Contents

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.

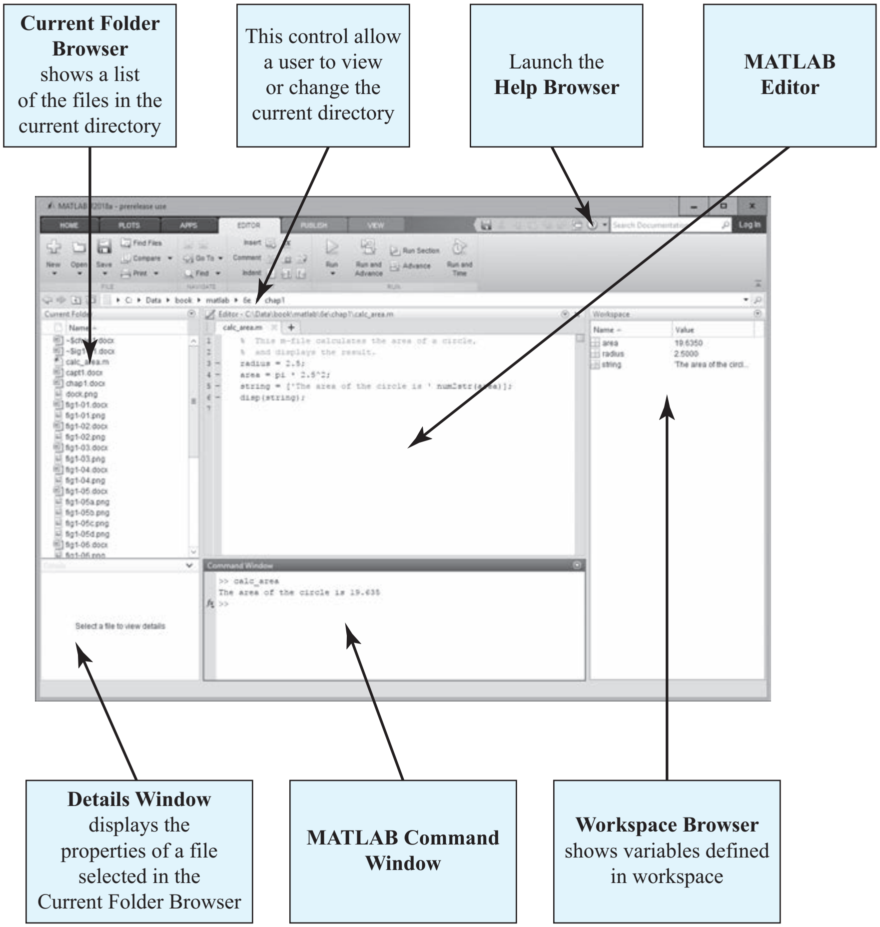


Figure 1.1 The default MATLAB desktop. The exact appearance of the desktop may differ slightly on different types of computers.

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

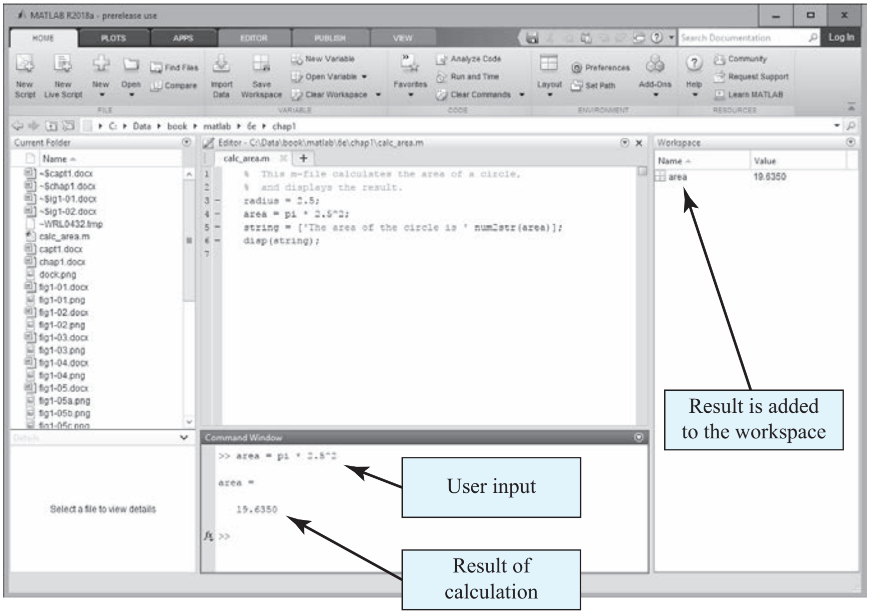


Figure 1.2 The Command Window appears in the center of the desktop. You enter commands and see responses here.

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

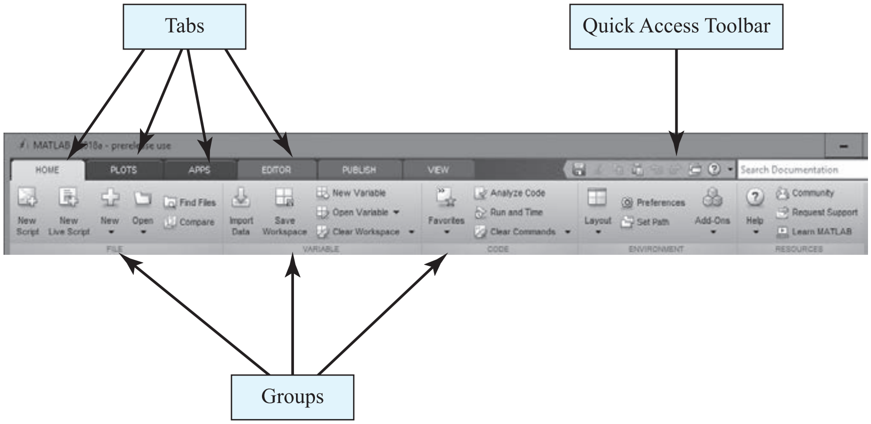


Figure 1.3 The Toolstrip, which allows you to select from a wide variety of MATLAB tools and commands.

“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

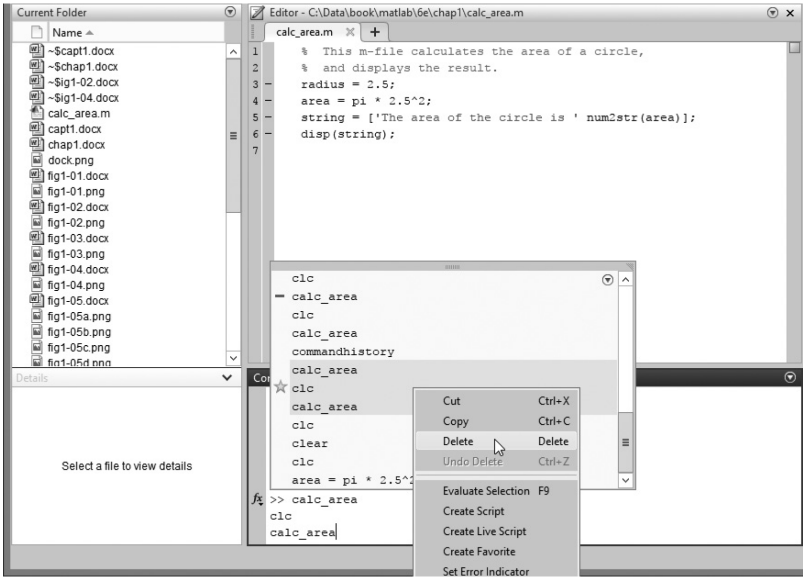


Figure 1.4 The Command History Window, showing three commands being deleted.

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.

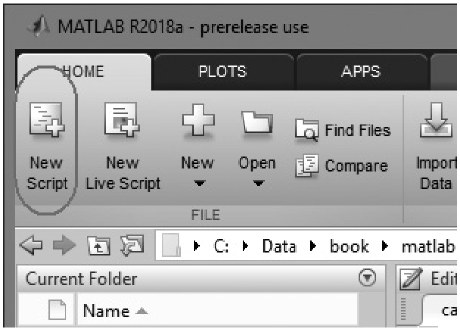
An Edit/Debug Window displaying a simple M-file called `calc_area.m` is shown in Figure 1.5. This file calculates the area of a circle given its radius and displays the result. By default, the Edit Window is docked to the desktop, as shown in Figure 1.5c. The Edit Window can also be undocked from the 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.

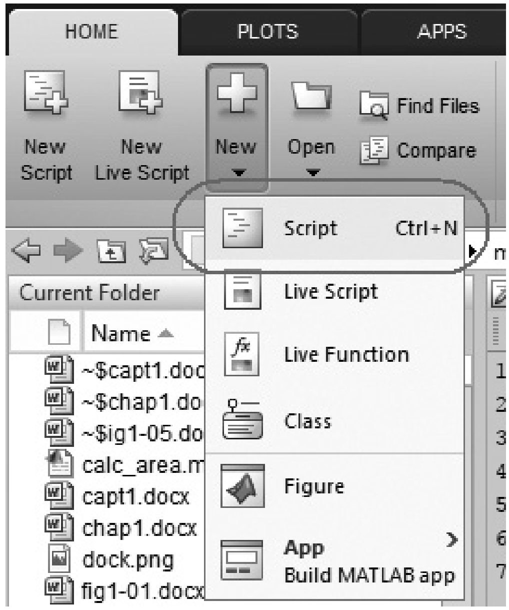
After an M-file is saved, it may be executed by typing its name in the Command Window. For the M-file in Figure 1.5, the results are:

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

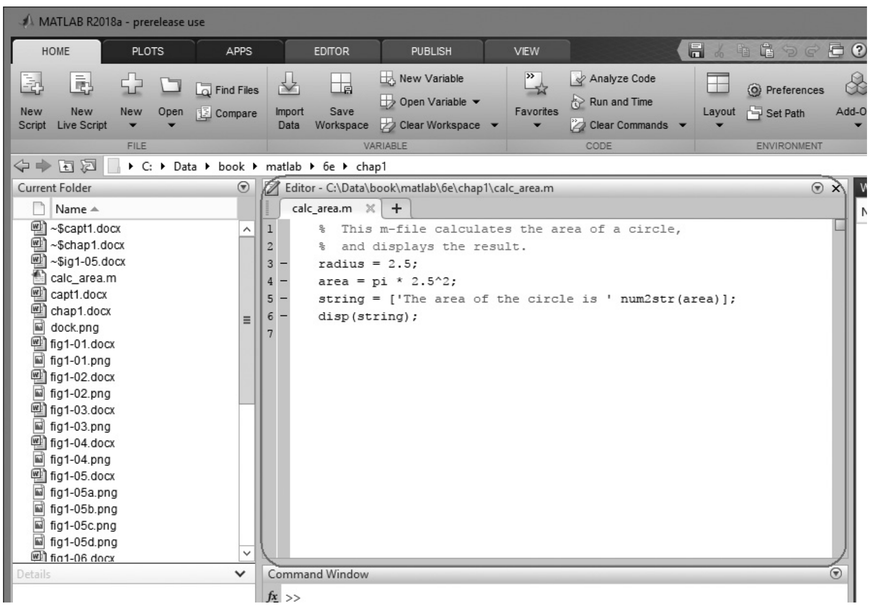
The Edit Window also doubles as a debugger, as we shall see in Chapter 2.


MATLAB R2018a - prerelease use
HOME
PLOTS
APPS
New Script
New Live Script
New
Open
Find Files
Compare
Import Data
FILE
C: ▸ Data ▸ book ▸ matlab
Current Folder
Name

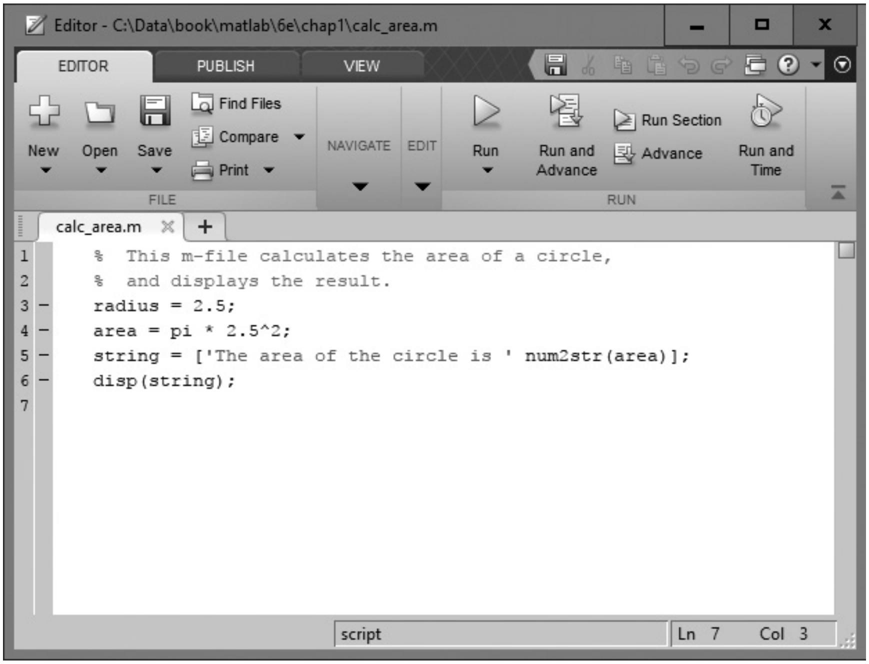
(a)


HOME
PLOTS
APPS
New Script
New Live Script
New
Open
Find Files
Compare
Script Ctrl+N
Live Script
Live Function
Class
Figure
App
Build MATLAB app
Current Folder
Name
~$capt1.doc
~$chap1.do
~$ig1-05.do
calc_area.m
capt1.docx
chap1.docx
dock.png
fig1-01.docx

(b)


MATLAB R2018a - prerelease use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
New Script
New Live Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Favorites
Analyze Code
Run and Time
Clear Commands
Layout
Preferences
Set Path
Add-O
FILE
VARIABLE
CODE
ENVIRONMENT
C: ▸ Data ▸ book ▸ matlab ▸ 6e ▸ chap1
Current Folder
Name
~$capt1.docx
~$chap1.docx
~$ig1-05.docx
calc_area.m
capt1.docx
chap1.docx
dock.png
fig1-01.docx
fig1-01.png
fig1-02.docx
fig1-02.png
fig1-03.docx
fig1-03.png
fig1-04.docx
fig1-04.png
fig1-05.docx
fig1-05a.png
fig1-05b.png
fig1-05c.png
fig1-05d.png
Details
Editor - C:\Data\book\matlab\6e\chap1\calc_area.m
calc_area.m
% This m-file calculates the area of a circle,
% and displays the result.
radius = 2.5;
area = pi * 2.5^2;
string = ['The area of the circle is ' num2str(area)];
disp(string);
Command Window
fx >>

(c)



(d)

Figure 1.5 (a) Creating a new M-file with the "New Script" command. (b) Creating a new M-file with the "New >> Script" selection from the Toolbar. (c) The MATLAB Editor, docked to the MATLAB desktop. (See color insert.) (d) The MATLAB Editor, displayed as an independent window.

1.3.6 Figure Windows

A **Figure Window** is used to display MATLAB graphics. A figure can be a two- or three-dimensional plot of data, an image, or a GUI. A simple script file that calculates and plots the function sin *x* is as follows:

```
% sin_x.m: This M-file calculates and plots the
% function sin(x) for 0 <= x <= 6.
x = 0:0.1:6
y = sin(x)
plot(x,y)
```

If this file is saved under the name `sin_x.m`, then a user can execute the file by typing "`sin_x`" in the Command Window. When this script file is executed, MATLAB opens a figure window and plots the function sin *x* in it. The resulting plot is shown in Figure 1.6.

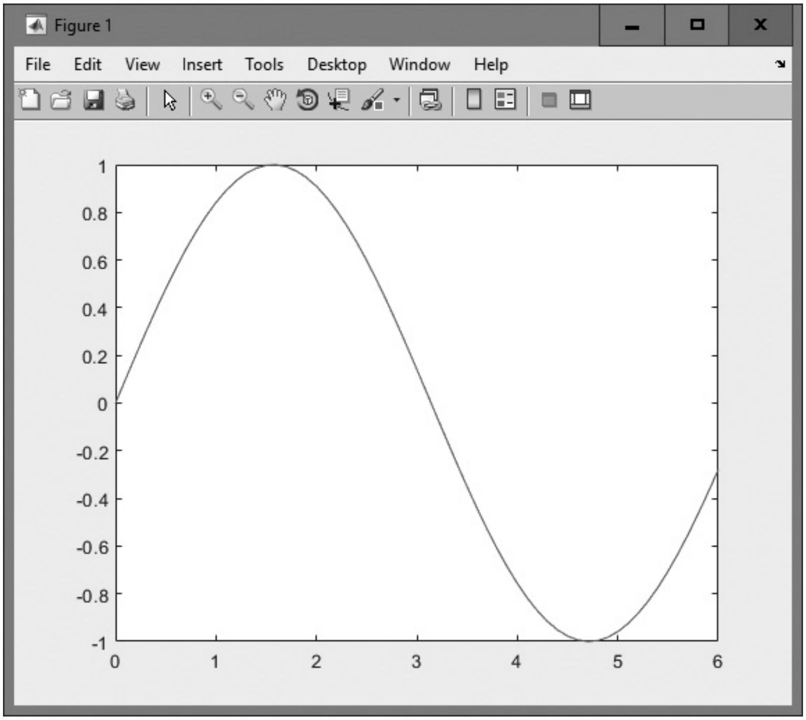


Figure 1.6 MATLAB plot of sin *x* versus *x*.

1.3.7 Docking and Undocking Windows

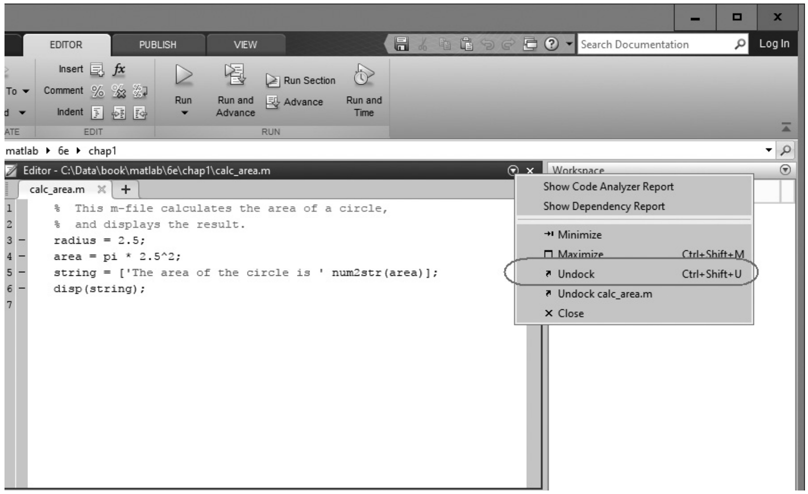
MATLAB windows such as the Command Window, the Edit/Debugging Window, and Figure Windows can either be *docked* to the desktop, or they can be *undocked*. When a window is docked, it appears as a pane within the MATLAB desktop. When it is undocked, it appears as an independent window on the computer screen separate from the desktop. When a window is docked to the desktop, it can be undocked by selecting the small down arrow in the upper-right corner and selecting the "Undock" option from the popup menu (see Figure 1.7a). When a window is an independent window, it can be docked to the desktop by selecting the small down arrow in the upper-right corner and selecting the "Dock" option from the popup menu (see Figure 1.7b).

1.3.8 The MATLAB Workspace

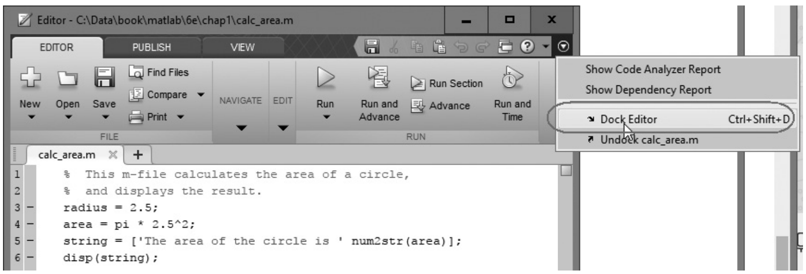
A statement like

```
z = 10
```

creates a variable named z, stores the value 10 in it, and saves it in a part of computer memory known as the **workspace**. A workspace is the collection of all the variables and arrays that can be used by MATLAB when a particular command, M-file, or function is executing. All commands executed in the Command Window (and all



(a)



(b)

Figure 1.7 (a) Selecting the "Undock" option from the menu displayed after clicking the small down arrow in the upper-right corner of a pane. (b) Selecting the "Dock" option after clicking the small down arrow in the upper-right corner of an undocked window.

script files executed from the Command Window) share a common workspace, so they can all share variables. As we will see later, MATLAB functions differ from script files in that each function has its own separate workspace.

A list of the variables and arrays in the current workspace can be generated with the `whos` command. For example, after M-files `calc_area` and `sin_x` are executed, the workspace contains the following variables.

```
» whos
  Name        Size           Bytes  Class    Attributes

  area        1x1                8  double
  radius      1x1                8  double
  string      1x32              64  char
  x           1x61             488  double
  y           1x61             488  double
```

Script file `calc_area` created variables `area`, `radius`, and `string`, while script file `sin_x` created variables `x` and `y`. Note that all of the variables are in the same workspace, so if two script files are executed in succession, the second script file can use variables created by the first script file.

The contents of any variable or array may be determined by typing the appropriate name in the Command Window. For example, the contents of `string` can be found as follows:

```
» string
string =
The area of the circle is 19.635
```

A variable can be deleted from the workspace with the `clear` command. The `clear` command takes the form

```
clear var1 var2 ...
```

where `var1` and `var2` are the names of the variables to be deleted. The command `clear variables` or simply `clear` deletes all variables from the current workspace.

1.3.9 The Workspace Browser

The contents of the current workspace can also be examined with a GUI-based Workspace Browser. The Workspace Browser appears by default in the right side of the desktop. It provides a graphic display of the same information as the `whos` command, and it also shows the actual contents of each array if the information is short enough to fit within the display area. The Workspace Browser is dynamically updated whenever the contents of the workspace change.

A typical Workspace Browser window is shown in Figure 1.8. As you can see, it displays the same information as the `whos` command. Double-clicking on any variable in the window will bring up the Array Editor, which allows the user to modify the information stored in the variable.

One or more variables may be deleted from the workspace by selecting them in the Workspace Browser with the mouse and pressing the delete key, or by right-clicking with the mouse and selecting the delete option.

1.3.10 The Current Folder Browser

The Current Folder Browser is displayed on the upper-left side of the desktop. It shows all the files in the currently selected folder, and allows the user to edit or execute any desired file. You can double-click on any M-file to open it in the MATLAB editor, or you can right-click it and select "Run" to execute it. The Current Folder Browser is shown in Figure 1.9. A toolbar above the browser is used to select the current folder to display.

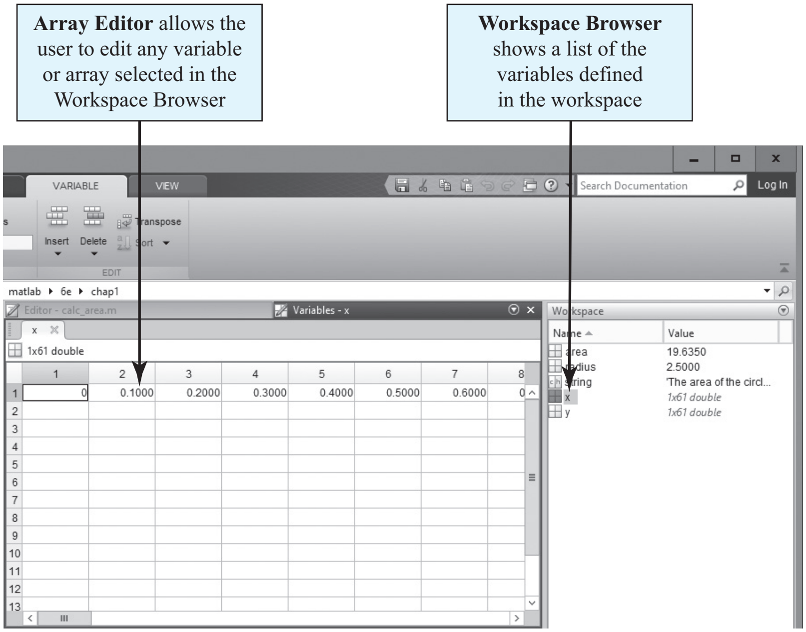


Figure 1.8 The Workspace Browser and Array Editor. The Array Editor is invoked by double-clicking a variable in the Workspace Browser. It allows you to change the values contained in a variable or array.

1.3.11 Getting Help

There are three ways to get help in MATLAB. The preferred method is to use the Help Browser. The Help Browser can be started by selecting the ? icon from the Toolstrip or by typing `doc` or `helpwin` in the Command Window. A user can get help by browsing the MATLAB documentation, or he or she can search for the details of a particular command. The Help Browser is shown in Figure 1.10.

There are also two command-line-oriented ways to get help. The first way is to type `help` or `help` followed by a function name in the Command Window. If you just type `help`, MATLAB will display a list of possible help topics in the Command Window. If a specific function or a toolbox name is included, help will be provided for that particular function or toolbox.

The second way to get help is the `lookfor` command. The `lookfor` command differs from the `help` command in that the `help` command searches for an exact function name match, while the `lookfor` command searches the quick summary information in each function for a match. This makes `lookfor` slower than `help`, but it improves the chances of getting back useful information. For example, suppose that you were looking for a function to take the inverse of a matrix. Since MATLAB does not have a function named `inverse`, the command

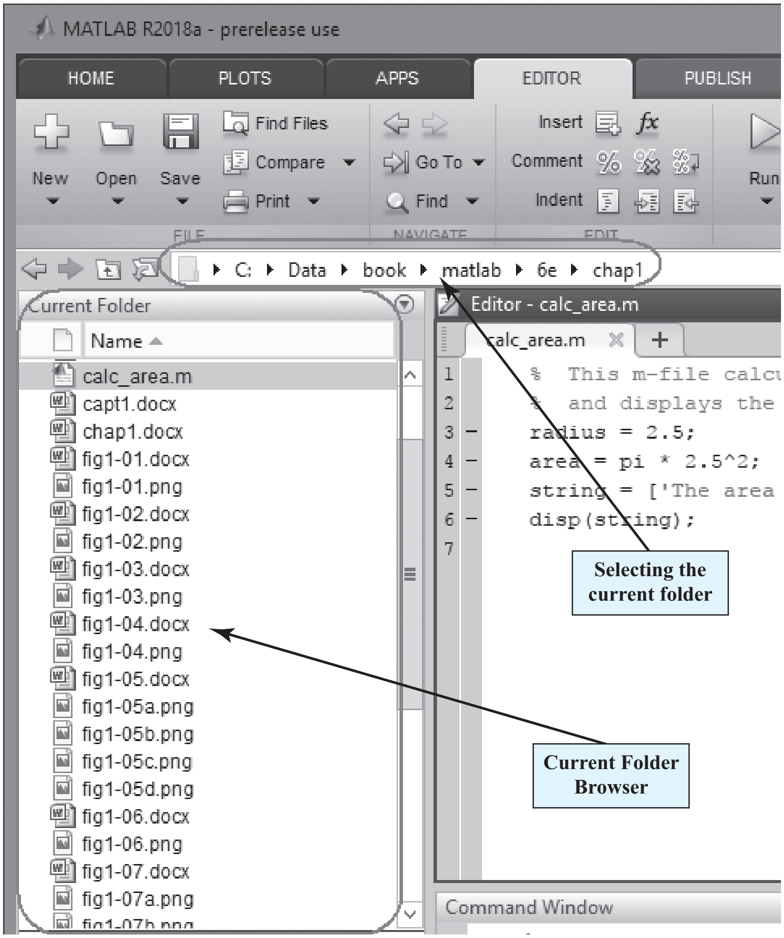


Figure 1.9 The Current Folder Browser.

"help inverse" will produce nothing. On the other hand, the command "look-for inverse" will produce the following results:

```
» lookfor inverse
ifft          - Inverse discrete Fourier transform.
ifft2         - Two-dimensional inverse discrete Fourier transform.
ifftn         - N-dimensional inverse discrete Fourier transform.
ifftshift     - Inverse FFT shift.
acos          - Inverse cosine, result in radians.
acosd         - Inverse cosine, result in degrees.
acosh         - Inverse hyperbolic cosine.
acot          - Inverse cotangent, result in radian.
acotd         - Inverse cotangent, result in degrees.
acoth         - Inverse hyperbolic cotangent.
```

```
acsc          - Inverse cosecant, result in radian.
acscd         - Inverse cosecant, result in degrees.
acsch         - Inverse hyperbolic cosecant.
asec          - Inverse secant, result in radians.
asecd         - Inverse secant, result in degrees.
asech         - Inverse hyperbolic secant.
asin          - Inverse sine, result in radians.
asind         - Inverse sine, result in degrees.
asinh         - Inverse hyperbolic sine.
atan          - Inverse tangent, result in radians.
atan2         - Four quadrant inverse tangent.
atan2d        - Four quadrant inverse tangent, result in degrees.
atand         - Inverse tangent, result in degrees.
atanh         - Inverse hyperbolic tangent.
invhilb       - Inverse Hilbert matrix.
ipermute      - Inverse permute array dimensions.
dramadah      - Matrix of zeros and ones with large determinant
                or inverse.
invhess       - Inverse of an upper Hessenberg matrix.
inv           - Matrix inverse.
pinv          - Pseudoinverse.
...
```

From this list, we can see that the function of interest is named `inv`.

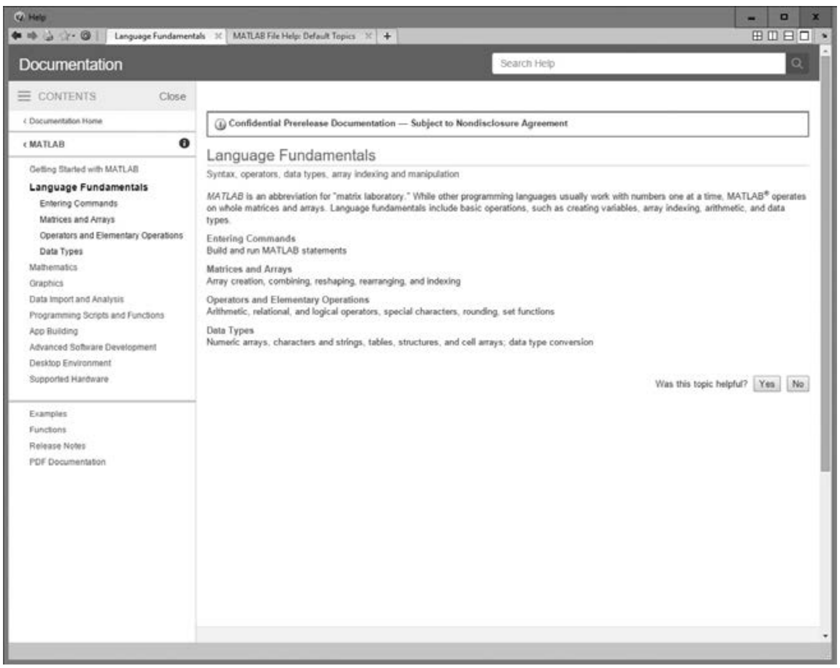


Figure 1.10 The Help Browser.

1.3.12 A Few Important Commands

If you are new to MATLAB, a few demonstrations may help to give you a feel for its capabilities. To run MATLAB's built-in demonstrations, type `demo` in the Command Window.

The contents of the Command Window can be cleared at any time using the `clc` command, and the contents of the current Figure Window can be cleared at any time using the `clf` command. The variables in the workspace can be cleared with the `clear` command. As we have seen, the contents of the workspace persist between the executions of separate commands and M-files, so it is possible for the results of one problem to have an effect on the next one that you may attempt to solve. To avoid this possibility, it is a good idea to issue the `clear` command at the start of each new independent calculation.

Another important command is the **abort** command. If an M-file appears to be running for too long, it may contain an infinite loop, and it will never terminate. In this case, the user can regain control by typing control-c (abbreviated `^c`) in the Command Window. This command is entered by holding down the control key while typing a "c". When MATLAB detects a `^c`, it interrupts the running program and returns a command prompt.

There is also an auto-complete feature in MATLAB. If a user starts to type a command and then presses the Tab key, a popup list of recently typed commands and MATLAB functions that match the string will be displayed (see Figure 1.11). The user can complete the command by selecting one of the items from the list.

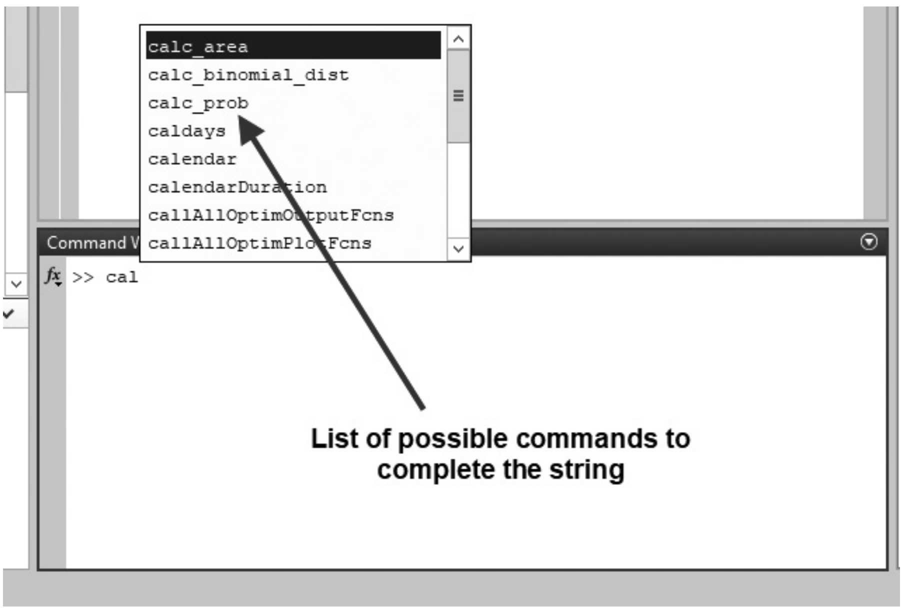


Figure 1.11 If you type a partial command and then hit the Tab key, MATLAB will pop up a window of suggested commands or functions that match the string.