

MATLAB® HANDBOOK

WITH APPLICATIONS TO
MATHEMATICS, SCIENCE,
ENGINEERING, AND FINANCE



DAVID BÁEZ-LÓPEZ
DAVID ALFREDO BÁEZ VILLEGAS



CRC Press
Taylor & Francis Group

A CHAPMAN & HALL BOOK

MATLAB[®] Handbook with Applications to Mathematics, Science, Engineering, and Finance

José Miguel David Báez-López
David Alfredo Báez Villegas



CRC Press

Taylor & Francis Group

Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

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

© 2019 by Taylor & Francis Group, LLC
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works

Printed on acid-free paper

International Standard Book Number-13: 978-1-138-62645-4 (Hardback)

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

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

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

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

Library of Congress Cataloging-in-Publication Data

Names: Báez López, David, author. | Báez Villegas, David Alfredo, author.
Title: MATLAB handbook with applications to mathematics, science, engineering,
and finance / José Miguel David Báez-López and David Alfredo Báez Villegas.
Description: Boca Raton, Florida : CRC Press, [2019].
Identifiers: LCCN 2018030135 | ISBN 9781138626454 (hardback : alk. paper) |
ISBN 9781315228457 (ebook).
Subjects: LCSH: Numerical analysis--Data processing. | MATLAB.
Classification: LCC QA297 .B27 2019 | DDC 510.285/536--dc23
LC record available at <https://lccn.loc.gov/2018030135>

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

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

To Gary and Laura, my children and Ofelia, my wife.
David Báez-López



Taylor & Francis

Taylor & Francis Group

<http://taylorandfrancis.com>

Contents

Preface	xi
List of Figures	xiii
List of Tables	xix
1 Introduction to MATLAB	1
1.1 Introduction	1
1.1.1 Book Organization	2
1.1.2 Chapter Organization	2
1.2 Starting MATLAB	3
1.3 Simple Calculations in MATLAB	4
1.3.1 Elementary Functions	5
1.4 Variables	7
1.4.1 Variable Types	9
1.5 Strings	10
1.6 Saving a Session and Its Variables	14
1.7 Input/Output Instructions	16
1.7.1 Formatted Output	16
1.7.2 Data Input	18
1.8 Help	19
1.8.1 Help Page	20
1.9 Concluding Remarks	22
1.10 Bibliography	22
2 Plotting with MATLAB	23
2.1 Introduction	23
2.2 Two-dimensional Plotting	24
2.2.1 Plotting from the Workspace	30
2.3 Plot Options	33
2.4 Other Two-dimensional Plots	36
2.4.1 Polar Plots	37
2.4.2 Bar Plot	38
2.4.3 Stairs Plot	38

2.4.4	Histogram Plot	40
2.4.5	Stem Plot	40
2.4.6	Compass Plot	41
2.4.7	Pie Plot	41
2.5	Subplots	43
2.6	Three-dimensional Plots	43
2.6.1	The Instruction <code>plot3</code>	44
2.6.2	Mesh Plot	45
2.6.3	Surf Plot	49
2.6.4	Contour Plot	51
2.7	Observation Point	56
2.8	Structure of Objects in MATLAB	57
2.9	Hierarchy of MATLAB Objects	59
2.10	Concluding Remarks	60
3	Variables and Functions	61
3.1	Variables	61
3.1.1	Symbolic Variables	62
3.2	Functions	64
3.2.1	MATLAB Elementary Functions	67
3.2.2	Using Symbolic Functions	68
3.2.3	Plots for Symbolic Functions	69
3.2.4	Function Evaluation Using <code>eval</code> and <code>feval</code>	71
3.2.5	The Tool <code>funtool</code>	72
3.3	Polynomials	73
3.4	Curve Fitting	79
3.4.1	Cubic Spline Fitting	80
3.4.2	The Tool <code>Basic Fitting</code>	81
3.5	Solution of Equations	82
3.6	Execution Time, Date, and Time of the Day	88
3.7	Concluding Remarks	90
4	Matrices and Linear Algebra	91
4.1	Introduction	91
4.2	Matrices	92
4.3	Basic Operations with Matrices	96
4.3.1	The Dot Operator	98
4.4	The Characteristic Polynomial	98
4.5	Vectors	99
4.5.1	Norm of a Vector	103
4.5.2	Vector Generation	104
4.6	Dot and Cross Products	105
4.6.1	Dot Product	105

4.6.2	Cross Product	106
4.7	Matrix and Vector Functions	106
4.8	Systems of Simultaneous Linear Equations	107
4.8.1	LU Factorization	109
4.9	Eigenvalues and Eigenvectors	110
4.10	Cell Arrays	112
4.11	Structures	113
4.12	Concluding Remarks	114
5	Calculus	115
5.1	Introduction	115
5.2	Limits of Functions	116
5.3	Limits of Sequences	118
5.4	Continuity	119
5.5	Derivatives	121
5.6	Integration	124
5.7	Series	128
5.8	Differential Equations	130
5.8.1	Numerical Solution of Differential Equations	133
5.9	Concluding Remarks	134
6	Programming in MATLAB	135
6.1	Introduction	135
6.2	Creating m-files	136
6.3	Basic Programming Instructions in MATLAB	137
6.3.1	The Instruction <code>if-end</code>	137
6.3.2	The Instruction <code>if-else-end</code>	139
6.3.3	The Instruction <code>elseif</code>	140
6.3.4	The Statement <code>switch-case</code>	141
6.3.5	The Instruction <code>for</code>	142
6.3.6	Nested Loops	143
6.3.7	The <code>while</code> Loop	144
6.4	Functions	145
6.5	Variables of Functions	148
6.5.1	Global Variables	151
6.5.2	The Instruction <code>return</code>	152
6.5.3	The Instructions <code>nargin</code> and <code>nargout</code>	154
6.5.4	Recursive Functions	154
6.6	File Management	155
6.6.1	File Opening and Closing	155
6.7	Writing Information to a File	157
6.7.1	Reading and Writing Formatted Data	158
6.7.2	Reading and Writing Binary Files	162

6.8	Passing Data Between MATLAB and Excel	165
6.8.1	Exporting Data to Excel	165
6.8.2	Exporting Excel Files to MATLAB	168
6.8.3	Reading Data from Excel Files	169
6.9	Publishing m-files from MATLAB	170
6.9.1	Cell Programming	170
6.9.2	Publishing m-files	172
6.10	Concluding Remarks	176
7	Object-Oriented Programming	179
7.1	Introduction	179
7.2	The Object-Oriented Programming Paradigm	180
7.3	Classes in MATLAB	181
7.3.1	Creation and Use of a Class	182
7.3.2	Declaration and Use of Setters	183
7.3.3	Inheritance	185
7.3.4	Constructor	186
7.3.5	Direct and Indirect Access to Properties	187
7.3.6	Public and Private Methods	190
7.3.7	Overriding Methods	193
7.3.8	Overloading	195
7.4	Examples	200
7.5	Conclusions	205
8	Graphical User Interfaces	207
8.1	Creation of a GUI with the Tool GUIDE	207
8.1.1	Starting GUIDE	208
8.1.2	Properties of Objects in a GUI	208
8.1.3	A Simple GUI	210
8.2	Examples	215
8.3	Deployment of MATLAB Graphical User Interfaces	223
8.4	Concluding Remarks	225
9	Simulink	227
9.1	The Simulink Environment	227
9.1.1	A Basic Example	228
9.2	Continuous and Discrete Systems	231
9.3	Subsystems	235
9.3.1	Masking Subsystems	237
9.3.2	Icon & Ports Tab	239
9.3.3	Parameters & Dialog Window	239
9.3.4	Initialization Tab	240

9.3.5	Documentation Tab	241
9.4	Examples	241
9.5	Concluding Remarks	243
9.6	References	244
10	MATLAB Applications to Engineering	245
10.1	Introduction	245
10.2	Applications to Signals and Systems	245
10.3	Applications in Digital Signal Processing	254
10.4	Applications in Control	258
10.5	Applications to Chemical Engineering	263
10.6	Applications in Food Engineering	268
10.7	Applications in Civil Engineering	271
10.8	Applications in Mechanical Engineering	277
10.9	Bibliography	282
11	MATLAB Applications to Physics	283
11.1	Introduction	283
11.2	Examples in Kinematics	283
11.3	Examples in Dynamics	290
11.4	Applications in Astronomy	298
11.5	Applications in Electricity and Magnetism	301
11.6	Applications in Optics	304
11.7	Applications in Modern Physics	307
11.8	Concluding Remarks	310
11.9	Bibliography	310
12	Applications to Finance	311
12.1	Introduction	311
12.2	The Financial and Financial Derivatives Toolboxes	311
12.3	The Financial Derivatives Toolbox	318
12.4	The Black-Scholes Analysis	320
12.4.1	American Options	324
12.5	The Greek Letters	331
12.6	Conclusions	336
12.7	References	336
13	Image Processing in MATLAB	337
13.1	Introduction	337
13.2	Reading and Writing Images	339
13.3	Resolution of the Images	340
13.4	Spatial Filtering	342

13.5 The Discrete Fourier Transform	345
13.6 Color Image Processing	348
13.7 Morphological Image Processing	350
13.8 Concluding Remarks	353
Index	355

Preface

Mathematics is used in almost every field of knowledge. It is a necessary tool in engineering, physics, science, finance, biology, chemistry, and accounting, to name a few. Every day, new disciplines emerge that require massive computational effort. We can mention Artificial Intelligence as an example of them. Mathematics is taught at all levels of education, from kindergarten and elementary school to college and graduate school. Thus, most people have a fairly good knowledge of some area of mathematics. Unfortunately, most students and mathematics users are not taught using mathematics software tools like MATLAB[®]¹, among other similar tools such as Mathematica and MathCAD, which allow users to solve mathematics problems when they arise in their corresponding fields of expertise. The purpose of this handbook is to allow mathematics users to learn and master the mathematics software package MATLAB.

MATLAB integrates computation, visualization, and programming to produce a powerful tool for a number of different tasks in mathematics. MATLAB is the acronym for MATrix LABoratory. In MATLAB, every mathematical variable or quantity is treated in matrix form.

With MATLAB, we can perform complex mathematical tasks with relatively simple programs. This is possible because MATLAB has close to 10,000 built-in functions, from simple ones such as differentiation, integration, and plotting, to optimization functions which require no user programming.

Many of the functions mentioned above are grouped into toolboxes specially dedicated to some field of science, finance, or engineering.

Another important topic covered in the handbook is object-oriented programming, the paradigm that allows us to create graphical user interfaces. The main concepts of this paradigm, as well some examples, are presented.

Simulink[®] is another software package that runs from MATLAB. Simulink simulates systems at the block level. Thus, it is ideal for scientific and engineering system simulation. Simulink is described in [Chapter 9](#) and some examples show the great advantages of using it for system modelling and simulation.

There are many books available on MATLAB, but a unique feature of this handbook is that it can be used by novices and experienced users alike. It is written for the first time MATLAB user who wants to learn the basics of MATLAB, but, at the same time, it can be used by users with a basic MATLAB

¹For contact information write to Mathworks, Inc., 3 Apple Hill Dr., Natick, MA 01760-2098, USA, E-mail: info@mathworks.com, Web: www.mathworks.com.

knowledge who want to learn advanced topics such as programming, creating executables, publishing results directly from MATLAB programs, and creating graphical user interfaces. In addition, for experienced users, it has chapters with MATLAB applications in engineering, physics, finance, image processing, and optimization. Each and every one of the examples and exercises were solved using MATLAB Releases 2017b and 2018b.

The authors wish to thank those in the Book program from The MathWorks, Inc. The authors also wish to thank the staff at Taylor & Francis Group.

David Báez-López

David Alfredo Báez Villegas

List of Figures

1.1	MATLAB main window.	3
1.2	Instructions and variables saved with <code>diary</code>	15
1.3	Selecting instructions and variables in the Command History window using the mouse right button.	16
1.4	Instructions and variables saved in the file <code>variables.m</code>	17
1.5	Help window.	21
1.6	Search results for the function <code>diff</code>	21
2.1	Plot of <code>cos(x)</code>	25
2.2	Using a cursor in a plot.	25
2.3	Two curves in the same plot using <code>hold on</code>	26
2.4	Two traces in the same figure using pairs <code>x</code> , <code>y</code>	27
2.5	Text information added to plot of <code>sin x</code> and <code>cos x</code> functions.	28
2.6	Semilog plot for <code>sin(x)</code> . The <code>x</code> -axis is in a log scale.	28
2.7	Semilog plot for <code>exp(x^2)</code> . The <code>y</code> -axis is in a log scale.	29
2.8	Multiple plot.	30
2.9	Plot with a few points for the <code>x</code> -axis.	31
2.10	Workspace window with variables <code>x</code> , <code>y</code> created.	31
2.11	Selection of the plot type.	32
2.12	Plot catalog window.	32
2.13	Plot for the variable <code>y</code>	33
2.14	Icon for editing the figure properties.	33
2.15	Text on a plot.	35
2.16	Change of axes' limits.	36
2.17	Plot with axis' limits menu displayed.	37
2.18	Polar plot of a spiral.	38
2.19	Bar plot.	39
2.20	Stairs plot.	39
2.21	Histogram plot.	40
2.22	Stem plot.	41
2.23	Compass plot.	42
2.24	Pie plot.	42
2.25	Multiple plots using the instruction <code>subplot</code>	44
2.26	Helix plot using <code>plot3</code>	45
2.27	Three two-dimensional curves in a three-dimensional plot.	46

2.28	Mesh plot.	47
2.29	Transparent mesh plot using hidden off	47
2.30	Black mesh plot using options.	48
2.31	Mesh plot with a contour using meshc	48
2.32	Mesh plot with a zero plane using meshz	49
2.33	Plot of sphere with surf	50
2.34	Plot of sphere without lines.	50
2.35	Plot of sphere with smoothed colors.	51
2.36	Surface plot with lighting.	52
2.37	Contour plot.	52
2.38	Three-dimensional contour plot.	53
2.39	Three-dimensional contour plot with pcolor	53
2.40	Waterfall plot.	54
2.41	A quiver plot.	55
2.42	Contour plot with values.	55
2.43	Viewpoint for a three-dimensional plot.	56
2.44	Mesh plot with a different viewpoint.	57
2.45	Rotate button.	57
2.46	Plot in Figure 5.	59
2.47	Hierarchy of MATLAB objects.	60
3.1	MATLAB Editor window.	66
3.2	Ball's path.	67
3.3	Plot of $f(x) = x^2 \sin(x)$	70
3.4	Plot of $f(x) = x^2 \sin(x)$ in the range -20 to 20.	70
3.5	Plot of $f(x) = \exp(-x/2) \sin(x/2)$ in the interval $[-5, 6]$	71
3.6	Windows for the function funtool	72
3.7	Windows for the function funtool with a = 2 and $[-1, 1]$	73
3.8	Plot of polynomial $6x^3 + 3x^2 - 7x + 0.4$	75
3.9	Plot of data points and third and fifth order polynomials.	81
3.10	Plot of data points and the spline.	82
3.11	Choice of Basic Fitting from the Tools menu.	83
3.12	Window to select the interpolating polynomials.	83
3.13	The three windows for the Basic Fitting Tool	84
3.14	Curves of the 5th degree polynomial and spline in the top plot. Residuals plot in the bottom plot.	84
3.15	Plot of the function $x^3 - \sin(x) - 3$	87
4.1	MATLAB main window.	94
4.2	MATLAB array editor window.	94
4.3	MATLAB and array editor with new elements.	95
4.4	Cell array.	112
5.1	Integration paths.	126
5.2	Solution of the differential equation $dy/dt = -2yt$	134

6.1	Menu to create an m-file.	137
6.2	WordPad window showing the contents of binary.dat	164
6.3	Data in the file countries.csv	166
6.4	Part 1 of the Import Wizard . Here we indicate that the data is separated by commas or tabs.	167
6.5	Part 2 of the Import Wizard . Here we indicate that the data is separated by commas.	167
6.6	Part 3 of the Import Wizard . Here we select each column and set the data format.	167
6.7	Data from countries.csv in Excel.	168
6.8	Data in Excel for Numbers.csv	169
6.9	Data in Excel for years.xlsx	169
6.10	Import wizard for Excel files.	170
6.11	Import wizard for variables.	170
6.12	m-file divided in cells.	172
6.13	Publishing preferences.	172
6.14	Top part of the deployed pdf document.	175
6.15	Top part of the published file in HTML format.	176
7.1	Creating a class in MATLAB.	182
7.2	Help for viewing information about the superclass handle . .	183
7.3	Characteristics of the object a and the variable frac	185
8.1	GUIDE Quick Start window.	208
8.2	A blank GUI.	209
8.3	Property inspector.	210
8.4	GUI with required objects.	211
8.5	GUI with objects stretched to its final size and final labels. .	212
8.6	Plot of function $e^{(x/10)}\sin(x)$ from 0 to 9π	214
8.7	Initial layout for the GUI.	215
8.8	GUI layout with strings and sizes changed.	216
8.9	String for the pop-up menu. Click on the icon to the right of String	216
8.10	A run for the temperature conversion GUI.	219
8.11	GUI layout.	221
8.12	Final GUI layout.	221
8.13	Final GUI with data.	223
8.14	Deployment tool.	224
8.15	Loading the files to be deployed.	224
8.16	Creating the executable file to be deployed.	225
9.1	Simulink Icon.	228
9.2	Simulink menu window.	228
9.3	Simulink model window.	229
9.4	Simulink libraries window.	229

9.5	Creation of the model.	230
9.6	Properties for the Transfer Function block.	231
9.7	Input and output signals in Scope.	232
9.8	Continuous model.	233
9.9	Output signal.	234
9.10	Discrete-time model.	235
9.11	Mixed-mode model.	235
9.12	Subsystem from an existing model, a) Original model, b) Selection of the blocks and lines for the subsystem, c) Path to create the subsystem, d) Main model window.	236
9.13	Subsystem from a subsystem block.	237
9.14	Mask Editor pane.	238
9.15	Editing the Gain blocks to A0 and A1	238
9.16	Parameters & Dialog tab.	240
9.17	Prompt for parameters A0 and A1	240
9.18	Description and Help.	241
9.19	Digital counter with unused states.	242
9.20	Model for retirement savings.	243
10.1	Bode plots.	246
10.2	Bode plots for a third order function.	247
10.3	Effect of changing ξ in a second order transfer function.	248
10.4	Unit-impulse and unit-step responses.	249
10.5	Step response for different values of ξ	249
10.6	Mesh plot of step response for different values of ξ	250
10.7	Sequence x_3 resulting from the convolution of x_1 and x_2	255
10.8	Block diagram of a plant with feedback.	259
10.9	Root locus of a feedback system.	260
10.10	Model for the Hubble telescope.	261
10.11	Response for the Hubble telescope.	262
10.12	Flux versus time.	265
10.13	Molar fraction.	265
10.14	Concentration profiles for the components.	267
10.15	3D plot for specific heat versus temperature and time.	269
10.16	Drying curves for different number of terms in the series.	271
10.17	Deflected beam.	272
10.18	Beam deflection.	274
10.19	Support.	274
10.20	Reactions at the support.	275
10.21	Components of the reactions.	275
10.22	Mass-spring-damper system.	277
10.23	Effect of harmonic movement at the base.	278
10.24	Displacement and speed plots.	278
10.25	Structure excited by an impulse.	280

10.26	Plots of displacement and velocity vs. time and velocity vs. time.	282
11.1	Distance traveled by both vehicles.	286
11.2	Plot of speed vs. time.	286
11.3	Plot of the y-coordinate for the position vs. time.	290
11.4	Plot of the y-coordinate for the position vs. time when air resistance is taken into account.	292
11.5	Plot of angular position vs. time.	294
11.6	Plot of angular position and speed vs. time for the exact equations.	295
11.8	Orbit of a planet around the Sun.	300
11.9	Mercury's orbit as seen from the Earth.	301
11.10	Electric field lines.	303
11.11	Magnetic field lines.	305
11.12	Diffraction patterns for a two slit screen when the slit distance is $d = 50\lambda$	306
11.13	Diffraction patterns for a two slit screen when the slit distance is $d = 6\lambda$	307
11.14	Interference pattern.	308
11.15	Comparison between Newtonian and relativistic energies. . .	310
12.1	Plot of Real price vs. yield	318
12.2	Window for derivtool with some preloaded data.	319
12.3	Option 101 selected in derivtool	319
12.4	Window where sensitivities can be chosen.	320
12.5	Values for Vega sensitivity are shown.	321
12.6	Hedging in derivtool	321
12.7	Cumulative probability distribution function.	322
12.8	Variation of the put option with the interest rate and time. .	324
12.9	Binomial stock price movement.	324
12.10	Complete binomial tree.	325
12.11	Binomial tree.	330
12.12	Binomial tree.	331
12.13	Plot of Rho for a put option vs. interest rate and stock price. .	334
12.14	Plot of delta for a put option vs. strike price and stock price. .	334
12.15	Plot of Theta for a call option vs. interest rate and stock price.	335
13.1	Pixels of an image.	338
13.2	Ordering of pixels.	338
13.3	Lena image. This image is 512×512.	339
13.4	Part of the image corresponding to imshow(lena_gray (210: 400, 210: 500))	340
13.5	Gray levels for row 131 for the image lena_gray	341

13.6	X-ray image with a limited dynamic range.	341
13.7	Histogram of the image of Figure 13.6.	342
13.8	Equalized histogram.	343
13.9	Equalized image.	343
13.10	Filtered image of a disc with radius 9.	346
13.11	A square to obtain the Discrete Fourier Transform.	346
13.12	Obtaining the Fourier Transform in two dimensions.	347
13.13	Lena image with its RGB components.	349
13.14	Lena in the HSV format and its components.	349
13.15	Structuring element and image to be processed.	350
13.16	Result from an (a) erosion, (b) dilation. The original image is the black square. The structuring element is the small square.	350
13.17	Test image for the four morphological operations.	351
13.18	Results of the morphological operations: erosion, dilation, opening and closing.	351

List of Tables

1.1	Basic MATLAB operations and precedence	5
1.2	Elementary functions	6
1.3	Predefined constants in MATLAB	7
1.4	Formats for displaying numerical values	8
1.5	Data types	10
1.6	Functions for strings	12
2.1	Color codes for traces	34
2.2	Trace markers	34
2.3	Trace style	34
3.1	Experimental data	79
4.1	Matrix commands	93
4.2	Special matrices	95
4.3	Matrix operations	99
4.4	Vector operations	101
6.1	Relational operators	138
6.2	Logic operators	138
6.3	Permit codes to open files	155
6.4	File handles	156
6.5	Options for variable precision	162
6.6	Commands for MATLAB LaTeX	174
8.1	Important icons in the GUIDE toolbar.	209
8.2	Most used properties for objects in a GUI.	211
8.3	Tag names.	217
9.1	Options for the Icon and Ports tab.	239
10.1	Data for specific heat.	268
13.1	IMFilter options.	344
13.2	Types of filters. The format is <code>fspecial(Parameters)</code> . . .	345



Taylor & Francis

Taylor & Francis Group

<http://taylorandfrancis.com>

Chapter 1

Introduction to MATLAB

1.1	Introduction	1
1.1.1	Book Organization	2
1.1.2	Chapter Organization	2
1.2	Starting MATLAB	3
1.3	Simple Calculations in MATLAB	4
1.3.1	Elementary Functions	5
1.4	Variables	7
1.4.1	Variable Types	9
1.5	Strings	10
1.6	Saving a Session and Its Variables	14
1.7	Input/Output Instructions	16
1.7.1	Formatted Output	16
1.7.2	Data Input	18
1.8	Help	19
1.8.1	Help Page	20
1.9	Concluding Remarks	22
1.10	Bibliography	22

1.1 Introduction

MATLAB[®] is a very high-level powerful system designed for technical computing. It integrates in the same software environment computation, programming, and visualization. It also has a very easy mathematical notation. Some of the most common applications for technical computing are as varied as:

- Algorithmic development
- Modeling and simulation
- Data analysis
- Plotting
- Graphical User Interfaces
- Rapid prototyping
- Toolboxes
- Vectorization

MATLAB is an acronym for **MAT**rix **LAB**oratory and it was originally developed to perform matrix calculations. MATLAB includes a programming language which is, probably, more powerful than traditional programming languages such as C, C++, C#, VisualBasic, and Python, to name a few.

MATLAB was developed in 1984 by Cleve Moler and Jack Little, who founded The MathWorks, Inc. in Natick, Massachusetts. The first version of MATLAB only had about eighty functions. The very latest version includes more than ten thousand functions.

Besides MATLAB, The MathWorks has developed a series of software packages, called toolboxes, written in the MATLAB programming language. These toolboxes can perform a number of calculations in several branches of engineering, economics, finance, physics, and mathematics, among others. It is hard to imagine an area of knowledge where MATLAB does not have an application. In the coming chapters we will use examples of the different areas where MATLAB has applications to illustrate how to use MATLAB.

1.1.1 Book Organization

The book is organized in the following way. The first two chapters are an introduction to MATLAB. The following three [chapters, 3 to 5](#), cover basic calculations in MATLAB. The topics include linear algebra, calculus, and plotting. The next three [chapters, 6 to 9](#), cover programming, advanced programming techniques, graphical user interface (GUI) development, and Simulink® which is a MATLAB-based GUI useful for system modeling and simulation. The last five chapters cover a broad set of examples illustrating applications in several engineering disciplines, physics, and finance.

1.1.2 Chapter Organization

The chapter is organized as follows. It begins with a MATLAB environment description. It describes the basic layout desktop and explains the purpose of the different windows available. It continues with basic calculations describing how to create and store variables. Different formats for variables are introduced. Basic functions are presented and the set of elementary function is described. Strings are introduced and some operations on strings are described. A very useful characteristic of MATLAB is the fact that all the variables, functions, and calculations are stored in the **Command History** and thus, they can be stored in a file, or a program can be created from them. This is covered in detail in this chapter. Finally, it is explained how to use MATLAB Help.

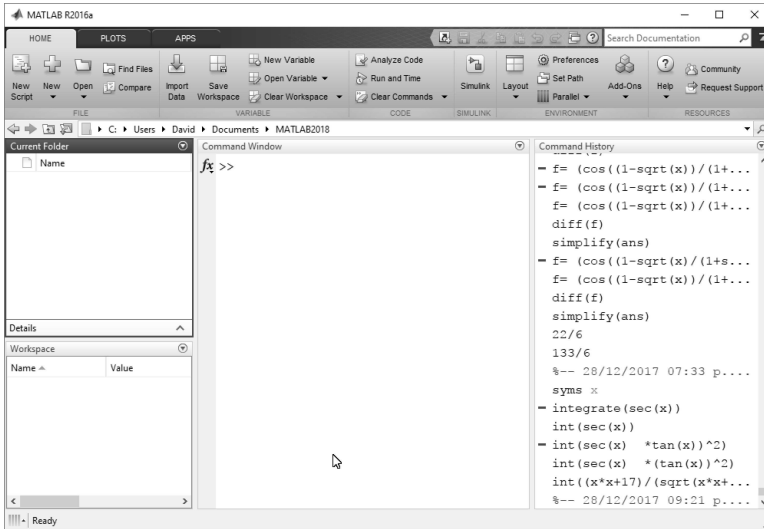


FIGURE 1.1: MATLAB main window.

1.2 Starting MATLAB

To install MATLAB, follow the instructions provided by The MathWorks, Inc. to install the software and the licenses. When the installation is finished, a MATLAB icon will be displayed on the Desktop and in the Programs path. When we click on the icon, MATLAB will open, as shown in Figure 1.1. In the main menu in the Desktop we can display up to four windows:

1. Command Window,
2. Workspace,
3. Current Directory, and
4. Command History.

Each of these windows can be attached to the main window, but we can have any of them as a separate window by using the right upper arrow. We can reattach the window using the same arrow.

The **Command window** is the window where we enter variable values and instructions to perform calculations. The **Workspace window** contains the variables created in the current session. The **Current Directory** displays the files and folders in the current directory. The **Command History** displays the instructions executed in the current and previous sessions, unless it has been deleted. Each of these windows has a pulldown menu that can be displayed with the triangle at the right-hand upper corner for each window.

In the main window we also have a set of tabs which are HOME, PLOTS, and APPS. The HOME tab is shown by default and it contains a toolbar with standard icons like **New Script**, **New**, **Open**, and the like. There are some icons to **Import Data** and **Save Workspace**. Other icons are related to coding like **Analyze Code** and **Run and Time**. There is an icon for **Simulink**, the systems simulator which is covered in [Chapter 9](#). Several of the icons in the HOME toolbar are discussed and used in the book. The PLOTS tab shows the different plots that can be made in MATLAB. We only need to select the variables we wish to plot. In the APPS tab we have the apps available in the toolboxes. We can also add the apps that we develop.

1.3 Simple Calculations in MATLAB

MATLAB can perform simple calculations as if it were a simple calculator. The prompt symbol `>>` indicates that MATLAB is ready. For example, if we wish to add two numbers such as $2 + 3$, we simply write in the **Command Window** after the MATLAB prompt the desired operation as:

```
>> 2 + 3
```

and press ENTER, the output that MATLAB gives is

```
ans =  
5
```

We use the convention that the data we enter in MATLAB and the results from MATLAB are written in typewriter font.

We now show in [Table 1.1](#) the basic operations and the precedence they have in MATLAB. The precedence means the priority order to perform the different operations. So, exponentiation has the highest priority, multiplication and division have the following higher priority, and finally, addition and subtraction have the lowest priority. Thus, for example, in the operation

```
>> 2*3 + 4*7^4
```

The exponentiation 7^4 is performed first, then the multiplication $2*3$ and 4 times the result of 7^4 , and finally, the addition of the results of $2*3$ and $4*7^4$.

Precedence can be user-modified by using parentheses. Operations enclosed by parentheses are performed first. For example,

```
>> 2^3*4
```

TABLE 1.1: Basic MATLAB operations and precedence

Operation	Symbol	Example	Precedence
Addition	+	$15 + 8 = 23$	3
Subtraction	-	$16 - 11 = 5$	3
Multiplication	*	$3 * 7.2 = 21.6$	2
Division	/	$18 / 3 = 6$	2
Exponentiation	\wedge	$3 \wedge 4 = 81$	1

gives the result

```
ans =
    32
```

Since the first operation is $2 \wedge 3$, which gives 8, then this result is multiplied by 4. Instead, we can use parentheses as in

```
>> 2^(3*4)
```

and we obtain

```
ans =
   4096
```

because MATLAB performs first $(3 * 4)$ which gives 12 and then $2 \wedge 12$ which gives 4096.

1.3.1 Elementary Functions

MATLAB has a set of functions for everyday use. These functions are available in a set called **Elementary Functions**. The trigonometric functions fall within this set. For example, to calculate `sin(1)`

```
>> sin(1)

ans =
    0.8415
```

For the case of the trigonometric functions, the argument is in radians. Some of the elementary functions are available in [Table 1.2](#). For a listing of all of the elementary functions, we enter `help elfcn` in the **Command Window**.

TABLE 1.2: Elementary functions

Function	MATLAB Notation
$\sin x$	<code>sin (x)</code>
$\cos x$	<code>cos (x)</code>
$\tan x$	<code>tan (x)</code>
\sqrt{x}	<code>sqrt (x)</code>
$\log_{10}(x)$	<code>log10 (x)</code>
$\ln (x)$	<code>log (x)</code>
$ x $	<code>abs (x)</code>
e^x	<code>exp (x)</code>

The word **elfcn** is an acronym for **e**lementary **f**unctions. As mentioned above, MATLAB has more than ten thousand functions. Some of the most elementary ones are written in C and most of the functions are written in the MATLAB language.

For the case of trigonometric functions, a simple example is to find the sine of the irrational number π . If we approximate π by 3.1416, then

```
>> sin (3.1416)
```

```
ans =
-7.3464e-006
```

which is a good approximation to the exact result. MATLAB has a predefined value for the constant π and which is simply called **pi**. Then, we have

```
>> sin(pi)
```

```
ans =
1.2246e-016
```

which is closer to the expected result. Other examples are

```
>> sqrt (2)
```

```
ans =
1.4142
```

```
>> log10 (1000)
```

```
ans = 3.0000
```

Some of the predefined constants are shown in [Table 1.3](#).

TABLE 1.3: Predefined constants in MATLAB

MATLAB	
<code>pi</code>	3.14159265
<code>i</code>	Imaginary unit = $\sqrt{-1}$.
<code>j</code>	Same as <code>i</code> .
<code>eps</code>	Precision of the floating point operations, $2.22 \cdot 10^{-16}$
<code>Inf</code>	Infinity
<code>NaN</code>	Not a number.

1.4 Variables

Variables are created in MATLAB as they are defined. That is, there is no need to predefine them together with their type as it has to be done in languages such as C, Fortran, and Visual Basic, to name a few. For example, the variable `alpha` is created the first time we use it as:

```
>> alpha = 32
```

```
alpha =
    32
```

Now, this variable appears in the **Workspace** window and will remain there until we delete it with the instruction `clear` which erases all variables stored in the **Workspace**. To see a list of variables in the **Workspace** we only need to type `who`. For example, if the MATLAB session just started, the only variable is `alpha`. Then, we obtain

```
>> who
```

```
Your variables are:
alpha
```

Variable names can have up to 63 characters. If a variable name is longer than 63 characters, the name will be truncated to 63 characters.

The number of digits used by MATLAB can be changed according to the format chosen for this purpose. The formats available are shown in [Table 1.4](#). We show the formats with the irrational number $\sqrt{2}$.

Each time MATLAB performs an action, the result is displayed in the **Command Window**. Sometimes we are only interested in the final results and not in the intermediate ones. We can suppress intermediate results by typing a semicolon after the instruction at the end of the line. For example:

TABLE 1.4: Formats for displaying numerical values

MATLAB format	Displayed value	Comments
format short	1.4142	5 digits
format	1.4142	Same as format short
format long	1.414213562373095	16 digits
format short e	1.4142e+00	5 digits plus exponent
format long e	1.414213562373095e+00	16 digits plus exponent
format hex	3ff6a09e667f3bcd	Hexadecimal
format bank	1.14	2 decimals (for currency)
format +	+	Positive or negative
format rat	1393/985	Approximates to a ratio

```
>> 23 + 32

ans =
    55

>> 23 + 32;
```

The first time we calculate `23 + 32` we get an answer, the second time we have a semicolon after the indicated addition and we **do not** get an answer at all; however, the result is stored in the variable `ans`.

MATLAB variable names are case sensitive, so care must be taken when writing long programs as we see in [Chapter 6](#). Thus, variable `A1` is different from variable `a1`.

For a set of numbers written between brackets as in

```
v = [ pi 2 -4 8 sin(2)]
```

The variable `v` is called a vector, in this case a row vector. In a column vector, the elements are separated by semicolons, thus it is written as

```
w = [2 ; 3; -7; sqrt(3)]
```

In the column vector `w` the elements are separated by a semicolon. This indicates that the elements after the semicolon are located in a different row. Thus, MATLAB gives as an answer to these two vectors:

```
>> v = [ pi 2 -4 8 sin(2)]

v =

    3.1416    2.0000   -4.0000    8.0000    0.9093
```

```
>> w = [2 ; 3; -7; sqrt(3)]

w =
  2.0000
  3.0000
 -7.0000
  1.7321
```

In the case of row vector \mathbf{v} , the elements can be separated by either blanks or by commas, while in a column vector elements are separated by semicolons. A vector has dimension n if it has n elements. Thus, vector \mathbf{v} has dimension 5 whereas vector \mathbf{w} has dimension 4.

If we have two row vectors with the same dimension, we define the dot product by

$$\mathbf{x}*\mathbf{y}'$$

but if we have two column vectors with the same dimension, we define the dot product by

$$\mathbf{x}'*\mathbf{y}$$

The result is a scalar. Thus, for vectors \mathbf{v} , \mathbf{w} given by

```
>> v = [-1 4 7 -9]

v =
 -1 4 7 -9

>> w = [9 8 -6 3]

w =
 9 8 -6 3

>> v*w'

ans =
 -46
```

1.4.1 Variable Types

Variables have a type in MATLAB. The available types are shown in [Table 1.5](#) and we have also the size in bytes that each type uses in the memory.

TABLE 1.5: Data types

Data type	Name	Size
<code>double</code>	double precision	8 bytes
<code>single</code>	single precision	4 bytes
<code>int8</code>	8-bit signed integer	1 byte
<code>int16</code>	16-bit signed integer	2 bytes
<code>int32</code>	32-bit signed integer	4 bytes
<code>int64</code>	64-bit signed integer	8 bytes
<code>uint8</code>	8-bit unsigned integer	1 byte
<code>uint16</code>	16-bit unsigned integer	2 bytes
<code>uint32</code>	32-bit unsigned integer	4 bytes
<code>uint64</code>	64-bit unsigned integer	8 bytes

1.5 Strings

A string of characters is a series of letters, and/or numbers, and/or any other symbols. A string is defined by enclosing the set of characters between single quotes. As examples of strings we have ‘book’, ‘technical computing’, ‘2332’. A variable in MATLAB can be a string. For example,

```
>> a = 'taylor'

a =
taylor

>> b = '1 a 2 b 3 c!"#'
```

```
b =
1 a 2 b 3 c!"#
```

If for any reason a string has to have a quote as a part of the string, we simply repeat the quote as in

```
>> a = 'Laura''s bear'
```

```
a =
Laura's bear
```

In this example, the blank space is also a character in the string.

The elements in a string have a position. Thus `a(k)` is the k th element in the string. The number k is called the index. For the string `a` given above, `a(1) = L` and `a(10) = e`,

```
>> a(1)
```

```
ans =  
L
```

```
>> a(10)
```

```
ans =  
e
```

To specify a range of characters we use `a(k:n)` which indicates the characters from the k th to the n th. For example,

```
>> a(3:9)
```

```
ans =  
  
ura's b
```

We can also use `a(3:2:9)` which indicates the set of characters in a string from index 3 to index 9 but incrementing the index by 2, that is, we are looking for `a(3)`, `a(5)`, `a(7)` and `a(9)`,

```
>> a(3:2:9)
```

```
ans =  
  
uasb
```

We can concatenate several strings by using

```
New_string = [string 1, string 2, string 3]
```

For example, if we have three strings `a1`, `a2`, and `a3`, defined by (there is a blank space after each country name)

```
>> a1 = 'Canada '; a2 = 'Mexico '; a3 = 'USA ';
```

We can form

```
>> b = [a1, a2]
```


TABLE 1.6: Functions for strings

MATLAB	Function
<code>length</code>	Number of characters in a string.
<code>strcmp</code>	Compares two strings.
<code>str2num</code>	Converts a string to a numerical value.
<code>num2str</code>	Converts a number to a string.
<code>strrep</code>	Replaces characters in a string with different characters.
<code>upper</code>	Changes lowercase characters to uppercase.
<code>lower</code>	Changes uppercase characters to lowercase.

```

b =

    Canada Mexico

>> b2 = [a1, a3]

b2 =

    Canada USA

```

We can also form new strings with elements from the strings already defined as

```

>> y1 = [a1(1:3), a2(1:3), a3(1:2)]

y1 =

    CanMexUS

```

There exists a set of functions that can be performed on strings. [Table 1.6](#) shows a list of some of the most used ones.

For the function `length` and the string `a = 'This is Chapter 1'`, we have

```

>> length(a)

ans =

    17

```

The instruction `strcmp(a, b)` compares two strings `a` and `b`. If they are equal the result is unity, if they are not equal the result is zero. For example, for the strings `a1` and `a2` given above,

```

>> strcmp(a1, a2)

```

```
ans =  
    logical  
    0
```

```
>> strcmp(a2, 'Mexico ')
```

```
ans =  
    logical  
    1
```

The instruction `str2num` converts a string to a number only if the string characters are numbers, as

```
>> d = str2num('2010')
```

```
d =  
2010
```

The instruction `num2str` transforms a number into a string. For the number `a = 810`

```
>> num2str(a)
```

```
ans =  
    '810'
```

which is a string. The instruction `strrep` (`c1` , `c2` , `c3`) replaces the string `c2` by string `c3` in the string `c1`. For example, in the string 'MATLAB is a high level programming language' we can replace the string 'level' by 'level technical' with

```
>> c1 = 'MATLAB is a high level programming language' ;  
>> c2 = 'level';  
>> c3 = 'level technical';  
>> strrep( c1, c2, c3)
```

```
ans =  
    'MATLAB is a high level technical programming language'
```

The instructions `upper` and `lower` convert lowercase characters to uppercase and vice versa, respectively. For the strings `a = 'American Continent'` and `b = 'European Continent'` we have:

```
h = upper(a)
k = upper(b)
lower(h)
lower(k)

h = 'AMERICAN CONTINENT'
k = 'EUROPEAN CONTINENT'
ans =
'american continent'
ans =
'european continent'
```

1.6 Saving a Session and Its Variables

A variable and its value are kept only during the session and only until the user uses the instruction `clear` which deletes all the variables in the session. However, in several cases it is desired to have them available in another session. In other cases, we wish to save not only the variables, but also the instructions executed during the session. That is, we might need to keep the complete session. This can be done using the instruction `diary` followed by the file name. The instructions and variables will be stored in that file until the instruction `diary off` is executed, as in the following,

```
diary file_name
:
diary off
```

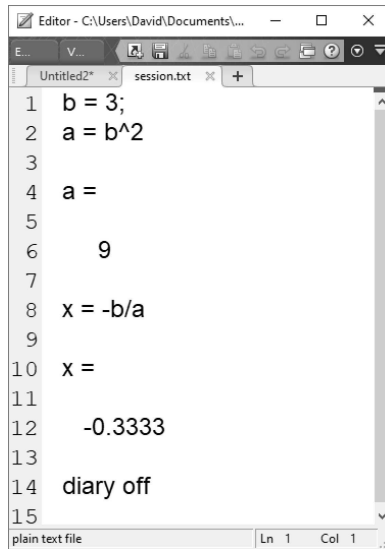
where the dots indicate the instructions and variables. Let us consider the following session

```
>> a = 1; b = 2;
>> diary session.txt
>> b = 3;
>> a = b^2

a =
    9.0000

>> x = -b/a

x =
   -0.3333
```



```
1 b = 3;  
2 a = b^2  
3  
4 a =  
5  
6 9  
7  
8 x = -b/a  
9  
10 x =  
11  
12 -0.3333  
13  
14 diary off  
15
```

FIGURE 1.2: Instructions and variables saved with `diary`.

```
>> diary off
```

```
>> d = 3*x
```

```
d =  
-1.0000
```

Now we take a look at the **Current Folder** window and we observe that there is a new file called `session.txt`. If we open it with **File** → **Open** we see that all the variables and instructions together with the responses executed after `diary session.txt` and before `diary off` are stored in the file. This is shown in Figure 1.2. If a file name is not specified, the default file name is `diary.m`.

Another way to save information is by using the icon **Save Workspace** which is available from the tools bar. In this case we **only** save variable information. We cannot save instructions with this option. In this case the file name has the extension `.mat`. For this session we save the data in the file `sessionOne.mat`.

Alternatively, we can also execute

```
>> save('sessionOne')
```

To load the variables in a later session we simply execute

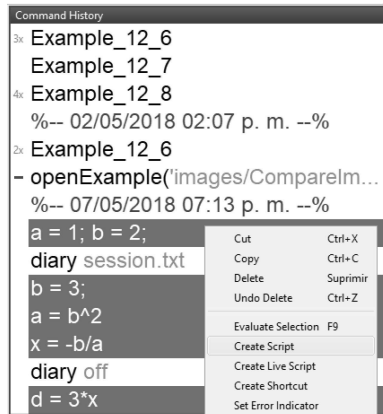


FIGURE 1.3: Selecting instructions and variables in the Command History window using the mouse right button.

```
>> load ('sessionOne')
```

We can also use icon **Open** and select the file variables.

From the **Command History** window we can also save information. By selecting with the left button mouse the instructions and variables and then with a right click we select the option **Create Script** as shown in Figure 1.3. The file name is saved with the extension **.m**. The final file is shown in Figure 1.4.

1.7 Input/Output Instructions

Up to now we have used MATLAB as a calculator, where the result is written right after we hit the ENTER key. Sometimes this is not very convenient, especially if we are not interested in intermediate results but rather at the final one. Fortunately, we can omit intermediate outputs by typing a semicolon after the instruction, as we saw in Section 1.4. Now we see how to read and write data from a to a file, respectively.

1.7.1 Formatted Output

Besides the direct output obtained after entering an instruction and hitting the ENTER key, we can use the instruction **fprintf**. We illustrate its use with an example. To write the string **'This is chapter 1.'** we use:

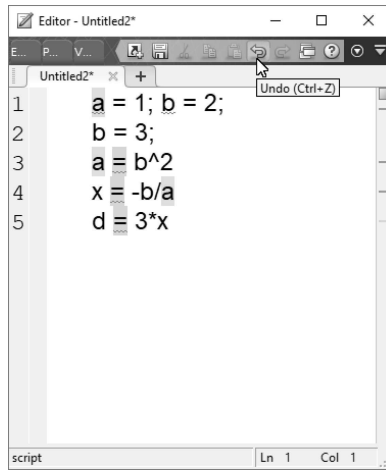


FIGURE 1.4: Instructions and variables saved in the file `variables.m`.

```
>> fprintf('This is chapter 1.\n');
```

This is chapter 1.

We note several things. First we added `\n` which indicates that there is a line feed after writing the desired string. If we now use again the `fprintf` instruction, the string included there will be written in the following line. Second, the semicolon does not preclude writing the desired output. And third, there is not a line with `ans` as before. If we continue writing data with `fprintf` but we do not use `\n`, the output is going to be written in the same line. This is so because the instruction `fprintf` does not make a line feed by itself, but we have to indicate it. For example,

```
>> fprintf('This ');fprintf('is ');fprintf('Chapter 1.');
```

This is Chapter 1.

```
>> fprintf('This \n');fprintf('is \n');fprintf('Chapter 1.\n');
```

This
is
Chapter 1.

The first time we executed the `fprintf` instruction, the strings were written in the same row because we omitted the line feed part. In the second `fprintf` instruction we placed an `\n` before ending each string. Thus, after each `fprintf` instruction there is a line feed.

We can format the output variables. We have different formats to do this. The formats are:

- f - floating point
- g - fixed or floating point
- i - integer
- c - character
- s - string
- d - double precision
- e - exponential notation

We can also use `fprintf` with numerical quantities. For example, for the number `a = pi`, we can use

```
>> fprintf ( 'The number is %12.8f\n', pi )
```

```
The number is 3.14159265
```

which means that the number `pi` is written as a fixed point number with 8 decimal places.

1.7.2 Data Input

The easiest way to input data to MATLAB is through the keyboard, in the same way we have done it so far. Another way is to use the instruction `input` which has the format

```
x = input ( 'Enter the value of x' )
```

This instruction displays the string between quotes and waits for the value of `x`. So we can have the following:

```
>> x = input ( 'Enter the value of x ' ) ;
```

```
Enter the value of x 56
```

```
>> fprintf ( '\n You entered the number %g.\n ', x)
```

```
You entered the number 56.
```

The instruction `input` does not generate a line feed. As mentioned before, the line feed can only be generated with `\n`, which can go anywhere within the string.

If we enter an alphanumeric value instead of a numerical one, we get an