CENGAGE

MATLAB®

PROGRAMMING FOR ENGINEERS

SIXTH EDITION

Updated for MATLAB R2018a

STEPHEN J. CHAPMAN

MATLAB®
Programming for Engineers

MATLAB® Programming for Engineers

Sixth Edition

Stephen J. Chapman
BAE Systems Australia

CENGAGE

Australia • Brazil • Mexico • Singapore • Spain • United Kingdom • United States

***MATLAB Programming for Engineers*, Sixth Edition**
Stephen J. Chapman

Product Director, Global Engineering: Timothy L. Anderson

Senior Product Assistant: Alexander Sham

Content Developer: MariCarmen Constable

Associate Marketing Manager: Tori Sitcawich

Content Manager: Marianne Groth

IP Analyst: Nancy Dillon

IP Project Manager: Jillian Shafer

Production Service: RPK Editorial Services, Inc.

Compositor: MPS Limited

Senior Designer: Diana Graham

Cover Image: iStockPhoto.com/ Henrik5000

Manufacturing Planner: Doug Wilke

Library of Congress Control Number: 2018965078

Student Edition:
ISBN: 978-0-357-03039-4

Loose-leaf Edition:
ISBN: 978-0-357-03051-6

Cengage
20 Channel Center Street
Boston, MA 02210
USA

Cengage is a leading provider of customized learning solutions with employees residing in nearly 40 different countries and sales in more than 125 countries around the world. Find your local representative at **www.cengage.com**.

Cengage products are represented in Canada by Nelson Education, Ltd.

To learn more about Cengage platforms and services, register or access your online learning solution, or purchase materials for your course, visit **www.cengage.com**.

Printed in the United States of America
Print Number: 01 Print Year: 2018

This book is dedicated with love to my youngest daughter Devorah,
who just finished high school. The last *one!*

Preface

MATLAB (short for MATrix LABoratory) is a special-purpose computer program optimized to perform engineering and scientific calculations. It started life as a program designed to perform matrix mathematics, but over the years it has grown into a flexible computing system capable of solving essentially any technical problem.

The MATLAB program implements the MATLAB language and provides an extensive library of predefined functions to make technical programming tasks easier and more efficient. This extremely wide variety of functions makes it much easier to solve technical problems in MATLAB than in other languages such as Fortran or C. This book introduces the MATLAB language as it is implemented in version R2018a and shows how to use it to solve typical technical problems.

This book teaches MATLAB as a technical programming language, showing students how to write clean, efficient, and documented programs. It makes no pretense at being a complete description of all of MATLAB's hundreds of functions. Instead, it teaches the student how to use MATLAB as a computer language and how to locate any desired function with MATLAB's extensive on-line help facilities.

The first eight chapters of the text are designed to serve as the text for an "Introduction to Programming/Problem Solving" course for freshman engineering students. This material should fit comfortably into a 9-week, 3-hour-per-week course. The remaining chapters cover advanced topics such as I/O, object-oriented programming, and graphical user interfaces (GUIs). These chapters may be covered in a longer course or used as a reference by engineering students or practicing engineers who use MATLAB as a part of their coursework or employment.

Changes in the Sixth Edition

The sixth edition of this book is specifically devoted to MATLAB R2018a. In the four years since the last release, there have been many changes in MATLAB.

The most significant of these changes include the introduction of the App Designer, which includes a whole new paradigm for creating MATLAB apps; a new family of plotting functions; and strings. There have also been many smaller improvements throughout the program. The book has been revised to reflect these changes.

The major changes in this edition of the book include:

- An increase in the number of MATLAB applications featured in the chapters, with more end-of-chapter exercises using them.
- More extensive coverage of plots in Chapter 3 and Chapter 8. The discussion now includes most of the currently recommended plot types in MATLAB. Older deprecated plot types have been dropped from coverage as the new ones have been added.
- Coverage of the new `string` data type, along with changes in the support for character arrays.
- Coverage of the time data types: `dateTime`, `duration`, and `calendarDuration`.
- Coverage of table arrays.
- A completely rewritten Chapter 14 featuring the new App Designer and class-based GUIs.
- An extra on-line Chapter 15 featuring the older GUIDE-based GUIs; this chapter can be downloaded from the book's website.

The Advantages of MATLAB for Technical Programming

MATLAB has many advantages compared to conventional computer languages for technical problem solving. Among them are:

1. **Ease of Use**
   MATLAB is an interpreted language, like many versions of Basic. Like Basic, it is very easy to use. The program can be used as a scratch pad to evaluate expressions typed at the command line, or it can be used to execute large pre-written programs. Programs may be easily written and modified with the built-in integrated development environment and debugged with the MATLAB debugger. Because the language is so easy to use, it is ideal for educational use and for the rapid prototyping of new programs.

   Many program development tools are provided to make the program easy to use. They include an integrated editor/debugger, on-line documentation and manuals, a workspace browser, and extensive demos.

2. **Platform Independence**
   MATLAB is supported on many different computer systems, providing a large measure of platform independence. At the time of this writing, the language is supported on Windows 7/8/10, Linux, and the Mac. Programs written on any platform will run on all of the other platforms, and data files written on any platform may be read transparently on any other platform.

As a result, programs written in MATLAB can migrate to new platforms when the needs of the user change.

3. **Predefined Functions**
   MATLAB comes complete with an extensive library of predefined functions that provide tested and prepackaged solutions to many basic technical tasks. For example, suppose that you are writing a program that must calculate the statistics associated with an input data set. In most languages, you would need to write your own subroutines or functions to implement calculations such as the arithmetic mean, standard deviation, median, and so forth. These and hundreds of other functions are built right into the MATLAB language, making your job much easier.

   In addition to the large library of functions built into the basic MATLAB language, there are many special-purpose toolboxes available to help solve complex problems in specific areas. For example, a user can buy standard toolboxes to solve problems in Signal Processing, Control Systems, Communications, Image Processing, and Neural Networks, among many others.

4. **Device-Independent Plotting**
   Unlike other computer languages, MATLAB has many integral plotting and imaging commands. The plots and images can be displayed on any graphical output device supported by the computer on which MATLAB is running. This capability makes MATLAB an outstanding tool for visualizing technical data.

5. **Graphical User Interface**
   MATLAB includes tools that allow a programmer to interactively construct a GUI for his or her program. With this capability, the programmer can design sophisticated data analysis programs that can be operated by relatively inexperienced users.

Features of This Book

Many features of this book are designed to emphasize the proper way to write reliable MATLAB programs. These features should serve a student well as he or she is first learning MATLAB and should also be useful to the practitioner on the job. These features include:

1. **Emphasis on Top-Down Design Methodology**
   The book introduces a top-down design methodology in Chapter 4 and then uses it consistently throughout the rest of the book. This methodology encourages a student to think about the proper design of a program *before* beginning to code. It emphasizes the importance of clearly defining the problem to be solved and the required inputs and outputs before any other work is begun. Once the problem is properly defined, the methodology teaches the student to employ stepwise refinement to break the task down

into successively smaller sub-tasks, and to implement the sub-tasks as separate subroutines or functions. Finally, it teaches the importance of testing at all stages of the process, both unit testing of the component routines and exhaustive testing of the final product.

The formal design process taught by the book may be summarized as follows:

1. *Clearly state the problem that you are trying to solve*.
2. *Define the inputs required by the program and the outputs to be produced by the program*.
3. *Describe the algorithm that you intend to implement in the program*. This step involves top-down design and stepwise decomposition, using pseudocode or flow charts.
4. *Turn the algorithm into MATLAB statements*.
5. *Test the MATLAB program*. This step includes unit testing of specific functions as well as exhaustive testing of the final program with many different data sets.

2. **Emphasis on Functions**
The book emphasizes the use of functions to logically decompose tasks into smaller subtasks. It teaches the advantages of functions for data hiding. It also emphasizes the importance of unit testing functions before they are combined into the final program. In addition, the book teaches about the common mistakes made with functions and how to avoid them.

3. **Emphasis on MATLAB Tools**
The book teaches the proper use of MATLAB's built-in tools to make programming and debugging easier. The tools covered include the Editor/Debugger, Workspace Browser, Help Browser, and GUI design tools.

4. **Good Programming Practice Boxes**
These boxes highlight good programming practices when they are introduced for the convenience of the student. In addition, the good programming practices introduced in a chapter are summarized at the end of the chapter. An example Good Programming Practice Box is as follows:

Good Programming Practice

Always indent the body of an `if` construct by two or more spaces to improve the readability of the code.

5. **Programming Pitfalls Boxes**
These boxes highlight common errors so that they can be avoided. An example Programming Pitfalls Box is as follows:

Programming Pitfalls

Make sure that your variable names are unique in the first 31 characters. Otherwise, MATLAB will not be able to tell the difference between them.

6. **Emphasis on Data Structures**
   Chapter 10 contains a detailed discussion of MATLAB data structures, including sparse arrays, cell arrays, and structure arrays. The proper use of these data structures is illustrated in the chapters on handle graphics (Chapter 13) and graphical user interfaces (Chapter 14).
7. **Emphasis on Object-Oriented MATLAB**
   Chapter 12 includes an introduction to object-oriented programming (OOP) and describes the MATLAB implementation of OOP in detail. This information is then applied in the discussion of App Designer GUIs.

Pedagogical Features

The first eight chapters of this book are specifically designed to be used in a freshman "Introduction to Program/Problem Solving" course. It should be possible to cover this material comfortably in a 9-week, 3-hour-per-week course. If there is insufficient time to cover all of the material in a particular Engineering program, Chapter 8 may be omitted, and the remaining material will still teach the fundamentals of programming and using MATLAB to solve problems. This feature should appeal to harassed engineering educators trying to cram ever more material into a finite curriculum.

The remaining chapters cover advanced material that will be useful to the engineer and engineering students as they progress in their careers. This material includes advanced I/O, object-oriented programming, and the design of GUIs for programs.

The book includes several features designed to aid student comprehension. A total of 20 quizzes appear scattered throughout the chapters, with answers to all questions included in Appendix B. These quizzes can serve as a useful self-test of comprehension. In addition, there are approximately 230 end-of-chapter exercises. Answers to all exercises are included in the Instructor's Solutions Manual. Good programming practices are highlighted in all chapters with special Good Programming Practice boxes, and common errors are highlighted in Programming Pitfalls boxes. End-of-chapter materials include Summaries of Good Programming Practice and Summaries of MATLAB Commands and Functions.

The book is accompanied by an Instructor's Solutions Manual, which contains the solutions to all end-of-chapter exercises. The source code for all examples in

the book is available from the book's website at https://login.cengage.com, and the source code for all solutions in the Instructor's Manual is available separately to instructors.

A Final Note to the User

No matter how hard I try to proofread a document like this book, it is inevitable that some typographical errors will slip through and appear in print. If you should spot any such errors, please drop me a note via the publisher, and I will do my best to get these errors eliminated from subsequent printings and editions. Thank you very much for your help in this matter.

I will maintain a complete list of errata and corrections at the book's website, which is available through https://login.cengage.com. Please check that site for any updates and/or corrections.

Acknowledgments

I would like to thank all my friends at Cengage Learning for the support they have given me in getting this book to market.

In addition, I would like to thank my wife Rosa, and our children Avi, David, Rachel, Aaron, Sarah, Naomi, Shira, and Devorah for their help and encouragement.

Stephen J. Chapman
Melbourne, Australia

Digital Resources

New Digital Solution for Your Engineering Classroom

WebAssign is a powerful digital solution designed by educators to enrich the engineering teaching and learning experience. With a robust computational engine at its core, WebAssign provides extensive content, instant assessment, and superior support.

WebAssign's powerful question editor allows engineering instructors to create their own questions or modify existing questions. Each question can use any combination of text, mathematical equations and formulas, sound, pictures, video, and interactive HTML elements. Numbers, words, phrases, graphics, and sound or video files can be randomized so that each student receives a different version of the same question.

In addition to common question types such as multiple choice, fill-in-the-blank, essay, and numerical, you can also incorporate robust answer entry palettes (mathPad, chemPad, calcPad, physPad, pencilPad, Graphing Tool) to input and grade symbolic expressions, equations, matrices, and chemical structures using powerful computer algebra systems. You can even use Camtasia to embed "clicker" questions that are automatically scored and recorded in the GradeBook.

WebAssign Offers Engineering Instructors the Following

- The ability to create and edit algorithmic and numerical exercises.
- The opportunity to generate randomized iterations of algorithmic and numerical exercises. When instructors assign numerical WebAssign homework exercises (engineering math exercises), the WebAssign program offers them the ability to generate and assign their students differing versions of the same engineering math exercise. The computational engine extends beyond and provides the luxury of solving for correct solutions/answers.
- The ability to create and customize numerical questions, allowing students to enter units, use a specific number of significant digits, use a specific number of decimal places, respond with a computed answer, or answer within a different tolerance value than the default.

Visit https://www.webassign.com/instructors/features/ to learn more. To create an account, instructors can go directly to the signup page at http://www.webassign.net/signup.html.

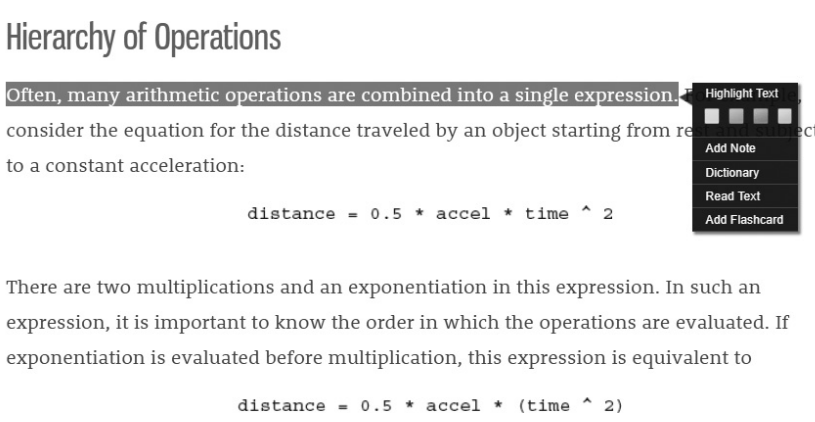
MindTap Reader

Available via WebAssign and our digital subscription service, Cengage Unlimited, **MindTap Reader** is Cengage's next-generation eBook for engineering students.

The MindTap Reader provides more than just text learning for the student. It offers a variety of tools to help our future engineers learn chapter concepts in a way that resonates with their workflow and learning styles.

- **Personalize their experience**

Within the MindTap Reader, students can highlight key concepts, add notes, and bookmark pages. These are collected in My Notes, ensuring they will have their own study guide when it comes time to study for exams.

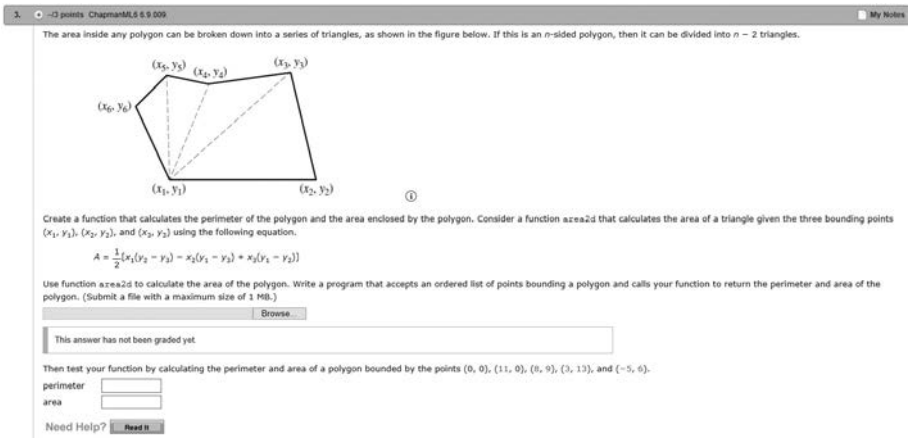


- **Flexibility at their fingertips**

With access to Merriam-Webster's Dictionary and the book's internal glossary, students can personalize their study experience by creating and collating their own custom flashcards. The ReadSpeaker feature reads text aloud to students, so they can learn on the go—wherever they are.

- **Review concepts at point of use**

Within WebAssign, a "Read It" button at the bottom of each question links students to corresponding sections of the textbook, enabling access to the MindTap Reader at the precise moment of learning. A "Watch It" button causes a short video to play. These videos allow students to better understand and review the problem they need to complete, enabling support at the precise moment of learning.



The MindTap Mobile App

Available on iOS and Android smartphones, the MindTap Mobile App provides convenience. Students can access their entire textbook anyplace and anytime. They can take notes, highlight important passages, and have their text read aloud whether they are on-line or off.

To download the mobile app, visit https://www.cengage.com/mindtap/mobileapp.

CENGAGE UNLIMITED

All-You-Can-Learn Access with Cengage Unlimited

Cengage Unlimited is the first-of-its-kind digital subscription that gives students total and on-demand access to all the digital learning platforms, eBooks, on-line homework, and study tools Cengage has to offer—in one place, for one price. With Cengage Unlimited, students get access to their WebAssign courseware, as well as content in other Cengage platforms and course areas from day one. That's 70 disciplines and 675 courses worth of material, including engineering.

With Cengage Unlimited, students get **unlimited access** to a library of more than 22,000 products. To learn more, visit https://www.cengage.com/unlimited.

Contents

Chapter 3 Two-Dimensional Plots 111

Chapter 4 Branching Statements and Program Design 157

Chapter 12 User-Defined Classes and Object-Oriented Programming 615

Chapter 13 Handle Graphics and Animation 685

Chapter 15 Guide-Based Graphical User Interfaces (On-line Only)

Chapter 1

Introduction to MATLAB

MATLAB (short for MATrix LABoratory) is a special-purpose computer program optimized to perform engineering and scientific calculations. It started life as a program designed to perform matrix mathematics, but over the years it has grown into a flexible computing system capable of solving essentially any technical problem.

The MATLAB program implements the MATLAB programming language and provides a very extensive library of predefined functions to make technical programming tasks easier and more efficient. This book introduces the MATLAB language as it is implemented in MATLAB Version 2018A and shows how to use it to solve typical technical problems.

MATLAB is a huge program with an incredibly rich variety of functions. Even the basic version of MATLAB without any toolkits is much richer than other technical programming languages. There are more than 1000 functions in the basic MATLAB product alone, and the toolkits extend this capability with many more functions in various specialties. Furthermore, these functions often solve very complex problems (solving differential equations, inverting matrices, and so forth) in a *single step*, saving large amounts of time. Doing the same thing in another computer language usually involves writing complex programs yourself or buying a third-party software package (such as IMSL, the Intel® Math Kernel Library, or the NAG software libraries) that contains the functions.

The built-in MATLAB functions are almost always better than anything that an individual engineer could write on his or her own because many people have worked on them, and they have been tested against many different data sets. These functions are also robust, producing sensible results for wide ranges of input data and gracefully handling error conditions.

This book makes no attempt to introduce users to all of MATLAB's functions. Instead, it teaches users the basics of how to write, debug, and optimize good MATLAB programs, and it introduces a subset of the most important functions used to solve common scientific and engineering problems. Just as importantly, it teaches

the scientist or engineer how to use MATLAB's own tools to locate the right function for a specific purpose from the enormous variety of choices available. In addition, it teaches how to use MATLAB to solve many practical engineering problems, such as vector and matrix algebra, curve fitting, differential equations, and data plotting.

The MATLAB program is a combination of a procedural programming language, an integrated development environment (IDE) that includes an editor and debugger, and an extremely rich set of functions that perform many types of technical calculations.

The MATLAB language is a procedural programming language, meaning that the engineer writes *procedures*, which are effectively mathematical recipes for solving a problem. This makes MATLAB very similar to other procedural languages such as C or Fortran. However, the extremely rich list of predefined functions and plotting tools makes it superior to these other languages for many engineering analysis applications.

In addition, the MATLAB language includes object-oriented extensions that allow engineers to write object-oriented programs. These extensions are similar to other object-oriented languages such as C++ or Java.

1.1 The Advantages of MATLAB

MATLAB has many advantages compared to conventional computer languages for technical problem solving. Among them are the following:

1. **Ease of Use**
   MATLAB is an interpreted language, like many versions of Basic. Like Basic, it is very easy to use. The program can be used as a scratch pad to evaluate expressions typed at the command line, or it can be used to execute large prewritten programs. Programs may be easily written and modified with the built-in integrated development environment and debugged with the MATLAB debugger. Because the language is so easy to use, it is ideal for the rapid prototyping of new programs.

   Many program development tools are provided to make the program easy to use. They include an integrated editor/debugger, on-line documentation and manuals, a workspace browser, and extensive demos.
2. **Platform Independence**
   MATLAB is supported on many different computer systems and thus enables a large measure of platform independence. At the time of this writing, the language is supported on Windows 7/8.1/10, Linux, and the Apple Mac operating system. Programs written on any platform will run on all of the other platforms, and data files written on any platform may be read transparently on any other platform. As a result, programs written in MATLAB can migrate to new platforms when the needs of the user change.
3. **Predefined Functions**
   MATLAB comes complete with an extensive library of predefined functions that provide tested and prepackaged solutions to many basic technical tasks. For example, suppose that you are writing a program that must calculate the

statistics associated with an input data set. In most languages, you would need to write your own subroutines or functions to implement calculations such as the arithmetic mean, standard deviation, median, and so forth. These and hundreds of other functions are built right into the MATLAB language, making your job much easier.

In addition to the large library of functions built into the basic MATLAB language, there are many special-purpose toolboxes available to help solve complex problems in specific areas. For example, you can buy standard toolboxes to solve problems in signal processing, control systems, communications, image processing, and neural networks, among many others. There is also an extensive collection of free user-contributed MATLAB programs that are shared through the MATLAB website.

4. **Device-Independent Plotting**
   Unlike most other computer languages, MATLAB has many integral plotting and imaging commands. The plots and images can be displayed on any graphical output device supported by the computer on which MATLAB is running. This capability makes MATLAB an outstanding tool for visualizing technical data.
5. **Graphical User Interface**
   MATLAB includes tools that allow an engineer to interactively construct a graphical user interface (GUI) for his or her program, and also to produce Web apps. With this capability, an engineer can design sophisticated data analysis programs that can be operated by relatively inexperienced users.
6. **MATLAB Compilers**
   MATLAB's flexibility and platform independence is achieved by compiling MATLAB programs into a device-independent p-code, and then interpreting the p-code instructions at run-time. This approach is similar to that used by Microsoft's Visual Basic language or by Java. Unfortunately, the resulting programs sometimes executed slowly because the MATLAB code is interpreted rather than compiled. Newer versions of MATLAB have partially overcome this problem by introducing just-in-time (JIT) compiler technology. The JIT compiler compiles portions of the MATLAB code as it is executed to increase overall speed.

   A separate MATLAB Coder is also available. The MATLAB Coder generates portable and readable C and C++ code from MATLAB code. This converted code can then be compiled and included in programs written in other languages. In addition, legacy code written in other languages can be compiled and used within MATLAB.

1.2 Disadvantages of MATLAB

MATLAB has two principal disadvantages. The first is that it is an interpreted language and therefore can execute more slowly than compiled languages. This problem can be mitigated by properly structuring the MATLAB program to maximize the performance of vectorized code and by using the JIT compiler.

The second disadvantage is cost: a full copy of MATLAB is 5 to 10 times more expensive than a conventional C or Fortran compiler. This relatively high cost is more than offset by the reduced time required for an engineer or scientist to create a working program, so MATLAB is cost-effective for businesses. However, it is too expensive for most individuals to consider purchasing. Fortunately, there is also an inexpensive student edition of MATLAB, which is a great tool for students wishing to learn the language. The student edition of MATLAB is essentially identical to the full edition.

1.3 The MATLAB Environment

The fundamental unit of data in any MATLAB program is the **array**. An array is a collection of data values organized into rows and columns and known by a single name. Individual data values within an array can be accessed by including the name of the array followed by subscripts in parentheses that identify the row and column of the particular value. Even scalars are treated as arrays by MATLAB—they are simply arrays with only one row and one column. We will learn how to create and manipulate MATLAB arrays in Section 1.4.

When MATLAB executes, it can display several types of windows that accept commands or display information. The three most important types of windows are Command Windows, where commands may be entered; Figure Windows, which display plots and graphs; and Edit Windows, which permit a user to create and modify MATLAB programs. We will see examples of all three types of windows in this section.

In addition, MATLAB can display other windows that provide help and that allow the user to examine the values of variables defined in memory. We will examine some of these additional windows here, and examine the others when we discuss how to debug MATLAB programs.

1.3.1 The MATLAB Desktop

When you start MATLAB Version 2018A, a special window called the MATLAB desktop appears. The desktop is a window that contains other windows showing MATLAB data, plus toolbars and a "Toolstrip" or "Ribbon Bar" similar to that used by Windows 10 or Microsoft Office. By default, most MATLAB tools are "docked" to the desktop, so that they appear inside the desktop window. However, the user can choose to "undock" any or all tools, making them appear in windows separate from the desktop.

The default configuration of the MATLAB desktop is shown in Figure 1.1. It integrates many tools for managing files, variables, and applications within the MATLAB environment.

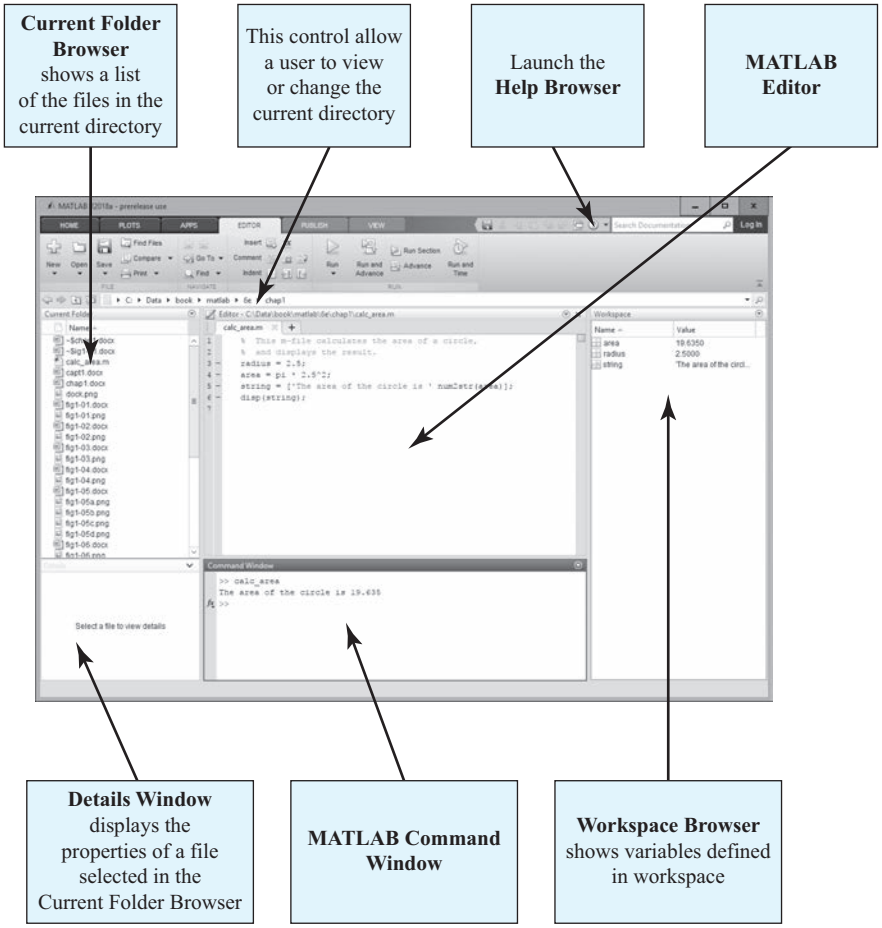


Figure 1.1 The default MATLAB desktop. The exact appearance of the desktop may differ slightly on different types of computers.

The major tools within or accessible from the MATLAB desktop are:

- The Command Window
- The Toolstrip
- The Documents Window, including the Editor/Debugger and Array Editor
- Figure Windows
- The Workspace Browser
- The Current Folder Browser, with the Details Window
- The Help Browser
- The Path Browser
- A Popup Command History Window

Table 1.1: Tools and Windows Included in the MATLAB Desktop

Tool	Description
Command Window	A window where the user can type commands and see immediate results, or where the user can execute scripts or functions
Toolstrip	A strip across the top of the desktop containing icons to select functions and tools, arranged in tabs and sections of related functions
Command History Window	A window that displays recently used commands, accessed by clicking the up arrow when typing in the Command Window
Document Window	A window that displays MATLAB files and allows the user to edit or debug them
Figure Window	A window that displays a MATLAB plot
Workspace Browser	A window that displays the names and values of variables stored in the MATLAB workspace
Current Folder Browser	A window that displays the names of files in the current directory. If a file is selected in the Current Folder Browser, details about the file will appear in the Details Window
Help Browser	A tool to get help for MATLAB functions, accessed by clicking the "Help" button on the Toolstrip
Path Browser	A tool to display the MATLAB search path, accessed by clicking the "Set Path" button on the Home tab of the Toolstrip

The functions of these tools are summarized in Table 1.1. We will discuss them in later sections of this chapter.

1.3.2 The Command Window

The bottom center of the default MATLAB desktop contains the **Command Window**. A user can enter interactive commands at the command prompt (») in the Command Window, and they will be executed on the spot.

As an example of a simple interactive calculation, suppose that you wanted to calculate the area of a circle with a radius of 2.5 m. The equation for this area of a circle is

$$A = \pi r^2 \tag{1.1}$$

where r is the radius of the circle and A is the area of the circle. This equation can be evaluated in the MATLAB Command Window by typing:

```
» area = pi * 2.5^2
area =
   19.6350
```

where `*` is the multiplication symbol and `^` is the exponential symbol. MATLAB calculates the answer as soon as the Enter key is pressed, and stores the answer in a variable (really a 1×1 array) called `area`. The contents of the variable are

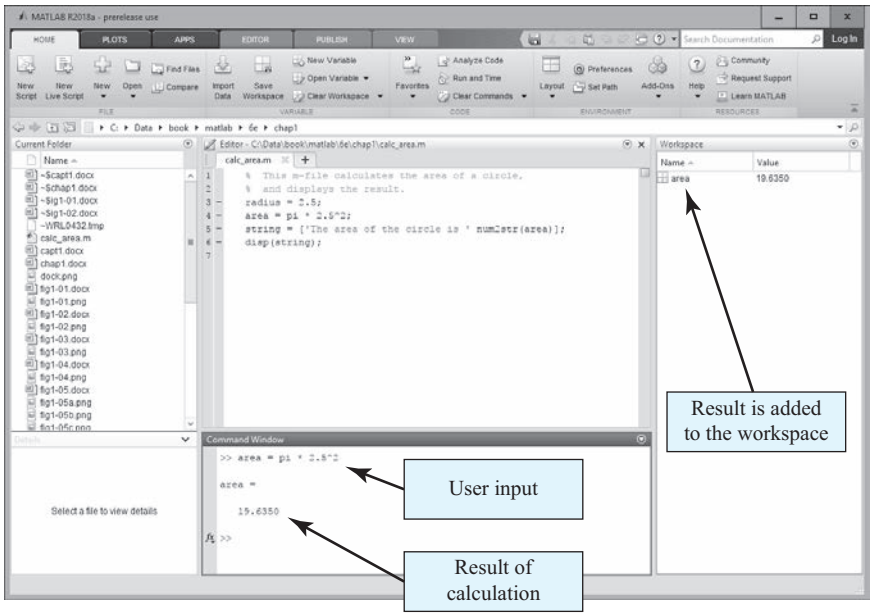


Figure 1.2 The Command Window appears in the center of the desktop. You enter commands and see responses here.

displayed in the Command Window as shown in Figure 1.2, and the variable can be used in further calculations. (Note that π is predefined in MATLAB, so we can just use `pi` without first declaring it to be 3.141592 …).

If a statement is too long to type on a single line, it may be continued on successive lines by typing an **ellipsis** (`...`) at the end of the first line and then continuing on the next line. For example, the following two statements are identical.

```
x1 = 1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6
```

and

```
x1 = 1 + 1/2 + 1/3 + 1/4 ...
   + 1/5 + 1/6
```

Instead of typing commands directly in the Command Window, a series of commands can be placed into a file, and the entire file can be executed by typing its name in the Command Window. Such files are called **script files**. Script files (and functions, which we will see later) are also known as **M-files** because they have a file extension of "`.m`".

1.3.3 The Toolstrip

The Toolstrip (see Figure 1.3) is a bar of tools that appears across the top of the desktop. The controls on the Toolstrip are organized into related categories of functions, first by tabs, and then by groups. For example, the tabs visible in Figure 1.3 are

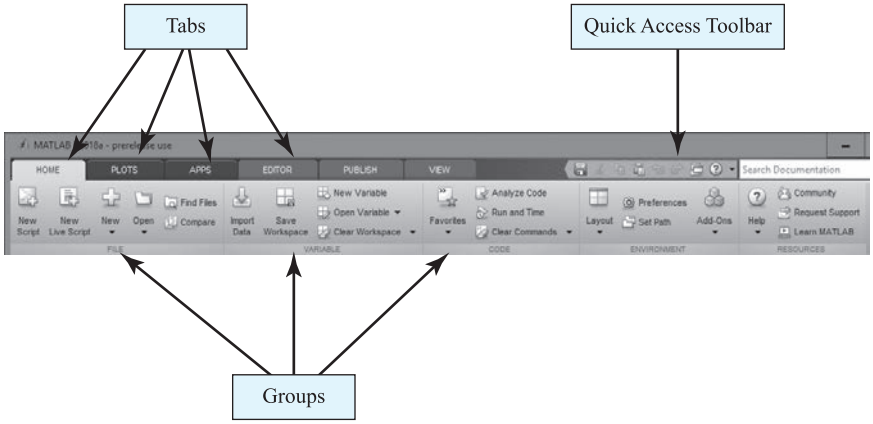


Figure 1.3 The Toolstrip, which allows you to select from a wide variety of MATLAB tools and commands.

"Home", "Plots", "Apps", "Editor", and so forth. When one of the tabs is selected, a series of controls grouped into sections is displayed. In the Home tab, the sections are "File", "Variable", "Code", and so forth. With practice, the logical grouping of commands helps the user to quickly locate any desired function.

In addition, the upper-right corner of the Toolstrip contains the Quick Access Toolbar, which is where you can customize the interface and display the most commonly used commands and functions at all times. To customize the functions displayed there, right-click on the toolbar and select the Customize option from the popup menu.

1.3.4 The Command History Window

The Command History Window displays a list of the commands that a user has previously entered in the Command Window. The list of commands can extend back to previous executions of the program. Commands remain in the list until they are deleted. To display the Command History Window, press the up arrow key while typing in the Command Window. To reexecute any command, simply double-click it with the left mouse button. To delete one or more commands from the Command History Window, select the commands and right-click them with the mouse. A popup menu will be displayed that allows the user to delete the items (see Figure 1.4).

1.3.5 The Document Window

A **Document Window** (also called an **Edit/Debug Window**) is used to create new M-files or to modify existing ones. An Edit/Debug Window is created automatically when you create a new M-file or open an existing one. You can create a new

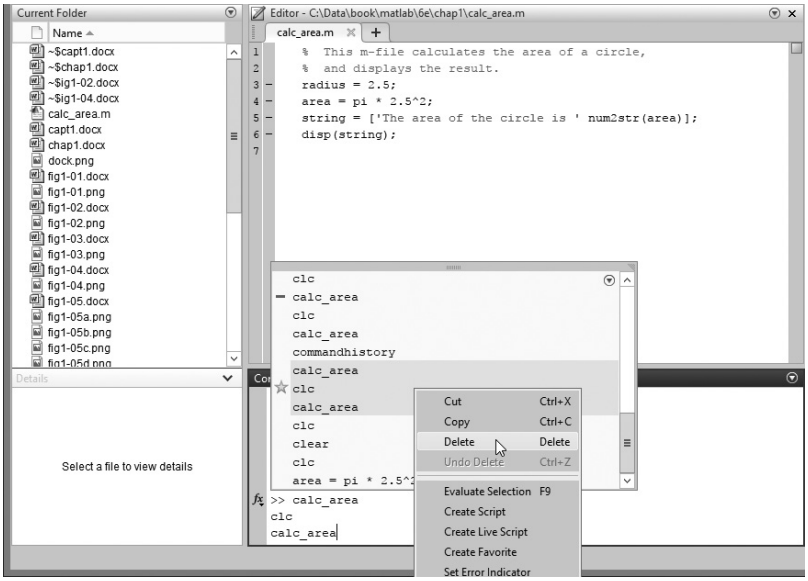


Figure 1.4 The Command History Window, showing three commands being deleted.

M-file with the "New Script" command from the "File" group on the Toolstrip (Figure 1.5a), or by clicking the New icon and selecting Script from the popup menu (Figure 1.5b). You can open an existing M-file file with the Open command from the "File" section on the Toolstrip.

An Edit/Debug Window displaying a simple M-file called `calc_area.m` is shown in Figure 1.5. This file calculates the area of a circle given its radius and displays the result. By default, the Edit Window is docked to the desktop, as shown in Figure 1.5c. The Edit Window can also be undocked from the MATLAB desktop. In that case, it appears within a container called the Documents Window, as shown in Figure 1.5d. We will learn how to dock and undock a window later in this chapter.

The Edit Window is essentially a programming text editor, with the MATLAB language's features highlighted in different colors. Comments in an M-file file appear in green, variables and numbers appear in black, complete character strings appear in magenta, incomplete character strings appear in red, and language keywords appear in blue.

After an M-file is saved, it may be executed by typing its name in the Command Window. For the M-file in Figure 1.5, the results are:

```
» calc_area
The area of the circle is 19.635
```

The Edit Window also doubles as a debugger, as we shall see in Chapter 2.