

Lesson 2B.

CHAPTER 2. MATLAB FUNDAMENTALS

2.1 Introduction

- This Chapter and the Chapters that follow cover many concepts in computer programming and how to apply them to solving Engineering type problems.
- We do this on the MATLAB platform.
- Recall at the end of Chapter 1, it was mentioned that a number of building blocks in any programming language are:
 - (a) the arithmetic statement or assignment
 - (b) output/input statements, including graphs and tables
 - (c) loops
 - (d) alternate paths (conditional branches).
 - (e) functions, both built in and self written.

Items (a)-(d) are covered in this Chapter as well as some of the elementary built in functions of item (e).

Item (e) is covered in Chapter 3. Examples of self written functions and MATLAB's functions that solve particular types of mathematical problems are covered throughout the book.

- MATLAB is a software program for numeric computation, data analysis and graphics. One advantage that MATLAB has for engineers over programming

languages such as C or C++ is that the MATLAB program includes functions which numerically solve:

- (a). large systems of linear algebraic equations
- (b). systems of ordinary differential equations
- (c). roots of transcendental equations
- (d). integrals
- (e). statistical problems
- (f). optimization problems
- (g). control systems problems
- (h) many other types of problems encountered in engineering.

MATLAB also offers toolboxes (which must be purchased separately) that are designed to solve problems in specialized areas.

In this chapter the following items are covered:

- The MATLAB desktop environment.
- Constructing a program (also called a script) in MATLAB.
- MATLAB fundamentals and basic commands, including `clear`, `clc`, colon operator, arithmetic operators, trigonometric functions, logarithmic and exponential functions, and other useful functions such as `max`, `min`, and `length`.
- Output/input in MATLAB, including the `fprintf` and `input` statements.
- MATLAB program flow, including `for` loops, `while` loops, `if` and `if-elseif` statements, and the `switch` group statement.
- MATLAB graphics, including the `plot` and `subplot` commands.

- Interpolation and MATLAB's `interp1` function.
- Working with characters and strings.
- MATLAB's `textscan` function and cells.
- Debugging a program

Many example scripts are included throughout the chapter to illustrate these various topics.

2.2 MATLAB's Desktop

- Mathworks, the company that developer of MATLAB, normally updates their version of MATLAB every six months. In 2012, Mathworks introduced MATLAB version R2012b whose desktop was very different from their MATLAB version R2012a. Thus, getting started in MATLAB version R2012b was very different than getting started in MATLAB version R2012a. In this Textbook, Sections 2.2 and 2.3 discusses the MATLAB desktop windows and how to construct a script in MATLAB is based on MATLAB version R2014a.
- The MATLAB program can usually be started through the “All Programs” option in the Windows start menu and double clicking on the MATLAB listing, or by a MATLAB icon that is on the desktop.
- Upon startup, a new window will open containing the MATLAB “desktop” (not to be confused with the Windows desktop) and one or more MATLAB windows will open within the MATLAB desktop (see Figure 2.1 for the default configuration).

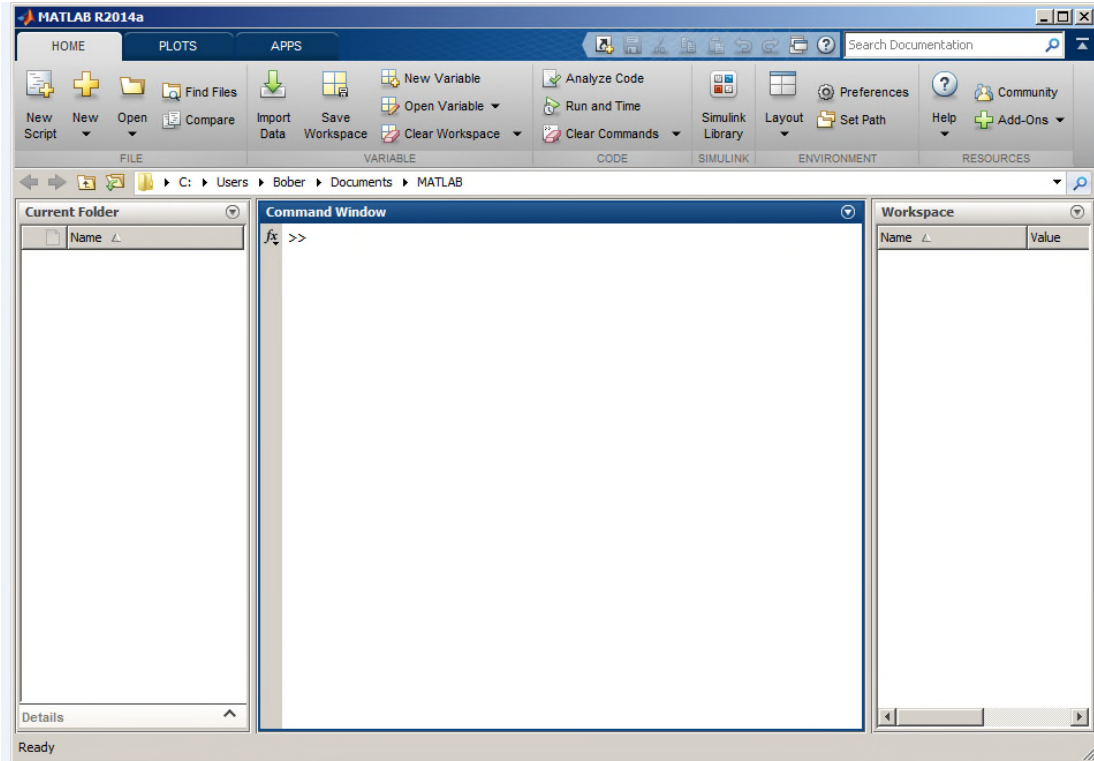


Figure 2.1. MATLAB desktop windows for the three column option.

- The main windows in the default configuration are:
 - a. Command Window
 - b. Current Folder
 - c. Workspace.
- You can customize the MATLAB windows that appear upon startup by clicking on ‘*layout*’ option in the toolbar and checking (or unchecking) the windows that you wish to appear on the MATLAB desktop. Figure 2.1 shows the Command Window (in the center), the Current Folder Window (on the left), the Workspace Window on the right (in some configurations this window may appear below the Current Folder window) and a Long Narrow Box containing the Path to the Current Folder (just below the Toolbar and just above the Command Window).

We will designate this Long Narrow Box as the Path Box. These windows and the Path Box are summarized as follows:

- *Command Window*: in the Command Window you can enter commands and data, make calculations and print results. You can write a script (program) in the Command Window and execute the script. However, writing a script directly into the Command Window is discouraged because it will not be saved, and if an error is made, the entire script must be retyped. By using the up arrow (↑) key on your keyboard, the previous command can be retrieved (and edited) for re-execution.
- *Path Box*: This box gives the path to the Current Folder. **To run a MATLAB script, the script needs to be in the folder associated with the path listed in this box .**
- *Current Folder Window* (on the left): This window lists all the files in the Current Folder whose path is given in the Path Box. By double clicking on a file in this window, the file will open within MATLAB.
- *Workspace Window* : This window will be on the right for a three column option (see Figure 2.1) or below the Current Folder Window for the two column option (see Figure 2.2). The two or three column option can be selected from the *layout* options in the Toolstrip. The Workspace Window contains all the commands entered into the Command Window.

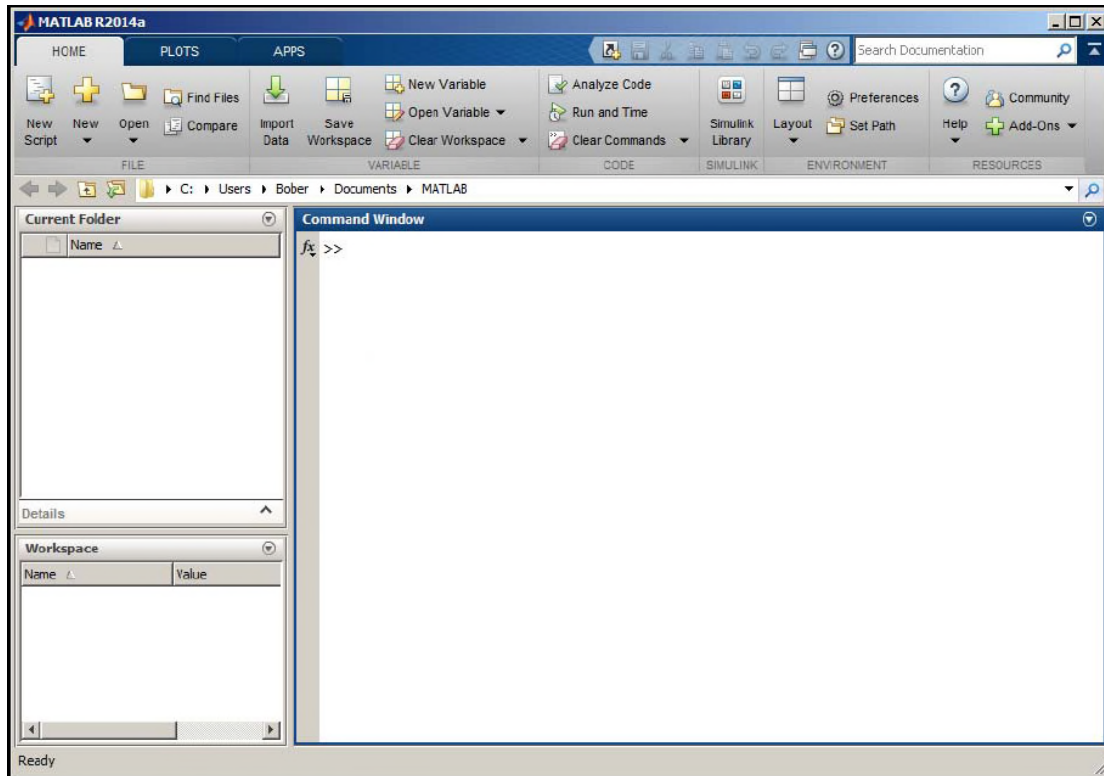


Figure 2.2. MATLAB desktop windows for two column option

2.3 Constructing a Program in MATLAB

The following list summarizes the steps for writing a script in MATLAB:

1. If available, start the MATLAB program by double-clicking on the MATLAB icon on the Window's desktop. If not available, go to the Window's *Start* menu, click on *All Programs*, find the MATLAB program among the list of available programs and double-click on it. This will open up the MATLAB desktop.
2. Click on the *New Script icon* in the Toolstrip in MATLAB's desktop. This brings up a new *Editor Window* just above the Command Window (see Figure 2.3).

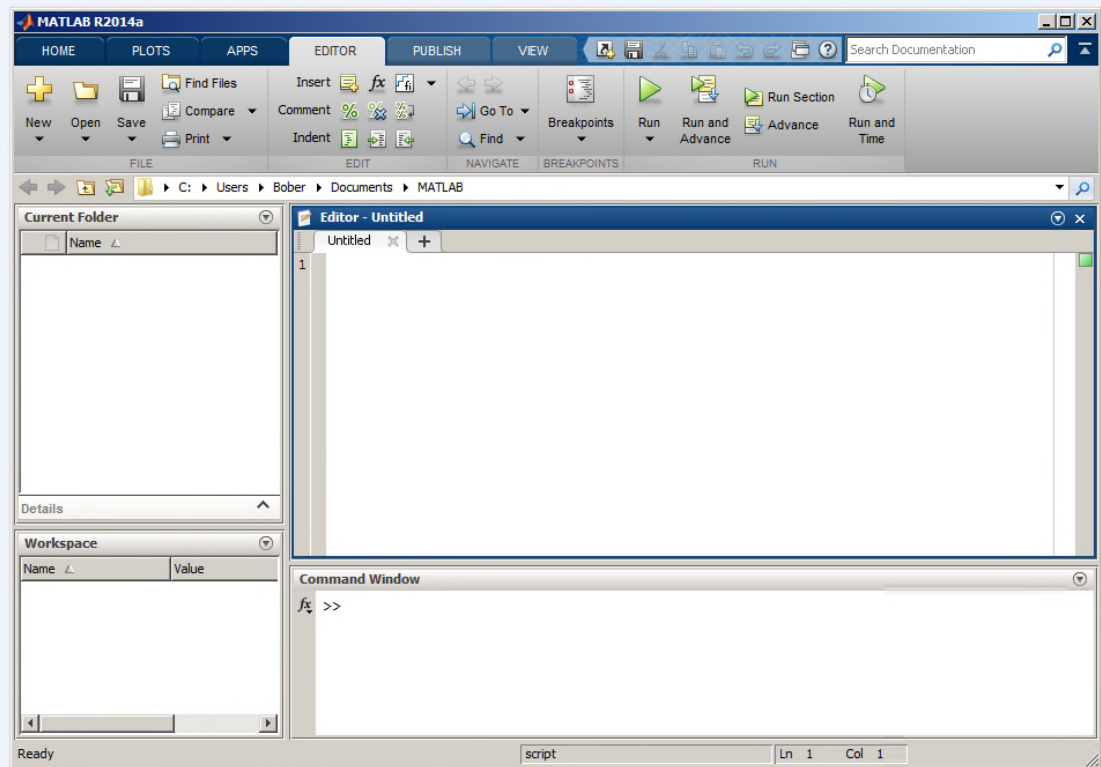


Figure 2.3. Editor Window just above the Command Window.

3. Type your program into the Editor Window.
4. When you are finished typing in the program, save the script by clicking on the *save* icon in the Toolstrip (see Figure 2.3). A dialog box will appear in which you are to select the folder (left column) and which you are to type in the name of the script (file name dialog box) that you which to save (see Figure 2.4). It is best to use a folder that contains **only** your own MATLAB scripts.

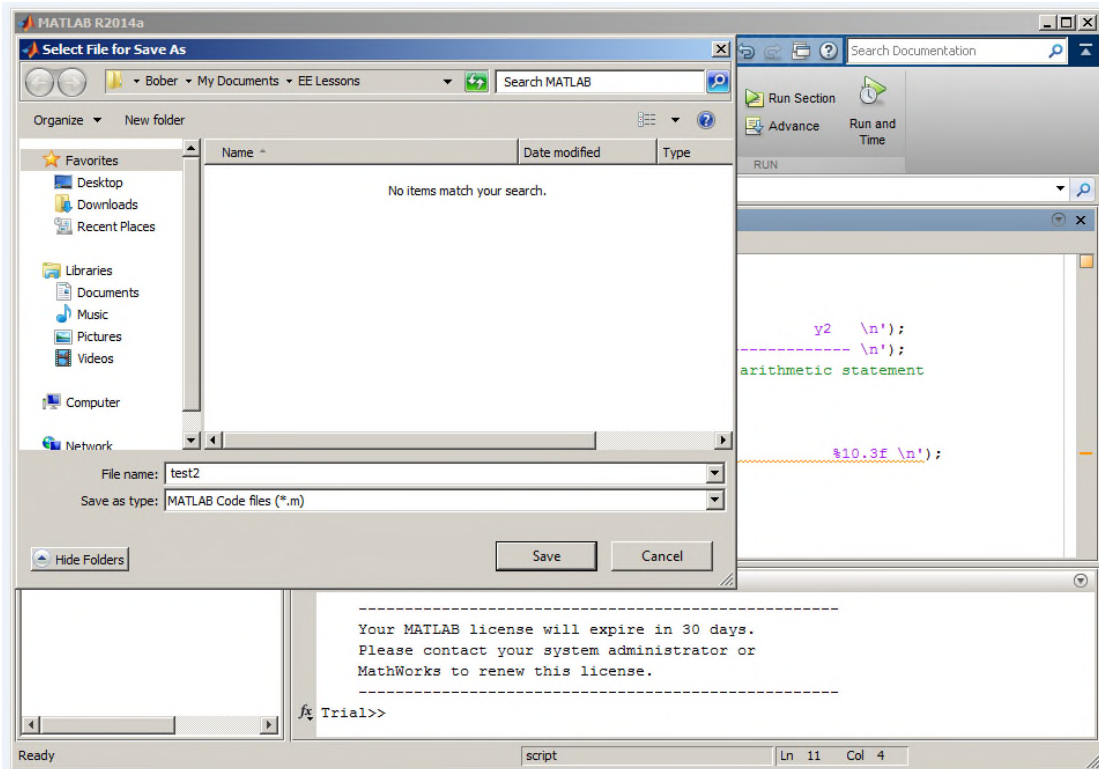


Figure 2.4 Select folder (left column) and type in the name of the script (file name dialog box)

5. You may then run the script in the script window by clicking on the *Save and Run* green arrow in the Toolstrip (see Figure 2.

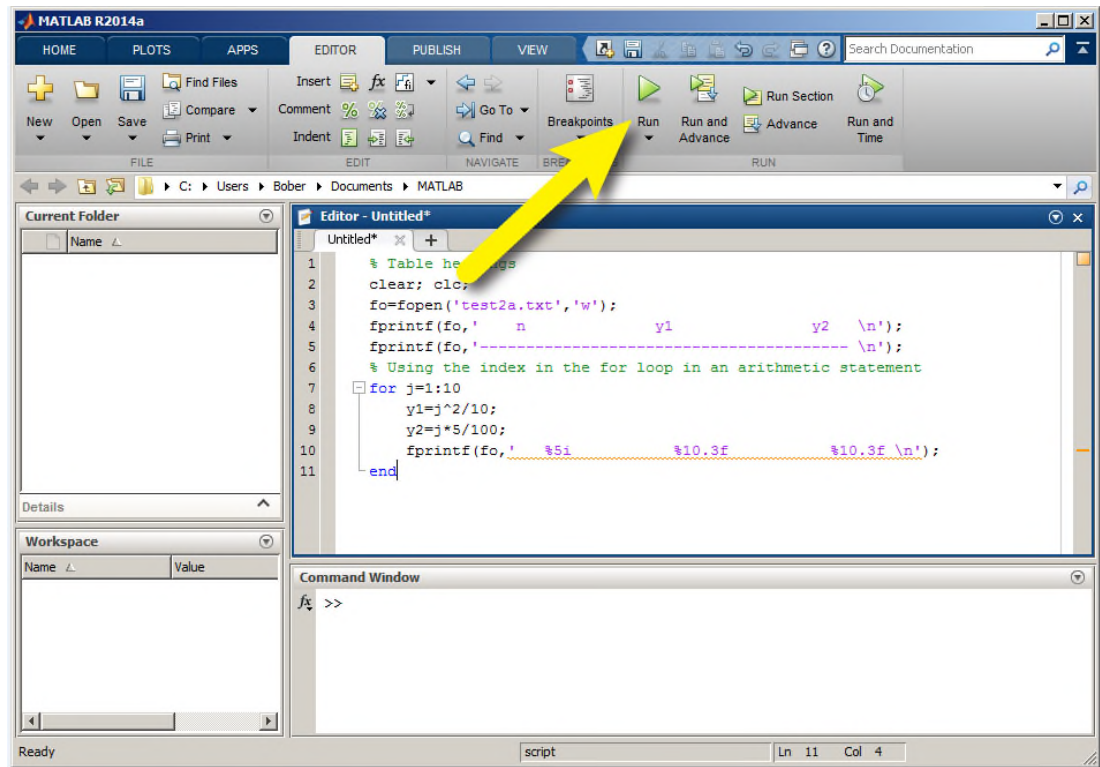


Figure 2.5. Save and run icon in the toolbar.

Alternatively, you can run the script from the command window by typing the script name (without the *.m* extension) after the MATLAB prompt (`>>`). For example, if the program has been saved as *heat.m*, then type *heat* after the MATLAB prompt (`>>`), as shown below:

```
>> heat
```

6. If you try to run your script and your script is not in the Current Folder whose path is listed in the Path Box, a dialog box will appear giving you the option of changing the MATLAB current folder or adding your folder to MATLAB's path (see Figure 2.6).

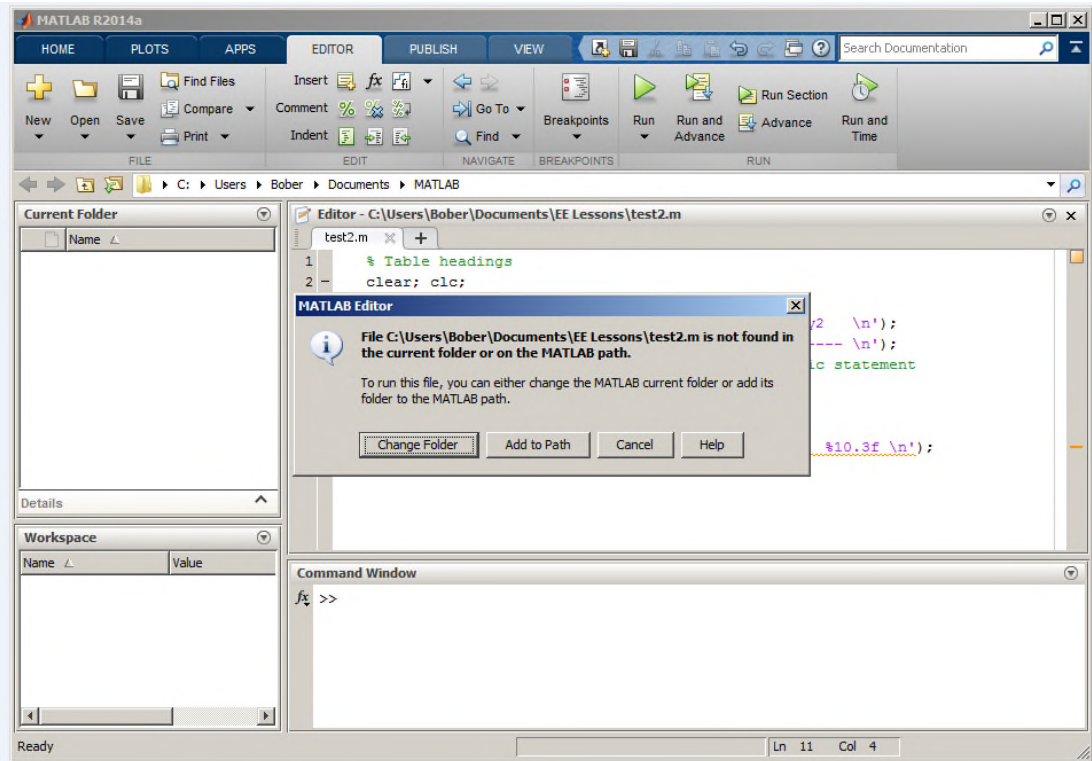


Figure 2.6 Dialog box for changing folder or path

6. If you need additional help on getting started, you can click on the *Help* icon in the Toolstrip in MATLAB's desktop. In the window that opens (see Figure 2.7) you can type in an item of interest in the search box, or you can click on the MATLAB option which brings up the window shown in Figure 2.8.

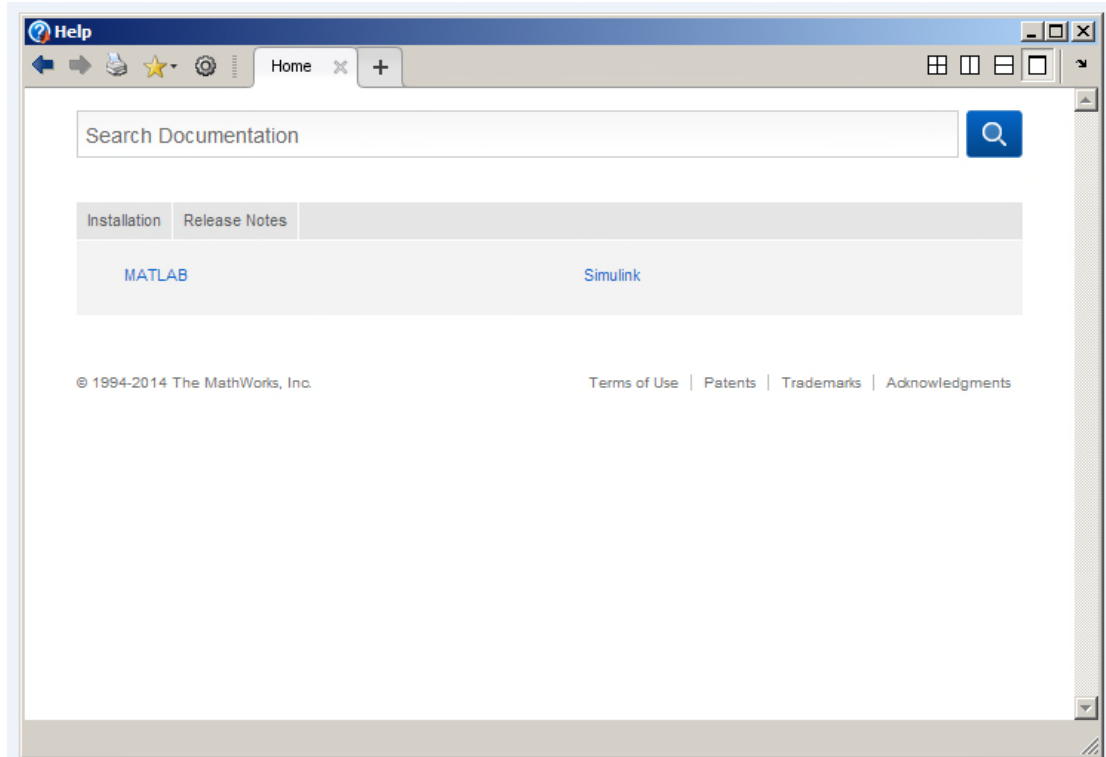


Figure 2.7. Help window.

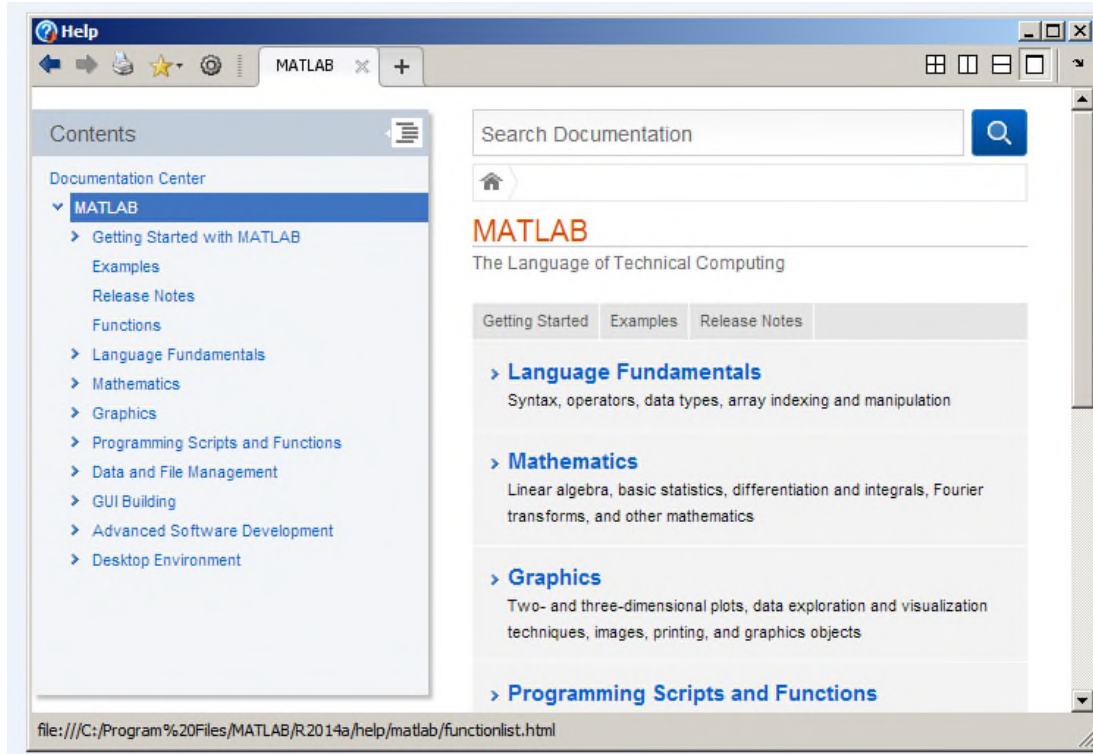


Figure 2.8 Topics in the MATLAB help window.

Review 2.1:

1. What are the two alternative ways to start the MATLAB program?
2. What are the windows in the MATLAB's default desktop?
3. It is best to write a MATLAB script (program) in the **Editor** Window. From MATLAB's default desktop how does one open the **Editor** window?
4. After you have completed writing a script in the appropriate window, what is the next step?
5. What happens if you try to run a script and the folder containing the script is not listed in the Path Box in the Command window?
6. Name two ways to execute a script.

7. In MATLAB, what is the file name extension for saved scripts?

2.4 MATLAB Fundamentals

- Variable names

- must start with a letter.
- can contain letters, digits and the underscore character.
- can be of any length, but must be unique within the first 19 characters.

Note: do not use a variable name that is the same as a file name, a MATLAB function name or a self-written function name.

- MATLAB command names and variable names are case sensitive.
- Semicolons are usually placed after variable definitions and program statements when you do not want the command echoed to the screen. In the absence of a semicolon, the defined variable appears on the screen. For example, if you entered the following assignment in the command window:

```
>> A = [3 4 7 6]
```

In the command window, you would see

```
A =
```

```
3 4 7 6
```

```
>>
```

Alternatively, if you add the semicolon after the assignment, then your command is entered but there is nothing printed to the screen, and the prompt immediately appears for you to enter your next command:

```
>> A = [3 4 7 6];
```

```
>>
```

- The percent sign (%) is used for a comment line.
- A separate Graphics Window opens to display plots and graphs.
- MATLAB's *clear*, *clc*, *clf* and *Ctl-C* commands.

`clear` removes all variables and data from the work space.

`clc` clears the Command Window.

`clf` clears the Graphics Window.

`Ctl-C` aborts a program which may be running in an infinite loop.

- Commands are case sensitive. Use lowercase letters for commands.
- The `quit` command or exit command terminates MATLAB.
- The `save` command saves variables or data in the work space of the Current Folder. The data file name will have the *.mat* extension.
- User-defined functions (also called *self-written* functions) are also saved as M-files.
- Scripts and functions are saved as ASCII text files. Thus, they may be written either in the built-in Script Window or in Notepad or in any word processor (saved as a text file).
- The basic data structure in MATLAB is a matrix.
- A matrix is surrounded by brackets and may have an arbitrary number of rows

and columns; for example, the matrix $A = \begin{bmatrix} 1 & 3 \\ 6 & 5 \end{bmatrix}$ may be entered into MATLAB

as

```
>> A = [ 1 3 <enter>
```

```
6 5 ]; <enter>
```

or

```
>> A = [ 1 3 ; 6 5 ]; <enter>
```

where the semicolon within the brackets indicates the start of a new row within the matrix.

- A matrix of 1 row and 1 column is a scalar; example:

```
>> A = [ 3.5 ];
```

Alternatively, MATLAB also accepts **A=3.5** (without brackets) as a scalar.

- A matrix consisting of 1 row and several columns, or 1 column and several rows is considered a vector; example:

```
>> A = [ 2 3 6 5 ] (row vector)
```

```
>> A = [ 2
```

```
3
```

```
6
```

```
5 ] (column vector)
```

A matrix can be defined by including a second matrix as one of the elements;

Example:

```
>> B = [ 1.5 3.1 ];
```

```
>> C = [ 4.0 B ]; (thus C = [ 4.0 1.5 3.1] )
```

- A specific element of matrix C can be selected by writing

```
>> a = C(2); (thus a = 1.5)
```

If you wish to select the last element in a vector, you can write

```
>> a = C(end); (thus a = 3.1 )
```

- The colon operator (:) may be used to:

1. Create a new matrix from an existing matrix; examples:

$$\text{if } A = \begin{bmatrix} 5 & 7 & 10 \\ 2 & 5 & 2 \\ 1 & 3 & 1 \end{bmatrix}$$

$$\text{then } x = A(:, 1) \text{ gives } x = \begin{bmatrix} 5 \\ 2 \\ 1 \end{bmatrix}$$

The colon in the expression $A(:, 1)$ implies all the rows in matrix A , and the 1 implies column 1.

$$x = A(:, 2:3) \text{ gives } x = \begin{bmatrix} 7 & 10 \\ 5 & 2 \\ 3 & 1 \end{bmatrix}$$

The first colon in the expression $A(:, 2:3)$ implies all the rows in A , and the 2:3 implies columns 2 and 3.

We can also write

$$y = A(1, :) \text{ which gives } y = [5 \ 7 \ 10]$$

The 1 implies the first row and the colon implies all the columns.

2. Colon can also be used to generate a series of numbers. The format is:

$n = \text{starting value} : \text{step size} : \text{final value}$. If the step size is omitted, the default step size is one. Example:

$$n = 1:8 \text{ gives } n = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8].$$

To increment in steps of 2 use

$$n = 1:2:7 \text{ gives } n = [1 \ 3 \ 5 \ 7]$$

These type of expressions are often used in a `for` loop, which is discussed later.