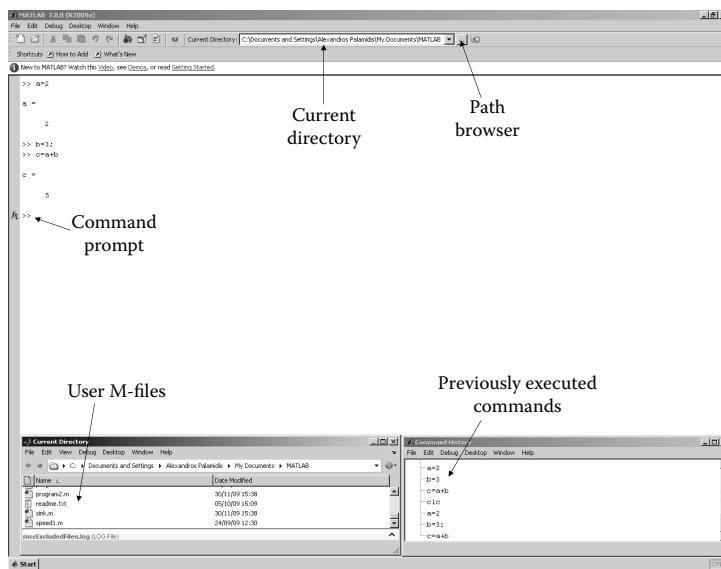
We are ready to start our introduction to the programming tool MATLAB.

1.1 What Is MATLAB?

MATLAB is a high-level technical computing language equipped with a user-friendly interface. Its name stems from the words MATrix and LABoratory as it is based on the use of matrices. MATLAB is a product of Mathworks, and the first version of MATLAB was released in 1984. It was first adopted by the control engineering and applied mathematics communities, but as years passed, numerous functions (categorized in toolboxes) were embedded making MATLAB an extremely powerful tool useful for scientists and engineers from various disciplines. For example, MATLAB can be used in a wide range of applications, such as telecommunications, signal and image processing, control, mathematics, financial modeling, bioengineering, aeronautics, and many more. It is also very useful and popular in education because a student can easily derive handy results without facing many difficulties, as the language syntax and structure is very simple and the data analysis and visualization is straightforward. With the help of MATLAB in this book the reader is expected to understand in a comprehensive and practical way the difficult concepts of the signals and systems theory.

1.2 Working Environment

The working environment of MATLAB consists of several windows that are illustrated in Figure 1.1.

**FIGURE 1.1**

The working environment of MATLAB.

- The *Command* window is the main window of the program. The commands are typed at the command prompt and the obtained results also appear in the command window.
- In the *Command History* window one can see all the previously executed commands.
- Moreover, at the lower left part of the figure we notice the *Current Directory*. The current directory specifies the active folder of MATLAB. By default the current directory is the folder *work*.

There are two possible ways to work with MATLAB. The first is through the command window where the user types and executes the commands one by one, while the second option is to write a sequence of commands in the text editor, i.e., to formulate a program and execute the commands in a batch mode. This kind of programs are known as *scripts*, and are just like any other program written in another programming language. A script file is executed either directly from the editor window from the menu *Debug*—*Run* or by typing the script's name at the command prompt. In any case, the script file must be saved in the *Current Directory*. The extension of this file type is *.m*, and this is why scripts (together with functions) are sometimes called M-Files.

1.3 Getting Started

In this section, we illustrate how to perform some basic operations on MATLAB. The presentation is done in the form of tables. In the left part we give the MATLAB command, in the middle we present the result of the executed command, while at the right part the command and the corresponding results are explained.

1.3.1 Simple Arithmetic Operations

The four operators $+$, $-$, $*$, and $/$ are used as usual to add, subtract, multiply, and divide two numbers. In MATLAB a left division operator \ is also available, and the power of a number is computed by the operator $^$.

Commands	Results	Comments
$3+5$	$ans = 8$	% Addition
$5-2$	$ans = 3$	% Subtraction
$6*7$	$ans = 42$	% Multiplication
$2/4$	$ans = 0.5$	% Division
$2\backslash 4$	$ans = 2$	% Left division
2^3	$ans = 8$	% Power

1.3.2 Comments

Notice the symbol %. The symbol % is used to insert comments. Anything written to the right of % is considered comment and is not taken into account from MATLAB.

1.3.3 The Variable *ans*

The symbol *ans* is a variable. By default, the result of the last executed MATLAB command is assigned to the variable *ans*. To see the value which is stored in *ans*, simply type the word *ans* at the command prompt.

Commands	Results	Comments
<i>ans</i>	$ans = 8$	The result of the most recently executed command is assigned to the variable <i>ans</i> .
$ans = 3$	$ans = 3$	Additionally, a value can be assigned directly to <i>ans</i> .

Remark

In order to recall a previously executed command you can press the up-arrow key of your keyboard.

1.3.4 Priority of Operations

Multiple operations can be performed in a single statement. The priority in which the operations are executed is common to the one used in all programming languages: power, multiplication-division, addition-subtraction. Also, the parentheses are used in the usual way.

Commands	Results	Comments
$3+7*2$	$ans = 17$	The multiplication is executed before the addition.
$2^2+2*(3-2*(1+14/7))$	$ans = -2$	The priority is specified from the parentheses.

A decimal number is defined by employing the dot operator. If the integer part is zero it can be omitted for brevity. A negative number is defined by writing the minus sign before the number.

Commands	Results	Comments
$0.4 + 1.2$	$ans = 1.6000$	Operations between decimal numbers.
$.2 + .4$	$ans = 0.6000$	If the integer part of a decimal number is zero it is possible to omit it.
$-3-1$	$ans = -4$	Operations between negative numbers.

A useful format of writing a number is to express it as a power of ten. This format is often called scientific. The short way of writing a number x in the form $y \cdot 10^n$ is as follows.

Commands	Results	Comments
$1e3$	$ans = 1000$	$1 \cdot 10^3$.
$3e1$	$ans = 30$	$3 \cdot 10^1$.
$4e-2$	$ans = 0.0400$	$4 \cdot 10^{-2}$.
$36e9$	$ans = 3.6000e + 10$	$36 \cdot 10^9$. The result is written as $3.6 \cdot 10^{10}$.

1.3.5 Constants

In MATLAB there are also constants. Some of them are presented in the table below.

Commands	Results	Comments
pi	$ans = 3.1416$	Mathematical constant π .
i	$ans = 0 + 1.0000i$	Imaginary unit i (used to define complex numbers).
j	$ans = 0 + 1.0000i$	Alternative representation of the imaginary unit.
inf	$ans = Inf$	Infinity (∞).
$-inf$	$ans = -Inf$	Minus infinity ($-\infty$).
eps	$ans = 2.2204e - 016$	Spacing of floating point numbers. This is the highest accuracy in which the result of an operation can be calculated.
NaN	$ans = NaN$	Unidentified number (not a number).
$realmax$	$ans = 1.7977e+308$	Largest positive floating point number defined in MATLAB.
$realmin$	$ans = 2.2251e - 308$	Smallest positive floating point number defined in MATLAB.

1.3.6 Built-In Functions

There are many built-in functions available in MATLAB. Some of them are presented below.

Commands	Results	Comments
<code>sqrt(2)</code>	<code>ans = 1.4142</code>	Square root.
<code>abs(-4)</code>	<code>ans = 4</code>	Absolute value.
<code>sign(3)</code>	<code>ans = 1</code>	The function <i>sign</i> returns 1 if the input number is positive, 0 if the input is zero, and -1 if the input is negative.
<code>exp(1)</code>	<code>ans = 2.7183</code>	Exponential.
<code>log(1)</code>	<code>ans = 0</code>	Natural logarithm.
<code>log10(100)</code>	<code>ans = 2</code>	Base 10 logarithm.
<code>log2(4)</code>	<code>ans = 2</code>	Base 2 logarithm.
<code>sin(pi/2)</code>	<code>ans = 1</code>	Sine.
<code>cos(pi)</code>	<code>ans = -1</code>	Cosine.
<code>tan(pi/4)</code>	<code>ans = 1</code>	Tangent.
<code>asin(0.5)</code>	<code>ans = 0.5236</code>	Arcsine (inverse sine).
<code>acos(-1)</code>	<code>ans = 3.1416</code>	Arccosine (inverse cosine).
<code>atan(1)</code>	<code>ans = 0.7854</code>	Arctangent (inverse tangent).
<code>sinh(0.1)</code>	<code>ans = 0.1002</code>	Hyperbolic sine.
<code>cosh(0.2)</code>	<code>ans = 1.0201</code>	Hyperbolic cosine.
<code>tanh(pi/6)</code>	<code>ans = 0.4805</code>	Hyperbolic tangent.
<code>ceil(0.6)</code>	<code>ans = 1</code>	Rounding toward the closer larger integer.
<code>floor(0.6)</code>	<code>ans = 0</code>	Rounding toward the closer smaller integer.
<code>round(0.6)</code>	<code>ans = 1</code>	Rounding toward the closer integer.
<code>fix(0.6)</code>	<code>ans = 0</code>	Rounding toward the closer-to-zero integer.
<code>factorial(5)</code>	<code>ans = 120</code>	Factorial function.

Remark

The input argument of a trigonometric function must be in *radians*. To convert degrees to radians we multiply degrees by $\pi/180$.

Example

Compute the cosine of 60° .

In order to not get confused, an easy way is to use the proportion method:

$$\frac{180}{60} = \frac{\pi}{x} \Rightarrow x = 60 \frac{\pi}{180}.$$

Hence,

Commands	Results	Comments
<code>cos(60*pi/180)</code>	<code>ans = 0.5000</code>	Cosine of 60°

1.3.7 Variables

At MATLAB there is no need to declare the variables before assigning them a value. A variable can have any name as long as it does not start with a number and it does not include spaces or special characters. Finally, there is a distinction between capital and small letters.

Example

Compute the expressions

- $\sin(45^\circ)^2 + \cos(45^\circ)^2$
- $\sin(45^\circ) \cdot \cos(45^\circ)$

Commands	Results	Comments
<code>theta=45*pi/180</code>	<code>theta = 0.7854</code>	First, we convert the 45° to radians. The result is assigned to a variable named <i>theta</i> .
<code>a=sin(theta)</code>	<code>a = 0.7071</code>	Definition of the variable <i>a</i> .
<code>b=cos(theta)</code>	<code>b = 0.7071</code>	Definition of the variable <i>b</i> .
<code>c=a^2+b^2</code>	<code>c = 1</code>	The result of the first expression is assigned to the variable <i>c</i> .
<code>a*b</code>	<code>ans = 0.5000</code>	The second expression is computed and the result is stored in the default variable <i>ans</i> .
<code>d=c+ans</code>	<code>d = 1.5000</code>	The variable <i>ans</i> can be used for further computations.
<code>A=6</code>	<code>A = 6</code>	Definition of the variable <i>A</i> .
<code>a=3</code>	<code>a = 3</code>	Definition of the variable <i>a</i> .
<code>c=A+a</code>	<code>c = 9</code>	It is obvious that there is distinction between small and capital letters.

Sometimes it is tedious to have all results displayed. In order to not display a result, we can type a question mark ";" after the command.

Commands	Results	Comments
<code>d=A+a;</code>		Nothing is displayed at the command window. This of course does not mean that the command is not executed.
<code>d</code>	<code>d = 9</code>	To see the result simply type the variable to which it is assigned.

Finally we mention that text can be also assigned to a variable. The text is inserted between single quotes.

Commands	Results	Comments
x='hello'	x = hello	Text assignment to a variable.

1.3.8 Format

The option of specifying the number of decimal digits that are displayed is available in MATLAB. Also, there are other options to set the output format. By default, four decimal digits are displayed, and there are blank lines between the command the variable and the result. By executing the command `format` we return to the default appearance. The possible formats are illustrated below using π .

Commands	Results	Comments
<code>format pi</code>	<code>ans = 3.1416</code>	Default format, i.e., 4 decimal digits and blank horizontal lines.
<code>format compact pi</code>	<code>ans = 3.1416</code>	Suppression of blank lines.
<code>format loose pi</code>	<code>ans = 3.1416</code>	Blank lines are inserted between the commands the variable and the result.
<code>format short pi</code>	<code>ans = 3.1416</code>	Format with 4 decimal digits.
<code>format long pi</code>	<code>ans = 3.14159265358979</code>	Format with 14 decimal digits.
<code>format short e pi</code>	<code>ans = 3.1416e+000</code>	Format with 4 decimal digits and the number expressed as power of 10.
<code>format long e pi</code>	<code>ans = 3.141592653589793e+000</code>	Format with 14 decimal digits and the number expressed as power of 10.
<code>format bank pi</code>	<code>ans = 3.14</code>	Format with 2 decimal digits.
<code>format hex a=1</code>	<code>a = 3ff000000000000</code>	Hexadecimal format.

In this book we use compact format in order to save space.

1.3.9 Help in MATLAB

`help-lookfor-what-which`

A complete manual of the MATLAB commands is offered through the command prompt (also available as an html help file). The command `help` in MATLAB is the counterpart of command `man` in Linux. Typing `help` and the name of the command displays a short

explanation of how the command is used, available syntaxes, examples, and similar commands. If one types only `help`, all the available MATLAB commands (divided in categories) with a short description are displayed in the command window.

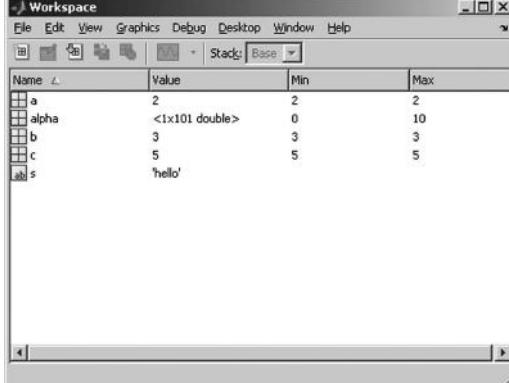
Commands	Results	Comments
<code>help cos</code>	<pre>COS Cosine. COS(X) is the cosine of the elements of X. Overloaded methods help sym/cos.m</pre>	Help about the command <code>cos</code> .
<code>help plot</code>	<pre>PLOT Linear plot. PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix, then the vector...... See also SEMILOGX, SEMILOGY.....</pre>	Help about the command <code>plot</code> . Notice that similar commands are also displayed.

Some other related commands are

Commands	Comments
<code>look for text</code>	Returns all commands and functions where the word “ <i>text</i> ” exists.
<code>what folder</code>	Returns all the files that are in the directory “ <i>folder</i> ”.
<code>which name</code>	Returns the directory in which the file with filename “ <i>name</i> ” is.

1.4 Memory Management

The variables are saved in a memory region named `workspace`. To view the saved variables the command `workspace` is used. Alternatively, one may use the commands `who` and `whos`.

Commands	Results	Comments
<code>workspace</code>		The workspace window is displayed. Inside the window the assigned variables are shown. Editing an existing variable is possible through the workspace.
<code>who</code> <code>whos</code>	Your variables are: a alpha b c s Name Size Bytes Class a 1x1 8 double array alpha 1x101 808 double array b 1x1 8 double array c 1x1 8 double array s 1x5 10 char array Grand total is 109 elements using 842 bytes	The assigned variables. The assigned variables and their size and type.

1.4.1 Commands save-load-exit-quit

Assigned variables stay in memory while MATLAB is running. If we close and reopen it the variables are lost. We can use the command `save` in order to store permanently the assigned variables. The saved variables can be retrieved back to the memory by typing the command `load`.

Commands	Comments
<code>save variables</code>	The workspace variables are saved in a file named <i>variables.mat</i> . If the <code>save</code> command is executed without specifying an output file the variables are saved in a file named <i>matlab.mat</i> . The <i>variables.mat</i> file is saved in the working directory.
<code>exit</code> <code>% or</code> <code>quit</code>	The commands <code>exit</code> and <code>quit</code> terminate MATLAB.
We open MATLAB.	
<code>load variables</code>	The variables that are saved in the file <i>variables.mat</i> are loaded in MATLAB and can be used now.

1.4.2 The Command `clear`

The command `clear` is employed in order to delete a variable from the memory. The syntax is `clear` and the variable(s) name(s).

Remark

If the command `clear` is executed without specifying the variable to be deleted, all workplace variables are deleted. The command `clear` (or `clear all`) should be used to clear the memory before executing large programs.

Commands	Results	Comments
<code>a = 5</code>	<code>a = 5.00</code>	Assign a value to variable <i>a</i> .
<code>clear a</code>		Deleting variable <i>a</i> from memory.
<code>a</code>	??? Undefined function or variable 'a'.	Indeed variable <i>a</i> is not defined.
<code>clear</code>		Deleting all variables from the memory.
<code>who</code>		Indeed the memory is null.

1.5 Commands `diary` and `clc`

Sometimes it is necessary to store the commands and whatever else is displayed in the command window. The command `diary` is employed to achieve that. Typing `diary` on anything that is displayed in the command window is recorded in a file named *diary.dia*. The *diary.dia* file is saved in the working directory. Typing `diary off` stops the recording process.

Commands	Comments
<code>diary memoir.dia</code>	The file <i>memoir.dia</i> is created.
<code>A = 5 B = 6;</code>	Commands are recorded in the file <i>memoir.dia</i> .
<code>diary off</code>	Stop of the record process.

The command `clc` (named after the initial letters of clear command) clears the command window from anything written before. It must not be confused with the command `clear` as it does not delete anything from the memory.

1.6 Vectors

MATLAB (named after MATrix LABoratory) is a programming tool specialized for working with matrices (or arrays). Even the scalars are considered as matrices of size 1×1 . In this section, we introduce a special case of matrices, the vectors.

1.6.1 Row Vectors

To define a row vector, the elements of the vector are given into square brackets [...]. Spaces or commas are inserted between the elements. A vector element is specified by its index. The index numbering starts from 1, namely, the index of the first element of a vector is 1. In order to refer to a position in a vector, parentheses are used.

Commands	Results	Comments
<code>a = [1 2 3 4 5]</code>	<code>a = 1 2 3 4 5</code>	Row vector a of 5 elements.
<code>b = [-6, -7, -8, -9, -10]</code>	<code>b = -6 -7 -8 -9 -10</code>	Row vector b of 5 negative elements.
<code>c = [0.3, 8.5, 6, -2.4]</code>	<code>c = 0.30 8.50 6.00 -2.40</code>	Row vector c of 4 decimal elements.
<code>b(3)</code>	<code>ans = -8</code>	The third element of vector b .
<code>b(6)</code>	<code>??? Index exceeds matrix dimensions.</code>	Vector b has 5 elements. Thus, referring to its 6th position causes an error.

1.6.2 Commands `length`/`size`

In order to compute the number of elements of a vector, the appropriate command is the command `length`. A similar command is the command `size`. Command `size` is usually employed to compute the dimensions of a matrix. However, vectors are special case of matrices; thus, using the command `size` to compute the length of a vector is completely valid.

Commands	Results	Comments
<code>length(a)</code>	<code>ans = 5</code>	Number of elements of vector a .
<code>size(a)</code>	<code>ans = 1 5</code>	Vector a consists of 1 row and 5 columns.

It is also possible to edit only one specific element of a vector by referring to the position (index) of the vector. If between the position and the rest vector elements there are no defined elements, the null positions are filled with zeros.

Commands		Results					Comments
a	a = 1	2 3 4 5					Vector <i>a</i> is already defined.
a(3) = 99	a = 1	2 99 4 5					The value 99 is inserted at the third position of vector <i>a</i> , replacing the previous value.
a(10) = 23	a = 1 0	2 99 4 5 0 0 0 23					The value 23 is inserted at the tenth position of <i>a</i> . Notice that the intermediate (6th, 7th, 8th, and 9th) positions are filled with zeros.

This way of defining row vectors is quite hard and inappropriate for large-size vectors. In case that the vector elements are equally spaced a vector is defined by giving the first element, the step, that is, the distance between two consecutive elements and the last element separated by the colon operator “:”. In other words, a vector with equally spaced elements is defined using the syntax *a*=First element:Step>Last element. If the step is 1, it can be omitted. The step can be negative, decimal, or even irrational number. Furthermore, if a vector is defined in this way the use of square brackets is not necessary.

Commands		Results							Comments
a=1:2:11	a = 1	3 5 7 9 11							The elements of vector <i>a</i> start from 1, end at 11, and are spaced by 2.
b=1:6	b = 1	2 3 4 5 6							If no step is specified, by default the step considered is 1.
c=0:0.2:1	c = 0	0.2 0.4 0.6 0.8 1.0							Decimal step.
d=3:-1:-3	d = 3	2 1 0 -1 -2 -3							Negative step.
e=0:pi/6:pi	e = 0	0.5236 1.0472 1.5708 2.0944 2.6180 3.1416							Vector <i>e</i> is defined from 0 to π with step $\pi/6$.

1.6.3 Addition/Subtraction

Operations between vectors are usually performed between the corresponding vector elements, i.e., they are *operations per element*. A necessary condition for the operation between two vectors is that the vectors must be of same size.

- Suppose that $a = [a_1 \ a_2 \ \dots \ a_n]$ and $b = [b_1 \ b_2 \ \dots \ b_n]$. Then, $a + b = [a_1 + b_1 \ a_2 + b_2 \ \dots \ a_n + b_n]$ and $a - b = [a_1 - b_1 \ a_2 - b_2 \ \dots \ a_n - b_n]$.

Commands		Results							Comments
add=a+b	add = 2	5 8 11 14 17							Vector <i>add</i> is the sum of vectors <i>a</i> and <i>b</i> .
sub=a-b	sub = 0	1 2 3 4 5							Subtraction of vectors.

1.6.4 Multiplication, Division, and Power

For this type of operations, attention must be paid to the fact that this is an element per element operation. To perform multiplication and division between two vectors or to compute the power of a vector the dot operator “.” has to be inserted before the operator.

- Suppose that $a = [a_1 \ a_2 \ \dots \ a_n]$ and $b = [b_1 \ b_2 \ \dots \ b_n]$. Then

$$\begin{aligned} a.*b &= [a_1 b_1 \ a_2 b_2 \ \dots \ a_n b_n] \\ a./b &= [a_1/b_1 \ a_2/b_2 \ \dots \ a_n/b_n] \\ a.\backslash b &= [a_1 \backslash b_1 \ a_2 \backslash b_2 \ \dots \ a_n \backslash b_n] \\ a.^k &= [a_1^k \ a_2^k \ \dots \ a_n^k], \text{ where } k \text{ is a number} \\ k.^a &= [k^{a_1} \ k^{a_2} \ \dots \ k^{a_n}], \text{ where } k \text{ is a number} \\ a.^b &= \left[a_1^{b_1} \ a_2^{b_2} \ \dots \ a_n^{b_n} \right] \end{aligned}$$

Commands	Results	Comments
a	$a = 1 \ 3 \ 5 \ 7 \ 9 \ 11$	Vectors a and b .
b	$b = 1 \ 2 \ 3 \ 4 \ 5 \ 6$	
a.*b	$ans = 1 \ 6 \ 15 \ 28 \ 45 \ 66$	Multiplication element per element.
a./b	$ans = 1.00 \ 1.50 \ 1.66 \ 1.75$ 1.80 1.83	Division element per element.
a.\b	$ans = 1.0000 \ 0.6667 \ 0.6000$ 0.5714 0.5556 0.5455	Left division element per element.
a.^2	$ans = 1 \ 9 \ 25 \ 49 \ 81 \ 121$	Each element of a is raised to the power of 2.
2.^a	$ans = 2 \ 8 \ 32 \ 128 \ 512 \ 2048$	2 is raised to each element of a .
a.^b	$ans = 1 \ 9 \ 125 \ 2401 \ 59049 \ 1771561$	Each element of a is raised to the power of the corresponding element of b .

In case that a vector is multiplied or divided by a number, the use of dot operator is not necessary. The majority of the built-in MATLAB functions can be applied to vectors. The result is a vector of the same size in which the function is applied to each element.

Commands	Results	Comments
$a = 0:\pi/2:2*\pi$	$a = 0 \ 1.5708 \ 3.1416$ 4.7124 6.2832	Vector a from 0 to 2π with step $\pi/2$. In other words, $a = [0 \ \pi/2 \ 3\pi/2 \ 2\pi]$.
$b = 3*a$	$b = 0 \ 4.7124 \ 9.4248$ 14.1372 18.8496	Multiplication between a number and a vector. The dot operator is not used.
$c = \cos(a)$	$c = 1.0000 \ 0.0000 \ -1.0000$ 0.0000 1.0000	Applying the built-in function \cos at vector a yields a vector c with elements $c = [\cos(0) \ \cos(\pi/2) \ \cos(\pi) \ \cos(3\pi/2) \ \cos(2\pi)]$.

1.6.5 Column Vectors

So far the vectors considered were row vectors. The way to define a column vector is quite similar. In this case, the elements are also inserted between square brackets. However, the elements are not separated by comma but by a question mark “;”. A second way to define a column vector is given now: After opening the square bracket and inserting the first element pressing the Enter key moves us to the next line where the second element is inserted. This process continues for all vector elements. Closing the square brackets and pressing Enter provides us a column vector. A third way is to define a row vector and append the apostrophe (or transpose operator) “'”. The transpose operator transposes the vector, i.e., converts the rows into columns and the columns into rows.

Commands	Results	Comments
<code>a = [1;2;3;4]</code>	<code>a =</code> 1 2 3 4	First way of creating a column vector. The question mark is used to change row.
<code>b = [5 6 7 8]</code>	<code>b =</code> 5 6 7 8	Second way of creating a column vector. By pressing Enter when the square brackets are open also changes row.
<code>c = [9 10 11 12] d = c'</code>	<code>c =</code> 9 10 11 12 <code>d =</code> 9 10 11 12	Third way of creating a column vector. The transpose operator is used to convert a row vector into a column vector.
<code>e = d'</code>	<code>e =</code> 9 10 11 12	The transpose operator also converts a column vector into a row vector.
<code>z = [1:0.1:100]';</code>		Column vector z from 1 to 100 with step 0.1.
<code>size(z)</code>	<code>ans =</code> 991 1	Indeed z is a an array (or matrix) of 991 rows and 1 column.
<code>length(z)</code>	<code>ans =</code> 991	The <code>length</code> command in a column vector returns the number of elements.

1.6.6 Dot Product of Two Vectors

In order to compute the dot product of two vectors, the vectors must be of the same size. Suppose that $a = [a_1 \ a_2 \ \dots \ a_n]$ and $b = [b_1 \ b_2 \ \dots \ b_n]$. The dot product of the vectors a, b is given by

$$ab = \sum_{i=1}^n a_i b_i. \quad (1.1)$$

The dot product of two vectors is a scalar. There are three ways to compute the dot product of two row vectors a and b .

1. The command `dot (a, b)` directly computes the dot product of the vectors a and b .
2. The dot product can be computed according to its definition, that is, can be computed as `sum(a.*b)`. If the command `sum` is applied to a vector it returns the sum of the vector elements.
3. A third way based on the multiplication between two matrices is presented in Section 1.7.4.1.

Commands	Results	Comments
<code>a = 1:4</code>	<code>a = 1 2 3 4</code>	
<code>b = 2:5</code>	<code>b = 2 3 4 5</code>	Definition of the row vectors a and b .
<code>dot1 = dot (a, b)</code>	<code>dot1 = 40</code>	Dot product computed by the command <code>dot</code> .
<code>dot2 = sum(a.*b)</code>	<code>dot2 = 40</code>	The dot product is computed according to its definition. Notice that the use of the dot operator before the multiplication is necessary.

1.6.7 Useful Commands

`sum-cumsum-prod-diff-max-min-sort-mean-median`

In this section, we introduce various useful (when working with vectors) commands. As already discussed, the command `sum` returns the sum of the elements of a vector. The command `cumsum` returns a vector whose elements are the cumulative sum of the previous elements. The command `prod` is the product of the vector elements, while the command `diff` returns a vector in which each element is given by its subtraction with the previous element. The commands `max` and `min` return the largest and smallest elements of the vector, respectively, as well as their index. The command `sort` sorts the vector elements in ascending (by default) or descending order. The command `mean` computes the mean value, while the command `median` returns the median value. All these commands are suitable also for matrices by slightly changing their syntax.

Commands	Results	Comments
<code>a = [4 2 7 0 6]</code>	<code>a = 4 2 7 0 6</code>	Definition of vector a .
<code>s = sum(a)</code>	<code>s = 19</code>	Sum of the elements of a .
<code>c = cumsum(a)</code>	<code>c = 4 6 13 13 19</code>	Cumulative sum. The result is obtained as $[4, 4+2, 4+2+7, 4+2+7+0, 4+2+7+0+6]$.
<code>p = prod(a)</code>	<code>p = 0</code>	Product of all elements.
<code>d = diff(a)</code>	<code>d = -2 5 -7 6</code>	Difference between two consecutive elements, i.e., $d(1) = a(2) - a(1)$, etc.
<code>[m, i] = max(a)</code>	<code>m = 7 i = 3</code>	The largest value is assigned to variable m , and its index is assigned to variable i .
<code>[m, i] = min(a)</code>	<code>m = 0 i = 4</code>	The smaller value is assigned to variable m , and its index is assigned to variable i .
<code>max(a)</code>	<code>ans = 7</code>	If no output variable is specified, only the largest value is returned.
<code>mean(a)</code>	<code>ans = 3.8000</code>	Mean value of the elements.
<code>median(a)</code>	<code>ans = 4</code>	Median value of the vector.
<code>sort(a)</code>	<code>ans = 0 2 4 6 7</code>	Sorting in ascending order.
<code>sort(a, 'descend')</code>	<code>ans = 7 6 4 2 0</code>	Sorting in descending order.

1.7 Matrices

In this section, we introduce how a matrix (or array) is defined in MATLAB. The use of square brackets is necessary. The first way is to type the elements of the first row separated by commas or spaces and then insert a question mark (which corresponds in changing a row), type the elements of the second row, and so on. A second way is to type the elements of the first row, then press Enter to move to the second row, etc. Finally, a third way appropriate for matrices whose elements in each row are equally spaced is to consider a matrix row as a row vector, i.e., to create each matrix row using the syntax First element:Step:Last element and change row by inserting a question mark after the last element of every row.

Commands	Results	Comments
<code>A = [1 2 3; 4 5 6; 7 8, 9]</code>	$\begin{matrix} A = & 1 & 2 & 3 \\ & 4 & 5 & 6 \\ & 7 & 8 & 9 \end{matrix}$	The matrix A is created. It consists of three rows and three columns.
<code>A = [1 2 3 4 5 6 7, 8, 9]</code>	$\begin{matrix} A = & 1 & 2 & 3 \\ & 4 & 5 & 6 \\ & 7 & 8 & 9 \end{matrix}$	Second way of creating a matrix.
<code>D = [0:.1:.5; 0:−.1:−.5; 0:pi/5:pi]</code>	$\begin{matrix} D = & 0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 \\ & 0 & -0.1 & -0.2 & -0.3 & -0.4 & -0.50 \\ & 0 & 0.63 & 1.26 & 1.88 & 2.51 & 3.14 \end{matrix}$	Third way of creating a matrix.

The elements of a matrix are indexed using *two* indices. The first index specifies the row and the second index specifies the column. To refer to a matrix position parentheses are used. The matrix dimensions are computed by the command size.

Commands	Results	Comments
<code>A(1,1)</code>	<code>ans = 1</code>	The first element of matrix A (first row-first column).
<code>A(3,2)</code>	<code>ans = 8</code>	The element in the third row-second column of A.
<code>size(A)</code>	<code>ans = 3 3</code>	The dimensions of matrix A (3×3).

The majority of the built-in MATLAB functions can be also applied to matrices. The result is a matrix of the same size in which the function is applied to the matrix elements.

Commands	Results	Comments
<code>A = [0 pi/2; pi 3*pi/2]</code>	$\begin{matrix} A = & 0 & 1.5708 \\ & 3.1416 & 4.7124 \end{matrix}$	$A = \begin{bmatrix} 0 & \pi/2 \\ \pi & 3\pi/2 \end{bmatrix}$
<code>B = cos(A)</code>	$\begin{matrix} B = & 1.0000 & 0.0000 \\ & -1.0000 & -0.0000 \end{matrix}$	$B = \begin{bmatrix} \cos(0) & \cos(\pi/2) \\ \cos(\pi) & \cos(3\pi/2) \end{bmatrix}$

1.7.1 Matrix Concatenation

Two matrices **A** and **B** that have the same number of rows can be concatenated in order to create a new matrix **C** with the same number of rows. The number of columns of **C** is the sum of the number of columns of **A** and **B**. This process is named matrix concatenation and is applied by typing **[A B]** or **[A, B]**.

Commands	Results					Comments
$\mathbf{A} = [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9]$	$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$					Matrix A of size 3×3 .
$\mathbf{B} = [1 \ 1; \ 2 \ 2; \ 3 \ 3]$	$\begin{matrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{matrix}$					Matrix B of size 3×2 .
$\mathbf{C} = [\mathbf{A} \ \mathbf{B}]$	$\begin{matrix} 1 & 2 & 3 & 1 & 1 \\ 4 & 5 & 6 & 2 & 2 \\ 7 & 8 & 9 & 3 & 3 \end{matrix}$					Matrix concatenation. A new matrix C is created that includes the elements of both A and B .
$[\mathbf{m}, \mathbf{n}] = \text{size}(\mathbf{C})$	$\begin{matrix} \mathbf{m} = 3 \\ \mathbf{n} = 5 \end{matrix}$					The dimensions of matrix C are 3 rows and 5 columns.

Remark

The process of concatenating matrices is a very useful process. The importance of this process will become clear in Section 1.8.10 where we will learn how to define and plot piecewise functions.

Two matrices **A** and **B** that have the same number of columns can be also concatenated if the second matrix **B** is placed below **A** in order to create a new matrix **C** with the same number of columns. The number of rows of **C** is the sum of the number of rows of **A** and **B**. The statement used in this case is **[A; B]**.

Commands	Results					Comments
$\mathbf{A} = [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9]$	$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$					Matrix A of size 3×3 .
$\mathbf{B} = [1 \ 1 \ 1; \ 2 \ 2 \ 2]$	$\begin{matrix} 1 & 1 & 1 \\ 2 & 2 & 2 \end{matrix}$					Matrix B of size 2×3 .
$\mathbf{C} = [\mathbf{A}; \mathbf{B}]$	$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 1 & 1 & 1 \\ 2 & 2 & 2 \end{matrix}$					Matrix concatenation. A new matrix C is created that includes the elements of both A and B .
$[\mathbf{m}, \mathbf{n}] = \text{size}(\mathbf{C})$	$\begin{matrix} \mathbf{m} = 5 \\ \mathbf{n} = 3 \end{matrix}$					The dimensions of matrix C are 5 rows and 3 columns.

1.7.2 Working with Matrices

The way of referring to one element of a matrix was introduced earlier. Additionally, it is possible to refer in rows, columns, or sub-matrices. This is achieved by employing the colon

operator “:” For example, the statement that returns the k th row of \mathbf{A} is $\mathbf{A}(k, :)$. Similarly, the statement that returns the m th column of \mathbf{A} is $\mathbf{A}(:, m)$. A sub-matrix from the k_1 row to the k_2 row and from the m_1 column to the m_2 column of \mathbf{A} is derived by typing $\mathbf{A}(k_1:k_2, m_1:m_2)$. Finally, the statement $\mathbf{A}(:)$ converts matrix \mathbf{A} into a column vector.

Commands	Results	Comments
$\mathbf{A} = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$	$\begin{matrix} \mathbf{A} = 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$	Matrix \mathbf{A} of size 3×3 .
$\mathbf{A}(:, :)$	$\begin{matrix} \mathbf{ans} = 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$	The whole matrix \mathbf{A} . <i>Explanation:</i> The operator “:” means that all rows and all columns are selected.
$\mathbf{A}(2, :)$	$\begin{matrix} \mathbf{ans} = 4 & 5 & 6 \end{matrix}$	The second row of \mathbf{A} . <i>Explanation:</i> From the rows only the second one is picked. From the columns all of them are selected.
$\mathbf{A}(:, 3)$	$\begin{matrix} \mathbf{ans} = 3 \\ 6 \\ 9 \end{matrix}$	The third column of \mathbf{A} . <i>Explanation:</i> All rows are selected, while only the third column is chosen.
$\mathbf{A}(1:2, 1:2)$	$\begin{matrix} \mathbf{ans} = 1 & 2 \\ 4 & 5 \end{matrix}$	The sub-matrix consists of the first two rows and columns of \mathbf{A} . <i>Explanation:</i> From the rows we select from the 1st to the 2nd using step 1. Thus, the 1st and 2nd rows are selected. Similarly, only the first two columns are chosen.
$\mathbf{A}(1:2:3, 1:2:3)$	$\begin{matrix} \mathbf{ans} = 1 & 3 \\ 7 & 9 \end{matrix}$	The sub-matrix contains the first and the third rows and columns of \mathbf{A} . <i>Explanation:</i> From the rows we pick from the 1st to the 3rd using step 2. Thus, the 1st and 3rd rows are selected. Similarly for the columns.
$\mathbf{A}(1:2:3, 1:2)$	$\begin{matrix} \mathbf{ans} = 1 & 2 \\ 7 & 8 \end{matrix}$	The sub-matrix contains the first and the third rows and the first and second columns of \mathbf{A} . <i>Explanation:</i> From the rows we pick from the 1st to the 3rd using step 2. Thus, the 1st and 3rd rows are selected. From the columns we pick from the 1st to the 2nd using step 1. Thus, the 1st and 2nd columns are selected.
$\mathbf{A}(1:2:3, :)$	$\begin{matrix} \mathbf{ans} = 1 & 2 & 3 \\ 7 & 8 & 9 \end{matrix}$	The sub-matrix contains the first and the third rows of \mathbf{A} . <i>Explanation:</i> From the rows we pick from the 1st to the 3rd using step 2. Thus, the 1st and 3rd rows are selected. Using the colon operator “:” all columns are selected.
$\mathbf{A}(:)$	$\begin{matrix} \mathbf{ans} = 1 \\ 4 \\ 7 \\ 2 \\ 5 \\ 8 \\ 3 \\ 6 \\ 9 \end{matrix}$	The statement $\mathbf{A}(:)$ returns the matrix \mathbf{A} as a column vector. The elements are sorted by column.

1.7.3 Addition/Subtraction

The basic operations in matrices are performed very easily. Suppose that **A** and **B** are matrices of size $M \times N$ (i.e., they have M rows and N columns) and let the elements of **A** and **B** be denoted by a_{ij} and b_{ij} , $i = 1, 2, \dots, M$, $j = 1, 2, \dots, N$, respectively. The sum **A + B** is a new matrix **C** of size $M \times N$ and elements c_{ij} given by $c_{ij} = a_{ij} + b_{ij}$. Similarly, the subtraction **A - B** results in a new matrix **D** of size $M \times N$ and elements d_{ij} given by $d_{ij} = a_{ij} - b_{ij}$.

Commands	Results	Comments
	$\begin{array}{ccc} A = & 1 & 2 & 3 \\ & 4 & 5 & 6 \end{array}$	
$A = [1 \ 2 \ 3 \ ; \ 4 \ 5 \ 6 \ ; \ 7 \ 8 \ 9]$	$\begin{array}{ccc} & 7 & 8 & 9 \end{array}$	
$B = [1 \ 1 \ 1 \ ; \ 3 \ 3 \ 3 \ ; \ 5 \ 5 \ 5]$	$\begin{array}{ccc} B = & 1 & 1 & 1 \\ & 3 & 3 & 3 \\ & 5 & 5 & 5 \end{array}$	Matrices A and B of size 3×3 .
$C = A + B$	$\begin{array}{ccc} C = & 2 & 3 & 4 \\ & 7 & 8 & 9 \\ & 12 & 13 & 14 \end{array}$	Matrix addition.
$D = A - B$	$\begin{array}{ccc} D = & 0 & 1 & 2 \\ & 1 & 2 & 3 \\ & 2 & 3 & 4 \end{array}$	Matrix subtraction.

1.7.4 Multiplication of Matrices

The necessary condition for the multiplication between two matrices is that the number of columns of the first matrix must be equal to the number of rows of the second matrix. The result is a matrix that has the same number of rows with the first matrix and the same number of columns with the second one. Suppose that matrix **A** has M rows and N columns and matrix **B** has N rows and K columns. Then matrix **C = A · B** has M rows and K columns. The mathematical expression is

$$(M \times N) \cdot (N \times K) = (M \times K). \quad (1.2)$$

Let $a_{ij}, i = 1, 2, \dots, M, j = 1, 2, \dots, N$ and $b_{ij}, i = 1, 2, \dots, N, j = 1, 2, \dots, K$ denote the elements of **A** and **B**, respectively. The matrix **C** given by the product **A · B** is of size $(M \times K)$ and its elements are computed by

$$c_{ij} = \sum_{p=1}^N a_{ip} b_{pj}, \quad i = 1, 2, \dots, M, \quad j = 1, 2, \dots, K. \quad (1.3)$$

Commands	Results	Comments
$A = [1 \ 2 \ 3; \ 4 \ 5 \ 6]$ $B = [1 \ 1 \ 1 \ 1; \ 2 \ 2 \ 2 \ 2; \ 3 \ 3 \ 3 \ 3]$	$\begin{array}{ccccc} A = & 1 & 2 & 3 \\ & 4 & 5 & 6 \\ B = & 1 & 1 & 1 & 1 \\ & 2 & 2 & 2 & 2 \\ & 3 & 3 & 3 & 3 \end{array}$	Matrix A is of size 2×3 , while matrix B is of size 3×4 .
$C = A * B$	$\begin{array}{cccc} C = & 14 & 14 & 14 \\ & 32 & 32 & 32 \end{array}$	Matrix C is the product $\mathbf{A} \cdot \mathbf{B}$ and its size is 2×4 .
$D = B * A$??? Error using ⇒ * Inner matrix dimensions must agree.	The product $\mathbf{B} \cdot \mathbf{A}$ is not a valid operation as it does not fulfill condition (1.2).

Although the commutative property stands in matrix addition, namely, $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$, when two matrices are multiplied the commutative property is not applicable, i.e., $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$.

Commands	Results	Comments
$A = [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9];$ $B = [3 \ 2 \ 1; \ 6 \ 5 \ 4; \ 9 \ 8 \ 7];$		Matrices A and B of size 3×3 .
$A * B$	$\begin{array}{ccc} ans = & 42 & 36 & 30 \\ & 96 & 81 & 66 \\ & 150 & 126 & 102 \end{array}$	The product $\mathbf{A} \cdot \mathbf{B}$.
$B * A$	$\begin{array}{ccc} ans = & 18 & 24 & 30 \\ & 54 & 69 & 84 \\ & 90 & 114 & 138 \end{array}$	$\mathbf{B} \cdot \mathbf{A}$ is not equal to $\mathbf{A} \cdot \mathbf{B}$.

Matrix multiplication must not be confused with the multiplication between vectors. Recall that vector multiplication is an element per element operation and is implemented by inserting the dot operator before the multiplication operator. Multiplication per element can be performed between two matrices **A** and **B**, if they are of same size.

Commands	Results	Comments
$A = [1 \ 2 \ 3; \ 4 \ 5 \ 6; \ 7 \ 8 \ 9];$ $B = [3 \ 2 \ 1; \ 6 \ 5 \ 4; \ 9 \ 8 \ 7];$		Matrices A and B of size 3×3 .
$A * B$	$\begin{array}{ccc} ans = & 42 & 36 & 30 \\ & 96 & 81 & 66 \\ & 150 & 126 & 102 \end{array}$	The product $\mathbf{A} \cdot \mathbf{B}$.
$A . * B$	$\begin{array}{ccc} ans = & 3 & 4 & 3 \\ & 24 & 25 & 24 \\ & 63 & 64 & 63 \end{array}$	The element per element multiplication is implemented using the <i>dot operator</i> . The two results are different.

1.7.4.1 The Dot Product as a Special Case of Matrix Multiplication

The dot product was defined in Section 1.6.6. In this section, we present an alternative way of computing the dot product between two row vectors a and b based on matrix multiplication. Let N denote the number of elements of each vector. If row vector a is considered as matrix its size is $1 \times N$. Suppose now that vector b is converted into a column vector. Its size is now $N \times 1$. Thus, according to relationship (1.2) the dimensions of the product $d = a^*b'$ are given by

$$(1 \times N) \cdot (N \times 1) = (1 \times 1); \quad (1.4)$$

that is, their product is a scalar. Moreover, d is computed according to Equation 1.3 as

$$d_{ij} = \sum_{p=1}^N a_{ip} b_{pj}, \quad i = 1, \quad j = 1, \quad (1.5)$$

which is equivalent to Equation 1.1 that describes the dot product of vectors.

Example

Compute the dot product of the vectors $a = [1, 2, 3]$ and $b = [4, 5, 6]$. Verify your result with the MATLAB command `dot`.

Commands	Results	Comments
<code>a = 1:3;</code>	<code>ans = 4</code>	
<code>b = 4:6;</code>	<code>5</code>	Definition of the row vectors a and b . The transpose operator converts b into a column vector.
<code>b'</code>	<code>6</code>	
<code>size(a)</code>	<code>ans = 1</code>	3
<code>size(b')</code>	<code>ans = 3</code>	1
		The size of a is 1×3 , while the size of b' is 3×1 .
<code>d = a*b'</code>	<code>d = 32</code>	The dot product is computed as $a \cdot b'$. Notice that we do not use the dot operator before the multiplication operator as we implement a multiplication between matrices.
<code>dot(a, b)</code>	<code>ans = 32</code>	Verification of the result with use of the command <code>dot</code> .

1.7.5 Power of a Matrix

In order to raise a matrix A to a power, A must be a square matrix; that is, the number of rows must be the same as the number of columns. The power of a matrix is actually a multiplication of the matrix with itself as many times as the power is. To raise each element of a matrix to a power, the dot operator is inserted before the power operator.

Commands	Results			Comments
$A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$	A =	1	2	3
		4	5	6
		7	8	9
A^3	ans =	468	576	684
		1062	1305	1548
		1656	2034	2412
$A \cdot A \cdot A$	ans =	468	576	684
		1062	1305	1548
		1656	2034	2412
$A.^3$	ans =	1	8	27
		64	125	216
		343	512	729

1.7.6 Inverse of a Matrix

The inverse matrix of a matrix A is denoted by A^{-1} . If A^{-1} is multiplied with A , their product is the identity (or unit) matrix I , that is, a matrix with ones in the main diagonal and zeros elsewhere. The command that creates the inverse matrix of a matrix A is $\text{inv}(A)$. Alternatively, the inverse matrix is computed by typing A^{-1} .

Commands	Results			Comments
$A = [2 \ 4 \ 6; 1 \ 1 \ 1; 3 \ 4 \ 1]$	A =	2	4	6
		1	1	1
		3	4	1
$B = \text{inv}(A)$	B =	-0.3750	2.5000	-0.2500
		0.2500	-2.0000	0.5000
		0.1250	0.5000	-0.2500
$C = A^{-1}$	C =	-0.3750	2.5000	-0.2500
		0.2500	-2.0000	0.5000
		0.1250	0.5000	-0.2500
$A \cdot B$	ans =	1.0000	0.0000	0
		0.0000	1.0000	0
		0.0000	0.0000	1.0000
$B \cdot A$	ans =	1.0000	0.0000	0.0000
		0.0000	1.0000	0.0000
		0.0000	0	1.0000

1.7.7 Determinant of a Matrix

The inverse of a matrix A does not always exist. The necessary conditions for the existence of the inverse matrix are (a) A must be square and (b) A must be invertible. A matrix is invertible if its determinant is not zero. The determinant of a matrix is calculated by executing the command $\det(A)$.

Commands	Results	Comments
<code>A = [1 2 3; 4 5 6]</code>	$\begin{matrix} A = 1 & 2 & 3 \\ & 4 & 5 & 6 \end{matrix}$	Matrix \mathbf{A} of size 2×3 .
<code>inv(A)</code>	??? Error using ⇒ inv Matrix must be square.	\mathbf{A} is not square, hence the inverse matrix does not exist.
<code>A = [2 4 6; 1 1 1; 3 4 1]</code>	$\begin{matrix} A = 2 & 4 & 6 \\ 1 & 1 & 1 \\ 3 & 4 & 1 \end{matrix}$	Square matrix \mathbf{A} of size 3×3 .
<code>det(A)</code>	<code>ans = 8</code>	The determinant of \mathbf{A} is not zero; thus \mathbf{A} is invertible.
<code>inv(A)</code>	$\begin{matrix} ans = -0.375 & 2.500 & -0.250 \\ 0.250 & -2.000 & 0.500 \\ 0.125 & 0.500 & -0.250 \end{matrix}$	The inverse matrix \mathbf{A}^{-1} .
<code>B = [1 1 1; 2 2 2; 3 3 3]</code>	$\begin{matrix} B = 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{matrix}$	Square matrix \mathbf{B} of size 3×3 .
<code>det(B)</code>	<code>ans = 0</code>	The determinant of \mathbf{B} is zero; thus \mathbf{B} is <i>not</i> invertible.
<code>inv(B)</code>	Warning: Matrix is singular to working precision. (Type "warning off MATLAB:singularMatrix" to suppress this warning.) $\begin{matrix} ans = Inf & Inf & Inf \\ Inf & Inf & Inf \\ Inf & Inf & Inf \end{matrix}$	Indeed the inverse matrix \mathbf{B}^{-1} cannot be computed.

1.7.8 Division of Matrices

The division between matrices is defined in MATLAB as the product of the first matrix with the inverse of the second matrix if left division is applied, and vice versa if right division is applied. In other words, $\mathbf{A}/\mathbf{B} = \mathbf{A} \cdot \mathbf{B}^{-1}$ and $\mathbf{A}\backslash\mathbf{B} = \mathbf{A}^{-1} \cdot \mathbf{B}$. As usual, if the dot operator is employed the division is implemented as an element per element operation.

Commands	Results	Comments
$A = [2 \ 4 \ 6; 1 \ 1 \ 1; 3 \ 4 \ 1];$ $B = [3 \ 2 \ 2; 1 \ 4 \ 1; 3 \ 2 \ 1];$		Invertible square matrices A and B.
A/B	$\begin{matrix} \text{ans} = 4.8000 & 0.8000 & -4.4000 \\ 0.6000 & 0.1000 & -0.3000 \\ -0.4000 & 0.6000 & 1.2000 \end{matrix}$	Left division.
$A*inv(B)$	$\begin{matrix} \text{ans} = 4.8000 & 0.8000 & -4.4000 \\ 0.6000 & 0.1000 & -0.3000 \\ -0.4000 & 0.6000 & 1.2000 \end{matrix}$	Equivalent operation to left division.
$A\backslash B$	$\begin{matrix} \text{ans} = 0.6250 & 8.7500 & 1.5000 \\ 0.2500 & -6.5000 & -1.0000 \\ 0.1250 & 1.7500 & 0.5000 \end{matrix}$	Right division.
$inv(A)*B$	$\begin{matrix} \text{ans} = 0.6250 & 8.7500 & 1.5000 \\ 0.2500 & -6.5000 & -1.0000 \\ 0.1250 & 1.7500 & 0.5000 \end{matrix}$	Equivalent operation to right division.
$C = A./B$	$\begin{matrix} C = 0.6667 & 2.0000 & 3.0000 \\ 1.0000 & 0.2500 & 1.0000 \\ 1.0000 & 2.0000 & 1.0000 \end{matrix}$	Element per element division, that is, $c_{ij} = a_{ij}/b_{ij}$.

1.7.9 Transpose of a Matrix

The transpose of a matrix \mathbf{A} (usually denoted by \mathbf{A}^T) is a matrix whose rows are the columns of \mathbf{A} and its columns are the rows of \mathbf{A} . Let \mathbf{B} denote the transpose of \mathbf{A} . The elements of \mathbf{B} represented by b_{ij} are given by $b_{ij} = a_{ji}$. The transposed matrix \mathbf{A}^T is derived by applying the transpose operator $\ll'\gg$ after matrix \mathbf{A} , i.e., by typing $\mathbf{B}=\mathbf{A}'$.

Commands	Results	Comments
$A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9];$	$\begin{matrix} \text{A} = 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$	Matrix \mathbf{A} of size 3×3 .
$B = A'$	$\begin{matrix} \text{B} = 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{matrix}$	The transposed matrix \mathbf{A}^T .
B'	$\begin{matrix} \text{ans} = 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$	Transposing \mathbf{A}^T results the initial matrix \mathbf{A} .
$A = [1 \ 2 \ 3; 4 \ 5 \ 6]$	$\begin{matrix} \text{A} = 1 & 2 & 3 \\ 4 & 5 & 6 \end{matrix}$	Matrix \mathbf{A} of size 2×3 .
$B = A'$	$\begin{matrix} \text{B} = 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{matrix}$	The transpose of \mathbf{A} is of size 3×2 .

Remark

Recall that the transpose operator $\ll'\gg$ was employed to convert a row vector into a column vector and vice versa.

Remark

The transpose of a matrix must not be confused with the inversion of a matrix.

1.7.10 Special Forms of Matrices

`ones-zeros-rand-eye-magic-hilb`

The command `ones(M,N)` creates a matrix of size $M \times N$ with ones. Typing `ones(M)` returns a square matrix with ones. Similarly, the command `zeros(M,N)` creates a matrix of size $M \times N$ with zeros. The command `rand(M,N)` returns an $M \times N$ matrix with random (see Section 1.17 for details) elements. The command `eye(M,N)` defines an $M \times N$ matrix with ones in the main diagonal and zeros elsewhere. The command `magic(M)` creates a square matrix with the property that the sum of each row or column is equal. Finally, the command `hilb(N)` returns an $N \times N$ Hilbert matrix.

Commands	Results	Comments
<code>ones(2,3)</code>	<code>ans = 1 1 1 1 1 1</code>	Matrix of size 2×3 with ones.
<code>zeros(1,4)</code>	<code>ans = 0 0 0 0</code>	Matrix of size 1×4 (or vector of length 4) with zeros.
<code>rand(3)</code>	<code>ans = 0.9501 0.4860 0.4565 0.2311 0.8913 0.0185 0.6068 0.7621 0.8214</code>	Matrix of size 3×3 with random elements. If there is one input argument to the command, the obtained matrix is square.
<code>eye(4,2)</code>	<code>ans = 1 0 0 1 0 0 0 0</code>	Matrix of size 4×2 with ones in the main diagonal and zeros elsewhere.
<code>eye(3)</code>	<code>ans = 1 0 0 0 1 0 0 0 1</code>	The identity matrix \mathbf{I} of size 3×3 .
<code>A=magic(3)</code>	<code>A = 8 1 6 3 5 7 4 9 2</code>	Magic matrix.
<code>hilb(3)</code>	<code>ans = 1.0000 0.5000 0.3333 0.5000 0.3333 0.2500 0.3333 0.2500 0.2000</code>	Hilbert matrix.

1.7.11 Useful Commands

`diag-triu-tril-toeplitz-hankel-find-sum-rank-norm-eig`

In this section, we introduce some other commands that are applicable to matrices.

Commands	Results	Comments
<code>A = [1 2 3; 4 5 6; 7 8 9]</code>	$\begin{matrix} A = 1 & 2 & 3 \\ & 4 & 5 & 6 \\ & 7 & 8 & 9 \end{matrix}$	Matrix A of size 3×3 .
<code>diag(A)</code>	$\begin{matrix} ans = 1 \\ & 5 \\ & 9 \end{matrix}$	Command <code>diag</code> returns a column vector with the elements of the main diagonal.
<code>B=diag([-1 2 3])</code>	$\begin{matrix} B = -1 & 0 & 0 \\ & 0 & 2 & 0 \\ & 0 & 0 & 3 \end{matrix}$	Command <code>diag</code> can be also used to create a diagonal matrix. The elements of the main diagonal are the ones given as input to the command, while all other matrix elements are zero.
<code>triu(A)</code>	$\begin{matrix} ans = 1 & 2 & 3 \\ & 0 & 5 & 6 \\ & 0 & 0 & 9 \end{matrix}$	Conversion to an upper triangular matrix.
<code>tril(A)</code>	$\begin{matrix} ans = 1 & 0 & 0 \\ & 4 & 5 & 0 \\ & 7 & 8 & 9 \end{matrix}$	Conversion to a lower triangular matrix.
<code>toeplitz(A)</code>	$\begin{matrix} ans = 1 & 4 & 7 & 2 & 5 & 8 & 3 & 6 & 9 \\ & 4 & 1 & 4 & 7 & 2 & 5 & 8 & 3 & 6 \\ & 7 & 4 & 1 & 4 & 7 & 2 & 5 & 8 & 3 \end{matrix}$	Conversion to a Toeplitz matrix.
<code>hankel(A)</code>	$\begin{matrix} ans = 1 & 4 & 7 & 2 & 5 & 8 & 3 & 6 & 9 \\ & 4 & 7 & 2 & 5 & 8 & 3 & 6 & 9 & 0 \\ & 7 & 2 & 5 & 8 & 3 & 6 & 9 & 0 & 0 \\ & 2 & 5 & 8 & 3 & 6 & 9 & 0 & 0 & 0 \\ & 5 & 8 & 3 & 6 & 9 & 0 & 0 & 0 & 0 \\ & 8 & 3 & 6 & 9 & 0 & 0 & 0 & 0 & 0 \\ & 3 & 6 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 6 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$	Conversion to a Hankel matrix.
<code>find(B)</code>	$\begin{matrix} ans = 1 \\ & 5 \\ & 9 \end{matrix}$	The command <code>find</code> returns the indices of the nonzero elements of B . The matrix is considered vector, hence the indices are scalars.
<code>find(B>2)</code>	$\begin{matrix} ans = 9 \end{matrix}$	Returns the indices of elements that are larger than 2.
<code>sum(A)</code>	$\begin{matrix} ans = 12 & 15 & 18 \end{matrix}$	The sum of each column of A .
<code>sum(A, 2)</code>	$\begin{matrix} ans = 6 \\ & 15 \\ & 24 \end{matrix}$	The sum of each row of A .
<code>sum(sum(A))</code>	$\begin{matrix} ans = 45 \end{matrix}$	The sum of all elements of A .
<code>rank(B)</code>	$\begin{matrix} ans = 3 \end{matrix}$	The rank of a matrix is the number of linearly independent rows or columns.
<code>norm(B)</code>	$\begin{matrix} ans = 3 \end{matrix}$	The norm of B .
<code>eig(B)</code>	$\begin{matrix} ans = -1 \\ & 2 \\ & 3 \end{matrix}$	The eigenvalues of B . With the proper syntax the command <code>eig</code> returns also the eigenvectors of a matrix.

1.8 Plotting with MATLAB

MATLAB is a very reliable and powerful tool for plotting. A graph is constructed as a set of points in two or three dimensions. These points are (typically) connected with a solid line. Commonly the graph of a function is desired. However, in MATLAB a plot is done using *vectors* or *matrices* and not functions.

1.8.1 Plotting in Two Dimensions

Suppose that we want to plot a function $y(x)$, where x is the independent variable. The procedure to plot $y(x)$ is as follows: First the vector x is created, such as $a \leq x \leq b$, where a, b are scalars. The function $y(x)$ will be plotted over the interval $[a, b]$. Next, we create the vector y , which is of the same length as x ; that is, the two vectors have an equal number of elements. The value of each element of y is the value of $y(x)$ calculated for each element of x . Finally, the function $y(x)$ is plotted by typing `plot(x, y)`.

Example

Plot the function $y(x) = x^2$, $-2 \leq x \leq 2$.

Commands	Results	Comments
<code>x = -2:2</code>	<code>x = -2 -1 0 1 2</code>	Vector x is defined from -2 to 2 with step 1 .
<code>length(x)</code>	<code>ans = 5.00</code>	Vector x has 5 elements.
<code>y = x.^2</code>	<code>y = 4 1 0 1 4</code>	Vector y is created. Notice that $y = [(-2)^2 (-1)^2 0^2 1^2 2^2]$. Also notice the dot operator which is inserted before the power operator to ensure that this is an element per element operation.
<code>length(y)</code>	<code>ans = 5.00</code>	Vector y has the same number of elements to x .
<code>plot(x, y)</code>		Graph of the function $y(x) = x^2$, $-2 \leq x \leq 2$.

The obtained graph of $y(x) = x^2$, $-2 \leq x \leq 2$ is not exactly what was expected. This is due to the way that the command `plot` works. As mentioned before, the graph is done at the points $[x(1), y(1)]$, $[x(2), y(2)]$, ... and next these points are connected with solid line. More specifically, in this graph the points with coordinates $[-2, 4], [-1, 1], [0, 0], [1, 1], [2, 4]$

are connected. In order to achieve a better graph, more points in the vector x and consequently in y are required. Therefore, by using a smaller step in the definition of x a better graph of $y(x)$ is obtained.

Commands	Results					Comments
$x = -2.00$	-1.90	-1.80	-1.70	-1.60		
-1.50	-1.40	-1.30	-1.20	-1.10		
-1.00	-0.90	-0.80	-0.70	-0.60		
-0.50	-0.40	-0.30	-0.20	-0.10		The vector x is defined from -2 to 2 with step 0.1.
$x = -2:0.1:2$	0	0.10	0.20	0.30	0.40	0.50
	0.70	0.80	0.90	1.00	1.10	1.20
	1.30	1.40	1.50	1.60	1.70	1.80
	1.90	2.00				
$plot(x, y)$??? Error using ⇒ plot Vectors must be the same lengths.					Trying to execute the command <code>plot</code> we get an error warning as <code>length(x) = 41</code> and <code>length(y) = 5</code> .
$y = x.^2;$						Vector y has to be redefined according to the new vector x .
$length(y)$	ans = 41.00					Now the two vectors have the same number of elements and the function $y(x) = x^2$, $-2 \leq x \leq 2$ can be plotted.
$plot(x, y)$						This time the graph is satisfactory.
$plot(y)$						It is possible to execute the command <code>plot</code> with only one input argument. The result appears to be correct. However, the x -axis is not the interval $[-2, 2]$ but $[1, 41]$. More specifically, each element of y (recall that y consists of 41 elements) is plotted versus its index in the vector y and not versus the corresponding element of x .

1.8.2 The Fig File

The window in which a graph is displayed is known as *figure*. Figures are files with extension *.fig*. A figure can be saved outside MATLAB in the most common formats (jpg, eps, etc.). The file save dialog is depicted in Figure 1.2.

1.8.3 The Command `linspace`

A vector x is defined by specifying its initial value, the step, and its final value. The problem with this approach is that it is not always easy to calculate the number of elements that the vector consists of. For example, it is quite hard to estimate the number of elements of the vector defined by the statement $x = [-2*pi:0.3:2*pi]$. The command `linspace` provides an alternative way of creating vectors. The advantage is that the number of vector elements is defined through the command. The syntax is $x = \text{linspace}(\text{initial_value}, \text{final_value}, \text{number_of_elements})$. The produced points, that is, the vector elements are equally spaced. The command `logspace` has the same use and syntax with `linspace`, but the points are logarithmically spaced. Moreover, the initial and final values are considered as powers of 10.

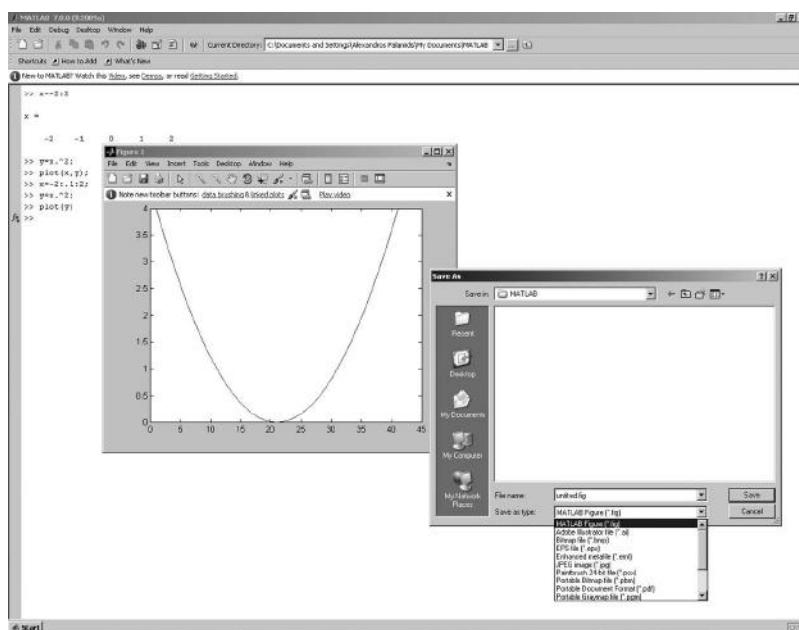


FIGURE 1.2

A figure can be exported from MATLAB in the most common formats.

Commands	Results						Comments
<code>x = linspace(0,1,5)</code>	<code>ans = 0</code>	0.25	0.50	0.75	1.00		Vector x from 0 to 1 with 5 elements.
<code>x = 0:1/4:1</code>	<code>ans = 0</code>	0.25	0.50	0.75	1.00		Equivalent vector creation.
<code>logspace(0,1,5)</code>	<code>ans = 1.00</code>	1.78	3.16	5.62	10.00		The command <code>logspace</code> creates 5 logarithmically spaced points on the interval $[10^0 \ 10^1]$.

1.8.4 Plotting Several Functions in One Figure

It is possible to plot more than one function in the same figure by employing a different syntax of the command `plot`.

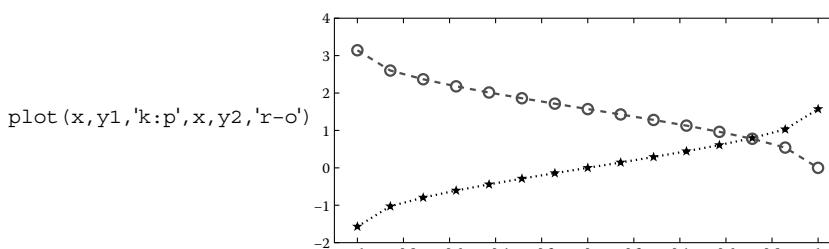
Example

Plot the functions $y(x) = x^2 \cos(x)$, $g(x) = x \cos(x)$, and $f(x) = 2^x \sin(x)$, $0 \leq x \leq 2\pi$, in the same figure.

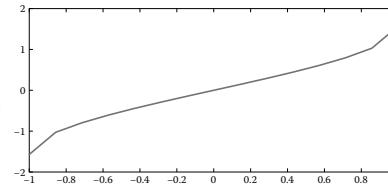
Commands	Results	Comments
<code>x = linspace(0,2*pi,100);</code>		Definition of vector x .
<code>y = (x.^2).*cos(x);</code> <code>g = x.*cos(x);</code> <code>f = (2.^x).*sin(x);</code>		Definition of vectors y , g , and f .
<code>plot(x,y,x,g,x,f)</code>		Using this syntax of the command <code>plot</code> , it is possible to plot many functions in the same figure. Each function is plotted in different color.

In the previous examples, the functions were plotted with predefined colors and line type (solid). It is possible to create a graph using colors, symbols used to draw the points, and type of line that connects the points of your choice. This is achieved by applying one more input argument to the command `plot`. The new argument is a series of special characters given in single quotes. The available special characters are presented in the table below.

Symbol	Color	Symbol	Point Type	Symbol	Line Type
b	Blue	.	Point	-	Solid
g	Green	o	Circle	:	Dotted
r	Red	x	x-mark	-.	Dashdot
c	Cyan	+	Plus	--	Dashed
m	Magenta	*	Star		
y	Yellow	s	Square		
k	Black	d	Diamond		
w	White	<, >	Triangle		
		p	Pentagram		
		h	Hexagram		

Commands	Results	Comments
<pre>x=linspace(-1,1,15); y1=asin(x); y2=acos(x);</pre> 		<p>Definition of vector x. Definition of vectors y_1 and y_2.</p>

Another available syntax of the command `plot` that specifies the color and the line width is presented below.

Commands	Results	Comments
<pre>plot(x,y1,'Color',[1 0 1], 'LineWidth',4)</pre> 		<p>The vector that appears after the text "Color" specifies the RGB elements that take values on $[0 \ 1]$. Number 4 determines the line width.</p>

If one plots a new graph, by executing the command `plot` the old graph is discarded. A command that can be used in order to hold the current plot and draw the next one in the same figure is the command `hold`. This command can be used as a switch between the two modes (preserve graph/discard graph). More easily, one can type `hold on` to apply the preserve graph state and `hold off` to return to the default mode.

Commands	Results	Comments
<code>x=linspace(0, 2*pi, 150); plot(x, cos(x));</code>		Definition of vector x and direct plot of $\cos(x)$.
<code>hold on</code>		The old graph is preserved.
<code>plot(x, sin(x));</code>		The new function is plotted together with the old one.
<code>hold off</code>		Switch to the non-hold mode.
<code>plot(x, tan(x))</code>		The old graph is discarded and the new function is plotted alone.

1.8.5 Formatting a Figure

`grid-title-xlabel-ylabel-legend-text-gtext-axis-xlim-ylim`

The command `grid on` adds grid lines to the graph, while the command `grid off` removes the grid lines. Simply typing `grid` is a switch between the two modes. Text beside the x -axis and y -axis can be added using the commands `xlabel` and `ylabel`, respectively. A graph title is inserted by the command `title`.

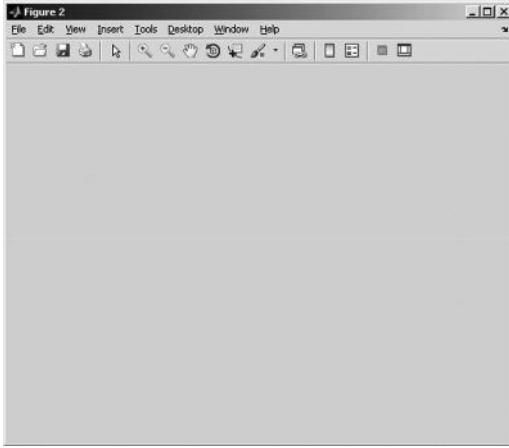
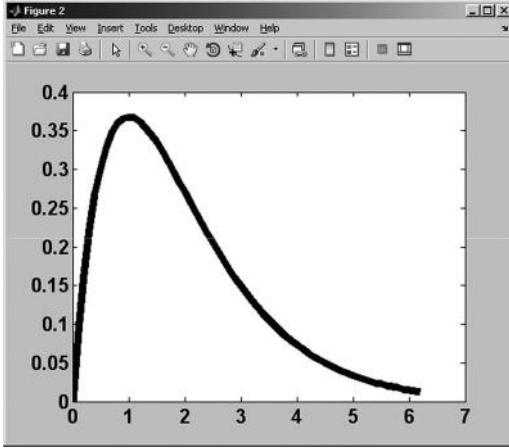
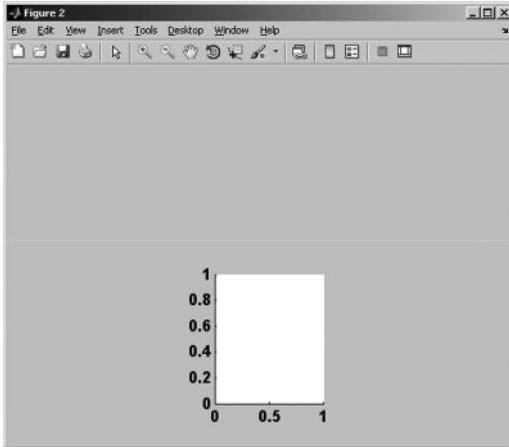
Commands	Results	Comments
<pre>x=linspace(0,2*pi,150); plot(x,cos(x),'r*',x,sin(x), 'k')</pre>		Definition of the vector x and plot of $\cos(x)$ and $\sin(x)$.
<pre>grid</pre>		Insertion of grid lines.
<pre>xlabel('x-axis') ylabel('y-axis') title('Graph of cos(x) and sin(x)')</pre>	<p style="text-align: center;">Graph of $\cos(x)$ and $\sin(x)$</p>	Labels in the axes and graph title.

Text can be added at a specific point in the plot by typing the command `text(x,y,'string')`, where (x,y) are the coordinates of the point where the string written in the quotes is displayed. A simpler way to insert text inside the graph is to use the command `gtext('string')`, where the text is inserted at a position indicated with the mouse. The command `legend('string1', 'string2', ...)` puts a legend on the current plot using the specified strings as labels. The legend is erased by typing `legend off`. Finally, the command `axis([x_min, x_max, y_min, y_max])` specifies the limits for the two axis. If only the limits of the x -axis need to be changed the command `xlim([x_min, x_max])` is appropriate. Similarly, executing the command `ylim([y_min, y_max])` sets the limits of y -axis.

Commands	Results	Comments
<pre>text(3,0.3,'string1') gtext('string2') legend('cos(x)','sin(x)')</pre>		The text “string1” is placed at the point with coordinates [3 0.3], while the text “string2” is placed with the mouse. The legend command clarifies which function corresponds to which curve. The legend is by default placed at the upper left part of the graph.
<pre>axis([2,4,-1,0])</pre>		Change of limits. The x-axis is at [2 4] and the y-axis is at [-1 0].

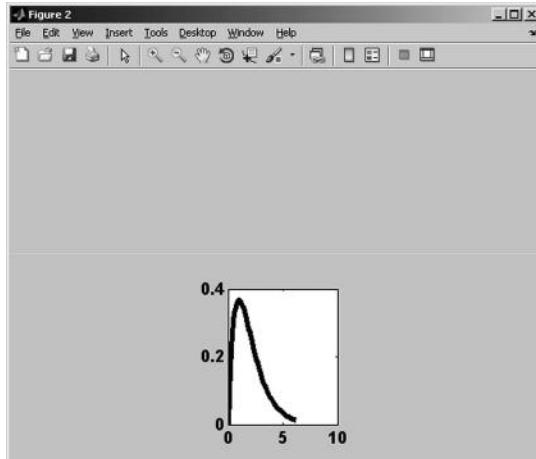
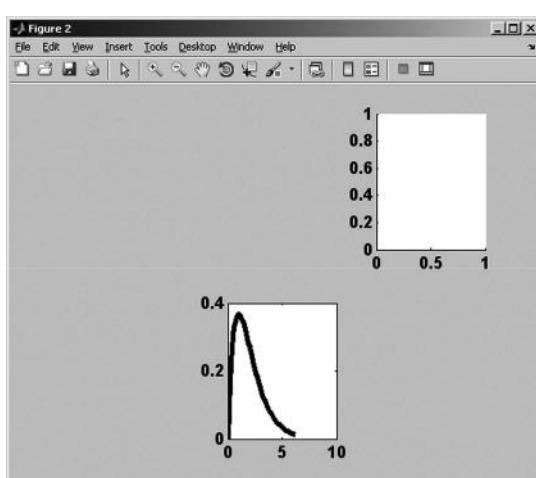
1.8.6 Plotting in Different Figures

Up to this point, all plots were made in a single figure named *Figure 1*. By typing at the command prompt `figure`, a new figure with name *Figure 2* appears without closing the old figure. Typing `figure(k)` creates a new figure with name *Figure k*. This is the active figure where the next graph will appear. The command `subplot(m, n, p)` or `subplot(mnp)` splits the figure window into $m \times n$ small subfigures and makes the p th subfigure active.

Commands	Results	Comments
figure		A new figure with name <i>Figure 2</i> is created. The old figure is preserved.
<pre>x=0:.1:2*pi; plot(x,x.*exp(-x));</pre>		The window <i>Figure 2</i> is the active one, i.e., the next graph is plotted in it.
subplot(2,3,5)		The figure is split in $2 \times 3 = 6$ subfigures. The active subfigure is the fifth.

(continued)

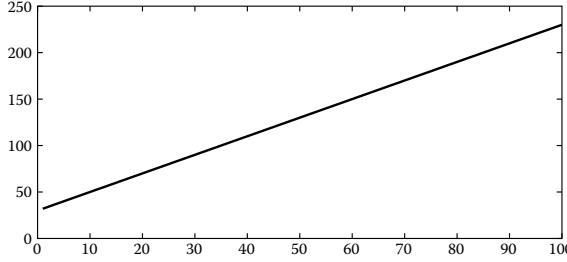
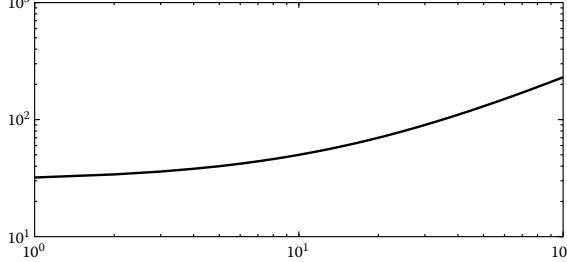
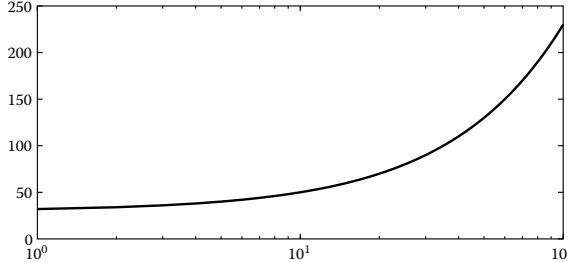
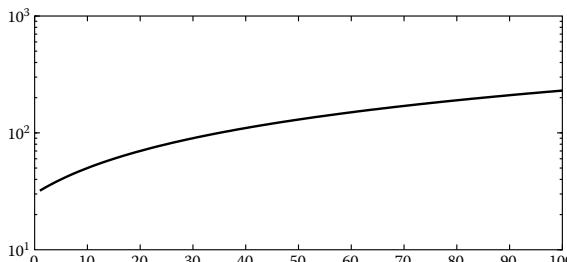
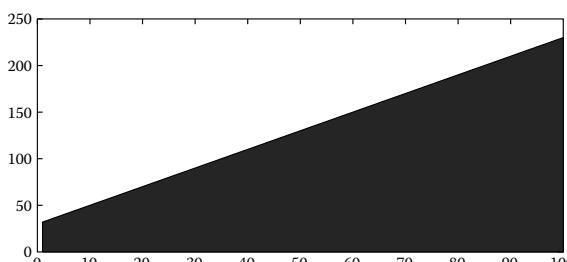
(continued)

Commands	Results	Comments
<code>plot(x, x.*exp(-x))</code>		Plotting in the active subfigure.
<code>subplot(2, 3, 3)</code>		Activation of the third subfigure.

1.8.7 Commands for Plotting

`loglog-semilogx-semilogy-area-fplot-ezplot`

Besides `plot` there are also some other commands that can be employed for plotting a function. Suppose that x and y are the two available vectors. The command `loglog(x, y)` plots y versus x with both axis being logarithmically scaled. Typing `semilogx(x, y)` only the x -axis is in logarithmic scale, while typing `semilogy(x, y)` only the y -axis is in logarithmic scale. The command `area(x, y)` produces a stacked area plot. The command `fplot` provides an easy way of plotting a function without the need to specify vectors. Its syntax is `fplot('fun', [x_min, x_max])`, where fun is the function that is plotted in the interval $[x_{min}, x_{max}]$. A related command with a similar syntax is the command `ezplot`. Additionally, `ezplot` is useful when plotting symbolic expressions. Symbolic expressions are discussed in Section 1.15.

Commands	Results	Comments
<pre>x = 1:100; y = 2*x+30; plot(x,y)</pre>	 A linear plot of the function $y(x) = 2x + 30$ for $x \in [0, 100]$. The x-axis ranges from 0 to 100 with major ticks every 10 units. The y-axis ranges from 0 to 250 with major ticks every 50 units. A straight line starts at approximately (0, 30) and ends at approximately (100, 230).	Graph of the function $y(x) = 2x + 30$, $1 \leq x \leq 100$.
<pre>loglog(x,y)</pre>	 A log-log plot of the same function $y(x) = 2x + 30$. Both axes are logarithmic, ranging from 10^0 to 10^3 . The curve appears as a straight line with a positive slope.	Graph of $y(x)$ in logarithmic scale axes.
<pre>semilogx(x,y)</pre>	 A semi-log plot where the x-axis is logarithmic and the y-axis is linear. The x-axis ranges from 10^0 to 10^2 with major ticks at $10^0, 10^1, 10^2$. The y-axis ranges from 0 to 250 with major ticks every 50 units. The curve is exponential-like, starting at approximately (1, 30) and increasing rapidly as x increases.	Graph of $y(x)$ with the x -axis being logarithmically scaled.
<pre>semilogy(x,y)</pre>	 A semi-log plot where the y-axis is logarithmic and the x-axis is linear. The y-axis ranges from 10^1 to 10^3 with major ticks at $10^1, 10^2, 10^3$. The x-axis ranges from 0 to 100 with major ticks every 10 units. The curve is exponential-like, starting at approximately (0, 30) and increasing slowly as x increases.	Graph of $y(x)$ with the y -axis being logarithmically scaled.
<pre>area(x,y)</pre>	 A stacked area plot of the function $y(x) = 2x + 30$ over the interval $[0, 100]$. The area under the curve is filled with a solid black color.	A stacked area plot of $y(x)$.

(continued)

(continued)

Commands	Results	Comments
<pre>fplot('2*x+30', [1,100]) % or ezplot('2*x+30', [1,100])</pre>		Fast ways of plotting $y(x)$. By using <code>fplot</code> or <code>ezplot</code> there is no need to create vectors.

1.8.8 Plotting Discrete-Time Functions

The command `stem`

A discrete time function is a function of the form $f[n], n \in \mathbb{Z}$, where \mathbb{Z} denotes the set of integer numbers. In this case, the appropriate command for plotting a function $f[n]$ is the command `stem(n, f)`. The command `stem` plots the data sequence given in vector f versus vector n . The graph consists of lines from the x -axis terminated with circles on the data value.

Commands	Results	Comments
<pre>n = -3:3 f = n.^2</pre>	<pre>n = -3 -2 -1 0 1 2 3 f = 9 4 1 0 1 4 9</pre>	The step used in vector n is 1. The function $f[n] = n^2$ is defined.
<code>stem(n, f)</code>		Graph of $f[n] = n^2$, $-3 \leq n \leq 3$ with use of the command <code>stem</code> .
<code>stem(n, f, 'r*')</code>		This syntax of <code>stem</code> creates a graph with red stars.

1.8.9 Graph in Polar Coordinates

In MATLAB it is possible to create a plot using polar coordinates. The command is `polar(theta,r)`, where *theta* is the vector of angles given in radians and *r* is the vector of distances from the axis origin.

Commands	Results	Comments
<pre> theta = 0 1.3963 2.7925 theta=linspace(0,4*pi,10) r=theta r = 0 1.3963 2.7925 4.1888 5.5851 6.9813 8.3776 9.7738 11.1701 12.5664 r = 0 1.3963 2.7925 4.1888 5.5851 6.9813 8.3776 9.7738 11.1701 12.5664 polar(theta,r,'-*') </pre>	<pre> theta = 0 1.3963 2.7925 theta=linspace(0,4*pi,10) r=theta r = 0 1.3963 2.7925 4.1888 5.5851 6.9813 8.3776 9.7738 11.1701 12.5664 r = 0 1.3963 2.7925 4.1888 5.5851 6.9813 8.3776 9.7738 11.1701 12.5664 polar(theta,r,'-*') </pre>	The function $r = \theta$ is plotted, where <i>r</i> is the radius and θ defines the angles in radians.

1.8.10 Piecewise Functions

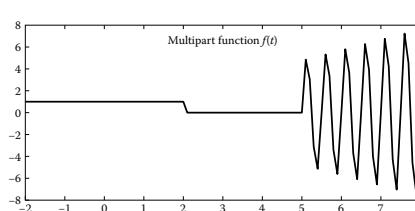
In this section, we discuss the way of defining and plotting functions with more than one part.

Example

Plot the function

$$f(t) = \begin{cases} 1, & -2 \leq t \leq 2 \\ 0, & 2 < t < 5 \\ t \sin(4\pi t), & 5 \leq t \leq 8 \end{cases}.$$

First, $f(t)$ is defined separately at each different time interval, i.e., three different functions $f_1(t)$, $f_2(t)$, and $f_3(t)$ are defined over the corresponding time intervals $[-2, 2]$, $(2, 5)$, and $[5, 8]$. Then, the various time intervals and the corresponding functions are concatenated (see Section 1.7.1 for details) in order to form the vector of time t and the vector of function $f(t)$. Finally, the piecewise function $f(t)$ is plotted as usual.

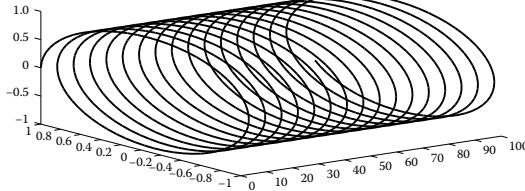
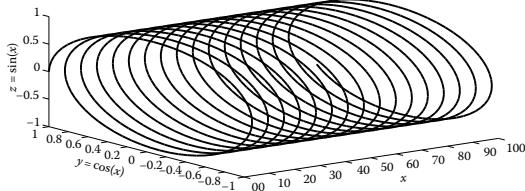
Commands	Results	Comments
$t1 = -2:.1:2;$ $t2 = 2.1:.1:4.9;$ $t3 = 5:.1:8;$		Three different (non-overlapping) time intervals are defined. However, the intervals could overlap without problem in the points of discontinuity of $f(t)$ as time is continuous.
$f1 = ones(size(t1));$ $f2 = zeros(size(t2));$ $f3 = t3.*sin(4*pi*t3);$		The first branch of $f(t)$ is defined over the corresponding time interval. The created vector $f1$ is a vector of ones with same size to $t1$. A common mistake that has to be avoided is to write $f1 = 1$. In this case, the two vectors are not of the same size and the plot cannot be made.
$t = [t1 t2 t3];$ $f = [f1 f2 f3];$		Similarly, the second branch of $f(t)$ is a vector of zeros with the same size to $t2$.
$plot(t,f)$ $title('Multi-part function f(t)')$		The third branch of $f(t)$ is defined over $t3$. The three time intervals are concatenated to form a time vector from -2 to 8 .
		Similarly, the three branches of $f(t)$ are concatenated to form the multipart function.
		The graph of $f(t)$ is completed without problem.

1.8.11 Plotting in Three Dimensions

In this section, we introduce the way of plotting curves or surfaces in the three-dimensional (3-D) space.

1.8.11.1 Plotting Curves in Three Dimensions

The command `plot3` is employed in order to plot a curve in the 3-D space. This command is a 3-D counterpart of the command `plot`. The syntax is `plot3(x, y, z)`, where x , y , and z are vectors of the same length. The plotted line connects the points whose coordinates are the elements of x , y , and z . The commands used to format a 2-D graph are also applicable to the 3-D space if the proper syntax is used. For example, the command `text` has four input arguments (three coordinates and the string). To insert a label in the z -axis the command `zlabel` is employed. Commands like `grid`, `box`, `subplot`, etc. have the same functionality and syntax.

Commands	Results	Comments
<pre>x = 0 : .1 : 100; y = cos(x); z = sin(x);</pre> <pre>plot3(x, y, z)</pre>		Definition of the function $f(x) = (x, \cos(x), \sin(x))$, $0 \leq x \leq 100$. Graph of $f(x, y, z)$ in the 3-D space.
<pre>xlabel('x') ylabel('y = cos(x)') zlabel('z = sin(x)')</pre>		Labels in the three axes.

1.8.11.2 Plotting Surfaces in Three Dimensions

In case we want to plot a surface, the procedure that must be followed is slightly harder. A surface is specified by a function of two independent variables of the form $z = f(x, y)$, $x_1 \leq x \leq x_2$, $y_1 \leq y \leq y_2$. The procedure followed is:

- The vectors x and y are defined in the proper limits.
- With the command `meshgrid` the vectors x and y are converted in two matrices \mathbf{X} and \mathbf{Y} . The rows of \mathbf{X} are copies of vector x and the columns of \mathbf{Y} are copies of vector y .
- A new matrix \mathbf{Z} is determined in terms of the matrices \mathbf{X} and \mathbf{Y} according to the function $f(x, y)$; that is, we define $\mathbf{Z} = f(\mathbf{X}, \mathbf{Y})$.
- The 3-dimensional graph, i.e., the surface is obtained with the command `mesh(X, Y, Z)` or with the similar command `surf(X, Y, Z)`.

Example

Plot the surface that is described by the function $z = f(x, y) = x + y, 1 \leq x \leq 5, 1 \leq y \leq 5$. Moreover, for comparison reasons plot the curve defined by the same function.

Commands	Results	Comments
<code>x=1:5;</code>	<code>x = 1 2 3 4 5</code>	First step: Definition of vectors x and y .
<code>y=1:5;</code>	<code>y = 1 2 3 4 5</code>	
	<code>X = 1 2 3 4 5</code>	
	<code>1 2 3 4 5</code>	
	<code>1 2 3 4 5</code>	
	<code>1 2 3 4 5</code>	
	<code>1 2 3 4 5</code>	
<code>[X, Y] = meshgrid(x, y)</code>	<code>Y = 1 1 1 1 1</code>	Second step: Conversion of the vectors x and y in matrices \mathbf{X} and \mathbf{Y} with use of the command <code>meshgrid</code> .
	<code>2 2 2 2 2</code>	
	<code>3 3 3 3 3</code>	
	<code>4 4 4 4 4</code>	
	<code>5 5 5 5 5</code>	
	<code>Z = 2 3 4 5 6</code>	
	<code>3 4 5 6 7</code>	
<code>Z=X+Y</code>	<code>4 5 6 7 8</code>	Matrix \mathbf{Z} is defined according to the relationship $\mathbf{Z} = f(\mathbf{X}, \mathbf{Y})$.
	<code>5 6 7 8 9</code>	
	<code>6 7 8 9 10</code>	
<code>mesh(X, Y, Z)</code>		The surface graph is obtained by the command <code>mesh</code> .
<code>surf(X, Y, Z)</code>		The command <code>surf</code> provides a surface with colors.
<code>z=x+y</code> <code>plot3(x, y, z)</code>		The curve is plotted for comparison reasons. Notice that in order to plot a curve we use vectors, but a surface is plotted with use of matrices.

Finally, we mention some other commands that are useful for plotting in the 3-D space. The command `waterfall` is the same as `mesh` except that the column lines of the mesh are not drawn. The command `bar3` creates a 3-D bar graph, while using the command `sphere` a unit sphere can be created or just be plotted in the 3-D space.

1.9 Complex Numbers

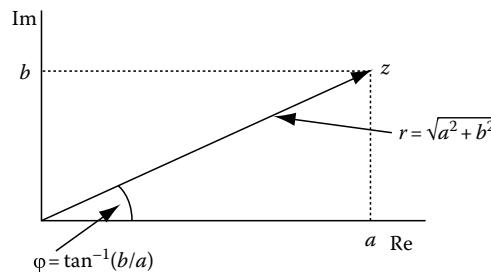
Up to this point only real numbers were considered. In MATLAB it is possible to work with complex numbers. A complex number is of the form $z = a + bi$ or $z = a + bj$, where i and j denote the imaginary unit which is given by $i = j = \sqrt{-1}$. The letters i and j are constants and recognized from MATLAB as the imaginary unit. The *real part* of z is a , while the *imaginary part* of z is b .

Commands	Results	Comments
<code>sqrt(-1)</code>	<code>ans = 0 + 1.0000i</code>	The imaginary unit i is given by $i = \sqrt{-1}$.
<code>z=3+2*i</code>	<code>z = 3.0000 + 2.0000i</code>	The complex number $3+2i$.
<code>z=3+2*j</code>	<code>z = 3.0000 + 2.0000i</code>	Instead of i equivalently one may use j .
<code>z=3+2i</code>	<code>z = 3.0000 + 2.0000i</code>	The multiplication operator can be omitted when a complex number is defined.
<code>z=3+2j</code>	<code>z = 3.0000 + 2.0000i</code>	Also, if j is used in the complex number the multiplication operator can be omitted.

1.9.1 Useful Commands

`real`-`imag`-`abs`-`angle`-`phase`-`conj`

The real part of a complex number $z = a + bi$ is easily found by typing `a = real(z)`, while the command `b = imag(z)` returns the imaginary part of z . The *magnitude* (or absolute value or modulus) of z is given by $|z| = \sqrt{a^2 + b^2}$. The command used to compute the magnitude of a complex number is `abs(z)`. The magnitude represents the distance from the axis origin. Finally, the phase (or angle or argument) of a complex number is computed by the commands `angle(z)` or `phase(z)`. The result is in radians. The angle φ is given by $\varphi = \tan^{-1}(b/a)$. The command `conj(z)` returns the complex conjugate of z . The complex conjugate of $z = a + bi$ is usually denoted by z^* and is given by $z^* = a - bi$. An alternative way to compute the complex conjugate of z is to use the transpose operator, i.e., to type z' . In the following graph (Figure 1.3), a complex number $z = a + bi$ is plotted in the complex plane. The horizontal axis of a complex plane corresponds to the real part of z , while the vertical axis represents the imaginary part of z .

**FIGURE 1.3**

Graph of a complex number $z = a + bi$ in the complex plane. The symbol Re denotes the real axis, while symbol Im denotes the imaginary axis. The magnitude is indicated by r while the angle is denoted by φ .

Commands	Results	Comments
<code>a = 4;</code> <code>b = 3;</code> <code>z = a+b*i</code>	<code>z = 4.0000 + 3.0000i</code>	Complex number $4 + 3i$.
<code>real(z)</code>	<code>ans = 4</code>	Real part of z .
<code>imag(z)</code>	<code>ans = 3</code>	Imaginary part of z .
<code>abs(z)</code>	<code>ans = 5</code>	Magnitude of z .
<code>sqrt(a^2+b^2)</code>	<code>ans = 5</code>	Indeed, the magnitude is given by $ z = \sqrt{a^2 + b^2}$.
<code>angle(z)</code>	<code>ans = 0.6435</code>	The angle of z in radians.
<code>phase(z)</code>	<code>ans = 0.6435</code>	Alternative computation of the angle.
<code>atan(b/a)</code>	<code>ans = 0.6435</code>	Indeed, the angle of z is given by $\phi = \tan^{-1}(b/a)$.
<code>angle(z)*180/pi</code>	<code>ans = 36.8699</code>	Conversion from radians to degrees.
<code>conj(z)</code>	<code>ans = 4.0000 - 3.0000i</code>	Complex conjugate.
<code>z'</code>	<code>ans = 4.0000 - 3.0000i</code>	Alternative computation of the complex conjugate.
<code>z*conj(z)</code>	<code>ans = 25</code>	An important property of complex numbers is revealed here. More specifically $zz^* = z ^2$, where $ z $ is the magnitude of z .

1.9.2 Forms of Complex Numbers

- Exponential form.** A complex number $z = a + bi$ is easily expressed in polar coordinates if written in the form $z = Ae^{j\varphi}$, where $A = |z|$ is the magnitude of z and $\varphi = \tan^{-1}(b/a)$ is the angle of z .
- Trigonometric form.** Euler's formula states that $e^{jx} = \cos(x) + j\sin(x)$. Hence, a complex number $z = a + bi$ can be written as $z = A(\cos(\varphi) + j\sin(\varphi))$, where $a = A\cos(\varphi)$, $b = A\sin(\varphi)$, $A = |z|$, and $\varphi = \tan^{-1}(b/a)$.

Commands	Results	Comments
<code>z = 4+3i</code>	$z = 4.0000 + 3.0000i$	Complex number $4 + 3i$.
<code>A = abs(z)</code>	$A = 5$	Magnitude of z .
<code>fi = angle(z)</code>	$fi = 0.6435$	Angle of z .
<code>z1 = A*exp(i*fi)</code>	$z1 = 4.0000 + 3.0000i$	Exponential equivalent form of z .
<code>A*cos(fi)</code>	$ans = 4$	Indeed $a = A \cos(\varphi)$.
<code>A*sin(fi)</code>	$ans = 3$	And also $b = A \sin(\varphi)$.
<code>A*(cos(fi)+i*sin(fi))</code>	$ans = 4.0000 + 3.0000i$	Trigonometric equivalent form of z .
<code>z3 = exp(i)</code>	$z3 = 0.5403 + 0.8415i$	The term $z3 = e^i$ is $0.5403 + 0.8415i$.
<code>abs(z3)</code>	$ans = 1$	The magnitude of $z3$ is $A = 1$.
<code>angle(z3)</code>	$ans = 1$	The angle of $z3$ is $\varphi = 1$ rad.

Sometimes we refer to these two forms as polar form of a complex number.

1.9.3 Operations with Complex Numbers

The addition and subtraction of two complex numbers is computed by simply adding and subtracting their corresponding real and imaginary parts. Suppose that $z_1 = a + bi$ and $z_2 = c + di$. Then,

$$z_1 \pm z_2 = (a \pm b) + (c \pm d)i. \quad (1.6)$$

In order to compute the multiplication and the division of two complex numbers or the power of a complex number we take into account that $i^2 = -1$. Thus,

$$z_1 z_2 = (a + bi)(c + di) = (ac - bd) + (ad + bc)i. \quad (1.7)$$

$$\frac{z_1}{z_2} = \left(\frac{ac + bd}{c^2 + d^2} \right) + \left(\frac{bc - ad}{c^2 + d^2} \right) i. \quad (1.8)$$

$$z_1^2 = (a + bi)(a + bi) = (a^2 - b^2) + (2ab)i. \quad (1.9)$$

Finally, we can define vectors or matrices of complex numbers as usual.

Commands	Results	Comments
<code>z1 = 1+2i</code>	$z1 = 1.0000 + 2.0000i$	Definition of the complex numbers $z1$ and $z2$.
<code>z2 = 2+1i</code>	$z2 = 2.0000 + 1.0000i$	
<code>z1+z2</code>	$ans = 3.0000 + 3.0000i$	Sum of complex numbers.
<code>z1-z2</code>	$ans = -1.0000 + 1.0000i$	Subtraction of complex numbers.
<code>z1*z2</code>	$ans = 0 + 5.0000i$	Product of complex numbers.
<code>z1/z2</code>	$ans = 0.8000 + 0.6000i$	Division of complex numbers.
<code>z1^2</code>	$ans = -3.0000 + 4.0000i$	Power of a complex number.

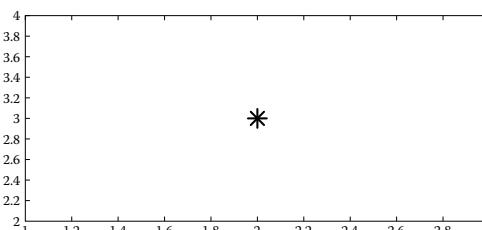
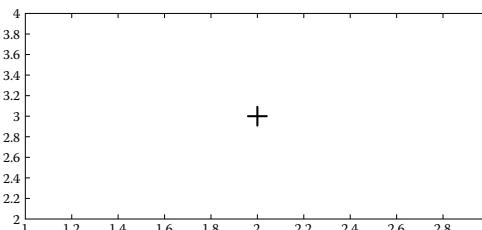
(continued)

(continued)

Commands	Results		Comments
<code>a = [z1 z2]</code>	<code>a = 1.00 + 2.00i</code>	<code>2.00 + 1.00i</code>	Row vector of complex numbers
<code>b = [z1; z2; z1]</code>	<code>b = 1.0000 + 2.0000i</code>	<code>2.0000 + 1.0000i</code>	Column vector of complex numbers.
<code>B = [z1 z2; z2 z1]</code>	<code>B = 1.00 + 2.00i</code>	<code>2.00 + 1.00i</code>	Matrix of complex numbers.

1.9.4 Graph of Complex Numbers

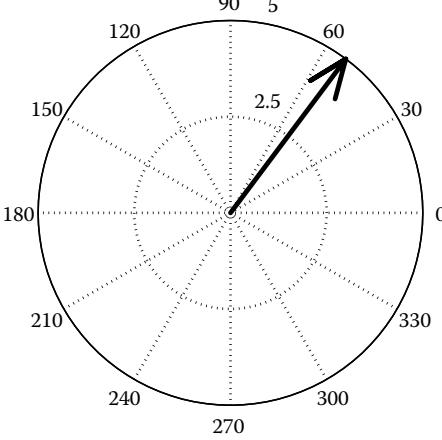
A complex number is usually plotted in the complex plane. Recall that the horizontal axis is the real axis and the vertical axis is the imaginary axis. Suppose that $z = a + bi$. Typing `plot(z)` produces a graph of an (almost invisible) point at the coordinates $[a, b]$. In order to make this point visible one can use the syntax `plot(z, '*')`, that is, to plot the complex number with a star.

Commands	Results	Comments
<code>z1 = 2+3i</code>	<code>z1 = 2.0000 + 3.0000i</code>	Definition of <code>z1</code> .
<code>plot(z1, '*')</code>		Graph of <code>z1</code> in the complex plane with a star. The star is placed at the coordinates [2, 3].
<code>a = real(z1);</code> <code>b = imag(z1);</code> <code>plot(a, b, '*')</code>		This analytical way is given in order to understand that plotting in the complex plane is a graph of the real versus the imaginary part of a complex number.
<code>plot(real(z1), imag(z1), '+')</code>		Plot of <code>z1</code> in the complex plane with a cross.

A vector of complex numbers is plotted in the complex plane exactly in the same way. By typing `plot(z)`, the points of the vector elements are connected with a solid line.

Commands	Results	Comments
<pre> z1 = 2+3i; z2 = 3-2i; z3 = -1+2i; z4 = -1-i; z5 = 0+0i; z = [z1 z2 z3 z4 z5] </pre>	<pre> z = 2.00+3.00i 3.00-2.00i -1.00+2.00i -1.00-1.00i 0 </pre>	A vector of complex numbers.
<pre> plot(z,'*') axis([-1.5 3.5 -2.5 3.5]) </pre>		Graph of the vector elements in the complex plane.
<pre> x = 0:0.1:10 z = x.^2+x*i plot(z) </pre>		Plotting with line a vector of complex elements.

As mentioned earlier a complex number is often expressed in polar form, i.e., in the form $z = Ae^{i\varphi}$, where $A = |z|$ is the magnitude and $\varphi = \tan^{-1}(b/a)$ is the angle of z . A suitable command to obtain a plot of this form is the command `compass(z)`. Here, z is drawn as an arrow emanating from the axis origin, with magnitude A and angle φ .

Commands	Results	Comments
<pre>z = 3+4i; mag = abs(z) ang = angle(z)*180/pi</pre> <pre>compass(z)</pre>	<pre>mag = 5 ang = 53.1301</pre> 	<p>The magnitude of z is 5 and the angle is approximately 53°.</p> <p>Indeed, the magnitude of the arrow is 5 and the angle is 53.1301°.</p>

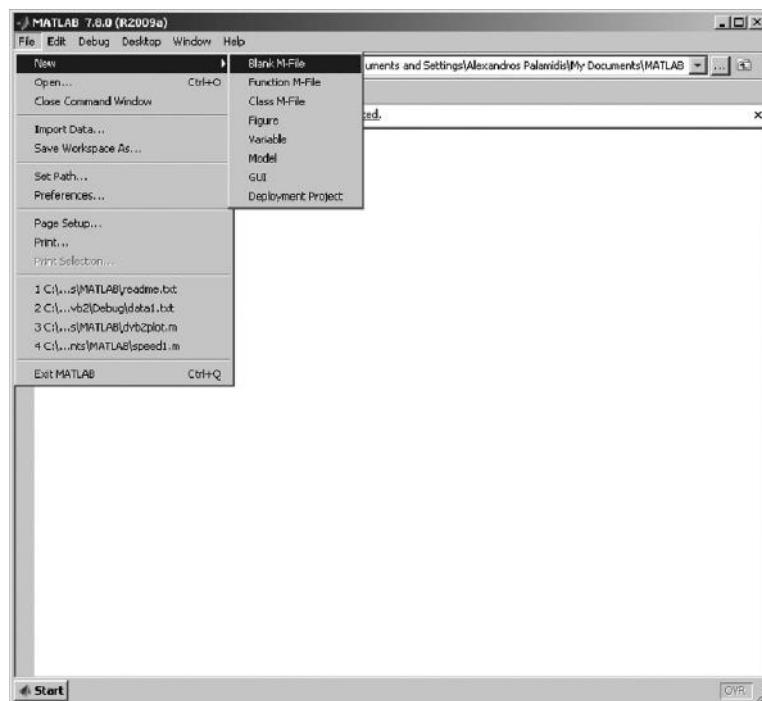
1.10 M-Files

All commands given so far were typed in the command prompt and immediately executed by MATLAB. In order to write big programs with many commands that are executed all together, the program must first be written in a text editor. In this editor, one can type all the needed commands to form a program, save the program, and execute it any time he or she wants. The text files are called *M-Files* due to their suffix **.m*. An M-File is created from the main menu by following the path File→New→ Blank M-File (Figure 1.4). An M-File must be saved in the *Current Directory* of MATLAB. In case an M-File is saved in another location in order to execute it, the directory in which it is contained must be set as the *Current Directory*. Typing the command `pwd` in the command prompt returns the *Current Directory*, while typing `cd foldername` changes the current working directory, i.e., sets the directory named *foldername* as the *Current Directory*.

There are two categories of M-Files: the *scripts* and the *functions*.

1.10.1 Scripts

Scripts are M-files with MATLAB commands. Their name must have a *.m* suffix, namely, is of the form *name.m*. One should be careful not to use as a filename the name of a command or that of another file. A script is executed either through the menu of the editor (Debug → Run) or by simply typing at the command prompt the name of the script without including the *.m* suffix. Scripts are suitable for solving problems that require many commands.

**FIGURE 1.4**

Creation of a new blank M-file.

Example

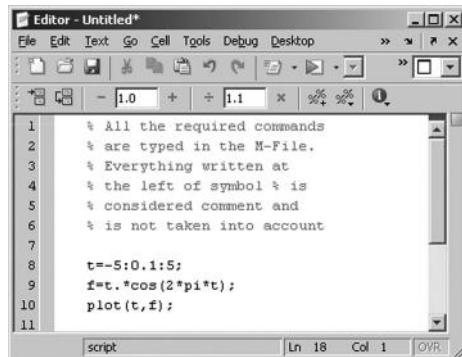
Write a program that plots the function $f(t) = t \cos(2\pi t)$, $-5 \leq t \leq 5$.

Commands	Results/Comments
	A new M-File is created.

File → New → Blank M-File

(continued)

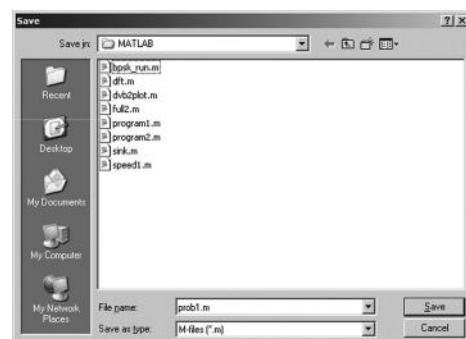
(continued)

Commands**Results/Comments**


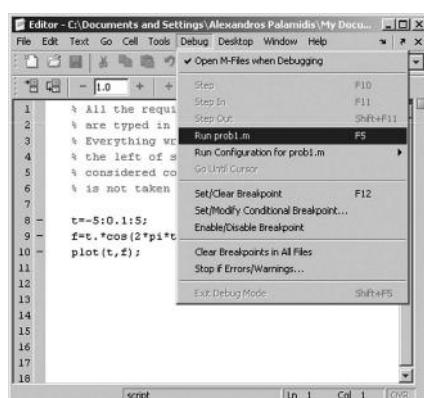
```
% All the required commands
% are typed in the M-File.
%
% Everything written at
% the left of symbol % is
% considered comment and
% is not taken into account
%
t=-5:0.1:5;
f=t.*cos(2*pi*t);
plot(t,f);
```

All commands that are needed are typed in the M-File.
Notice that nothing is executed.

The M-File is saved in the *Current Directory* as *prob1.m*

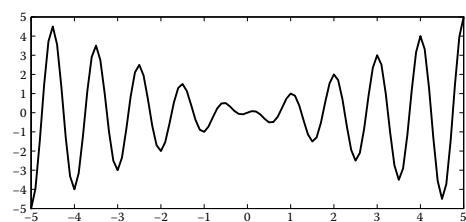


File→Save



```
% All the required commands
% are typed in
%
% Everything written at
% the left of %
% is considered comment and
% is not taken into account
%
t=-5:0.1:5;
f=t.*cos(2*pi*t);
plot(t,f);
```

The program is executed from the editor menu (Debug → Run) and the graph of $f(t)$ is obtained.



(continued)

Commands	Results/Comments
prob1	A second way to execute a script is to type the name of the script at the command prompt and press Enter.

Script *prob1.m* is saved and can be executed at anytime. The advantage of scripts is that they are implemented very easily. As a matter of fact, it is like writing in the command prompt and executing all commands together. The disadvantage is the lack of flexibility. More specifically, the script *prob1.m* will always plot the function $f(t) = t \cos(2\pi t)$ for $-5 \leq t \leq 5$. In order to gain flexibility we can use another type of M-Files, *functions*.

1.10.2 Functions

Functions are also M-Files, that is, are files with extension *.m* and must be saved in the *Current Directory* of MATLAB. The difference between functions and scripts is that a function accepts one or more input arguments and returns one or more output arguments. To declare that an M-File is a function the first line of the M-file must contain the syntax definition. More specifically, the first line of the M-file must be of the form $\text{function}[y_1, y_2, \dots, y_n] = \text{name}(x_1, x_2, \dots, x_m)$. The variables y_1, y_2, \dots, y_n are the outputs of the function while x_1, x_2, \dots, x_m are the input arguments. In case there is only one output, the square brackets are not necessary. The “*name*” specifies the name of the function. In order to execute a function, first the M-File is saved in the *Current Directory*.

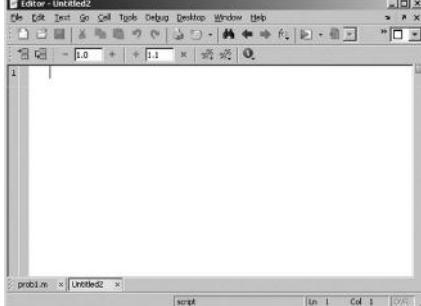
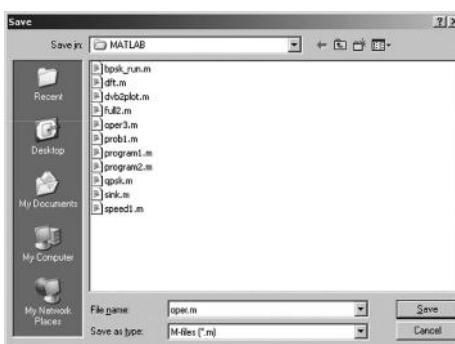
Remark

The name given to the function must be the same as the filename by which the M-File will be saved in the *Current Directory*.

In contrast to scripts, functions are executed *only* from the command prompt, by typing the function name and specifying the appropriate (in number and type) input and output arguments.

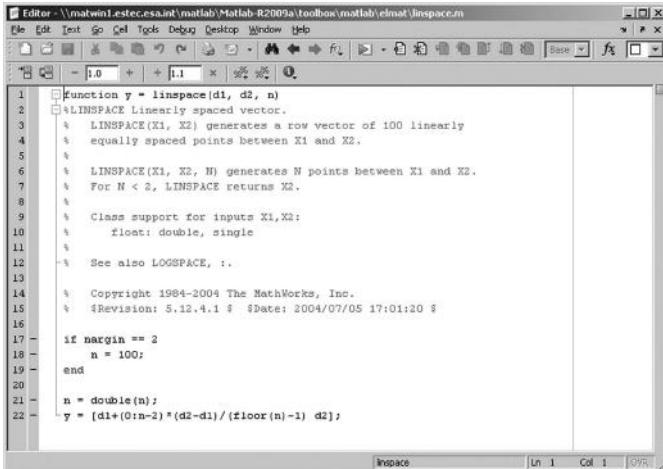
Example

Write a function that accepts as input arguments two matrices and returns their sum and their product.

Commands	Results	Comments
		A new M-File is created.
From the main menu we follow the path: File → New → Blank M-File		
		At the first line we specify that this M-file is a function. More specifically, a function named <i>oper</i> accepts as input arguments the variables <i>A</i> and <i>B</i> and returns as outputs the variables <i>sm</i> and <i>pro</i> . Next, a brief comment explains what this function does followed by the MATLAB statements that define the two outputs.
		The M-File is saved in the <i>Current Directory</i> with the same filename to the function's name, i.e., it is saved as <i>oper.m</i> .

Commands	Results	Comments
oper	??? Input argument 'A' is undefined. Error in ⇒ C:\MATLAB6p5\work\oper.m On line 4 ⇒ sm=A+B;	Trying to execute the function by typing only the function's name causes an error. This is expected as there are no input arguments defined.
C = [1 2; 1 2]; D = [2 3; 2 3]; oper(C,D)	ans = 3 5 3 5	The input arguments, namely, two matrices C and D are defined in the command window and the function is now executed. However, only the sum of the two input matrices is returned, assigned to the default variable ans. This is due to the fact that no output variables are specified.
C = [1 2; 1 2]; D = [2 3; 2 3]; [AD, MUL] = oper(C,D)	AD = 3 5 3 5 MUL = 6 9 6 9	The function is executed by also specifying two output variables, AD and MUL. The sum of C and D is assigned to variable AD, while the product C*D is assigned to variable MUL. This is the correct way to execute a function.
E = [1 1; 2 3]; F = [1 2; 3 4]; [AD, MUL] = oper(E,F)	AD = 2 3 5 7 MUL = 4 6 11 16	The function oper.m is executed with two different matrices as input arguments.
[AD, MUL] = oper(3,5)	AD = 8 MUL = 15	The function oper.m accepts scalar numbers as input arguments and returns their sum and product. It is clear that a function offers a high degree of flexibility compared to a script file.

Typing help and the function's name returns the first comments written in the function's M-File. Most MATLAB commands are written as functions. In order to view (and possibly edit) a MATLAB command type open and the name of the command.

Commands	Results	Comments
help oper	This function computes the sum and the product of two matrices	These are the comments written in the function oper.m.
open linspace	 The screenshot shows the MATLAB Editor window with the file 'linspace.m' open. The code defines a function that generates linearly spaced vectors. It includes documentation strings (docstrings) explaining the function's purpose, inputs (xi, x2, n), outputs (y), and usage examples. It also includes copyright information and revision details.	The command open linspace displays the M-File of the built-in command (or function) linspace.

Looking into the function *linspace.m*, we notice a variable named *nargin*. This variable is present in all built-in MATLAB functions. The value of *nargin* is the number of input arguments that the function accepts during its call. There is also a variable named *nargout*, whose value is the number of outputs that the function returns. Using those two variables a programmer can control (e.g., to provide an error message) function calls with different than the expected input/output arguments. A similar command to the command *open* is the command *type*. It has the same use and syntax (e.g., *type linspace*), but the code of the built-in function *linspace* is displayed in the command window.

1.11 Input/Output Commands

The command `a = input ('string')` is employed in order to get input data from a user. This command displays at the command window the text “*string*” and then waits for input data from the keyboard. The user input is assigned to variable *a*. Using the syntax `b = input ('string', 's')` the user input is considered a string. To display a variable *A* in the command window simply type `disp (A)`. The syntax `disp ('string')` displays the string in the quotes. To display both text and the arithmetic value of a variable, first the value of the variable is converted to text by using the command `num2str(A)` and then the two strings are concatenated according to the process introduced in Section 1.7.1. The command `display(A)` has the same use and syntax to `disp`, but additionally displays the name of the variable. Another output display command is the command `fprintf ('format', A)`. This command will be further discussed in Section 1.12.

Commands and User Input	Results	Comments
<code>a = input ('Insert number')</code>	Insert number	The text in the quotes is displayed. MATLAB expects input from the keyboard.
67.8	<code>a = 67.8000</code>	Number 67.8 is the input from the user and is assigned to variable <i>a</i> .
<code>A = input ('Insert matrix')</code>	Insert matrix	The text in the quotes is displayed. MATLAB expects input from the keyboard.
<code>[1 2; 3 4]</code>	<code>A = 1 2 3 4</code>	Matrix [1 2; 3 4] is the input from the user and is assigned to variable <i>A</i> .
<code>b = input ('text ? ', 's')</code>		Using this syntax the user input is considered a string.
hello	<code>b = hello</code>	The string <i>hello</i> is assigned to variable <i>b</i> .
<code>disp('The matrix is')</code>	The matrix is	Text displayed using the command <code>disp</code> .
<code>disp(A)</code>	<code>1 2 3 4</code>	Variable displayed using the command <code>disp</code> .
<code>display(A)</code>	<code>A = 1 2 3 4</code>	Using the command <code>display</code> we display the name of the variable and its value.
<code>c = num2str(a)</code>	<code>c = 67.8</code>	The value of <i>a</i> is converted to string and is assigned to variable <i>c</i> .

(continued)

Commands and User Input	Results	Comments
<code>d='The number is'</code>	<code>d = The number is</code>	Text is assigned to variable <i>d</i> .
<code>e=[d c];</code>		The strings <i>d</i> and <i>c</i> are concatenated.
<code>disp(e)</code>	<code>The number is 67.8</code>	Text and variable appear in the same line.
<code>disp(['The number is' num2str(a)])</code>	<code>The number is 67.8</code>	Direct execution of the four previous commands.
<code>fprintf('%s %f','The number is',a)</code>	<code>The number is 67.8</code>	Use of the command <code>fprintf</code> to produce the same output.

1.12 File Management

In MATLAB, it is possible to create external files in order to store the data. The command `fopen(file1)` opens the file with filename *file1* for read access. Using the syntax `fid=fopen('file1')` opens the file with filename *file1* and returns a number stored in the variable *fid* that is used as a pointer to the file *file1*. A special character is used within the command `fopen`. Character “r” opens a file in read mode, character “w” opens a file in write mode by deleting the previous contents, while character “a” opens the file in write mode without deleting its contents. The data is inserted at the end of the file. Character “r+” specifies read and write mode, while character “w+” specifies read and write mode with deletion of the previous contents. For example, the command `fid=fopen('file1', 'r')` opens the file *file1* in read mode. The enumeration of *fid* starts from 3. The value *fid*=1 denotes that data are shown in the display while value 2 directs the data in the device specified to display errors (e.g., a printer). The command `fclose(fid)` closes the file pointed by *fid*. To complete the changes made in a file, the file must be closed; thus, using the `fclose` command is obligatory. The command `fprintf(fid,format,variables)` stores the data of the specified *variables* in the file pointed by *fid*. The *format* indicates the way that data is stored in the file. If the variables are matrices, data are read by column. The available formats and control characters are illustrated in the table below.

Control Characters	Available Formats
\n New line	%e Exponential
\t Tab	%f Floating point
\f New page	%u Integer
	%s String

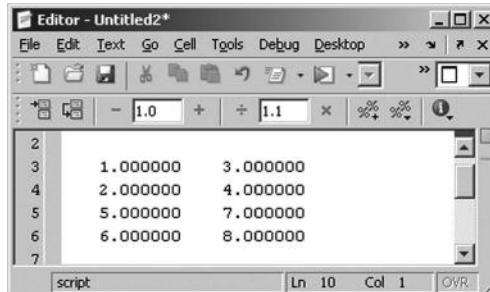
Finally, the command `fscanf(fid, format, matrix_size)` reads serially the data from the file specified by *fid* and stores them in a matrix. The matrix size is given as $[m,n]$.

Example

Store the matrices $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$ in a file named *file1.txt*. Next, read the data from the file *file1.txt* and store them in a matrix \mathbf{C} .

Commands	Results	Comments
$\mathbf{A} = [1 \ 2 ; 3 \ 4]$	$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	Matrices \mathbf{A} and \mathbf{B} are defined.
$\mathbf{B} = [5 \ 6 ; 7 \ 8]$	$\mathbf{B} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$	
<code>fid=fopen('file1.txt','w')</code>	<code>fid = 3</code>	The file <i>file1.txt</i> is created and saved in the <i>Current Directory</i> of MATLAB.
<code>fprintf(fid,'%f %f \n',A,B);</code>		The data of \mathbf{A} and \mathbf{B} are stored in the file specified by <i>fid</i> , i.e., are stored in the file <i>file1.txt</i> .
<code>fclose(fid);</code>		The file is closed and the recording is completed.

From the main menu (File → Open) we open the file *file1.txt* in the built-in text editor.



The file *file1.txt*.

`fid=fopen('file1.txt','r');`

File1.txt is opened in read mode.

```
c=1
3
2
4
5
7
6
8
```

`c=fscanf(fid, '%f', [8,1])`

The data are read serially and stored in a 8×1 matrix c .

1.13 Logical/Relational Operators

A logic expression is an expression whose outcome is *True* (1) or *False* (0). A simple example of such an expression is a logical operation or relationship. The main relational operators are given in the table below.

Relational Operator	Command	Explanation	Examples	
$= =$	eq	Equal	$20 == 20$ ans = 1	eq(20,20) ans = 1
$>$	gt	Greater than	$3 > 5$ ans = 0	gt(5,3) ans = 1
$<$	lt	Less than	$3 < 5$ ans = 1	lt(5,3) ans = 0
$\sim =$	ne	Not equal	$2 \sim = 2$ ans = 0	ne(2,3) ans = 1
\geq	ge	Greater than or equal	$5 \geq 5$ ans = 1	ge(4,5) ans = 0
\leq	le	Less than or equal	$5 \leq 4$ ans = 0	le(5,5) ans = 1

Logical operations can be also implemented between matrices. Some related commands are given in the following table.

Command	Explanation	Examples
<code>isequal(a,b)</code>	Returns 1 if matrices <i>a</i> and <i>b</i> are equal.	<code>a = [1 2] ; b = [1 2] ; isequal(a,b) ans = 1</code>
<code>isempty(a)</code>	Returns 1 if matrix <i>a</i> has no elements.	<code>isempty(a) ans = 0</code> <code>a = [] ; isempty(a) ans = 1</code>
<code>isnan(a)</code>	Returns 1 if argument <i>a</i> is NaN.	<code>a = 0/0 Warning: Divide by zero. a = NaN isnan(a) ans = 1</code>
<code>isinf(a)</code>	Returns 1 if argument <i>a</i> is infinity.	<code>a = 1/0 Warning: Divide by zero. a = Inf isinf(a) ans = 1</code>

Finally, there are also logical operators.

Operator	Explanation	Example
&	Logical operator AND.	$1 \& 0$ ans = 0
	Logical operator OR.	$1 0$ ans = 1
~	Logical operator NOT.	~ 0 ans = 1
xor()	Logical operator XOR.	xor(1,1) ans = 0
all (condition)	Returns 1 if the condition is fulfilled for all elements of the vector.	a = [1 2]; b = [3 2]; all(a < b) ans = 0
any (condition)	Returns 1 if the condition is fulfilled for any element of the vector.	any(a < b) ans = 1

Regarding the commands `all` and `any`, if no condition is specified the logical operation is implemented referring to zero. For example, the command `all(a)` returns 1 if all elements of `a` are nonzero. Even though there is a defined priority between the logical operators, the authors suggest using parenthesis for safety. Finally, we mention that logical operations between matrices are implemented as logical operations between the corresponding elements and return a matrix of the same size with elements that are the result of the operation.

1.14 Control Flow

`if-elseif-else, switch-case-otherwise, for, while, break, continue, return`

The syntax of the conditional statement `if` is

```

if condition1
    Statements
elseif condition2
    Statements
else
    Statements
end

```

The command `switch` is an alternative to `if`, appropriate when dealing with multiple cases. The syntax of command `switch` is as follows.

```

switch (variable)
case {value1}
    Statements
case {value2,value3,...}
    Statements
otherwise
    Statements
end

```

Command `switch` is suitable to test equalities in contrast to command `if` that tests also inequalities. Note that command `break` must not be used together with `switch`. Also, notice that braces {} are used instead of parentheses.

Example

Write a program (in a M-File) that accepts a mark between 1 and 10 as user input. According to the input mark, the program returns an answer pass (for mark ≥ 5) or fail.

Three different equivalent programs are given.

M-Files and Their Execution from the Command Prompt	Results	Comments
<pre> a = input ('mark?') if (a>0 &a<5) disp('fail') elseif (a>= 5 & a<= 10) disp('pass') else disp('choose again') end % File→Save as“if1.m” if1 </pre>	<pre> mark ? 3 fail </pre>	The statements <code>if-elseif-else</code> . The user input is number 3. Notice the use of the logical operator & (i.e., AND).
<pre> a = input ('mark?') if (a>0 &a<5) disp('fail') end if (a>= 5 & a<= 10) disp('pass') end if (a<= 0 a>10) disp('choose again') end % File→Save as“if2.m” if2 </pre>	<pre> mark ?16 choose again </pre>	The requested program is implemented using the simple syntax of the <code>if</code> statement. Notice the use of the logical operator (i.e., OR).
<pre> a = input ('mark?') switch(a) case{1,2,3,4} disp('fail') case{5,6,7,8,9,10} disp('pass') otherwise disp('choose again') end % File→ Save as“swi.m” swi </pre>	<pre> mark ? 8 pass </pre>	The statement <code>switch</code> . Program <code>swi.m</code> is equivalent to the two previous implementations.

The command `for` is used to repeat statements a specific number of times. The syntax is

```
for counter = vector
Statements
end
```

The command `while` is used to repeat statements an indefinite number of times. The syntax is

```
while (condition)
Statements
end
```

The command `break` is employed in order to stop a loop (*for* or *while* loop) before its completion. The command `continue` passes control to the next iteration of loop, i.e., it is used to stop one step of a loop. The command `return` is usually employed in functions to cause an early stop. Finally, with the key combinations *Ctrl-C* and *Ctrl-Break* one can terminate any statement under execution. To illustrate the use of loops several examples are given below.

M-Files and Their Execution from the Command line	Results	Comments
<pre>% a = 1:5 % for i = a for i=1:5 b(i) = i^2; end display(b) % File→ Save as"for1.m" for1</pre>	<pre>b = 1 4 9 16 25</pre>	Loop implemented with the command <code>for</code> . The comments at the first 2 lines illustrate an alternative way to define the loop-counter <i>i</i> . In this program, a <code>for</code> -loop is used to create a vector whose elements are the squares of the numbers 1 to 5
<pre>a=1; while(a>= 0) disp('positive to loop') disp('negative to exit') a=input('enter number'); end % File→ Save as"while1.m" while1</pre>	<pre>positive to loop, negative to exit enter number 89 positive to loop, negative to exit enter number 2 positive to loop, negative to exit enter number -5</pre>	Loop implemented with command <code>while</code> . Notice that variable <i>a</i> must be initialized before being used in the loop. In this program, the loop continues as long as the user input is not a negative number.

(continued)

M-Files and Their Execution from the Command line	Results	Comments
<pre>a=1; while(1) a=input('Mark ?') if (a>0 & a<5) disp('fail') break; elseif (a>=5 & a<=10) disp('pass') break; else disp('choose again'); end % File→ Save as "while2.m" while2</pre>	<pre>Mark ? 11 choose again Mark ? -6 choose again Mark ? 7 pass</pre>	By typing <code>while(1)</code> an infinite loop is created. In order to terminate the infinite loop the command <code>break</code> is used. In this program, if the user input is not valid (i.e., between 1 and 10) the user is requested to enter again his mark.
<pre>clear; for i=1:5 if (i == 3) continue; end b(i) = i^2; end display(b) % File-> Save as "cont1.m" cont1</pre>	<pre>b = 1 4 0 16 25</pre>	Using the command <code>continue</code> causes the non-execution of the statements for $i=3$. The command <code>continue</code> is used together with an <code>end</code> statement.
<pre>function B=det1(A) if det(A) == 0 disp('non invertible matrix') B=NaN; return else B=inv(A); end % File-> Save as "det1.m" B=det1([2 2;1 1]);</pre>	<pre>non invertible matrix</pre>	The command <code>return</code> terminates the function <code>det1.m</code> if the matrix determinant is zero. Notice that a value must be assigned to the output argument of the function. Hence, we set <code>B=NaN</code> .

Finally, something that should be mentioned is that when programming with MATLAB we should avoid using loops as they are computationally expensive. Most of the times, the same process can be implemented using vectors/matrices.

Commands	Results	Comments
<pre>for i=1:10 a(i)=i^2; a = 1 4 9 16 25 36 end 49 64 81 100 a</pre>	<pre>a = 1 4 9 16 25 36</pre>	Implementation with loop.
<pre>i=1:10; a=i.^2</pre>	<pre>a = 1 4 9 16 25 36</pre>	Implementation with vectors.

1.15 Symbolic Variables

The types of variables defined and used so far are matrices, vectors, scalars, and strings. In MATLAB, another variable type is available: the symbolic variable (or object). A symbolic variable is defined by the commands `sym` and `syms`. More specifically, by typing `x = sym('x')` or `syms x y z` if more than one symbolic variable is needed, the variables x y z are defined as symbolic variables. The use of symbolic variables allows the computation of limits, integrals, derivatives, etc.

Commands	Results	Comments
<code>x = sym('x')</code>	$x = x$	Definition of the symbolic variable x .
<code>a = limit(sin(x)/x, 0)</code>	$a = 1$	Computation of $\lim_{x \rightarrow 0} (\sin(x)/x)$.
<code>syms y z w</code>		Easy way to define multiple symbolic variables.
<code>y</code> <code>z</code> <code>w</code>	$y = y$ $z = z$ $w = w$	Confirmation that y , z , and w are symbolic variables.
<code>A = [x 2*y; z-w, z+w]</code>	$A = [x, 2*y]$ $[z-w, z+w]$	Symbolic matrices can be also defined.
<code>det(A)</code>	$ans = x*z+x*w-2*y*z+2*y*w$	The determinant of A .
<code>F = 2*x^2+3*x+4</code>	$F = 2*x^2+3*x+4$	Polynomial of symbolic variables. The use of dot operator before the power operator is not necessary since x is not a vector but a symbolic variable. F is called a symbolic expression.

1.15.1 Differentiation of a Function

To compute the derivative of a function $f(x)$, first x must be declared as a symbolic variable. Then, the function f is defined in terms of the independent variable x and the derivative is computed by typing the command `diff(f)`. Typing `diff(f, n)` returns the n th derivative of $f(x)$. If the function has several symbolic variables, the syntax `diff(f, variable, n)` returns the n th partial derivative with respect to the variable specified in the `diff` command.

Example

Compute the partial derivatives of the function $f(x, y) = e^{-x} \cos(y)$.

Commands	Results	Comments
<code>syms x y</code>		Declaration of the symbolic variables x and y .
<code>f = exp(-x)*cos(y)</code>	<code>f = exp(-x)*cos(y)</code>	The function $f(x, y)$ is a symbolic expression.
<code>diff(f, x)</code>	<code>ans = -exp(-x)*cos(y)</code>	Partial derivative $\partial f / \partial x$.
<code>diff(f, y)</code>	<code>ans = -exp(-x)*sin(y)</code>	Partial derivative $\partial f / \partial y$.
<code>diff(f, x, 2)</code>	<code>ans = exp(-x)*cos(y)</code>	Second derivative of f with respect to x .
<code>t = diff(f, x)</code> <code>diff(t, x)</code>	<code>t = -exp(-x)*cos(y)</code> <code>ans = exp(-x)*cos(y)</code>	Alternative computation of the second derivative of f with respect to x . The result of the first derivative is assigned to variable t and the derivative of t is the desired result.

1.15.2 Integration of a Function

The command used to compute the integral of a function is the command `int`. Suppose that f is a function. Typing `int(f, variable)` returns the indefinite integral of f with respect to the variable specified at the `int` command. Of course, it is not always possible to compute an integral or write the integration result in a closed-form expression. A definite integral is computed by `int(f, variable, lower-limit, upper-limit)`.

Commands	Results	Comments
<code>syms x t</code>		Declaration of the symbolic variables x and t .
<code>f = x^2</code> <code>g = x^2 + exp(-t)</code>	<code>f = x^2</code> <code>g = x^2 + exp(-t)</code>	The functions $f(x) = x^2$ and $g(x, t) = x^2 + e^{-t}$ are defined as symbolic expressions.
<code>int(f, x)</code>	<code>ans = 1/3*x^3</code>	Computation of $\int x^2 dx$.
<code>pretty(ans)</code>	<code>1/3 x^3</code>	The command <code>pretty</code> prints symbolic output in a format that resembles typeset mathematics.
<code>int(f, x, -1, 1)</code>	<code>ans = 2/3</code>	Computation of $\int_{-1}^1 x^2 dx$.
<code>int(exp(-x^2), x, -inf, inf)</code>	<code>ans = pi^(1/2)</code>	Computation of $\int_{-\infty}^{\infty} e^{-x^2} dx$. To define $\pm\infty$ in MATLAB we write $\pm inf$. Note that the function can be directly defined inside the command <code>int</code> .
<code>z = int(g, x)</code> <code>int(z, t)</code>	<code>z = 1/3*x^3 + exp(-t)*x</code> <code>ans = 1/3*x^3*t - exp(-t)*x</code>	Computation of the double indefinite integral $\int \int g(x, t) dx dt$.

1.15.3 Summation of a Function

The command `symsum` computes symbolic summations. The most convenient syntax is `symsum(expression, index, lower-limit, upper-limit)`.

Commands	Results	Comments
<code>syms w k f = w^k symsum(f, k, 0, inf)</code>	$f = w^k$ $ans = -1/(w-1)$	Computation of $\sum_{k=0}^{\infty} w^k$. The result is $1/(1-w)$.

1.15.4 Rational Form

The command `numden` is employed in order to convert a function written in a complicated form into a simple rational expression. If f is the complicated expression, typing `[n, d] = numden(f)` returns the numerator n and the denominator d of the equivalent simplified rational expression of f .

Commands	Results	Comments
<code>syms x</code>		Definition of the symbolic variable x .
<code>f = (x^2) / (x+4) + 3 / (x-3)</code>	$f = x^2 / (x+4) + 3 / (x-3)$	The complicated written function is $f(x) = \frac{x^2}{x+4} + \frac{3}{x-3}$.
<code>[n, d] = numden(f)</code>	$n = x^3 - 3*x^2 + 3*x + 12$ $d = (x+4) * (x-3)$	The numerator and denominator of the rational form of $f(x)$, or in other words, $f(x) = \frac{x^3 - 3x^2 + 3x + 12}{(x+4)(x-3)}$.

1.15.5 Solving Algebraic Equations

The command `solve` computes the roots of a symbolic expression f . The best practice is to specify the variable that you solve for, and not let MATLAB choose what is best for you.

Commands	Results	Comments
<code>syms x</code>		Definition of the symbolic variable x .
<code>f = x^2 + 6*x + 5</code>	$f = x^2 + 6*x + 5$	The expression $x^2 + 6x + 5 = 0$ is assigned to variable f .
<code>solve(f, x)</code>	$ans = [-5; -1]$	Solving f for x . The roots are -5 and -1 .
<code>solve('x^2 + 6*x + 5 = 0', x)</code>	$ans = -1$ -5	Direct computation of the solution of equation $x^2 + 6x + 5 = 0$.

The command `solve` can be also applied when dealing with systems of two or more algebraic equations. The appropriate syntax in this case is `solve('eqn1', 'eqn2', ..., 'eqnN', 'var1, var2, ..., varN')`, where $eqnm$ denotes the m th equation and $varm$ the m th variable to solve for.

Example

Compute x and y for the system $y = 2x + 3$ and $y = 4x + 5$.

Commands	Results	Comments
<code>[x, y] = solve ('y=2*x+3', 'y=4*x+5', 'x, y')</code>	$x = -1$ $y = 1$	Solution of a system of algebraic equations. Notice that it is not necessary to declare symbolic variables in order to use the command <code>solve</code> . Also notice that it is possible to assign the output of <code>solve</code> to a variable.

1.15.6 Solving Differential Equations

It is quite easy to solve ordinary differential equations, by using a similar to `solve` command, named `dsolve`. The syntax is `dsolve('f','initial-conditions', 'independent-variable')`, where f is a n th order differential equation, the initial conditions are the values (usually at zero) of the $n - 1$ derivatives, while the independent variable is usually omitted. By default, t is considered as the independent variable. Regarding the definition of the differential equation f , the first derivative of y is denoted by Dy , the second derivative of y is denoted by $D2y$, and so on.

Example

Compute the solution of the differential equation $y'(t) + y(t) + 1 = 0, y(0) = 0$.

Commands	Results	Comments
<code>f = 'Dy+y+1'</code>	<code>f = Dy+y+1</code>	Definition of the differential equation. The quotes are required.
<code>it = 'y(0) = 0'</code>	<code>it = y(0) = 0</code>	The initial condition, i.e., the value of $y(t)$ at $t = 0$ is defined. Again quotes must be used.
<code>dsolve(f, it)</code>	<code>ans = -1+exp(-t)</code>	Solution of the differential equation. The independent variable is omitted.
<code>dsolve('Dy+y+1', 'y(0) = 0', 't')</code>	<code>ans = -1+exp(-t)</code>	Direct computation of the solution of the differential equation.

Example

Compute the solution of the differential equation $y''(t) - 1 = 0, y(1) = 1, y'(2) = 2$.

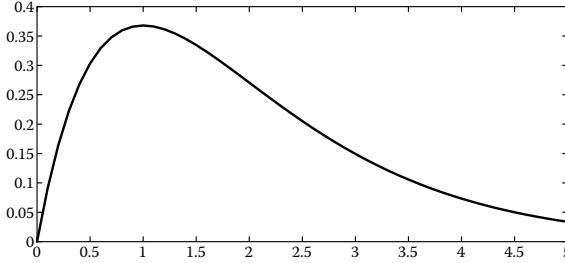
Commands	Results	Comments
<code>f = 'D2y-1 = 0'</code>	<code>f = D2y-1 = 0</code>	Definition of the differential equation.
<code>it = 'y(1) = 1, Dy(2) = 2'</code>	<code>it = y(1) = 1,</code> <code>Dy(2) = 2</code>	The conditions are the value of $y(t)$ at $t = 1$ and the value of $y'(t)$ at $t = 2$.
<code>y = dsolve(f, it)</code>	<code>y = 1/2*t^2+1/2</code>	The function $y(t) = (t^2 + 1)/2$ is the solution of the differential equation.

1.15.7 The Command `subs`

The command `subs` is used to replace one or more variables (symbolic or numerical) of a symbolic expression with some others. The syntax is `subs(f, old, new)`, where f is the expression, old are the replaced variable(s), and new are the values or the variables that replace the old ones.

Commands	Results	Comments
<code>syms y x z f = 2*x+y+4</code>	$f = 2*x+y+4$	The symbolic expression f has two symbolic variables.
<code>w = subs(f, y, 3)</code>	$w = 2*x+7$	Variable y is replaced by number 3.
<code>q = subs(w, x, z)</code>	$q = 2*z+7$	Variable x is replaced by variable z .
<code>subs(q, 7, 11)</code>	$ans = 2*z+11$	Number 7 is replaced by number 11.
<code>subs(f, [x, y], [z, 3])</code>	$ans = 2*z+7$	The variables x and y of the expression f are replaced by z and 3, respectively.

The command `subs` is very useful as it makes possible the graph implementation of a symbolic expression.

Commands	Results	Comments
<code>syms t y y=t*exp(-t)</code>	$y = t*exp(-t)$	The function $y(t) = te^{-t}$ is defined as symbolic expression.
<code>plot(t, y)</code>	??? Error using ⇒ plot Conversion to double from sym is not possible.	Trying to plot $y(t)$ in the usual way causes an error.
<code>t1 = 0:.1:5; y1 = subs(y, t, t1); plot(t1, y1);</code>		The symbolic variable t is replaced by the vector $t1$. The output $y1$ becomes also a vector. The plot of $y(t) = te^{-t}$ is done in the time interval specified from $t1$.

1.16 Polynomials

There are many ways to represent a polynomial in MATLAB. The most convenient way is to represent it as a row vector whose elements correspond to the polynomial coefficients in descending order. Of course, if a polynomial term is missing (or in other words, its coefficient is zero), the corresponding vector element is set to zero. For example, the

polynomial $p(x) = 2x^3 + 5x - 6$ is represented by $p = [2 \ 0 \ 5 \ -6]$. Suppose that p and q are vectors that represent the polynomials $p(x)$ and $q(x)$, respectively. Operations between $p(x)$ and $q(x)$ are implemented as follows:

- Addition and subtraction: If $p(x)$ and $q(x)$ are of the same order, then p and q are vectors of the same length and the operations of addition and subtraction are element per element operations. If $p(x)$ and $q(x)$ are not of the same order, the vector of smaller size must be padded at its beginning with zeros.
- Multiplication: The product $p(x) \cdot q(x)$ is computed by typing the command `conv(p, q)`.
- Division: The command `[a, b] = deconv(p, q)` returns the quotient a and the remainder b of the division $p(x)/q(x)$.

Commands	Results				Comments
<code>p1 = [5 6];</code> <code>p2 = [7 0 2];</code>					Definition of the polynomials $p_1(x) = 5x + 6$ and $p_2(x) = 7x^2 + 2$.
<code>p = conv(p1, p2)</code>	<code>p =</code>	35	42	10	The product $p_1(x)p_2(x)$ is the polynomial $p(x) = 35x^3 + 42x^2 + 10x + 12$.
<code>[a, b] = deconv(p2, p1)</code>	<code>a =</code>	1.40	-1.68		The ratio $p_2(x)/p_1(x)$ is $a(x) + b(x)/p_1(x) = 1.4x - 1.68 + 12.08/(5x + 6)$.
<code>conv(a, p1) + b</code>	<code>ans =</code>	7.00	0.00	2.00	Confirmation of the polynomial division. Indeed $a(x)p_1(x) + b(x) = p_2(x)$.
<code>p1 = [0 5 6];</code> <code>ad = p1+p2</code> <code>sub = p2-p1</code>	<code>ad =</code>	7	5	8	To add and subtract $p_1(x)$ and $p_2(x)$ one zero element is padded in the beginning of vector $p1$.
<code>sub =</code>	<code>sub =</code>	7	-5	-4	

Other operations between polynomials that are supported in MATLAB are

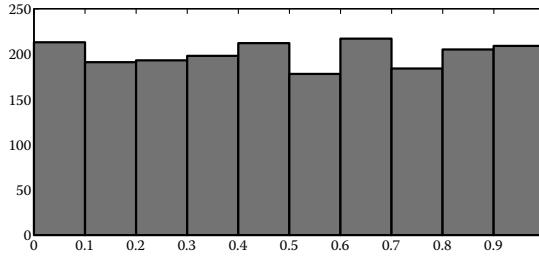
- Polynomial roots computation: The command `r = roots(p)` computes the roots of the polynomial p and stores them in a column vector r .
- If the roots r of a polynomial are known, typing `p = poly(r)` returns the coefficients of the polynomial.
- The derivative of a polynomial is computed by the command `h = polyder(p)`.
- Finally, to evaluate a polynomial $p(x)$ at a specific value x_0 , i.e., to compute $p(x_0)$ the suitable command is `polyval(p, x0)`.

Commands	Results				Comments
<code>p = [1 -0.5 3]</code>	<code>r =</code>	0.2500	+ 1.7139i		Definition of the polynomial $p(x) = x^2 - 0.5x + 3$
<code>r = roots(p)</code>		0.2500	- 1.7139i		and computation of its roots.
<code>p = poly(r)</code>	<code>p =</code>	1.00	-0.50	3.00	Construction of the polynomial from its roots.
<code>h = polyder(p)</code>	<code>h =</code>	2.0000	-0.5000		The derivative of $p(x)$ is $h(x) = p'(x) = 2x - 0.5$.
<code>polyval(p, 2)</code>	<code>ans =</code>	6			Evaluation of $p(x)$ for $x = 2$.

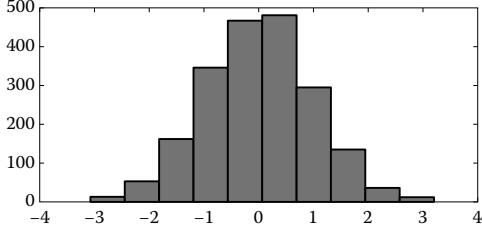
1.17 (Pseudo)Random Numbers

`rand randn hist pie mean var`

Random numbers are a subject of great importance and find applications in many areas of interest such as simulation or cryptography. In MATLAB, there is the possibility to generate random (or pseudorandom) numbers distributed according to a specific distribution function. The command `x = rand(N, K)` returns a matrix of size $N \times K$ whose elements are *uniformly* distributed in $[0 \ 1]$. To create random numbers uniformly distributed in an interval $[a, b]$ the associated statement is `x = a + (b - a) * rand(N, K)`. In order to generate *normally* distributed random numbers, we use the command `randn`. The syntax `x = randn(N, K)` returns a matrix of size $N \times K$ whose elements are distributed according to the normal (or Gaussian) distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$. The appropriate statement to generate random numbers distributed according to a normal distribution with mean m and variance s^2 is `x = m + s * randn(N, K)`. The histogram of a vector x whose elements are random numbers is plotted by the command `hist(x, k)`, where k is the numbers of bins. A pie plot of the data in vector x is implemented by the command `pie(x)`. Finally, command `mean(x)` computes the mean value of the vector elements, while command `var(x)` calculates the variance of the values in x .

Commands	Results			Comments	
<code>x = rand(4, 3)</code>	<code>x =</code>	0.9218 0.7382 0.1763 0.4057	0.9355 0.9169 0.4103 0.8936	0.0579 0.3529 0.8132 0.0099	Matrix of 4 rows and 3 columns whose elements are uniformly distributed over the interval $[0 \ 1]$.
<code>x = 2 + (5 - 2) * rand(4, 3)</code>	<code>x =</code>	2.4167 2.6083 2.5962 3.8114	2.8166 2.5964 2.0458 4.2404	3.3353 4.7954 3.3980 3.2559	Matrix of 4 rows and 3 columns whose elements are uniformly distributed over the interval $[2 \ 5]$.
<code>x = randn(4, 3)</code>	<code>x =</code>	-0.4326 -1.6656 0.1253 0.2877	-1.1465 1.1909 1.1892 -0.0376	0.3273 0.1746 -0.1867 0.7258	Matrix of 4 rows and 3 columns whose elements are distributed according to the normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$.
<code>x = 4 + 5 * randn(4, 3)</code>	<code>x =</code>	1.0584 14.9159 3.3180 4.5697	9.3338 4.2964 3.5218 -0.1617	5.4721 -2.6809 7.5716 12.1178	Matrix of 4 rows and 3 columns whose elements are distributed according to the normal distribution with mean $\mu = 4$ and variance $\sigma^2 = 25$.
<code>x = rand(1, 2000); hist(x, 10)</code>	 <p>A histogram titled "Histogram" showing the frequency distribution of 2000 random numbers. The x-axis is labeled from 0 to 1 with major ticks every 0.1. The y-axis is labeled from 0 to 250 with major ticks every 50. The distribution is uniform, with each of the 10 bins having approximately 200 counts.</p>				2000 random numbers uniformly distributed in $[0 \ 1]$ and the corresponding histogram.

(continued)

Commands	Results	Comments
$m = \text{mean}(x)$ $s2 = \text{var}(x)$ $x = \text{randn}(1, 2000);$ $\text{hist}(x, 10)$	$m = 0.5020$ $s2 = 0.0845$ 	Mean value and variance of the uniformly distributed random numbers. The estimated mean value of the samples approximates the true one which is given by $(a + b)/2 = 0.5$. The estimated variance also approximates the true variance which is given by $\sigma^2 = (b - a)^2/12 = 0.0833$.
$m = \text{mean}(x)$ $s2 = \text{var}(x)$	$m = 0.0012$ $s2 = 0.9783$	2000 random numbers distributed according to the normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$ and the corresponding histogram.

1.18 Solved Problems

Problem 1

- Create a vector $a = [0, 0.1, 0.2, \dots, 10]$ and a vector $b = [\cos(0), \cos(0.2), \cos(0.4), \dots, \cos(20)]$.
- Compute the following:
 - $c = a/b$
 - $d = a^4$
 - The dot product of a and b

Solution

```

a = 0:.1:10;
b = cos([0:.2:20]);
c = a./b;
d = a.^4;
prod = dot(a, b)

```

Problem 2

1. Create a 3×3 matrix \mathbf{A} whose elements are random numbers uniformly distributed in $[0 \ 1]$. Compute
 - a. The inverse matrix.
 - b. The transpose of matrix \mathbf{A} .
 - c. The determinant of \mathbf{A} .
 - d. The size of \mathbf{A} .
 - e. The second column of \mathbf{A} .
2. Create a sub-matrix \mathbf{B} with the elements of the first and third rows of \mathbf{A} .

Solution

1. $\mathbf{A} = \text{rand}(3)$
 - a. $\text{inv}(\mathbf{A})$
 - b. \mathbf{A}'
 - c. $\det(\mathbf{A})$
 - d. $\text{size}(\mathbf{A})$
 - e. $\mathbf{A}(:, 2)$
2. $\mathbf{B} = \mathbf{A}(1:2:3, :)$

Problem 3

- a. Create a function that accepts as input argument a number in radians and returns its value in degrees.
- b. Compute (through your function) how many degrees is $\pi/4$ radians.

Solution

- a.

```
function y = degtorad(x)
y = x*180/pi;
```
- b.

```
z = degtorad(pi/4);
% To verify your result execute the built-in MATLAB command deg2rad.m
```

Problem 4

- a. Create a function that plots the function $\text{sinc}(x) = \sin(\pi x)/(\pi x)$.
- b. Plot (through your function) $\text{sinc}(x)$ in the interval $[-2\pi, 2\pi]$.

Solution

```
a. function y=sink(x)
y=sin(pi*x)./(pi*x);
plot(x,y);
```

There is a small problem at $x = 0$. To see how to combat it open the built-in MATLAB command `sinc.m` by typing in the command prompt `open sinc`.

```
b. sink([-2*pi:.1:2*pi]);
```

Problem 5

- a. Write a function that accepts as input arguments a vector x and two scalars m and s and plots the function $g(x) = \frac{1}{s\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-m}{s})^2}$.
- b. Plot $y(x)$ for a vector x from -5 to 5 , $m=0$ and $s=1$.

Solution

```
a. function g=gau(x,m,s)
g=(1/(s*sqrt(2*pi)))*exp(-0.5*((x-m)/s).^2);
plot(x,g);
```

```
b. gau([-5:.1:5], 0,1)
```

Problem 6

- a. Write a function that accepts as input argument a complex number and returns as output
- o Its magnitude
 - o Its angle
 - o Its real part
 - o Its imaginary part
- b. Compute the above quantities for the complex numbers
- o i
 - o $-i$
 - o 1
 - o e^{3+4i}

Solution

```
a. function [mag, ang, re, im] = comple(x)
mag=abs(x)
ang=angle(x)
re=real(x)
im=imag(x)

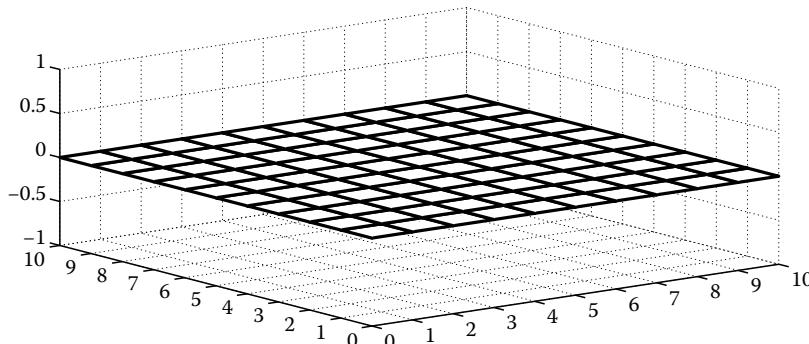
b. [a,b,c,d]=comple(i);
[a,b,c,d]=comple(-i);
[a,b,c,d]=comple(1);
[a,b,c,d]=comple(exp(3+4i));
```

Problem 7

- Generate a row vector of 500 linearly equally spaced points between 0 and 2π .
- Plot on the same figure in the interval $[0 \quad 2\pi]$ the functions $f(x) = xe^{-x}$, $0 \leq x \leq 2\pi$ and $y(x) = 2^{\cos(x)}$, $0 \leq x \leq 2\pi$.
- Add a title (of your choice) to the graph.
- Label the two axis of the graph.
- Insert a legend for all functions that appear in the graph.
- Plot the above functions at a second figure in two different subfigures.

Solution

```
x=linspace(0,2*pi,500);
f=x.*exp(-x);
y=2.^cos(x);
plot(x,f,x,y);
title('Two functions');
xlabel('x-axis');
ylabel('y-axis');
legend('f(x)', 'y(x)');
figure(2);
subplot(211);
plot(x,f);
subplot(212);
plot(x,y);
```

Problem 8

Plot the graph that is depicted above. Notice that $0 \leq x \leq 10$, $0 \leq y \leq 10$ and $z = 0$.

Solution

```
x=0:10;
y=x;
[X,Y]=meshgrid(x,y);
Z=zeros(size(X));
mesh(X,Y,Z)
```

Problem 9

- Compute the partial derivatives of the function $f(x, t) = \cos(x) + \sin(t) + e^t$.
- Calculate the integral $\int_0^\infty te^{-t} dt$.
- Calculate the double indefinite integral $\iint x^3 e^t dx dt$.
- Compute the summation $\sum_{k=0}^{\infty} \frac{x^k}{k!}$.
- Express in rational form the function $f(x) = \frac{3}{x+2} + \frac{x}{x^2+1}$.
- Find the solution of the equation $f(x) = x^3 + 2x^2 - x - 2$.

Solution

```
a. syms x t k w n
f = cos(x)+sin(t)+exp(t);
diff(f,x);
diff(f,t);

b. x=t*exp(-t);
int(x,t,0,inf)

c. f = (x^3)*exp(t);
a = int(f,x);
b = int(a,t);

d. f = (x^k) /'k!'
symsum(f,k,0,inf)

e. f = 3/(x+2)+x/(x^2+1);
[n,d] = numden(f);
f = n/d;

f. f = x^3+2*x^2-x-2
solve(f,x)
```

Problem 10 Compute and plot the solution of the differential equation $y''(t) + y(t) = 1$, $y(0) = 0, y'(0) = 0$.

Solution

```
syms y t
sol = dsolve('D2y+y=1','y(0)=0','Dy(0)=0','t');
t = -5:.1:5;
y = subs(sol,t);
plot(t,y)
```

Problem 11

- Create a vector a with elements from 1 to 100. Store the data of a in a file with filename *test.txt*.
- Create a vector b with elements from 100 to 1 and store the data of b in the file *test.txt* without deleting the data of a .

- c. Create a matrix \mathbf{B} of size 1×200 and save in \mathbf{B} the data of the file *test.txt*.
- d. Save matrix \mathbf{B} and vectors a and b in a file *matrix.mat* and next erase them from the memory.
- e. Retrieve only matrix \mathbf{B} from the file *matrix.mat* and load it in the memory.

Solution

```

a. a=1:100;
fid=fopen('test.txt','w');
fprintf(fid,'%f \n',a);
fclose(fid)

b. b=100:-1:1;
fid=fopen('test.txt','a');
fprintf(fid,'%f \n',b);
fclose(fid)

c. fid=fopen('test.txt','r');
B=fscanf(fid, '%f', [1,200]);
d. save matrix B a b
e. clear B a b
load matrix B

```

Problem 12

- a. Create a vector of 5000 random numbers distributed according to the normal distribution with mean $\mu = 5$ and variance $\sigma^2 = 3$.
- b. Plot the histogram.
- c. Compute the mean value and the variance of the samples.
- d. Create a vector of 5000 random numbers uniformly distributed in the interval from $a = 1$ to $b = 10$.
- e. Plot the histogram.
- f. Compute the mean value and the variance of the samples.

Solution

```

a. x=5+sqrt(3)*randn(5000,1);
b. hist(x,100)
c. mean(x)
    var(x)
d. x=1+9*rand(5000,1);
e. hist(x,10);
f. mean(x);
    var(x);

```

1.19 Homework Problems

1. Display the numbers 1 through 10 as long as their squares and their square roots.
2. Compute the sum of the squares of the numbers 1 through 10.
3. Create a vector with elements $[0, \frac{2}{4}, \frac{4}{6}, \frac{6}{8}, \dots, \frac{18}{20}]$.
4. Create the vector $x = [1, 2, \dots, 100]$. Assign the even numbers of x to a vector y .
5. Create the vector $x = [1, 2, \dots, 100]$. Assign the numbers that are multiples of 3 to a vector y .
6. Consider the vectors $add = a + b = [13 12 11 10 10]$ and $sub = a - b = [-11 -6 -14 8]$. Find the vectors a and b .
7. Create an algorithm that computes the product of two matrices according to Equation 1.3.
8. Plot the function $f(t) = te^{-t}$, $0 \leq t \leq 5$.
9. Consider the function $h(t) = \text{sinc}(t/T) \frac{\cos(\pi\beta t/T)}{1-(4\beta^2 t^2/T^2)}$. This function describes a raised cosine filter in the time domain. Suppose that $T = 1$ and $\beta = 0.5$ and plot $h(t)$ in the time interval $-5T \leq t \leq 5T$.
10. Split a figure into six subfigures and plot the function $h(t)$ described in the previous exercise combining each time the values $T = 1$ and $T = 3$ with $\beta = 0$, $\beta = 0.5$, and $\beta = 1$. Insert a title in each subfigure that describes the parameters T and β .
11. Read the contents of the matrix $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ row by row and store them in a vector.
12. Create a function that accepts two complex numbers as input arguments and returns their product and their division.
13. Create a function that accepts two numbers as input arguments and returns one with the larger absolute value. The function must return an error message if the number of arguments is not 2.
14. Create a function that accepts a vector as input argument and returns
 - a. The vector sorted in descending order
 - b. The number of the vector elements that are zero
 - c. The elements (in a new vector) that are greater than zero
15. A prime number is a natural number that has exactly two distinct natural number divisors: 1 and itself. The first two prime numbers are 1 and 2. Write a program that finds the prime numbers between 3 and 101. Tip: Use the command `rem` to check if the remainder of a division is zero.

16. Suppose that you have a text file with temperature measurements in degrees Fahrenheit. Write a program that reads data from this text file, converts the measurements in degrees Celsius, and stores the converted data in another text file.
17. Solve the following system of algebraic equations: $y = 1 - x^2$ and $y = 1 + x$.
18. Solve the following system of differential equations: $x'(t) = -y(t)$, $y'(t) = -x(t)$, $x(0) = 4$, $y(0) = 3$. Next, plot the functions $x(t)$ and $y(t)$ in the time interval $0 \leq t \leq 5$.

2

Signals

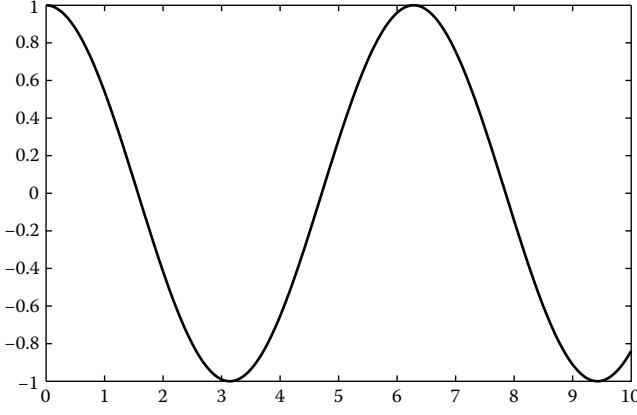
In this chapter, we introduce the concept of signals. Signals are classified according to their basic characteristics and properties. Furthermore, several elementary basic signals are introduced. A signal is defined as any natural quantity that varies according to one or more independent variables such as time or space. Time is usually the independent variable, but other variables like frequency can also be considered. Examples of signals are a sound, an image, an electrical current or voltage, a transmitted message, and many others. From a mathematical point of view, a signal is described by a function of one or more independent variables. According to the number of independent variables, a signal is characterized as a one-dimensional (1-D), a two-dimensional (2-D), or a multidimensional signal.

2.1 Categorization by the Variable Type

There are three main categories where a signal can be classified according to the type of the independent and dependent variables.

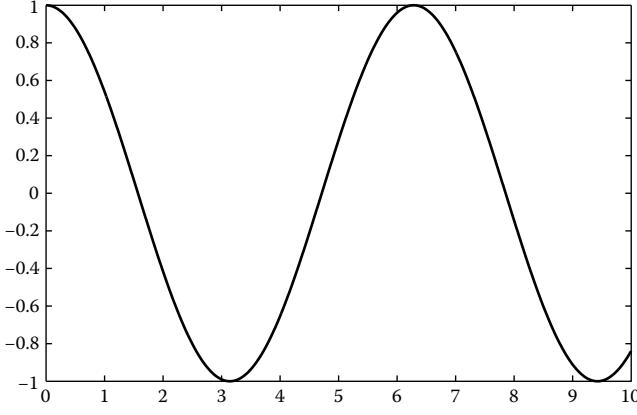
2.1.1 Continuous-Time Signals

A signal is called continuous-time (or analog) signal if the independent variable (time) is defined in a continuous interval. For 1-D signals, the domain of a signal is a continuous interval of the real axis. In other words, for continuous-time signals the independent variable t is continuous. Moreover, the dependent value that usually denotes the amplitude of the signal is also a continuous variable. An example of such a signal is speech as a function of time. An analog signal is expressed by a function $x(t)$, where t takes real values. Unfortunately, in MATLAB®, and generally on a computer, the work is done in discrete time. However, a continuous-time signal or function is approximated satisfactorily by using the corresponding discrete-time functions with very small time step. In the following example, the analog signal $y(t) = \cos(t)$, $0 \leq t \leq 10$ is defined and plotted.

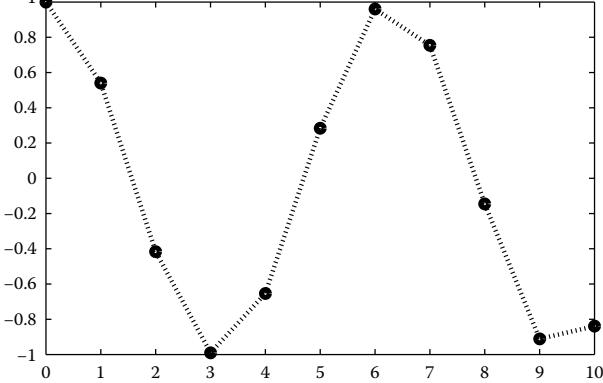
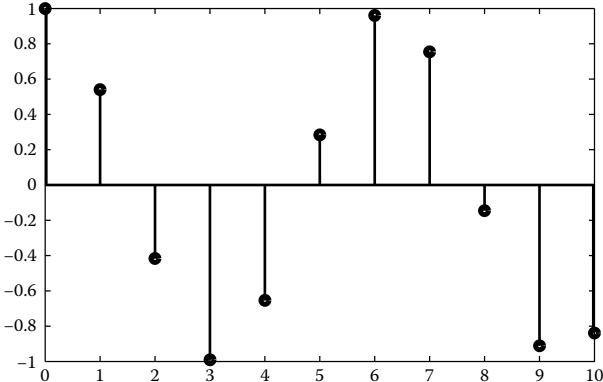
Commands	Results	Comments
$t = 0 : 0.01 : 10$ $y = \cos(t);$ $\text{plot}(t, y)$		Time (the independent variable t) is defined by using a very small step (time step = 0.01) in the continuous domain $0 \leq t \leq 10$. The dependent variable $y(t)$ is defined in the continuous set of values $-1 \leq y(t) \leq 1$. The analog signal is drawn by using the command <code>plot</code> .

2.1.2 Discrete-Time Signals

A signal is called a discrete-time signal if the independent variable (time) is defined in a discrete interval (e.g., the set of integer numbers), while the dependent variable is defined in a continuous set of values. In the following example, the discrete-time signal $y[n] = \cos[n]$ is plotted. Note that when referring to discrete time the variable n is typically used to represent the time.

Commands	Results	Comments
$n = 0 : 10$ $y = \cos(n);$		Discrete time n is defined with step 1. The dependent variable $y[n]$ is defined in the continuous set of values $-1 \leq y[n] \leq 1$.

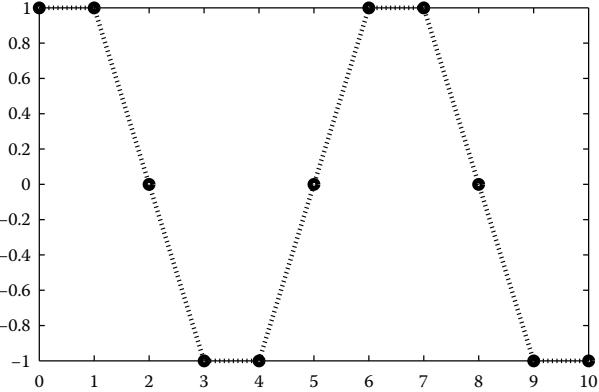
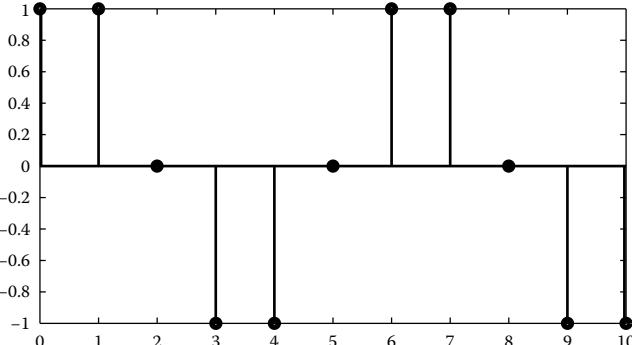
(continued)

Commands	Results	Comments
<code>plot(n, y, ':o')</code>		Graph of the discrete-time signal $y[n]$ by using the command <code>plot</code> with the proper syntax. The points determined by the circles are the values of $y[n]$.
<code>stem(n, y)</code>		Using the command <code>stem</code> is more appropriate when dealing with discrete-time signals.

A discrete-time signal $x[n]$ is usually obtained by *sampling* a continuous-time signal $x(t)$ at a constant rate. Suppose that T_s is the sampling period, that is every T_s s we sample the value of $x(t)$. Suppose also that $n \in \mathbb{Z}$, i.e., $n = 0, \pm 1, \pm 2, \dots$. The sequence of the samples $x[nT_s]$, $n \in \mathbb{Z}$ derived from the continuous-time signal $x(t)$ is sometimes called time series and denotes a discrete-time signal. The sampling period T_s is constant and thus can be omitted from the notation. In this book (see for example the two previous figures) we assume that $T_s = 1$.

2.1.3 Digital Signals

Digital signals are the signals that both independent and dependent variables take values from a discrete set. In the following example, the signal $y[n] = \cos[n]$ is again plotted, but we use the command `round` to limit the set of values that $y[n]$ can take. That is, $y[n]$ can be $-1, 0$, or 1 .

Commands	Results	Comments
$n = 0 : 10$ $y = \cos(n);$ $y = \text{round}(y);$ <code>plot(n, y, 'o')</code>		Discrete time n (independent variable) is defined with step 1. Definition of the dependent variable $y[n]$. The variable $y[n]$ is rounded toward the nearest integer; that is, it takes values from a discrete set.
<code>stem(n, y)</code>		Graph of the discrete-time signal by using the command <code>plot</code> with the proper syntax. The points determined by the circles are the values of the variable $y[n]$. Using the command <code>stem</code> is more appropriate also for digital signals.

In Section 2.2, we discuss continuous-time signals. The basic signals are introduced and the most common properties and categories are presented.

2.2 Basic Continuous-Time Signals

In this section, we present the basic continuous-time signals along with the way that they are implemented and plotted in MATLAB.

2.2.1 Sinusoidal Signals

The first basic category presented is that of sinusoidal signals. This type of signal is of the form $x(t) = A \cos(\Omega t + \theta)$, where Ω is the angular frequency, given in rad/s, A is the amplitude of the sinusoidal signal, and θ is the phase (in radians). Sinusoidal signals are periodic signals with fundamental period T given by $T = 2\pi/\Omega$ s. Finally, a useful quantity is the frequency f given in Hertz. Frequency f is defined by $f = 1/T$ or $f = \Omega/2\pi$.

Example

Plot the signal $x(t) = 3 \cos(3\pi t + \pi/3)$ in four periods.

First, the period T is calculated as $T = 2\pi/\Omega = 2\pi/3\pi = 2/3$. Hence, the MATLAB implementation is as follows.

Commands	Results	Comments
$A = 3;$		The amplitude of the signal is 3.
$\Omega = 3 * pi;$		The angular frequency is 3π .
$\theta = pi/3;$		The phase is $\pi/3$.
$T = 2 * pi / \Omega;$		The period is $2/3$.
$t = 0 : 0.01 : 4 * T;$		The time is defined from 0 to $4T$.
$x = A * cos(\Omega * t + \theta);$		Signal definition.
$plot(t, x)$		The signal is plotted in time of four periods.

When referring to sinusoidal signals we refer both to cosines and sines, as a cosine and a sine are in fact the same signal with a $\theta = \pi/2$ phase difference. In the figure below the signals $\cos(t)$ and $\sin(t + \pi/2)$ are plotted for time of one period.

Commands	Results	Comments
<pre>t = 0:0.1:2*pi; x1=cos(t); x2=sin(t+pi/2); plot(t,x1,t,x2,'o') xlim([0 2*pi])</pre>		<p>It is clear that the signals $\cos(t)$ and $\sin(t + \pi/2)$ are identical. The signal $\cos(t)$ is plotted with the solid line, while $\sin(t + \pi/2)$ is plotted with the circles. One can say that cosine is a sine with phase $\theta = \pi/2$.</p>

2.2.2 Exponential Signals

Exponential signals are signals of the form $x(t) = Ae^{bt}$. If $b > 0$, $x(t)$ is an increasing function while if $b < 0$, $x(t)$ is a decreasing function. At $t = 0$ the signal takes the value $x(0) = A$ as $e^{bt} = 1$.

Example

Plot the signals $x(t) = 3e^{0.4t}$ and $y(t) = 2e^{-0.9t}$ in the time interval $-2 \leq t \leq 5$.

Commands	Results/Comments
<pre>t = -2:.1:5; x = 3*exp(0.4*t); y = 2*exp(-0.9*t); plot(t,x,t,y,:'); legend('x(t)', 'y(t)')</pre>	

2.2.3 Complex Exponential Signals

Another signal highly associated with the sinusoidal signals is the complex exponential signal $Ae^{j\Omega t + \theta}$, which is also periodic with fundamental period given by $T = 2\pi/\Omega$. This is derived straightforwardly from Euler's formula:

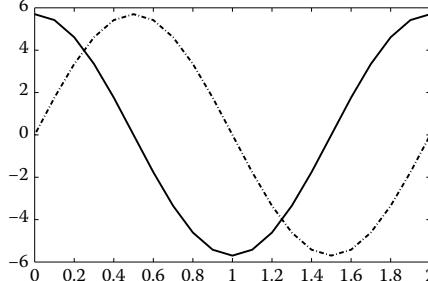
$$Ae^{j\Omega t + \theta} = A(\cos(\Omega t + \theta) + j \sin(\Omega t + \theta)). \quad (2.1)$$

From Equation 2.1, we conclude that $\text{Re}\{Ae^{j\Omega t + \theta}\} = A\cos(\Omega t + \theta)$ and $\text{Im}\{Ae^{j\Omega t + \theta}\} = A\sin(\Omega t + \theta)$, where $\text{Re}\{z\}$ is the real part and $\text{Im}\{z\}$ is the imaginary part of a complex number z .

Example

Plot the real and imaginary parts of the signal $y(t) = 2e^{j\pi t + \pi/3}$ in time of one period.

First, the period is calculated as $T = 2\pi/\Omega = 2\pi/\pi = 2$. Thus,

Commands	Results	Comments
<pre>t = 0:.1:2; y_re = real(2*exp(j*pi*t+pi/3)); y_im = imag(2*exp(j*pi*t+pi/3)); plot(t,y_re,t,y_im,'-.'');</pre> 		<p>Definition of the real and imaginary parts of the signal $y(t) = 2e^{j\pi t + \pi/3}$</p> <p>The real part of $y(t)$ is $2 \cos(\pi t + \pi/3)$ while the imaginary part of $y(t)$ is $2 \sin(\pi t + \pi/3)$.</p>

It is very easy to perform operations between complex exponential signals. For example, the product of two complex exponential signals is easily computed by simply adding their exponents. More precisely, if $y_1(t) = Ae^{j\Omega_1 t}$ and $y_2(t) = Be^{j\Omega_2 t}$ then $y_3(t) = y_1(t) \cdot y_2(t) = A \cdot B \cdot e^{j(\Omega_1 + \Omega_2)t}$.

Example

Plot the real parts of the signals $x(t) = 2e^{j\pi t}$ $3e^{j2\pi t}$ and $y(t) = 6e^{j3\pi t}$.

Commands	Results	Comments
<pre>t = 0:0.1:5; x = (2*exp(j*pi*t)).*(3*exp(j*2*pi*t)) y=6*exp(j*3*pi*t);plot(t,real(x),t, real(y)',ko')</pre>		The real parts of $x(t)$ and $y(t)$ are the same.

The signals $Ae^{j\Omega t+\theta}$, $A \cos(\Omega t + \theta)$, and $A \sin(\Omega t + \theta)$ have a key role in signal processing as any periodic signal can be expressed as a sum of single frequency signals. We will discuss this point further in a Chapter 5.

2.2.4 Unit Step Function

Another basic signal is the unit step function $u(t)$. The unit step function is given by

$$u(t) = \begin{cases} 1, & t > 0 \\ \frac{1}{2}, & t = 0 \\ 0, & t < 0 \end{cases}. \quad (2.2)$$

However, $u(t)$ is a continuous-time signal; thus the value at $t=0$ can be omitted for convenience and $u(t)$ can be defined as

$$u(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases}. \quad (2.3)$$

The MATLAB command that generates the unit step function is the command `heaviside(t)`.

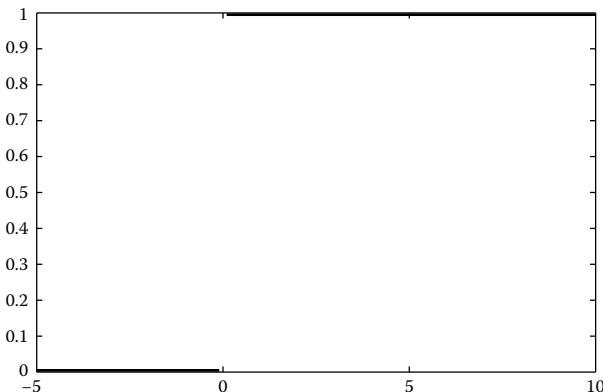
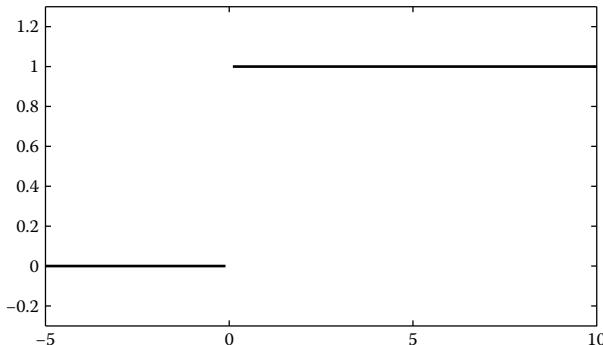
According to MATLAB programmers, unit step function is given by

$$u(t) = \begin{cases} 1, & t > 0 \\ 0, & t \leq 0 \end{cases}, \quad (2.4)$$

i.e., it is not defined at $t=0$. In the following example three different methods of defining and plotting the unit step function are presented.

First Method

With use of the command `heaviside`.

Commands	Results	Comments
<code>t = -5:0.1:10;</code>	<code>t = -5, . . . , -0.1, 0, 0.1, . . . , 10</code>	Definition of the time interval $-5 \leq t \leq 10$.
<code>u = heaviside(t)</code>	<code>u = 0, . . . , 0, NaN, 1, . . . , 1</code>	Definition of $u(t)$.
<code>plot(t, u)</code>		Graph of $u(t)$.
<code>ylim([-0.3 1.3])</code>		Change of axis for better appearance of the graph.

Second Method

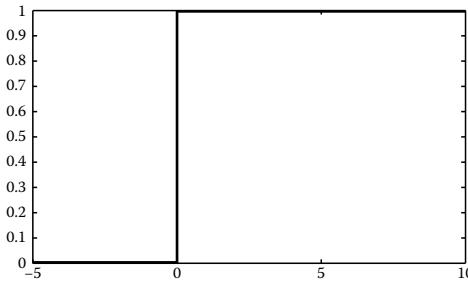
Use of the technique of defining and plotting piecewise functions.

Commands	Results	Comments
<code>t1 = -5:.1:0</code>	<code>t1 = -5, -4.9, . . . , 0</code>	Definition of the first time interval $-5 \leq t \leq 0$.
<code>t2 = 0:.1:10</code>	<code>t2 = 0, 0.1, . . . , 10</code>	Definition of the second time interval $0 \leq t \leq 10$.
<code>u1 = zeros(size(t1))</code>	<code>u1 = 0, 0, . . . , 0</code>	Implementation of the part of $u(t)$ that corresponds to time $t1$.
<code>u2 = ones(size(t2));</code>	<code>u2 = 1, 1, . . . , 1</code>	Implementation of the part of $u(t)$ that corresponds to time $t2$.

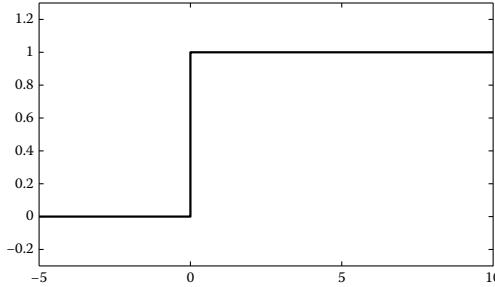
(continued)

(continued)

Commands	Results	Comments
<code>t = [t1 t2];</code>	$t = -5, \dots, -0.1, 0, \dots, 10$	Concatenation of the two time vectors.
<code>u = [u1 u2];</code>	$u = 0, \dots, 0, 1, \dots, 1$	Concatenation of the two function vectors.



Graph of the unit step function $u(t)$ in the time interval $0 \leq t \leq 10$.



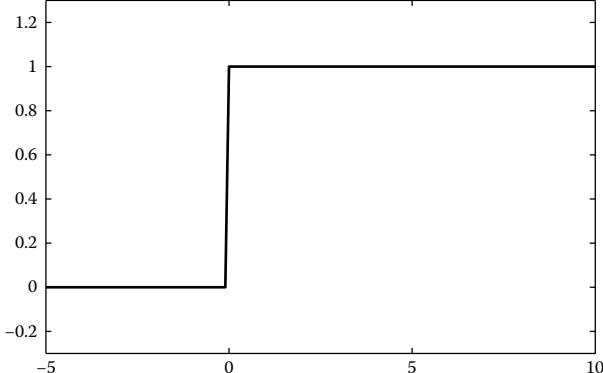
Change of axis for better appearance of the graph.

Third Method

Implementation with specific number of zeros and ones.

Commands	Results	Comments
<code>t = -5:.1:10;</code>	$t = -5, \dots, -0.1, 0, \dots, 10$	Time definition.
<code>u = [zeros(1,50) ones(1,101)];</code>	$u = 0, \dots, 0, 1, \dots, 1$	The vector t consists of 151 elements. Thus, the first 50 elements of t are matched with zeros while the next 101 elements (including $t=0$) of t are matched with ones. The two vectors are concatenated.

(continued)

Commands	Results	Comments
<pre>plot(t,u); ylim([-0.3 1.3])</pre>		Graph of the unit step function $u(t)$ in the time interval $0 \leq t \leq 10$.

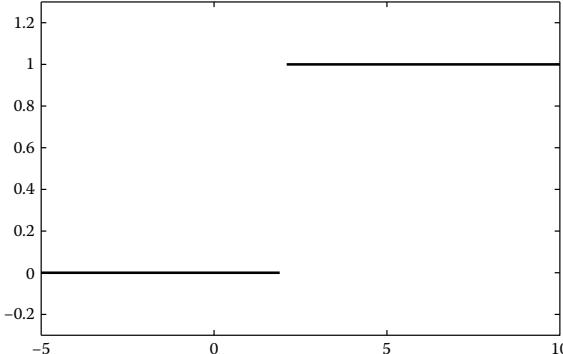
The general form of the unit step function is

$$u(t - t_0) = \begin{cases} 1, & t - t_0 \geq 0 \Rightarrow t \geq t_0 \\ 0, & t - t_0 < 0 \Rightarrow t < t_0 \end{cases}. \quad (2.5)$$

Suppose that we want to define and plot the unit step function for $t_0 = 2$, i.e., we want to define and plot the function $u(t - 2)$.

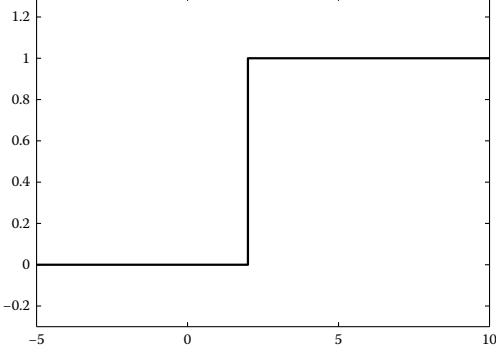
First Method

With use of the command `heaviside`.

Commands	Results	Comments
<pre>t = -5:0.1:10; u = heaviside(t-2) plot(t,u) ylim([-0.3 1.3])</pre>		Definition of the time interval $-5 \leq t \leq 10$. Definition of $u(t - 2)$. Graph of $u(t - 2)$. Notice that function $u(t - 2)$ becomes 1 from the time instance $t = 2$ and afterward. Thus, $u(t - 2)$ is a shifted by 2 units to the right version of $u(t)$.

Second Method

Use of the technique of defining multipart functions.

Commands	Results	Comments
<code>t1=-5:.1:2</code>	<code>t1 = -5, -4.9, . . . , 2</code>	Definition of the first time interval $-5 \leq t \leq 2$.
<code>t2=2:.1:10</code>	<code>t2 = 2, 0.1, . . . , 10</code>	Definition of the second time interval $2 \leq t \leq 10$.
<code>u1=zeros(size(t1));</code>	<code>u1 = 0, 0, . . . , 0</code>	Implementation of the vector of $u(t)$ that corresponds to time t_1 .
<code>u2=ones(size(t2));</code>	<code>u2 = 1, 1, . . . , 1</code>	Implementation of the vector of $u(t)$ that corresponds to time t_2 .
<code>t=[t1 t2];</code>		Concatenation of the two time vectors.
<code>u=[u1 u2];</code>		Concatenation of the two function vectors.
<code>plot(t,u) ylim([-0.3 1.3])</code>		Graph of the signal $u(t-2)$.

Finally, we mention that the command `heaviside` can be also used with symbolic variables.

Commands	Results	Comments
<code>syms t</code>		Definition of t as a symbolic variable.
<code>u=heaviside(t)</code>	<code>u=heaviside(t)</code>	Definition of $u(t)$ as a symbolic expression.
<code>diff(u,t)</code>	<code>ans = dirac(t)</code>	The derivative of $u(t)$ is the Dirac delta function $\delta(t)$.

2.2.5 Unit Impulse or Dirac Delta Function

The Dirac function $\delta(t)$, strictly speaking, is not a function but is defined through its properties. The main property is

$$\int_{-\infty}^{\infty} f(t)\delta(t) = f(0), \quad (2.6)$$

where $f(\cdot)$ is an arbitrary function. Suppose that $f(t) = 1$, $t \in (-\infty, \infty)$. Then (2.6) becomes

$$\int_{-\infty}^{\infty} \delta(t) = 1. \quad (2.7)$$

For practical reasons, $\delta(t)$ can be loosely defined as a function that is infinite at $t=0$ and zero elsewhere. This is the way that $\delta(t)$ is implemented from the MATLAB programmers. The mathematical expression is

$$\delta(t) = \begin{cases} \infty, & t = 0 \\ 0, & t \neq 0 \end{cases}. \quad (2.8)$$

An alternative definition for the Dirac function that is usually applicable when dealing with discrete-time signals is given now. In this case, we refer to $\delta(t)$ as the *Delta* or the *Kronecker* function. The mathematical definition of the delta function is

$$\delta(t) = \begin{cases} 1, & t = 0 \\ 0, & t \neq 0 \end{cases}. \quad (2.9)$$

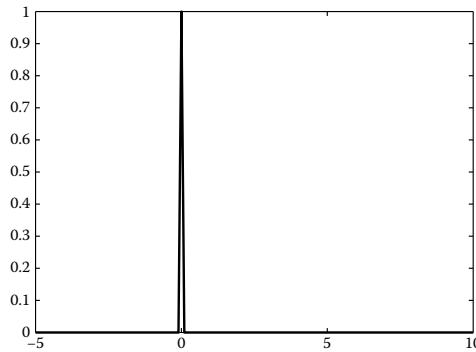
Note that if the definition given by (2.9) is used, the graph of $\delta(t)$ we get in MATLAB is closer to the one met at the theory of signals and systems. The MATLAB implementation of $\delta(t)$ is as follows.

Commands	Results	Comments
<code>t1 = -5:.1:-0.1;</code>		Definition of the first time interval $-5 \leq t < 0$.
<code>t2 = 0;</code>		The second time interval is defined only for one time instance, namely for $t = 0$.
<code>t3 = 0.1:.1:10;</code>		Definition of the third time interval $0 < t \leq 10$.
<code>d1 = zeros(size(t1));</code>		Implementation of the part of $\delta(t)$ that corresponds to time $t1$.
<code>d2 = 1;</code>		Implementation of the part (vector of one element) of $\delta(t)$ that corresponds to time $t2$.

(continued)

(continued)

Commands	Results	Comments
<code>d3=zeros(size(t3));</code>		Implementation of the part of $\delta(t)$ that corresponds to time $t3$.
<code>t=[t1 t2 t3];</code>		Concatenation of the three time vectors.
<code>D=[d1 d2 d3];</code>		Concatenation of the three function vectors.

`plot(t,d)`Graph of the Dirac function $\delta(t)$.

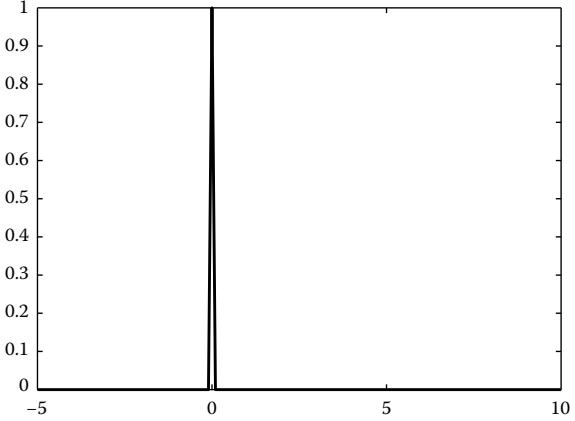
The signal defined according to (2.9) can be plotted with the help of the command `gauspuls(t)`.

Remark

The command `gauspuls` is not created in MATLAB for that purpose but it is convenient for us to use it in the special way that is presented in this book.

Commands	Results	Comments
<code>t=-5:.1:10;</code>		Definition of time.
<code>s=gauspuls(t)</code>		Definition of the unit pulse.

(continued)

Commands	Results	Comments
<code>plot(t,s)</code>		The two graphs are identical.

In case that the first definition of the Dirac function is used, i.e.,

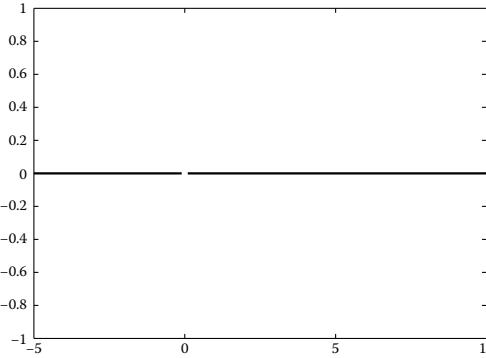
$$\delta(t) = \begin{cases} \infty, & t = 0 \\ 0, & t \neq 0 \end{cases}$$

the MATLAB implementation is

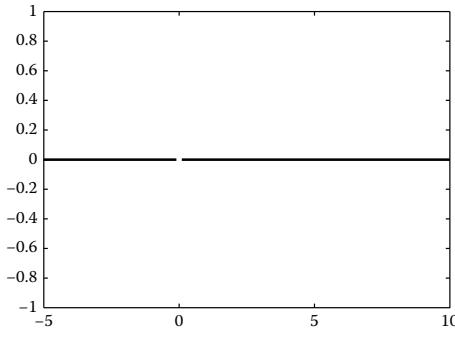
Commands	Results	Comments
<code>t = -5:.1:10</code>	<code>t = -5, . . . , -0.1, 0, 0.1, . . . , 10</code>	Definition of the time interval.
<code>d = [zeros(1,50) inf zeros(1,100)]; u = 0, . . . , 0, inf, 0, . . . , 0</code>		The vector t consists of 151 elements. Thus, the first 50 elements of t are matched with zeros, the element with index 51 (that corresponds to $t=0$) is matched with inf , while the next 100 elements of t are matched again with zeros. The three vectors are concatenated.

(continued)

(continued)

Commands	Results	Comments
<code>plot(t,d)</code>		Graph of $\delta(t)$. Notice that at the time instance $t = 0$ there is a gap in the graph that denotes ∞ .

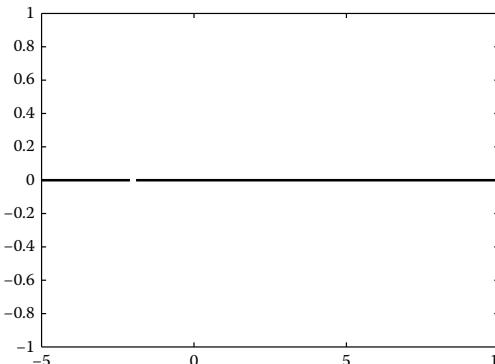
The MATLAB command that defines the Dirac function is the command `dirac(t)`. The command `dirac` is demonstrated in the following example.

Commands	Results	Comments
<code>t = -5:.1:10</code>	<code>t = -5, . . . -0.1, 0, 0.1, 10</code>	Definition of the time interval $-5 \leq t \leq 10$.
<code>d = dirac(t);</code>	<code>d = 0, 0, inf, 0, 0</code>	Definition of $\delta(t)$.
<code>plot(t,d)</code>		Graph of $\delta(t)$. The obtained graph is similar to that of the previous example.

The general form of the Dirac function is

$$\delta(t - t_0) = \begin{cases} \infty, & t = t_0 \\ 0, & t \neq t_0 \end{cases}. \quad (2.10)$$

Suppose that we want to define and plot the Dirac function for $t_0 = -2$, i.e., we want to define and plot the function $\delta(t + 2)$.

Commands	Results	Comments
<code>t = -5:0.1:10;</code>		Definition of the time interval $-5 \leq t \leq 10$.
<code>d=dirac(t+2)</code>		Definition of $\delta(t + 2)$.
<code>plot(t,d)</code>		Graph of $\delta(t + 2)$. The function $\delta(t + 2)$ becomes infinite at $t = -2$. The signal $\delta(t)$ is shifted by 2 units to the left.

Finally, we mention that the `dirac` command can be also used with symbolic variables.

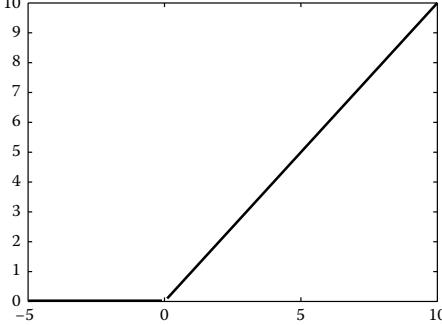
Commands	Results	Comments
<code>syms t</code>		Definition of the symbolic variable t .
<code>d=dirac(t)</code>	<code>d=dirac(t)</code>	Definition of $\delta(t)$.
<code>int(d,t,-inf,inf)</code>	<code>ans = 1</code>	Confirmation of Equation 2.7, i.e., $\int_{-\infty}^{\infty} \delta(t) = 1$.

2.2.6 Ramp Function

The unit-ramp function $r(t)$ is defined in terms of the unit step function $u(t)$ as

$$r(t) = t \cdot u(t) = \begin{cases} t, & t \geq 0 \\ 0, & t < 0 \end{cases}. \quad (2.11)$$

Therefore, in order to define the ramp function, first we have to construct the unit step function.

Commands	Results	Comments
$t = -5:0.1:10;$		Definition of the time interval $-5 \leq t \leq 10$.
$r = t .* \text{heaviside}(t);$		Definition of the ramp function $r(t) = t \cdot u(t)$.
$\text{plot}(t, r)$		Graph of $r(t)$.

An alternative way to construct the ramp function is by using the technique of piecewise functions.

Commands	Results	Comments
$t1 = -5:-0.1;$		Definition of the first time interval $-5 \leq t < 0$.
$t2 = 0:1:10;$		Definition of the second time interval $0 \leq t \leq 10$.
$r1 = \text{zeros}(\text{size}(t1));$		The first part of $r(t)$ that corresponds to time $t1$ is constructed.
$r2 = t2;$		The second part of $r(t)$ that corresponds to time $t2$ is constructed.
$t = [t1 t2];$		Time concatenation.

(continued)

Commands	Results	Comments
<pre>r = [r1 r2]; plot(t, r)</pre>		Function concatenation. Graph of the ramp function $r(t)$.

The general form of the (unit) ramp function is

$$r(t - t_0) = (t - t_0)u(t - t_0) = \begin{cases} t - t_0, & t \geq t_0 \\ 0, & t < t_0 \end{cases}. \quad (2.12)$$

Suppose that we want to define and plot the ramp function for $t_0 = 1$, i.e., we want to define and plot the function $r(t - 1)$. The MATLAB implementation is

Commands	Results	Comments
<pre>t = -5:.1:10; r = (t-1).*heaviside(t-1) plot(t, r)</pre>		Definition of the first time interval $-5 \leq t \leq 10$. The function $r(t - 1)$ is defined as $r(t - 1) = (t - 1)u(t - 1)$. Graph of $r(t - 1)$. The function $r(t - 1)$ is the function $r(t)$ shifted by 1 unit to the right.

Finally, it is noted that the derivative of the unit-ramp function $r(t)$ is the unit step function $u(t)$, or equivalently the integral of $u(t)$ equals $r(t)$.

Commands	Results	Comments
<code>syms t</code>		Definition of the symbolic variable t .
<code>u=heaviside(t)</code>	<code>u=heaviside(t)</code>	Definition of $u(t)$.
<code>int(u,t)</code>	<code>ans = heaviside(t)*t</code>	$\int u(t)dt = tu(t) = r(t)$.

2.2.7 Rectangular Pulse Function

The rectangular pulse function $pT(t)$ is a rectangular pulse with unit amplitude and duration T . It is defined in terms of the unit step function $u(t)$ as

$$pT(t) = u\left(t + \frac{T}{2}\right) - u\left(t - \frac{T}{2}\right) = \begin{cases} 1, & -T/2 \leq t \leq T/2 \\ 0, & \text{elsewhere} \end{cases}. \quad (2.13)$$

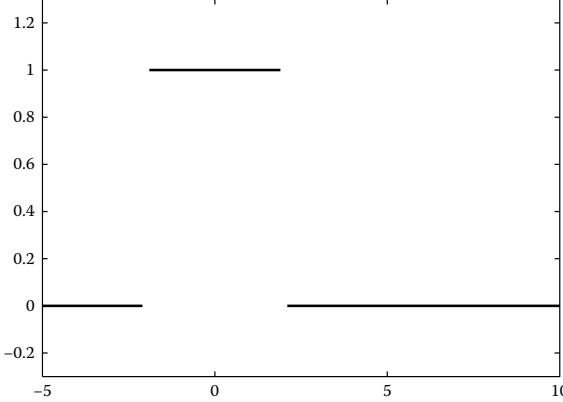
Thus, for the implementation of a rectangular pulse function, two unit step functions have to be first defined. Suppose that the rectangular pulse has duration $T=4$, namely the signal is $p4(t)$. Substituting the term $T=4$ in (2.13) yields

$$p4(t) = u\left(t + \frac{4}{2}\right) - u\left(t - \frac{4}{2}\right) = u(t+2) - u(t-2). \quad (2.14)$$

The MATLAB implementation is

Commands	Results	Comments
<code>t=-5:.1:10;</code>	<code>t = -5, . . . , -2, -1.9, . . . , 2, 2.1, . . . , 10</code>	Definition of the time interval $-5 \leq t \leq 10$.
<code>u1=heaviside(t+2); u1=0, . . . , NaN, 1, . . . , 1, 1, . . . , 1</code>		Definition of $u(t+2)$.
<code>u2=heaviside(t-2); u2=0, . . . , 0, 0, . . . , NaN, 1, . . . , 1</code>		Definition of $u(t-2)$.
<code>p=u1-u2;</code>	<code>p = 0, . . . , NaN, 1, . . . , NaN, 0, . . . , 0</code>	The rectangular pulse is built according to (2.14).

(continued)

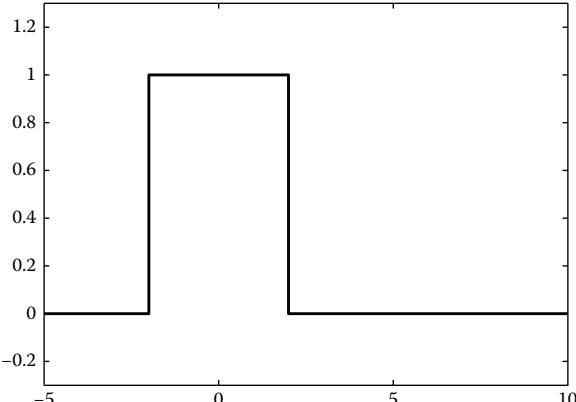
Commands	Results	Comments
<pre>plot(t,p) ylim([-0.3 1.3])</pre>	 The graph shows a rectangular pulse centered at $t=0$ with a width of $T=4$. The signal is zero for $t \in [-5, -2]$, reaches a maximum value of 1 for $t \in (-2, 2)$, and is zero again for $t \in (2, 10]$.	Graph of the rectangular pulse.

The signal $p4(t)$ is a rectangular pulse of unit magnitude and duration $T=4$, centered at $t=0$. A second way that can be used is to define the signal as a three-part function. Thus, in MATLAB we type

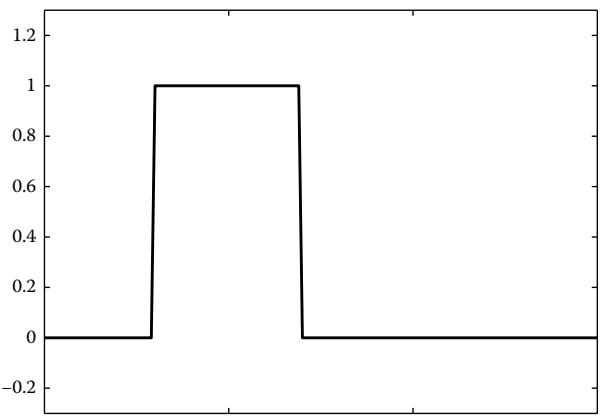
Commands	Results	Comments
<code>t1 = -5:.1:-2;</code>		Definition of the first time interval $-5 \leq t \leq -2$.
<code>t2 = -2:.1:2;</code>		Definition of the second time interval $-2 \leq t \leq 2$.
<code>t3 = 2:.1:10;</code>		Definition of the third time interval $2 \leq t \leq 10$.
<code>p1=zeros(size(t1));</code>		Function $pT(t)$ is zero for $-5 \leq t \leq -2$.
<code>p2=ones(size(t2));</code>		Function $pT(t)$ is one for $-2 \leq t \leq 2$.
<code>p3=zeros(size(t3));</code>		Function $pT(t)$ is zero for $2 \leq t \leq 10$.
<code>t = [t1 t2 t3];</code>		Time concatenation.

(continued)

(continued)

Commands	Results	Comments
<pre>p = [p1 p2 p3] ; plot(t,p); ylim([-0.3 1.3]);</pre>		Function concatenation. Graph of the signal and modification of the <i>y</i> -axis for better appearance.

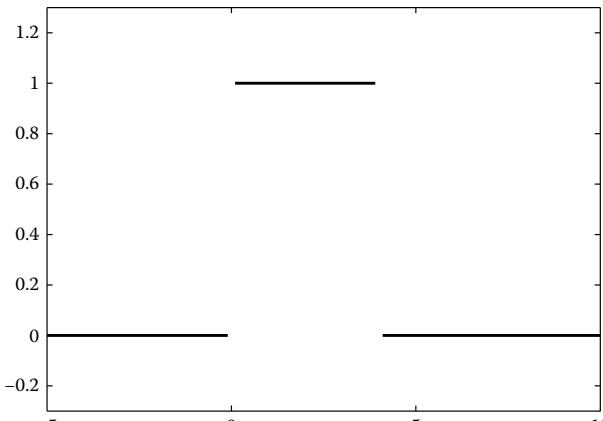
Finally, a third way to create a rectangular pulse signal is by using the command `rectpuls`. Using the syntax `rectpuls(t)` we get a rectangular pulse of duration $T=1$. Typing `rectpuls(t, a)` results in a rectangular pulse of duration $T=a$. Therefore, in order to define the signal $p_4(t)$, one types

Commands	Results	Comments
<pre>t = -5:.1:10; s = rectpuls(t, 4);</pre>		Time definition. Rectangular pulse of width $T=4$.

Suppose now that the signal $p_4(t - 2)$ is needed. According to the definition of rectangular pulse given in (2.13), we get

$$p4(t-2) = u\left(t - 2 + \frac{4}{2}\right) - u\left(t - 2 - \frac{4}{2}\right) = u(t) - u(t-4). \quad (2.15)$$

Therefore, in MATLAB we type

Commands	Results	Comments
<code>t = -5:.1:10</code>		Definition of the time interval $-5 \leq t \leq 10$.
<code>u1 = heaviside(t)</code>		Definition of $u(t)$.
<code>u2 = heaviside(t-4)</code>		Definition of $u(t-4)$.
<code>p = u1-u2</code>		The rectangular pulse is defined according to Equation 2.15.
<code>plot(t, p)</code> <code>ylim([-0.3 1.3])</code>		Graph of $p4(t-2)$.

We observe that the signal $p4(t-2)$ is similar to $p4(t)$ but shifted by 2 units to the right. Hence, by changing appropriately the time, a signal can be shifted to the left or to the right on the horizontal axis.

2.3 Discrete-Time Signals

In this section, we discuss a different category of signals: the discrete-time signals. Discrete-time signals are sequences. A sequence $x[n]$, $n \in (-\infty, \infty)$ consists of infinite (real or complex) elements or samples. Of course, in MATLAB we deal mostly with definite-time signals, i.e., with sequences of the form $x[n]$, $n_1 \leq n \leq n_2$. Discrete-time signals are treated in a similar way to the one presented for the continuous-time signals. However, there are two main differences:

1. The definition of the time. For discrete-time signals, time is defined by using `step` 1.
2. The graph of discrete-time signals is obtained by using the `stem` command. `stem` is similar to `plot` but is suitable for discrete-time signals.

2.3.1 Unit Impulse Sequence

The unit impulse sequence $\delta[n]$ is the counterpart of the Dirac delta function when dealing with discrete-time signals. It also known as Kronecker delta. The mathematical expression of $\delta[n]$ is

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}. \quad (2.16)$$

In the general case, the unit impulse sequence is given by

$$\delta[n - n_0] = \begin{cases} 1, & n = n_0 \\ 0, & n \neq n_0 \end{cases}. \quad (2.17)$$

In order to define and plot $\delta[n]$ there are numerous options. The first is by using the command `gauspuls`.

Commands	Results	Comments																
<pre>n=-3:3 d=gauspuls(n)</pre>	<table border="0"> <tr> <td style="padding-right: 20px;"><code>n =</code></td><td>-3</td><td>-2</td><td>-1</td><td>0</td><td>1</td><td>2</td><td>3</td></tr> <tr> <td><code>d =</code></td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr> </table>	<code>n =</code>	-3	-2	-1	0	1	2	3	<code>d =</code>	0	0	0	1	0	0	0	Definition of the discrete-time vector (with step 1) and of the unit impulse sequence $\delta[n]$.
<code>n =</code>	-3	-2	-1	0	1	2	3											
<code>d =</code>	0	0	0	1	0	0	0											
<pre>stem(n,d)</pre>		The unit impulse sequence graph is implemented with the command <code>stem</code> .																
<pre>d=gauspuls(n-2); stem(n,d)</pre>		Definition and graph of $\delta[n - 2]$. The sequence $\delta[n - 2]$ is shifted by 2 units to the right compared to $\delta[n]$.																

A second (more analytical) way is given now. The unit impulse sequence is considered as a three-branched (discrete time) function.

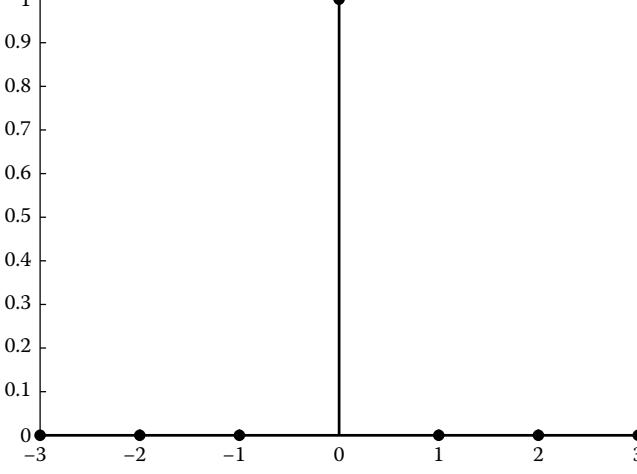
Commands	Results	Comments
<pre>n1 = -3:-1; n2 = 0; n3 = 1:3; n = [n1 n2 n3]</pre>	<pre>n = -3 -2 -1 0 1 2 3</pre>	The discrete time intervals are separately defined and then concatenated.
<pre>d1=zeros(size(n1)); d2=1; d3=zeros(size(n3)); d = [d1 d2 d3]</pre>	<pre>d = 0 0 0 1 0 0 0</pre>	The unit impulse sequence $\delta[n]$ is zero for $-3 \leq n \leq -1$, one for $n=0$, and zero for $1 \leq n \leq 3$.
<pre>stem(n,d)</pre>		Graph of $\delta[n]$.

A third way that can be used in order to obtain $\delta[n]$ is to use the statement $x = (n == 0)$, where n is the time vector and x is the signal. This statement returns one if $n=0$ and zero if $n \neq 0$.

Commands	Results/Comments
<pre>n1 = -3; n2 = 3; n = n1:n2 d = (n == 0)</pre>	<pre>n = -3 -2 -1 0 1 2 3 d = 0 0 0 1 0 0 0</pre>

(continued)

(continued)

Commands	Results/Comments
<code>stem(n,d)</code>	

2.3.2 Unit Step Sequence

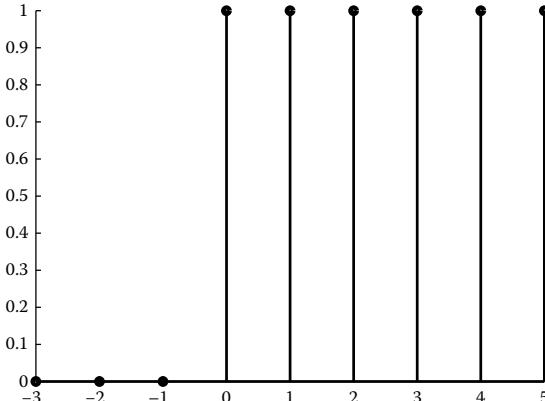
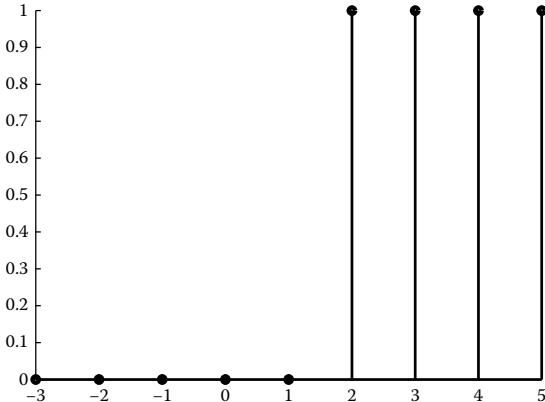
The unit step sequence $u[n]$ is the counterpart of the unit step function when dealing with discrete-time signals. The mathematical expression is

$$u[n] = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}. \quad (2.18)$$

In the general case, the unit step sequence is given by

$$u[n - n_0] = \begin{cases} 1, & n \geq n_0 \\ 0, & n < n_0 \end{cases}. \quad (2.19)$$

Due to the fact that $u[n]$ and $u[n - n_0]$ are 1 at $n=0$ and $n=n_0$, respectively, the use of the command `heaviside` is not possible. The solution is to treat $u[n]$ as a two-part discrete-time function.

Commands	Results	Comments
<pre>n1 = -3:-1; n2 = 0:5; n = [n1 n2]; u1=zeros(size(n1)); u2=ones(size(n2)); u = [u1 u2]; stem(n,u)</pre>		Graph of $u[n]$ over the (discrete) time interval $-3 \leq n \leq 5$.
<pre>n1 = -3:1; n2 = 2:5; n = [n1 n2]; u1=zeros(size(n1)); u2=ones(size(n2)); u2=ones(size(n2)); u = [u1 u2]; stem(n,u)</pre>		Graph of $u[n - 2]$ over the time interval $-3 \leq n \leq 5$.

An alternative way of defining a unit step sequence $u[n - n_0]$ is to use one of the statements $x = (n \geq n_0)$ or $x = (n - n_0) \geq 0$, where n is the time vector, x is the unit step signal, and n_0 is the point in which the signal turns from zero to one.

Commands	Results	Comments
<pre>n=-3:5 n0=0; u=(n>=n0) stem(n,u)</pre>		Graph of $u[n]$ over the time interval $-3 \leq n \leq 5$.
<pre>n=-3:5 n0=2; u=((n-n0)>=0) stem(n,u)</pre>		Graph of $u[n-2]$ over the time interval $-3 \leq n \leq 5$.

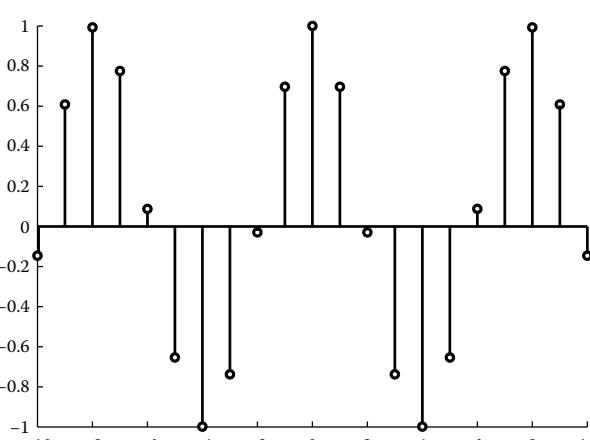
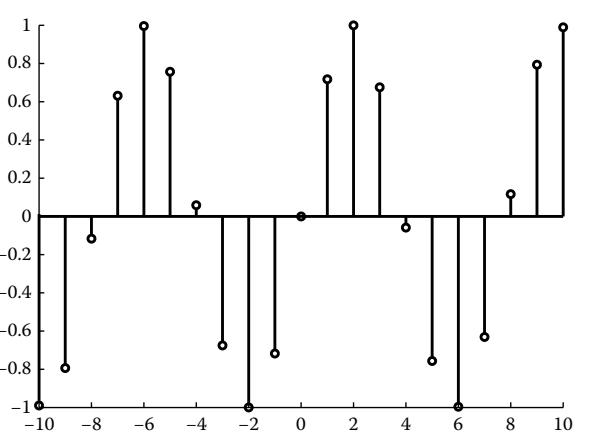
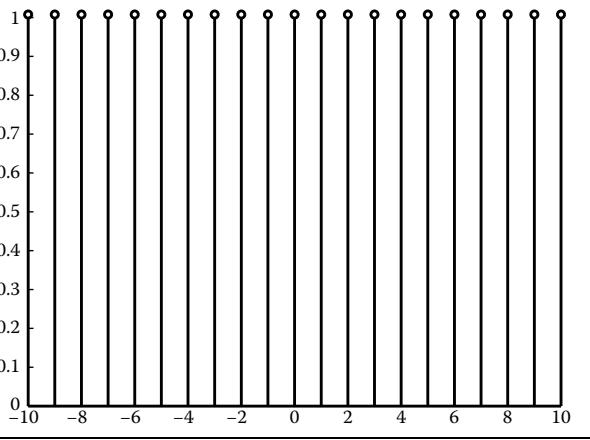
2.3.3 Real Exponential Sequence

The mathematical expression of a real-valued exponential sequence is $x[n]=a^n$, $a \in \mathbb{R}$. The sequence $x[n]$ is ascending if $|a| > 1$, and descending if $|a| < 1$.

Commands	Results	Comments																				
<pre>n = -3 : 5 a1 = 0.8; x1 = a1.^n; stem(n, x1);</pre>	<table border="1"> <caption>Data for x[n] = 0.8^n</caption> <thead> <tr> <th>n</th> <th>x[n]</th> </tr> </thead> <tbody> <tr><td>-3</td><td>0.000</td></tr> <tr><td>-2</td><td>0.160</td></tr> <tr><td>-1</td><td>0.128</td></tr> <tr><td>0</td><td>0.100</td></tr> <tr><td>1</td><td>0.080</td></tr> <tr><td>2</td><td>0.064</td></tr> <tr><td>3</td><td>0.051</td></tr> <tr><td>4</td><td>0.040</td></tr> <tr><td>5</td><td>0.032</td></tr> </tbody> </table>	n	x[n]	-3	0.000	-2	0.160	-1	0.128	0	0.100	1	0.080	2	0.064	3	0.051	4	0.040	5	0.032	Graph of the real-valued exponential sequence $x[n] = 0.8^n$, $-3 \leq n \leq 5$.
n	x[n]																					
-3	0.000																					
-2	0.160																					
-1	0.128																					
0	0.100																					
1	0.080																					
2	0.064																					
3	0.051																					
4	0.040																					
5	0.032																					
<pre>a2 = 1.2; x2 = a2.^n; stem(n, x2);</pre>	<table border="1"> <caption>Data for x[n] = 1.2^n</caption> <thead> <tr> <th>n</th> <th>x[n]</th> </tr> </thead> <tbody> <tr><td>-3</td><td>0.000</td></tr> <tr><td>-2</td><td>0.064</td></tr> <tr><td>-1</td><td>0.080</td></tr> <tr><td>0</td><td>0.100</td></tr> <tr><td>1</td><td>0.120</td></tr> <tr><td>2</td><td>0.144</td></tr> <tr><td>3</td><td>0.1728</td></tr> <tr><td>4</td><td>0.20736</td></tr> <tr><td>5</td><td>0.248832</td></tr> </tbody> </table>	n	x[n]	-3	0.000	-2	0.064	-1	0.080	0	0.100	1	0.120	2	0.144	3	0.1728	4	0.20736	5	0.248832	Graph of the real-valued exponential sequence $x[n] = 1.2^n$, $-3 \leq n \leq 5$.
n	x[n]																					
-3	0.000																					
-2	0.064																					
-1	0.080																					
0	0.100																					
1	0.120																					
2	0.144																					
3	0.1728																					
4	0.20736																					
5	0.248832																					

2.3.4 Complex Exponential Sequence

A complex exponential sequence is a signal of the form $x[n] = e^{j\omega n}$, or alternatively (from Euler's formula), is a signal of the form $x[n] = \cos(\omega n) + j\sin(\omega n)$.

Commands	Results	Comments
<pre>n=-10:10; w=0.8; x=exp(j*w*n); stem(n,real(x));</pre>		Definition of the complex exponential sequence $x[n] = e^{j0.8n}$, $-10 \leq n \leq 10$ and graph of its real part.
<pre>stem(n,imag(x))</pre>		Graph of the imaginary part of $x[n] = e^{j0.8n}$.
<pre>stem(n,abs(x))</pre>		Graph of the magnitude of $x[n] = e^{j0.8n}$.

(continued)

Commands	Results	Comments
<code>stem(n,angle(x))</code>		Graph of the angle of $x[n] = e^{j0.8n}$.

A more general form for a complex exponential sequence is $x[n] = r^n e^{j\omega n}$. This is equivalent to the signal $x[n] = z^n$, where z is a complex number. The two forms are related according to

$$z = r e^{j\omega}. \quad (2.20)$$

Commands	Results	Comments
<pre>n=-10:10; r=0.9; w=1; x=(r.^n).*exp(j*w*n); stem(n,real(x));</pre>		Definition of the complex exponential sequence $x[n] = 0.9^n e^{jn}$, $-10 \leq n \leq 10$ and graph of its real part.
<code>stem(n,imag(x))</code>		Graph of the imaginary part of $x[n] = 0.9^n e^{jn}$.

(continued)

(continued)

Commands	Results	Comments
<code>stem(n,abs(x))</code>		Graph of the magnitude of $x[n] = 0.9^n e^{jn}$.
<code>stem(n,angle(x))</code>		Graph of the angle of $x[n] = 0.9^n e^{jn}$.
<code>z = 0.9*exp(j*1)</code>	$z = 0.4863 + 0.7573i$	Computation of the complex number z according to (2.20).
<code>n = -10:10;</code> <code>x = z.^n;</code> <code>stem(n,real(x));</code>		The signal $x[n] = z^n$ is defined and its real and imaginary parts are plotted. The graphs are similar to those derived for the signal $x[n] = r^n e^{j\omega n}$, hence, relationship (2.20) is valid.

(continued)

Commands	Results	Comments
<code>stem(n,imag(x));</code>		

2.3.5 Sinusoidal Sequence

The sinusoidal sequence is defined by an expression of the form $x[n] = A \cos(\omega n + \varphi)$ or $x[n] = A \sin(\omega n + \varphi)$, where A is the amplitude, φ is the phase, and ω is the angular frequency. A sinusoidal sequence is easily plotted in MATLAB.

Example

Plot the sinusoidal sequences $x[n] = 2 \cos(n/2 + \pi/4)$, $0 \leq n \leq 20$ and $y[n] = 2 \cos(n\pi/6 + \pi/4)$ $y[n] = 2 \cos(n\pi/6 + \pi/4)$, $0 \leq n \leq 20$.

Commands	Results
<pre>n=0:20; x=2*cos(1/2*n+pi/4); stem(n,x) legend('x[n]') grid</pre>	

(continued)

(continued)

Commands	Results
<pre>y=2*cos(pi/6*n+pi/4); stem(n,y) legend('y[n]') grid</pre>	

Notice that the first sinusoidal sequence $x[n]$ is *not* periodic. We will discuss this further in Section 2.4.1. As stated in the beginning of the chapter, a discrete-time signal $x[n]$ is usually obtained by sampling a continuous-time signal $x(t)$ with a sampling interval T_s . Although sampling is a very general issue, we will restrict ourselves to the case of sampling a continuous-time sinusoidal signal. Suppose that $x(t) = \cos(\Omega t)$ is the signal from which we will derive the discrete-time signal $x[n] = \cos(\omega n) = \cos(\omega n T_s)$. According to the Nyquist–Shannon sampling theorem the sampling rate $f_s = 1/T_s$ must be at least two times larger than the frequency $f = \Omega/2\pi$ of the sinusoidal signal in order to completely determine the continuous-time signal $x(t)$ from the sequence of its samples $x[n] \equiv x[nT_s]$. The mathematical expression is

$$f_s > 2f. \quad (2.21)$$

In other words, the sampling interval T_s must satisfy the following relationship:

$$T_s < \frac{1}{2f} = \frac{\pi}{\Omega}. \quad (2.22)$$

Example

Let $x(t) = \cos(7t)$ be a continuous-time signal. Sample the sequence $x[nT_s]$ for $T_s = \pi/7$ and $T_s = \pi/4$ and plot in the same figure the graphs of the three signals.

In the first case where $T_s = \pi/7$, we expect a discrete-time signal close to $x(t)$ since $T_s \leq \pi/\Omega = \pi/7$. On the other hand, for $T_s = \pi/4$ the Nyquist criterion is not fulfilled as $T_s > \pi/\Omega$; thus $f_s = 1/T_s = 4/\pi < 2f = 2(\Omega/2\pi) = 7/\pi$.

Commands	Results
<pre>t = 0:0.01:10; x = cos(7*t); Ts1 = pi/7; ts1 = 0:Ts1:10; xs1 = cos(7*ts1); Ts2 = pi/4; ts2 = 0:Ts2:10; xs2 = cos(7*ts2); plot(t,x,ts1,xs1,:o',ts2, xs2,:+') legend('x(t)', 'x[n], T_s = \pi/7', 'x[n], T_s = \pi/4')</pre>	

In the first case $T_s = \pi/7$, the signal $x[n]$ is close to $x(t)$, and $x(t)$ can be reconstructed from the samples of $x[n]$ through a process that is not described in this book. On the other hand, the samples obtained with sampling interval $T_s = \pi/4$ are not sufficient enough to determine $x(t)$.

2.4 Properties of Signals

In this section, the basic properties that characterize a signal are introduced.

2.4.1 Periodic Signals

A continuous-time signal $x(t)$ is periodic if there is a positive number T such that

$$x(t) = x(t + T), \quad \forall t \in \mathbb{R}. \quad (2.23)$$

If T is the smallest positive value for which the previous relationship holds, then T is called the fundamental period of the signal. Moreover, if $k \in \mathbb{Z}$ another expression that denotes the periodicity of a signal is

$$x(t) = x(t + kT), \quad \forall t \in \mathbb{R}.$$

For example, a sinusoidal signal $x(t) = A \cos(\Omega t + \theta)$ is periodic with fundamental period $T = 2\pi/\omega$, but $T = 4\pi/\omega$, $T = 6\pi/\omega$, $T = 8\pi/\omega$, etc., are also periods of the signal $x(t)$.

A discrete-time signal is periodic if there is a positive integer number N such that

$$x[n] = x[n + N], \quad \forall n \in \mathbb{Z}. \quad (2.24)$$

Example

Verify that the signal $x(t) = \sin(t)$ is periodic.

First, the period of the signal is computed as $T = 2\pi/\Omega = 2\pi/1 = 2\pi$. Of course, it is not possible to verify the periodicity for all integers k , i.e., to verify that $x(t) = x(t + kT)$, $k \in \mathbb{Z}$. The periodicity of $x(t)$ is verified for $1 \leq k \leq 10$ and time $1 \leq t \leq 5$.

Commands	Results	Comments
<code>t = 1:5; x = sin(t); T = 2*pi;</code>	$x = 0.84 \quad 0.91 \quad 0.14 \quad -0.76 \quad -0.96$	The signal $x(t) = \sin(t)$ is defined over the time interval $1 \leq t \leq 5$.
<code>for k=1:10 xk(k, :) = sin(t+k*T) end</code>	$xk = 0.84 \quad 0.91 \quad 0.14 \quad -0.76 \quad -0.96$ $0.84 \quad 0.91 \quad 0.14 \quad -0.76 \quad -0.96$	The period of $x(t)$ is $T = 2\pi$.
<code>xk</code>	$xk = \begin{bmatrix} x(t+T) \\ x(t+2T) \\ x(t+3T) \\ \dots \\ x(t+10T) \end{bmatrix}$	Using a for-loop we create a 10-row matrix. Each row of the matrix corresponds to the values of $x(t + kT)$ for a specific k .
		The matrix xk is
		All matrix rows are equal to the original signal $x(t)$. Hence, $x(t) = \sin(t)$ is a periodic signal.

As illustrated in Section 2.3.5, a sinusoidal sequence $x[n] = \cos(\omega n + \varphi)$ is not always periodic, i.e., condition (2.24) is not always satisfied. A discrete-time signal $x[n]$ is periodic if there are integer positive numbers m, N , such that

$$\omega = \frac{2\pi m}{N}. \quad (2.25)$$

Therefore, if we look into the first example of the previous section, we notice that indeed the sequence $x[n] = 2 \cos(n/2 + \pi/4)$ is not periodic as there are no integer positive numbers m, N to satisfy the condition $\omega = 1/2 = 2\pi m/N$. On the other hand, the sequence $y[n] = 2 \cos(n\pi/6 + \pi/4)$ is periodic, as for $m=1$ and $N=12$ or for $m=2$ and $N=24$ the condition (2.25) is satisfied since $\omega = \pi/6 = 2\pi m/N$. Moreover, $N=12$ is the fundamental period of the discrete-time sinusoidal signal $y[n]$.

2.4.1.1 Sum of Periodic Continuous-Time Signals

Suppose that $x_1(t)$ and $x_2(t)$ are periodic signals with periods T_1 and T_2 , respectively. In this chapter, the condition that must be fulfilled in order for the signal $x(t) = x_1(t) + x_2(t)$ to be also periodic is given. Moreover, the period of $x(t)$ is computed. Since $x_1(t)$ and $x_2(t)$ are periodic, we get

$$x_1(t) = x_1(t + mT_1), m \in \mathbb{Z} \quad \text{and} \quad x_2(t) = x_2(t + kT_2), k \in \mathbb{Z} \quad (2.26)$$

Thus,

$$x(t) = x_1(t) + x_2(t) = x_1(t + mT_1) + x_2(t + kT_2). \quad (2.27)$$

Suppose that $x(t)$ is indeed periodic with period T . Then, the following relationship also holds:

$$x(t) = x(t + T) = x_1(t + T) + x_2(t + T). \quad (2.28)$$

Combining (2.27) and (2.28) yields

$$x_1(t + T) + x_2(t + T) = x_1(t + mT_1) + x_2(t + kT_2). \quad (2.29)$$

Equation 2.29 is valid if

$$mT_1 = kT_2 = T \quad \text{where } m, k \in \mathbb{Z}. \quad (2.30)$$

If m, k are prime numbers, then the period of the signal $x(t) = x_1(t) + x_2(t)$ is computed directly according to Equation 2.30.

Example

Plot the signal $x(t) = \cos(t) + \sin(3t)$ in time of three periods.

The period T_1 of $\cos(t)$ is computed as $T_1 = 2\pi/\Omega = 2\pi$, while the period T_2 of $\sin(3t)$ is computed as $T_2 = 2\pi/\Omega = 2\pi/3$. Hence, the period T of $x(t)$ is calculated according to Equation 2.30 for $m=1$ and $k=3$, namely, $T=2\pi$.

Therefore, in MATLAB we type

Commands	Results	Comments
<pre>t = 0 : .1 : 6*pi; x = cos(t) + sin(3*t); plot(t,x)</pre>		<p>Time definition ($T = 2\pi \Rightarrow 3T = 6\pi$).</p> <p>The signal $x(t)$.</p> <p>From the graph of $x(t)$ one can see that $x(t)$ is indeed periodic with period $T = 2\pi$.</p>

2.4.1.2 Construction of Periodic Signals

The signals constructed so far through the MATLAB built-in functions (e.g., the rectangular pulse) were defined over the time of one period. In MATLAB, there are several available commands that can be used to construct periodic signals. The command `gensig` is $[s, t] = \text{gensig}(\text{type}, T, t, ts)$, where `type` can be '`sin`' for a sine signal, '`square`' for a rectangular periodic pulse, and '`pulse`' for periodic unit pulses. T is the period and t is the time duration of the signal. Finally, ts is the spacing of the time samples. By default $ts = 10^{-3}$ s. The signal is assigned to variable s , while time is assigned to variable t .

Commands	Results	Comments
<pre>[s,t] = gensig('square',3,20,0.01) plot(t,s) ylim([- .3 1.3])</pre>		Squares pulses that are repeated with period $T=3$, time duration $t=20$, and sampling time $ts=0.01$ s.
<pre>[s,t] = gensig('pulse',2,10); plot(t,s) ylim([- .3 1.3])</pre>		Unit pulses repeated with period $T=2$ over the time interval $0 \leq t \leq 10$.
		Graph of repeated unit pulses.

Another useful command is the command `square`. Its output is actually a squared sine signal with peaks at +1 and -1. Its syntax is like the `sin` function. In other words, the command `square(t)` generates a periodic square wave of period $T=2\pi/\Omega=2\pi$.

Commands	Results	Comments
<code>t = 0:0.1:10;</code>		Time definition.
<code>s = square(t);</code>		Signal definition.
<code>plot(t,s);</code> <code>ylim([-1.3 1.3])</code>		The period of $s(t)$ is $T=2\pi$ and the amplitude is ± 1 .
<code>s2 = square(2*pi*t);</code>		Definition of a second signal in the same time.
<code>plot(t,s2);</code> <code>ylim([-1.3 1.3]) ;</code>		The period of $s_2(t)$ is $T=2\pi/\Omega=2\pi/2\pi=1$.

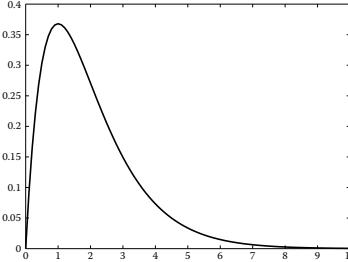
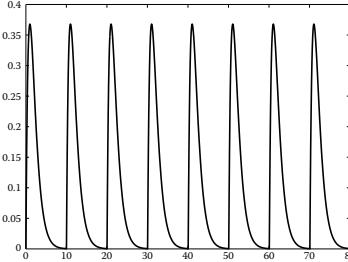
A similar signal is generated by the command `sawtooth`. Its syntax is `sawtooth(t, T)` and the output is a triangle wave with period T . If no period is specified the default period is $T=2\pi/\Omega=2\pi$.

Commands	Results	Comments
<pre>t = 0:0.1:20; s = sawtooth(t);</pre> <pre>plot(t,s); ylim([-1.3 1.3])</pre>		<p>Definition of a triangle wave for $0 \leq t \leq 20$.</p> <p>Indeed the period is $T = 2\pi$, while the signal takes values in $[-1,1]$.</p>

All periodic signals that were constructed so far had a precise form (e.g., square). Suppose that we need to repeat an arbitrary signal (e.g., the signal $x(t) = te^{-t}$, $0 \leq t \leq 10$) several times, i.e., to construct a periodic signal with period $T = 10$ that in one period is given by $x(t) = te^{-t}$, $0 \leq t \leq 10$. The solution to this problem is to use the command `repmat`. Its syntax is `xp = repmat(x, N, K)`. This command accepts as input argument a vector x and repeats it for N rows and K columns. The result is assigned to the $N \times K$ matrix xp . The use of the command `repmat` is illustrated in the table below.

Commands	Results	Comments
<pre>x = [1 2]; N = 3; K = 4;</pre> <pre>xp = repmat(x,N,K)</pre>	<pre>x = [1 2] N = 3; K = 4; xp = 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2</pre>	<p>Definition of the vector x and of the number of rows and columns.</p> <p>The matrix xp is a replica of vector x in N rows and K columns.</p>

We now return to the original problem, namely, to repeat (suppose eight times) the signal $x(t) = te^{-t}$, $0 \leq t \leq 10$. The process followed is

Commands	Results	Comments
<pre>t = 0 : .1 : 10; x = t .* exp(-t); plot(t,x);</pre>		The signal $x(t)$ is defined over the time of one period (here $T=10$).
<pre>xp = repmat(x, 1, 8); tp = linspace(0, 80, length(xp)); plot(tp, xp)</pre>		<p>Vector x is replicated in 1 row and 8 columns.</p> <p>The time that the periodic signal xp will be plotted is estimated by dividing the time interval $0 \leq tp \leq 8T$ in a number of points equal to the number of elements of xp.</p>

A discrete-time periodic signal can be also constructed easily by using the command `repmat`.

Example

Plot in 10 periods the periodic signal that in one period ($T=4$) is given by

$$x[n] = \begin{cases} 1, & n = 0, 1 \\ -1, & n = 2, 3 \end{cases}.$$

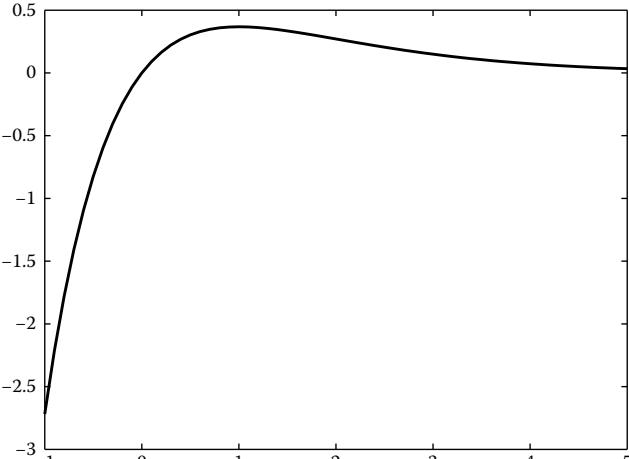
Commands	Results	Comments
<pre>N=10; x=[1 1 -1 -1]; xp=repmat(x,1,N); n=0:length(xp)-1; stem(n,xp);</pre>		The signal is repeated $N=10$ times.

2.4.2 Causal Signals

A signal $x(t)$ (or $x[n]$) is causal if it is zero for all negative values of time, i.e., $x(t)=0$, $t < 0$ (or $x[n]=0$, $n < 0$). Non-causal signals are signals that have nonzero values in both positive and negative time. Finally, a signal that is zero for all positive times is called anti-causal.

Commands	Results	Comments
<pre>t1=0:0.1:5; x1=t1.*exp(-t1); plot(t1,x1)</pre>		Causal signal.

(continued)

Commands	Results	Comments
<pre>t2=-1:0.1:5; x2=t2.*exp(-t2); plot(t2,x2)</pre>		Non-causal signal.

2.4.3 Even and Odd Signals

A signal $x(t)$ is even (or has even symmetry or is an even function of t) if $x(-t)=x(t)$, $-\infty < t < \infty$.

A signal is odd (or has odd symmetry or is an odd function of t) if $x(-t)=-x(t)$, $-\infty < t < \infty$.

Example

Find out if the signals $x(t)=t^2$ and $y(t)=t^3$ are even or odd.

Of course, we cannot examine these two signals over the time interval $(-\infty, \infty)$. The symmetry of $x(t)$ and $y(t)$ is tested in the time interval $-4 \leq t \leq 4$.

Commands	Results	Comments
$t1=0:4$	$t1 = 0 \quad 1.00 \quad 2.00 \quad 3.00 \quad 4.00$	The time t .
$t2=0:-1:-4$	$t2 = 0 \quad -1.00 \quad -2.00 \quad -3.00 \quad -4.00$	The time $-t$.
$x1=t1.^2$	$x1 = 0 \quad 1.00 \quad 4.00 \quad 9.00 \quad 16.00$	The signal $x(t)$.
$x2=t2.^2$	$x2 = 0 \quad 1.00 \quad 4.00 \quad 9.00 \quad 16.00$	The signal $x(-t)$. It is clear that $x(t)=x(-t)$, hence, $x(t)=t^2$ is an even signal.
$y1=t1.^3$	$y1 = 0 \quad 1.00 \quad 8.00 \quad 27.00 \quad 64.00$	The signal $y(t)$.
$y2=t2.^3$	$y2 = 0 \quad -1.00 \quad -8.00 \quad -27.00 \quad -64.00$	The signal $y(-t)$. It is clear that $-y(t)=y(-t)$, hence, $y(t)=t^3$ is an odd signal.

Remark

A signal is not always even or odd. For example, the signal $x(t) = t^2/(t+5)$ is neither odd nor even.

All signals can be expressed as the sum of an odd signal $x_o(t)$ and an even signal $x_e(t)$; that is, any signal $x(t)$ is expressed as $x(t) = x_e(t) + x_o(t)$, where

$$\begin{aligned}x_e(t) &= \frac{1}{2}[x(t) + x(-t)] \\x_o(t) &= \frac{1}{2}[x(t) - x(-t)].\end{aligned}\tag{2.31}$$

In the discrete-time signals case, the same rules are valid. A signal $x[n]$ is even if $x[n] = x[-n]$ and odd if $-x[n] = x[-n]$. Moreover, a discrete-time signal $x[n]$ can be expressed as the sum of an even signal $x_e[n]$ and an odd signal $x_o[n]$, where $x_e[n]$ and $x_o[n]$ are given by

$$\begin{aligned}x_e[n] &= \frac{1}{2}(x[n] + x[-n]) \\x_o[n] &= \frac{1}{2}(x[n] - x[-n]).\end{aligned}\tag{2.32}$$

Example

Separate the unit step sequence $u[n]$, $-5 \leq n \leq 5$ in even and odd parts. Verify your outcome by constructing the unit step sequence from its even and odd parts.

In order to derive the even and odd parts of $x[n]$ we first compute the signal $x[-n]$, which is given by

$$x[-n] = u[-n] = \begin{cases} 1, & n \leq 0 \\ 0, & n > 0 \end{cases}.$$

Commands	Results	Comments
<pre>n = -5 : 5; u = (n >= 0); stem(n, u);</pre>		Definition and graph of $x[n] = u[n]$, $-5 \leq n \leq 5$.

(continued)

Commands	Results	Comments
<pre>u_n = (n<= 0) ue = 1/2*(u+ u_n); stem(n,ue);</pre>		First we define the signal $x[-n] = u[-n]$. Next, the even part $x_e[n]$ is computed according to the upper part of (2.32).
<pre>uo = 1/2*(u- u_n); stem(n,uo);</pre>		Computation and graph of the odd part $x_o[n]$ according to the lower part of (2.32).
<pre>stem(n,ue+uo)</pre>		Graph of the signal $x_e[n] + x_o[n]$. Indeed it is identical to $u[n]$.

2.4.4 Energy and Power Signals

An important classification of signals is between (finite) energy and (finite) power signals. The energy E_x of a continuous-time signal $x(t)$ is computed according to

$$E_x = \lim_{T \rightarrow \infty} \int_{-T}^T |x(t)|^2 dt = \int_{-\infty}^{\infty} |x(t)|^2 dt, \quad (2.33)$$

where $|x(t)|$ is the absolute value of $x(t)$. A signal $x(t)$ is an energy signal if its energy is definite, i.e., $0 < E_x < \infty$.

The power P_x of a signal $x(t)$ is given by

$$P_x = \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^T |x(t)|^2 dt. \quad (2.34)$$

If $x(t)$ is periodic, the power is computed from the relationship

$$P_x = \frac{1}{T} \int_{t_0}^{t_0+T} |x(t)|^2 dt. \quad (2.35)$$

A signal $x(t)$ is called a power signal if $0 < P_x < \infty$. In order to calculate the energy or the power of a signal, recall that the command `limit(F, x, a)` computes the limit of the function F when the symbolic variable x tends to a .

Example

Verify that $x(t) = u(t)$ is a power-type signal while $x(t) = u(t) - u(t - 1)$ is an energy-type signal.

First the signal $x(t) = u(t)$ is considered.

Commands	Results	Comments
<code>syms t T</code>		Symbolic variables definition.
<code>x = heaviside(t);</code>		Definition of $x(t) = u(t)$.
<code>d = int(abs(x)^2, t, -T, T);</code>		Calculation of $\int_{-T}^T x(t) ^2 dt$.
<code>Ex = limit(d, T, inf)</code>	<code>Ex = Inf</code>	The energy of $x(t) = u(t)$ given by $E_x = \lim_{T \rightarrow \infty} \int_{-T}^T x(t) ^2 dt$ is infinite. Hence, $u(t)$ is not an energy signal.
<code>Px = limit((1/(2*T))*d, T, inf)</code>	<code>Px = 1/2</code>	The power of $x(t) = u(t)$ is computed as $P_x = \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^T x(t) ^2 dt$. The power is $P_x = 0.5$; thus $u(t)$ is a power signal.

For the signal $x(t) = u(t) - u(t - 1)$ we get

Commands	Results	Comments
<code>syms t T</code>		Symbolic variables definition.
<code>x=heaviside(t)-heaviside(t-1);</code>		Definition of $x(t) = u(t) - u(t-1)$.
<code>d=int(abs(x)^2,t,-T,T);</code>		Calculation of $\int_{-T}^T x(t) ^2 dt$.
<code>Px=limit((1/(2*T))*d,T,inf)</code>	$Px = 0$	The power computed by $P_x = \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^T x(t) ^2 dt$ is zero, hence, $x(t) = u(t) - u(t-1)$ is not a power signal.
<code>Ex=limit(d,T,inf)</code>	$Ex = 1$	The energy given by $E_x = \lim_{T \rightarrow \infty} \int_{-T}^T x(t) ^2 dt$ is $E_x = 1$; thus $u(t) - u(t-1)$ is an energy signal.

Remark

If a signal is of finite energy, (i.e., is energy signal) the signal power is zero. On the other hand, if a signal is a power signal its energy is infinite.

Remark

There are signals that are neither energy signals nor power signals. For example, the signal $x(t) = t$ does not belong to any of these categories.

Many times it is desired to compute the energy or the power of a signal in a definite time interval $t_1 \leq t \leq t_2$. In this case, the energy is given by

$$E_x = \int_{t_1}^{t_2} |x(t)|^2 dt, \quad (2.36)$$

while the power is calculated according to

$$P_x = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} |x(t)|^2 dt. \quad (2.37)$$

Example

Compute the energy and the power of the signal $x(t) = 2\cos(\pi t)$ in one period time.

The period of $x(t) = 2\cos(\pi t)$ is $T = 2\pi/\Omega = 2\pi/\pi = 2$.

Commands	Results	Comments
<code>t1 = 0; t2 = 2; syms t x = 2*cos(pi*t);</code>		We define the time of one period as $T = t_2 - t_1$. Moreover, in order to solve the integrals (2.36) and (2.37) t is declared as a symbolic variable; thus X becomes a symbolic expression.
<code>Ex = int(abs(x)^2, t, t1, t2)</code>	<code>Ex = 4</code>	The energy in one period
<code>Px = (1/(t2-t1))*int(abs(x)^2, t, t1, t2)</code>	<code>Px = 2</code>	The power in one period

The energy and power of a discrete-time signal are computed from quite similar expressions. The energy is given by

$$E_x = \sum_{n=-\infty}^{\infty} |x[n]|^2 = \lim_{N \rightarrow \infty} \sum_{n=-N}^N |x[n]|^2, \quad (2.38)$$

while the power is computed according to

$$P_x = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N |x[n]|^2. \quad (2.39)$$

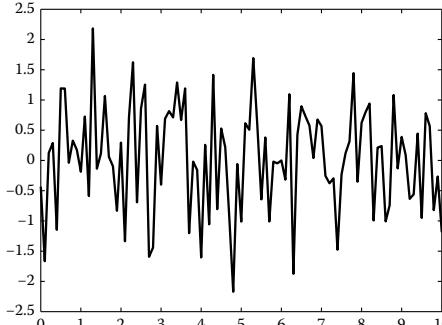
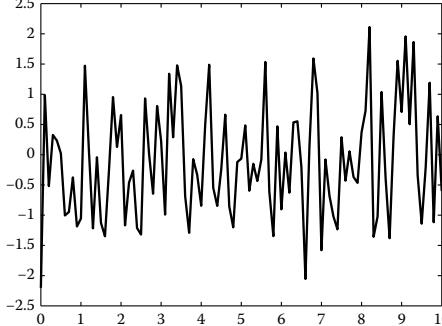
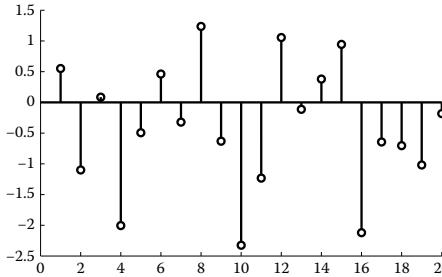
Example

Compute the energy of the signal $x[n] = \cos(n)$, $-5 \leq n \leq 5$.

Commands	Results	Comments
<code>N = 5; n = -N:N;</code>	<code>E = 4.9058</code>	
<code>x = cos(n); E = sum((abs(x)).^2)</code>	<code>E = 4.9058</code>	The energy of the signal $x[n]$.
<code>P = 1/(2*N+1)*E</code>	<code>P = 0.4460</code>	The average power of $x[n]$.

2.4.5 Deterministic and Stochastic Signals

One other important classification is the one between deterministic and stochastic signals. A signal is deterministic if no randomness is involved in its values, i.e., if each value is fixed and can be determined by a mathematical expression. On the other hand, a stochastic or random signal has randomness and its values cannot be predicted. An example of a deterministic signal is the unit step function as its values (1 or 0) are known for any value of t . An example of a random signal is thermal noise produced by electronic devices. Thermal noise is usually modeled as a sequence of random numbers distributed according to the normal distribution with mean value $\mu = 0$ and variance $\sigma^2 = 1$. In the next example, a stochastic signal is plotted.

Commands	Results	Comments
<pre>t = 0:.1:10; x1 = randn(size(t)); plot(t,x1);</pre>		Definition of time interval. The thermal noise signal is defined by using the command <code>randn</code> .
<pre>t = 0:.1:10; x2 = randn(size(t)); plot(t,x2);</pre>		The exactly same process generates a different, also random, signal $x_2(t)$.
<pre>x = randn(20,1); stem(x);</pre>		Random discrete-time signal.

The field of statistical signal processing is out of the scope of this book as we mainly concentrate on deterministic signals.

2.5 Transformations of the Time Variable for Continuous-Time Signals

In many cases, there are signals related to each other with operations performed on the independent variable, namely, the time. In this section, we examine the basic operations that are performed on the independent variable.

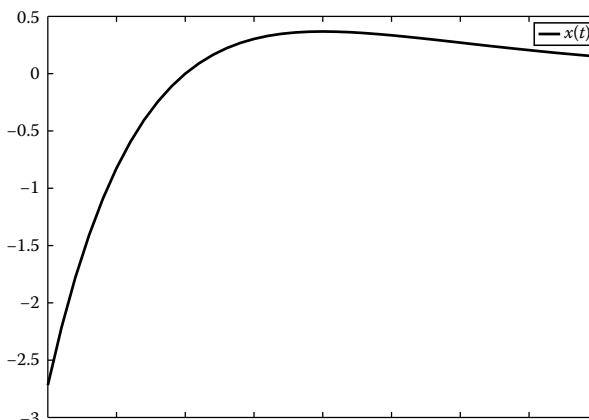
2.5.1 Time Reversal or Reflection

The first operation discussed is the signal's reflection. A signal $y(t)$ is a reflection or a reflected version of $x(t)$ about the vertical axis if $y(t) = x(-t)$.

The operation of time reversal is actually an alternation of the signal values between negative and positive time. Assume that x is the vector that denotes the signal $x(t)$ in time t . The MATLAB statement that plots the reflected version of $x(t)$ is `plot (-t, x)`.

Example

Suppose that $x(t) = te^{-t}$, $-1 \leq t \leq 3$. Plot the signal $x(-t)$.

Commands	Results	Comments
<pre>t = -1:.1:3; x = t.*exp(-t); plot(t,x); legend('x(t)')</pre>		Graph of the original signal $x(t)$.

(continued)

Commands	Results	Comments
<pre>plot(-t,x); legend('x(-t)')</pre>		Graph of the reflected version $y(t)=x(-t)$. In order to plot $x(-t)$ it is enough to replace t by $-t$ at the plot command.

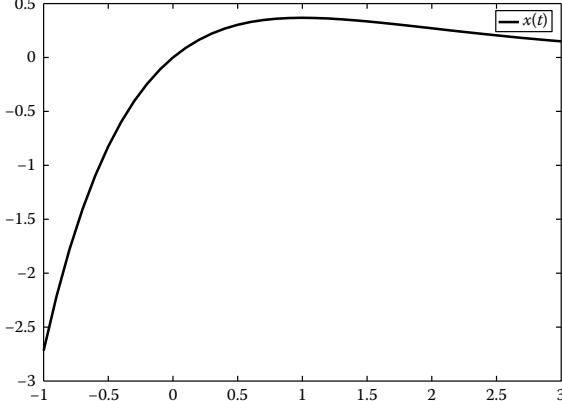
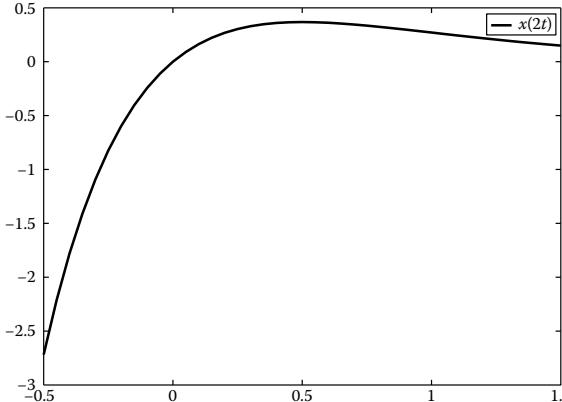
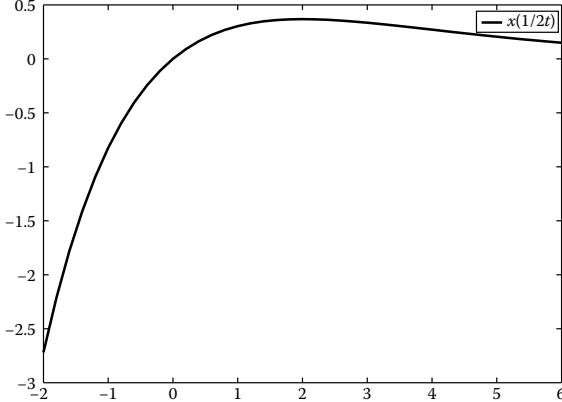
2.5.2 Time Scaling

The second operation discussed is time scaling. A signal $x_1(t)$ is a compressed version of $x(t)$ if $x_1(t) = x(at)$, $a > 1$. The time compression practically means that the time duration of the signal is reduced by a factor a . On the other hand, a signal $x_2(t)$ is an expanded version of $x(t)$ if $x_2(t) = x(at)$, $0 < a < 1$. In this case, the time duration of the signal is increased by a factor $1/a$.

In order to plot in MATLAB a time-scaled version of $x(t)$, namely, a signal of the form $y(t) = x(at)$, the statement employed is `plot((1/a)*t, x)`. In contrast to what someone would expect the vector of time t must be multiplied by $1/a$ and not a .

Example

Consider again the continuous-time signal $x(t) = te^{-t}$, $-1 \leq t \leq 3$. We will plot the signal $x_1(t) = x(2t)$, which is a time compression of $x(t)$ by a factor $a=2$; and the signal $x_2(t) = x(0.5t)$, which is a time expansion of $x(t)$ by a factor $1/a=2$.

Commands	Results	Comments
<pre>t = -1:.1:3; x = t.*exp(-t); plot(t,x); legend('x(t)');</pre>	 A plot of the exponential decay function $x(t) = t \cdot e^{-t}$ for $t \in [-1, 3]$. The x-axis ranges from -1 to 3 with major ticks every 0.5 units. The y-axis ranges from -3 to 0.5 with major ticks every 0.5 units. The curve starts at approximately (-1, -2.7), passes through (0, 0), and approaches a horizontal asymptote at $y = 0$ as $t \rightarrow 3$.	Graph of $x(t)$. The signal lies in the time interval $-1 \leq t \leq 3$.
<pre>a = 2; plot((1/a)*t,x) legend('x(2t)');</pre>	 A plot of the compressed exponential decay function $x(2t) = 2t \cdot e^{-2t}$ for $t \in [-0.5, 1.5]$. The x-axis ranges from -0.5 to 1.5 with major ticks every 0.5 units. The y-axis ranges from -3 to 0.5 with major ticks every 0.5 units. The curve starts at approximately (-0.5, -2.7), passes through (0, 0), and approaches a horizontal asymptote at $y = 0$ as $t \rightarrow 1.5$.	Graph of $x(2t)$, which is a compressed version of $x(t)$. Notice that the signal lies in the time interval $-0.5 \leq t \leq 1.5$.
<pre>a = 1/2; plot((1/a)*t,x) legend('x(1/2 t)');</pre>	 A plot of the expanded exponential decay function $x(1/2t) = 0.5t \cdot e^{-0.5t}$ for $t \in [-2, 6]$. The x-axis ranges from -2 to 6 with major ticks every 1 unit. The y-axis ranges from -3 to 0.5 with major ticks every 0.5 units. The curve starts at approximately (-2, -2.7), passes through (0, 0), and approaches a horizontal asymptote at $y = 0$ as $t \rightarrow 6$.	Graph of $x(t/2)$, which is an expanded version of $x(t)$. Notice that the signal lies in the time interval $-2 \leq t \leq 6$.

2.5.3 Time Shifting

A third operation performed on time is one of time shifting. A signal $y(t)$ is a time-shifted version of $x(t)$ if $y(t) = x(t - t_0)$, where t_0 is the time shift. If $t_0 > 0$, the signal $y(t) = x(t - t_0)$ is shifted by t_0 units to the right (i.e., toward $+\infty$); while if $t_0 < 0$, the signal $y(t) = x(t - t_0)$ is shifted by t_0 units to the left (i.e., toward $-\infty$). The time shifting is implemented in MATLAB in an opposite way to what may be expected. More specifically, in order to plot the signal $x(t - t_0)$ the corresponding MATLAB statement is `plot(t+t0,x)`.

Example

The signal $x(t) = te^{-t}$, $-1 \leq t \leq 3$ is again considered. We will plot the signals $x_1(t) = x(t - 2)$, that is, a shifted version of $x(t)$ by two units to the right (here $t_0 = 2$) and $x_2(t) = x(t + 3) = x(t - (-3))$, that is, a shifted version of $x(t)$ by 3 units to the left (here $t_0 = -3$).

Commands	Results	Comments
<pre>t = -1:.1:3; x = t.*exp(-t); plot(t,x); legend('x(t)');</pre>		Graph of $x(t)$. The signal lies in the time interval $-1 \leq t \leq 3$.
<pre>t0 = 2; plot(t+t0,x) legend('x(t-2)');</pre>		Graph of $x(t - 2)$. The signal $x(t - 2)$ is shifted two units to the right compared to $x(t)$, i.e., it lies in the time interval $1 \leq t \leq 5$.

(continued)

(continued)

Commands	Results	Comments
<pre>t0 = -3; plot(t+t0,x); legend('x(t+3)');</pre>		Graph of $x(t + 3)$. The signal $x(t + 3)$ is shifted 3 units to the left compared to $x(t)$, i.e., it lies in the time interval $-4 \leq t \leq 0$.

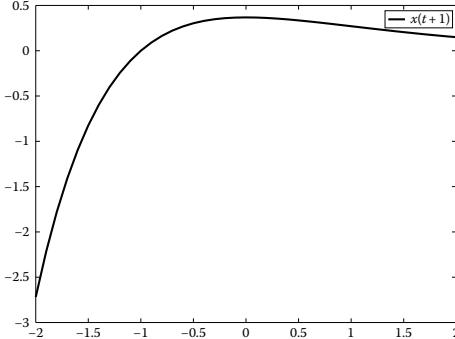
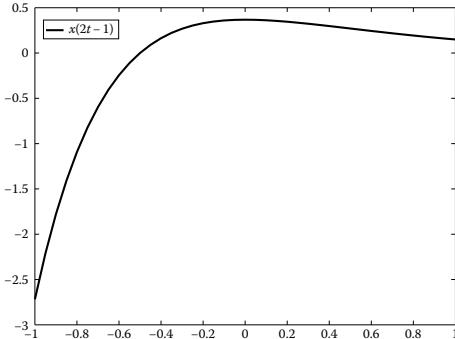
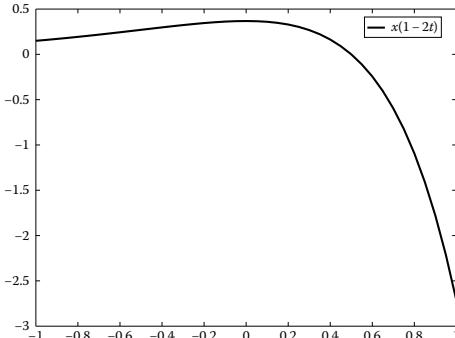
Example

Suppose that $x(t) = te^{-t}$, $-1 \leq t \leq 3$. Plot the signal $q(t) = x(1 - 2t)$.

For the signal $x(1 - 2t)$, all three operations discussed so far (reversal, scaling, and shifting) are performed. In such a case, the graph of the signal has to be obtained according to a specific order. First, we plot the time-shifted version, then time scaling is applied, and finally the operation of time reversal is performed in order to obtain the desired signal.

Commands	Results	Comments
<pre>t = -1:.1:3; x = t.*exp(-t); plot(t,x); legend('x(t)');</pre>		Graph of $x(t)$. The signal lies in the time interval $-1 \leq t \leq 3$.

(continued)

Commands	Results	Comments
<pre>plot(t-1,x) legend('x(t+1)')</pre>		The first operation performed is that of time shifting; that is, the signal $y(t) = x(t + 1)$ is obtained. The signal $x(t + 1)$ lies in the time interval $-2 \leq t \leq 2$.
<pre>plot(0.5*(t-1),x) legend('x(2t-1)')</pre>		Next, the time-scaling operation is applied. The signal $z(t) = y(2t) = x(2t + 1)$ is plotted. Notice that $x(2t + 1)$ lies in the time interval $-1 \leq t \leq 1$. Also notice that the entire expression associated with time is multiplied by the factor 1/2.
<pre>plot(-0.5*(t-1),x) legend('x(1-2t)')</pre>		Finally, the signal $z(t)$ is reflected in order to obtain the signal $q(t) = z(-t) = x(1 - 2t)$. Notice that the entire expression associated with time is reversed.

This order (shifting-scaling-reversal) must be strictly followed in order to correctly plot a signal. The fact that the entire expression (which is the first argument of the command `plot`) is multiplied by the scaling factor and afterward reversed is easily explained if we consider that it represents the time interval in which the signal is plotted.

2.6 Transformations of the Time Variable for Discrete-Time Signals

Suppose that $x[n]$ is a discrete-time signal defined over the time interval specified by n , and n_0 is an integer number. The three transformations of the time variable for discrete-time signals are the following:

- *Time shifting*. This operation is similar to the continuous time operation. More specifically, the signal $y[n] = x[n - n_0]$ is shifted by n_0 units (or samples) to the right if $n_0 > 0$, and is shifted by n_0 units to the left if $n_0 < 0$. The associated MATLAB statement is `stem(n+n0, x)`.
- *Time reversal* is also similar to the operation performed for continuous-time signals. It is described by the relationship $y[n] = x[-n]$, where $y[n]$ is the reflected (about the vertical axis) signal. Time reversal is implemented in MATLAB by typing `stem(-n, x)`.
- *Time scaling*. This transformation is somehow different from the one described in the continuous time case. The relationship that describes the time-scaling operation is $y[n] = x[an]$. If $a > 1$ and $a \in \mathbb{Z}$, the time-scaling operation is called *downsampling* or decimation. Notice that a must be integer as $x[n]$ is undefined for fractional values of n . The downsampling operation results in time compression of the signal. Moreover, some samples of $x[n]$ are lost. The downsampling operation is implemented in MATLAB by using the command `y = downsample(x, a)` or the statement `y = x(1:a:end)`. The variable `end` represents the last index in an indexing expression. If $0 < a < 1$, the signal $y[n] = x[an]$ is a time-expanded version of $x[n]$. In this case, $(1/a) - 1$ zeros are inserted between two consecutive samples of $x[n]$. The time-expansion operation is called *upsampling*, and is implemented by the MATLAB command `y = upsample(x, 1/a)` or with the statement `y(1:1/a:end) = x`.

Remark

There are some limitations on the values that a can take. In case of downsampling a must be a positive integer, and in case of upsampling $1/a$ must be a positive integer.

The downsampling and upsampling processes are illustrated in the next example for the discrete-time signal $x[n] = [1, 2, 3, 4, 5, 6]$, $0 \leq n \leq 5$.

Commands	Results	Comments
<code>x = [1, 2, 3, 4, 5, 6]</code>	<code>x = 1 2 3 4 5 6</code>	The discrete-time signal $x[n] = [1, 2, 3, 4, 5, 6]$, $0 \leq n \leq 5$.
<code>a = 2;</code> <code>xds = downsample(x, a)</code>	<code>xds = 1 3 5</code>	The downsampled version of $x[n]$.

(continued)

Commands	Results	Comments
<code>xds = x(1:a:end)</code>	<code>xds = 1 3 5</code>	Alternative computation of the down-sampled signal.
<code>a = 1/2;</code> <code>xups = upsample(x, 1/a)</code>	<code>xups = 1 0 2 0 3</code> 0 4 0 5 0 6 0	Upsampling operation on $x[n]$.
<code>xups = zeros(1, 1/a*length(x))</code> <code>xups(1:1:a:end) = x</code>	<code>xups = 1 0 2 0 3</code> 0 4 0 5 0 6 0	Upsampling operation performed in an alternative way. Notice that the variable in which the upsampled signal is stored must be first defined as a vector of zeros with length $(1/a) \cdot \text{length}(x[n])$.

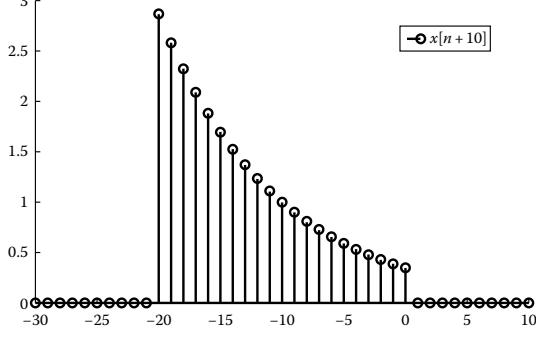
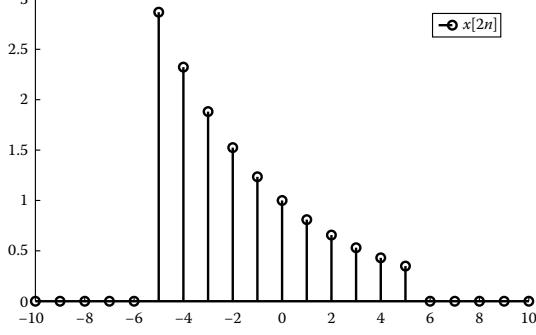
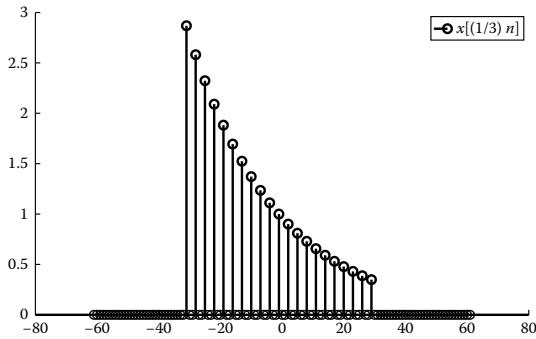
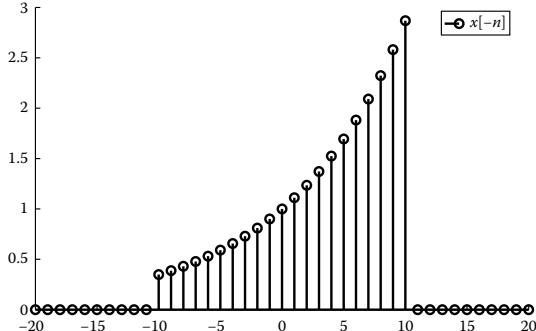
Example

Consider the sequence $x[n] = 0.9^n$, $-10 \leq n \leq 10$. Plot the sequences $x[n - 10]$, $x[n + 10]$, $x[3n]$, $x[n/3]$, and $x[-n]$.

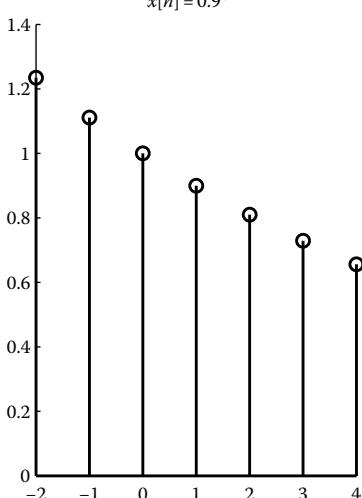
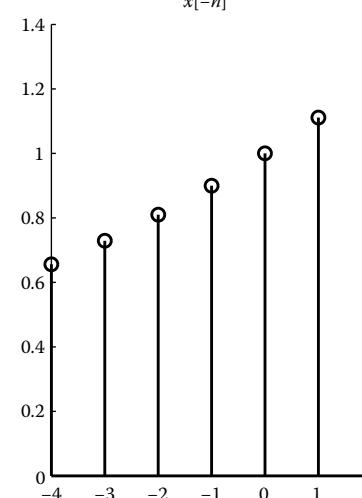
Commands	Results	Comments
<pre>n = -20:20; p = ((n >= -10) & (n <= 10)); x = (0.9.^n).*p; stem(n,x); legend('x[n]');</pre>		Graph of $x[n] = 0.9^n$, $-10 \leq n \leq 10$.
<pre>stem(n+10,x) legend('x[n-10]');</pre>		Graph of $x[n - 10]$. The signal $x[n]$ is shifted by 10 units to the right.

(continued)

(continued)

Commands	Results	Comments
<code>stem(n-10,x) legend('x[n+10]')</code>		Graph of $x[n + 10]$. The signal $x[n]$ is shifted by 10 units to the left.
<code>a = 2; xd = downsample(x,a); nd = -10:10; stem(nd,xd); legend('x[2n]')</code>		Graph of $x[2n]$. The signal $x[n]$ is compressed by a factor 2.
<code>a = 1/3 xup = upsample(x,1/a) nup = -61:61 stem(nup,xup) legend('x[(1/3)n]')</code>		Graph of $x[(1/3)n]$. The signal $x[n]$ is expanded by a factor 3.
<code>stem(-n,x) legend('x[-n]')</code>		Graph of $x[-n]$, which is a reflected version of $x[n]$.

An alternative way to obtain the signal $x[-n]$ is by using the command `fliplr`. This command flips the order of the vector elements. To demonstrate its use, the signal $x[n] = 0.9^n$, $-2 \leq n \leq 4$ is considered.

Commands	Results
<code>n = -2:4;</code>	$n = -2 \quad -1 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4$
<code>n1 = -fliplr(n)</code>	$n1 = -4 \quad -3 \quad -2 \quad -1 \quad 0 \quad 1 \quad 2$
<code>x = 0.9.^n</code>	$x = 1.23 \quad 1.11 \quad 1.00 \quad 0.90 \quad 0.81 \quad 0.73 \quad 0.66$
<code>x1 = fliplr(x)</code>	$x1 = 0.66 \quad 0.73 \quad 0.81 \quad 0.90 \quad 1.00 \quad 1.11 \quad 1.23$
	$x[n] = 0.9^n$
<code>subplot(121);</code> <code>stem(n,x);</code> <code>title('x[n] = 0.9^n');</code> <code>subplot(122);</code> <code>stem(n1,x1);</code> <code>title('x[-n]');</code>	
	$x[-n]$
	

2.7 Solved Problems

Problem 1 Plot the continuous-time signal $x(t) = u(t+1) - u(t-2) + u(t-4)$

- a. Without using the command `heaviside`
- b. Using the command `heaviside`

Solution

In case of an expression with multiple unit step functions, the procedure is to create a table in which the value of each unit step function is computed at the various time intervals.

	$-\infty < t < -1$	$-1 \leq t < 2$	$2 \leq t < 4$	$4 \leq t < \infty$
$u(t+1)$	0	1	1	1
$u(t-2)$	0	0	1	1
$u(t-4)$	0	0	0	1
$x(t)$	0	1	0	1

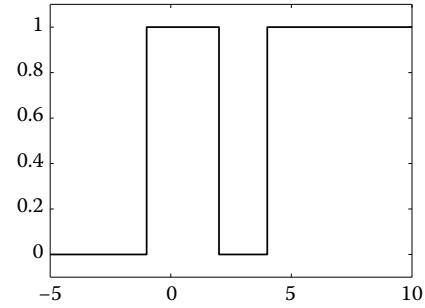
Hence,

$$x(t) = \begin{cases} 0, & t < -1 \\ 1, & -1 \leq t < 2 \\ 0, & 2 \leq t < 4 \\ 1, & t \geq 4 \end{cases}$$

The MATLAB implementation is as follows.

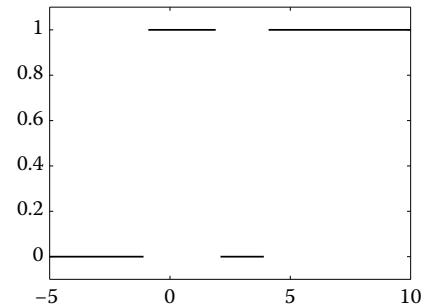
a.

```
t1=-5:.1:-1;
t2=-1:.1:2;
t3=2:.1:4;
t4=4:.1:10;
x1=zeros(size(t1));
x2=ones(size(t2));
x3=zeros(size(t3));
x4=ones(size(t4));
t=[t1 t2 t3 t4];
x=[x1 x2 x3 x4];
plot(t,x);
ylim([-0.1 1.1]);
```



b.

```
t=-5:.1:10;
x=heaviside(t+1)-heaviside(t-2)+heaviside(t-4);
plot(t,x)
ylim([-0.1 1.1]);
```



Problem 2 Plot the signal $x(t) = u(t + 1) u(t - 1)$

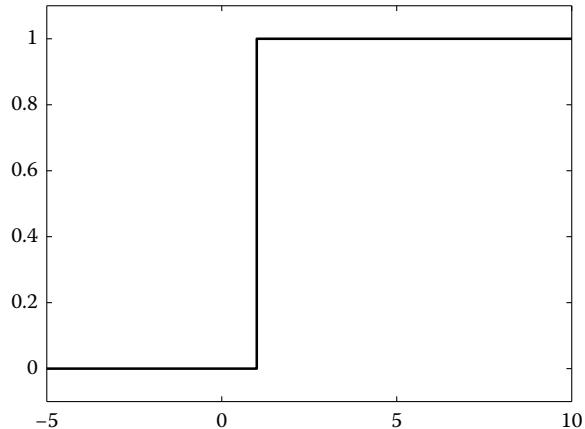
- Without using the command heaviside
- Using the command heaviside

Solution

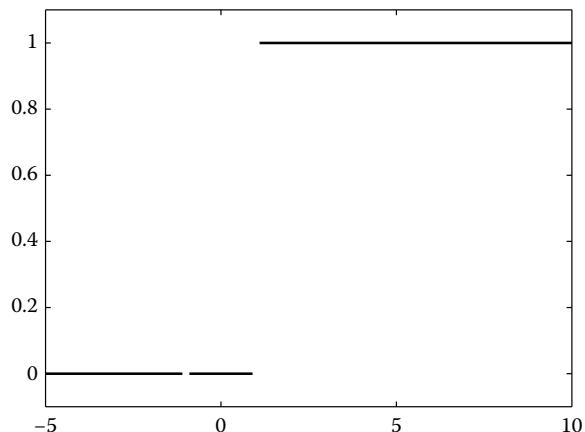
Again a similar table is created.

	$-\infty < t < -1$	$-1 \leq t < 1$	$1 \leq t < \infty$
$u(t + 1)$	0	1	1
$u(t - 1)$	0	0	1
$x(t)$	0	0	1

a.
 $t1 = -5 : .1 : 1;$
 $t2 = 1 : .1 : 10;$
 $x1 = zeros(size(t1));$
 $x2 = ones(size(t2));$
 $t = [t1 t2];$
 $x = [x1 x2];$
 $plot(t, x);$
 $ylim([-0.1 1.1]);$



b.
 $t = -5 : .1 : 10;$
 $x = heaviside(t+1) .* heaviside(t-1);$
 $plot(t, x);$
 $ylim([-0.1 1.1]);$



Problem 3 Verify that $\int_{-\infty}^t \delta(r)dr = u(t)$.

Solution

```
syms r t
int(dirac(r),r,-inf,t)      ans = heaviside(t)
```

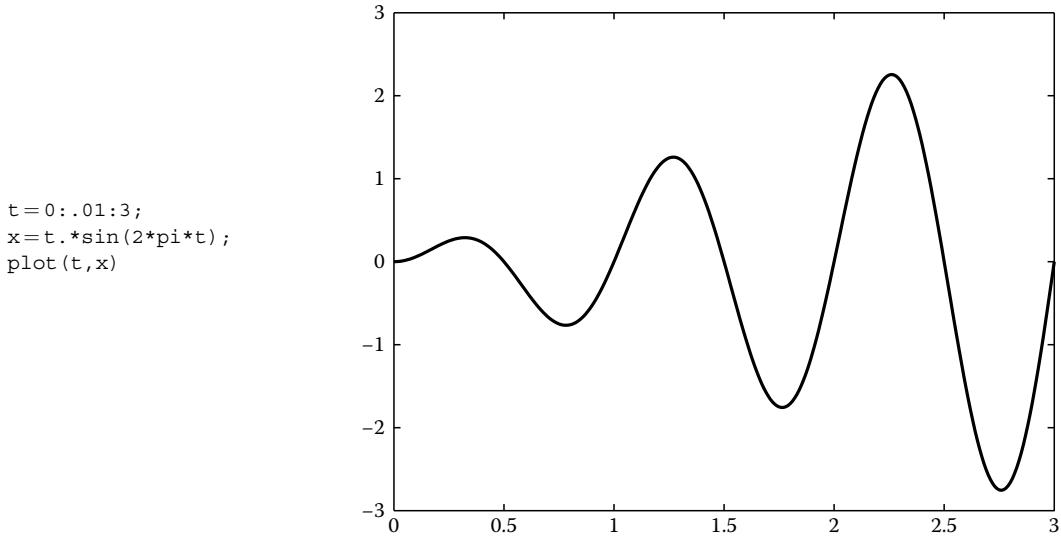
Problem 4 Plot the signal $x(t) = t \sin(2\pi t)(u(t) - u(t - 3))$.

Solution

The first approach is to express the signal as

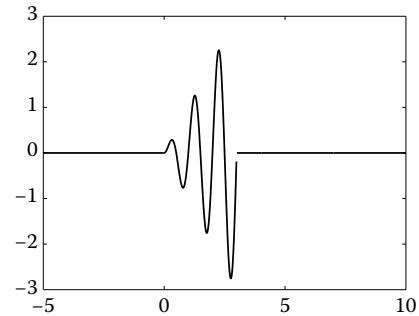
$$x(t) = \begin{cases} t \sin(2\pi t), & 0 \leq t \leq 3 \\ 0, & t < 0 \text{ and } t > 3 \end{cases}.$$

Therefore the MATLAB code is



The other approach is to use the command `heaviside`.

```
t = -5:.01:10;
x = t.*sin(2*pi*t).*(heaviside(t)-heaviside(t-3));
plot(t,x)
```

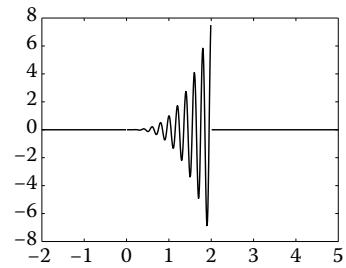


Problem 5 Plot the signal $x(t) = t^3 \cos(10\pi t) p2(t - 1)$, where $pT(t)$ is a rectangular pulse of duration T .

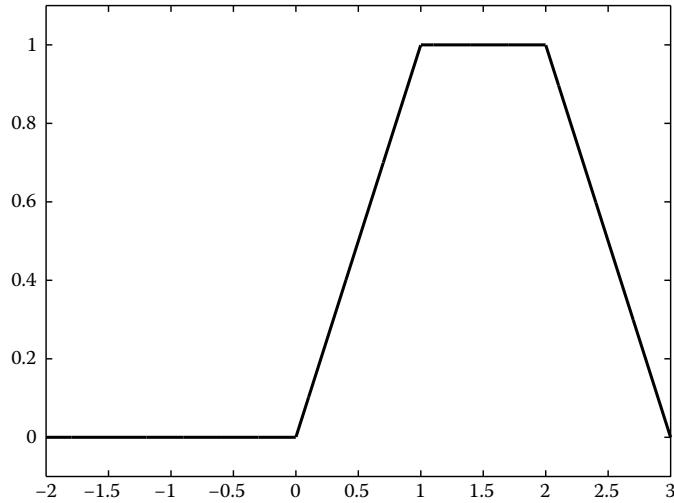
Solution

The rectangular pulse $p2(t - 1)$ is given by $p2(t - 1) = u(t - 1 + 2/2) - u(t - 1 - 2/2) = u(t) - u(t - 2)$. Hence,

```
t = -2:.01:5;
x = (t.^3).*cos(10*pi*t).*(heaviside(t)-heaviside(t-2));
plot(t,x)
```



Problem 6 Express the signal depicted in the figure as a sum of ramp functions. Plot your result for verification.

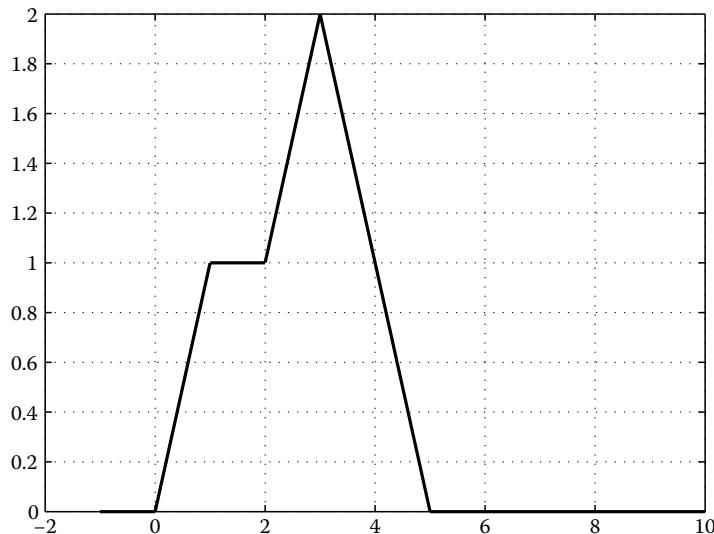


Solution

The signal is $x(t) = r(t) - r(t - 1) - r(t - 2)$. Hence, the MATLAB code is

```
t=-2:.001:3;
x=t.*heaviside(t)-(t-1).*heaviside(t-1)-(t-2).*heaviside(t-2);
plot(t,x)
ylim([-0.1 1.1]);
```

Problem 7 Express the signal depicted in the figure as a sum of ramp functions. Plot your result for verification.

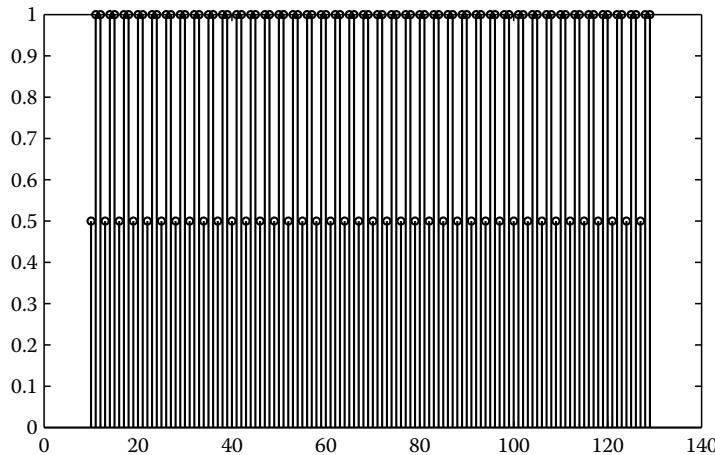


Solution

The signal is $x(t) = r(t) - r(t-1) + r(t-2) - 2r(t-3) + r(t-5)$. Hence, the MATLAB code is

```
t = -1:0.001:10;
x = t.*heaviside(t)-(t-1).*heaviside(t-1)+(t-2).*heaviside(t-2)-
2*(t-3).*heaviside(t-3)+(t-5).*heaviside(t-5);
plot(t,x);
grid;
```

Problem 8 Draw the periodic discrete-time signal $x[n]$ that is depicted in the figure. The signal $x[n]$ in one period is given by $x[n] = [0.5, 1, 1]$. Notice that the signal begins at the time instance $n = 10$.

**Solution**

```
x = [0.5 1 1];
xp = repmat(x, 1, 40);
stem(10:length(xp)+9, xp);
```

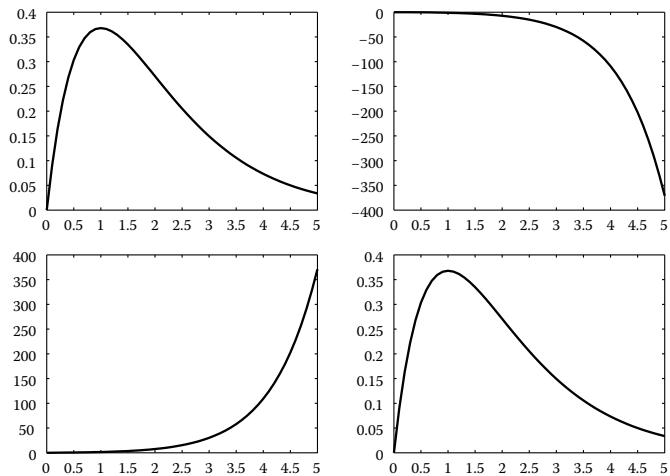
Problem 9 Consider the signal $x(t) = te^{-t}$, $0 \leq t \leq 5$. Plot

- The signal $x(t)$
- The even decomposition $x_e(t)$ of $x(t)$
- The odd decomposition $x_o(t)$ of $x(t)$
- The signal $y(t) = x_e(t) + x_o(t)$

Solution

The even decomposition $x_e(t)$ is computed according to $x_e(t) = (x(t) + x(-t))/2 = (te^{-t} + (-te^t))/2 = t(e^{-t} - e^t)/2$, while the odd decomposition $x_o(t)$ is given by $x_o(t) = (x(t) - x(-t))/2 = (te^{-t} - (-te^t))/2 = t(e^{-t} + e^t)/2$.

```
t = 0:.1:5;
x = t.*exp(-t);
xe = 0.5*t.* (exp(-t)-exp(t));
xo = 0.5*t.* (exp(-t)+exp(t));
subplot(221);
plot(t,x);
subplot(222);
plot(t,xe);
subplot(223);
plot(t,xo);
subplot(224);
plot(t,xe+xo);
```



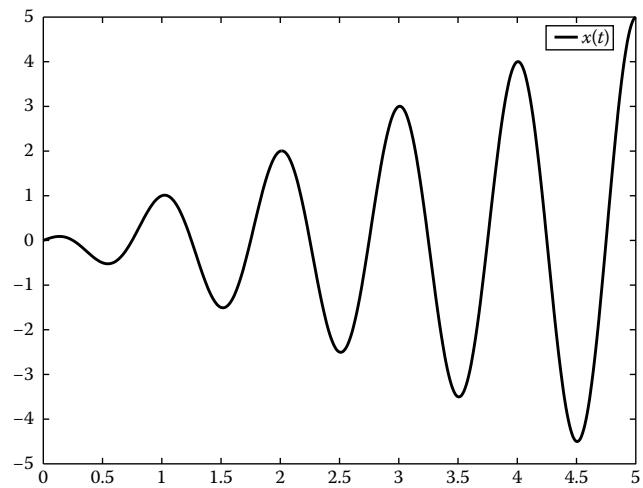
The first and the fourth graphs are same; hence, we have computed correctly the even and odd decompositions of $x(t)$.

Problem 10 Suppose that $x(t) = t \cos(2\pi t)$, $0 \leq t \leq 5$. Plot the signals

- $x(t)$
- $x(-t)$
- $x(t/5)$
- $x(1 + 3t)$
- $x(-1 - 3t)$

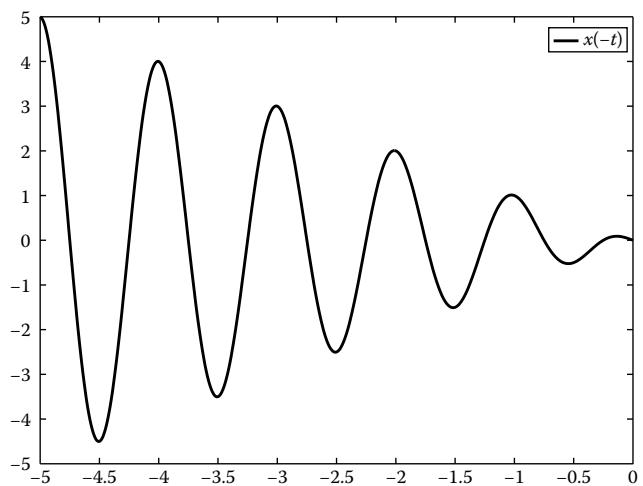
Solution

```
t = 0:0.01:5;
x = t.*cos(2*pi*t);
plot(t,x);
legend('x(t)');
```

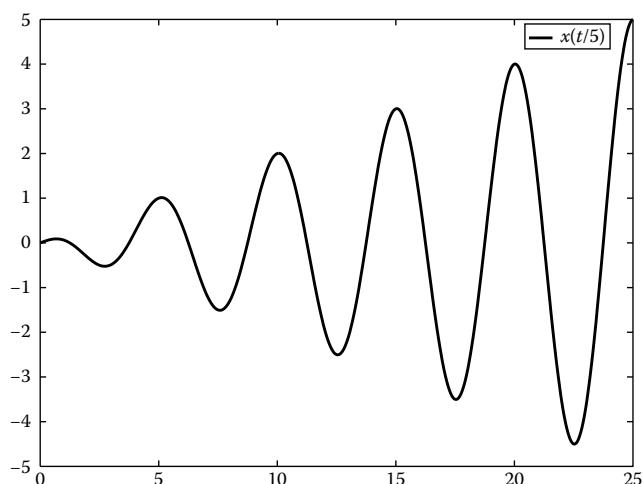


(continued)

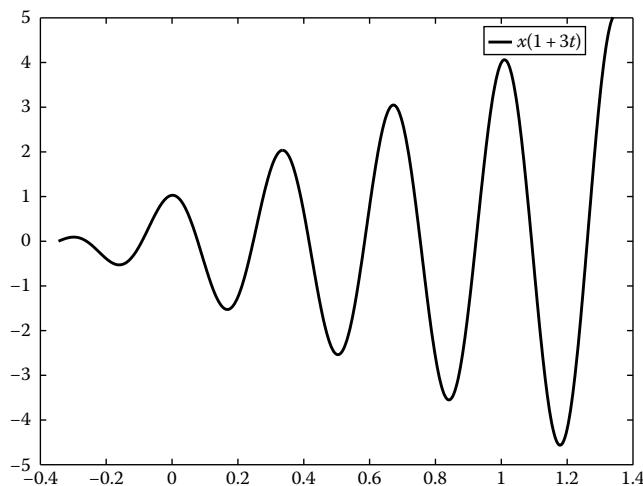
```
plot(-t,x)
legend('x(-t)');
```



```
plot(5*t,x)
legend('x(t/5)');
```



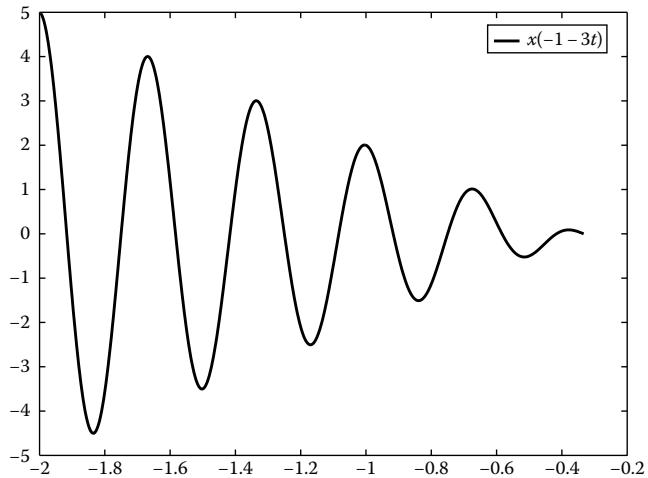
```
plot((1/3)*(-1+t),x)
legend('x(1+3t)');
```



(continued)

(continued)

```
plot(-(1/3)*(1+t),x)
legend('x(-1-3t)');
```

**Problem 11**

- Plot the unit-ramp sequence $r[n]$ in the time interval $-3 \leq n \leq 6$.
- Plot the unit-ramp sequence $r[n+1]$ in the same time interval.

Solution

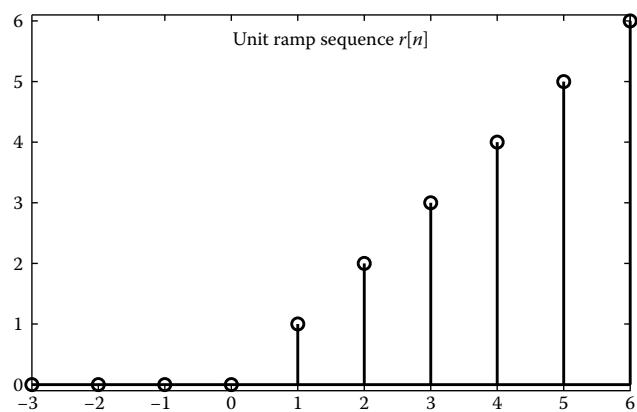
The unit-ramp sequence $r[n]$ is defined by

$$r[n] = n \cdot u[n] = \begin{cases} n, & n \geq 0 \\ 0, & n < 0 \end{cases},$$

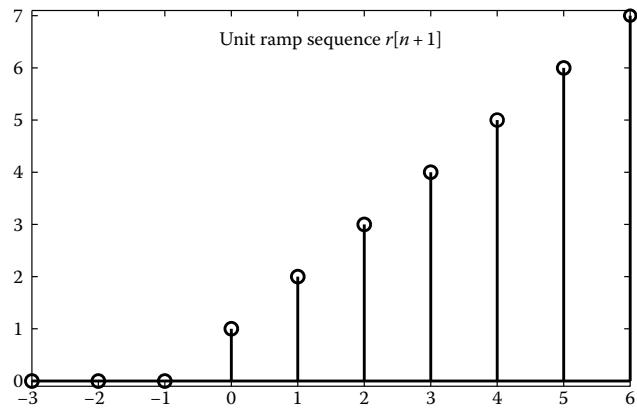
while the general form of a ramp sequence $r[n - n_0]$ is

$$r[n - n_0] = (n - n_0)u[n - n_0] = \begin{cases} n - n_0, & n \geq n_0 \\ 0, & n < n_0 \end{cases}.$$

```
a.
n=-3:6;
u=(n>=0);
r=n.*u;
stem(n,r)
title('Unit ramp sequence r[n]')
ylim([-1.1 6.1])
```



```
b.
n=-3:6;
u1=(n>=-1);
r=(n+1).*u1;
stem(n,r)
title('Unit ramp sequence r[n+1]')
ylim([-1.1 7.1])
```



2.8 Homework Problems

1. Plot in the time interval $-5 \leq t \leq 10$ the signal $x(t) = r(t+5) - r(t+3) - r(t+2) + r(t) + u(t) + u(t-2) + u(t-5) - 3u(t-8)$.
2. Plot in time of three periods the real and the imaginary parts of the signal $x(t) = 3e^{-j2t}$.
3. Plot in four periods the signal $x(t) = \cos(2\pi t) + \sin(3\pi t)$.
4. Consider the signal $x(t) = te^{-0.1t} \cos(t)$, $0 \leq t \leq 20$. Plot
 - The signal $x(t)$
 - The even decomposition $x_e(t)$ of $x(t)$
 - The odd decomposition $x_o(t)$ of $x(t)$
 - The signal $y(t) = x_e(t) + x_o(t)$

5. Compute the energy of the discrete-time signal $x[n] = 0.9^n u[n]$.
6. Suppose that $x(t) = \begin{cases} t, & 0 \leq t \leq 2 \\ 4-t, & 2 < t \leq 4 \end{cases}$. Plot the signals $x(t)$, $x(-t)$, $x(t/2)$, $x(2+4t)$, and $x(-2-4t)$.
7. Write a function that accepts a number t_0 as input argument and returns the unit step function $u(t - t_0)$. Your function should also create the graph of $u(t - t_0)$ in a proper time interval.
8. Write a function that accepts a number t_0 as input argument and returns the Dirac function $\delta(t - t_0)$. Your function should also create the graph of $\delta(t - t_0)$ in a proper time interval.
9. Write a function that accepts a number t_0 as input argument and returns the ramp function $r(t - t_0)$. Your function should also create the graph of $r(t - t_0)$ in a proper time interval.
10. Write a function that accepts a number T as input argument and returns the rectangular function $pT(t)$. Your function should also create the graph of $pT(t)$ in a proper time interval.
11. Write a function that accepts the numbers T and t_0 as input arguments and returns the rectangular function $pT(t - t_0)$. Your function should also create the graph of $pT(t - t_0)$ in a proper time interval.
12. Write a function that accepts a number n_0 as input argument and returns the unit step sequence $u[n - n_0]$. Your function should also create the graph of $u[n - n_0]$ in a proper interval.
13. Write a function that accepts a number n_0 as input argument and returns the unit impulse sequence $\delta[n - n_0]$. Your function should also create the graph of $\delta[n - n_0]$ in a proper interval.
14. Write a function that accepts a signal $x[n]$, the discrete time n and a number n_0 as input arguments, and returns the shifted signal $x[n - n_0]$. Your function should also create the graph of $x[n - n_0]$ in a proper interval.
15. Write a function that accepts a signal $x[n]$ and a number a as input arguments and returns the scaled signal $x[a \cdot n]$. Your function should also create the graph of $x[a \cdot n]$ in a proper interval.
16. Write a function that accepts two numbers r , and ω , and a time interval n as input arguments and returns the complex exponential sequence $x[n] = r^n e^{j\omega n}$. Your function should also plot the real part, the imaginary part, the magnitude, and the phase of $x[n]$.

3

Systems

In Chapter 2, we discussed the basic continuous-time and discrete-time signals. In this chapter, we introduce the concept of systems. The main categories in which systems are classified are given and the basic properties of systems are described. A system is a concept with a very wide meaning used in various expressions of everyday life. From an electrical engineering perspective, a system is an entity that manipulates one or more signals to perform an operation and returns the operation result as one or more signals. A more formal definition is that a continuous-time or discrete-time system is an entity that transforms an input signal $x(t)$ or $x[n]$ into an output signal $y(t)$ or $y[n]$ according to a specified operation. The transformation of $x(t)$ or $x[n]$ is denoted by $y(t) = S\{x(t)\}$ or $y[n] = S\{x[n]\}$, where S denotes the system. In simple words, one can say that a system responds to an input signal $x(t)$ or $x[n]$ by an output signal $y(t)$ or $y[n]$ (see Figures 3.1 and 3.2).

From a signals and systems perspective, a system is considered a *black box*, which means that it is examined in terms of its input, output, and properties without the need to know its internal working. An example of a system is an electrical guitar. Consider the music note played by the guitarist as the input signal. The electrical guitar, the amplifier, and the sound boxes constitute the system. The output signal is the actual sound that the audience hears. The output signal is different (louder, i.e., has a higher amplitude and usually a longer duration) from the input signal. Considering a black-box approach, we are interested in the input and the output signals and not the internal implementation of the system (e.g., the circuit of the amplifier).

3.1 Systems Classification

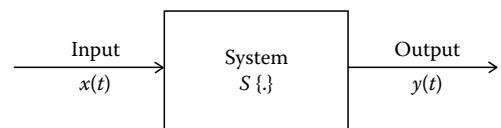
In this section, systems are divided into categories according to some criteria.

3.1.1 Classification according to the Number of Inputs and Outputs

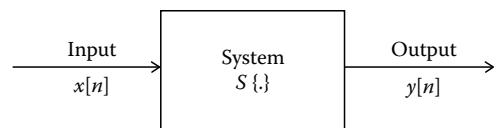
The first criterion is the number of input signals that the system accepts and the number of output signals it returns.

- Single-input single-output (SISO) systems

This is the simpler category. A continuous-time SISO system accepts a single input signal and returns a single-output signal. A SISO system is depicted in Figure 3.1. A typical input/output (i/o) relationship that describes a SISO system is a relationship of the form $y(t) = a \cdot x(t - t_0)$, where a is scalar, t_0 is the time delay, $x(t)$ is the input signal, and $y(t)$ is the response of the system to the input signal.

**FIGURE 3.1**

Block diagram representation of a continuous-time system.

**FIGURE 3.2**

Block diagram representation of a discrete-time system.

Example

The i/o relationship that describes a system S is $y(t) = x(t - 2)$. Compute and plot the system response to the input signal $x(t) = t \cos(2\pi t)$, $0 \leq t \leq 3$.

Commands	Results	Comments
<pre>t = 0:.01:3; x = t.*cos(2*pi*t); plot(t,x); title('Input signal x(t)')</pre>		Graph of the input signal $x(t)$.
<pre>plot(t+2,x); title('Output signal y(t)')</pre>		Graph of the output signal $y(t)$.

- Multiple-input single-output (MISO) systems accept multiple inputs but return a single output. For example, a MISO system is described by an i/o relationship of the form $y(t) = x_1(t) + x_2(t)$ or $y(t) = x_1(t) \cdot x_2(t) \cdot x_3(t)$. A MISO system is illustrated in Figure 3.3.

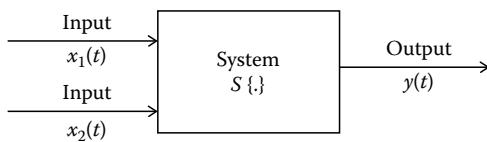


FIGURE 3.3
Block diagram representation of a MISO system.

Example

Suppose that a MISO system S is described by the i/o relationship $y(t) = x_1(t) + x_2(t) \cdot x_3(t)$. Compute and plot the system's output if the input signals are given by $x_1(t) = u(t) - u(t - 3)$, $x_2(t) = t \sin(t)$, $0 \leq t \leq 4$, and $x_3(t) = t \cos(t)$, $0 \leq t \leq 4$.

Commands	Results	Comments
<pre>t = 0:.01:4; x1 = heaviside(t) - heaviside(t-3); x2 = t.*sin(t); x3 = t.*cos(t); plot(t,x1,t,x2,:',t,x3,'-'); legend('x1(t)', 'x2(t)', 'x3(t)')</pre>		Graph of the input signals $x_1(t)$, $x_2(t)$, and $x_3(t)$.
<pre>y=x1+x2.*x3; plot(t,y); legend('Output y(t)')</pre>		Graph of the system response $y(t)$.

- Single-input multiple-output (SIMO) systems accept a single input and return multiple outputs. For example, a SIMO system is described by the i/o relationships $y_1(t) = x_1^2(t)$ and $y_2(t) = 1 - 3x_1(t)$. The graphic representation of a SIMO system is given in Figure 3.4.
- Multiple-input multiple-output (MIMO) systems accept multiple inputs and also return multiple outputs. For example, a MIMO system is described by the i/o relationships $y_1(t) = x_1(t) + x_2(t)$ and $y_2(t) = x_1(t) \cdot x_2(t) \cdot x_3(t)$. The block diagram representation of a MIMO system is given in Figure 3.5.

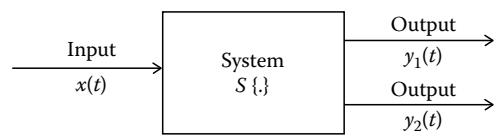


FIGURE 3.4
Block diagram representation of a SIMO system.

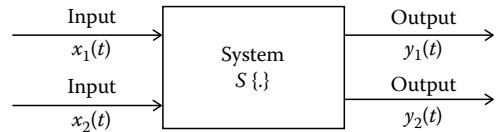


FIGURE 3.5
Block diagram representation of a MIMO system.

Example

Suppose that a MIMO system is described by the i/o relationships $y_1(t) = x_1(t) + x_2(t)$ and $y_2(t) = x_1(t) - x_2(t)$. Compute and plot the system response to the input signals $x_1(t) = u(t)$ and $x_2(t) = 0.5 \cdot u(t - 1)$.

Commands	Results	Comments																					
<pre>t = 0:.01:4; x1 = heaviside(t); x2 = 0.5*heaviside(t-1); plot(t,x1,t,x2,':'); ylim([-0.1 1.4]); legend('x1(t)', 'x2(t)')</pre>	<table border="1"> <caption>Data for Input Signals</caption> <thead> <tr> <th>t</th> <th>x1(t)</th> <th>x2(t)</th> </tr> </thead> <tbody> <tr><td>0.0</td><td>0.0</td><td>0.0</td></tr> <tr><td>0.5</td><td>1.0</td><td>0.0</td></tr> <tr><td>1.0</td><td>1.0</td><td>0.5</td></tr> <tr><td>2.0</td><td>1.0</td><td>0.5</td></tr> <tr><td>3.0</td><td>1.0</td><td>0.5</td></tr> <tr><td>4.0</td><td>1.0</td><td>0.5</td></tr> </tbody> </table>	t	x1(t)	x2(t)	0.0	0.0	0.0	0.5	1.0	0.0	1.0	1.0	0.5	2.0	1.0	0.5	3.0	1.0	0.5	4.0	1.0	0.5	Graph of input signals.
t	x1(t)	x2(t)																					
0.0	0.0	0.0																					
0.5	1.0	0.0																					
1.0	1.0	0.5																					
2.0	1.0	0.5																					
3.0	1.0	0.5																					
4.0	1.0	0.5																					
<pre>y1=x1+x2; y2=x1-x2; plot(t,y1,t,y2,':'); ylim([-0.1 2]); legend('y1(t)', 'y2(t)')</pre>	<table border="1"> <caption>Data for Output Signals</caption> <thead> <tr> <th>t</th> <th>y1(t)</th> <th>y2(t)</th> </tr> </thead> <tbody> <tr><td>0.0</td><td>1.0</td><td>0.0</td></tr> <tr><td>0.5</td><td>1.0</td><td>0.0</td></tr> <tr><td>1.0</td><td>1.5</td><td>0.5</td></tr> <tr><td>2.0</td><td>1.5</td><td>0.5</td></tr> <tr><td>3.0</td><td>1.5</td><td>0.5</td></tr> <tr><td>4.0</td><td>1.5</td><td>0.5</td></tr> </tbody> </table>	t	y1(t)	y2(t)	0.0	1.0	0.0	0.5	1.0	0.0	1.0	1.5	0.5	2.0	1.5	0.5	3.0	1.5	0.5	4.0	1.5	0.5	Graph of output signals.
t	y1(t)	y2(t)																					
0.0	1.0	0.0																					
0.5	1.0	0.0																					
1.0	1.5	0.5																					
2.0	1.5	0.5																					
3.0	1.5	0.5																					
4.0	1.5	0.5																					

3.1.2 Continuous-Time and Discrete-Time Signals

A second classification of systems is related to the “nature” of the input and output signals. More specifically, a continuous-time system is a system in which continuous-time input signals result in continuous-time output signals. On the other hand, a discrete-time system is a system in which the system response to discrete-time input signal(s) is discrete-time output signal(s). Furthermore, there are also hybrid systems in which a continuous-time input signal is transformed into a discrete-time output signal and vice versa. An analog to digital (A/D) converter is an example of a hybrid system.

3.1.3 Deterministic and Stochastic Systems

A third classification of systems is between stochastic and deterministic systems. A system is deterministic if no randomness is involved in the input and output signals. A nondeterministic system is called stochastic. A stochastic system (unlike a deterministic one) does not always produce the same output for a given input.

3.2 Properties of Systems

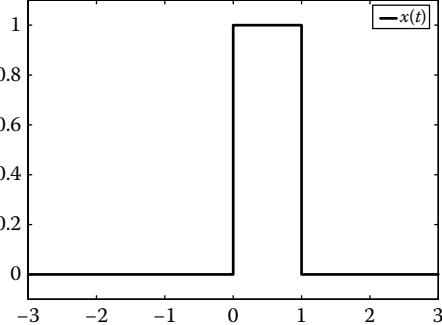
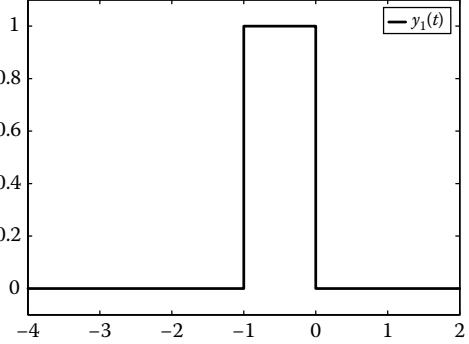
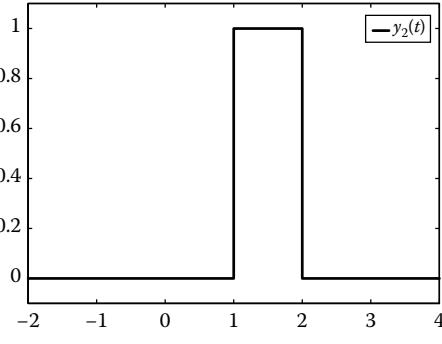
In this section, we introduce the basic system properties. An illustrative example accompanies each property. Of course, an example does not prove a property. However, the illustrated examples are carefully selected in order to correspond with the introduced property. The following properties apply to both continuous-time and discrete-time systems.

3.2.1 Causal and Noncausal Systems

A system is causal if the system output $y(t_0)$ at time $t = t_0$ does not depend on values of the input $x(t)$ for $t > t_0$. In other words, for any input signal $x(t)$, the corresponding output $y(t)$ depends only on the present and past values of $x(t)$. So, if the input to a causal system is zero for $t < t_0$ the output of this system is also zero for $t < t_0$. Correspondingly, a discrete-time system is causal if its output $y[n_0]$ at time $n = n_0$ depends only on the values of the input signal $x[n]$ for $n \leq n_0$. All natural systems are causal. However, in engineering there are many noncausal systems. For example, off-line data processing is a noncausal system.

Example

Suppose that a system S_1 is described by the i/o relationship $y(t) = x(t + 1)$ while the i/o relationship of a system S_2 is given by $y(t) = x(t - 1)$. Using the input signal $x(t) = u(t) - u(t - 1)$ find out if the two systems are causal.

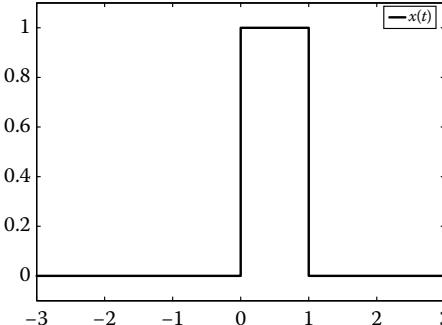
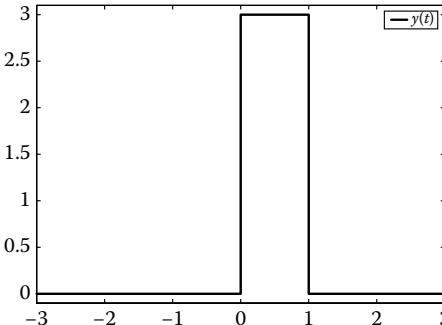
Commands	Results	Comments
<pre>t1=-3:.1:0; x1=zeros(size(t1)); t2=0:.1:1; x2=ones(size(t2)); t3=1:.1:3; x3=zeros(size(t3)); t=[t1 t2 t3]; x=[x1 x2 x3]; plot(t,x); ylim([-0.1 1.1]); legend('x(t)')</pre>		Definition and graph in the time interval $-3 \leq t \leq 3$ of the input signal $x(t) = u(t) - u(t - 1) = \begin{cases} 1, & 0 \leq t \leq 1 \\ 0, & \text{elsewhere} \end{cases}$.
<pre>plot(t-1,x) ylim([-0.1 1.1]); legend('y_1(t)')</pre>		The output of S_1 is given by $y(t) = x(t + 1)$. The input $x(t)$ is zero for $t < 0$ but the output $y(t)$ is nonzero for $t < 0$, i.e., $y(t)$ depends on future values of $x(t)$; thus system S_1 is <i>not</i> causal.
<pre>plot(t+1,x) ylim([-0.1 1.1]); legend('y_2(t)')</pre>		The output of S_2 is given by $y(t) = x(t - 1)$. The output is zero for $t < 1$, i.e., $y(t)$ depends only on past values of $x(t)$; thus system S_2 is causal.

3.2.2 Static (Memoryless) and Dynamic (with Memory) Systems

A system is static or memoryless if for any input signal $x(t)$ or $x[n]$ the corresponding output $y(t)$ or $y[n]$ depends only on the value of the input signal at the same time. A nonstatic system is called dynamic or dynamical.

Example

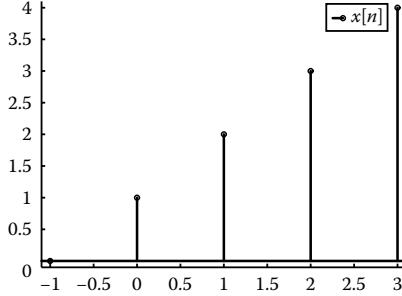
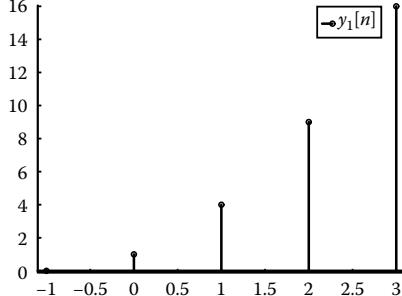
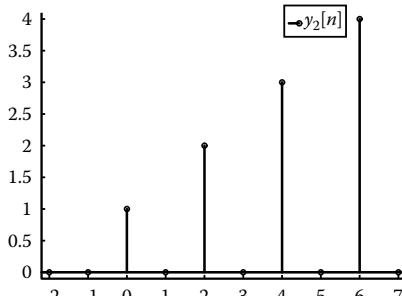
Using the input signal $x(t) = u(t) - u(t - 1)$ find out if the systems described by the i/o relationships $y(t) = 3x(t)$ and $y(t) = x(t) + x(t - 1)$ are static or dynamic.

Commands	Results	Comments
<pre>t1=-3:.1:0; x1=zeros(size(t1)); t2=0:.1:1; x2=ones(size(t2)); t3=1:.1:3;x3 =zeros(size(t3)); t=[t1 t2 t3]; x=[x1 x2 x3]; plot(t,x); ylim([-0.1 1.1]); legend('x(t)')</pre>		<p>Definition and graph in the time interval $-3 \leq t \leq 3$ of the input signal $x(t) = u(t) - u(t - 1) = \begin{cases} 1, & 0 \leq t \leq 1 \\ 0, & \text{elsewhere} \end{cases}$.</p>
<pre>plot(t,3*x); ylim([-0.1 3.1]); legend('y(t)')</pre>		<p>The output of the system with i/o relationship $y(t) = 3x(t)$ depends only on the value of the input at the same time. Hence, it is a static (or memoryless) system.</p>

In order to determine if the second system described by the i/o relationship $y(t) = x(t) + x(t - 1)$ is static or dynamic, recall that $x(t) = u(t) - u(t - 1) = 1, 0 \leq t \leq 1$; thus $x(t - 1) = u(t - 1) - u(t - 2) = 1, 1 \leq t \leq 2$ and so $y(t) = u(t) - u(t - 2) = 1, 0 \leq t \leq 2$. The values of $y(t)$ depend on past values of $x(t)$ so the system is dynamic.

Example

Determine if the discrete-time systems described by the i/o relationships $y[n] = x^2[n]$ and $y[n] = x[n/2]$ are static or dynamic. Use the input signal $x[n] = [0 1 2 3 4], -1 \leq n \leq 3$.

Commands	Results	Comments
<pre>n=-1:3; x=[0 1 2 3 4]; stem(n,x); axis([-1.1 3.1 -.1 4.1]); legend('x[n]')</pre>		Graph of the discrete-time input signal $x[n] = [0 1 2 3 4]$, $-1 \leq n \leq 3$.
<pre>y=x.^2; stem(n,y); axis([-1.1 3.1 -.1 16.1]); legend('y_1[n]')</pre>		Graph of the system output $y[n] = x^2[n]$. Each value of $y[n]$ depends only on the value of $x[n]$ for the same n . Hence, the system is static.
<pre>a=1/2; y=upsample(x,1/a) stem(-2:7,y) axis([-2.2 7.2 -.1 4.1]); legend('y_2[n]')</pre>		The response $y[n]$ of the system described by the i/o relationship $y[n] = x[n/2]$ is computed by upsampling the input signal $x[n]$. This is clearly a system with memory (dynamic) as, for example, the value of $y[n]$ for $n = 6$ depends on the value of $x[n]$ for $n = 3$.

Remark

All static systems are causal.

3.2.3 Linear and Nonlinear Systems

Let $y(t)$ denote the response of a system S to an input signal $x(t)$, that is, $y(t) = S\{x(t)\}$. System S is linear if for any input signals $x_1(t)$ and $x_2(t)$ and any scalars a_1 and a_2 the following relationship holds:

$$S\{a_1x_1(t) + a_2x_2(t)\} = a_1S\{x_1(t)\} + a_2S\{x_2(t)\}. \quad (3.1)$$

In other words, the response of a linear system to an input that is a linear combination of two signals is the linear combination of the responses of the system to each one of these signals. The linearity property is generalized for any number of input signals, and this is often referred to as the principle of superposition. The linearity property is a combination of two other properties: the additivity property and the homogeneity property. A system S satisfies the additivity property if for any input signals $x_1(t)$ and $x_2(t)$

$$S\{x_1(t) + x_2(t)\} = S\{x_1(t)\} + S\{x_2(t)\}, \quad (3.2)$$

while the homogeneity property implies that for any scalar a and any input signal $x(t)$,

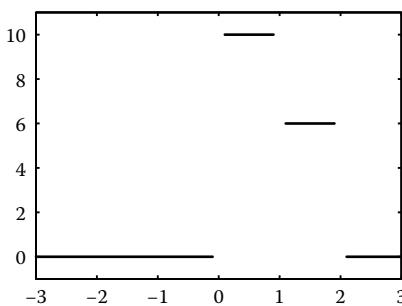
$$S\{ax(t)\} = aS\{x(t)\}. \quad (3.3)$$

Example

Let $x_1(t) = u(t) - u(t - 1)$ and $x_2(t) = u(t) - u(t - 2)$ be input signals to the systems described by the i/o relationships $y(t) = 2x(t)$ and $y(t) = x^2(t)$. Determine if the linearity property holds for these two systems.

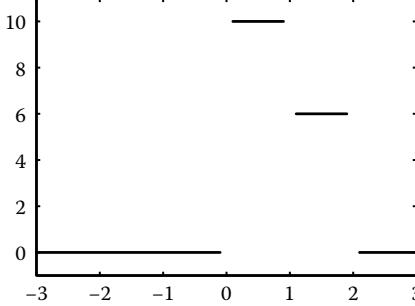
To examine if the systems are linear, we use the scalars $a_1 = 2$ and $a_2 = 3$. The time interval considered is $-3 \leq t \leq 3$.

For the system described by the i/o relationship $y(t) = 2x(t)$ the procedure followed is

Commands	Results	Comments
$t = -3 : 1 : 3;$		Definition of the input signals $x_1(t)$ and $x_2(t)$.
$x1 = heaviside(t) - heaviside(t-1);$ $x2 = heaviside(t) - heaviside(t-2);$		
% Computation of the left side of Equation 3.1.		
$a1 = 2;$ $a2 = 3;$ $z = a1*x1 + a2*x2;$		The expression $a_1x_1(t) + a_2x_2(t)$ is defined.
$y = 2*z;$ $plot(t, y);$ $ylim([-1 11]);$		The left side of Equation 3.1, namely, $S\{a_1x_1(t) + a_2x_2(t)\}$ is computed and the result is plotted.

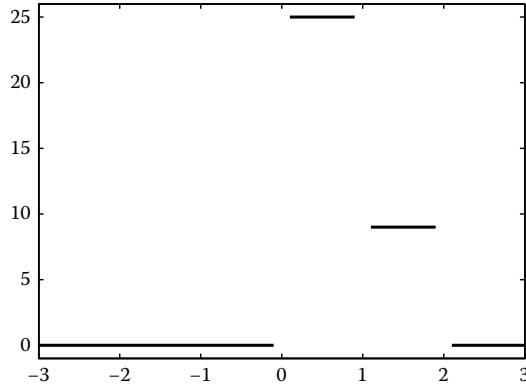
(continued)

(continued)

Commands	Results	Comments
% Computation of the right side of Equation 3.1. $\begin{aligned} z1 &= 2*x1; \\ z2 &= 2*x2; \\ y &= a1*z1 + a2*z2; \\ \text{plot}(t, y); \\ \text{ylim}([-1 11]); \end{aligned}$		Definition of $S\{x_1(t)\}$ and $S\{x_2(t)\}$. The right side of Equation 3.1, namely, $a_1S\{x_1(t)\} + a_2S\{x_2(t)\}$, is computed and the result is plotted.

The two graphs obtained are identical, hence, the two sides of Equation 3.1 are equal. Therefore, the system described by the i/o relationship $y(t) = 2x(t)$ is linear.

Next we examine if the linearity property holds for the system with i/o relationship $y(t) = x^2(t)$. The procedure followed is the same as the previous one.

Commands	Results	Comments
% Computation of the left side of Equation 3.1. $\begin{aligned} t &= -3:1:3; \\ x1 &= \text{heaviside}(t) - \text{heaviside}(t-1); \\ x2 &= \text{heaviside}(t) - \text{heaviside}(t-2); \\ a1 &= 2; \\ a2 &= 3; \\ z &= a1*x1 + a2*x2; \\ y &= z.^2; \\ \text{plot}(t, y); \\ \text{ylim}([-1 26]); \end{aligned}$		Definition of the input signals $x_1(t)$ and $x_2(t)$. The expression $a_1x_1(t) + a_2x_2(t)$ is defined. The left side of Equation 3.1, namely, $S\{a_1x_1(t) + a_2x_2(t)\}$, is computed and the result is plotted.

(continued)

Commands	Results	Comments
% Computation of the right side of Equation 3.1. z1=x1.^2; z2=x2.^2; y=a1*z1+a2*z2; plot(t,y); ylim([-1 6]);		Definition of $S\{x_1(t)\}$ and $S\{x_2(t)\}$. The right side of Equation 3.1, namely, $a_1S\{x_1(t)\} + a_2S\{x_2(t)\}$, is computed and the result is plotted.

The two obtained graphs are not alike, hence, the linearity property is not fulfilled. Consequently the system with i/o relationship $y(t) = x^2(t)$ is *not* linear.

Example

Determine if the linearity property holds for the discrete-time systems described by the i/o relationships $y[n] = 2^{x[n]}$ and $y[n] = nx[n]$. Consider the input signals $x_1[n] = 0.8^n$, $0 \leq n \leq 5$ and $x_2[n] = \cos(n)$, $0 \leq n \leq 5$.

The linearity property holds for a discrete-time system if

$$S\{a_1x_1[n] + a_2x_2[n]\} = a_1S\{x_1[n]\} + a_2S\{x_2[n]\}. \quad (3.4)$$

First, the system with i/o relationship $y[n] = 2^{x[n]}$ is examined.

Commands	Results	Comments
n=0:5; x1=0.8.^n; x2=cos(n); a1=2; a2=3; z=a1*x1+a2*x2; y1=2.^z	y1 = 32.0000 9.3237 1.0221 0.2595 0.4532 2.8409	Computation of the left side of (3.4).

z1=2.^x1; z2=2.^x2; y2=a1*z1+a2*z2	y2 = 10.0000 7.8450 5.3649 4.3625 4.5637 6.1618	Computation of the right side of (3.4). The two sides are not equal, hence, the system with i/o relationship $y[n] = 2^{x[n]}$ is not linear.
--	--	--

The same procedure is followed to examine the second system with i/o $y[n] = nx[n]$.

Commands	Results	Comments
<pre>n=0:5; x1=0.8.^n; x2=cos(n); a1=2; a2=3; z=a1*x1+a2*x2; y1=n.*z</pre>	<pre>y1 = 0 3.2209 0.0631 -5.8379 -4.5669 7.5317</pre>	Computation of the left side of (3.4).
<pre>z1=n.*x1; z2=n.*x2; y2=a1*z1+a2*z2</pre>	<pre>y2 = 0 3.2209 0.0631 -5.8379 -4.5669 7.5317</pre>	Computation of the right side of (3.4). The two sides are equal, hence, the system with i/o $y[n] = nx[n]$ is linear.

3.2.4 Time-Invariant and Time-Variant Systems

A system is time invariant, if a time shift in the input signal results in the same time shift in the output signal. In other words, if $y(t)$ is the response of a time-invariant system to an input signal $x(t)$, then the system response to the input signal $x(t - t_0)$ is $y(t - t_0)$. The mathematical expression is

$$y(t - t_0) = S\{x(t - t_0)\}. \quad (3.5)$$

Equivalently, a discrete-time system is time or (more appropriately) shift invariant if

$$y[n - n_0] = S\{x[n - n_0]\}. \quad (3.6)$$

From Equations 3.5 and 3.6, we conclude that if a system is time invariant, the amplitude of the output signal is the same independent of the time instance the input is applied. The difference is a time shift in the output signal. A non-time-invariant system is called time-varying or time-variant system.

Example

Suppose that the response of a system S to an input signal $x(t)$ is $y(t) = te^{-t}x(t)$. Determine if this system is time invariant by using the input signal $x(t) = u(t) - u(t - 5)$.

In order to determine if the system is time invariant, first we compute and plot the system response $y(t)$ to the given input signal $x(t) = u(t) - u(t - 5)$. Next the computed output $y(t)$ is shifted by 3 units to the right to represent the signal $y_1(t) = y(t - 3)$. This process corresponds to the left side of Equation 3.5. As for the right side of Equation 3.5 first the

input signal $x(t)$ is shifted 3 units to the right in order to represent the signal $x(t - 3)$. Next the system response $y_2(t) = S\{x(t - 3)\}$ is computed and plotted. If the two derived system responses are equal, the system under consideration is time invariant.

Commands	Results	Comments
<pre>t=-5:.001:10; p=heaviside(t)-heaviside(t-5); y=t.*exp(-t).*p; plot(t,y) ylim([-0.05 .4]); legend('y(t)')</pre>		The response $y(t)$ of the system to the input signal $x(t) = u(t) - u(t - 5)$ is $y(t) = te^{-t}[u(t) - u(t - 5)] = te^{-t}$, $0 \leq t \leq 5$.
<pre>plot(t+3,y) ylim([-0.05 .4]); legend('y(t-3)')</pre>		The output signal $y(t)$ is shifted 3 units to the right in order to obtain the signal $y_1(t) = y(t - 3)$.

The input signal, $x_2(t) = x(t - 3)$ is given by $x_2(t) = u(t - 3) - u(t - 8)$. Thus the system response $y_2(t) = S\{x_2(t)\} = S\{x(t - 3)\}$ is computed as $y_2(t) = te^{-t} [u(t - 3) - u(t - 8)]$.

Commands	Results
<pre>t=-5:.001:10; p=heaviside(t-3)-heaviside(t-8); y2=t.*exp(-t).*p; plot(t,y2) ylim([-0.01 .2]); legend('S[x(t-3)]')</pre>	

The two obtained graphs are not alike; thus the system described by the i/o relationship $y(t) = te^{-t}x(t)$ is time variant. A rule of thumb is that if the output of the system depends on time t outside of $x(t)$ the system is not time invariant.

Example

Consider a system described by the i/o relationship $y(t) = 1 - 2x(t - 1)$. Determine if this is a time-invariant system by using the input signal $x(t) = \cos(t)[u(t) - u(t - 10)]$.

The system response to the input signal $x(t) = \cos(t)[u(t) - u(t - 10)]$ is $y(t) = 1 - 2\cos(t - 1)[u(t - 1) - u(t - 11)]$.

First $y(t)$ is defined and plotted. After that, the computed output $y(t)$ is shifted 4 units to the right to represent the signal $y_1(t) = y(t - 4)$.

Commands	Results	Comments
<pre>t = -5:.01:20; p = heaviside(t-1) - heaviside(t-11); y = 1-2*cos(t-1).*p; plot(t,y) legend('y(t)')</pre>		The system response $y(t)$ to the input $x(t)$ is defined and plotted.
<pre>p = heaviside(t) - heaviside(t-10); x = cos(t).*p; plot(t+1,1-2*x) legend('y(t)')</pre>		Alternative way of plotting the signal $y(t)$ in terms of $x(t)$.

(continued)

Commands	Results	Comments
<pre>plot(t+4,y) legend('y(t-4)')</pre>		The shifted by 4 units to the right output signal $y(t - 4)$.

Next the signal $x(t)$ is shifted 4 units to the right to represent the signal $x(t - 4)$. The response of the system $y_2(t) = S\{x(t - 4)\}$ is computed and plotted. If the two derived system responses are equal, the system under consideration is time invariant. The shifted signal $x_2(t) = x(t - 4)$ is computed as $x_2(t) = \cos(t - 4)[u(t - 4) - u(t - 14)]$. Hence, the system response $y_2(t) = S\{x(t - 4)\}$ is given by $y_2(t) = 1 - 2\cos(t - 5)[u(t - 5) - u(t - 15)]$.

Commands	Results	Comments
<pre>p=heaviside(t-4)-heaviside(t-14); x=cos(t-4).*p; plot(t,x)</pre>		Graph of the shifted input signal $x(t - 4)$.

(continued)

(continued)

Commands	Results	Comments
<pre>p=heaviside(t-4)-heaviside(t-14); x=cos(t-4).*p; plot(t+1,1-2*x)</pre>		First way to plot the system response $y_2(t)$. The response $y_2(t) = S\{x(t-4)\}$ to the input signal $x_2(t) = x(t-4)$ is plotted in terms of $x_2(t)$.
<pre>t=-5:0.1:20; p=heaviside(t-5)-heaviside(t-15); y2=1-2*cos(t-5).*p; plot(t,y2); legend('S[x(t-4)]')</pre>		Second way to plot the system response $y_2(t)$. The response $y_2(t) = S\{x(t-4)\}$ of the system to the shifted signal $x_2(t)$ is computed as $y_2(t) = 1 - 2\cos(t-5)[u(t-5) - u(t-15)]$. The obtained graph is same as the one above.

The shifted output signal $y(t-4)$ is the same as that of the response of the system $S\{x(t-4)\}$. Therefore, the system described by the i/o relationship $y(t) = 1 - 2x(t-1)$ is time invariant.

Example

Consider a system described by the i/o relationship $y(t) = x(2t)$. Find out if this is a time-invariant system by using the input signal $x(t) = u(t+2) - u(t-2)$.

First, we compute the system response $y(t)$ to the given input signal $x(t) = u(t+2) - u(t-2)$. Next, the computed output $y(t)$ is shifted 2 units to the right to represent the signal $y(t-2)$.

Commands	Results	Comments
<pre>t=-5:.1:10; x=heaviside(t+2)-heaviside(t-2); plot(t,x); ylim([- .1 1.1]);</pre>		The input signal $x(t) = u(t+2) - u(t-2)$.
<pre>plot((1/2)*t,x); ylim([- .1 1.1]);</pre>		The system response $y(t) = x(2t)$ to the input signal $x(t) = u(t+2) - u(t-2)$.
<pre>plot((1/2)*t+2,x); ylim([- .1 1.1]); Legend('y_1(t)')</pre>		The shifted by 2 units signal $y(t-2)$. The mathematical expression for $y_1(t) = y(t-2)$ is $y_1(t) = u(t+1) - u(t+3)$.

Next, the input signal $x(t)$ is shifted 2 units to the right to represent the signal $x(t-2)$. The system response $y_2(t) = S[x(t-2)]$ is computed and plotted. If the two derived system responses are equal, the system under consideration is time invariant.

Commands	Results	Comments
<pre>plot(t+2,x); ylim([- .1 1.1]); legend('x(t-2)')</pre>		The input signal $x(t)$ is shifted by $t_0 = 2$ units to the right, i.e., we plot the signal $x(t - 2)$.
<pre>t=-5:.1:10; x2=heaviside(t)-heaviside(t-4); plot(t,x2); ylim([- .1 1.1]);</pre>		The shifted input signal $x_2(t) = x(t - 4)$ is actually given by $x_2(t) = u(t) - u(t - 4)$. The (shifted) input signal is defined and plotted for confirmation.
<pre>plot((1/2)*t,x2); ylim([- .1 1.1]); legend('y_2(t)')</pre>		The system response $y_2(t) = x_2(2t)$ is plotted. The mathematical expression for $y_2(t)$ is $y_2(t) = u(t) - u(t - 2)$.

The two derived system responses are not equal; thus the system described by the i/o relationship $y(t) = x(2t)$ is *not* time invariant.

Example

Determine if the discrete-time system described by the i/o relationship $y[n] = x^2[n]$ is shift invariant. Use the input signal $x[n] = 0.8^n(u[n] - u[n - 5])$.

The procedure in the discrete-time case is exactly the same as the one followed in the continuous-time case. So first, the computed output $y[n]$ is shifted by 2 units to the right to obtain the delayed signal $y[n - 2]$. Next, the input signal $x[n]$ is shifted 2 units to the right to obtain the delayed signal $x[n - 2]$. The system response $S\{x[n - 2]\}$ to the input signal $x[n - 2]$ is computed and if it is similar to $y[n - 2]$, the system under consideration is shift invariant.

Commands	Results	Comments
<pre>n = 0:5; x = 0.8.^n; y = x.^2; stem(n,y); xlim([-1 5.1]) legend('y[n]')</pre>		The system response $y[n] = S\{x[n]\}$.
<pre>stem(n+2,y) legend('y[n-2]') xlim([1.9 7.1])</pre>		The shifted by two units signal $y_1[n] = y[n - 2]$.
<pre>n=2:7; y2 = (0.8.^{(n-2)}).^2; stem(n,y2); xlim([1.9 7.1]) legend('S[x[n-2]]')</pre>		The system response $y_2[n] = S\{x[n - 2]\}$ to the input signal $x[n - 2]$ is $y_2[n] = 0.8^{n-2} (u[n - 2] - u[n - 7])$.

It is obvious that $y[n - 2] = S\{x[n - 2]\}$; thus the discrete-time system described by the i/o relationship $y[n] = x^2[n]$ is shift invariant.

3.2.5 Invertible and Non-Invertible Systems

A system is invertible if the input signal $x(t)$ that is applied to the system can be derived from the system response $y(t)$. In other words, a system is invertible if the i/o relationship $y(t) = S\{x(t)\}$ is one to one, namely, if different input values correspond to different output values.

Example

Determine if the systems S_1 and S_2 described by the i/o relationships $y_1[n] = 3x[n]$ and $y_2[n] = x^2[n]$, respectively, are invertible. Consider the signal $x[n] = 2n$, $-2 \leq n \leq 2$ as the input signal.

Commands	Results	Comments
<code>n=-2:2; x=2*n</code>	$x = -4 \quad -2 \quad 0 \quad 2 \quad 4$	Input signal $x[n]$.
<code>y1=3*x</code>	$y1 = -12 \quad -6 \quad 0 \quad 6 \quad 12$	The output signal $y_1[n] = 3x[n]$ of the system S_1 .
<code>y2=x.^2</code>	$y2 = 16 \quad 4 \quad 0 \quad 4 \quad 16$	The output signal $y_2[n] = x^2[n]$ of the system S_2 .

The i/o relationship of the first system is one to one as different input values result in different output values. Hence, the system described by the i/o relationship $y_1(t) = 3x(t)$ is invertible. On the other hand, the i/o relationship of the second system is not one to one as different input values result in same output values. Hence, the system described by the i/o relationship $y_2(t) = x^2(t)$ is *not* invertible.

3.2.5.1 Construction of the Inverse System

The inverse system accepts the output $y(t)$ of the invertible system as its input and returns the input signal $x(t)$ that was applied in the invertible system as its output.

Example

Derive the inverse systems of the systems described by the i/o relationships $y_1[n] = 3x[n]$ and $y_2[n] = x^2[n]$.

The i/o relationship of the first system is $z_1[n] = (1/3)y_1[n]$, while for the second system (despite the fact that it is not invertible) we will try the more reasonable i/o relationship $z_2[n] = \sqrt{y_2[n]}$. Recall that the output signals $y_1[n]$ and $y_2[n]$ of the two systems were computed in the previous example.

Commands	Results	Comments
<code>y1=[-12 -6 0 6 12];</code>		The output signal $y_1[n] = 3x[n]$ is applied as input to the inverse system.
<code>z1=(1/3)*y1</code>	$z1 = -4 \quad -2 \quad 0 \quad 2 \quad 4$	The response $z_1[n]$ of the system described by the i/o relationship $z_1[n] = (1/3)y_1[n]$ to the input signal $y_1[n]$ is equal to the original input signal $x[n]$. Hence, it is the inverse system.
<code>y2=[16 4 0 4 16];</code>		The output signal $y_2[n] = x^2[n]$, is applied as input to the inverse(?) system.

(continued)

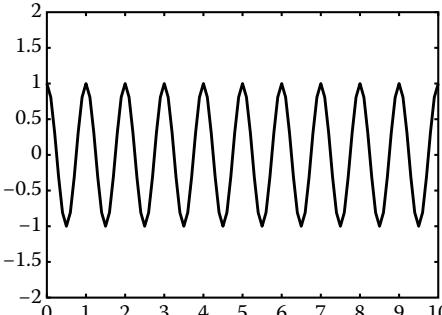
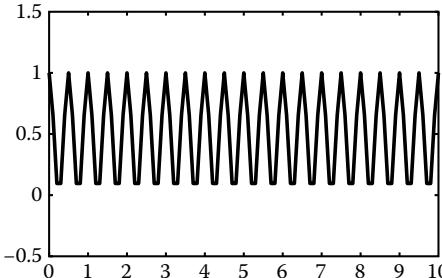
Commands	Results	Comments
<code>z2 = sqrt(y2)</code>	$\begin{array}{cccccc} z2 & = & 4 & \quad 2 & 0 & 2 & 4 \end{array}$	The response $z_2[n]$ of the system described by the i/o relationship $z_2[n] = \sqrt{y_2[n]}$ to the input signal $y_2[n]$ is <i>not</i> equal to the original input signal $x[n]$. As expected it is <i>not</i> the inverse system as the original system (with i/o $y_2[n] = x^2[n]$) is not invertible.

3.2.6 Stable and Unstable Systems

Stability is a very important system property. The practical meaning of a stable system is that for a small applied input the system response is also small (does not diverge). A more formal definition is that a system is stable or bounded-input bounded-output (BIBO) stable if the system response to any bounded-input signal is a bounded-output signal. The mathematical expression is as follows: Suppose that a positive number $M < \infty$ exists, such that $|x(t)| \leq M$. The system is stable if $\forall t \in \mathbb{R}$ a positive number $N < \infty$ exists, such that $|y(t)| \leq N$. A non-stable system is called unstable.

Example

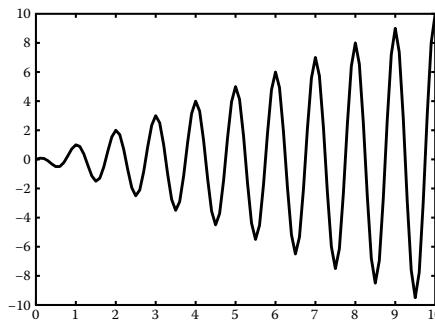
Suppose that an input signal $x(t) = \cos(2\pi t)$ is applied to two systems described by the i/o relationships $y_1(t) = x^2(t)$ and $y_2(t) = tx(t)$. Determine if these two systems are stable.

Commands	Results	Comments
<code>t = 0:.1:10;</code> <code>x = cos(2*pi*t);</code> <code>plot(t,x);</code> <code>ylim([-2 2]);</code>		Definition and graph of $x(t)$. The input signal is bounded as $-1 \leq x(t) \leq 1$, namely, $x(t)$ is bounded by $M = 1$ as $ x(t) \leq M$.
<code>y1 = x.^2;</code> <code>plot(t,y1);</code> <code>ylim([-0.5 1.5]);</code>		Definition and graph of $y_1(t)$. The output signal $y_1(t)$ is bounded as $0 \leq y_1(t) \leq 1$, namely, $y_1(t)$ is bounded by $N = 1$, as $ y_1(t) \leq N$. Hence, the system described by the i/o relationship $y_1(t) = x^2(t)$ is BIBO stable.

(continued)

(continued)

```
y2=t.*x;
plot(t,y2);
```



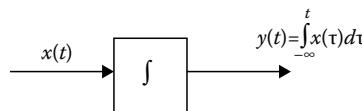
Definition and graph of $y_2(t)$. The output signal $y_2(t)$ is not bounded as its amplitude is getting larger as time passes. Hence, the system with i/o relationship $y_1(t) = tx(t)$ is not BIBO stable.

Moreover, in order to be sure about our conclusion the values of the signals must be computed when $t \rightarrow \infty$. This is easily done by employing the command `limit`.

Commands	Results	Comments
<code>syms t x=cos(2*pi*t); limit(x,t,inf)</code>	<code>ans = -1 .. 1</code>	Computation of $\lim_{t \rightarrow \infty} [x(t)]$. The value lies in the interval [-1 1]. Thus, the input signal $x(t)$ is bounded at infinity.
<code>y1=x^2; limit(y1,t,inf)</code>	<code>ans = 0 .. 1</code>	Computation of $\lim_{t \rightarrow \infty} [y_1(t)]$. The value lies in the interval [0 1]. Thus, the output signal $y_1(t)$ is bounded at infinity and the system is BIBO stable.
<code>y2=t*x; limit(y2,t,inf)</code>	<code>ans = NaN</code>	Computation of $\lim_{t \rightarrow \infty} [y_2(t)]$. The limit does not exist; thus the output signal $y_2(t)$ is not bounded and the second system is unstable.

3.3 Solved Problems

Problem 1 An integrator system is depicted in the figure below.



The i/o relationship of this system is

$$y(t) = \int_{-\infty}^t x(\tau) d\tau.$$

Determine if the integrator system is

- a. Linear or not linear
- b. Static or dynamic

- c. Causal or noncausal
- d. Time invariant or time variant
- e. Stable or unstable

Solution

- a. Regarding the linearity property

The input signals $x_1(t) = u(t) - u(t - 2)$, $x_2(t) = u(t) - u(t - 3)$, and the scalars $a_1 = 2$, $a_2 = 3$ are considered to examine the linearity property of the system.

```
syms t r
x1=heaviside(r)-heaviside(r-2);
x2=heaviside(r)-heaviside(r-3);
```

First the two input signals are defined.

The relationship that must be fulfilled in order for the system to be linear is $S\{a_1x_1(t) + a_2x_2(t)\} = a_1S\{x_1(t)\} + a_2S\{x_2(t)\}$.

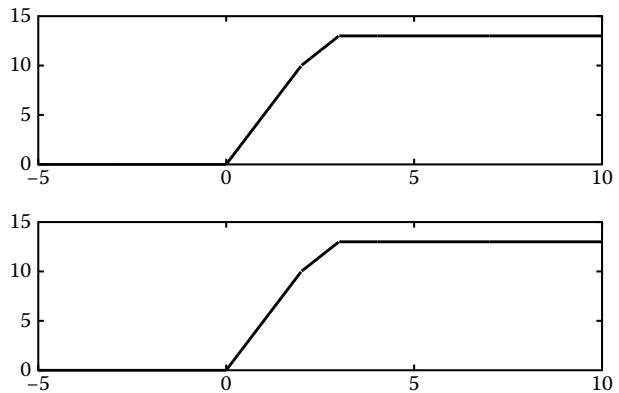
```
a1=2;
a2=3;
z=a1*x1+a2*x2;
y1=int(z,r,-inf,t);
z1=int(x1,r,-inf,t);
z2=int(x2,r,-inf,t);
y2=a1*z1+a2*z2;
```

Definition of the left side of the relationship.

Definition of the right side of the relationship.

In order to plot the two sides, the symbolic variables are substituted with vectors.

```
t=-5:0.01:10;
y1=subs(y1,t);
y2=subs(y2,t);
subplot(211);
plot(t,y1);
subplot(212);
plot(t,y2);
```



The left side is illustrated in the upper part of the figure, while the right side is depicted in the lower part. The two graphs are identical, hence, the linearity property holds for the integrator system.

b. Static or dynamic

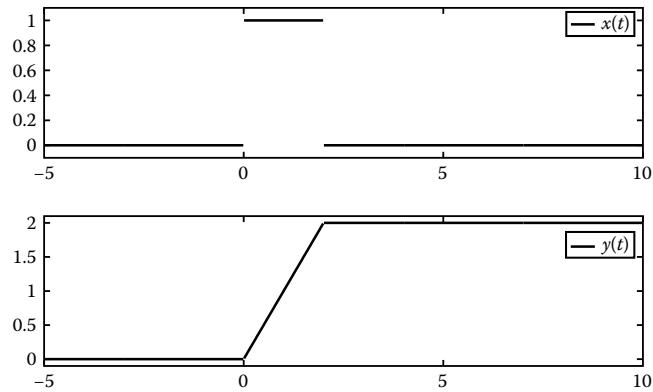
The input signal $x_1(t) = u(t) - u(t - 2)$ is used to examine if the system has memory, i.e., if it is dynamic.

Computation of the output signal $y(t)$.

```
syms t r
x=heaviside(r)-heaviside(r-2);
y=int(x,r,-inf,t);
```

In order to plot the input and output signals, the symbolic variables are substituted with vectors.

```
t=-5:0.01:10;
x=subs(x,t);
y=subs(y,t);
subplot(211);
plot(t,x);
legend('x(t)')
ylim([-0.1 1.1]);
subplot(212);
plot(t,y);
legend('y(t)')
ylim([-0.1 2.1]);
```



The output signal $y(t)$ (depicted in the lower part of the figure) depends on the previous values of the input signal. To make this point more clear, consider that the output signal corresponds to the area bounded by the graph of the input signal and the t -axis and so the output value is increasing linearly while the input signal remains constant. Therefore, we conclude that an integrator system is a dynamic system.

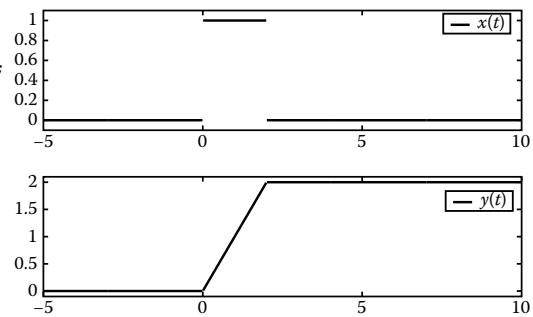
c. Causal or noncausal

The output signal $y(t)$ at t_0 is given by $y(t_0) = \int_{-\infty}^{t_0} x(\tau)d\tau$; that is, it depends only on the values of $x(t)$ for $t \leq t_0$. Thus the integrator system is a causal system. An alternative explanation is that from the graphs of $x(t)$ and $y(t)$ derived in the previous query we notice that $x(t) = 0$, $t < 0$ yields $y(t) = 0$, $t < 0$. Hence, $y(t)$ does not depend on the future values of $x(t)$; thus the system is causal.

d. Time invariant or time variant

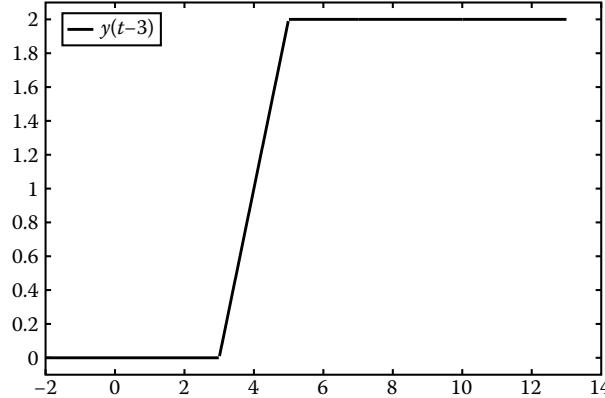
The response $y(t)$ of the system to the input signal $x_1(t) = u(t) - u(t - 2)$ is depicted in the previous figure. The mathematical expression for $y(t)$ is $y(t) = r(t) - r(t - 2) = t \cdot u(t) - (t - 2) \cdot u(t - 2)$, where $r(t)$ denotes the ramp function. For confirmation, both signals $x(t)$ and $y(t)$ are plotted.

```
t=-5:.01:10;
x=heaviside(t)-heaviside(t-2);
y=t.*heaviside(t)-(t-2).*heaviside(t-2);
subplot(211);
plot(t,x);
ylim([-0.1 1.1]);
legend('x(t)')
subplot(212);
plot(t,y);
ylim([-0.1 2.1]);
legend('y(t)')
```



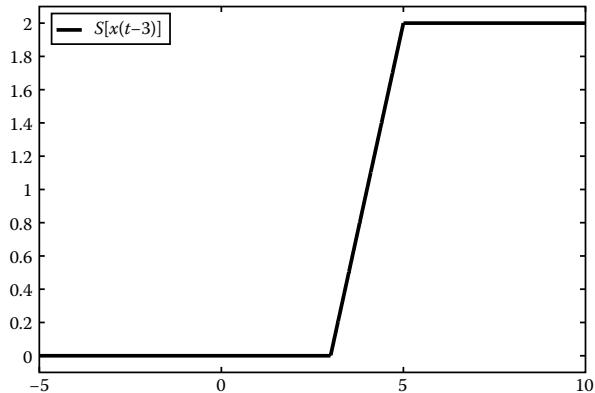
The shifted output signal is given by $y(t - 3) = (t - 3) \cdot u(t - 3) - (t - 5) \cdot u(t - 5)$. Its MATLAB® implementation and graph are

```
plot(t+3,y)
ylim([-0.1 2.1]);
legend('y(t-3)')
```



The shifted (by 3 units) input signal $x(t - 3)$ is given by $x_1(t) = u(t - 3) - u(t - 5)$. The response of the system to $x(t - 3)$ is computed as $y(t) = S\{x(t - 3)\} = \int_{-\infty}^{t-3} x(\tau)d\tau$.

```
syms t r
x=heaviside(r)-heaviside(r-2);
y=int(x,r,-inf,t-3)
tt=-5:.001:10;
y2=subs(y,t,tt);
plot(tt,y2)
ylim([-0.1 2.1]);
legend('S[x(t-3)]')
```



The two graphs are the same; thus the integrator system is time invariant.

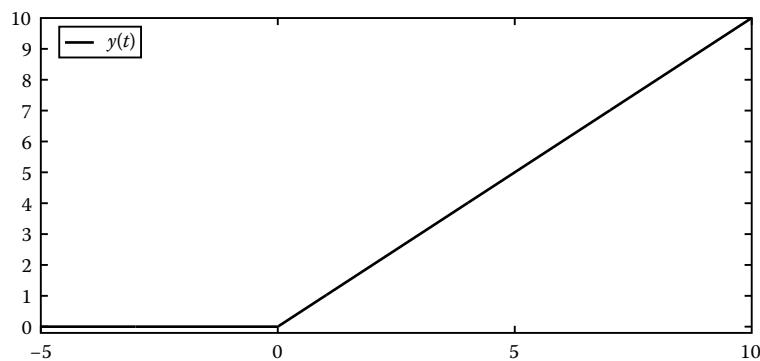
e. Stable or unstable

The stability property of the integrator system is examined by using the input signal $x(t) = u(t)$.

```
syms t r
x=heaviside(r);
y=int(x,r,-inf,t)
t=-5:.01:10;
y=subs(y,t);
plot(t,y);
legend('y(t)')
```

The system response to $x(t) = u(t)$ is computed as
 $y = \text{heaviside}(t) \cdot t$

Graph of the output signal $y(t) = tu(t) = r(t)$.



While the input signal $x(t) = u(t)$ is bounded, the output signal $y(t) = r(t) = t \cdot u(t)$ grows linearly as time passes. Probably this is an unstable system. To assure our conclusion, we furthermore compute the limit at $+\infty$.

```
syms t
y=t*heaviside(t);
limit(y,t,inf)
```

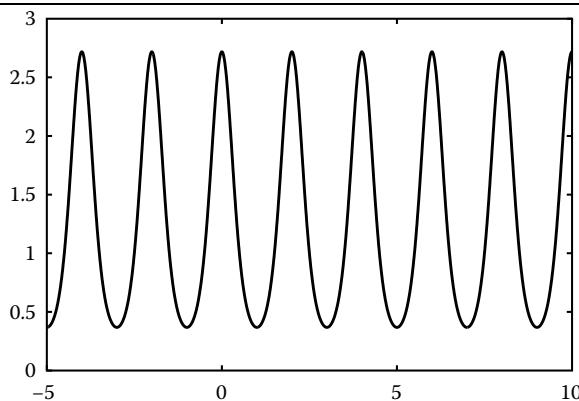
The limit is infinity; that is, the output signal is unbounded.
Therefore the integrator system is unstable.
ans = Inf

Problem 2 Determine if the system described by the i/o relationship $y(t) = e^{x(t)}$ is stable.

Solution

First the system response $y(t)$ to the bounded input $x(t) = \cos(\pi t)$ is computed and plotted while next the limits of $y(t)$ at $\pm\infty$ are derived.

```
t = -5:0.01:10;
y = exp(cos(pi*t));
plot(t,y);
```



```
syms t
x = cos(t);
y = exp(x);
limit(y,t,inf)
ans = exp(-1) .. exp(1)

limit(y,t,-inf)
ans = exp(-1) .. exp(1)
```

Computation of $\lim_{t \rightarrow \infty} [y(t)]$.
The value of the output signal when t tends to infinity lies in the interval $[\exp(-1) \exp(1)] = [0.3679 \ 2.7183]$.

Computation of $\lim_{t \rightarrow -\infty} [y(t)]$.
The value of the output signal when t tends to minus infinity also lies in the interval $[\exp(-1) \exp(1)] = [0.3679 \ 2.7183]$.

Therefore we conclude that the system under consideration is stable.

Problem 3 Find out if the system described by the i/o relationship $y(t) = x(t/4)$ is causal.

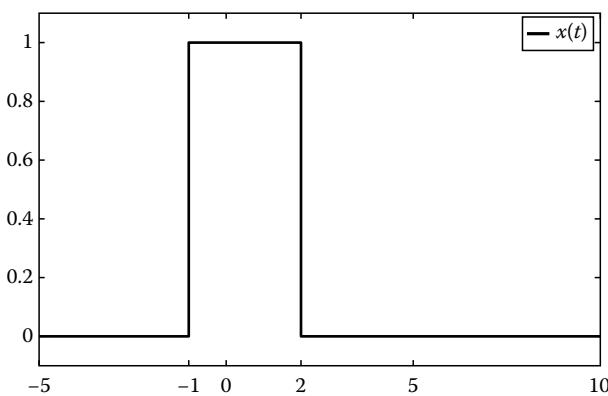
Solution

The input signal $x(t) = u(t + 1) - u(t - 2)$ is considered; thus the signal $x(t)$ is defined as

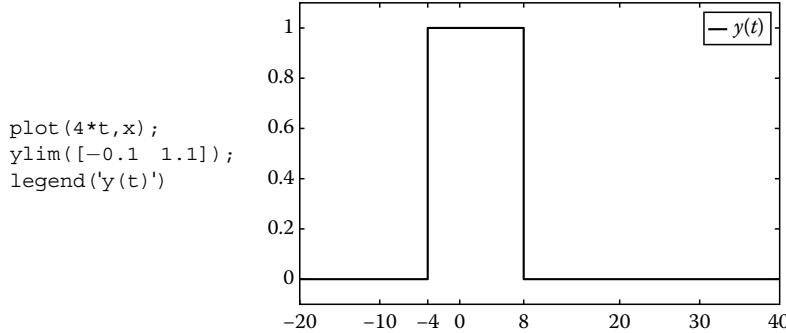
$$x(t) = \begin{cases} 0, & t < -1 \\ 1, & -1 \leq t \leq 2 \\ 0, & t > 2 \end{cases}$$

The graph is implemented in MATLAB according to the second definition of $x(t)$.

```
t1 = -5:.1:-1;
x1 = zeros(size(t1));
t2 = -1:.1:2;
x2 = ones(size(t2));
t3 = 2:.1:10;
x3 = zeros(size(t3));
t = [t1 t2 t3];
x = [x1 x2 x3];
plot(t,x);
ylim([-0.1 1.1]);
legend('x(t)')
```



The graph of the output signal $y(t)$ is derived by typing



The input signal is zero for $t < -1$ but the output signal is not zero for $t < -1$; thus it depends from future values of $x(t)$. Therefore the system is not causal.

Remark

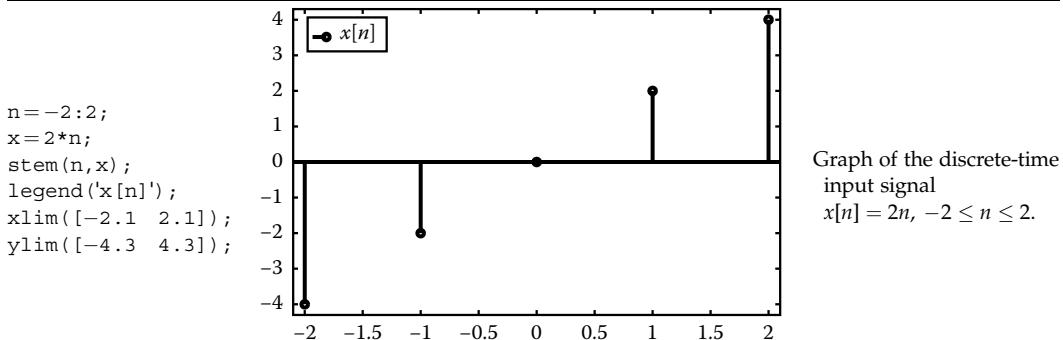
If the input signal $x(t) = u(t) - u(t - 2)$ was considered, we would conclude that the system is causal, a conclusion that is not correct. Hence, in order to prove that a property does not hold, giving an appropriate example is enough. On the other hand, a property is not proven by an example. Nevertheless, the authors have carefully selected the given examples for the practical approach adopted in this book to be consistent with theory.

Problem 4 Find out if the discrete-time system described by the i/o relationship $y[n] = x[-n]$ is

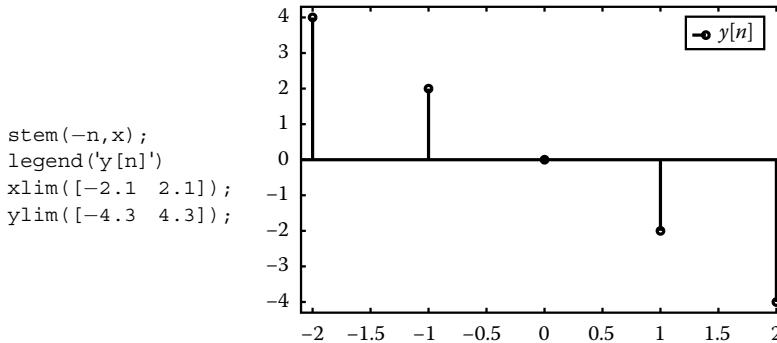
- Static or dynamic
- Causal or noncausal
- Linear or not linear
- Shift invariant or shift variant

Solution

- a. The input signal $x[n] = 2n$, $-2 \leq n \leq 2$ is considered.



(continued)



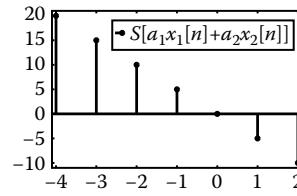
Graph of the response $y[n]$ of the system. The system is dynamic, as for example the value of $y[n]$ for $n = 2$ depends on the value of $x[n]$ for $n = -2$.

- b. The system is not causal as the output depends on future values of the input. For example the value of $y[n]$ for $n = -1$ depends on the value of $x[n]$ for $n = 1$.
- c. We consider the input signals $x_1[n] = 2n$, $-2 \leq n \leq 4$, $x_2[n] = n/3$, $-2 \leq n \leq 4$, and the scalars $a_1 = 2$, $a_2 = 3$. Recall that a discrete-time system is linear if $S\{a_1x_1[n] + a_2x_2[n]\} = a_1S\{x_1[n]\} + a_2S\{x_2[n]\}$.

```

n=-2:4;
x1=2*n;
x2=n/3;
a1=2; a2=3;
z=a1*x1+a2*x2;
n1=-fliplr(n);
y=fliplr(z);
stem(n1,y);
xlim([-4.1 2.1]);
ylim([-11 21]);
legend('S[a_1x_1[n]+a_2x_2[n]]')

```

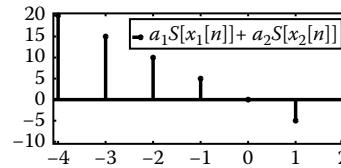


The left side of the mathematical expression for linearity is computed by using the command `fliplr`. Below, we compute and plot the right side of the linearity expression.

```

y1=fliplr(x1);
y2=fliplr(x2);
stem(n1,a1*y1+a2*y2);
legend('a_1S[x_1[n]]+a_2S[x_2[n]]')
xlim([-4.1 2.1]);
ylim([-11 21]);

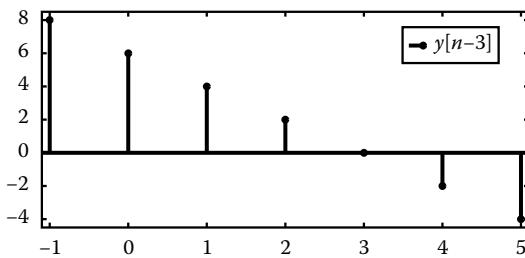
```



The two graphs are alike, hence, the system described by the equation $y[n] = x[-n]$ is linear.

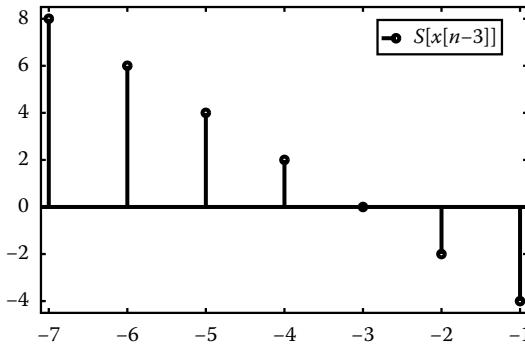
- d. A discrete-time system is shift invariant if $y[n - n_0] = S\{x[n - n_0]\}$. Let $x[n] = 2n$, $-2 \leq n \leq 4$ be the input signal and $n_0 = 3$ is the shift.

```
n=-2:4;
x=2*n;
y=fliplr(x);
ny=-fliplr(n);
stem(ny+3,y);
xlim([-1.1 5.1]);
ylim([-4.5 8.5]);
legend('y[n-3]');
```



First, we compute the left side of the relationship; that is, we obtain the signal $y[n - 3]$ by shifting $y[n]$ 3 units to the right.

```
stem(-(n+3),x)
xlim([-7.1 -.9]);
ylim([-4.5 8.5]);
legend('S[x[n-3]]');
```

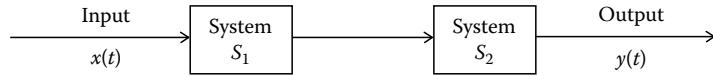


The response of the system to the input signal $x[n - 3]$ is computed by first shifting the signal $x[n]$ 3 units to the right and then reversing the derived graph. The two graphs are not the same; thus the system is not shift invariant.

3.4 Homework Problems

- Suppose that the response of a system S_1 to the input signal $x(t) = e^{-t}u(t - 1)$ is $y_1(t) = te^{-t}u(t)$. On the other hand, suppose that the response of a second system S_2 to the input signal $x(t)$ is $y_2(t) = te^{-t}u(t - 2)$. Find out if the two systems are causal.
- Find out if the system described by the i/o relationship $y(t) = \int_0^{\infty} x(t - \tau)e^{-3\tau}d\tau$ is causal.
- Find out if the discrete-time system with i/o relationship $y[n] = x[2n]$ is static or dynamic.
- Find out if the system with i/o relationship $y(t) = \cos(x(t))$ is static or dynamic.
- Determine if the system with i/o relationship $y(t) = \begin{cases} x(t), & t \geq 0 \\ 0 & t < 0 \end{cases}$ is homogeneous and additive.
- Determine if the system with i/o relationship $y[n] = n^2x[n]$ is linear.
- Determine whether the system with i/o relationship $y(t) = x(t) + x(t - 1)$ is linear.
- Determine whether the discrete-time system with i/o relationships $y[n] = x[n^2]$ and $y[n] = x^2[n]$ are linear.
- Suppose that the i/o relationships of the discrete-time systems S_1 , S_2 , and S_3 are $y_1[n] = 5x[n]$, $y_2[n] = nx[n]$, and $y_3[n] = x[5n]$. Determine if these three systems are time invariant.
- Determine if the system with i/o relationship $y(t) = 3x(t) + 2\cos(\pi t/3)$ is linear and time invariant.

11. Find out if the system described by the i/o relationship $y(t) = \int_{t-2}^{t+2} x(\tau)d\tau$ is a linear time-invariant system.
12. Find out if the systems with i/o relationships $y(t) = \cos(x(t))$, $y(t) = x(t)\cos(t)$, $y(t) = x(t)e^{-t}$, and $y(t) = e^{-x(t)}$ are invertible. If the answer is positive derive the inverse system.
13. Determine if the discrete-time system with i/o relationship $y[n] = x[n] + x[n - 1]$ is stable.
14. Determine if the discrete-time systems with i/o relationships $y[n] = x[n]/(n + 1)$ and $y[n] = x[n]/(n + 1)u[n]$ are stable.
15. Suppose that the output $y(t)$ of the system that is depicted in the figure below is equal to the applied input signal $x(t)$. Find the i/o relationship of the system S_2 if the i/o relationship of the system S_1 is $y_1(t) = \log_2(x(t))$.



This page intentionally left blank

4

Time Domain System Analysis

In this chapter, we introduce the concept of impulse response of a system and we discuss the possible interconnections between two or more systems. Moreover, the process of convolution, which is one of the most important concepts in signals and systems theory, is established for both continuous- and discrete-time systems. Furthermore, we present how a system can be described by a differential or a difference equation, and finally we discuss the finite impulse response and infinite impulse response filters.

4.1 Impulse Response

The meaning of impulse response of a system is easily derived if one considers the terms that it is named from. The term, “response,” denotes the output of a system, while the term, “impulse,” denotes that the input signal applied to the system is the unit impulse or Dirac delta function. Hence, the *impulse response* of a causal linear and time-invariant continuous-time system is its output when the Dirac delta function is the input applied to the system. This relationship is graphically illustrated in the block diagram representation shown in Figure 4.1. The impulse response is usually denoted by $h(t)$. Mathematically, the above definition is expressed as

$$h(t) = S\{\delta(t)\}. \quad (4.1)$$

4.2 Continuous-Time Convolution

The impulse response of a linear time-invariant system completely specifies the system. More specifically, if the impulse response of a system is known one can compute the system output for any input signal. We now present one of the most important topics in signals and systems theory.

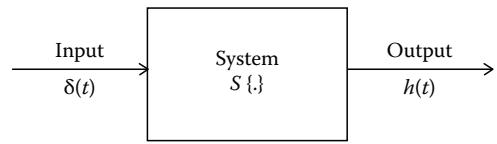
The response (or output) of a system to any input signal is computed by the *convolution* of the input signal with the impulse response of the system.

Suppose that $y(t)$ denotes the output of the system, $x(t)$ is the input signal, and $h(t)$ is the impulse response of the system. The mathematical expression of the convolution relationship is

$$y(t) = x(t) * h(t), \quad (4.2)$$

FIGURE 4.1

Block diagram representation of system S . The signal $x(t) = \delta(t)$ is the input signal applied to the system; that is, the Dirac delta function is the input signal, and the output of the system is $y(t) = h(t)$; that is, the output of the system is the system impulse response.



where the symbol $*$ denotes convolution, and it must not be confused with the multiplication symbol. The calculation of the convolution between two signals involves the computation of an integral. The complete form of (4.2) is

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t - \tau)d\tau. \quad (4.3)$$

By alternating variables τ and t , Equation 4.3 becomes

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(t - \tau)h(\tau)d\tau. \quad (4.4)$$

Equations 4.3 and 4.4 are equivalent and express the *convolution integral*.

4.2.1 Computation of Convolution

In order to calculate the convolution between two signals, a specific (and not exactly trivial) computational procedure has to be followed. The convolution computational procedure is introduced through an example.

Example

A linear time-invariant system is described by the impulse response

$$h(t) = \begin{cases} 1 - t, & 0 \leq t \leq 1 \\ 0, & \text{elsewhere} \end{cases}.$$

Calculate the response of the system to the input signal

$$x(t) = \begin{cases} 1, & 0 \leq t \leq 2 \\ 0, & \text{elsewhere} \end{cases}.$$

In order to compute the convolution between $x(t)$ and $h(t)$ the computational procedure is implemented in the following steps.

Step 1: The input and impulse response signals are plotted in the τ -axis; that is, t is replaced with τ , or in other words the signals

$$h(\tau) = \begin{cases} 1 - \tau, & 0 \leq \tau \leq 1 \\ 0, & \text{elsewhere} \end{cases} \quad \text{and} \quad x(\tau) = \begin{cases} 1 - \tau, & 0 \leq \tau \leq 1 \\ 0, & \text{elsewhere} \end{cases}$$

are defined and plotted.

Commands	Results	Comments
<pre> tx1=-2:.1:0; tx2=0:.1:2; tx3=2:.1:4; tx=[tx1 tx2 tx3]; x1=zeros(size(tx1)); x2=ones(size(tx2)); x3=zeros(size(tx3)); x=[x1 x2 x3]; </pre> <pre> th1=-2:.1:0; th2=0:.1:1; th3=1:.1:4; th=[th1 th2 th3]; h1=zeros(size(th1)); h2=1-th2; h3=zeros(size(th3)); h=[h1 h2 h3]; plot(tx,x,th,h,:*) ylim([-1 1.1]) legend('x(\tau)', 'h(\tau)') grid </pre>		The signals $x(\tau)$ and $h(\tau)$ are defined in the usual way of constructing two-part functions. The input signal $x(\tau)$ is plotted with solid line, while the impulse response signal $h(\tau)$ is plotted with dotted line and asterisks. Both signals are defined in the time interval $-2 \leq \tau \leq 4$. Notice that signals are plotted in the τ -axis and how Greek letters are inserted into the legend.

Step 2: The second step is known as *reflection*. One of the two signals is selected (in this example $h(\tau)$ is chosen, but $x(\tau)$ could have been also selected) and its symmetric with respect to the vertical axis, i.e., $h(-\tau)$ is plotted.

Commands	Results
<pre> plot(tx,x,-th,h,:*) legend('x(\tau)', 'h(-\tau)') ylim([-1 1.1]) </pre>	

Step 3: The third step is *shifting*. The signal $h(-\tau)$ is shifted by t ; that is, the signal $h(t - \tau)$ is plotted. Note that t is a constant, as the variable of the defined signals is variable τ .

Commands	Results	Comments
<pre>t = -2; plot(tx,x,-th+t,h,'.*') ylim([-1 1.1]) legend('x(\tau)', 'h(t-\tau)')</pre>		<p>The signal $h(t - \tau)$ is plotted for $t = -2$. Notice that $h(t - \tau)$ is shifted by 2 units to the left compared to $h(-\tau)$.</p>

A very useful (for the future computations) observation is the point in the τ -axis where the right end of $h(t - \tau)$ is. The right end of $h(t - \tau)$ for $t = -2$; that is, the right end of $h(-2 - \tau)$ is at the point $\tau = -2$. In the previous graph, we observe that the right end of $h(-\tau)$, that is, the right end of $h(0 - \tau)$ is at $\tau = 0$. Hence, we conclude that in the general case the right end of $h(t - \tau)$ is at the point $\tau = t$. On the other hand, the left end of $h(t - \tau)$ is at $\tau = t - T$, where T is the duration of $h(t - \tau)$. In this example $T = 1$.

Step 4: The fourth step is the *sliding* step. The value of the output signal $y(t)$ at time t , that is, the value of the convolution between the input signal and the impulse response signal at time t , depends on the overlap between $x(\tau)$ and $h(t - \tau)$ at time t . In order to compute the convolution for all t we have to slide the signal $h(t - \tau)$ from $-\infty$ toward $+\infty$ and to derive the kind (or stage) of overlap in reference to the value of t . Note that the signal $x(t)$ remains still.

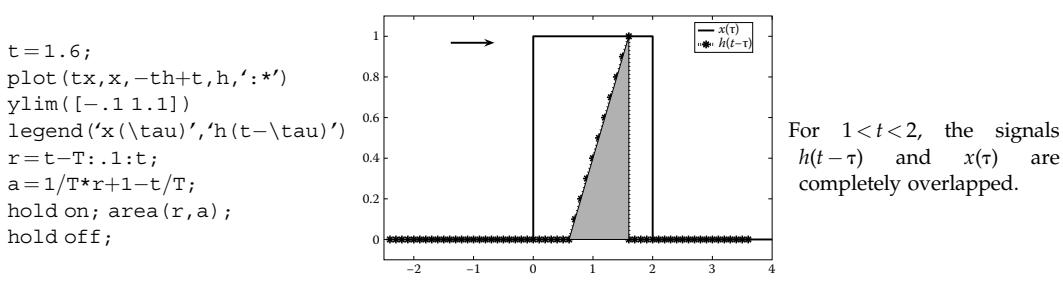
- First stage: Zero overlap

Commands	Results	Comments
<pre>t = -2; plot(tx,x,-th+t,h,'.*') ylim([-1 1.1]) legend('x(\tau)', 'h(t-\tau)')</pre>		<p>For $t < 0$ (in this figure $t = -2$) the signals $h(t - \tau)$ and $x(\tau)$ do not overlap. Thus, the output is $y(t) = 0$.</p>

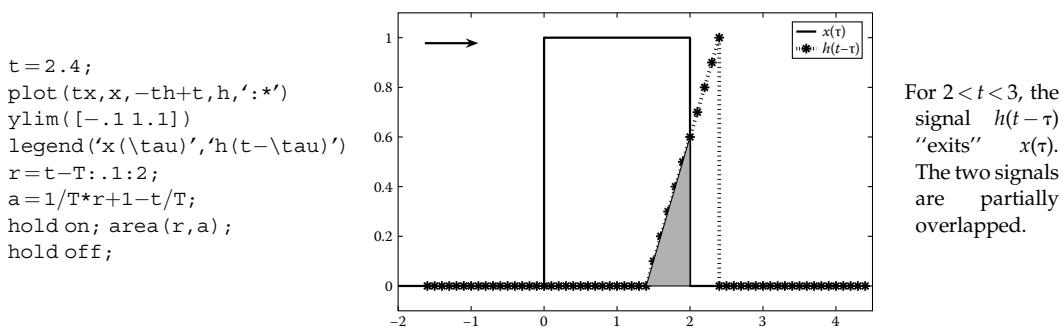
- Second stage: Partial overlap

Commands	Results	Comments
<pre>t = 0.5; plot(tx,x,-th+t,h,:'*') ylim([-1 1.1]) legend('x(\tau)', 'h(t-\tau)') %the following code %produces the shadowed %area plot. T=1; r=0:.1:t; a=1/T*r+1-t/T; hold on; area(r,a); hold off;</pre>		<p>For $0 < t < 1$, the signal $h(t - \tau)$ is "entering" at $x(\tau)$. The two signals are partially overlapped. The overlapped part is shaded.</p>

- Third stage: Complete overlap



- Fourth stage: Exit—partial overlap



- Fifth stage: Zero overlap

Commands	Results	Comments
<pre>t = 3.6; plot(tx,x,-th+t,h,:'*') ylim([-1 1.1]) legend('x(\tau)', 'h(t-\tau)') </pre> <div style="border: 1px solid black; padding: 2px; display: inline-block;"> ZERO OVERLAP STAGE </div>		For $t > 3$, the signal $h(t - \tau)$ does not overlap with $x(\tau)$. Hence, the output is $y(t) = 0$.

Step 5: Specification of the limits and calculation of the convolution integral. Having derived the time intervals for the various stages of overlap, the convolution integral is computed separately for each stage. Before computing the integrals the integration limits have to be specified. Recall that the right end of the signal $h(t - \tau)$ is at the point $\tau = t$, while the left end of $h(t - \tau)$ is at the point $\tau = t - T = t - 1$. Hence,

1. For $t < 0$, the two signals do not overlap (zero-overlap stage). Thus, the response of the system is $y(t) = 0$.
2. For $0 < t < 1$, the two signals start to overlap (entry stage—partial overlap). The limits of the integral are the limits that specify the shadowed area. Hence, $y(t) = \int_0^t x(\tau)h(t - \tau)d\tau$. The input signal $x(\tau)$ is given by $x(\tau) = 1$, while the impulse response signal $h(t - \tau)$ is given by $h(t - \tau) = 1 - (t - \tau) = 1 - t + \tau$. The expression of $h(t - \tau)$ is derived by substituting t with $t - \tau$ in $h(t)$. Thus, the integral that has to be calculated is $y(t) = \int_0^t 1(1 - t + \tau)d\tau = \int_0^t 1 - t + \tau d\tau$.

Commands	Results	Comments
<pre>syms t r f=1-t+r; y=int(f,r,0,t)</pre>	$y = t - 1/2 * t^2$	The integral is computed and the response of the system at the entry stage is $y(t) = t - t^2/2$, $0 < t < 1$.

3. For $1 < t < 2$, the two signals overlap completely (complete overlap stage). The only difference to the previous calculation is the integral limits. In this case, the output is given by $y(t) = \int_{t-1}^t 1 - t + \tau d\tau$.

Commands	Results	Comments
<code>y=int(f,r,t-1,t)</code> <code>simplify(y)</code>	$y = 1-t+1/2*t^2-1/2*(t-1)^2$ $ans=1/2$	The integral is computed and the result is simplified. The response of the system at the complete overlap stage is $y(t) = 0.5, 1 < t < 2$.

4. For $2 < t < 3$, the two signals overlap partially (exit stage). Thus, the output is given by $y(t) = \int_{t-1}^2 1 - t + \tau d\tau$.

Commands	Results	Comments
<code>y=int(f,r,t-1,2)</code>	$y = 5-t-t*(3-t)-1/2*(t-1)^2$	The response of the system at the exit stage is $y(t) = (3-t)^2/2, 2 < t < 3$.

5. For $t > 3$, there is no overlap; thus, the output is $y(t) = 0, t > 3$.

Combining all the derived results we, conclude that the response of a system with impulse response $h(t) = 1 - t, 0 \leq t \leq 1$ to the input signal $x(t) = 1, 0 \leq t \leq 2$ is

$$y(t) = \begin{cases} t - t^2/2, & 0 \leq t \leq 1 \\ 1/2, & 1 \leq t \leq 2 \\ (3 - t)^2/2, & 2 \leq t \leq 3 \\ 0, & t < 0 \text{ and } t > 3 \end{cases}.$$

Finally, the output $y(t)$ is plotted in MATLAB® according to the technique of plotting multipart functions.

Commands	Results
<pre>t1=0:.1:1; t2=1:.1:2; t3=2:.1:3; y1=t1-(t1.^2)/2; y2=0.5*ones(size(t2)); y3=0.5*((3-t3).^2); plot(t1,y1,t2,y2,'.',t3,y3,:') ylim([0 0.6]) title('Output signal y(t)');</pre>	<p style="text-align: right;">Output signal $y(t)$</p>

4.2.2 The Command conv

The computational process followed in the previous chapter is the analytical way of deriving the convolution between two signals. In MATLAB, the command `conv` allows the direct computation of the convolution between two signals. In order to illustrate the `conv` command we consider the previously used example, namely, an impulse response signal given by $h(t) = 1 - t$, $0 \leq t \leq 1$ and an input signal given by $x(t) = 1$, $0 \leq t \leq 2$. There are four rules that have to be applied for successfully computing the convolution between two continuous-time signals.

- *First rule:* The two signals (input and impulse response) should be defined in the same time interval. Hence, both signals are defined in the time interval $a \leq t \leq b$, where $t=a$ is the first time instance that at least one of the signals $x(t)$ or $h(t)$ is not zero and $t=b$ is the last time instance that at least one of the two signals is not zero. In our example the time interval is $0 \leq t \leq 2$. Thus the input signal is $x(t) = 1$, $0 \leq t \leq 2$ and the impulse response signal is expressed as

$$h(t) = \begin{cases} 1 - t, & 0 \leq t \leq 1 \\ 0, & 1 \leq t \leq 2 \end{cases}.$$

- *Second rule:* When a signal consists of multiple parts, the time intervals in which each part is defined must not overlap. Therefore the impulse response signal is defined as

$$h(t) = \begin{cases} 1 - t, & 0 \leq t \leq 1 \\ 0, & 1 < t \leq 2 \end{cases}.$$

Notice that the equality at $t=1$ is placed only at the upper part.

The MATLAB implementation of the definition of the signal is as follows.

Commands	Comments
<code>step = 0.01;</code>	The time step has to be quite small in order to approximate accurately the continuous-time signals.
<code>t = 0:step:2; x = ones(size(t));</code>	The input signal $x(t) = 1, 0 \leq t \leq 2$ is defined.
<code>t1 = 0:step:1; t2 = 1+step:step:2;</code>	The time intervals for the two parts of $h(t)$ are defined in such a way that they do not overlap. More specifically, vector $t2$ is defined from the time instance $1 + \text{step}$, i.e., is defined from the time instance 1.01. (second rule). Also notice that the time step used at the definition of the two signals must be the same.
<code>h1 = 1-t1; h2 = zeros(size(t2)); h = [h1 h2];</code>	The impulse response $h(t)$ is defined in the wider time interval, namely, in the time interval where the input $x(t)$ is defined (first rule).

Having defined the input and impulse response signals the response of the system can be computed by convoluting the two signals. The convolution is implemented by the command $y = \text{conv}(x, h)$. However, there is still one detail that needs to be addressed and is described at the next rule.

- *Third rule:* The output of the `conv` command has to be multiplied with the time step used in the definition of the input and impulse response signals, in order to correctly compute the output of the system. This rule emerges from the fact that the convolution integral is approximated by sum in MATLAB.

Commands	Comments
<code>y = conv(x, h) * step;</code>	The response $y(t)$ of the system is computed by convoluting the input signal $x(t)$ with the impulse response signal $h(t)$ and multiplying the result with the time step (third rule).

The response of the system is now computed and the only thing left to do is to plot it. However, the number of elements of the output vector y is not equal to the number of elements of the vectors x or h .

The precise relationship is $\text{length}(y) = \text{length}(x) + \text{length}(h) - 1$.

Commands	Results	Comments
<code>length(y)</code>	<code>ans = 401</code>	
<code>length(x)</code>	<code>ans = 201</code>	
<code>length(h)</code>	<code>ans = 201</code>	Indeed the relationship $\text{length}(y) = \text{length}(x) + \text{length}(h) - 1$ is true.

To overcome this, the fourth rule must be applied.

- *Fourth rule:* The output of the system is plotted in the double time interval of the one in which the input and impulse response signals are defined.

Commands	Results	Comments
<pre>ty = 0 : step : 4; plot(ty, y);</pre>		<p>The time interval in which the output signal y will be plotted is $0 \leq t \leq 4$, namely, it is double from the time interval where the vectors x and h are defined.</p> <p>The response of the system $y(t)$ computed from the convolution between the input $x(t)$ and the impulse response $h(t)$ is plotted.</p>

The output signal that was obtained through the MATLAB implementation (by using the command `conv`) is the same as the one derived with the analytical approach in Section 4.2.1. In this point the necessity for applying the four rules must be already clear. In case that the first or second rule was not followed it would not be possible to apply the fourth rule hardening the proper plotting of the output of the system. However, as we will discuss in Section 4.6 there is an alternative way, also applicable to continuous-time signals, to implement the convolution between two signals.

4.2.3 Deconvolution

Suppose that the impulse response of a system $h(t)$ and the output of a system $y(t)$ are available and we want to compute the input signal $x(t)$ that was applied to the system in order to generate the output $y(t)$. This process is called deconvolution and is implemented in MATLAB by using the command `deconv`. The syntax is `x = deconv(y, h)`, where x is the input vector, h is the impulse response vector, and y is the system response vector. The deconvolution process is also useful for determining the impulse response of a system if the input and output signals are known. In this case, the command is `h = deconv(y, x)`.

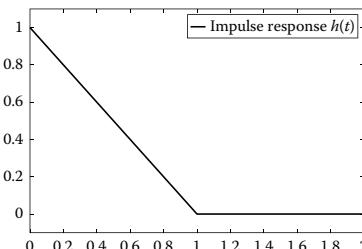
Remark

The commutativity property is not valid for the `deconv` command; hence, the output signal must be the first input argument of the command `deconv`.

Example

We will consider the same signals used in the previous example. Therefore, the problem is to compute the impulse response $h(t)$ of a system when the response of the system to the input $x(t) = 1, 0 \leq t \leq 2$ is the signal $y(t)$, which is depicted in the previous figure.

The signals $x(t)$ and $y(t)$ and the time t are already defined in the previous example, so we can directly use them to compute the impulse response $h(t)$.

Commands	Results	Comments
<pre>hh=deconv(y,x)*(1/step); plot(t,hh) ylim([-1 1.1]); legend('impulse response h(t)')</pre> 		The impulse response $h(t)$ is computed by multiplying the outcome of the command <code>deconv</code> by the quantity <code>(1/step)</code> as deconvolution is the inverse operation of convolution.

Indeed, the impulse response derived by the deconvolution process is the true one, i.e.,

$$hh(t) = h(t) = \begin{cases} 1 - t, & 0 \leq t \leq 1 \\ 0, & 1 < t \leq 2 \end{cases}.$$

4.2.4 Continuous-Time Convolution Examples

In this section, the convolution between various input and impulse response signals is illustrated through numerous examples.

Example

Suppose that a linear time-invariant (LTI) system is described by the impulse response $h(t) = e^{-t}u(t)$. Compute the response of the system to the input signal

$$x(t) = \begin{cases} 0.6, & -1 \leq t \leq 0.5 \\ 0.3, & 0.5 \leq t \leq 3 \\ 0, & t < -1 \text{ and } t > 3 \end{cases}.$$

First, the output is derived by the analytical way; that is, the convolution integrals are defined and computed for each stage. The graphic illustration of the convolution at each

stage is also given. Next, the result is confirmed by computing the output of the system with use of the `conv` command in two slightly different ways. Notice that the input signal consists of two nonzero parts. This fact complicates the computational process.

Step 1: The input and impulse response signals are defined and plotted in the τ -axis.

Commands	Results	Comments
<pre>th1=linspace(0,10,1001); h1=exp(-th1); h=[0 h1]; th=[0 th1]; tx=[-1 -1 0.5 0.5 3 3]; x=[0 0.6 0.6 0.3 0.3 0]; plot(tx,x,':',th,h) legend('x(\tau)','h(\tau)')</pre>		<p>Notice that a zero element is embedded into the impulse response vector and its corresponding time vector. This is done in order to plot the vertical line at $\tau = 0$.</p>

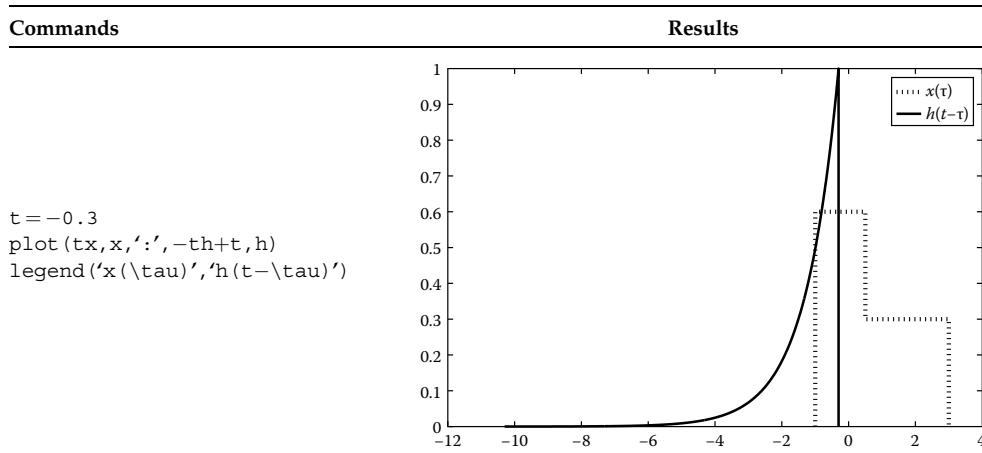
Steps 2–5: For brevity, Steps 2 through 5 of the computational process are combined; that is, we slide the signal $h(t - \tau)$ from $-\infty$ to $+\infty$ and compute the convolution integrals within the appropriate limits at various stages.

- First stage: Zero overlap.

Commands	Results
<pre>t=-3 plot(tx,x,:,-th+t,h) legend('x(\tau)','h(t-\tau)')</pre>	

For $t < -1$, the input and impulse response signals do not overlap; thus the output of the system is $y(t) = 0$.

- Second stage: Partial overlap of $h(t - \tau)$ with the first part of $x(\tau)$.



For $-1 < t < 0.5$, the impulse response signal $h(t - \tau)$ overlaps partially with the first part of $x(\tau)$, while there is no overlap with the second part of $x(\tau)$. The convolution integral in this stage is computed as

$$\begin{aligned} y(t) &= \int_{-1}^t x(\tau)h(t - \tau)d\tau = \int_{-1}^t 0.6e^{-(t-\tau)}d\tau \\ &= 0.6e^{-t} \int_{-1}^t e^\tau d\tau = 0.6 - 0.6e^{-t-1}. \end{aligned}$$

The result is verified in MATLAB.

Commands	Results/Comments
<pre>syms t r f = 0.6*exp(-(t-r)); y = int(f,r,-1,t)</pre>	$y = 3/5 - 3/5 \exp(-t-1)$ Notice that the value of $x(\tau)$ for $-1 < t < 0.5$ is 0.6.

- Third stage: The impulse response signal $h(t - \tau)$ overlaps completely with the first part of $x(\tau)$ and partially with the second part of $x(\tau)$.

Commands	Results
<pre>t=1.4; plot(tx,x,:,-th+t,h) legend('x(\tau)', 'h(t-\tau)')</pre>	

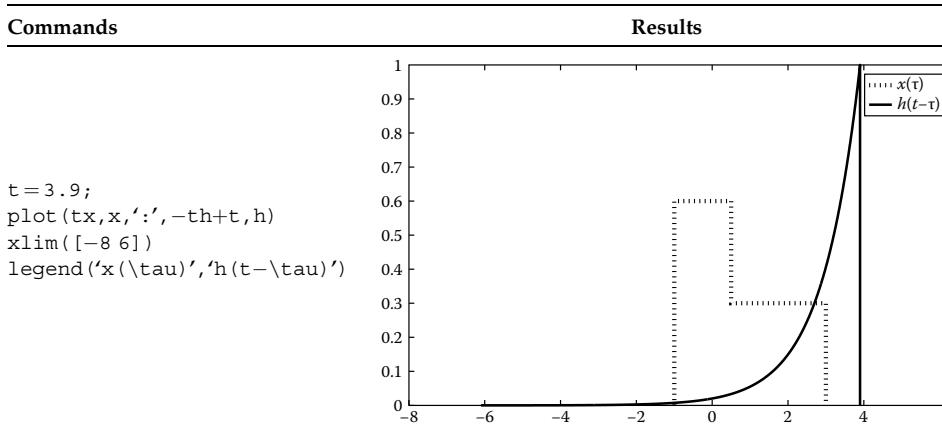
This stage takes place for $0.5 < t < 3$. There are two integrals that need to be calculated, corresponding to the different values of $x(\tau)$. Hence, the output signal is

$$\begin{aligned} y(t) &= \int_{-1}^{0.5} 0.6e^{-(t-\tau)} d\tau + \int_{0.5}^t 0.3e^{-(t-\tau)} d\tau = 0.6e^{-t} \int_{-1}^{0.5} e^\tau d\tau + 0.3e^{-t} \int_{0.5}^t e^\tau d\tau \\ &= 0.6e^{-t}(e^{0.5} - e^{-1}) + 0.3e^{-t}(e^t - e^{0.5}) = 0.3e^{-t+0.5} - 0.6e^{-t-1} + 0.3. \end{aligned}$$

In MATLAB, the above expression is computed as follows.

Commands	Results
<pre>syms t r f1=0.6*exp(-(t-r)); f2=0.3*exp(-(t-r)); y=int(f1,r,-1,0.5)+int(f2,r,0.5,t)</pre>	$y = \frac{3}{10}e^{-t+1/2} - \frac{3}{5}e^{-t-1} + \frac{3}{10}$

- Fourth stage: Complete overlap of $h(t - \tau)$ with both parts of $x(\tau)$.



The fourth stage takes place for $t > 3$. The convolution integral is calculated as

$$\begin{aligned} y(t) &= \int_{-1}^{0.5} 0.6e^{-(t-\tau)} d\tau + \int_{0.5}^3 0.3e^{-(t-\tau)} d\tau = 0.6e^{-t} \int_{-1}^{0.5} e^\tau d\tau + 0.3e^{-t} \int_{0.5}^3 e^\tau d\tau \\ &= 0.6e^{-t}(e^{0.5} - e^{-1}) + 0.3e^{-t}(e^3 - e^{0.5}) = 0.3e^{-t+0.5} - 0.6e^{-t-1} + 0.3e^{-t+3}. \end{aligned}$$

In MATLAB, the above expression is computed as follows.

Commands	Results
<pre>syms t r f1 = 0.6*exp(-(t-r)); f2 = 0.3*exp(-(t-r)); y=int(f1,r,-1,0.5)+int(f2,r,0.5,3)</pre>	$y = 3/10*exp(-t+1/2)-3/5*exp(-t-1) +3/10*exp(-t+3)$

An additional stage could be expected, the one that $h(t - \tau)$ exits from $x(\tau)$. However, looking into the definition of

$$h(t) = e^{-t}u(t) = \begin{cases} e^{-t}, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

we see that $h(t)$ (theoretically) is nonzero for any $t \in [0, +\infty)$. Therefore, $h(-\tau)$ is nonzero for any $\tau \in (-\infty, 0]$, and $h(t - \tau)$ is nonzero for any $\tau \in (-\infty, t]$. Thus, for $t > 3$, the signals $h(t - \tau)$ and $x(\tau)$ completely overlap. Consequently, there is no exit stage in the convolution computational process of this example. Assembling the expressions calculated for the output at the various stages of convolution, the response of the system to the input signal $x(t)$ is given by

$$y(t) = \begin{cases} 0, & t < -1 \\ 0.6 - 0.6e^{-t-1}, & -1 \leq t \leq 0.5 \\ 0.3e^{-t+0.5} - 0.6e^{-t-1} + 0.3, & 0.5 \leq t \leq 3 \\ 0.3e^{-t+0.5} - 0.6e^{-t-1} + 0.3e^{-t+3}, & t > 3 \end{cases}.$$

The system response is plotted using the technique of plotting piecewise functions.

Commands	Results
<pre>t1=-1:.1:0.5; y1=0.6*(1-exp(-1-t1)); t2=0.5:.1:3; y2=0.3*exp(-t2+0.5)-0.6*exp(-t2-1)+0.3 t3=3:.1:10; y3=0.3*exp(-t3)*(exp(0.5)-2*exp(-1)+exp(3)); t=[t1 t2 t3]; y=[y1 y2 y3]; plot(t,y) title('Output signal y(t)')</pre>	

Next, two slightly different approaches implemented in MATLAB in order to calculate again the convolution between the input signal

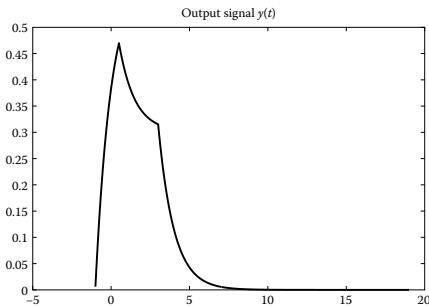
$$x(t) = \begin{cases} 0.6, & -1 \leq t < 0.5 \\ 0.3, & 0.5 \leq t \leq 3 \\ 0, & t < -1 \text{ and } t > 3 \end{cases}$$

and the impulse response signal $h(t) = e^{-t}u(t)$ are illustrated. In order to implement the computation in MATLAB an assumption must be made. As mentioned earlier, the signal $h(t) = e^{-t}u(t)$ is nonzero for $t \geq 0$. Therefore, in MATLAB it should be defined over the time interval $[0, +\infty)$. Of course, this is not possible. From the graph of $h(t)$, we notice that for $t > 10$ its value approaches zero. Thus, considering that $h(t) = 0$, $t > 10$ makes no difference from a practical viewpoint. So from now on, any function of the form $f(t) = e^{-at}u(t)$, $a \geq 1$ will be defined as $f(t) = e^{-at}(u(t) - u(t - 10)) = e^{-at}$, $0 \leq t \leq 10$.

First approach: The input signal $x(t)$ is shifted by one unit to the right in order to have nonzero value from the time instance $t = 0$. In other words, the input signal is now given by

$$x(t) = \begin{cases} 0.6, & 0 \leq t < 1.5 \\ 0.3, & 1.5 \leq t \leq 4 \\ 0, & t > 4 \end{cases}$$

This time shift does not cause any problem as the system is time invariant. After the convolution process is completed, the computed output has to be shifted back by one unit to the left. The shift-back operation is implemented when we plot the output signal.

Commands	Results	Comments
<pre>t1 = [0:0.01:1.5]; t2 = [1.5+0.01:0.01:4] t3 = [4.01:0.01:10]; x1 = 0.6*ones(size(t1)); x2 = 0.3*ones(size(t2)); x3 = zeros(size(t3)); x = [x1 x2 x3]; h = exp(-[t1 t2 t3]);</pre> <pre>y = conv(x, h)*0.01;</pre> <pre>plot(-1:0.01:19, y) title('Output signal y(t)')</pre>		<p>The first rule is applied; that is, the input and impulse response signals are defined over the same time interval ($0 \leq t \leq 10$), while the partial time intervals are constructed in a way that they do not overlap (second rule).</p> <p>The output of the system is computed by multiplying the result of the convolution with the time step (third rule).</p>
		<p>The output is plotted in the double time interval (fourth rule). Notice that the output signal is shifted by one unit to the left since the input signal was shifted one unit to the right.</p>

The response of the system obtained is the same as the one derived with the analytical approach.

Second approach: The input and impulse response are defined in the same (the wider) time interval, i.e., in the time interval $-1 \leq t \leq 10$. Thus the two signals are now given by

$$x(t) = \begin{cases} 0.6, & -1 \leq t \leq 0.5 \\ 0.3, & 0.5 < t \leq 3 \\ 0, & 3 < t \leq 10 \end{cases} \quad \text{and} \quad h(t) = \begin{cases} 0, & -1 \leq t < 0 \\ e^{-t}, & 0 \leq t \leq 10 \end{cases}.$$

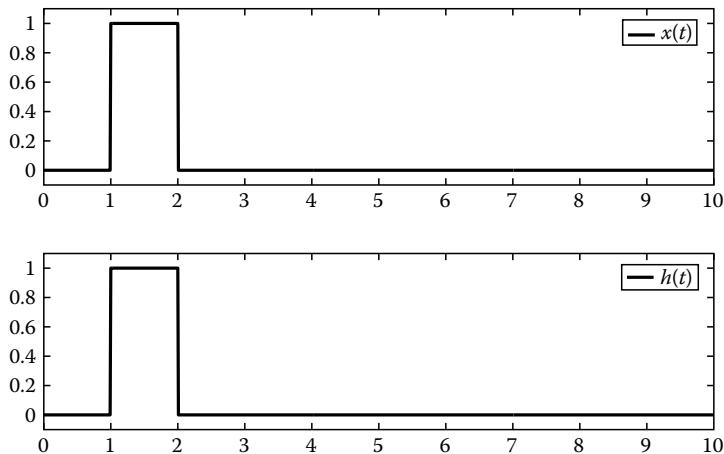
The difference in this approach is at the time interval in which the output signal is plotted. The time interval must be doubled for both negative and positive values of t .

Commands	Results	Comments
<pre>t1 = [-1:0.01:0.5]; x1 = .6*ones(size(t1)); t2 = [.5+0.01:0.01:3]; x2 = .3*ones(size(t2)); t3 = [3.01:0.01:10]; x3 = zeros(size(t3)); x = [x1 x2 x3]; t1 = -1:-.01:-.01; t2 = 0:.01:10; h1 = zeros(size(t1)); h2 = exp(-t2); h = [h1 h2]; y = conv(x,h)*.01; plot(-2:.01:20,y)</pre>		<p>The output $y(t)$ is plotted over the time interval $-2 \leq t \leq 20$ as the input and impulse response signals are defined in the time interval $-1 \leq t \leq 10$. Thus, when the input or the impulse response signals are nonzero for $t < 0$ the time interval where the output signal is plotted must be doubled for positive and negative values of t.</p>

The computed response of the system is again the same as the one derived by the analytical approach.

Example

Compute and plot the convolution between the signals that appear in the figure below.



The mathematical expression of the two signals is $x(t) = h(t) = 1$, $1 \leq t \leq 2$, or more formally the two signals are given by

$$x(t) = h(t) = \begin{cases} 0, & 0 < t < 1 \\ 1, & 1 \leq t \leq 2 \\ 0, & 2 < t \leq 10 \end{cases}$$

Therefore,

Commands	Results	Comments
<pre>t1=0:.01:1-0.01; t2=1:.01:2; t3=2.01:.01:10; t=[t1 t2 t3]; x1=zeros(size(t1)); x2=ones(size(t2)); x3=zeros(size(t3)); x=[x1 x2 x3]; h=x; y=conv(x,h)*.01; plot(0:.01:20,y); axis([0 6 -.1 1.1]) legend('y(t)')</pre>		Graph of the output signal.

In order to make clear why the output takes nonzero values from $t = 2$ and afterward, the signals $x(\tau)$ and $h(t - \tau)$ are plotted for $t = 0$ and $t = 2$.

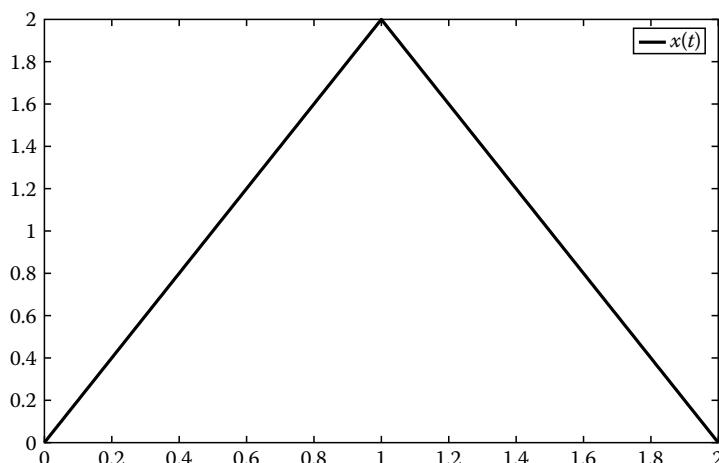
Commands	Results	Comments
<pre>plot(t,x,0-t,h,:') legend('x(t)', 'h(t-\tau)=h(0-\tau)') ylim([-1 1.1])</pre>		For $t = 0$ there is no overlap; thus the response of the system is zero.
<pre>plot(t,x,2-t,h,:') ylim([-1 1.1]) legend('x(t)', 'h(t-\tau)=h(2-\tau)')</pre>		For $t = 2$ the signal $h(t - \tau)$ is starting to enter $x(\tau)$. Thus, for $t > 2$ there is overlap between $x(\tau)$ and $h(t - \tau)$, and the response of the system is not zero.

Finally, we mention that since the time in which the overlap starts (and the output is nonzero) is known, the definition of the zero parts of the two signals could have been avoided. This method will be discussed further in the discrete-time convolution section. As expected, the outcome provided by this second approach is the same.

Commands	Results	Comments
<pre>t=1:.01:2; x=ones(size(t)); h=ones(size(t)); y=conv(x,h)*.01; plot(2:.01:4,y) legend('y(t)')</pre>		Only the nonzero parts of $x(t)$ and $h(t)$ are considered in the computation. The plot of the output $y(t)$ starts at $t = 2$.

Example

Suppose that a system is described by the impulse response $h(t) = \cos(2\pi t)(u(t) - u(t - 4))$. Compute and plot the response of the system to the input signal that is shown in the figure below.



First, the mathematical expression of the input signal $x(t)$ must be derived. The input signal is a two-part function, where each part is a straight line. So, in order to determine the mathematical expression of the signal the straight line equation $x(t) = at + b$ is employed. To find a and b it is enough to examine the values of $x(t)$ in two points. The straight line equation for the first part is $x_1(t) = a_1t + b_1$. At the axis origin (i.e., at $t = 0$), we get $0 = a_10 + b_1 \Rightarrow b_1 = 0$. At the end of the first part (i.e., at $t = 1$), we get $2 = a_11 + 0 \Rightarrow a_1 = 2$. Hence, the mathematical expression of the first part is $x_1(t) = 2t, 0 \leq t \leq 1$. For the second part the straight line equation is $x_2(t) = a_2t + b_2$. At $t = 1$, we get $2 = a_21 + b_2 \Rightarrow b_2 = 2 - a_2$; while at $t = 2$, we get $0 = a_22 + b_2 \Rightarrow a_2 = b_2/2$. Thus, $a_2 = -2$, $b_2 = 4$, and the mathematical expression of the second part is $x_2(t) = 4 - 2t, 1 \leq t \leq 2$. Therefore, the input signal is given by

$$x(t) = \begin{cases} 2t, & 0 \leq t \leq 1 \\ 4 - 2t, & 1 \leq t \leq 2 \\ 0, & 2 < t \leq 4 \end{cases}$$

The mathematical expression of the impulse response is $h(t) = \cos(2\pi t)(u(t) - u(t-4))$, which is simplified into $h(t) = \cos(2\pi t)$, $0 \leq t \leq 4$. In order to apply the first rule (i.e., to define both signals at the proper time interval) the input signal is written as

$$x(t) = \begin{cases} 2t, & 0 \leq t \leq 1 \\ 4 - 2t, & 1 < t \leq 2 \\ 0, & 2 < t \leq 4 \end{cases}$$

Next, the procedure is followed as usual.

Commands	Results	Comments
<pre>t1=0:.01:1; t2=1.01:.01:2; t3=2.01:.01:4; x1=2*t1; x2=4-2*t2; x3=zeros(size(t3)); x=[x1 x2 x3]; t=[t1 t2 t3]; h=cos(2*pi*t); y=conv(x,h)*.01; plot(0:.01:8,y)</pre>		The system response $y(t)$.

4.3 Convolution Properties

In this section, we introduce the main properties of convolution through illustrative examples.

- Commutative property

For two signals $h_1(t)$ and $h_2(t)$ the commutative property stands; that is,

$$h_1(t) * h_2(t) = h_2(t) * h_1(t). \quad (4.5)$$

Example

Verify the commutative property of the convolution supposing that $h_1(t) = 1$, $0 \leq t \leq 5$ and $h_2(t) = 2e^{-2t}$, $0 \leq t \leq 5$.

The left side of (4.5), i.e., the signal $y(t) = h_1(t) * h_2(t)$ is computed and plotted first while the right side of (4.5), namely, $z(t) = h_2(t) * h_1(t)$ is computed and plotted at the second row.

Commands	Results	Comments
<pre>t = 0 : .01 : 5; h1 = ones(size(t)); h2 = 2*exp(-2*t); y = conv(h1,h2)*0.01; plot(0:.01:10,y); title('h_1(t)*h_2(t)')</pre>		The left side of (4.5).
<pre>z = conv(h2,h1)*0.01; plot(0:.01:10,z); title('h_2(t)*h_1(t)')</pre>		The right side of (4.5).

The two signals ($y(t)$ and $z(t)$) are identical; thus the commutative property is clearly verified.

- Associative property

For three signals $h_1(t)$, $h_2(t)$, and $x(t)$ the associative property stands; that is,

$$h_2(t) * [h_1(t) * x(t)] = [h_2(t) * h_1(t)] * x(t). \quad (4.6)$$

Example

Verify the associative property of the convolution supposing that $h_1(t) = (1/\pi)t$, $0 \leq t \leq 5$; $h_2(t) = 2e^{-2t}$, $0 \leq t \leq 5$; and $x(t) = u(t) - u(t - 5)$.

For the left side of (4.6), which is $y(t) = h_2(t) * [h_1(t) * x(t)]$, first the convolution $y_1(t) = h_1(t) * x(t)$ is computed. Next, the signal $h_2(t)$ is defined in the same time interval with $y_1(t)$ (at $0 \leq t \leq 10$) and the result of their convolution $y(t)$ is plotted.

Commands	Results	Comments
<pre>t = 0:.01:5; x=ones(size(t)); h1=1/pi*t; y1=conv(h1,x)*.01; h2=2*exp(-2*t); th2=5.01:.01:10; hh2=zeros(size(th2)); h2=[h2 hh2]; y=conv(h2,y1)*0.01; plot(0:.01:20,y); title('h_2(t)*(h_1(t)*x(t))')</pre>	<p>The graph shows a bell-shaped curve starting near zero at t=0, reaching a maximum value of approximately 3.7 at t=6, and returning to zero at t=10. The x-axis ranges from 0 to 20 with major ticks every 2 units. The y-axis ranges from 0 to 4 with major ticks every 0.5 units. The title of the plot is $h_2(t) * (h_1(t) * x(t))$.</p>	The left side of (4.6).

For the right side of (4.6), which is $z(t) = [h_2(t) * h_1(t)] * x(t) = [h_1(t) * h_2(t)] * x(t)$, first the convolution $z_1(t) = h_1(t) * h_2(t)$ is computed. Next, the signal $x(t)$ is defined in the same time interval with $z_1(t)$ (at $0 \leq t \leq 10$) and the result of their convolution $z(t)$ is plotted.

Commands	Results	Comments
<pre>t = 0:.01:5; h1=1/pi*t; h2=2*exp(-2*t); z1=conv(h1,h2)*0.01; x=ones(size(t)); tx=5.01:.01:10; xx=zeros(size(tx)); x=[x xx]; z=conv(z1,x)*0.01; plot(0:.01:20,z); title('(h_2(t)*h_1(t))*x(t)')</pre>	<p>The graph shows a bell-shaped curve starting near zero at t=0, reaching a maximum value of approximately 3.7 at t=6, and returning to zero at t=10. The x-axis ranges from 0 to 20 with major ticks every 2 units. The y-axis ranges from 0 to 4 with major ticks every 0.5 units. The title of the plot is $(h_2(t)*h_1(t))*x(t)$.</p>	The right side of (4.6).

The two graphs are identical; thus the associative property is clearly verified.

- Distributive property

For three signals $h_1(t)$, $h_2(t)$, and $x(t)$ the distributive property stands; that is,

$$[h_1(t) + h_2(t)] * x(t) = h_1(t) * x(t) + h_2(t) * x(t). \quad (4.7)$$

Example

Illustrate the distributive property of the convolution by using the signals $h_1(t) = \cos(\pi t)$, $0 \leq t \leq 5$; $h_2(t) = 2e^{-2t}$, $0 \leq t \leq 5$; and $x(t) = u(t) - u(t-5)$.

In a similar vein to the two previous examples, the left side of (4.7), that is, $y(t) = [h_1(t) + h_2(t)] * x(t)$ is compared to the right side of (4.7), that is, $z(t) = h_1(t) * x(t) + h_2(t) * x(t)$.

Commands	Results	Comments
<pre>t = 0:.01:5; x=ones(size(t)); h1=cos(pi*t); h2=2*exp(-2*t); h=h1+h2; y=conv(h,x)*0.01; plot(0:.01:10,y); title('[h_1(t)+h_2(t)]*x(t)')</pre>		The left side of (4.7).
<pre>t = 0:.01:5; x=ones(size(t)); h1=cos(pi*t); h2=2*exp(-2*t); z1=conv(h1,x)*0.01; z2=conv(h2,x)*0.01; z=z1+z2; plot(0:.01:10,z); title('h_1(t)*x(t)+h_2(t)*x(t)')</pre>		The right side of (4.7).

The two graphs are the same; thus the distributive property of convolution is clearly illustrated.

- Identity property

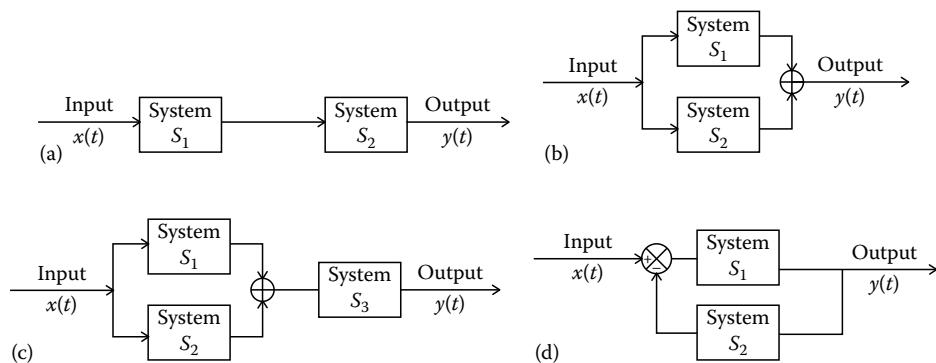
If $\delta(t)$ is the Dirac delta function, then for any signal $h(t)$ the following expression is true:

$$h(t) * \delta(t) = h(t). \quad (4.8)$$

This property is straightforwardly proven from the definition of the Dirac function. Nevertheless, an illustrative example is provided in the section of the discrete-time convolution.

4.4 Interconnections of Systems

Systems may be interconnections of other (sub)-systems. The basic interconnections are the cascade, the parallel, the mixed, and the feedback. The block diagrams of these four types are illustrated in Figure 4.2.

**FIGURE 4.2**

Basic interconnections of (sub) systems: (a) cascade, (b) parallel, (c) mixed, and (d) feedback.

- When two systems S_1 and S_2 are cascade (or serially) connected (Figure 4.2a), the output of the first system is the input of the second system. The block diagram of two parallel interconnected systems is presented in Figure 4.2b. The same input signal is applied to the two parallel-connected systems, and the outputs of S_1 and S_2 are combined (in this block diagram added) to generate the overall output. The mixed interconnection is a combination of cascade and parallel interconnections. In the block diagram of Figure 4.2c, systems S_1 and S_2 are parallel connected and their output is input to the cascade-connected system S_3 . Finally, in Figure 4.2d the feedback interconnection block diagram is depicted. The output of S_1 is input to S_2 , while the output of S_2 is fed back to S_1 and combined with the input signal produce the overall output of the system.

The interconnected subsystems can be considered as one system, i.e., an equivalent system described by one overall impulse response. In order to compute the output and the impulse response of the equivalent overall system for the various types of interconnections, suppose that the subsystem S_1 is described by the impulse response $h_1(t) = te^{-3t}[u(t) - u(t - 3)]$ and the subsystem S_2 by the impulse response $h_2(t) = t\cos(2\pi t)[u(t) - u(t - 3)]$. Finally, let $x(t) = u(t) - u(t - 3)$ be the input signal.

- Cascade interconnection

The output of S_1 is input to S_2 . Thus, the output of the equivalent system is computed as $y(t) = [x(t) * h_1(t)] * h_2(t)$; that is, the input signal is first convoluted with the impulse response of S_1 and the computed output is convoluted with the impulse response of S_2 .

Commands	Results	Comments
<pre>t = 0:0.01:3; x = ones(size(t)); h1 = t.*exp(-3*t); y1=conv(x,h1)*0.01; t1 = 0:0.01:3; h2a=t1.*cos(2*pi*t1); t2 = 3.01:.01:6; h2b=zeros(size(t2)); h2 = [h2a h2b]; y=conv(y1,h2)*0.01; plot(0:.01:12,y) legend('y(t)')</pre>		Graph of the output $y(t) = [x(t) * h_1(t)] * h_2(t)$.

To compute the impulse response of the overall system, the associative property of the convolution is applied. More specifically, applying the associative property to the output relationship $y(t) = [x(t) * h_1(t)] * h_2(t)$ yields $y(t) = x(t) * [h_1(t) * h_2(t)]$. Consequently, the impulse response of the overall equivalent system, if the subsystems are cascade connected, is given by $h(t) = h_1(t) * h_2(t)$. Straightforwardly, the system response to $x(t)$ is given by $y(t) = x(t) * h(t)$.

Commands	Results	Comments
<pre>t = 0:0.01:3; h1=t.*exp(-3*t); h2=t.*cos(2*pi*t); h=conv(h1,h2)*0.01; plot(0:.01:6,h) legend('h(t)')</pre>		The impulse response $h(t)$ is obtained by $h(t) = h_1(t) * h_2(t)$.
<pre>t1 = 0:0.01:3; t2 = 3.01:.01:6; x1=ones(size(t1)); x2=zeros(size(t2)); x = [x1 x2]; y=conv(x,h)*0.01; plot(0:.01:12,y) legend('y(t)')</pre>		The output of the system $y(t)$ is computed from the convolution between the input signal $x(t)$ and the overall impulse response $h(t)$, which was derived using the associative property.

The two graphs are identical; hence, the computation of the impulse response and the output signal of the equivalent system are correct.

- Parallel interconnection

In this type of interconnection, the same input signal is applied to both subsystems. The two outputs of the subsystems are added to obtain the final output. The mathematical expression is $y(t) = h_1(t) * x(t) + h_2(t) * x(t)$.

Commands	Results	Comments
<pre>t = 0:0.01:3; h1 = t.*exp(-3*t); h2 = t.*cos(2*pi*t); x = ones(size(t)); y1 = conv(h1,x).*0.01 y2 = conv(h2,x).*0.01; y = y1+y2; plot(0:.01:6,y) legend('y(t)')</pre>		<p>The response of the system to the input signal $x(t)$ is computed by adding the outcome of the convolutions between the input signal and the impulse responses of the subsystems; that is, it is computed as $y(t) = h_1(t) * x(t) + h_2(t) * x(t)$.</p>

To compute the impulse response of the overall system the distributive property of the convolution is applied. More specifically, applying the distributive property to the output relationship $y(t) = h_1(t) * x(t) + h_2(t) * x(t)$ yields $y(t) = [h_1(t) + h_2(t)] * x(t)$. Consequently, the impulse response of the equivalent system when the subsystems are parallel connected is given by $h(t) = h_1(t) + h_2(t)$. Straightforwardly, the output of the system is given by $y(t) = x(t) * h(t)$. To verify our conclusion, we consider the same signals used in the cascade interconnection case, namely, $h_1(t) = te^{-3t}[u(t) - u(t - 3)]$, $h_2(t) = t \cos(2\pi t)[u(t) - u(t - 3)]$, and $x(t) = u(t) - u(t - 3)$.

Commands	Results	Comments
<pre>t = 0:0.01:3; h1 = t.*exp(-3*t); h2 = t.*cos(2*pi*t); h = h1+h2; plot(0:.01:3,h) legend('h(t)')</pre>		<p>The overall impulse response of the system is the sum of the impulse responses of the subsystems; that is, it is computed by $h(t) = h_1(t) + h_2(t)$.</p>

(continued)

Commands	Results	Comments
<pre>x=ones(size(t)); y=conv(x,h)*0.01; plot(0:.01:6,y) legend('y(t)')</pre>		<p>The output of the system $y(t)$ is computed from the convolution between the input signal $x(t)$ and the impulse response $h(t)$, which was derived using the distributive property.</p>

The graphs of the system response are identical; hence, our computation of the impulse response and the output of the equivalent system is accurate. The implementation of a mixed interconnection is left (see Sections 4.12 and 4.13) as an exercise to the reader. The feedback interconnection will be discussed in Chapter 11.

4.5 Stability

The concept of stability was introduced in Chapter 3. A system is bounded-input bounded-output (BIBO) stable if for any bounded applied input, the response of the system is also bounded. The knowledge of the impulse response of an LTI system allows us to specify a new criterion about the stability of a system.

An LTI system is BIBO stable if its impulse response is absolutely integrable on $(-\infty, +\infty)$.

The mathematical expression is

$$\int_{-\infty}^{\infty} |h(t)| dt < \infty. \quad (4.9)$$

Example

A system is described by the impulse response $h(t) = e^{-t^2}$. Tell if this is a BIBO stable system and verify your conclusion.

A system is BIBO stable if the condition given in (4.9) is satisfied; that is, we have to examine if its impulse response is absolutely integrable.

Commands	Results	Comments
<pre>syms t h=exp(-t.^2); int(abs(h),t,-inf,inf)</pre>	ans = pi^(1/2)	Condition (4.9) is fulfilled; hence, the system under consideration is BIBO stable.

In order to verify that this is a BIBO stable system, the bounded input signal $x(t) = u(t+10) - u(t-10)$ is applied to the system. The response of the system is expected to be also bounded.

Commands	Results	Comments
<pre>t=-10:.1:10; x1=ones(size(t)); h=exp(-t.^2); y1=conv(x1,h)*0.1; plot(-20:.1:20,y1) title('System response to u(t+10)-u(t-10)')</pre>		The system response $y(t)$ is computed from the convolution of $x(t)$ with $h(t)$.

Indeed, the response of the system is bounded ($|y(t)| < M = 2$); thus the BIBO stability of the system is verified.

Example

A system is described by the impulse response $h(t) = t^2$. Tell if this is a BIBO stable system.

Again the condition (4.9) is examined.

Commands	Results	Comments
<pre>syms t h=t^2; int(abs(h),t,-inf,inf)</pre>	ans = Inf	Condition (4.9) is not fulfilled; hence, the system is <i>not</i> BIBO stable.

4.6 Discrete-Time Convolution

An LTI discrete-time system is (equivalently to the continuous-time case) completely described by its impulse response, which is usually denoted by $h[n]$. The impulse response of an LTI discrete-time system is the output of the system when the unit impulse sequence (or Kronecker delta function) is applied as input. Even though the unit impulse input consists of one nonzero term, the impulse response signal $h[n]$ usually consists of more than one nonzero elements. The explanation is that the system is dynamic (with memory); that is, the system responds over various time instances to the input applied at $t=0$. The knowledge of the impulse response $h[n]$ of a system allows the computation of the response $y[n]$ of the system to any input signal $x[n]$. The output signal is computed by the discrete-time convolution. The mathematical expression of discrete-time convolution is

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] = \sum_{k=-\infty}^{+\infty} h[k]x[n-k]. \quad (4.10)$$

Expression (4.10) is the discrete-time counterpart of (4.3) and is known as the *convolution sum*. The convolution between two discrete-time signals is computed (according to the same rules implied for the continuous-time convolution) by using the MATLAB command conv.

4.6.1 The Unit Impulse Sequence as Input to a System

In this section, the impulse response of a discrete-time system is discussed in detail. More specifically, it is established that if the unit impulse sequence $\delta[n]$ is the input to a system the output of the system is the impulse response $h[n]$ of the system. This is the identity property of the convolution that was stated in Section 4.3.

Example

Suppose that a discrete-time system is described by the impulse response $h[n] = [2 \ 4 \ 1 \ 3]$, $0 \leq n \leq 3$. Compute the response of the system to the input signal

$$x[n] = \delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}.$$

The response of the system is computed by convoluting the impulse response $h[n]$ with the input signal $\delta[n]$. Recall that $\delta[n]$ must be defined in the same time interval to $h[n]$; that is $0 \leq n \leq 3$. For illustration purposes we plot the two signals.

Commands	Results	Comments
<pre>x = [1 0 0 0]; n = 0:3; stem(n,x) axis([-2 3 .2 -.1 1 1]) legend('x[n] = \delta[n]')</pre>		Input signal $x[n] = \delta[n]$ plotted in $0 \leq n \leq 3$.
<pre>h = [2 4 3 1]; n = 0:3; stem(n,h) axis([-2 3 .2 -.1 4 .1]) legend('h[n]')</pre>		Impulse response $h[n] = [2, 4, 1, 3]$, $0 \leq n \leq 3$.

The discrete-time convolution is computed in a similar process to the continuous-time convolution with use of the command `conv`. The difference is that the output of the command `conv` does not have to be multiplied by the time step since the time step is 1. The similarity is at the number of elements of the output vector $y[n]$ compared to the number of elements of the input and impulse response vectors. The exact relationship is again $\text{length}(y) = \text{length}(x) + \text{length}(h) - 1$. Therefore, the output of the system is plotted in the double time interval from the input and impulse response signals.

Commands	Results	Comments
<pre>y = conv(x,h); stem(0:6,y) axis([-2 6 .2 -.1 4 .1]) legend('y[n] = h[n]')</pre>		The output $y[n]$ is computed and plotted for $0 \leq n \leq 6$. As expected the output and impulse response signals are the same.

The output and impulse response signals are the same; thus the identity property clearly stands. The system under consideration is a linear shift-invariant system. If at a linear and shift-invariant system, with impulse response $h[n]$, the input $\delta[n - k]$, i.e., a shifted unit impulse sequence is applied, then the response of the system is $h[n - k]$; that is, the impulse response is also shifted by k units.

Commands	Results	Comments
<pre>n = 0 : 3; x = [0 1 0 0]; stem(n,x); axis([- .1 3 .1 -.1 1.1]) legend('x[n] = \delta[n-1]')</pre>		The shifted unit impulse sequence $x[n] = \delta[n - 1]$ is input to the system.
<pre>h = [2 4 3 1]; n = 0 : 3; stem(n,h) axis([- .2 6 .2 -.1 4 .1]) legend('h[n]')</pre>		Impulse response $h[n] = [2 4 3 1]$, $0 \leq n \leq 3$.
<pre>y = conv(x,h); stem(0:6,y) axis([- .2 6 .2 -.1 4 .1]) legend('y[n] = h[n-1]')</pre>		The output of the system is its impulse response shifted by 1 unit to the right.

As expected, the response $y[n]$ of the system to the input signal $\delta[n - 1]$ is $h[n - 1]$. More generally, the response of a linear shift-invariant system to a shifted unit impulse sequence $\delta[n - k]$ is shifted by k units version of its impulse response, i.e., the output of the system is $h[n - k]$.

4.6.2 Computation of Discrete-Time Convolution

The process that has to be followed to analytically derive the convolution sum given in (4.10) is alike but similar to the one followed in the continuous-time case. First, the input and impulse response signals are plotted in the k -axis (corresponding to the τ -axis of the continuous-time case). Then one of the two signals is reversed about the amplitude axis, and its reflection is shifted from $-\infty$ to ∞ by changing appropriately the value of n . The output of the system is computed from the overlapping values of $x[n]$ and $h[n-k]$ according to the convolution sum $y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]$. The convolution procedure for two discrete-time signals is demonstrated by using the signals $x[n]=[1, 2]$, $0 \leq n \leq 1$ and $h[n]=[2, 1, 1, 1]$, $0 \leq n \leq 3$.

Step 1: The two signals are plotted at the k -axis.

Commands	Results	Comments
<pre> kx = [0 1]; x = [1 2]; stem(kx,x); legend('x[k]') axis([-1 3.1 -1 2.1]) </pre>		Input signal $x[k]$.
<pre> kh = 0:3; h = [2,1,1,1]; stem(kh,h); axis([-1 3.1 -1 2.1]) legend('h[k]') </pre>		Impulse response signal $h[k]$.

Step 2: The reflected version of $h[k]$, namely, $h[-k]$ is plotted.

Commands	Results	Comments
<pre>stem(-kh,h) axis([-3.1 .1 -.1 2.1]) legend('h[-k]')</pre>		The reflected signal $h[-k]$.

Step 3: The signal $h[n - k]$ is shifted from $-\infty$ to ∞ by changing appropriately the value of n . The output of the system is computed at *each shift* through Equation 4.10, in which only the values of the overlapping parts of $x[k]$ and $h[n - k]$ are considered. The procedure is alike but similar to the one followed at the continuous-time case. Instead of computing the integral of $x(\tau) \cdot h(t - \tau)$, we calculate the sum of $x[k] \cdot h[n - k]$. Moreover the stages of the process are less. First there is zero overlap, next there is overlap, and finally again there is no overlap.

- First stage: Zero overlap. This stage occurs for $n \leq -1$.

Commands	Results	Comments
<pre>stem(kx,x) axis([-5.1 3.1 -.1 2.1]) legend('x[k]')</pre>		Input signal $x[k]$.

(continued)

Commands	Results	Comments
<pre>n=-2; stem(-kh+n,h); axis([-5.1 3.1 -.1 2.1]) legend('h[-2-k]')</pre>		Impulse response signal $h[n - k]$ for $n = -2$.

Obviously, the two signals do not overlap. Thus, the output is $y[n] = 0, n \leq -1$.

- Second stage: Overlap. The signals $x[n]$ and $h[n - k]$ overlap for $0 \leq n \leq 4$.

In order to compute the output, the two signals are plotted for $n = 0, 1, \dots, 4$. The output is computed through the convolution sum by taking into account only the overlapping samples.

For $n = 0$,

Commands	Results
<pre>stem(kx,x) axis([-5.1 3.1 -.1 2.1]) legend('x(k)')</pre>	
<pre>n=0; stem(-kh+n,h); axis([-5.1 3.1 -.1 2.1]) legend('h(n-k)=h(0-k)')</pre>	

For $n = 0$, the signals $x[k]$ and $h[n - k] = h[0 - k]$ have one overlapping sample at $k = 0$. The output of the system for $n = 0$; that is, $y[n] = y[0]$ is computed as $y[0] = 1 \cdot 2 = 2$, where 1 is the value of $x[k]$ and 2 is the value of $h[n - k]$ at the overlapping sample. In order to understand better the output calculation, notice that according to the definition of the convolution sum $y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n - k]$ the overlapping samples of $x[k]$ and $h[n - k]$ are multiplied. The sum is applied when more than one samples overlap. This is the case for $n = 1$.

Commands	Results
<pre>stem(kx,x) axis([-5.1 3.1 -.1 2.1]) legend('x[k]')</pre>	<p>A stem plot showing the signal $x[k]$ versus k. The x-axis ranges from -5 to 3, and the y-axis ranges from 0 to 2. There are two vertical stems: one at $k = 0$ reaching a height of 1, and another at $k = 1$ reaching a height of 2. Both stems have open circles at their tips. A legend box in the top right corner contains a circle icon and the text "x[k]".</p>
<pre>n=1; stem(-kh+n,h); axis([-5.1 3.1 -.1 2.1]) legend('h[n-k] = h[1-k]')</pre>	<p>A stem plot showing the signal $h[n - k] = h[1 - k]$ versus k. The x-axis ranges from -5 to 3, and the y-axis ranges from 0 to 2. There are three vertical stems: one at $k = -2$ reaching a height of 1, another at $k = -1$ reaching a height of 1, and a third at $k = 0$ reaching a height of 1. All three stems have open circles at their tips. A legend box in the top left corner contains a circle icon and the text "h[n-k] = h[1-k]".</p>

For $n = 1$, the signals $x[k]$ and $h[n - k] = h[1 - k]$ overlap with two samples, namely, at $k = 0$ and at $k = 1$. The output for $n = 1$, that is, $y[n] = y[1]$ is computed as $y[1] = 2 \cdot 2 + 1 \cdot 1$, where 2 is the value of $x[k]$ and $h[n - k]$ at $k = 1$, while 1 is the value of $x[k]$ and $h[n - k]$ at $k = 0$. Hence, $y[1] = 5$.

For $n=2$, we have

Commands	Results
<pre> stem(kx,x) axis([-5.1 3.1 -.1 2.1]) legend('x[k]') </pre>	
<pre> n=2; stem(-kh+n,h); axis([-5.1 3.1 -.1 2.1]) legend('h[n-k]=h[2-k]') </pre>	

For $n=2$, $x[k]$ and $h[n-k]=h[2-k]$ overlap at two points, namely, at $k=0$ and at $k=1$. The output for $n=2$, that is, $y[n]=y[2]$ is computed as $y[2]=2 \cdot 1 + 1 \cdot 1$, where 2 is the value

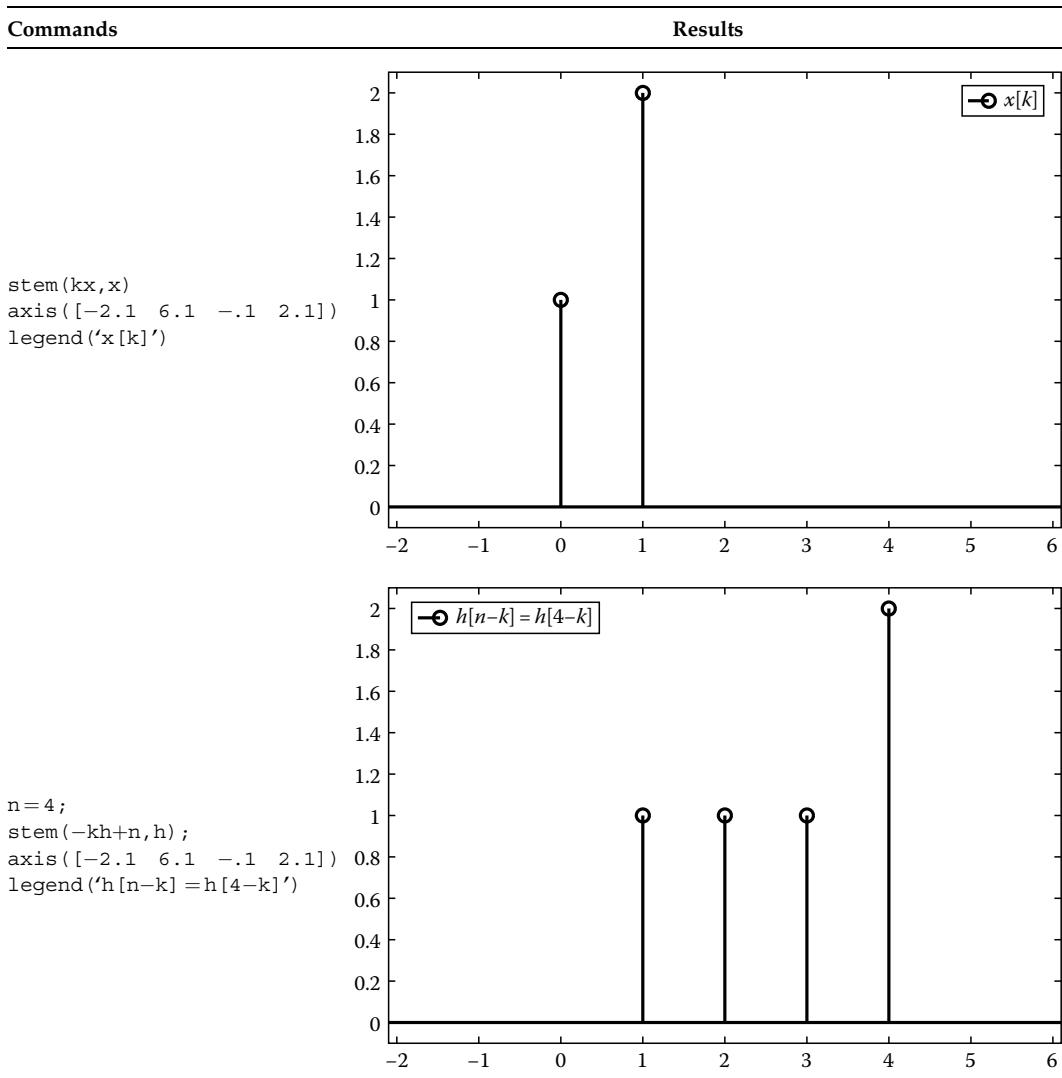
of $x[k]$ and 1 is the value of $h[n-k]$ at $k=1$, while 1 is the value of $x[k]$ and $h[n-k]$ at $k=0$. Hence, $y[2]=3$.

For $n=3$, we have

Commands	Results
<pre> stem(kx,x) axis([-2.1 6.1 -.1 2.1]) legend('x[k]') </pre>	<p>A stem plot showing the signal $x[k]$ versus k. The x-axis ranges from -2 to 6, and the y-axis ranges from 0 to 2. There are two vertical stems: one at $k=0$ reaching a height of 1, and another at $k=1$ reaching a height of 2. A legend in the top right corner identifies the symbol as $x[k]$.</p>
<pre> n=3; stem(-kh+n,h); axis([-2.1 6.1 -.1 2.1]) legend('h[n-k] = h[3-k]') </pre>	<p>A stem plot showing the signal $h[n-k] = h[3-k]$ versus k. The x-axis ranges from -2 to 6, and the y-axis ranges from 0 to 2. There are four vertical stems: one at $k=-3$ reaching a height of 2, and three at $k=0, 1, 2$ each reaching a height of 1. A legend in the top right corner identifies the symbol as $h[n-k] = h[3-k]$.</p>

The output for $n = 3$, i.e., $y[n] = y[3]$ is computed as $y[3] = 2 \cdot 1 + 1 \cdot 1 = 3$.

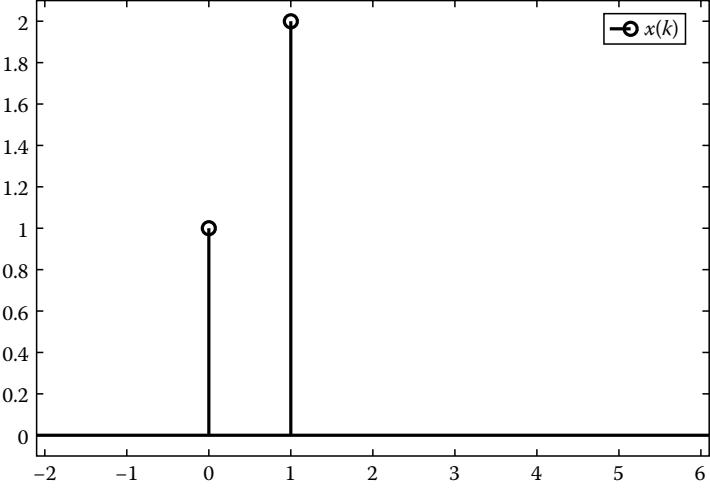
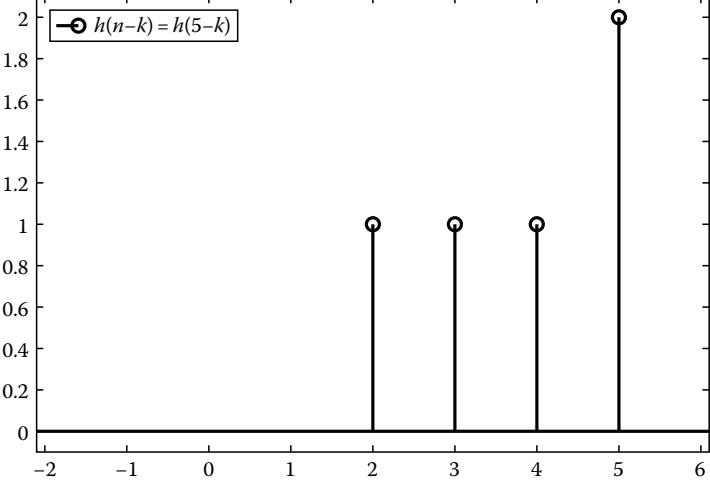
For $n = 4$, we have



For $n = 4$, there is only one overlap point at $k = 1$. Thus, the output for $n = 4$ is $y[4] = 2 \cdot 1 = 2$.

- Third stage: Zero overlap. For $n \geq 5$, the input and impulse response signals do not overlap, hence, $y[n] = 0$, $n \geq 5$.

For $n=5$,

Commands	Results
<pre>stem(kx,x) axis([-2.1 6.1 -.1 2.1]) legend('x(k)')</pre>	
<pre>n=5; stem(-kh+n,h); axis([-2.1 6.1 -.1 2.1]) legend('h(n-k)=h(5-k)')</pre>	

Combining the derived results we conclude that the response of a system with impulse response $h[n] = [2, 1, 1, 1]$, $0 \leq n \leq 3$ to the input signal $x[n] = [1, 2]$, $0 \leq n \leq 1$ is $y[n] = [2, 5, 3, 3, 2]$, $0 \leq n \leq 4$. The output signal is plotted in the figure below.

Commands	Results	Comments
<pre>n = 0 : 4; y = [2, 5, 3, 3, 2]; stem(n, y); axis([-0.1 4.1 -0.1 5.1]) legend('y[n]')</pre>		The discrete-time system response $y[n]$.

We should mention that there are some other methods available for the analytical computation of discrete (and continuous)-time convolution. However, the authors believe that the graphical illustration is the best way to understand the convolution process.

4.6.3 Discrete-Time Convolution Examples

In this section, several examples on computing the convolution between two discrete-time signals by using the command `conv` are given. There are two basic principles that should be considered when computing the convolution between two discrete-time signals $x[n]$ and $h[n]$.

The Two Principles of Convolution

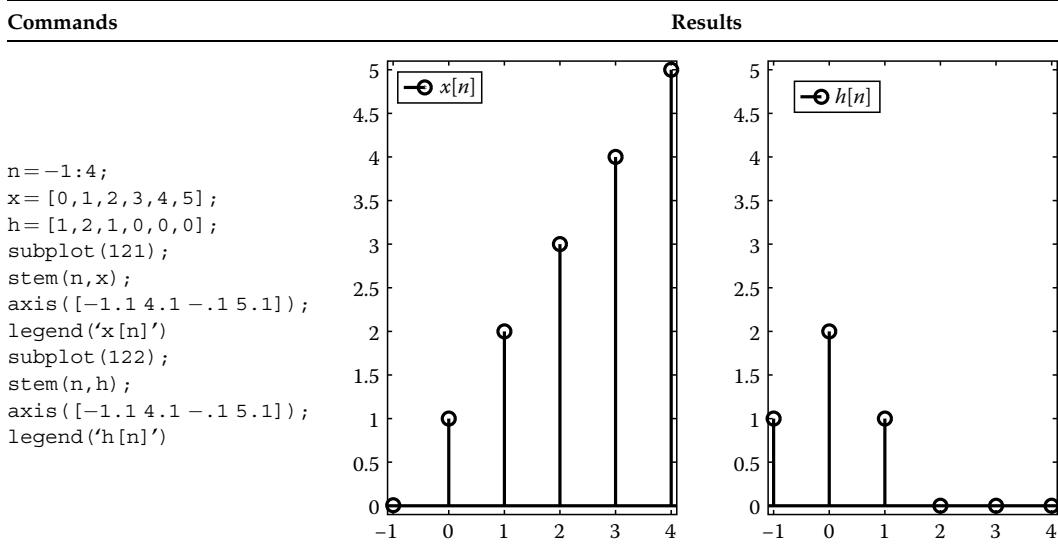
1. Suppose that the length of vector x is N and the length of vector h is M . The outcome of the command $y = \text{conv}(x, h)$ is a vector y of length $M + N - 1$. In other words, $\text{length}(y) = \text{length}(x) + \text{length}(h) - 1$.
2. If the nonzero values of $x[n]$ are in the interval $[a_x, b_x]$ and the nonzero values of $h[n]$ are in the interval $[a_h, b_h]$ then the nonzero values of the output $y[n]$ are in the interval $[a_x + a_h, b_x + b_h]$.

Example

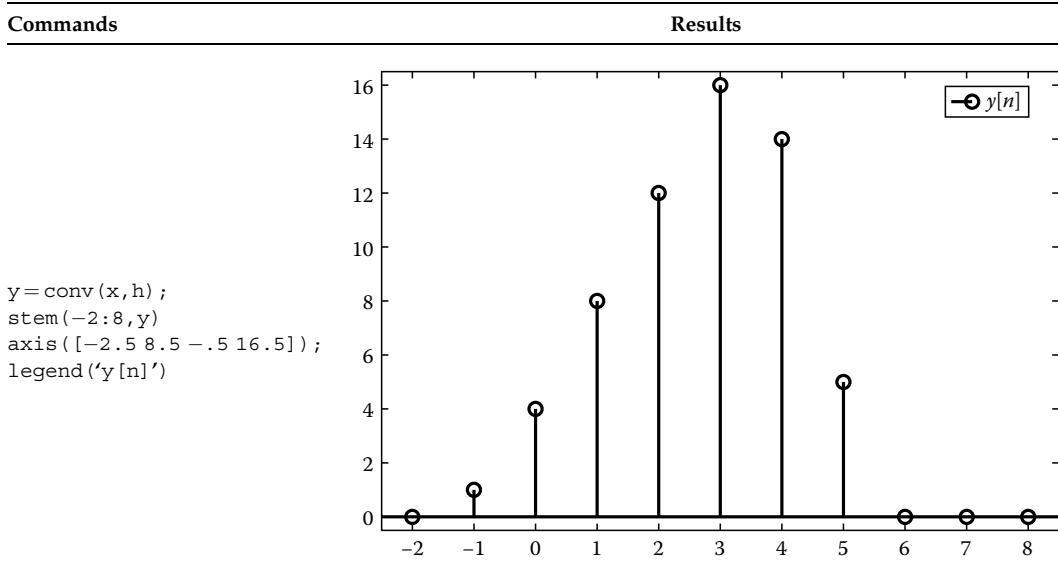
Suppose that the impulse response of a system is $h[n] = [1, 2, 1, 0, 0, 0]$, $-1 \leq n \leq 4$. Compute the response of the system to the input signal $x[n] = [1, 2, 3, 4, 5]$, $0 \leq n \leq 4$.

Two different but equivalent procedures of computing the output $y[n]$, which is the convolution between $x[n]$ and $h[n]$, are introduced.

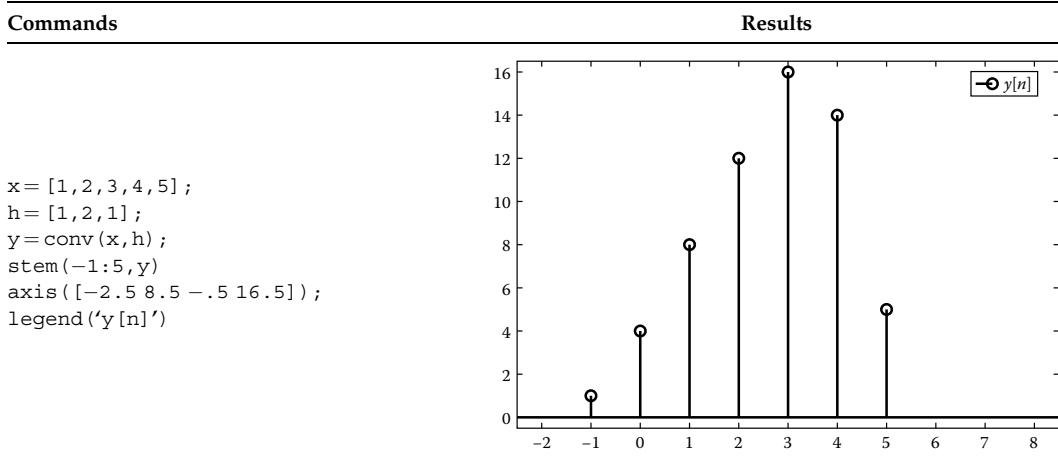
First procedure: According to the four rules of convolution established for continuous-time signals in Section 4.2.2. First, the two signals are defined in the same time interval (first rule). Hence, the input signal is formulated as $x[n] = [0, 1, 2, 3, 4, 5]$, $-1 \leq n \leq 4$, while the mathematical expression of the impulse response signal is $h[n] = [1, 2, 1, 0, 0, 0]$, $-1 \leq n \leq 4$. For illustration purposes $x[n]$ and $h[n]$ are plotted below.



There is no need to apply the second rule as the two signals consist of only one part. Moreover, the third rule, which is the multiplication between the output of the conv command and the time step, is trivial in the discrete-time case since the value of the time step is one. Therefore, after the convolution computation we proceed according to the fourth rule, i.e., we plot the output signal in the double time interval from the one that the input and the impulse response signals are defined.



Second procedure: The second approach is somewhat easier from the first. It is based on the second principle of convolution. The idea is to convolute only the nonzero parts of the signals. The output obtained is plotted in the appropriate time interval that is defined according to the second principle. Since only the nonzero parts of $x[n]$ and $h[n]$ are considered, the two signals are defined as $x[n]=[1,2,3,4,5]$, $0 \leq n \leq 4$ and $h[n]=[1,2,1]$, $-1 \leq n \leq 1$. The output $y[n]$ is plotted in the time interval $-1 \leq n \leq 5$. In order to understand better the selected time interval in which the output signal will be plotted, notice that the point a_x where $x[n]$ takes the first nonzero value is at $n=0$, while the point a_h where $h[n]$ takes the first nonzero value is at $n=-1$. Thus, $y[n]$ takes the first nonzero value at the point $a_y = a_x + a_h = -1$. Similarly, the end points are at $n=1$ and $n=4$ for $x[n]$ and $h[n]$, respectively; thus the end point of $y[n]$ is at $n=5$.



As expected, the output signals obtained from the two different procedures are the same.

Example

Suppose that the impulse response of a system is

$$h[n] = \begin{cases} n, & -5 \leq n \leq 5 \\ 0, & \text{elsewhere} \end{cases}.$$

Compute the response of the system to the input signal

$$x[n] = \begin{cases} 1, & 10 \leq n \leq 20 \\ 0, & \text{elsewhere} \end{cases}.$$

Again the response of the system is computed through the two different procedures introduced in the previous example.

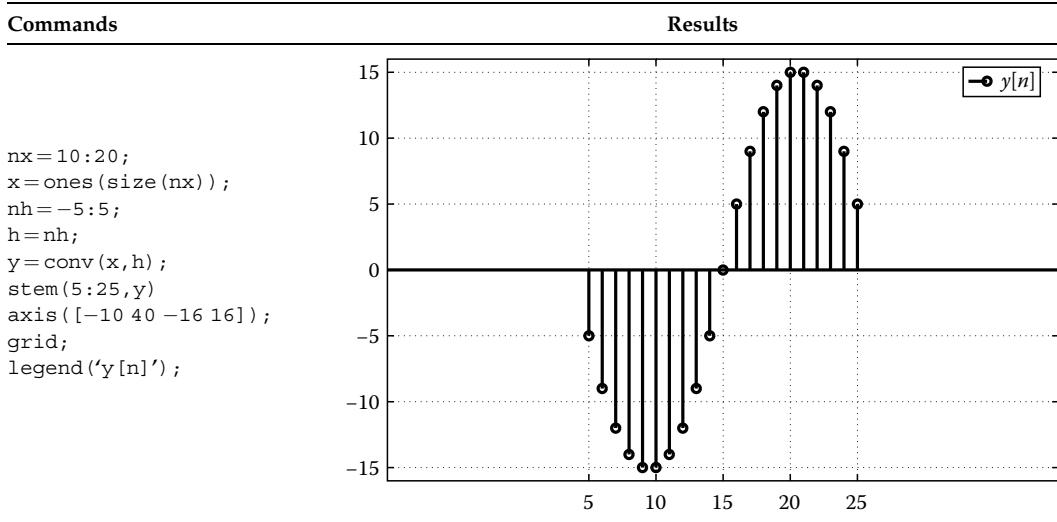
First procedure: The input and impulse response signals are defined in the same interval, i.e., are defined as

$$x[n] = \begin{cases} 0, & -5 \leq n \leq 9 \\ 1, & 10 \leq n \leq 20 \end{cases} \quad \text{and} \quad h[n] = \begin{cases} n, & -5 \leq n \leq 5 \\ 0, & 6 \leq n \leq 20 \end{cases}.$$

The output of the system obtained by their convolution is plotted in the time interval $-10 \leq n \leq 40$.

Commands	Results	Comments
<pre>n1=-5:9; x1=zeros(size(n1)); n2=10:20; x2=ones(size(n2)); x=[x1 x2]; n1=-5:5; h1=n1; n2=6:20; h2=zeros(size(n2)); h=[h1 h2]; y=conv(x,h); stem(-10:40,y); ylim([-16 16]); grid; legend('y[n]');</pre>		<p>The output of the system $y[n] = x[n]*h[n]$.</p>

Second procedure: Only the nonzero parts of $x[n]$ and $h[n]$ are defined. According to the second principle, the nonzero elements of the output $y[n] = x[n]*h[n]$ of the system lie in the time interval [5, 25].



The output signals obtained from the two different procedures are equal; thus the two computational procedures are equivalent.

4.7 Systems Described by Difference Equations

In the previous sections, we have computed the convolution of two finite element sequences. Suppose now that the impulse response sequence $h[n]$ of a system consists of infinite nonzero terms. In this case, the output of the system cannot be computed with use of the command `conv`. In order to deal with this case, a different way of describing an LTI system is now established. A discrete-time LTI system can be completely described by a *difference equation*. Difference equations are equations of the form

$$\sum_{k=0}^N a_k y[n-k] = \sum_{k=0}^M b_k x[n-k]. \quad (4.11)$$

The system information that is present in the possibly infinite nonzero terms of the impulse response $h[n]$ of the system is revealed from the finite coefficients of the difference equation a_k and b_k .

Suppose that $a_0 = 1$. Then the output of the system is given by

$$y[n] = - \sum_{k=1}^N a_k y[n-k] + \sum_{k=0}^M b_k x[n-k]. \quad (4.12)$$

The value of $y[n]$ for $n = 0$, i.e., $y[0]$, is computed directly from (4.12) as

$$y[0] = - \sum_{k=1}^N a_k y[-k] + \sum_{k=0}^M b_k x[-k]. \quad (4.13)$$

Thus, in order to calculate $y[0]$, except from the value of the input signal $x[0]$ at $n = 0$, we must also know the initial conditions $x[-1], \dots, x[-M]$ and $y[-1], \dots, y[-N]$.

4.8 Filters

An LTI system is also called filter due to the fact that the input signal is modified or filtered according to some rules that characterize the system. A discrete-time filter is described by a linear difference equation with constant coefficients of the form

$$y[n] = \sum_{k=0}^M b[k]x[n-k] - \sum_{k=1}^N a[k]y[n-k]. \quad (4.14)$$

The impulse response $h[n]$ of the filter is the solution of the difference equation if the input signal applied to the signal is the unit impulse sequence, namely, if $x[n] = \delta[n]$. LTI filters are classified in two categories based on the length of their impulse response sequence. These two categories are the *finite impulse response (FIR)* filters and the *infinite impulse response (IIR)* filters. A FIR filter has impulse response of finite duration and is described by a difference equation of the form

$$y[n] = \sum_{k=0}^M b[k]x[n-k], \quad (4.15)$$

while the impulse response of a FIR filter is given by

$$h[n] = \sum_{k=0}^M b[k]\delta[n-k] \quad (4.16)$$

On the other hand, an IIR filter has an impulse response of infinite duration and is described by the difference equation given in (4.11) or equivalently in (4.14).

4.8.1 The Command `filter`

The response of a filter (or system) described by a linear difference equation with constant coefficients is calculated by using the MATLAB command `filter`. Suppose that a system is described by the difference equation given in (4.11). Hence, the input vector $x = [x[0], x[1], \dots, x[n]]$ and the coefficient vectors $a = [a[0], a[1], \dots, a[N]]$ and $b = [b[0], b[1], \dots,$

$b[M]$] are available. The command `y=filter(b,a,x)` returns the discrete-time response $y=[y[0], y[1], \dots, y[n]]$ of the system to the applied input sequence $x=[x[0], x[1], \dots, x[n]]$.

Remark

The output signal $y[n]$ has the same number of samples (or elements) with the input signal $x[n]$.

Example

Consider the system described by the difference equation $y[n] = 0.5y[n - 1] - 0.1y[n - 2] + 0.1x[n] - 0.5x[n - 1] + x[n - 2]$ and assume that the initial conditions are zero.

Compute in the time interval $0 \leq n \leq 9$

- The response of the system $y[n]$ to the input signal $x[n] = [1, 2, -1], 0 \leq n \leq 2$
- The impulse response $h[n]$ of the system
- The difference equation is written as $y[n] - 0.5y[n - 1] + 0.1y[n - 2] = 0.1x[n] - 0.5x[n - 1] + x[n - 2]$. Hence, in MATLAB we type

Commands	Results	Comments
<pre>N = 10; a = [1 -0.5 0.1]; b = [0.1 -0.5 1]; x = [1 2 -1 zeros(1, 7)]; y = filter(b, a, x); stem(0:N-1, y); legend('y[n]'); </pre>		The response of the system $y[n]$ to the input signal $x[n] = [1, 2, -1], 0 \leq n \leq 2$.

The output signal $y[n]$ is directly computed as the outcome of the command `filter`. Notice that $x[n]$ is padded with zeros in order to define it for $0 \leq n \leq 9$. The coefficients $b[k]$ of the input signal $x[n]$ are the first argument in the `filter` command, while the coefficients $a[k]$ of the output signal $y[n]$ are the second argument.

- Two alternative approaches are given in order to compute the impulse response $h[n]$ of the system.

First approach: The command `filter` is employed. Recall that if $x[n] = \delta[n]$ the solution $y[n]$ of the difference equation is actually the impulse response $h[n]$. Hence, the procedure is to apply the unit impulse sequence $\delta[n]$ as input to the system and compute the response of the system which is the impulse response $h[n]$ of the system. The unit impulse sequence $\delta[n]$ is defined as

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & 0 < n \leq 9 \end{cases}$$

Commands	Results	Comments
<pre>N=10; a=[1 -0.5 0.1]; b=[0.1 -0.5 1]; d=[1 zeros(1,N-1)]; h=filter(b,a,d); stem(0:N-1,h); xlim([-0.5 10]); legend('h[n]'); </pre>		The impulse response of the system $h[n]$.

Second approach: The second approach is to substitute the input $x[n]$ by $\delta[n]$ and directly compute the solution from the difference equation based on a recursive algorithm implemented with a for-loop. Since $y[n] = x[n] = 0$, $n < 0$, for $n = 0$ the difference equation becomes $y[0] = 0.1x[0] = 0.1\delta[0] = 0.1 = h[0]$, while for $n = 1$ the difference equation becomes $y[1] = 0.5y[0] + 0.1x[1] - 0.5x[0] = 0.5 \cdot 0.1 + 0.1\delta[1] - 0.5\delta[0] = 0.05 + 0 - 0.5 = -0.45 = h[1]$ and so on for $n = 3, 4, \dots$. However, in MATLAB we cannot use zero index in a matrix; hence, in order to implement the for-loop the output signal $y[n]$ and the input signal $\delta[n]$ must be shifted by 1 unit to the right. Thus, the input signal from

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

becomes

$$\delta[n] = \begin{cases} 1, & n = 1 \\ 0, & n \neq 1 \end{cases}$$

Hence, applying the newly defined $\delta[n]$ to the difference equation yields the following:

$$\text{For } n = 1 \Rightarrow y[1] = h[1] = 0.1x[1] = 0.1\delta[1] = 0.1$$

$$\text{For } n = 2 \Rightarrow y[2] = h[2] = 0.5h[1] - 0.5x[1] = 0.5h[1] - 0.5\delta[1] = 0.5 \cdot 0.1 - 0.5 \cdot 1 = -0.45.$$

These two initial values are enough. The MATLAB code is

Commands	Results	Comments
<pre>N=10; x=zeros(N,1); x(1)=1; h(1)=0.1; h(2)=-0.45; for m=3:N h(m)=0.5*h(m-1)-0.1*h(m-2)+0.1*x(m)-0.5*x(m-1)+x(m-2); end stem(0:N-1,h); xlim([-0.3 9.3]); legend('h[n]');</pre>		The impulse response $h[n]$ of the system.

The two graphs are same; hence, the two approaches are equivalent. The command `filter` can be used to compute the response of a system described by a difference equation also when the initial conditions are not zero. In this case, the proper syntax is `y = filter(b, a, x, zi)`, where `zi` is a vector associated with the initial conditions. However, in order to correctly determine vector `zi` one more MATLAB command must be employed. More specifically, one has to type `zi =filtic(b, a, yit, xit)` where $yit = [y[-1], y[-2], \dots, y[-nb]]$ and $xit = [x[-1], x[-2], \dots, x[-na]]$ are the initial conditions of $y[n]$ and $x[n]$, respectively. The syntax `zi =filtic(b, a, yit)` implies that $x[n]$ is zero for $n < 0$.

Example

Consider the system described by the difference equation $y[n] = 0.5y[n-1] - 0.1y[n-2] + 0.1x[n] - 0.5x[n-1] + x[n-2]$ with initial conditions $y[-1] = 0.3$, $y[-2] = 0.5$, and $x[-1] = x[-2] = 0.4$. Compute for $0 \leq n \leq 9$ the response of the system $y[n]$ to the input signal $x[n] = [1, 2, -1], 0 \leq n \leq 2$. Compare your result to the one derived in the previous example where the initial conditions were zero.

Commands	Results	Comments
<pre>a=[1 -0.5 0.1]; b=[0.1 -0.5 1]; yit=[0.3 0.5]; xit=[0.4 0.4]; zi=filtic(b,a,yit,xit)</pre>	$zi = -5.1000 - 2.6000$	First, the vector <code>zi</code> is defined with use of the command <code>filtic</code> .

(continued)

(continued)

Commands	Results	Comments
<pre>n=0:9; x=[1 2 -1 zeros(1,7)]; y=filter(b,a,x,zi); stem(n,y); xlim([-0.3 9.3]); title('y[n] with initial conditions');</pre>		The response $y[n]$ of the system to $x[n] = [1, 2, -1], 0 \leq n \leq 2$ (and generally to any input signal) is different when the initial conditions are different.

To verify our result (and illustrate one more way of working with difference equations) we compute the output signal directly from the difference equation using the recursive algorithm that was implemented in the previous example for the computation of the impulse response.

The first two values of $y[n]$, i.e., $y[0]$ and $y[1]$ are computed as $y[0] = 0.5y[-1] - 0.1y[-2] + 0.1x[0] - 0.5x[-1] + x[-2] = 0.4$ and $y[1] = 0.5y[0] - 0.1y[-1] + 0.1x[1] - 0.5x[0] + x[-1] = 0.27$. As in the previous example, due to the fact that we cannot use zero indices the signals $x[n]$ and $y[n]$ are shifted 1 unit to the right. The recursive algorithm is

Commands	Results	Comments
<pre>L=10; x=[1 2 -1 zeros(1,7)]; y=[0.4 0.27 zeros(1,8)]; for m=3:L y(m)=0.5*y(m-1)-0.1*y(m-2)+0.1*x(m)-0.5*x(m-1)+x(m-2); end stem(0:L-1,y); xlim([-0.3 9.3]); legend('y[n]');</pre>		The output signal $y[n]$ computed directly from the difference equation that describes the system.

As expected the two graphs are identical; hence, the two ways are equivalent.

4.8.2 Infinite Impulse Response Filters

In this section, we introduce the infinite (duration) impulse response (IIR) filters through analytical examples.

Example

Consider the system described by the linear difference equation with constant coefficients $y[n] - 1.1y[n-1] + 0.9y[n-2] = x[n]$ with zero initial conditions. Compute and plot for the time interval $0 \leq n \leq 100$

- The response of the system $y[n]$ to the input signal $x[n] = [1, 2, -1], 0 \leq n \leq 2$
- The impulse response of the system $h[n]$

The system described by the difference equation is an *infinite impulse response filter (IIR)*. This means that the system impulse response consists of infinite nonzero samples. However, we calculate only the first 101 samples of $h[n]$; that is, we calculate $h[n], 0 \leq n \leq 100$.

- The vectors a , b , and x are defined and the output is calculated with use of the command `filter`.

Vector a consists of the coefficients of the samples of the output signal $y[n]$, while vector b consists of the coefficients of the input signal samples $x[n]$. Finally, the input signal $x[n]$ must be defined for $0 \leq n \leq 100$. This is done by padding vector x with 98 zeros.

Commands	Results	Comments
<pre> a = [1 -1.1 0.9]; b=1; n=0:100; x1=[1 2 -1]; n2=3:100; x2=zeros(size(n2)); x=[x1 x2]; y=filter(b,a,x); stem(n,y); axis([-2 102 -3 3.5]); legend('y[n]'); </pre>		Graph of the output signal $y[n]$.

- The impulse response of the filter is derived by applying the signal $x[n] = \delta[n]$ as input to the IIR filter. The impulse response is computed using the command `filter`.

Commands	Results	Comments
<pre>a = [1 -1.1 0.9]; b = 1; n = 0:100; x = [1, zeros(1,100)]; h = filter(b,a,x); stem(n,h); axis([-2 102 -1 1.2]); legend('h[n]');</pre>		The impulse response of the filter $h[n]$.

Notice that the impulse response $h[n]$ of the IIR filter has indeed infinite nonzero samples. In the next example, the relationship between the command `filter` and the command `conv` is discussed.

Example

Consider the system (which is an IIR filter) described by the difference equation $y[n] - 1.1y[n - 1] + 0.5y[n - 2] + 0.3y[n - 4] = 0.5x[n] - 0.2x[n - 1]$. Compute and plot

- a. The impulse response of the system for $0 \leq n \leq 10$
- b. The response $y[n]$ of the system to the input signal $x[n] = [5, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0]$, $0 \leq n \leq 10$, by using the command `conv` and the command `filter`
- a. Recall that the statement $x=(n==0)$ returns 1 if $n=0$ and 0 if $n \neq 0$. Hence, if $n=0, 1, \dots, 10$ the vector x corresponds to the unit impulse sequence $\delta[n] = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$, $0 \leq n \leq 10$. After defining $\delta[n]$, we compute the impulse response of the system by executing the command `filter`.

Commands	Results	Comments
<pre>n = 0:10; a = [1 -1.1 0.5 0 0.3]; b = [0.5 -0.2]; x = (n == 0); h = filter(b,a,x); stem(n,h) xlim([-0.5 10.5]); legend('h[n]');</pre>		The impulse response $h[n]$ of the filter.

b1. Computation of the output signal with use of the command `conv`.

Having calculated the impulse response $h[n]$, the output signal is easily obtained by convoluting the input sequence $x[n]$ with the impulse response $h[n]$. The derived output sequence $y[n]$ consists of 21 samples.

Commands	Results	Comments
<pre> x = [5 1 1 1 0 0 1 1 1 0 0]; y1=conv(x,h); stem(0:20,y1); xlim([-0.5 20.5]); legend('y[n] from conv') </pre>		Output $y[n]$ computed with use of the command <code>conv</code> .

b2. Computation of the output signal with use of the command `filter`.

In order to obtain 21 samples as output from the command `filter`, the input signal has to be defined for $0 \leq n \leq 20$, i.e., has to be padded with zeros. This is done with the statement $x(21) = 0$. This statement pads with zeros the undefined positions of x up to the 21st position. The `filter` command can now be applied to compute the 21 samples of the output signal.

Commands	Results	Comments
<pre> a=[1 -1.1 0.5 0 0.3]; b=[0.5 -0.2]; x=[5 1 1 1 0 0 1 1 1 0 0]; x(21)=0; y2=filter(b,a,x); stem(0:20,y2); xlim([-0.5 20.5]); legend('y[n] from filter') </pre>		Output $y[n]$ computed with use of the command <code>filter</code> .

The two graphs are same only at the first 11 samples. So which of the two computed outputs is the correct one? The answer is that the result provided by the command `filter` is the correct one. The result provided by the `conv` command is partially wrong as the

impulse response of an IIR filter is of infinite duration; that is, the impulse response $h[n]$ consists of infinite nonzero terms. However, in the beginning of our example, only the first 11 samples of $h[n]$ were computed. All other samples were considered zero; thus the computed $h[n]$ is a truncated version of the real one. Having computed the first 11 samples of $h[n]$ results in the correct computation of the first 11 samples of $y[n]$. The rest of the samples obtained with use of the command `conv` are not accurate. The conclusion is that when dealing with an IIR filter, if the output is derived based on the use of the system impulse response, the result will not be accurate because of the truncation of the impulse response samples.

Hence, the best way to deal with IIR filters is to describe them by their difference equation and use the MATLAB command `filter` in order to compute the output signal.

4.8.3 Finite Impulse Response Filters

In this section, we introduce the finite-duration impulse response (FIR) filters through analytical examples. As mentioned earlier, the output of a FIR filter is computed by

$$y[n] = \sum_{k=0}^M b[k]x[n - k], \quad (4.17)$$

while its impulse response is given by

$$h[n] = \sum_{k=0}^M b[k]\delta[n - k]. \quad (4.18)$$

Example

Consider a system (which is a FIR filter) described by the difference equation $y[n] = 3x[n] - 2x[n - 1] + 4x[n - 2]$. Compute for $0 \leq n \leq 10$

- a. The impulse response $h[n]$ of the system
- b. The response $y[n]$ of the system to the input signal $x[n] = [1, -1, 3], 0 \leq n \leq 2$

- a. The impulse response $h[n]$ is computed with use of the command `filter` by applying the unit impulse sequence $\delta[n]$ as input signal.

Commands	Results	Comments
<pre>a=1; b=[3 -2 4]; n=0:10; x=[1 zeros(1,10)]; h=filter(b,a,x); stem(n,h);</pre>		<p>The impulse response of the system is $h[n] = [3, -2, 4]$, $0 \leq n \leq 2$.</p>

Notice that the impulse response of the system is same with the coefficients $b[k]$ of the difference equation. This is an expected result (see Equation 4.18) as the impulse response of a FIR filter is

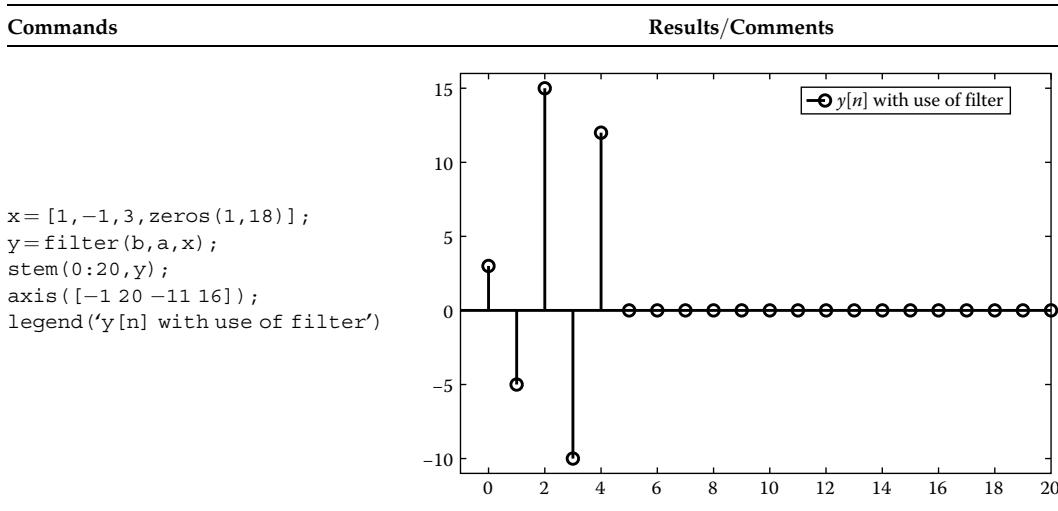
$$h[n] = \sum_{k=0}^M b[k]\delta[n-k] = b[0]\delta[n] + b[1]\delta[n-1] + \dots + b[M]\delta[n-M] = \begin{cases} b[0], & n = 0 \\ b[1], & n = 1 \\ \vdots \\ b[M], & n = M. \end{cases}$$

- b. The response $y[n]$ of the system to the input signal $x[n]$ is computed by two different (and equivalent for FIR filters) ways.

First way: With use of the command conv.

Commands	Results
<pre>x=[1 -1 3 zeros(1,8)]; y=conv(x,h); stem(0:20,y); axis([-1 20 -11 16]); legend('y[n] with use of conv')</pre>	

Second way: With use of the command `filter`. In a similar manner to the procedure followed in the IIR filter case, the input signal $x[n]$ is padded with the appropriate number of zeros.



The two graphs are the same, therefore we conclude that when we deal with FIR filters we can use both commands `conv` and `filter` to compute the output signal.

4.9 Stability Criterion for Discrete-Time Systems

In Section 4.5, we have established a criterion about the stability of continuous-time LTI systems. More specifically, it was stated that a system is stable if the impulse response of the system is absolutely integrable. For discrete-time systems a similar criterion can be established. More specifically, a discrete-time linear shift-invariant system is stable if and only if its impulse response $h[n]$ is absolutely summable. The mathematical expression is

$$\sum_{n=-\infty}^{\infty} |h[n]| < \infty. \quad (4.19)$$

Example

A system is described by the impulse response $h[n] = (1/2^n)u[n]$. Tell if this is a BIBO stable system.

A discrete-time system is BIBO stable if the condition given in (4.19) is satisfied; that is, we examine if the impulse response of the system is absolutely summable.

Commands	Results	Comments
<code>syms n h = 1/(2^n) symsum(abs(h), n, 0, inf)</code>	<code>ans = 2</code>	Condition (4.19) is fulfilled; hence, the system under consideration is BIBO stable.

4.10 Systems Described by Differential Equations

In Section 4.7, we saw how a discrete-time system is described by a difference equation. Similarly, a continuous-time system can be described by a differential equation. More specifically, the input and output signals are related through a linear ordinary differential equation with constant coefficients; i.e., the input–output relationship of the system is of the form

$$a_n \frac{d^n y(t)}{dt^n} + \cdots + a_1 \frac{dy(t)}{dt} + a_0 y(t) = b_m \frac{d^m x(t)}{dt^m} + \cdots + b_1 \frac{dx(t)}{dt} + b_0 x(t). \quad (4.20)$$

The response $y(t)$ of the system to a given input signal $x(t)$ is easily obtained by solving the differential equation. Recall that the MATLAB command `dsolve` is employed to solve ordinary differential equations. Recall also that for a given function $x(t)$ there are many different solutions of this differential equation depending on the initial values (or initial conditions) $y(0), y'(0), \dots, y^{(m-1)}(0)$. If the initial conditions are zero, that is, $y(0) = y'(0) = \dots = y^{(m-1)}(0) = 0$, then the system is said to be in *initial rest*.

Example

Suppose that a system is described by the first-order differential equation $y'(t) + y(t) = x(t)$. Suppose also that the system has zero initial conditions; that is, $y(0) = 0$. Compute the response of the system to the input signal $x(t) = e^{-t} u(t)$.

Commands	Results	Comments
<code>f = 'Dy+y-exp(-t)*heaviside(t)' y = dsolve(f, 'y(0) = 0')</code>	<code>f = Dy+y-exp(-t)*heaviside(t) y = exp(-t)*heaviside(t)*t</code>	Definition of the differential equation $y'(t) + y(t) = x(t)$, where $x(t) = e^{-t} u(t)$.

(continued)

(continued)

Commands	Results	Comments
<pre>ezplot(y, [0 10]) title('y(t) from differential equation')</pre>	<p>$y(t)$ from differential equation</p>	The response of the system to the input signal $x(t) = e^{-t}u(t)$ is derived by solving the differential equation. The computed output signal is $y(t) = te^{-t}u(t)$.

In Chapter 11, we will learn how to derive the impulse response of a system described by a differential equation. For now, take as granted that the impulse response $h(t)$ of the system described by the differential equation $y'(t) + y(t) = x(t)$, $y(0) = 0$ is $h(t) = e^{-t}u(t)$. We confirm that the response of the system to $x(t) = e^{-t}u(t)$ is $y(t) = te^{-t}u(t)$ by convoluting the input signal with the impulse response $h(t)$ of the system.

Commands	Results	Comments
<pre>t = 0 : 0.001 : 5; x = exp(-t); h = exp(-t); y = conv(x,h)*0.001; plot(0:0.001:10,y); title('y(t) from convolution')</pre>	<p>$y(t)$ from convolution</p>	The output $y(t)$ of the system is computed by convoluting the input signal $x(t) = e^{-t}u(t)$ with the impulse response signal $h(t) = e^{-t}u(t)$. The obtained graph is same as the previous one; hence, $y(t) = te^{-t}u(t)$ is indeed the output signal.

4.11 Step Response of a System

A last topic that is discussed in this chapter is the step response of a system. The step response $s(t)$ of a continuous-time system is defined as the response of the system to the unit step function $u(t)$, and correspondingly the step response $s[n]$ of a discrete-time system is the response of the system to the unit step sequence $u[n]$.

Example

Compute in the time interval $0 \leq n \leq 100$ the step response of the system described by the difference equation $y[n] = y[n - 1] + 0.2x[n] + 0.1x[n - 1]$.

The input $x[n] = u[n]$, $0 \leq n \leq 100$ is applied to the system and the output signal $y[n]$ that is returned from the command filter is the step response of the system.

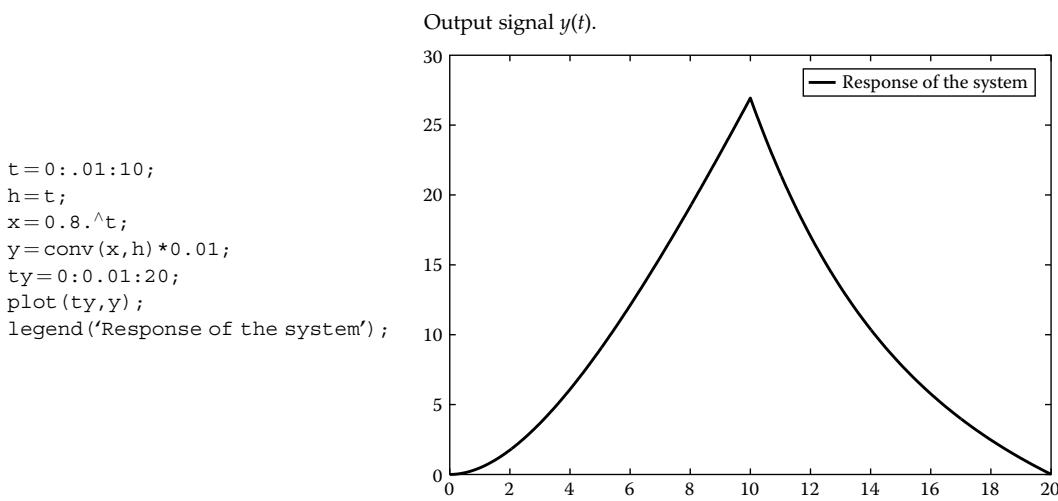
Commands	Results	Comments
<pre>a = [1 -1]; b = [.2, 1]; n = 0:100; u = ones(size(n)); s = filter(b, a, u); stem(0:100, s); title('Step response of the system')</pre>		<p>The step response of the system in the time interval $0 \leq n \leq 100$.</p>

The step response of a system can be also computed by the commands step and dstep in the continuous-time and discrete-time case, respectively. These two commands are discussed in Chapter 11.

4.12 Solved Problems

Problem 1 A system is described by the impulse response $h(t) = t$, $0 \leq t \leq 10$. Compute and plot the response of the system to the input signal $x(t) = 0.8^t$, $0 \leq t \leq 10$.

Solution

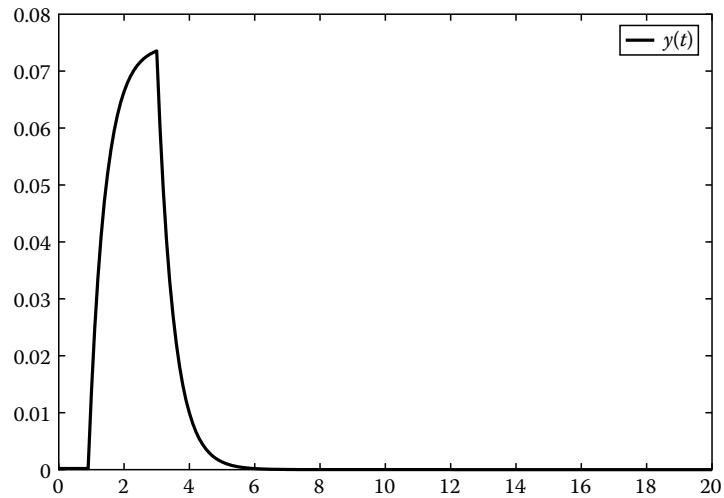


Problem 2 A system is described by the impulse response $h(t) = e^{-2t}u(t - 1)$. Compute and plot the response of the system to the input signal $x(t) = u(t) - u(t - 2)$.

Solution

```
t1=0:.1:.9;
t2=1:.1:10;
h1=zeros(size(t1));
h2=exp(-2*t2);
h=[h1 h2];
t1=0:.1:2;
t2=2.1:.1:10;
x1=ones(size(t1));
x2=zeros(size(t2));
x=[x1 x2];
y=conv(x,h)*0.1;
plot(0:.1:20,y);
legend('y(t)');
```

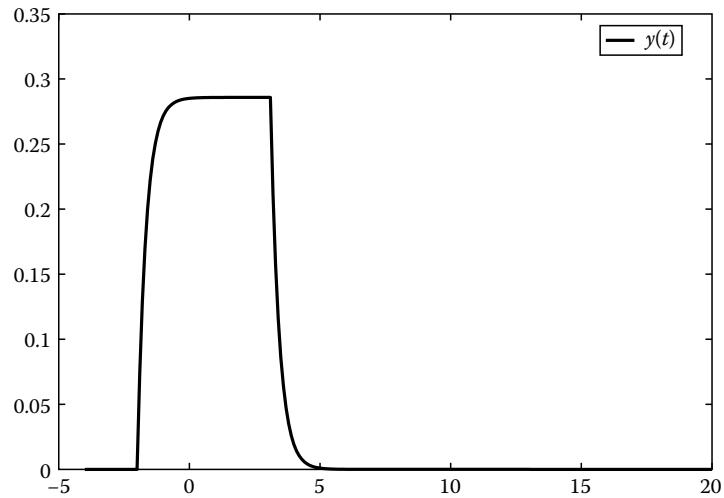
The response $y(t)$ of the system.



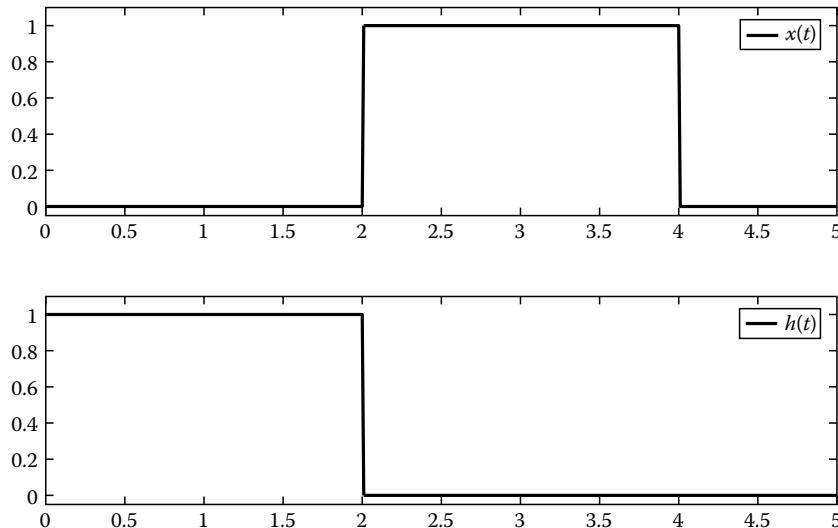
Problem 3 The input signal $x(t) = u(t + 2) - u(t - 3)$ is applied to a system described by the impulse response $h(t) = e^{-3t}u(t)$. Compute and plot the output signal of the system.

Solution

```
t1=-2:.1:3;
x1=ones(size(t1));
t2=3.1:.1:10;
x2=zeros(size(t2));
x=[x1 x2];
t1=-2:.1:0;
h1=zeros(size(t1));
t2=0.1:.1:10;
h2=exp(-3*t2);
h=[h1 h2];
y=conv(x,h)*.1;
plot(-4:.1:20,y)
legend('y(t)');
```



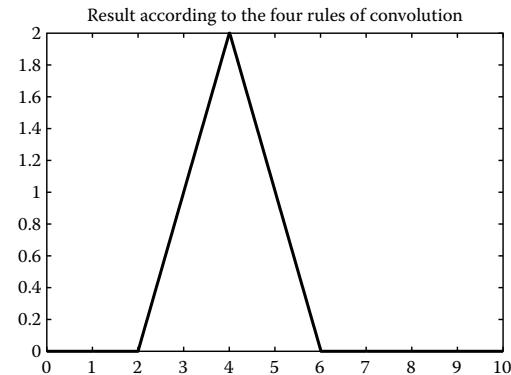
Problem 4 Compute and plot the convolution between the signals that are depicted in the figure below.



Solution

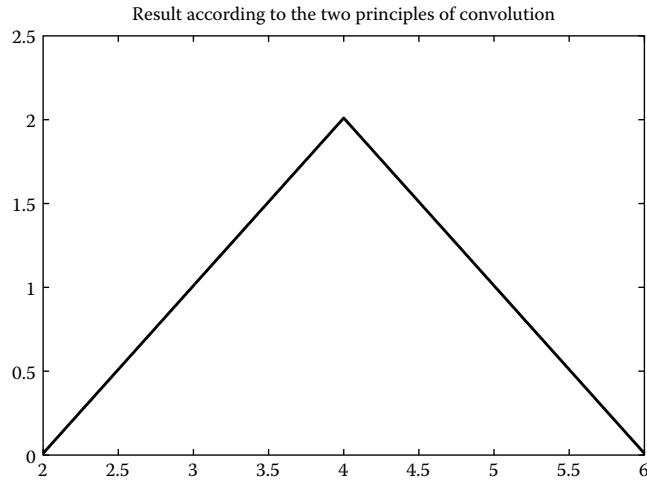
First the convolution is computed according to the four rules established in the continuous-time convolution section.

```
t1 = 0:.01:2;
t2 = 2.01:.01:4;
t3 = 4.01:.01:5;
x1 = zeros(size(t1));
x2 = ones(size(t2));
x3 = zeros(size(t3));
x = [x1 x2 x3];
h1 = ones(size(t1));
h2 = zeros(size([t2 t3]));
h = [h1 h2];
y = conv(x,h)*0.01;
plot(0:0.01:10,y);
title('Result according to the
four rules of convolution')
```



Alternatively, the convolution is computed according to the two principles introduced in the discrete-time convolution section. As it is illustrated in this problem, the two principles can also be applied successfully also to the continuous-time case.

```
t1=2:.01:4;
x=ones(size(t1));
t2=0:.01:2;
h=ones(size(t2));
y=conv(x,h)*0.01;
plot(2:.01:6,y)
title('Result according to the
two principles of convolution')
```

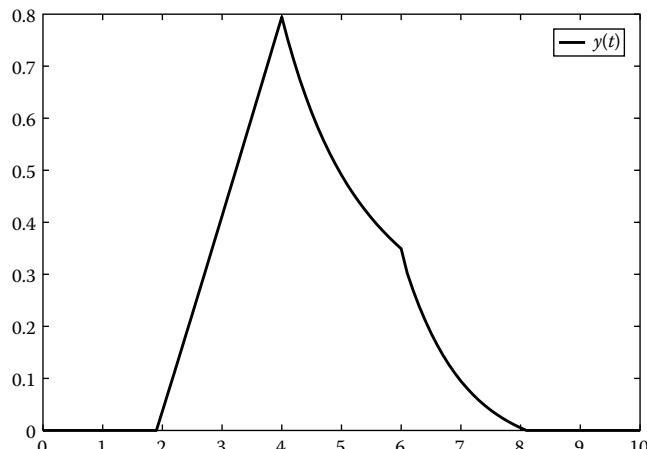


Problem 5 A system is described by the impulse response $h(t) = e^{-t}(u(t-1) - u(t-3))$. Compute and plot the response of the system to the input signal

$$x(t) = \begin{cases} t, & 1 \leq t \leq 3 \\ 1, & 3 < t \leq 5 \end{cases}.$$

Solution

```
t1=0:.1:.9;
t2=1:.1:3;
t3=3.1:.1:5;
x1=zeros(size(t1));
x2=t2;
x3=ones(size(t3));
x=[x1 x2 x3];
h1=zeros(size(t1));
h2=exp(-t2);
h3=zeros(size(t3));
h=[h1 h2 h3];
y=conv(x,h)*0.1;
plot(0:.1:10,y);
legend('y(t)')
```



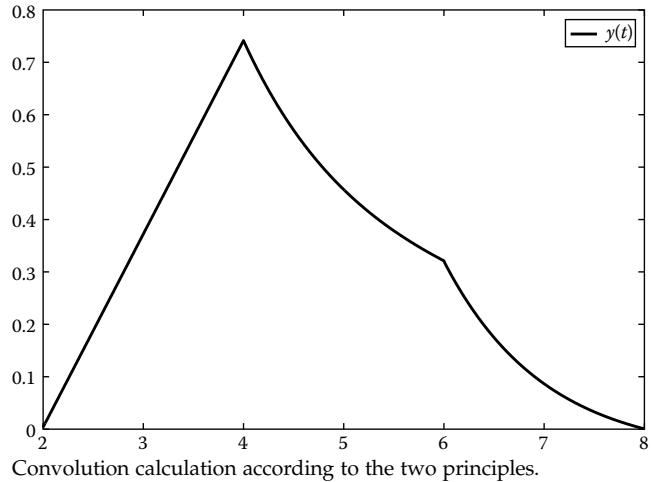
Convolution calculation according to the four rules.

Alternatively:

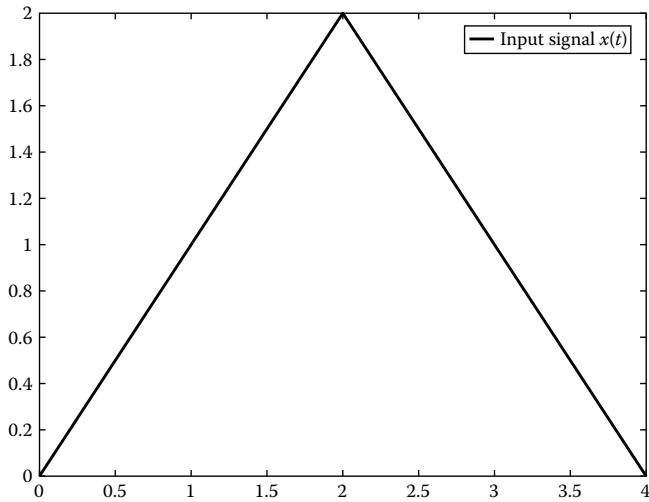
```

th=1:.01:3;
h=exp(-th);
tx1=1:.01:3;
x1=tx1;
tx2=3.01:.01:5;
x2=ones(size(tx2));
x=[x1 x2];
y=conv(x,h)*0.01;
plot(2:.01:8,y);
legend('y(t)')

```



Problem 6 Suppose that the impulse response of a system is $h(t) = te^{-t}u(t)$. Compute and plot the response of the system to the input signal that is depicted in the figure below.



Solution

First, the mathematical expression of $x(t)$ is derived. The first part of the input signal is $x_1(t) = t$. To derive the second part, one needs to compute the coefficients a and b of the straight line equation $x_2(t) = at + b$. The computation is made at the points $t = 2$ and $t = 4$.

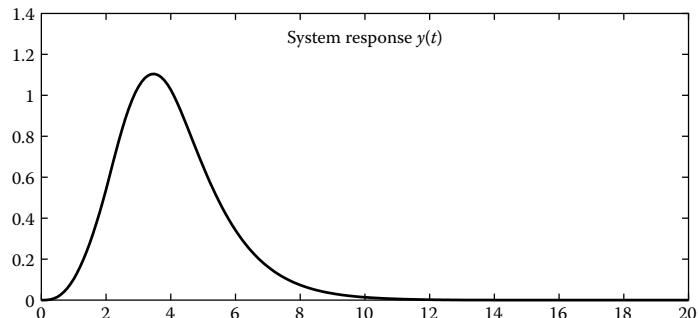
For $t=2$, we get $2=a \cdot 2+b \Rightarrow b=2-2a$. For $t=4$, we get $0=a \cdot 4+b \Rightarrow b=-4a$. Hence, $a=-1$ and $b=4$. Therefore, the input signal $x(t)$ is given by

$$x(t) = \begin{cases} t, & 0 \leq t \leq 2 \\ 4-t, & 2 < t \leq 4 \end{cases}.$$

The response of the system is computed as usual.

Computation and graph of the output signal.

```
t1=0:.1:2;
t2=2.1:.1:4;
t3=4.1:.1:10;
x1=t1;
x2=4-t2;
x3=zeros(size(t3));
x=[x1 x2 x3];
t=0:.1:10;
h=t.*exp(-t);
y=conv(x,h)*0.1;
plot(0:.1:20,y);
title('System response y(t)')
```



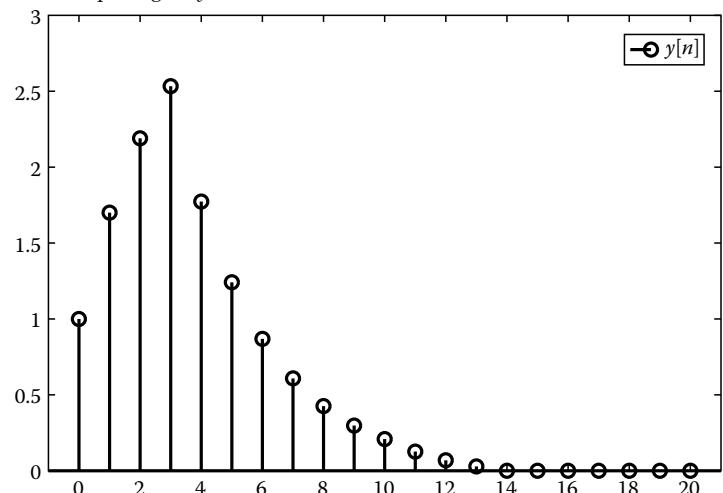
Problem 7 Calculate the convolution between the discrete-time signals $h[n]=0.7^n$, $0 \leq n \leq 10$ and $x[n]=u[n]-u[n-4]$.

Solution

The signals $x[n]$ and $h[n]$ are first defined in the same time interval. Next, their convolution is computed with the help of the command conv.

The output signal $y[n]$.

```
n=0:10;
u=ones(size(n));
n1=0:3;
u4_1=zeros(size(n1));
n2=4:10;
u4_2=ones(size(n2));
u4=[u4_1 u4_2];
x=u-u4;
h=0.7.^n;
y=conv(x,h);
stem(0:20,y);
xlim([-1 21]);
legend('y[n]')
```



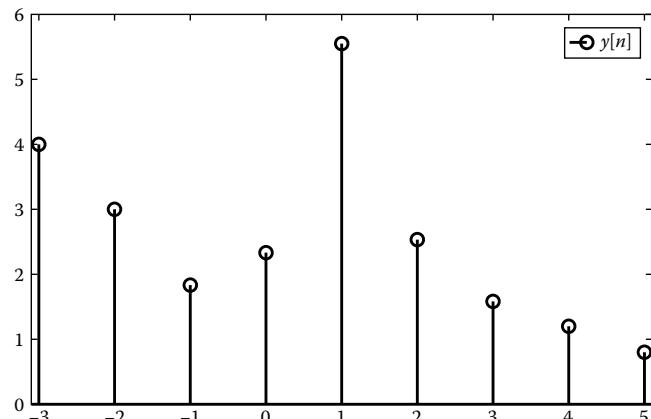
Problem 8 Write a function that computes and plots the convolution of two sequences. The function must accept as arguments the two signals and the time intervals in which the sequences are defined. Execute your function for $x[n] = n^2$, $-2 \leq n \leq 2$ and $h[n] = 1/(n+2)$, $-1 \leq n \leq 3$.

Solution

```
function [y,n] = convd(x,n1,h,n2)
a=n1(1)+n2(1);
b=n1(end)+n2(end);
n=a:b;
y=conv(x,h);
stem(n,y);
legend('y[n]')

```

```
n1=-2:2;
x=n1.^2;
n2=-1:3;
h=1./(n2+2);
[y,n] = convd(x,n1,h,n2)
```



The time interval where the output signal $y[n]$ lies is computed according to the two principles of discrete-time convolution.

Problem 9 Consider a system with zero initial conditions described by the difference equation $y[n] = -y[n-1] - 0.5y[n-2] + 0.2x[n] + 0.1x[n-1] + 0.1x[n-2]$.

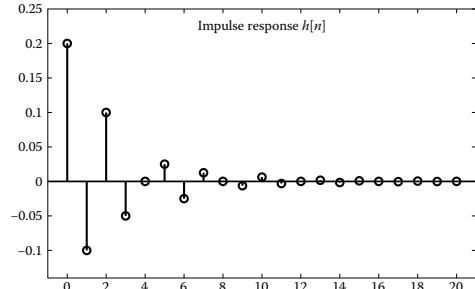
Compute and plot in the time interval $0 \leq n \leq 20$

- The impulse response of the system
- The step response of the system
- The response of the system to the input signal $x[n] = n \cdot 0.9^n$, $0 \leq n \leq 5$

Solution

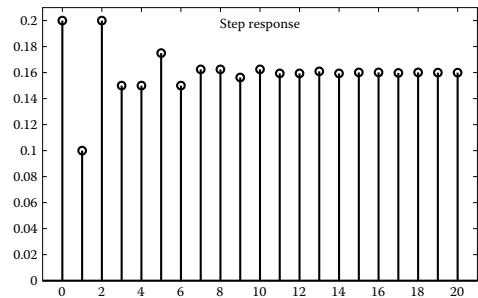
a.

```
a = [1 1 0.5];
b = [.2.1.1];
d = [1 zeros(1, 20)];
h = filter(b, a, d);
stem(0:20, h);
title('Impulse response h[n]')
xlim([-1 21])
```



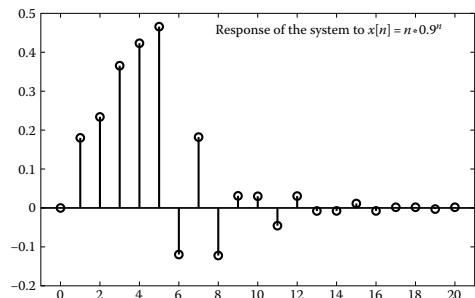
b.

```
a = [1 1 0.5];
b = [.2.1.1];
u = ones(1, 21);
s = filter(b, a, u);
stem(0:20, s);
title('Step response')
xlim([-1 21])
```



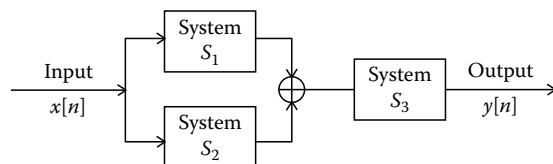
c.

```
a = [1 1 0.5];
b = [.2.1.1];
nx1 = 0:5;
x1 = nx1.* (0.9.^nx1);
nx2 = 6:20;
x2 = zeros(size(nx2));
x = [x1 x2];
y = filter(b, a, x);
stem(0:20, y);
title('Response of the system to x[n] = n*0.9^n')
xlim([-1 21])
```



Problem 10 For the system depicted in the figure below, the impulse responses of the discrete-time interconnected subsystems S_1 , S_2 , and S_3 are $h_1[n]=[2, 3, 4]$, $0 \leq n \leq 2$; $h_2[n]=[-1, 3, 1]$, $0 \leq n \leq 2$; and $h_3[n]=[1, 1, -1]$, $0 \leq n \leq 2$, respectively. Compute

- The impulse response $h[n]$ of the overall equivalent system
- The response of the overall system to the input signal $x[n] = u[n] - u[n - 2]$



Solution

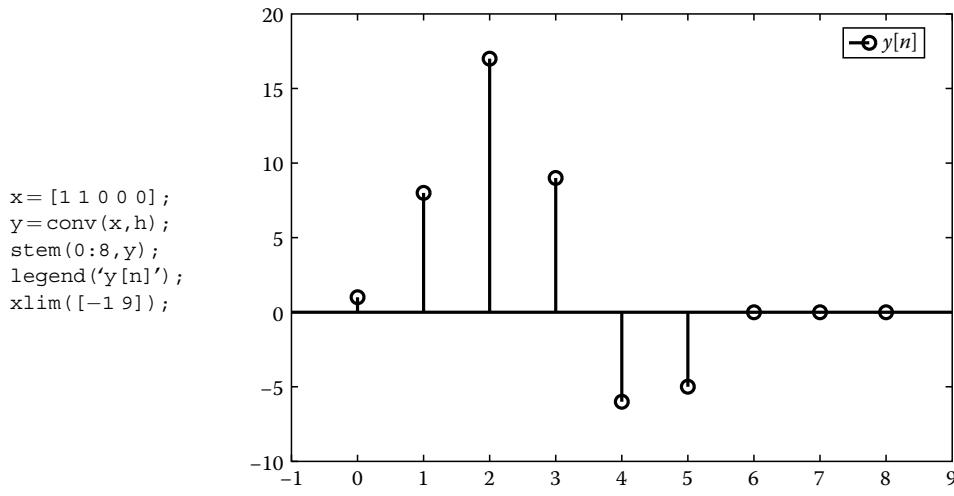
- a. Subsystems S_1 and S_2 are parallel connected. Hence, the impulse response of their interconnection is given by $h_{12} = h_1 + h_2$. The subsystem S_3 is cascade connected with them; thus the impulse response of the equivalent system is given by $h = h_{12} * h_3$.

```

h1 = [2, 3, 4];
h2 = [-1, 3, 1];
h12 = h1+h2;
h3 = [1 1 -1];
h = conv(h12, h3)

```

- b. The output is obtained by convoluting the input signal with the impulse response of the equivalent system.

**4.13 Homework Problems**

1. A system is described by the impulse response $h(t) = u(t) - u(t - 3)$. Compute and plot the response of the system to the input signal $x(t) = 2[u(t) - u(t - 3)]$.
2. Compute and plot the convolution between the signals $x(t) = 3te^{-t}u(t - 1)$ and $h(t) = t \cos(t)[u(t + 1) - u(t - 4)]$.

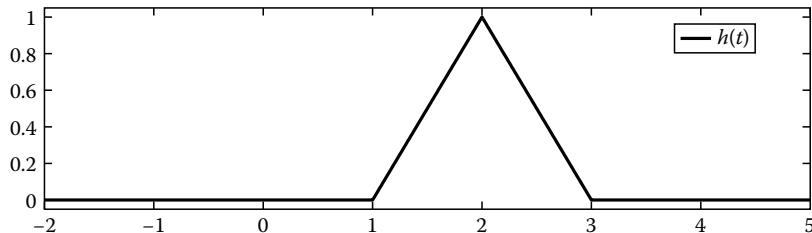
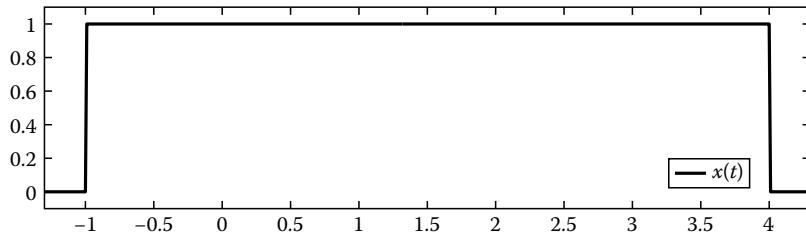
3. A system is described by the impulse response signal $h(t) = u(t) - u(t - 3)$.

Compute and plot the response of the system to the input signal

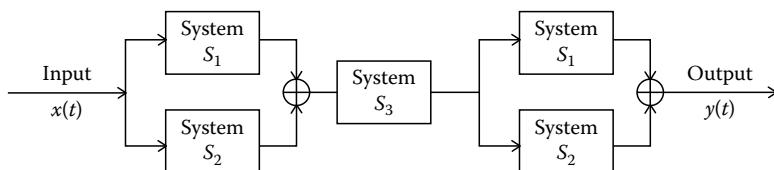
$$x(t) = \begin{cases} 1, & -2 \leq t \leq 0 \\ e^{-3t}, & 0 \leq t \leq 2 \end{cases}.$$

Moreover, plot the two signals during the various stages of their convolution.

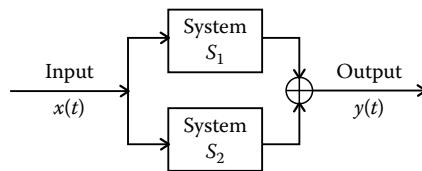
4. Compute the convolution between the signals that are depicted in the figure below.



5. The response of a system to the input signal $x(t) = \cos(t)[u(t) - u(t - 3)]$ is $y(t) = t[u(t) - u(t - 6)]$. Plot the impulse response of the system.
6. A system is described by the impulse response $h(t) = e^{-2t}u(t)$. Plot the input signal that is applied to the system if the output signal is $y(t) = te^{-t}u(t)$.
7. A system is described by the impulse response $h(t) = e^{-2t}u(t)$. Plot the impulse response of the inverse system.
8. Suppose that the impulse responses of the subsystems S_1 , S_2 , and S_3 that are connected as shown in the figure below are $h_1(t) = t \cos(2\pi t)$, $0 \leq t \leq 4$; $h_2(t) = te^{-2t}$, $0 \leq t \leq 4$; and $h_3(t) = u(t) - u(t - 5)$. Compute and plot in the appropriate time interval the impulse response of the overall system and the response of the overall system to the input signal $x(t) = te^{-2t}[u(t) - u(t - 2)]$.



9. Suppose that the impulse responses of the subsystems S_1 and S_2 that are connected as shown in the figure below are $h_1(t) = e^{-3t}u(t)$ and $h_2(t) = te^{-2t}u(t)$. Determine if the overall system is BIBO stable.



10. Compute the convolution between the complex sequences $x = [3 + 2j, 1 + j, 4 + 6j]$ and $h = [1 - 2j, j, 3 - 2j, 2]$.
11. Suppose that a discrete-time system is described by the impulse response $h[n] = 0.8^n$, $0 \leq n \leq 10$. Compute and plot the response of the system to the input signal $x[n] = \sqrt{1/n}$, $1 \leq n \leq 5$.
12. Compute the impulse response and the step response of the system described by the difference equation $y[n] = 0.8y[n - 2] + x[n] - 0.5x[n - 1]$ with zero initial conditions.
13. Consider the system described by the difference equation $y[n] = 0.8y[n - 1] + x[n] - 0.5x[n - 2]$ and assume zero initial conditions.
Compute and plot in the time interval $0 \leq n \leq 100$
- The response of the system $y[n]$ to the input signal $x[n] = 0.9^n(u[n] - u[n - 100])$
 - The impulse response $h[n]$ of the system
14. Consider the system described by the difference equation $y[n] = 0.8y[n - 1] + x[n] - 0.5x[n - 2]$ and assume that $y[-1] = 3$, $y[-2] = 4$, and $x[-1] = 1$. Compute and plot in the time interval $0 \leq n \leq 100$ the response of the system $y[n]$ to the input signal $x[n] = 0.9^n(u[n] - u[n - 100])$. Compare the output signal to the one obtained in the previous problem.
15. Suppose that a system is described by the second-order differential equation $y''(t) + 2y'(t) + 2y(t) = x(t)$. Suppose also that the system has zero initial conditions; that is, $y(0) = y'(0) = 0$. Compute the response of the system to the input signal $x(t) = e^{-t}u(t)$.
16. The impulse response of the system described by the differential equation $y''(t) + 2y'(t) + 2y(t) = x(t)$, $y(0) = y'(0) = 0$ is $h(t) = e^{-t}\sin(t)u(t)$. Verify for $0 \leq t \leq 10$ the output signal that was computed in the previous exercise.
17. The impulse response of a system S_1 is $h_1[n] = 0.9^n u[n]$. A system S_2 is described by the difference equation $y[n] = 0.1x[n] - 0.5x[n - 1] + x[n - 2]$ with zero initial conditions. Find out if the systems S_1 and S_2 are BIBO stable.

This page intentionally left blank

5

Fourier Series

In this chapter, we introduce a way of analyzing/decomposing a continuous-time signal into frequency components given by sinusoidal signals. This process is crucial in the signal processing field since it reveals the frequency content of a signal and simplifies the calculation of a system's output. This analysis is based on the use of *Fourier series*. Up to this point, all signals were expressed in the time domain. With the use of the Fourier series, a signal is expressed in the frequency domain and sometimes a frequency representation of a signal reveals more information about the signal than its time domain representation. There are three different and equivalent ways that can be used in order to express a signal into a sum of simple oscillating functions, i.e., into a sum of sines, cosines, or complex exponentials. In this chapter, the symbols n and k are often swapped in order for the code written in the examples to be in accordance with the theoretical mathematical equations.

5.1 Orthogonality of Complex Exponential Signals

Suppose that $x_m(t)$ and $x_k(t)$ are two complex-valued continuous-time periodic signals with period T . These two signals are orthogonal if their inner product is zero. The inner product of $x_m(t)$ and $x_k(t)$ is given by

$$I = \int_{t_0}^{t_0+T} x_m(t)x_k^*(t)dt, \quad (5.1)$$

where $x_k^*(t)$ is the complex conjugate of $x_k(t)$. If $I=0$ for $m \neq k$, the signals $x_m(t)$ and $x_k(t)$ are orthogonal. Suppose now that $x_m(t) = e^{jm\Omega_0 t}$, $x_k(t) = e^{jk\Omega_0 t}$, and $k, m \in \mathbb{Z}$. These two signals are orthogonal as

$$I = \int_0^T e^{jm\Omega_0 t} e^{-jk\Omega_0 t} dt = \int_0^T e^{j(m-k)\Omega_0 t} dt = \begin{cases} T, & k = m \\ 0, & k \neq m \end{cases}. \quad (5.2)$$

In order to verify the orthogonality of complex exponential signals, consider the signals $x(t) = e^{j3(2\pi/T)t}$ and $y(t) = e^{j5(2\pi/T)t}$. In this case, the complex conjugate of $y(t)$ is $y^*(t) = e^{-j5(2\pi/T)t}$.

Commands	Results	Comments
<pre> syms t T w = 2*pi/T x = exp(j*3*w*t); yc = exp(-j*5*w*t); I = int(x*yc,t,0,T) yc = exp(-j*3*w*t); I = int(x*yc,t,0,T) </pre>	<pre>I = 0</pre>	<p>The signals $x(t) = e^{j3(2\pi/T)t}$ and $y(t) = e^{-j5(2\pi/T)t}$ are indeed orthogonal as their inner product is zero.</p>

5.2 Complex Exponential Fourier Series

Suppose that a signal $x(t)$ is defined in the time interval $[t_0, t_0 + T]$. Then, $x(t)$ is expressed in exponential Fourier series form as

$$x(t) = \sum_{k=-\infty}^{\infty} a_k e^{jk\Omega_0 t}, \quad t \in [t_0, t_0 + T], \quad (5.3)$$

where

Ω_0 is the fundamental frequency, and is given by $\Omega_0 = (2\pi/T)$
 t_0, T are real numbers

The terms a_k that appear in Equation 5.3 are given by

$$a_k = \frac{1}{T} \int_{t_0}^{t_0+T} x(t) e^{-jk\Omega_0 t} dt, \quad (5.4)$$

where the orthogonality property of complex exponential signals was taken into account to derive a_k . The complex coefficients a_k are called complex exponential Fourier series coefficients, while a_0 is real number and is called the constant or dc component. Each coefficient a_k corresponds to the projection of the signal $x(t)$ on the k th orthogonal component $e^{jk\Omega_0 t}$, and indicates the spectrum content of the signal $x(t)$ at the frequency $k\Omega_0$, which is known as the k th harmonic. The Fourier series expansion is valid only in the interval $[t_0, t_0 + T]$, and the value of T defines the fundamental frequency Ω_0 . As the Fourier series coefficients represent the signal in the frequency domain, we refer to them as the spectral coefficients of the signal.

Example

Expand in complex exponential Fourier series the signal $x(t) = e^{-t}$, $0 \leq t \leq 3$.

The first thing that needs to be done is to define the quantities $t_0 = 0$, $T = 3$, and $\Omega_0 = (2\pi/T)$. Moreover, the signal $x(t)$ is defined as symbolic expression.

Commands	Results	Comments
<pre>t0 = 0; T = 3; w = 2*pi/T; syms t x = exp(-t); ezplot(x, [t0 t0+T]);</pre>	<p>The graph shows a smooth, monotonically decreasing curve starting at (0, 1) and approaching the x-axis as t increases towards 3. The plot is titled "exp(-t)".</p>	<p>Definition and graph of the signal $x(t)$ in the time interval $[t_0, t_0 + T]$.</p>

Afterward, the coefficients a_k are computed according to Equation 5.4. Looking into Equation 5.3, we observe that infinite Fourier coefficients a_k have to be calculated. Of course, this computation cannot be done in an analytical way. Fortunately, as the index k approaches toward $+\infty$ or toward $-\infty$, the Fourier coefficients a_k are approaching zero. Thus, $x(t)$ can be satisfactorily approximated by using a finite number of complex exponential Fourier series terms. Consequently, by computing the coefficients a_k for $-100 \leq k \leq 100$, i.e., by using the first 201 complex exponential terms, a good approximation of $x(t)$ is expected. The approximate signal is denoted by $xx(t)$, and is computed by

$$xx(t) = \sum_{k=-K}^{K} a_k e^{jk\Omega_0 t}, \quad t \in [t_0, t_0 + T]. \quad (5.5)$$

Commands	Comments
<pre>for k = -100:100 a(k+101) = (1/T) * int(x*exp(-j*k*w*t), t, t0, t0+T); end</pre>	Calculation of the coefficients a_k according to Equation 5.4.

In order to define the vector a that contains the Fourier series coefficients a_k , $-100 \leq k \leq 100$, the syntax $a(k+101)$ is used for programming reasons, since in MATLAB® the index of a vector cannot be zero or negative. Having calculated the coefficients a_k , the signal $x(t)$ is approximated according to Equation 5.3, or more precisely according to Equation 5.5. Note that Equation 5.3 is sometimes called *synthesis* equation, while we refer to Equation 5.4 as the *analysis* equation.

Commands	Results/Comments
<pre> for k = -100:100 ex(k+101) = exp(j*k*w*t); end xx = sum(a.*ex); ezplot(xx, [t0 t0+T]); title('Approximation with 201 terms') </pre>	<p>Initially, the quantities $e^{jk\Omega_0 t}$, $-100 \leq k \leq 100$ are computed, and afterward the signal is approximated according to Equation 5.5.</p>

The plotted signal $xx(t)$ that is computed with the use of the complex exponential Fourier series is almost identical with the original signal $x(t)$. In order to understand the importance of the number of terms used for the approximation of the original signal $x(t)$, the approximate signal $xx(t)$ is constructed for different values of k . First, the signal $x(t)$ is approximated by three exponential terms, i.e., the coefficients a_k are computed for $-1 \leq k \leq 1$.

Commands	Results/Comments
<pre> clear a ex; for k = -1:1 a(k+2) = (1/T)*int(x*exp(-j*k*w*t), t, t0, t0+T); end for k = -1:1 ex(k+2) = exp(j*k*w*t); end xx = sum(a.*ex); ezplot(xx, [t0 t0+T]); title('Approximation with 3 terms') </pre>	<p>When three terms are used for the approximation of $x(t)$ by $xx(t)$, i.e., $-1 \leq k \leq 1$, the approximate signal $xx(t)$ is pretty dissimilar from the original signal $x(t)$.</p>

Next, the signal $x(t)$ is approximated by 11 exponential terms, i.e., the coefficients a_k are computed for $-5 \leq k \leq 5$.

Commands	Results/Comments
<pre> for k=-5:5 a(k+6) = (1/T) * int (x*exp(-j*k*w*t), t, t0, t0+T); end for k=-5:5 ex(k+6) = exp(j*k*w*t); end xx = sum(a.*ex); ezplot(xx, [t0 t0+T]); title('Approximation with 11 terms') </pre>	<p>It is clear that even when 11 terms are used, namely, $-5 \leq k \leq 5$, the approximation of $x(t)$ by $xx(t)$ is not good as $xx(t)$ is pretty dissimilar from the original signal $x(t)$.</p>

Finally, the signal $x(t)$ is approximated by 41 exponential terms, i.e., the coefficients a_k are computed for $-20 \leq k \leq 20$.

Commands	Results/Comments
<pre> for k=-20:20 a(k+21) = (1/T) * int (x*exp(-j*k*w*t), t, t0, t0+T); end for k=-20:20 ex(k+21) = exp(j*k*w*t); end xx = sum(a.*ex); ezplot(xx, [t0 t0+T]); title('Approximation with 41 terms') </pre>	<p>The signal $xx(t)$ is now computed from 41 terms ($-20 \leq k \leq 20$) and is starting to look quite similar to the original signal $x(t)$. Thus, this is a quite satisfactory approximation.</p>

From the above analysis, it is clear that when many exponential terms are being considered in the construction of the approximate signal, a better approximation of the original signal is obtained. As illustrated in the beginning of the example, when 201 terms were used for the construction of the approximate signal, the obtained approximation was very good.

5.3 Trigonometric Fourier Series

In this section, we introduce a second form of Fourier series. Suppose that a signal $x(t)$ is defined in the time interval $[t_0, t_0 + T]$. Then $x(t)$, by using the trigonometric Fourier series, can be expressed in the time interval $[t_0, t_0 + T]$ as a sum of sinusoidal signals, namely, sines and cosines, where each signal has frequency $k\Omega_0$ rad/s.

The mathematical expression is

$$x(t) = a_0 + \sum_{k=1}^{\infty} b_k \cos(k\Omega_0 t) + \sum_{k=1}^{\infty} c_k \sin(k\Omega_0 t). \quad (5.6)$$

The coefficients $a_0, b_1, b_2, \dots, c_1, c_2, \dots$ of the trigonometric Fourier series are computed by

$$a_0 = \frac{1}{T} \int_{t_0}^{t_0+T} x(t) dt. \quad (5.7)$$

$$b_n = \frac{2}{T} \int_{t_0}^{t_0+T} x(t) \cos(n\Omega_0 t) dt, \quad n = 1, 2, \dots \quad (5.8)$$

$$c_n = \frac{2}{T} \int_{t_0}^{t_0+T} x(t) \sin(n\Omega_0 t) dt, \quad n = 1, 2, \dots \quad (5.9)$$

Example

The signal that will be expanded is the same signal used at the previous example. Thus, the problem is to expand in trigonometric Fourier series the signal $x(t) = e^{-t}$, $0 \leq t \leq 3$.

First, the trigonometric Fourier coefficients b_n, c_n and the dc component a_0 are computed according to Equations 5.8, 5.9, and 5.7, respectively, for $n = 1, 2, \dots, 200$. Next, $x(t)$ is approximated according to the relationship

$$x(t) = a_0 + \sum_{k=1}^N b_k \cos(k\Omega_0 t) + \sum_{k=1}^N c_k \sin(k\Omega_0 t). \quad (5.10)$$

Commands	Results/Comments
<pre>T = 3; t0 = 0; w = 2*pi/T; syms t x = exp(-t);</pre>	Definition of $t_0 = 0$, $T = 3$, $\Omega_0 = (2\pi/T)$ and of the signal $x(t)$.
<pre>a0 = (1/T)*int (x,t,t0,t0+T); for n=1:200 b(n) = (2/T)*int (x*cos(n*w*t),t,t0,t0+T); end for n=1:200 c(n) = (2/T)*int (x*sin(n*w*t),t,t0,t0+T); end</pre>	Computation of the trigonometric coefficients according to Equations 5.7 through 5.9.
<pre>k = 1:200; xx = a0+sum(b.*cos(k*w*t))+sum(c.*sin(k*w*t))</pre>	The signal $x(t)$ is approximated according to Equation 5.6, or more precisely according to Equation 5.10 with $N = 200$.

(continued)

Commands	Results/Comments
<pre>ezplot(xx, [t0 t0+T]); title('Approximation with 201 terms')</pre>	<p>Graph of the approximate signal $xx(t)$ that was computed by the terms of the trigonometric Fourier series.</p>

The approximation with 201 terms (200 trigonometric Fourier series terms + the dc component) of the original signal $x(t)$ is very good.

In order to understand the importance of the number of terms used for the approximation of the original signal $x(t)$, the approximate signal $xx(t)$ is constructed for different values of n .

Approximation with 6 terms ($n = 1, \dots, 5$).

Commands	Results/Comments
<pre>clear b c for n=1:5 b(n) = (2/T)*int(x*cos(n*w*t), t, t0, t0+T); c(n) = (2/T)*int(x*sin(n*w*t), t, t0, t0+T); end k=1:5; xx=a0+sum(b.*cos(k*w*t))+sum(c.*sin(k*w*t)) ezplot(xx, [t0 t0+T]); title('Approximation with 6 terms')</pre>	<p>When the signal $xx(t)$ approximates $x(t)$ with 6 terms ($n = 1, \dots, 5$), it is pretty dissimilar from the original signal $x(t)$.</p>

Approximation with 21 terms ($n = 1, \dots, 20$).

Commands	Results/Comments
<pre> for n=1:20 b(n) = (2/T)*int(x*cos(n*w*t),t,t0,t0+T); c(n) = (2/T)*int(x*sin(n*w*t),t,t0,t0+T); end k=1:20; xx=a0+sum(b.*cos(k*w*t))+sum(c.*sin(k*w*t)); ezplot(xx,[t0 t0+T]); title('Approximation with 21 terms') </pre>	<p>The signal $xx(t)$ is now computed by 21 terms ($1 \leq n \leq 20$) and is quite similar to the original signal $x(t)$. This is a quite satisfactory approximation.</p>

5.4 Fourier Series in the Cosine with Phase Form

A third form of Fourier series is derived by using the trigonometric property

$$b_k \cos(k\Omega_0 t) + c_k \sin(k\Omega_0 t) = A_k \cos(k\Omega_0 t + \theta_k). \quad (5.11)$$

Here, the signal $x(t)$ (which is defined in the time interval $[t_0, t_0 + T]$) is expressed as

$$x(t) = a_0 + \sum_{k=1}^{\infty} A_k \cos(k\Omega_0 t + \theta_k). \quad (5.12)$$

Thus, the signal $x(t)$ is expanded in a sum of sinusoids with different amplitudes and phases. The coefficients a_0, A_k, θ_k are given by

$$a_0 = \frac{1}{T} \int_{t_0}^{t_0+T} x(t) dt. \quad (5.13)$$

$$A_k = \sqrt{b_k^2 + c_k^2}, \quad k = 1, 2, \dots \quad (5.14)$$

$$\theta_k = \begin{cases} \tan^{-1}\left(-\frac{c_k}{b_k}\right) = -\tan^{-1}\left(\frac{c_k}{b_k}\right), & \text{dk} = 1, 2, \dots, \text{when } b_k \geq 0 \\ \pi + \tan^{-1}\left(-\frac{c_k}{b_k}\right), & k = 1, 2, \dots, \text{when } b_k < 0 \end{cases}. \quad (5.15)$$

Due to the two-part definition of the phases θ_k , it is more suitable (and secure) to use the atan2 command instead of atan, when computing the phase angle.

Example

Expand the signal $x(t) = e^{-t^2}$, $-2 \leq t \leq 2$ in the Fourier series in the cosine with phase form.

First, we plot the signal $x(t)$ in order to use it as a reference for the approximate signal that will be constructed according to Equation 5.12. Afterward, the coefficients a_0, b_n, c_n of the trigonometric Fourier series are calculated and the approximate signal is plotted, using 101 trigonometric Fourier series terms.

Commands	Results/Comments
<pre>T = 4; t0 = -2; w = 2*pi/T; syms t n x = exp(-t^2); ezplot(x, [t0 t0+T]); legend('x(t)')</pre>	<p>Definition of $t_0 = 0$, $T = 3$, $\Omega_0 = 2\pi/T$ and graph of the signal $x(t)$.</p>
<pre>a0 = (1/T)*int(x,t,t0,t0+T); b = (2/T)*int(x*cos(n*w*t),t,t0,t0+T); c = (2/T)*int(x*sin(n*w*t),t,t0,t0+T); n1 = 1:100; bn = subs(b,n,n1); cn = subs(c,n,n1); xx = a0 + sum(bn.*cos(n1*w*t)) + sum(cn.*sin(n1*w*t)); ezplot(xx, [-2 2]); title('Approximation with 101 terms');</pre>	<p>Calculation of the trigonometric coefficients according to Equations 5.7 through 5.9, approximation of $x(t)$ according to (5.6), and graph of the approximate signal. Notice that in this implementation, the use of for-loop is avoided. This is the optimal programming practice in MATLAB.</p>

Having computed the coefficients a_0, b_n, c_n of the trigonometric form, the amplitudes A_k and phases θ_k of the cosine with phase form can be easily calculated. Afterward, the signal is approximated according to Equation 5.12.

Commands	Results	Comments
<pre>k=1:100; A=sqrt(bn.^2+cn.^2); thita=atan2(-cn,bn); xx=a0+sum(A.*cos(k*w*t+thita))</pre>		Calculation of the amplitudes and phases according to (5.14) and (5.15).
<pre>ezplot(xx, [-2 2]) title('Approximation with 101 terms')</pre>		Expansion of the signal according to (5.12). Approximation according to the Fourier Series in cosine with phase form.

The three graphs are identical; thus it is clear that the third form of Fourier series, namely, the cosine with phase form is equivalent with the two previous forms of Fourier series.

5.5 Plotting the Fourier Series Coefficients

In the previous sections, the Fourier coefficients of the three forms of Fourier series were computed. In this section, the way of plotting them is presented. Once again, the signal $x(t)=e^{-t}$, $0 \leq t \leq 3$ is considered. The coefficients a_k of the complex exponential form are the first that will be plotted for $-6 \leq k \leq 6$ and for $-40 \leq k \leq 40$. In the usual case, the coefficients a_k are complex numbers. As presented in Chapter 1, a complex number z can be expressed as $z=|z|e^{j\angle z}$, where $|z|$ is the magnitude and $\angle z$ is the angle of z . Therefore, in order to create the graph of the coefficients a_k of the complex exponential form, the magnitude and the angle of each coefficient have to be plotted.

Commands	Results/Comments
<pre> syms t k n x=exp(-t); t0=0; T=3; w=2*pi/T; a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T) k1=-6:6; ak=subs(a,k,k1); stem(k1,abs(ak)); legend(' a_k , k = -6:6') </pre>	<p>The signal $x(t) = e^{-t}$, $0 \leq t \leq 3$ is defined and the magnitude of the coefficients a_k is plotted for $-6 \leq k \leq 6$.</p>
<pre> stem(k1,angle(ak)); legend('\angle a_k, k = -6:6') </pre>	<p>The angles $\angle a_k$ are plotted for $-6 \leq k \leq 6$. It is worth noticing the way that the angle symbol is drawn through the legend command.</p>
<pre> k2=-40:40; ak2=subs(a,k,k2); stem(k2,abs(ak2)); legend(' a_k k = -40:40') </pre>	
<pre> stem(k2,angle(ak2)); legend('\angle a_k k = -40:40') </pre>	

Next, the dc component a_0 and the coefficients b_n, c_n of the trigonometric form are plotted for various values of n .

Commands	Results
<pre>a0 = (1/T)*int(x,t0,t0+T); stem(0,eval(a0)); legend('a_0')</pre>	
<pre>b = (2/T)*int(x*cos(n*w*t),t,t0,t0+T) n1 = 1:10; bn = subs(b,n,n1); stem(n1, bn); legend('b_n')</pre>	
<pre>n11=1:40; bn1=subs(b,n,n11); stem(n11, bn1); legend('b_n n=1:40')</pre>	
<pre>c = (2/T)*int(x*sin(n*w*t),t,t0,t0+T); n2 = 1:10; cn = subs(c,n,n2); stem(n2, cn); legend('c_n')</pre>	

(continued)

Commands	Results
<pre>n21=1:40; cn1=subs(c,n,n21); stem(n21,cn1); legend('c_n n = 1:40')</pre>	

Finally, the amplitudes A_n and the phases θ_n of the cosine with phase form are plotted for various values of n . The phases θ_n are computed using two alternative ways: using the atan and the atan2 commands. In this example, the outcome of these commands is the same but, as mentioned earlier, the atan2 is the preferred command.

Commands	Results
<pre>A=sqrt(b^2+c^2); n1=1:6; An=subs(A,n,n1); stem(n1,An); legend('A_n n = 1:6')</pre>	
<pre>n11=1:40; An1=subs(A,n,n11); stem(n11,An1); legend('A_n n = 1:40')</pre>	

(continued)

(continued)

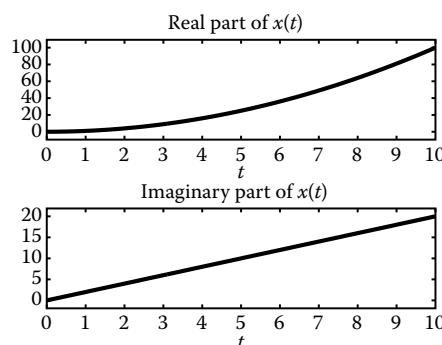
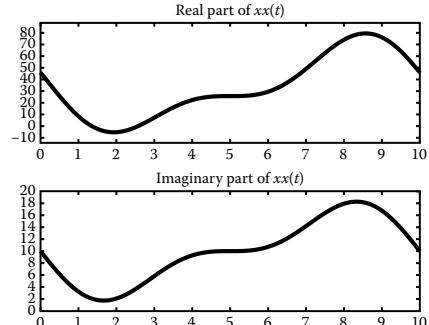
Commands	Results
<pre>thita=-atan(c/b); n1=1:6; thitan=subs(thita,n,n1); stem(n1,thitan); legend('\theta_n')</pre>	
<pre>thita2=atan2(-cn, bn); stem(n1, thita2(1:6)) legend ('\theta_n using atan2')</pre>	
<pre>n11=1:40; thitan1=subs(thita,n,n11); stem(n11,thitan1); legend ('\theta_n n = 1:40')</pre>	
<pre>thita21=atan2(-cn1, bn1); stem(n11, thita21(1:40)) legend ('\theta_n using atan2')</pre>	

5.6 Fourier Series of Complex Signals

So far, we were dealing with real signals. In this section, we examine the Fourier series representation of a complex-valued signal.

Example

Compute the coefficients of the complex exponential Fourier series and of the trigonometric Fourier series of the complex signal $x(t) = t^2 + j2\pi t$, $0 \leq t \leq 10$. Moreover, plot the approximate signals using 5 and 41 components of the complex exponential form and 5 and 21 components of the trigonometric form.

Commands	Results/Comments
<pre> syms t t0 = 0; T = 10; w = 2*pi/T; x = t^2 + j*2*pi*t; subplot(211); ezplot(real(x), [t0 T]); title ('Real part of x(t)'); subplot(212); ezplot(imag(x), [t0 T]); title ('Imaginary part of x(t)'); </pre>	<p>The signal $x(t) = t^2 + j2\pi t$ is complex; hence, its real and imaginary parts are plotted separately.</p> 
<pre> for k = -2:2 a(k+3) = (1/T) * int(x*exp(-j*k*w*t), t, t0, t0+T); ex(k+3) = exp(j*k*w*t); end xx = sum(a.*ex); subplot(211); ezplot(real(xx), [t0 T]); title ('Real part of xx(t)'); subplot(212); ezplot(imag(xx), [t0 T]); title ('Imaginary part of xx(t)'); </pre>	<p>Computation of the first five coefficients a_k of the complex exponential form and graph of the approximate signal $xx(t)$ obtained with five components.</p> 

(continued)

(continued)

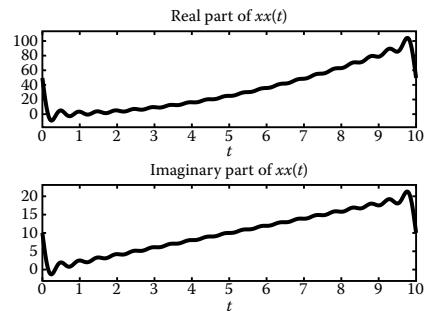
Commands**Results/Comments**

```

for k=-20:20
a(k+21) = (1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T);
ex(k+21) = exp(j*k*w*t);
end
xx = sum(a.*ex);
subplot(211);
ezplot(real(xx),[t0 T]);
title('Real part of xx(t)');
subplot(212);
ezplot(imag(xx),[t0 T]);
title('Imaginary part of xx(t)');

```

Computation of the first 41 coefficients a_k of the complex exponential form and graph of the approximate signal $xx(t)$ obtained with 41 components.

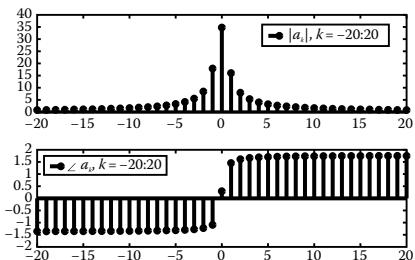


```

a1 = eval(a)
subplot(211);
stem(-20:20,abs(a1));
legend('|a_k|, k = -20:20');
subplot(212);
stem(-20:20,angle(a1));
legend('\angle a_k, k = -20:20')

```

The coefficients a_k of the complex exponential form are complex; thus their magnitude and phase are plotted.

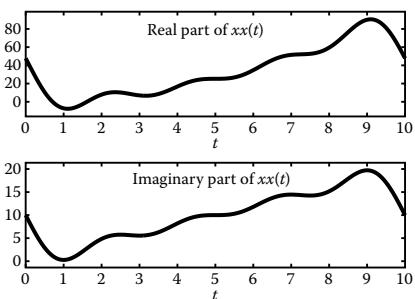


```

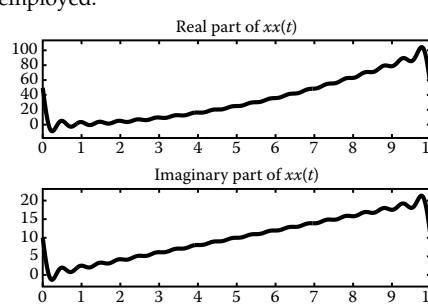
a0 = (1/T)*int(x,t,t0,t0+T);
for n=1:4
b(n) = (2/T)*int(x*cos(n*w*t),t,t0,t0+T);
c(n) = (2/T)*int(x*sin(n*w*t),t,t0,t0+T);
end
k=1:4;
xx=a0+sum(b.*cos(k*w*t))+sum(c.*sin(k*w*t))
subplot(211);
ezplot(real(xx),[t0 T]);
title('Real part of xx(t)');
subplot(212);
ezplot(imag(xx),[t0 T]);
title('Imaginary part of xx(t)');

```

Computation of the first five coefficients of the trigonometric form (four trigonometric Fourier series terms + the dc component) and graph of the corresponding approximate signal $xx(t)$.



(continued)

Commands	Results/Comments
<pre>a0 = 100/3+10*i b = [100/pi^2, 25/pi^2, 100/9/pi^2, 25/4/pi^2] c = [(-100-20*i)/pi, (-50-10*i)/pi, (-100/3-20/3*i)/pi, (-25-5*i)/pi]</pre> <p> $a_0 = 100/3 + 10i$ $b = [100/\pi^2, 25/\pi^2, 100/9\pi^2, 25/4\pi^2]$ $c = [(-100 - 20i)/\pi, (-50 - 10i)/\pi, (-100/3 - 20/3i)/\pi, (-25 - 5i)/\pi]$ </p> <pre>a0 = (1/T)*int (x,t,t0,t0+T); for n=1:20 b(n) = (2/T)*int (x*cos(n*w*t),t,t0,t0+T); c(n) = (2/T)*int (x*sin(n*w*t),t,t0,t0+T); end k=1:20; xx=a0+sum(b.*cos(k*w*t))+sum(c.*sin(k*w*t)) subplot(211); ezplot(real(xx),[t0 T]); title('Real part of xx(t)'); subplot(212); ezplot(imag(xx),[t0 T]); title('Imaginary part of xx(t)');</pre>	<p>We observe that the coefficients of the trigonometric Fourier series form are complex numbers if the signal is complex.</p> <p>The approximation of the signal $x(t)$ (real and imaginary part) is quite good when 21 components of the trigonometric Fourier series form are employed.</p> 

The computation of the coefficients of the Fourier series in the cosine with phase form is left as a homework (see Sections 5.14 and 5.15).

5.7 Fourier Series of Periodic Signals

In the previous sections, the Fourier series expansion of a signal was defined in a closed time interval $[t_0, t_0 + T]$. Beyond this interval, the Fourier series expansion does not always converge to the original signal $x(t)$. In this section, we introduce the case where the signal $x(t)$ is a periodic signal with period T , i.e., $x(t) = x(t + T)$. In this case, the Fourier series is also periodic with period T ; thus it converges to $x(t)$ for $-\infty < t < \infty$.

Example

Approximate, using the three forms of Fourier series, the periodic signal that in one period is given by

$$x(t) = \begin{cases} 1, & 0 \leq t \leq 1 \\ -1, & 1 \leq t \leq 2 \end{cases}.$$

First, the signal $x(t)$ is plotted over the time of five periods for reference reasons.

Commands	Results/Comments
<pre>t1=0:.01:1; t2=1.01:.01:2; x1=ones(size(t1)); x2=-ones(size(t2)); x=[x1 x2]; xp=repmat(x,1,5); t=linspace(0,10,length(xp)); plot(t,xp)</pre>	<p>The signal $x(t)$ in five periods.</p>

Afterward, the two-part signal $x(t)$ is defined as the single symbolic expression $x(t) = u(t) - 2u(t-1)$, $0 \leq t \leq 2$, where $u(t)$ is the unit step function. Note that a periodic signal $x(t)$ is entirely determined by its values over one period. Thus, the symbolic expression of $x(t)$ is defined only for the time interval of interest, namely, for $0 \leq t \leq 2$ (one period). The defined symbolic expression is plotted in one period for confirmation.

Commands	Results
<pre>syms t x=heaviside(t)-2*heaviside(t-1); ezplot(x,[0 2]);</pre>	

Finally, the complex exponential Fourier series coefficients a_k are calculated and the approximate signal $xx(t)$ is computed and plotted for $0 \leq t \leq 10$, i.e., for time of five periods. As in the previous examples, $xx(t)$ is computed and plotted for various numbers of exponential terms used.

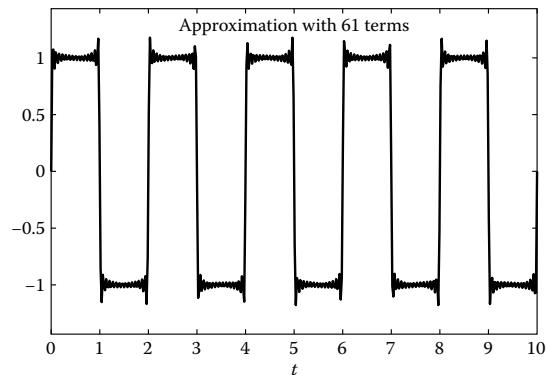
Commands	Results
<pre> k = -2:2; t0 = 0; T = 2; w = 2*pi/T; a = (1/T)*int(x*exp(-j*k*w*t), t, t0, t0+T) </pre>	$a = [0, 2*i/\pi, 0, -2*i/\pi, 0]$
<pre> xx = sum(a.*exp(j*k*w*t)); ezplot(xx, [0 10]) title('Approximation with 5 terms') </pre>	
<pre> k = -5:5; a = (1/T)*int(x*exp(-j*k*w*t), t, t0, t0+T); xx = sum(a.*exp(j*k*w*t)); ezplot(xx, [0 10]) title('Approximation with 11 terms') </pre>	
<pre> k = -10:10; a = (1/T)*int(x*exp(-j*k*w*t), t, t0, t0+T); xx = sum(a.*exp(j*k*w*t)); ezplot(xx, [0 10]) title('Approximation with 21 terms') </pre>	

(continued)

(continued)

Commands**Results**

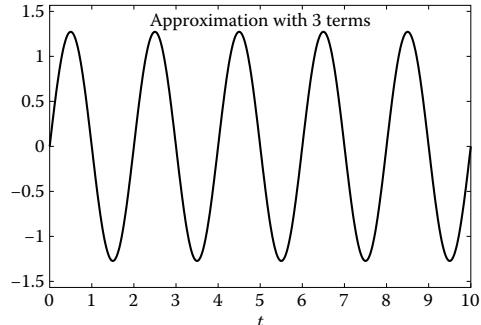
```
k = -30:30;
a = (1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T);
xx = sum(a.*exp(j*k*w*t));
ezplot(xx, [0 10])
title('Approximation with 61 terms')
```



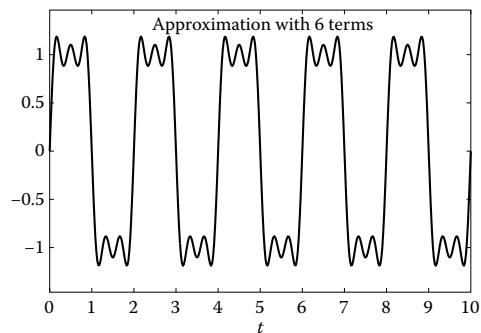
The same process is repeated for the trigonometric Fourier series coefficients a_0, b_k, c_k . The approximate signal is computed and plotted for various numbers of trigonometric terms used.

Commands**Results**

```
a0 = (1/T)*int(x,t0,t0+T);
n = 1:2;
b = (2/T)*int(x*cos(n*w*t),t,t0,t0+T);
c = (2/T)*int(x*sin(n*w*t),t,t0,t0+T);
xx = a0+sum(b.*cos(n*w*t))+sum(c.*sin(n*w*t));
ezplot(xx, [0 10])
title('Approximation with 3 terms')
```



```
n = 1:5;
b = (2/T)*int(x*cos(n*w*t),t,t0,t0+T);
c = (2/T)*int(x*sin(n*w*t),t,t0,t0+T);
xx = a0+sum(b.*cos(n*w*t))+sum(c.*sin(n*w*t));
ezplot(xx, [0 10])
title('Approximation with 6 terms')
```



(continued)

Commands	Results
<pre>n=1:40; b = (2/T)*int(x*cos(n*w*t),t,t0,t0+T); c = (2/T)*int(x*sin(n*w*t),t,t0,t0+T); xx=a0+sum(b.*cos(n*w*t))+sum(c.*sin(n*w*t)); ezplot(xx,[0 10]) title('Approximation with 41 terms')</pre>	

Finally, the approximation process according to the Fourier series in the cosine with phase form is implemented. The trigonometric Fourier series coefficients a_0 , b_k , c_k are already computed for $k=1:40$; thus the amplitudes A_k and the phases θ_k are easily derived.

Commands	Results
<pre>for n=1:2 A(n)=sqrt(b(n)^2+c(n)^2); thita(n)=atan2(-eval(c(n)),eval(b(n))); end k=1:2; xx=a0+sum(A.*cos(k*w*t+thita)) ezplot(xx,[0 10]) title('Approximation with 3 terms')</pre>	
<pre>for n=1:5 A(n)=sqrt(b(n)^2+c(n)^2); thita(n)=atan2(-eval(c(n)),eval(b(n))); end k=1:5; xx=a0+sum(A.*cos(k*w*t+thita)); ezplot(xx,[0 10]) title('Approximation with 6 terms')</pre>	

(continued)

(continued)

Commands	Results
<pre>for n=1:40 A(n)=sqrt(b(n)^2+c(n)^2); thita(n)=atan2(-eval(c(n)), eval(b(n))); end k=1:40; xx=a0+sum(A.*cos(k*w*t+thita)); ezplot(xx, [0 10]) title('Approximation with 41 terms')</pre>	

In all the above graphs, an oscillation at the discontinuity points of the signal is observed. This phenomenon is known as the *Gibbs phenomenon*. It is noted that the oscillation amplitude is independent from the number of approximation terms used. On the contrary, when the number of used terms tends to infinity, the time interval in which the oscillation appears tends to zero. Thus, the Gibbs phenomenon is eliminated when the number of terms used for the signal approximation is infinite.

5.8 Line Spectra

In Section 5.2, it was stated that the complex exponential Fourier series coefficients a_k specify the spectrum content of a signal $x(t)$ at the harmonic frequencies $k\Omega_0$, $k = \pm 1, \pm 2, \dots$. The graph of the magnitudes $|a_k|$ versus the frequency Ω at the harmonic frequencies $k\Omega_0$ is known as the magnitude spectrum, while the graph of the angles $\angle a_k$ (usually in degrees) at the harmonic frequencies $k\Omega_0$ is known as the phase spectrum. The magnitude spectrum and the phase spectrum constitute the *line spectra*.

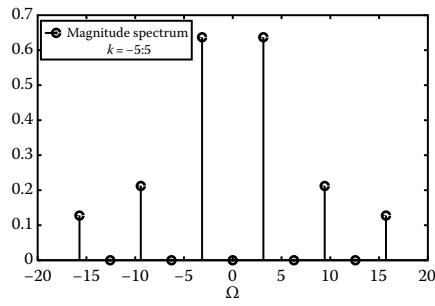
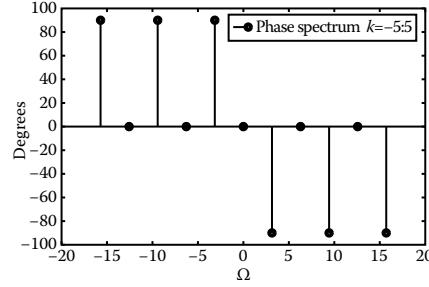
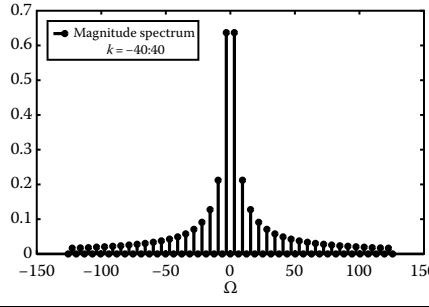
Example

Consider the periodic signal $x(t)$ that in one period is defined by

$$x(t) = \begin{cases} -1, & -1 \leq t \leq 0 \\ 1, & 0 \leq t \leq 1 \end{cases}.$$

Plot the line spectra of $x(t)$.

First, the signal is defined as a single symbolic expression. Next, the vector containing the harmonic frequencies is specified. Finally, the complex exponential Fourier series coefficients a_k are computed and the line spectra are plotted for $-5 \leq k \leq 5$ and $-40 \leq k \leq 40$.

Commands	Results/Comments																								
<pre> syms t k n x=heaviside(t)-2*heaviside(t-1); t0=0; T=2; w=2*pi/T; k=-5:5; a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T); wk=w*k </pre>	<p>First, the signal $x(t)$ is defined by $x(t) = u(t) - 2u(t-1)$.</p>																								
<pre> stem(wk,abs(eval(a))) legend('Magnitude spectrum k = -5:5') xlabel('\Omega') </pre>	<p>The fundamental frequency Ω_0 is given by $\Omega_0 = 2\pi/T = \pi$. Thus, the vector containing the harmonic frequencies is given by $k\Omega_0$, $-5 \leq k \leq 5$; that is, is given by $[-5\pi, -4\pi, \dots, 5\pi]$. Indeed,</p> <table border="0"> <tr> <td>$wk =$</td> <td>-15.7080</td> <td>-12.5664</td> <td>-9.4248</td> <td>-6.2832</td> </tr> <tr> <td></td> <td>-3.1416</td> <td>0</td> <td>3.1416</td> <td>6.2832</td> </tr> <tr> <td></td> <td>12.5664</td> <td>15.7080</td> <td></td> <td>9.4248</td> </tr> </table>	$wk =$	-15.7080	-12.5664	-9.4248	-6.2832		-3.1416	0	3.1416	6.2832		12.5664	15.7080		9.4248									
$wk =$	-15.7080	-12.5664	-9.4248	-6.2832																					
	-3.1416	0	3.1416	6.2832																					
	12.5664	15.7080		9.4248																					
	<p>The magnitudes a_k are plotted versus the harmonic frequencies $k\Omega_0$.</p>  <table border="1"> <caption>Magnitude spectrum data</caption> <thead> <tr> <th>Harmonic Frequency ($k\Omega_0$)</th> <th>Magnitude (a_k)</th> </tr> </thead> <tbody> <tr><td>-15.7080</td><td>~0.12</td></tr> <tr><td>-12.5664</td><td>~0.22</td></tr> <tr><td>-9.4248</td><td>~0.02</td></tr> <tr><td>-6.2832</td><td>~0.02</td></tr> <tr><td>-3.1416</td><td>~0.02</td></tr> <tr><td>0</td><td>~0.02</td></tr> <tr><td>3.1416</td><td>~0.02</td></tr> <tr><td>6.2832</td><td>~0.22</td></tr> <tr><td>9.4248</td><td>~0.02</td></tr> <tr><td>12.5664</td><td>~0.12</td></tr> <tr><td>15.7080</td><td>~0.02</td></tr> </tbody> </table>	Harmonic Frequency ($k\Omega_0$)	Magnitude ($ a_k $)	-15.7080	~0.12	-12.5664	~0.22	-9.4248	~0.02	-6.2832	~0.02	-3.1416	~0.02	0	~0.02	3.1416	~0.02	6.2832	~0.22	9.4248	~0.02	12.5664	~0.12	15.7080	~0.02
Harmonic Frequency ($k\Omega_0$)	Magnitude ($ a_k $)																								
-15.7080	~0.12																								
-12.5664	~0.22																								
-9.4248	~0.02																								
-6.2832	~0.02																								
-3.1416	~0.02																								
0	~0.02																								
3.1416	~0.02																								
6.2832	~0.22																								
9.4248	~0.02																								
12.5664	~0.12																								
15.7080	~0.02																								
<pre> stem(wk,angle(eval(a))*180/pi) legend('Phase spectrum k = -5:5') xlabel('\Omega') ylabel('Degrees') </pre>	<p>The angles $\angle a_k$ are converted from radians to degrees and plotted versus the harmonic frequencies $k\Omega_0$.</p>  <table border="1"> <caption>Phase spectrum data (in Degrees)</caption> <thead> <tr> <th>Harmonic Frequency ($k\Omega_0$)</th> <th>Angle ($\angle a_k$)</th> </tr> </thead> <tbody> <tr><td>-15.7080</td><td>~85</td></tr> <tr><td>-12.5664</td><td>~85</td></tr> <tr><td>-9.4248</td><td>~0</td></tr> <tr><td>-6.2832</td><td>~0</td></tr> <tr><td>-3.1416</td><td>~0</td></tr> <tr><td>0</td><td>~0</td></tr> <tr><td>3.1416</td><td>~0</td></tr> <tr><td>6.2832</td><td>~0</td></tr> <tr><td>9.4248</td><td>~-100</td></tr> <tr><td>12.5664</td><td>~-100</td></tr> <tr><td>15.7080</td><td>~-100</td></tr> </tbody> </table>	Harmonic Frequency ($k\Omega_0$)	Angle ($\angle a_k$)	-15.7080	~85	-12.5664	~85	-9.4248	~0	-6.2832	~0	-3.1416	~0	0	~0	3.1416	~0	6.2832	~0	9.4248	~-100	12.5664	~-100	15.7080	~-100
Harmonic Frequency ($k\Omega_0$)	Angle ($\angle a_k$)																								
-15.7080	~85																								
-12.5664	~85																								
-9.4248	~0																								
-6.2832	~0																								
-3.1416	~0																								
0	~0																								
3.1416	~0																								
6.2832	~0																								
9.4248	~-100																								
12.5664	~-100																								
15.7080	~-100																								
<pre> k=-40:40; a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T); wk=w*k stem(wk,abs(eval(a))) legend('Magnitude spectrum k = -40:40') xlabel('\Omega') </pre>	 <table border="1"> <caption>Magnitude spectrum data for k=-40 to 40</caption> <thead> <tr> <th>Harmonic Frequency ($k\Omega_0$)</th> <th>Magnitude (a_k)</th> </tr> </thead> <tbody> <tr><td>0</td><td>~0.65</td></tr> <tr><td>±10</td><td>~0.15</td></tr> <tr><td>±20</td><td>~0.05</td></tr> <tr><td>±30</td><td>~0.02</td></tr> <tr><td>±40</td><td>~0.01</td></tr> </tbody> </table>	Harmonic Frequency ($k\Omega_0$)	Magnitude ($ a_k $)	0	~0.65	±10	~0.15	±20	~0.05	±30	~0.02	±40	~0.01												
Harmonic Frequency ($k\Omega_0$)	Magnitude ($ a_k $)																								
0	~0.65																								
±10	~0.15																								
±20	~0.05																								
±30	~0.02																								
±40	~0.01																								

(continued)

(continued)

Commands	Results/Comments
<pre>stem(wk,angle(eval(a))*180/pi) legend('Phase spectrum k=-40:40') xlabel ('\Omega') ylabel ('Degrees')</pre>	

5.9 Properties of Fourier Series

The Fourier series coefficients derived through the analysis equation determine completely a periodic signal $x(t)$. Hence, one can say that the complex exponential coefficients a_k and the signal $x(t)$ are a Fourier series pair. This relationship is denoted by $x(t) \leftrightarrow a_k$. In this section, we present some properties of the Fourier series. The properties are verified through examples.

5.9.1 Linearity

Suppose that the complex exponential Fourier series coefficients of the periodic signals $x(t)$ and $y(t)$ are denoted by a_k and b_k , respectively, or in other words $x(t) \leftrightarrow a_k$ and $y(t) \leftrightarrow b_k$. Moreover, let z_1, z_2 denote two complex numbers. Then,

$$z_1x(t) + z_2y(t) \leftrightarrow z_1a_k + z_2b_k. \quad (5.16)$$

To verify the linearity property, we consider the periodic signals $x(t) = \cos(t)$, $y(t) = \sin(2t)$, and the scalars $z_1 = 3 + 2i$ and $z_2 = 2$.

Commands

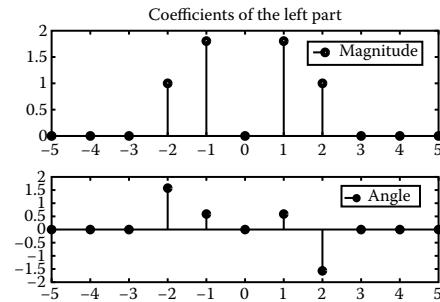
```
t0 = 0;
T = 2*pi;
w = 2*pi/T;
syms t
z1 = 3+2i; z2 = 2;
x = cos(t); y = sin(2*t);
f = z1*x+z2*y;
k = -5:5;
left = (1/T)*int(f*exp(-j*k*w*t),t,t0,t0+T);
left = eval(left);
subplot(211);
stem(k,abs(left));
legend('Magnitude');
title('Coefficients of the left part');
subplot(212);
stem(k,angle(left));
legend('Angle');
```



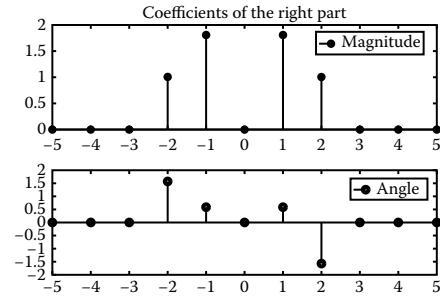
```
a = (1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T);
b = (1/T)*int(y*exp(-j*k*w*t),t,t0,t0+T);
right = z1*a + z2*b;
subplot(211);
right = eval(right);
stem(k,abs(right));
legend('Magnitude');
title('Coefficients of the right part');
subplot(212);
stem(k,angle(right));
legend('Angle');
```

Results/Comments

First, we determine the (complex exponential) Fourier series coefficients of the left part; that is, we compute the coefficients of the signal $f(t) = z_1x(t) + z_2y(t)$. The period of $f(t)$ is $T=2\pi$.



In order to derive the right part of (5.16), first we compute the coefficients a_k and b_k , and then we formulate the right part of (5.16).



The two graphs are identical; thus the linearity property is verified.

5.9.2 Time Shifting

A shift in time of the periodic signal results on a phase change of the Fourier series coefficients. So, if $x(t) \leftrightarrow a_k$, the exact relationship is

$$x(t - t_1) \leftrightarrow e^{-jk\Omega_0 t_1} a_k. \quad (5.17)$$

In order to verify the time-shifting property, we consider the periodic signal that in one period is given by $x(t) = te^{-5t}$, $0 \leq t \leq 10$. Moreover we set $t_1 = 3$. Consequently, the signal $x(t - t_1)$ is given by $x(t - t_1) = x(t - 3) = (t - 3)e^{-5(t-3)}$.

Commands

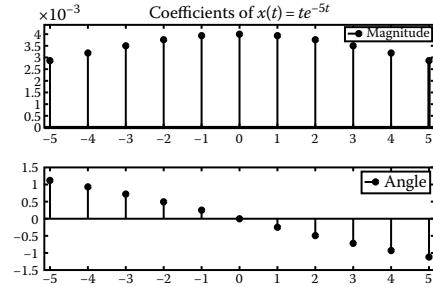
```
t0 = 0;
T = 10;
w = 2*pi/T;
syms t
x = t*exp(-5*t)
k = -5:5;
a = (1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T);
a1 = eval(a);
subplot(211);
stem(k,abs(a1));
title('Coefficients of x(t) = te^-5^t');
legend('Magnitude');
subplot(212);
stem(k,angle(a1));
legend('Angle');
```

```
t1 = 3;
right = exp(-j*k*w*t1).*a;
right = eval(right);
subplot(211);
stem(k,abs(right));
legend('Magnitude');
title('Right part');
subplot(212);
stem(k,angle(right));
legend('Angle');
```

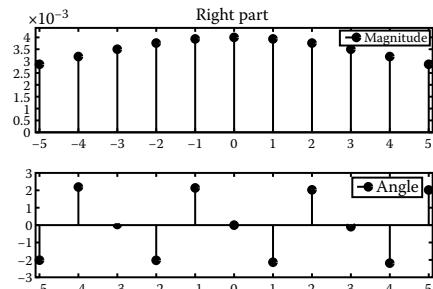
```
x = (t-t1).*exp(-5*(t-t1));
a = (1/T)*int(x*exp(-j*k*w*t),t,t0+t1,t0+T+t1);
coe = eval(a);
subplot(211);
stem(k,abs(coe));
legend('Magnitude');
title('Coefficients of (t-3)exp(-5(t-3))');
subplot(212);
stem(k,angle(coe));
legend('Angle');
```

Results/Comments

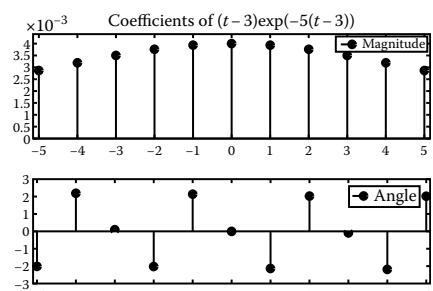
First, the Fourier series exponential coefficients a_k , $-5 \leq k \leq 5$ for the signal $x(t) = te^{-5t}$, $0 \leq t \leq 10$ are computed and plotted.



Next, we compute the right part of (5.17) by multiplying the coefficients a_k by the quantity $e^{-jk\Omega_0 t_1}$.



Finally, we define the signal $y(t) = (t-3)e^{-5(t-3)}$, which is a time-shifted version of $x(t)$, and compute the corresponding Fourier series coefficients.



The two last graphs are identical; hence, the time-shift property is confirmed. Comparing the two last graphs with the first one, we notice that indeed the magnitude does not change, but the phase is different.

5.9.3 Time Reversal

The Fourier series coefficients of the reflected version of a signal $x(t)$ are also a reflection of the coefficients of $x(t)$. So, if $x(t) \leftrightarrow a_k$, the mathematical expression is

$$x(-t) \leftrightarrow a_{-k}. \quad (5.18)$$

In order to validate the time-reversal property, we consider the periodic signal that in one period is given by $x(t) = t \cos(t)$, $0 \leq t \leq 2\pi$.

Commands	Results/Comments
<pre>t0=0; T=2*pi; w=2*pi/T; syms t x=t*cos(t); k=-5:5; a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T); a1=eval(a); subplot(211); stem(k,real(a1)); legend('Re[a_k]'); title('Coefficients of x(t)'); subplot(212); stem(k,imag(a1)); legend('Im[a_k]');</pre>	<p>First, the Fourier series exponential coefficients a_k, $-5 \leq k \leq 5$ for the signal $x(t) = t \cos(t)$, $0 \leq t \leq 2\pi$ are computed and plotted.</p>
<pre>x_=-t*cos(-t); b=(1/T)*int(x_*exp(-j*k*w*t),t,t0-T,t0); b1=eval(b); subplot(211); stem(k,real(b1)); legend('Re[b_k]'); title('Coefficients of x(-t)'); subplot(212); stem(k,imag(b1)); legend('Im[b_k]');</pre>	<p>Next, we compute the coefficients b_k of the time reversed version $x(-t)$, and we notice that $b_k = a_{-k}$. Hence, the time-reversal property of (5.18) is confirmed.</p>

5.9.4 Time Scaling

The Fourier series coefficients of a time-scaled version $x(\lambda t)$ of $x(t)$ do not change. On the other hand, the fundamental period of the scaled version becomes T/λ , and the fundamental frequency becomes $\lambda\Omega_0$. The mathematical expression is

$$x(\lambda t) \leftrightarrow a_k. \quad (5.19)$$

The time-scaling property is confirmed by using the periodic signal that in one period is given by $x(t) = t \cos(t)$, $0 \leq t \leq 2\pi$.

Commands	Results/Comments																																				
<pre> syms t t0 = 0; T=2*pi; w=2*pi/T; x=t*cos(t); k=-5:5; a = (1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T); a1=eval(a) subplot(211); stem(k,abs(a1)); legend('Magnitude'); title('Coefficients of x(t)'); subplot(212); stem(k,angle(a1)); legend('Angle'); </pre>	<p>First, the Fourier series exponential coefficients a_k, $-5 \leq k \leq 5$ for the signal $x(t) = t \cos(t)$, $0 \leq t \leq 2\pi$ are computed and plotted.</p> <table border="1"> <caption>Coefficients of $x(t)$</caption> <thead> <tr> <th>k</th> <th>Magnitude</th> <th>Angle</th> </tr> </thead> <tbody> <tr><td>-5</td><td>~0.2</td><td>-1.57</td></tr> <tr><td>-4</td><td>~0.3</td><td>-1.57</td></tr> <tr><td>-3</td><td>~0.35</td><td>-1.57</td></tr> <tr><td>-2</td><td>~0.45</td><td>-1.57</td></tr> <tr><td>-1</td><td>1.0</td><td>0.0</td></tr> <tr><td>0</td><td>0.0</td><td>0.0</td></tr> <tr><td>1</td><td>1.0</td><td>0.0</td></tr> <tr><td>2</td><td>~0.45</td><td>1.57</td></tr> <tr><td>3</td><td>~0.35</td><td>1.57</td></tr> <tr><td>4</td><td>~0.3</td><td>1.57</td></tr> <tr><td>5</td><td>~0.2</td><td>1.57</td></tr> </tbody> </table>	k	Magnitude	Angle	-5	~0.2	-1.57	-4	~0.3	-1.57	-3	~0.35	-1.57	-2	~0.45	-1.57	-1	1.0	0.0	0	0.0	0.0	1	1.0	0.0	2	~0.45	1.57	3	~0.35	1.57	4	~0.3	1.57	5	~0.2	1.57
k	Magnitude	Angle																																			
-5	~0.2	-1.57																																			
-4	~0.3	-1.57																																			
-3	~0.35	-1.57																																			
-2	~0.45	-1.57																																			
-1	1.0	0.0																																			
0	0.0	0.0																																			
1	1.0	0.0																																			
2	~0.45	1.57																																			
3	~0.35	1.57																																			
4	~0.3	1.57																																			
5	~0.2	1.57																																			
<pre> lamda = 2; T = T/ lamda; w=2*pi/T; x = lamda *t*cos(lamda *t); k=-5:5; a = (1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T); a1=eval(a) subplot(211); stem(k,abs(a1)); legend('Magnitude'); title('Coefficients of x(2t)'); subplot(212); stem(k,angle(a1)); legend('Angle'); </pre>	<p>Next, we compute the coefficients b_k of the time-scaled signal $x(2t) = 2t \cos(2t)$, $0 \leq t \leq \pi$, and we notice that $b_k = a_k$. Hence, the time-scaling property is confirmed.</p> <table border="1"> <caption>Coefficients of $x(2t)$</caption> <thead> <tr> <th>k</th> <th>Magnitude</th> <th>Angle</th> </tr> </thead> <tbody> <tr><td>-5</td><td>~0.2</td><td>-1.57</td></tr> <tr><td>-4</td><td>~0.3</td><td>-1.57</td></tr> <tr><td>-3</td><td>~0.35</td><td>-1.57</td></tr> <tr><td>-2</td><td>~0.45</td><td>-1.57</td></tr> <tr><td>-1</td><td>1.0</td><td>0.0</td></tr> <tr><td>0</td><td>0.0</td><td>0.0</td></tr> <tr><td>1</td><td>1.0</td><td>0.0</td></tr> <tr><td>2</td><td>~0.45</td><td>1.57</td></tr> <tr><td>3</td><td>~0.35</td><td>1.57</td></tr> <tr><td>4</td><td>~0.3</td><td>1.57</td></tr> <tr><td>5</td><td>~0.2</td><td>1.57</td></tr> </tbody> </table>	k	Magnitude	Angle	-5	~0.2	-1.57	-4	~0.3	-1.57	-3	~0.35	-1.57	-2	~0.45	-1.57	-1	1.0	0.0	0	0.0	0.0	1	1.0	0.0	2	~0.45	1.57	3	~0.35	1.57	4	~0.3	1.57	5	~0.2	1.57
k	Magnitude	Angle																																			
-5	~0.2	-1.57																																			
-4	~0.3	-1.57																																			
-3	~0.35	-1.57																																			
-2	~0.45	-1.57																																			
-1	1.0	0.0																																			
0	0.0	0.0																																			
1	1.0	0.0																																			
2	~0.45	1.57																																			
3	~0.35	1.57																																			
4	~0.3	1.57																																			
5	~0.2	1.57																																			

5.9.5 Signal Multiplication

The Fourier series coefficients of the product of two signals equals the convolution of the Fourier series coefficients of each signal. Suppose that $x(t) \leftrightarrow a_k$ and $y(t) \leftrightarrow b_k$.

Then,

$$x(t)y(t) \leftrightarrow a_k * b_k, \quad (5.20)$$

where $*$ denotes (discrete time) convolution. To verify property (5.20), we consider the signals $x(t) = \cos(t)$ and $y(t) = \sin(t)$.

Commands	Results/Comments
<pre> syms t t0=0; T=2*pi; w=2*pi/T; x=cos(t); k=-5:5; a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T); a1=eval(a); y=sin(t); b=(1/T)*int(y*exp(-j*k*w*t),t,t0,t0+T); b1=eval(b); left=conv(a1,b1); subplot(211); stem(-10:10,abs(left)); legend('Magnitude'); title('a_k*b_k'); subplot(212); stem(-10:10,angle(left)); legend('Angle'); </pre>	<p>First, we compute the exponential Fourier series coefficients a_k, $-5 \leq k \leq 5$ and b_k, $-5 \leq k \leq 5$, and the left part of (5.20) is computed by their convolution. Notice that the convolution $d_k = a_k * b_k$ is implemented between two complex-valued sequences. As discussed in the Chapter 4, the procedure is same as the one between real-valued sequences.</p>
<pre> z=x*y; k=-10:10; c=(1/T)*int(z*exp(-j*k*w*t),t,t0,t0+T); c1=eval(c) subplot(211); stem(k,abs(c1)); legend('Magnitude'); title('Coefficients of x(t)y(t)'); subplot(212); stem(k,angle(c1)); legend('Angle'); </pre>	<p>Next, we compute the Fourier series coefficients c_k, $-10 \leq k \leq 10$ of the signal $z(t) = \cos(t)\sin(t)$, and we notice that $c_k = d_k$. Hence, property (5.20) is verified.</p>

5.10 Symmetry

In this section, we discuss the coefficients of the various Fourier series forms in relation to the symmetry of the signal $x(t)$.

5.10.1 Even Symmetry

Recall that when a signal $x(t)$ is an even function of t , the relationship $x(-t) = x(t)$ stands. In this case, the trigonometric Fourier series coefficients c_k are zero. In order to demonstrate this property, the coefficients c_k of an even signal are computed for the time of one period, i.e., $-T/2 \leq t \leq T/2$ in two parts. First, the coefficients c_k are computed for $T/2 \leq t \leq 0$;

that is, c_k are computed for the signal $x(-t)$. However, because of the even symmetry $x(-t)=x(t)$. Thus,

Commands	Results
<pre>syms x t k T w=2*pi/T; c1=(2/T)*int(x*sin(k*w*t),t,-T/2,0);</pre>	$c1 = x * (-1 + \cos(k * \pi)) / k / \pi$

Next the coefficients c_k are computed for $0 \leq t \leq T/2$ and in order to calculate c_k for the signal $x(t)$, $-T/2 \leq t \leq T/2$ the two parts are added.

Commands	Results
$c2 = (2/T) * \text{int}(x * \sin(k * w * t), t, 0, T/2);$ $c = c1 + c2$	$c2 = -x * (-1 + \cos(k * \pi)) / k / \pi$ $c = 0$

Indeed, the trigonometric Fourier series coefficients c_k of a signal with even symmetry are zero. Regarding the coefficients a_k of the complex exponential Fourier series, when the signal $x(t)$ is even, the relationship $a_k = a_{-k}$ stands, namely, the coefficients also have even symmetry. In order to demonstrate this property, the first 11 coefficients a_k of the signal that in one period is defined as $x(t) = t^2$, $-2 \leq t \leq 2$ are computed and plotted. It is well known that $x(t) = t^2$ is a signal with even symmetry.

Commands	Results
<pre>t0 = -2; T = 4; w = 2*pi/T; syms t x = t^2; k = -5:5; a = (1/T) * int(x * exp(-j*k*w*t), t, t0, t0+T); stem(k, eval(a)) legend('a_k')</pre>	

From the above graph, it is clear that the complex exponential Fourier series coefficients of an even signal have also even symmetry, namely, $a_k = a_{-k}$.

5.10.2 Odd Symmetry

Recall that when a signal $x(t)$ is an odd function of t , the relationship $x(-t) = -x(t)$ stands. In this case, the trigonometric Fourier series coefficients b_k are zero. In order to demonstrate this property, the same procedure used in the previous section is followed. The coefficients b_k of an odd signal are computed for one period time, i.e., $-T/2 \leq t \leq T/2$ in two parts.

First, the coefficients b_k are computed for $-T/2 \leq t \leq 0$; that is, b_k are computed for the signal $x(-t)$. However, because of the even symmetry $x(-t) = -x(t)$. Thus,

Commands	Results
<pre>syms x t k T w=2*pi/T; b1=(2/T)*int(-x*cos(k*w*t),t,-T/2,0)</pre>	$b1 = -\sin(k\pi)x/k\pi$

Next, the coefficients b_k are computed for $0 \leq t \leq T/2$, and in order to calculate the coefficients b_k for the signal $x(t)$, $-T/2 \leq t \leq T/2$ the two parts are added.

Commands	Results
<pre>b2=(2/T)*int(x*cos(k*w*t),t,0,T/2) b=b1+b2</pre>	$b2 = \sin(k\pi)x/k\pi$ $b = 0$

Indeed, the trigonometric Fourier series coefficients b_k of a signal with odd symmetry are zero. Regarding the coefficients a_k of the complex exponential Fourier series, when the signal $x(t)$ is odd, the relationship $a_k = -a_{-k}$ stands, namely, the coefficients have also odd symmetry. In order to demonstrate this property, the first 11 coefficients a_k of the signal, which in one period is defined as $x(t) = t$, $-2 \leq t \leq 2$, are computed and plotted. Of course, $x(t) = t$ is a signal with odd symmetry.

Commands	Results	Comments
<pre>t0=-2; T=4; w=2*pi/T; syms tx=t; k=-5:5; a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T) stem(k,imag(eval(a))) legend('a_k')</pre>	$a = [2/5*i/pi, -1/2*i/pi, 2/3*i/pi, -i/pi, 2*i/pi, 0, -2*i/pi, i/pi, -2/3*i/pi, 1/2*i/pi, -2/5*i/pi]$	The coefficients a_k are complex numbers with odd symmetry and zero real part. In order to illustrate the odd symmetry, the imaginary part is presented in the figure below.

To sum up the last two sections, if a signal $x(t)$ is real and even, the complex exponential Fourier series coefficients a_k are real and even, and the trigonometric Fourier series coefficients c_k are zero. On the other hand, if a signal is real and odd, the complex exponential Fourier series coefficients a_k are imaginary and odd, and the trigonometric Fourier series coefficients b_k are zero.

5.11 Parseval's Identity

In this section, we discuss an important identity that allows us to compute the average power of a signal from its Fourier series coefficients. More specifically, the average power of a periodic signal $x(t)$ with period T equals the sum of the squares of the complex exponential Fourier series coefficients. The mathematical expression is

$$P_x = \frac{1}{T} \int_T |x(t)|^2 dt = \sum_{k=-\infty}^{\infty} |a_k|^2. \quad (5.21)$$

Moreover, if the signal $x(t)$ is real, Equation 5.21 is transformed to

$$P_x = \frac{1}{T} \int_T |x(t)|^2 dt = a_0^2 + 2 \sum_{k=1}^{\infty} |a_k|^2. \quad (5.22)$$

Example

Calculate the average power of the periodic signal $x(t) = \sin(t)$ and verify your result by calculating the average power from the complex exponential Fourier series coefficients.

Commands	Results	Comments
<pre> syms t x=sin(t); T=2*pi; t0=0; P=(1/T)*int(abs(x)^2,t0,t0+T); P=eval(P) </pre>	<pre>P = 0.5000</pre>	The average power is computed directly from the left part of Equation 5.21.
<pre> w=2*pi/T; k=-6:6; a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T) eval(a) </pre>	<pre> ans = 0 0 0 0 0 0+0.50i 0 0 -0.50i 0 0 0 0 </pre>	Calculation of the coefficients a_k .
<pre> P=sum((abs(a)).^2); eval(P) </pre>	<pre>ans = 0.5000</pre>	The average power is computed according to the right part of (5.21) and the result is verified.
<pre> a0=(1/T)*int(x,t0,t0+T); P=a0^2+2*sum((abs(a(7:13))).^2) eval(P) </pre>	<pre>ans = 0.5000</pre>	The average power is computed according to Equation 5.22 and the result is once again verified.

This example is somewhat trivial, due to the fact that there are only two nonzero coefficients a_k . In the following example the case of infinite nonzero coefficients a_k is discussed.

Example

Calculate the average power of a signal $x(t)$ that in one period is given by $x(t) = t$, $-1 \leq t \leq 1$. Verify the result by calculating the average power from the complex exponential Fourier series coefficients.

Commands	Results	Comments
<pre> syms t x=t; t0=-1; T=2; w=2*pi/T; P=(1/T)*int(abs(x)^2,t0,t0+T) </pre>	$P = 1/3$	The average power of the signal $x(t) = t$, $-1 \leq t \leq 1$.
<pre> k=-3:3; a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T) </pre>	$a = [1/3*i/pi, -1/2*i/pi, i/pi, 0, -i/pi, 1/2*i/pi, -1/3*i/pi]$	Calculation of the coefficients a_k for $-3 \leq k \leq 3$. In this case, there are infinite nonzero coefficients.
<pre> P=sum((abs(a)).^2); eval(P) </pre>	$ans = 0.2758$	The average power is calculated according to the right part of (5.21), but the result is not close to the real value, which is 0.3333.
<pre> k=-8:8; a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T); ans=0.3095 P=sum((abs(a)).^2); eval(P) </pre>		The average power is now computed for $-8 \leq k \leq 8$, and the result approaches somehow the real one.
<pre> k=-20:20; a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T); ans=0.3235 P=sum((abs(a)).^2); eval(P) </pre>		The average power is computed for $-20 \leq k \leq 20$, and the result appears to be quite close to the real one.
<pre> k=-100:100; a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T); ans=0.3313 P=sum((abs(a)).^2); eval(P) </pre>		Finally, the average power is computed for $-100 \leq k \leq 100$, and the result is very close to the real value of the power.

5.12 Criterion for the Approximation of a Signal by a Fourier Series Expansion

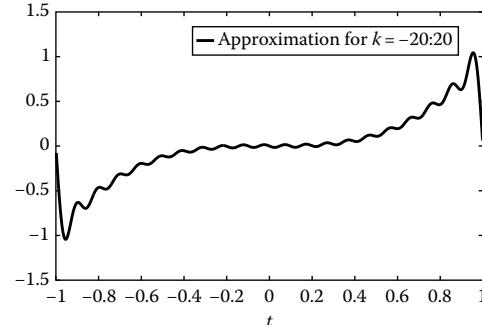
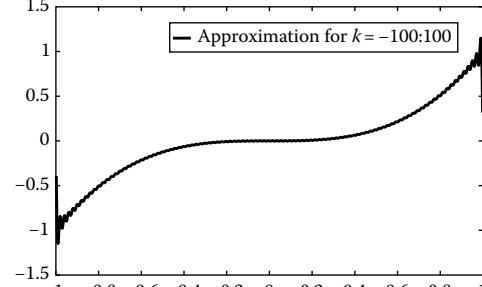
Taking into account the previous remarks, a criterion regarding the “quality” of an approximation of a periodic signal can be established. The approximation degree of a periodic signal $x(t)$ when $2K+1$ complex exponential Fourier series terms are used, i.e., when $x(t)$ is approximated by $\sum_{k=-K}^K a_k e^{jk\Omega_0 t}$, can be measured by the percentage of the average power of the signal that is contained in the complex exponential Fourier series coefficients. The mathematical expression for the approximation percentage is $\left(\frac{\sum_{k=-K}^K |a_k|^2}{P_x} \right) \cdot 100\%$.

Example

Calculate the approximation percentage for the signal that in one period is given by $x(t) = t^3$, $-1 \leq t \leq 1$, when using 7, 41, and 201 complex exponential Fourier series terms. Moreover, plot the approximate signal for each case.

Commands	Results/Comments
<pre> syms t x=t^3; t0=-1; T=2; w=2*pi/T; Px=(1/T)*int(abs(x)^2,t0,t0+T) ezplot(x, [-1 1]); legend('x(t)') </pre>	<p>The average power of the signal $x(t) = t^3$, $-1 \leq t \leq 1$ is $P_x = 1/7$. The signal $x(t)$ is plotted in the figure below.</p>
<pre> k=-3:3; a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T) xx=sum(a.*exp(j*k*w*t)); ezplot(xx, [-1 1]); legend('Approximation for k = -3:3') </pre>	<p>First, the coefficients a_k are calculated for $-3 \leq k \leq 3$ and the approximate signal is computed and plotted in the figure below. It is clear that this is not a good approximation.</p>
<pre> Pa=(abs(a)).^2; percentage=sum(Pa/Px) eval(percentage) % or alternatively Pa=sum((abs(a)).^2) per=Pa/Px eval(per) </pre>	<p>The approximation percentage is computed, and as expected is quite low, namely 61%.</p> <p style="text-align: right;">ans = 0.6101</p>

(continued)

Commands	Results/Comments
	Graph of the approximate signal when 41 terms are used, i.e., for $-20 \leq k \leq 20$. The approximate signal is now quite similar to the original.
<pre>k=-20:20; a=(1/T)*int(x*exp(-j*k*w*t), t,t0,t0+T); xx=sum(a.*exp(j*k*w*t)); ezplot(xx, [-1 1]); legend('Approximation for k = -20:20');</pre>	
<pre>Pa=(abs(a)).^2; percentage=sum(Pa/Px); eval(percentage)</pre>	<p>As expected, the approximation percentage is quite high, namely 93%.</p> <p style="text-align: right;">ans = 0.9309</p>
	Graph of the approximate signal when 201 terms are used, i.e., for $-100 \leq k \leq 100$.
<pre>k=-100:100; a=(1/T)*int(x*exp(-j*k*w*t), t,t0,t0+T); xx=sum(a.*exp(j*k*w*t)); ezplot(xx, [-1 1]); legend('Approximation for k = -100:100');</pre>	
<pre>Pa=sum((abs(a)).^2); per=Pa/Px; eval(per)</pre>	<p>The above graph illustrates that the approximation is very good. Indeed the approximation percentage is 98.6%.</p> <p style="text-align: right;">ans = 0.9859</p>

5.13 Relationship between Complex Exponential and Trigonometric Fourier Series Coefficients

In this section, the relationship between the complex exponential Fourier series coefficients a_k and the trigonometric Fourier series coefficients b_k and c_k is established. The following three equations describe this relationship mathematically:

$$b_k = a_k + a_{-k}, \quad k = 1, 2, \dots \quad (5.23)$$

$$c_k = j \cdot (a_k - a_{-k}), \quad k = 1, 2, \dots \quad (5.24)$$

$$a_k = \frac{1}{2}(b_k - j \cdot c_k), \quad k = 1, 2, \dots \quad (5.25)$$

Example

Verify Equations 5.23 through 5.25 by using the periodic signal that in one period is given by $x(t) = e^{-t}$, $0 \leq t \leq 5$.

In order to confirm the validity of (5.23) and (5.24), the exponential coefficients a_k are computed for $-6 \leq k \leq 6$ and the trigonometric coefficients b_n , c_n are computed for $1 \leq n \leq 6$ according to their definitions given in Equations 5.4, 5.8, and 5.9, respectively. Next, the coefficients b_k and c_k are computed according to (5.23), (5.24), and it is easily derived that coefficients b_n and c_n are equal to b_k and c_k .

Commands	Results	Comments																		
<pre> syms t; x=exp(-t); t0=0; T=5; w=2*pi/T; k=-6:6; n=1:6; b=(2/T)*int(x*cos(n*w*t),t,t0,t0+T); b=eval(b) c=(2/T)*int(x*sin(n*w*t),t,t0,t0+T); c=eval(c) </pre>	<table> <tbody> <tr> <td></td><td>b = 0.1540</td><td>0.0543</td><td>0.0261</td></tr> <tr> <td></td><td></td><td>0.0151</td><td>0.0098</td><td>0.0069 Calculation of b_n,</td></tr> <tr> <td></td><td>c = 0.1936</td><td>0.1365</td><td>0.0985 c_n from (5.8) and</td></tr> <tr> <td></td><td></td><td>0.0760</td><td>0.0617</td><td>(5.9).</td></tr> </tbody> </table>		b = 0.1540	0.0543	0.0261			0.0151	0.0098	0.0069 Calculation of b_n ,		c = 0.1936	0.1365	0.0985 c_n from (5.8) and			0.0760	0.0617	(5.9).	
	b = 0.1540	0.0543	0.0261																	
		0.0151	0.0098	0.0069 Calculation of b_n ,																
	c = 0.1936	0.1365	0.0985 c_n from (5.8) and																	
		0.0760	0.0617	(5.9).																
<pre> a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T); a=eval(a); for i=1:6 bb(7-i)=a(14-i)+a(i); cc(7-i)=j*(a(14-i)-a(i)); end bb cc </pre>	<table> <tbody> <tr> <td></td><td>bb = 0.1540</td><td>0.0543</td><td>0.0261</td></tr> <tr> <td></td><td>0.0151</td><td>0.0098</td><td>0.0069</td></tr> <tr> <td></td><td>cc = 0.1936</td><td>0.1365</td><td>0.0985</td></tr> <tr> <td></td><td>0.0760</td><td>0.0617</td><td>0.0518</td></tr> </tbody> </table>		bb = 0.1540	0.0543	0.0261		0.0151	0.0098	0.0069		cc = 0.1936	0.1365	0.0985		0.0760	0.0617	0.0518	<p>Calculation of a_k from (5.4) and b_k, c_k from (5.23) and (5.24), respectively.</p> <p>The coefficients b_k, c_k are equal to the coefficients b_n, c_n, which were computed above; hence, the validity of Equations 5.23 and 5.24 is clearly illustrated.</p>		
	bb = 0.1540	0.0543	0.0261																	
	0.0151	0.0098	0.0069																	
	cc = 0.1936	0.1365	0.0985																	
	0.0760	0.0617	0.0518																	

In order to verify (5.25), the coefficients a_n are computed for $1 \leq n \leq 6$ according to (5.25), and are compared with the coefficients a_k that have been already computed for $1 \leq k \leq 6$ according to (5.4).

Commands	Results	Comments									
<pre> an=(1/2)*(b-j*c) </pre>	<table> <tbody> <tr> <td>an = 0.0770-0.0968i</td><td>0.0272-0.0682i</td><td></td></tr> <tr> <td>0.0131-0.0492i</td><td>0.0076-0.0380i</td><td>Computation of a_n for $1 \leq n \leq 6$ according to (5.25).</td></tr> <tr> <td>0.0049-0.0308i</td><td>0.0034-0.0259i</td><td></td></tr> </tbody> </table>	an = 0.0770-0.0968i	0.0272-0.0682i		0.0131-0.0492i	0.0076-0.0380i	Computation of a_n for $1 \leq n \leq 6$ according to (5.25).	0.0049-0.0308i	0.0034-0.0259i		
an = 0.0770-0.0968i	0.0272-0.0682i										
0.0131-0.0492i	0.0076-0.0380i	Computation of a_n for $1 \leq n \leq 6$ according to (5.25).									
0.0049-0.0308i	0.0034-0.0259i										
<pre> ak(1:6)=a(8:13) </pre>	<table> <tbody> <tr> <td>ak = 0.0770-0.0968i</td><td>0.0272-0.0682i</td><td></td></tr> <tr> <td>0.0131-0.0492i</td><td>0.0076-0.0380i</td><td>The coefficients a_k for $1 \leq k \leq 6$ are already computed according to (5.4).</td></tr> <tr> <td>0.0049-0.0308i</td><td>0.0034-0.0259i</td><td></td></tr> </tbody> </table>	ak = 0.0770-0.0968i	0.0272-0.0682i		0.0131-0.0492i	0.0076-0.0380i	The coefficients a_k for $1 \leq k \leq 6$ are already computed according to (5.4).	0.0049-0.0308i	0.0034-0.0259i		
ak = 0.0770-0.0968i	0.0272-0.0682i										
0.0131-0.0492i	0.0076-0.0380i	The coefficients a_k for $1 \leq k \leq 6$ are already computed according to (5.4).									
0.0049-0.0308i	0.0034-0.0259i										
		The coefficients given by Equations 5.4 and 5.25 are equal; thus the validity of (5.25) is clear.									

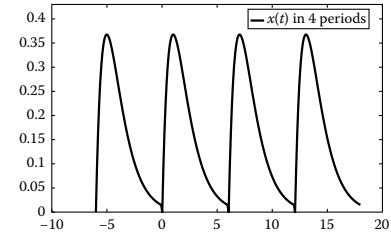
5.14 Solved Problems

Problem 1 The periodic signal $x(t)$ is defined in one period as $x(t) = te^{-t}$, $0 \leq t \leq 6$. Plot in time of 4 periods the approximate signals using 81 terms of the complex exponential and of the trigonometric forms of Fourier series. For comparison reasons, plot the original signal $x(t)$ over the same time interval.

Solution

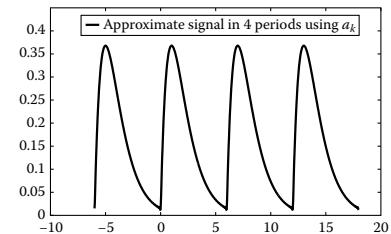
The signal $x(t)$ in four periods.

```
t = 0:.1:6;
x = t.*exp(-t);
xp = repmat(x, 1, 4);
tp = linspace(-6, 18, length(xp));
plot(tp, xp)
legend('x(t) in 4 periods')
```



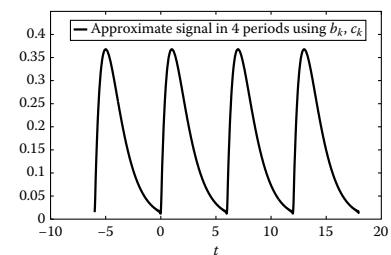
```
t0 = 0;
T = 6;
w = 2*pi/T;
syms t
x = t.*exp(-t);
k = -40:40;
a = (1/T)*int(x*exp(-j*k*w*t), t, t0, t0+T);
xx = sum(a.*exp(j*k*w*t));
ezplot(xx, [-6 18]);
legend('approximate signal in 4 periods using a_k')
```

The approximate signal computed from the complex exponential terms is plotted in four periods.



```
a0 = (1/T)*int(x, t0, t0+T);
n = 1:81;
b = (2/T)*int(x*cos(n*w*t), t, t0, t0+T);
c = (2/T)*int(x*sin(n*w*t), t, t0, t0+T);
xx = a0 + sum(b.*cos(n*w*t)) + sum(c.*sin(n*w*t))
ezplot(xx, [-6 18]);
legend('approximate signal in 4 periods using b_k, c_k')
```

The approximate signal computed from the trigonometric terms is plotted in four periods.



Problem 2 Plot the coefficients of the three Fourier series forms for the periodic signal that in one period is defined by $x(t) = e^{-t^2}$, $-3 \leq t \leq 3$.

Solution

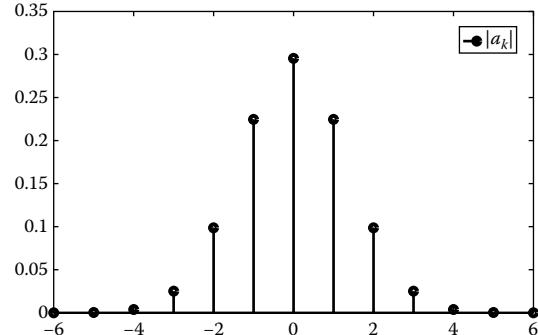
First, the coefficients a_k of the complex exponential form are computed for $-6 \leq k \leq 6$. In order to plot the coefficients a_k , the magnitudes $|a_k|$ and angles $\angle a_k$ are plotted.

```

syms t k
x=exp(-t^2);
t0=-3;
T=6;
w=2*pi/T
a=(1/T)*int(x*exp(-j*k*w*t), t,t0,t0+T);
k1=-6:6;
ak=subs(a,k,k1);
stem(k1,abs(ak));
legend('|a_k|')

```

The magnitudes $|a_k|$.

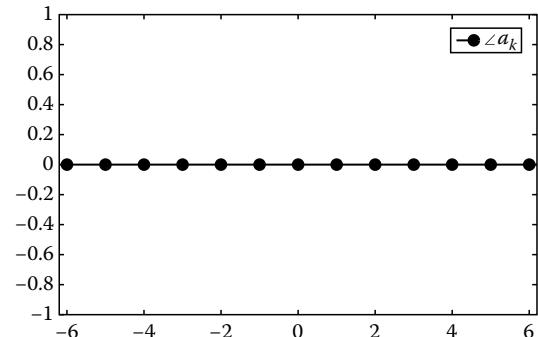


The angles $\angle a_k$.

```

stem(k1,angle(ak));
legend('\angle a_k')

```



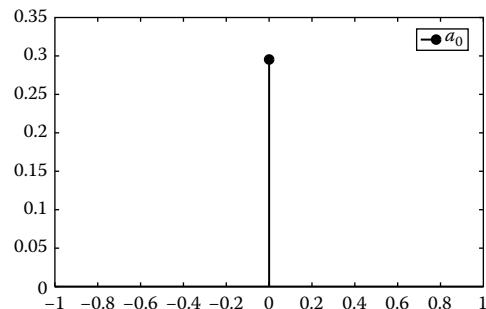
Next, the dc component a_0 and the coefficients b_n , c_n of the trigonometric form are computed and plotted for $1 \leq n \leq 10$.

```

syms n
a0=(1/T)*int(x,t0,t0+T);
stem(0,eval(a0))
legend('a_0');

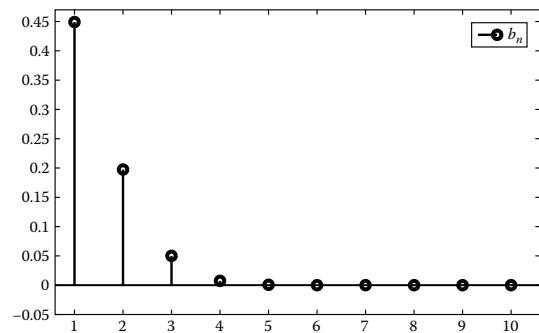
```

The dc component a_0 .

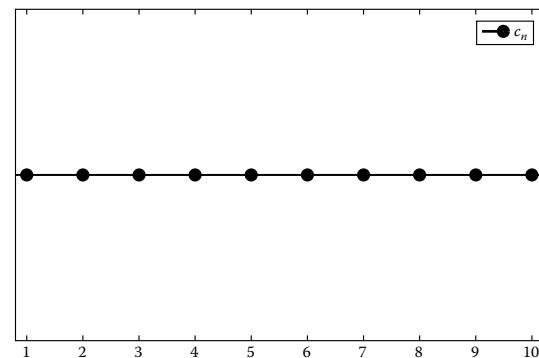


(continued)

```
b = (2/T) * int(x*cos(n*w*t),t,t0,t0+T)
n1=1:10;
bn=subs(b,n,n1);
stem(n1, bn);
legend('b_n');
```

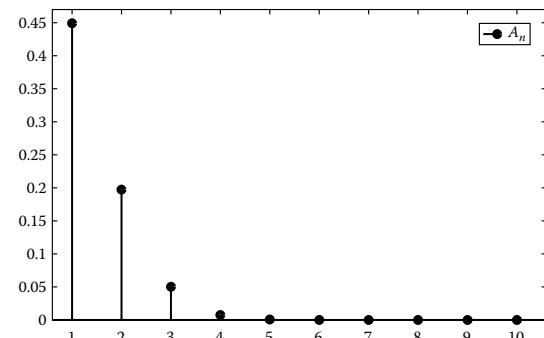
The coefficients b_n .

```
c = (2/T) * int(x*sin(n*w*t),t,t0,t0+T);
cn=subs(c,n,n1);
stem(n1, cn);
legend('c_n');
ylim([-1e-10,1e-10])
```

The coefficients c_n are zero due to the even symmetry of the signal.

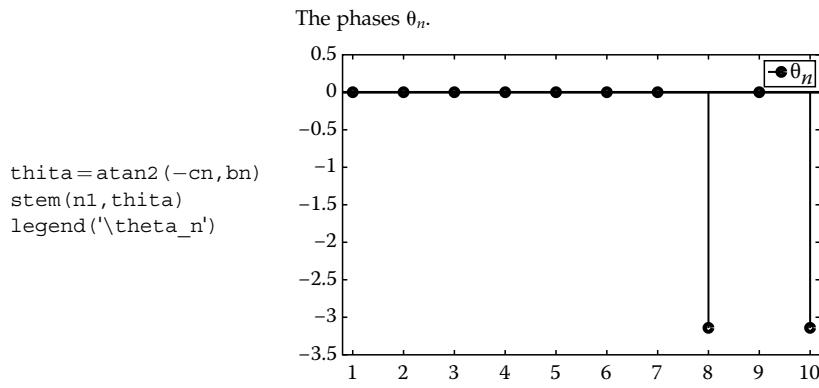
Finally, the amplitudes and phases of the cosine with phase form are computed and plotted for $1 \leq n \leq 10$.

```
A=sqrt(b.^2+c.^2);
An=subs(A,n,n1);
stem(n1, An);
legend('A_n')
```

The amplitudes A_n .

(continued)

(continued)



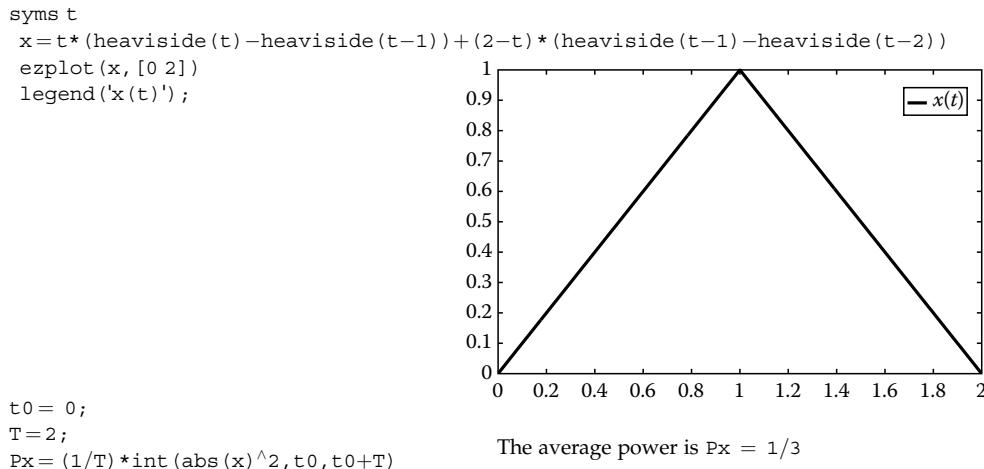
Problem 3 The periodic signal $x(t)$ in one period is given by

$$x(t) = \begin{cases} t, & 0 \leq t \leq 1 \\ 2-t, & 1 \leq t \leq 2 \end{cases}.$$

Calculate the approximation percentage when the signal $x(t)$ is approximated by 3, 5, 7, and 17 terms of the complex exponential Fourier series. Furthermore, plot the approximate signal for each case.

Solution

First, the signal $x(t)$ is expressed as a single symbolic expression, namely, is expressed as $x(t) = t[u(t) - u(t-1)] + (2-t)[u(t-1) - u(t-2)]$. For reference reasons, $x(t)$ is plotted in the figure below in one period. Moreover, the average power of the signal is computed.

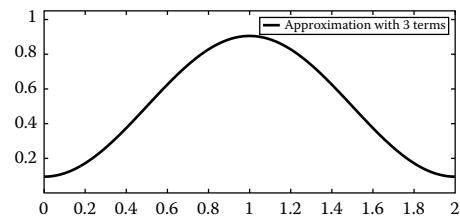


Next, the complex exponential Fourier series coefficients a_k are calculated for different values of k , and the corresponding approximation percentage is also computed.

```

k=-1:1;
w=2*pi/T;
a=(1/T)*int(x*exp(-j*k*w*t), t,t0,t0+T);
xx=sum(a.*exp(j*k*w*t));
ezplot(xx, [0 2])
legend('Approximation with 3 terms')

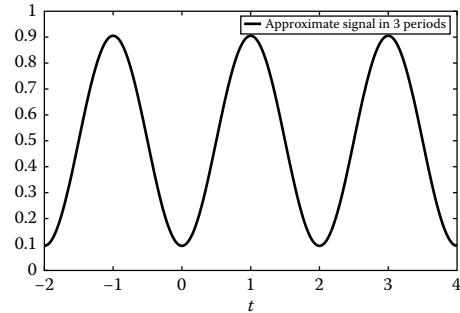
```



```

ezplot(xx, [-2 4])
legend('Approximate signal in 3 periods')

```



```

Pa = sum( (abs(a)).^2 );
per = Pa/Px
percentage = eval(per)

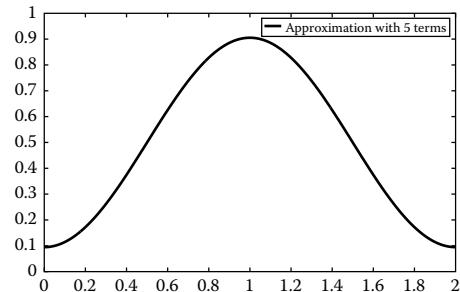
```

The approximation percentage is 99.64%.
percentage = 0.9964

```

k=-2:2;
a=(1/T)*int(x*exp(-j*k*w*t), t,t0,t0+T);
xx=sum(a.*exp(j*k*w*t));
ezplot(xx, [0 2])
legend('Approximation with 5 terms')

```



```

Pa = sum( (abs(a)).^2 );
per = Pa/Px
percentage = eval(per)

```

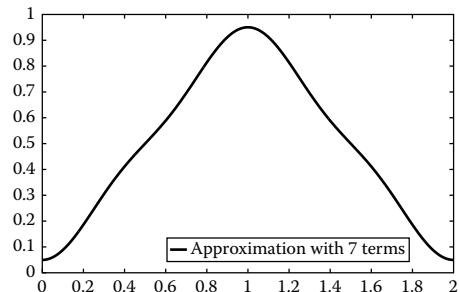
The approximation percentage is 99.64%.
The estimated approximation percentage is exactly the same. This is explained by the fact that the coefficients a_2 and a_{-2} are zero; thus they do not contribute to the Fourier series expansion of the signal.

percentage = 0.9964

(continued)

(continued)

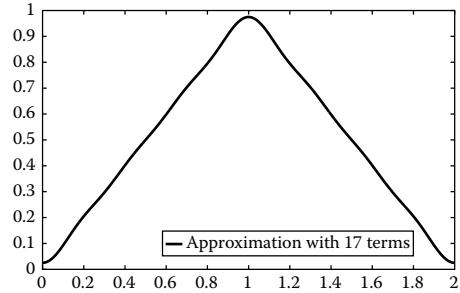
```
k=-3:3;
a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T);
xx=sum(a.*exp(j*k*w*t));
ezplot(xx,[0 2])
legend('Approximation with 7 terms')
```



```
Pa = sum( (abs(a)).^2 );
per = Pa/Px;
percentage = eval(per)
```

The approximation percentage is 99.94%.
percentage = 0.9994

```
k=-8:8;
a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T);
xx=sum(a.*exp(j*k*w*t));
ezplot(xx,[0 2])
legend('Approximation with 17 terms')
```



```
Pa = sum( (abs(a)).^2 );
per = Pa/Px;
percentage = eval(per)
```

The approximation percentage is 99.99% and the approximate signal is almost identical to the original.
percentage = 0.9999

Problem 4 The periodic signal $x(t)$ in one period is given by

$$x(t) = \begin{cases} 1, & 0 \leq t \leq 1 \\ 0, & 1 \leq t \leq 2 \end{cases}$$

Plot in one period the approximate signals using 41 and 201 terms of the complex exponential Fourier series. Furthermore, each time plot the complex exponential coefficients.

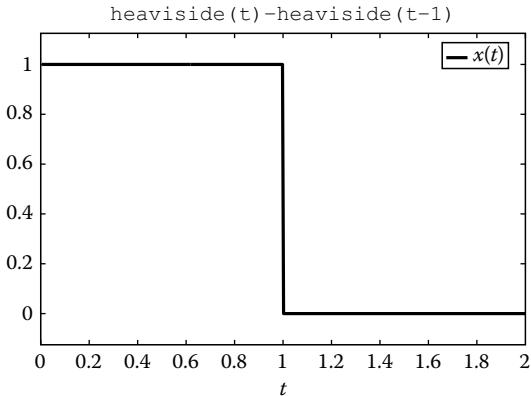
Solution

First, the signal $x(t)$ is expressed as a single symbolic expression, namely, is expressed as $x(t) = u(t) - u(t - 1)$. For reference reasons, $x(t)$ is plotted in the figure below.

```

syms t
t0 = 0;
T=2;
w=2*pi/T;
x=heaviside(t)-heaviside(t-1);
ezplot(x, [0 2])
legend('x(t)')

```

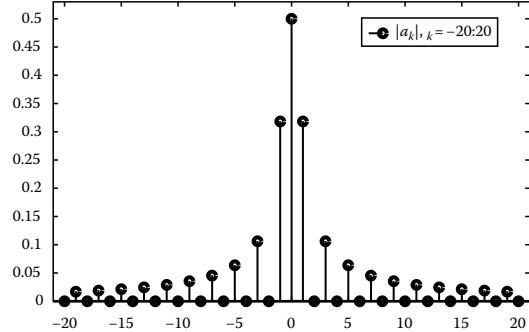


Next, the magnitudes and the angles of the coefficients a_k are computed and plotted in the figures below.

```

k=-20:20;
a=(1/T)*int(x*exp(-j*k*w*t),t,t0,t0+T);
a=eval(a);
stem(k,abs(a));
legend('|a_k|', k=-20:20);

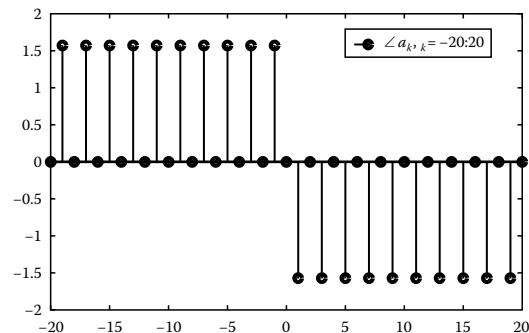
```



```

stem(k,angle(a));
legend('\angle a_k, k = -20:20');

```



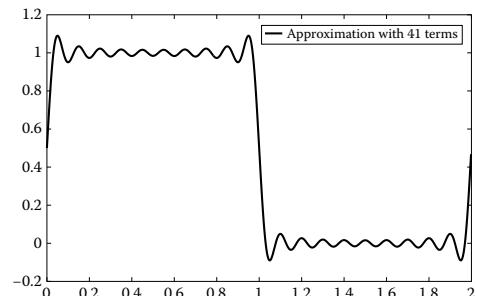
(continued)

(continued)

```

xx = sum(a.*exp(j*k*w*t)) ;
ezplot(xx, [0 2])
legend('Approximation with 41 terms')

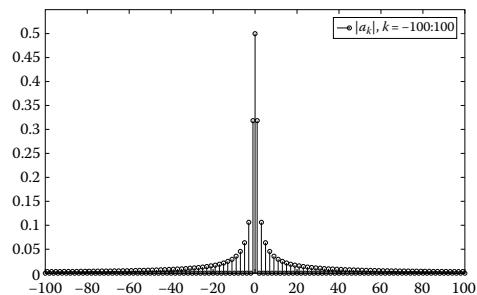
```



```

k = -100:100;
a = (1/T)*int(x*exp(-j*k*w*t), t, t0, t0+T);
a = eval(a);
stem(k, abs(a));
legend('|a_k|, k = -100:100');

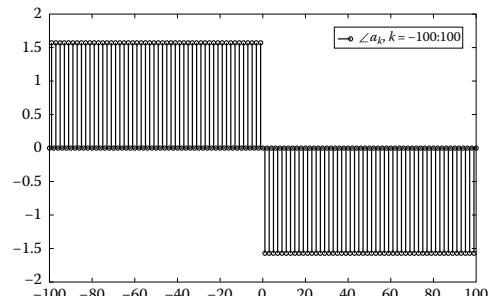
```



```

stem(k, angle(a));
legend('\angle a_k, k = -100:100');

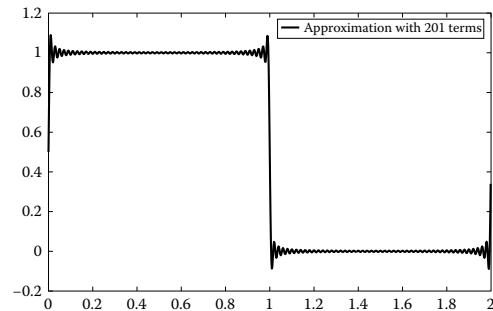
```



```

xx = sum(a.*exp(j*k*w*t));
ezplot(xx, [0 2])
legend('Approximation with 201 terms')

```



Problem 5 The periodic signal $x(t)$ in a period is given by

$$x(t) = \begin{cases} 1, & 0 \leq t \leq 1 \\ 0, & 1 \leq t \leq 2 \end{cases}.$$

Plot in one period the approximate signals using 41 and 201 terms of the trigonometric Fourier series. Furthermore plot the coefficients of the trigonometric form for every case.

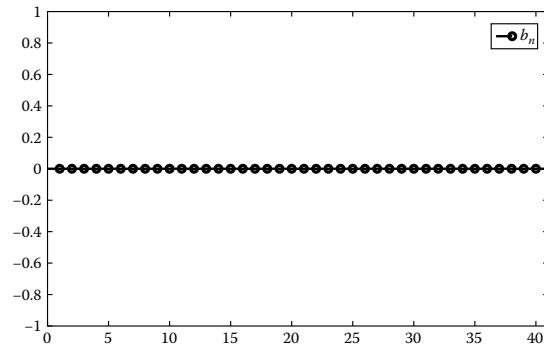
Solution

```
syms t
x=heaviside(t)-heaviside(t-1)
t0=0;
T=2;
w=2*pi/T;
```

The signal $x(t)$ is expressed as a single symbolic expression (see previous problem for details). Moreover, t_0, T , and Ω_0 are defined.

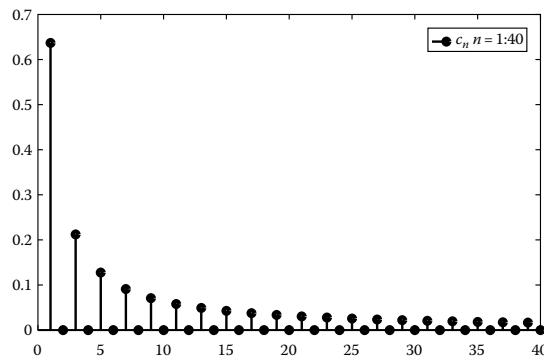
```
n=1:40;
b=(2/T)*int(x*cos(n*w*t),t,t0,t0+T)
stem(n,eval(b))
legend('b_n')
```

The coefficients b_n are zero.



```
c=(2/T)*int(x*sin(n*w*t),t,t0,t0+T);
stem(n,eval(c))
legend('c_n n=1:40')
```

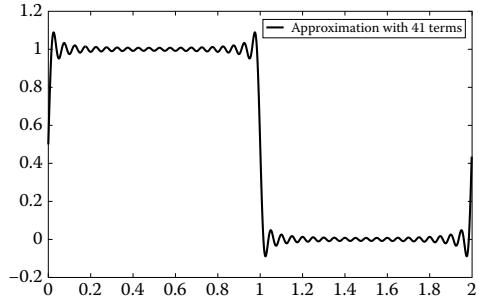
The coefficients c_n .



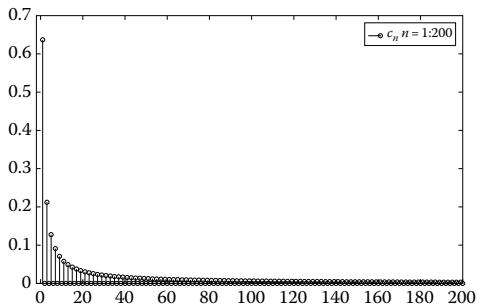
(continued)

(continued)

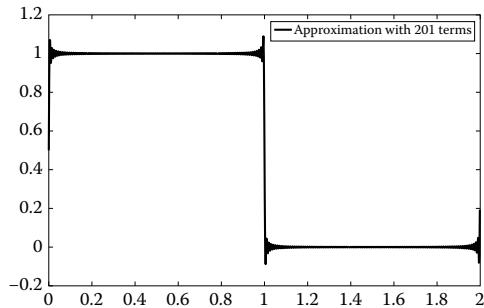
```
a0 = (1/T)*int(x,t0,t0+T);
xx=a0+sum(b.*cos(n*w*t))+sum(c.*sin(n*w*t))
ezplot(xx,[0 2])
legend('Approximation with 41 terms')
```



```
n=1:200;
b=(2/T)*int(x*cos(n*w*t),t,t0,t0+T)
c=(2/T)*int(x*sin(n*w*t),t,t0,t0+T);
stem(n,eval(c));
legend('c_n n=1:200')
```



```
xx=a0+sum(b.*cos(n*w*t))+sum(c.*sin(n*w*t))
ezplot(xx,[0 2])
legend('Approximation with 201 terms')
```



Problem 6 The periodic signal $x(t)$ in one period is given by

$$x(t) = \begin{cases} 1, & 0 \leq t \leq 1 \\ 0, & 1 \leq t \leq 2 \end{cases}.$$

Plot in one period the approximate signals using 41 and 201 terms of the Fourier series-Series in the cosine with phase form. Furthermore, each time plot the coefficients (i.e., the amplitudes and phases) of the cosine with phase form.

Solution

```

syms t
x=heaviside(t)-heaviside(t-1)
t0=0;
T=2;
w=2*pi/T;

n=1:40;
b=(2/T)*int(x*cos(n*w*t),t,t0,t0+T)
c=(2/T)*int(x*sin(n*w*t),t,t0,t0+T)

```

The signal $x(t)$ is expressed as a single symbolic expression and t_0 , T , and Ω_0 are defined.

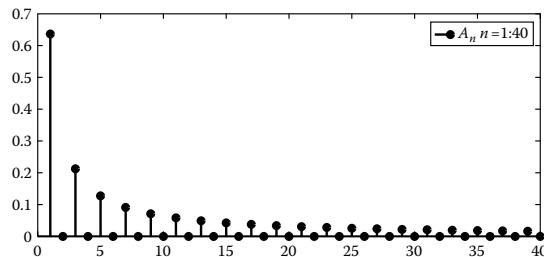
```

A=sqrt(b.^2+c.^2);
stem(n,eval(A));
legend('A_n n=1:40')

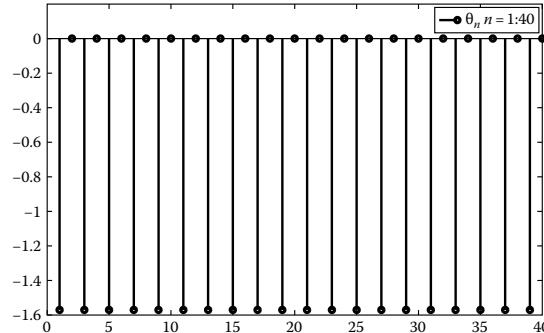
```

The coefficients b_n and c_n of the trigonometric form are computed.

Graph of the amplitudes A_n .



Graph of the phases θ_n .



```

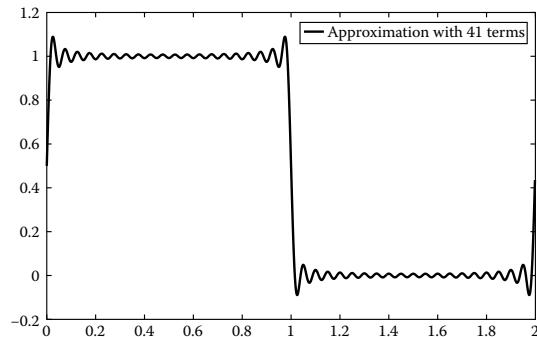
thita=atan2(-eval(c),eval(b));
stem(n,thita)
legend('theta_n n=1:40')

```

```

a0=(1/T)*int(x,t0,t0+T);
xx=a0+sum(A.*cos(n*w*t+thita))
ezplot(xx,[0 2])
legend('Approximation with 41 terms')

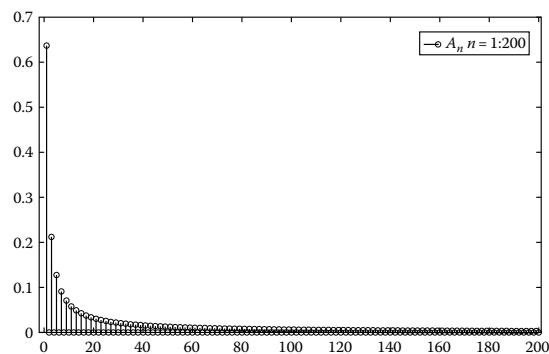
```



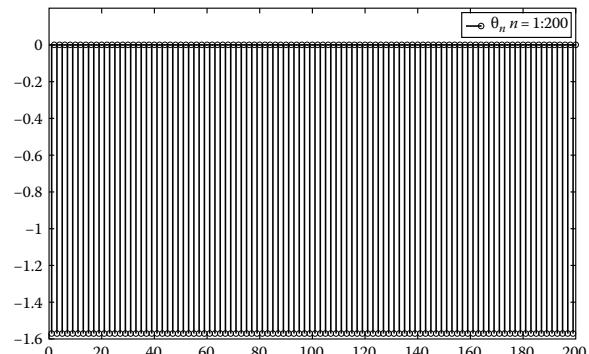
(continued)

(continued)

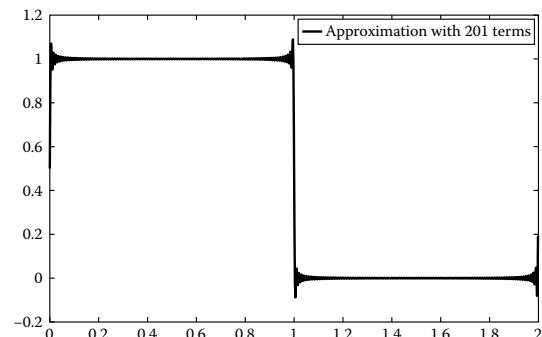
```
n=1:200;
b=(2/T)*int(x*cos(n*w*t),t,t0,t0+T);
c=(2/T)*int(x*sin(n*w*t),t,t0,t0+T);
A=sqrt(b.^2+c.^2);
stem(n,eval(A));
legend('A_n n=1:200')
```



```
thita= atan2(-eval(c),eval(b));
stem(n,thita)
legend('\theta_n n=1:200')
```



```
xx=a0+sum(A.*cos(n*w*t+thita))
ezplot(xx,[0 2])
legend('Approximation with 201 terms')
```



5.15 Homework Problems

1. The inner product of two discrete time periodic signals with period N is given by $I = \sum_N x_m[n]x_k^*[n]$. Find out if the signals $e^{j2\pi n/6}$ and $e^{j4\pi n/6+\pi/4}$ are orthogonal.
2. Suppose that m and n are positive integers and verify the following identities:
 - a. $\int_{-\pi}^{\pi} \sin(nt) \sin(mt) dt = \begin{cases} 0, & m \neq n \\ \pi, & m = n \end{cases}$
 - b. $\int_{-\pi}^{\pi} \cos(nt) \cos(mt) dt = \begin{cases} 0, & m \neq n \\ \pi, & m = n \end{cases}$
 - c. $\int_{-\pi}^{\pi} \sin(nt) \cos(mt) dt = 0$
 - d. $\int_{-\pi}^{\pi} \sin(nt) dt = 0$
 - e. $\int_{-\pi}^{\pi} \cos(nt) dt = 0$
3. Compute and plot the coefficients of the Fourier series in the cosine with phase form of the complex signal $x(t) = t^2 + j2\pi t$, $0 \leq t \leq 10$.
4. Approximate (by a number of terms of your choice) through the three forms of Fourier series expansion the periodic signal $x(t) = 2 \cos(t) + \sin(4t)$. Each time
 - a. Plot the approximate signal in time of five periods.
 - b. Compute and plot the Fourier series coefficients.

Moreover,

 - c. Plot the line spectra.
 - d. Compute the approximation percentage.
5. Approximate (by a number of terms of your choice) through the three forms of Fourier series expansion the periodic signal that in one period is given by

$$x(t) = \begin{cases} \sin(2\pi t), & 0 \leq t \leq 1/2 \\ 0, & 1/2 \leq t \leq 1 \end{cases}$$

Each time

- a. Plot the approximate signal in time of four periods.
- b. Compute and plot the Fourier series coefficients.

Moreover,

- c. Plot the line spectra.
- d. Compute the approximation percentage.

6. Approximate (by a number of terms of your choice) through the three forms of Fourier series expansion the periodic signal that in one period is given by

$$x(t) = \begin{cases} \sin(\pi t), & 0 \leq t \leq 1 \\ -2 \sin(\pi t), & 1 \leq t \leq 2 \end{cases}$$

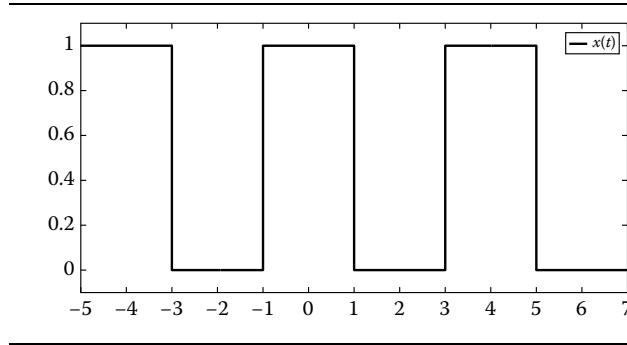
Each time

- Plot the approximate signal in time of six periods.
- Compute and plot the Fourier series coefficients.

Moreover,

- Plot the line spectra.
- Compute the approximation percentage.

- Approximate (using a number of terms of your choice) through the three forms of Fourier series expansion the periodic signal that is depicted in the following figure



Each time,

- Plot the approximate signal in time of three periods.
- Compute and plot the Fourier series coefficients.

Moreover,

- Plot the line spectra.
- Compute the approximation percentage.

- Approximate (by a number of terms of your choice) through the three forms of Fourier series expansion the periodic signal that in one period is given by

$$x(t) = \begin{cases} 1, & 0 \leq t \leq 1 \\ 0, & 1 < t \leq 2 \end{cases}$$

Each time,

- Plot the approximate signal in time of four periods.
- Compute and plot the Fourier series coefficients.

Moreover,

- Plot the line spectra.
- Compute the approximation percentage.

9. Approximate (by a number of terms of your choice) through the three forms of Fourier series expansion the periodic signal that in one period is given by

$$x(t) = \begin{cases} 1, & 0 \leq t \leq 2 \\ 0, & 2 < t \leq 4 \end{cases}$$

Each time

- a. Plot the approximate signal in time of four periods.
- b. Compute and plot the Fourier series coefficients.

Moreover,

- c. Plot the line spectra.
- d. Compute the approximation percentage.

10. Approximate (by a number of terms of your choice) through the three forms of Fourier series expansion the periodic signal that in one period is given by $x(t) = 0.5t$, $0 \leq t \leq 2$. Each time,

- a. Plot the approximate signal in time of three periods.
- b. Compute and plot the Fourier series coefficients.

Moreover,

- c. Plot the line spectra.
- d. Compute the approximation percentage.

11. Approximate (by a number of terms of your choice) through the three forms of Fourier series expansion the periodic signal that in one period is given by

$$x(t) = \begin{cases} 2t, & 0 \leq t \leq 0.5 \\ 2 - 2t, & 0.5 \leq t \leq 1.5 \\ 2t - 4, & 1.5 \leq t \leq 2 \end{cases}$$

Each time,

- a. Plot the approximate signal in time of four periods.
- b. Compute and plot the Fourier series coefficients.

Moreover,

- c. Plot the line spectra.
- d. Compute the approximation percentage.

12. Approximate (by a number of terms of your choice) through the three forms of Fourier series expansion the periodic signal that in one period is given by

$$x(t) = \sin\left(\frac{\pi}{2}t\right), \quad 0 \leq t \leq 2$$

Each time,

- Plot the approximate signal in time of three periods.
- Compute and plot the Fourier series coefficients.

Moreover,

- Plot the line spectra.
- Compute the approximation percentage.

13. Approximate (by a number of terms of your choice) through the three forms of Fourier series expansion the periodic signal that in one period is given by

$$x(t) = \begin{cases} (2/3)t, & 0 \leq t \leq 1.5 \\ 2 - (2/3)t, & 1.5 \leq t \leq 3 \end{cases}$$

Each time,

- Plot the approximate signal in time of three periods.
- Compute and plot the Fourier series coefficients.

Moreover,

- Plot the line spectra.
- Compute the approximation percentage.

14. Verify (using a signal of your choice) the following relationships:

- $|a_k| = |a_{-k}|$
- $\angle a_k = -\angle a_{-k}$
- $b_k = 2 \operatorname{Re}\{a_k\}$
- $c_k = -2 \operatorname{Im}\{a_k\}$
- $A_k = 2|a_k|$
- $P_x = a_0^2 + \frac{1}{2} \sum_{k=1}^{\infty} (b_k^2 + c_k^2)$

6

Fourier Transform

In Chapter 5, we introduced how a periodic signal defined for $-\infty < t < \infty$ or a signal defined in a closed time interval $a \leq t \leq b$ can be expanded in a Fourier series, i.e., how it can be expressed as a linear combination of infinite sinusoidal signals. The Fourier series expansion reveals the frequency content of a signal. In this chapter, we introduce a generalized concept that expresses in the frequency domain (almost), any signal defined in the time domain for $-\infty < t < \infty$. The *Fourier transform* is the way to express in the frequency domain a signal that is given in the time domain. In this chapter, we discuss the Fourier transform of continuous-time signals, which is known as continuous-time Fourier transform (CTFT). By applying Fourier transform to a continuous-time signal $x(t)$, we obtain a representation of the signal at the cyclic frequency domain Ω or equivalently at the frequency domain, f .

6.1 Mathematical Definition

As previously stated, the Fourier transform expresses a signal (or function) $x(t)$ in the (cyclic) frequency domain; that is, the signal is described by a function $X(\Omega)$. The Fourier transform is denoted by the symbol $F\{.\}$; that is, one can write

$$X(\Omega) = F\{x(t)\}. \quad (6.1)$$

In other words, the Fourier transform of a signal $x(t)$ is a signal $X(\Omega)$. An alternative way of writing (6.1) is

$$x(t) \xrightarrow{F} X(\Omega). \quad (6.2)$$

The mathematical expression of the Fourier transform is

$$X(\Omega) = F\{x(t)\} = \int_{-\infty}^{\infty} x(t) \cdot e^{-j\Omega t} dt. \quad (6.3)$$

From (6.3) it is clear that $X(\Omega)$ is a complex function of Ω . In case the Fourier transform of $x(t)$ has to be expressed in the frequency domain f , then substituting Ω by $2\pi f$ in (6.3) yields

$$X(f) = F\{x(t)\} = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft} dt. \quad (6.4)$$

In order to return from the frequency domain back to the time domain the *inverse Fourier transform* is applied. The inverse Fourier transform is denoted by the symbol $F^{-1}\{.\}$; that is, one can write

$$x(t) = F^{-1}\{X(\Omega)\}, \quad (6.5)$$

or alternatively

$$X(\Omega) \xrightarrow{F^{-1}} x(t). \quad (6.6)$$

The mathematical expression of the inverse Fourier transform is

$$x(t) = F^{-1}\{X(\Omega)\} = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\Omega) \cdot e^{j\Omega t} d\Omega. \quad (6.7)$$

Using f instead of Ω , Equation 6.7 becomes

$$x(t) = F^{-1}\{X(f)\} = \int_{-\infty}^{\infty} X(f) \cdot e^{j2\pi ft} df. \quad (6.8)$$

The cyclic frequency Ω is measured in rad/s, while the frequency f is measured in Hertz. The Fourier transform of a signal is called (frequency) *spectrum*.

6.2 The Commands `fourier` and `ifourier`

The computation of the integrals given in (6.3), (6.4), (6.7), and (6.8) is not always a trivial matter. However, in MATLAB® there is the possibility to compute directly the Fourier transform $X(\Omega)$ of a signal $x(t)$ by using the command `fourier`. Correspondingly, the inverse Fourier transform is computed by using the command `ifourier`. Before executing these two commands, time t and frequency Ω must be declared as symbolic variables. Recall that a symbolic variable is defined by using the command `syms`.

Example

Compute the Fourier transform of the function $x(t) = e^{-t^2}$.

Commands	Results	Comments
<pre>syms t w x=exp(-t^2); fourier(x) int(x*exp(-j*w*t),t,-inf,inf)</pre>	<pre>ans = pi^(1/2) * exp(-1/4*w^2) ans = exp(-1/4*w^2) * pi^(1/2)</pre>	<p>The Fourier transform of $x(t)$ is $X(\Omega) = \sqrt{\pi} \cdot e^{-(\Omega^2/4)}$</p> <p>The result is verified according to (6.3).</p>

Example

Compute the inverse Fourier transform of the function $X(\Omega) = 1/(1 + j\Omega)$.

Commands	Results	Comments
<code>X=1/(1+j*w); ifourier(X)</code>	<code>ans = exp(-x)*heaviside(x)</code>	The inverse Fourier transform of a function is expressed with x as the independent variable. The result is $e^{-x}u(x)$.

However, it is more appropriate to use t as the independent variable. This is accomplished by using the syntax `ifourier(X, t)`.

Commands	Results	Comments
<code>X=1/(1+j*w); ifourier(X, t)</code>	<code>ans = exp(-t)*heaviside(t)</code>	The inverse Fourier transform is expressed with t as the independent variable. The result is $e^{-t}u(t)$.

The command `fourier` should be executed as `fourier(x, w)`. Using this syntax is optimal since it allows the computation of constant functions.

Commands	Results	Comments
<code>x=1; fourier(x)</code>	?? Function 'fourier' is not defined for values of class 'double'.	The simple syntax is not appropriate for computing the Fourier transform of constant functions.
<code>fourier(x, w)</code>	<code>ans = 2*pi*dirac(w)</code>	Using the complete syntax allows us to compute the Fourier transform (if it exists) of any function.
<code>syms s fourier(x, s)</code>	<code>ans = 2*pi*dirac(s)</code>	Changing the variable in which the result is given.
<code>syms n X=1/(1+j*w); ifourier(X, n)</code>	<code>ans = exp(-n) *heaviside(n)</code>	A variable change is also possible at the <code>ifourier</code> command.

Example

Compute the Fourier transform (or spectrum) of the signal $e^{-t}u(t)$ and compare your result with that of the previous example.

Commands	Results	Comments
<code>syms t w x=exp(-t)*heaviside(t); X=fourier(x, w)</code>	<code>X = 1/(1+i*w)</code>	The Fourier transform $X(\Omega)$ of $x(t) = e^{-t}u(t)$ is $X(\Omega) = 1/(1 + j\Omega)$.

Comparing the derived result with that of the previous example, we notice that the functions $x(t) = e^{-t}u(t)$ and $X(\Omega) = 1/(1 + j\Omega)$ are a Fourier transform pair. In other words, the Fourier transform of $x(t) = e^{-t}u(t)$ is $X(\Omega) = 1/(1 + j\Omega)$ while the inverse Fourier transform of $X(\Omega) = 1/(1 + j\Omega)$ is $x(t) = e^{-t}u(t)$. Sometimes a Fourier transform pair is denoted by $x(t) \leftrightarrow X(\Omega)$. Thus in our case $e^{-t}u(t) \leftrightarrow 1/(1 + j\Omega)$. In Section 6.3 we present the most common Fourier transform pairs.

6.3 Fourier Transform Pairs

In this section, the most common Fourier transform pairs are presented. The computation of the Fourier and inverse Fourier transforms requires the computation of the integrals given in (6.4) and (6.7), and sometimes can be quite hard. This is the reason that already computed Fourier transform pairs are used to compute the Fourier or the inverse Fourier transform of a function. Thus, the computational procedure is to express a complicated function of interest in terms of functions with known Fourier (or inverse Fourier) transform and then based on the properties of Fourier transform to compute the Fourier (or inverse Fourier) transform of the complicated function. In the following table, the most common pairs are given. The illustrated Fourier transform pairs are confirmed by using the commands `fouriern` and `ifouriern`.

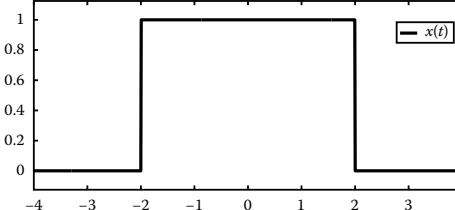
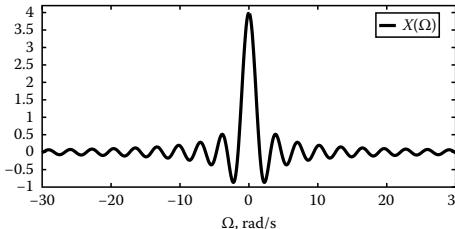
Time Domain	Frequency Domain	Commands	Results
$x(t)$	$X(\Omega)$	<code>syms t w w0 t0</code>	
$\delta(t)$	1	<code>x=dirac(t); fourier(x,w)</code>	<code>ans = 1</code>
1	$2\pi\delta(\Omega)$	<code>fourier(1,w)</code>	<code>ans = 2*pi*dirac(w)</code>
$u(t)$	$(1/j\Omega) + \pi\delta(\Omega)$	<code>X=1/(j*w)+pi*dirac(w); ifourier(X,t)</code>	<code>ans = heaviside(t)</code>
$\delta(t - t_0)$	$e^{-j\Omega t_0}$	<code>x=dirac(t-t0); fourier(x,w)</code>	<code>ans = exp(-i*t0*w)</code>
$e^{j\Omega_0 t}$	$2\pi\delta(\Omega - \Omega_0)$	<code>X=2*pi*dirac(w-w0); ifourier(X,t)</code>	<code>ans = exp(i*w0*t)</code>
$\cos(\Omega_0 t)$	$\pi\delta(\Omega - \Omega_0) + \pi\delta(\Omega + \Omega_0)$	<code>X=pi*(dirac(w-w0)+dirac(w+w0)); ifourier(X,t)</code>	<code>ans = cos(w0*t)</code>
$\sin(\Omega_0 t)$	$(\pi/j)\delta(\Omega - \Omega_0) - (\pi/j)\delta(\Omega + \Omega_0)$	<code>X=(pi/j)*(dirac(w-w0)-dirac(w+w0)); x=ifourier(X,t)</code>	<code>x = sin(w0*t)</code>
$e^{-at}u(t),$ $\text{Re}(a) > 0$	$1/(j\Omega + a)$	<code>a=8; x=exp(-a*t)*heaviside(t); X=fourier(x,w)</code>	<code>X = 1/(8+i*w)</code>
$te^{-at}u(t),$ $\text{Re}(a) > 0$	$1/(j\Omega + a)^2$	<code>x=t*exp(-a*t)*heaviside(t); fourier(x,w)</code>	<code>ans = 1/(8+i*w)^2</code>
$(t^{n-1}/(n-1)!)e^{-at}u(t),$ $\text{Re}(a) > 0$	$1/(j\Omega + a)^n$	<code>n=4; X=1/(j*w+a)^n; ifourier(X,t)</code>	<code>ans = 1/6*t^3 *exp(-8*t) *heaviside(t)</code>

Finally, a very important Fourier transform pair is presented in the next example.

Example

Verify the Fourier transform pair $pT(t) \leftrightarrow T \sin(\Omega T/2)/(\Omega T/2)$, where $pT(t)$ is a rectangular pulse of duration T , given by

$$pT(t) = 1, |t| \leq T/2 \Rightarrow pT(t) = \begin{cases} 1, & -T/2 \leq t \leq T/2 \\ 0, & \text{elsewhere} \end{cases}.$$

Commands	Results	Comments
<pre> syms t w T x=heaviside(t+T/2)- heaviside(t-T/2); xx=subs(x,T,4); ezplot(xx, [-4 4]) legend('x(t)') </pre>		<p>Definition of the rectangular pulse $x(t) = pT(t) = u(t + T/2) - u(t - T/2)$ and graph of $pT(t)$ for $T = 4$, i.e., we plot the signal $p4(t)$.</p>
<pre> X1=fourier(x,w) </pre>	$X1 = 2 * \sin(\frac{1}{2}w * T) / w$	<p>Fourier transform of $pT(t)$.</p>
<pre> X2=T*(sin(w*T/2))/(w*T/2) </pre>	$X2 = 2 * \sin(1/2 * w * T) / w$	<p>The signal $X(\Omega) = T \sin(\Omega T/2) / (\Omega T/2)$ is defined in the proper frequency interval and is equal to the Fourier transform of $x(t)$. Thus, the Fourier transform pair given in the example is confirmed.</p>
<pre> ww=[-30:.1:-1.1.1:1.30]; X=subs(X1,w,ww) X=subs(X,T,4); plot(ww,X) xlabel('\Omega rad/s') legend('X(\Omega)') </pre>		<p>Graph of the Fourier transform $X(\Omega)$ in the frequency domain. Notice that $pT(t)$ is transformed into a $\text{sinc}(\cdot)$ signal in the frequency domain.</p>

6.4 Properties of Fourier Transform

In this section, we present the main properties of the Fourier transform. Each property is verified by an appropriate example.

1. *Linearity.* If $X_1(\Omega) = F\{x_1(t)\}$ and $X_2(\Omega) = F\{x_2(t)\}$, then for any scalars a_1 and a_2 the following property stands:

$$F\{a_1x_1(t) + a_2x_2(t)\} = a_1X_1(\Omega) + a_2X_2(\Omega). \quad (6.9)$$

Commands	Results	Comments
<pre>syms t w a1=3; a2 =4; x1=exp(-3*t)*heaviside(t); Left = (15+10*i*w-3*w^2)/ x2=t*exp(-t)*heaviside(t); (3+i*w)/(1+i*w)^2 Le=a1*x1+a2*x2; Left=fourier(Le) X1=fourier(x1); X2=fourier(x2); Right=a1*X1+a2*X2</pre>	$\text{Right} = \frac{3/(3+i*w) + 4/(1+i*w)^2}{}$	Equation 6.9 is verified for $\alpha_1=3$, $\alpha_2=4$, $x_1(t)=e^{-3t}u(t)$, and $x_2(t)=te^{-t}u(t)$. First, we compute the left side of Equation 6.9 while the right side is computed below.
<pre>simplify(Left -Right)</pre>	$\text{ans} = 0$	The two derived sides in a first sight seem different. However, their difference is zero; hence, the linearity property is verified.

2. Time shifting. If $X(\Omega)=F\{x(t)\}$, then for any t_0 ,

$$F\{x(t - t_0)\} = e^{-j\Omega t_0} X(\Omega). \quad (6.10)$$

Commands	Results	Comments
<pre>syms t w x=cos(t); t0=2; xt0=cos(t-t0); Left=fourier(xt0,w)</pre>	$\text{Left} = \exp(-2*i*w) * \pi * (\text{dirac}(w-1) + \text{dirac}(w+1))$	Equation 6.10 is verified by considering the signal $x(t)=\cos(t)$ and the time shift $t_0=2$. First, the left side of Equation 6.10 is computed.
<pre>X=fourier(x,w); Right=exp(-j*w*t0)*X</pre>	$\text{Right} = \exp(-2*i*w) * \pi * (\text{dirac}(w-1) + \text{dirac}(w+1))$	The right side of Equation 6.10 is computed and is equal to the left; hence, the Fourier transform time-shifting property is confirmed.

3. Frequency shifting. If $X(\Omega)=F\{x(t)\}$, then for any Ω_0 ,

$$F\{e^{j\Omega_0 t}x(t)\} = X(\Omega - \Omega_0). \quad (6.11)$$

Commands	Results	Comments
<pre>syms t w x=t^3; w0=3; Le=exp(j*w0*t)*x; Left=fourier(Le,w)</pre>	$\text{Left} = -2*i*\pi * \text{dirac}(3, w-3)$	The signal $x(t)=t^3$ and the shift $\Omega_0=3$ are considered to confirm (6.11). First, we calculate the left side. The right side is computed below.
<pre>X=fourier(x,w) Right=subs(X,w,w-w0)</pre>	$\text{Right} = -2*i*\pi * \text{dirac}(3, w-3)$	In order to obtain $X(\Omega - \Omega_0)$ we substitute Ω by $\Omega - \Omega_0$ with use of the command subs. The two sides are equal; thus the frequency shifting property is validated.

4. *Scaling in time and frequency.* If $X(\Omega) = F\{x(t)\}$, then for any scalar $b > 0$,

$$F\{x(bt)\} = \frac{1}{|b|} X\left(\frac{\Omega}{b}\right), \quad (6.12)$$

and

$$F\left\{\frac{1}{|b|} x\left(\frac{t}{b}\right)\right\} = X(b\Omega). \quad (6.13)$$

The natural meaning of this property is that if the signal is expanded in time ($b < 1$), then its spectrum is compressed to lower frequencies. On the other hand, a time compression of a signal ($b > 1$) results in an expansion of the signal spectrum to higher frequencies.

Commands	Results	Comments
<pre> syms t w b=3; x=heaviside(t+1)-heaviside(t-1); ezplot(x, [-2 2]); legend('x(t)'); </pre>		<p>The signal $x(t) = u(t+1) - u(t-1)$ and the scalar $b = 3$ are considered to confirm the time-scaling property. The signal $x(t)$ is plotted in this graph while its spectrum $X(\Omega)$ is plotted in the next graph. In this way we can observe the scaling effects that take place in time and in frequency.</p>
<pre> X=fourier(x,w); ezplot(X, [-40 40]); legend('X(\Omega)'); xlabel('\Omega') </pre>		<p>The signal $X(\Omega)$ is plotted in the frequency interval $-40 \leq \Omega \leq 40$ rad/s.</p>
<pre> xb=subs(x,t,b*t); ezplot(xb, [-2 2]); legend('x(bt)', b=3); </pre>		<p>The compressed in time version of $x(t)$. The signal $x(bt)$ is compressed by a factor of 3 compared to $x(t)$.</p>

(continued)

(continued)

Commands	Results	Comments
Xb=fourier(xb,w); ezplot(Xb, [-40 40]) legend('F(x(bt))') xlabel('Omega')		Computation and graph of the left side of (6.12). We notice that the spectrum of the compressed signal is expanded (by a factor of 3) compared to the spectrum of the original signal.
Ri=subs(X,w,w/b); Right=(1/abs(b))*Ri; ezplot(Right, [-40 40]); legend('(1/ b)*X(Omega/b)') xlabel('Omega')		The right side of (6.12) is computed and plotted. The derived graph is same as the one obtained from the left side; hence, the scaling property is confirmed.

Verification of (6.13) is left as a homework problem (see Section 6.10, Problem 7) to the reader.

5. *Time reversal.* If $X(\Omega) = F\{x(t)\}$, then

$$F\{x(-t)\} = X(-\Omega). \quad (6.14)$$

Commands	Results	Comments
x=t*heaviside(t); X=fourier(x,w); Right=i*(-pi*dirac(1,w)*w^2+i)/w^2 Right=subs(X,w,-w)	Right = $i * (-\pi * \text{dirac}(1, w) * w^2 + i) / w^2$	The signal $x(t) = tu(t)$ is considered. The right side of (6.14) is computed.
x_t=subs(x,t,-t); Left=fourier(x_t,w)	Left = $i * (-\pi * \text{dirac}(1, w) * w^2 + i) / w^2$	The left side is computed, and since the two sides are equal, the property given in (6.14) is verified.

6. *Duality.* If $X(\Omega) = F\{x(t)\}$, then

$$F\{X(t)\} = 2\pi x(-\Omega). \quad (6.15)$$

Commands	Results	Comments
<pre>x = exp(-t) * heaviside(t); X = fourier(x) Xt = subs(X, w, t) Left = fourier(Xt)</pre>	<pre>Left = 2*pi*exp(w)*heaviside(-w)</pre>	The left side of (6.15) is evaluated for the signal $x(t) = e^{-t}u(t)$ by first computing the spectrum $X(\Omega)$, then substituting Ω with t to obtain $X(t)$, and finally applying Fourier transform to $X(t)$.
<pre>x_w = subs(x, t, -w); Right = 2*pi*x_w</pre>	<pre>Right = 2*pi*exp(w)*heaviside(-w)</pre>	The right side is computed by substituting in $x(t)$ time t by $-\Omega$. The two sides are same; hence, the duality property is verified.

7. *Conjugation.* If $X(\Omega) = F\{x(t)\}$, then

$$F\{x^*(t)\} = X^*(-\Omega) \quad (6.16)$$

and

$$F\{x^*(-t)\} = X^*(\Omega), \quad (6.17)$$

where $h^*(.)$ denotes the complex conjugate of $h(.)$. We will confirm Equation 6.17 by considering the signal $x(t) = e^{-2t} u(t)$.

Commands	Results	Comments
<pre>x = exp(-2*t) * heaviside(t); x_ = subs(x, t, -t) Left = fourier(x_, w)</pre>	<pre>x_ = exp(2*t) * heaviside(-t) Left = 1/(2-i*w)</pre>	To compute the left side of Equation 6.17 first we define the signal $x(-t)$ by substituting t with $-t$ in $x(t)$. Since $x(-t)$ is real $x^*(-t) = x(-t)$ and the Fourier transform of $x(-t)$ yields the left side of Equation 6.17.
<pre>X = fourier(x, w); Right = conj(X)</pre>	<pre>Right = 1/(2-i*conj(w))</pre>	As for the right side of Equation 6.17, first we compute the Fourier transform $X(\Omega)$ of $x(t)$ and then compute its conjugate. Ω is real, so $\Omega^* = \Omega$; thus the property is verified.

8. *Differentiation in time and frequency.* If $X(\Omega) = F\{x(t)\}$, then

$$F\left\{\frac{dx(t)}{dt}\right\} = j\Omega X(\Omega) \quad (6.18)$$

and

$$F\{tx(t)\} = j\frac{dX(\Omega)}{d\Omega}. \quad (6.19)$$

Commands	Results	Comments
<code>x=exp(-3*t) *heaviside(t); der=diff(x,t); Left=fourier(der,w)</code>	<code>Left = i*w/(3+i*w)</code>	Equations 6.18 and 6.19 are verified by using the signal $x(t)=e^{-3t} u(t)$. The left side of Equation 6.18 is calculated.
<code>X=fourier(x,w); Right=j*w*X</code>	<code>Right = i*w/(3+i*w)</code>	The right side of Equation 6.18 is computed and the two sides are equal. Thus, the differentiation in time property is confirmed.
<code>Left=fourier(t*x,w)</code>	<code>Left = 1/(3+i*w)^2</code>	The left side of Equation 6.19.
<code>der=diff(X,w); Right=j*der</code>	<code>Right = 1/(3+i*w)^2</code>	The right side of Equation 6.19 equals the left one; hence, the differentiation in frequency property is also verified.

9. *Integration.* If $X(\Omega)=F\{x(t)\}$, then

$$F\left\{\int_{-\infty}^t x(\tau)d\tau\right\} = \frac{1}{j\Omega} X(\Omega) + \pi X(0)\delta(\Omega). \quad (6.20)$$

Commands	Results	Comments
<code>syms t r w x=exp(r)*heaviside(-r)+exp(-r)*heaviside(r); integ=int(x,r,-inf,t); Left=fourier(integ,w)</code>	<code>Left=2*pi*dirac(w) -2*i/(1+w^2)/w</code>	Definition of the signal $x(t)=e^t u(-t) + e^{-t} u(t)$
<code>X=fourier(x,w); X0=subs(X,w,0); Right=(1/(j*w))*X+pi*X0*dirac(w)</code>	<code>Right=2*pi*dirac(w) -2*i/(1+w^2)/w</code>	The left side of (6.20).
		The right side of (6.20) is same as the left one; hence, the integration property of the Fourier transform is validated.

10. *Fourier transform of the even and odd parts of a signal.* If $X(\Omega)=F\{x(t)\}$, $x_e(t)$ denotes the even part of $x(t)$ and $x_o(t)$ is the odd part of $x(t)$, then

$$F\{x_e(t)\} = \text{Re}\{X(\Omega)\} \quad (6.21)$$

and

$$F\{x_o(t)\} = j \text{Im}\{X(\Omega)\}. \quad (6.22)$$

In words, the Fourier transform of the even part of a signal equals the real part of the Fourier transform of the signal, while the Fourier transform of the odd part of a signal equals the product of j with the imaginary part of the Fourier transform of the signal.

Commands	Results	Comments
<pre>syms t w x = exp(-t) * heaviside(t); x_t = subs(x, t, -t); xe = 0.5 * (x+x_t)</pre>	$\begin{aligned}x &= \exp(-t) * \text{heaviside}(t) \\x_t &= \exp(t) * \text{heaviside}(-t)\end{aligned}$ $\begin{aligned}xe &= 1/2 * \exp(-t) \\&\quad * \text{heaviside}(t) \\&\quad + 1/2 * \exp(t) * \text{heaviside}(-t)\end{aligned}$	The signal $x(t) = e^{-t}u(t)$ is defined and by substituting t with $-t$ we obtain the signal $x(-t) = e^tu(-t)$.
<pre>xo = 0.5 * (x-x_t);</pre>	$\begin{aligned}xo &= 1/2 * \exp(-t) \\&\quad * \text{heaviside}(t) - 1/2 * \exp(t) \\&\quad * \text{heaviside}(-t)\end{aligned}$	The even part of $x(t)$ is $x_e(t) = [x(t) + x(-t)]/2$.
<pre>xe+xo</pre>	$\text{ans} = \exp(-t) * \text{heaviside}(t)$	The odd part of $x(t)$ is $x_o(t) = [x(t) - x(-t)]/2$.
<pre>X = fourier(x, w)</pre>	$X = 1/(1+w*i)$	We confirm that $x(t) = x_e(t) + x_o(t)$.
<pre>Xe = fourier(xe, w)</pre>	$Xe = 1/(1+w^2)$	Computation of the spectrum $X(\Omega)$.
<pre>Xre = real(X)</pre>	$\begin{aligned}Xre &= 1/2/(1+w*i) + 1/2/ \\&\quad (1-i*conj(w))\end{aligned}$	Left side of (6.21).
<pre>Xre = subs(Xre, conj(w), w)</pre>	$\begin{aligned}Xre &= 1/2/(1-i*w) \\&\quad + 1/2/(1+w*i)\end{aligned}$	Right side of (6.21).
<pre>dif = Xe-Xre simplify(dif)</pre>	$\text{ans} = 0$	Frequency w is real valued, hence, $\text{conj}(w) = w$.
<pre>Xo = fourier(xo, w)</pre>	$Xo = -i*w/(1+w^2)$	The difference of the two sides is zero; thus (6.21) is verified.
<pre>Xim = imag(X)</pre>	$Xim = -1/2*i*(1/(1+w*i) - 1/ \\(1-i*conj(w)))$	Left side of (6.22).
<pre>Xim = subs(Xim, conj(w), w)</pre>	$Xim = -1/2*i*(1/(1+w*i) - 1/ \\(1-i*w))$	Right side of (6.22).
<pre>dif = Xo-j*Xim; simplify(dif)</pre>	$\text{ans} = 0$	Frequency w is real, hence, $\text{conj}(w) = w$.
		The difference of the two sides is zero; thus (6.22) is verified.

6.5 Convolution in Time and Frequency

In this section, we discuss two very useful theorems referring to the convolution between two signals either in the time domain or in the frequency domain. These two theorems allow us to avoid the direct computation of the convolution between two signals. Let $X_1(\Omega) = F\{x_1(t)\}$ denote the Fourier transform of $x_1(t)$, and $X_2(\Omega) = F\{x_2(t)\}$ denote the Fourier transform of $x_2(t)$. Then, the Fourier transform of the convolution of $x_1(t)$ and $x_2(t)$ equals the product of the corresponding Fourier transforms $X_1(\Omega)$ and $X_2(\Omega)$. In other words, the convolution in the time domain is transformed into multiplication in the frequency domain. The mathematical expression is

$$F\{x_1(t) * x_2(t)\} = X_1(\Omega) \cdot X_2(\Omega), \quad (6.23)$$

where the symbol $*$ denotes convolution. Applying inverse Fourier transform in both (left and right) sides of Equation 6.23 yields $F^{-1}\{F\{x_1(t) * x_2(t)\}\} = F^{-1}\{X_1(\Omega) \cdot X_2(\Omega)\}$. Therefore, Equation 6.23 is equivalently written as

$$x_1(t) * x_2(t) = F^{-1}\{X_1(\Omega) \cdot X_2(\Omega)\}. \quad (6.24)$$

Correspondingly, the convolution of two signals in the frequency domain is transformed into multiplication in the time domain. The precise relationship is

$$F\{x_1(t)x_2(t)\} = \frac{1}{2\pi} X_1(\Omega) * X_2(\Omega). \quad (6.25)$$

In order to verify Equation 6.23, it is enough to verify (6.24). The signals $x_1(t) = u(t) - u(t-2)$ and $x_2(t) = u(t) - u(t-4)$ are used for that purpose.

Commands	Results	Comments
<pre> syms t w x1=heaviside(t)-heaviside(t-2); x2=heaviside(t)-heaviside(t-4); X1=fourier(x1,w); X2=fourier(x2,w); right=ifourier(X1*X2,t); ezplot(right,[0 8]); </pre>		Computation and graph of the right side of (6.24).
<pre> t1=0:.01:2; t2=2.01:.01:4; x1=[ones(size(t1)) zeros(size(t2))]; x2=ones(size([t1 t2])); y=conv(x1,x2)*.01; plot(0:.01:8,y); </pre>		Computation and graph of the convolution between $x_1(t)$ and $x_2(t)$ (left side of (6.24)).

The two graphs are identical; hence, the convolution theorem of the Fourier transform is confirmed.

6.6 Symmetry of the Real and Imaginary Parts of Fourier Transform

The Fourier transform $X(\Omega)$ of a signal $x(t)$ is usually a complex function, so it can be written in the form

$$X(\Omega) = \operatorname{Re}\{X(\Omega)\} + j\operatorname{Im}\{X(\Omega)\}. \quad (6.26)$$

Commands	Results	Comments
<code>syms t w x=exp(-2*t)*heaviside(t); X=fourier(x,w)</code>	$x = \exp(-2t) * \text{heaviside}(t)$ $X = 1/(2+w*i)$	The signal $x(t) = e^{-2t}u(t)$ is considered. The Fourier transform $X(\Omega)$ of $x(t)$ (which is the left side of (6.26)) is computed.
<code>Xre=real(X); Xim=imag(X); Right=Xre+j*Xim</code>	$\text{Right} = 1/(2+w*i)$	The right side of (6.26) is equal to the left; hence, the analysis in real and imaginary parts is valid also in the frequency domain.

If $x(t)$ is a real function, the following relationships hold:

$$\text{Re}\{X(-\Omega)\} = \text{Re}\{X(\Omega)\}. \quad (6.27)$$

$$\text{Im}\{X(-\Omega)\} = -\text{Im}\{X(\Omega)\}. \quad (6.28)$$

$$X(-\Omega) = X^*(\Omega). \quad (6.29)$$

These equations state that the real part of the Fourier transform of a real signal is an even function, while the imaginary part of the Fourier transform of a real signal is an odd function.

Commands	Results	Comments
<code>X_w=subs(X,w,-w); Left=real(X_w); Right=real(X);</code>		Computation of the two sides of (6.27).
<code>A=Left-Right; A=subs(A,conj(w),w); simplify(A)</code>	$\text{ans} = 0$	Subtracting the two sides and taking into account that $\text{conj}(w) = w$ validates (6.27).
<code>Left=imag(X_w); Right=-imag(X);</code>		Computation of the two sides of (6.28).
<code>A=Left-Right; A=subs(A,conj(w),w); simplify(A)</code>	$\text{ans} = 0$	Subtracting the two sides and taking into account that $\text{conj}(w) = w$ validates (6.28).
<code>X_w=subs(X,w,-w)</code>	$X_w = 1/(2-i*w)$	The left side of (6.29).
<code>Xc=conj(X); Xc=subs(Xc,conj(w),w)</code>	$Xc = 1/(2-i*w)$	The right side of (6.29) is equal to the left and so (6.29) is also confirmed.

6.7 Parseval's Theorem

Parseval's theorem states that the energy of a continuous-time signal can be computed in the frequency domain from the Fourier transform of the signal. More specifically, the energy of a signal $x(t)$ is computed by

$$E_x = \int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(\Omega)|^2 d\Omega. \quad (6.30)$$

The quantity $|X(\Omega)|^2$ specifies the energy distribution over the frequency spectrum and is called *energy spectral density* of the signal $x(t)$.

Commands	Results	Comments
<pre>syms t w x = exp(-t^2);</pre>		In order to confirm Parseval's theorem the energy of the signal $x(t) = e^{-t^2}$ is computed in both (time and frequency) domains.
<pre>Et = int ((abs(x))^2, t, -inf, inf); eval(Et)</pre> <pre>X=fourier(x,w); Ew=(1/(2*pi))*int((abs(X))^2, w, -inf, inf); ans = 1.2533 eval(Ew)</pre>	<pre>ans = 1.2533</pre>	The energy of $x(t)$ is computed in the time domain according to $\int_{-\infty}^{\infty} x(t) ^2 dt$. The energy of $x(t)$ is computed in the frequency domain according to $(1/2\pi) \int_{-\infty}^{\infty} X(\Omega) ^2 d\Omega$. The result is the same; hence, Parseval's theorem is verified.

6.8 Autocorrelation and Cross-Correlation

In this section, we introduce two very important concepts: autocorrelation and cross-correlation. Suppose that $x(t)$ is an energy-type signal. The autocorrelation function $R_x(\tau)$ of $x(t)$ is defined as

$$R_x(\tau) = \int_{-\infty}^{\infty} x(t)x^*(t - \tau)dt = \int_{-\infty}^{\infty} x(t + \tau)x^*(t)dt. \quad (6.31)$$

Example

Compute the autocorrelation function of the signal $x(t) = e^{-t}u(t - 1)$.

Commands	Results	Comments
<pre>syms t r w x1 = exp(-t)*heaviside(t-1);</pre>		Definition of $x(t)$
<pre>x_2 = subs(x1, t, t-r); x2 = conj(x_2)</pre>		Derivation of $x^*(t - \tau)$.
<pre>R = int (x1*x2, t, -inf, inf) R = simplify(R)</pre>	<pre>R=1/2*exp(-2+r)-1/2*exp(-2+r) *heaviside(r)+1/2*exp(-r-2)* heaviside(r)</pre>	The autocorrelation function is computed according to the first part of (6.31).

We now present some basic properties of the autocorrelation function.

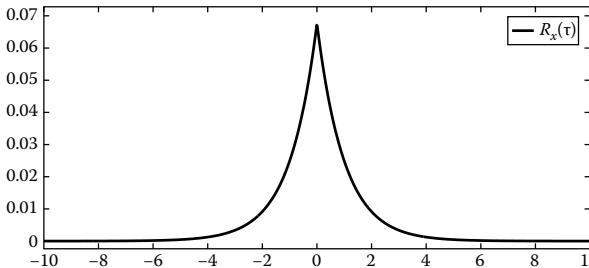
1. The energy of a signal $x(t)$ is equal to its autocorrelation function $R_x(\tau)$ evaluated at $\tau = 0$.

$$R_x(0) = E_x. \quad (6.32)$$

Commands	Results	Comments
R0=limit(R,r,0); eval(R0)	ans = 0.0677	The autocorrelation function is evaluated at $\tau=0$, that is, we compute $R_x(0)$. Notice that the limit command is employed as $R_x(\tau)$ contains the term heaviside(τ) and is not defined at $\tau=0$.
X=fourier(x1,w); Integ= int((abs(X)) ² ,w,-inf,inf); Ew=(1/(2*pi))*Integ; eval(Ew)	ans = 0.0677	The energy E_x of $x(t)$ is computed by using Parseval's theorem, and since $R_x(0)=E_x$ Equation 6.32 is verified.

2. The maximum value of $R_x(\tau)$ is at $\tau=0$.

$$|R_x(\tau)| \leq R_x(0). \quad (6.33)$$

Commands	Results	Comments
ezplot(R, [-10 10]) legend('R_x(\tau)')	 A plot of the autocorrelation function $R_x(\tau)$ versus τ . The x-axis ranges from -10 to 10, and the y-axis ranges from 0 to 0.07. The curve is zero for $\tau < 0$ and $\tau > 0$, with a single sharp peak centered at $\tau = 0$ reaching a maximum value of approximately 0.0677. A legend in the top right corner identifies the curve as $R_x(\tau)$.	Graph of the autocorrelation function $R_x(\tau)$. Indeed $R_x(\tau)$ takes its maximum value for $\tau=0$.

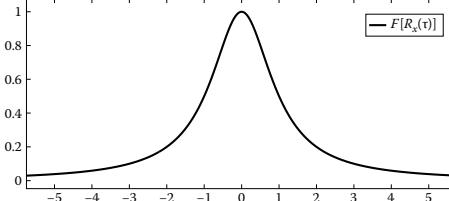
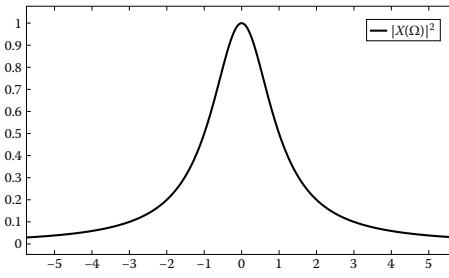
3. The Fourier transform of the autocorrelation function of an energy signal is equal to the energy spectral density of the signal.

$$F\{R_x(\tau)\} = |X(\Omega)|^2. \quad (6.34)$$

Commands	Results	Comments
syms t w r x1=exp(-t)*heaviside(t); x_2=subs(x1,t,t-r); x2=conj(x_2);		This time we consider the signal $x(t)=e^{-t} u(t)$. Derivation of $x^*(t-\tau)$.

(continued)

(continued)

Commands	Results	Comments
<pre>R = int (x1*x2, t,-inf,inf); Left=fourier(R,w); ezplot(Left) legend('F[R_x(\tau)]')</pre>		Computation and graph of the Fourier transform of the autocorrelation function (left part of (6.34)).
<pre>X=fourier(x1,w); Right=abs(X)^2; ezplot(Right) legend(' X(\Omega) ^2')</pre>		Computation and graph of the energy spectral density of x(t) (right part of (6.34)).

In MATLAB, the autocorrelation function of a signal is computed by using the command `xcorr`. The syntax is `R=xcorr(x)*step`, where `step` is the time step used in the definition of the signal $x(t)$. If the length of x is M , the outcome of command `xcorr` (here is the vector R) is of length $2M - 1$. Hence, R must be plotted in the double time interval from that of $x(t)$. More specifically, if $x(t)$ is defined in the time interval $[0, T]$, then its autocorrelation function is plotted in the time interval $[-T, T]$. The command `xcorr` is used in a similar way to the command `conv`.

Example

Compute the autocorrelation function of the signal $x(t) = e^{-t}u(t - 1)$ using the command `xcorr`.

The autocorrelation of $x(t)$ is already computed and plotted in the previous example. Hence, with the use of `xcorr` we will confirm our previous result.

Commands	Results	Comments
<pre>step=0.01; t1=0:step:1-step; x1=zeros(size(t1)); t2=1:step:10; x2=exp(-t2); x=[x1 x2];</pre>		<p>The signal $x(t) = e^{-t} u(t - 1)$, $0 \leq t \leq 10$ is defined as</p> $x(t) = \begin{cases} 0, & 0 \leq t < 1 \\ e^{-t}, & 1 \leq t \leq 10 \end{cases}$

(continued)

Commands	Results	Comments
<pre>R=xcorr(x)*step; plot(-10: step:10,R) legend('R_x(\tau) with xcorr')</pre>		

The obtained graph derived from using the command `xcorr` is the same as the one derived with the analytical computation of the autocorrelation function.

The *cross-correlation* of two signals $x(t)$ and $y(t)$ is a measure of the similarity between $x(t)$ and $y(t)$. The mathematical expression of the cross-correlation is

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} x(t)y * (t - \tau)dt = \int_{-\infty}^{\infty} x(t + \tau)y * (t)dt. \quad (6.35)$$

Example

Compute the cross-correlation of the signals $x(t) = t[u(t) - u(t - 5)]$ and $y(t) = (t - 2)[u(t - 2) - u(t - 7)]$.

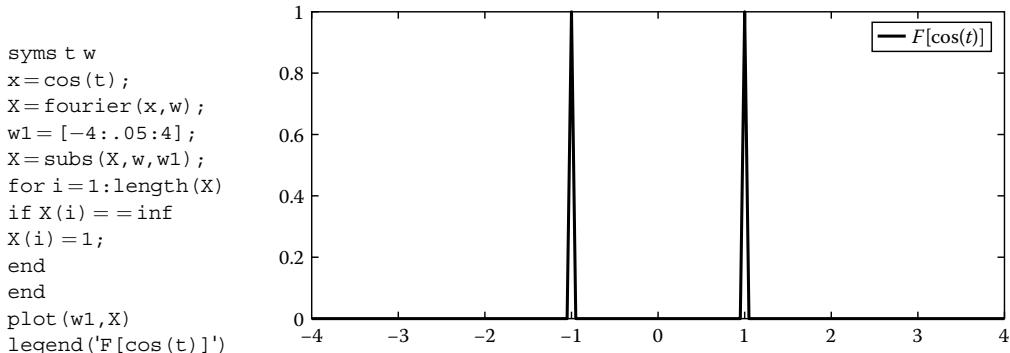
The signal $y(t)$ is a delayed version of $x(t)$ by $\tau = 2$. Hence, we expect the maximum value of $R_{xy}(\tau)$ to be at $\tau = -2$.

Commands	Results	Comments
<pre>t1=0:0.1:5; x1=t1; t2=5.1:0.1:7; x2=zeros(size(t2)); x=[x1 x2]; t1=0:0.1:1.9; y1=zeros(size(t1)); t2=2:0.1:7; y2=t2-2; y=[y1 y2]; R=xcorr(x,y)*0.1; plot(-7:0.1:7,R) legend('R_x_y(\tau)') grid</pre>		<p>Definition of the signals $x(t)$ and $y(t)$ for $0 \leq t \leq 7$ and graph of their cross-correlation. The maximum value of the cross-correlation reveals the delay of the second signal.</p>

6.9 Solved Problems

Problem 1 Plot the Fourier transform of the continuous-time signal $x(t) = \cos(t)$.

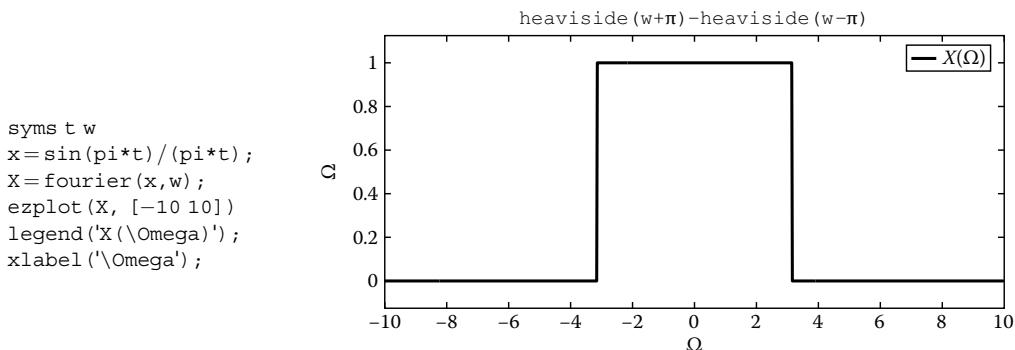
Solution



A for-loop is used in order to substitute $+\infty$ with 1 at the points $\Omega = \pm 1$. This process is done to derive a better graph of the Dirac functions of $X(\Omega)$ at $\Omega = \pm 1$.

Problem 2 Plot the Fourier transform of the continuous-time signal $x(t) = \sin(\pi t)/(\pi t) = \text{sinc}(t)$.

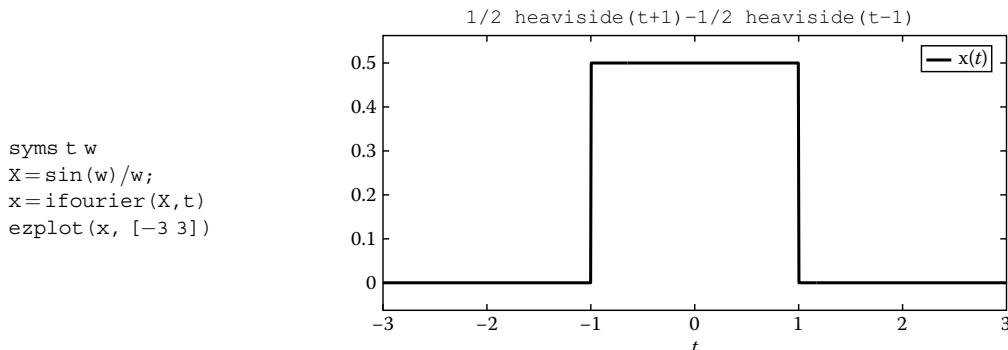
Solution



The Fourier transform of a sinc(.) function is a rectangular pulse. Recall that in Section 6.3 we have established that the Fourier transform of a rectangular pulse is a sinc(.) function.

Problem 3 Plot the inverse Fourier transform of the signal $X(\Omega) = \sin \Omega / \Omega$.

Solution



The inverse Fourier transform of function $\sin \Omega / \Omega$ is a rectangular pulse of amplitude $A = 0.5$.

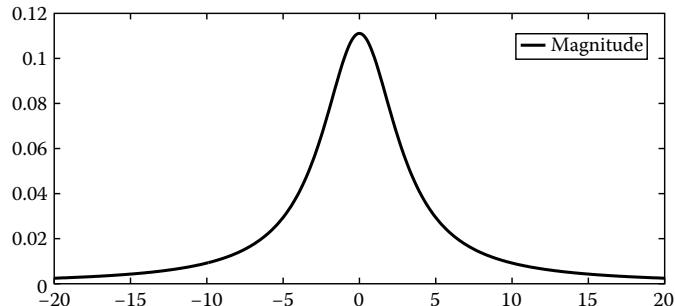
Problem 4 Suppose that a signal $x(t)$ is given by $x(t) = te^{-3t}$. Compute the Fourier transform $X(\Omega)$ of the signal $x(t)$ and plot for $-20 \leq \Omega \leq 20$ rad/s:

- The magnitude of $X(\Omega)$
- The angle of $X(\Omega)$
- The real part of $X(\Omega)$
- The imaginary part of $X(\Omega)$

Solution

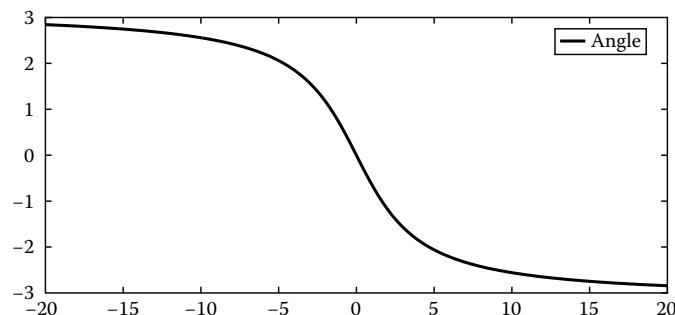
a.

```
syms t w
x=t*exp(-3*t)*heaviside(t);
X=fourier(x,w);
w=-20:.1:20;
X=subs(X,w);
plot(w,abs(X));
legend('magnitude')
```



b.

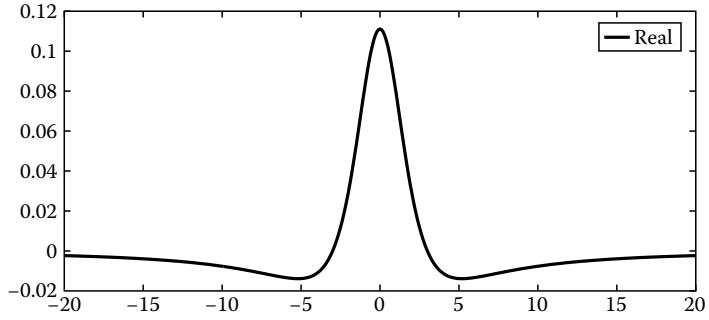
```
plot(w,angle(X));
legend('angle')
```



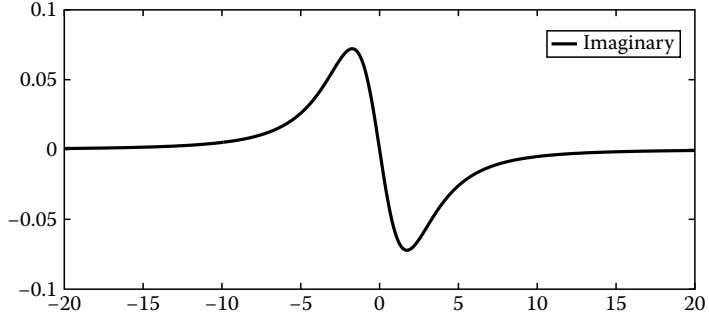
(continued)

(continued)

c.
`plot(w,real(X))
 legend('real')`



d.
`plot(w,imag(X))
 legend('imaginary')`



Problem 5 Verify the Fourier transform properties.

- a. $F\{x(t)\cos(\Omega_0 t)\} = (1/2)[X(\Omega + \Omega_0) + X(\Omega - \Omega_0)]$
 b. $F\{x(t)\sin(\Omega_0 t)\} = (1/2)[X(\Omega + \Omega_0) - X(\Omega - \Omega_0)],$

by using the signal $x(t) = e^{-t} u(t)$ and $\Omega_0 = 3$.

Solution

a.
`syms t w
 x=exp(-t)*heaviside(t);
 w0=3;
 Left=fourier(x*cos(w0*t),w)
 X=fourier(x,w);
 Xw1=subs(X,w,w+w0);
 Xw2=subs(X,w,w-w0);
 Right=0.5*(Xw1+Xw2);
 Right=simplify(Right)`

b.
`Left=fourier(x*sin(w0*t),w)
 Right=(j/2)*(Xw1-Xw2);
 Right=simplify(Right)`

The left side of the first property is computed as

$$\text{Left} = (1+w^2)/(1-3*i+w^2)/(1+3*i+w^2)$$

The right side equals the left; thus the first property is verified.

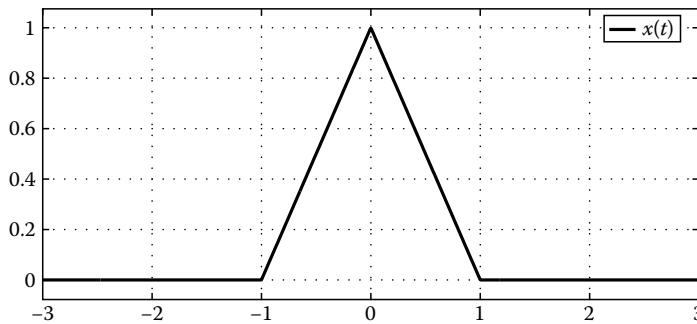
$$\text{Right} = (1+w^2)/(1-3*i+w^2)/(1+3*i+w^2)$$

The two sides of the second property are equal; hence, the second property is also verified.

$$\text{Left} = 3/(1-3*i+w^2)/(1+3*i+w^2)$$

$$\text{Right} = 3/(1+3*i+i*w)/(1-3*i+i*w)$$

Problem 6 Compute and plot the Fourier transform of the signal $x(t)$ that is depicted in the following figure.



Solution

The mathematical expression of $x(t)$ is

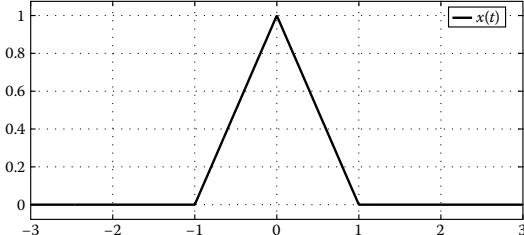
$$x(t) = \begin{cases} t+1, & -1 \leq t \leq 0 \\ -t+1, & 0 \leq t \leq 1 \end{cases}.$$

Next, the signal $x(t)$ is written as a single symbolic expression:

$$x(t) = (t+1)[u(t+1) - u(t)] + (-t+1)[u(t) - u(t-1)].$$

The signal $x(t)$ is plotted according to its symbolic definition for confirmation.

```
syms t w
x = (t+1)*(heaviside(t+1) - heaviside(t)) +
    +(1-t)*(heaviside(t) - heaviside(t-1));
ezplot(x, [-3 3]);
legend('x(t)');
grid
```



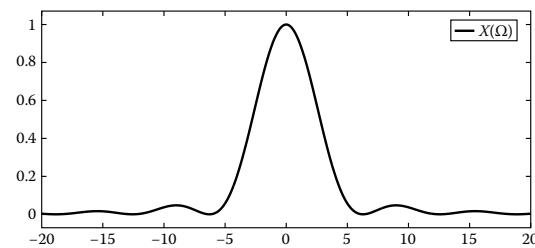
```
X=fourier(x,w)
```

The Fourier transform $X(\Omega)$ of $x(t)$ is

$$X = 4/w^2 * \sin(1/2*w)^2.$$

Graph of $X(\Omega)$.

```
ezplot(X, [-20 20])
legend('X(\rmOmega)')
```



Problem 7 Suppose that $y(x) = xe^{-x} u(x)$. Compute the convolution between the signals $y_1(t) = dy(t)/dt$ and $y_2(t) = \int_{-\infty}^t y(r)dr$.

Solution

The solution is to apply the convolution theorem of the Fourier transform; that is, first the Fourier transforms of $y_1(t)$ and $y_2(t)$ are computed and the inverse Fourier of their product is the result of the convolution between $y_1(t)$ and $y_2(t)$. To compute the Fourier transforms of the signals $y_1(t)$ and $y_2(t)$ we take into account the differentiation and integration properties of Fourier transform.

```
syms x t r w
y=x*exp(-x)*heaviside(x);
Y=fourier(y,w)
Y1=j*w*Y;
Y0=subs(Y,w,0)
Y2=(1/(j*w))*Y+pi*Y0*dirac(w);
```

Convol = ifourier (Y1*Y2,t)

```
ezplot(Convol, [0 10])
legend('Convolution Result')
```

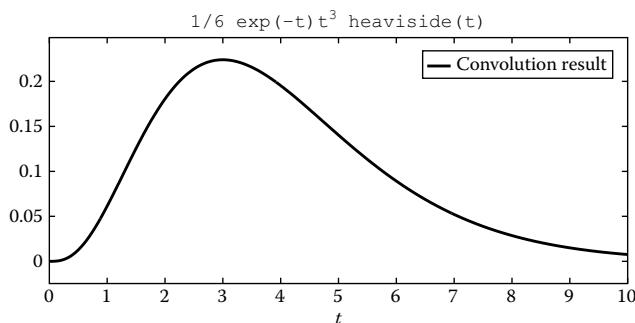
The signal $y(t)$ is defined.

The Fourier transform $Y(\Omega)$.

Computation of $Y_1(\Omega)$ according to the differentiation property.

Computation of $Y_2(\Omega)$ according to the integration property.

Inverse Fourier transform to compute the convolution between $y_1(t)$ and $y_2(t)$.



Problem 8 Verify that the energy of the signal $x(t) = \cos(t)$ is zero.

Solution

```
syms t w
x=cos(t);
Et=int((abs(x))^2,t,-inf,inf)
```

```
X=fourier(x,w);
Ew=(1/(2*pi))*int((abs(X))^2,w,-inf,inf)
```

The energy computation in the time domain is not possible, as the integral $\int_{-\infty}^{\infty} |x(t)|^2 dt$ cannot be computed.

Et = NaN

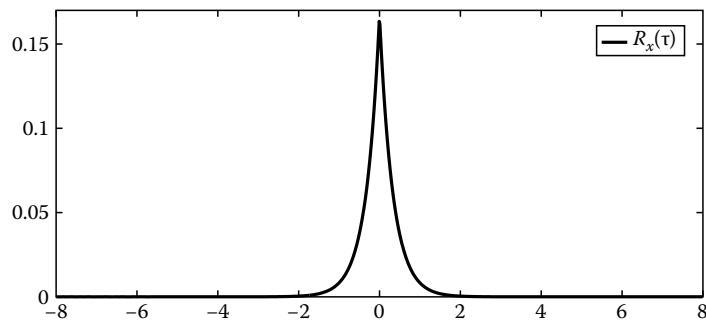
However, applying the Parseval's theorem allows the energy computation in the frequency domain according to $(1/2\pi) \int_{-\infty}^{\infty} |X(\Omega)|^2 d\Omega$.

Ew = 0

Problem 9 Compute and plot the autocorrelation function of the signal $x(t) = e^{-3t} u(t)$.

Solution

```
syms t r
x=exp(-3*t)*heaviside(t);
x1=x;
x_2=subs(x1,t,t-r);
x2=conj(x_2);
R=int(x1*x2,t,-inf,inf);
ezplot(R, [-8 8]);
legend('R_x(\tau)');
ylim([0 0.17]);
```



Problem 10 Consider the signal $x(t) = e^{-3t} u(t)$. Compute and plot the Fourier transform of its autocorrelation function as well as the energy spectral density of $x(t)$.

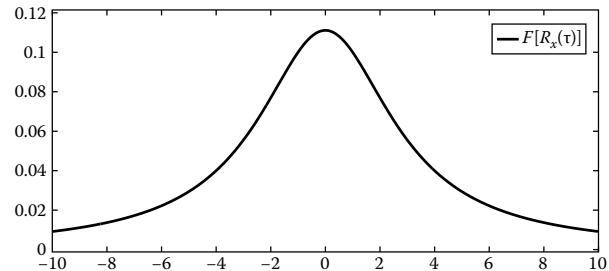
Solution

```
syms t w r
x=exp(-3*t)*heaviside(t);
x1=x;
x_2=subs(x1,t,t-r);
x2=conj(x_2);
R=int(x1*x2,t,-inf,inf)
A=fourier(R,w);
```

We compute the autocorrelation function $R_x(\tau)$ and its Fourier transform $F\{R_x(t)\}$.

Graph of $F\{R_x(t)\}$.

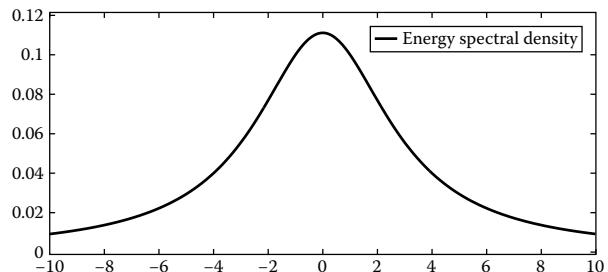
```
ezplot(A, [-10 10])
legend('F[R_x(\tau)]')
```



```
X=fourier(x,w)
B=abs(X)^2;
```

Computation of the energy spectral density of $x(t)$.

```
ezplot(B, [-10 10])
legend('Energy spectral density')
```



6.10 Homework Problems

1. Compute and plot the Fourier transform of the signals

a. $x_1(t) = e^{5t}u(-t)$

b. $x_2(t) = e^{-5t}u(t)$

c. $x_3(t) = te^{5t}u(-t)$

d. $x_4(t) = te^{-5t}u(t)$

2. Compute and plot the inverse Fourier transform of the signal $H(\Omega) = e^{-(j\Omega)/5}$

3. Compute and plot the Fourier transform of the signal

$$x(t) = \begin{cases} 1, & 0 \leq t < 1 \\ 0, & 1 \leq t < 2 \\ 1, & 2 \leq t < 3 \\ 0, & 3 \leq t < 4 \end{cases}$$

4. Compute and plot the Fourier transform of the signal

$$x(t) = \begin{cases} 1, & 0 \leq t < 1 \\ 0, & 1 \leq t < 2 \\ 1, & 2 \leq t < 3 \\ 0, & 3 \leq t < 4 \\ 1, & 4 \leq t < 5 \\ 0, & 5 \leq t < 6 \end{cases}$$

5. Compute and plot the Fourier transform of the signal

$$x(t) = \begin{cases} 1, & 0 \leq t < 0.5 \\ 0, & 0.5 \leq t < 1 \\ 1, & 1 \leq t < 1.5 \\ 0, & 1.5 \leq t < 2 \end{cases}$$

6. Compute and plot the Fourier transform of the signal

$$x(t) = \begin{cases} 1, & 0 \leq t < 0.5 \\ 0, & 0.5 < t < 1 \\ 1, & 1 \leq t < 1.5 \\ 0, & 1.5 \leq t < 2 \\ 1, & 2 \leq t < 2.5 \\ 0, & 2.5 \leq t < 3 \end{cases}$$

7. Verify the frequency scaling property of the Fourier transform.

8. Compute the convolution between the signals $h(t) = e^{-t}u(t)$ and $x(t) = e^{-t}\cos(2\pi t)u(t)$.

9. Compute the energy of the signal $x(t) = e^{-0.1t}\cos(t)u(t)$.

10. Compute the energy of the signal $x(t) = 2e^{-t}u(t)$

a. In the time domain

b. In the frequency domain

c. From its autocorrelation function

11. Plot the autocorrelation of the signal $x(t) = 0.8^t u(t)$.
12. Compute and plot the cross-correlation between the signals:
 - a. $x(t) = te^{-t} u(t)$ and $h(t) = e^{-t} u(t)$
 - b. $x(t) = e^{-t} u(t)$ and $h(t) = e^{-2t} u(t)$
 - c. $x(t) = e^{-t} u(t)$ and $h(t) = e^{-2t} u(t-1)$
13. Verify that $\int_{-\infty}^{\infty} x_1(t)x_2(t)dt = (1/2\pi) \int_{-\infty}^{\infty} X_1^*(\Omega)X_2(\Omega)d\Omega$.
14. Compute and plot the convolution between the signals $x(t) = e^{-t^2}$ and $h(t) = e^{-t} u(t) + e^t u(-t)$.
15. For power-type signals, the time average autocorrelation function $R_x(\tau)$ of a signal $x(t)$ is defined as $R_x(\tau) = \lim_{T \rightarrow \infty} (1/T) \int_{-T/2}^{T/2} x(t)x^*(t - \tau)dt$.
 - a. Compute and plot the time average autocorrelation function of the signal $x(t) = \cos(t)$.
 - b. The power of a signal is computed as $P_x = \lim_{T \rightarrow \infty} (1/T) \int_{-T/2}^{T/2} |x(t)|^2 dt$. Verify that $R_x(0) = P_x$.

This page intentionally left blank

7

Fourier Analysis of Discrete-Time Signals

In Chapter 6, we introduced the Fourier transform of continuous-time signals. In this chapter, we discuss how Fourier transform can be applied to discrete-time signals. There are two types of Fourier transform applicable to discrete-time signals: the *discrete-time Fourier transform* and the *discrete Fourier transform*.

7.1 Discrete-Time Fourier Transform

Discrete-time Fourier transform (DTFT) is the counterpart transform to the continuous-time Fourier transform (CTFT) when dealing with discrete-time signals, i.e., when dealing with signals described by a function $x[n]$, $n \in \mathbb{Z}$. The mathematical expression of the DTFT is

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}. \quad (7.1)$$

The DTFT is a continuous complex-valued function of the cyclic frequency ω . A sufficient condition for the existence of the DTFT $X(\omega)$ of a signal $x[n]$ is

$$\sum_{n=-\infty}^{\infty} |x[n]| < \infty. \quad (7.2)$$

Moreover, the DTFT is always a *periodic* function with period 2π . In order to return from the frequency domain ω back to the discrete-time domain n , we apply inverse DTFT. The mathematical expression of the inverse DTFT is

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega)e^{j\omega n}d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega)e^{j\omega n}d\omega. \quad (7.3)$$

The computation of the DTFT $X(\omega)$ of a signal $x[n]$ is easily obtained directly from (7.1).

Example

Compute the DTFT $X(\omega)$ of the signal $x[n] = 0.8^n$, $0 \leq n \leq 20$ and plot $X(\omega)$ over the frequency intervals $-\pi \leq \omega \leq \pi$ and $-5\pi \leq \omega \leq 5\pi$.

From (7.1) it is obvious that the DTFT of a sequence is a complex-valued function. Hence, we plot both its magnitude and phase.

Commands	Results	Comments
<pre> syms w n = 0:20; x = 0.8.^n; X = sum(x.*exp(-j*w*n)); </pre>		Definition of $x[n] = 0.8^n$, $0 \leq n \leq 20$.
		Computation of the DTFT of $x[n]$ according to (7.1).
<pre> ezplot(abs(X), [-pi pi]) title('Magnitude of DTFT') ylim([0 5.4]) </pre>		Graph of the magnitude of the DTFT $X(\omega)$ for $-\pi \leq \omega \leq \pi$.
<pre> w1 = -pi:.01:pi; XX = subs(X,w,w1); plot(w1,angle(XX)); xlim([-pi pi]) title('Phase of DTFT') </pre>		Graph of the phase of the DTFT $X(\omega)$ for $-\pi \leq \omega \leq \pi$.
<pre> ezplot(abs(X), [-5*pi 5*pi]); title('Magnitude of DTFT in 5 periods') ylim([0 5.8]) </pre>		Graph of the magnitude of the DTFT $X(\omega)$ for $-5\pi \leq \omega \leq 5\pi$.
<pre> w1 = -5*pi:.01:5*pi; XX = subs(X,w,w1); plot(w1,angle(XX)); xlim([-5*pi 5*pi]) title('Phase of DTFT in 5 periods') </pre>		Graph of the phase of the DTFT $X(\omega)$ for $-5\pi \leq \omega \leq 5\pi$.

From the last two graphs it is clear that DTFT is a periodic function with period 2π .

Example

Compute and plot the DTFT of the signal $x[n] = 0.6^n u[n]$.

The signal $x[n]$ is defined over the time interval $[0, +\infty)$ as $x[n] = 0.6^n u[n]$, which is equivalent to $x[n] = 0.6^n$, $n \geq 0$. The length of the sequence $x[n]$ is not finite; hence, the DTFT $X(\omega)$ of $x[n]$ has to be computed with use of the command `symsum`.

Commands	Results	Comments
<pre>syms n w x = 0.6^n</pre>	$x = (3/5)^n$	Definition of $x[n] = 0.6^n u[n]$.
<pre>X = symsum(x*exp(-j*w*n), n, 0, inf)</pre>	$X = 5*\exp(i*w)/(3+5*\exp(i*w))$	The DTFT of $x[n]$ is computed according to (7.1) with use of the command <code>symsum</code> .
<pre>w1 = -pi:.01:pi; X_ = subs(X, w, w1); plot(w1, abs(X_)); legend(' X(\omega) ') xlim([-pi pi])</pre>		Graph of the magnitude of the DTFT $X(\omega)$ for $-\pi \leq \omega \leq \pi$.
<pre>plot(w1, angle(X_)); xlim([-pi pi]) legend('angle X(\omega)')</pre>		Graph of the phase of the DTFT $X(\omega)$ for $-\pi \leq \omega \leq \pi$.

7.2 Properties of Discrete-Time Fourier Transform

In this section, we introduce the main properties of the DTFT. The properties are presented through appropriate examples. Suppose that the DTFT of a signal $x[n]$ is denoted by $X(\omega)$ or alternatively the functions $x[n]$ and $X(\omega)$ form a DTFT pair, that is, $x[n] \leftrightarrow X(\omega)$. The following properties hold for the DTFT:

1. *Linearity.* If $x_1[n] \leftrightarrow X_1(\omega)$, $x_2[n] \leftrightarrow X_2(\omega)$ and a, b scalars, then

$$ax_1[n] + bx_2[n] \leftrightarrow aX_1(\omega) + bX_2(\omega). \quad (7.4)$$

Commands	Results	Comments
<pre> syms w n = 0:10; x1=0.8.^n; x2=0.7.^n; a=3; b=4; L=a*x1+b*x2; Left=sum(L.*exp(-j*w*n)); ezplot(abs(Left), [-pi pi]) title(' DTFT(ax_1[n]+bx_2[n]) ') </pre>		<p>We consider the signals $x_1[n] = 0.8^n$ and $x_2[n] = 0.7^n$, both defined for $0 \leq n \leq 10$ and $a=3, b=4$. The DTFT of the left side of (7.4) is computed straightforwardly.</p>
<pre> w1=-pi:.01:pi; LL=subs(Left,w,w1) plot(w1,angle(LL)); xlim([-pi pi]) title('Phase of DTFT') </pre>		<p>The graph of the magnitude of the DTFT of the left side is depicted above while the phase is illustrated in the left tab.</p>
<pre> X1= sum(x1.*exp(-j*w*n)); X2= sum(x2.*exp(-j*w*n)); Right=a*X1+b*X2; ezplot(abs(Right), [-pi pi]) title('Magnitude of right side') </pre>		<p>The right side of (7.4) is computed and its magnitude is plotted.</p>
<pre> w1=-pi:.01:pi; RR=subs(Right,w,w1) plot(w1,angle(RR)); xlim([-pi pi]) title('Phase of right part') </pre>		<p>Graph of the phase of the right side. The corresponding graphs are identical; hence, the linearity property of DTFT is validated.</p>

2. Time shifting. If $x[n] \leftrightarrow X(\omega)$ and n_0 integer, then

$$x[n - n_0] \leftrightarrow X(\omega)e^{-j\omega n_0}. \quad (7.5)$$

We consider the signal $x[n] = [1, 2, 3, 4]$, $0 \leq n \leq 3$ and $n_0 = 2$. Hence, $x[n - n_0]$ is given by $x[n - 2] = [0, 0, 1, 2, 3, 4]$, $0 \leq n \leq 5$.

Commands	Results	Comments
<pre> syms w n = 0:3; x = [1,2,3,4]; n=0:5; xn0 = [0 0 x]; %$x[n-n_0] = [0,0,1,2,3,4]$ Left = sum(xn0.*exp(-j*w*n)); ezplot(abs(Left), [-pi pi]) title(' DTFT(x[n-n_0]) ') ylim([2 11]); </pre>		Graph of the magnitude of the DTFT of $x[n - n_0]$.
<pre> w1=-pi:.01:pi; L=subs(Left,w,w1); plot(w1,angle(L)); xlim([-pi pi]) title('Phase of DTFT') </pre>		Graph of the phase of the DTFT of $x[n - n_0]$.
<pre> n=0:3; n0=2; x = [1,2,3,4]; X = sum(x.*exp(-j*w*n)); R=X.*exp(-j*w*n0); w1=-pi:.01:pi; R=subs(R,w,w1); plot(w1,abs(R)); title('Magnitude of right part') ylim([2 11]); xlim([-pi pi]) </pre>		The right side of (7.5) is computed and its magnitude is plotted.
<pre> plot(w1,angle(R)); xlim([-pi pi]) title('Phase of right part') xlim([-pi pi]) </pre>		Graph of the phase of the right part. The corresponding graphs are identical; hence, the time-shifting property of DTFT is verified.

3. Frequency shifting. If $x[n] \leftrightarrow X(\omega)$ and ω_0 is real, then

$$x[n]e^{j\omega_0 n} \leftrightarrow X(\omega - \omega_0). \quad (7.6)$$

Commands	Results	Comments
<pre> syms w w0 = 3; n = 0:10; x=0.8.^n; L=x.*exp(j*w0*n); Left = sum(L.*exp(-j*w*n)); ezplot(abs(Left), [-pi pi]) title(' DTFT(x[n]exp(3jn)) ') </pre>	<p>The graph shows the magnitude of the Discrete-Time Fourier Transform (DTFT) of the signal $x[n]e^{j\omega_0 n}$. The x-axis represents frequency ω from -3 to 3, and the y-axis represents magnitude from 0 to 4. The magnitude starts at approximately 4.0 at $\omega = -3$, drops sharply to about 1.5 at $\omega \approx -2.5$, and then exhibits a periodic oscillatory behavior between approximately 1.0 and 1.5 for $\omega > -2.5$.</p>	<p>We consider the signal $x[n] = 0.8^n$, $0 \leq n \leq 10$ and $\omega_0 = 3$. The magnitude of the DTFT of $x[n]e^{j\omega_0 n}$ is depicted in the figure on the left.</p>
<pre> w1=-pi:.01:pi; L=subs(Left,w,w1); plot(w1,angle(L)); xlim([-pi pi]) title('Phase of DTFT') </pre>	<p>The graph shows the phase of the DTFT of $x[n]e^{j\omega_0 n}$. The x-axis represents frequency ω from -3 to 3, and the y-axis represents phase from -1 to 1. The phase starts at approximately -0.8 at $\omega = -3$, crosses zero at $\omega \approx -2.5$, and then exhibits a periodic oscillatory behavior between approximately -0.5 and 0.5 for $\omega > -2.5$.</p>	<p>Graph of the phase of the DTFT of $x[n]e^{j\omega_0 n}$.</p>
<pre> X = sum(x.*exp(-j*w*n)); R = subs(X,w,w-w0); ezplot(abs(R), [-pi pi]); title('Magnitude of right part') </pre>	<p>The graph shows the magnitude of the right side of the equation. The x-axis represents frequency w from -3 to 3, and the y-axis represents magnitude from 0 to 4. The magnitude starts at approximately 4.0 at $w = -3$, drops sharply to about 1.5 at $w \approx -2.5$, and then exhibits a periodic oscillatory behavior between approximately 1.0 and 1.5 for $w > -2.5$.</p>	<p>The right side of (7.6) is computed and its magnitude is plotted.</p>
<pre> w1=-pi:.01:pi; R=subs(R,w,w1); plot(w1,angle(R)); xlim([-pi pi]) title('Phase of right part') </pre>	<p>The graph shows the phase of the right side. The x-axis represents frequency w from -3 to 3, and the y-axis represents phase from -1 to 1. The phase starts at approximately -0.8 at $w = -3$, crosses zero at $w \approx -2.5$, and then exhibits a periodic oscillatory behavior between approximately -0.5 and 0.5 for $w > -2.5$.</p>	<p>Graph of the phase of the right side. The corresponding graphs are identical; hence, the frequency-shifting property of DTFT is verified.</p>

4. Time reversal. If $x[n] \leftrightarrow X(\omega)$, then

$$x[-n] \leftrightarrow X(-\omega). \quad (7.7)$$

Commands	Results	Comments
<pre> syms w n = -5:0; x=-n Left = sum(x.*exp(-j*w*n)); ezplot(abs(Left), [-pi pi]) title(' DTFT(x[-n]) ') </pre>		<p>We consider the signal $x[n] = n$, $0 \leq n \leq 5$. Thus, $x[-n] = -n$, $-5 \leq n \leq 0$. The magnitude of the DTFT of $x[-n]$ is depicted in the figure on the left.</p>
<pre> w1=-pi:.01:pi; LL=subs(Left,w,w1) plot(w1,angle(LL)); xlim([-pi pi]) title('\angle DTFT(x[-n])') </pre>		<p>Graph of the phase of the DTFT of $x[-n]$.</p>
<pre> n=0:5; x=n; X = sum(x.*exp(-j*w*n)); Right=subs(X,w,-w); ezplot(abs(Right), [-pi pi]) title('Magnitude of right part') </pre>		<p>In order to compute the right side of (7.7), first we derive the DTFT $X(\omega)$ of $x[n]$ and then substitute ω by $-\omega$ to obtain $X(-\omega)$.</p>
<pre> w1=-pi:.01:pi; RR=subs(Right,w,w1) plot(w1,angle(RR)); xlim([-pi pi]) title('Phase of right part') </pre>		<p>Graph of the phase of $X(-\omega)$. The corresponding graphs are identical; hence, the time-reversal property of DTFT is confirmed.</p>

5. *Differentiation in frequency.* If $x[n] \leftrightarrow X(\omega)$, then

$$\frac{n}{j} x[n] \leftrightarrow \frac{dX(\omega)}{d\omega}. \quad (7.8)$$

To verify the differentiation in frequency property, we will use the DTFT pair $a^n u[n] \leftrightarrow \frac{1}{1-ae^{-j\omega}}$, $|a| < 1$.

Commands

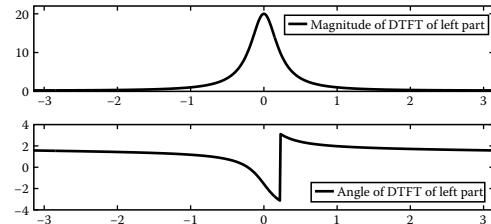
```

syms n
x = 0.8.^n
L = symsum(n/j*x*exp(-j*w*n), n, 0, inf);
subplot(211);
w1 = -pi:.01:pi;
Li = subs(L,w,w1);
plot(w1,abs(Li));
legend('magnitude of DTFT of left part')
subplot(212);
plot(w1,angle(Li))
legend('angle of DTFT of left part')

```

Results/Comments

We consider the signal $x[n] = 0.8^n u[n]$. Thus, its DTFT is computed by $X(\omega) = \sum_{n=0}^{\infty} 0.8^n e^{-j\omega n}$. Notice that we use the command `symsum` to compute the DTFT of $(n/j)x[n]$.

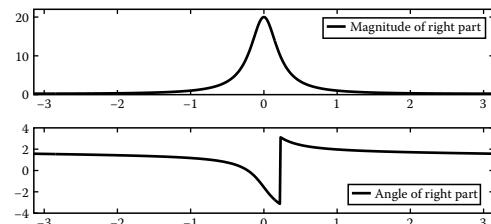


```

syms n w
X = 1/(1-0.8*exp(-j*w));
R = diff(X,w);
subplot(211);
w1 = -pi:.01:pi;
Ri = subs(R,w,w1);
plot(w1,abs(Ri));
legend('magnitude of right part')
subplot(212);
plot(w1,angle(Ri))
legend('angle of right part')

```

The DTFT of $x[n] = 0.8^n u[n]$ is $X(\omega) = 1/(1 - 0.8e^{-j\omega})$. We compute and plot the derivative of $X(\omega)$, and since the two graphs are same the differentiation in frequency property is verified.



6. Time difference. If $x[n] \leftrightarrow X(\omega)$, then

$$x[n] - x[n-1] \leftrightarrow (1 - e^{-j\omega})X(\omega). \quad (7.9)$$

Commands

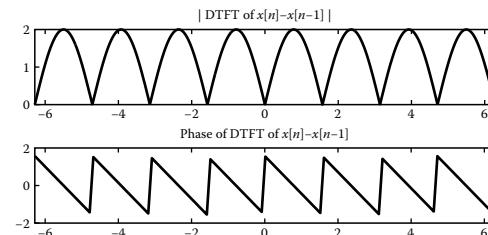
```

syms w
n = 0:4;
x = [1, 1, 1, 1, 0];
x1 = [0, 1, 1, 1, 1];
L = x-x1;
Left = sum(L.*exp(-j*w*n));
subplot(211);
ezplot(abs(Left));
title('| DTFT of x[n]-x[n-1] |')
w1 = -2*pi:.1:2*pi;
Left = subs(Left,w,w1);
subplot(212);
plot(w1,angle(Left));
title('Phase of DTFT of x[n]-x[n-1]')
xlim([-2*pi 2*pi])

```

Results/Comments

We consider the signal $x[n] = u[n] - u[n-4]$. Hence, the delayed signal $x[n-1]$ is $x[n-1] = u[n-1] - u[n-5]$. Both signals are defined for $0 \leq n \leq 4$. The DTFT of $x[n] - x[n-1]$ is plotted for $-2\pi \leq \omega \leq 2\pi$.



(continued)

Commands	Results/Comments
<pre>X = sum(x.*exp(-j*w*n)); Right = (1-exp(-j*w))*X; subplot(211); ezplot(abs(Right)); title(' (1-exp(jw))*X(w) '); subplot(212); Right = subs(Right,w,w1); plot(w1,angle(Right)); title('\angle (1-exp(jw))*X(w)'); xlim([-2*pi 2*pi])</pre>	<p>Computation of the DTFT of $x[n]$ and the graph of $(1-e^{j\omega})X(\omega)$, $-2\pi \leq \omega \leq 2\pi$. The two graphs are identical; hence, the time-difference property of DTFT is confirmed.</p>

7. Convolution. If $x[n] \leftrightarrow X(\omega)$ and $h[n] \leftrightarrow H(\omega)$, then

$$x[n]*h[n] \leftrightarrow X(\omega)Y(\omega), \quad (7.10)$$

where the symbol * denotes convolution. This property was established in the continuous-time case but is valid also for discrete-time signals.

Commands	Results/Comments
<pre>n=0:5; x=1./(n+1); h=0.9.^n; y=conv(x,h); n1=0:10; Y = sum(y.*exp(-j*w*n1)); subplot(211); ezplot(abs(Y), [-pi pi]); title('Magnitude of Y(\omega)'); subplot(212); w1=-pi:.01:pi; Y1=subs(Y,w,w1); plot(w1,angle(Y1)); title('Angle of Y(\omega)'); xlim([-pi pi])</pre> <pre>xx=[x 0 0 0 0]; hh=[h 0 0 0 0]; X= sum(xx.*exp(-j*w*n1)); H= sum(hh.*exp(-j*w*n1)); Y=X.*H; subplot(211); ezplot(abs(Y), [-pi pi]); title('Magnitude of X(\omega)H(\omega)'); subplot(212); Y1=subs(Y,w,w1); plot(w1,angle(Y1)); title('Angle of X(\omega)H(\omega)'); xlim([-pi pi])</pre>	<p>We consider the signals $x[n] = 1/(n+1)$ and $h[n] = 0.9^n$, both defined for $0 \leq n \leq 5$. We compute and plot the DTFT $Y(\omega)$ of their convolution $y[n] = x[n]*h[n]$. Notice that n in the DTFT calculation is defined as $0 \leq n \leq 10$, since the length of $y[n]$ is 11 samples.</p> <p>The signals $x[n]$ and $y[n]$ are zero padded in order to have length of 11 samples. The DTFTs $X(\omega)$ and $Y(\omega)$ are computed from the zero-padded signals, and since the graph of $X(\omega)Y(\omega)$ is same as the one obtained from the left side of (7.10) the convolution property is confirmed.</p>

7.3 Parseval's Theorem for Discrete-Time Fourier Transform

Parseval's theorem was established in Chapter 6 for continuous-time signals. Parseval's theorem is also valid for discrete-time signals. It states that the energy of a discrete-time signal $x[n]$ is computed in the frequency domain according to

$$E_x = \sum_{n=-\infty}^{\infty} |x[n]|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(\omega)|^2 d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(\omega)|^2 d\omega, \quad (7.11)$$

where

$X(\omega)$ is the DTFT of $x[n]$

$|X(\omega)|^2$ is the *energy spectral density* of the signal $x[n]$

To verify Parseval's theorem we will use the DTFT pair $a^n u[n] \leftrightarrow 1/(1 - ae^{-j\omega})$, $|a| < 1$.

Commands	Results	Comments
<pre> syms n w a = 0.6; x = a^n; E = symsum(abs(x)^2, n, 0, inf) E = eval(E) X = 1/(1-a*exp(-j*w)); EsD = abs(X)^2 E = 1/(2*pi)*int(EsD, w, -pi, pi) E = eval(E) </pre>	<pre>E = 1.5625</pre>	<p>The energy of the signal $x[n] = 0.6^n u[n]$ is computed in the time domain according to $\sum_{n=-\infty}^{\infty} x[n] ^2$.</p>

7.4 Discrete Fourier Transform

In order to implement DTFT analysis of a signal on a computer, it is necessary to obtain samples ω_k from the frequency ω . This leads to a second type of Fourier transform appropriate for discrete-time signals, the discrete Fourier transform (DFT). The N -point DFT of a discrete-time signal $x[n]$ is denoted by X_k or $X(k)$ or $X(\omega_k)$. It is defined in the discrete-time interval $0 \leq n \leq N - 1$, and is computed according to

$$X_k = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi n k}{N}}, \quad k = 0, 1, \dots, N - 1. \quad (7.12)$$

The DFT X_k of a sequence $x[n]$ is a function of k and is completely specified by its values for $k = 0, 1, \dots, N - 1$, that is, from the N values X_1, X_2, \dots, X_{N-1} . Typically, X_k are complex numbers. Hence, they can be expressed in polar form as

$$X_k = |X_k| e^{j \angle X_k}, \quad k = 0, 1, \dots, N - 1, \quad (7.13)$$

where

$|X_k|$ is the magnitude of X_k

$\angle X_k$ is the phase of X_k

Alternatively, X_k is expressed as

$$X_k = \operatorname{Re}\{X_k\} + j \operatorname{Im}\{X_k\}, \quad k = 0, 1, \dots, N-1, \quad (7.14)$$

where $\operatorname{Re}\{X_k\}$ is the real part of X_k and is given by

$$\operatorname{Re}\{X_k\} = x(0) + \sum_{n=1}^{N-1} x[n] \cos\left(\frac{2\pi nk}{N}\right) = \sum_{n=0}^{N-1} x[n] \cos\left(\frac{2\pi nk}{N}\right), \quad (7.15)$$

where the second equality is derived from the fact that $x(0)\cos(0) = x(0)$. The imaginary part $\operatorname{Im}\{X_k\}$ of X_k is given by

$$\operatorname{Im}\{X_k\} = -\sum_{n=0}^{N-1} x[n] \sin\left(\frac{2\pi nk}{N}\right). \quad (7.16)$$

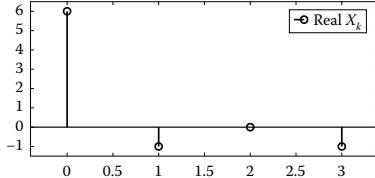
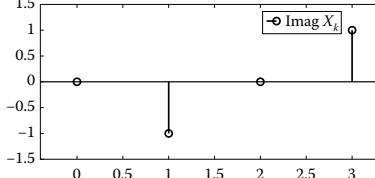
Example

Compute the DFT X_k of the sequence $x[n] = [1, 2, 2, 1]$, $0 \leq n \leq 3$. Plot the magnitude, the phase, the real part, and the imaginary part of X_k . Finally, verify Equations 7.13 through 7.16.

Commands	Results	Comments
<pre> x = [1 2 2 1]; N = length(x); for k = 0:N-1 for n = 0:N-1 X(n+1) = x(n+1) * exp(-j*2*pi*k*n/N); end Xk(k+1) = sum(X); end Xk </pre>	<pre> Xk = 6.0000 -1.0000 - 1.0000i 0 - 0.0000i -1.0000 + 1.0000i </pre>	The DFT X_k of the sequence $x[n]$ is computed according to Equation 7.12. Notice that X_k is a complex-valued sequence.
<pre> mag = abs(Xk); stem(0:N-1, mag); legend ('\ X_k ') xlim([-0.5 3.5]); ylim([-0.5 6.5]); </pre>		Graph of the magnitude $ X_k $ of the DFT X_k .
<pre> phas = angle(Xk); stem(0:N-1, phas); legend ('\angle X_k') xlim([-0.4 3.4]); ylim([-3.5 3]); </pre>		Graph of the phase $\angle X_k$ of the DFT X_k .

(continued)

(continued)

Commands	Results	Comments
<pre>mag.*exp(j*phas)</pre>	<pre>ans = 6.0000 -1.0000 - 1.0000i 0.0000 - 0.0000i -1.0000 + 1.0000i</pre>	Confirmation of Equation 7.13.
<pre>re=real(Xk); stem(0:N-1,re); xlim([- .4 3.4]); ylim([-1.5 6.5]); legend ('real Xk')</pre>		Graph of the real part of X_k .
<pre>im=imag(Xk); stem(0:N-1,im); xlim([- .4 3.4]); ylim([-1.5 1.5]); legend ('imag Xk')</pre>		Graph of the imaginary part of X_k .
<pre>re+j*im for k=0:N-1 for n=0:N-1 Xre(n+1)=x(n+1)*cos(2*pi*k*n/N); end Xr(k+1) = sum(Xre); end Xr Xim=zeros(size(x)) N=length(x); for k=1:N-1 for n=0:N-1 Xima(n+1)=x(n+1)*sin(2*pi*k*n/N); end Xim(k+1) = -sum(Xima); end Xim</pre>	<pre>ans = 6.0000 -1.0000 - 1.0000i 0.0000 - 0.0000i -1.0000 + 1.0000i</pre> <pre>Xr = 6.0000 -1.0000 0 -1.0000</pre> <pre>Xim = 0 -1.0000 -0.0000 1.0000</pre>	Equation 7.14 is verified as $\text{Re}[X_k] + j\text{Im}[X_k]$ is equal to the computed DFT X_k . The real part of X_k is computed according to Equation 7.15, and since it is same as the one derived from X_k , Equation 7.15 is also verified. The imaginary part of X_k is computed according to Equation 7.16, and since it is same as the one derived from X_k , Equation 7.16 is also confirmed.

Example

Write a function that computes the DFT X_k of a sequence $x[n]$. Compute through your function the DFT of the discrete-time signal $x[n] = [1, -2, 2, 1], 0 \leq n \leq 3$.

Commands	Results/Comments
<pre>function Xk=dft(x); N=length(x); for k=0:N-1 for n=0:N-1 X(n+1)=x(n+1)*exp(-j*2*pi*k*n/N); end Xk(k+1)=sum(X); end x=[1 -2 2 1]; Xk=dft(x)</pre>	<p>The function <i>dft.m</i> is based on the code written in the previous example. The sequence $x[n]$ is the input argument, while the DFT X_k is the output argument of the function.</p> <p>The function <i>dft.m</i> is executed from the command prompt. The DFT X_k of $x[n]$ is</p> $\begin{aligned} Xk &= 2.0000 \quad -1.0000 + 3.0000i \\ &\quad 4.0000 + 0.0000i \quad -1.0000 - 3.0000i \end{aligned}$

7.5 Properties of Discrete Fourier Transform

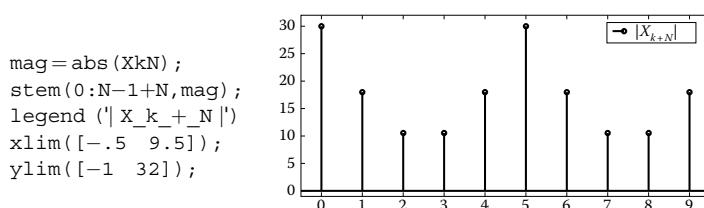
In this section, we present the basic properties of DFT.

1. *Periodicity.* The N -point DFT sequence X_k of a sequence $x[n]$ is periodic with period N . The mathematical expression is

$$X_{k+N} = X_k. \quad (7.17)$$

To verify the periodicity of DFT we consider the signal $x[n] = n^2$, $0 \leq n \leq 4$. In this case $N = 5$.

Commands	Results	Comments
<pre>n=0:4; x=n.^2; X=dft(x)</pre>	$\begin{aligned} X &= 30.0000 \quad -5.2639 + 17.2048i \\ &\quad -9.7361 + 4.0615i \quad -9.7361 - 4.0615i \\ &\quad -5.2639 - 17.2048i \end{aligned}$	<p>The DFT of the sequence $x[n]$ is computed via the function <i>dft.m</i> that was created in the previous example.</p>
<pre>N=length(x); for k=0:N-1+N for n=0:N-1 X(n+1)=x(n+1)*exp(-j*2*pi*k*n/N); end XkN(k+1)=sum(X); end XkN</pre>	$\begin{aligned} XkN &= 30.0000 \quad -5.2639 + 17.2048i \\ &\quad -9.7361 + 4.0615i \quad -9.7361 - 4.0615i \\ &\quad -5.2639 - 17.2048i \quad 30.0000 + 0.0000i \\ &\quad -5.2639 + 17.2048i \quad -9.7361 + 4.0615i \\ &\quad -9.7361 - 4.0615i \quad -5.2639 - 17.2048i \end{aligned}$	<p>In order to compute X_{k+N}, we define k from 0 to $2N - 1$. It is clear that X_{k+N} is periodic with period $N = 5$.</p>



Magnitude of X_{k+N} .

(continued)

(continued)

Commands	Results	Comments
<pre>phas = angle(XkN); stem(0:N-1+N, phas); legend ('\angle X_k+_N'); xlim([- .5 9.5]);</pre>		Phase of X_{k+N} .

2. *Linearity.* If the DFTs of the sequences $x_1[n]$ and $x_2[n]$ are denoted by $X_1(k)$ and $X_2(k)$, respectively, then for any scalars a, b

$$\text{DFT}\{ax_1[n] + bx_2[n]\} = aX_1(k) + bX_2(k). \quad (7.18)$$

Commands	Results	Comments
<pre>x1 = [1 2 3 4 5]; x2 = [1 3 5 2 1]; a = 3; b = 4;</pre>		The sequences $x_1[n] = [1, 2, 3, 4, 5]$ and $x_2[n] = [1, 3, 5, 2, 1]$ are considered. Moreover, we set $a = 3$ and $b = 4$.
<pre>Left = dft(a*x1+b*x2)</pre>	<pre>Left = 93.0000 -21.2082 + 4.3390i -7.7918 - 9.1473i -7.7918 + 9.1473i -21.2082 - 4.3390i</pre>	The left side of (7.18) is computed through the function <i>dft.m</i> .
<pre>X1 = dft(x1); X2 = dft(x2); Right = a*X1+b*X2</pre>	<pre>Right = 93.0000 -21.2082 + 4.3390i -7.7918 - 9.1473i -7.7918 + 9.1473i -21.2082 - 4.3390i</pre>	The right side of (7.18) is equal to the left; hence, the linearity property of the DFT is confirmed.

3. *Symmetry.* If X_k is the DFT of $x[n]$, then

$$X_k = X_{((N-k)_N)}^*, \quad 0 \leq k \leq N - 1, \quad (7.19)$$

where

 X_k^* denotes the complex conjugate of X_k $X_{((N-k)_N)}$ is a circularly shifted version of X_k $((N-k)_N) \equiv (N - k) \bmod N$

The circular shift of a sequence is discussed in details in Section 7.7. From Equation 7.19 we conclude that if N is odd then $|X_k|$ is even symmetric about $k = N/2$ and $\angle X_k$ is odd symmetric about $k = N/2$.

Commands	Results	Comments
<code>x = [1 2 3 4 5]; Xk = dft(x)</code>	$Xk = 15.0000 \quad -2.5000 + 3.4410i$ $\quad \quad \quad -2.5000 + 0.8123i \quad -2.5000 - 0.8123i$ $\quad \quad \quad -2.5000 - 3.4410i$	Definition of $x[n] = [1, 2, 3, 4, 5]$, $0 \leq n \leq 4$ and computation of the DFT X_k of $x[n]$.
<code>k = 0 : 4 N = 5 R = Xk(1+mod(N-k, N)) Xnk = conj(R)</code>	$Xnk = 15.0000 \quad -2.5000 + 3.4410i$ $\quad \quad \quad -2.5000 + 0.8123i \quad -2.5000 - 0.8123i$ $\quad \quad \quad -2.5000 - 3.4410i$	The right part of Equation 7.19 is computed, and since it is equal to X_k , Equation 7.19 is verified.

4. *Parseval's theorem for DFT:* Parseval's theorem states that the energy of a discrete-time signal $x[n]$ can be computed from the DFT X_k of $x[n]$ according to

$$E = \frac{1}{N} \sum_{k=0}^{N-1} |X_k|^2. \quad (7.20)$$

We consider the signal $x[n] = 1/(n+1)$, $0 \leq n \leq 10$. Recall that the energy of a discrete-time signal is calculated according to $E = \sum_{n=0}^{N-1} |x[n]|^2$.

Commands	Results	Comments
<code>n = 0 : 10; x = 1 ./ (n+1); En = sum(abs(x) .^ 2)</code>	$En = 1.5580$	Energy computed at the discrete-time domain.
<code>N = length(x); X = dft(x); Edft = (1/N) * sum(abs(X) .^ 2)</code>	$Edft = 1.5580$	Energy computed according to Parseval's theorem from the DFT of $x[n]$.

7.6 Inverse Discrete Fourier Transform

Suppose that the DFT X_k of a discrete-time signal $x[n]$ is known. The signal $x[n]$ can be derived from the N DFT points X_k , $k = 1, 2, \dots, N - 1$ by applying the *inverse discrete Fourier transform (IDFT)*. The IDFT of a sequence X_k is given by

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j \frac{2\pi n k}{N}}, \quad n = 0, 1, \dots, N - 1. \quad (7.21)$$

Example

Compute the IDFT of the sequence $X_k = [6, -1 - j, 0, -1 + j]$, $0 \leq k \leq 3$.

Commands	Results	Comments
Xk = [6, -1-j, 0, -1+j]; N = length(Xk); for n=0:N-1 for k=0:N-1 xn(k+1) = Xk(k+1)*exp(j*2*pi*n*k/N); end x(n+1) = sum(xn); end x = (1/N)*x	x = 1.0000 2.0000 - 0.0000i 2.0000 - 0.0000i 1.0000 + 0.0000i	Definition of X_k . The IDFT of X_k is computed according to Equation 7.21. The result is the discrete-time signal $x[n] = [1, 2, 2, 1]$, $0 \leq n \leq 3$.

Example

Write a function that computes the IDFT of a sequence X_k and compute through your function the IDFT of the sequence $X_k = [6, -1-j, 0, -1+j], 0 \leq k \leq 3$.

Commands	Results	Comments
function x=idft(Xk) N=length(Xk); for n=0:N-1 for k=0:N-1 xn(k+1) = Xk(k+1)*exp(j*2*pi*n*k/N); end x(n+1) = sum(xn); end x = (1/N)*x; Xk = [6, -1-j, 0, -1+j]; x = idft(Xk)	x = 1.0000 2.0000 - 0.0000i 2.0000 - 0.0000i 1.0000 + 0.0000i	The function <i>idft.m</i> is based on the code written in the previous example. The sequence X_k is the input argument, while the IDFT of X_k is the output argument of the function. The function <i>idft.m</i> is executed from the command prompt and the IDFT $x[n]$ of X_k is computed.

7.7 Circular Shift of a Sequence

In this section, we introduce the concept of a circular shift of a sequence. When a sequence $x[n]$ defined over the interval $[0, N-1]$ is circularly shifted by m samples (or elements), the outcome is again a sequence $x_{c,m}[n]$ defined over the same interval $[0, N-1]$, where the first $N-m$ samples of $x[n]$, i.e., the samples $x[0]$ to $x[N-m-1]$ are shifted by m samples to the right, while the rest m samples, i.e., the samples $x[N-m]$ to $x[N-1]$ are placed at the beginning of the sequence, taking over the first m positions. The circular shifted sequence $x_{c,m}[n]$ is mathematically described by the following relationship:

$$x_{c,m}[n] = x[((n-m))_N], \quad (7.22)$$

where $((n-m))_N \equiv (n-m) \text{ modulo } N$.

In order to completely understand the concept of a circular shift, consider it as a clockwise rotation of elements placed in a circle. In Figure 7.1a circularly shifted by two samples sequence is depicted.

Example

Compute the circularly shifted sequence $x_{c,m}[n]$ for $m=2$ if $x[n]=[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8]$, $0 \leq n \leq 7$.

Commands	Results	Comments
$x = [.1, .2, .3, .4, .5, .6, .7, .8]$ $N = 8;$ $m = 2;$ x'	ans = 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000	Definition of $x[n]$, N , and m . For illustration purposes, $x[n]$ is depicted as column vector.
$n = 0;$ $p = \text{mod}((n-m), N)$	p = 6	We compute the position of the element of $x[n]$ that will be placed first in the circularly shifted sequence $x_{c,2}[n]$. The position is given by $(n - m) \bmod N$.
$xc(n+1) = x(p+1);$ $xcm = xc'$	ans = 0.7000	The first element $x_{c,2}[0]$ of the circularly shifted sequence $x_{c,2}[n]$ is the seventh element of $x[n]$, namely, is the element $x[6] = 0.7$. But as MATLAB® does not support zero index, the sample with value 0.7 is the sample with index 7 of the defined in MATLAB vector x .
$n = 1;$ $p = \text{mod}((n-m), N)$ $xc(n+1) = x(p+1);$ $xcm = xc'$	p = 7 xcm = 0.7000 0.8000	The second element $x_{c,2}[1]$ of $x_{c,2}[n]$ is the eighth element of $x[n]$, namely, is the element $x[7] = 0.8$.
$n = 2;$ $p = \text{mod}((n-m), N)$ $xc(n+1) = x(p+1);$ $xcm = xc'$	p = 0 xcm = 0.7000 0.8000 0.1000	The third element $x_{c,2}[2]$ of $x_{c,2}[n]$ is the first element of $x[n]$, namely, is the element $x[0] = 0.1$.
$n = 3;$ $p = \text{mod}((n-m), N)$ $xc(n+1) = x(p+1);$ $xcm = xc'$	p = 1 xcm = 0.7000 0.8000 0.1000 0.2000	The fourth element $x_{c,2}[3]$ of $x_{c,2}[n]$ is the second element of $x[n]$, namely, is the element $x[1] = 0.2$.
$n = 4;$ $p = \text{mod}((n-m), N)$ $xc(n+1) = x(p+1);$ $xcm = xc'$	p = 2 xcm = 0.7000 0.8000 0.1000 0.2000 0.3000	The fifth element $x_{c,2}[4]$ of $x_{c,2}[n]$ is the third element of $x[n]$, namely, is the element $x[2] = 0.3$.

(continued)

(continued)

Commands	Results	Comments
<pre> xcm = 0.7000 for n=5:N-1 p=mod((n-m) , N); xc(n+1)=x(p+1); end xcm=xc'</pre>	<pre> 0.8000 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000</pre>	Deriving the rest three elements of $x_{c,2}[n]$ in the same way yields the circularly shifted sequence $x_{c,2}[n]$.

An easier way to compute a circularly shifted sequence is provided by the MATLAB command `circshift`. The appropriate syntax when using the command for a real sequence is `xcm=circshift(x', m)`, where m is the desired shift and x' is the sequence $x[n]$ given as a *column vector*. If $x[n]$ has complex values, then the appropriate syntax is `xcm=circshift(x.', m)`; that is, we have to insert the dot operator before the transpose operator, as the statement z' computes the conjugate transpose of a complex vector z .

Commands	Results	Comments
<pre> m = 2; xcm=circshift(x', m)</pre>	<pre> 0.7000 0.8000 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000</pre>	The circularly shifted by $m=2$ samples sequence $x_{c,m}[n]$. The sequence $x[n]$ is applied as a column vector to the command <code>circshift</code> .

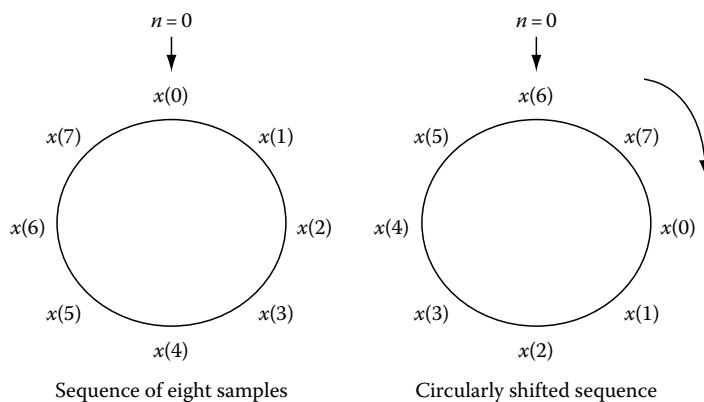
Example

Sketch the sequence $x[n]$ of the previous example and the circularly shifted sequences $x_{c,1}[n]$, $x_{c,2}[n]$, and $x_{c,4}[n]$.

Commands	Results	Comments
<pre> stem(0:N-1,x) xlim([- .5 N-1+.5]); ylim([- .1 1]); legend('x[n]')</pre>		The sequence $x[n]$.

(continued)

Commands	Results	Comments
<pre>m=1; xcm1=circshift(x',m); stem(0:N-1,xcm1); xlim([-0.5 N-1+0.5]); ylim([-0.1 1]); legend('x_c_,_m[n] , m=1')</pre>		The sequence $x_{c,1}[n]$.
<pre>m=2; xcm2=circshift(x',m); stem(0:N-1,xcm2); xlim([-0.5 N-1+0.5]); ylim([-0.1 1]); legend('x_c_,_m[n] , m=2')</pre>		The sequence $x_{c,2}[n]$.
<pre>m=4; xcm4=circshift(x',m); stem(0:N-1,xcm4); xlim([-0.5 N-1+0.5]); ylim([-0.1 1]); legend('x_c_,_m[n] , m=4')</pre>		The sequence $x_{c,4}[n]$.

**FIGURE 7.1**The sequence $x[n]$ is rotated clockwise to produce the circularly shifted sequence $x_{c,2}[n]$.

7.7.1 Discrete Fourier Transform of a Circularly Shifted Sequence

In this section, we introduce two properties of DFT related to the circular shift operation. Let X_k denote the DFT of $x[n]$. Then,

$$DFT\{x[((n-m))_N]\} = X_k e^{-j \frac{2\pi nm}{N}} \quad (7.23)$$

and

$$DFT\{x[n]\} e^{j \frac{2\pi nm}{N}} = X_{((k-m))_N}. \quad (7.24)$$

The first relationship is the circular shift in time property, while the second relationship is the circular shift in frequency property. First, the circular shift in time property, given by (7.23), is verified.

Commands	Results	Comments
<pre> x = [1 0 3 4 7]; m = 2; xc = circshift(x', m); Left = dft(xc) Left.' X = dft(x); N = length(x); k = 0:N-1; Right = X.*exp(-j*2*pi*m*k/N) Right.' </pre>	<pre> ans = 15.0000 6.2812 - 4.3920i -3.7812 - 1.4001i -3.7812 + 1.4001i 6.2812 + 4.3920i ans = 15.0000 6.2812 - 4.3920i -3.7812 - 1.4001i -37812+1.4001i 6.2812 + 4.3920i. </pre>	<p>The sequence $x[n] = [1, 0, 3, 4, 7]$, $0 \leq n \leq 4$ is considered. The applied shift is $m = 2$.</p> <p>The left part of (7.23) is computed by first deriving the circularly shifted sequence $x_{c,2}[n]$ and next computing its DFT.</p> <p>The right part of (7.23) is equal to the left; hence, the circular shift in time property is verified.</p>

Remark

Notice that in order to convert the row vectors *Left* and *Right* to column vectors we type *Left.'* and *Right.'*, respectively. The dot operator before the transpose operator is *required* in order to obtain the transpose without conjugation. Next, we verify the circular shift in frequency property which is given by (7.24).

Commands	Results	Comments
<pre>x = [1 0 3 4 7]; N = length(x); n = 0:N-1; m = 2; Left = dft(x.*exp(j*2*pi*n*m/N)); Left.'</pre>	<pre>ans = -2.5000 + 3.1634i -2.5000 + 7.2452i 15.0000 - 2.5000 -7.2452i -2.5000 - 3.1634i</pre>	The sequence $x[n] = [1, 0, 3, 4, 7]$, $0 \leq n \leq 4$ is again considered. The applied shift is $m = 2$. The left part of (7.24) is easily derived.
<pre>X = dft(x); Right = circshift(x.', m)</pre>	<pre>Right = -2.5000 + 3.1634i -2.5000 + 7.2452i 15.0000 - 2.5000 -7.2452i -2.5000 - 3.1634i</pre>	To obtain the right part of (7.24), first we apply DFT to the sequence $x[n]$ in order to compute X_k , and then X_k is circularly shifted by two samples to obtain $X_{((k-2)_N)}$. The two parts are equal; thus the circular shift in frequency property is also verified.

7.8 Circular Convolution

In this section, we introduce the concept of circular convolution. Circular convolution is similar but not the same as the (linear) convolution which was introduced in Chapter 4. The circular convolution between two discrete-time signals $x[n]$ and $h[n]$ is computed according to

$$x[n] \otimes h[n] = \sum_{m=0}^{N-1} x[m]h[((n-m))_N], \quad (7.25)$$

where

the symbol \otimes denotes circular convolution

$h[((n-m))_N]$ is the sequence $h[n]$ circularly shifted by m samples

To demonstrate how circular convolution is computed, we consider the discrete-time signals $x[n] = [1, 0, 2.5, 1.5]$, $0 \leq n \leq 3$ and $h[n] = [1, 1, 0.5, 2]$, $0 \leq n \leq 3$. If we were computing the linear convolution between $x[n]$ and $h[n]$ we would first derive the reversed signal $h[-m]$, and by changing the value of n we would slide the signal $h[n-m]$ from $-\infty$ to ∞ in order to compute the linear convolution.

In the case of circular convolution, first we derive the circular time-reversed signal $h[((-m))_N]$. Afterward, by changing appropriately the values of n we obtain $h[((n-m))_N]$, and we compute the circular convolution according to Equation 7.25. In order to derive $h[((-m))_N]$, we take into account that

$$((-m))_N = \begin{cases} 0, & m = 0 \\ N - m, & m = 1, 2, \dots, N - 1 \end{cases} \quad (7.26)$$

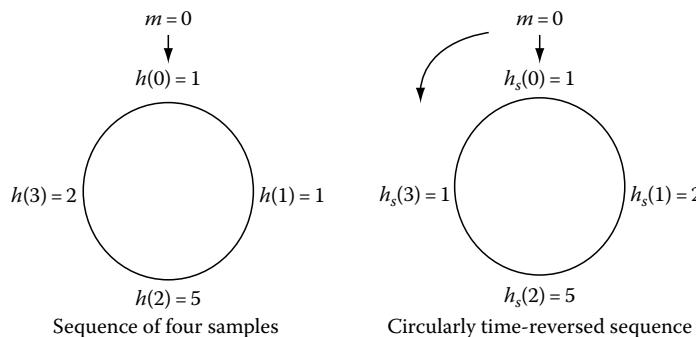
Commands	Results	Comments
<pre>N=4 for m=0:3 p(m+1)=mod(-m,N); end p</pre>	<p style="text-align: center;">$p = 0 \quad 3 \quad 2 \quad 1$</p>	Confirmation of Equation 7.26.

Therefore, the circular time-reversed signal $h_s[m] = h[(-(m))_N]$ is computed according to the following procedure.

Commands	Results	Comments
<pre>h = [1, 1, 0.5, 2]; for m=0:3 hs(1+m) = h(1+p(m+1)); end hs'</pre>	<p style="text-align: center;">$hs = 1.0000 \quad 2.0000 \quad 0.5000 \quad 1.0000$</p>	$\begin{aligned} h_s[0] &= h[(0)_4] \\ &= h[0] = 1 \\ h_s[1] &= h[(-1)_4] \\ &= h[4-1] \\ &= h[3] = 2 \\ h_s[2] &= h[(-2)_4] \\ &= h[4-2] \\ &= h[2] = 0.5 \\ h_s[3] &= h[(-3)_4] \\ &= h[4-3] \\ &= h[1] = 1 \end{aligned}$

Graph of the circular time-reversed sequence $h_s[m] = h[(-(m))_N]$.

To completely understand how to derive the circularly time-reversed signal, consider that the elements of the sequence are placed in a circle and we are picking them in anticlockwise order to yield the circular time-reversed sequence. This process is depicted in Figure 7.2 for the discrete-time signal $h[n] = [1, 1, 0.5, 2]$.

**FIGURE 7.2**

The circularly time-reversed sequence $h_s[n]$ is derived by picking the samples of $h[n]$ in anticlockwise order.

Next, the computation of the circular convolution between $x[n]$ and $h[n]$ is graphically illustrated, likewise the graphic illustration of the linear convolution in Chapter 4.

Remark

Changing the value of n from 0 to $N - 1$ results in a *circular shift* of $h[((n - m))_N]$.

The convolution is computed according to Equation 7.25, i.e., for the signals $x[n]$ and $h[n]$ is given by

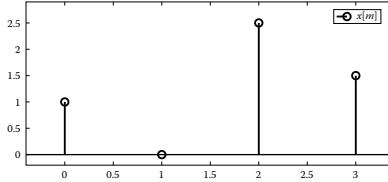
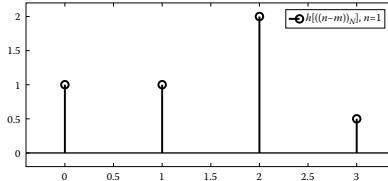
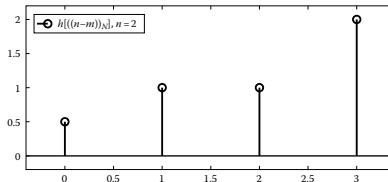
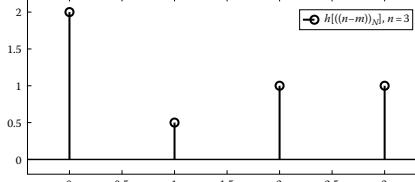
$$y[n] = x[n] \otimes h[n] = \sum_{m=0}^3 x[m]h[((n - m))_4], \quad n = 0, 1, 2, 3.$$

Therefore, it is necessary to compute $h[((n - m))_4]$ for $n = 0, 1, 2, 3$. The sequence $h[((n - m))_4]$ for $n = 0, 1, 2, 3$ is graphically illustrated along with $x[m]$ in order to demonstrate the overlapping samples of the two sequences.

Commands	Results	Comments
<pre>N = 4; x = [1, 0, 2.5, 1.5]; m = 0:N-1; stem(m,x); axis([-4 3.4 -.2 2.9]); legend('x[m]')</pre>		The discrete-time signal $x[m]$.
<pre>n = 0; hsn = circshift(hs', n); stem(m, hsn); axis([-4 3.4 -.2 2.2]); legend('h[((n-m))_N], n=0')</pre>		The circular time-reversed sequence $h[((0-m))_N] = h[((-m))_N] = h_s[m]$.
<pre>clear y; y(n+1) = sum(x.*hsn') % or alternatively % y(n+1) = x*hsn;</pre>	$y = 3.7500$	The circular convolution for $n = 0$, that is, $y[0]$ is computed from the overlapping samples of the two sequences according to Equation 7.25. Hence, $y[0]$ is calculated by $y[0] = \sum_{m=0}^3 x[m]h[((0 - m))_4] = \sum_{m=0}^3 x[m]h_s[m] \Rightarrow y[0] = x[0]h_s[0] + x[1]h_s[1] + x[2]h_s[2] + x[3]h_s[3] \Rightarrow y[0] = 1 \cdot 1 + 0 \cdot 2 + 2.5 \cdot 0.5 + 1.5 \cdot 1 = 3.75$.

(continued)

(continued)

Commands	Results	Comments
<pre>m = 0:N-1; stem(m,x); axis([- .4 3.4 -.2 2.9]); legend('x[m]')</pre>		The discrete time signal $x[n]$ is plotted again for convenience.
<pre>n=1; hsn=circshift(hs',n); stem(m,hsn); axis([- .4 3.4 -.2 2.2]); legend ('h[((n-m))_N], n=1')</pre>		The signal $h[((1-m))_N]$.
$y(n+1) = \text{sum}(x.*hsn')$	$y = 3.7500 \quad 6.7500$	The result for $n = 1$, that is, $y[1]$ is computed from the overlapping samples of the sequences $x[m]$ and $h[((1-m))_4]$ according to the relationship $y[1] = \sum_{m=0}^3 x[m]h[((1-m))_4] = 6.75$.
<pre>n=2; hsn=circshift(hs',n); stem(m,hsn); axis([- .4 3.4 -.2 2.2]); legend ('h[((n-m))_N], n=2')</pre>		The signal $h[((2-m))_N]$.
$y(n+1) = \text{sum}(x.*hsn')$	$y = 3.7500 \quad 6.7500 \quad 6.0000$	For $n = 2$, $y[n] = y[2]$ is given by $y[2] = \sum_{m=0}^3 x[m]h[((2-m))_4] = 6$.
<pre>n=3; hsn=circshift(hs',n); stem(m,hsn); axis([- .4 3.4 -.2 2.2]); legend ('h[((n-m))_N], n=3')</pre>		The signal $h[((3-m))_N]$.

(continued)

Commands	Results	Comments
$y(n+1) = x * hsn$	$y = 3.7500 \quad 6.7500 \quad 6.0000 \quad 6.0000$	For $n=3$, $y[n]=y[3]$ is given by $y[3] = \sum_{m=0}^3 x[m] h[(3-m)_4] = 6$.
<pre>stem(m,y) axis([-0.4 3.4 -0.2 7.2]); title('circular convolution') legend('y[n]')</pre>		Graph of the circular convolution $y[n] = x[n] \otimes h[n]$.

7.8.1 Discrete Fourier Transform of Circular Convolution

Recall that the CTFT of the linear convolution of two continuous-time signals is equal to the product of the Fourier transforms of these two signals. Recall also that the DTFT of the linear convolution of two discrete-time signals is equal to the product of the DTFTs of these two signals. Regarding the DFT, a similar property referring to the circular convolution stands.

The DFT of the *circular* convolution of two discrete-time signals is equal to the product of the DFTs of the two signals.

Let $X_1(k)$ and $X_2(k)$ denote the DFTs of the sequences $x_1[n]$ and $x_2[n]$ respectively. The mathematical expression of the above-mentioned property is

$$DFT\{x_1[n] \otimes x_2[n]\} = X_1(k) \cdot X_2(k) \quad (7.27)$$

To confirm (7.27), we consider the discrete-time signals $x[n]=[1, 0, 2.5, 1.5]$, $0 \leq n \leq 3$ and $h[n]=[1, 1, 0.5, 2]$, $0 \leq n \leq 3$. Their circular convolution is already computed in the previous section as $y[n]=x_1[n] \otimes x_2[n]=[3.75, 6.75, 6, 6]$, $0 \leq n \leq 3$.

Commands	Results	Comments
$y = [3.75 \ 6.75 \ 6 \ 6];$ $\text{Left} = \text{dft}(y);$ $\text{Left}.'$	$\text{ans} = 22.5000$ $-2.2500 - 0.7500i$ $-3.0000 - 0.0000i$ $-2.2500 + 0.7500i$	The signal $y[n] = x_1[n] \otimes x_2[n]$ is defined, and by computing its DFT, we obtain the left side of (7.27).
$x_1 = [1 \ 0 \ 2.5 \ 1.5];$ $x_2 = [1 \ 1.5 \ 2];$ $X_1 = \text{dft}(x_1);$ $X_2 = \text{dft}(x_2);$ $\text{Right} = X_1.*X_2;$ $\text{Right}.'$	$\text{ans} = 22.5000$ $-2.2500 - 0.7500i$ $-3.0000 - 0.0000i$ $-2.2500 + 0.7500i$	The DFTs $X_1(k)$ and $X_2(k)$ of $x_1[n]$ and $x_2[n]$, respectively, are computed. The right side of (7.27) is derived by multiplying $X_1(k)$ with $X_2(k)$. The two sides are same, hence (7.27) is verified.

7.8.2 Relationship between Linear and Circular Convolution

The linear convolution between two sequences $x_1[n]$ and $x_2[n]$ is a sequence $y_1[n]$, where the number of samples of $y_1[n]$ is given by the relationship $\text{length}(y_1) = \text{length}(x_1) + \text{length}(x_2) - 1$. Hence, if $x_1[n]$ and $x_2[n]$ have N samples, the sequence $y_1[n] = x_1[n] * x_2[n]$ has $2N - 1$ elements. On the other hand, the circular convolution $y_2[n]$ between $x_1[n]$ and $x_2[n]$ if $x_1[n]$ and $x_2[n]$ are sequences of N samples is also a sequence of N samples. In order to make the circular convolution equivalent to the linear convolution, $N - 1$ zero samples have to be padded at the end of the sequences that are circularly convoluted. Thus, $x_1[n]$ and $x_2[n]$ become sequences of $N + N - 1 = 2N - 1$ samples with their last $N - 1$ samples being equal to zero. Let $x_{11}[n]$ and $x_{22}[n]$ denote the zero-padded versions of $x_1[n]$ and $x_2[n]$, respectively. Then the sequence $y_{22}[n] = x_{11}[n] \otimes x_{22}[n]$ is a sequence of $2N - 1$ samples, and more importantly the samples of $y_{22}[n]$ are the same as the samples of the linear convolution $y_1[n] = x_1[n] * x_2[n]$.

Example

Consider the sequences $x_1[n] = [1, 2, 3, 4], 0 \leq n \leq 3$ and $x_2[n] = [3, 2, 5, 1], 0 \leq n \leq 3$. Compute

- The $N = 4$ point circular convolution $x_1[n] \otimes x_2[n]$
- The $M = 2N - 1 = 7$ point circular convolution $x_{11}[n] \otimes x_{22}[n]$, where $x_{11}[n] = [1, 2, 3, 4, 0, 0, 0], 0 \leq n \leq 6$ and $x_{22}[n] = [3, 2, 5, 1, 0, 0, 0], 0 \leq n \leq 6$
- The linear convolution $x_1[n] * x_2[n]$

Commands	Results	Comments
a. <pre>x1 = [1,2,3,4]; x2 = [3,2,5,1]; N = length(x1); for m=0:N-1 p(m+1) = mod(-m,N); x2s(1+m) = x2(1+p(m+1)); end x2s</pre>	<pre>x2s = 3 1 5 2</pre>	Definition of $x_1[n]$, $x_2[n]$, and N .
<pre>for n=0:N-1 x2sn=circshift(x2s',n); y2(n+1)=x1*x2sn; end y2</pre>	<pre>y2 = 28 31 22 29</pre>	We compute the circular time-reversed version of $x_2[n]$. Computation of the 4-point circular convolution $x_1[n] \otimes x_2[n]$.
b. <pre>x11 = [x1 0 0 0]; x22 = [x2 0 0 0]; N = length(x11); for m=0:N-1 p(m+1) = mod(-m,N); end for m=0:N-1 x22s(1+m) = x22(1+p(m+1)); end for n=0:N-1 x22sn=circshift(x22s',n); y22(n+1) = x11*x22sn; end y22</pre>	<pre>y22 = 3 8 18 29 25 23 4</pre>	The sequences $x_1[n]$ and $x_2[n]$ are zero-padded in order to create $x_{11}[n]$ and $x_{22}[n]$. Computation of the circular time-reversed version of $x_{22}[n]$.
c. <pre>y1 = conv(x1, x2)</pre>	<pre>y1 = 3 8 18 29 25 23 4</pre>	Computation of the 7-point circular convolution $x_{11}[n] \otimes x_{22}[n]$. The linear convolution $x_1[n]*x_2[n]$ is calculated by using the command <code>conv</code> , and is equal to the circular convolution of the zero-padded sequences.

7.9 Fast Fourier Transform

Computing an N -point DFT or IDFT directly from its definition can be a computationally expensive process. More specifically, looking into Equation 7.12 we notice that for every value that k takes, i.e., for every X_k , N multiplications must be performed. Thus, in order to compute the entire sequence X_k , $k = 0, \dots, N - 1$, we must perform N^2 multiplications. If the discrete-time signal $x[n]$ is complex valued, things get difficult, since one multiplication of two complex numbers requires four multiplications between real numbers. Hence, computing the DFT (or IDFT) directly from the definition is usually too slow for real-time applications. In order to reduce the computational effort needed, an efficient algorithm (with many variants) is available for the DFT computation. This algorithm is called *fast Fourier transform algorithm* or *FFT algorithm*. FFT is based on a “divide and conquer” technique; that is, the original problem of N points is divided in two symmetric subproblems of $N/2$ points. If $N/2$ is even number, the problem of $N/2$ points is divided in two subproblems

of $N/4$ points. If N is a power of 2, i.e., $N = 2^p$ then only a 2-point DFT has to be computed. The DFT X_k of a sequence $x[n]$ is computed in MATLAB through a FFT algorithm, with the command `fft`. The syntax is $X = \text{fft}(x)$, where x is the sequence $x[n]$ and X is the DFT X_k .

Commands	Results	Comments
<code>x = [1 2 3];</code> <code>Xk = fft(x)</code>	$Xk = 6.0000 - 1.0000i$ 1.0981 + 1.3660i -4.0981 - 0.3660i	The DFT of the sequence $x[n] = [1, 2, 3]$, $0 \leq n \leq 2$ computed by the <code>fft</code> command.
<code>Xk = dft(x)</code>	$Xk = 6.0000 - 1.0000i$ 1.0981 + 1.3660i -4.0981 - 0.3660i	The DFT of the same sequence computed by the function <code>dft.m</code> that was created in Section 7.4. Of course, the result in both cases is the same.

Remark

FFT must not be confused with DFT. FFT is an algorithm that computes the DFT of a sequence.

To illustrate the efficiency of the FFT algorithm, we compare the execution speed of the MATLAB command `fft` with that of the function `dft.m` created in Section 7.4. The benchmarking is performed by using the MATLAB commands `tic` and `toc`. The command `toc` measures the time elapsed from the execution of the `tic` command. We create an M-File named `tictoc.m` that is used for the benchmarking test.

M-FILE `tictoc.m` and Its Execution from the Command

prompt	Results	Comments
<pre>N=20 n=0:N-1; x=(-1).^(n); disp('DFT') tic X=dft(x); toc disp('FFT') tic X=fft(x); toc N=2^12 n=0:N-1; x=(-1).^(n); disp('DFT') tic X=dft(x); toc disp('FFT') tic X=fft(x); toc tictoc</pre>	<p>N = 20 DFT Elapsed time is 0.009416 s. FDT Elapsed time is 0.000027 s. N = 4096 DFT Elapsed time is 10.780808 s. FFT Elapsed time is 0.000171 s.</p>	<p>The sequence $x[n] = (-1)^n$, $0 \leq n \leq N-1$ is defined first for $N=20$ and then for $N=2^{12}$. The N-point DFT of $x[n]$ is computed through the <code>dft.m</code> function and through the <code>fft</code> command.</p> <p>The difference in the execution time is very large in both cases ($N=20$ and $N=2^{12}$). In the second case ($N=2^{12}$), the DFT of $x[n]$ was computed in 10 s through the <code>dft.m</code> function (i.e., by computing the DFT according to its definition) while by using the <code>fft</code> command (i.e., by using the FFT algorithm) the DFT of $x[n]$ was computed in 171 μs.</p>

Small execution time is a crucial task in real-time applications. A real-time application that uses the IDFTs and DFTs of sequences is the orthogonal frequency division multiplexing (OFDM) telecommunication system. Of course, the DFTs are computed by an FFT algorithm. Hence, from now on, when a DFT of a sequence has to be calculated, we will use the command `fft(x)`. Next, one more benchmarking test is implemented for the FFT algorithm. This time the performance of FFT is evaluated in relation to the value of N ; that is, we explore the difference in the execution time if N is a power of 2. The M-File `tictoc2.m` is used for this test.

**M-FILE `tictoc2.m`
and Its Execution
from the Command
prompt**

prompt	Results	Comments
<code>N = 2^20</code>		
<code>n = 0:N-1;</code>		
<code>x = (-1).^n;</code>		
<code>tic</code>		
<code>X = fft(x);</code>		
<code>toc</code>		
<code>N = 950000</code>		
<code>n = 0:N-1;</code>		
<code>x = (-1).^n;</code>	<code>N = 1048576</code>	The performance evaluation is done using the sequence $x[n] = (-1)^n$, $0 \leq n \leq N-1$ for $N = 2^{20} = 1,048,576$, $N = 950,000$, $N = 850,000$, $N = 750,000$, and $N = 650,000$.
<code>tic</code>	<code>Elapsed time is 0.191865 s.</code>	
<code>X = fft(x);</code>		
<code>toc</code>		
<code>N = 850000</code>	<code>N = 950000</code>	We notice that if N is a power of 2; that is, if $N = 2^p$, the DFT of an N -point sequence is computed faster than the DFT of a sequence with $M < N$ points if M is not a power of 2.
<code>n = 0:N-1;</code>	<code>Elapsed time is 0.341129 s.</code>	
<code>x = (-1).^n;</code>	<code>N = 850000</code>	
<code>tic</code>	<code>Elapsed time is 0.310672 s.</code>	
<code>X = fft(x);</code>		
<code>toc</code>		
<code>N = 750000</code>	<code>N = 750000</code>	More specifically, the DFT of the sequence $x[n] = (-1)^n$, $0 \leq n \leq N-1$ for $N = 2^{20} = 1,048,576$ was computed faster than the DFTs of the same sequence for $N = 950,000$, $N = 850,000$, and $N = 750,000$.
<code>n = 0:N-1;</code>	<code>Elapsed time is 0.195277 s.</code>	
<code>x = (-1).^n;</code>	<code>N = 650000</code>	
<code>tic</code>	<code>Elapsed time is 0.164688 s.</code>	
<code>X = fft(x);</code>		
<code>toc</code>		
<code>N = 650000</code>		
<code>n = 0:N-1;</code>		
<code>x = (-1).^n;</code>		
<code>tic</code>		
<code>X = fft(x);</code>		
<code>toc</code>		
<code>tictoc2</code>		

An alternative syntax of the `fft` command is `X = fft(x, N)`. Using this syntax we derive the N -point DFT of an M -point sequence. If $M > N$, the sequence $x[n]$ is truncated; while if $M < N$, the sequence $x[n]$ is zero-padded.

Commands	Results	Comments
<code>x = [1 2 3 4];</code> <code>fft(x)</code>	<code>ans = 10.0000 -2.0000 + 2.0000i -2.0000 -2.0000 - 2.0000i</code>	4-point DFT of the sequence $x[n] = [1, 2, 3, 4]$, $0 \leq n \leq M-1$, where $M=4$.
<code>fft(x, 6)</code>	<code>ans = 10.0000 -3.5000 - 4.3301i 2.5000 + 0.8660i -2.0000 2.5000 - 0.8660i -3.5000 + 4.3301i</code>	Case $M < N$. 6-point DFT of the sequence $x[n] = [1, 2, 3, 4], 0 \leq n \leq 3$.
<code>xx = [x 0 0];</code> <code>fft(xx)</code>	<code>ans = 10.0000 -3.5000 - 4.3301i 2.5000 + 0.8660i -2.0000 2.5000 - 0.8660i -3.5000 + 4.3301i</code>	The sequence $x[n]$ is padded with two zeros and its FFT is computed. In this way, we verify that by using the command <code>fft(x, 6)</code> $N-M=2$ zeros are padded at the end of $x[n]$ to convert it into a 6-point sequence.
<code>fft(x, 3)</code>	<code>ans = 6.0000 -1.5000 + 0.8660i -1.5000 - 0.8660i</code>	Case $M > N$. 3-point DFT of the sequence $x[n] = [1, 2, 3, 4], 0 \leq n \leq 3$.
<code>xx = [1 2 3];</code> <code>fft(xx)</code>	<code>ans = 6.0000 -1.5000 + 0.8660i -1.5000 - 0.8660i</code>	The sequence $x[n]$ is truncated by one sample and its FFT is computed. Indeed by using the command <code>fft(x, 3)</code> , the last point of $x[n]$ is discarded to convert $x[n]$ into a 3-point sequence.

The IDFT of a sequence X_k is computed by the MATLAB command `ifft`. The syntax is $\mathbf{x} = \text{ifft}(\mathbf{X})$ or $\mathbf{x} = \text{ifft}(\mathbf{X}, N)$ if an N -point IDFT is required.

Commands	Results	Comments
<code>X = [10, -2+2j, -2, -2-2j];</code> <code>ifft(X)</code>	<code>ans = 1 2 3 4</code>	IDFT of the sequence $X_k = [10, -2+2i, -2, -2-2i], 0 \leq k \leq N-1$, where $N=4$.
<code>idft(X)</code>	<code>ans = 1.0000 2.0000 + 0.0000i 3.0000 - 0.0000i 4.0000 - 0.0000i</code>	The IDFT of X_k is computed through the function <code>idft.m</code> that was created in Section 7.6. As expected, the two results are same.
<code>ifft(X, 6)</code>	<code>ans = 0.6667 1.7113 - 0.0774i 1.3780 - 0.5000i 2.0000 1.9553 - 0.5000i 2.2887 + 1.0774i</code>	6-point IDFT of X_k .
<code>X(6) = 0</code>	<code>X = 10 -2+2i -2 -2-2i 0 0</code>	X_k is zero padded in order to convert it into a 6-point sequence and its IDFT is computed below.
<code>ifft(X)</code>	<code>ans = 0.6667 1.7113 - 0.0774i 1.3780 - 0.5000i 2.0000 1.9553 - 0.5000i 2.2887 + 1.0774i</code>	Indeed, by using the command <code>ifft(X, 6)</code> $N-M=2$, zeros are padded at the end of X_k to convert it into a 6-point sequence.

7.10 Relationship between DFT and DTFT

In this section, we establish the relationship between the two types of Fourier transform that are applicable to discrete-time signals, namely, the DFT and the DTFT. For convenience, the mathematical expressions of the two transforms are given again. The DTFT of a signal $x[n]$ is given by

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}, \quad (7.28)$$

while the DFT of $x[n]$ is given by

$$X_k = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi nk}{N}}, \quad k = 0, 1, \dots, N-1. \quad (7.29)$$

Suppose that $x[n]$ is a truncated signal, i.e., $x[n]=0$ for $n < 0$ and $n \geq N$. In this case Equation 7.28 becomes

$$X(\omega) = \sum_{n=0}^{N-1} x[n]e^{-j\omega n}. \quad (7.30)$$

Looking into Equations 7.30 and 7.29, we conclude that

$$X_k = X(\omega)|_{\omega=\frac{2\pi k}{N}} = X\left(\frac{2\pi k}{N}\right), \quad k = 0, 1, \dots, N-1. \quad (7.31)$$

Equation 7.31 states that the DFT X_k is actually a sampling in the frequency of the DTFT $X(\omega)$, or in other words the DFT is a sampling of the continuous spectrum of a discrete-time signal. More precisely, we can state that the DFT sequence X_k is equal to the DTFT $X(\omega)$ when $X(\omega)$ is evaluated at the points $\omega_k = 2\pi k/N$, $k = 0, 1, \dots, N-1$. To verify the relationship between DTFT and DFT, consider the truncated discrete-time signal $x[n] = 0.9^n$, $0 \leq n \leq 7$. The DTFT is evaluated at the frequencies $\omega_k = 2\pi k/N$, $k = 0, 1, \dots, N-1$. and is compared to the DFT of $x[n]$.

Commands	Results	Comments
<pre>n = 0 : 7; x = 0.9.^n; syms w Xdtft = sum(x.*exp(-j*w*n));</pre>	<pre>ans = 5.6953 0.3855 - 0.6747i 0.3147 - 0.2832i 0.3023 - 0.1176i 0.2998 - 0.0000i 0.3023 + 0.1176i 0.3147 + 0.2832i 0.3855 + 0.6747i</pre>	Definition of $x[n]$ and computation of its DTFT $X(\omega)$.
<pre>N = 8; k = 0:N-1; wk = 2*pi*k/N; XXdtft = subs(Xdtft,w,wk); XXdtft.'</pre>	<pre>ans = 5.6953 0.3855 - 0.6747i 0.3147 - 0.2832i 0.3023 - 0.1176i 0.2998 - 0.0000i 0.3023 + 0.1176i 0.3147 + 0.2832i 0.3855 + 0.6747i</pre>	The DTFT $X(\omega)$ of $x[n]$ is evaluated at the frequency points $\omega_k = 2\pi k/N$, $k = 0, 1, \dots, N-1$.
<pre>X = fft(x) X.'</pre>	<pre>ans = 5.6953 0.3855 - 0.6747i 0.3147 - 0.2832i 0.3023 - 0.1176i 0.2998 - 0.0000i 0.3023 + 0.1176i 0.3147 + 0.2832i 0.3855 + 0.6747i</pre>	The DFT X_k of $x[n]$ is computed, and is equal to the values that $X(\omega)$ takes for $\omega_k = 2\pi k/N$, $k = 0, 1, \dots, N-1$.

Example

Let $x[n]$ be a random sequence of 21 elements. Plot in the same figure the DTFT $X(\omega)$ of $x[n]$ for $0 \leq \omega \leq 2\pi$ and the DFT of $x[n]$ versus the frequencies $\omega_k = 2\pi k/N$, $k = 0, 1, \dots, N-1$.

Commands	Results	Comments
<pre>x = randn(1, 21);</pre>		Definition of a random sequence $x[n]$.
<pre>n = 0 : 20; syms w Xdtft = sum(x.*exp(-j*w*n));</pre>		The DTFT $X(\omega)$ of $x[n]$ is computed.
<pre>Xdft = fft(x);</pre>		Computation of the DFT X_k of $x[n]$.
<pre>N = length(Xdft); k = 0:N-1; wk = 2*pi*k/N;</pre>		Computation of the frequency points $\omega_k = 2\pi k/N$, $k = 0, 1, \dots, N-1$.

(continued)

Commands	Results	Comments
<pre>ezplot(abs(Xdtft), [0 2*pi]); hold on plot(wk, abs(Xdft), 'o') hold off legend('DTFT', 'DFT')</pre>		First, the magnitude of DTFT is plotted over the interval $0 \leq \omega \leq 2\pi$. Next, the magnitude of the DFT is plotted versus the evaluated at the previous step frequency points ω_k .
<pre>w1 = 0:.01:2*pi; XXdtft = subs(Xdtft, w, w1); plot(w1,angle(XXdtft)); hold on plot(wk,angle(Xdft), 'o') legend('DTFT', 'DFT') hold off</pre>		Graph of the phase of the two transforms.

The two transforms match up at the frequency points $\omega_k = 2\pi k/N$, $k = 0, 1, \dots, N-1$; hence, the relationship between DTFT and DFT described by Equation 7.31 is verified. Suppose now that we want to plot the DTFT and the DFT of a sequence in the frequency interval $-\pi \leq \omega \leq \pi$, that is, to include the negative frequencies in the graph. In this case, we have to shift the zero-frequency component of the computed DFT sequence to the center of the signal spectrum. This is implemented in MATLAB by using the command `fftshift`. The syntax `fftshift(X)` swaps the left and right halves of vector X .

Commands	Results	Comments
<pre>n=0:11; x=0.9.^n; syms w Xdtft=sum(x.*exp(-j*w*n)); Xdft=fft(x); Xdft.'</pre>	<pre>ans = 7.1757 0.6302 - 1.2857i 0.4337 - 0.6146i 0.3964 - 0.3568i 0.3839 - 0.2064i 0.3790 - 0.0959i 0.3777 0.3790 + 0.0959i 0.3839 + 0.2064i 0.3964 + 0.3568i 0.4337 + 0.6146i 0.6302 + 1.2857i</pre>	Definition of $x[n] = 0.9^n$, $0 \leq n \leq 11$ and computation of its DTFT and its DFT. The DFT samples are shown in the middle tab.
<pre>N=12; k=-N/2:N/2-1; wk=2*pi*k/N; wk'</pre>	<pre>ans = -3.1416 - 2.6180 -2.0944 - 1.5708 -1.0472 - 0.5236 0 0.5236 1.0472 1.5708 2.0944 2.6180</pre>	Evaluation of the frequency points $\omega_k = 2\pi k/N$, $-N/2 \leq k \leq (N/2)-1$.

(continued)

(continued)

Commands	Results	Comments
<pre>Xshift = fftshift(Xdft) Xshift.'</pre>	<pre>ans = 0.3777 0.3790 + 0.0959i 0.3839 + 0.2064i 0.3964 + 0.3568i 0.4337 + 0.6146i 0.6302 + 1.2857i 7.1757 0.6302 - 1.2857i 0.4337 - 0.6146i 0.3964 - 0.3568i 0.3839 - 0.2064i 0.3790 - 0.0959i</pre>	Demonstration of the <code>fftshift</code> operation. The zero-frequency component is placed at the center of the spectrum and the left and right halves of vector <code>Xdft</code> are swapped.
<pre>ezplot(abs(Xdtft), [-pi, pi]); hold ; plot(wk,abs(Xshift), 'o') hold on ylim([0 8]) legend('DTFT', 'DFT') title(' DTFT & DFT ') hold off</pre>		Graph of the magnitudes of the DTFT and of the DFT of $x[n]$. Notice that we use the vector <code>Xshift</code> that was formulated via the <code>fftshift</code> command to plot the graph of the DFT of $x[n]$.
<pre>w1=-pi:.01:pi; Xdtft=subs(Xdtft,w,w1); plot(w1,angle(Xdtft)); hold on plot(wk,angle(Xshift), 'o') legend('DTFT', 'DFT') xlim([-pi pi]) title('\angle DTFT &\angle DFT') hold off</pre>		Graph of the phases of the DTFT and of the DFT of $x[n]$.

7.11 Relationship between Fourier Transform and Discrete Fourier Transform

In this section, we discuss how the CTFT is approximated by the DFT. Suppose that $X(\Omega)$ denotes the Fourier transform of a continuous-time signal $x(t)$. The procedure followed in order to approximate $X(\Omega)$ through samples obtained from the FFT algorithm is

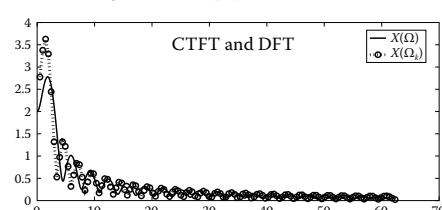
- The signal $x(t)$ is sampled with sampling time T ; that is, we obtain the discrete-time signal $x[nT]$, $n = 0, 1, \dots, N - 1$.
- The DFT X_k , $k = 0, 1, \dots, N - 1$ of the discrete-time signal $x[nT]$ is computed.
- The Fourier transform $X(\Omega)$ can be approximated at the frequencies $\Omega_k = 2\pi k/NT$, $k = 0, 1, \dots, N - 1$ from the DFT samples according to

$$X(\Omega_k) = NT \frac{1 - e^{-j\frac{2\pi k}{N}}}{j2\pi k} X_k, \quad k = 0, 1, \dots, N-1. \quad (7.32)$$

As N is getting larger or the sampling time T is getting smaller, we obtain a better approximation of $X(\Omega)$ by the sequence $X(\Omega_k)$, which is computed according to (7.32).

Example

Compute and plot the magnitude of the Fourier transform $X(\Omega)$ of the continuous-time signal $x(t) = 2 - 3t$, $0 \leq t \leq 2$. Also compute and plot in the same graph the magnitude of the approximate sequence $X(\Omega_k)$ for $N = 128$ and $T = 0.1$.

Commands	Results/Comments
$T = .1;$ $N = 128;$ $t = 0 : 1 / (N * T) : 2;$ $x = 2 - 3 * t;$ $Xk = fft(x, N);$ $k = 0 : N - 1;$ $wk = 2 * pi * k / (N * T);$ $Xwk = (N * T * (1 - exp(-j * 2 * pi * k / N)) . / (j * 2 * pi * k)) .* Xk$	The sequence $x[nT]$, $n = 0, 1, \dots, N-1$ is defined for $N = 128$ and $T = 0.1$.
$\text{syms } t \text{ w}$ $x = (2 - 3 * t) * (\text{heaviside}(t) - \text{heaviside}(t - 2));$ $X = \text{fourier}(x, w);$ $\text{ezplot}(\text{abs}(X), [0 \text{ } wk(N)]);$ hold on $\text{plot}(wk, \text{abs}(Xwk), ':o')$ hold off $\text{legend}('X(\Omega)', 'X(\Omega_k)')$ $\text{ylim}([0 \text{ } 4])$ $\text{title}('CTFT and DFT')$	Computation of the DFT X_k of $x[nT]$. The frequency points Ω_k are evaluated and the sequence $X(\Omega_k)$ is computed according to (7.32). The signal $x(t)$ is declared as symbolic expression and its Fourier transform $X(\Omega)$ is computed.
	Graph of $X(\Omega)$ and of $X(\Omega_k)$ over the frequency interval $\Omega_0 \leq \Omega \leq \Omega_{N-1}$.
	

7.12 Linear Convolution Computation via Fast Fourier Transform

In this chapter, we have seen that the DFT of the circular convolution of two sequences is equal to the product of their DFTs. Moreover, it was established that the linear convolution of two sequences is equivalent to their circular convolution if the two sequences are properly padded with zeros. Furthermore, we discussed that the N -point DFT of an M -point sequence with $M < N$ is computed by padding the sequence with $N - M$ zeros. Finally, recall that the linear convolution of two sequences $x_1[n]$ of N_1 samples and $x_2[n]$ of N_2 samples is a sequence $y[n]$ of $M = N_1 + N_2 - 1$ samples. In this section, by combining the

above-mentioned matters, we introduce an alternative way of computing the linear convolution of two sequences. Suppose that $X_1(k)$ and $X_2(k)$ are the DFTs of the N_1 -point sequence $x_1[n]$ and N_2 -point sequence $x_2[n]$, respectively. The procedure of computing the linear convolution $x_1[n] * x_2[n]$ is

- a. Compute the $N = N_1 + N_2 - 1$ point DFTs of $x_1[n]$ and $x_2[n]$.
- b. Multiply the two DFTs.
- c. The IDFT of the product is equal to the linear convolution of $x_1[n]$ and $x_2[n]$.

Commands	Results	Comments
<code>x1 = [1 2 0 5]; x2 = [3 2 1]; N1 = 4; N2 = 3;</code>	<code>x1 = 1 2 0 5 x2 = 3 2 1</code>	The sequences $x_1[n]=[1,2,0,5]$, $0 \leq n \leq N_1-1$, where $N_1=4$ and $x_2[n]=[3,2,1]$, $0 \leq n \leq N_2-1$, where $N_2=3$ are defined.
<code>N=N1+N2-1; X1=fft(x1,N); X2=fft(x2,N); PROD=X1.*X2;</code>	<code>CON = 3 8 5 17 10 5</code>	The $N = N_1 + N_2 - 1 = 6$ point DFTs $X_1(k)$ and $X_2(k)$ of $x_1[n]$ and $x_2[n]$, respectively, are computed.
<code>CON=ifft(PROD)</code>	<code>CON = 3 8 5 17 10 5</code>	The product $X_1(k) \cdot X_2(k)$ is calculated.
<code>y=conv(x1,x2)</code>	<code>y = 3 8 5 17 10 5</code>	Applying IDFT at the product we obtain the result of the linear convolution between $x_1[n]$ and $x_2[n]$.
		The result is confirmed by computing the linear convolution of the two sequences via the command <code>conv</code> .

7.13 Solved Problems

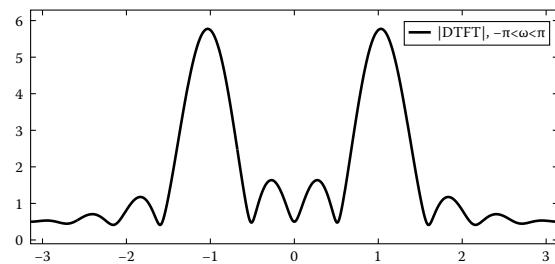
Problem 1 Plot the magnitude and the phase of the DTFT $X(\omega)$ of the signal $x[n] = \cos(\pi n/3)$, $0 \leq n \leq 10$ over the frequency intervals $-\pi \leq \omega \leq \pi$ and $-3\pi \leq \omega \leq 3\pi$.

Solution

```

syms w
n = 0:10;
x = cos((1/3)*pi*n);
X = sum(x.*exp(-j*w*n));
ezplot(abs(X), [-pi pi])
legend('|DTFT|, -\pi<\omega<\pi')

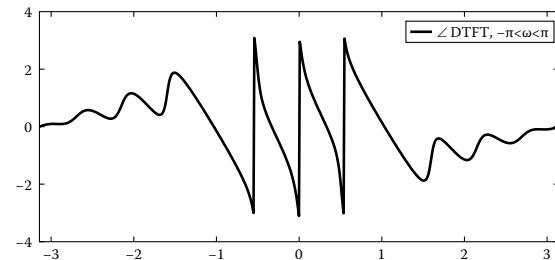
```



```

w1 = -pi:.01:pi;
XX = subs(X,w,w1);
plot(w1,angle(XX))
xlim([-pi pi]);
legend('\angle DTFT, -\pi<\omega<\pi')

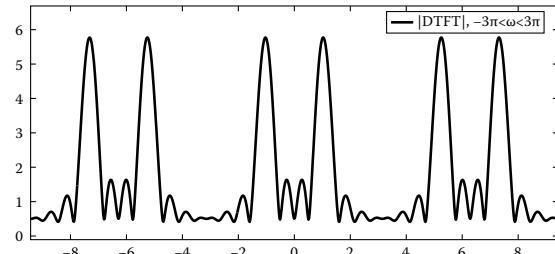
```



```

ezplot(abs(X), [-3*pi 3*pi])
legend('|DTFT|, -3\pi<\omega<3\pi')

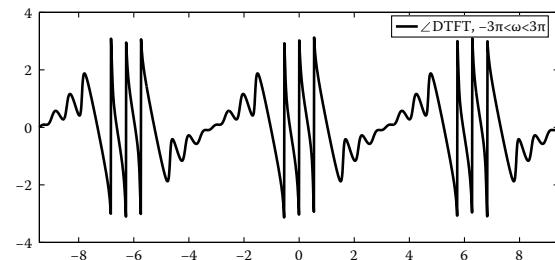
```



```

w1 = -3*pi:.01:3*pi;
XX = subs(X,w,w1);
plot(w1,angle(XX));
legend('\angle DTFT, -3\pi<\omega<3\pi')
xlim([-3*pi 3*pi]);

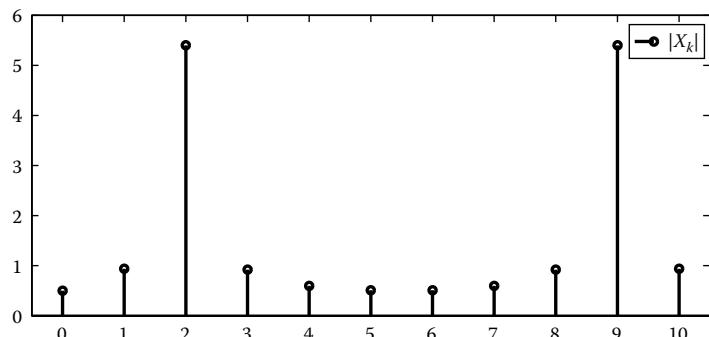
```



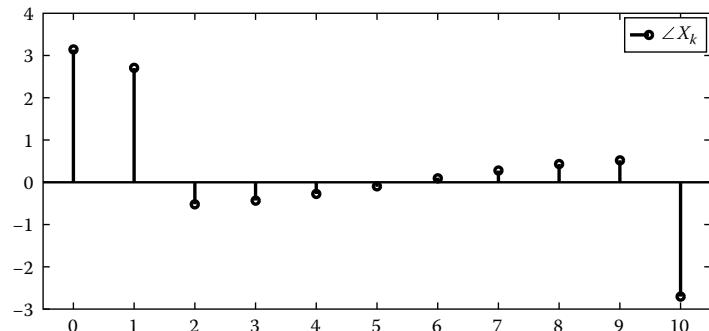
Problem 2 Plot the magnitude, the angle, the real part, and the imaginary part of the DFT X_k of the signal $x[n] = \cos(\pi n/3)$, $0 \leq n \leq 10$.

Solution

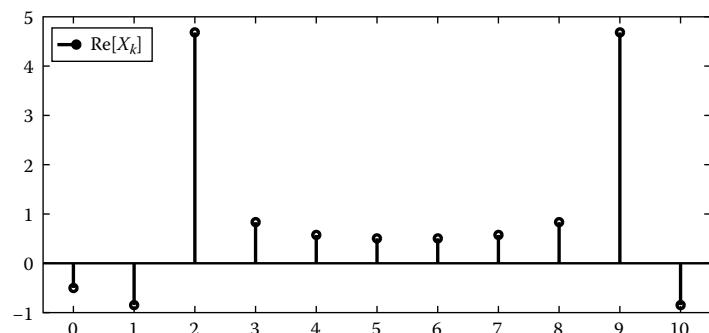
```
n=0:10;
x=cos( (1/3)*pi*n);
Xk=fft(x);
N=length(x);
k=0:N-1;
stem(k,abs(Xk));
xlim([- .5 10.5]);
legend('|X_k|')
```



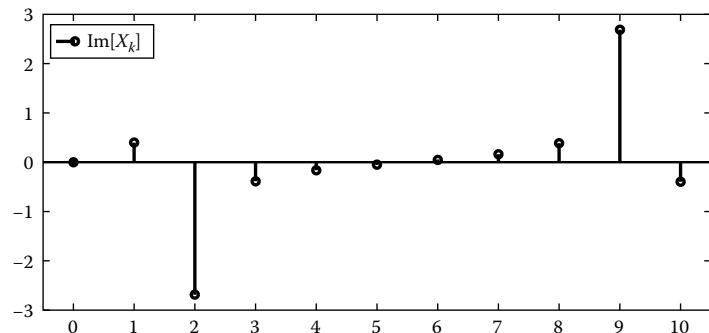
```
stem(k,angle(Xk));
xlim([- .5 10.5]);
legend('\angle X_k')
```



```
stem(k,real(Xk));
xlim([- .5 10.5]);
legend('RE[X_k]')
```



```
stem(k,imag(Xk));
xlim([- .5 10.5]);
legend('IM[X_k]')
```



Problem 3 Using the signal $x[n] = \cos(\pi n/3)$, $0 \leq n \leq 8$ confirm that

- a. $X_k = |X_k|e^{j\angle X_k}$ and
- b. $X_k = \text{Re}\{X_k\} + j \text{Im}\{X_k\}$,

where $|X_k|$ is the magnitude, $\angle X_k$ the phase, $\text{Re}\{X_k\}$ the real part, and $\text{Im}\{X_k\}$ the imaginary part of the DFT X_k of the signal $x[n]$.

Solution

```

n=0:8;
x=cos( (1/3)*pi*n);
Xk=fft(x);
Xk.'
ans = 1.0000    1.0000+2.4161i
      1.0000-3.0176i   1.0000-0.8660i
      1.0000-0.2376i   1.0000+0.2376i
      1.0000+0.8660i   1.0000+3.0176i
      1.0000-2.4161i

ans = 1.0000    1.0000+2.4161i
      1.0000-3.0176i   1.0000-0.8660i
a. A=abs(Xk).*exp(j*angle(Xk));
A.'
      1.0000-0.2376i   1.0000+0.2376i
      1.0000+0.8660i   1.0000+3.0176i
      1.0000-2.4161i

ans = 1.0000    1.0000+2.4161i
      1.0000-3.0176i   1.0000-0.8660i
b. B=real(Xk)+j*imag(Xk);
B.'
      1.0000-0.2376i   1.0000+0.2376i
      1.0000+0.8660i   1.0000+3.0176i
      1.0000-2.4161i

```

Problem 4

- a. Write a function that computes the circular convolution of two sequences that have the same number of samples.
- b. Compute through your function the circular convolution of the sequences $x_1[n] = [1, 2, 3, 4, 5]$ and $x_2[n] = [5, 4, 3, 2, 1]$.
- c. Confirm your result by using the commands `fft - ifft`.

Solution

```

a.
function y=circonv(x,h)
N=length(x);
for m=0:N-1
    hs(1+m)=h(1+mod(-m,N));
end
for n=0:N-1
    hsn=circshift(hs.',n);
    y(n+1)=x*hsn;
end

b.
x1=[1 2 3 4 5];      y = 45 40 40 45 55
x2=[ 5 4 3 2 1];
y=circonv(x1,x2)

c.
X1=fft(x1);          y = 45 40 40 45 55
X2=fft(x2);
y=ifft(X1.*X2)

```

Problem 5 Compute the circular convolution of the sequences $x_1[n]=[1,2,3,4]$ and $x_2[n]=[5,4,3,2,1]$

- In the discrete-time domain
- With use of the commands `fft` and `ifft`

Solution

```

a.
x1=[1 2 3 4];
x2=[ 5 4 3 2 1];
x1=[x1 0];
N=length(x1);
for m=0:N-1
    x2s(1+m)=x2(1+mod(-m,N));
end
for n=0:N-1
    x2sn=circshift(x2s',n);
    y(n+1)=x1*x2sn;
end
y

b.
X1=fft(x1,N);          y = 25      25      30      40      30
X2=fft(x2,N);
y=ifft(X1.*X2)

```

Problem 6 Calculate the linear convolution of the sequences $x_1[n] = [1, 2, 3, 4]$ and $x_2[n] = [5, 4, 3, 2, 1]$

- At the discrete-time domain by using the command *conv*
- With use of circular convolution
- With use of commands *fft* and *ifft*

Solution

```

a.
x1=[1 2 3 4];
y = 5      14     26     40     30     20     11     4
x2=[ 5 4 3 2 1];
y=conv(x1,x2)

b.
N1=length(x1);      y = 5      14     26     40     30     20     11     4
N2=length(x2);      where circonv.m is the function created in Problem 4.
M=N1+N2-1;
x1(M)=0;
x2(M)=0;
y=circonv(x1,x2)

c.
X1=fft(x1,M);      y = 5.0000    14.0000   26.0000   40.0000   30.0000
X2=fft(x2,M);      20.0000    11.0000    4.0000
y=ifft(X1.*X2)

```

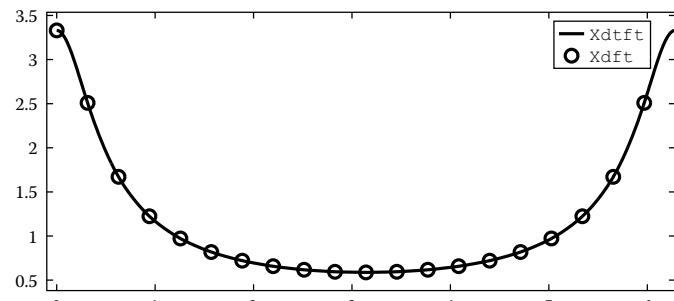
Problem 7 Consider the sequence $x[n] = 0.7^n$, $0 \leq n \leq 19$. Plot in the same graph the DTFT of $x[n]$ over the frequency interval $0 \leq \omega \leq 2\pi$ and the DFT of $x[n]$ versus the frequency points $\omega_k = 2\pi k/N$, $k = 0, 1, \dots, N - 1$.

Solution

```

n=0:19;
x=0.7.^n;
syms w
Xdtft=sum(x.*exp(-j*w*n));
Xdft=dft(x);
N=length(Xdft);
k=0:N-1;
wk=2*pi*k/N;
ezplot(abs(Xdtft), [0 2*pi]);
hold on
plot(wk,abs(Xdft),'o')
legend('Xdtft', 'Xdft')

```



Problem 8

- Write a function that computes the circular convolution of two sequences that do not necessarily have the same number of elements.
- Write a function that computes the circular convolution of two sequences with use of the commands `fft` and `ifft`.
- Compute through your functions the circular convolution of $x_1[n] = [1, 2, 3, 4]$ and $x_2[n] = [5, 4, 3, 2, 1]$.

Solution

<pre>a. function y=circonv2(x,h) N1=length(x); N2=length(h); N=max(N1,N2); if N>N1 x=[x zeros(1,N-length(x))]; end if N> N2 h=[h zeros(1,N-length(h))]; end for m=0:N-1 p(m+1)=mod(-m,N); hs(1+m)=h(1+p(m+1)); end for n=0:N-1 hsn=circshift(hs.',n); y(n+1)=x*hsn;hs.'; end c. x1=[1,2,3,4]; x2=[5,4,3,2,1]; y1=circonv2(x1,x2) y2=circonv3(x1,x2)</pre>	<pre>b. function y=circonv3(x,h); N1=length(x); N2=length(h); N=max(N1,N2); X=fft(x,N); H=fft(h,N); y=ifft(X.*H);</pre>
	$y_1 = 25 \quad 25 \quad 30 \quad 40 \quad 30$ $y_2 = 25 \quad 25 \quad 30 \quad 40 \quad 30$

Problem 9 Calculate the execution time of the functions `circonv2.m` and `circonv3.m` that were created in the previous problem. Use the sequences $x[n] = 1.01^n$, $0 \leq n \leq 1000$ and $x_2[n] = \cos(n)$, $0 \leq n \leq 900$.

Solution

<pre>n1=0:10e3; x1=1.01.^n1; n2=0:9e2; x2=cos(n2); disp ('circonv2') tic y=circonv2(x1,x2); toc disp ('circonv3') tic y=circonv3(x1,x2); toc bench</pre>	<p>The script M-File <code>bench.m</code> is created and executed from the command prompt.</p> <p>As expected the <code>fft-ifft</code> implementation is faster than the for-loop implementation.</p> <p><code>circonv2</code> Elapsed time is 14.055622 s.</p> <p><code>circonv3</code> Elapsed time is 0.378998 s.</p>
--	---

Problem 10 Compute the energy of the discrete-time signal $x[n] = 1.01^n$, $0 \leq n \leq 100$

- At the discrete-time domain
- From the DFT X_k of $x[n]$
- From the DTFT $X(\omega)$ of $x[n]$

Solution

In this problem, Parseval's theorem is applied for discrete-time signals. More specifically, the energy of an N -point discrete-time signal $x[n]$ is given by $E = \sum_{n=0}^{N-1} |x[n]|^2$. Parseval's theorem states that the energy can be computed from the DFT X_k of $x[n]$ according to $E = (1/N) \sum_{k=0}^{N-1} |X_k|^2$. Finally, recall from Section 7.3 that energy can be computed from the DTFT $X(\omega)$ of $x[n]$ as $E = (1/2\pi) \int_{2\pi} |X(\omega)|^2 d\omega$.

<code>n = 0:1e2;</code>	Energy computed at the discrete-time domain.
<code>x = 1.01.^n;</code>	<code>En = 321.5458</code>
<code>En = sum(abs(x).^2)</code>	
<code>N = length(x);</code>	Energy computed according to Parseval's
<code>X = fft(x);</code>	theorem from the DFT of $x[n]$.
<code>Edft = (1/N) * sum(abs(X).^2)</code>	<code>Edft = 321.5458</code>
<code>syms w</code>	
<code>X = sum(x.*exp(-j*w*n));</code>	Energy computed according to Parseval's
<code>Esd = abs(X)^2</code>	theorem from the DTFT of $x[n]$.
<code>E = 1/(2*pi) * int(Esd,w,-pi,pi)</code>	<code>Edtft = 321.5458</code>
<code>Edtft = eval(E)</code>	

Problem 11

- Write a function that computes the linear convolution of two sequences with use of the commands `fft` and `ifft`.
- Compute using your function, the linear convolution of the sequences $x_1[n] = [1, 2, 3, 4]$, $0 \leq n \leq 3$ and $x_2[n] = [7, 6, 5, 4, 3, 2, 1]$, $0 \leq n \leq 6$.
- Confirm your result by using the command `conv`.

Solution

a. <code>function y = linconv(x, h);</code>	
<code> N1 = length(x);</code>	
<code> N2 = length(h);</code>	
<code> N = N1 + N2 - 1;</code>	
<code> X = fft(x, N);</code>	
<code> H = fft(h, N);</code>	
<code> y = ifft(X.*H);</code>	
b. <code>x1 = 1:4;</code>	<code>y = 7.0000 20.0000 38.0000</code>
<code>x2 = 7:-1:1;</code>	<code> 60.0000 50.0000 40.0000</code>
<code>y = linconv(x1, x2)</code>	<code> 30.0000 20.0000 11.0000 4.0000</code>
c. <code>y = conv(x1, x2)</code>	<code>y = 7 20 38 60 50 40 30 20 11 4</code>

7.14 Homework Problems

1. Consider the signal $x[n] = 0.9^n$.
 - a. Compute and plot the DTFT of the signal $x[n]u[n]$.
 - b. Compute and plot the DTFT of the signal $x[n]u[n-5]$.
 - c. Compute and plot the DTFT of the signal $x[-n]u[-n]$.
2. Verify the DTFT pair $(n+1)a^n u[n] \leftrightarrow 1/(1 - ae^{-j\omega})^2$, where $|a| < 1$.
3. Verify that the DTFT $X(\omega)$ of the signal $x[n] = \sin(\omega_0 n)/\pi n$ is

$$X(\omega) = \begin{cases} 0, & -\pi \leq \omega < -\omega_0 \\ 1, & -\omega_0 \leq \omega \leq \omega_0 \\ 0, & \omega_0 \leq \omega < \pi \end{cases}, \text{ where } |\omega_0| < \pi.$$

4. Compute the energy of the signal $x[n] = \sin(2.5n)/\pi n$.
5. Compute and plot the DTFTs of the rectangular pulses

$$x_1[n] = \begin{cases} 1, & -3 \leq n \leq 3 \\ 0, & \text{elsewhere} \end{cases}$$

and

$$x_2[n] = \begin{cases} 1, & -10 \leq n \leq 10 \\ 0, & \text{elsewhere} \end{cases}.$$

6. Consider a sequence $x[n]$ of $N = 10$ samples. Find the DFT of $x[n]$ when

$$x[n] = \begin{cases} 1, & n = 0 \\ 0, & \text{elsewhere} \end{cases}$$

and when

$$x[n] = \begin{cases} 1, & n = 4 \\ 0, & \text{elsewhere} \end{cases}.$$

7. Compute the N -point DFT of the rectangular pulse $x[n] = \begin{cases} 1, & 0 \leq n \leq 9 \\ 0, & \text{elsewhere} \end{cases}$ when $N = 10$, $N = 5$, and $N = 15$.
8. Compute and plot the DTFT and the DFT of the sequence $x[n] = 3^{-n}$, $0 \leq n \leq 20$ at the frequency interval $-\pi \leq \omega \leq \pi$ rad/s.
9. Plot the sequence $x[n] = n^2$, $0 \leq n \leq 10$ and the circularly shifted sequences $x_{c,2}[n]$, $x_{c,5}[n]$, and $x_{c,8}[n]$.
10. Compute the circular convolution of the sequences $x[n] = 1/(n+1)$, $0 \leq n \leq 10$ and $h[n] = n$, $0 \leq n \leq 15$.
11. Calculate the linear convolution of the sequences $x[n] = 1/(n+1)$, $0 \leq n \leq 8$ and $h[n] = \sqrt{n}$, $0 \leq n \leq 12$ with use of circular convolution and also by using the commands `fft` and `ifft`.

12. Write a function that accepts a sequence $x[n]$ and a number N as input arguments and computes the N -point DFT of $x[n]$.
13. Write a function that accepts a sequence $x[n]$ and the time n as input arguments and computes and plots the DTFT of $x[n]$.
14. Write a function that accepts a sequence X_k and a number N as input arguments and computes the N -point IDFT of X_k .
15. Verify that if X_k is the DFT of a real N -point sequence $x[n]$, then $X_k = X_{N-k}^*$.

This page intentionally left blank

8

Frequency Response

In this chapter, we introduce the concept of a system frequency response. First, we discuss the frequency response of continuous-time systems, while the discrete-time counterpart follows.

8.1 Continuous-Time Frequency Response

A system is completely specified by its impulse response $h(t)$. The system response $y(t)$ to an input signal $x(t)$ is computed by convoluting the impulse response with the input signal; that is, the system response is given by

$$y(t) = h(t) * x(t). \quad (8.1)$$

Applying Fourier transform to both sides of (8.1) yields

$$Y(\Omega) = H(\Omega)X(\Omega), \quad (8.2)$$

where

$X(\Omega)$ and $Y(\Omega)$ are the Fourier transforms of the input signal and of the output signal, respectively

$H(\Omega)$ is called *frequency response*, and is the Fourier transform of the impulse response $h(t)$ of the system

The frequency response is a complete description of a system in the (cyclic) frequency domain. The mathematical expression of the frequency response is straightforwardly computed from Equation 8.2 as

$$H(\Omega) = F\{h(t)\} = \frac{Y(\Omega)}{X(\Omega)}. \quad (8.3)$$

The frequency response $H(\Omega)$ is usually a complex-valued function, and this is why sometimes it is denoted by $H(j\Omega)$. As any complex function, $H(\Omega)$ can be written in the form

$$H(\Omega) = |H(\Omega)|e^{j\angle H(\Omega)}, \quad (8.4)$$

where

$|H(\Omega)|$ is the magnitude of $H(\Omega)$

$\angle H(\Omega)$ is the phase of $H(\Omega)$

Example

Compute the frequency response $H(\Omega)$ of a system described by the impulse response $h(t) = e^{-2t}u(t)$. Plot the frequency response magnitude and phase for $0 \leq \Omega \leq 10$ rad/s. Finally verify Equation 8.4.

Commands	Results	Comments
<pre>syms t w h=exp(-t)*heaviside(t)</pre>	$h = \exp(-t) * \text{heaviside}(t)$	Definition of the system impulse response $h(t)$.
<pre>H=fourier(h,w)</pre>	$H = 1 / (1 + i * w)$	The frequency response $H(\Omega)$ is the Fourier transform of $h(t)$.
<pre>w1=0:.1:10; HH=subs(H,w,w1); plot(w1,abs(HH)); title('Frequency response magnitude')</pre>		The magnitude of $H(\Omega)$ is plotted in the frequency interval $0 \leq \Omega \leq 10$. The magnitude is computed using the command <code>abs</code> .
<pre>plot(w1,angle(HH)) title('phase H(\Omega) in radians')</pre>		The phase of $H(\Omega)$ is plotted in the same frequency interval. The command <code>angle</code> is used to compute the phase. The result is in radians.
<pre>dif=HH-abs(HH).*exp(j*angle(HH)); subplot(211) plot(w1,real(dif)); ylim([-1e-7 1e-7]) legend('real part') subplot(212) plot(w1,imag(dif)); ylim([-1e-7 1e-7]) legend('imaginary part')</pre>		The result (real and imaginary part) of the operation $H(\Omega) - H(\Omega) e^{j\angle H(\Omega)}$ is plotted. Since it is practically zero, Equation 8.4 is verified.

Example

The response of a system to the input signal $x(t) = e^{-3t}u(t)$ is $y(t) = te^{-3t}u(t)$. Compute

- The frequency response $H(j\Omega)$ of the system.
- The impulse response $h(t)$ of the system. Verify your result by computing the convolution between $x(t)$ and $h(t)$.

Commands	Results	Comments
<pre>syms t w x=exp(-3*t)*heaviside(t); y=t*exp(-3*t)*heaviside(t);</pre>		The input signal $x(t) = e^{-3t}u(t)$ and the output signal $y(t) = te^{-3t}u(t)$ are defined as symbolic expressions.
<pre>X=fourier(x,w); Y=fourier(y,w);</pre>		Computation of the Fourier transforms $X(\Omega)$ and $Y(\Omega)$ of the input and output signals $x(t)$ and $y(t)$, respectively.
<pre>H=Y/X</pre>	$H = 1/(3+i*w)$	The frequency response is given by $H(\Omega) = Y(\Omega)/X(\Omega)$.
<pre>h=ifourier(H,t)</pre>	$h = \exp(-3*t)*heaviside(t)$	The impulse response $h(t)$ is computed as the inverse Fourier transform of the frequency response; that is, $h(t) = F^{-1}\{H(\Omega)\}$.

To verify our results, the output is calculated by the convolution between the input signal $x(t) = e^{-3t}u(t)$ and the obtained impulse response $h(t) = e^{-3t}u(t)$. For reference reasons the output signal $y(t) = te^{-3t}u(t)$ is plotted in the figure below.

Commands	Results	Comments
<pre>syms t y=t*exp(-3*t)*heaviside(t); legend('y(t)') ylim([0 0.15])</pre>		The output signal $y(t) = te^{-3t}u(t)$.
<pre>t=0:.01:5; x=exp(-3*t); h=x; y=conv(x,h)*.01; plot(0:.01:10,y); legend('x(t)*h(t)') ylim([0 0.15])</pre>		The output derived from the convolution between the input and impulse response is same as $y(t)$; hence, the impulse response was computed correctly.

From Equations 8.2 and 8.4, we easily derive the following two important relationships:

$$|Y(\Omega)| = |H(\Omega)||X(\Omega)| \quad (8.5)$$

and

$$\angle Y(\Omega) = \angle H(\Omega) + \angle X(\Omega). \quad (8.6)$$

Finally, we should note that there are books that refer to the frequency response as transfer function. In this book, the term transfer function refers solely to the Laplace transform of the impulse response and is defined in the s (or complex frequency) domain, while the term frequency response is used exclusively when we refer to the Fourier transform of a system's impulse response.

8.2 The Command `freqs`

The frequency response of a system is usually expressed in rational form, i.e., it is given in the form

$$H(j\Omega) = \frac{B(j\Omega)}{A(j\Omega)} = \frac{b_n(j\Omega)^n + b_{n-1}(j\Omega)^{n-1} + \cdots + b_1(j\Omega) + b_0}{a_m(j\Omega)^m + a_{m-1}(j\Omega)^{m-1} + \cdots + a_1(j\Omega) + a_0}. \quad (8.7)$$

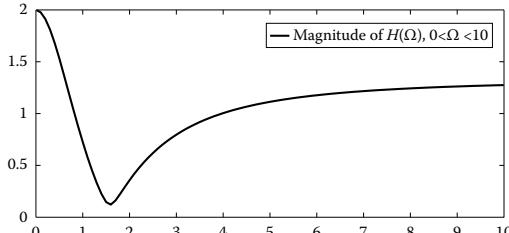
In this case, a very easy way to compute and plot the frequency response $H(\Omega)$ (denoted also by $H(j\Omega)$) is available in MATLAB®. The command that computes and plots $H(j\Omega)$ is the command `freqs`. Its syntax is `H = freqs(num, den, w)`, where `num` is the vector of the coefficients of the numerator polynomial, `den` is the corresponding vector for the denominator, and `w` is the vector of the frequencies at which the frequency response H is evaluated.

Example

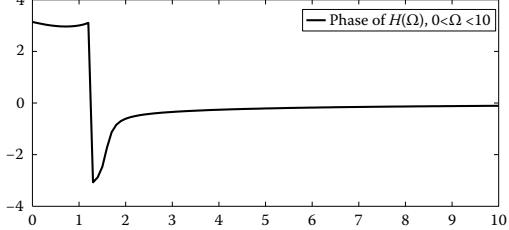
Plot the magnitude and the phase of the frequency response

$$H(j\Omega) = \frac{B(j\Omega)}{A(j\Omega)} = \frac{8(j\Omega)^2 + 2(j\Omega) + 20}{6(j\Omega)^2 - 5(j\Omega) - 10}$$

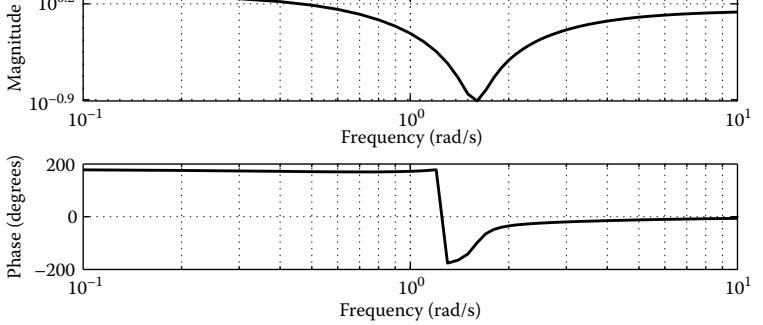
in the frequency interval $0 \leq \Omega \leq 10$ rad/s.

Commands	Results/Comments
<code>num = [8 2 20];</code> <code>den = [6 -5 -10];</code> <code>w = 0 : .1 : 10;</code>	The vectors of the coefficients of the numerator and denominator polynomials as well as the frequency vector are defined. Notice that the coefficients given in <code>num</code> and <code>den</code> must correspond to powers of $j\Omega$.
<code>H = freqs (num, den, w)</code>	The frequency response $H(j\Omega)$ is evaluated at the points specified in the frequency vector.
<code>plot(w, abs(H));</code> <code>legend('Magnitude of H(\Omega), 0 < \Omega < 10')</code>	The magnitude of $H(j\Omega)$ is plotted for $0 \leq \Omega \leq 10$ rad/s.
	 <p>Magnitude of $H(\Omega)$, $0 < \Omega < 10$</p>

(continued)

Commands	Results/Comments
<pre>plot(w,angle(H)); legend('Phase of H(\Omega)', '0<\Omega <10')</pre>	The phase of $H(j\Omega)$ is plotted for $0 \leq \Omega \leq 10$ rad/s. 

An alternative syntax for the command `freqs` is given below. If no output variable is specified, the magnitude and the angle of $H(j\Omega)$ are directly plotted in the same figure. The magnitude is plotted in dB, while the phase is plotted in degrees. Additionally, the frequency axis is logarithmically scaled.

Commands	Results/Comments
<pre>freqs(num, den, w)</pre>	

If no frequency vector is specified, i.e., the syntax is `freqs(num, den)`, then by default a 200 sample vector with values from 0 to 10 is considered as frequency vector. If a different number of samples is required (e.g., L samples), the appropriate syntax is `freqs(num, den, L)`. Finally, using the syntax `[H, w] = freqs(num, den, L)` except from the frequency response vector, we also define the frequency vector.

Commands	Results	Comments
<pre>[H, w] = freqs(num, den); length(w)</pre>	<pre>ans = 200</pre>	The frequency response and frequency vectors consist of 200 elements. The frequency is defined over the interval $0 \leq \Omega \leq 10$ rad/s.
<pre>[H, w] = freqs(num, den, 722) length(w)</pre>	<pre>ans = 722</pre>	The frequency response and frequency vectors consist of 722 elements. The frequency Ω is again defined over the interval $0 \leq \Omega \leq 10$ rad/s.

To properly define a frequency response of a system in MATLAB, the coefficients given in *num* and *den* must correspond to powers of $j\Omega$. Hence, if a term in (8.7) is not of the form $a_n(j\Omega)^n$, it has to be converted into the proper form by modifying its coefficient. Finally, if a term of $j\Omega$ is not present, a zero must be placed in the corresponding position of the coefficients vector.

Example

Plot in the frequency interval $0 \leq \Omega \leq 20$ rad/s the frequency response

$$H(j\Omega) = \frac{\Omega^3 + \Omega^2 - 5\Omega + 1}{3\Omega^2 - 1}.$$

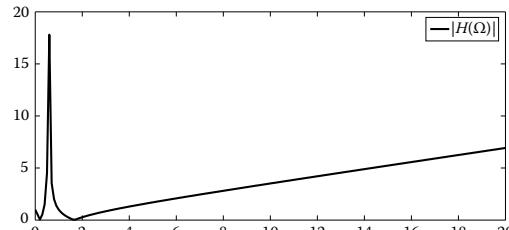
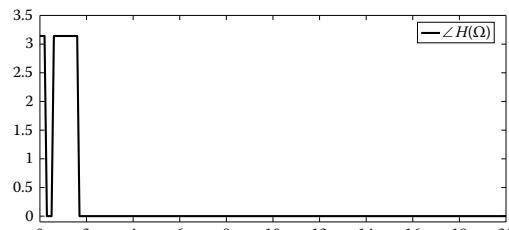
First, $H(j\Omega)$ is converted into the appropriate form (given in (8.7)); that is, $H(j\Omega)$ is written as

$$H(j\Omega) = \frac{\Omega^3 + \Omega^2 - 5\Omega + 1}{3\Omega^2 - 1} = \frac{1/j^3(j\Omega)^3 + 1/j^2(j\Omega)^2 - 5/j(j\Omega) + 1}{3/j^2(j\Omega)^2 + 0j\Omega - 1}.$$

Commands	Results	Comments
<pre> num=[1/(j^3) 1/(j^2) -5/j 1]; den=[3/(j^2) 0 -1]; w=0:.1:20; H=freqs(num,den,w); plot(w,abs(H)); legend(' H(\Omega) ') </pre>	<pre> num=0+1.00i -1.00 0+5.00i 1.00 den=-3 0 -1 </pre> <p>The magnitude plot shows a sharp peak at $\Omega = 0$ reaching approximately 17. As Ω increases, the magnitude decreases rapidly, crossing zero at $\Omega \approx 1.8$, and then increases monotonically towards 7 at $\Omega = 20$.</p>	<p>The obtained result is expected as $j^2 = -1$, $j^3 = -j$, and $1/j = -j$. Hence, $1/j^2 = 1/(-1) = -1$ and $1/j^3 = -1/j = -j/j^2 = -j/(-1) = j$.</p>
<pre> plot(w,angle(H)); ylim([-0.5 3.5]); legend('angle H(\Omega)') </pre>	<p>The phase plot shows a step function starting at π (approximately 3.14) for $\Omega < 2$ and dropping to 0 for $\Omega > 2$.</p>	

To confirm the result obtained by the *freqs* command, the magnitude and the phase of a frequency response $H(j\Omega)$ are computed and plotted in the way that normal functions are computed and plotted in MATLAB. More specifically, first the frequency interval is defined and $H(j\Omega)$ is defined according to its mathematical expression with Ω as the

independent variable. Next, the magnitude and phase of $H(j\Omega)$ are plotted in the defined frequency interval.

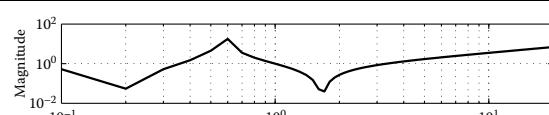
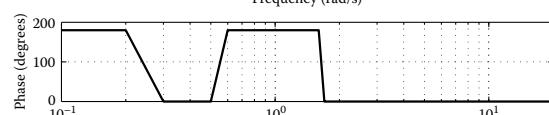
Commands	Results/Comments
<pre>w = 0 : .1 : 20; H = (w.^3+w.^2-5*w+1) ./ (3*w.^2-1);</pre>	<p>The frequency response $H(j\Omega) = (\Omega^3 + \Omega^2 - 5\Omega + 1)/(3\Omega^2 - 1)$ is defined in the frequency interval $0 \leq \Omega \leq 20$ rad/s.</p> 
<pre>plot(w,abs(H)); legend(' H(\Omega) ')</pre> <pre>plot(w,angle(H)); ylim([-0.5 3.5]); legend('\angle H(\Omega)')</pre>	

The graphs obtained using the “classic” approach are similar to the ones obtained by the `freqs` command. Hence, the two ways are equivalent.

Remark 1: A number A expressed in volts is converted to dB according to the relationship: $A_{dB} = 20 \log_{10}(A)$.

Remark 2: Radians are converted to degrees according to the relationship: $A_{degrees} = A_{radians}(180/\pi)$.

Finally, recall that using the command `loglog` instead of `plot` returns the graph of a function with both axes in logarithmic scale. Alternatively, the commands `semilogx` and `semilogy` plot the function with only one logarithmically scaled axis.

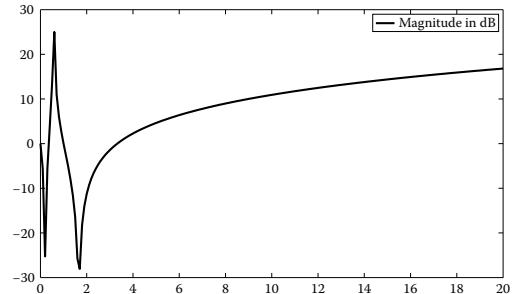
Commands	Results/Comments
<pre>freqs(num,den,w);</pre>	 

(continued)

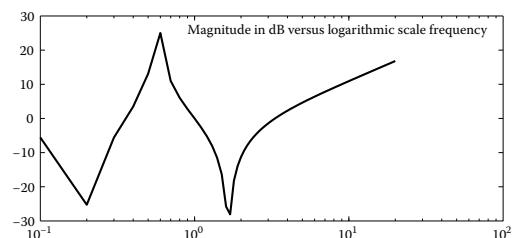
(continued)

Commands**Results/Comments**

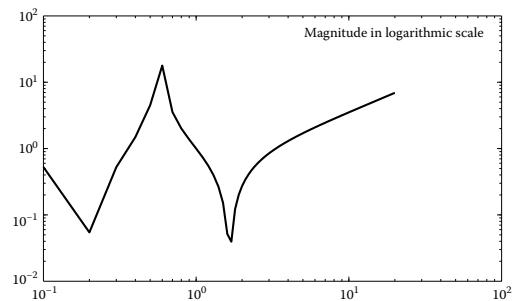
```
plot(w,20*log10(abs(H)));
legend('Magnitude in dB')
```



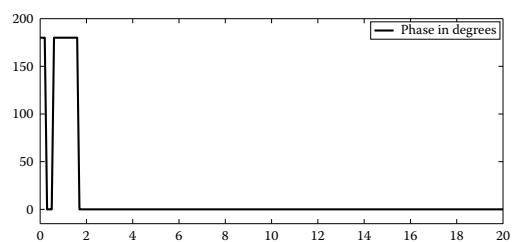
```
semilogx(w,20*log10(abs(H)));
title('Magnitude in dB versus logarithmic scale frequency')
```



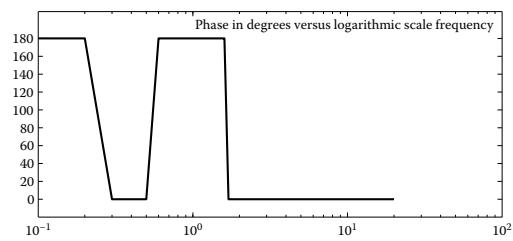
```
loglog(w,abs(H));
title('Magnitude in logarithmic scale')
```



```
plot(w,angle(H)*180/pi);
ylim([-15 200]);
legend('Phase in degrees')
```



```
semilogx(w,angle(H)*180/pi)
ylim([-20 200]);
title('Phase in degrees versus logarithmic scale frequency')
```



8.2.1 The Command `invfreqs`

It is obvious from the name of the command that command `invfreqs` is used for the inverse operation from command `freqs`. More specifically, the frequency response vector H and the corresponding vector of frequencies w are now known (i.e., are the input arguments), and the numerator and denominator of the frequency response are computed through the command `invfreqs`. The syntax is $[num, den] = \text{invfreqs}(H, w, N, M)$, where H is the frequency response vector, w is the vector of the corresponding frequencies, while N and M specify the order of the numerator and denominator polynomials of the frequency response $H(j\Omega)$. The output arguments num and den are the computed coefficients of the frequency response polynomials $B(j\Omega)$ and $A(j\Omega)$ (numerator and denominator, respectively).

Example

A system is described by the frequency response

$$H(j\Omega) = \frac{-2\Omega^2 + 7j\Omega}{3(j\Omega)^2 + 2}.$$

Compute and plot the magnitude and the phase of $H(j\Omega)$, and verify your result by using the `invfreqs` command.

Commands	Results/Comments						
Definition and graph of $H(j\Omega)$.							
<pre>num = [-2/j^2 7 0]; den = [3 0 2]; w = 0:.1:10; H = freqs(num, den, w); freqs(num, den, w)</pre>							
Multiplying the numerator and the denominator by 3 we verify the result, i.e., the coefficients obtained by the command <code>invfreqs</code> are same as the coefficients of $H(j\Omega)$.							
<pre>[num1, den1] = invfreqs(H, w, 2, 2)</pre> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">num1 = 0.6667</td> <td style="padding: 2px;">2.3333</td> <td style="padding: 2px;">0.0000</td> </tr> <tr> <td style="padding: 2px;">den1 = 1.0000</td> <td style="padding: 2px;">-0.0000</td> <td style="padding: 2px;">0.6667</td> </tr> </table>		num1 = 0.6667	2.3333	0.0000	den1 = 1.0000	-0.0000	0.6667
num1 = 0.6667	2.3333	0.0000					
den1 = 1.0000	-0.0000	0.6667					

The `invfreqs` command can be used to specify equivalent frequency responses when we are given the evaluated (at the frequencies of the vector w) frequency response H .

Example

Derive equivalent frequency responses of

$$H(j\Omega) = \frac{-2\Omega^2 + 7j\Omega}{3(j\Omega)^2 + 2}$$

(which is the frequency response of the previous example) that

- The polynomials of the numerator and denominator are of third order.
- The numerator polynomial is of tenth order and the denominator polynomial is of seventh order.

Commands	Results/Comments
----------	------------------

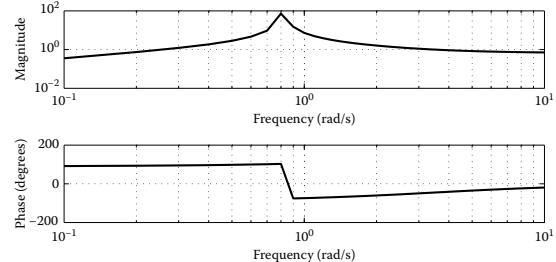
The derived frequency response is

$$H_2(j\Omega) = \frac{0.3333(j\Omega)^3 + 0.1802(j\Omega)^2 - 0.9729j\Omega}{(j\Omega)^3 - 1.4594(j\Omega)^2 + 0.6667j\Omega - 0.9729}.$$

$$\begin{array}{cccccc} \text{num2} & = & 0.3333 & 0.1802 & -0.9729 & 0.0000 \\ \text{den2} & = & 1.0000 & -1.4594 & 0.6667 & -0.9729 \end{array}$$

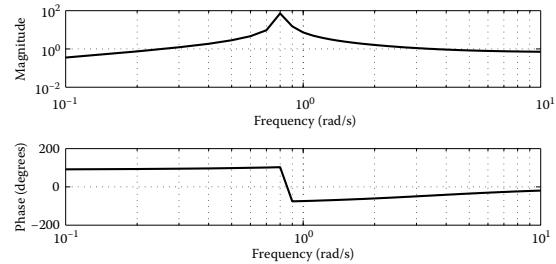
The frequency response $H_2(j\Omega)$ is same as $H(j\Omega)$ and the two frequency responses are equivalent (at least for $0 \leq \Omega \leq 10$ rad/s).

`freqs(num2, den2, w)`



The frequency response $H_3(j\Omega)$ where the numerator polynomial is of tenth order and the denominator polynomial is of seventh order is also equivalent to $H(j\Omega)$ and $H_2(j\Omega)$.

`[num3, den3] = invfreqs(H, w, 10, 7); freqs(num3, den3, w)`



Remark

The command `invfreqs` computes the mathematical expression of a frequency response according to the least-squares method. Thus, the result obtained from this command is an approximation and it is not always accurate.

8.3 The Command `lsim`

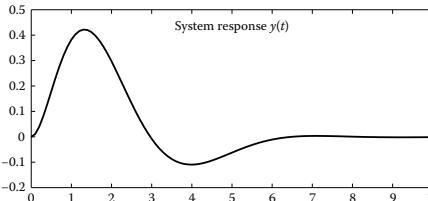
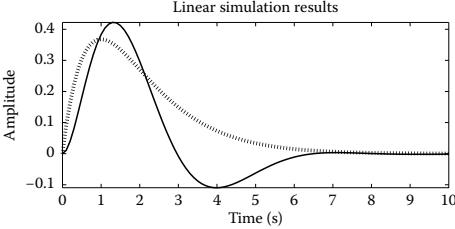
In this section, we introduce an easy way to compute the system response $y(t)$ to an arbitrary input signal $x(t)$ if the frequency response of the system $H(j\Omega)$ is known. The related command is the command `lsim`. The syntax is $y = \text{lsim}(\text{num}, \text{den}, x, t)$, where num is the vector of the coefficients of the numerator polynomial of the frequency response $H(j\Omega)$, den is the corresponding vector for the denominator of $H(j\Omega)$, x is the input signal, and t is the time in which the input signal is applied and the output signal is computed. The coefficients given in num and den must correspond to powers of $j\Omega$. Hence, if a term of $H(j\Omega)$ is not of the form $a_n(j\Omega)^n$ it has to be converted into the proper form by modifying its coefficient. There are three problems that might arise when using the `lsim` command. The first is that unlike the `freqs` command the input vectors num and den cannot contain complex coefficients. The solution is to substitute j^2 by -1 when appropriate. The second is that the order of the numerator polynomial has to be smaller or equal to the order of the denominator polynomial, while the third one is that an initial variation takes place in the output signal computation for a short period of time. Finally, we note that if we do not specify an output argument; that is, if we use the syntax $\text{lsim}(\text{num}, \text{den}, x, t)$ the outcome of the command is a graph of the output signal $y(t)$ together with the input signal $x(t)$ in the specified time t .

Example

A system is described by the frequency response

$$H(j\Omega) = \frac{5j\Omega + 2}{2(j\Omega)^2 + 3j\Omega + 4}.$$

Compute and plot the response $y(t)$ of the system to the input signal $x(t) = te^{-t}$, $0 \leq t \leq 10$.

Commands	Results	Comments
<pre>num = [5 2]; den = [2 3 4]; t = 0 : 1 : 10; x = t.*exp(-t); y = lsim(num, den, x, t); plot(t, y); title('System response y(t)')</pre> <pre>lsim(num, den, x, t);</pre>		<p>The response $y(t)$ of the system to the input signal $x(t)$ is computed with the command <code>lsim</code>.</p>
		<p>Graph in the same figure of the input signal (depicted with the dotted line) and of the system response (depicted with the solid line) in the time interval $0 \leq t \leq 10$.</p>

Example

A system is described by the frequency response

$$H(j\Omega) = \frac{-2\Omega^2 + 7}{3(j\Omega)^2 + 2j\Omega + 1}.$$

Compute and plot the response $y(t)$ of the system to the input signal $x(t) = 1/(t+1), 0 \leq t \leq 20$.

First, we express $H(j\Omega)$ in the appropriate form, that is, in terms of $j\Omega$. Thus,

$$H(j\Omega) = \frac{-2/j^2(j\Omega)^2 + 0j\Omega + 7}{3(j\Omega)^2 + 2j\Omega + 1}.$$

But $j^2 = -1 \Rightarrow -2/j^2 = 2$. Hence, $H(j\Omega)$ becomes

$$H(j\Omega) = \frac{2(j\Omega)^2 + 0j\Omega + 7}{3(j\Omega)^2 + 2j\Omega + 1}.$$

Commands	Results	Comments
<pre>num = [2 0 7]; den = [3 2 1]; t = 0:.1:20; x = 1./(t+1); y=lsim(num,den,x,t); plot(t,y); title('System response y(t)')</pre>		<p>The response $y(t)$ of the system to the input signal $x(t)$ is computed with the command <code>lsim</code>. Notice the way that vector <code>num</code> is formulated.</p>
<pre>lsim(num,den,x,t);</pre>		<p>Graph in the same figure of the input signal $x(t)$ (with the dotted line) and of the system response $y(t)$ (with the solid line) in the time interval $0 \leq t \leq 20$.</p>

8.4 System Response to Sinusoidal Input

In this section, a fundamental relationship referring to the response of a system to a sinusoidal input signal is established. Suppose that a system is described by a frequency response $H(\Omega)$. The response of a system to the input signal $x(t) = A \cos(\Omega_0 t + \varphi)$ is given by

$$y(t) = A |H(\Omega_0)| \cos(\Omega_0 t + \varphi + \angle H(\Omega_0)), \quad (8.8)$$

where

$H(\Omega_0)$ is the value of $H(\Omega)$ at the frequency $\Omega = \Omega_0$
 $|H(\Omega_0)|, \angle H(\Omega_0)$ are the magnitude and phase of $H(\Omega_0)$, respectively

Correspondingly, the response of a system to the input signal $x(t) = A \sin(\Omega_0 t + \varphi)$ is given by

$$y(t) = A|H(\Omega_0)| \sin(\Omega_0 t + \varphi + \angle H(\Omega_0)). \quad (8.9)$$

The importance of these two relationships is comprehensible if one considers that by using Fourier series, any signal can be written as a sum of sinusoidal signals.

Example

A system is described by the frequency response

$$H(\Omega) = \frac{3(j\Omega)^2 + 4j\Omega + 2}{-\Omega^2 + j\Omega + 3}.$$

Compute and plot over the time interval $0 \leq t \leq 30$ the response of the system to the input signals (a) $x_1(t) = 3 \cos(t + \pi/3)$ and (b) $x_2(t) = \sin((\pi/2)t + \pi/4)$.

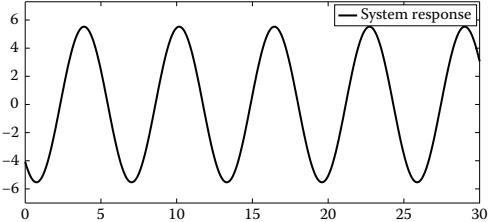
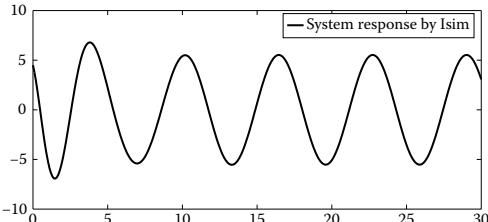
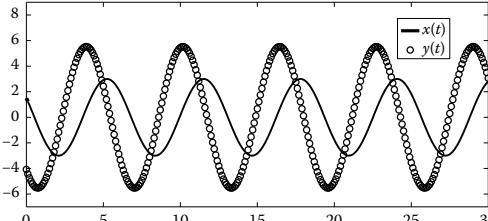
According to Equations 8.8 and 8.9, the system response to $x_1(t) = 3 \cos(t + \pi/3)$ is given by $y(t) = 3|H(1)| \cos(1 \cdot t + \pi/3 + \angle H(1))$, where $H(1)$ is the frequency response evaluated at $\Omega = \Omega_0 = 1$ rad/s and 3 is the amplitude of the input signal. Correspondingly, the system response to $x_2(t) = \sin((\pi/2)t + \pi/4)$ is given by $y(t) = 1|H(\pi/2)| \sin((\pi/2)t + \pi/4 + \angle H(\pi/2))$, where $H(\pi/2)$ is the frequency response evaluated at $\Omega = \Omega_0 = \pi/2$ and the amplitude of the input signal is 1. The result in both cases is verified by the command `lsim`. However, the coefficients of vectors `num` and `den` have to be real. Therefore, instead of `den = [-1/j^2 1 3]` (which is the coefficients vector of the denominator) we write `den = [1 3]`. Therefore,

- a. The system response to the input $x_1(t) = 3 \cos(t + \pi/3)$ is computed as follows.

Commands	Results/Comments
<pre>t = 0 : .1 : 30; x1 = 3 * cos(t + pi/3); plot(t, x1); legend('Input signal x(t)'); ylim([-4 4]);</pre>	<p>Definition and graph of the input signal $x_1(t) = 3 \cos(t + \pi/3)$.</p>

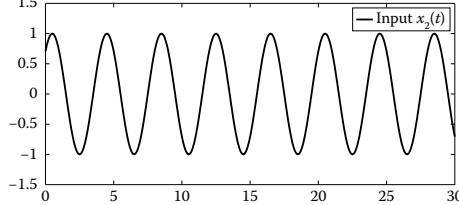
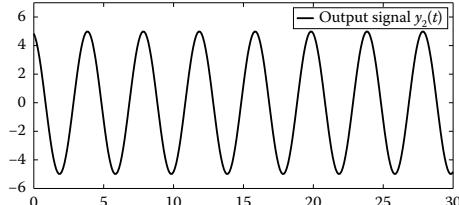
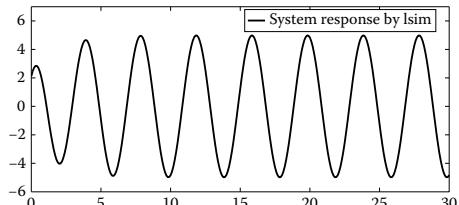
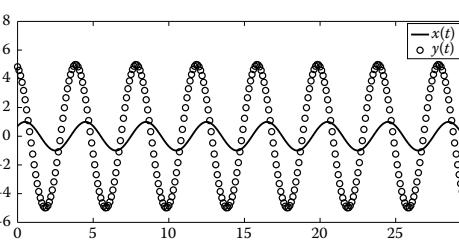
(continued)

(continued)

Commands	Results/Comments
w0 = 1; Hw0 = (3*(j*w0)^2 + 4*j*w0 + 2) / (-w0^2 + j*w0 + 3); magn = abs(Hw0); phas = angle(Hw0);	Evaluation of $H(\Omega_0)$. $Hw0 = 0.4000 + 1.8000i$ $magn = 1.8439$ $phas = 1.3521$
y1 = 3*abs(Hw0)*cos(t+pi/3+angle(Hw0)); plot(t,y1); legend('System response'); ylim([-7 7]);	Computation and graph of the output signal $y_1(t)$ according to Equation 8.8. 
num = [3 4 2]; den = [1 1 3]; y_ls = lsim(num, den, x1, t); plot(t, y_ls); legend('System response by lsim');	Computation and graph of the output signal $y_1(t)$ with use of the <code>lsim</code> command. Except from the expected initial variation, the result obtained from <code>lsim</code> is same as the one derived according to Equation 8.8. 
plot(t, x1, t, y1, 'o'); legend('x(t)', 'y(t)'); ylim([-7 9]);	Graph of input and output signals. 

From the last figure, the relationship (given in Equation 8.8) between the input and output signals becomes clear. The two signals have the *same frequency*, namely, $\Omega = \Omega_0 = 1$ rad/s. The amplitude of the output signal is $A|H(\Omega_0)| = 3 * 1.8439 = 5.5317$, and the phase of the output signal is $\varphi + \angle H(\Omega_0) = \pi/3 + 1.3521 = 2.3993$ rad.

- b. In order to compute the system response to the input signal $x_2(t) = \sin((\pi/2)t + \pi/4)$, we follow the same procedure.

Commands	Results/Comments
	Definition and graph of the input signal $x_2(t) = \sin((\pi/2)t + \pi/4)$.
<pre>t = 0 : .1 : 30; x2 = sin((pi/2)*t+pi/4); plot(t,x2); ylim([-1.5 1.5]); legend('input x_2(t)');</pre>	
<pre>w0=pi/2; Hw0=(3*(j*w0)^2+4*j*w0+2)/ (-w0^2+j*w0+3) mag=abs(Hw0) ph=angle(Hw0)</pre>	Evaluation of $H(\Omega_0)$. $Hw0 = 2.5417 + 4.3009i$ $mag = 4.9958$ $ph = 1.0370$
<pre>y2=1*mag*sin(w0*t+pi/4+ph); plot(t,y2); legend('output signal y_2(t)') ylim([-6 7]);</pre>	Computation and graph of the system response according to Equation 8.9.
	
<pre>num=[3 4 2]; den=[1 1 3]; y1s=lsim(num,den,x2,t); plot(t,y1s); legend('system response by lsim')</pre>	Computation and graph of the output signal with use of the <code>lsim</code> command. Except from the expected initial variation, the result obtained from <code>lsim</code> is same as the one derived according to Equation 8.9.
	
	Graph of input and output signals.
<pre>plot(t,x2,t,y1s,'o') legend('x(t)','y(t)') ylim([-6 8]);</pre>	

Also in this case, the conclusions about the input/output relationship are the same: The two signals have the same frequency $\Omega = \Omega_0 = \pi/2$, the amplitude of the output signal is $A|H(\Omega_0)| = 1*4.9958 = 4.9958$, and the phase of the output signal is $\varphi + \angle H(\Omega_0) = \pi/4 + 1.0370 = 1.8224$ rad.

We now present an alternative method that is useful if the frequency vector w and the frequency response vector H are available but the mathematical expression of the frequency response is not known. This is the case when the frequency response of a system is measured in several frequencies. We consider the same example as before, namely, the frequency response of the system is

$$H(\Omega) = \frac{3(j\Omega)^2 + 4j\Omega + 2}{-\Omega^2 + j\Omega + 3}$$

and the input signal is given by $x(t) = 3 \cos(t + \pi/3)$, $0 \leq t \leq 30$. Before introducing the alternative method we discuss how to compute which element of a vector has the closest value to a random number. Suppose that the problem is to determine which value of vector $a = [-4, -3, \dots, 4]$ is closer to the value $z = 2.3$.

Commands	Results	Comments
<code>a = -4 : 4 z = 2.3;</code>	<code>a = -4 -3 -2 -1 0 1 2 3 4</code>	
<code>a - z</code>	<code>ans = -6.3 -5.3 -4.3 -3.3 -2.3 -1.3 -0.3 0.7 1.7</code>	
<code>abs(a - z)</code>	<code>ans = 6.3 5.3 4.3 3.3 2.3 1.3 0.3 0.7 1.7</code>	
<code>[m, i] = min(abs(a - z))</code>	<code>m = 0.30 i = 7.00</code>	The command <code>min</code> returns the smallest element of the vector $ a - 2.3 $ (at m) and more importantly the <code>index</code> of the element with the minimum value (at i).
<code>a(i)</code>	<code>ans = 2.00</code>	The element of vector a with index i is the one that is closest to the value $z = 2.3$.

Now, the alternative computational procedure can be described as follows.

Commands	Results	Comments
<code>num = [3 4 2]; den = [1 1 3]; [H, w] = freqs(num, den);</code>		Computation of the vectors of frequency response H and of frequency w .
<code>w0 = 1; [m, i] = min(abs(w - w0));</code>		We estimate the index of the element of w that has the closest value to $\Omega_0 = 1$. Recall that Ω_0 is the frequency of the input signal.

(continued)

Commands	Results	Comments
w(i)	ans = 0.9895	The closest value is 0.9895.
Hw0 = H(i) mag = abs(Hw0) phas = angle(Hw0)	Hw0 = 0.3994 + 1.7629i mag = 1.8076 phas = 1.3480	We evaluate $H(j\Omega_0)$, i.e., we evaluate the frequency response $H(j\Omega)$ for $\Omega = \Omega_0$. Notice that $H(i)$ is the value of $H(j\Omega)$ that corresponds to the frequency $w(i)$.
t = 0:.1:30; y1 = 3*mag*cos(w(i)*t+pi/4+phas) plot(t,y1) legend('Output y(t)') ylim([-6 8]);		The output signal is computed according to Equation 8.8. The derived graph is (almost) the same as the one obtained before; hence, the two methods are (almost) equivalent.

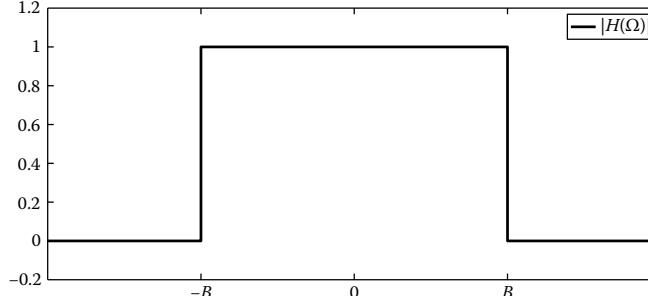
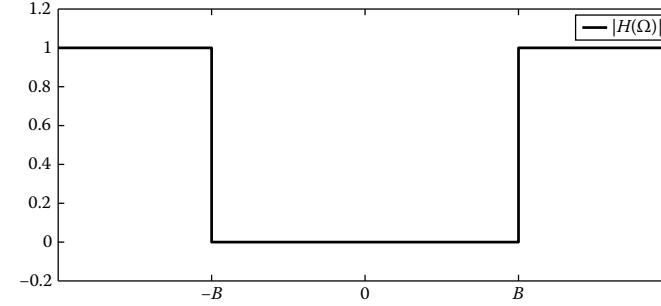
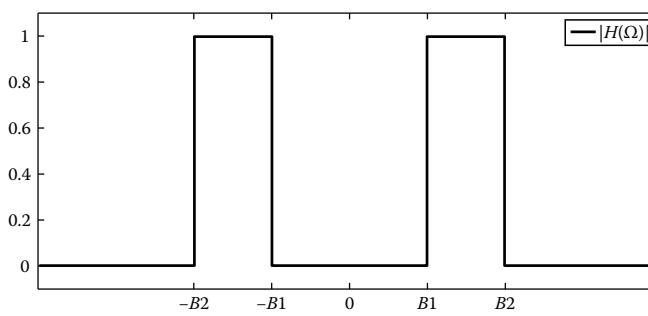
8.5 Ideal Filters

In relationship (8.5) it is stated that $|Y(\Omega)| = |H(\Omega)||X(\Omega)|$, i.e., the magnitude of the output signal in the frequency domain is the product of the magnitude of the spectrum of the input signal and the magnitude of the frequency response. Suppose that $|H(\Omega)|$ is zero over a frequency band (or interval) $B_1 \leq \Omega \leq B_2$. The magnitude of the output signal $|Y(\Omega)|$ will be also zero in the frequency band $B_1 \leq \Omega \leq B_2$. From this point of view, the system described by the frequency response $H(\Omega)$ is a *filter* that blocks (or cuts) the input signal over a specific frequency interval. Depending on the frequency zone that is blocked, there are four basic types of filters.

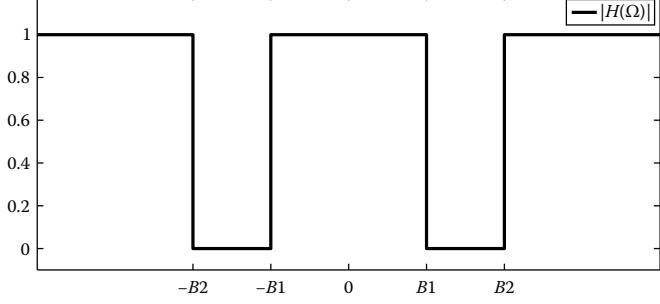
- *Low-pass filters*. The input signal passes at the low frequencies and is cut at the high frequencies.
- *High-pass filters*. The input signal passes at the high frequencies and is cut at the low frequencies.
- *Band-pass filters*. The signal passes over a frequency band and is cut at all other frequencies outside the band.
- *Band-stop filters*. The signal is cut over a frequency band while it passes at all other frequencies.

The set of frequencies Ω such that $|H(\Omega)| > 0$ is called the passband of the filter, while the set of frequencies Ω such that $|H(\Omega)| = 0$ is called the stopband of the filter. The input signal passes undistorted from the passband if the magnitude of the frequency response

is unity in the passband. Next, we illustrate the amplitude responses of the four basic types of filters.

Filter Type/Mathematical Expression/Filter Graph	Commands
<p>Low-pass filter</p> $ H(\Omega) = \begin{cases} 1, & -B \leq \Omega \leq B \\ 0, & \Omega > B \end{cases}$ 	<pre>B = 10; w1 = -20:-B; Hm1 = zeros(size(w1)); w2 = -B:B; Hm2 = ones(size(w2)); w3 = B:20; Hm3 = zeros(size(w3)); w = [w1 w2 w3]; Hm = [Hm1 Hm2 Hm3]; plot(w,Hm) legend(' H(\Omega) '); ylim([- .2 1.2])</pre>
<p>High-pass filter</p> $ H(\Omega) = \begin{cases} 0, & -B \leq \Omega \leq B \\ 1, & \Omega > B \end{cases}$ 	<pre>w1 = -20:-B; Hm1 = ones(size(w1)); w2 = -B:B; Hm2 = zeros(size(w2)); w3 = B:20; Hm3 = ones(size(w3)); w = [w1 w2 w3]; Hm = [Hm1 Hm2 Hm3]; plot(w,Hm) legend(' H(\Omega) '); ylim([- .2 1.2])</pre>
<p>Band-pass filter</p> $ H(\Omega) = \begin{cases} 1, & B_1 \leq \Omega \leq B_2 \\ 0, & \text{elsewhere} \end{cases} = \begin{cases} 1, & -B_2 \leq \Omega \leq -B_1 \\ 1, & B_1 \leq \Omega \leq B_2 \\ 0, & \text{elsewhere} \end{cases}$ 	<pre>B1 = 5; B2 = 10; w1 = -20:-B2; Hm1 = zeros(size(w1)); w2 = -B2:-B1; Hm2 = ones(size(w2)); w3 = -B1:B1; Hm3 = zeros(size(w3)); w4 = B1:B2; Hm4 = ones(size(w4)); w5 = B2:20; Hm5 = zeros(size(w5)); w = [w1 w2 w3 w4 w5]; Hm = [Hm1 Hm2 Hm3 Hm4 Hm5]; plot(w,Hm) legend(' H(\Omega) ');</pre>

(continued)

Filter Type/Mathematical Expression/Filter Graph	Commands
Band-stop filter $ H(\Omega) = \begin{cases} 0, & B_1 \leq \Omega \leq B_2 \\ 1, & \text{elsewhere} \end{cases} = \begin{cases} 0, & -B_2 \leq \Omega \leq -B_1 \\ 0, & B_1 \leq \Omega \leq B_2 \\ 1, & \text{elsewhere} \end{cases}$ 	<pre> B1 = 5; B2 = 10; w1 = -20:-B2; Hm1 = ones(size(w1)); w2 = -B2:-B1; Hm2 = zeros(size(w2)); w3 = -B1:B1; Hm3 = ones(size(w3)); w4 = B1:B2; Hm4 = zeros(size(w4)); w5 = B2:20; Hm5 = ones(size(w5)); w = [w1 w2 w3 w4 w5]; Hm = [Hm1 Hm2 Hm3 Hm4 Hm5]; plot(w,Hm); legend(' H(\Omega) '); </pre>

Up to this point we discussed the magnitude of the frequency response and we concluded that a signal passes undistorted from a specific range of frequencies if the magnitude response of the filter is unity in this range. However, the phase of the frequency response of a filter plays a key role in the possible distortion of a signal passing from that filter. In order to avoid the signal distortion, the phase of the frequency response must be a linear function of Ω , or more specifically the phase response of the filter at the passband must be of the form

$$\angle H(\Omega) = -\Omega t_d, \quad \Omega \in \text{passband} \quad (8.10)$$

where $t_d > 0$ is a constant. Filters that have unity amplitude response in the passband and satisfy Equation 8.10 are called *ideal* filters. The frequency response of an ideal filter is given by

$$H(\Omega) = \begin{cases} e^{-j\Omega t_d}, & \Omega \in \text{passband} \\ 0, & \Omega \in \text{stopband} \end{cases}. \quad (8.11)$$

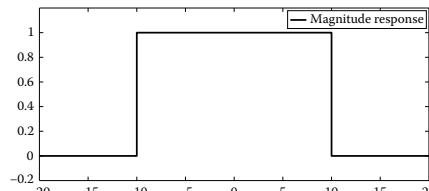
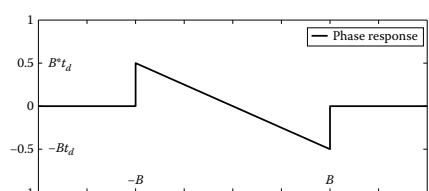
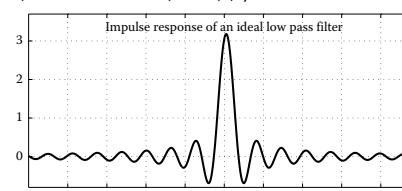
Example

Plot the frequency response of an ideal low-pass filter. Also, compute and plot the impulse response of this filter.

The frequency response of an ideal low-pass filter is given by

$$H(\Omega) = \begin{cases} e^{-j\Omega t_d}, & -B \leq \Omega \leq B \\ 0, & \Omega < -B \text{ and } \Omega > B' \end{cases}$$

where B is sometimes called the cutoff frequency. The MATLAB implementation is as follows.

Commands	Results	Comments
<pre>B = 10; td = .05; w1 = -20:-B; H1=zeros(size(w1)); w2 = -B:B; H2 = exp(-j*w2*td); w3=B:20; H3=zeros(size(w3)); w = [w1 w2 w3]; H = [H1 H2 H3]; plot(w,abs(H)); ylim([-2.2 1.2]); legend('Magnitude response'); plot(w,angle(H)); ylim([-2*B*td 2*B*td]); gtext('Bt_d') gtext('-Bt_d') gtext('B') gtext('-B') legend ('Phase response'); syms t w H=exp(-j*w*td)*(heaviside(w+B)-heaviside(w-B)); h=ifourier(H,t); ezplot(h, [-5 5]); ylim([-0.8 3.5]); title('Impulse response of an ideal low pass filter') grid;</pre>	 	Definition of the frequency response of an ideal low-pass filter and graph of its magnitude response.
		Indeed, the magnitude of the frequency response in the passband $-B \leq \Omega \leq B$, $B = 10 \text{ rad/s}$ is 1 and the slope of the phase is $-td$.
		The impulse response $h(t)$ of the filter is the inverse Fourier transform of the frequency response $H(\Omega)$. The impulse response of an ideal low-pass filter is a $\text{sinc}(\cdot)$ function centered at $t = t_d$.

Example

Compute the response $y(t)$ of an ideal low-pass filter with bandwidth $B = 10 \text{ rad/s}$ to the input signal $x(t) = 3 \cos(4t + \pi/3) + 5 \sin(15t + \pi/4)$.

The frequency of the term $5 \sin(15t + \pi/4)$ is $\Omega_0 = 15 \text{ rad/s}$; thus this term will be cut by the low-pass filter with bandwidth $B = 10 \text{ rad/s}$. On the other hand, the term $3 \cos(4t + \pi/3)$ is expected to pass undistorted since its frequency is $\Omega_0 = 4 \text{ rad/s}$. Therefore, the expected output signal is $y(t) = 3 \cos(4t + (\pi/3) + \theta)$. We present two methods for the computation of the output signal.

First method: The first approach is to consider the relationship that defines the response of a system to a sinusoidal signal. Therefore, combining Equations 8.8 and 8.9, the response $y(t)$ to the sinusoidal input signal $x(t)$ is given by

$$y(t) = 3|H(4)| \cos\left(4t + \frac{\pi}{3} + \angle H(4)\right) + 5|H(15)| \sin\left(15t + \frac{\pi}{4} + \angle H(15)\right).$$

The MATLAB implementation is as follows.

Commands	Results/Comments
<pre>B = 10; td = .05; syms w t H = exp(-j*w*td) * (heaviside(w+B) - heaviside(w-B)); w = 4; Hw0_1 = eval(H) w = 15; Hw0_2 = eval(H) mag1 = abs(Hw0_1) phas1 = angle(Hw0_1) mag2 = abs(Hw0_2) phas2 = angle(Hw0_2) y = 3*mag1*cos(4*t+pi/3+phas1)+5*mag2*sin(15*t+pi/4+phas2)</pre>	<p>We define the frequency response $H(\Omega)$ of the ideal low-pass filter with bandwidth $B = 10$ rad/s.</p> <p>$H(\Omega)$ is computed at the frequencies $\Omega_0 = 4$ and $\Omega_0 = 15$, and we get</p> <p>$Hw0_1 = 0.9801 - 0.1987i$ $Hw0_2 = 0$</p> <p>Computation of $H(4)$, $\angle H(4)$, $H(15)$, and $\angle H(15)$.</p> <p>$mag1 = 1$ $phas1 = -0.2000$ $mag2 = 0$ $phas2 = 0$</p> <p>We compute the output signal according to the relationship of the response of a system to a sinusoidal signal, and we obtain the expected result:</p> <p>$y = 3\cos(4t + 1/3\pi) - 1/5$</p>

The term $5 \sin(15t + \pi/4)$ is completely eliminated by the ideal low-pass filter, and the output signal $y(t)$ is almost the same as the part of the input signal with frequency $\Omega_0 < B$; that is, $y(t)$ is almost the same as $3 \cos(4t + \pi/3)$. The difference is a phase change θ in $y(t)$ which is computed as $\theta = -\Omega_0 t_d = -4*0.05 = -1/5$ rad. This change of phase expresses a *time delay*. More specifically, the relationship that describes the response (in the passband) of an ideal filter to a sinusoidal input signal $x(t) = A \cos(\Omega_0 t)$ is

$$y(t) = A|H(\Omega_0)| \cos(\Omega_0(t - t_d)). \quad (8.12)$$

Second method: The second approach is based on Equation 8.2, namely, is based on the relationship $Y(\Omega) = H(\Omega)X(\Omega)$. Hence, the procedure is to compute the Fourier transform $X(\Omega)$ of the input signal $x(t)$, multiply it with the frequency response $H(\Omega)$ to compute the spectrum $Y(\Omega)$ of the output signal, and then apply inverse Fourier transform to derive the response $y(t)$ of the system in the time domain.

Commands	Results	Comments
<pre>B = 10; td = 0.05; syms t w H = exp(-j*w*td) * (heaviside(w+B) - heaviside(w-B));</pre>		<p>We define the frequency response $H(\Omega)$ of the ideal low-pass filter with bandwidth $B = 10$ rad/s.</p>

(continued)

(continued)

Commands	Results	Comments
<pre>x = 3*cos(4*t+pi/3)+5*sin(15*t+pi/4); X=fourier(x,w);</pre>		We define the input signal $x(t)$ and compute its Fourier transform $X(\Omega)$.
<pre>Y=H*X; Y=simplify(Y);</pre>	$y = \frac{3}{4}i\exp(4it) * (-1)^{(1/5)\pi} * 3^{(1/2)} + \frac{3}{4}\exp(4it) * (-1)^{(-1/5)\pi} + 3^{(1/2)}\exp(-4it) * (-1)^{(1/5)\pi} - \frac{3}{4}i\exp(-4it) * (-1)^{(1/5)\pi} * 3^{(1/2)}$	The spectrum of the output signal $Y(\Omega)$ is computed according to the relationship $Y(\Omega) = X(\Omega)H(\Omega)$.
<pre>y=ifourier(Y,t) ezplot(y, [0 10]) t=0:.1:10 ya=3*cos(4*t+1/3*pi-1/5); hold on plot(t,ya,:o') legend('2nd way','1 st way') ylim([-3.5 5]); hold off</pre>		The output signal $y(t)$ is derived by applying inverse Fourier transform to $Y(\Omega)$. The derived result does not seem same as the previously obtained output signal $y(t) = 3 \cos(4t + \pi/3 + 1/5)$. However, plotting in the same figure the derived output signal together with the one obtained from the first method, we see that the two signals are the same; hence, the two methods are equivalent.

8.6 Frequency Response of Discrete-Time Systems

In this section, we introduce the frequency response of a discrete-time system. Suppose that a discrete-time system is described by an impulse response $h[n]$. Then, the frequency response $H(\omega)$ is defined as the discrete-time Fourier transform (DTFT) of the impulse response $h[n]$ of the system. The mathematical expression is

$$H(\omega) = DTFT\{h[n]\} = \sum_{n=-\infty}^{\infty} h[n]e^{-j\omega n}. \quad (8.13)$$

Example

Compute and plot in the frequency intervals $0 \leq \omega \leq 2\pi$ and $-5\pi \leq \omega \leq 5\pi$ rad/s the frequency response $H(\omega)$ of the discrete-time system described by the impulse response $h[n] = [3, 5, 2, 1], 0 \leq n \leq 3$.

Commands	Results	Comments
<pre>n = 0 : 3; h = [3 5 2 1]; syms w H = sum(h.*exp(-j*w*n))</pre>	$H = 3 + 5 \exp(-i\omega) + 2 \exp(-2i\omega) + \exp(-3i\omega)$	We define the impulse response signal $h[n]$ and the DTFT of $h[n]$ is the frequency response $H(\omega)$ of the system.
<pre>ezplot(abs(H), [0 2*pi]) title(' H(\omega) , 0<\omega<2\pi')</pre>		The magnitude response of the system is plotted in the frequency interval $0 \leq \omega \leq 2\pi$ rad/s.
<pre>ezplot(abs(H), [-5*pi 5*pi]) title(' H(\omega) ,-5\pi<\omega<5\pi')</pre>		The magnitude response of the system is plotted in the frequency interval $-5\pi \leq \omega \leq 5\pi$ rad/s.
<pre>w1 = 0 : .1 : 2*pi; HH = subs(H, w, w1); plot(w1, angle(HH)); title('\angle H(\omega), 0<\omega<2\pi')</pre>		Phase response for $0 \leq \omega \leq 2\pi$ rad/s.
<pre>w1 = -5*pi : .1 : 5*pi; HH = subs(H, w, w1); plot(w1, angle(HH)); xlim([-5*pi 5*pi]) title('\angle H(\omega), -5\pi<\omega<5\pi')</pre>		Phase response for $-5\pi \leq \omega \leq 5\pi$ rad/s.

As expected due to the DTFT, the frequency response of a discrete-time system is periodic with period $T = 2\pi$.

Example

Compute the frequency response $H(\omega)$ of a system with impulse response $h[n] = (2/3)^n u[n]$.

Commands	Results	Comments
<pre>syms n w h = (2/3)^n; H = symsum(h*exp(-j*w*n), n, 0, inf)</pre>	$H = \frac{3 \exp(i\omega)}{-2 + 3 \exp(i\omega)}$	In this case, the impulse response $h[n]$ has infinite terms; hence, its DTFT which is the frequency response $H(\omega)$ is computed with use of the command <code>symsum</code> .

8.7 The Command freqz

The frequency response of a discrete-time system is usually a function of the form

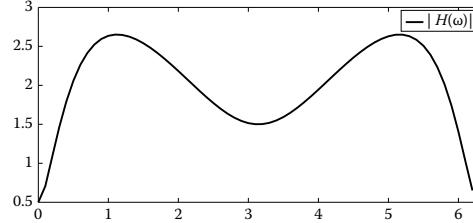
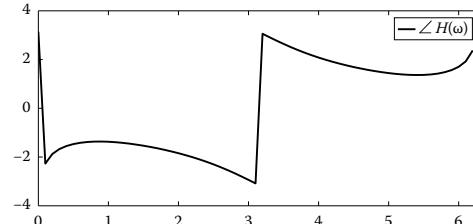
$$H(\omega) = \frac{B(\omega)}{A(\omega)} = \frac{b_0 + b_1 e^{-j\omega} + \dots + b_n e^{-jn\omega}}{a_0 + a_1 e^{-j\omega} + \dots + a_m e^{-jm\omega}}. \quad (8.14)$$

In order to compute the values of frequency response $H(\omega)$ over a frequency interval ω , we can use the command `freqz`. Its syntax and use is similar to the command `freqs` which was employed to compute the frequency response of a continuous-time system. A common syntax is `H = freqz (num, den, w)`, where w is the vector of frequencies, `num` and `den` are the coefficients of the numerator and denominator polynomials of Equation 8.14, respectively, and H is the vector of frequency response. The syntax `[H, w] = freqz (num, den, N)` evaluates N points of $H(\omega)$ for $0 \leq \omega \leq \pi$ rad/s, which corresponds to the upper part of the unit circle. This part will become clear when we discuss the concept of transfer function in Chapter 11. If N is omitted, its default value is 512. In order to compute the frequency response over the whole unit circle ($0 \leq \omega \leq 2\pi$), the appropriate syntax is `[H, w] = freqz (num, den, N, 'whole')`. Finally, typing just `freqz (num, den, N)` or `freqz (num, den, w)` returns the graph of the magnitude (in dB) and the phase (in degrees) of the frequency response versus the normalized by π frequency ω .

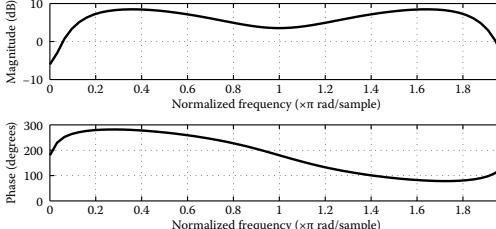
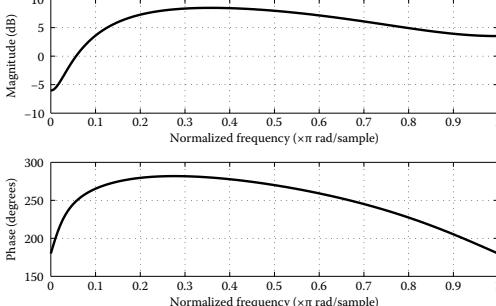
Example

Plot the discrete-time frequency response

$$H(\omega) = \frac{3 + 5e^{-j\omega} - 7e^{-2j\omega}}{2 - 4e^{-j\omega}}.$$

Commands	Results/Comments
<pre>w = 0 : .1 : 2*pi; num = [3 5 -7]; den = [2 -4]; H = freqz (num, den, w); plot (w, abs (H)) legend (' H(\omega) ') xlim ([0 2*pi])</pre>	<p>The frequency response is computed for $0 \leq \omega \leq 2\pi$ and its magnitude is plotted.</p> 
<pre>plot (w, angle (H)) xlim ([0 2*pi]) legend ('\angle H(\omega)')</pre>	<p>Phase response for $0 \leq \omega \leq 2\pi$ rad/s.</p> 

(continued)

Commands	Results/Comments
	Magnitude and phase response of $H(\omega)$ in the normalized by π frequency interval w .
<code>freqz(num, den, w);</code>	 <p>Magnitude and phase response of $H(\omega)$ in the normalized by π upper part ($0 \leq \omega \leq \pi$) of the unit circle.</p> 

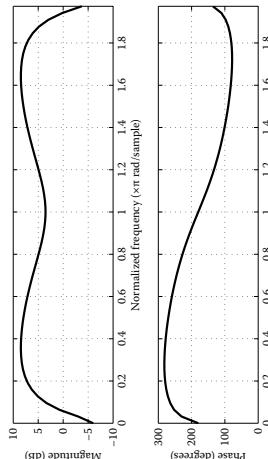
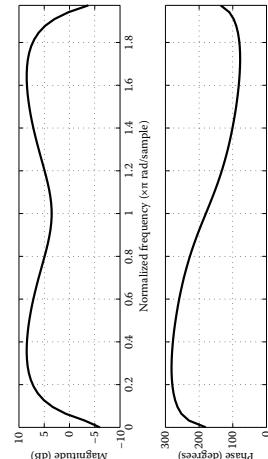
8.7.1 The Command `invfreqz`

The command `invfreqz` is used for the inverse operation from `freqz`. More specifically, the frequency response H evaluated at the frequencies specified in w , and the corresponding vector of the frequencies w are now known (i.e., are the input arguments), and the numerator and denominator of the frequency response are computed through the command `invfreqs`. The syntax is `[num, den] = invfreqz(H, w, N, M)`, where H is the frequency response vector, w is the vector of corresponding frequencies, and N and M specify the number of terms of $e^{-j\omega}$ of the numerator and denominator of the discrete-time frequency response $H(\omega)$, respectively. The output arguments `num` and `den` are the computed coefficients of the frequency response polynomials $B(\omega)$ and $A(\omega)$. Moreover, the `invfreqs` command can be used to specify equivalent frequency responses given the evaluated (at the frequencies of the vector w) frequency response vector H .

Example

Suppose that the vectors H and w , which were obtained in the previous example, are available. Derive the mathematical expression of the discrete-time frequency response $H(\omega)$ and estimate an equivalent frequency response $H_2(\omega)$ with four terms of $e^{-j\omega}$ in the numerator and denominator.

In this example, we will not define again vectors H and w , as this is already done in the previous example.

Commands	Results	Comments
<pre>[num1, den1] = invfreqz (H, w, 2, 1)</pre>	<pre>num1 = 1.5000 2.5000 -3.5000 den1 = 1.0000 -2.0000</pre>	The mathematical expression of $H(\omega)$ is computed with use of the command <code>invfreqz</code> , and is same (if we multiply <code>num1</code> and <code>den1</code> by 2) to the frequency response of the previous example $H(\omega) = \frac{3 + 5e^{-j\omega} - 7e^{-2j\omega}}{2 - 4e^{-j\omega}}.$
<pre>freqz (num1, den1, w)</pre>		Graph of the estimated frequency response $H(\omega)$.
<pre>[num2, den2] = invfreqz (H, w, 4, 4)</pre>	<pre>num2 = 1.5000 0.9638 -7.1480 1.7715 2.5381 den2 = 1.0000 -3.0241 1.3231 1.4503 0.0000</pre>	Equivalent frequency response with four terms of $e^{-j\omega}$ in numerator and denominator. $H(\omega) = \frac{1.5 + 0.9638e^{-j\omega} - 7.148e^{-2j\omega} + 1.7715e^{-3j\omega} + 2.5381e^{-3j\omega}}{1 - 3.0241e^{-j\omega} + 1.3231e^{-2j\omega} + 1.4503e^{-3j\omega}}.$
<pre>freqz (num2, den2, w)</pre>		Graph of the estimated equivalent frequency response $H_2(\omega)$, which is indeed same as $H(\omega)$.

8.8 System Response to Discrete-Time Sinusoidal Input

The case where a discrete-time sinusoidal signal is applied to a discrete-time system is treated in a similar way to the one introduced in the continuous-time case. Thus, suppose that a sinusoidal sequence $x[n] = A \cos(\omega_0 n + \theta)$ is the input signal applied to a discrete-time system with frequency response $H(\omega)$. Then, the response $y[n]$ of the system to $x[n]$ is computed according to the relationship

$$y[n] = A|H(\omega_0)| \cos(\omega_0 n + \theta + \angle H(\omega_0)). \quad (8.15)$$

Example

Compute the response of the discrete-time system with frequency response $H(\omega) = (3 + e^{-j\omega}) / (2 + 4e^{-j\omega})$ to the input signal $x[n] = 2 \cos(4n + \pi/3)$. Also, plot for $-10 \leq n \leq 10$ in the same figure the output sequence $y[n]$ with the input sequence $x[n]$.

Commands	Results	Comments
<pre>w0 = 4; Hw0 = (3+exp(-j*w0)) / (2+4*exp(-j*w0)); mag = abs(Hw0) phas = angle(Hw0)</pre>	<pre>mag = 0.7981 phas = -1.4591</pre>	We compute the quantities $ H(\omega_0) $ and $\angle H(\omega_0)$ for $\omega_0 = 4$, which is the frequency of the input signal.
<pre>syms n y=2*mag*cos(w0*n+pi/3+phas)</pre>	<pre>y = 1.5963*cos(4*n+1/3*pi - 1.4591)</pre>	The system response $y[n]$ is computed according to Equation 8.15.

The two signals have the same frequency ω_0 , but the amplitude of the output is smaller than the amplitude of the input by a factor of $|H(\omega_0)| = 0.7981$, and moreover there is a phase difference of $\angle H(\omega_0) = -1.4591$.

8.9 Moving Average Filter

In this section, we introduce a special type of filter called the moving average filter. A moving average filter is a simple FIR filter that is described by the input/output relationship

$$y[n] = \frac{1}{N} (x[n] + x[n-1] + \dots + x[n-N+1]) = \frac{1}{N} \sum_{k=0}^{N-1} x[n-k]. \quad (8.16)$$

The time-shifting property of DTFT states that if k is an integer number then

$$\text{DTFT}\{x[n - k]\} = X(\omega)e^{-jk\omega}. \quad (8.17)$$

Applying DTFT to (8.16) due to Equation 8.17 yields

$$\begin{aligned} Y(\omega) &= \frac{1}{N}(X(\omega) + X(\omega)e^{-j\omega} + \dots + X(\omega)e^{-j(N-1)\omega}) \\ \Rightarrow Y(\omega) &= \frac{1}{N}(1 + e^{-j\omega} + \dots + e^{-j(N-1)\omega})X(\omega). \end{aligned} \quad (8.18)$$

Hence, the frequency response of a moving average filter is given by

$$H(\omega) = \frac{Y(\omega)}{X(\omega)} = \frac{1}{N} \sum_{k=0}^{N-1} e^{-jk\omega}. \quad (8.19)$$

Equation 8.19 is equivalently written as

$$H(\omega) = \frac{\sin(N\omega/2)}{N \sin(\omega/2)} e^{-j(N-1)\omega/2}. \quad (8.20)$$

Finally, the impulse response of a moving average filter is given by

$$h[n] = \begin{cases} \frac{1}{N}, & 0 \leq n \leq N-1 \\ 0, & \text{elsewhere} \end{cases}. \quad (8.21)$$

Example

Compute and plot the frequency response $H(\omega)$ of a $N = 3$ point moving average filter.

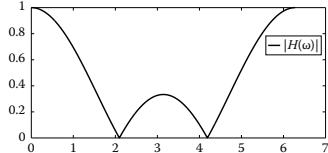
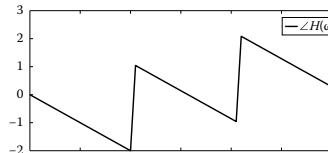
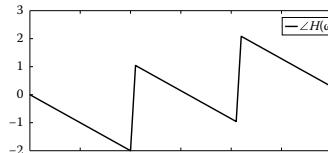
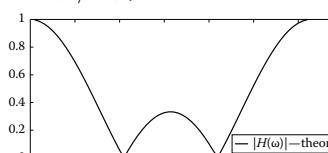
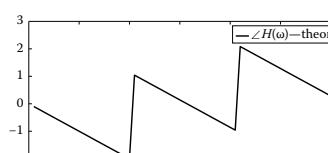
According to (8.16) and (8.20), the input/output relationship of the moving average filter is $y[n] = 1/3(x[n] + x[n - 1] + x[n - 2])$, and the frequency response of the filter is given by

$$H(\omega) = \frac{\sin(3\omega/2)}{3 \sin(\omega/2)} e^{-j\omega}.$$

We will use this $H(\omega)$ only to confirm our solution, which is based on the time-shifting property of DTFT given in Equation 8.17.

Commands	Results	Comments
<pre>syms X w X1=X*exp(-j*w); X2=X*exp(-j*2*w); Y=(1/3)*(X+X1+X2);</pre>		We define the DTFTs of $x[n - 1]$ and $x[n - 2]$ according to Equation 8.17.
		We apply DTFT in both sides of the input/output relationship of the moving average filter.

(continued)

Commands	Results	Comments
$H = Y/X;$ $H = \text{simplify}(H)$	$H = 1/3 + 1/3 * \exp(-i*w) + 1/3 * \exp(-2*i*w)$ 	The frequency response of the filter is computed from the relationship $H(\omega) = Y(\omega)/X(\omega)$.
$\text{ezplot}(\text{abs}(H), [0 2*\pi]);$ $\text{legend}(' \text{H}(\omega) ');$		Magnitude response for $0 \leq \omega \leq 2\pi$ rad/s.
$w1 = 0 : .1 : 2*\pi;$ $HH = \text{subs}(H, w, w1);$ $\text{plot}(w1, \text{angle}(HH));$ $\text{xlim}([0 2*\pi]);$ $\text{legend}(' \text{angle H}(\omega) ');$		Phase response for $0 \leq \omega \leq 2\pi$ rad/s.
$w = 0 : .1 : 2*\pi;$ $Htheor = \sin(3*w/2) .* \exp(-j*w) ./ (3 * \sin(w/2));$ $\text{plot}(w, \text{abs}(Htheor));$ $\text{xlim}([0 2*\pi]);$ $\text{ylim}([-2 1.2]);$ $\text{legend}(' \text{H}(\omega) - \text{theory}');$	 	The magnitude and the phase of the frequency response are plotted according to the theoretical relationship (8.20).
$\text{plot}(w, \text{angle}(Htheor));$ $\text{xlim}([0 2*\pi]);$ $\text{ylim}([-2.3 2.3]);$ $\text{legend}(' \text{angle H}(\omega) - \text{theory}')$		

The corresponding graphs are identical; hence, the frequency response computation is correct.

8.10 Solved Problems

Problem 1 Compute and plot in the frequency interval $-20 \leq \Omega \leq 20$ rad/s the frequency response of a continuous-time system with impulse response

- a. $h[t] = p2(t)$
- b. $h[t] = p10(t)$

where $pT(t)$ is a rectangular pulse of duration $t = T$.

Solution

A rectangular pulse is defined as $pT(t) = u(t + T/2) - u(t - T/2)$.

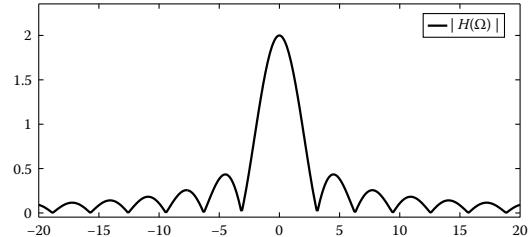
The impulse response $p2(t)$ is defined as

$p2(t) = u(t + 1) - u(t - 1)$ and the Fourier transform

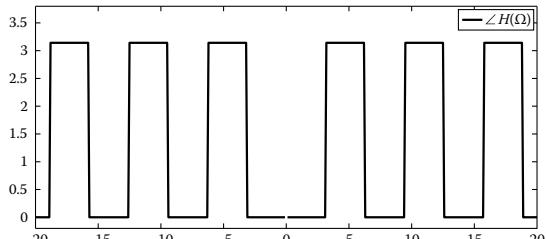
of $p2(t)$ is the frequency response $H(\Omega)$ of the system.

$$H = 2/w \sin(w)$$

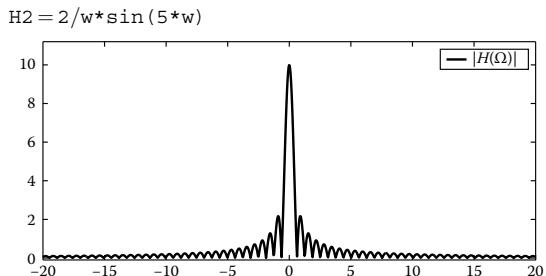
```
a. syms t w
h=heaviside(t+1)-heaviside(t-1)
H=fourier(h,w)
ezplot(abs(H), [-20 20]);
legend('| H(\Omega) |')
```



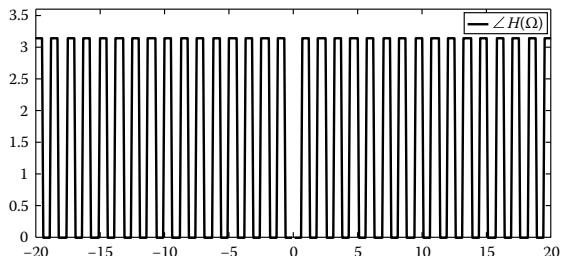
```
w1=-20:.1:20;
HH=subs(H,w,w1);
plot(w1,angle(HH));
ylim([-2 3.8]);
legend('\angle H(\Omega)')
```



```
b. h2=heaviside(t+5)-heaviside(t-5);
H2=fourier(h2,w)
ezplot(abs(H2), [-20 20]);
ylim([-1 11.2]);
legend('| H(\Omega) | ')
```



```
w1=-20:.1:20;
HH=subs(H2,w,w1);
plot(w1,angle(HH));
legend('\angle H(\Omega)')
```



Problem 2 Plot the magnitude and the phase of the frequency response

$$H(j\Omega) = \frac{8\Omega^2 + 2j\Omega + 9}{6j\Omega^2 - 10}$$

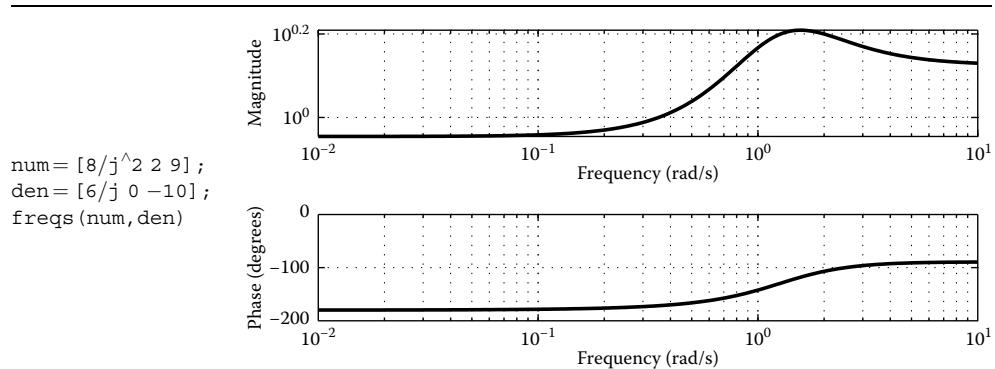
in the frequency range $0 \leq \Omega \leq 10$ rad/s.

Solution

The frequency response $H(j\Omega)$ is written as

$$H(j\Omega) = \frac{8\Omega^2 + 2j\Omega + 9}{6j\Omega^2 - 10} = \frac{(8/j^2)(j\Omega)^2 + 2j\Omega + 9}{(6/j^2)(j\Omega)^2 + 0j\Omega - 10}.$$

Hence,



Problem 3 Plot the magnitude and the phase of the frequency response

$$H(j\Omega) = \frac{8j\Omega^3 + 2\Omega - 9}{3\Omega^3 + 3j\Omega^2 + \Omega - 10}$$

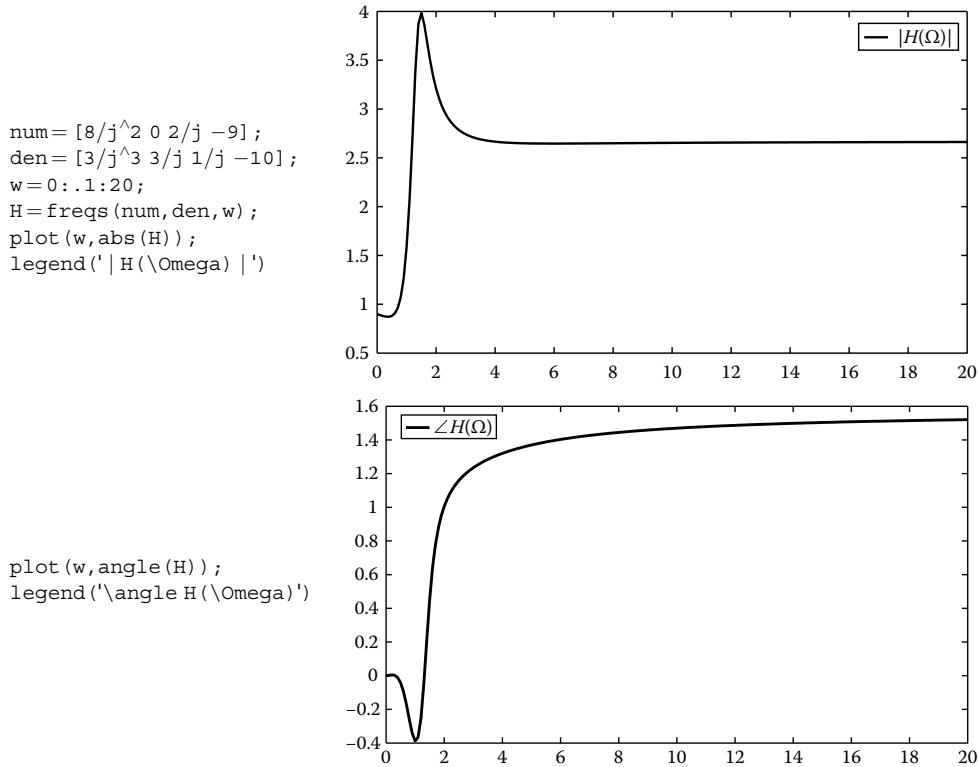
in the frequency range $0 \leq \Omega \leq 20$ rad/s.

Solution

The frequency response $H(j\Omega)$ is written as

$$H(j\Omega) = \frac{8j\Omega^3 + 2\Omega - 9}{3\Omega^3 + 3j\Omega^2 + \Omega - 10} = \frac{(8/j^3)(j\Omega)^3 + 0(j\Omega)^2 + (2/j)j\Omega - 9}{(3/j^3)(j\Omega)^3 + (3/j)(j\Omega)^2 + (1/j)j\Omega - 10}.$$

Thus,



Problem 4 Suppose that a system is described by the frequency response

$$H_1(j\Omega) = \frac{3(j\Omega)^2 - 4(j\Omega) + 1}{(j\Omega)^3 + 2(j\Omega)}.$$

Find equivalent system with frequency response of the form

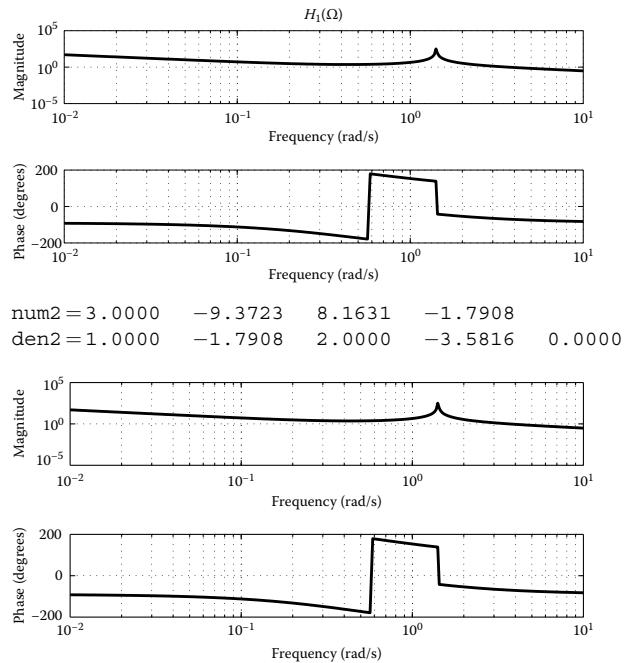
$$H_2(j\Omega) = \frac{b_3(j\Omega)^3 + b_2(j\Omega)^2 + b_1(j\Omega) + b_0}{a_4(j\Omega)^4 + a_3(j\Omega)^3 + a_2(j\Omega)^2 + a_1(j\Omega) + a_0}.$$

Solution

We will use the command `invfreqs`. First, we plot the frequency response $H_1(j\Omega)$.

```
num1 = [3 -4 1];
den1 = [1 0 2 0];
[H1,w] = freqs(num1,den1);
freqs(num1,den1)
title('H_1(\Omega)')

[num2,den2] = invfreqs(H1,w,3,4)
```



```
freqs(num2,den2)
```

The two graphs are alike; hence, the frequency response

$$H_2(j\Omega) = \frac{3(j\Omega)^3 - 9.3723(j\Omega)^2 + 8.1631(j\Omega) + 1.7908}{(j\Omega)^4 - 1.7908(j\Omega)^3 + 2(j\Omega)^2 + 3.5816(j\Omega)}$$

is equivalent to $H_1(j\Omega)$.

Problem 5 Compute and plot the response of a system with frequency response

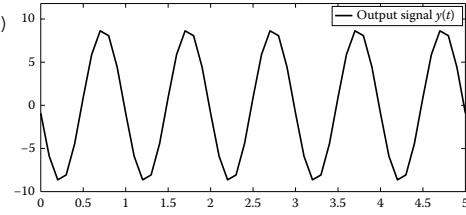
$$H(j\Omega) = \frac{3(j\Omega)^3 + 4(j\Omega) + 2}{(j\Omega)^2 + 2(j\Omega) + 1}$$

to the input signal $x(t) = 3 \cos(2\pi t + \pi/2)$, $0 \leq t \leq 5$.

Solution

We will use the relationship of system response to sinusoidal inputs given in Equation 8.8.

```
w0 = 2*pi;
Hw0 = (3*(j*w0)^2 + 4*j*w0 + 2) / ((j*w0)^2 + 2*j*w0 + 1)
mag = abs(Hw0);
phas = angle(Hw0);
t = 0:.1:5;
y = 3*mag*cos(w0*t+pi/2+phas);
plot(t,y)
legend('Output signal y(t)')
```



Problem 6 Compute and plot the response of a system with frequency response

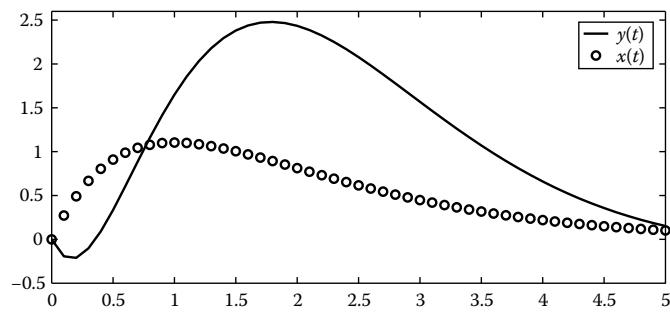
$$H(j\Omega) = \frac{\Omega^2 + 4(j\Omega) + 2}{(j\Omega)^2 + 2(j\Omega) + 1}$$

to the input signal $x(t) = 3te^{-t}$, $0 \leq t \leq 5$. Moreover, plot in the same figure the input signal.

Solution

The system response is computed with the command `lsim`, but we have to be careful in order not to use complex coefficients in vectors `num` and `den`.

```
num = [-1 4 2];
den = [1 2 1];
t = 0:.1:5;
x = 3*t.*exp(-t);
y = lsim(num, den, x, t);
plot(t, y, t, x, 'o')
legend('y(t)', 'x(t)')
```



Problem 7 Plot the frequency response and the impulse response of an ideal bandpass filter with cutoff frequencies $B_1 = 10$ rad/s and $B_2 = 20$ rad/s.

Solution

The frequency response of an ideal bandpass filter is given by

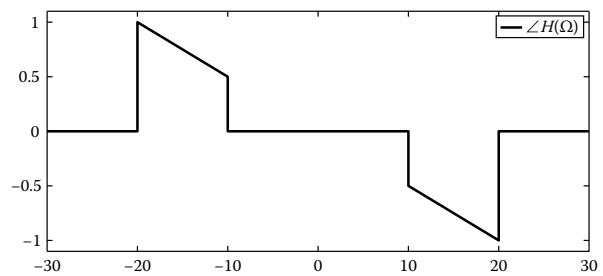
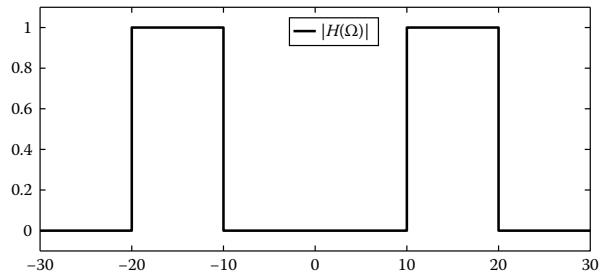
$$H(\Omega) = \begin{cases} e^{-j\Omega t_d}, & B_1 \leq |\Omega| \leq B_2 \\ 0, & \text{elsewhere} \end{cases} = \begin{cases} e^{-j\Omega t_d}, & -B_2 \leq \Omega \leq -B_1 \\ e^{j\Omega t_d}, & B_1 \leq \Omega \leq B_2 \\ 0, & \text{elsewhere} \end{cases}.$$

```

td = .05;
B1 = 10;
B2 = 20;
w1 = -30:-B2;
H1 = zeros(size(w1));
w2 = -B2:-B1;
H2 = exp(-j*w2*td);
w3 = -B1:B1;
H3 = zeros(size(w3));
w4 = B1:B2;
H4 = exp(-j*w4*td);
w5 = B2:30;
H5 = zeros(size(w5));
w = [w1 w2 w3 w4 w5];
H = [H1 H2 H3 H4 H5];
plot(w,abs(H))
ylim([-2 1.2]);
legend('|H(\Omega)|')
plot(w,angle(H));
ylim([-1.1 1.1]);
legend('\angle H(\Omega)')

```

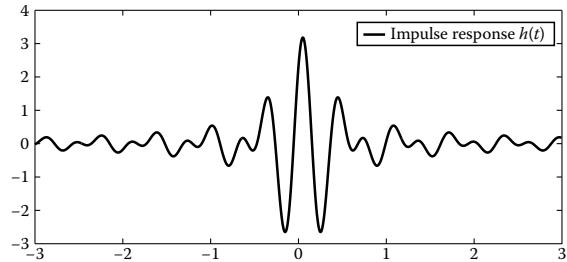
Graph of frequency response.



```

syms t w
H1 = exp(-j*w*td)*(heaviside(w+B2)-heaviside(w+B1));
H2 = exp(-j*w*td)*(heaviside(w-B1)-heaviside(w-B2));
h1 = ifourier(H1,t);
h2 = ifourier(H2,t);
h = h1+h2;
ezplot(h, [-3 3]);
ylim([-4 4]);
legend('Impulse response h(t)')

```



The impulse response of an ideal bandpass filter consists of a sinusoid within a sinc envelope.

Problem 8 Consider an ideal low-pass filter with cutoff frequency $B = 10$ rad/s. Compute the response of the filter to the input signal $x(t) = 2 \sin(30t)/\pi t$.

Solution

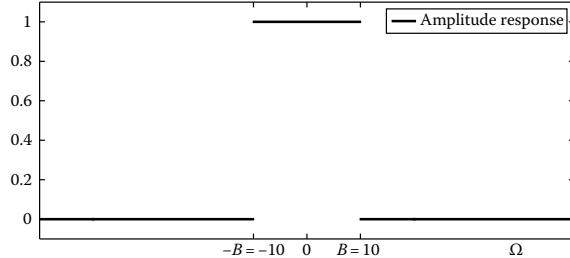
The frequency response of an ideal low-pass filter is given by

$$H(\Omega) = \begin{cases} e^{-j\Omega t_d}, & -B \leq \Omega \leq B \\ 0, & \Omega < -B \text{ and } \Omega > B \end{cases}.$$

The procedure is to first compute the Fourier $X(\Omega)$ of $x(t)$, then compute the output signal in the frequency domain according to the relationship $Y(\Omega) = X(\Omega)H(\Omega)$, and finally apply inverse Fourier transform to $Y(\Omega)$ to derive the output signal $y(t)$ in the time domain.

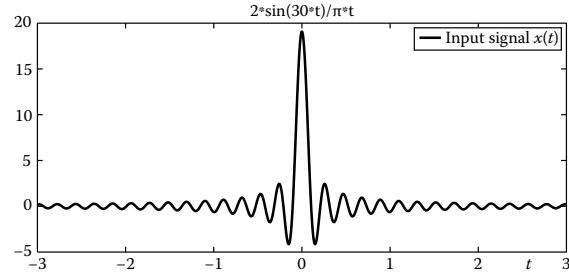
```
syms t w
td=.05;
B= 10;
H=exp(-j*w*td)*(heaviside(w+B)-heaviside(w-B));
w1=-50:.1:50;
H1=subs(H,w,w1);
plot(w1,abs(H1));
legend('Amplitude response')
ylim([- .1 1.1]);
xlabel('\Omega')
```

Definition of the frequency response of the filter and graph of its magnitude for $-50 \leq \Omega \leq 50$ rad/s.



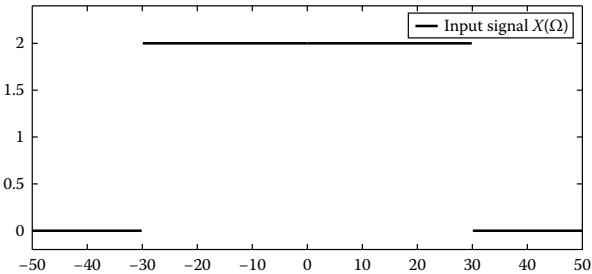
```
x=2*sin(30*t)/(pi*t);
ezplot(x, [-3 3])
ylim([-5 20]);
legend('Input signal x(t)')
```

Graph of input signal $x(t)$, $-3 \leq t \leq 3$ s.



Computation of $X(\Omega)$ and graph of $|X(\Omega)|$, $-50 \leq \Omega \leq 50$ rad/s.

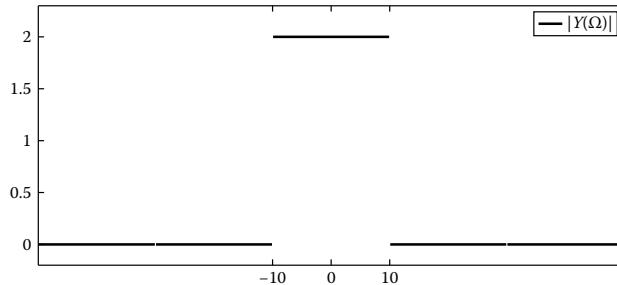
```
X=fourier(x,w);
w1=-50:.1:50;
XX=subs(X,w,w1);
plot(w1,abs(XX));
legend('Input signal X(\Omega)')
ylim([- .2 2.4]);
```



(continued)

We compute $Y(\Omega)$ and plot $|Y(\Omega)|$, $-50 \leq \Omega \leq 50$ rad/s.

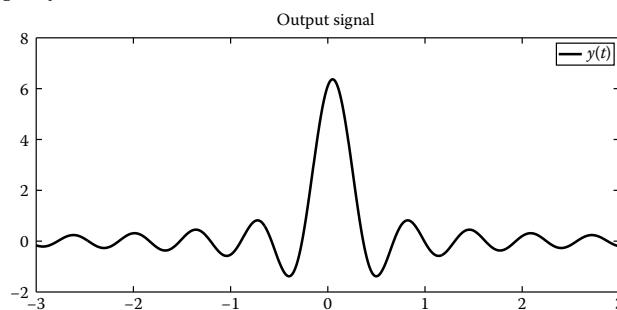
```
Y=X*H;
YY=subs(Y,w,w1);
plot(w1,abs(YY));
ylim([- .2 2.3]);
legend(' | Y(\Omega) |')
```



The input signal had passed undistorted through the filter in the passband $-B \leq \Omega \leq B$ rad/s (i.e., in the frequency range $-10 \leq \Omega \leq 10$ rad/s) and is cut (its magnitude is zero) in the stopband, namely, outside the frequency range $-B \leq \Omega \leq B$ rad/s.

By applying inverse Fourier transform to $Y(\Omega)$, we compute the output signal $y[t]$.

```
y=ifourier(Y);
ezplot(y, [-3 3])
 ylim([-1.5 8]);
title ('Output signal')
legend('y(t) ')
```



The output signal is plotted and we see that it is oscillating with lower frequency compared to the input signal, since the high-frequency components of the signal were cut from the ideal low-pass filter.

Problem 9 Suppose that a sinusoidal signal $x(t) = \cos(t)$ is transmitted over an additive white Gaussian noise channel (or system). With the help of the commands `fft` and `ifft` remove the noise from the input signal $x(t)$.

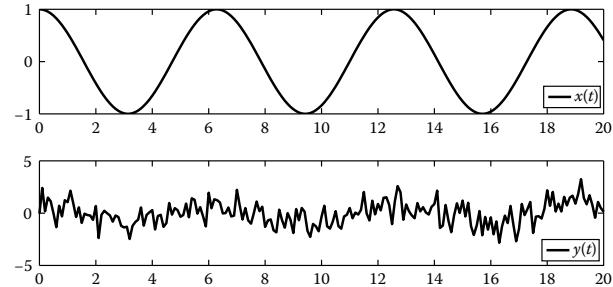
Solution

```
t = 0:1:20;
x = cos(t);
w = randn(size(t));
y = x+w;
```

Definition of $x(t)$ (which is actually a sampling of the continuous-time signal) and of the noisy received signal $y(t)$. Additive white Gaussian noise is modeled as a Gaussian process $w(t)$ with zero mean; thus $y(t)$ is given by $y(t) = x(t) + w(t)$.

We plot the input signal $x(t)$ and the output signal $y(t)$ in order to see the distortion of $y(t)$.

```
subplot(2,1,1);
plot(t,x)
legend('x(t)');
subplot(2,1,2);
plot(t,y)
legend('y(t)');
```



We compute the DFTs X_k and Y_k of $x(t)$ and $y(t)$, respectively.

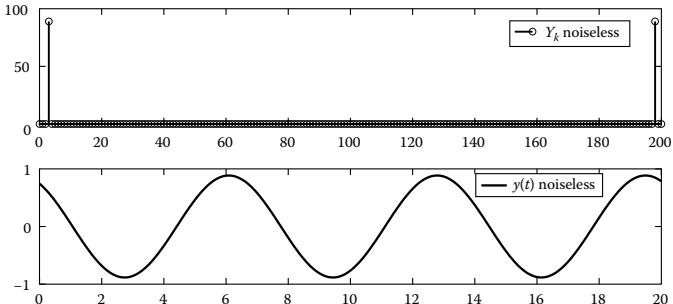
```
X=fft(x);
Y=fft(y);
N=length(X);
subplot(2,1,1);
stem(0:N-1,abs(X));
legend('X_k');
subplot(2,1,2);
stem(0:N-1,abs(Y));
legend('Y_k');

for i=1:length(Y)
if abs(Y(i))<50
Yclean(i)=0;
else
Yclean(i)=Y(i);
end
end
```

The samples of Y_k that are below a threshold are removed. Here the threshold is 50, but generally the threshold is subject to the noise power.

```
Yclean=ifft(Yclean);
subplot(2,1,1);
stem(0:N-1,abs(Yclean));
legend('Y_k noiseless');
subplot(2,1,2);
plot(t,Yclean)
legend('y(t) noiseless');
```

The IDFT of the “clean” Y_k is the filtered signal $y_c[t]$. Indeed, the filtered version of $y[t]$ is very similar to the input signal $x(t)$; hence, the noise removal process is successful.



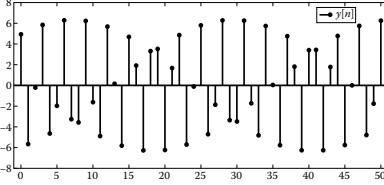
Problem 10 Consider a discrete-time system with frequency response

$$H(\omega) = \frac{3j\omega + 2}{\omega + 1}.$$

Compute the system response $y[n]$ to the input signal $x[n] = 3 \sin(2n + 1)$.

Solution

The system response $y[n]$ in the discrete-time domain is computed according to $y[n] = 3|H(2)| \sin(2n + 1 + \angle H(2))$, where $|H(2)|$ is the magnitude and $\angle H(2)$ is the angle of the frequency response $H(\omega)$ evaluated at the frequency of the input signal $\omega = \omega_0 = 2$ rad/s.

Commands	Results	Comments
<pre> syms n w0 = 2; Hw0 = (3*j*w0+2) / (w0+1); mag = abs(Hw0); phas = angle(Hw0); y = 3*mag*sin(w0*n+1+phas); n = 0:50; y = subs(y, n); stem(n, y); legend('y [n]'); </pre>		Calculation of $ H(\omega_0) $ and $\angle H(\omega_0)$ for $\omega_0 = 2$.

8.11 Homework Problems

- Suppose that a system is described by the impulse response $h(t) = te^{-t}u(t)$. Compute and plot the frequency response of the system.
- Suppose that a system is described by the frequency response $H(j\Omega) = 4/(3 + j\Omega)^3$. Compute and plot the impulse response of the system.
- Suppose that a system is described by the impulse response $h(t) = 2\text{sinc}(t)$. Plot the magnitude of the frequency response of the system.
- Suppose that a system is described by the impulse response $h(t) = 2\text{sinc}(t)\sin(t)$. Plot the magnitude of the frequency response of the system.
- Suppose that a system is described by the frequency response

$$H_1(j\Omega) = \frac{3\Omega^2 - 4j\Omega + 1}{\Omega^3 + 2j\Omega}.$$

Find an equivalent system with frequency response of the form

$$H_2(j\Omega) = \frac{b_3(j\Omega)^3 + b_2(j\Omega)^2 + b_1(j\Omega) + b_0}{a_4(j\Omega)^4 + a_3(j\Omega)^3 + a_2(j\Omega)^2 + a_1(j\Omega) + a_0}.$$

6. Consider a system described by the frequency response

$$H(j\Omega) = \frac{-2\Omega^2 + j\Omega}{-3\Omega^2 + 2j\Omega + 1}.$$

Compute and plot the response of the system to the input signal

$$x(t) = \cos(t)/(t+1), \quad 0 \leq t \leq 100.$$

7. Consider a system described by the frequency response $H(j\Omega) = 3/(j\Omega + 2)$. Compute and plot the response of the system to the input signal $x(t) = te^{-t}u(t)$. Verify your result by computing the convolution between the input signal and the impulse response of the system.
8. A system is described by the frequency response

$$H(j\Omega) = \frac{-2\Omega^2 - j\Omega}{-\Omega^2 + 4j\Omega + 3}.$$

Use the relationship that describes the response of a system to sinusoidal input signals to compute and plot over the time interval $0 \leq t \leq 50$ the response of the system to the input signals

- a. $x_1(t) = 2 \cos(2t + \pi/6)$
- b. $x_2(t) = 4 \sin((\pi/2)t + \pi/3)$
- c. $x_3(t) = x_1(t) + x_2(t)$

Confirm your derived results with the command `lsim`.

9. Plot the impulse response of an ideal bandstop filter that cuts the input signal at the frequency range $20 \leq \Omega \leq 30$ rad/s.
10. Plot the impulse response of an ideal high-pass filter that cuts the input signal for frequencies $\Omega < 30$ rad/s.
11. Consider an ideal high-pass filter with cutoff frequency $B = 20$ rad/s. Compute the response of the filter to the input signal $x(t) = 2 \cos(10t)/(\pi t)$.
12. Compute the frequency response of a discrete-time system when the system is described by the following impulse response:
- a. $h[n] = \begin{cases} n^2, & 0 \leq n \leq 4 \\ 0, & \text{elsewhere} \end{cases}$
 - b. $h[n] = 0.9^n u[n]$

13. Plot the discrete-time frequency responses

$$H_1(\omega) = \frac{3 + 5e^{-j\omega} - 7e^{-2j\omega}}{2 - 4e^{-j\omega}}$$

and

$$H_2(\omega) = \frac{5e^{-j\omega} + 7e^{-2j\omega}}{4e^{-j\omega} + 1}.$$

14. Suppose that a discrete-time system is described by the frequency response

$$H(\omega) = \frac{3e^{-j\omega} - 7e^{-3j\omega}}{3 + 4e^{-j\omega} + 6e^{-2j\omega} - 8e^{-3j\omega}}.$$

Find the frequency response of an equivalent system whose frequency response has five terms of $e^{-j\omega}$ in the numerator and six terms of $e^{-j\omega}$ in the denominator.

15. Compute the response of the discrete-time system with frequency response

$$H(\omega) = \frac{4 + 2e^{-j\omega}}{3 + 5e^{-j\omega}}$$

to the input signal $x[n] = 2 \sin(3n + \pi/3)$. Also, plot for $-10 \leq n \leq 10$ in the same figure the output sequence $y[n]$ with the input sequence $x[n]$.

16. Compute and plot the frequency response $H(\omega)$ of a 4-point moving average filter.
 17. Design a bandpass ideal filter that causes a delay of 0.1s to the input signal $x(t) = \cos(15t)$.
 18. Suppose that the signal $x(t) = \sin(t) + \cos(2\pi t)$ is transmitted over an additive white Gaussian noise channel. With the help of the commands `fft` and `ifft` remove the noise from the received signal $y(t)$.

This page intentionally left blank

9

Laplace Transform

In this chapter, we introduce the Laplace transform of continuous-time signals. The Laplace transform is an operation that “transfers” a continuous-time signal from the time domain t into the complex frequency domain s , and is a valuable tool in the signal processing field.

9.1 Mathematical Definition

A continuous-time signal described in the time domain t is a signal given by a function $f(t)$. As previously mentioned, the Laplace transform expresses a signal in the complex frequency domain s (or s -domain); that is, a signal is described by a function $F(s)$. Laplace transform is denoted by the symbol $L\{.\}$; that is, one can write

$$F(s) = L\{f(t)\}. \quad (9.1)$$

In other words, the Laplace transform of a function $f(t)$ is a function $F(s)$. An alternative way of writing (9.1) is

$$f(t) \xrightarrow{L} F(s). \quad (9.2)$$

There are two available forms of Laplace transform. The first is the two-sided (or bilateral) Laplace transform where the Laplace transform $F(s)$ of a function $f(t)$ is given by

$$F(s) = L\{f(t)\} = \int_{-\infty}^{\infty} f(t)e^{-st}dt. \quad (9.3)$$

Variable s is a complex-valued number, and thus can be written as $s = \sigma + j\Omega$. This relationship reveals the association between the Laplace transform and the Fourier transform. More specifically, substituting s by $j\Omega$ in (9.3) yields the Fourier transform $X(\Omega)$ of $x(t)$. Setting the lower limit of the integral to zero yields the one-sided (or unilateral) Laplace transform, which is described by the relationship

$$F(s) = L\{f(t)\} = \int_0^{\infty} f(t)e^{-st}dt. \quad (9.4)$$

The use of the one-sided Laplace transform is better suited for the purposes of this book, as the signals considered are usually causal signals. In order to return from the s -domain back to the time domain, the inverse Laplace transform is applied. The inverse Laplace transform is denoted by the symbol $L^{-1}\{.\}$; that is, one can write

$$f(t) = L^{-1}\{F(s)\}, \quad (9.5)$$

or alternatively

$$F(s) \xrightarrow{L^{-1}} f(t). \quad (9.6)$$

The mathematical expression that describes the inverse Laplace transform of a function $F(s)$ is

$$f(t) = L^{-1}\{F(s)\} = \frac{1}{2\pi j} \int_{\sigma-j\infty}^{\sigma+j\infty} F(s)e^{st} dt. \quad (9.7)$$

9.2 Commands `laplace` and `ilaplace`

The computation of the integrals given in (9.3), (9.4), and (9.7) can sometimes be quite hard. This is the reason that already computed Laplace transform pairs are used to compute the Laplace transform and the inverse Laplace transform of a function (or signal). However, in MATLAB®, the Laplace transform $F(s)$ of a function $f(t)$ is easily computed by executing the command `laplace`. Moreover, the inverse Laplace transform of a function $F(s)$ is obtained by executing the command `ilaplace`. Before using these two commands, the declaration of complex frequency s and of time t as symbolic variables is necessary. Recall that in order to define a symbolic variable the command `syms` is used. Finally, note that the command `laplace` computes the unilateral Laplace transform of a function.

Example

Compute the (unilateral) Laplace transform of the function $f(t) = e^{-t}$.

Commands	Results	Comments
<code>syms t s</code> <code>t = t</code> <code>s = s</code>		The command <code>syms</code> is used to define the symbolic variables t and s . Next, we confirm that t and s are symbolic variables.
<code>f = exp(-t);</code> <code>laplace(f)</code>	<code>ans = 1/(1+s)</code>	The function $f(t) = e^{-t}$ is defined as symbolic expression and the (unilateral) Laplace transform $F(s) = L\{f(t)\}$ is computed. The result is $F(s) = 1/(1 + s)$. That is, $L\{e^{-t}\} = 1/(1 + s)$.

Example

Compute the inverse Laplace transform of the function $F(s) = 1/(1 + s)$.

Commands	Results	Comments
<code>syms t s</code>		The command <code>syms</code> is used to define the symbolic variables t and s .
<code>F = 1/(1+s);</code> <code>ilaplace(F)</code>	<code>ans = exp(-t)</code>	The function $F(s) = 1/(1 + s)$ is defined as symbolic expression and the inverse Laplace transform $f(t) = L^{-1}\{F(s)\}$ is computed. The result is $f(t) = e^{-t}$.

Functions $f(t) = e^{-t}$ and $F(s) = 1/(1 + s)$ are a Laplace transform pair. In other words, the (unilateral) Laplace transform of $f(t) = e^{-t}$ is $F(s) = 1/(1 + s)$, while the inverse Laplace transform of $F(s) = 1/(1 + s)$ is $f(t) = e^{-t}$. A Laplace transform pair is denoted by $f(t) \leftrightarrow F(s)$. In our case, the Laplace transform pair is $e^{-t}u(t) \leftrightarrow 1/(1 + s)$. In Section 9.4, the most common Laplace transform pairs are presented. At the moment, let us introduce alternative syntaxes of the `laplace` and `ilaplace` commands. The most effective and in the same time simple syntax of the command `laplace` is `laplace(f, s)`. In this way, the Laplace transform of the signal $f(t)$ is expressed with the second input argument (here is s) as the independent variable. This syntax is optimal and makes possible the computation of the Laplace transform in every case. Such a case is when the Laplace transform of a constant function has to be computed.

Commands	Results	Comments
<code>syms t s</code> <code>f = 1;</code> <code>laplace(f)</code>	<code>??? Function 'laplace' is not defined for values of class 'double'.</code>	When the simple syntax of the command <code>laplace</code> is used, MATLAB cannot compute the Laplace transform of the constant function $f(t) = 1$.
<code>laplace(f, s)</code>	<code>ans = 1/s</code>	Using the optimal syntax the Laplace transform of $f(t) = 1$ is computed. The result is $F(s) = L\{1\} = 1/s$.

Moreover, the Laplace transform of a function can be expressed in terms of another variable (e.g., w).

Commands	Results	Comments
<code>syms t s w</code> <code>f = exp(-t);</code> <code>laplace(f, w)</code>	<code>ans = 1/(1+w)</code>	The Laplace transform of $f(t) = e^{-t}$ is expressed with w as the independent variable.
<code>laplace(f)</code>	<code>ans = 1/(1+s)</code>	By default, the Laplace transform is expressed with s as the independent variable.

Finally, a common mistake that has to be avoided is trying to express the Laplace transform of a function $f(t)$ by using the variable t as the independent variable.

Commands	Results	Comments
<code>f = exp(-t); laplace(f, t)</code>	??? Error using ==> sym.maple.Error, (in laplace) Third parameter t must be free of the second parameter t.	This is a logic mistake if one considers that the command is to transfer the signal from the domain of t to the domain of t .

However, variable t can be used (even if there is no particular meaning in doing that) as the independent variable of the Laplace transform of a function, if the function is defined in terms of another variable.

Commands	Results	Comments
<code>syms n f = exp(-n); laplace(f, t)</code>	<code>ans = 1/(1+t)</code>	The function $f(n) = e^{-n}$ is defined and the Laplace transform of $f(n)$ is expressed in terms of t .
<code>f = exp(-s); laplace(f)</code>	<code>ans = 1/(1+t)</code>	If the independent variable is s , the Laplace transform is expressed by default in terms of t .

Notice that if a symbolic variable is once declared through the command `syms`, we do not have to declare it each time. Regarding the inverse Laplace transform, the optimal syntax of the command `ilaplace` is `ilaplace(F, t)`. In this way, the inverse Laplace transform of the signal $F(s)$ is expressed with the second input argument (here is t) as the independent variable. This syntax is optimal and makes possible the computation of the inverse Laplace transform in every case. Such a case is when the inverse Laplace transform of a constant function has to be computed.

Commands	Results	Comments
<code>syms t s F = 1; ilaplace(F)</code>	??? Function 'ilaplace' is not defined for values of class 'double'.	When the simple syntax of the command <code>ilaplace</code> is used, MATLAB cannot compute the inverse Laplace transform of the constant function $F(s) = 1$.
<code>ilaplace(F, t)</code>	<code>ans = dirac(t)</code>	By using the optimal syntax of the <code>ilaplace</code> command the inverse Laplace transform of $F(s) = 1$ is computed. The result is $f(t) = L^{-1}[F(s)] = \delta(t)$.

Moreover, there is the possibility to express the inverse Laplace transform of a function $F(s)$ with a different variable (e.g., n) as independent variable. Furthermore, trying to express the inverse Laplace transform of a function $F(s)$ by using the variable s as the independent variable has to be avoided. Finally, (and even if there is no particular meaning in doing that) the variable s can be used as the independent variable of the inverse Laplace transform of a function, if the function is defined in terms of another variable.

Commands	Results	Comments
<pre>syms n F = 1/(s+2) f = ilaplace(F, n)</pre>	<pre>f = exp(-2*n)</pre> <p>??? Error using ==> sym.maple.Error, (in invlaplace) Third parameter s must be free of second parameter s</p>	The inverse Laplace transform of the function $F(s) = 1/(s + 2)$, expressed with n as the independent variable.
<pre>syms w F = 1/(w+2); ilaplace(F, s)</pre>	<pre>ans = exp(-2*s)</pre>	This is a logic mistake if one considers that the command is to transfer the signal from the s -domain to the s -domain.
		The function $F(w) = 1/(w + 2)$ is defined and the inverse Laplace transform is expressed with s as the independent variable.

Finally, an alternative available syntax for the Laplace transform computation is `laplace(f, t, s)`; that is, the function f is transferred from t to s , while the inverse Laplace transform is computed using the command `ilaplace(F, s, t)`; that is, the function F is transferred from s to t .

9.3 Region of Convergence

The Laplace transform of a function does not always exist as the integral of (9.3) does not always converge. Consider, for example, the signal $x(t) = e^{-at}u(t)$, $a \in \mathbb{R}$. The Laplace transform $X(s)$ of $x(t)$ is given by

$$X(s) = L\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-st}dt = \int_{-\infty}^{\infty} e^{-at}u(t)e^{-st}dt = \int_0^{\infty} e^{-(a+s)t}dt = -\frac{1}{s+a}(e^{-(a+s)t}|_{t \rightarrow \infty} - 1).$$

Recall that s is a complex variable, hence the term $e^{-(a+s)t}|_{t \rightarrow \infty}$ does not become infinite as long as $\operatorname{Re}\{s + a\} > 0$. Hence, the Laplace transform of $x(t) = e^{-at}u(t)$, $a \in \mathbb{R}$ converges as long as $\operatorname{Re}\{s\} > -a$. The area that corresponds to the values of s such that $\operatorname{Re}\{s\} > -a$ is called region of convergence (ROC). The ROC of the Laplace transform of $x(t) = e^{-at}u(t)$ is depicted in Figure 9.1.

Hence, for $\operatorname{Re}\{s\} > -a$, the Laplace transform of $x(t) = e^{-at}u(t)$, $a \in \mathbb{R}$ is $X(s) = 1/(s + a)$. In this book, due to the fact that we focus on the MATLAB implementation, referring to the ROC of the Laplace transform of a signal is omitted for simplicity. However, for clarity reasons the ROC of the Laplace transform of a signal must be specified. For example, the Laplace transform of $x(t) = -e^{-at}u(-t)$, $a \in \mathbb{R}$ is $X(s) = 1/(s + a)$ with ROC $\operatorname{Re}\{s\} < -a$. In other words, the signals $x_1(t) = e^{-at}u(t)$, $a \in \mathbb{R}$ and $x_2(t) = -e^{-at}u(-t)$, $a \in \mathbb{R}$ have the same Laplace transform but with different ROC.

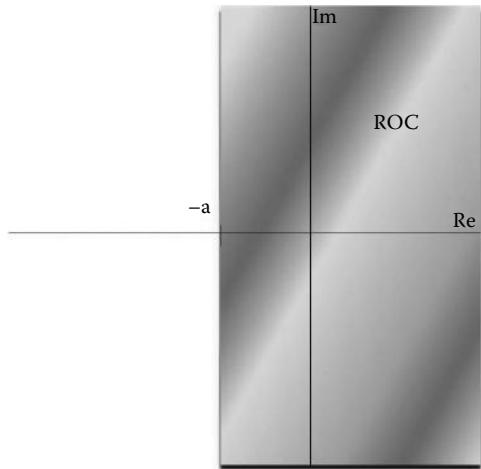


FIGURE 9.1
Region of convergence of the Laplace transform of $x(t) = e^{-at}u(t)$.

9.4 Laplace Transform Pairs

In this section, the most common Laplace transform pairs are illustrated. As mentioned in Section 9.2, the computation of the Laplace or the inverse Laplace transforms requires the computation of the integrals given in (9.3), (9.4), and (9.7), and it can be quite hard. This is the reason that already computed Laplace transform pairs are used to compute the Laplace or the inverse Laplace transform of a function. Thus, the computational procedure is to express a complicated function of interest in terms of functions with known Laplace (or inverse Laplace) transform, and then based on the properties of Laplace transform to compute the Laplace (or inverse Laplace) transform of the complicated function. In the table below, the most common Laplace transform pairs are given. The illustrated pairs are confirmed by using the commands `laplace` and `ilaplace`. Recall that command `laplace` computes the unilateral Laplace transform ($t \geq 0$). Thus, the unit step function (implemented in MATLAB with the command `heaviside`) can be omitted.

Time Domain	s-Domain	Commands	Results
$x(t)$	$X(s)$	<code>syms s t a w</code>	—
$\delta(t)$	1	<code>laplace(dirac(t), s)</code>	<code>ans = 1</code>
$u(t)$	$1/s$	<code>u=heaviside(t); laplace(u, s)</code>	<code>ans = 1/s</code>
$\frac{d^n\delta(t)}{dt^n}$	s^n	<code>n=4 x=diff(dirac(t), n); X= laplace(x, s)</code>	$X=s^4$
$e^{-at}u(t)$	$1/(s + a)$	<code>u=heaviside(t); x=exp(-a*t)*u; laplace(x, s)</code>	<code>ans = 1/(s+a)</code>
$e^{+at}u(t)$	$1/(s - a)$	<code>ilaplace(1/(s-a), t)</code>	<code>ans = exp(a*t)</code>
$e^{j\omega_0 t}u(t)$	$1/(s - j\omega_0)$	<code>laplace(exp(j*w*t), s)</code>	<code>ans = 1/(s - i*w)</code>
$\cos(\omega_0 t)u(t)$	$s/(s^2 + \omega_0^2)$	<code>X= s/(s^2+w^2); x= ilaplace(X)</code>	<code>x=cos(w*t)</code>

(continued)

Time Domain	s-Domain	Commands	Results
$\sin(\omega_0 t)u(t)$	$\omega_0 / (s^2 + \omega_0^2)$	$x = \sin(w*t);$ $X = \text{laplace}(x)$	$X = w / (s^2 + w^2)$
$e^{-at} \cos(\omega_0 t)u(t)$	$(s+a) / ((s+a)^2 + \omega_0^2)$	$x = \exp(-a*t) * \cos(w*t);$ $\text{laplace}(x, s)$	$\text{ans} = (s+a) / ((s+a)^2 + w^2)$
$e^{-at} \sin(\omega_0 t)u(t)$	$\omega_0 / ((s+a)^2 + \omega_0^2)$	$X = w / ((s+a)^2 + w^2);$ $x = \text{ilaplace}(X, t)$	$x = \exp(-a*t) * \sin(w*t)$
$t^n u(t)$	$n! / s^{n+1}$	$x = (t^5) * \text{heaviside}(t);$ $\text{laplace}(x, t, s)$	$\text{ans} = 120 / s^6$ % n=5
$e^{-at} t^n u(t)$	$n! / (s+a)^{n+1}$	$X = \text{factorial}(10) /$ $(s+a)^{11};$ $\text{ilaplace}(X, t)$	$\text{ans} = t^{10}$ *exp(-a*t) % n=10
$t \cos(\omega_0 t)u(t)$	$(s^2 - \omega_0^2) / (s^2 + \omega_0^2)^2$	$x = t * \cos(w*t);$ $\text{laplace}(x, t, s)$	$\text{ans} = 1 / (s^2 + w^2)$ ^2 * (s^2 - w^2)
$(t^{n-1} / (n-1)) e^{-at} u(t)$	$1 / (s+a)^n$	$X = 1 / (s+a)^6;$ $x = \text{ilaplace}(X, t)$	$x = 1 / 120 * t^5$ *exp(-a*t) % n=6

9.5 Laplace Transform Properties and Theorems

In this section, we present the main properties of the Laplace transform. Each property is verified by an appropriate example.

1. *Linearity.* Let $X_1(s) = L\{x_1(t)\}$ and $X_2(s) = L\{x_2(t)\}$. Then for any scalars a_1, a_2 the following property stands:

$$L\{a_1 x_1(t) + a_2 x_2(t)\} = a_1 X_1(s) + a_2 X_2(s). \quad (9.8)$$

Commands	Results	Comments
<pre>syms t s x1 = exp(-t); x2 = cos(t); a1 = 3; a2 = 4; Le = a1*x1+a2*x2; Left = laplace(Le, s)</pre>	$\text{Left} = 3 / (1+s) + 4*s / (s^2+1)$	Equation 9.8 is verified for $a_1 = 3, a_2 = 4, x_1(t) = e^{-t}u(t)$, and $x_2(t) = \cos(t)u(t)$. First, the left side of Equation 9.8 is computed. The right side is computed below.
<pre>X1 = laplace(x1); X2 = laplace(x2); Right = a1*X1+a2*X2</pre>	$\text{Right} = 3 / (1+s) + 4*s / (s^2+1)$	The two sides are equal; thus, the linearity property is clearly confirmed.

2. *Time shifting.* If $X(s) = L\{x(t)u(t)\}$, then for any $t_0 > 0$

$$L\{x(t - t_0)u(t - t_0)\} = e^{-st_0}X(s). \quad (9.9)$$

Commands	Results	Comments
<pre>syms t s t0 = 2; Le = cos(t-t0)*heaviside(t-t0); Left = laplace(Le)</pre>	$\text{Left} = \exp(-2*s) * s / (s^2 + 1)$	Equation 9.9 is verified by using the signal $x(t) = \cos(t)u(t)$ and the time shift $t_0 = 2$. First, the left side of Equation 9.9 is computed. The right side is computed below.
$X = \text{laplace}(\cos(t), s);$ $\text{Right} = \exp(-s*t0) * X$	$\text{Right} = \exp(-2*s) * s / (s^2 + 1)$	The two sides are equal; thus, the time-shifting property is validated.

3. *Complex frequency shifting.* If $X(s) = L\{x(t)u(t)\}$, then for any s_0 ,

$$L\{x(t)e^{s_0 t}\} = X(s - s_0). \quad (9.10)$$

Commands	Results	Comments
$x = t^3;$ $s0 = 2;$ $f = x * \exp(s0*t);$ $L = \text{laplace}(f, s)$	$L = 6 / (s - 2)^4$	Equation 9.10 is verified by using the signal $x(t) = t^3 u(t)$ and the frequency shift $s_0 = 2$. First, the left side of Equation 9.10 is computed. The right side is computed below.
$X = \text{laplace}(x, s);$ $R = \text{subs}(X, s, s-2)$	$R = 6 / (s - 2)^4$	In order to compute $X(s - s_0)$, the variable s in $X(s)$ is substituted by $s - 2$. The two sides are equal; thus, the complex frequency-shifting property is valid.

4. *Time scaling.* If $X(s) = L\{x(t)\}$, then for any $b > 0$,

$$L\{x(bt)\} = \frac{1}{b}X\left(\frac{s}{b}\right). \quad (9.11)$$

Commands	Results	Comments
syms b $le = \exp(-b*2*t);$ $L = \text{laplace}(le, s)$	$L = 1 / (s + 2*b)$	Equation 9.11 is verified by using the signal $x(t) = e^{-2t}u(t)$, while b is defined as symbolic variable. First, the left side of Equation 9.11 is computed. The right side is computed below.
$x = \exp(-2*t);$ $X = \text{laplace}(x, s)$	$X = 1 / (s + 2)$	Calculation of $X(s)$.
$R = (1/b) * \text{subs}(X, s, s/b)$	$R = 1/b / (s/b + 2)$	In order to compute $(1/b)X(s/b)$, the variable s in $X(s)$ is substituted by s/b .
$\text{simplify}(R)$	$\text{ans} = 1 / (s + 2*b)$	The result is shortened by using the command <code>simplify</code> and the time-scaling property is verified.

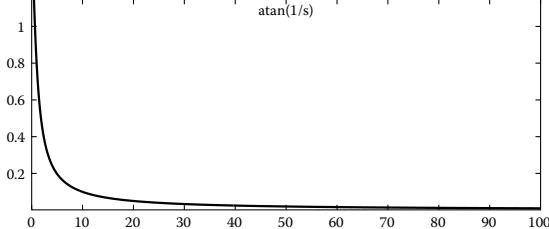
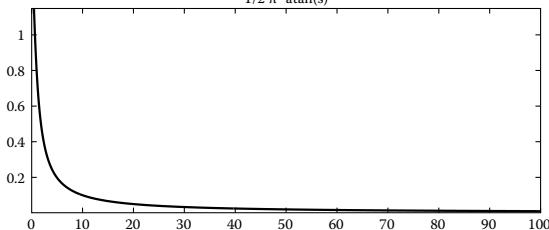
5. *Differentiation in the s-domain.* If $X(s) = L\{x(t)\}$ and $n = 1, 2, \dots$ then

$$L\{(-t)^n x(t)\} = \frac{d^n X(s)}{ds^n}. \quad (9.12)$$

Commands	Results	Comments
<code>x = exp(t); f = (-t)^5 * x; L = laplace(f, s)</code>	$L = -120/(s-1)^6$	Equation 9.12 is verified by using the signal $x(t) = e^t u(t)$ and $n = 5$. First, the left side of Equation 9.12 is computed. The right side is computed below.
<code>X = laplace(x, s); R = diff(X, 5)</code>	$R = -120/(s-1)^6$	The two sides are equal; thus, the property of differentiation in the s-domain is clearly illustrated.

6. *Integration in the complex frequency.* If $X(s) = L\{x(t)\}$, then

$$L\left\{\frac{x(t)}{t}\right\} = \int_s^\infty X(u)du. \quad (9.13)$$

Commands	Results	Comments
<code>x = sin(t); L = laplace(x/t, s); ezplot(L, [0 100])</code>	 A plot titled "atan(1/s)" showing a curve starting at (0, 1) and decaying towards zero as s increases to 100. The x-axis ranges from 0 to 100 with major ticks every 10 units. The y-axis ranges from 0 to 1 with major ticks every 0.2 units.	Equation 9.13 is verified by using the signal $x(t) = \sin(t)u(t)$. First, the left side of Equation 9.13 is computed and plotted.
<code>syms u X = laplace(x, u); R = int(X, u, s, inf); ezplot(R, [0 100])</code>	 A plot titled "1/2 pi - atan(s)" showing a curve starting at (0, 1) and decaying towards zero as s increases to 100. The x-axis ranges from 0 to 100 with major ticks every 10 units. The y-axis ranges from 0 to 1 with major ticks every 0.2 units.	In order to compute the right side of Equation 9.13 the Laplace transform of $x(t)$ is first computed, followed by the integral calculation. The result is also plotted.

Note that using the function `ezplot` is the easiest way of plotting a symbolic expression in an interval of interest (here $0 \leq s \leq 100$).

7. *Differentiation in the time domain.* If $X(s) = L\{x(t)\}$, then

$$L\left\{\frac{dx(t)}{dt}\right\} = sX(s) - x(0). \quad (9.14)$$

For higher-order differentiation, the following relationship stands:

$$L\left\{\frac{d^n x(t)}{dt^n}\right\} = s^n X(s) - \sum_{k=0}^{n-1} s^{n-k-1} x^{(k)}(0). \quad (9.15)$$

Commands	Results	Comments
<code>x = cos(t); L = diff(x, t); laplace(L, s)</code>	<code>ans = -1/(s^2+1)</code>	Equation 9.14 is verified by using the signal $x(t) = \cos(t)u(t)$. Notice that $x(0) = \cos(0) = 1$. First, the left side of Equation 9.14 is computed.
<code>X = laplace(x, s); x0 = 1; R = s*X-x0; simplify(R)</code>	<code>ans = -1/(s^2+1)</code>	The right side of Equation 9.14 is calculated and is equal to the left; thus the differentiation property clearly stands.

8. *Integration in the time domain.* If $X(s) = L\{x(t)\}$, then

$$L\left\{\int_{-\infty}^t x(r)dr\right\} = \frac{X(s)}{s} + \frac{\int_0^\infty x(r)dr}{s}. \quad (9.16)$$

Commands	Results	Comments
<code>syms r x = exp(3*r); le = int(x, r, -inf, t); L = laplace(le, s)</code>	<code>L = 1/3/(s-3)</code>	The signal $x(t) = e^{3t}u(t)$ is considered. First the left side of (9.16) is calculated.
<code>X = laplace(x, s); xr = int(x, r, -inf, 0) R = X/s + xr/s simplify(R)</code>	<code>ans = 1/3/(s-3)</code>	The right side is calculated and is equal to the left; thus, the integration property is verified.

9. *Initial value theorem.* If $X(s) = L\{x(t)\}$, then

$$\lim_{s \rightarrow \infty} sX(s) = x(0). \quad (9.17)$$

Commands	Results	Comments
<code>x=sin(t); X=laplace(x,s); limit(s*X,s,inf)</code>	<code>ans = 0</code>	The signal $x(t) = \sin(t)u(t)$ is considered. In this case, $x(0) = \sin(0) = 0$. Indeed, $\lim_{s \rightarrow \infty} sX(s) = 0 = x(0)$. Thus, the initial value theorem clearly stands.

10. *Final value theorem.* If $X(s) = L\{x(t)\}$, then

$$\lim_{t \rightarrow \infty} x(t) = \lim_{s \rightarrow 0} sX(s). \quad (9.18)$$

Commands	Results	Comments
<code>x=1-exp(-t); limit(x,t,inf)</code>	<code>ans = 1</code>	The signal $x(t) = (1 - e^{-t})u(t)$ is considered. First, the left side of (9.18) is calculated.
<code>X=laplace(x,s); limit(s*X,s,0)</code>	<code>ans = 1</code>	The right side is calculated and is equal to the left; thus, the final value theorem is confirmed.

9.6 Partial Fraction Expansion of a Rational Function

In the usual case, the Laplace transform of a function is expressed as a rational function of s , that is, is given as a ratio of two polynomials of s . The mathematical expression is

$$X(s) = \frac{B(s)}{A(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0}, \quad (9.19)$$

where a_i and b_i are real scalars. First, we discuss the case $m < n$; that is, the case that the degree of the polynomial $B(s)$ of the numerator is lower than the degree of the polynomial $A(s)$ of the denominator. Suppose that λ_i are the roots of $A(s)$. The following cases are considered:

1. *The roots λ_i are distinct;* that is, every root appears only once. In this case, $A(s)$ is written in the factored form $A(s) = a_n \prod_{i=1}^n (s - \lambda_i)$, while $X(s)$ is written as

$$X(s) = \frac{c_1}{s - \lambda_1} + \frac{c_2}{s - \lambda_2} + \cdots + \frac{c_n}{s - \lambda_n}. \quad (9.20)$$

The coefficients c_1, \dots, c_n are computed according to

$$c_i = \lim_{s \rightarrow \lambda_i} (s - \lambda_i) X(s), \quad (9.21)$$

or equivalently according to

$$c_i = [(s - \lambda_i)X(s)]_{s=\lambda_i}. \quad (9.22)$$

Example

Express in partial fraction form the signal that in the s -domain is given by

$$X(s) = \frac{s^2 + 3s + 1}{s^3 + 5s^2 + 2s - 8}.$$

Commands	Results	Comments
<code>A = [1 5 2 -8]; rt = roots(A)</code>	<code>rt = -4.0000 -2.0000 1.0000</code>	First, the roots of the denominator are calculated.
<code>syms s X = (s^2+3*s+1)/(s^3+5*s^2+2*s-8); c1 = limit((s-rt(1))*X,s,rt(1)) c2 = limit((s-rt(2))*X,s,rt(2)) c3 = limit((s-rt(3))*X,s,rt(3))</code>	<code>c1 = 1/2 c2 = 1/6 c3 = 1/3</code>	The coefficients are calculated according to Equation 9.21.
<code>X1 = simplify((s-rt(1))*X); X2 = simplify((s-rt(2))*X); X3 = simplify((s-rt(3))*X); c1 = subs(X1,s,rt(1)) c2 = subs(X2,s,rt(2)) c3 = subs(X3,s,rt(3))</code>	<code>c1 = 0.5000 c2 = 0.1667 c3 = 0.3333</code>	Alternative calculation of the coefficients according to Equation 9.22.

Therefore,

$$X(s) = \frac{s^2 + 3s + 1}{s^3 + 5s^2 + 2s - 8} = \frac{1/2}{s + 4} + \frac{1/6}{s + 2} + \frac{1/3}{s - 1}.$$

One may ask the reason for doing this procedure (i.e., expressing $X(s)$ from rational form to partial fraction form). Looking into the Laplace transform pairs that were presented in Section 9.4, we notice the pair $e^{-at}u(t) \leftrightarrow 1/(s + a)$. Thus, the inverse Laplace transform of

$$X(s) = \frac{1/2}{s + 4} + \frac{1/6}{s + 2} + \frac{1/3}{s - 1}$$

can be directly computed as $x(t) = (1/2)e^{-4t} + (1/6)e^{-2t} + (1/3)e^t$. The result is confirmed by using the command `ilaplace`.

Commands	Results	Comments
<code>syms t s ilaplace(X,t)</code>	<code>ans = 1/2*exp(-4*t)+1/3*exp(t)+ 1/6*exp(-2*t)</code>	Confirmation of the theoretical derivation.

2. The roots λ_i are repeated. Suppose that root λ_1 is repeated r times and all other roots are distinct. In this case, $A(s)$ is written in the factored form

$$A(s) = a_n(s - \lambda_1)^r \prod_{i=r+1}^n (s - \lambda_i),$$

while $X(s)$ is written as

$$X(s) = \frac{c_1}{s - \lambda_1} + \frac{c_2}{(s - \lambda_1)^2} + \cdots + \frac{c_r}{(s - \lambda_1)^r} + \frac{c_{r+1}}{s - \lambda_{r+1}} + \cdots + \frac{c_n}{s - \lambda_n}. \quad (9.23)$$

The coefficients c_1, \dots, c_n are calculated according to

$$\begin{aligned} c_i &= \lim_{s \rightarrow \lambda_1} \frac{1}{(r-i)!} \frac{d^{r-i}[(s - \lambda_1)^r X(s)]}{ds^{r-i}}, \quad i = 1, \dots, r, \\ c_i &= \lim_{s \rightarrow \lambda_i} (s - \lambda_i) X(s), \quad i = r+1, \dots, n \end{aligned} \quad (9.24)$$

Example

Express in partial fraction form the signal

$$X(s) = \frac{s^2 + 3s + 1}{s^3 - 3s + 2}.$$

Commands	Results/Comments
<code>A = [1 0 -3 2]; rt = roots(A)</code>	First, the roots of the denominator are calculated. The root $\lambda = 1$ is repeated two times. Notice that the coefficient of s^2 is zero and must be taken into account when formulating matrix A. <code>rt = -2.0000 1.0000 1.0000</code>
<code>syms s X = (s^2+3*s+1)/(s^3-3*s+2); c1=limit((s-rt(1))*X,s,rt(1)) % alternatively % c1 = subs((s-rt(1))*X,s,rt(1))</code>	The coefficient c_1 that corresponds to the distinct root $\lambda = -2$ is computed according to the lower part of (9.24). In reality, the coefficient that corresponds to $\lambda = -2$ is coefficient $c_r, r = 3$, but since the distinct root is the first sample of vector rt we name the coefficient according to the MATLAB implementation. Thus, c_1 is <code>c1 = -1/9</code>
<code>r = 2; i = 1; f = ((s-1)^r) *X; d = diff(f,s,r-i); fact = 1/factorial(r-i); c2 = limit(fact*d,s,1)</code>	In order to compute the coefficient c_2 (that corresponds to $i = 1$), first we set $r = 2$ as there are two repeated roots. Calculation of $f(s) = (s - \lambda_1)^r X(s)$. Calculation of $\frac{d^{r-i}f(s)}{ds^{r-i}}$, that is, of $\frac{d^{r-i}[(s - \lambda_1)^r X(s)]}{ds^{r-i}}$.
<code>i = 2; d = diff(f,s,r-i); fact = 1/factorial(r-i); c3 = limit(fact*d,s,1)</code>	Calculation of $1/(r - i)!$ The coefficient c_2 is computed according to the upper part of (9.24). <code>c2 = 10/9</code> The same procedure is followed for the calculation of c_3 (here $i = 2$). <code>c3 = 5/3</code>

Therefore,

$$X(s) = \frac{s^2 + 3s + 1}{s^3 - 3s + 2} = \frac{-1/9}{s+2} + \frac{10/9}{s-1} + \frac{5/3}{(s-1)^2}.$$

The inverse Laplace transform is directly computed by using the Laplace transform pair $e^{-at}t^n u(t) \leftrightarrow n!/(s+a)^{n+1}$. Thus, the inverse Laplace transform of $X(s)$ is $x(t) = L^{-1}\{X(s)\} = -(1/9)e^{-2t} + (10/9)e^t + (5/3)te^t$. The result is confirmed by using the command `ilaplace`.

Commands	Result	Comments
<code>syms t s ilaplace(X, t)</code>	<code>ans = -1/9*exp(-2*t) + (5/3*t+10/9)*exp(t)</code>	Confirmation of the theoretical derivation.

Let us consider now the case where $m \geq n$, that is, the degree of the numerator polynomial $B(s)$ is greater than or equal to the degree of the denominator polynomial $A(s)$. In this case, we have to implement the division between $B(s)$ and $A(s)$. The signal $X(s)$ is written as

$$X(s) = \frac{B(s)}{A(s)} = K(s) + \frac{G(s)}{A(s)}.$$

The division of two polynomials is computed by the command `deconv`. A more detailed explanation of division between polynomials is given in Section 1.16. After the division, the process of partial fraction expansion is applied to the rational part of $X(s)$, that is, is applied to $G(s)/A(s)$.

Example

Express in partial fraction form the signal

$$X(s) = \frac{s^2 - 12s + 11}{s^2 + 4s + 3}.$$

Commands	Results	Comments
<code>num = [1 -12 11]; den = [1 4 3]; [k, g] = deconv(num, den)</code>	<code>k = 1 g = 0 -16 8</code>	The signal $X(s)$ is written as $X(s) = 1 + \frac{-16s + 8}{s^2 + 4s + 3}$.
<code>rt = roots(den)</code>	<code>rt = -3 -1</code>	Calculation of the roots of the denominator.
<code>syms s GA = (-16*s+8)/(s^2+4*s+3);</code>		Definition of $G(s)/A(s)$.
<code>c1 = limit((s-rt(1))*GA, s, rt(1)) c2 = limit((s-rt(2))*GA, s, rt(2))</code>	<code>c1 = -28 c2 = 12</code>	Calculation of the coefficients c_1 and c_2 according to Equation 9.21.

Therefore, the signal $X(s)$ is written in partial fraction form as

$$X(s) = 1 - \frac{28}{s+3} + \frac{12}{s+1}.$$

9.6.1 The Command `residue`

In Section 9.6, we introduced a procedure that has to be followed in order to express a function from rational form to partial fraction form. Alternatively, this procedure can be implemented directly in MATLAB by using the command `residue`. The syntax is $[R, P, K] = \text{residue}(B, A)$, where B is the vector containing the coefficients of the numerator polynomial and A is the vector of the coefficients of the denominator polynomial. Suppose that a signal $X(s)$ is written in the rational form

$$X(s) = \frac{B(s)}{A(s)} = \frac{b_ms^m + b_{m-1}s^{m-1} + \dots + b_1s + b_0}{a_ns^n + a_{n-1}s^{n-1} + \dots + a_1s + a_0}.$$

By defining the vectors $B = [b_m, \dots, b_0]$ and $A = [a_n, \dots, a_0]$, and executing the command $[R, P, K] = \text{residue}(B, A)$, the signal $X(s)$ can be written in partial fraction form as

$$X(s) = \frac{r_1}{s - p_1} + \frac{r_2}{s - p_2} + \dots + \frac{r_n}{s - p_n} + K,$$

where

$$R = [r_1, r_2, \dots, r_n]$$

$$P = [p_1, p_2, \dots, p_n]$$

K is the residue of the division of the two polynomials

K is null when $m < n$. The command `residue` is illustrated in the following examples.

Example

Express in partial fraction form the signal

$$X(s) = \frac{s^2 + 3s + 1}{s^3 + 5s^2 + 2s - 8}.$$

Commands	Results	Comments
<pre>num = [1 3 1]; den = [1 5 2 -8]; [R, P, K] = residue(num, den)</pre>	<pre>R = 0.5000 0.1667 0.3333 P = -4.0000 -2.0000 1.0000 K = []</pre>	<p>The signal $X(s)$ is expanded in partial fraction form. The result is similar to the one obtained at the second example of the previous section. Notice that K is empty; that is, there is no residue.</p>
<pre>syms s X = R(1)/(s - P(1)) + R(2)/(s - P(2)) + R(3)/(s - P(3)) pretty(X)</pre>	<p>The signal $X(s)$ is written as $1/2\frac{1}{s+4} + 1/6\frac{1}{s+2} + 1/3\frac{1}{s-1}$.</p>	

Example

Express in partial fraction form the signal

$$X(s) = \frac{s^2 + 3s + 1}{s^3 - 3s + 2}.$$

Commands	Results		Comments
<pre>num = [1 3 1]; den = [1 0 -3 2]; [R, P, K] = residue (num, den)</pre>	R = -0.1111 1.6667 P = -2.0000 1.0000 K = []	1.1111 1.0000	The signal $X(s)$ is written in partial fraction form as $X(s) = -0.1111/(s+2) + 1.1111/(s-1) + 1.6667/(s-1)^2$. The result is same as the one obtained at the second example of the previous section.

Example

Express in partial fraction form the signal

$$X(s) = \frac{s^2 - 12s + 11}{s^2 + 4s + 3}.$$

In this case, the degree of the numerator and denominator polynomials is equal. Also notice that this is the same signal expanded in partial fraction form in the last example of the previous section.

Commands	Results		Comments
<pre>num = [1 -12 11]; den = [1 4 3]; [R, P, K] = residue (num, den)</pre>	R = -28 12 P = -3 -1 K = 1		The signal $X(s)$ is written in the partial fraction form as $X(s) = 1 - \frac{28}{s+3} + \frac{12}{s+1}$. The result is same as the one derived analytically.

Example

Express in partial fraction form the signal

$$X(s) = \frac{s^2 - 3s + 2}{s^2 + 4s + 5}.$$

In this case, the roots of the denominator polynomial are complex numbers. The complex roots do not really change anything in the procedure followed.

Commands	Results		Comments
<pre>n = [1 -3 2]; d = [1 4 5]; [R, P, K] = residue (n, d)</pre>	R = -3.50 - 5.50i -3.50 + 5.50i P = -2.00 + 1.00i -2.00 - 1.00i K = 1		The signal $X(s)$ is expressed in partial fraction form as $X(s) = \frac{-3.5 - 5.5j}{s + 2 - j} + \frac{-3.5 + 5.5j}{s + 2 + j} + 1$.

Example

Express in partial fraction form the signal

$$X(s) = \frac{s^3 - 3s + 2}{s^2 + 4s + 5}.$$

In this case, the degree of the denominator polynomial is lower than the degree of the numerator polynomial. Moreover, the roots of the denominator polynomial are complex numbers.

Commands	Results	Comments
<code>n = [1 0 -3 2];</code>	$R = 4.00 - 3.00i$	The signal $X(s)$ is expressed in partial fraction
<code>d = [1 4 5];</code>	$4.00 + 3.00i$	form as $X(s) = \frac{4 - 3j}{s + 2 - j} + \frac{4 + 3j}{s + 2 + j} + s - 4$.
<code>[R, P, K] = residue(n, d)</code>	$P = -2.00 + 1.00i$	Notice the way that the residue $K(s)$ is defined.
	$-2.00 - 1.00i$	
	$K = 1 - 4$	

Finally, one thing that is worth noticing is the fact that the vector P consists of the roots of the denominator polynomial.

Commands	Result	Comments
<code>d = [1 4 5];</code> <code>roots(d)</code>	$ans = -2.0000 + 1.0000i$ $-2.0000 - 1.0000i$	Indeed, vector P contains the roots of the denominator polynomial.

The command `residue` can be also used for the inverse operation; that is, it can be used to express into rational form a signal that is written in partial fraction form. The appropriate syntax in this case is `[B, A] = residue(R, P, K)`. The vectors R, P, K of the partial fraction form are the input arguments, while the vectors B and A that contain the coefficients of the numerator and denominator polynomials, respectively, are computed by the `residue` command.

Example

Express in rational form the signal

$$X(s) = \frac{1}{s+2} + \frac{2}{s-1} + \frac{3}{s+1}.$$

Commands	Results	Comments
<code>R = [1 2 3];</code> <code>P = [-2 1 -1];</code> <code>K = []</code> <code>[n, d] = residue(R, P, K)</code>	$n = 6 \quad 9 \quad -3$ $d = 1 \quad 2 \quad -1 \quad -2$	Inverse use of the command <code>residue</code> .
<code>syms s</code> <code>X = (n(1)*s^2 + n(2)*s + n(3)) /</code> <code>(d(1)*s^3 + d(2)*s^2 + d(3)*s + d(4))</code> <code>pretty(X)</code>	$\frac{6s^2 + 9s - 3}{s^3 + 2s^2 - s - 2}$	The signal $X(s)$ is expressed in rational form.

9.7 Convolution in Time and in Complex Frequency

In this section, two very important theorems referring to the convolution of two signals in the time domain, as well as to the convolution of two signals in the s -domain, are presented.

9.7.1 Convolution in the Time Domain

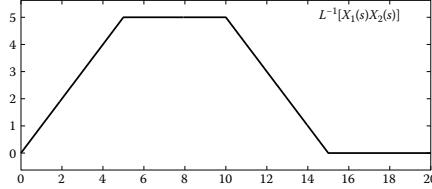
In Chapter 4, we discussed the convolution between two signals $x_1(t)$ and $x_2(t)$. In the chapters that followed, we stated that convolution is transformed into product in the frequency domain. In this section, we present a similar result for the Laplace transform. So let $X_1(s) = L\{x_1(t)\}$ denote the Laplace transform of $x_1(t)$ and $X_2(s) = L\{x_2(t)\}$ denote the Laplace transform of $x_2(t)$. Then, the Laplace transform of the convolution between $x_1(t)$ and $x_2(t)$ is equal to the product of the corresponding Laplace transforms $X_1(s)$ and $X_2(s)$. In other words, convolution in the time domain is transformed into multiplication in the complex frequency domain. The mathematical expression is

$$L\{x_1(t) * x_2(t)\} = X_1(s) \cdot X_2(s), \quad (9.25)$$

where the symbol $*$ denotes convolution. Applying inverse Laplace transform in both (left and right) parts of Equation 9.25 yields $L^{-1}\{L\{x_1(t) * x_2(t)\}\} = L^{-1}\{X_1(s) \cdot X_2(s)\}$. Therefore, Equation 9.25 is equivalently written as

$$x_1(t) * x_2(t) = L^{-1}\{X_1(s) \cdot X_2(s)\}. \quad (9.26)$$

Hence, in order to verify Equation 9.25, it is enough to verify (9.26). The signals $x_1(t) = u(t) - u(t - 5)$ and $x_2(t) = u(t) - u(t - 10)$ are used for that purpose.

Commands	Result	Comments
<pre>syms t s x1=heaviside(t)-heaviside(t-5); x2=heaviside(t)-heaviside(t-10); X1=laplace(x1,s); X2=laplace(x2,s); R=ilaplace(X1*X2,t); ezplot(R,[0 20]); title('L^-1[X_1(s)X_2(s)]')</pre>		First, the right side of (9.26) is computed and the result is plotted.

(continued)

Commands	Result	Comments
<pre>t1=0:.01:5; t2=5.01:.01:10; x1=[ones(size(t1)), zeros(size(t2))]; x2=ones(size([t1 t2])); y=conv(x1,x2)*.01; plot(0:.01:20,y); title('Convolution') ylim([-0.5 5.5])</pre>		The convolution between $x_1(t)$ and $x_2(t)$ (left side of (9.26)) is calculated and the result is plotted.

The two graphs are identical; thus Equation 9.25 is confirmed.

9.7.2 Convolution in the Complex Frequency Domain

The corresponding theorem referring to the convolution in the s -domain is presented in this section. Let $X_1(s) = L\{x_1(t)\}$ denote the Laplace transform of $x_1(t)$ and $X_2(s) = L\{x_2(t)\}$ denote the Laplace transform of $x_2(t)$. Then, the Laplace transform of the product $x_1(t)x_2(t)$ is equal to the convolution of the corresponding Laplace transforms $X_1(s)$ and $X_2(s)$ multiplied by the term $1/2\pi j$. The mathematical expression is

$$L\{x_1(t)x_2(t)\} = \frac{1}{2\pi j} X_1(s) * X_2(s). \quad (9.27)$$

Example

Compute the convolution in the complex frequency domain between the signals $X_1(s) = 1/(s^2 + 4)$ and $X_2(s) = 8/(s - 3)$.

The convolution of $X_1(s)$ and $X_2(s)$ is computed according to the left side of (9.27).

Commands	Result	Comments
<pre>syms t s X1=1/(s^2+4); X2=8/(s-3); x1=ilaplace(X1,t); x2=ilaplace(X2,t); con=(2*pi*j)*laplace(x1*x2)</pre>	<pre>con=4*i*pi/(1/4*(s-3)^2+1)</pre>	Convolution computation of $X_1(s)$ and $X_2(s)$ according to the left side of (9.27).

9.8 Using the Laplace Transform to Solve Differential Equations

In this section, we introduce a method for solving linear differential equations with constant coefficients with use of the Laplace transform. A linear differential equation with constant coefficients is described by the relationship

$$a_n \frac{d^n y(t)}{dt^n} + a_{n-1} \frac{d^{n-1} y(t)}{dt^{n-1}} + \cdots + a_1 \frac{dy(t)}{dt} + a_0 y = x(t). \quad (9.28)$$

The coefficients $a_i, i = 1, \dots, n$ are real numbers. Furthermore, the initial conditions $y(0) = b_0, y'(0) = b_1, \dots, y^{(n-1)}(0) = b_{n-1}$ are known, while the term $y^{(i)}(0)$ denotes the i th degree derivative of $y(t)$ evaluated at $t = 0$. Recall that the solution of a differential equation is a function $y(t)$ that satisfies the differential equation for any t and also satisfies the initial conditions. The Laplace transform property that is used to solve this type of differential equations is the differentiation in the time-domain property, given by Equation 9.15. For convenience the differentiation property is also stated below:

$$L\left\{\frac{d^n x(t)}{dt^n}\right\} = sL\left\{\frac{d^{n-1} x(t)}{dt^{n-1}}\right\} - x^{(n)}(0). \quad (9.29)$$

The computational procedure is

1. The Laplace transform is applied to both sides of the differential equation.
2. From the linearity property the Laplace transform of a sum equals the sum of the Laplace transforms of the sum terms. Moreover, the scalars a_i are intergraded out of the Laplace transforms.
3. The Laplace transforms of the derivatives are computed according to Equation 9.29.
4. The algebraic equation that comes up is solved for $Y(s)$.
5. The inverse Laplace transform of $Y(s)$ is evaluated, that is, $y(t)$ is computed. The function $y(t)$ is the solution of the differential equation.

Example

Find the solution of the differential equation $y''(t) + 3y'(t) + 2y(t) = e^{-t}, y(0) = 2, y'(0) = 3$.

The differential equation of this example is a second-order differential equation; thus, Equation 9.29 is evaluated for $n = 1$ and $n = 2$. The following relationships hold:

$$L\{y(t)\} = Y(s). \quad (9.30)$$

$$L\{y'(t)\} = sL\{y(t)\} - y(0) \Rightarrow \quad (9.31)$$

$$L\{y'(t)\} = sY(s) - y(0). \quad (9.32)$$

$$L\{y''(t)\} = sL\{y'(t)\} - y'(0) \Rightarrow \quad (9.33)$$

$$L\{y''(t)\} = s^2 Y(s) - sy(0) - y'(0). \quad (9.34)$$

The computational procedure is

1. Applying Laplace transform to both sides of the differential equation yields $L\{y''(t) + 3y'(t) + 2y(t)\} = L\{e^{-t}\}$.
2. Due to the linearity property we get $L\{y''(t)\} + 3L\{y'(t)\} + 2L\{y(t)\} = 1/(s+1)$, where the Laplace transform pair $e^{-at}u(t) \leftrightarrow 1/(s+a)$ is applied.

3. The Laplace transforms $L\{y''(t)\}$, $L\{y'(t)\}$, and $L\{y(t)\}$ are substituted according to Equations 9.30 through 9.34, and the differential equation is converted to the algebraic equation $s^2Y(s) - 2s - 3 + 3(sY(s) - 2) + 2Y(s) = 1/(s + 1)$, where the initial conditions $y(0) = 2$ and $y'(0) = 3$ have been taken into account.

4. We solve the equation for $Y(s)$ and get

$$Y(s) = \frac{2s + 1/(s + 1) + 9}{s^2 + 3s + 2}.$$

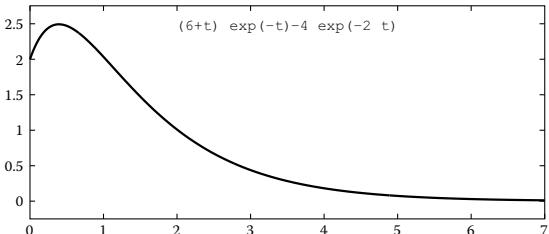
5. The solution of the differential equation is given by the inverse Laplace transform of $Y(s)$; that is, $y(t) = L^{-1}\{Y(s)\} = 4e^{-2t} + (6 + t)e^{-t}$.

A similar computational procedure is implemented in MATLAB in order to compute the solution of the differential equation.

Commands	Results	Comments
<code>syms t s Y</code>		$Y(s)$ is denoted by Y .
<code>X=laplace(exp(-t),s)</code>		The Laplace transform of the right side of the differential equation is computed.
<code>y0=2; yd0=3;</code>		The initial conditions are defined.
<code>Y1=s*Y-y0;</code>		$L\{y'(t)\}$ is denoted by $Y1$, and $Y1$ is defined according to Equation 9.32.
<code>Y2=s*Y1-yd0;</code>		$L\{y''(t)\}$ is denoted by $Y2$, and $Y2$ is defined according to Equation 9.33.
<code>%y2=(s^2)*Y-s*y0-yd0;</code>		Alternatively, $Y2$ can be defined according to Equation 9.34.
<code>G=Y2+3*Y1+2*Y-X;</code>		The term G is a function of Y and s .
<code>SOL=solve(G,Y)</code>	$SOL = (2*s^2+11*s+10)/(s+1)/$ $(s^2+3*s+2)$	Using the command <code>solve</code> allows us to solve for Y . This is the solution of the differential equation expressed in the complex frequency domain. The obtained solution is same as the analytically computed solution.

(continued)

(continued)

Commands	Results	Comments
<code>y_t = ilaplace(SOL, t)</code> <code>ezplot (y_t, [0 7])</code>	$y_t = -4 \cdot \exp(-2 \cdot t) + (t+6) \cdot \exp(-t).$ 	Applying inverse Laplace transform yields the solution $y(t)$ of the differential equation. The solution $y(t)$ is a symbolic expression, thus is plotted with use of the <code>ezplot</code> command.

To confirm that $y(t)$ is in fact the solution of the differential equation, $y(t)$ is inserted in the differential equation. If it satisfies the differential equation as well as the initial conditions, then it is indeed its solution.

Commands	Results	Comments
<code>test = diff(y_t, 2) + 3 * diff(y_t) + 2 * y_t - exp(-t)</code>	<code>test = 0</code>	Indeed, $y(t)$ satisfies the differential equation.
<code>t = 0;</code> <code>y0 = eval(y_t)</code> <code>yd0 = eval(diff(y_t))</code>	<code>y0 = 2</code> <code>yd0 = 3</code>	The functions $y(t)$ and $y'(t)$ are evaluated for $t = 0$, and the initial values are also satisfied.
<code>y0 = subs(y_t, 0)</code> <code>yd0 = subs(diff(y_t), 0)</code>	<code>y0 = 2</code> <code>yd0 = 3</code>	Alternative confirmation for the initial values.

9.9 Solved Problems

Problem 1 Compute the unilateral Laplace transform of the function $f(t) = -1.25 + 3.5te^{-2t} + 1.25e^{-2t}$. Confirm your answer by evaluating the inverse Laplace transform of the result.

Solution

```

syms s t
f=-1.25+3.5*t*exp(-2*t) +1.25*exp(-2*t);
F=laplace(f,s)

simplify(F)

pretty(ans)

f=ilaplace(F,t);
simplify(f);
pretty(ans)

```

The Laplace transform $F(s)$ of $f(t)$.

$$F = -\frac{5}{4}s + \frac{7}{2}/(s+2)^2 + \frac{5}{4}/(s+2)$$

Shortening $F(s)$.

$$\text{ans} = (s-5)/s/(s+2)^2$$

Writing $F(s)$ in a nice-looking way.

$$(s-5)/(s(s+2)^2)$$

The result is verified by computing the inverse Laplace transform.

$$-\frac{5}{4} + \frac{7}{2}t \exp(-2t) + \frac{5}{4} \exp(-2t)$$

Problem 2 Evaluate the Laplace transform of $z(t) = y(t)$ for a generic function $y(t)$. Next, replace $y(t)$ with $\sin(t)$ and compute $z(t)$ according to the appropriate initial conditions.

Solution

```

sym'y(t)';
syms t s
z=laplace( diff('y(t)',2),s)

z=subs(z,'y(t)',sin(t))

z=subs(z,'y(0)',0)

z=subs(z,'D(y)(0)',1)

simplify(z)

x=sin(t);
x2=diff(x,2,t)
laplace(x2,s)

```

Computation of $z(t) = x(t) = y(t)$.

$$z = s * (s * \text{laplace}(y(t), t, s) - y(0)) - D(y)(0)$$

Replacement of $y(t)$ by $\sin(t)$.

$$z = s * (s / (s^2 + 1) - y(0)) - D(y)(0)$$

Replacement of $y(0)$ by $\sin(0) = 0$.

$$z = s^2 / (s^2 + 1) - D(y)(0)$$

Replacement of $y'(0)$ by $\sin'(0) = \cos(0) = 1$.

$$z = s^2 / (s^2 + 1) - 1$$

$\text{ans} = -1 / (s^2 + 1)$
This is the expected result as $-\sin(\omega_0 t)u(t) \leftrightarrow -\omega_0 / (s^2 + \omega_0^2)$.

Confirmation of the obtained result in a straightforward way.

$$\text{ans} = -1 / (s^2 + 1)$$

Problem 3 Compute the inverse Laplace transform of the function $Y(s) = 1/s + 2/(s+4) + 1/(s+5)$.

Solution

```

syms s t
Y=1/s +2/(s+4)+1/(s+5) ;
y=ilaplace(Y,t)

```

The inverse Laplace Transform of $Y(s)$ is

$$y = 1 + 2 \exp(-4t) + \exp(-5t)$$

Problem 4 Express in partial fraction form the signal

$$X(s) = \frac{3s^2 + 2s + 1}{s^3 + s}.$$

Solution

<pre>num=[3 2 1]; den=[1 0 1 0]; [R,P,K]=residue(num,den); X=R(1)/(s-P(1))+R(2)/(s-P(2))+R(3)/(s-P(3)); pretty(X)</pre>	<pre>R= 1.0000 - 1.0000i 1.0000 + 1.0000i 1.0000 P= 0 + 1.0000i 0 - 1.0000i 0 K= [] X(s)= 1-j 1+j 1 s-j s+j s</pre>
---	---

Problem 5 Express in the partial fraction form the signal

$$X(s) = \frac{s^2 + 5s + 4}{s^4 + 1}.$$

Verify your result by computing the inverse Laplace transform from both forms (partial fraction and rational).

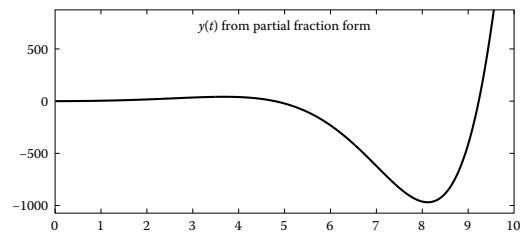
Solution

<pre>num=[1 5 4]; den=[1 0 0 0 1]; [R,P,K]=residue(num,den); K=[]</pre>	<pre>R= 0.5303 + 0.3661i 0.5303 - 0.3661i - 0.5303 - 2.1339i - 0.5303 + 2.1339i P= - 0.7071 + 0.7071i - 0.7071 - 0.7071i 0.7071 + 0.7071i 0.7071 - 0.7071i</pre>
---	--

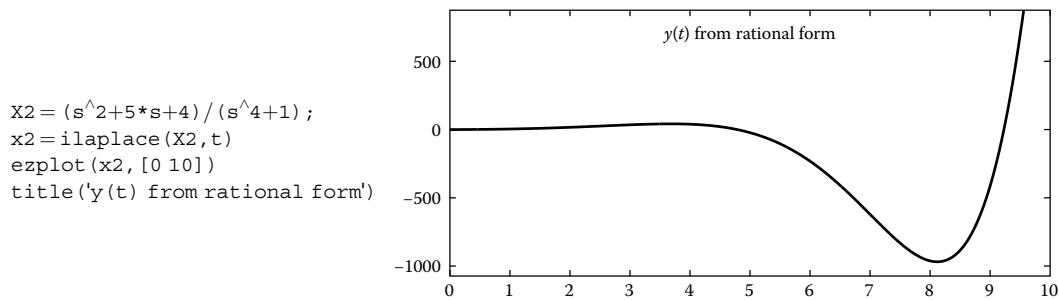
Hence, the signal $X(s)$ is written as

$$X(s) = \frac{0.5303 + 0.3661i}{s - (-0.7071 + 0.7071i)} + \frac{0.5303 - 0.3661i}{s - (-0.7071 - 0.7071i)} + \frac{-0.5303 - 2.1339i}{s - (0.7071 + 0.7071i)} + \frac{-0.5303 + 2.1339i}{s - (0.7071 - 0.7071i)}.$$

```
syms s
X1=R(1)/(s-P(1))+R(2)/(s-P(2))+R(3)/(s-P(3))+R(4)/(s-P(4));
x1=ilaplace(X1,t);
ezplot(x1,[0 10])
title('y(t) from partial fraction form')
```



(continued)

**Problem 6**

- a. Express in the partial fraction form the signal

$$X(s) = \frac{s^4 + 3s^3 - 8s^2 + 7s + 13}{s^2 + 2s + 1}.$$

Verify your result by using

- b. The reverse syntax of the command `residue`.
c. The command `numden`.

Solution

a. num = [1 3 -8 7 13];	R = 28 -4
den = [1 2 1];	P = -1 -1
[R, P, K] = residue (num, den)	K = 1 1 -11

There is a repeated root in the denominator polynomial. Moreover, the residue K is not null. Thus, signal $X(s)$ is expressed as $X(s) = \frac{28}{s+1} - \frac{4}{(s+1)^2} + s^2 + s - 11$.

b. `[num, den] = residue(R, P, K)`

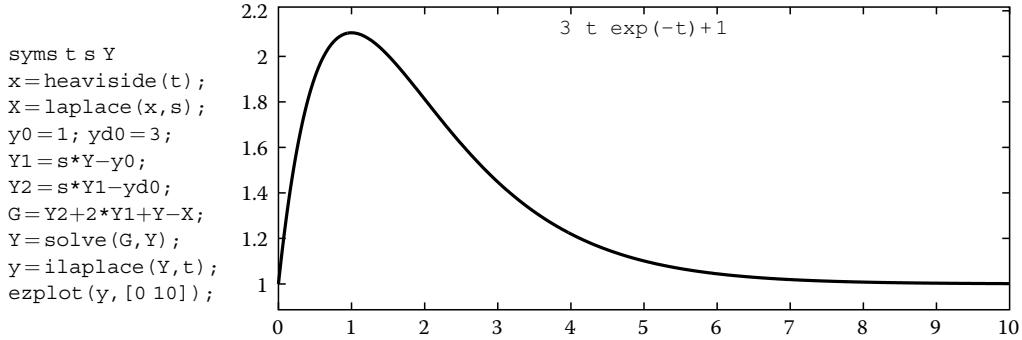
num = 1 3 -8 7 13
den = 1 2 1

c. `syms s`
 $X = R(1)/(s-P(1)) + R(2)/((s-P(2))^2) +$
 $K(1)*s^2 + K(2)*s + K(3);$
 $[n, d] = \text{numden}(X);$
 $X = n/d$

Problem 7 Use the Laplace transform to compute the solution of the differential equation $y''(t) + 2y'(t) = u(t) - y(t)$, $y(0) = 1$, $y'(0) = 3$.

Plot the solution in the time interval $0 \leq t \leq 10$.

Solution

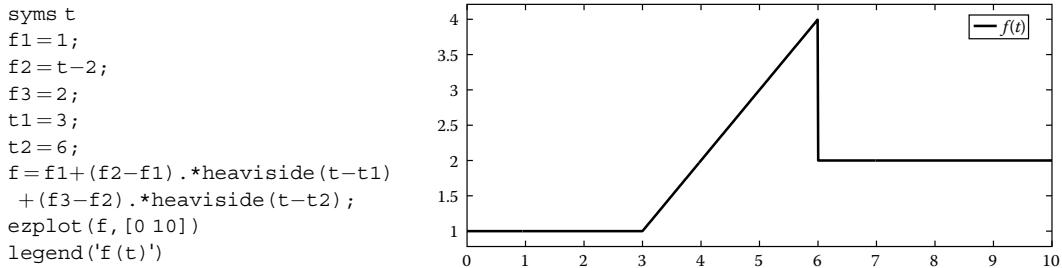


Problem 8 Use the Laplace transform to find the solution of the differential equation $y''(t) + 3y'(t) + 2y(t) = f(t)$, $y(0) = 1$, $y'(0) = 3$. The function $f(t)$ is given by

$$f(t) = \begin{cases} 1, & t < 3 \\ t-2 & 3 < t < 6 \\ 2, & t > 6 \end{cases}$$

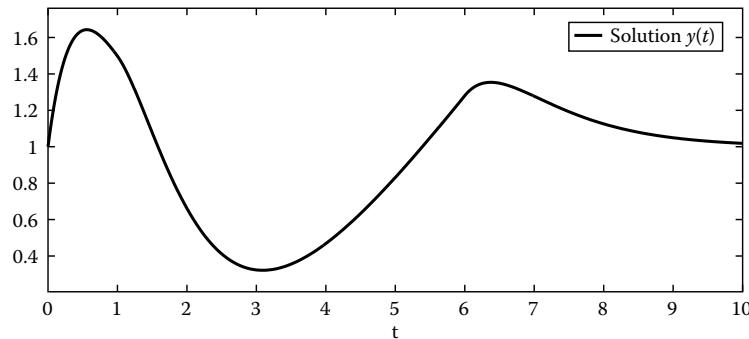
Solution

First, the function $f(t)$ is defined as a single symbolic expression. The piecewise function $f(t)$ is written as $f(t) = f_1(t) + [f_2(t) - f_1(t)]u(t - t_1) + [f_3(t) - f_2(t)]u(t - t_2)$, where $t_1 = 3$, $t_2 = 6$, $f_1 = 1$, $f_2 = t - 2$, and $f_3 = 2$. For confirmation $f(t)$ is plotted.



Next, the solution of the differential equation is obtained by following the usual computational procedure.

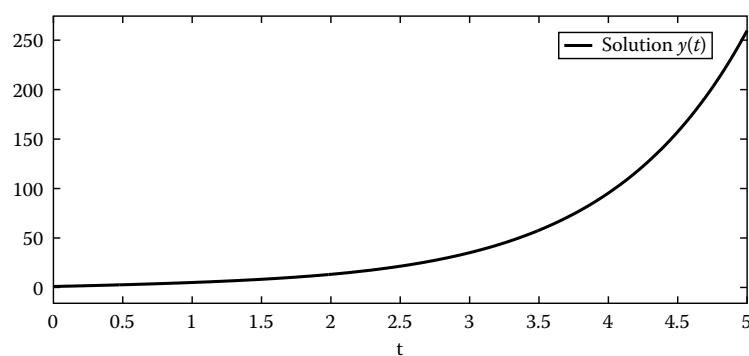
```
syms s Y
y0=1; yd0=3;
F=laplace(f,s);
Y1=s*Y-y0;
Y2=s*Y1-yd0;
G=Y2+3*Y1+2*Y-F;
Y=solve(G,Y);
y=ilaplace(Y,t);
ezplot(y,[0,10]);
legend('Solution y(t)')
```



Problem 9 Use the Laplace transform to find the solution of the differential equation $y'''(t) + y'(t) - 2y(t) = \delta(t)$, where $y(0) = 1$, $y'(0) = 3$, $y''(0) = 1$, and $\delta(t)$ is the Dirac function.

Solution

```
syms t s Y
y0=1; yd0=3; y2d0=1;
x=dirac(t);
X=laplace(x,s);
Y1=s*Y-y0;
Y2=s*Y1-yd0;
Y3=s*Y2-y2d0;
G=Y3+Y1-2*Y-X;
Y=solve(G,Y);
y=ilaplace(Y,t);
ezplot(y,[0 5]);
legend('Solution y(t)')
```



9.10 Homework Problems

1. Compute the unilateral Laplace transform of the signals
 - a. $x(t) = (e^{-3t} + e^{3t})$
 - b. $x(t) = (1 + e^{3t})/e^{-3t}$
 - c. $x(t) = 3 \cos(2t) + \sin(3\pi t)$
 - d. $x(t) = t \sin(t)$
 - e. $x(t) = t^4 e^t$
 - f. $x(t) = u(t) + u(t - 1) + u(t - 2)$
 - g. $x(t) = \delta(t) + \delta(t - 1)$
 - h. $x(t) = e^{-t} \cos(t)$

2. Compute the inverse Laplace transform of the signals

a. $X(s) = 1 + \frac{1}{s} + \frac{1}{s^2} + \frac{1}{s^3} + \frac{1}{s^4} + \frac{1}{s^5}$

b. $X(s) = \frac{1}{s - 4j} + \frac{2}{s - 4}$

c. $X(s) = \frac{1}{s + 4} + \frac{s}{s + 4}$

d. $X(s) = \frac{s}{s^2 + 4} - \frac{2}{s^2 + 4}$

3. The unit step function $u(t)$ is defined in MATLAB by the command `heaviside(t)`. Evaluate $u(t)$ for $t = 0$.

4. Verify using the Laplace transform that $\lim_{t \rightarrow \infty} \frac{\sin(t)}{t} = 0$.

5. Verify using the Laplace transform that $\lim_{t \rightarrow 0} \frac{\sin(t)}{t} = 1$.

6. Express in partial fraction form the signals

a. $X(s) = \frac{s^2 - s + 2}{s^3 - 2s^2 - 5s + 6}$

b. $X(s) = \frac{s^3 - 2s^2 - 5s + 6}{s^2 - s + 2}$

c. $X(s) = \frac{s^2 - 4s + 3}{s^4 + 7s^3 + 18s^2 + 20s + 8}$

d. $X(s) = \frac{s^4 + 7s^3 + 18s^2 + 20s + 8}{s^2 - 4s + 3}$

7. Express in rational form the signals

a. $X(s) = \frac{3}{s + 4} + \frac{2}{s - 1}$

b. $X(s) = \frac{1}{s - 1} + \frac{2}{(s - 1)^2} + \frac{3}{(s - 1)^3}$

c. $X(s) = \frac{3}{s + 4} + \frac{2}{s - 1} + s^2 + 2s + 6$

8. Compute the convolution between the signals $x(t) = 5e^{-t}u(t)$ and $h(t) = te^{-t}u(t)$.

9. Compute the convolution between the signals $X_1(s) = \frac{1}{(s+1)^2}$ and $X_2(s) = \frac{2}{(s+1)^3}$.

10. Find the solution of the differential equation $2y''(t) + y'(t) = te^{-t}$, $y(0) = 2$, $y'(0) = 1$.

11. Compute and plot the solution of the differential equation $y'''(t) - 5y''(t) + 8y'(t) - 4y(t) = 0$, $y(0) = 2$, $y'(0) = 1$, $y''(0) = -1$.

12. Compute and plot the solution of the differential equation $2y''(t) = y'(t) + x(t)$, $y(0) = 0$, $y'(0) = -1$, where $x(t) = \begin{cases} e^{0.1t}, & 0 < t < 1 \\ 1.1, & t > 1 \end{cases}$.

13. Verify that $L\left\{\frac{d^2y(t)}{dt^2}\right\} = s^2Y(s) - sy(0) - y'(0)$.

14. Confirm that $x'(0) = \lim_{s \rightarrow \infty} [s^2X(s) - sx(0)]$, where $x'(0)$ is the first derivative of $x(t)$ evaluated at $t = 0$.

15. Verify that $L\left\{\int_0^t x(r)dr\right\} = \frac{X(s)}{s}$.

10

z-Transform

In this chapter, we introduce the *z*-transform. *z*-transform is the counterpart of Laplace transform when dealing with discrete-time signals. It is employed to transform difference equations that describe the input/output relationships of discrete-time systems into algebraic equations and is a very useful tool for the analysis and design of discrete-time systems.

10.1 Mathematical Definition

As Laplace transform is a more general transform compared to the Fourier transform for continuous-time signals, *z*-transform is a more general transform than discrete-time Fourier transform when dealing with discrete-time signals. A discrete-time signal is defined in the discrete-time domain n ; that is, it is given by a function $f[n]$, $n \in \mathbb{Z}$. *z*-transform is denoted by the symbol $Z\{\cdot\}$ and expresses a signal in the *z*-domain, i.e., the signal is given by a function $F(z)$. The mathematical expression is

$$F(z) = Z\{f[n]\}. \quad (10.1)$$

In other words, the *z*-transform of a function $f[n]$ is a function $F(z)$. The mathematical expression of the two-sided (or bilateral) *z*-transform is

$$F(z) = Z\{f[n]\} = \sum_{n=-\infty}^{\infty} f[n]z^{-n}, \quad (10.2)$$

where z is a complex variable. Setting the lower limit of the sum from minus infinity to zero yields the one-sided (or unilateral) *z*-transform whose mathematical expression is

$$F(z) = Z\{f[n]\} = \sum_{n=0}^{\infty} f[n]z^{-n}. \quad (10.3)$$

In order to return from the *z*-domain back to the discrete-time domain, the inverse *z*-transform is applied. The inverse *z*-transform is denoted by the symbol $Z^{-1}\{\cdot\}$; that is, one can write

$$f[n] = Z^{-1}\{F(z)\}. \quad (10.4)$$

The mathematical expression of the inverse z -transform is

$$x[n] = \frac{1}{2\pi j} \oint X(z) z^{n-1} dz. \quad (10.5)$$

The z -transform of a sequence is easily computed by Equations 10.2 or 10.3.

Example

Compute the z -transform of the sequence $x[n] = [3, 5, 4, 3]$, $0 \leq n \leq 3$.

The z -transform of the sequence $x[n]$ is directly computed from Equation 10.3.

Commands	Results	Comments
<pre>syms z x0 = 3; x1 = 5; x2 = 4; x3 = 3; Xz = x0*(z^0) + x1*(z^-1) + x2*(z^-2) + x3*(z^-3) pretty(Xz)</pre>	$3 + \frac{5}{z} + \frac{4}{z^2} + \frac{3}{z^3}$	z -Transform of the sequence $x[n]$.

An alternative and more elegant computation of the z -transform of $x[n]$ is

Commands	Results	Comments
<pre>syms z x = [3 5 4 3]; n = [0 1 2 3]; X = sum(x.* (z.^-n)) pretty(X)</pre>	$3 + \frac{5}{z} + \frac{4}{z^2} + \frac{3}{z^3}$	Alternative way of computing the z -transform of the sequence $x[n]$.

Finally, if $x[n]$ is a sequence of infinite samples, we must use the command `symsum` to compute its z -transform.

Example

Compute the z -transform of the sequence $x[n] = 0.9^n u[n]$.

Commands	Results	Comments
<pre>syms n z x = 0.9^n; X = symsum(x.* (z.^-n), n, 0, inf)</pre>	$X = 10*z / (10*z - 9)$	The z -transform of the sequence $x[n] = 0.9^n u[n]$ is $X(z) = \frac{z}{z - 0.9}$.

10.2 Commands `ztrans` and `iztrans`

In MATLAB®, the z -transform $F(z)$ of a sequence $f[n]$ is computed easily by using the command `ztrans`. Moreover, the inverse z -transform of a function $F(z)$ is computed by

using the command `iztrans`. Before using these two commands, the declaration of the complex variable z and of the discrete time n as symbolic variables is necessary. Recall that in order to define a symbolic variable, the command `syms` is used. The commands `ztrans` and `iztrans` are used exactly in the same way as the commands `laplace` and `ilaplace` are used to compute the Laplace and inverse Laplace transforms of a function. Finally, we note that the command `ztrans` computes the unilateral z-transform.

Example

Compute the (unilateral) z-transform of the sequence $f[n] = 2^n$.

Commands	Results	Comments
<code>syms n z f = 2^n; ztrans(f) simplify(ans)</code>	<code>ans = z / (z - 2)</code>	The z-transform of the sequence $f[n] = 2^n$.

Example

Compute the inverse z-transform of the function $F(z) = z/(z - 2)$.

Commands	Results	Comments
<code>syms n z F = z / (z - 2); iztrans(F)</code>	<code>ans = 2^n</code>	Inverse z-transform of the function $F(z) = \frac{z}{z - 2}$.

The functions $f[n] = 2^n u[n]$ and $F(z) = z/(z - 2)$ are a z-transform pair. In other words, the z-transform of $f[n] = 2^n u[n]$ is $F(z) = z/(z - 2)$, while the inverse z-transform of $F(z) = z/(z - 2)$ is the sequence $f[n] = 2^n u[n]$. A z-transform pair is denoted as $x[n] \leftrightarrow X(z)$. In our case $2^n u[n] \leftrightarrow z/(z - 2)$. In Section 10.4, we present the most common z-transform pairs. At the moment, we discuss alternative syntaxes of the commands `ztrans` and `iztrans`. The most effective, and in the same time simple, syntax of the command `ztrans` is `ztrans(f, z)`. In this way, the z-transform of a sequence $f[n]$ is expressed with the second input argument (here is z) as the independent variable. Using this syntax is optimal and makes possible the computation of the z-transform in every case. Such a case is when the z-transform of a constant function has to be computed.

Commands	Results	Comments
<code>syms n z f = 1; ztrans(f)</code>	<code>??? Function 'ztrans' is not defined for values of class 'double'.</code>	When the simple syntax of the command <code>ztrans</code> is used, MATLAB cannot compute the z-transform of the constant sequence $f[n] = 1$.
<code>ztrans(f, z)</code>	<code>ans = z / (z - 1)</code>	Using the optimal syntax of <code>ztrans</code> the z-transform of $f[n] = 1$ is computed. The result is $F(z) = Z\{1\} = z/(z - 1)$.

The z-transform of a function can be expressed in terms of another variable (e.g., w).

Commands	Results	Comments
<code>syms w f = n^2 ztrans(f, w)</code>	<code>ans = w*(w+1)/(w-1)^3</code>	The z-transform of $f[n] = n^2$ expressed with symbol w as the independent variable.
<code>ztrans(f)</code>	<code>ans = z*(z+1)/(z-1)^3</code>	By default the z-transform is expressed with z as the independent variable.

Of course, if a symbolic variable is once declared (and not erased using the command `clear`), it does not have to be declared every time it is used. Regarding the inverse z-transform, the optimal syntax of the command `iztrans` is `iztrans(F, n)`. In this way, the inverse z-transform of the signal $F(z)$ is expressed in terms of the variable that appears as the second input argument (here is n). Correspondingly, with the `ztrans` command, there is the possibility of expressing the inverse z-transform of a function $F(z)$ with alternative variable (e.g., t) as independent variable.

Commands	Results	Comments
<code>syms n z t F = z/(z-3); f = iztrans(F, t)</code>	<code>f = 3^t</code>	The inverse z-transform of the signal $F(z) = z/(z - 3)$ expressed with t as independent variable.

Finally, an alternative available syntax for the z-transform computation is `ztrans(f, n, z)`; that is, the function f is transferred from n to z , while the inverse z-transform is computed by using the command `iztrans(F, z, n)`; that is, the function F is transferred from z to n .

10.3 Region of Convergence

The z-transform of a sequence does not always exist. The region of convergence (ROC) of the z-transform of a sequence $x[n]$ is the range of z for which the z-transform of $x[n]$ is not infinite. For example, the z-transform $X(z)$ of the sequence $x[n] = 0.8^n u[n]$ is computed as $X(z) = \sum_{n=0}^{\infty} 0.8^n z^{-n} = \sum_{n=0}^{\infty} (0.8/z)^n$. Thus $X(z)$ converges; that is, $X(z) < \infty$ if $|0.8/z| < 1 \Rightarrow |z| > 0.8$. Therefore, the ROC of the z-transform $X(z)$ of the signal $x[n] = 0.8^n u[n]$ is $|z| > 0.8$.

10.4 z-Transform Pairs

In this section, the most common z-transform pairs are presented. In the general case, the z-transform or the inverse z-transform of a function cannot be easily computed directly from Equations 10.2, 10.3, and 10.5. This is the reason that already computed z-transform pairs are used to compute the z-transform or the inverse z-transform of a complicated function. Thus, the computational procedure is to express a complicated function of

interest in terms of functions with known z (or inverse z) transform and then based on the properties of z -transform to compute the z (or inverse z) transform of the complicated function. In the table below, the most common z -transform pairs are given. The illustrated pairs are confirmed by using the commands `ztrans` and `iztrans`. It should be noted that since the command `ztrans` computes the one-sided z -transform ($n \geq 0$), the unit step sequence $u[n]$ can be omitted from the signal definition.

Discrete-Time Domain	z -Domain	Commands	Results
$x[n]$	$X(z)$	<code>syms n z a w</code>	
$\delta[n]$	1	<code>f=dirac(n);</code> <code>ztrans(f,z)</code>	<code>ans=dirac(0)</code> <code>% $\delta(0) = 1$.</code>
$u[n]$	$z/(z-1)$	<code>f=heaviside(n)</code> <code>ztrans(f,z)</code>	<code>ans=z/(z-1)</code>
$n \cdot u[n]$	$z/(z-1)^2$	<code>ztrans(n,z)</code>	<code>ans=z/(z-1)^2</code>
$a^n u[n]$	$z/(z-a)$	<code>F=z/(z-a);</code> <code>f=iztrans(F,n)</code>	<code>f=a^n</code>
$n a^n u[n]$	$\frac{az}{(z-a)^2}$	<code>f=n*a^n;</code> <code>ztrans(f,z)</code>	<code>ans=z*a/(-z+a)^2</code>
$\cos(\omega_0 n)u[n]$	$\frac{z^2 - z \cos(\omega_0)}{z^2 - 2z \cos(\omega_0) + 1}$	<code>f=cos(w*n)</code> <code>ztrans(f,z)</code>	<code>ans=(z-cos(w))*z/(z^2-2*z*cos(w)+1)</code>
$\sin(\omega_0 n)u[n]$	$\frac{z \sin(\omega_0)}{z^2 - 2z \cos(\omega_0) + 1}$	<code>f=sin(w*n);</code> <code>ztrans(f,z)</code>	<code>ans=z*sin(w)/(z^2-2*z*cos(w)+1)</code>
$a^n \cos(\omega_0 n)u[n]$	$\frac{z^2 - az \cos(\omega_0)}{z^2 - 2az \cos(\omega_0) + a^2}$	<code>f=(a^n)*cos(w*n)</code> <code>ztrans(f,z);</code> <code>simplify(ans)</code>	<code>ans=-(-z+cos(w)*a)*z/(z^2-2*z*cos(w)*a+a^2)</code>
$a^n \sin(\omega_0 n)u[n]$	$\frac{az \sin(\omega_0)}{z^2 - 2az \cos(\omega_0) + a^2}$	<code>f=(a^n)*sin(w*n)</code> <code>ztrans(f,z);</code> <code>simplify(ans)</code>	<code>ans=z*sin(w)*a/(z^2-2*z*cos(w)*a+a^2)</code>

10.5 Properties of z -Transform

In this section, the main properties of the z -transform are introduced and verified through appropriate examples.

1. *Linearity:* If $X_1(z) = Z\{x_1[n]\}$ and $X_2(z) = Z\{x_2[n]\}$, then for any scalars a_1, a_2 ,

$$Z\{a_1 x_1[n] + a_2 x_2[n]\} = a_1 X_1(z) + a_2 X_2(z). \quad (10.6)$$

Commands	Results	Comments
<pre>syms n z x1 = n^2; x2 = 2^n; a1 = 3; a2 = 4; Le = a1*x1+a2*x2; Left = ztrans(Le,z)</pre>	$\text{Left} = \frac{3z(z+1)}{(z-1)^3 + 2z}$	To verify Equation 10.6, we consider $a_1=3$, $a_2=4$, and the sequences $x_1[n]=n^2u[n]$, $x_2[n]=2^n u[n]$. The left side is computed first.
<pre>X1 = ztrans(x1); X2 = ztrans(x2); Right = a1*X1+a2*X2</pre>	$\text{Right} = \frac{3z(z+1)}{(z-1)^3 + 2z}$	Computation of the right side of (10.6). The two sides are equal; hence, the linearity property of z-transform is verified.

2. Right shift of $x[n]u[n]$: If $X(z) = Z\{x[n]\}$ and $x[n]$ is causal, i.e., $x[n] = 0$, $n < 0$, then for any positive integer m ,

$$Z\{x[n-m]u[n-m]\} = z^{-m}X(z). \quad (10.7)$$

Commands	Results	Comments
<pre>m = 2 x = 3^(n-m) * heaviside(n-m); Left = ztrans(x,z) Left = simplify(Left)</pre>	$\text{Left} = \frac{1}{z/(z-3)}$	The right-shifting property is validated using the discrete-time signal $x[n]=3^n u[n]$ and the scalar $m=2$. First, we compute the left side of (10.7).
<pre>x = 3^n * heaviside(n); X = ztrans(x,z); Right = (z^(-m)) * X Right = simplify(Right)</pre>	$\text{Right} = \frac{1}{z/(z-3)}$	The right side of (10.7) is computed and is equal to the left one; hence, the right-shifting property is confirmed.

3. Right shift of $x[n]$: If $X(z) = Z\{x[n]\}$, then for any positive integer m ,

$$Z\{x[n-1]\} = z^{-1}X(z) + x[-1] \quad (10.8)$$

$$Z\{x[n-2]\} = z^{-2}X(z) + x[-2] + z^{-1}x[-1] \quad (10.9)$$

⋮

$$Z\{x[n-m]\} = z^{-m}X(z) + x[-m] + z^{-1}x[-m+1] + \cdots + z^{-m+1}x[-1]. \quad (10.10)$$

Commands	Results	Comments
<pre>n = -3:3; x = 0.8.^n; xminus3 = x(1) xminus2 = x(2) xminus1 = x(3)</pre>	$\begin{aligned} \text{xminus3} &= 1.9531 \\ \text{xminus2} &= 1.5625 \\ \text{xminus1} &= 1.2500 \end{aligned}$	To confirm the right-shifting property, the signal $x[n]=0.8^n$, $-3 \leq n \leq 3$ is first defined in order to calculate the values $x[-1]$, $x[-2]$, and $x[-3]$.

(continued)

Commands	Results	Comments
<pre>syms n z xn1=0.8^(n-1); Left=ztrans(xn1,z); simplify(Left)</pre>	$ans = 25/4*z/(5*z-4)$	First, we verify (10.8). The result that appears in the middle tab is the left side of (10.8).
<pre>x=0.8^n; X=ztrans(x,z); Right=z^-1*X+xminus1; simplify(Right)</pre>	$ans = 25/4*z/(5*z-4)$	The right side of (10.8) is computed, and since it is equal to the left one, (10.8) is verified.
<pre>xn2=0.8^(n-2); Left=ztrans(xn2,z); simplify(Left)</pre>	$ans = 125/16*z/(5*z-4)$	The left side of (10.9).
$Right = z^{-2}X + x_{\text{minus}2} + z^{-1}x_{\text{minus}1};$ <pre>simplify(Right)</pre>	$ans = 125/16*z/(5*z-4)$	The right side of (10.9) is equal to the left one; hence, the right-shifting property of z-transform is demonstrated.

4. *Left shift in time:* If $X(z) = Z\{x[n]\}$, then for any positive integer m ,

$$Z\{x[n+1]\} = zX(z) - x[0]z \quad (10.11)$$

$$Z\{x[n+2]\} = z^2X(z) - x[0]z^2 - x[1]z \quad (10.12)$$

 \vdots

$$Z\{x[n+m]\} = z^mX(z) - x[0]z^m - x[1]z^{m-1} - \dots - x[m-1]z. \quad (10.13)$$

Commands	Results	Comments
$x_0 = 0.8^0$ $x_1 = 0.8^1$	$x_0 = 1$ $x_1 = 0.8000$	To confirm the left-shifting property the signal $x[n] = 0.8^n u[n]$ is again considered. First, $x[0]$ and $x[1]$ are evaluated.
<pre>syms n z; xn1=0.8^(n+1); Left=ztrans(xn1,z); simplify(Left)</pre>	$ans = 4*z/(5*z-4)$	Initially, we verify Equation 10.11. The result that appears at the middle tab is the left side of (10.11).
<pre>x=0.8^n; X=ztrans(x,z); Right=z*X - x0*z; simplify(Right)</pre>	$ans = 4*z/(5*z-4)$	The right side of (10.11) is computed and it is equal to the left one; hence, (10.11) is verified.
$xn2=0.8^{(n+2)}$ $Left=ztrans(xn2,z);$ <pre>simplify(Left)</pre>	$ans = 16/5*z/(5*z-4)$	The left side of (10.12).
$Right = z^2X - x_0z^2 - x_1z;$ <pre>simplify(Right)</pre>	$ans = 16/5*z/(5*z-4)$	The right side of (10.12) is equal to the left one, hence the left-shifting property of z-transform is demonstrated.

5. *Scaling in the z-domain:* If $X(z) = Z\{x[n]\}$, then for any scalar a ,

$$Z\{a^{-n}x[n]\} = X(az) \quad (10.14)$$

and

$$Z\{a^n x[n]\} = X\left(\frac{z}{a}\right). \quad (10.15)$$

An important special case is when $a = e^{j\omega_0}$, for which (10.15) becomes

$$Z\{e^{j\omega_0 n} x[n]\} = X(e^{-j\omega_0} z). \quad (10.16)$$

Commands	Results	Comments
<pre>syms n z x=4^n; X=ztrans(x,z); X=simplify(X)</pre>	$X = z / (z - 4)$	Equations 10.14 and 10.15 are verified by using the signal $x[n] = 4^n u[n]$.
<pre>syms a Right=subs(X,z,a*z)</pre>	$\text{Right} = a * z / (a * z - 4)$	In order to derive the right side of (10.14), z in $X(z)$ is substituted by $a \cdot z$.
<pre>L=a^(-n)*x; Left=ztrans(L,z); simplify(Left)</pre>	$\text{ans} = a * z / (a * z - 4)$	The left side of (10.14) is equal to the right one, hence Equation 10.14 is confirmed.
<pre>Right=subs(X,z,z/a); simplify(Right)</pre>	$\text{ans} = -z / (-z + 4 * a)$	In order to compute the right side of (10.15), z in $X(z)$ is substituted by z/a .
<pre>L=a^n*x; Left=ztrans(L,z); simplify(Left)</pre>	$\text{ans} = -z / (-z + 4 * a)$	The left side of (10.15) is equal to the right one, hence Equation 10.15 is also confirmed.
<pre>syms w0 Right=subs(X,z,exp(-j*w0)*z)</pre>	$\text{Right} = \exp(-i * w0) * z / (\exp(-i * w0) * z - 4)$	The right side of (10.16).
<pre>L=exp(j*w0*n)*x; Left=ztrans(L,z); simplify(Left-Right)</pre>	$\text{ans} = 0$	The left side is computed, and since the difference of the two sides is zero (10.16) is confirmed.

6. *Time reversal:* If $X(z) = Z\{x[n]\}$, then

$$Z\{x[-n]\} = X(z^{-1}). \quad (10.17)$$

Commands	Results	Comments
<pre>syms z x=[1 2 3 4]; n=[0 1 2 3]; X=sum(x.* (z.^(-n))); Right=subs(X,z,z^-1)</pre>	$\text{Right} = 1 + 2 * z + 3 * z^2 + 4 * z^3$	The signal $x[n] = [1, 2, 3, 4]$, $0 \leq n \leq 3$ is considered. The right side is computed by substituting z with z^{-1} in $X(z)$.

(continued)

Commands	Results	Comments
<code>nrev = [-3, -2, -1, 0]; xrev = [4, 3, 2, 1]; Left = sum(xrev.* (z.^-nrev))</code>	$\text{Left} = 1 + 2z + 3z^2 + 4z^3$	To compute the left side of (10.17), first the time $-n$ and the signal $x[-n]$ are defined. The z-transform of $x[-n]$, which is the left side of (10.17) is equal to the right side; thus the time-reversal property is confirmed.

7. *Differentiation in the z-domain:* If $X(z) = Z\{x[n]\}$, then

$$Z\{n \cdot x[n]\} = -z \frac{dX(z)}{dz}. \quad (10.18)$$

Commands	Results	Comments
<code>syms n z x = 0.9^n; Left = ztrans(n*x, z)</code>	$\text{Left} = 90z / (10z - 9)^2$	Equation 10.18 is confirmed by using the discrete-time signal $x[n] = 0.9^n u[n]$. First, we compute the left side.
<code>X = ztrans(x, z); d = diff(X, z); Right = -z*d simplify(Right)</code>	$\text{ans} = 90z / (10z - 9)^2$	The right side of (10.18) equals the left one, hence the differentiation in the z-domain property is verified.

8. *Summation:* If $X(z) = Z\{x[n]\}$ and $x[n]$ is a causal discrete-time signal, then

$$Z\left\{\sum_{i=0}^n x[i]\right\} = \frac{z}{z-1} X(z). \quad (10.19)$$

Commands	Results	Comments
<code>syms n z x = n^2; s = symsum(x, n, 0, n); Left = ztrans(s, z); Left = simplify(Left)</code>	$\text{Left} = z^2 * (z+1) / (z-1)^4$	The signal $x[n] = n^2 u[n]$ is considered. First, we compute the left side of (10.19). The sum $\sum_{i=0}^n x[i]$ is computed using the command <code>symsum</code> .
<code>X = ztrans(x, z); Right = (z/(z-1))*X</code>	$\text{Right} = z^2 * (z+1) / (z-1)^4$	The right side of (10.19) is equal to the left one, hence the summation property is validated.

9. *Convolution in the time domain:* If $X_1(z) = Z\{x_1[n]\}$ and $X_2(z) = Z\{x_2[n]\}$, then

$$Z\{x_1[n] * x_2[n]\} = X_1(z)X_2(z). \quad (10.20)$$

Applying inverse z -transform to both sides of (10.20) yields

$$x_1[n] * x_2[n] = Z^{-1}\{X_1(z)X_2(z)\}. \quad (10.21)$$

To verify Equation 10.20 it is enough to verify (10.21).

Commands	Results	Comments
<pre>n=0:50; x1=0.9.^n; x2=0.8.^n; y=conv(x1,x2); stem(0:100,y) legend('Convolution');</pre>		The left side of (10.21) is computed for the discrete-time signals $x_1[n] = 0.9^n$, $0 \leq n \leq 50$ and $x_2[n] = 0.8^n$, $0 \leq n \leq 50$. The result of their convolution is plotted.
<pre>syms n z x1=0.9.^n; x2=0.8.^n; X1=ztrans(x1,z); X2=ztrans(x2,z); Right=iztrans(X1*X2); n=0:100; Right=subs(Right,n); stem(0:100,Right) legend('Z^-1[X_1(z) X_2(z)]');</pre>		Next, we compute and plot the right side of (10.21). The two graphs are identical; hence, the convolution property of the z -transform is confirmed.

10. *Complex conjugation:* If $X(z) = Z\{x[n]\}$ and $x^*[n]$ is the complex conjugate of $x[n]$, then

$$Z\{x^*[n]\} = X^*(z^*). \quad (10.22)$$

Commands	Results	Comments
<pre>syms n z x=(2+3*i)^n; xc=conj(x); Left=ztrans(xc)</pre>	$\text{Left} = z / (-2 + 3i + z)$	Equation 10.22 is verified using the signal $x[n] = (2 + 3i)^n u[n]$. First, the left side of Equation 10.22 is computed.
<pre>x=(2+3*i)^n; X=ztrans(x,z); Xc=conj(X); Right=subs(Xc,z,conj(z))</pre>	$\text{Right} = (2 + 3i) * z / (2z + 3i * z - 13)$	The right side is derived by first computing the conjugate of $X(z)$ and then substituting z by z^* .
<pre>simplify(Right-Left)</pre>	$\text{ans} = 0$	The difference of the two sides is zero, so property (10.22) is confirmed.

11. *Initial value theorem:* If $X(z) = Z\{x[n]\}$, then the initial values of $x[n]$ can be computed according to the following relationships:

$$\begin{aligned} x[0] &= \lim_{z \rightarrow \infty} X(z) \\ x[1] &= \lim_{z \rightarrow \infty} [zX(z) - zx[0]] \\ &\vdots \\ x[m] &= \lim_{z \rightarrow \infty} [z^m X(z) - z^m x[0] - z^{m-1} x[1] - \cdots - zx[m-1]] \end{aligned} \quad (10.23)$$

Commands	Results	Comments
<pre>syms n z x = (n+1)^2; X = ztrans(x, z); x0 = limit(X, z, inf) x1 = limit(z*X-z*x0, z, inf) x2 = limit((z^2)*x-(z^2)*x0-z*x1, z, inf)</pre>	<pre>x0 = 1 x1 = 4 x2 = 9</pre>	<p>Suppose that $x[n] = (n+1)^2 u[n]$. Thus, $x[0] = 1$, $x[1] = 4$, $x(2) = 9$, etc.</p> <p>The initial values $x[0]$, $x[1]$, $x[2]$ are computed according to the relationships given in (10.23), and the validity of (10.23) is demonstrated.</p>

12. *Final value theorem:* If $X(z) = Z\{x[n]\}$ and the limit of $x[n]$ when n tends to infinity exists, then

$$\lim_{n \rightarrow \infty} x[n] = \lim_{z \rightarrow 1} [(z-1)X(z)] = \lim_{z \rightarrow 1} [(1-z^{-1})X(z)]. \quad (10.24)$$

Commands	Results	Comments
<pre>syms n z x = 0.8^n; limit(x, n, inf)</pre>	ans = 0	The limit when $n \rightarrow \infty$ is computed for the sequence $x[n] = 0.8^n u[n]$ (left side of (10.24)).
<pre>X = ztrans(x, z); limit((z-1)*x, z, 1)</pre>	ans = 0	The middle part of (10.24) is computed and is equal to the left one.
<pre>limit((1-z^-1)*x, z, 1)</pre>	ans = 0	The right side of (10.24) is computed and is also equal to the left one. Hence, the final value theorem is verified.

10.6 Partial Fraction Expansion of a Rational Function

The z-transform of a sequence is usually expressed as a rational function of z , i.e., it is written as a ratio of two polynomials of z . The mathematical expression is

$$X(z) = \frac{B(z)}{A(z)} = \frac{b_m z^m + b_{m-1} z^{m-1} + \cdots + b_1 z + b_0}{a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + a_0}, \quad (10.25)$$

where a_i, b_i are real numbers. In order to express the function $X(z)$ in a partial fraction form, we follow a similar approach to that followed in the Laplace transform chapter. Hence, the first case considered is when $m < n$; that is, when the degree of the numerator polynomial $B(z)$ is lower than the degree of the denominator polynomial $A(z)$. Suppose that λ_i are the roots of $A(z)$. The following cases are considered:

1. *The roots λ_i are distinct*; that is, every root appears only once. In this case, the denominator is factored as $A(z) = a_n \prod_{i=1}^n z - \lambda_i$ and the signal $X(z)$ is written as

$$X(z) = \frac{c_1}{z - \lambda_1} + \frac{c_2}{z - \lambda_2} + \cdots + \frac{c_n}{z - \lambda_n}, \quad (10.26)$$

where the coefficients c_1, \dots, c_n are computed according to

$$c_i = \lim_{z \rightarrow \lambda_i} [(z - \lambda_i)X(z)]. \quad (10.27)$$

Example

Express in the partial fraction form the signal which in the z -domain is given by $X(z) = \frac{z^2 + 3z + 1}{z^3 + 5z^2 + 2z - 8}$.

Commands	Results	Comments
<code>A = [1 5 2 -8]; ro = roots(A)</code>	<code>ro = -4.0000 -2.0000 1.0000</code>	The vector containing the coefficients of the denominator polynomial is defined and its roots are computed.
<code>syms z X = (z^2+3*z+1)/(z^3+5*z^2+2*z-8); c1 = limit((z-ro(1))*X, z, ro(1)) c2 = limit((z-ro(2))*X, z, ro(2)) c3 = limit((z-ro(3))*X, z, ro(3))</code>	<code>c1 = 1/2 c2 = 1/6 c3 = 1/3</code>	The coefficients c_i are calculated according to Equation 10.27.

Hence, we obtain

$$X(z) = \frac{z^2 + 3z + 1}{z^3 + 5z^2 + 2z - 8} = \frac{1/2}{z + 4} + \frac{1/6}{z + 2} + \frac{1/3}{z - 1}.$$

2. *The roots λ_i are repeated*. Suppose that the root λ_1 is repeated r times and all other roots are distinct. In this case, $A(z)$ is written in the factored form $A(z) = a_n(z - \lambda_1)^r \prod_{i=r+1}^n z - \lambda_i$, while $X(s)$ is written as

$$X(z) = \frac{c_1}{z - \lambda_1} + \frac{c_2}{(z - \lambda_1)^2} + \cdots + \frac{c_r}{(z - \lambda_1)^r} + \frac{c_{r+1}}{z - \lambda_{r+1}} + \cdots + \frac{c_n}{z - \lambda_n}. \quad (10.28)$$

The coefficients c_1, \dots, c_n are given by

$$\begin{aligned} c_i &= \lim_{z \rightarrow \lambda_i} \frac{1}{(r-i)!} \frac{d^{r-i}[(z - \lambda_1)^r X(z)]}{dz^{r-i}}, \quad i = 1, \dots, r. \\ c_i &= \lim_{z \rightarrow \lambda_i} (z - \lambda_i) X(z), \quad i = r+1, \dots, n \end{aligned} \quad (10.29)$$

Example

Express in partial fraction expansion form the signal

$$X(z) = \frac{z^2 + 3z + 1}{z^3 - 3z + 2}.$$

Commands	Results	Comments
<code>A = [1 0 -3 2]; rt = roots(A)</code>	<code>rt = -2.0000 1.0000 1.0000</code>	First, the roots of the denominator are calculated. The root $\lambda = 1$ is repeated two times. Notice that the coefficient of z^2 is zero and must be taken into account when formulating matrix A .
<code>syms z X = (z^2+3*z+1)/(z^3-3*z+2); c1 = limit((z-rt(1))*X,z,rt(1))</code>	<code>c1 = -1/9</code>	The coefficient c_1 is computed according to the lower part of (10.29).
<code>r = 2</code>		In order to compute the coefficients c_2 and c_3 (that correspond to $i=1$ and $i=2$, respectively), we set $r=2$ as there are two repeated roots. First, we compute coefficient c_2 .
<code>f = ((z-1)^r)*X; di = diff(f,z,r-1); fact = 1/factorial(r-1); c2 = limit(fact*di,z,1)</code>	<code>c2 = 10/9</code>	Calculation of $(z - \lambda_1)^r X(z)$. Calculation of $\frac{d^{r-i}[(z - \lambda_1)^r X(z)]}{dz^{r-i}}$. Calculation of $\frac{1}{(r-i)!}$. Notice that $i=1$ when calculating c_2 .
<code>di = diff(f,z,r-2); fact = 1/factorial(r-2); c3 = limit(fact*di,z,1)</code>	<code>c3 = 5/3</code>	Coefficient c_2 is computed according to the upper part of (10.29). Coefficient c_3 is computed in the same way to c_2 , but here we set $i=2$.

Therefore,

$$X(z) = \frac{z^2 + 3z + 1}{z^3 - 3z + 2} = \frac{-1/9}{z + 2} + \frac{10/9}{z - 1} + \frac{5/3}{(z - 1)^2}.$$

10.6.1 Commands `residue` and `residuez`

As illustrated in Chapter 9, a signal in rational form can be expressed in partial fraction form by using the command `residue`. The command `residue` is used to confirm the result of the previous example.

Example

Express in partial fraction form the signal

$$X(z) = \frac{z^2 + 3z + 1}{z^3 - 3z + 2}.$$

Commands	Results	Comments
<code>num = [1 3 1]; den = [1 0 -3 2]</code>		The coefficients of numerator and denominator polynomials are defined as usual.
<code>[R, P, K] = residue(num, den)</code>	$R = -0.1111 \quad 1.1111$ $P = 1.6667 \quad 1.0000$ $K = []$	$X(z)$ is expressed in partial fraction form as $X(z) = \frac{-0.1111}{z+2} + \frac{1.1111}{z-1} + \frac{1.6667}{(z-1)^2}$, which is the same result as the one obtained by the analytical way.

Let us consider now the case where $m \geq n$, i.e., the degree of the numerator polynomial $B(z)$ is greater than or equal to the degree of the denominator polynomial $A(z)$. In a world without MATLAB, we would have to implement the division between $B(z)$ and $A(z)$, write the signal $X(z)$ in the form $K(z) + (G(z)/A(z))$, and then apply the relationships (10.26) through (10.29) to expand the signal as a sum of first-order rational functions. Fortunately, the command `residue` is applicable also in the case $m \geq n$.

Example

Express in partial fraction form the signal

$$X(z) = \frac{3z^3 + 8z^2 + 4}{z^2 + 5z + 4}.$$

Commands	Results	Comments
<code>n = [3 8 0 4]; d = [1 5 4]; [R, P, K] = residue(n, d)</code>	$R = 20 \quad 3$ $P = -4 \quad -1$ $K = 3 \quad -7$	The partial fraction expansion of $X(z)$ is $X(z) = \frac{20}{z+4} + \frac{3}{z+1} + 3z - 7$.

The obtained result is confirmed by using the reverse syntax of the `residue` command.

Commands	Results	Comments
<code>R = [20 3]; P = [-4 -1]; K = [3 -7]; [B, A] = residue(R, P, K)</code>	$B = 3 \quad 8 \quad 0 \quad 4$ $A = 1 \quad 5 \quad 4$	The rational function $X(z) = \frac{B(z)}{A(z)}$ is $X(z) = \frac{3z^3 + 8z^2 + 4}{z^2 + 5z + 4}$.

An alternative command (especially suitable for the z -transform) that can be used to expand a rational function into partial fraction form is the command `residuez`. The syntax is `[r, p, k] = residuez(num, den)`, where `num` and `den` denote the coefficients

of the numerator and denominator polynomials, respectively, in ascending powers of z^{-1} ; that is, $X(z)$ is of the form

$$X(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}. \quad (10.30)$$

The outcome of the command `residuez` is analyzed according to

$$X(z) = \frac{r_1}{1 - p_1 z^{-1}} + \frac{r_2}{1 - p_2 z^{-1}} + \cdots + \frac{r_n}{1 - p_n z^{-1}} + K(z), \quad (10.31)$$

where $K(z) = k_0 + k_1 z^{-1} + \cdots$. If there is a root p_1 of multiplicity q , the outcome of the command `residuez` is analyzed according to

$$\begin{aligned} X(z) = & \frac{r_1}{1 - p_1 z^{-1}} + \frac{r_2}{(1 - p_1 z^{-1})^2} + \cdots + \frac{r_q}{(1 - p_1 z^{-1})^q} + \frac{r_{q+1}}{1 - p_{q+1} z^{-1}} + \cdots \\ & + \frac{r_n}{1 - p_n z^{-1}} + K(z). \end{aligned} \quad (10.32)$$

The inverse syntax of the `residuez` command, i.e., `[num,den] = residuez(r,p,k)` does the inverse operation, that is, expresses a function from partial fraction form to rational form.

Example

Express in partial fraction form the signal

$$X(z) = \frac{1 - 8z^{-1} + 17z^{-2} + 2z^{-3} - 24z^{-4}}{1 + z^{-1} - 2z^{-2}}.$$

Commands	Results	Comments
<code>n = [1 -8 17 2 -24] ;</code> <code>d = [1 1 -2] ;</code> <code>[R, P, K] = residuez(n, d)</code>	$R = 5 \quad -4$ $P = -2 \quad 1$ $K = 0 \quad 5 \quad 12$	The partial fraction expansion of $X(z)$ is $X(z) = \frac{5}{1 + 2z^{-1}} + \frac{4}{1 - z^{-1}} + 5z^{-1} + 12z^{-2}.$

10.7 Using the z-Transform to Solve Difference Equations

In this section, we introduce a method for solving linear difference equations with constant coefficients by using the z -transform. For simplicity, we assume that the signals are causal, i.e., are zero for $n \leq 0$. In Section 10.8, an example of a difference equation with nonzero initial conditions is given. The basic property that is used to find the solution of a difference equation by employing the z -transform is the shifting property that for causal signals simplifies to

$$Z\{x[n - m]\} = z^{-m} X(z). \quad (10.33)$$

The general form of a difference equation is

$$y[n] = \sum_{k=0}^q b_k x[n-k] + \sum_{k=1}^p a_k y[n-k], \quad (10.34)$$

where b_k and a_k are constant numbers. Recall that the solution of a difference equation is a sequence $y[n]$ that satisfies the difference equation for any n . The computational process followed is similar to the one followed when solving differential equations by use of the Laplace transform. Hence, the steps followed are

1. The z -transform is applied to both sides of the difference equation.
2. Due to the linearity property, the z -transform of a sum equals the sum of the z -transforms of the sum terms. Moreover, the scalars b_k and a_k are intergraded out of the z -transforms.
3. The z -transforms of the shifted signals $x[n-k]$, $y[n-k]$ are evaluated according to (10.33).
4. The algebraic equation that comes up is solved for $Y(z)$.
5. The inverse z -transform of $Y(z)$ is calculated; that is, the sequence $y[n]$ is computed. The sequence $y[n]$ is the solution of the difference equation.

Example

Find the solution of the difference equation $y[n] + 0.5y[n-1] + 2y[n-2] = 0.9^n u[n]$, where $y[n] = 0$, $n < 0$.

From (10.33) we get

$$Z\{y[n]\} = Y(z). \quad (10.35)$$

$$Z\{y[n-1]\} = z^{-1}Z\{y[n]\} = z^{-1}Y(z). \quad (10.36)$$

$$Z\{y[n-2]\} = z^{-2}Z\{y[n]\} = z^{-2}Y(z). \quad (10.37)$$

The computational procedure is as follows.

1. Applying z -transform to both parts of the differential equation yields $Z\{y[n] + 0.5y[n-1] + 2y[n-2]\} = Z\{0.9^n u[n]\}$.
2. Due to the linearity property, we get

$$Z\{y[n]\} + 0.5Z\{y[n-1]\} + 2Z\{y[n-2]\} = \frac{z}{z - 0.9},$$

where the z -transform pair $a^n u[n] \leftrightarrow \frac{z}{z - a}$ is used.

3. The z -transforms of $y[n]$, $y[n - 1]$, and $y[n - 2]$ are substituted according to Equations 10.35 through 10.37, and the difference equation is converted to the algebraic equation

$$Y(z) = 0.5z^{-1}Y(z) + 2z^{-2}Y(z) = \frac{z}{z - 0.9}.$$

4. We solve the equation for $Y(z)$ and get

$$Y(z) = \frac{z}{(z - 0.9)(1 + 0.5z^{-1} + 2z^{-2})} = \frac{z^3}{(z - 0.9)(z^2 + 0.5z + 2)}.$$

5. The solution $y[n]$ of the difference equation is obtained by applying inverse z -transform to $Y(z)$, i.e., $y[n] = Z^{-1}\{Y(z)\}$.

A similar computational procedure is implemented in MATLAB in order to compute the solution of the given difference equation.

Commands	Results	Comments
<code>syms n z Y</code>		$Y(z)$ is denoted by Y .
<code>X=ztrans(0.9^n,z)</code>		The z -transform of the right side of the difference equation is computed.
<code>Y1=z^(-1)*Y;</code>		The z -transform of $y[n - 1]$, that is, $Z\{y[n - 1]\}$ is defined according to (10.36). The result is assigned to variable $Y1$.
<code>Y2=z^(-2)*Y;</code>		$Z\{y[n - 2]\}$ is denoted by $Y2$ and is defined according to (10.37).
This is the crucial point of the computational procedure. The term X is moved to the left side of the difference equation and the whole left side is assigned to a term G		
<code>G=Y+0.5*Y1+2*Y2-X;</code>		The term G is a function of Y and z .
<code>SOL=solve(G,Y);</code>	$\frac{z^3}{(z - 0.9)(z^2 + 0.5z + 2)}$	Using the command <code>solve</code> allows us to solve for Y . This is the solution of the differential equation expressed in the z -domain. The obtained solution is same as the analytically computed solution.
<code>pretty(SOL);</code>		
<code>y=iztrans(SOL,n);</code>		Applying inverse z -transform yields the solution $y[n]$ of the difference equation.

(continued)

(continued)

Commands	Results	Comments
<pre>n1=0:10; y_n=subs(y,n,n1); stem(n1,y_n) legend('y[n]') xlim([- .5 10.5])</pre>		The sequence $y[n]$ is defined as a symbolic expression; thus in order to implement its graph, the symbolic variable n is substituted by a vector.

In order to confirm that $y[n]$ is in fact the solution of the difference equation, $y[n]$ is inserted in the difference equation. If it satisfies the difference equation, then it is indeed its solution.

Commands	Results	Comments
<pre>yn1 = subs(y,n,n-1); yn2 = subs(y,n,n-2); test=y+0.5*yn1+2* yn2-0.9^n; test=simplify(test)</pre>	<pre>test = 0</pre>	The terms $y[n-1]$ and $y[n-2]$ are computed from the derived sequence $y[n]$. The obtained sequence $y[n]$ is indeed the solution of the difference equation $y[n] + 0.5y[n-1] + 2y[n-2] = 0.9^n$.

10.8 Solved Problems

Problem 1

- Compute the z -transform of the sequence $f_1[n] = [-3, 5, 6, 7, 8]$, $-2 \leq n \leq 2$.
- Compute the z -transform of the sequence $f_2[n] = [-3, 5, 6, 7, 8]$, $0 \leq n \leq 4$.

Solution

a. <pre>f = [-3, 5, 6, 7, 8]; n = -2:2; syms z F = sum(f.* (z.^-n)); pretty(F)</pre>	The z -transform $F_1(z)$ of the sequence $f_1[n] = [-3, 5, 6, 7, 8]$, $-2 \leq n \leq 2$ is $-3z^2 + 5z + 6 + 7z^{-1} + 8z^{-2}$.
b. <pre>n = 0:4; F = sum(f.* (z.^-n)); pretty(F)</pre>	The z -transform $F_2(z)$ of the sequence $f_2[n] = [-3, 5, 6, 7, 8]$, $0 \leq n \leq 4$ is $-3 + 5z^{-1} + 6z^{-2} + 7z^{-3} + 8z^{-4}$.

Problem 2

- Compute the z -transform of the discrete-time signal $x[n] = n^2 u[n]$.
- Confirm your result by computing the inverse z -transform of your outcome.

Solution

```
a. syms n z
x=n^2*heaviside(n);           X=z*(z+1)/(z-1)^3
X = ztrans(x,z)

b. iztrans(X,n)                ans=n^2
```

Problem 3

- Compute the z -transform of the discrete-time signal $x[n] = \cos(2\pi n)u[n]$.
- Confirm your answer by computing the inverse z -transform of your outcome.

Solution

```
a. syms n z
x=cos(2*pi*n);           X=z/(z-1)
X = ztrans(x,z)

b. iztrans(X,n)            ans=1
```

With a first glance, the result is not confirmed by the inverse z -transform operation. However, considering that n takes only integer values yields $\cos(2\pi n) = 1, \forall n \in \mathbb{Z}$, and the result is confirmed.

Problem 4 Find the inverse z -transform of $X(z) = \frac{2z + 3}{z^2 + 5z + 6}$

- When $X(z)$ is in rational form
- When $X(z)$ is in partial fraction form

Solution

num = [2 3] den = [1 5 6]; [R, P, K] = residue(num, den)	R = 3.0000 -1.0000 P = -3.0000 -2.0000 K = []
--	---

The partial fraction form of $X(z)$ is $X(z) = \frac{3}{z+3} - \frac{1}{z+2}$.

```
a. syms n z
X = (2*z+3)/(z^2+5*z+6);
iztrans(X,n)                                ans = 1/2*charfcn[0](n)+1/2*
                                              (-2)^n-(-3)^n

b. X = 3/(z+3)-1/(z+2)
iztrans(X,n)                                ans = 1/2*charfcn[0](n)+1/2*
                                              (-2)^n-(-3)^n
```

The derived result is same in both cases. However, an unknown function `charfcn` appears in our results. To learn more about this function we type `mhelp charfcn`, and the help text displayed in the command window states that `charfcn [A] (x)` is 1 if x belongs to set A and 0 if not. In our case, the statement `charfcn [0] (n)` is 1 for $n = 0$ and 0 elsewhere.

Problem 5 Compute the z-transform of the discrete-time signal

$$f[n] = \begin{cases} 0.9^n, & 0 \leq n \leq 3 \\ 2^{-n}, & 4 \leq n \leq 6 \\ 1, & 7 \leq n \leq 10 \end{cases}$$

- a. With use of the command `ztrans`.
- b. Without using the command `ztrans`.
- c. Confirm your result by computing the inverse z-transform of your outcome.

Solution

First, the signal $f[n]$ is defined as a single symbolic expression. The three-part function $f[n]$ is written as

$$f[n] = f_1[n](u[n] - u[n - n_1]) + f_2[n](u[n - n_1] - u[n - n_2]) + f_3[n](u[n - n_2] - u[n - n_3]),$$

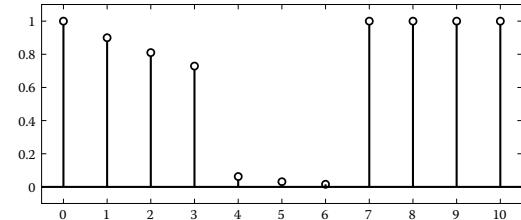
where $f_1 = 0.9^n$, $f_2 = 2^{-n}$, $f_3 = 1$, $n_1 = 4$, $n_2 = 7$, and $n_3 = 11$. For confirmation, $f[n]$ is plotted according to the technique of plotting multipart functions and according to the derived single symbolic expression.

```

n1=0:3;
f1=0.9.^n1;
n2=4:6;
f2=2.^(-n2);
n3=7:10;
f3=ones(size(n3));
n=[n1 n2 n3];
f=[f1 f2 f3];
stem(n,f);
axis([- .5 10.5 -.1 1.1]);

```

Graph of $f[n]$ using the technique of plotting multipart functions.

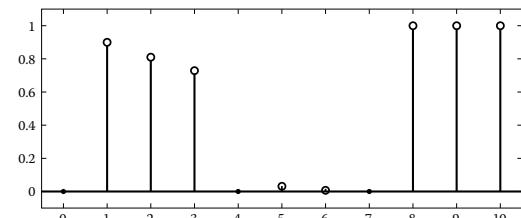


```

syms n z
n1=4;
n2=7;
n3=11;
f1=0.9^n;
f2=2.^(-n);
f3=1;
f=f1*(heaviside(n)-heaviside(n-n1))
+f2*(heaviside(n-n1)-heaviside(n-n2))
+f3*(heaviside(n-n2)-heaviside(n-n3));
n_s=0:10;
f_s=subs(f,n_s);
stem(n_s,f_s);
axis([- .5 10.5 -.1 1.1]);

```

Graph of $f[n]$ according to its symbolic expression. The graphs are alike except from the points $n=0,4,7$ where $f[n]$ has no value. This is due to the way the command `heaviside` is defined in MATLAB. However, this fact does not affect the z-transform computation.



(continued)

a. $F_1 = \text{ztrans}(f, z)$

$$F_1 = 1 + 9/10/z + 81/100/z^2 + 729/1000/z^3 + 1/16/z^4 + 1/32/z^5 + 1/64/z^6 + 1/z^7 + 1/z^8 + 1/z^9 + 1/z^{10}$$

The signal is defined as a three-part function and its z -transform is computed according to its definition.

```
b. n1 = 0:3;
f1 = 0.9.^n1;
n2 = 4:6;
f2 = 2.^(-n2);
n3 = 7:10;
f3 = ones(size(n3));
n = [n1 n2 n3];
f = [f1 f2 f3];
F2 = sum(f.*(z.^-n))
```

$$F_2 = 1 + 9/10/z + 81/100/z^2 + 729/1000/z^3 + 1/16/z^4 + 1/32/z^5 + 1/64/z^6 + 1/z^7 + 1/z^8 + 1/z^9 + 1/z^{10}$$

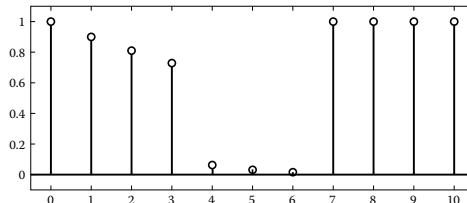
The two derived results are the same.

```
c. syms n
ftest = iztrans(F1, n)
```

$$\begin{aligned} f_{\text{test}} = & \text{charfcn}[0](n) + 9/10*\text{charfcn}[1](n) + \\ & 81/100*\text{charfcn}[2](n) + 729/1000*\text{charfcn}[3](n) + 1/16*\text{charfcn}[4](n) + \\ & 1/32*\text{charfcn}[5](n) + 1/64*\text{charfcn}[6](n) + \text{charfcn}[7](n) + \\ & \text{charfcn}[8](n) + \text{charfcn}[9](n) + \text{charfcn}[10](n) \end{aligned}$$

Variable f_{test} is the inverse z -transform of the computed function $F(z)$. Hence, we expect that f_{test} is equal to the original signal $f[n]$.

```
n = 0:10;
ftest1 = subs(ftest, n);
stem(n, ftest1);
axis([-0.5 10.5 -0.1 1.1]);
```



Indeed, the graph of f_{test} is identical to that of $f[n]$; hence, the computation of the z -transform $F(z)$ of $f[n]$ is correct.

Problem 6

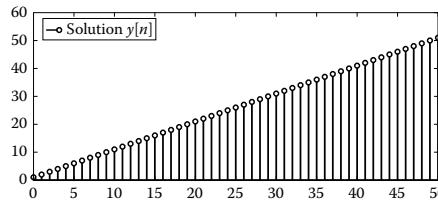
- Using z -transform find the solution of the difference equation $y[n] - y[n-1] = u[n]$.
- Plot the solution for $0 \leq n \leq 50$.
- Confirm your result by inserting the obtained solution in the difference equation.

Solution

```
a. syms n z Y
x=heaviside(n);
X=ztrans(x,z);
Y1=z^(-1)*Y;
G=Y-Y1-X;
SOL=solve(G,Y);
y=iztrans(SOL,n)
```

$y = 1+n$ The solution $y[n]$

```
b. n1=0:50;
yn=subs(y,n,n1);
stem(n1,yn);
legend('Solution y[n]');
```



```
c. yntest=y;
yn_1test=subs(y,n,n-1);
test=1-heaviside(n)
test=yntest-yn_1test-x
```

The result is zero for
 $n \geq 0$.

Problem 7

- Use z-transform to find the solution of the difference equation $y[n] - y[n - 1] = x[n] + x[n - 1]$, where $x[n] = 0.8^n u[n]$.
- Plot the solution for $0 \leq n \leq 20$.
- Confirm your result by inserting the obtained solution in the difference equation.

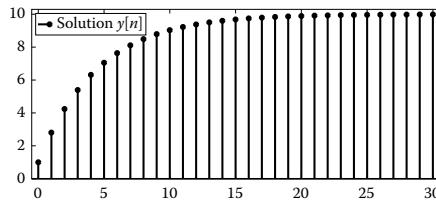
Solution

```
a. syms n z Y
x = 0.8^n;
X = ztrans(x, z);
X1 = z^(-1)*X;
Y1 = z^(-1)*Y;
G = Y - Y1 - X - X1;
SOL = solve(G, Y);
y = iztrans(SOL, n)
```

The solution $y[n]$ is
 $y = 10 - 9 * (4/5)^n$

Notice how the z-transform of $x[n-1]$ (denoted by $X1$) is defined.

```
b. n_s = 0:30;
y_s = subs(y, n, n_s);
stem(n_s, y_s);
legend('Solution y[n]');
```



```
c. xn = 0.8^(n);
xn_1 = .8^(n-1);
yn = 10 - 9 * (4/5)^n;
yn_1 = 10 - 9 * (4/5)^(n-1);
test = yn - yn_1 - xn - xn_1;
simplify(test)
```

ans = 0

Inserting the obtained solution in the difference equation yields zero; thus $y[n]$ is indeed the solution of the difference equation.

Problem 8 Use z-transform to find the solution of the difference equation $y[n] + 1.5y[n-1] + 0.5y[n-2] = x[n] + x[n-1]$, where $x[n] = 0.8^n u[n]$.

- Plot the solution for $0 \leq n \leq 20$.
- Confirm your result by inserting the obtained solution in the difference equation.

Solution

```
syms n z Y
x = 0.8^n;
X = ztrans(x, z);
X1 = z^(-1)*X;
Y1 = z^(-1)*Y;
Y2 = z^(-2)*Y;
G = Y + 1.5*Y1 + 0.5*Y2 - X - X1;
SOL = solve(G, Y);
y = iztrans(SOL, n)
```

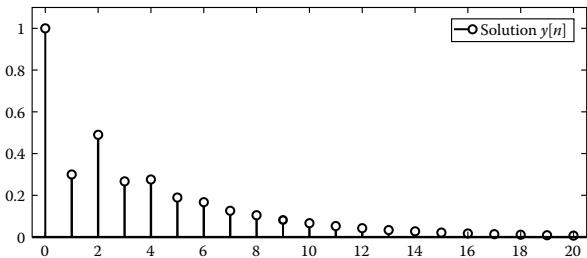
$$y = \frac{5}{13} \cdot (-\frac{1}{2})^n + \frac{8}{13} \cdot (\frac{4}{5})^n$$

The solution $y[n]$ of the difference equation.

(continued)

(continued)

```
a. n_s = 0:20;
y_s = subs(y,n,n_s);
stem(n_s,y_s);
legend('Solution y[n]');
xlim([-5 20.5])
ylim([0 1.1])
```



```
b. xn=x;
xn_1=0.8^(n-1);
yn=y;
yn_1=subs(y,n,n-1);
yn_2=subs(y,n,n-2);
test=yn+1.5*yn_1+0.5*yn_2-xn-xn_1
simplify(test)
```

ans = 0

Inserting the obtained solution in the difference equation yields zero; thus $y[n]$ is indeed the solution of the difference equation.

Problem 9 Compute and plot the solution of the difference equation $y[n] - 3y[n - 1] + y[n - 2] = x[n] - x[n - 1]$, where $x[n] = 0.9^n u[n]$ and the initial conditions are $y[-1] = -1$, $y[-2] = -2$. Moreover, verify your answer

- By examining if the derived solution satisfies the difference equation
- By computing the solution with use of the command *filter*

Solution

In this difference equation, we have nonzero initial conditions for $y[n]$. Thus, the z-transform of $y[n - 1]$ and $y[n - 2]$ are computed according to the property given in (10.8) and (10.9). Therefore, the z-transforms of $y[n]$, $y[n - 1]$, $y[n - 2]$, $x[n]$, and $x[n - 1]$ are given by

$$\begin{aligned} Z\{y[n]\} &= Y(z) \\ Z\{y[n - 1]\} &= z^{-1}Y(z) + y[-1] \\ Z\{y[n - 2]\} &= z^{-2}Y(z) + y[-2] + z^{-1}y[-1] \\ Z\{x[n]\} &= X(z) \\ Z\{x[n - 1]\} &= z^{-1}X(z) \end{aligned}$$

```

syms n z Y
x = 0.9^n;
X = ztrans(x, z);
X1 = z^(-1)*X;
Y_1 = -1;
Y_2 = -2
Y1 = z^(-1)*Y + Y_1;
Y2 = z^(-2)*Y + Y_2 + (z^(-1))*Y_1;
G = 2*Y - 3*Y1 + Y2 - X + X1;
SOL = solve(G, Y);
y = iztrans(SOL, n)

```

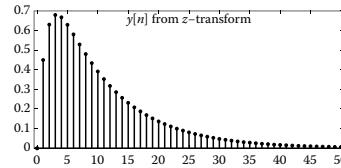
$$y = -\frac{9}{8} \left(\frac{1}{2}\right)^n + \frac{9}{8} \left(\frac{9}{10}\right)^n$$

The initial conditions of $y[n]$ are taken into account at the definition of the z -transforms of $y[n-1]$ and $y[n-2]$, and the solution of the difference equation with nonzero initial conditions is obtained

```

n1 = 0:50;
y_n = subs(y, n, n1);
stem(n1, y_n)
title('y[n] from z-transform');

```



Graph of the solution $y[n]$

```

a. xn = 0.9^(n);
xn_1 = .9^(n-1);
yn = -9/8*(1/2)^n + 9/8*(9/10)^n;
yn_1 = -9/8*(1/2)^(n-1) + 9/8*(9/10)^(n-1); ans = 0
yn_2 = -9/8*(1/2)^(n-2) + 9/8*(9/10)^(n-2);
test = 2*yn - 3*yn_1 + yn_2 - xn + xn_1;
simplify(test)

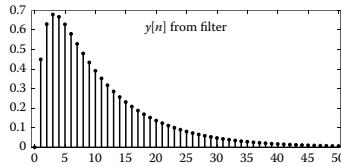
```

The obtained sequence $y[n]$ satisfies the difference equation

```

b. a = [2 -3 1];
b = [1 -1];
yit = [-1 -2];
zi =filtic(b,a,yit)
n = 0:50;
x = 0.9.^n;
y = filter(b,a,x,zi);
stem(n,y);
title('y[n] from filter');

```



The solution is derived with use of the command `filter`. Notice how the initial conditions are defined with the command `filtic`.

10.9 Homework Problems

1. Compute the z -transform of the signal $u[n] + nu[n] + n^2 u[n]$.
2. Compute the z -transforms of the signals $u[n]$ and $u[-n-1]$. Determine the ROC for each case.
3. Confirm the z -transform pair $\delta[n-k] \leftrightarrow \frac{1}{z^k}$.
4. Verify the z -transform pair $u[n] - u[n-k] \leftrightarrow \frac{z^k - 1}{z^k - z^{k-1}}$.

5. Confirm the z -transform pair $(n+1)u[n] \leftrightarrow \frac{z^2}{(z-1)^2}$.
6. Verify the z -transform pair $n^2u[n] \leftrightarrow \frac{z^2+z}{(z-1)^3}$.
7. Verify that $Z\{x[n]\cos(\omega_0 n)\} = \frac{1}{2}(X(e^{j\omega_0} z) + X(e^{-j\omega_0} z))$.
8. Confirm that $Z\{x[n]\sin(\omega_0 n)\} = \frac{j}{2}(X(e^{j\omega_0} z) - X(e^{-j\omega_0} z))$.
9. Verify that $Z\{n^2x[n]\} = z \frac{dX(z)}{dz} + z^2 \frac{d^2X(z)}{dz^2}$.
10. Suppose that $x[n]$ is a complex-valued sequence. Verify that
 - a. $\text{Re}\{x[n]\} = \frac{1}{2}(X(z) + X^*(z))$
 - b. $\text{Im}\{x[n]\} = \frac{1}{2j}(X(z) - X^*(z))$
11. Confirm that $Z\{x[n] - x[n-1]\} = (1 - z^{-1})X(z)$.
12. Verify that

$$Z\left\{\sum_{k=0}^n x[k]\right\} = \frac{1}{1 - z^{-1}} X(z).$$

13. Express in partial fraction form the signal

$$X(z) = \frac{2z^2 + z - 1}{z^3 - 3z + 2}.$$

Verify your answer by expressing the derived function in rational form.

14. Express in partial fraction form the signal

$$X(z) = \frac{z^3 - 3z + 2}{2z^2 + z - 1}.$$

Verify your answer by expressing the derived function in rational form.

15. Express in partial fraction form the signal

$$X(z) = \frac{5 - 11z^{-1}}{1 - 5z^{-1} + 6z^{-2}}.$$

Verify your answer by expressing the derived function in rational form.

16. Express in partial fraction form the signal

$$X(z) = \frac{12 - 38z^{-1} + 11z^{-2} + 3z^{-3} + 54z^{-4}}{1 - 5z^{-1} + 6z^{-2}}.$$

Verify your answer by expressing the derived function in rational form.

17. Compute and plot the solution of the difference equation $y[n] + y[n - 1] = 2x[n] + x[n - 1]$, where $x[n] = 0.8^n u[n]$ assuming zero initial conditions. Moreover, verify your answer (a) by examining if the derived solution satisfies the difference equation and (b) by computing the solution with use of the command `filter`.
18. Compute and plot the solution of the difference equation $2y[n] - 0.5y[n - 2] = 2x[n] - x[n - 2]$, where $x[n] = 0.9^n u[n]$ and the initial conditions are $y[-1] = 3$ and $y[-2] = 2$.

This page intentionally left blank

11

Transfer Function

In this chapter, we introduce the concept of transfer function for both continuous- and discrete-time systems. The transfer function is a valuable tool that simplifies the study and analysis of linear time-invariant (LTI) systems.

11.1 Continuous-Time Systems

An LTI system is completely described by a linear differential equation with constant coefficients, i.e., by an n th-order differential equation of the form

$$a_n \frac{d^n y(t)}{dt^n} + \cdots + a_1 \frac{dy(t)}{dt} + a_0 y(t) = b_m \frac{d^m x(t)}{dt^m} + \cdots + b_1 \frac{dx(t)}{dt} + b_0 x(t). \quad (11.1)$$

The coefficients a_i, b_j are constants. Moreover, suppose that the initial conditions are zero. Applying Laplace transform to both sides of (11.1) yields

$$a_n s^n Y(s) + a_{n-1} s^{n-1} Y(s) + \cdots + a_0 Y(s) = b_m s^m X(s) + b_{m-1} s^{m-1} X(s) + \cdots + b_0 X(s). \quad (11.2)$$

Solving for $Y(s)/X(s)$ yields

$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_0}. \quad (11.3)$$

The function $H(s)$ is called *transfer function* and is a complete description of an LTI system. Thus, the transfer function of an LTI system is defined as the ratio of the Laplace transform $Y(s)$ of the output signal $y(t)$ to the Laplace transform $X(s)$ of the input signal $x(t)$ that is applied to the system, supposing zero initial conditions. An alternative definition of transfer function is now presented. The response of a system in the time domain is computed by the convolution between the applied to the system input signal $x(t)$ and the impulse response $h(t)$ of the system, namely, is

$$y(t) = x(t) * h(t). \quad (11.4)$$

Applying Laplace transform to both sides of (11.4) and considering the convolution property of Laplace transform yields

$$Y(s) = X(s)H(s) \quad (11.5)$$

and

$$H(s) = \frac{Y(s)}{X(s)}, \quad (11.6)$$

where $H(s) = L\{h(t)\}$. Therefore, an alternative definition for the transfer function of a system is that *transfer function is the Laplace transform of the impulse response of the system*.

Example

Compute the transfer function $H(s)$ of a system described by the impulse response $h(t) = te^{-t} u(t)$.

Commands	Results	Comments
<pre>syms t s u=heaviside(t); h=t*exp(-t)*u; H=laplace(h,s)</pre>	$H = 1/(s+1)^2$	The transfer function $H(s) = 1/(s+1)^2$ is the Laplace transform of the impulse response $h(t) = te^{-t}u(t)$.

Example

Compute the transfer function of a system described by the differential equation $y''(t) + 3y'(t) - 2y(t) = x''(t) - x'(t) - 6x(t)$, $y(0) = y'(0) = x(0) = x'(0) = 0$.

First, the Laplace transform is applied to both sides of the differential equation. Next, we solve for $Y(s)$, and according to (11.6) the system transfer function $H(s)$ is obtained by dividing $Y(s)$ with $X(s)$.

Commands	Results/Comments
<code>syms t s X Y</code>	The time t , the complex frequency s , and the Laplace transforms of the input signal X and the output signal Y are defined as symbolic variables.
<code>Y1=s*Y;</code> <code>Y2=s*Y1;</code> <code>X1=s*X;</code> <code>X2=s*X1;</code>	The Laplace transforms of $y''(t)$, $y(t)$, $x''(t)$, $x(t)$ are declared as $Y2$, $Y1$, $X2$, $X1$, respectively, and defined according to the Laplace transform differentiation property, taking into account that the initial conditions are zero.
<code>G=Y2+3*Y1-2*Y-X2+X1+6*X;</code> <code>Y=solve(G,Y);</code>	We define a polynomial G that consists of both sides of the differential equation. Next, we solve for Y to obtain $Y(s)$.
<code>H=Y/X</code>	The system transfer function $H(s)$ is derived by the operation $Y(s)/X(s)$. The transfer function is $H = (-s-6+s^2)/(-2+s^2+3*s)$

The transfer function $H(s)$ specifies completely the system as it encompasses all the information about the coefficients and the order of the differential equation that describes the system. In other words, if the system transfer function is known, the differential equation describing the input/output (i/o) relationship of the system can be derived and vice versa.

Example

An LTI system is described by the transfer function

$$H(s) = \frac{s^2 - s - 6}{s^2 + 3s - 2}.$$

Notice that it is the transfer function obtained in the previous example. Write down the i/o differential equation that describes the system.

Commands	Results/Comments
<code>syms x x1 x2 y y1 y2</code>	The input and output signals $x(t)$, $y(t)$, and their first and second derivatives $x'(t)$, $x''(t)$, $y'(t)$, $y''(t)$ are declared as the symbolic variables x , y , $x1$, $x2$, $y1$, $y2$, respectively.
<code>B = [1 -1 -6];</code> <code>A = [1 3 -2];</code>	The coefficients of the numerator and denominator polynomials are defined as the vectors B and A , correspondingly.
<code>g = A(1)*y2 + A(2)*y1 + A(3)*y - B(1)*x2 - B(2)*x1 - B(3)*x</code>	The elements of B and A are also the coefficients of the i/o differential equation. Hence, the differential equation that describes the system is $g = y_2 + 3y_1 - 2y - x_2 + x_1 + 6x$ which for $g=0$ is the same as that of the previous example.

11.2 The `tf` Command

A transfer function is usually given in rational form, namely, is written as a ratio of two polynomials.

$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_0}{a_n s^n + a_{n-1} s^{n-1} + \dots + a_0}. \quad (11.7)$$

In order to define a transfer function in MATLAB® we can use the command `tf`. The syntax is `H = tf (num, den)`, where *num* and *den* are vectors containing the polynomial coefficients of the numerator and denominator, respectively. The outcome *H* is a special type of MATLAB variable named *TF object*, and represents the transfer function of a system.

Example

Create the transfer function $H(s) = s^2/(s^2 + 3s + 1)$.

Commands	Results	Comments
<code>num = [1 0 0];</code> <code>den = [1 3 1];</code> <code>Hs = tf (num, den)</code>	Transfer function: $\frac{s^2}{s^2 + 3s + 1}$	Only the vectors of the numerator and denominator coefficients are required to declare a TF object. The result represents a system transfer function.

A nice feature of the `tf` command is that a time delay can be applied to the system specified from a transfer function by using the syntax `H = tf (num, den, 'inputdelay', m)`, where m is the needed delay.

Example

A system is described by the transfer function

$$H(s) = \frac{s+2}{s^2 + 3s + 1}.$$

Modify appropriately the transfer function to cause a time delay of 2 units to the system response.

Commands	Results
<pre>num = [1 2]; den = [1 3 1]; H = tf (num, den, 'inputdelay', 2)</pre>	<p>Transfer function:</p> $\exp(-2 * s) * \frac{s+2}{s^2 + 3s + 1}.$

The transfer function that causes a 2-units delay to the system is the system transfer function multiplied by the factor e^{-2s} . This relationship is based on the Laplace transform time-shifting property and becomes clear in the next example.

Example

Compute and plot the impulse responses $h_1(t)$ and $h_2(t)$ of the systems with transfer functions $H_1(s) = s/(s^2 + 4)$ and $H_2(s) = e^{-3s} s/(s^2 + 4)$, respectively.

Commands	Results	Comments
<pre>syms t s H1 = s/(s^2+4); h1 = ilaplace(H1,t); ezplot(h1, [0 20]); grid; legend('h_1(t)')</pre>		<p>The impulse response $h_1(t) = \cos(2t)$ (plotted for $0 \leq t \leq 20$) is obtained from the inverse Laplace transform of the transfer function $H_1(s) = s/(s^2 + 4)$.</p>
<pre>H2 = exp(-3*s) * (s/(s^2+4)); h2 = ilaplace(H2,t) ezplot(h2, [0 20]); grid; legend('h_2(t)')</pre>		<p>The impulse response $h_2(t) = \cos(2t - 6)u(t - 3)$ is obtained from the inverse Laplace transform of the transfer function $H_2(s) = e^{-3s} s/(s^2 + 4)$.</p>

It is obvious that the impulse response $h_2(t)$, which corresponds to the transfer function $H_2(s) = e^{-3s} s/(s^2 + 4) = e^{-3s} H_1(s)$, is the same as $h_1(t)$ but 3 units shifted to the right; thus it

causes a delay of 3 units to an applied input. This phenomenon is easily explained by the time-shifting property of Laplace transform $L\{x(t - t_0)u(t - t_0)\} = e^{-st_0} X(s)$, in which it is stated that multiplying a signal in the complex frequency domain by e^{-st_0} results in delay of t_0 in the time domain.

11.3 Stability of Continuous-Time Systems

In Chapter 4, we have established a criterion for the stability of a system. More specifically, it was stated that a system is bounded-input bounded-output (BIBO) stable if the impulse response of the system is absolutely integrable. The mathematical expression is

$$\int_{-\infty}^{\infty} |h(t)| < \infty. \quad (11.8)$$

In this section, an alternative criterion regarding the BIBO stability of a system is given. First, two useful quantities must be defined. Suppose that a transfer function $H(s) = Y(s)/X(s)$ is written in the rational form of (11.7). Then,

- The roots of the numerator polynomial are called *zeros* of the system.
- The roots of the denominator polynomial are called *poles* of the system.

Recall also that the roots of a polynomial in the general case are complex numbers. The criterion about the stability of a system can be established now.

If all poles of a system transfer function are in the *left* half of the complex plane, the system is BIBO stable.

Equivalently, if the real parts of all poles are negative, the system is BIBO stable.

If a single pole is in the right half of the complex plane (or equivalently has positive real part), then the system is unstable. Finally, if a pole lies on the imaginary axis (or equivalently has zero real part), the system is critically stable.

Example

Determine if a system with transfer function

$$H(s) = \frac{3s^2 + 5}{s^3 + 3s^2 + 6s + 4}$$

is stable.

Commands	Results	Comments
<pre>den = [1 3 6 4]; poles = roots(den)</pre>	<pre>poles = -1.0000 + 1.7321i -1.0000 - 1.7321i -1.0000</pre>	The poles are computed, and since all poles have negative real part according to the stability criterion the system is stable.

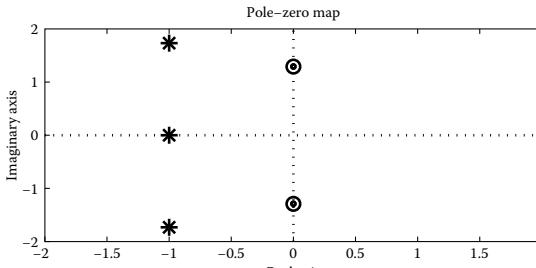
The poles are plotted in the complex plane in the following way.

Commands	Results
<pre>plot(real(poles), imag(poles), '*') xlim([-2 2]) legend('Poles') % Or more easily % plot(poles, '*')</pre>	

The poles are in the left half of the complex plane; thus the system is stable. In order to plot both poles and zeros in the same figure, we execute the following commands.

Commands	Results	Comments
<pre>num = [3 0 5]; zeros = roots(num); plot(real(poles), imag(poles), '*', real(zeros), imag(zeros), 'o') xlim([-2 2]) legend('poles', 'zeros')</pre>		The zeros are computed by finding the roots of the numerator polynomial. The poles are plotted with asterisks and the zeros are plotted with circles.

An alternative way of computing and plotting the poles and zeros of a transfer function is now introduced.

Commands	Results	Comments
<pre>num = [3 0 5]; den = [1 3 6 4]; H = tf(num, den)</pre>	Transfer function: $\frac{3s^2 + 5}{s^3 + 3s^2 + 6s + 4}$	The transfer function is defined by using the command <code>tf</code> .
<pre>poles = pole(H) zeros = zero(H)</pre>	$\text{poles} = -1.0000 + 1.7321i \quad -1.0000 - 1.7321i \quad -1.0000$ $\text{zeros} = 0 + 1.2910i \quad 0 - 1.2910i$	The poles are computed by the command <code>pole</code> , while the zeros are the outcome of the command <code>zero</code> .
<pre>pzmap(H) xlim([-2 2]) % Or alternatively pzmap(num, den);</pre>		The graph of poles and zeros in the complex plane is obtained by using the command <code>pzmap</code> .

11.4 Transfer Function in Zero/Pole/Gain Form

The transfer function of a linear single-input single-output (SISO) system can be written in a zero/pole/gain form, i.e., can be written in the form

$$H(s) = K \frac{(s - z_1)(s - z_2) \cdots (s - z_m)}{(s - p_1)(s - p_2) \cdots (s - p_n)}, \quad (11.9)$$

where z_1, z_2, \dots, z_m are the zeros of the system, p_1, p_2, \dots, p_n are the poles of the system, and coefficient K is called the gain of the system. The command used in order to convert a transfer function from rational form to zero/pole/gain form is the command `zpk`.

Example

Express the transfer function

$$H(s) = \frac{2s + 1}{s^2 + 3s + 2}$$

in zero/pole/gain form.

Commands	Results	Comments
<pre>num = [2 1]; den = [1 3 2]; H = tf(num, den); H2 = zpk(H)</pre>	Zero/pole/gain: $\frac{2(s + 0.5)}{(s + 1)(s + 2)}$	The transfer function $H(s)$ written in zero/pole/gain form. The output of the <code>zpk</code> command is a <code>zpk object</code> . For $H(s)$, the zero is $z_1 = -0.5$, the poles are $p_1 = -1$ and $p_2 = -2$, and the gain is $K = 2$.

One more MATLAB command that can be used to compute the zeros, poles, and gain of a transfer function is the command `tf2zp`. Its syntax is `[z, p, k] = tf2zp(num, den)`, where z is the vector of zeros z_i , p is the vector of poles p_i , and k is the gain K . It is also possible to do the inverse operation, that is, to convert a transfer function from zero/pole/gain form to rational form by using the command `[n, d] = zp2tf(z, p, k)`.

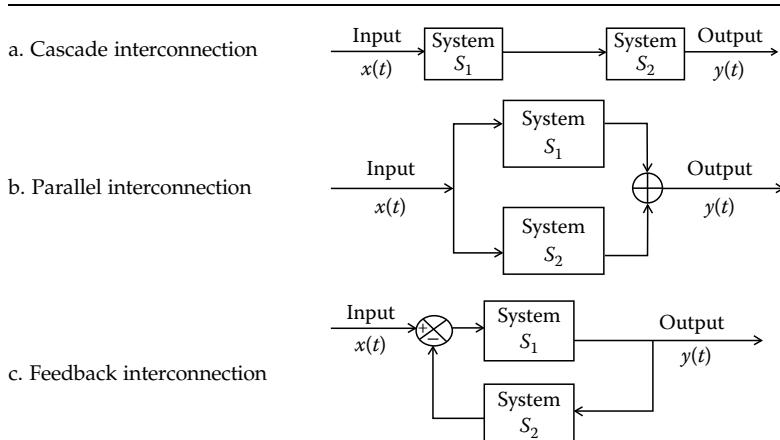
Commands	Results	Comments
<code>num = [2 1]; den = [1 3 2]; [z, p, k] = tf2zp(num, den)</code>	$z = -0.5000$ $p = -2 -1$ $k = 2$	The zeros z_i , the poles p_i , and the gain K of the zero/pole/gain form.
<code>[n, d] = zp2tf(z, p, k)</code>	$n = 0 \ 2 \ 1$ $d = 1 \ 3 \ 2$	Using the command <code>zp2tf</code> , we obtain the numerator and denominator polynomials of the rational form.

11.5 Interconnections of Systems

In this section, we introduce the way of dealing with the basic interconnections between systems in terms of each system's transfer function. Recall from Chapter 4 that the basic possible interconnections between (sub)systems are the cascade, the parallel, the mixed, and the feedback.

Example

The transfer function of subsystem S_1 is $H_1(s) = 1/500s^2$, while the transfer function of subsystem S_2 is $H_2(s) = (s + 1)/(s + 2)$. Calculate the overall system transfer function $H(s)$ for the cases of interconnections that are shown below.



- a. The overall transfer function $H(s)$ of the cascade-connected systems is computed by the command `series`.

Commands	Results	Comments
<pre>num1 = 1; den1 = [500 0 0]; num2 = [1 1]; den2 = [1 2]; [num, den] = series(num1, den1, num2, den2)</pre>	<pre>num = 0 0 1 1 den = 500 1000 0 0</pre>	The numerator and denominator coefficients of the two subsystems are the input arguments of the <code>series</code> command. The command returns the numerator and denominator coefficients of the overall transfer function $H(s)$.
<pre>printsys(num, den)</pre>	<pre>num/den = s + 1 500s^3 + 1000s^2</pre>	The command <code>printsys</code> prints the derived coefficients in rational form.

Alternatively, and more elegantly, the `series` command can be used with TF objects.

Commands	Results	Comments
<pre>H1 = tf(num1, den1); H2 = tf(num2, den2); H = series(H1, H2)</pre>	<pre>Transfer function: s + 1 500s^3 + 1000s^2</pre>	The transfer functions of the two subsystems are defined as TF objects. The output of <code>series</code> command (which is the overall transfer function) is also a TF object.

To confirm the result derived from the `series` command, recall that for two cascade-connected systems with impulse responses $h_1(t)$ and $h_2(t)$, respectively, the overall impulse response $h(t)$ is given by the convolution between $h_1(t)$ and $h_2(t)$, namely, $h(t) = h_1(t) * h_2(t)$. The overall transfer function $H(s)$ is the Laplace transform of the overall impulse response $h(t)$ given by $h(t) = h_1(t) * h_2(t)$. But convolution in the time domain is transformed into multiplication in the complex frequency domain; thus $H(s)$ is computed as $H(s) = H_1(s)H_2(s)$.

Commands	Results	Comments
<pre>num = conv(num1, num2) den = conv(den1, den2) printsys(num, den)</pre>	<pre>num/den = s + 1 500s^3 + 1000s^2</pre>	The command <code>conv</code> is used to compute the product $H_1(s)H_2(s)$.
<pre>H = H1 * H2</pre>	<pre>Transfer function: s + 1 500s^3 + 1000s^2</pre>	Alternatively, we can directly compute the product of two TF objects.

The result obtained from the last procedure is equal to the previous two; therefore, we conclude that if two subsystems are cascade connected the transfer function of the overall system is equal to the product of the transfer functions of the subsystems.

- b. The overall system transfer function $H(s)$ of two parallel-connected subsystems is computed by the command `parallel`.

Commands	Results	Comments
<pre>num1 = 1; den1 = [500 0 0]; num2 = [1 1]; den2 = [1 2]; H1 = tf(num1,den1); H2 = tf(num2,den2); H = parallel(H1,H2)</pre>	<p>Transfer function:</p> $\frac{500s^3 + 500s^2 + s + 2}{500s^3 + 1000s^2}$	The system transfer function $H(s)$ when the system consists of two parallel-connected subsystems.

To verify the result obtained from the `parallel` command, recall that for two parallel-connected subsystems with impulse responses $h_1(t)$ and $h_2(t)$, the overall impulse response $h(t)$ is computed by the sum of $h_1(t)$ and $h_2(t)$, namely, $h(t) = h_1(t) + h_2(t)$. Thus, first the impulse responses of the two subsystems are added and the Laplace transform of the result is the system transfer function. Alternatively (due to the linearity property of Laplace transform), the system transfer function can be computed from the sum of the transfer functions of the subsystems $H_1(s) = L\{h_1(t)\}$ and $H_2(s) = L\{h_2(t)\}$, i.e., $H(s) = H_1(s) + H_2(s)$.

Commands	Results	Comments
<pre>syms s t H1 = 1/(500*s^2); h1 = ilaplace(H1,t); H2 = (s+1)/(s+2); h2 = ilaplace(H2,t); h = h1+h2; H = laplace(h,s); H = simplify(H); pretty(H)</pre>	<p>Transfer function:</p> $\frac{500s^3 + 500s^2 + s + 2}{500s(s^2 + 2s)}$	The system transfer function is computed by adding the impulse responses of the subsystems and applying Laplace transform to the result.
<pre>H1 = 1/(500*s^2); H2 = (s+1)/(s+2); H = H1+H2; H = simplify(H); pretty(H)</pre>	<p>Transfer function:</p> $\frac{500s^3 + 500s^2 + s + 2}{500s(s^2 + 2s)}$	Alternatively, the system transfer function is directly computed by adding the transfer functions of the subsystems.
<pre>H1 = tf(num1,den1); H2 = tf(num2,den2); H = H1+H2</pre>	<p>Transfer function:</p> $\frac{500s^3 + 500s^2 + s + 2}{500s(s^2 + 2s)}$	Finally, we can directly compute the system transfer function by adding the transfer functions of the subsystems if these are defined as <code>TF</code> objects.

To conclude, if a system consists by two parallel-connected subsystems, the system transfer function is the sum of the transfer functions of the two subsystems.

- c. The overall system transfer function $H(s)$ for two feedback-connected subsystems is called closed-loop transfer function, and is given by the mathematical expression

$$H(s) = \frac{H_1(s)}{1 + H_1(s)H_2(s)}, \quad (11.10)$$

where

$H_1(s)$ is the transfer function of the direct branch

$H_2(s)$ is the transfer function of the feedback branch

In a feedback interconnection, the closed-loop transfer function is computed by the MATLAB command `feedback`.

Commands	Results	Comments
<pre>num1 = 1; den1 = [500 0 0]; num2 = [1 1]; den2 = [1 2]; H1 = tf(num1,den1); H2 = tf(num2,den2); H = feedback(H1,H2)</pre>	<p style="text-align: center;">Transfer function:</p> $\frac{s + 2}{500s^3 + 1000s^2 + s + 1}$	The closed-loop transfer function $H(s)$.

To verify the result obtained from the `feedback` command, we compute the closed-loop transfer function according to Equation 11.10.

Commands	Results	Comments
<pre>syms s H1 = 1/(500*s^2); H2 = (s+1)/(s+2); H = H1/(1+H1*H2); simplify(H)</pre>	$ans = (s+2)/(500s^3+1000s^2+s+1)$	The result obtained from the <code>feedback</code> command is confirmed according to Equation 11.10.

11.6 Continuous-Time System Response

Suppose that $H(s)$ is the transfer function of a system. The system response $y(t)$ to an input signal $x(t)$ is the inverse Laplace transform of $Y(s) = H(s)X(s)$. If the system is stable, the system response $y(t)$ after a definite time interval T_s , called reset time, converges to a constant state. This state of $y(t)$ is called steady state and is denoted by $y_{ss}(t)$. The state of $y(t)$ for $0 \leq t \leq T_s$ is called transient state and is denoted by $y_{ts}(t)$. If the transfer function is known and expressed in rational form, one can compute the system response to any input signal by using the command `lsim`. The syntax is `y = lsim(num, den, x, t)`, where `num` and `den` are the numerator and denominator coefficient vectors of the transfer function, `x` is the input signal, and `t` is the time in which the input signal is applied and the output signal is computed. An alternative syntax is `y = lsim(H, x, t)`, where `H` can be a TF or a ZPK object that represents the transfer function.

Example

A system is specified by the transfer function

$$H(s) = \frac{10}{s^2 + 2s + 10}.$$

Compute and plot the system response $y(t)$ to the input signal $x(t) = \cos(2\pi t)$, $0 \leq t \leq 10$.

Commands	Results	Comments
<pre>n=10; d=[1 2 10]; H=tf(n,d); t=0:.1:10; x=cos(2*pi*t); y=lsim(H,x,t); plot(t,y); legend('y(t)');</pre>		<p>First, $H(s)$ is defined with the <code>tf</code> command, while next using the <code>lsim</code> command the system response $y(t)$ to the input $x(t)$ is computed over the time interval in which $x(t)$ is defined.</p>
<pre>y=lsim(n,d,x,t); plot(t,y); legend('y(t)');</pre>		<p>Output computation using an alternative syntax of the <code>lsim</code> command.</p>
<pre>lsim(H,x,t)</pre>		<p>If the command <code>lsim</code> is executed without specifying an output argument, the output and the input signals are plotted in the same figure. The input signal $x(t)$ is depicted with the dotted line, while the output signal $y(t)$ is plotted with the solid line.</p>

Notice that initially there is a variation at the system response, but after the time instance $t=2$ the output signal reaches its steady state.

Recall that *step response* is the system response to a unit step input signal $u(t)$, while *impulse response* is the system response to the Dirac delta input signal $\delta(t)$. In order to compute the step response and the impulse response of a system, one can use the commands `step` and `impulse`, respectively.

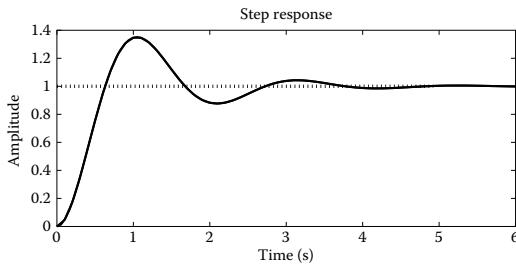
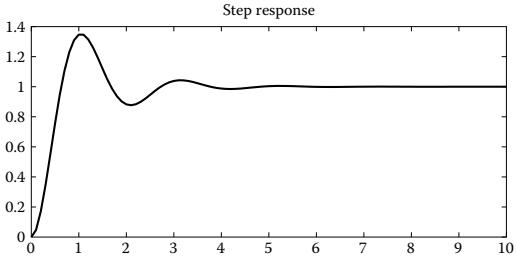
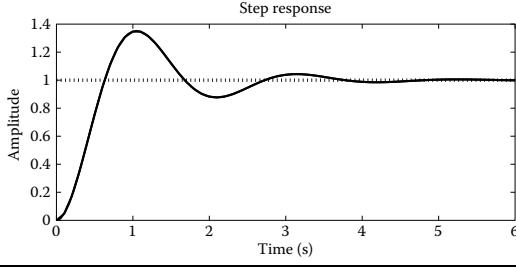
Example

Suppose that a system is specified by the transfer function

$$H(s) = \frac{10}{s^2 + 2s + 10}.$$

Compute and plot

- a. The system step response $s(t)$
- b. The system impulse response $h(t)$
- a. The system step response is computed by the command `step`.

Commands	Results	Comments
<pre>n=10; d=[1 2 10]; H=tf(n,d); step(H)</pre>	 <p>Step response</p> <p>This plot shows the step response of the system over time. The x-axis represents Time (s) from 0 to 6, and the y-axis represents Amplitude from 0 to 1.4. The response starts at (0,0), rises sharply to a peak of about 1.3 at t=1, crosses the steady-state value of 1.0 at t ≈ 1.5, and then damps out with decreasing amplitude, eventually settling to a steady-state value of 1.0 after about 3 seconds.</p>	<p>The <code>step</code> command plots the step response up to the point in which the output reaches its steady state. The steady state of the system is depicted with the dashed line.</p>
<pre>t=0:.1:10; s=step(H,t); plot(t,s); title('Step response')</pre>	 <p>Step response</p> <p>This plot shows the step response over a longer time interval from 0 to 10 seconds. The transient behavior is similar to the first plot, but the system continues to oscillate around the steady-state value of 1.0, with the transient component gradually dying out over the 10-second period.</p>	<p>However, a different time interval can be specified. Here, the step response is computed and plotted for $0 \leq t \leq 10$. Notice that the steady state is $y_{ss}(t)=1$.</p>
<pre>Hzpk=zpk(H); step(Hzpk)</pre>	 <p>Step response</p> <p>This plot shows the step response of the system in zero/pole/gain form. The transient behavior is identical to the first plot, where the system reaches a steady-state value of 1.0 after approximately 3 seconds.</p>	<p>The command <code>step</code> is also applicable if the transfer function is in zero/pole/gain form.</p>

To verify the obtained result the system step response is computed as $y(t) = L^{-1}\{Y(s)\} = L^{-1}\{H(s)X(s)\}$, where $X(s) = L\{x(t)\}$ and the input signal $x(t)$ is the unit step function $u(t)$.

Commands	Results	Comments
<pre> syms t s x=heaviside(t); X=laplace(x,s); H=10/(s^2+2*s+10); Y=H*X; y=ilaplace(Y,t); t=0:.1:6; s=subs(y,t); plot(t,s) title('Step response') </pre>	<p style="text-align: center;">Step response</p>	The result of the step command is verified by computing the system response to the input signal $u(t)$.

An alternative verification is to use the `lsim` command to compute the system response to the input signal $x(t) = u(t)$.

Commands	Results	Comments
<pre> n=10; d=[1 2 10]; t=0:.1:6; x=ones(size(t)); s=lsim(n,d,x,t); plot(t,s); legend('s(t)'); </pre>		The step response of the system computed by the <code>lsim</code> command.

- b. The impulse response of the system is computed by the command `impulse`. The syntaxes illustrated for the `step` command are also applicable for the command `impulse`.

Commands	Results	Comments
<pre> n=10; d=[1 2 10]; H=tf(n,d); impulse(H) </pre>	<p style="text-align: center;">Impulse response</p>	The system impulse response is plotted with use of the command <code>impulse</code> .

To verify our result, recall that $h(t) = L^{-1}\{H(s)\}$.

Commands	Results	Comments
<pre> syms s t H=10/(s^2+2*s+10); h=ilaplace(H,t); t=0:1:6; h=subs(h,t); plot(t,h) legend('h(t)') </pre>		The impulse response is computed as the inverse Laplace transform of the transfer function.

11.7 Discrete-Time Systems

The transfer function $H(z)$ of a discrete-time system is defined in a similar way to the transfer function of continuous-time systems. So the transfer function of an LTI discrete-time system is defined as the ratio of the z -transform $Y(z)$ of the system's output signal to the z -transform $X(z)$ of the input signal that is applied to the system, supposing zero initial conditions. The mathematical expression is

$$H(z) = \frac{Y(z)}{X(z)}. \quad (11.11)$$

An alternative definition for a discrete-time system transfer function is that the transfer function $H(z)$ of a system is the z -transform of the impulse response $h[n]$ of the system. The mathematical expression is

$$H(z) = Z\{h[n]\}. \quad (11.12)$$

Example

Compute the transfer function of the discrete-time system with impulse response $h[n] = 2^n u[n]$.

Commands	Results	Comments
<pre> syms n z h=2^n; H=ztrans(h,z) H=simplify(H) </pre>	$H = z / (z - 2)$	The transfer function $H(z)$ is computed directly from (11.12)

The transfer function of a discrete-time system can be derived also from the difference equation that describes the system.

Example

Compute the transfer function of a system described by the difference equation $y[n] - y[n-1] = x[n] + x[n-1]$ assuming that the initial conditions are zero.

First, z-transform is applied to both sides of the difference equation. The initial conditions are zero; hence, the property $Z[x[n-1]] = z^{-1}Z[x[n]] = z^{-1}X(z)$ is valid and can be used. Next, the difference equation (which is now transformed into algebraic equation) is solved for $Y(z)$, and the transfer function $H(z)$ is obtained by dividing $Y(z)$ with $X(z)$.

Commands	Results	Comments
<pre>syms n z X Y Y1 = (z^-1)*Y; X1 = (z^-1)*X; G = Y-Y1-X-X1; Y = solve(G, Y); H = Y/X</pre>	$H = (z+1)/(z-1)$	The transfer function $H(z)$ is derived by first solving the z-transform of the difference equation for $Y(z)$ and then dividing $Y(z)$ by $X(z)$ according to (11.11).

The coefficients of the denominator polynomial of the system transfer function are the same as the coefficients of the output signal y in the difference equation, while the coefficients of the numerator polynomial of the system transfer function are the same as the coefficients of the input signal x in the difference equation. Hence, if a system transfer function is known, we can directly derive the difference equation that specifies the system, and vice versa.

11.8 The Command `tf` for Discrete-Time Systems

The use and syntax of the `tf` command in the discrete-time case is similar to the one in the continuous-time case except from the fact that we have to indicate one more input argument that specifies the sampling time. The syntax of the `tf` command for discrete-time systems is $H = \text{tf}(\text{num}, \text{den}, Ts)$, where *num* and *den* are the numerator and denominator coefficients of the transfer function and *Ts* is the sampling time.

Commands	Results	Comments
<pre>num = [2 1]; den = [1 3 2]; Ts = 0.4; H = tf(num, den, Ts)</pre>	Transfer function: $\frac{2 * z + 1}{z^2 + 3 * z + 2}$ Sampling time: 0.4	If we specify the sampling time argument, the transfer function is written in terms of z .

11.9 Stability of Discrete-Time Systems

The calculation of the zeros and poles of a discrete-time transfer function is very important as the knowledge of the poles provides a criterion for the stability of a system. The commands introduced for the continuous-time case are also used for discrete-time systems.

Example

Compute and plot the poles and the zeros of the transfer function

$$H(z) = \frac{2z + 1}{z^2 + 3z + 2}.$$

First way

Commands	Results	Comments
<pre>n = [2 1]; d = [1 3 2]; zer = roots(n); pol = roots(d); plot(real(pol), imag(pol), '*', real(zer), imag(zer), 'o') xlim([-3 1]); legend('poles', 'zeros');</pre>		Computation and graph of the zeros and poles of $H(z)$.

Second way

Commands	Results	Comments
<pre>H = tf(n, d, 0.1); poles = pole(H) zeros = zero(H) pzmap(H) xlim([-3 1.2])</pre>	<p>Pole-zero map</p>	Calculation of the zeros and poles of $H(z)$. The poles and zeros are plotted with use of the command <code>pzmap</code> .

Stability criterion: In the z -plane the stability criterion is not the imaginary axis, but the *unit circle* $|z| = 1$. Hence,

A discrete-time system is stable if all the poles of its transfer function lie inside the unit circle. If one pole lies outside the unit circle the system is unstable.

As one can see in the previous figure, the command `pzmap` besides the zeros and poles of the transfer function also plots the unit circle in the complex plane. Thus, from the above figure we conclude that the system with transfer function

$$H(z) = \frac{2z + 1}{z^2 + 3z + 2}$$

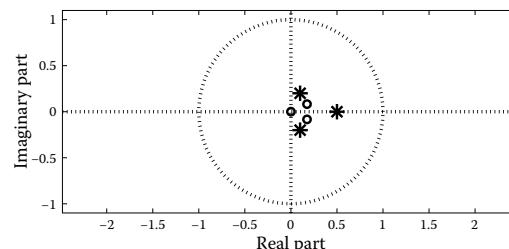
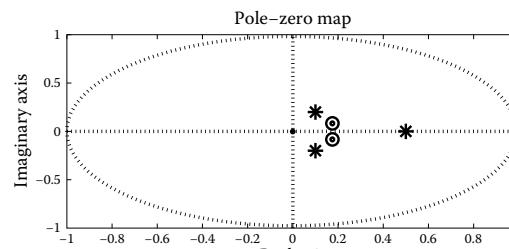
is not stable since the pole $p_1 = -2$ is not inside the unit circle. Another useful command that plots the poles, the zeros, and the unit circle in the z -plane is the command `zplane`. Its syntax is `zplane (num, den)`, where *num* and *den* are the coefficients of the numerator and denominator of the system transfer function, respectively.

Example

Determine if the discrete-time system with transfer function

$$H(z) = \frac{3z^2 - 1.4z + 0.15}{z^3 - 0.7z^2 + 0.15z - 0.025}$$

is stable.

Commands	Results	Comments
<pre>n = [4 -1.4 .15]; d = [1 -.7 .15 -.025]; zplane(n,d)</pre>		Graph of poles and zeros of $H(z)$ together with the unit circle.
<pre>H = tf (n, d, 0.1); pzmap (H)</pre>		Equivalent graph obtained with use of the command <code>pzmap</code> .

All the poles of $H(z)$ lie inside the unit circle, hence the system is stable. The poles lie inside the unit circle if their magnitude is less than one. To verify this statement, we calculate the poles of $H(z)$, and we compute their magnitudes.

Commands	Results	Comments
<code>poles=pole(H)</code>	<code>poles = 0.50 0.10 + 0.20i 0.10 - 0.20i</code>	The poles of the transfer function $H(z)$.
<code>mag=abs(poles)</code>	<code>mag = 0.5000 0.2236 0.2236</code>	The magnitudes of the poles are less than 1; hence, the system is stable.

Finally, we mention that the transfer function of a discrete-time system written in rational form can be expressed in zero/pole/gain form with use of the command `zpk`.

Example

Express in zero/pole/gain form the transfer function

$$H(z) = \frac{2z^2 - 1}{z^2 + z - 12}.$$

Commands	Results	Comments
<code>n = [2 0 -1];</code> <code>d = [1 1 -12];</code> <code>H = tf(n, d, 0.5);</code> <code>zpk(H)</code>	Zero/pole/gain: $\frac{2(z - 0.7071)(z + 0.7071)}{(z + 4)(z - 3)}$ Sampling time: 0.5	Zero/pole/gain form of the transfer function $H(z)$.

11.10 Discrete-Time System Response

In this section, we will discuss how to compute the response of a discrete-time system to various input signals when the system transfer function is known.

11.10.1 Step Response

The step response $s[n]$ of a discrete-time system is the response of the system to the unit step sequence

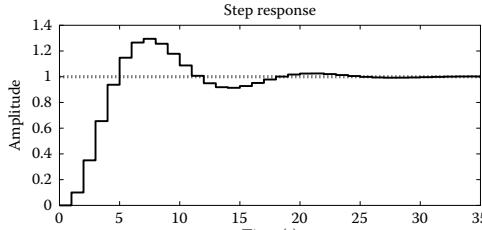
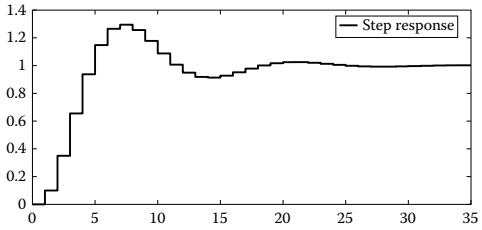
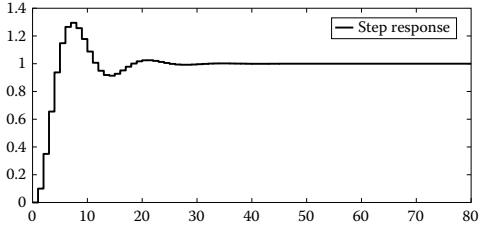
$$u[n] = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

The response of a system to $u[n]$ is computed with the help of the command `dstep`, while the graph of step response is implemented with the command `stairs`. The syntax of `dstep` is `y = dstep(num, den)`, where `num` and `den` are the coefficients of the numerator and denominator of the system transfer function, respectively.

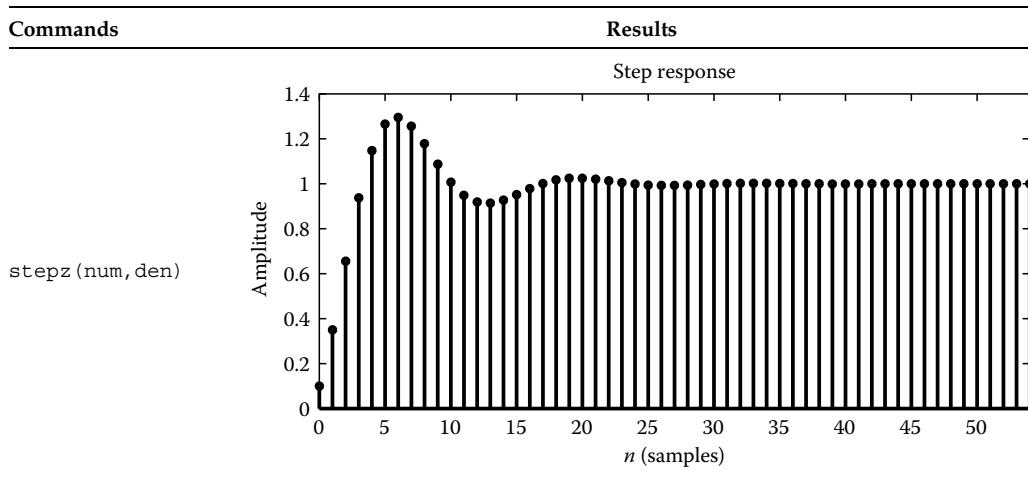
Example

Compute the step response of the discrete-time system with transfer function

$$H(z) = \frac{0.1z - 0.1}{z^2 - 1.5z + 0.7}.$$

Commands	Results	Comments
<pre>num = [.1 .1]; den = [1 -1.5 0.7]; dstep(num,den)</pre>		<p>If no output argument is specified, the command <code>dstep</code> plots directly the step response $s[n]$ of the discrete-time system. The output signal is plotted up to the point where it reaches its steady state. The steady state of the system is depicted with the dashed line.</p>
<pre>s=dstep(num,den) stairs(0:length(s)-1,s); legend('Step response')</pre>		<p>The step response signal $s[n]$ is computed by the command <code>dstep</code> and is plotted with use of the command <code>stairs</code>.</p>
<pre>n=0:80; s=dstep(num,den,n); stairs(n,s) legend('Step response')</pre>		<p>Graph of step response for $0 \leq n \leq 80$.</p>

An alternative way in order to compute and plot the step response of the system is to use the command `stepz`. The syntax is `y = stepz (num, den)`, where *num* and *den* are the coefficients of the numerator and denominator of the system transfer function, respectively.



11.10.2 Impulse Response

The impulse response $h[n]$ of a discrete-time system is the response of the system to the unit impulse sequence (or delta function)

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}.$$

The response $h[n]$ of a system to $\delta[n]$ is computed with the help of the command `dimpulse`, while the graph of impulse response is also implemented by the command `stairs`. The syntax of `dimpulse` is `y = dimpulse (num, den)`, where *num* and *den* are the coefficients of the numerator and denominator of the system transfer function, respectively.

Example

Compute and plot the impulse response of the discrete-time system with transfer function

$$H(z) = \frac{0.1z - 0.1}{z^2 - 1.5z + 0.7}.$$

Commands	Results	Comments
<pre>num = [.1 .1]; den = [1 -1.5 0.7]; dimpulse(num, den)</pre>		<p>The impulse response of the discrete-time system is plotted up to the point where it reaches its steady state. The steady state of the system is depicted with the dashed line.</p>
<pre>h = dimpulse(num, den); stairs(0:length(h)-1, h) legend('Impulse response')</pre>		<p>The impulse response $h[n]$ is computed by the command <code>dimpulse</code> and is plotted by the command <code>stairs</code>.</p>
<pre>n = 0:50; y = dimpulse(num, den, n); stairs(n, y) legend('h[n]')</pre>		<p>The impulse response $h[n]$ is defined and plotted for $0 \leq n \leq 50$.</p>

Finally, one may use the command `impz` to compute and plot the impulse response of a system.

Commands	Results
<pre>num = [.1 .1]; den = [1 -1.5 0.7]; impz(num, den)</pre>	

11.10.3 The Command `dlsim`

If the transfer function of a discrete-time system is known, the response of the system to an arbitrary input signal is computed by the command `dlsim`. Its syntax is `y = dlsim(num, den, x)`, where `num` and `den` are the coefficients of the numerator and denominator of the system transfer function, respectively.

Example

Compute and plot with use of the command `dlsim` the impulse response $h[n]$ of the discrete-time system with transfer function

$$H(z) = \frac{0.1z - 0.1}{z^2 - 1.5z + 0.7}.$$

The impulse response $h[n]$ of the system was computed in the previous example with use of the command `dimpulse`. In this example, in order to find $h[n]$, we set $x[n] = \delta[n]$, i.e., the unit impulse sequence $\delta[n]$ is the applied input signal. The response of the system to $\delta[n]$ is the impulse response $h[n]$.

Commands	Results	Comments
<pre>num = [.1 .1]; den = [1 -1.5 0.7]; n = 0:50; x = [1 zeros(1,50)]; y = dlsim(num,den,x); stairs(n,y) legend('y[n] = h[n]');</pre>		<p>The response $y[n]$ of the system to $\delta[n]$ is the impulse response $h[n]$ of the system.</p>

As expected, the obtained impulse response $h[n]$ is the same as the one derived in the previous example.

Example

Compute the response of the discrete-time system with transfer function

$$H(z) = \frac{0.1z - 0.1}{z^2 - 1.5z + 0.7}$$

to the input signal $x[n] = (-1)^n$, $0 \leq n \leq 50$.

Commands	Results
<pre> num = [0.1 0.1]; den = [1 -1.5 0.7]; n = 0:50; x = (-1).^n; y = dlsim(num, den, x); stairs(n, y); legend ('Output signal y[n]') </pre>	

To confirm our result we will compute the system response by using the command `filter`. This is very logic as the coefficients of the difference equation that describes the system are equal to the coefficients of the transfer function $H(z)$.

Commands	Results
<pre> y2 = filter(num, den, x); stairs(n, y2); legend ('Output signal y[n]') </pre>	

11.11 Conversion between Continuous-Time and Discrete-Time Systems

A continuous-time system S_c described by a transfer function $H(s)$ can be converted to a discrete-time system S_d described by a transfer function $H(z)$ with use of the command `c2d`. The appropriate syntax is `sysdis = c2d(syscont, Ts, method)`, where `syscont` denotes the continuous-time system, `sysdis` is the discrete-time system, `Ts` is the sampling time, and `method` is the selected discretization method. The available discretization methods are

- “zoh”: Zero-order hold on the inputs. This is the default method.
- “foh”: Linear interpolation of inputs.
- “imp”: Impulse-invariant discretization.
- “tustin”: Bilinear (Tustin) approximation.
- “prewarp”: Tustin approximation with frequency prewarping.
- “matched”: Matched pole-zero method.

Example

Discretize the continuous-time system with transfer function $H(s) = (s + 10)/s$ with use of the zero-order hold method.

Commands	Results	Comments
<code>n = [1 10]; d = [1 0]; Hs = tf(n, d)</code>	Transfer function: $\frac{s + 10}{s}$	Definition of the continuous-time transfer function $H(s)$ with use of the <code>tf</code> command.
<code>Ts = 0.2; Hz = c2d(Hs, Ts, 'zoh')</code>	Transfer function: $\frac{z + 1}{z - 1}$ Sampling time: 0.2	The discretized transfer function is $H(z) = \frac{z + 1}{z - 1}$.

On the other hand, a discrete-time system can be converted to a continuous-time system with the command `d2c`.

Example: Convert the discrete-time system with transfer function $H(z) = (z + 1)/(z - 1)$ to a continuous-time system.

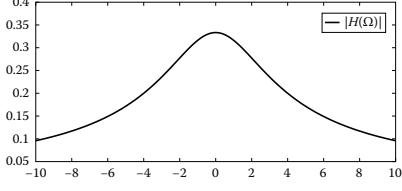
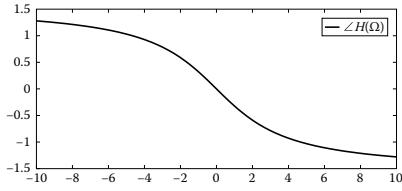
Commands	Results	Comments
<code>num = [1 1]; den = [1 -1]; Ts = 0.2; Hz = tf(num, den, Ts)</code>	Transfer function: $\frac{z + 1}{z - 1}$ Sampling time: 0.2	Definition of the discrete-time transfer function $H(z)$.
<code>Hs = d2c(Hz, 'zoh')</code>	Transfer function: $\frac{s + 10}{s}$	The equivalent continuous-time transfer function is $H(s) = \frac{s + 10}{s}$.

11.12 Transfer Function and Frequency Response

In this section, we introduce the relationship between the transfer function $H(s)$ (or $H(z)$) and the frequency response $H(j\Omega)$ (or $H(\omega)$) of a continuous-time (or discrete-time) system. First, we discuss the continuous-time case. If the system is stable, the frequency response $H(j\Omega)$ of the system is computed by substituting s by $j\Omega$ in $H(s)$. In order to numerically compute $H(j\Omega)$ over a frequency range $\Omega_1 \leq \Omega \leq \Omega_2$ rad/s from the system transfer function $H(s)$, one can use the command `freqresp`. Its syntax is `Hw = freqresp(Hs, w)`, where Hs denotes the transfer function $H(s)$ of the system, w is the vector containing the frequencies, and Hw is the frequency response of the system evaluated at the frequencies specified in vector w . However, the output argument Hw is a 3-D matrix. In the simple case of a SISO system, the value of the frequency response for a specific frequency is contained in the third dimension of the 3-D matrix Hw . Hence, in order to plot the magnitude of the computed frequency response the appropriate statement is `plot(w, abs(Hw(:, :, :)))`, and correspondingly the graph of the phase is derived with the statement `plot(w, angle(Hw(:, :, :)))`.

Example

Compute and plot the frequency response of the system with transfer function $H(s) = 1/(s + 3)$.

Commands	Results	Comments
<pre>Hs=tf(1,[1 3])</pre>	Transfer function: $\frac{1}{s+3}$	Definition of the system transfer function $H(s)$.
<pre>w=-10:.1:10; Hw=freqresp(Hs,w); plot(w,abs(Hw(:,1))); legend(' H(j\Omega) ');</pre>		The graph of the frequency response $H(j\Omega)$ is obtained by plotting the last argument of the 3-D matrix Hw versus the frequency Ω .
<pre>plot(w,angle(Hw(:,1))); legend('\angle H(j\Omega)');</pre>		The phase of the frequency response $H(j\Omega)$.
<pre>w=-2:2 Hw=freqresp(Hs,w)</pre>	<pre>Hw(:,:,1) = 0.2308 + 0.1538i Hw(:,:,2) = 0.3000 + 0.1000i Hw(:,:,3) = 0.3333 Hw(:,:,4) = 0.3000 - 0.1000i Hw(:,:,5) = 0.2308 - 0.1538i</pre>	Illustration of the 3-D matrix Hw . The frequency response is evaluated at the frequencies specified in vector w . For example, for $\Omega = -1$ rad/s $H(j\Omega) = 0.3 + 0.1j$.

We now discuss how to compute the frequency response $H(\omega)$ of a discrete-time system via the transfer function $H(z)$ of the discrete-time system. Suppose that a discrete-time signal is described by a sequence $x[n]$ that is zero for $n < 0$. The discrete-time Fourier transform (DTFT) $X(\omega)$ of $x[n]$ is computed by $X(\omega) = \sum_{n=0}^{\infty} x[n]e^{-j\omega n}$. On the other hand, the z-transform $X(z)$ of $x[n]$ is given by $X(z) = \sum_{n=0}^{\infty} x[n]z^{-n}$. Thus, we can derive the DTFT $X(\omega)$ of $x[n]$ by substituting in $X(z)$ the variable z with $e^{j\omega}$. This is the reason that in some books the DTFT of a signal $x[n]$ is denoted as $X(e^{j\omega})$. Thus, if

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^K b_k z^{-k}}{\sum_{m=0}^M a_m z^{-m}}$$

is the transfer function of a discrete-time system, substituting z by $e^{j\omega}$ in $H(z)$ yields the discrete-time frequency response of the system

$$H(\omega) = \frac{Y(\omega)}{X(\omega)} = \frac{\sum_{k=0}^K b_k e^{-j\omega k}}{\sum_{m=0}^M a_m e^{-j\omega m}}.$$

Example

Compute with the help of z-transform the frequency response $H(\omega)$ of a discrete-time system with impulse response $h[n] = [3, 5, 2, 1]$, $0 \leq n \leq 3$.

Commands	Results	Comments
<pre>n = 0 : 3; h = [3 5 2 1]; syms z w Htf = sum(h.*z.^-n);</pre>		We define the impulse response $h[n]$, and by applying z-transform to $h[n]$ we compute the transfer function $H(z)$ of the system.
<pre>H = subs(Htf, z, exp(j*w)); H = simplify(H)</pre> <pre>Hw = sum(h.*exp(-j*w*n))</pre>	$H = 3 + 5 \exp(-i\omega) + 2 \exp(-2i\omega) + \exp(-3i\omega)$ $Hw = 3 + 5 \exp(-i\omega) + 2 \exp(-2i\omega) + \exp(-3i\omega)$	We substitute z by $e^{j\omega}$ in the transfer function $H(z)$, and we derive the frequency response $H(\omega)$ of the system. To confirm our result, we compute $H(\omega)$ by applying DTFT to the impulse response $h[n]$.

Example

Compute the frequency response $H(\omega)$ of the discrete-time system with impulse response $h[n] = (2/3)^n u[n]$.

Commands	Results	Comments
<pre>syms n z w h = (2/3)^n * heaviside(n); Hz = ztrans(h, z);</pre> <pre>H = subs(Hz, z, exp(j*w)); H = simplify(H)</pre>		Definition of the impulse response $h[n]$ and computation via z-transform of the transfer function $H(z)$.
$h = (2/3)^n;$ $Hw = symsum(h*exp(-j*w*n), n, 0, inf)$	$H = 3 \exp(i\omega) / (3 \exp(i\omega) - 2)$ $Hw = 3 \exp(i\omega) / (-2 + 3 \exp(i\omega))$	Substituting z by $e^{j\omega}$ in $H(z)$ yields the discrete-time frequency response $H(\omega)$. Confirmation of our result by directly computing the DTFT of $h[n]$.

Example

Compute with the help of z-transform the frequency response $H(\omega)$ of a system described by the difference equation $y[n] = 0.9y[n - 1] + x[n]$ with zero initial conditions.

In order to compute the transfer function $H(z)$ of the system we will apply z-transform to both sides of the difference equation taking into account that $Z\{y[n - 1]\} = z^{-1} Z\{y[n]\} = z^{-1} Y(z)$. Next, we will substitute z by $e^{j\omega}$ in order to derive the frequency response $H(\omega)$ of the system.

Commands	Results	Comments
<pre>syms z w Yz X Y1z = z^(-1)*Yz; G = Yz - 0.9*Y1z - X; Yz = solve(G, Yz);</pre>		We apply z -transform to both sides of the difference equation and solve for $Y(z)$.
<pre>Hz = Yz/X</pre>	$Hz = 10 / (10 + 9 * z)$	The transfer function is computed as $H(z) = \frac{Y(z)}{H(z)}.$
<pre>Hw = subs(Hz, z, exp(j*w))</pre>	$Hw = 10 / (10 + 9 * \exp(i * w))$	Substituting z by $e^{j\omega}$ in $H(z)$ yields the discrete-time frequency response $H(\omega)$.

Example

Recall that an N -point moving average filter is described by the i/o relationship $y[n] = (1/N) \sum_{k=0}^{N-1} x[n-k]$, while its impulse response is given by $h[n] = 1/N$, $0 \leq n \leq N-1$. Compute with the help of z -transform the frequency response and the impulse response of a 3-point moving average filter.

The i/o relationship of a 3-point moving average filter is given by $y[n] = 1/3 (x[n] + x[n-1] + x[n-2])$, while its impulse response is given by $h[n] = 1/3$, $0 \leq n \leq 2$. The MATLAB implementation is as follows.

Commands	Results	Comments
<pre>syms z w Xz Yz X1 = z^(-1)*Xz; X2 = z^(-2)*Xz; Yz = (1/3)*(Xz + X1 + X2);</pre>		We denote the z -transforms of $x[n-1]$ and $x[n-2]$ as $X1$ and $X2$, respectively, and define them according to the time-shifting property of z -transform. Next, we apply z -transform to both sides of the difference equation.
<pre>Hz = Yz/Xz Hz = simplify(Hz)</pre>	$Hz = 1/3 * (z^2 + z + 1) / z^2$	The transfer function is computed as $H(z) = \frac{Y(z)}{H(z)}.$
<pre>Hw = subs(Hz, z, exp(j*w)) Hw = simplify(Hw)</pre>	$Hw = 1/3 + 1/3 * \exp(-i * w) + 1/3 * \exp(-2 * i * w)$	We substitute z by $e^{j\omega}$ at $H(z)$, and we obtain the frequency response $H(\omega)$ of the moving average filter.
<pre>hn = iztrans(Hz) hn = charfcn[2](n) + 1/3 * charfcn[1](n) + 1/3 * charfcn[0](n)</pre>		In order to derive the impulse response $h[n]$ of the moving average filter, we compute the inverse z -transform of the transfer function $H(z)$. Recall that the function $charfcn[A](x)$ returns 1 if x is in set A and zero otherwise. Hence, $hn = 1/3$ if $n = 0, 1, 2$ and zero for any other value of n .

11.13 Bode Plot

A Bode diagram is a logarithmically scaled graph of a system transfer function $H(s)$ versus frequency Ω that in reality shows the system frequency response. A Bode plot is a combination of a Bode magnitude plot and a Bode phase plot. A Bode diagram is

implemented with the command `bode`. The syntax is `bode (Hs, w)`, where H_s denotes the transfer function and w is the vector containing the *positive* frequencies for which $H(s)$ is evaluated. The magnitude is given in dB, the phase in degrees, and the frequency in rad/s. The syntax `[mag, phas] = bode (Hs, w)` returns the magnitude and the phase of $H(s)$ evaluated at the frequency points specified in vector w . Alternatively, one may use the command `freqs` in order to obtain the transfer function H evaluated at the points specified in w by executing the command `H = freqs (num, den, w)` and then plot the magnitude by typing `semilogx(w, 20*log10(abs(H)))` and the phase by typing `semilogx(w, angle(H)*180/pi)`.

Example

Create the Bode diagram of the system with transfer function

$$H(s) = \frac{s + 2}{s^2 + 3s + 1}.$$

Commands	Results	Comments
<pre>num = [1 2]; den = [1 3 1]; H = tf(num, den); w = 0:.1:100; bode(H, w)</pre>		Bode diagram of the system. The magnitude is in dB while the phase is in degrees. The frequency axis is in logarithmic scale.
<pre>H = freqs(num, den, w); subplot(211); semilogx(w, 20*log10(abs(H))) title('Bode plot') legend('Magnitude in dB') subplot(212); semilogx(w, angle(H)*180/pi) legend('Phase in degrees') xlabel('Frequency in rad/s')</pre>		Bode plot implemented with use of the command <code>freqs</code> .

11.14 State-Space Representation

Before describing what state representation is, let us introduce some notation that is used in this section. The output signal which is the response of a system is denoted as usual by $y(t)$, but the input signal which is applied to the system will be denoted by $v(t)$ rather than $x(t)$. By $x(t)$, we will denote the *state* of the system. The state $x(t)$ of a system is a column vector called *state vector* and is defined as

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{n-1}(t) \\ x_n(t) \end{bmatrix},$$

where $x_1(t), x_2(t), \dots, x_n(t)$ are called the state variables. The state $x(t_0)$ of the system can be considered as the “stored energy” of the system at time t_0 . Moreover, we denote $\dot{x}_1(t) = dx_1(t)/dt$ and consequently

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_{n-1}(t) \\ \dot{x}_n(t) \end{bmatrix}.$$

In this chapter, a continuous-time system is usually described by a transfer function $H(s)$ of the form

$$H(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_0}{a_n s^n + a_{n-1} s^{n-1} + \dots + a_0},$$

or equivalently by an n th-order differential equation of the form

$$a_n \frac{d^n y(t)}{dt^n} + \dots + a_1 \frac{dy(t)}{dt} + a_0 y(t) = b_m \frac{d^m v(t)}{dt^m} + \dots + b_1 \frac{dv(t)}{dt} + b_0 v(t).$$

In this section, we introduce a new way of describing an LTI system. This new way is called state space modeling, and the system is now described by two equations. The first one is called state (evolution) equation, and its mathematical description is

$$\dot{x}(t) = Ax(t) + Bv(t), \quad (11.13)$$

where $x(t)$ and $\dot{x}(t)$ are column vectors of n elements and A is a matrix of size $n \times n$, is called *transition* matrix.

Supposing that the system under consideration is SISO, the input signal $v(t)$ is 1-D, hence the column vector B is of size $n \times 1$. B is called input or *control* matrix. The second equation that describes the system is called *output* or *observation equation*, and is given by

$$y(t) = Cx(t) + Dv(t), \quad (11.14)$$

where the size of the *output* matrix C is $1 \times n$; that is, C is a column vector and D is called feed-forward matrix. In the SISO case D is a scalar. If the order of the denominator polynomial of the system transfer function is greater than the order of the numerator polynomial of the transfer function, then D is zero. A state space model is defined in MATLAB by the command `ss`. Its syntax is `sys = ss(A, B, C, D)`, where A , B , C , and D are

the matrices of the state space model, and sys is a state-space (SS) object that represents the continuous-time state space model. In order to extract the matrices A , B , C , and D from a previously defined state space model, we can use the command `ssdata`. The syntax is $[A, B, C, D] = \text{ssdata}(sys)$.

Example

Create the state space model described by the equations

$$\dot{x}(t) = \begin{bmatrix} 0.1 & 1 \\ -1.5 & -2 \end{bmatrix}x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix}v(t)$$

and $y(t) = [1 \ 2]x(t) + 0.1v(t)$. Next, verify your result by using the command `ssdata`.

Commands	Results	Comments
<pre> A = [0.1 1; -1.5 -2]; B = [1; 0]; C = [1 2]; D = 0.1; sys = ss(A, B, C, D) </pre>	<pre> a = x1 x2 x1 0.1 1 x2 -1.5 -2 b = u1 x1 1 x2 0 c = x1 x2 y1 1 2 d = u1 y1 0.1 continuous-time model </pre> <pre> A = 0.1000 1.0000 -1.5000 -2.0000 B = 1 0 C = 1 2 D = 0.1000 </pre>	<p>We define the matrices A, B, C, D, and the state space model is created with use of the command <code>ss</code>. The variable sys is an SS object which is a special type of variable.</p> <p>Extraction of the A, B, C, and D matrices of the state model from the SS object sys.</p>

After a state space model of a continuous-time system is defined, one can compute the response of the system to an arbitrary input signal by using the command `lsim`. Moreover, the step response and the impulse response of the system are computed via the commands `step` and `impulse`, respectively. Finally, one useful command when dealing with state space models is the command `initial`. The command `initial` computes the initial condition response. Its syntax is $y = \text{initial}(sys, x_0, t)$, where sys is the system defined with the command `ss` and x_0 is a vector that specifies the initial state of the system.

Example

Compute and plot the response of the system defined in the previous example to the input signal $v(t) = 20te^{-t}$, $0 \leq t \leq 10$. Moreover, compute and plot the step response and the impulse response of the system. Finally, compute and plot the initial state response of the system when the initial state of the system is

$$x_0 = \begin{bmatrix} 10 \\ 20 \end{bmatrix}.$$

Commands	Results/Comments
	The response of the system defined by its state space model is computed with use of the command <code>lsim</code> .
<pre>t = 0:.1:10; v=20*t.*exp(-t); y=lsim(sys,v,t); plot(t,y); title('Output signal y(t)');</pre>	
	Alternative syntax of the command <code>lsim</code> when the system is specified by a state space model. The dotted line represents the input signal, while with the solid line we have plotted the system response.
<pre>lsim(A,B,C,D,v,t)</pre>	
<pre>x0 = [10;20]; lsim(sys,v,t,x0)</pre>	
	The impulse response $h(t)$ of the system is computed with use of the command <code>impulse</code> .
<pre>h=impulse(sys,t); plot(t,h) title('Impulse response h(t)');</pre>	

(continued)

Commands	Results/Comments
	This syntax of <code>impulse</code> also returns the <i>state trajectory</i> . The 2-D state process is plotted in the figure.
<pre>[h,t,x] = impulse(sys); plot(t,x(:,1),t,x(:,2),':') legend('x_1(t)','x_2(t)'); title('State evolution')</pre>	
	The step response $s(t)$ of the system is computed with use of the command <code>step</code> .
<pre>s=step(sys,t); plot(t,s) title('Step response');</pre>	
	The initial state response of the system is the output of the system when no input signal is applied.
<pre>y=initial(sys,x0,t) plot(t,y) title('Initial state response');</pre>	

11.14.1 Construction of a State-Space Model

Although there are several ways to construct a state space model of a system from the system transfer function, we restrict ourselves to the case that a system is described by a strictly proper transfer function. A strictly proper transfer function is of the form

$$H(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_0}, \quad \text{where } m < n.$$

Suppose also that the system is a SISO system. The equivalent state space model is given by

$$\dot{x}(t) = \underbrace{\begin{bmatrix} -a_{n-1} & -a_{n-2} & \cdots & -a_1 & -a_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & 0 & \ddots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}}_A x(t) + \underbrace{\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_B v(t) \quad (11.15)$$

$$y(t) = \underbrace{[0, \dots, 0, b_m, b_{m-1}, \dots, b_0]}_C x(t) + \underbrace{0}_D v(t) \quad (11.16)$$

This realization is a modified version of the *controllable canonical form*. Another common realization is the *observable canonical form*. In MATLAB, it is possible to define a state space model from the system transfer function, and vice versa. This is done with the commands `tf2ss` and `ss2tf`, respectively. The syntax of `tf2ss` is `[A, B, C, D] = tf2ss(num, den)` where *num* and *den* are the coefficients of the numerator and denominator polynomials of the transfer function, respectively, and *A*, *B*, *C*, *D* are the matrices of the state space model given in Equations 11.15 and 11.16.

Example

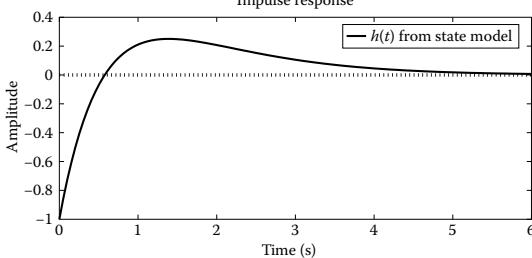
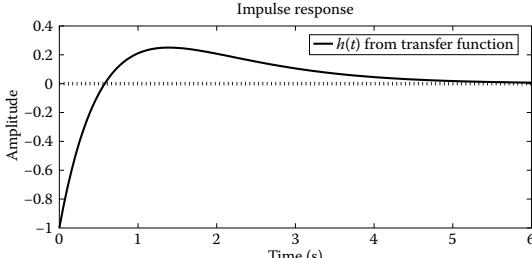
Determine the state space representation of the continuous-time system with transfer function

$$H(s) = \frac{s^2 + 1.5s + 2}{s^2 + 2.5s + 1.5}.$$

Confirm that the two representations are equivalent by computing the impulse response of the system in each case.

Commands	Results/Comments
<pre>num = [1 1.5 2]; den = [1 2.5 1.5]; sys = tf2ss(num, den); [A, B, C, D] = tf2ss(num, den)</pre>	<p>The matrices <i>A</i>, <i>B</i>, <i>C</i>, <i>D</i> of the state space representation are calculated via the <code>tf2ss</code> command. Hence, the equations of the state space model are</p> $\dot{x}(t) = \begin{bmatrix} -2.5 & -1.5 \\ 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} v(t) \text{ and}$ $y(t) = [-1 \ 0.5] x(t) + v(t).$ $\begin{aligned} A &= -2.5000 & -1.5000 \\ & & 1.0000 & 0 \\ B &= 1 \\ & & 0 \\ C &= -1.0000 & 0.5000 \\ D &= 1 \end{aligned}$

(continued)

Commands	Results/Comments
<code>impulse (A, B, C, D); legend('h(t) from state model')</code>	The impulse response of the system is plotted according to the state space representation of the system. 
<code>impulse (num, den) legend('h(t) from transfer function')</code>	The impulse response is plotted according to the transfer function representation of the system. 

In order to compute the transfer function of a SISO system described by a state space model, the statement is `[num, den] = ss2tf (A, B, C, D, 1)`, where the fifth input argument determines which input signal is used for the transfer function calculation. In the SISO case, we have only one input signal; hence, the fifth input argument is 1.

Example

Determine the transfer function $H(s)$ of a system described by the state space model

$$\dot{x}(t) = \begin{bmatrix} -2.5 & -1.5 \\ 1 & 0 \end{bmatrix}x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix}v(t)$$

and $y(t) = [-1 \ 0.5]x(t) + v(t)$.

The matrices A , B , C , and D were already computed in the previous example.

Commands	Results	Comments
<code>[num, den] = ss2tf (A, B, C, D, 1); H = tf (num, den)</code>	Transfer function: $\frac{s^2 + 1.5s + 2}{s^2 + 2.5s + 1.5}$	The transfer function $H(s)$ is same as the one of the previous example, hence our computation is correct.

Two other similar commands are the command `zp2ss` with syntax `[A, B, C, D] = zp2ss (z, p, k)`, which is used to convert a transfer function given in zero/pole/gain form into a state space model; and the command `ss2zp` with syntax `[z, p, k] = ss2zp (A, B, C, D, 1)`, which is used to convert a state space model into a transfer function written in zero/pole/gain form.

11.14.2 Discrete-Time State-Space Models

In this section, we introduce the state space representation of a discrete-time system. The *state evolution equation* of a discrete-time state space model is

$$x[n+1] = Ax[n] + Bv[n], \quad (11.17)$$

where

$x[n]$ is the n -element state vector

$$x[n] = \begin{bmatrix} x_1[n] \\ x_2[n] \\ \vdots \\ x_{n-1}[n] \\ x_n[n] \end{bmatrix}$$

$v[n]$ is the discrete-time input signal

A is the transition matrix

B is the control matrix

The *observation equation* is given by

$$y[n] = Cx[n] + Dv[n] \quad (11.18)$$

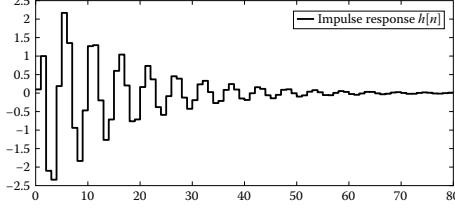
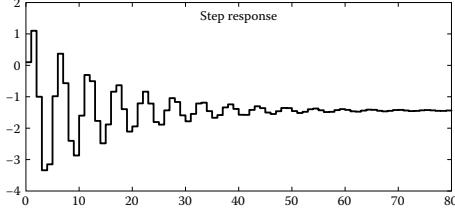
In order to determine a discrete-time state space model, the appropriate syntax of the `ss` command is `sys = ss(A, B, C, D, Ts)`, where A , B , C , and D are the matrices of the state space model and Ts specifies the sampling time. If we set $Ts = -1$, the sampling time is left unspecified. Correspondingly, to the continuous-time case, the discrete-time system transfer function $H(z)$ is calculated from a state space model via the command `ss2tf` by determining the appropriate sampling time. The impulse response of the system is computed by the command `dimpulse`, the step response by the command `dstep` while the system response to an arbitrary discrete-time input signal is calculated with the help of the command `dlsim`.

Example

Create the discrete-time state space model described by the equations

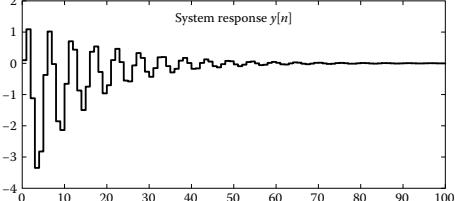
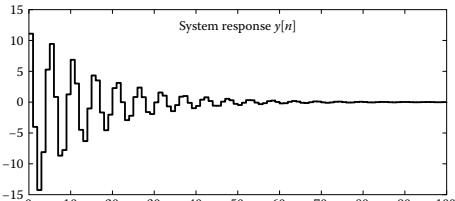
$$x[n+1] = \begin{bmatrix} 0.9 & 0.7 \\ -1.5 & -0.2 \end{bmatrix}x[n] + \begin{bmatrix} 1 \\ 0 \end{bmatrix}v[n]$$

and $y[n] = [1 \ 2]x[n] + 0.1v[n]$. Verify your outcome by using the command `ssdata`. Moreover, compute the transfer function $H(z)$, the impulse response $h[n]$, and the step response $s[n]$ of the discrete-time system. Finally, compute the system response $y[n]$ to the input signal $v[n] = 0.9^n$, $0 \leq n \leq 100$.

Commands	Results	Comments
<pre>A = [.9 .7; -1.5 -.2]; B = [1; 0]; C = [1 2]; D = 0.1; Ts = 0.1; sys = ss(A, B, C, D, Ts)</pre>	<pre>a = x1 x2 x1 0.9 0.7 x2 -1.5 -0.2 b= u1 x1 1 x2 0 c = x1 x2 y1 1 2 d= u1 y1 0.1</pre> <p>Sampling time: 0.1 discrete-time model.</p>	First, we define the matrices A , B , C , D , and the sampling time T_s . The state space model is created with use of the command <code>ss</code> . Indicating a sampling time results in a discrete-time state space model.
<pre>[A, B, C, D] = ssdata(sys)</pre>	<pre>A = 0.9000 0.7000 -1.5000 -0.2000 B = 1 0 C = 1 2 D = 0.1000</pre>	Extraction of the A , B , C , and D matrices of the state model from the SS object <code>sys</code> .
<pre>[num, den] = ss2tf(A, B, C, D, 1) H = tf(num, den, Ts)</pre>	<p>Transfer function:</p> $\frac{0.1z^2 + 0.93z - 2.713}{z^2 - 0.7z + 0.87}$ <p>Sampling time: 0.1</p>	The transfer function $H(z)$ of the discrete-time system. Notice that the sampling time used for the specification of $H(z)$ is same as the one used in the state space model creation.
<pre>n = 0:80; h = dimpulse(A, B, C, D, 1, n); stairs(n, h) legend('Impulse response h[n]')</pre>		Graph of the impulse response $h[n]$ of the system according to its state space representation. Notice that since we deal with a discrete-time system, the appropriate command is the command <code>dimpulse</code> .
<pre>s = dstep(A, B, C, D, 1, n) stairs(n, s) title('Step response')</pre>		Graph of the step response $s[n]$ of the system according to its state space representation with use of the command <code>dstep</code> .

(continued)

(continued)

Commands	Results	Comments
<pre>n = 0:100 v = 0.9.^n y = dlsim(A, B, C, D, v) stairs(n, y) title('System response y[n]')</pre>		The system response $y[n]$ to the input signal $v[n] = 0.9^n$, $0 \leq n \leq 100$ is computed with the command <code>dlsim</code> .
<pre>x0 = [3 4]'; y = dlsim(A, B, C, D, v, x0) stairs(n, y) title('System response y[n]')</pre>		The system response $y[n]$ to the input signal $v[n] = 0.9^n$, $0 \leq n \leq 100$ when the initial state is $x_0 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$.

11.15 Solved Problems

Problem 1 Determine if the LTI system with transfer function

$$H(s) = \frac{H_1(s)}{H_2(s)}$$

is BIBO stable, where

$$H_1(s) = \frac{2s^2 + 1}{s^3 + 3s^2 + 3s + 1}$$

and

$$H_2(s) = \frac{(s+1)(s+2)}{(s+2j)(s-2j)(s+3)}.$$

Solution

```
num1 = [1 1];
num2 = [1 2];
numH2 = conv(num1, num2);
den1 = [1 2j];
den2 = [1 -2j];
den3 = [1 3];
den12 = conv(den1, den2);
denH2 = conv(den12, den3);
```

First, $H_2(s)$ is expressed in rational form by using the command `conv` to execute the multiplications of the polynomials of the numerator and denominator of $H_2(s)$.

(continued)

```

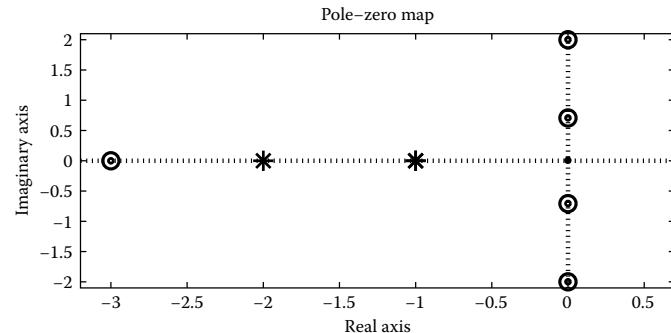
numH1 = [2 0 1];
numH = conv(numH1, denH2);
denH1 = [1 3 3 1];
denH = conv(denH1, numH2);

H=tf(numH, denH);
poles=pole(H);
pzmap(H)

```

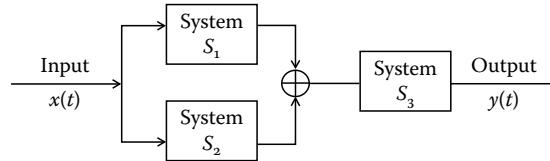
The numerator of $H(s)$ is computed by multiplying (using `conv`) the numerator polynomial of $H_1(s)$ with the denominator polynomial of $H_2(s)$. The corresponding operation is done to obtain the denominator of $H(s)$.

$H(s)$ is defined as a TF object, the poles are computed, and (together with the zeros) are plotted.



The poles are at the left half of the complex plane (as their real part is negative); thus the system is BIBO stable.

Problem 2 Compute the transfer function of the system that is shown in the figure below. The transfer functions of the subsystems are $H_1(s) = s/(s + 4)$, $H_2(s) = s/(s^2 + 4)$, and $H_3(s) = s^2/(s^2 + 4)$.



Solution

```

num1 = [1 0];
den1 = [1 4];
H1 = tf(num1, den1);
num2 = [1 0];
den2 = [1 0 4];
H2 = tf(num2, den2);
H12 = parallel(H1, H2);
% H12 = H1 + H2.

```

```

num3 = [1 0 0];
den3 = [1 0 4];
H3 = tf(num3, den3);
H = series(H12, H3)
% H = H12 * H3

```

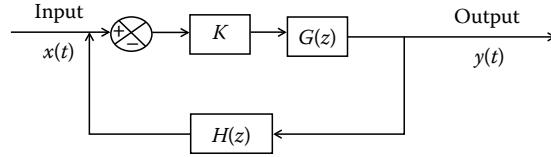
The transfer function of the two parallel-connected subsystems is

$$\text{Transfer function: } \frac{s^3 + s^2 + 8s}{s^3 + 4s^2 + 4s + 16}.$$

The overall system transfer function is

$$\text{Transfer function: } \frac{s^5 + s^4 + 8s^3}{s^5 + 4s^4 + 8s^3 + 32s^2 + 16s + 64}.$$

Problem 3 Compute the closed-loop transfer function of the system that is depicted in the figure below if $G(z) = 0.1/(z - 0.5)$, $H(z) = 0.5/(z - 0.1)$, and $K = 2$.



Solution

```

K=2;
Ts=-1
G=tf(0.1, [1 -0.5], Ts);
H=tf(0.5, [1 -0.1], Ts);
F=feedback(K*G, H)
  
```

Transfer function: $\frac{0.2z - 0.02}{z^2 - 0.6z + 0.15}$.
Sampling time: unspecified.

Problem 4 Compute and plot in the same figure the step responses of two systems with transfer functions

$$H_1(s) = \frac{10}{(s+1)(s+0.2-10i)(s+0.2+10i)}$$

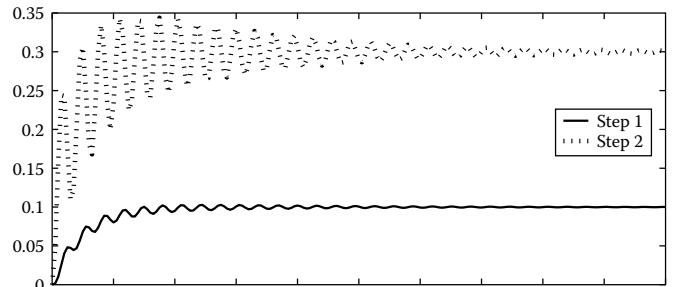
and

$$H_2(s) = \frac{10(s+3)}{(s+1)(s+0.2-10i)(s+0.2+10i)}.$$

Solution

```

z1=[];
p1=[-1 -.2+10j -.2-10j];
k1=10;
H=zpk(z1,p1,k1);
t=0:.1:20
y1=step(H,t)
z2=-3
p2=[-1 -.2+10j -.2-10j];
k2=10;
[num,den]=zp2tf(z2,p2,k2);
y2=step(num,den,t)
plot(t,y1,t,y2,':')
legend('Step1', 'Step2')
  
```



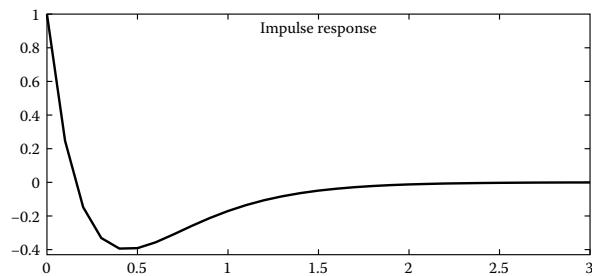
Problem 5 Compute and plot in the time interval $0 \leq t \leq 3$ s the impulse response of a system with transfer function

$$H(s) = \frac{(s+2)(s-3)}{s^3 + 9s^2 + 26s + 24}.$$

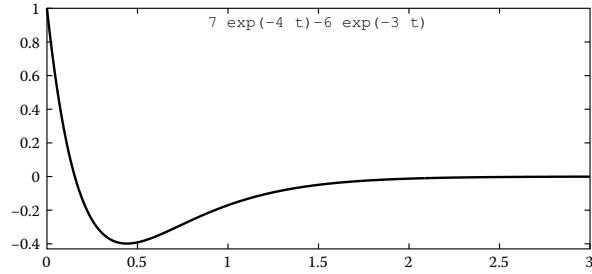
Verify your result by computing the impulse response of the system with use of the inverse Laplace transform.

Solution

```
num=conv([1 2],[1 -3]);
den=[1 9 26 24];
t=0:.1:3;
h=impulse(num,den,t);
plot(t,h)
title('Impulse response')
```



```
syms s
H=(s+2)*(s-3)/(s^3+9*s^2+26*s+24);
h=ilaplace(H)
ezplot(h,[0 3])
ylim([- .4 1]);
```



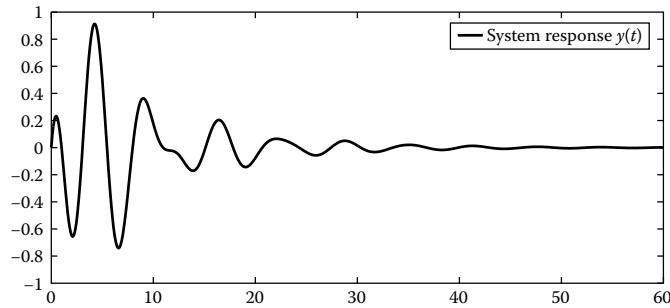
Problem 6 Compute and plot the response of the system with transfer function

$$H(s) = \frac{s^2 - 1}{s^3 + 2s^2 + 3s + 4}$$

to the input signal $x(t) = e^{-0.1t} \cos(t)$, $0 \leq t \leq 60$.

Solution

```
num=[1 0 -1];
den=[1 2 3 4];
H=tf(num,den);
t=0:.1:60;
x=exp(-0.1*t).*cos(t);
y=lsim(H,x,t);
plot(t,y);
legend('System response y(t)');
```



Problem 7 Consider the discrete-time system described by the transfer function

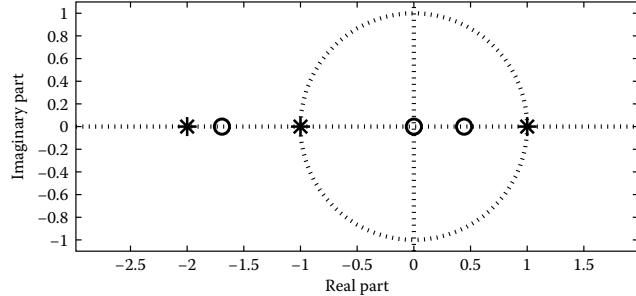
$$H(z) = \frac{8z^2 + 10z - 6}{z^3 + 2z^2 - z - 2}.$$

- a. Find out if the system is stable.
- b. Express the transfer function in zero/pole/gain form.
- c. Express the transfer function in partial fraction form.

Solution

The command `zplane` plots one extra zero at the axis origin when the order of the numerator polynomial is lower than the order of the denominator polynomial.

a. `num = [8 10 -6];
den = [1 2 -1 -2];
zplane (num, den)`



One pole (plotted with the asterisk) is not inside the unit circle, hence the system is not stable.

Zero/pole/gain:

b. `Ts = 0.1;
H = tf (num, den, Ts);
H = zpk (H)`

Sampling time: 0.1

$$R = 2.0000 \quad 2.0000 \quad 4.0000$$

$$P = -2.0000 \quad 1.0000$$

c. `[R, P, K] = residue (num, den)`

$$-1.0000$$

$$K = []$$

The transfer function in partial fraction form is given by

$$H(z) = \frac{2}{z+2} + \frac{2}{z-1} + \frac{4}{z+1}.$$

Problem 8 Consider a discrete-time system with transfer function

$$H_1(z) = \frac{z^2}{z^2 + 0.2z + 0.01}.$$

- a. Find the transfer function $H_2(z)$ of a system with the same behavior that causes a delay of 2 units to an applied input signal. Also, plot for $0 \leq n \leq 8$.
- b. The impulse response of the system with transfer function $H_1(z)$.
- c. The impulse response of the system with transfer function $H_2(z)$.

Solution

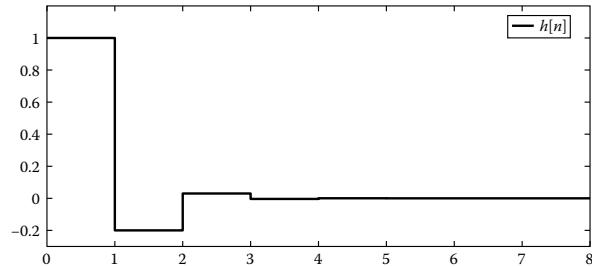
```
a. num = [1 0 0];
den = [1 .2 .01];
Ts = 0.1;
H1 = tf(num, den, Ts);
H2 = tf(num, den, Ts, 'inputdelay', 2)
```

Transfer function:

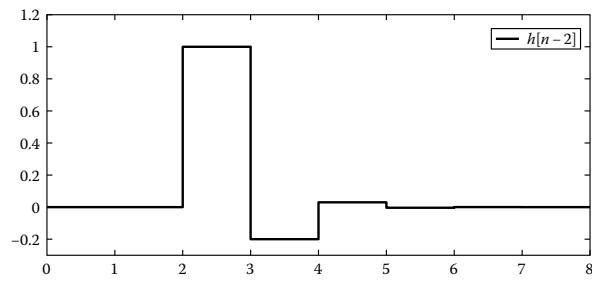
$$z^{-2} * \frac{z^2}{z^2 + 0.2z + 0.01}.$$

Sampling time: 0.1

```
b. n = 0 : 8;
h = dimpulse(num, den, n);
stairs(n, h)
legend('h[n]')
```



```
c. num = 1;
h2 = dimpulse(num, den, n);
stairs(n, h2)
legend('h[n-2]')
```



Problem 9 Consider the system described by the difference equation $y[n] = 1.6y[n - 1] - 0.8y[n - 2] + 0.01x[n] + 0.03x[n - 1] + 0.015x[n - 2]$ with zero initial conditions. Compute and plot the response $y[n]$ of the system to the input signal $x[n]$ when

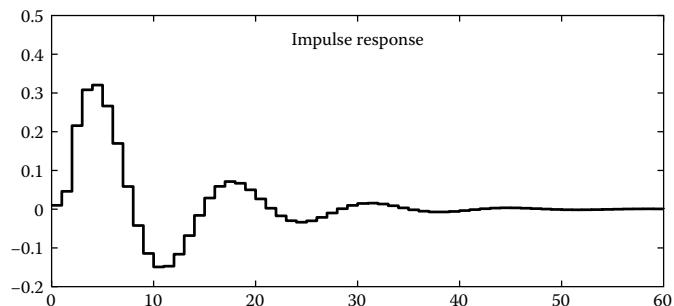
- a. $x[n]$ is the unit impulse sequence $\delta[n]$.
- b. $x[n]$ is the unit step sequence $u[n]$.
- c. $x[n]$ is the unit ramp sequence $r[n]$ (in two ways).

Solution

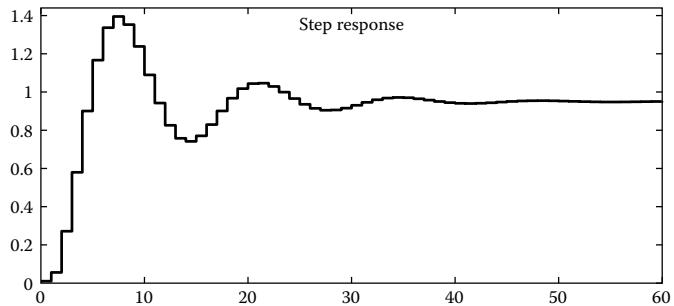
```
num = [.01 0.03 0.15];
den = [1 -1.6 0.8];
printsys(num, den, 'z')
```

The system transfer function $H(z)$ is $\frac{0.01z^2 + 0.03z + 0.15}{z^2 - 1.6z + 0.8}$.

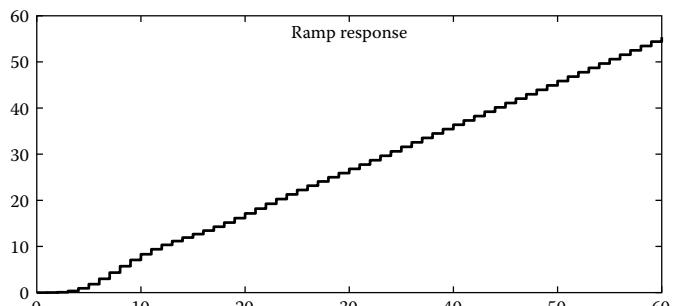
```
a. num = [.01 0.03 0.15];
den = [1 -1.6 0.8];
k = 0:60;
y1 = dimpulse(num, den, k);
stairs(k, y1)
title('Impulse Response')
```



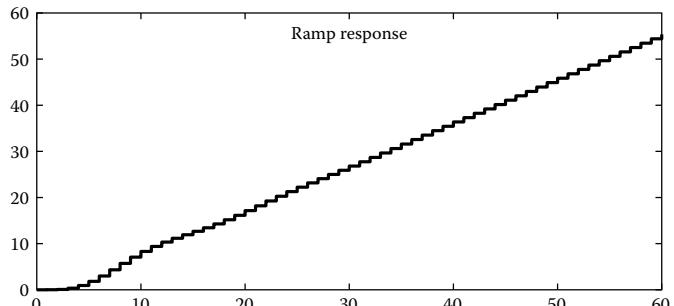
```
b. y2 = dstep(num, den, k);
stairs(k, y2)
title('Step Response')
```



```
c1. ramp = k;
y31 = filter(num, den, ramp);
stairs(k, y31);
title('Ramp Response');
```



```
c2. y32 = dlsim(num, den, ramp);
stairs(k, y32);
title('Ramp Response');
```



Problem 10 Consider a discrete-time system with frequency response

$$H(\omega) = \frac{0.1 + 0.1e^{-j\omega} + 0.18e^{-2j\omega} + 0.18e^{-3j\omega} + 0.09e^{-4j\omega} + 0.09e^{-5j\omega}}{1 - 1.5e^{-j\omega} + 2.2e^{-2j\omega} - 1.5e^{-3j\omega} + 0.8e^{-4j\omega} - 0.18e^{-5j\omega}}.$$

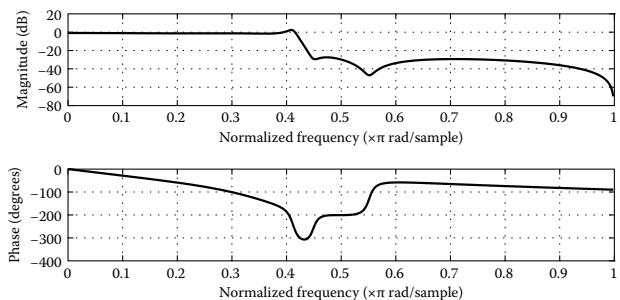
- a. Plot the frequency response of the system.
- b. Compute and plot the impulse response and the step response of the system.

Solution

The frequency response of the discrete-time system.

a.

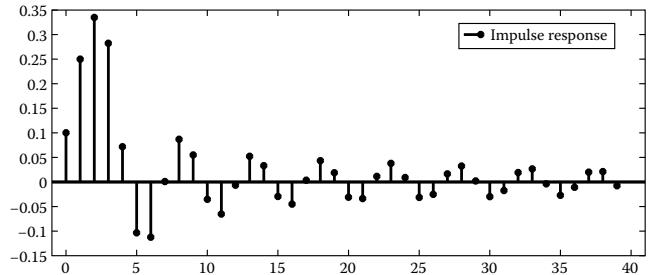
```
num = [.1 .1 .18 .18 .09 .09];
den = [1 -1.5 2.2 -1.5 0.8 -0.18];
freqz(num,den)
```



b. The system transfer function $H(z) = \frac{0.1z^5+0.1z^4+0.18z^3+0.18z^2+0.09z+0.09}{z^5-1.5z^4+2.2z^3-1.5z^2+0.8z-0.18}$ is derived from the frequency response $H(\omega)$ by substituting $e^{j\omega}$ with z and multiplying the numerator and the denominator by z^5 . Now, the impulse response and the step response of the system can be computed by the commands filter, dstep, and dimpulse.

Graph of the system impulse response obtained with the command filter.

```
delta= [1 zeros(1,39)];
h=filter(num,den,delta);
stem(0:39,h)
legend('Impulse response')
```

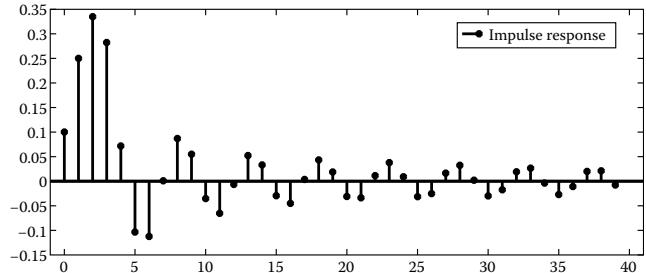


(continued)

(continued)

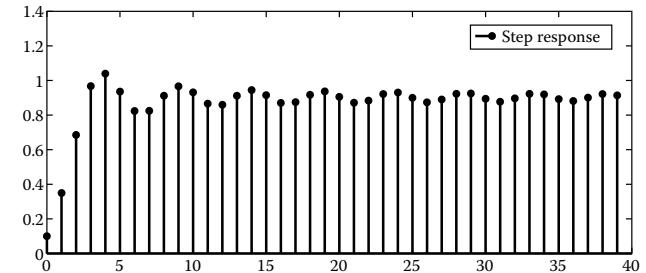
Graph of the system impulse response obtained with the command `dimpulse`.

```
h2=dimpulse(num,den,40)
stem(0:39,h2)
legend('Impulse response')
```



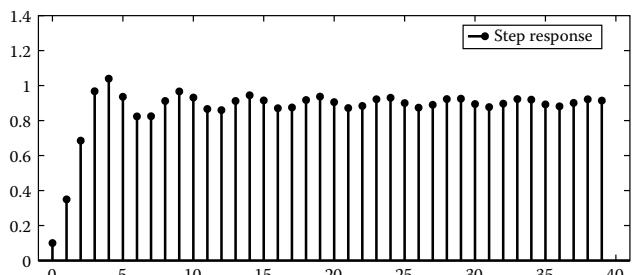
Graph of the system step response obtained with the command `filter`.

```
u=ones(1,40);
s=filter(num,den,u);
stem(0:39,s)
legend('Step response')
```



Graph of the system step response obtained with the command `dstep`.

```
s2=dstep(num,den,40);
stem(0:39,s2);
legend('Step response')
```



Problem 11 Determine the state space representation of the continuous-time system with transfer function

$$H(s) = \frac{s^2 + 2}{s^3 + s^2 + 4s + 2}.$$

Compute and plot in the time interval $0 \leq t \leq 20$ the impulse response and the step response of the system according to its state representation. Also, calculate the system response to the input signal

$$v(t) = \begin{cases} 2t, & 0 \leq t \leq 2 \\ 8 - 2t, & 2 < t \leq 4 \end{cases}$$

Finally, compute and plot the state process when the applied input signal is the Dirac function $\delta(t)$ and when the applied input signal is $v(t)$.

Solution

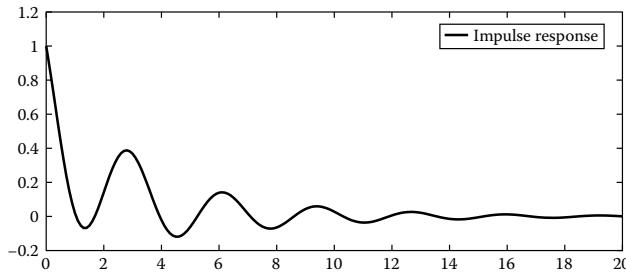
```
A = -1      -4      -2
      1       0       0
      0       1       0
B = 1
      0
      0
C = 1      0       2
      D = 0
```

The SS model of the system is

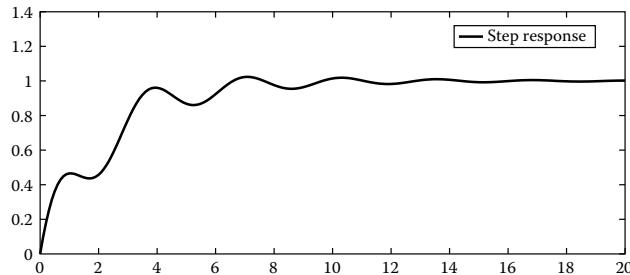
$$\dot{x}(t) = \begin{bmatrix} -1 & -4 & -2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} v(t)$$

$$y(t) = [1 \ 0 \ 2] v(t)$$

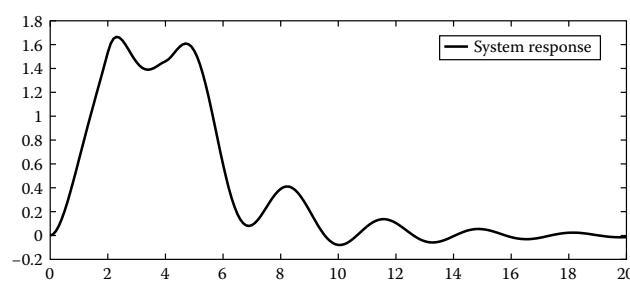
```
sys = ss(A, B, C, D);
t = 0:.1:20;
h = impulse(sys, t);
plot(t, h)
legend('Impulse response')
```



```
s = step(sys, t);
plot(t, s)
legend('Step response')
```



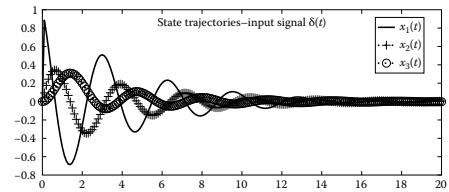
```
t1 = 0:.1:2;
t2 = 2.1:.1:4;
t3 = 4.1:.1:20;
t = [t1 t2 t3];
v1 = 2*t1;
v2 = 8-2*t2;
v3 = zeros(size(t3));
v = [v1 v2 v3];
y = lsim(sys, v, t);
plot(t, y);
legend('System response')
```



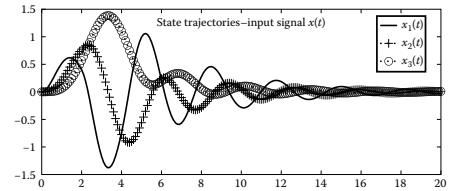
(continued)

(continued)

```
[y,t,x] = impulse(sys,t);
plot(t,x(:,1),t,x(:,2),'+',t,x(:,3),'o')
legend('x_1(t)','x_2(t)','x_3(t)')
title('State trajectories-input signal \delta(t)')
```



```
[y,t,x] = lsim(sys,v,t)
plot(t,x(:,1),t,x(:,2),'+',t,x(:,3),'o')
legend('x_1(t)','x_2(t)','x_3(t)')
title('State trajectories -input signal x(t) ')
```



Notice that the state of the system depends on the applied input signal.

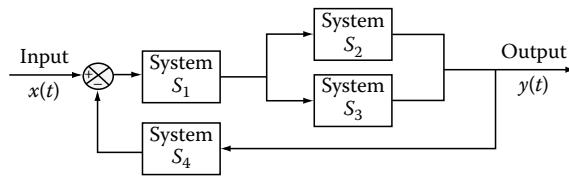
11.16 Homework Problems

1. Suppose that a continuous-time system is described by the impulse response $h(t) = te^{-t} \cos(t)u(t)$.
 - a. Compute the transfer function of the system.
 - b. Find out if the system is BIBO stable.
 - c. Plot the poles and the zeros of the system.
 - d. Write the transfer function in zero/pole/gain form.
2. Compute the transfer function of the systems described by the differential equations
 - a. $y''(t) + y(t) = x(t)$
 - b. $y''(t) + y(t) = x'(t) - x(t)$

Assume zero initial conditions for both differential equations.

3. When the input signal $x(t) = e^{-t} u(t)$ is applied to a system, the system response is $y(t) = te^{-t}u(t)$. Compute the system transfer function.
4. Compute the transfer function of a system that consists of three parallel-connected subsystems S_1 , S_2 , and S_3 with transfer functions $H_1(s) = \frac{3}{s+4}$, $H_2(s) = \frac{2}{s^2-2}$, and $H_3(s) = \frac{s+1}{s+5}$, respectively.

5. Compute the transfer function of the system that is depicted below. The transfer functions of the subsystems are $H_1(s) = \frac{1}{s+2}$, $H_2(s) = \frac{s+1}{s^2-2}$, $H_3(s) = \frac{2}{s+5}$, and $H_4(s) = \frac{s}{s^3+2s+1}$.



6. Suppose that a system is specified by the transfer function $H(s) = \frac{30s}{s^2 + 3s + 1}$. Compute and plot
- The step response of the system
 - The impulse response of the system
 - The response of the system to the input signal $x(t) = e^{-0.2t} \cos(t)$, $0 \leq t \leq 30$
7. Suppose that a discrete-time system is described by the impulse response $h[n] = 0.8^n u[n] + 0.9^n u[n]$.
- Compute the transfer function of the system.
 - Find out if the system is BIBO stable.
8. Compute the transfer function of the discrete-time system described by the difference equation $y[n] + y[n-1] - 0.3 y[n-2] = 2x[n]$. Assume that $y[n] = x[n] = 0$, $n < 0$.
9. Suppose that a discrete-time system is specified by the transfer function $H(z) = \frac{z}{-z^2 + 0.2z - 0.1}$. Compute and plot
- The step response of the system
 - The impulse response of the system
 - The response of the system to the input signal $x[n] = \frac{1}{n+1}$, $0 \leq n \leq 20$
10. Discretize the continuous-time system with transfer function $H(s) = (s+10)/s$ with use of the bilinear (Tustin) approximation method. Verify your result by converting the computed discrete-time system to continuous-time system.
11. Compute and plot the frequency response of the continuous-time system with transfer function $H(s) = \frac{s^2 + 3s + 5}{s^3 + 3s}$.
12. Compute and plot the frequency response of the discrete-time system with transfer function $H(z) = \frac{z}{z - 0.8}$.
13. Create the Bode diagram of the system with transfer function $H(s) = \frac{s}{s^2 + 1}$.

14. Suppose that a system is described by the state space model
 $\dot{x}(t) = \begin{bmatrix} 0.3 & 0.6 \\ -2.5 & -1 \end{bmatrix}x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix}v(t)$ and $y(t) = [1 \ 3]x(t) + 0.1 v(t)$.
- Compute and plot the impulse response of the system.
 - Compute and plot the step response of the system.
 - Compute and plot the response of the system to the input signal $v(t) = -3t^3e^{-t}$, $0 \leq t \leq 20$.
 - Compute and plot the response of the system to the input signal $v(t) = -3t^3e^{-t}$, $0 \leq t \leq 20$, when the initial state of the system is $x_0 = \begin{bmatrix} 5 \\ 10 \end{bmatrix}$.
 - Compute and plot the state trajectories for the four previous cases.
 - Calculate the system transfer function.
15. Suppose that a discrete-time system is described by the state space model
 $x[n+1] = \begin{bmatrix} -0.1 & 0.2 \\ 1 & 0 \end{bmatrix}x[n] + \begin{bmatrix} 1 \\ 0 \end{bmatrix}v(t)$ and $y[n] = [1 \ -3]x[n]$.
- Compute and plot the impulse response of the system.
 - Compute and plot the step response of the system.
 - Compute and plot the response of the system to the input signal
- $$v[n] = \frac{-3}{\sqrt{1+n}}, 0 \leq n \leq 20$$
- Compute and plot the response of the system to the input signal
- $$v[n] = \frac{-3}{\sqrt{1+n}}, 0 \leq n \leq 20$$
- when the initial state of the system is $x_0 = \begin{bmatrix} -5 \\ -10 \end{bmatrix}$.
- Compute and plot the state trajectories for the four previous cases.
 - Calculate the system transfer function.
16. Compute and plot for $0 \leq \Omega \leq 10$ rad/s the frequency response of the system described by the state space model $\dot{x}(t) = \begin{bmatrix} 0.2 & 1.1 \\ -1.5 & -2 \end{bmatrix}x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix}v(t)$ and $y(t) = [1 \ 1.8]x(t) + 0.1 v(t)$.
17. Compute and plot for $0 \leq \omega \leq 2\pi$ rad/s the frequency response of the discrete-time system described by the state space model $x[n+1] = \begin{bmatrix} 0.2 & 1.1 \\ -1.5 & -2 \end{bmatrix}x[n] + \begin{bmatrix} 1 \\ 0 \end{bmatrix}v[n]$ and $y[n] = [1 \ 1.8]x[n] + 0.1 v[n]$.

18. Determine the state space model of the system with transfer function $H(s) = \frac{3(s+2)}{(s+1)(s+3)(s+4)}$. Moreover,
- Compute and plot the impulse response of the system.
 - Compute and plot the step response of the system.
 - Compute and plot the response of the system to the input signal $v(t) = te^{-t}$, $0 \leq t \leq 10$.
 - Compute the initial state response when the initial state is $x_0 = \begin{bmatrix} 12 \\ -10 \\ -5 \end{bmatrix}$.
 - Compute and plot the state trajectories for the four previous cases.
19. Create the Bode diagram of the system described by the state space model $\dot{x}(t) = \begin{bmatrix} 0.2 & 1.1 \\ 1 & 0 \end{bmatrix}x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix}v(t)$ and $y(t) = [1 \ -1] x(t)$.

This page intentionally left blank

12

Suggested Laboratory Exercises

In this chapter, the authors propose eleven laboratory exercises that can be used in one semester laboratory course on signals and systems. The laboratory exercises consist entirely of problems as the necessary theory is already introduced in the main part of the book. We suppose that students have no previous experience in MATLAB®, and this is why the title of the first laboratory exercise is as follows.

12.1 Laboratory 1: Introduction to MATLAB

1. Compute the outcome of the expression $2 \cos^2(\pi) + \sqrt{4 \sin\left(\frac{\pi}{2}\right)} - e^2$.
2. Create a row vector a with elements $[1, 2, \dots, 5]$ and a row vector b with elements $[2^{-1}, 2^{-2}, \dots, 2^{-5}]$.
 - 2.1 Concatenate them and compute the length of the new vector.
 - 2.2 Compute the product of a and b .
 - 2.3 Convert vector b into a column vector and store the column vector into a new vector c .
 - 2.4 Create a row vector x with elements $[0, 1, \dots, 100]$ and a row vector y with elements $[0, -0.01, \dots, -1]$.
 - 2.5 Compute the dot product of x and y .
 - 2.6 Calculate the sum of the elements of x .
3. Create the matrices $\mathbf{A} = \begin{bmatrix} 12 & 3 & -6 \\ 2 & 8 & 11 \\ 2 & 1 & 1 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$.
 - 3.1 Compute the size of each matrix.
 - 3.2 Find the transpose matrix of \mathbf{B} . Store the transpose matrix into a matrix \mathbf{C} .
 - 3.3 Create a matrix \mathbf{D} by placing the elements of \mathbf{C} next to the elements of \mathbf{A} .
 - 3.4 Export the second row of \mathbf{A} .
 - 3.5 Export the first and the third columns of \mathbf{A} .
 - 3.6 Export the second and the third rows of \mathbf{A} .
 - 3.7 Compute the determinant of \mathbf{A} .
 - 3.8 Find the inverse matrix of \mathbf{A} . Verify that the computed matrix is indeed the inverse matrix of \mathbf{A} .
 - 3.9 Compute the element per element product between \mathbf{A} and the inverse matrix of \mathbf{A} .
 - 3.10 Find \mathbf{A}^2 .

4. Plot the function $f(x) = \cos(2\pi x)$, $0 \leq x \leq 5$.
 - 4.1 Insert title, grid, and legend in the graph.
 - 4.2 Plot in the same figure the function $g(x) = xe^{-x}$, $0 \leq x \leq 5$.
 - 4.3 Plot again the same functions, but this time in two different subfigures of the same figure.
 5. Define the complex numbers $z_1 = 3 + 4i$ and $z_2 = 5 - 4i$. Find out
 - 5.1 The real and the imaginary parts of z_1 .
 - 5.2 The magnitude and the phase of z_2 .
 - 5.3 The sum of z_1 and z_2 .
 - 5.4 Plot z_1 in the complex plane.
 6. Create a script (M-File) that computes the sum of the squares of numbers 1 through 10.
 7. Create a function that accepts a number b as input argument and generates the graph of the function $y(t) = te^{-bt}$, $0 \leq t \leq 5$. Vector y must be returned as an output from the function.
 8. Compute the first and the second derivative of $x(t) = \cos(t)$.
 9. Calculate the integrals $\int_{-1}^1 t^3 dt$ and $\int_{-\infty}^{\infty} e^{-t^2} dt$
-

12.2 Laboratory 2: Signals

1. Define and plot
 - 1.1 The continuous-time signal $x_1(t) = \cos(\pi t)$
 - 1.2 The discrete-time signal $x_2[n] = \cos(\pi n)$
 - 1.3 The real part and the imaginary part of the signal $x(t) = e^{3jt}$ in time of two periods
2. Plot in the time interval $-5 \leq t \leq 10$ the following signals:
 - 2.1 $u(t)$
 - 2.2 $u(t+3)$
 - 2.3 $u(t-2) - u(t+3)$
 - 2.4 $e^{-t}u(t+2)$
 - 2.5 $\delta(t)$
 - 2.6 $\delta(t-2)$
 - 2.7 $\delta(t-1) + \delta(t+2)$
 - 2.8 $r(t)$
 - 2.9 $r(t-1)$
 - 2.10 $r(t) - r(t-2) - r(t-5) + r(t-7)$
 - 2.11 $p4(t)$
 - 2.12 $p2(t-1)$

3. Tell if the signal $x(t) = t \cos(t)$ is symmetric, and if yes what kind of symmetry. Also, analyze $x(t)$ in even and odd parts and confirm your result.
 4. Determine if the signal $x(t) = t[u(t) - u(t - 8)]$ is an energy or power signal.
 5. Consider the signal $x(t) = t[u(t) - u(t - 5)]$. Plot the following:
 - 5.1 $x(-t)$
 - 5.2 $x(4t)$
 - 5.3 $x(0.3t)$
 - 5.4 $x(t + 2)$
 - 5.5 $x(t - 3)$
 - 5.6 $x(1 - 3t)$
-

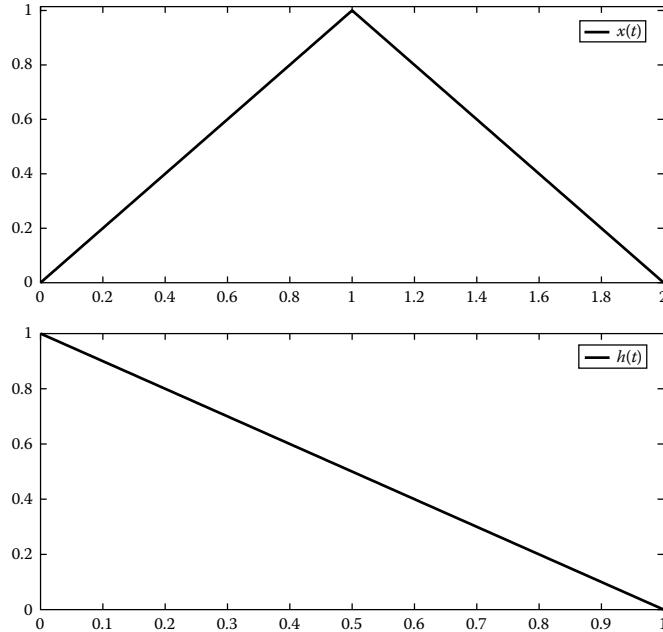
12.3 Laboratory 3: Systems

1. A differentiator system is a system in which the output is proportional to the time derivative of the input. So consider a differentiator system described by the i/o relationship $y(t) = dx(t)/dt$, where $x(t)$ is the applied to the system input signal and $y(t)$ is the output signal of the system. Determine if this system is
 - 1.1 Linear or nonlinear.
 - 1.2 Causal or non-causal.
 - 1.3 Static or dynamic.
 - 1.4 Time invariant or time variant.
 - 1.5 Stable or unstable.
 - 1.6 Invertible or not invertible. If your answer is invertible, try to define the inverse system.
 2. A discrete-time system is described by the i/o relationship $y[n] = x[n] - x[n - 1]$. Determine if the system is
 - 2.1 Linear or nonlinear
 - 2.2 Causal or non-causal
 - 2.3 Static or dynamic
 - 2.4 Shift invariant or shift variant
-

12.4 Laboratory 4: Time Domain System Analysis

1. Suppose that a system is described by the impulse response $h(t) = \sin(3\pi t)[u(t) - u(t - 5)]$. Compute and plot the system response to the input signal $x(t) = \cos(2\pi t)[u(t) - u(t - 5)]$.
2. Compute and plot the convolution between the signals $x(t) = \cos(\pi t)[u(t + 2) - u(t - 3)]$ and $h(t) = 5e^{-t} u(t)$.

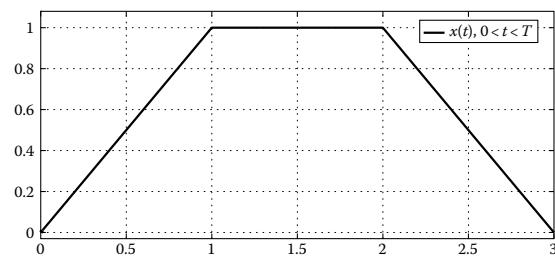
3. Compute and plot the convolution between the signals that are depicted in the figure below. Also, plot the two signals during the various stages of the convolution procedure.



4. Consider the system described by the difference equation $y[n] + 0.1y[n - 1] - y[n - 2] = x[n] + 2x[n - 1]$ and assume zero initial conditions. Compute and plot for $0 \leq n \leq 30$.
- 4.1 The impulse response $h[n]$ of the system.
 - 4.2 The step response $r[n]$ of the system.
 - 4.3 Finally, compute and plot the response $y[n]$ of the system to the input signal $x[n] = 0.8^n \cos(n)$, $0 \leq n \leq 50$.

12.5 Laboratory 5: Fourier Series

1. The periodic signal plotted in the figure on the right in one period is given by $x(t) = \begin{cases} t, & 0 \leq t \leq 1 \\ 1, & 1 \leq t \leq 2 \\ 3 - t, & 2 \leq t \leq 3 \end{cases}$



Compute and plot in one period the approximations of $x(t)$ using

- a. 3, 11, and 51 terms of the complex exponential Fourier series.
 - b. 3, 11, and 51 terms of the trigonometric Fourier series.
 - c. 3, 11, and 51 terms of the Fourier series in cosine with phase form.
 - d. Each time also plot the corresponding Fourier series coefficients.
 - e. Each time compute the approximation percentage.
 - f. Plot the line spectra.
 - g. Finally, plot the three “good” approximations in time of five periods.
2. Repeat the queries of the previous exercise for the periodic signal that in one period is given by

$$x(t) = \begin{cases} 2t, & 0 \leq t \leq 1 \\ 0, & 1 < t < 2 \end{cases} .$$

12.6 Laboratory 6: Fourier Transform

1. Compute and plot the Fourier transforms of the following functions:
 - 1.1 $\delta(t)$
 - 1.2 $u(t)$
 - 1.3 e^{j3t}
 - 1.4 $\sin(2\pi t)$
2. Compute and plot the inverse Fourier transforms of the following functions:
 - 2.1 1
 - 2.2 $e^{-4j\Omega}$
 - 2.3 $\pi\delta(\Omega - 2) + \pi\delta(\Omega + 2)$
 - 2.4 $1/(j\Omega + 3)$
3. Use the Fourier transform to compute (and plot) the convolution between the signals
 - 3.1 $x(t) = 2e^{-t}u(t)$ and $h(t) = te^{-t}u(t)$.
 - 3.2 $x(t) = u(t)$ and $h(t) = e^{-t}u(t)$.
 - 3.3 Confirm your result by computing the convolution of $x(t)$ and $h(t)$ with use of the command `conv`. Which is the assumption that must be made for the confirmation of the second case?
4. Compute the energy of the signal $x(t) = te^{-t}u(t)$ in the time domain and in the frequency domain.
5. Consider the signal $x(t) = 5^{-t}u(t)$
 - 5.1 Compute and plot the autocorrelation function of $x(t)$.
 - 5.2 Compute the energy of $x(t)$ via its autocorrelation function.

12.7 Laboratory 7: Fourier Analysis of Discrete-Time Systems

1. Compute the DTFTs $X_1(\omega)$ and $X_2(\omega)$ of the signals $x_1[n] = 0.6^n \cos(n)$, $0 \leq n \leq 3$ and $x_2[n] = 0.6^n \cos(n)u[n]$, respectively. Plot $X_1(\omega)$ and $X_2(\omega)$ over the frequency intervals $-\pi \leq \omega \leq \pi$ and $-4\pi \leq \omega \leq 4\pi$.
2. Compute the DFT X_k of the signal $x[n] = 0.9^n \sin(n)$, $0 \leq n \leq 10$. Plot the real part, the imaginary part, the magnitude, and the phase of X_k .
3. Consider the discrete-time signal $x[n] = 1/(n+1)$, $0 \leq n \leq 30$. Plot the DTFT $X(\omega)$ and the DFT X_k of $x[n]$ in the same figure in the frequency interval $-\pi \leq \omega \leq \pi$.
4. Consider the continuous-time signal $x(t) = u(t) - u(t-3)$. Approximate the Fourier transform $X(\Omega)$ of $x(t)$ by using the FFT algorithm. Plot $X(\Omega)$ and the derived approximation in the same figure. *Tip:* Try to choose the correct parameters N and T for the sampling of $x(t)$.
5. Plot the circular, shifted by 3 units version of the sequence $x[n] = n/(n+1)$, $0 \leq n \leq 10$.
6. Compute and plot the circular convolution of the discrete-time signals $x[n] = 0.3^n$, $0 \leq n \leq 6$ and $h[n] = 3^{-n}$, $0 \leq n \leq 6$.
7. Compute and plot the circular convolution of the discrete-time signals $x[n] = 0.3^n$, $0 \leq n \leq 6$ and $h[n] = 3^{-n}$, $0 \leq n \leq 10$.
8. Calculate the linear convolution of the sequences $x[n] = n^2$, $0 \leq n \leq 5$ and $h[n] = 0.8^n$, $0 \leq n \leq 8$.
 - 8.1 With use of circular convolution
 - 8.2 With use of the FFT algorithm
 - 8.3 With use of the command `conv`

12.8 Laboratory 8: Frequency Response

1. Compute and plot the frequency response of a system described by the impulse response $h(t) = (t^3/6) e^{-2t} u(t)$.
2. Plot the frequency response $H(j\Omega) = \frac{(j\Omega)^3 - 2(j\Omega)^2 + 3j\Omega - 4}{j\Omega^3 + \Omega^2 + 2\Omega}$.
3. Consider a system described by the frequency response $H(j\Omega) = \frac{(j\Omega)^2 + 2j\Omega + 3}{-\Omega^2 + j\Omega + 2}$.
Compute and plot the system response to the input signal $x(t) = 3 \sin(2\pi t + \pi/3)$
 - 3.1 According to the relationship that describes the system response to sinusoidal inputs
 - 3.2 Using the command `lsim`
4. Consider an ideal high-pass filter that cuts the input signal for frequencies $\Omega < 10$ rad/s.
 - 4.1 Plot the impulse response of the filter.

4.2 Compute and plot the system response to the input signal

$$x(t) = 6 \sin\left(t + \frac{\pi}{4}\right) + 8 \cos\left(12t + \frac{\pi}{6}\right).$$

5. Consider a discrete-time system with impulse response $h[n] = 0.8^n u[n] + 0.7^n \cos(n)u[n]$. Compute and plot the frequency response of the system.
 6. Suppose that a discrete-time system is described by the frequency response

$$H(\omega) = \frac{4 + 2e^{-2j\omega}}{3 + 5e^{-j\omega}}.$$

6.1 Plot the frequency response $H(\omega)$.

6.2 Compute and plot the system response to the discrete-time input signal $x[n] = 2 \cos(4n + (\pi/6))$.

12.9 Laboratory 9: Laplace Transform

1. Compute the unilateral Laplace of the following functions:

- 1.1 $\delta(t)$
- 1.2 $u(t)$
- 1.3 $e^{-3t}u(t)$
- 1.4 $\cos(2\pi t)u(t)$

2. Compute the inverse Laplace transforms of the functions

- 2.1 1
- 2.2 $1/s$
- 2.3 $1/(s+3)$
- 2.4 $1/(s-4j)$

3. Express in partial fraction form the functions

$$3.1 X_1(s) = \frac{s^3 - 7s^2 + 14s + 8}{s^3 - 8s^2 + 11s + 20}$$

$$3.2 X_2(s) = \frac{s^3 + s + 8}{s^2 + 2s + 1}$$

4. Find and plot the solution with use of Laplace transform of the following differential equations:

$$4.1 y''(t) + y'(t) + y(t) = \cos(2\pi t), y(0) = 1, y'(0) = -1$$

$$4.2 y''(t) - y(t) = 2u(t), y(0) = 1, y'(0) = -1$$

$$4.3 2y''(t) + 3y'(t) + y(t) = x(t), y(0) = 2, y'(0) = 4, \text{ and}$$

$$x(t) = \begin{cases} -1, & 0 < t < 1 \\ 1, & t \geq 1 \end{cases}$$

12.10 Laboratory 10: z-Transform

1. Compute the z -transforms of the sequences

$$1.1 \quad x[n] = [5, 4, 3, 2, 1], 0 \leq n \leq 4$$

$$1.2 \quad x[n] = [5, 4, 3, 2, 1], -2 \leq n \leq 2$$

$$1.3 \quad x[n] = \frac{1}{n} u[n - 1]$$

$$1.4 \quad x[n] = \frac{1}{n+3} u[n + 2]$$

2. Compute the unilateral z -transforms of the sequences

$$2.1 \quad x[n] = 2^n$$

$$2.2 \quad x[n] = 2^n \cos(n)$$

$$2.3 \quad x[n] = 2^n \sin(n)$$

$$2.4 \quad x[n] = 2^n e^{2jn}$$

3. Compute the inverse z -transforms of the functions

$$3.1 \quad 1$$

$$3.2 \quad \frac{z}{z - 1}$$

$$3.3 \quad \frac{z}{(z - 1)^2}$$

$$3.4 \quad \frac{3z}{(z - 3)^2}$$

4. Express in partial fraction form the functions

$$4.1 \quad X_1(z) = \frac{z^2 - 8}{z^3 - 8z^2 + 5z + 2}$$

$$4.2 \quad X_2(z) = \frac{1 - 8z^{-2}}{1 - 8z^{-1} + 5z^{-2} + 2z^{-3}}$$

$$4.3 \quad X_3(z) = \frac{z^{-1} - 8z^{-3}}{1 - 8z^{-1} + 5z^{-2} + 2z^{-3}}$$

4.4 Verify your three previous answers by expressing the derived functions in rational form.

5. Suppose that a system is described by the difference equation $y[n] + y[n - 1] - y[n - 2] - y[n - 3] = 0.9^n u[n]$ with zero initial conditions.

5.1 Compute with use of z -transform the solution of the difference equation.

5.2 Plot the solution for $0 \leq n \leq 30$.

Also, verify your answer

5.3 By examining if the derived solution satisfies the difference equation

5.4 By computing and plotting the solution with use of the command `filter`.

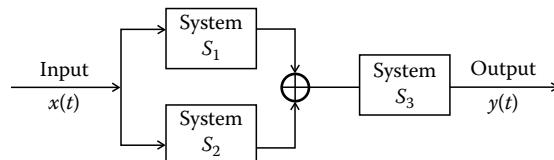
12.11 Laboratory 11: Transfer Function

1. A system is described by the impulse response $h(t) = t \cos(\pi t)u(t)$. Compute the transfer function $H(s)$ of the system.
2. A system is described by the differential equation $y''(t) - 3y'(t) + 2y(t) = x'(t) - 2x(t)$, $y'(0) = y(0) = x'(0) = x(0) = 0$. Compute the system transfer function $H(s)$.
3. Consider a system S_1 with transfer function

$$H_1(s) = \frac{3s^2 + 2}{s^2 + 2s + 6}.$$

- 3.1 Determine if the system is BIBO stable.
- 3.2 Write $H_1(s)$ in zero/pole/gain form.
- 3.3 Plot the zeros and the poles of the system in the complex plane.
- 3.4 Find the transfer function $H_2(s)$ of a system S_2 with the same behavior to S_1 that causes a delay of four units to an applied input signal. Determine if the new system is BIBO stable.
- 3.5 Confirm you result by plotting the impulse responses of the systems S_1 and S_2 .

4. Suppose that the subsystems S_1 , S_2 , and S_3 with transfers functions $H_1(s) = \frac{2}{s+2}$, $H_2(s) = \frac{s}{s+1}$, and $H_3(s) = \frac{s+1}{s+2}$, respectively, are connected as shown in the figure on the right.



- 4.1 Compute the transfer function $H(s)$ of the overall system.
- 4.2 Compute and plot the impulse response $h(t)$ of the system.
- 4.3 Compute and plot the step response $s(t)$ of the system.
- 4.4 Compute and plot the response $y(t)$ of the system to the input signal $x(t) = te^{-t} [u(t) - u(t - 8)]$.
- 4.5 Create the Bode diagram of the system.
5. Consider a discrete-time system with impulse response $h[n] = n \cdot 0.4^n u[n]$.
 - 5.1 Compute the transfer function $H(z)$ of the system.
 - 5.2 Determine if the system is BIBO stable.
 - 5.3 Compute and plot the step response $s[n]$ of the system.
 - 5.4 Compute and plot the response of the system to the input signal $x[n] = 0.9^n$, $0 \leq n \leq 30$.

6. Suppose that a system is described by the state-space model

$$\dot{x}(t) = \begin{bmatrix} -0.3 & 0.1 \\ 2 & -1 \end{bmatrix}x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix}v(t)$$

and $y(t) = [1 \ 1] x(t) + 0.5 v(t)$.

6.1 Compute and plot the impulse response of the system.

6.2 Compute and plot the step response of the system.

6.3 Compute and plot the response of the system to the input signal $v(t) = t \cos(t)$, $0 \leq t \leq 100$.

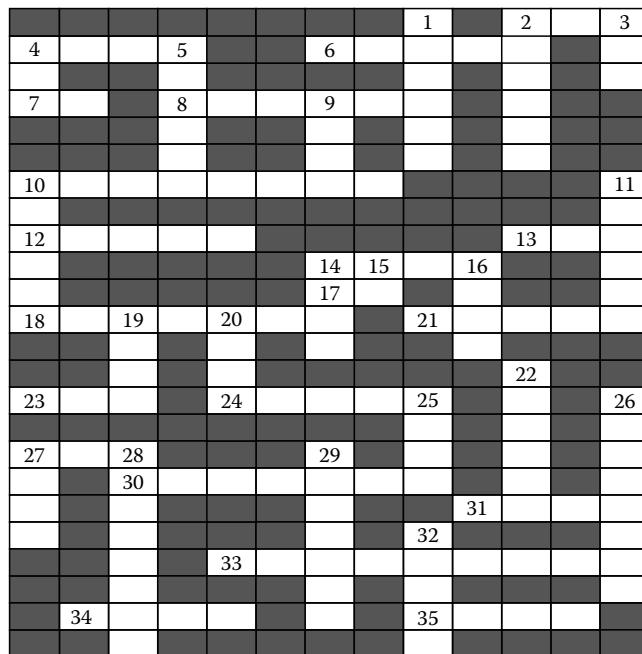
6.4 Compute and plot the initial condition response of the system when the initial state of the system is $x_0 = \begin{bmatrix} 10 \\ -10 \end{bmatrix}$.

6.5 Compute and plot the state trajectories of the previous query.

6.6 Determine if the system is BIBO stable.

Appendix A: Signal Crossword

Solve the crossword puzzle with MATLAB® commands



ACROSS: (2) Identity matrix (4) Executes a string as a statement (6) Discrete-time frequency response (7) π (8) Periodic signal generator (10) Systems connection (12) Limit of a symbolic expression (13) Addition of elements (14) Step response (17) Transfer function (18) Partial fraction expansion (21) Discrete-time system response (23) Cosine (24) Continuous-time frequency response (27) Conversion of continuous-time system to discrete-time system (30) Inverse transform (31) Part of a complex number (33) Unit-step function (34) Inverse sine (35) Inverse tangent.

DOWN: (1) Computes the number of elements of a vector (2) Creates the graph of a symbolic expression (3) Computes the eigenvalues (4) The inverse of the natural logarithm (5) Useful when more than one function is plotted in a figure (9) The Fourier transform of a rectangular pulse (10) Computes the output of an IIR filter (11) Cumulative sum (14) Plots discrete-time signals (15) The Laplace transform of an impulse response (16) Computes the roots of the denominator polynomial of a transfer function (19) Defines symbolic variables (20) Computes the derivative of a function (22) Zero-pole plot (25) Replaces variables in a symbolic expression (26) Systems interconnection (27) Polynomial multiplication (28) Computes the impulse response of discrete-time system (29) A transform (32) The derivative of $u(t)$.

This page intentionally left blank

Appendix B: Notation

\mathbf{A}^{-1}	The inverse of matrix \mathbf{A}
\mathbf{A}^T	The transpose of matrix \mathbf{A}
$x(t)$	Continuous-time signal
$x[n]$	Discrete-time signal
$x^*(t)$	The conjugate of $x(t)$
$\text{Re}\{x(t)\}$	The real part of $x(t)$
$\text{Im}\{x(t)\}$	The imaginary part of $x(t)$
$ x(t) $	The magnitude of $x(t)$
$\angle x(t)$	The phase of $x(t)$
$u(t)$	Unit-step function
$\delta(t)$	Dirac function
$r(t)$	Ramp function
$u[n]$	Unit-step sequence
$\delta[n]$	Delta sequence
$r[n]$	Ramp sequence
$h(t)$	Impulse response
$h[n]$	Discrete-time impulse response
$s(t)$	Step response
$s[n]$	Discrete-time step response
$F\{x(t)\}$	Fourier transform of $x(t)$
$F^{-1}\{X(\Omega)\}$	Inverse Fourier transform of $X(\Omega)$
$DTFT\{x[n]\}$	Discrete-time Fourier transform of $x[n]$
$IDTFT\{X(\omega)\}$	Inverse discrete-time Fourier transform of $X(\omega)$
$DFT\{x[n]\}$	Discrete Fourier transform of $x[n]$
$IDFT\{X_k\}$	Inverse discrete Fourier transform of X_k
$FFT\{x[n]\}$	Fast Fourier transform of $x[n]$
$IFFT\{X_k\}$	Inverse fast Fourier transform of X_k
$L\{x(t)\}$	Laplace transform of $x(t)$
$L^{-1}\{X(s)\}$	Inverse Laplace transform of $X(s)$
$Z\{x[n]\}$	z -Transform of $x[n]$
$Z^{-1}\{X(z)\}$	Inverse z -transform of $X(z)$
$H(\Omega)$	Continuous-time frequency response
$H(\omega)$	Discrete-time frequency response
$H(s)$	Continuous-time transfer function
$H(z)$	Discrete-time transfer function

This page intentionally left blank

Bibliography

1. Ali, T.E. and Karim, M.A., *Continuous Signals and Systems with MATLAB*, 2nd edition, CRC Press, Boca Raton, FL, 2008.
2. Boulet, B., *Fundamentals of Signals and Systems*, Da Vinci Engineering Press, Hingham, MA, 2006.
3. Fotopoulos, P. and Veloni, A.N., *Signals & Systems for Engineers* (in Greek), Synchroni Ekdotiki, Athens, Greece, 2008.
4. Hayes, M., *Schaum's Outline of Digital Signal Processing*, McGraw-Hill, New York, 1999.
5. Haykin, S. and Veen, B.V., *Signals and Systems*, Wiley, New York, 2003.
6. Ingle, V.K. and Proakis, J.G., *Digital Signal Processing Using MATLAB*, CL Engineering, 2006.
7. Kamen, E.W. and Heck, B.S., *Signals and Systems Using the Web and MATLAB*, Prentice-Hall, Upper Saddle River, NJ, 2006.
8. Karris, S., *Signals and Systems with MATLAB Applications*, Orchard Publications, Fremont, CA, 2007.
9. Karu, Z.Z., *Signals and Systems Made Ridiculously Simple*, ZiZi Press, Cambridge, MA, 1995.
10. Lathi, B.P., *Linear Systems and Signals*, Oxford University Press, Oxford, NY, 2005.
11. Oppenheim, A.V., Willsky, A.S., and Nawab, S.H., *Signals and Systems*, Prentice-Hall, Upper Saddle River, NJ, 1997.
12. Palamides, A.P. and Veloni, A.N., *Signals & Systems with MATLAB* (in Greek), Synchroni Ekdotiki, Athens, Greece, 2008.
13. Pouliarikas, A., *Signals and Systems Primer with MATLAB*, CRC Press, Boca Raton, FL, 2006.
14. Roberts, M.J., *Signals and Systems: Analysis of Signals through Linear Systems*, McGraw-Hill, New York, 2003.
15. Yang, W.Y., *Signals and Systems with MATLAB*, Springer, Berlin, Germany, 2009.

This page intentionally left blank

Signals and Systems Laboratory with MATLAB®

With its exhaustive coverage of relevant theory, **Signals and Systems Laboratory with MATLAB®** is a powerful resource that provides simple, detailed instructions on how to apply computer methods to *signals and systems* analysis. Written for laboratory work in a course on signals and systems, this book presents a corresponding MATLAB implementation for each theoretical concept introduced, making it a powerful learning tool for engineers, scientists, and students alike.

MATLAB code is used in problems and examples presented throughout the book. This code and other learning materials are available in a downloadable supplement.

Due to the extensive—and truly unique—integration of MATLAB throughout this book, the authors provide a complete tutorial on use of the language for signals and systems analysis. With more than 5000 lines of MATLAB code and more than 700 figures embedded in the text, the material teaches readers how to program in MATLAB and study signals and systems concepts at the same time, giving them the tools to harness the power of computers to quickly assess problems and then visualize their solutions.

Among its many useful features, this book:

- Offers complete coverage of the signals and systems theory, starting with elementary signals and concluding with state-space modeling
- Contains more than 400 examples and chapter-end solved problems
- Executes commands one-by-one at the MATLAB command prompt, and results, along with comments, are presented side-by side in two- or three-column tables

Additional pedagogical features:

- A detailed MATLAB tutorial to introduce a beginner programmer to the language
- Laboratory exercises that give students hands-on experience and help professors organize a course laboratory component
- Presentation of continuous- and discrete-time in parallel fashion, effectively illustrating the similarities and differences between the two
- Step-by-step examples that present data in tabular format and usually offer several different solutions to each problem

K11500



CRC Press
Taylor & Francis Group
an informa business
www.crcpress.com

6000 Broken Sound Parkway, NW
Suite 300, Boca Raton, FL 33487
270 Madison Avenue
New York, NY 10016
2 Park Square, Milton Park
Abingdon, Oxon OX14 4RN, UK

ISBN: 978-1-4398-3055-0
90000

9 781439 830550

www.crcpress.com