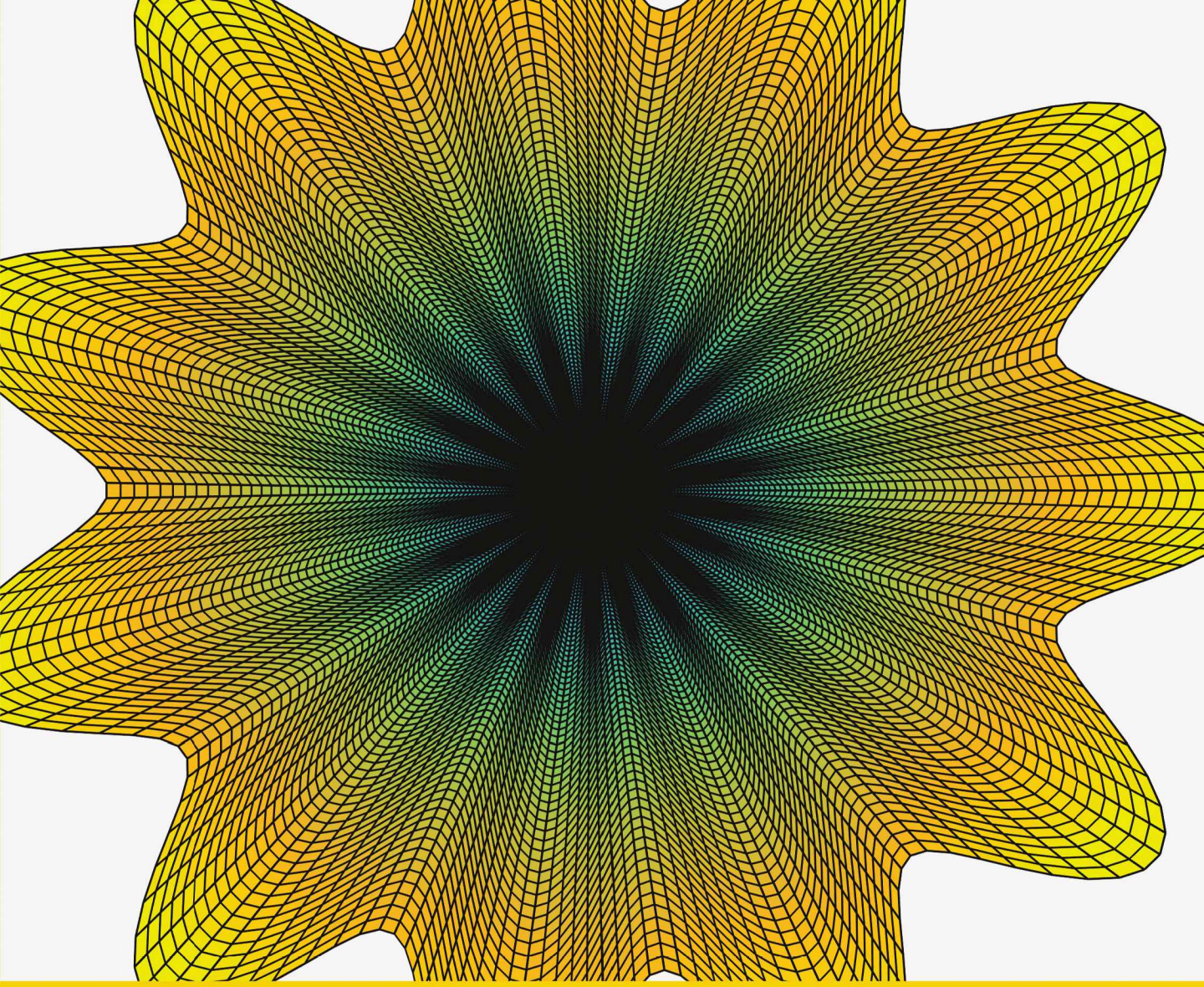

MATLAB®

A PRACTICAL INTRODUCTION TO PROGRAMMING AND PROBLEM SOLVING

FOURTH EDITION

MATLAB® examples

STORMY ATTAWAY

B H

MATLAB®

A Practical Introduction to Programming and Problem Solving

This page intentionally left blank

MATLAB®

A Practical Introduction to Programming and Problem Solving

Fourth Edition

Stormy Attaway

Department of Mechanical Engineering,
Boston University

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

ELSEVIER

Butterworth-Heinemann is an imprint of Elsevier

Butterworth-Heinemann is an imprint of Elsevier
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, UK
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, USA

First published 2009
Second edition 2012
Third edition 2013
Fourth edition 2017

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

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

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

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

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

ISBN: 978-0-12-804525-1

Publisher: Todd Green
Acquisition Editor: Stephen Merken
Developmental Editor: Nate McFadden
Production Project Manager: Sujatha Thirugnana Sambandam
Cover Designer: Greg Harris

Typeset by SPi Global, India

Dedication

This book is dedicated to all of my family: my mother Jane Conklin, my sister Catherine Attaway, my brother Banks Attaway, my stepmother Robyn Attaway, and my husband Ted de Winter.

This page intentionally left blank

Contents

Preface

MOTIVATION

The purpose of this book is to teach basic programming concepts and skills needed for basic problem solving, all using MATLAB® as the vehicle. MATLAB is a powerful software package that has built-in functions to accomplish a diverse range of tasks, from mathematical operations to three-dimensional imaging. Additionally, MATLAB has a complete set of programming constructs that allow users to customize programs to their own specifications.

There are many books that introduce MATLAB. There are two basic flavors of these books: those that demonstrate the use of the built-in functions in MATLAB, with a chapter or two on some programming concepts, and those that cover only the programming constructs without mentioning many of the built-in functions that make MATLAB efficient to use. Someone who learns just the built-in functions will be well-prepared to use MATLAB, but would not understand basic programming concepts. That person would not be able to then learn a language such as C++ or Java without taking another introductory course, or reading another book, on the programming concepts. Conversely, anyone who learns only programming concepts first (using any language) would tend to write highly inefficient code using control statements to solve problems, not realizing that in many cases these are not necessary in MATLAB.

Instead, this book takes a hybrid approach, introducing both the programming and the efficient uses. The challenge for students is that it is nearly impossible to predict whether they will in fact need to know programming concepts later on or whether a software package such as MATLAB will suffice for their careers. Therefore, the best approach for beginning students is to give them both: the programming concepts, and the efficient built-in functions. Since MATLAB is very easy to use, it is a perfect platform for this approach in teaching programming and problem solving.

As programming concepts are critically important to this book, emphasis is not placed on the time-saving features that evolve with every new MATLAB release.

For example, in most versions of MATLAB, statistics on variables are available readily in the Workspace Window. This is not shown in any detail in the book, as the availability of this feature depends on the version of the software, and because of the desire to explain the concepts in the book.

MODIFICATIONS IN FOURTH EDITION

The changes in the Fourth Edition of this book include the following:

- Use of MATLAB Version R2016a
- A new chapter on Object Oriented Programming, and its use (as of R2014b) in the graphics objects
- The new App Designer, which may eventually replace GUIs and uses object-oriented programming
- The new Live Editor, which creates Live Scripts including text, equations, plots, and code
- Modified and new end-of-chapter exercises
- More conceptual end-of-chapter exercises
- Reorganization of chapters and sections within chapters to make it easier to focus on programming concepts in order to cover more of the book in traditional courses
- More coverage of data structures including categorical arrays and tables
- Increased coverage of built-in functions in MATLAB

KEY FEATURES

Side-by-side Programming Concepts and Built-in Functions

The most important and unique feature of this book is that it teaches programming concepts and the use of the built-in functions in MATLAB, side-by-side. It starts with basic programming concepts such as variables, assignments, input/output, selection, and loop statements. Then, throughout the rest of the book many times a problem will be introduced and then solved using the "programming concept" and also using the "efficient method". This will not be done in every case to the point that it becomes tedious, but just enough to get the ideas across.

Systematic Approach

Another key feature is that the book takes a very systematic, step-by-step approach, building on concepts throughout the book. It is very tempting in a MATLAB text to show built-in functions or features early on with a note that says "we'll do this later". This book does not do that; functions are covered before they are used in examples. Additionally, basic programming concepts will be explained carefully and systematically. Very basic concepts such as

looping to calculate a sum, counting in a conditional loop, and error-checking are not found in many texts but are covered here.

File Input/Output

Many applications in engineering and the sciences involve manipulating large data sets that are stored in external files. Most MATLAB texts at least mention the **save** and **load** functions, and in some cases, also some of the lower-level file input/output functions. As file input and output is so fundamental to so many applications, this book will cover several low-level file input/output functions, as well as reading from and writing to spreadsheet files. Later chapters will also deal with audio and image files. These file input/output concepts are introduced gradually: first **load** and **save** in Chapter 3, then lower-level functions in Chapter 9, and finally sound and images in Chapter 13.

User-Defined Functions

User-defined functions are a very important programming concept, and yet many times the nuances and differences between concepts such as types of functions and function calls versus function headers can be very confusing to beginning programmers. Therefore, these concepts are introduced gradually. First, arguably the easiest type of functions to understand, those that calculate and return one single value, are demonstrated in Chapter 3. Later, functions that return no values and functions that return multiple values are introduced in Chapter 6. Finally, advanced function features are shown in Chapter 10.

Advanced Programming Concepts

In addition to the basics, some advanced programming concepts such as string manipulation, data structures (e.g., structures and cell arrays), recursion, anonymous functions, and variable number of arguments to functions are covered. Sorting and indexing are also addressed. All of these are again approached systematically; for example, cell arrays are covered before they are used in file input functions and as labels on pie charts.

Problem Solving Tools

In addition to the programming concepts, some basic mathematics necessary for solving many problems will be introduced. These will include statistical functions, solving sets of linear algebraic equations, and fitting curves to data. The use of complex numbers and some calculus (integration and differentiation) will also be introduced. The built-in functions in MATLAB to perform these tasks will be described.

Plots, Imaging, and Graphical User Interfaces

Simple two-dimensional plots are introduced very early in the book (Chapter 3) so that plot examples can be used throughout. A separate chapter, Chapter 12, shows more plot types, and demonstrates customizing plots and how the graphics properties are handled in MATLAB. This chapter makes use of strings and cell arrays to customize labels. Also, there is an introduction to image processing and the basics necessary to understand programming Graphical User Interfaces (GUIs) in Chapter 13.

Vectorized Code

Efficient uses of the capabilities of the built-in operators and functions in MATLAB are demonstrated throughout the book. In order to emphasize the importance of using MATLAB efficiently, the concepts and built-in functions necessary for writing vectorized code are treated very early in Chapter 2. Techniques such as preallocating vectors and using logical vectors are then covered in Chapter 5 as alternatives to selection statements and looping through vectors and matrices. Methods of determining how efficient the code is, are also covered.

Object-Oriented Programming

Creating objects and classes in MATLAB has been an option for some time, but as of R2014b, all Graphics objects are truly objects. Thus, object-oriented programming (OOP) is now a very important part of MATLAB programming. A new chapter has been introduced on this topic, and applications using App Designer reinforce the concepts.

LAYOUT OF TEXT

This text is divided into two parts: the first part covers programming constructs and demonstrates the programming method versus efficient use of built-in functions to solve problems. The second part covers tools that are used for basic problem solving, including plotting, image processing, and techniques to solve systems of linear algebraic equations, fit curves to data, and perform basic statistical analyses. The first six chapters cover the very basics in MATLAB and in programming, and are all prerequisites for the rest of the book. After that, many chapters in the problem solving section can be introduced when desired, to produce a customized flow of topics in the book. This is true to an extent, although the order of the chapters has been chosen carefully to ensure that the coverage is systematic.

The individual chapters are described here, as well as which topics are required for each chapter.

PART 1: INTRODUCTION TO PROGRAMMING USING MATLAB

Chapter 1: Introduction to MATLAB begins by covering the MATLAB Desktop Environment. Variables, assignment statements, and types are introduced. Mathematical and relational expressions and the operators used in them are covered, as are characters, random numbers, and the use of built-in functions and the Help browser.

Chapter 2: Vectors and Matrices introduces creating and manipulating vectors and matrices. Array operations and matrix operations (such as matrix multiplication) are explained. The use of vectors and matrices as function arguments, and functions that are written specifically for vectors and matrices are covered. Logical vectors and other concepts useful in vectorizing code are emphasized in this chapter.

Chapter 3: Introduction to MATLAB Programming introduces the idea of algorithms and scripts. This includes simple input and output, and commenting. Scripts are then used to create and customize simple plots, and to do file input and output. Finally, the concept of a user-defined function is introduced with only the type of function that calculates and returns a single value.

Chapter 4: Selection Statements introduces the use of logical expressions in **if** statements, with **else** and **elseif** clauses. The **switch** statement is also demonstrated, as is the concept of choosing from a menu. Also, functions that return **logical true** or **false** are covered.

Chapter 5: Loop Statements and Vectorizing Code introduces the concepts of counted (**for**) and conditional (**while**) loops. Many common uses such as summing and counting are covered. Nested loops are also introduced. Some more sophisticated uses of loops such as error-checking and combining loops and selection statements are also covered. Finally, vectorizing code using built-in functions and operators on vectors and matrices instead of looping through them, is demonstrated. Tips for writing efficient code are emphasized, and tools for analyzing code are introduced.

The concepts in the first five chapters are assumed throughout the rest of the book.

Chapter 6: MATLAB Programs covers more on scripts and user-defined functions. User-defined functions that return more than one value and also that do not return anything are introduced. The concept of a program in MATLAB which consists of a script that calls user-defined functions is demonstrated with examples. A longer menu-driven program is shown as a reference, but could be omitted. Subfunctions and scope of variables are also introduced, as are some debugging techniques. The Live Editor is introduced.

The concept of a program is used throughout the rest of the book.

Chapter 7: String Manipulation covers many built-in string manipulation functions as well as converting between string and number types. Several examples include using custom strings in plot labels and input prompts.

Chapter 8: Data Structures: Cell Arrays and Structures introduces two main data structures: cell arrays and structures. Once structures are covered, more complicated data structures such as nested structures and vectors of structures are also introduced. Cell arrays are used in several applications in later chapters, such as file input in Chapter 9, variable number of function arguments in Chapter 10, and plot labels in Chapter 12, and are therefore considered important and are covered first. The section on structures can be omitted, although the use of structure variables to store object properties is shown in Chapter 11. Other data structures such as categorical arrays and tables are also introduced. Methods of sorting are described. Finally, the concept of indexing into a vector is introduced. Sorting a vector of structures and indexing into a vector of structures are described, but these sections can be omitted.

PART II: ADVANCED TOPICS FOR PROBLEM SOLVING WITH MATLAB

Chapter 9: Advanced File Input and Output covers lower-level file input/output statements that require opening and closing the file. Functions that can read the entire file at once as well as those that require reading one line at a time are introduced and examples that demonstrate the differences in their uses are shown. Additionally, techniques for reading from and writing to spreadsheet files and also .mat files that store MATLAB variables are introduced. Cell arrays and string functions are used extensively in this chapter.

Chapter 10: Advanced Functions covers more advanced features of and types of functions, such as anonymous functions, nested functions, and recursive functions. Function handles, and their use both with anonymous functions and function functions are introduced. The concept of having a variable number of input and/or output arguments to a function is introduced; this is implemented using cell arrays. String functions are also used in several examples in this chapter. The section on recursive functions is at the end and may be omitted.

Chapter 11: Introduction to Object-Oriented Programming and Graphics As of version R2014b, all plot objects are actual objects. This chapter introduces Object-Oriented Programming (OOP) concepts and terminology using plot objects, and then expands to how to write your own class definitions and create your own objects.

Chapter 12: Advanced Plotting Techniques continues with more on the plot functions introduced in Chapter 3. Different two-dimensional plot types, such as logarithmic scale plots, pie charts, and histograms are introduced, as is customizing plots using cell arrays and string functions. Three-dimensional plot functions as well as some functions that create the coordinates for specified objects are demonstrated. The notion of Graphics is covered, and some graphics properties such as line width and color are introduced. Core graphics objects and their use by higher-level plotting functions are demonstrated. Applications that involve reading data from files and then plotting, use both cell arrays and string functions.

Chapter 13: Sights and Sounds briefly discusses sound files, and introduces image processing. An introduction to programming Graphical User Interfaces (GUIs) is also given, including the creation of a button group and embedding images in a GUI. Nested functions are used in the GUI examples. The GUI development environment, GUIDE, is introduced. The App Designer is also introduced; it creates OOP code and builds on the concepts from Chapter 11.

Chapter 14: Advanced Mathematics covers six basic topics: it starts with some of the built-in statistical and set operations in MATLAB then curve fitting, complex numbers, solving systems of linear algebraic equations, and integration and differentiation in calculus. Matrix solutions using the Gauss-Jordan and Gauss-Jordan elimination methods are described. Finally, some of the symbolic math toolbox functions are shown, including those that solve equations. This method returns a structure as a result.

PATH THROUGH THE BOOK

It has come to my attention that not all courses that use this text use all sections. In particular, not everyone gets to images and GUIs, which are the cool applications! I have reorganized some of the chapters and sections to make it easier to get to the fun, motivating applications including images and App Designer. What follows is a path through the book to get there, including which sections can be skipped.

Chapter 1: the last two sections, 1.7 and 1.8, can be skipped

Chapter 2: section 2.5 on matrix multiplication can be skipped

Chapters 3-4: are both fundamental

Chapter 5: the last section on Timing can be skipped

Chapter 6: the last two sections can be skipped

Chapter 7: the last section can be skipped

Chapter 8: cell arrays and structures are important, but the last 3 sections can be skipped

Chapter 9: this can be skipped entirely

Chapter 10: Variable number of arguments, nested functions, and anonymous functions are all used in App Designer but the last two sections can be skipped

Chapter 11: the first two sections are fundamental but the last can be skipped

Chapter 12: this can be skipped entirely

Chapter 13: most sections are independent although the concept of callback functions is explained in the GUI section and then used in the GUIDE and App Designer section

Chapter 14: all sections can be skipped

PEDAGOGICAL FEATURES

There are several pedagogical tools that are used throughout this book that are intended to make it easier to learn the material.

First, the book takes a conversational tone with sections called *"Quick Question!"*. These are designed to stimulate thought about the material that has just been covered. The question is posed, and then the answer is given. It will be most beneficial to the reader to try to think about the question before reading the answer! In any case, they should not be skipped over, as the answers often contain very useful information.

"Practice" problems are given throughout the chapters. These are very simple problems that drill the material just covered.

"Explore Other Interesting Features" This book is not intended to be a complete reference book, and cannot possibly cover all of the built-in functions and tools available in MATLAB; however, in every chapter there will be a list of functions and/or commands that are related to the chapter topics, which readers may wish to investigate.

When some problems are introduced, they are solved using both, *"The Programming Concept"* and also *"The Efficient Method"*. This facilitates understanding the built-in functions and operators in MATLAB as well as the underlying programming concepts. *"The Efficient Method"* highlights methods that will save time for the programmer, and in many cases, are also faster to execute in MATLAB.

Additionally, to aid the reader:

- Identifier names are shown in *italic*
- MATLAB function names are shown in **bold**
- Reserved words are shown in **bold and underlined**
- Key important terms are shown in ***bold and italic***

The end of chapter "**Summary**" contains where applicable, several sections:

- **Common Pitfalls**: a list of common mistakes that are made, and how to avoid them
- **Programming Style Guidelines**: In order to encourage "good" programs that others can actually understand, the programming chapters will have guidelines that will make programs easier to read and understand and therefore easier to work with and modify.
- **Key Terms**: a list of the key terms covered in the chapter, in sequence.
- **MATLAB Reserved Words**: a list of the reserved key words in MATLAB. Throughout the text, these are shown in bold, underlined type.
- **MATLAB Functions and Commands**: a list of the MATLAB built-in functions and commands covered in the chapter, in the order covered. Throughout the text, these are shown in bold type.
- **MATLAB Operators**: a list of the MATLAB operators covered in the chapter, in the order covered.
- **Exercises**: a comprehensive set of exercises, ranging from the rote to more engaging applications.

ADDITIONAL BOOK RESOURCES

A companion website with additional teaching resources is available for faculty using this book as a text for their course(s). Please visit www.textbooks.elsevier.com/9780128045251 to register for access to:

- Instructor solutions manual for end of chapter problems
- Instructor solutions manual for "Practice" problems
- Electronic figures from the text for creation of lecture slides
- Downloadable M-files for all examples in the text

Other book-related resources will also be posted there from time to time.

This page intentionally left blank

Acknowledgments

I am indebted to many, many family members, colleagues, mentors, and students.

Throughout the last 29 years of coordinating and teaching the basic computation courses for the College of Engineering at Boston University, I have been blessed with many fabulous students as well as graduate teaching fellows and undergraduate teaching assistants (TAs). There have been hundreds of TAs over the years, too many to name individually, but I thank them all for their support. In particular, the following TAs were very helpful in reviewing drafts of the original manuscript and subsequent editions and suggesting examples: Edy Tan, Megan Smith, Brandon Phillips, Carly Sherwood, Ashmita Randhawa, Kevin Ryan, Brian Hsu, Paul Vermilion, Ben Duong, Carlton Duffett, and Raaid Arshad. Kevin Ryan wrote the MATLAB scripts that were used to produce the cover illustrations. Vincent Barilla and Isabella Passaro helped proofread and created problems for the new chapter on OOP for this edition. Carlton and Raaid have been particularly helpful over the last couple of years and have developed many companion Cody Coursework problems.

A number of colleagues have been very encouraging throughout the years. In particular, I would like to thank Tom Bifano and Ron Roy for their support and motivation. I am also indebted to my mentors at Boston University, Bill Henneman of the Computer Science Department and Merrill Ebner of the Department of Manufacturing Engineering as well as Bob Cannon from the University of South Carolina.

I would like to thank all of the reviewers of the proposal and drafts of this book. Their comments have been extremely helpful, and I hope I have incorporated their suggestions to their satisfaction. They include: Dr. Montasir Abbas, Assistant Professor, Virginia Tech; Dr Ruijun Bu, Senior Lecturer, University of Liverpool; Dr. Antonio H. Costa, Professor, University of Massachusetts, Dartmouth; Dr. Patrick Flaherty, Assistant Professor, University of Massachusetts, Amherst; Dr. Daniel Fridline. Assistant Professor, SUNY Maritime College; Dr. Matthias Krauledat, Professor, Hochschule Rhein-Waal;

Dr. Roman Kuc, Professor, Yale University; Dr. Matthew A. Lackner, Associate Professor, University of Massachusetts, Amherst; Ryan Patrick, Lecturer, University of Nebraska, Lincoln; Alison Pechenick, Senior Lecturer, The University of Vermont; Dr. Charles Riedesel, Professor, University of Nebraska, Lincoln; Dr. Daniel F. Ryder, Jr., Associate Professor, Tufts University; Cheryl Schlittler, Instructor, University of Colorado, Colorado Springs; Dr. Geoffrey Shiflett, Associate Professor, University of Southern California; Dr. Hossein Tavana, Assistant Professor, University of Akron; Shaikh Md Rubayiat Tousif, Instructor, The University of Texas at Arlington; Phillip Wong, Lab Coordinator, Portland State University; and Dr. Vladislav V. Yakovlev, Professor, Texas A&M University.

Also, I thank those at Elsevier who helped to make this book possible including: Todd Green, Publisher; Stephen Merken, Acquisitions Editor; Nate McFadden, Sr. Developmental Editor; and Sujatha Thirugnana Sambandam, Production Project Manager.

Finally, thanks go to all of my family, especially my parents Roy Attaway and Jane Conklin, both of whom encouraged me at an early age to read and to write. Thanks also to my husband Ted de Winter for his encouragement and good-natured taking care of the weekend chores while I worked on this project!

The photo used in the image processing section was taken by Ron Roy.

Stormy Attaway
Department of Mechanical Engineering
Boston University

PART

Introduction to Programming Using MATLAB

This page intentionally left blank

CHAPTER 1

Introduction to MATLAB

KEY TERMS

prompt
programs
script files
toolstrip
variable
assignment statement
assignment operator
user
initializing
incrementing
decrementing
identifier names
reserved words
keywords
mnemonic
types
classes
double precision
floating point
unsigned
characters
strings
default
continuation operator
ellipsis
unary
operand
binary
scientific notation
exponential notation
precedence
associativity
nested parentheses
inner parentheses
help topics
call a function
arguments
returning values
tab completion
constants
random numbers
seed
pseudorandom
open interval
global stream
character encoding
character set
relational expression
Boolean expression
logical expression
relational operators
logical operators
scalars
short-circuit operators
truth table
commutative
roundoff errors
range
casting
type casting
saturation arithmetic
locale setting
logarithm
common logarithm
natural logarithm

CONTENTS

MATLAB® is a very powerful software package that has many built-in tools for solving problems and developing graphical illustrations. The simplest method for using the MATLAB product is interactively; an expression is entered by the user and MATLAB responds immediately with a result. It is also possible to write

MATLAB®. http://dx.doi.org/10.1016/B978-0-12-804525-1.00001-5

scripts and programs in MATLAB, which are essentially groups of commands that are executed sequentially.

This chapter will focus on the basics, including many operators and built-in functions that can be used in interactive expressions.

1.1 GETTING INTO MATLAB

MATLAB is a mathematical and graphical software package with numerical, graphical, and programming capabilities. It includes an integrated development environment, as well as both procedural and object-oriented programming constructs. It has built-in functions to perform many operations, and there are Toolboxes that can be added to augment these functions (e.g., for signal processing). There are versions available for different hardware platforms, in both professional and student editions. The MathWorks releases two versions of MATLAB annually, named by the year and 'a' or 'b'. This book covers the releases through Version R2016a. In cases where there have been changes in recent years, these are noted.

When the MATLAB software is started, a window opens in which the main part is the Command Window (see Fig. 1.1). In the Command Window, you should see:

```
>>
```

The >> is called the ***prompt***. In the Student Edition, the prompt instead is:

```
EDU>>
```

In the Command Window, MATLAB can be used interactively. At the prompt, any MATLAB command or expression can be entered, and MATLAB will respond immediately with the result.

It is also possible to write ***programs*** in MATLAB that are contained in ***script files*** or MATLAB code files. Programs will be introduced in Chapter 3.

The following commands can serve as an introduction to MATLAB and allow you to get help:

- **demo** will bring up MATLAB Examples in the Help Browser, which has examples of some of the features of MATLAB
- **help** will explain any function; **help help** will explain how help works
- **lookfor** searches through the help for a specific word or phrase (Note: this can take a long time)
- **doc** will bring up a documentation page in the Help Browser

To exit from MATLAB, either type **quit** or **exit** at the prompt, or click on MATLAB, then Quit MATLAB from the menu.

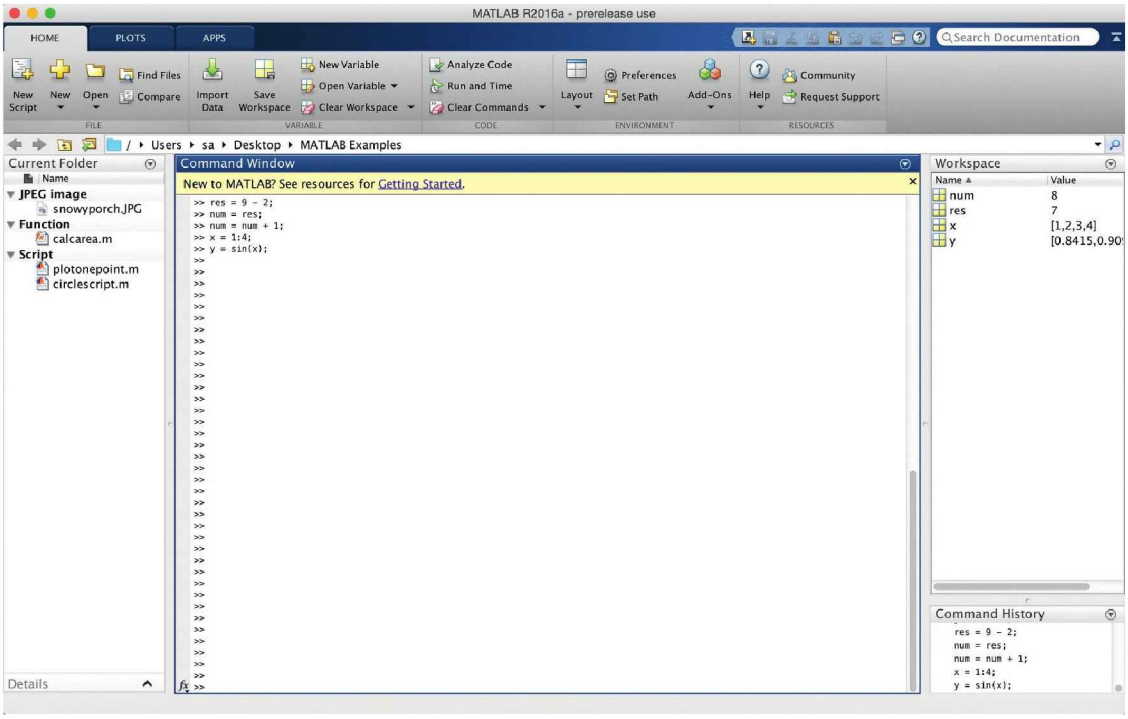

FIGURE 1.1
MATLAB Command Window.

1.2 THE MATLAB DESKTOP ENVIRONMENT

In addition to the Command Window, there are several other windows that can be opened and may be opened by default. What is described here is the default layout for these windows in Version R2016a, although there are other possible configurations. Different versions of MATLAB may show other configurations by default, and the layout can always be customized. Therefore, the main features will be described briefly here.

To the left of the Command Window is the Current Folder Window. The folder that is set as the Current Folder is where files will be saved. This window shows the files that are stored in the Current Folder. These can be grouped in many ways, for example by type, and sorted, for example by name. If a file is selected, information about that file is shown on the bottom where it says "Details."

To the right of the Command Window are the Workspace Window on top and the Command History Window on the bottom. The Command History Window shows commands that have been entered, not just in the current session (in the current Command Window), but previously as well. The Workspace Window will be described in the next section.

This default configuration can be altered by clicking the down arrow at the top right corner of each window. This will show a menu of options (different for each window), including, for example, closing that particular window and undocking that window. Once undocked, bringing up the menu and then clicking on the curled arrow pointing to the lower right will dock the window again. To make any of these windows the active window, click the mouse in it. By default the active window is the Command Window.

Beginning with Version 2012b, the Desktop now has a ***toolstrip***. By default, three tabs are shown ("HOME," "PLOTS," and "APPS"), although another, "SHORTCUTS," can be added.

Under the "HOME" tab there are many useful features, which are divided into functional sections "FILE," "VARIABLE," "CODE," "ENVIRONMENT," and "RESOURCES" (these labels can be seen on the very bottom of the grey toolstrip area). For example, under "ENVIRONMENT," hitting the down arrow under Layout allows for customization of the windows within the Desktop Environment including adding the SHORTCUTS tab. Other toolstrip features will be introduced in later chapters when the relevant material is explained.

1.3 VARIABLES AND ASSIGNMENT STATEMENTS

To store a value in a MATLAB session, or in a program, a ***variable*** is used. The Workspace Window shows variables that have been created and their values. One easy way to create a variable is to use an ***assignment statement***. The format of an assignment statement is

```
variablename = expression
```

The variable is always on the left, followed by the = symbol, which is the ***assignment operator*** (unlike in mathematics, the single equal sign does *not* mean equality), followed by an expression. The expression is evaluated and then that value is stored in the variable. Here is an example of how it would appear in the Command Window:

```
>> mynum = 6
mynum =
     6
>>
```

Here, the ***user*** (the person working in MATLAB) typed "mynum = 6" at the prompt, and MATLAB stored the integer 6 in the variable called *mynum*, and then displayed the result followed by the prompt again. As the equal sign is the assignment operator, and does not mean equality, the statement should be read as "mynum gets the value of 6" (*not* "mynum equals 6").

Note that the variable name must always be on the left, and the expression on the right. An error will occur if these are reversed.

```
>> 6 = mynum
   6 = mynum
     |
Error: The expression to the left of the equals sign is not a valid target
for an assignment.
>>
```

Putting a semicolon at the end of a statement suppresses the output. For example,

```
>> res = 9 - 2;
>>
```

This would assign the result of the expression on the right side, the value 7, to the variable *res*; it just does not show that result. Instead, another prompt appears immediately. However, at this point in the Workspace Window both the variables *mynum* and *res* and their values can be seen.

The spaces in a statement or expression do not affect the result, but make it easier to read. The following statement, which has no spaces, would accomplish exactly the same result as the previous statement:

Note
In the remainder of the text, the prompt that appears after the result will not be shown.

```
>> res = 9 - 2;
```

MATLAB uses a default variable named *ans* if an expression is typed at the prompt and it is not assigned to a variable. For example, the result of the expression 6 + 3 is stored in the variable *ans*:

```
>> 6 + 3
ans =
      9
```

This default variable, *ans*, is reused any time when only an expression, not an assignment statement, is typed at the prompt. Note that it is not a good idea to use *ans* as a name yourself or in expressions.

A shortcut for retyping commands is to hit the up arrow ↑, which will go back to the previously typed command(s). For example, if you decide to assign the result of the expression 6 + 3 to a variable named *result* instead of using the default variable *ans*, you could hit the up arrow and then the left arrow to modify the command rather than retyping the entire statement:

```
>> result = 6 + 3
result =
      9
```