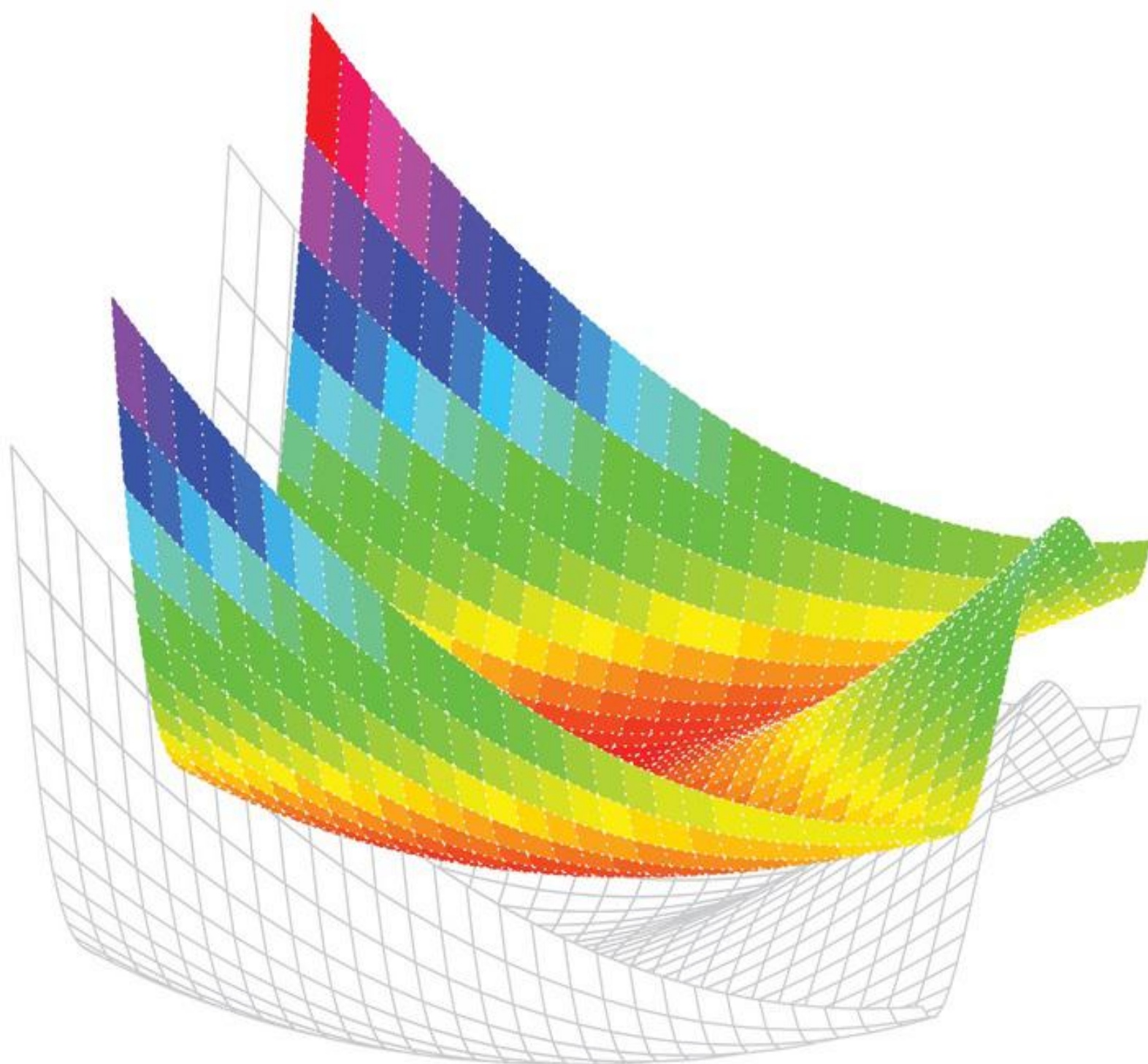


# MATLAB

AN INTRODUCTION WITH APPLICATIONS

6TH EDITION

AMOS  
GILAT



WILEY

# TABLE OF CONTENTS

[COVER](#)

[TITLE](#)

[COPYRIGHT](#)

[PREFACE](#)

[DEDICATION](#)

[INTRODUCTION](#)

[CHAPTER 1: STARTING WITH MATLAB](#)

[1.1 STARTING MATLAB, MATLAB WINDOWS](#)

[1.2 WORKING IN THE COMMAND WINDOW](#)

[1.3 ARITHMETIC OPERATIONS WITH SCALARS](#)

[1.4 DISPLAY FORMATS](#)

[1.5 ELEMENTARY MATH BUILT-IN FUNCTIONS](#)

[1.6 DEFINING SCALAR VARIABLES](#)

[1.7 USEFUL COMMANDS FOR MANAGING VARIABLES](#)

[1.8 SCRIPT FILES](#)

[1.9 EXAMPLES OF MATLAB APPLICATIONS](#)

[1.10 PROBLEMS](#)

[CHAPTER 2: CREATING ARRAYS](#)

[2.1 CREATING A ONE-DIMENSIONAL ARRAY \(VECTOR\)](#)

[2.2 CREATING A TWO-DIMENSIONAL ARRAY \(MATRIX\)](#)

[2.3 NOTES ABOUT VARIABLES IN MATLAB](#)

[2.4 THE TRANSPOSE OPERATOR](#)

[2.5 ARRAY ADDRESSING](#)

[2.6 USING A COLON : IN ADDRESSING ARRAYS](#)

[2.7 ADDING ELEMENTS TO EXISTING VARIABLES](#)

[2.8 DELETING ELEMENTS](#)

[2.9 BUILT-IN FUNCTIONS FOR HANDLING ARRAYS](#)

[2.10 STRINGS AND STRINGS AS VARIABLES](#)

[2.11 PROBLEMS](#)

[CHAPTER 3: MATHEMATICAL OPERATIONS WITH ARRAYS](#)

[3.1 ADDITION AND SUBTRACTION](#)

[3.2 ARRAY MULTIPLICATION](#)

[3.3 ARRAY DIVISION](#)

[3.4 ELEMENT-BY-ELEMENT OPERATIONS](#)

[3.5 USING ARRAYS IN MATLAB BUILT-IN MATH FUNCTIONS](#)

[3.6 BUILT-IN FUNCTIONS FOR ANALYZING ARRAYS](#)

[3.7 GENERATION OF RANDOM NUMBERS](#)

[3.8 EXAMPLES OF MATLAB APPLICATIONS](#)

[3.9 PROBLEMS](#)

## [CHAPTER 4: USING SCRIPT FILES AND MANAGING DATA](#)

[4.1 THE MATLAB WORKSPACE AND THE WORKSPACE WINDOW](#)

[4.2 INPUT TO A SCRIPT FILE](#)

[4.3 OUTPUT COMMANDS](#)

[4.4 THE `save` AND `load` COMMANDS](#)

[4.5 IMPORTING AND EXPORTING DATA](#)

[4.6 EXAMPLES OF MATLAB APPLICATIONS](#)

[4.7 PROBLEMS](#)

## [CHAPTER 5: TWO-DIMENSIONAL PLOTS](#)

[5.1 THE `plot` COMMAND](#)

[5.2 THE `fplot` COMMAND](#)

[5.3 PLOTTING MULTIPLE GRAPHS IN THE SAME PLOT](#)

[5.4 FORMATTING A PLOT](#)

[5.5 PLOTS WITH LOGARITHMIC AXES](#)

[5.6 PLOTS WITH ERROR BARS](#)

[5.7 PLOTS WITH SPECIAL GRAPHICS](#)

[5.8 HISTOGRAMS](#)

[5.9 POLAR PLOTS](#)

[5.10 PUTTING MULTIPLE PLOTS ON THE SAME PAGE](#)

[5.11 MULTIPLE FIGURE WINDOWS](#)

[5.12 PLOTTING USING THE PLOTS TOOLSTRIP](#)

[5.13 EXAMPLES OF MATLAB APPLICATIONS](#)

[5.14 PROBLEMS](#)

## [CHAPTER 6: PROGRAMMING IN MATLAB](#)

[6.1 RELATIONAL AND LOGICAL OPERATORS](#)

[6.2 CONDITIONAL STATEMENTS](#)

[6.3 THE `switch-case` STATEMENT](#)

[6.4 LOOPS](#)

[6.5 NESTED LOOPS AND NESTED CONDITIONAL STATEMENTS](#)

[6.6 THE `break` AND `continue` COMMANDS](#)

[6.7 EXAMPLES OF MATLAB APPLICATIONS](#)

[6.8 PROBLEMS](#)

## CHAPTER 7: USER-DEFINED FUNCTIONS AND FUNCTION FILES

### 7.1 CREATING A FUNCTION FILE

### 7.2 STRUCTURE OF A FUNCTION FILE

### 7.3 LOCAL AND GLOBAL VARIABLES

### 7.4 SAVING A FUNCTION FILE

### 7.5 USING A USER-DEFINED FUNCTION

### 7.6 EXAMPLES OF SIMPLE USER-DEFINED FUNCTIONS

### 7.7 COMPARISON BETWEEN SCRIPT FILES AND FUNCTION FILES

### 7.8 ANONYMOUS FUNCTIONS

### 7.9 FUNCTION FUNCTIONS

### 7.10 SUBFUNCTIONS

### 7.11 NESTED FUNCTIONS

### 7.12 EXAMPLES OF MATLAB APPLICATIONS

### 7.13 PROBLEMS

## CHAPTER 8: POLYNOMIALS, CURVE FITTING, AND INTERPOLATION

### 8.1 POLYNOMIALS

### 8.2 CURVE FITTING

### 8.3 INTERPOLATION

### 8.4 THE BASIC FITTING INTERFACE

### 8.5 EXAMPLES OF MATLAB APPLICATIONS

### 8.6 PROBLEMS

## CHAPTER 9: APPLICATIONS IN NUMERICAL ANALYSIS

### 9.1 SOLVING AN EQUATION WITH ONE VARIABLE

### 9.2 FINDING A MINIMUM OR A MAXIMUM OF A FUNCTION

### 9.3 NUMERICAL INTEGRATION

### 9.4 ORDINARY DIFFERENTIAL EQUATIONS

### 9.5 EXAMPLES OF MATLAB APPLICATIONS

### 9.6 PROBLEMS

## CHAPTER 10: THREE-DIMENSIONAL PLOTS

### 10.1 LINE PLOTS

### 10.2 MESH AND SURFACE PLOTS

### 10.3 PLOTS WITH SPECIAL GRAPHICS

### 10.4 THE `view` COMMAND

### 10.5 EXAMPLES OF MATLAB APPLICATIONS

### 10.6 PROBLEMS

## CHAPTER 11: SYMBOLIC MATH

### 11.1 SYMBOLIC OBJECTS AND SYMBOLIC EXPRESSIONS

[11.2 CHANGING THE FORM OF AN EXISTING SYMBOLIC EXPRESSION](#)

[11.3 SOLVING ALGEBRAIC EQUATIONS](#)

[11.4 DIFFERENTIATION](#)

[11.5 INTEGRATION](#)

[11.6 SOLVING AN ORDINARY DIFFERENTIAL EQUATION](#)

[11.7 PLOTTING SYMBOLIC EXPRESSIONS](#)

[11.8 NUMERICAL CALCULATIONS WITH SYMBOLIC EXPRESSIONS](#)

[11.9 EXAMPLES OF MATLAB APPLICATIONS](#)

[11.10 PROBLEMS](#)

[APPENDIX: SUMMARY OF CHARACTERS, COMMANDS, AND FUNCTIONS](#)

[INDEX](#)

[End User License Agreement](#)

## List of Illustrations

### CHAPTER 1: STARTING WITH MATLAB

[Figure 1-1: The default view of MATLAB desktop.](#)

[Figure 1-2: Example of a Figure Window.](#)

[Figure 1-3: Example of an Editor Window.](#)

[Figure 1-4: The Help Window.](#)

[Figure 1-5: The Command Window.](#)

[Figure 1-6: The Editor/Debugger Window.](#)

[Figure 1-7: A program typed in the Editor/Debugger Window.](#)

[Figure 1-8: The Current folder field in the Command Window.](#)

[Figure 1-9: Changing the current directory.](#)

[Figure 1-10: The Current Folder Window.](#)

### CHAPTER 2: CREATING ARRAYS

[Figure 2-1: Position of a point.](#)

### CHAPTER 4: USING SCRIPT FILES AND MANAGING DATA

[Figure 4-1: The Workspace Window.](#)

[Figure 4-2: The Variable Editor Window.](#)

[Figure 4-3: The VmphtoVkm.txt file opened in Word.](#)

[Figure 4-4: The FlbtoFN.txt file opened in Word.](#)

[Figure 4-5: Data saved in ASCII format.](#)

[Figure 4-6: Data saved as .txt file.](#)

[Figure 4-7: Excel spreadsheet with data.](#)

[Figure 4-8: Numerical ASCII data.](#)

[Figure 4-9: Import Wizard, first display.](#)

[Figure 4-10: Import Wizard, second display.](#)

## CHAPTER 5: TWO-DIMENSIONAL PLOTS

[Figure 5-1: Example of a formatted two-dimensional plot.](#)

[Figure 5-2: The Figure Window with a simple plot.](#)

[Figure 5-3: The Figure Window with a plot of the sales data.](#)

[Figure 5-4: The Figure Window with a plot of the function  \$y = 3.5^{-0.5x} \cos\(6x\)\$](#)

[Figure 5-5: A plot of the function  \$y = 3.5^{-0.5x} \cos\(6x\)\$  with large spacing.\)](#)

[Figure 5-6: A plot of the function  \$y = x^2 + 4\sin\(2x\) - 1\$ .](#)

[Figure 5-7: A plot of the function  \$y = 3x^3 - 26x + 10\$  and its first and second derivatives.](#)

[Figure 5-8: Formatting a plot using the Plot Editor.](#)

[Figure 5-9: Plots of  \$y=2^{\(-0.2x+10\)}\$  with linear, semilog, and log-log scales.](#)

[Figure 5-10: A plot with error bars.](#)

[Figure 5-11: Histogram of temperature data.](#)

[Figure 5-12: Two open Figure Windows.](#)

[Figure 5-13: Using the PLOTS Toolstrip.](#)

[Figure 5-14: Using the PLOTS Toolstrip.](#)

[Figure 5-15: Position, velocity, and acceleration of the piston vs. time.](#)

## CHAPTER 6: PROGRAMMING IN MATLAB

[Figure 6-1: The structure of the `if-end` conditional statement.](#)

[Figure 6-2: The structure of the `if-else-end` conditional statement.](#)

[Figure 6-3: The structure of the `if-elseif-else-end` conditional statement.](#)

[Figure 6-4: The structure of a `switch-case` statement.](#)

[Figure 6-5: The structure of a `for-end` loop.](#)

[Figure 6-6: The structure of a `while-end` loop.](#)

[Figure 6-7: Structure of nested loops.](#)

## CHAPTER 7: USER-DEFINED FUNCTIONS AND FUNCTION FILES

[Figure 7-1: The Editor/Debugger Window.](#)

[Figure 7-2: Structure of a typical function file.](#)

[Figure 7-3: A plot of the function  \$f\(x\) = e^{-0.17x}x^3 - 2x^2 + 0.8x - 3\$ .](#)

## CHAPTER 8: POLYNOMIALS, CURVE FITTING, AND INTERPOLATION

[Figure 8-1: Least squares fitting of first-degree polynomial to four points.](#)

[Figure 8-2: Fitting data with polynomials of different order.](#)

[Figure 8-3: The Basic Fitting Window.](#)

[Figure 8-4: A Figure Window modified by the Basic Fitting Interface.](#)

## CHAPTER 10: THREE-DIMENSIONAL PLOTS

[Figure 10-1: A plot of the function  \$x = \sqrt{t}\sin\(2t\)\$ ,  \$y = \sqrt{t}\cos\(2t\)\$ ,  \$z = 0.5t\$  for  \$0 \leq t \leq 6\pi\$ .](#)

[Figure 10-2: A grid in the  \$x y\$  plane for the domain  \$-1 \leq x \leq 3\$  and  \$1 \leq y \leq 4\$  with spacing of 1.](#)

[Figure 10-3: Azimuth and elevation angles.](#)

[Figure 10-4: A surface plot of the function  \$z = 1.8^{-1.5\sqrt{x^2+y^2}}\sin\(x\)\cos\(0.5y\)\$  with viewing angles of  \$az = 20^\circ\$  and  \$el = 35^\circ\$ .](#)

[Figure 10-5: A top view plot of the function  \$x = \sqrt{t}\sin\(2t\)\$ ,  \$y = \sqrt{t}\cos\(2t\)\$ ,  \$z = 0.5t\$  for  \$0 \leq t \leq 6\pi\$ .](#)

[Figure 10-6: Projections onto the  \$x z\$  plane of the function.](#)

[Figure 10-7: Projections onto the  \$y-z\$  plane of the function.](#)

## List of Tables

### CHAPTER 1: STARTING WITH MATLAB

[Table 1-1: MATLAB windows](#)

[Table 1-2: Display formats](#)

[Table 1-3: Elementary math functions](#)

[Table 1-4: Trigonometric math functions](#)

[Table 1-5: Rounding functions](#)

### CHAPTER 2: CREATING ARRAYS

[Table 2-1: Population data](#)

[Table 2-2: Built-in functions for handling arrays](#)

### CHAPTER 3: MATHEMATICAL OPERATIONS WITH ARRAYS

[Table 3-1: Built-in array functions](#)

[Table 3-2: The `rand` command](#)

[Table 3-3: The `randi` command](#)

### CHAPTER 9: APPLICATIONS IN NUMERICAL ANALYSIS

[Table 9-1: MATLAB ODE Solvers](#)



## CHAPTER 10: THREE-DIMENSIONAL PLOTS

[Table 10-1: Mesh and surface plots](#)

[Table 10-2: Specialized 3-D plots](#)

## CHAPTER 11: SYMBOLIC MATH

[Table 11-1: Plots with the `ezplot` command](#)



# **MATLAB® An Introduction with Applications**

**Sixth Edition**

**Amos Gilat**

Department of Mechanical and Aerospace Engineering  
The Ohio State University

**WILEY**

PUBLISHER	Laurie Rosatone
EDITORIAL DIRECTOR	Don Fowley
DEVELOPMENTAL EDITOR	Chris Nelson
EXECUTIVE MARKETING MANAGER	Dan Sayre
PRODUCTION EDITOR	Ashley Patterson
EDITORIAL ASSISTANT	Courtney Jordan
COVER DESIGN	Harry Nolan
COVER IMAGE	Amos Gilat

This book was set in Times New Roman MT Std. by Amos Gilat and printed and bound by Lightning Source, Inc.

Founded in 1807, John Wiley & Sons, Inc. has been a valued source of knowledge and understanding for more than 200 years, helping people around the world meet their needs and fulfill their aspirations. Our company is built on a foundation of principles that include responsibility to the communities we serve and where we live and work. In 2008, we launched a Corporate Citizenship Initiative, a global effort to address the environmental, social, economic, and ethical challenges we face in our business. Among the issues we are addressing are carbon impact, paper specifications and procurement, ethical conduct within our business and among our vendors, and community and charitable support. For more information, please visit our website: [www.wiley.com/go/citizenship](http://www.wiley.com/go/citizenship).

Copyright © 2017, 2014, 2011 John Wiley & Sons, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc. 222 Rosewood Drive, Danvers, MA 01923 (website [www.copyright.com](http://www.copyright.com)). Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030-5774, (201)748-6011, fax (201)748-6008, or online at: [www.wiley.com/go/permissions](http://www.wiley.com/go/permissions).

Evaluation copies are provided to qualified academics and professionals for review purposes only, for use in their courses during the next academic year. These copies are licensed and may not be sold or transferred to a third party. Upon completion of the review period, please return the evaluation copy to Wiley. Return instructions and a free of charge return mailing label are available at [www.wiley.com/go/returnlabel](http://www.wiley.com/go/returnlabel). If you have chosen to adopt this textbook for use in your course, please accept this book as your complimentary desk copy. Outside of the United States, please contact your local sales representative.

ISBN: 978-1-119-25683-0 (PBK)

ISBN 978-1-119-29931-8 (EVAL)

*Library of Congress Cataloging-in-Publication Data:*

Names: Gilat, Amos, author.

Title: MATLAB : an introduction with applications / Amos Gilat, Department of Mechanical and Aerospace Engineering, The Ohio State University.

Description: Sixth edition. | Hoboken, NJ : John Wiley & Sons, Inc., [2017] |

Includes index.

Identifiers: LCCN 2016029050 (print) | LCCN 2016030206 (ebook) | ISBN 9781119256830 (paper) | ISBN 9781119299547 (pdf) | ISBN 9781119299257 (epub)

Subjects: LCSH: MATLAB.

Classification: LCC QA297 .G48 2017 (print) | LCC QA297 (ebook) | DDC 518.0285/53--dc23

LC record available at <https://lccn.loc.gov/2016029050>

The inside back cover will contain printing identification and country of origin if omitted from this page. In addition, if the ISBN on the back cover differs from the ISBN on this page, the one on the back cover is correct

# PREFACE

MATLAB® is a very popular language for technical computing used by students, engineers, and scientists in universities, research institutes, and industries all over the world. The software is popular because it is powerful and easy to use. For university freshmen it can be thought of as the next tool to use after the graphic calculator in high school.

This book was written following several years of teaching the software to freshmen in an introductory engineering course. The objective was to write a book that teaches the software in a friendly, non-intimidating fashion. Therefore, the book is written in simple and direct language. In many places bullets, rather than lengthy text, are used to list facts and details that are related to a specific topic. The book includes numerous sample problems in mathematics, science, and engineering that are similar to problems encountered by new users of MATLAB.

This sixth edition of the book is updated to MATLAB Release 2016a. In addition, the end of chapter problems have been revised. In [Chapters 1](#) through [8](#) close to 70% of the problems are new or different than in previous editions.

I would like to thank several of my colleagues at The Ohio State University. Professor Richard Freuler for his comments, and Dr. Mike Parke for reviewing sections of the book and suggested modifications. I also appreciate the involvement and support of Professors Robert Gustafson, John Demel and Dr. John Merrill from the Engineering Education Innovation Center at The Ohio State University. Special thanks go to Professor Mike Lichtensteiger (OSU), and my daughter Tal Gilat (Marquette University), who carefully reviewed the first edition of the book and provided valuable comments and criticisms. Professor Brian Harper (OSU) has made a significant contribution to the new end of chapter problems in the present edition.

I would like to express my appreciation to all those who have reviewed earlier editions of the text at its various stages of development, including Betty Barr, University of Houston; Andrei G. Chakhovskoi, University of California, Davis; Roger King, University of Toledo; Richard Kwor, University of Colorado at Colorado Springs; Larry Lagerstrom, University of California, Davis; Yueh-Jaw Lin, University of Akron; H. David Sheets, Canisius College; Geb Thomas, University of Iowa; Brian Vick, Virginia Polytechnic Institute and State University; Jay Weitzen, University of Massachusetts, Lowell; and Jane Patterson Fife, The Ohio State University. In addition, I would like to acknowledge Chris Nelson who supported the production of the sixth edition.

I hope that the book will be useful and will help the users of MATLAB to enjoy the software.

Amos Gilat  
Columbus, Ohio  
May, 2016  
[gilat.1@osu.edu](mailto:gilat.1@osu.edu)



# INTRODUCTION

MATLAB is a powerful language for technical computing. The name MATLAB stands for MATrix LABoratory, because its basic data element is a matrix (array). MATLAB can be used for math computations, modeling and simulations, data analysis and processing, visualization and graphics, and algorithm development.

MATLAB is widely used in universities and colleges in introductory and advanced courses in mathematics, science, and especially engineering. In industry the software is used in research, development, and design. The standard MATLAB program has tools (functions) that can be used to solve common problems. In addition, MATLAB has optional toolboxes that are collections of specialized programs designed to solve specific types of problems. Examples include toolboxes for signal processing, symbolic calculations, and control systems.

Until recently, most of the users of MATLAB have been people with previous knowledge of programming languages such as FORTRAN and C who switched to MATLAB as the software became popular. Consequently, the majority of the literature that has been written about MATLAB assumes that the reader has knowledge of computer programming. Books about MATLAB often address advanced topics or applications that are specialized to a particular field. Today, however, MATLAB is being introduced to college students as the first (and often the only) computer program they will learn. For these students there is a need for a book that teaches MATLAB assuming no prior experience in computer programming.

## The Purpose of This Book

*MATLAB: An Introduction with Applications* is intended for students who are using MATLAB for the first time and have little or no experience in computer programming. It can be used as a textbook in freshmen engineering courses or in workshops where MATLAB is being taught. The book can also serve as a reference in more advanced science and engineering courses where MATLAB is used as a tool for solving problems. It also can be used for self-study of MATLAB by students and practicing engineers. In addition, the book can be a supplement or a secondary book in courses where MATLAB is used but the instructor does not have the time to cover it extensively.

## Topics Covered

MATLAB is a huge program, and therefore it is impossible to cover all of it in one book. This book focuses primarily on the foundations of MATLAB. The assumption is that once these foundations are well understood, the student will be able to learn advanced topics easily by using the information in the Help menu.

The order in which the topics are presented in this book was chosen carefully, based on several years of experience in teaching MATLAB in an introductory engineering course. The topics are presented in an order that allows the student to follow the book chapter after chapter. Every topic is presented completely in one place and then used in the following chapters.

The first chapter describes the basic structure and features of MATLAB and how to use the

program for simple arithmetic operations with scalars as with a calculator. Script files are introduced at the end of the chapter. They allow the student to write, save, and execute simple MATLAB programs. The next two chapters are devoted to the topic of arrays. MATLAB's basic data element is an array that does not require dimensioning. This concept, which makes MATLAB a very powerful program, can be a little difficult to grasp for students who have only limited knowledge of and experience with linear algebra and vector analysis. The concept of arrays is introduced gradually and then explained in extensive detail. [Chapter 2](#) describes how to create arrays, and [Chapter 3](#) covers mathematical operations with arrays.

Following the basics, more advanced topics that are related to script files and input and output of data are presented in [Chapter 4](#). This is followed by coverage of two-dimensional plotting in [Chapter 5](#). Programming with MATLAB is introduced in [Chapter 6](#). This includes flow control with conditional statements and loops. User-defined functions, anonymous functions, and function functions are covered next in [Chapter 7](#). The coverage of function files (user-defined functions) is intentionally separated from the subject of script files. This has proven to be easier to understand by students who are not familiar with similar concepts from other computer programs.

The next three chapters cover more advanced topics. [Chapter 8](#) describes how MATLAB can be used for carrying out calculations with polynomials, and how to use MATLAB for curve fitting and interpolation. [Chapter 9](#) covers applications of MATLAB in numerical analysis. It includes solving nonlinear equations, finding minimum or a maximum of a function, numerical integration, and solution of first-order ordinary differential equations. [Chapter 10](#) describes how to produce three-dimensional plots, an extension of the chapter on two-dimensional plots. [Chapter 11](#) covers in great detail how to use MATLAB in symbolic operations.

## The Framework of a Typical Chapter

In every chapter the topics are introduced gradually in an order that makes the concepts easy to understand. The use of MATLAB is demonstrated extensively within the text and by examples. Some of the longer examples in [Chapters 1–3](#) are titled as tutorials. Every use of MATLAB is printed with a different font and with a gray background. Additional explanations appear in boxed text with a white background. The idea is that the reader will execute these demonstrations and tutorials in order to gain experience in using MATLAB. In addition, every chapter includes formal sample problems that are examples of applications of MATLAB for solving problems in math, science, and engineering. Each example includes a problem statement and a detailed solution. Some sample problems are presented in the middle of the chapter. All of the chapters (except [Chapter 2](#)) have a section at the end with several sample problems of applications. It should be pointed out that problems with MATLAB can be solved in many different ways. The solutions of the sample problems are written such that they are easy to follow. This means that in many cases the problem can be solved by writing a shorter, or sometimes “trickier,” program. The students are encouraged to try to write their own solutions and compare the end results. At the end of each chapter there is a set of homework problems. They include general problems from math and science and problems from different disciplines of engineering.

## Symbolic Calculations

MATLAB is essentially a software for numerical calculations. Symbolic math operations, however, can be executed if the Symbolic Math toolbox is installed. The Symbolic Math toolbox is included in the student version of the software and can be added to the standard program.

## Software and Hardware

The MATLAB program, like most other software, is continually being developed and new versions are released frequently. This book covers MATLAB Version 9.0.0.341360, Release 2016a. It should be emphasized, however, that the book covers the basics of MATLAB, which do not change much from version to version. The book covers the use of MATLAB on computers that use the Windows operating system. Everything is essentially the same when MATLAB is used on other machines. The user is referred to the documentation of MATLAB for details on using MATLAB on other operating systems. It is assumed that the software is installed on the computer, and the user has basic knowledge of operating the computer.

## The Order of Topics in the Book

It is probably impossible to write a textbook where all the subjects are presented in an order that is suitable for everyone. The order of topics in this book is such that the fundamentals of MATLAB are covered first (arrays and array operations), and, as mentioned before, every topic is covered completely in one location, which makes the book easy to use as a reference. The order of the topics in this sixth edition is the same as in the previous edition. Programming is introduced before user-defined functions. This allows using programming in user-defined functions. Also, applications of MATLAB in numerical analysis follow [Chapter 8](#) which covers polynomials, curve fitting, and interpolation.



# CHAPTER 1

## STARTING WITH MATLAB

This chapter begins by describing the characteristics and purpose of the different windows in MATLAB. Next, the Command Window is introduced in detail. The chapter shows how to use MATLAB for arithmetic operations with scalars in much the way that a calculator is used. This includes the use of elementary math functions with scalars. The chapter then shows how to define scalar variables (the assignment operator) and how to use these variables in arithmetic calculations. The last section in the chapter introduces script files. It shows how to write, save, and execute simple MATLAB programs.

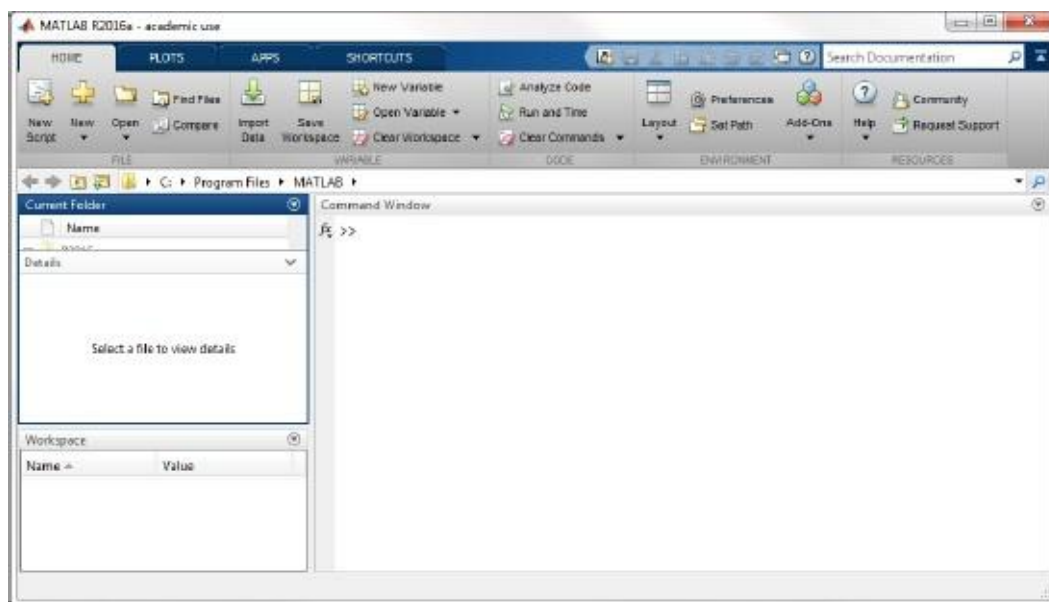
### 1.1 STARTING MATLAB, MATLAB WINDOWS

It is assumed that the software is installed on the computer, and that the user can start the program. Once the program starts, the MATLAB desktop window opens with the default layout, [Figure 1-1](#). The layout has a Toolstrip at the top, the Current Folder Toolbar below it, and four windows underneath. At the top of the Toolstrip there are three tabs: HOME, PLOTS, and APPS. Clicking on the tabs changes the icons in the Toolstrip. Commonly, MATLAB is used with the HOME tab selected. The associated icons are used for executing various commands, as explained later in this chapter. The PLOTS tab can be used to create plots, as explained in [Chapter 5 \(Section 5.12\)](#), and the APPS tab can be used for opening additional applications and Toolboxes of MATLAB.

#### *The default layout*

The default layout ([Figure 1-1](#)) consists of the following four windows that are displayed under the Toolstrip: the Command Window (the larger window), the Current Folder Window (on the top left), the Details Window and the Workspace Window (on the bottom left). A list of several MATLAB windows and their purposes is given in [Table 1-1](#).

Four of the windows—the Command Window, the Figure Window, the Editor Window, and the Help Window—are used extensively throughout the book and are briefly described on the following pages. More detailed descriptions are included in the chapters where they are used. The Command History Window, Current Folder Window, and the Workspace Window are described in [Sections 1.2](#), [1.8.4](#), and [4.1](#), respectively.



**Figure 1-1:** The default view of MATLAB desktop.

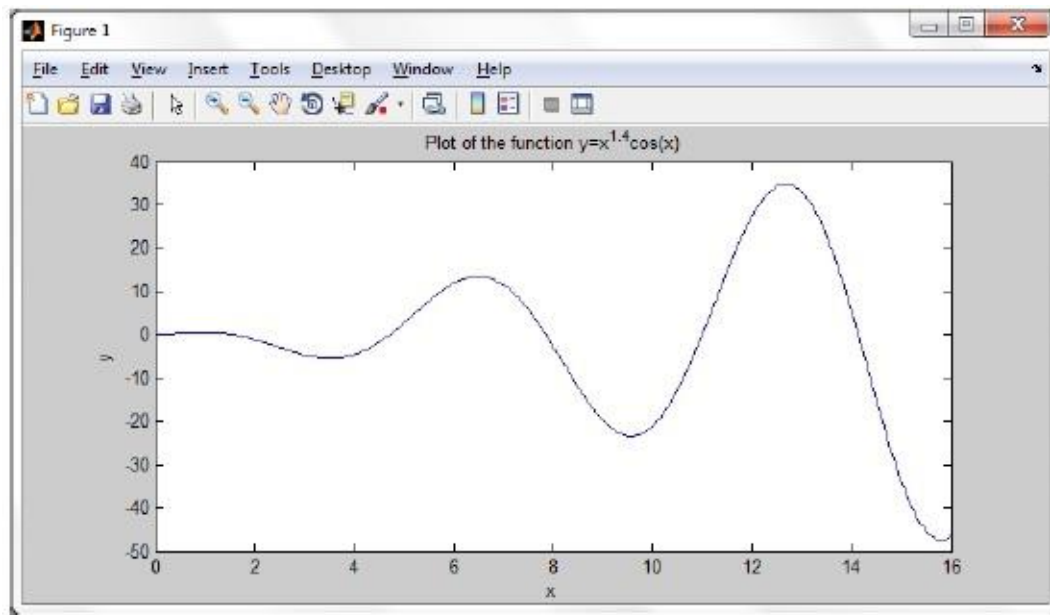
**Command Window:** The Command Window is MATLAB’s main window and opens when MATLAB is started. It is convenient to have the Command Window as the only visible window. This can be done either by closing all the other windows, or by selecting **Command Window Only** in the menu that opens when the **Layout** icon on the Toolstrip is selected. To close a window, click on the pull-down menu at the top right-hand side of the window and then select Close. Working in the Command Window is described in detail in [Section 1.2](#).

**TABLE 1-1:**

## MATLAB windows

Window	Purpose
Command Window	Main window, enters variables, runs programs.
Figure Window	Contains output from graphic commands.
Editor Window	Creates and debugs script and function files.
Help Window	Provides help information.
Command History Window	Logs commands entered in the Command Window.
Workspace Window	Provides information about the variables that are stored.
Current Folder Window	Shows the files in the current folder.

**Figure Window:** The Figure Window opens automatically when graphics commands are executed, and contains graphs created by these commands. An example of a Figure Window is shown in [Figure 1-2](#). A more detailed description of this window is given in [Chapter 5](#).

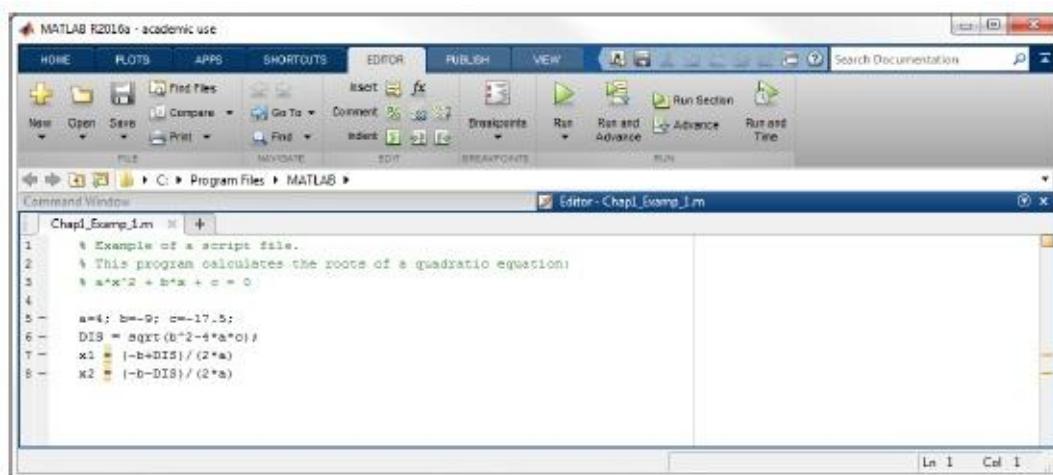


**Figure 1-2:** Example of a Figure Window.

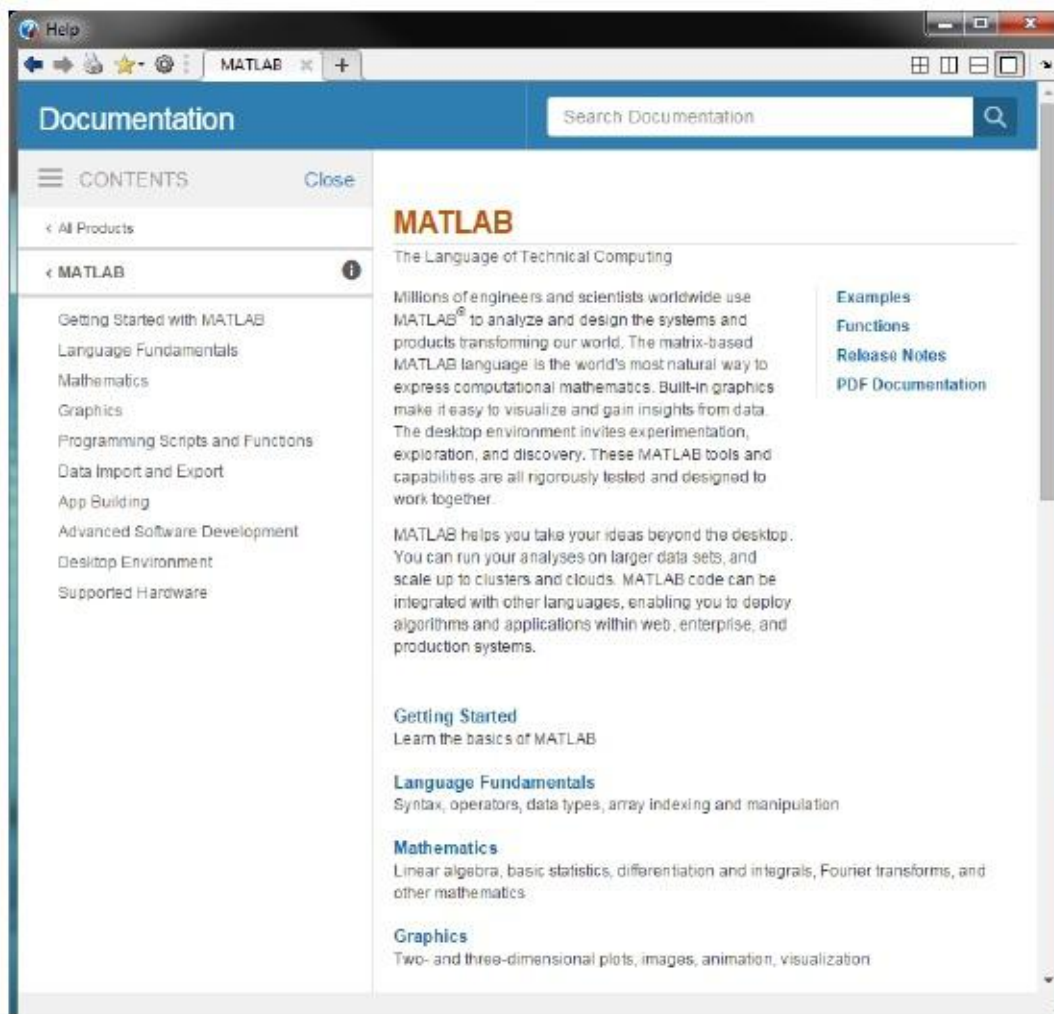
**Editor Window:** The Editor Window is used for writing and editing programs. This window is opened by clicking on the **New Script** icon in the Toolstrip, or by clicking on the **New** icon and then selecting **Script** from the menu that opens. An example of an Editor Window is shown in [Figure 1-3](#). More details on the Editor Window are given in [Section 1.8.2](#), where it is used for writing script files, and in [Chapter 7](#), where it is used to write function files.

**Help Window:** The Help Window contains help information. This window can be opened from the **Help** icon in the Toolstrip of the Command Window or the toolbar of any MATLAB window. The Help Window is interactive and can be used to obtain information on any feature of MATLAB. [Figure 1-4](#) shows an open Help Window.

When MATLAB is started for the first time, the screen looks like that shown in [Figure 1-1](#). For most beginners it is probably more convenient to close all the windows except the Command Window. The closed windows can be reopened by selecting them from the **layout** icon in the Toolstrip. The windows shown in [Figure 1-1](#) can be displayed by clicking on the **layout** icon and selecting **Default** in the menu that opens. The various windows in [Figure 1-1](#) are docked to the desktop. A window can be undocked (become a separate, independent window) by dragging it out. An independent window can be redocked by clicking on the pull-down menu at the top right-hand side of the window and then selecting **Dock**.



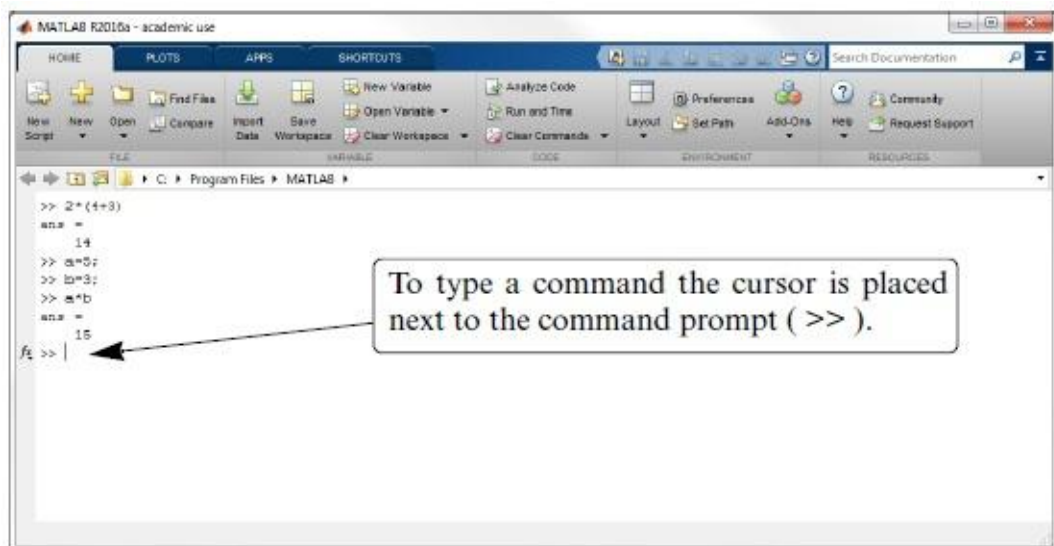
**Figure 1-3:** Example of an Editor Window.



**Figure 1-4: The Help Window.**

## 1.2 WORKING IN THE COMMAND WINDOW

The Command Window is MATLAB's main window and can be used for executing commands, opening other windows, running programs written by the user, and managing the software. An example of the Command Window, with several simple commands that will be explained later in this chapter, is shown in [Figure 1-5](#).



**Figure 1-5: The Command Window.**

## Notes for working in the Command Window:

- To type a command, the cursor must be placed next to the command prompt ( >> ).
- Once a command is typed and the **Enter** key is pressed, the command is executed. However, only the last command is executed. Everything executed previously (that might be still displayed) is unchanged.
- Several commands can be typed in the same line. This is done by typing a comma between the commands. When the **Enter** key is pressed, the commands are executed in order from left to right.
- It is not possible to go back to a previous line that is displayed in the Command Window, make a correction, and then re-execute the command.
- A previously typed command can be recalled to the command prompt with the up-arrow key (↑). When the command is displayed at the command prompt, it can be modified if needed and then executed. The down-arrow key (↓) can be used to move down the list of previously typed commands.
- If a command is too long to fit in one line, it can be continued to the next line by typing three periods ... (called an ellipsis) and pressing the **Enter** key. The continuation of the command is then typed in the new line. The command can continue line after line up to a total of 4,096 characters.

## The semicolon ( ; ):

When a command is typed in the Command Window and the **Enter** key is pressed, the command is executed. Any output that the command generates is displayed in the Command Window. If a semicolon ( ; ) is typed at the end of a command, the output of the command is not displayed. Typing a semicolon is useful when the result is obvious or known, or when the output is very large.

If several commands are typed in the same line, the output from any of the commands will not be displayed if a semicolon instead of a comma is typed between the commands.

## Typing %:

When the symbol % (percent) is typed at the beginning of a line, the line is designated as a comment. This means that when the **Enter** key is pressed the line is not executed. The % character followed by text (comment) can also be typed after a command (in the same line). This has no effect on the execution of the command.

Usually there is no need for comments in the Command Window. Comments, however, are frequently used in a program to add descriptions or to explain the program (see [Chapters 4 and 6](#)).

## The `clc` command:

The `clc` command (type `clc` and press **Enter**) clears the Command Window. After typing in the



Command Window for a while, the display may become very long. Once the `clc` command is executed, a clear window is displayed. The command does not change anything that was done before. For example, if some variables were defined previously (see [Section 1.6](#)), they still exist and can be used. The up-arrow key can also be used to recall commands that were typed before.

### The Command History Window:

The Command History Window lists the commands that have been entered in the Command Window. This includes commands from previous sessions. A command in the Command History Window can be used again in the Command Window. By double-clicking on the command, the command is reentered in the Command Window and executed. It is also possible to drag the command to the Command Window, make changes if needed, and then execute it. The list in the Command History Window can be cleared by selecting the lines to be deleted and then right-clicking the mouse and selecting **Delete Selection**. The whole history can be deleted by right-clicking the mouse and selecting choose **Clear Command History** in the menu that opens.

## 1.3 ARITHMETIC OPERATIONS WITH SCALARS

In this chapter we discuss only arithmetic operations with scalars, which are numbers. As will be explained later in the chapter, numbers can be used in arithmetic calculations directly (as with a calculator) or they can be assigned to variables, which can subsequently be used in calculations. The symbols of arithmetic operations are:

Operation	Symbol	Example
Addition	+	5 + 3
Subtraction	–	5 – 3
Multiplication	*	5 * 3
Right division	/	5 / 3
Left division	\	5 \ 3 = 3 / 5
Exponentiation	^	5 ^ 3 (means $5^3 = 125$ )

It should be pointed out here that all the symbols except the left division are the same as in most calculators. For scalars, the left division is the inverse of the right division. The left division, however, is mostly used for operations with arrays, which are discussed in [Chapter 3](#).

### 1.3.1 Order of Precedence

MATLAB executes the calculations according to the order of precedence displayed below. This order is the same as used in most calculators.

Precedence	Mathematical Operation
First	Parentheses. For nested parentheses, the innermost are executed first.
Second	Exponentiation.
Third	Multiplication, division (equal precedence).
Fourth	Addition and subtraction.

In an expression that has several operations, higher-precedence operations are executed before lower-precedence operations. If two or more operations have the same precedence, the expression is executed from left to right. As illustrated in the next section, parentheses can be used to change the order of calculations.

### 1.3.2 Using MATLAB as a Calculator

The simplest way to use MATLAB is as a calculator. This is done in the Command Window by typing a mathematical expression and pressing the **Enter** key. MATLAB calculates the expression and responds by displaying `ans =` followed by the numerical result of the expression in the next line. This is demonstrated in [Tutorial 1-1](#).

#### [Tutorial 1-1](#): Using MATLAB as a calculator.

```

>> 7+8/2
ans =
    11
>> (7+8)/2
ans =
    7.5000
>> 4+5/3+2
ans =
    7.6667
>> 5^3/2
ans =
    62.5000
>> 27^(1/3)+32^0.2
ans =
     5
>> 27^1/3+32^0.2
ans =
    11
>> 0.7854-(0.7854)^3/(1*2*3)+0.785^5/(1*2*3*4*5)...
- (0.785)^7/(1*2*3*4*5*6*7)
ans =
    0.7071
>>

```

Type and press Enter.

8/2 is executed first.

Type and press Enter.

7+8 is executed first.

5/3 is executed first.

5^3 is executed first, /2 is executed next.

1/3 is executed first, 27^(1/3) and 32^0.2 are executed next, and + is executed last.

27^1 and 32^0.2 are executed first, /3 is executed next, and + is executed last.

Type three periods ... (and press Enter) to continue the expression on the next line.

The last expression is the first four terms of the Taylor series for sin( $\pi/4$ ).



# 1.4 DISPLAY FORMATS

The user can control the format in which MATLAB displays output on the screen. In [Tutorial 1-1](#), the output format is fixed-point with four decimal digits (called `short`), which is the default format for numerical values. The format can be changed with the `format` command. Once the `format` command is entered, all the output that follows is displayed in the specified format. Several of the available formats are listed and described in [Table 1-2](#).

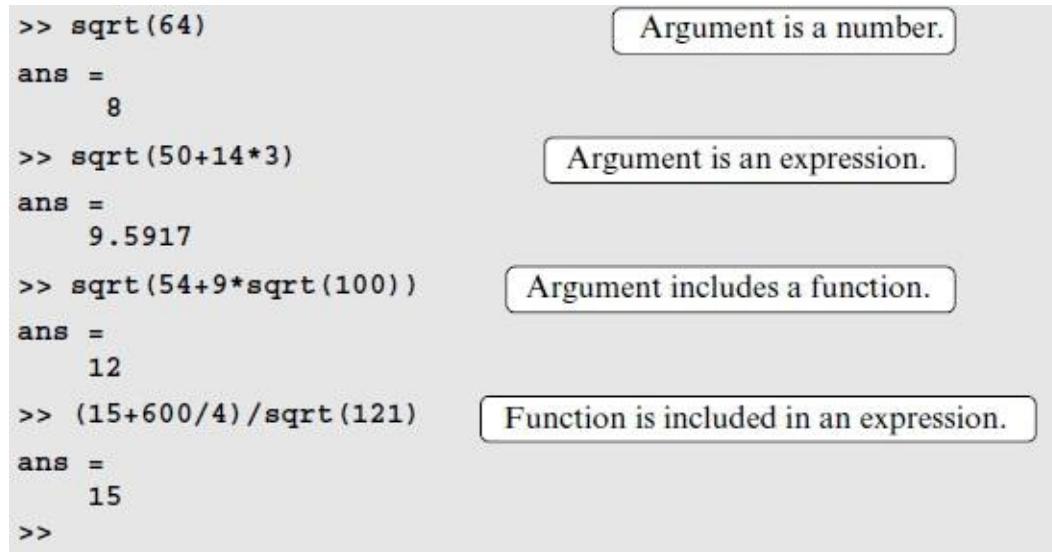
MATLAB has several other formats for displaying numbers. Details of these formats can be obtained by typing `help format` in the Command Window. The format in which numbers are displayed does not affect how MATLAB computes and saves numbers.

TABLE 1-2:		
Display formats		
Command	Description	Example
<code>format short</code>	Fixed-point with 4 decimal digits for: $0.001 \leq number \leq 1000$ Otherwise display format <code>short e</code> .	<pre>&gt;&gt; 290/7 ans =     41.4286</pre>
<code>format long</code>	Fixed-point with 15 decimal digits for: $0.001 \leq number \leq 100$ Otherwise display format <code>long e</code> .	<pre>&gt;&gt; 290/7 ans =     41.428571428571431</pre>
<code>format short e</code>	Scientific notation with 4 decimal digits.	<pre>&gt;&gt; 290/7 ans =     4.1429e+001</pre>
<code>format long e</code>	Scientific notation with 15 decimal digits.	<pre>&gt;&gt; 290/7 ans =     4.142857142857143e+001</pre>
<code>format short g</code>	Best of 5-digit fixed or floating point.	<pre>&gt;&gt; 290/7 ans =     41.429</pre>
<code>format long g</code>	Best of 15-digit fixed or floating point.	<pre>&gt;&gt; 290/7 ans =     41.4285714285714</pre>
<code>format bank</code>	Two decimal digits.	<pre>&gt;&gt; 290/7 ans =     41.43</pre>
<code>format compact</code>	Eliminates blank lines to allow more lines with information displayed on the screen.	
<code>format loose</code>	Adds blank lines (opposite of <code>compact</code> ).	

# 1.5 ELEMENTARY MATH BUILT-IN FUNCTIONS

In addition to basic arithmetic operations, expressions in MATLAB can include functions. MATLAB has a very large library of built-in functions. A function has a name and an argument in parentheses. For example, the function that calculates the square root of a number is `sqrt(x)`. Its name is `sqrt`, and the argument is `x`. When the function is used, the argument can be a number, a variable that has been assigned a numerical value (explained in [Section 1.6](#)), or a computable expression that can be made up of numbers and/or variables. Functions can also be included in arguments, as well as in expressions. [Tutorial 1-2](#) shows examples of using the function `sqrt(x)` when MATLAB is used as a calculator with scalars.

## [Tutorial 1-2](#): Using the `sqrt` built-in function.



```
>> sqrt(64)
ans =
     8

>> sqrt(50+14*3)
ans =
    9.5917

>> sqrt(54+9*sqrt(100))
ans =
    12

>> (15+600/4)/sqrt(121)
ans =
    15

>>
```

Argument is a number.

Argument is an expression.

Argument includes a function.

Function is included in an expression.

Some commonly used elementary MATLAB mathematical built-in functions are given in [Tables 1-3](#) through [1-5](#). A complete list of functions organized by category can be found in the Help Window.

**TABLE 1-3:****Elementary math functions**

Function	Description	Example
<code>sqrt(x)</code>	Square root.	<pre>&gt;&gt; sqrt(81) ans =     9</pre>
<code>nthroot(x,n)</code>	Real $n$ th root of a real number $x$ . (If $x$ is negative $n$ must be an odd integer.)	<pre>&gt;&gt; nthroot(80,5) ans =     2.4022</pre>
<code>exp(x)</code>	Exponential ( $e^x$ ).	<pre>&gt;&gt; exp(5) ans =    148.4132</pre>
<code>abs(x)</code>	Absolute value.	<pre>&gt;&gt; abs(-24) ans =     24</pre>
<code>log(x)</code>	Natural logarithm. Base $e$ logarithm ( $\ln$ ).	<pre>&gt;&gt; log(1000) ans =     6.9078</pre>
<code>log10(x)</code>	Base 10 logarithm.	<pre>&gt;&gt; log10(1000) ans =     3.0000</pre>
<code>factorial(x)</code>	The factorial function $x!$ ( $x$ must be a positive integer.)	<pre>&gt;&gt; factorial(5) ans =    120</pre>

**TABLE 1-4:****Trigonometric math functions**

Function	Description	Example
<code>sin(x)</code> <code>sind(x)</code>	Sine of angle $x$ ( $x$ in radians). Sine of angle $x$ ( $x$ in degrees).	<pre>&gt;&gt; sin(pi/6) ans =     0.5000</pre>
<code>cos(x)</code> <code>cosd(x)</code>	Cosine of angle $x$ ( $x$ in radians). Cosine of angle $x$ ( $x$ in degrees).	<pre>&gt;&gt; cosd(30) ans =     0.8660</pre>
<code>tan(x)</code> <code>tand(x)</code>	Tangent of angle $x$ ( $x$ in radians). Tangent of angle $x$ ( $x$ in degrees).	<pre>&gt;&gt; tan(pi/6) ans =     0.5774</pre>
<code>cot(x)</code> <code>cotd(x)</code>	Cotangent of angle $x$ ( $x$ in radians). Cotangent of angle $x$ ( $x$ in degrees).	<pre>&gt;&gt; cotd(30) ans =     1.7321</pre>

The inverse trigonometric functions are `asin(x)`, `acos(x)`, `atan(x)`, `acot(x)` for the angle in radians; and `asind(x)`, `acosd(x)`, `atand(x)`, `acotd(x)` for the angle in degrees. The hyperbolic trigonometric functions are `sinh(x)`, `cosh(x)`, `tanh(x)`, and `coth(x)`. [Table 1-4](#) uses `pi`, which is equal to  $\pi$  (see [Section 1.6.3](#)).

**TABLE 1-5:****Rounding functions**

Function	Description	Example
<code>round(x)</code>	Round to the nearest integer.	<pre>&gt;&gt; round(17/5) ans =     3</pre>
<code>fix(x)</code>	Round toward zero.	<pre>&gt;&gt; fix(13/5) ans =     2</pre>
<code>ceil(x)</code>	Round toward infinity.	<pre>&gt;&gt; ceil(11/5) ans =     3</pre>
<code>floor(x)</code>	Round toward minus infinity.	<pre>&gt;&gt; floor(-9/4) ans =    -3</pre>
<code>rem(x,y)</code>	Returns the remainder after $x$ is divided by $y$ .	<pre>&gt;&gt; rem(13,5) ans =     3</pre>
<code>sign(x)</code>	Signum function. Returns 1 if $x > 0$ , $-1$ if $x < 0$ , and 0 if $x = 0$ .	<pre>&gt;&gt; sign(5) ans =     1</pre>

## 1.6 DEFINING SCALAR VARIABLES

A variable is a name made of a letter or a combination of several letters (and digits) that is assigned a numerical value. Once a variable is assigned a numerical value, it can be used in mathematical expressions, in functions, and in any MATLAB statements and commands. A variable is actually a name of a memory location. When a new variable is defined, MATLAB allocates an appropriate memory space where the variable's assignment is stored. When the variable is used the stored data is used. If the variable is assigned a new value the content of the memory location is replaced. (In [Chapter 1](#) we consider only variables that are assigned numerical values that are scalars. Assigning and addressing variables that are arrays is discussed in [Chapter 2](#).)

### 1.6.1 The Assignment Operator

In MATLAB the `=` sign is called the assignment operator. The assignment operator assigns a value to a variable.

Variable\_name = A numerical value, or a computable expression

- The left-hand side of the assignment operator can include only one variable name. The right-hand side can be a number, or a computable expression that can include numbers and/or variables that were previously assigned numerical values. When the **Enter** key is pressed the numerical value of the right-hand side is assigned to the variable, and MATLAB displays



the variable and its assigned value in the next two lines.

The following shows how the assignment operator works.

```
>> x=15
x =
    15

>> x=3*x-12
x =
    33
>>
```

The number 15 is assigned to the variable x.

MATLAB displays the variable name and its assigned value.

A new value is assigned to x. The new value is 3 times the previous value of x minus 12.

The last statement ( $x = 3x - 12$ ) illustrates the difference between the assignment operator and the equal sign. If in this statement the = sign meant equal, the value of x would be 6 (solving the equation for x).

The use of previously defined variables to define a new variable is demonstrated next.

```
>> a=12
a =
    12

>> B=4
B =
     4

>> C=(a-B)+40-a/B*10
C =
    18
```

Assign 12 to a.

Assign 4 to B.

Assign the value of the expression on the right-hand side to the variable C.

- If a semicolon is typed at the end of the command, then when the **Enter** key is pressed, MATLAB does not display the variable with its assigned value (the variable still exists and is stored in memory).
- If a variable already exists, typing the variable's name and pressing the **Enter** key will display the variable and its value in the next two lines.

As an example, the last demonstration is repeated below using semicolons.

```
>> a=12;
>> B=4;
>> C=(a-B)+40-a/B*10;

>> C
C =
    18
```

The variables a, B, and C are defined but are not displayed, since a semicolon is typed at the end of each statement.

The value of the variable C is displayed by typing the name of the variable.

- Several assignments can be typed in the same line. The assignments must be separated with a comma (spaces can be added after the comma). When the **Enter** key is pressed, the assignments are executed from left to right and the variables and their assignments are displayed. A variable is not displayed if a semicolon is typed instead of a comma. For example, the assignments of the variables a, b, and c above can all be done in the same

line.

```
>> a=12, B=4; C=(a-B)+40-a/B*10
a =
    12
C =
    18
```

The variable B is not displayed because a semi-colon is typed at the end of the assignment.

- A variable that already exists can be reassigned a new value. For example:

```
>> ABB=72;
>> ABB=9;
>> ABB
ABB =
     9
>>
```

A value of 72 is assigned to the variable ABB.

A new value of 9 is assigned to the variable ABB.

The current value of the variable is displayed when the name of the variable is typed and the Enter key is pressed.

- Once a variable is defined it can be used as an argument in functions. For example:

```
>> x=0.75;
>> E=sin(x)^2+cos(x)^2
E =
     1
>>
```

## 1.6.2 Rules About Variable Names

A variable can be named according to the following rules:

- Must begin with a letter.
- Can be up to 63 characters long.
- Can contain letters, digits, and the underscore character.
- Cannot contain punctuation characters (e.g., period, comma, semicolon).
- MATLAB is case-sensitive: it distinguishes between uppercase and lowercase letters. For example, AA, Aa, aA, and aa are the names of four different variables.
- No spaces are allowed between characters (use the underscore where a space is desired).
- Avoid using the name of a built-in function for a variable (i.e., avoid using cos, sin, exp, sqrt, etc.). Once a function name is used to for a variable name, the function cannot be used.

## 1.6.3 Predefined Variables and Keywords

There are 20 words, called keywords, that are reserved by MATLAB for various purposes and cannot be used as variable names. These words are:

break case catch classdef continue else elseif end for function global if otherwise  
parfor persistent return spmd switch try while

When typed, these words appear in blue. An error message is displayed if the user tries to use a keyword as a variable name. (The keywords can be displayed by typing the command `iskeyword`.)

A number of frequently used variables are already defined when MATLAB is started. Some of the predefined variables are:

ans	A variable that has the value of the last expression that was not assigned to a specific variable (see <a href="#">Tutorial 1-1</a> ). If the user does not assign the value of an expression to a variable, MATLAB automatically stores the result in <code>ans</code> .
pi	The number $\pi$ .
eps	The smallest difference between two numbers. Equal to $2^{(-52)}$ , which is approximately $2.2204e-016$ .
inf	Used for infinity.
i	Defined as $\sqrt{-1}$ , which is: $0 + 1.0000i$ .
j	Same as <code>i</code> .
NaN	Stands for Not-a-Number. Used when MATLAB cannot determine a valid numeric value. Example: $0/0$ .

The predefined variables can be redefined to have any other value. The variables `pi`, `eps`, and `inf`, are usually not redefined since they are frequently used in many applications. Other predefined variables, such as `i` and `j`, are sometime redefined (commonly in association with loops) when complex numbers are not involved in the application.

## 1.7 USEFUL COMMANDS FOR MANAGING VARIABLES

The following are commands that can be used to eliminate variables or to obtain information about variables that have been created. When these commands are typed in the Command Window and the **Enter** key is pressed, either they provide information, or they perform a task as specified below.

Command	Outcome
<code>clear</code>	Removes all variables from the memory.
<code>clear x y z</code>	Removes only variables <code>x</code> , <code>y</code> , and <code>z</code> from the memory.
<code>who</code>	Displays a list of the variables currently in the memory.
<code>whos</code>	Displays a list of the variables currently in the memory and their sizes together with information about their bytes and class (see <a href="#">Section 4.1</a> ).

## 1.8 SCRIPT FILES

So far all the commands were typed in the Command Window and were executed when the **Enter** key was pressed. Although every MATLAB command can be executed in this way, using the Command Window to execute a series of commands—especially if they are related to each



other (a program)—is not convenient and may be difficult or even impossible. The commands in the Command Window cannot be saved and executed again. In addition, the Command Window is not interactive. This means that every time the **Enter** key is pressed only the last command is executed, and everything executed before is unchanged. If a change or a correction is needed in a command that was previously executed and the result of this command is used in commands that follow, all the commands have to be entered and executed again.

A different (better) way of executing commands with MATLAB is first to create a file with a list of commands (program), save it, and then run (execute) the file. When the file runs, the commands it contains are executed in the order that they are listed. If needed, the commands in the file can be corrected or changed and the file can be saved and run again. Files that are used for this purpose are called script files.

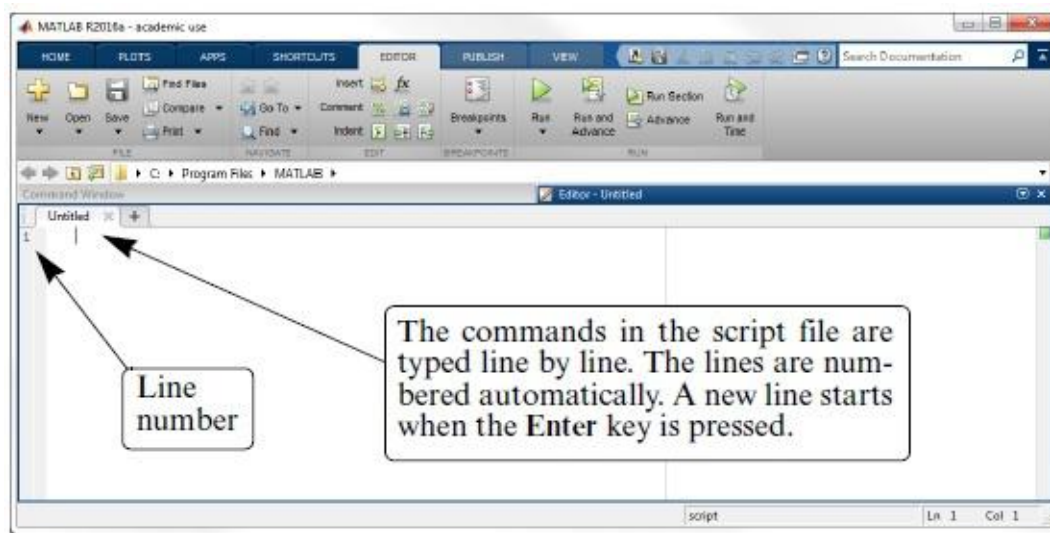
**IMPORTANT NOTE:** This section covers only the minimum required in order to run simple programs. This will allow the student to use script files when practicing the material that is presented in this and the next two chapters (instead of typing repeatedly in the Command Window). Script files are considered again in [Chapter 4](#), where many additional topics that are essential for understanding MATLAB and writing programs in script file are covered.

### 1.8.1 Notes About Script Files

- A script file is a sequence of MATLAB commands, also called a program.
- When a script file runs (is executed), MATLAB executes the commands in the order they are written, just as if they were typed in the Command Window.
- When a script file has a command that generates an output (e.g., assignment of a value to a variable without a semicolon at the end), the output is displayed in the Command Window.
- Using a script file is convenient because it can be edited (corrected or otherwise changed) and executed many times.
- Script files can be typed and edited in any text editor and then pasted into the MATLAB editor.
- Script files are also called M-files because the extension .m is used when they are saved.

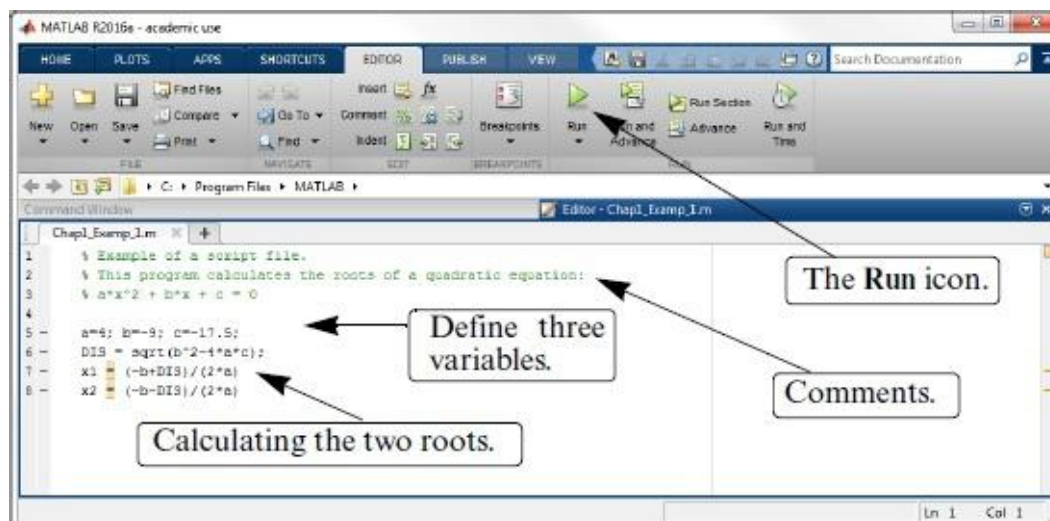
### 1.8.2 Creating and Saving a Script File

In MATLAB script files are created and edited in the Editor/Debugger Window. This window is opened from the Command Window by clicking on the **New Script** icon in the Toolstrip, or by clicking **New** in the Toolstrip and then selecting **Script** from the menu that open. An open Editor/Debugger Window is shown in [Figure 1-6](#).



**Figure 1-6: The Editor/Debugger Window.**

The Editor/Debugger Window has a Toolstrip at the top and three tabs EDITOR, PUBLISH, and VIEW above it. Clicking on the tabs changes the icons in the Toolstrip. Commonly, MATLAB is used with the HOME tab selected. The associated icons are used for executing various commands, as explained later in the Chapter. Once the window is open, the commands of the script file are typed line by line. MATLAB automatically numbers a new line every time the **Enter** key is pressed. The commands can also be typed in any text editor or word processor program and then copied and pasted in the Editor/ Debugger Window. An example of a short program typed in the Editor/Debugger Window is shown in [Figure 1-7](#). The first few lines in a script file are typically comments (which are not executed, since the first character in the line is %) that describe the program written in the script file.



**Figure 1-7: A program typed in the Editor/Debugger Window.**

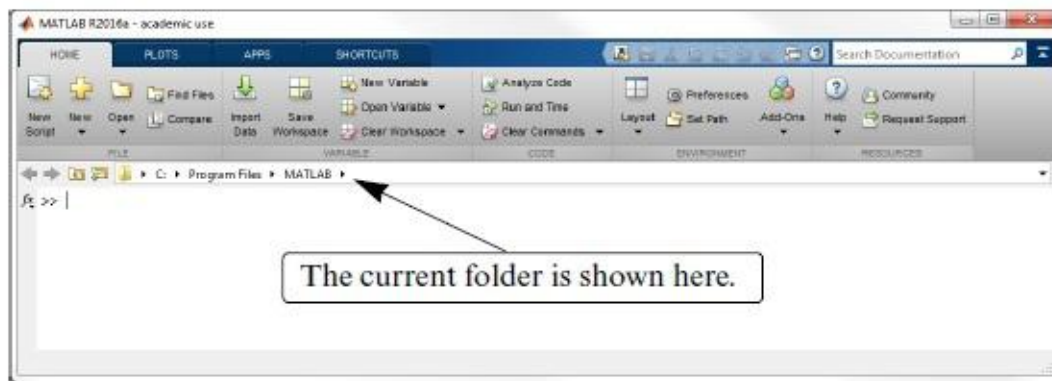
Before a script file can be executed it has to be saved. This is done by clicking **Save** in the Toolstrip and selecting **Save As...** from the menu that opens. When saved, MATLAB adds the extension .m to the name. The rules for naming a script file follow the rules of naming a variable (must begin with a letter, can include digits and underscore, *no spaces*, and up to 63 characters long). The names of user-defined variables, predefined variables, and MATLAB commands or functions should not be used as names of script files.

### 1.8.3 Running (Executing) a Script File

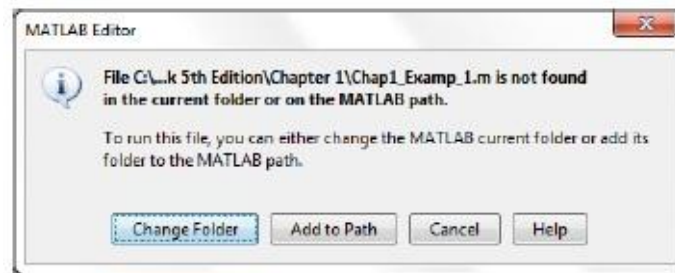
A script file can be executed either directly from the Editor Window by clicking on the **Run** icon (see [Figure 1-7](#)) or by typing the file name in the Command Window and then pressing the **Enter** key. For a file to be executed, MATLAB needs to know where the file is saved. The file will be executed if the folder where the file is saved is the current folder of MATLAB or if the folder is listed in the search path, as explained next.

### 1.8.4 Current Folder

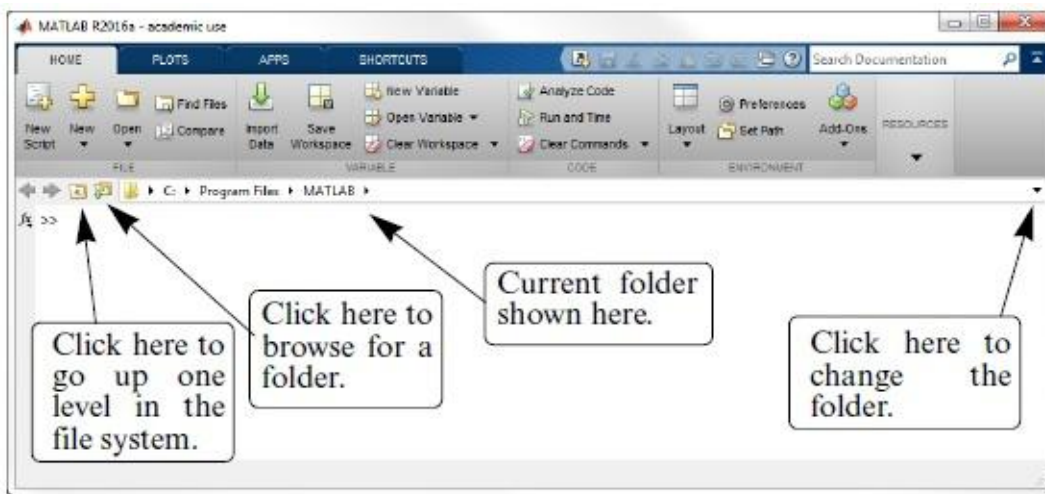
The current folder is shown in the “Current Folder” field in the desktop toolbar of the Command Window, as shown in [Figure 1-8](#). If an attempt is made to execute a script file by clicking on the **Run** icon (in the Editor Window) when the current folder is not the folder where the script file is saved, then the prompt shown in [Figure 1-9](#) opens. The user can then change the current folder to the folder where the script file is saved, or add it to the search path. Once two or more different current folders are used in a session, it is possible to switch from one to another in the **Current Folder** field in the Command Window. The current folder can also be changed in the Current Folder Window, shown in [Figure 1-10](#), which can be opened from the **Desktop** menu. The Current Folder can be changed by choosing the drive and folder where the file is saved.



[Figure 1-8](#): The Current folder field in the Command Window.



[Figure 1-9](#): Changing the current directory.



**Figure 1-10: The Current Folder Window.**

An alternative simple way to change the current folder is to use the `cd` command in the Command Window. To change the current folder to a different drive, type `cd`, space, and then the name of the directory followed by a colon `:` and press the **Enter** key. For example, to change the current folder to drive E (e.g., the flash drive) type `cd E:`. If the script file is saved in a folder within a drive, the path to that folder has to be specified. This is done by typing the path as a string in the `cd` command. For example, `cd('E:\Chapter 1')` sets the path to the folder [Chapter 1](#) in drive E. The following example shows how the current folder is changed to be drive E. Then the script file from [Figure 1-7](#), which was saved in drive E as `ProgramExample.m`, is executed by typing the name of the file and pressing the **Enter** key.

```
>> cd('E:\Chapter 1')
>> Chap1_Examp1
x1 =
    3.5000
x2 =
   -1.2500
```

The current directory is changed to drive E.

The script file is executed by typing the name of the file and pressing the Enter key.

The output generated by the script file (the roots x1 and x2) is displayed in the Command Window.

## 1.9 EXAMPLES OF MATLAB APPLICATIONS

## Sample Problem 1-1: Trigonometric identity

A trigonometric identity is given by:

Error parsing MathML: error on line 1 at column 169: Namespace prefix m on mspace is not defined

$$\cos^2 \frac{x}{2} = \frac{\tan x + \sin x}{2 \tan x}$$

Verify that the identity is correct by calculating each side of the equation, substituting  $x = \frac{\pi}{5}$ .

### Solution

The problem is solved by typing the following commands in the Command Window.

```
>> x=pi/5;
```

Define x.

```
>> LHS=cos(x/2)^2
```

Calculate the left-hand side.

```
LHS =  
0.9045
```

```
>> RHS=(tan(x)+sin(x))/(2*tan(x))
```

Calculate the right-hand side.

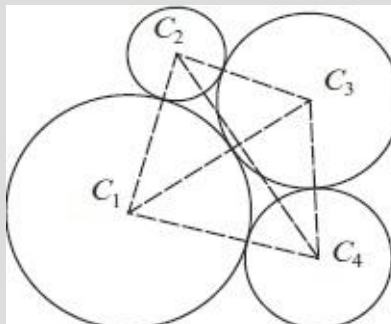
```
RHS =  
0.9045
```

## Sample Problem 1-2: Geometry and trigonometry

Four circles are placed as shown in the figure. At each point where two circles are in contact, they are tangent to each other. Determine the distance between the centers  $C_2$  and  $C_4$ .

The radii of the circles are:

$R_1 = 16$  mm,  $R_2 = 6.5$  mm,  $R_3 = 12$  mm, and  $R_4 = 9.5$  mm.



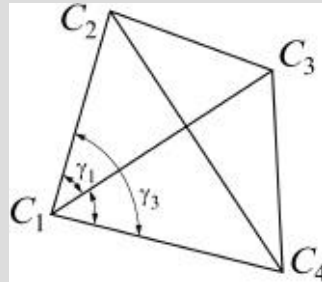
### Solution



The lines that connect the centers of the circles create four triangles. In two of the triangles,  $\Delta C_1C_2C_3$  and  $\Delta C_1C_3C_4$ , the lengths of all the sides are known. This information is used to calculate the angles  $\gamma_1$  and  $\gamma_2$  in these triangles by using the law of cosines. For example,  $\gamma_1$  is calculated from:

Error parsing MathML: error on line 1 at column 761: Namespace prefix m on mspace is not defined

$$(C_2C_3)^2 = (C_1C_2)^2 + (C_1C_3)^2 - (C_1C_2)(C_1C_3)\cos \gamma_1$$



Next, the length of the side  $C_2C_4$  is calculated by considering the triangle  $\Delta C_1C_2C_4$ . This is done, again, by using the law of cosines (the lengths  $C_1C_2$  and  $C_1C_4$  are known and the angle  $\gamma_3$  is the sum of the angles  $\gamma_1$  and  $\gamma_2$ ).

The problem is solved by writing the following program in a script file:

```
% Solution of Sample Problem 1-2
```

```
R1=16; R2=6.5; R3=12; R4=9.5;
```

```
C1C2=R1+R2; C1C3=R1+R3; C1C4=R1+R4;
```

```
C2C3=R2+R3; C3C4=R3+R4;
```

```
Gama1=acos((C1C2^2+C1C3^2-C2C3^2)/(2*C1C2*C1C3));
```

```
Gama2=acos((C1C3^2+C1C4^2-C3C4^2)/(2*C1C3*C1C4));
```

```
Gama3=Gama1+Gama2;
```

```
C2C4=sqrt(C1C2^2+C1C4^2-2*C1C2*C1C4*cos(Gama3))
```

Define the  $R$ 's.

Calculate the lengths of the sides.

Calculate  $\gamma_1$ ,  $\gamma_2$ , and  $\gamma_3$ .

Calculate the length of side  $C_2C_4$ .

When the script file is executed, the following (the value of the variable  $c2c4$ ) is displayed in the Command Window:

```
c2c4 =  
    33.5051
```

## Sample Problem 1-3: Heat transfer

An object with an initial temperature  $T_0$  of that is placed at time  $t = 0$  inside a chamber that has a constant temperature of  $T_s$  will experience a temperature change according to the equation

$$T = T_s + (T_0 - T_s)e^{-kt}$$

$$T = T_s + (T_0 - T_s)e^{-kt}$$

where  $T$  is the temperature of the object at time  $t$ , and  $k$  is a constant. A soda can at a temperature of  $120^\circ\text{F}$  (after being left in the car) is placed inside a refrigerator where the temperature is  $38^\circ\text{F}$ . Determine, to the nearest degree, the temperature of the can after three hours. Assume  $k = 0.45$ . First define all of the variables and then calculate the temperature using one MATLAB command.

### Solution

The problem is solved by typing the following commands in the Command Window.

```
>> Ts=38; T0=120; k=0.45; t=3;
```

```
>> T=round(Ts+(T0-Ts)*exp(-k*t))
```

```
T =  
    59
```

Round to the nearest integer.

## Sample Problem 1-4: Compounded interest

The balance  $B$  of a savings account after  $t$  years when a principal  $P$  is invested at an annual interest rate  $r$  and the interest is compounded  $n$  times a year is given by:

$$B = P \left( 1 + \frac{r}{n} \right)^{nt}$$

$$B = P \left( 1 + \frac{r}{n} \right)^{nt}$$

(1)

If the interest is compounded yearly, the balance is given by:



$$B = P(1 - r)^t$$

$$B = P(1 - r)^t$$

Suppose \$5,000 is invested for 17 years in one account for which the interest is compounded yearly. In addition, \$5,000 is invested in a second account in which the interest is compounded monthly. In both accounts the interest rate is 8.5%. Use MATLAB to determine how long (in years and months) it would take for the balance in the second account to be the same as the balance of the first account after 17 years.

## Solution

Follow these steps:

- Calculate  $B$  for \$5,000 invested in a yearly compounded interest account after 17 years using [Equation \(2\)](#).
- Calculate  $t$  for the  $B$  calculated in part (a), from the monthly compounded interest formula, [Equation \(1\)](#).
- Determine the number of years and months that correspond to  $t$ .

The problem is solved by writing the following program in a script file:

```
% Solution of Sample Problem 1-4
```

```
P=5000; r=0.085; ta=17; n=12;
```

```
B=P*(1+r)^ta
```

Step (a): Calculate B from Eq. (2).

```
t=log(B/P)/(n*log(1+r/n))
```

Step (b): Solve Eq. (1) for t, and calculate t.

```
years=fix(t)
```

Step (c): Determine the number of years.

```
months=ceil((t-years)*12)
```

Determine the number of months.

When the script file is executed, the following (the values of the variables  $B$ ,  $t$ ,  $\text{years}$ , and  $\text{months}$ ) is displayed in the Command Window:

```
>> format short g
```

```
B =
```

```
20011
```

```
t =
```

```
16.374
```

```
years =
```

```
16
```

```
months =
```

```
5
```

The values of the variables  $B$ ,  $t$ ,  $\text{years}$ , and  $\text{months}$  are displayed (since a semicolon was not typed at the end of any of the commands that calculate the values).

# 1.10 PROBLEMS

The following problems can be solved by writing commands in the Command Window or by writing a program in a script file and then executing the file.

1. Calculate:

$$(a) \left(5 - \frac{19}{7} + 2.5^3\right)^2 \left(5 - \frac{19}{7} + 2.5^3\right)^2$$

$$(b) 7 \times 3.1 + \frac{\sqrt{120}}{5} - 15^{5/3} \quad 7 \times 3.1 + \frac{\sqrt{120}}{5} - 15^{5/3}$$

2. Calculate:

$$(a) \sqrt[3]{8 + \frac{80}{2.6}} + e^{3.5} \quad \sqrt[3]{8 + \frac{80}{2.6}} + e^{3.5}$$

$$(b) \left(\frac{1}{\sqrt{75}} + \frac{73}{3.1^3}\right)^{1/4} + 55 \times 0.41 \left(\frac{1}{\sqrt{75}} + \frac{73}{3.1^3}\right)^{1/4} + 55 \times 0.41$$

3. Calculate:

$$(a) \frac{23 + \sqrt[3]{45}}{16 \times 0.7} + \log_{10} 589006 \quad \frac{23 + \sqrt[3]{45}}{16 \times 0.7} + \log_{10} 589006$$

$$(b) (36.1 - 2.25\pi)(e^{2.3} + \sqrt{20}) \quad (36.1 - 2.25\pi)(e^{2.3} + \sqrt{20})$$

4. Calculate:

$$(a) \frac{3.8^2}{2.75 - 41 \times 25} + \frac{5.2 + 1.8^5}{\sqrt{3.5}} \quad \frac{3.8^2}{2.75 - 41 \times 25} + \frac{5.2 + 1.8^5}{\sqrt{3.5}}$$

$$(b) \frac{2.1 \times 10^6 - 15.2 \times 10^5}{3 \cdot \sqrt[3]{6 \times 10^{11}}} \quad \frac{2.1 \times 10^6 - 15.2 \times 10^5}{3 \cdot \sqrt[3]{6 \times 10^{11}}}$$

5. Calculate:

$$(a) \frac{\sin(0.2\pi)}{\cos(\pi/6)} + \tan 72^\circ \quad \frac{\sin(0.2\pi)}{\cos(\pi/6)} + \tan 72^\circ$$

$$(b) (\tan 64^\circ \cos 15^\circ)^2 \frac{\sin^2 37^\circ}{\cos^2 20^\circ} \quad (\tan 64^\circ \cos 15^\circ)^2 + \frac{\sin^2 37^\circ}{\cos^2 20^\circ}$$

6. Define the variable  $z$  as  $z = 4.5$ ; then evaluate:

$$(a) 0.4z^4 + 3.1z^2 - 162.3z - 80.7 \quad 0.4z^4 + 3.1z^2 - 162.3z - 80.7$$

$$(b) (z^3 - 23)/(\sqrt[3]{z^2 + 17.5}) \quad (z^3 - 23)/(\sqrt[3]{z^2 + 17.5})$$

7. Define the variable  $t$  as  $t = 3.2$ ; then evaluate:

$$(a) \frac{1}{2} e^{2t} - 3.81t^3 \quad \frac{1}{2} e^{2t} - 3.81t^3$$

$$(b) \frac{6t^2+6t-2}{t^2-1} - \frac{6t^2+6t-2}{t^2-1}$$

8. Define the variables  $x$  and  $y$  as  $x = 6.5$  and  $y = 3.8$ ; then evaluate:

$$(a) (x^2+y^2)^{2/3} + \frac{xy}{y-x} - (x^2+y^2)^{2/3} + \frac{xy}{y-x}$$

$$(b) \frac{\sqrt{x+y}}{(x-y)^2} + 2x^2 - xy^2 - \frac{\sqrt{x+y}}{(x-y)^2} + 2x^2 - xy^2$$

9. Define the variables  $a$ ,  $b$ ,  $c$ , and  $d$  as:

$$c = 4.6, d = 1.7, a = cd^2, \text{ and } b = \frac{c+a}{c-d}; \quad b = \frac{c+a}{c-d}; \text{ then evaluate:}$$

(a)

Error parsing MathML: error on line 1 at column 37: Namespace prefix m on mrow is not defined

$$e^{d-b} + \sqrt[3]{c+a} - (ca)^d$$

$$(b) \frac{d}{c} + \left(\frac{ct}{b}\right)^2 - c^d - \frac{a}{b} - \frac{d}{c} + \left(\frac{ct}{b}\right)^2 - c^d - \frac{a}{b}$$

10. Two trigonometric identities are given by:

$$(a) \cos^2 x - \sin^2 x = 1 - 2\sin^2 x \quad \cos^2 x - \sin^2 x = 1 - 2\sin^2 x$$

$$(b) \frac{\tan x}{\sin x - 2\tan x} = \frac{1}{\cos x - 2} \quad \frac{\tan x}{\sin x - 2\tan x} = \frac{1}{\cos x - 2}$$

For each part, verify that the identity is correct by calculating the values of the left and right sides of the equation, substituting  $x = \pi / 10$ .

11. Two trigonometric identities are given by:

$$(a) (\sin x + \cos x)^2 = 1 + 2\sin x \cos x \quad (\sin x + \cos x)^2 = 1 + 2\sin x \cos x$$

$$(b) \frac{1-2\cos x-3\cos^2 x}{\sin^2 x} = \frac{1-3\cos x}{1-\cos x} \quad \frac{1-2\cos x-3\cos^2 x}{\sin^2 x} = \frac{1-3\cos x}{1-\cos x}$$

For each part, verify that the identity is correct by calculating the values of the left and right sides of the equation, substituting  $x = 20^\circ$ .

12. Define two variables:  $\alpha = \pi/8$ , and  $\beta = \pi/6$ . Using these variables, show that the following trigonometric identity is correct by calculating the values of the left and right sides of the equation.

$$\tan(\alpha + \beta) = \frac{\tan\alpha + \tan\beta}{1 - \tan\alpha\tan\beta}$$

$$\tan(\alpha + \beta) = \frac{\tan \alpha + \tan \beta}{1 - \tan \alpha \tan \beta}$$

13. Given:  $\int x^2 \cos x \, dx = 2x \cos x + (x^2 - 2) \sin x$ . Use

MATLAB to calculate the following definite integral:  $\int_{\pi/6}^{\pi/3} x^2 \cos x \, dx$ .

14. A rectangular box has the dimensions shown.

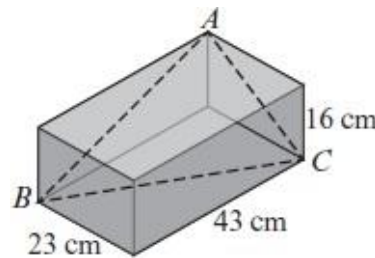
(a) Determine the angle  $BAC$  to the nearest degree.

(b) Determine the area of the triangle  $ABC$  to the nearest tenth of a centimeter.

Law of cosines:  $c^2 = a^2 + b^2 - 2ab \cos \gamma$

Heron's formula for triangular area:

$$A = \sqrt{p(p-a)(p-b)(p-c)}, \text{ where } p = (a+b+c)/2.$$

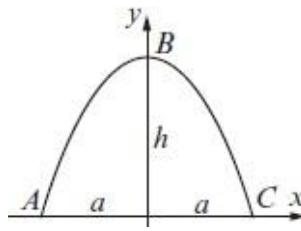


15. The arc length of a segment of a parabola  $ABC$  is given by:

$$L_{ABC} = \sqrt{a^2 + 4h^2} + \frac{a^2}{2h} \ln \left( \frac{2h}{a} + \sqrt{\left(\frac{2h}{a}\right)^2 + 1} \right)$$

$$L_{ABC} = \sqrt{a^2 + 4h^2} + \frac{a^2}{2h} \ln \left( \frac{2h}{a} + \sqrt{\left(\frac{2h}{a}\right)^2 + 1} \right)$$

Determine  $L_{ABC}$  if  $a=8$  in. and  $h=13$  in.



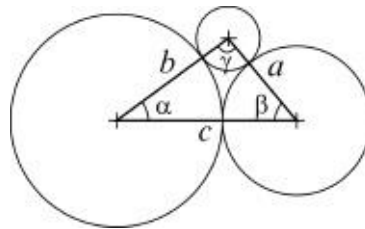
16. The three shown circles, with radius 15 in., 10.5 in., and 4.5 in., are tangent to each other.

(a) Calculate the angle  $\gamma$  (in degrees) by using the law of cosines.

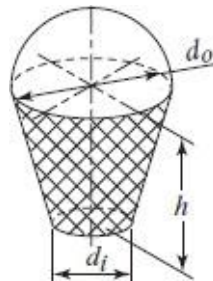
(Law of cosines:  $c^2 = a^2 + b^2 - 2ab \cos \gamma$ )

(b) Calculate the angles  $\gamma$  and  $\alpha$  (in degrees) using the law of sines.

(c) Check that the sum of the angles is  $180^\circ$ .



17. A frustum of cone is filled with ice cream such that the portion above the cone is a hemisphere. Define the variables  $d_i=1.25$  in.,  $d_o=2.25$  in.,  $h=2$  in., and determine the volume of the ice cream.



18. In the triangle shown  $a = 27$  in.,  $b = 43$  in., and  $c = 57$  in. Define  $a$ ,  $b$ , and  $c$  as variables, and then:

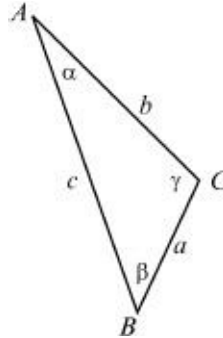
(a) Calculate the angles  $\alpha$ ,  $\beta$ , and  $\gamma$  by substituting the variables in the law of cosines.

(Law of cosines:  $c^2 = a^2 + b^2 - 2ab \cos \gamma$ )

(b) Verify the law of tangents by substituting the results into the right and left sides of:

$$\text{law of tangents : } \frac{b-c}{b+c} = \frac{\tan \left[ \frac{1}{2} (\beta - \gamma) \right]}{\tan \left[ \frac{1}{2} (\beta + \gamma) \right]}$$

$$\text{law of tangents: } \frac{b-c}{b+c} = \frac{\tan \left[ \frac{1}{2} (\beta - \gamma) \right]}{\tan \left[ \frac{1}{2} (\beta + \gamma) \right]}$$



19. For the triangle shown,  $\alpha = 72^\circ$ ,  $\beta = 43^\circ$ , and its perimeter is  $p = 114$  mm. Define  $\alpha$ ,  $\beta$ , and  $p$ , as variables, and then:

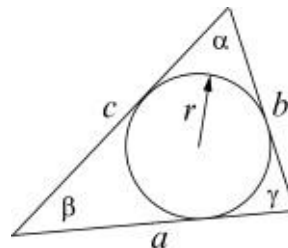
(a) Calculate the triangle sides (Use the law of sines).

(b) Calculate the radius  $r$  of the circle inscribed in the triangle using the formula:

$$r = \sqrt{\frac{(s-a)(s-b)(s-c)}{s}}$$

$$r = \sqrt{\frac{(s-a)(s-b)(s-c)}{s}}$$

where  $s = (a + b + c) / 2$ .



20. The distance  $d$  from a point  $P (x_p, y_p, z_p)$  to the line that passes through the two points  $A (x_A, y_A, z_A)$  and  $B (x_B, y_B, z_B)$  can be calculated by  $d = 2S / r$  where  $r$  is the distance



between the points A and B, given by  $r = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}$   
 $r = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}$  and S is the area of the triangle defined by the three points  
 calculated by  $S = \sqrt{s_1^2 + s_2^2 + s_3^2} S = \sqrt{s_1^2 + s_2^2 + s_3^2}$  where

$$s_1 = x_P y_A + x_A y_B + x_B y_P - (y_P x_A + y_A x_B + y_B x_P)$$

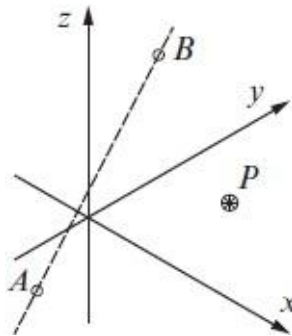
$$s_2 = y_P z_A + y_A z_B + y_B z_P - (z_P y_A + z_A y_B + z_B y_P)$$

$$s_3 = x_P z_A + x_A z_B + x_B z_P - (z_P x_A + z_A x_B + z_B x_P)$$

$$s_1 = x_P y_A + x_A y_B + x_B y_P - (y_P x_A + y_A x_B + y_B x_P)$$

$$s_2 = y_P z_A + y_A z_B + y_B z_P - (z_P y_A + z_A y_B + z_B y_P)$$

$s_3 = x_P z_A + x_A z_B + x_B z_P - (z_P x_A + z_A x_B + z_B x_P)$ . Determine the distance of point P (2, 6, -1) from the line that passes through point A(-2, -1.5, -3) and point B(-2.5, 6, 4). First define the variables  $x_P, y_P, z_P, x_A, y_A, z_A, x_B, y_B,$  and  $z_B$ , and then use the variable to calculate  $s_1, s_2, s_3$ , and  $r$ . Finally calculate S and  $d$ .

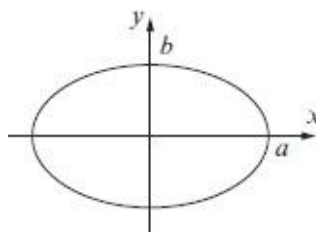


21. The perimeter of an ellipse can be approximated by:

$$P = \pi(a + b) \left( 3 - \frac{\sqrt{(3a + b)(a + 3b)}}{a + b} \right)$$

$$P = \pi(a + b) \left( 3 - \frac{\sqrt{(3a + b)(a + 3b)}}{a + b} \right)$$

Calculate the perimeter of an ellipse with  $a = 18$  in. and  $b = 7$  in.



22. A total of 4217 eggs have to be packed in boxes that can hold 36 eggs each. By typing one line (command) in the Command Window, calculate how many eggs will remain

unpacked if every box that is used has to be full. (Hint: Use MATLAB built-in function `fix`.)

23. A total of 777 people have to be transported using buses that have 46 seats and vans that have 12 seats. Calculate how many buses are needed if all the buses have to be full, and how many seats will remain empty in the vans if enough vans are used to transport all the people that did not fit into the buses. (Hint: Use MATLAB built-in functions `fix` and `ceil`.)

24. Change the display to `format long g`. Assign the number  $7E8/13$  to a variable, and then use the variable in a mathematical expression to calculate the following by typing one command:

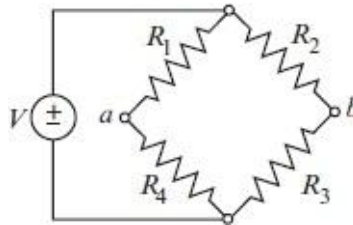
(a) Round the number to the nearest tenth.

(b) Round the number to the nearest million.

25. The voltage difference  $V_{ab}$  between points  $a$  and  $b$  in the Wheatstone bridge circuit is given by:

$$V_{ab} = V \left( \frac{c - d}{(c + 1)(d + 1)} \right)$$
$$V_{ab} = V \left( \frac{c - d}{(c + 1)(d + 1)} \right)$$

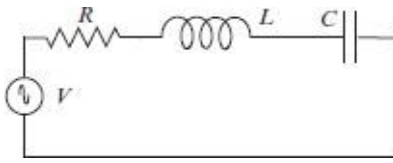
where  $c = R_2 / R_1$  and  $d = R_3 / R_4$ . Calculate the  $V_{ab}$  if  $V = 15 \text{ V}$ ,  $R_1 = 119.8 \text{ } \Omega$ ,  $R_2 = 120.5 \text{ } \Omega$ ,  $R_3 = 121.2 \text{ } \Omega$ , and  $R_4 = 119.3 \text{ } \Omega$



26. The current in a series  $RCL$  circuit is given by:

$$I = \frac{V}{\sqrt{R^2 + \left(\omega L - \frac{1}{\omega C}\right)^2}}$$
$$I = \frac{V}{\sqrt{R^2 + \left(\omega L - \frac{1}{\omega C}\right)^2}}$$

where  $\omega = 2\pi f$ . Calculate  $I$  for the circuit shown if the supply voltage is  $80 \text{ V}$ ,  $f = 50 \text{ Hz}$ ,  $R = 6 \text{ } \Omega$ ,  $L = 400 \times 10^{-3} \text{ H}$ , and  $C = 40 \times 10^{-6} \text{ F}$ .



27. The monthly payment  $M$  of a mortgage  $P$  for  $n$  years with a fixed annual interest rate  $r$  can be calculated by the formula:

$$M = P \frac{\frac{r}{12} + \left(1 + \frac{r}{12}\right)^{12n}}{\left(1 + \frac{r}{12}\right)^{12n} - 1}$$

$$M = P \frac{\frac{r}{12} \left(1 + \frac{r}{12}\right)^{12n}}{\left(1 + \frac{r}{12}\right)^{12n} - 1}$$

Determine the monthly payment of a 30-year \$450,000 mortgage with interest rate of 4.2% ( $r = 0.042$ ). Define the variables  $P$ ,  $r$ , and  $n$  and then use them in the formula to calculate  $M$ .

28. The number of permutations  ${}_nP_r$  of taking  $r$  objects out of  $n$  objects without repetition is given by:

$${}_nP_r = \frac{n!}{(n-r)!}$$

$${}_nP_r = \frac{n!}{(n-r)!}$$

(a) Determine how many six-letter passwords can be formed from the 26 letters in the English alphabet if a letter can only be used once.

(b) How many passwords can be formed if the digits 0, 1, 2, ..., 9 can be used in addition to the letters.

29. The number of combinations  $C_{n,r}$  of taking  $r$  objects out of  $n$  objects is given by:

$$C_{n,r} = \frac{n!}{r!(n-r)!}$$

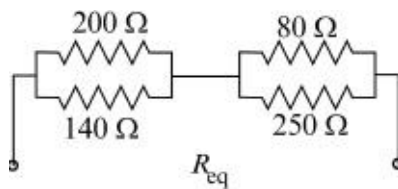
$$C_{n,r} = \frac{n!}{r!(n-r)!}$$

In the Powerball lottery game the player chooses five numbers from 1 through 59, and then the Powerball number from 1 through 35.

Determine how many combinations are possible by calculating  $C_{59,5} C_{35,1}$ . (Use the built-in function `factorial`.)

30. The equivalent resistance of two resistors  $R_1$  and  $R_2$  connected in parallel is given by

$R_{eq} = \frac{R_1 R_2}{R_1 + R_2}$ . The equivalent resistance of two resistors  $R_1$  and  $R_2$  connected in series is given by  $R_{eq} = R_1 + R_2$ . Determine the equivalent resistance of the four resistors in the circuit shown in the figure.

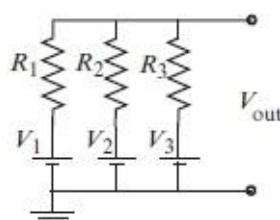


31. The output voltage  $V_{out}$  in the circuit shown is given by (Millman's theorem):

$$V_{out} = \frac{\frac{V_1}{R_1} + \frac{V_2}{R_2} + \frac{V_3}{R_3}}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}}$$

$$V_{out} = \frac{\frac{V_1}{R_1} + \frac{V_2}{R_2} + \frac{V_3}{R_3}}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}}$$

Calculate  $V_{out}$  given  $V_1 = 36$  V,  $V_2 = 28$  V,  $V_3 = 24$  V,  $R_1 = 400$  Ω,  $R_2 = 200$  Ω,  $R_3 = 600$  Ω.



32. Radioactive decay of carbon-14 is used for estimating the age of organic material. The decay is modeled with the exponential function  $f(t) = f(0)e^{kt}$ , where  $t$  is

time,  $f(0)$  is the amount of material at  $t = 0$ ,  $f(t)$  is the amount of material at time  $t$ , and  $k$  is a constant. Carbon-14 has a half-life of approximately 5,730 years. A sample taken from the ancient footprints of Acahualinca in Nicaragua shows that 77.45% of the initial ( $t = 0$ ) carbon-14 is present. Determine the estimated age of the footprint. Solve the problem by writing a program in a script file. The program first determines the constant  $k$ , then calculates  $t$  for  $f(t) = 0.7745 f(0)$ , and finally rounds the answer to the nearest year.

33. The greatest common divisor is the largest positive integer that divides the numbers without a remainder. For example, the greatest common divisor of 8 and 12 is 4. Use the MATLAB Help Window to find a MATLAB built-in function that determines the greatest common divisor of two numbers. Then use the function to show that the greatest common divisor of:

(a) 91 and 147 is 7.

(b) 555 and 962 is 37.

34. The amount of energy  $E$  (in joules) that is released by an earthquake is given by:

$$E = 1.74 \times 10^{19} \times 10^{1.44M}$$

$$E = 1.74 \times 10^{19} \times 10^{1.44M}$$

where  $M$  is the magnitude of the earthquake on the Richter scale.

(a) Determine the energy that was released from the Anchorage earthquake (1964, Alaska, USA), magnitude 9.2.

(b) The energy released in Lisbon earthquake (Portugal) in 1755 was onehalf the energy released in the Anchorage earthquake. Determine the magnitude of the earthquake in Lisbon on the Richter scale.

35. According to the Doppler effect of light, the perceived wavelength  $\lambda_p$  of a light source with a wavelength of  $\lambda_s$  of is given by:

$$\lambda_P = \lambda_S \sqrt{\frac{1 - \frac{v}{c}}{1 + \frac{v}{c}}}$$

$$\lambda_P = \lambda_S \sqrt{\frac{1 - \frac{v}{c}}{1 + \frac{v}{c}}}$$

where  $c$  is the speed of light (about  $300 \times 10^6$  m/s) and  $v$  is the speed the observer moves toward the light source. Calculate the speed the observer has to move in order to see a red light as green. Green wavelength is 530 nm and red wavelength is 630 nm.

36. Newton's law of cooling gives the temperature  $T(t)$  of an object at time  $t$  in terms of  $T_0$ , its temperature at  $t = 0$ , and  $T_s$ , the temperature of the surroundings.

$$T(t) = T_s + (T_0 - T_s)e^{-kt}$$

$$T(t) = T_s + (T_0 - T_s)e^{-kt}$$

A police officer arrives at a crime scene in a hotel room at 9:18 PM, where he finds a dead body. He immediately measures the body's temperature and finds it to be 79.5°F. Exactly one hour later he measures the temperature again and finds it to be 78.0°F. Determine the time of death, assuming that victim body temperature was normal (98.6°F) prior to death and that the room temperature was constant at 69°F.

37. The velocity  $v$  and the falling distance  $d$  as a function of time of a skydiver that experience the air resistance can be approximated by:

$$v(t) = \sqrt{\frac{mg}{k}} \tanh\left(\sqrt{\frac{kg}{m}} t\right) \text{ and } d(t) = \frac{m}{k} \ln\left[\cosh\left(\sqrt{\frac{kg}{m}} t\right)\right]$$

$$v(t) = \sqrt{\frac{mg}{k}} \tanh\left(\sqrt{\frac{kg}{m}} t\right) \text{ and } d(t) = \frac{m}{k} \ln\left[\cosh\left(\sqrt{\frac{kg}{m}} t\right)\right]$$

where  $k = 0.24 \text{ kg/m}$  is a constant,  $m$  is the skydiver mass,  $g = 9.81 \text{ m/s}^2$  is the acceleration due to gravity, and  $t$  is the time in seconds since the skydiver starts falling. Determine the velocity and the falling distance at  $t = 8 \text{ s}$  for a 95-kg skydiver

38. Use the Help Window to find a display format that displays the output as a ratio of integers. For example, the number 3.125 will be displayed as 25/8. Change the display to this format and execute the following operations:

(a)  $5/8 + 16/6$   $5/8 + 16/6$

(b)  $1/3 - 11/13 + 2.7^2$   $1/3 - 11/13 + 2.7^2$

39. Gosper's approximation for factorials is given by:



$$n! = \sqrt{\left(2n + \frac{1}{3}\right) \pi n^n e^{-n}}$$

$$n! = \sqrt{\left(2n + \frac{1}{3}\right) \pi n^n e^{-n}}$$

Use the formula for calculating 19!. Compare the result with the true value obtained with MATLAB's built-in function `factorial` by calculating the error ( $Error = (TrueVal - ApproxVal) / TrueVal$ ).

40. According to Newton's law of universal gravitation, the attraction force between two bodies is given by:

$$F = G \frac{m_1 m_2}{r^2}$$

$$F = G \frac{m_1 m_2}{r^2}$$

where  $m_1$  and  $m_2$  are the masses of the bodies,  $r$  is the distance between the bodies, and  $G = 6.67 \times 10^{-11} \text{ N-m}^2/\text{kg}^2$  is the universal gravitational constant. Determine how many times the attraction force between the sun and the Earth is larger than the attraction force between the Earth and the moon. The distance between the sun and Earth is  $149 \times 10^9 \text{ m}$ , the distance between the moon and Earth is  $384.4 \times 10^6 \text{ m}$ ,  $m_{Earth} = 5.98 \times 10^{28} \text{ kg}$ ,  $m_{sun} = 2.0 \times 10^{30} \text{ kg}$ , and  $m_{moon} = 7.36 \times 10^{22} \text{ kg}$ .

## CHAPTER 2

# CREATING ARRAYS

The array is a fundamental form that MATLAB uses to store and manipulate data. An array is a list of numbers arranged in rows and/or columns. The simplest array (one-dimensional) is a row or a column of numbers. A more complex array (two-dimensional) is a collection of numbers arranged in rows and columns. One use of arrays is to store information and data, as in a table. In science and engineering, one-dimensional arrays frequently represent vectors, and two-dimensional arrays often represent matrices. This chapter shows how to create and address arrays, and [Chapter 3](#) shows how to use arrays in mathematical operations. In addition to arrays made of numbers, arrays in MATLAB can also be a list of characters, which are called strings. Strings are discussed in [Section 2.10](#).

## 2.1 CREATING A ONE-DIMENSIONAL ARRAY (VECTOR)

A one-dimensional array is a list of numbers arranged in a row or a column. One example is the representation of the position of a point in space in a three-dimensional Cartesian coordinate system. As shown in [Figure 2-1](#), the position of point *A* is defined by a list of the three numbers 2, 4, and 5, which are the coordinates of the point.

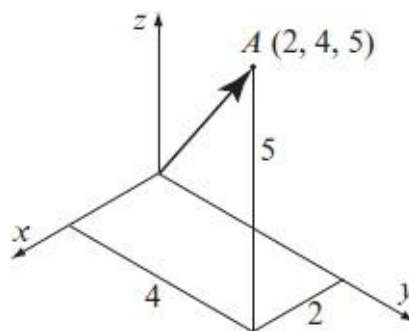
The position of point *A* can be expressed in terms of a position vector:

$$\mathbf{r}_A = 2\mathbf{i} + 4\mathbf{j} + 5\mathbf{k}$$

$$\mathbf{r}_A = 2\mathbf{i} + 4\mathbf{j} + 5\mathbf{k}$$

where **i**, **j**, and **k** are unit vectors in the direction of the *x*, *y*, and *z* axes, respectively. The numbers 2, 4, and 5 can be used to define a row or a column vector.

Any list of numbers can be set up as a vector. For example, [Table 2-1](#) contains population growth data that can be used to create two lists of numbers—one of the years and the other of the population values. Each list can be entered as elements in a vector with the numbers placed in a row or in a column.



**Figure 2-1:** Position of a point.

**TABLE 2-1:****Population data**

Year	1984	1986	1988	1990	1992	1994	1996
Population (millions)	127	130	136	145	158	178	211

In MATLAB, a vector is created by assigning the elements of the vector to a variable. This can be done in several ways depending on the source of the information that is used for the elements of the vector. When a vector contains specific numbers that are known (like the coordinates of point A), the value of each element is entered directly. Each element can also be a mathematical expression that can include predefined variables, numbers, and functions. Often, the elements of a row vector are a series of numbers with constant spacing. In such cases the vector can be created with MATLAB commands. A vector can also be created as the result of mathematical operations as explained in [Chapter 3](#).

Creating a vector from a known list of numbers:

The vector is created by typing the elements (numbers) inside square brackets [ ].

```
variable_name = [ type vector elements ]
```

**Row vector:** To create a row vector type the elements with a space or a comma between the elements inside the square brackets.

**Column vector:** To create a column vector type the left square bracket [ and then enter the elements with a semicolon between them, or press the **Enter** key after each element. Type the right square bracket ] after the last element.

[Tutorial 2-1](#) shows how the data from [Table 2-1](#) and the coordinates of point A are used to create row and column vectors.

## Tutorial 2-1: Creating vectors from given data.

```
>> yr=[1984 1986 1988 1990 1992 1994 1996]
```

The list of years is assigned to a row vector named yr.

```
yr =  
    1984    1986    1988    1990    1992    1994  
1996
```

```
>> pop=[127; 130; 136; 145; 158; 178; 211]
```

The population data is assigned to a column vector named pop.

```
pop =  
    127  
    130  
    136  
    145  
    158  
    178  
    211
```

```
    178  
    211
```

```
>> pntAH=[2, 4, 5]
```

The coordinates of point *A* are assigned to a row vector called pntAH.

```
pntAH =  
     2     4     5
```

```
>> pntAV=[2  
4  
5]
```

The coordinates of point *A* are assigned to a column vector called pntAV. (The Enter key is pressed after each element is typed.)

```
pntAV =  
     2  
     4  
     5  
  
>>
```

Creating a vector with constant spacing by specifying the first term, the spacing, and the last term:

In a vector with constant spacing, the difference between the elements is the same. For example, in the vector  $v = 2\ 4\ 6\ 8\ 10$ , the spacing between the elements is 2. A vector in which the first term is  $m$ , the spacing is  $q$ , and the last term is  $n$  is created by typing:

`variable_name = [m:q:n]`

or

`variable_name = m:q:n`

(The brackets are optional.)

Some examples are:

```

>> x=[1:2:13]
x =
     1     3     5     7     9    11    13
>> y=[1.5:0.1:2.1]
y =
    1.5000    1.6000    1.7000    1.8000    1.9000    2.0000
    2.1000
>> z=[-3:7]
z =
    -3    -2    -1     0     1     2     3     4     5     6
     7
>> xa=[21:-3:6]
xa =
    21    18    15    12     9     6
>>

```

First element 1, spacing 2, last element 13.

First element 1.5, spacing 0.1, last element 2.1.

First element -3, last term 7.  
If spacing is omitted, the default is 1.

First element 21, spacing -3, last term 6.

- If the numbers  $m$ ,  $q$ , and  $n$  are such that the value of  $n$  cannot be obtained by adding  $q$ 's to  $m$ , then (for positive  $n$ ) the last element in the vector will be the last number that does not exceed  $n$ .
- If only two numbers (the first and the last terms) are typed (the spacing is omitted), then the default for the spacing is 1.

Creating a vector with linear (equal) spacing by specifying the first and last terms, and the number of terms:

A vector with  $n$  elements that are linearly (equally) spaced in which the first element is  $xi$  and the last element is  $xf$  can be created by typing the `linspace` command (MATLAB determines the correct spacing):

```

variable_name = linspace(xi,xf,n)

```

First element      Last element      Number of elements

When the number of elements is omitted, the default is 100. Some examples are:

```
>> va=linspace(0,8,6)
va =
    0    1.6000    3.2000    4.8000    6.4000    8.0000
>> vb=linspace(30,10,11)
vb =
    30    28    26    24    22    20    18    16    14    12    10
>> u=linspace(49.5,0.5)
u =
    Columns 1 through 10
    49.5000    49.0051    48.5101    48.0152    47.5202    47.0253
    46.5303    46.0354    45.5404    45.0455
    .....
    Columns 91 through 100
         4.9545         4.4596         3.9646         3.4697         2.9747         2.4798
    1.9848         1.4899         0.9949         0.5000
>>
```

6 elements, first element 0, last element 8.

11 elements, first element 30, last element 10.

First element 49.5, last element 0.5.

When the number of elements is omitted, the default is 100.

100 elements are displayed.

## 2.2 CREATING A TWO-DIMENSIONAL ARRAY (MATRIX)

A two-dimensional array, also called a matrix, has numbers in rows and columns. Matrices can be used to store information like the arrangement in a table. Matrices play an important role in linear algebra and are used in science and engineering to describe many physical quantities.

In a square matrix the number of rows and the number of columns is equal. For example, the matrix

$$\begin{bmatrix} 7 & 4 & 9 \\ 3 & 8 & 1 \\ 6 & 5 & 3 \end{bmatrix} \times \begin{bmatrix} 3 & 8 & 1 \\ 6 & 5 & 3 \end{bmatrix} \text{ matrix}$$

is square, with three rows and three columns. In general, the number of rows and columns can be different. For example, the matrix:



31	26	14	18	5	30						
3	51	20	11	43	65	4	×	6	matrix		
28	6	15	61	34	22						
14	58	6	36	93	7						

31	26	14	18	5	30	
3	51	20	11	43	65	4 × 6 matrix
28	6	15	61	34	22	
14	58	6	36	93	7	

has four rows and six columns. A  $m \times n$  matrix has  $m$  rows and  $n$  columns, and  $m$  by  $n$  is called the size of the matrix.

A matrix is created by assigning the elements of the matrix to a variable. This is done by typing the elements, row by row, inside square brackets `[]`. First type the left bracket `[` then type the first row, separating the elements with spaces or commas. To type the next row type a semicolon or press **Enter**. Type the right bracket `]` at the end of the last row.

```
variable_name=[1st row elements; 2nd row elements; 3rd
               row elements; ... ;last row elements]
```

The elements that are entered can be numbers or mathematical expressions that may include numbers, predefined variables, and functions. *All the rows must have the same number of elements*. If an element is zero, it has to be entered as such. MATLAB displays an error message if an attempt is made to define an incomplete matrix. Examples of matrices defined in different ways are shown in [Tutorial 2-2](#).

**Tutorial 2-2: Creating matrices.**

```
>> a=[5 35 43; 4 76 81; 21 32 40]
a =
     5     35     43
     4     76     81
    21     32     40
```

*A semicolon is typed before a new line is entered.*

```
>> b = [7 2 76 33 8
1 98 6 25 6
5 54 68 9 0]
```

*The Enter key is pressed before a new line is entered.*

```
b =
     7     2     76     33     8
     1    98     6     25     6
     5    54    68     9     0
```

```
>> cd=6; e=3; h=4;
>> Mat=[e, cd*h, cos(pi/3); h^2, sqrt(h*h/cd), 14]
```

*Three variables are defined.*

```
Mat =
    3.0000    24.0000    0.5000
   16.0000    1.6330   14.0000
```

*Elements are defined by mathematical expressions.*

```
>>
```