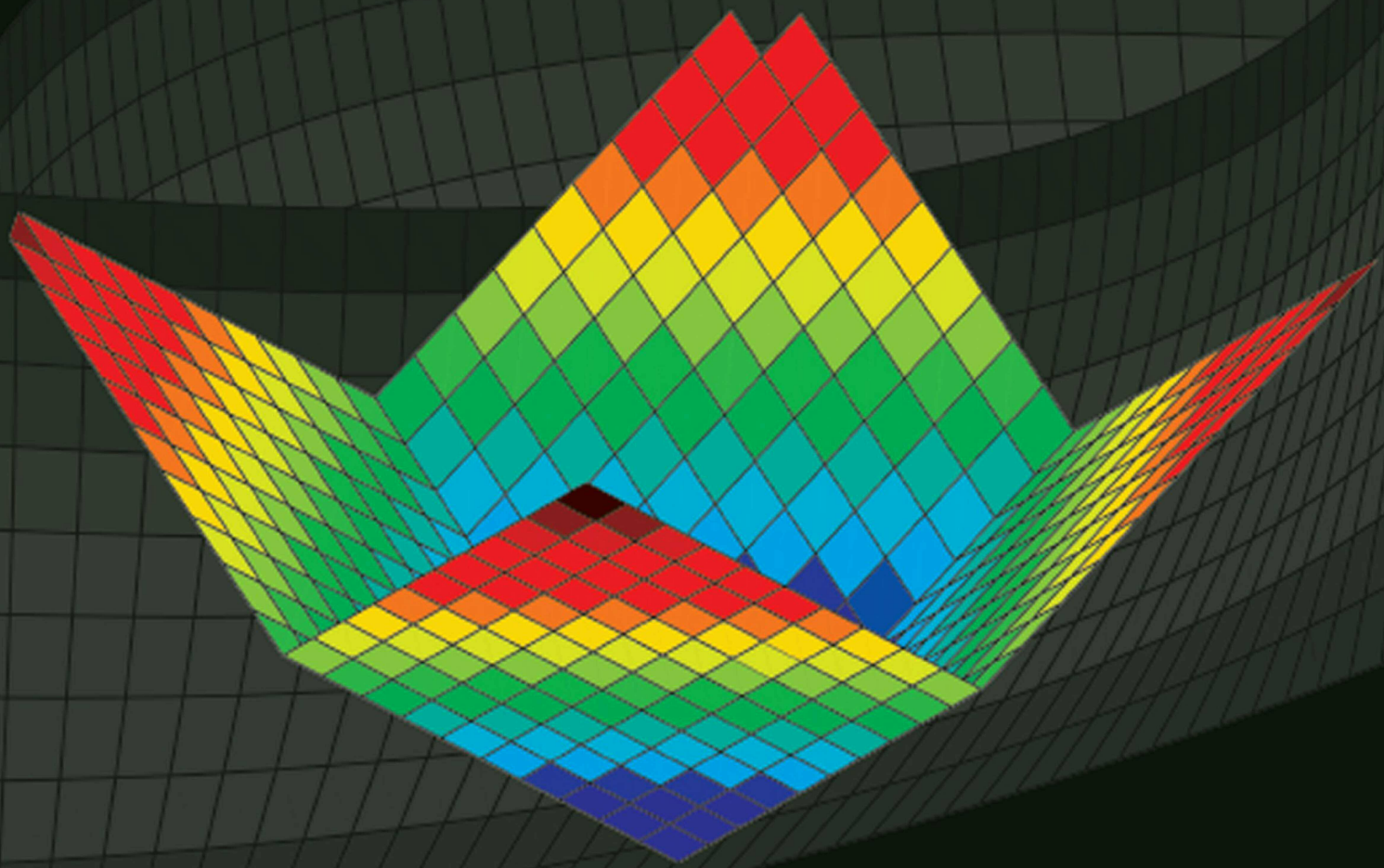
FIFTH EDITION

MATLAB®

An Introduction with Applications



AMOS GILAT

WILEY

MATLAB®

An Introduction with Applications

Fifth Edition

Amos Gilat

Department of Mechanical and Aerospace Engineering
The Ohio State University

WILEY

Chapter 1
Starting with TLAB

This chapter begins by describing the characteristics and purpose of the different windows in MATLAB. Next, the Command Window is introduced in detail. The chapter shows how to use MATLAB for arithmetic operations with scalars in much to the way that a calculator is used. This includes the use of elementary math functions with scalars. The chapter then shows how to define scalar variables (the assigmnent operator) and how to use these variables in arithmetic calcu-lations. The last section in the chapter introduces script files.

1.1 STARTING MATLAB, MATLAB WINDOWS

It is assumed that the software is installed on the computer, and that the user can start the program. Once the program starts, the MATLAB desktop window opens with the default layout, Figure 1-1. The layout has a Toolstrip at the top, the Current Folder Toolbar below it, and four windows underneath. At the top of the Toolstrip there are three tabs: HOME, PLOTS, and APPS. Clicking on the tabs changes the icons in the Toolstrip. Commonly, MATLAB is used with the HOME tab selected. The associated icons are used for executing various commands, as explained later in this chapter. The PLOTS tab can be used to

The default layout

The default layout (Figure 1-1) consists of the following four windows that are displayed under the Toolstrip: the Command Window (larger window at the center), the Curre nt Folder Window (on the left) and the Workspace and Command History windows (on the right). A list of several MATLAB windows and their purposes is given in Table 1-1.

Four of the windows—the Command Window, the Figure Window, the Editor Window, and the Help Window—are used extensively throughout the book and

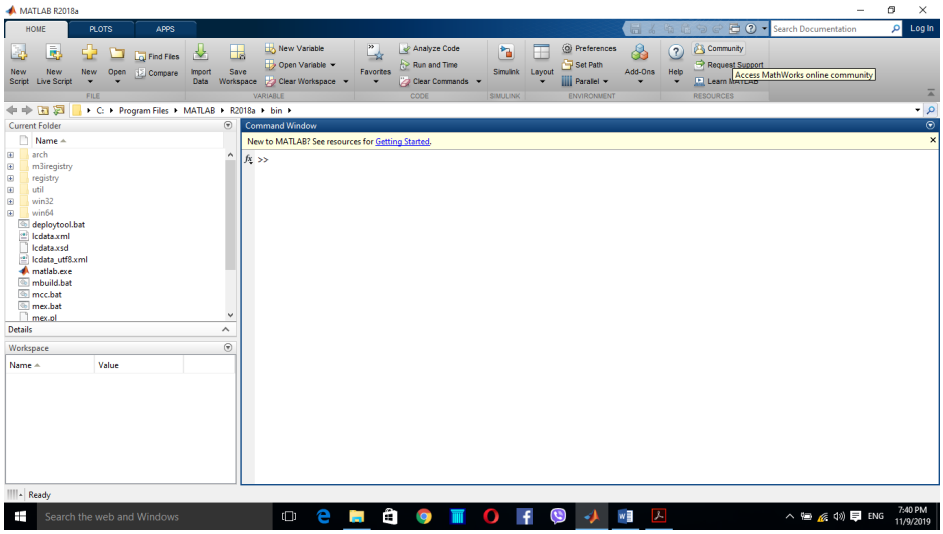


Figure 1-1: The default view of MATLAB desktop.

are briefly described on the following pages. More detailed descriptions are included in the chapters where they are used. The Command History Window, Current Folder Window, and the Workspace Window are described in Sections 1.2, 1.8.4, and 4.1, respectively.

Command Window: The Command Window is MATLAB's main window and opens when MATLAB is started. It is convenient to have the Command Window as the only visible window. This can be done either by closing all the other windows, or by selecting **Command Window Only** in the menu that opens when the **Layout** icon on the Toolstrip is selected. To close a window, click on the pull-down menu at the top right-hand side of the window and then select Close. Working in the Command Window is described in detail in Section 1.2.

Table 1-1: MATLAB windows

Window	**Purpose**
Command Window	Main window, enters variables, runs programs.
Editor Window	Creates and debugs script and function files.
Help Window	Provides help information.

Table 1-1: MATLAB windows

Window	Purpose
Workspace Window	Provides information about the variables that are stored.
Current Folder Window	Shows the files in the current folder.

Editor Window: The Editor Window is used for writing and editing programs. This window is opened by clicking on the **New Script** icon in the Toolstrip, or by clicking on the **New** icon and then selecting **Script** from the menu that opens. An example of an Editor Window is shown in Figure 1-3. More details on the Editor Window are given in Section 1.8.2, where it is used for writing script files, and in Chapter 7, where it is used to write function files.

Help Window: The Help Window contains help information. This window can be opened from the **Help** icon in the Toolstrip of the Command Window or the toolbar of any MATLAB window. The Help Window is interactive and can be used to obtain information on any feature of MATLAB. Figure 1-4 shows an open Help Window.

When MATLAB is started for the first time, the screen looks like that shown in Figure 1-1. For most beginners it is probably more convenient to close all the

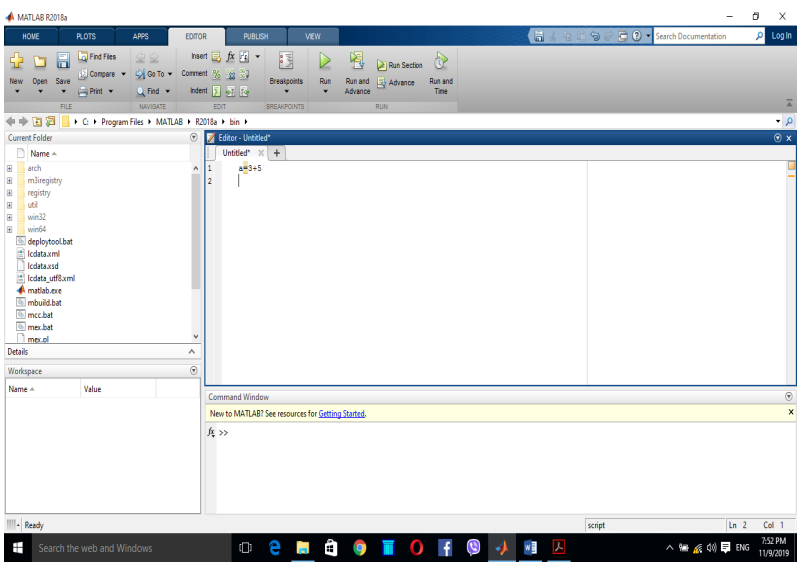


Figure 1-3: **Example of an Editor Window.**

windows except the Command Window. The closed windows can be reopened by selecting them from the **layout** icon in the Toolstrip. The windows shown in Figure 1-1 can be displayed by clicking on the **layout** icon and selecting **Default** in the menu that opens. The various windows in Figure 1-1 are docked to the desktop. A window can be undocked (become a separate, independent window) by dragging it out. An independent window can be redocked by clicking on the pull-down menu at the top right-hand side of the window and then selecting Dock.

1.2 WORKING IN THE COMMAND WINDOW

The Command Window is MATLAB's main window and can be used for executing commands, opening other windows, running programs written by the user, and managing the software. An example of the Command Window, with several simple commands that will be explained later in this chapter, is shown in Figure 1-5.

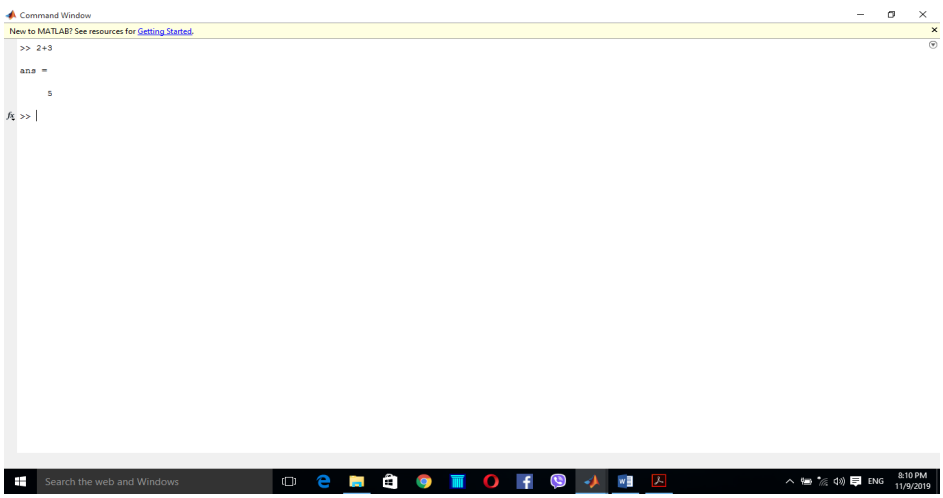


Figure 1-5: The Command Window.

Notes for working in the Command Window:

- To type a command, the cursor must be placed next to the command prompt (>>).
- Once a command is typed and the **Enter** key is pressed, the command is executed. However, only the last command is executed. Everything executed previously (that might be still displayed) is unchanged.
- Several commands can be typed in the same line. This is done by typing a comma between the commands. When the **Enter** key is pressed, the commands are executed in order from left to right.
- It is not possible to go back to a previous line that is displayed in the Command Window, make a correction, and then re-execute the command.

- A previously typed command can be recalled to the command prompt with the up-arrow key (↑). When the command is displayed at the command prompt, it can be modified if needed and then executed. The down-arrow key (↓) can be used to move down the list of previously typed commands.
- If a command is too long to fit in one line, it can be continued to the next line by typing three periods ... (called an ellipsis) and pressing the **Enter** key. The continuation of the command is then typed in the new line. The command can continue line after line up to a total of 4,096 characters.

The semicolon (;):

When a command is typed in the Command Window and the **Enter** key is pressed, the command is executed. Any output that the command generates is displayed in the Command Window. If a semicolon (;) is typed at the end of a command, the output of the command is not displayed. Typing a semicolon is useful when the result is obvious or known, or when the output is very large.

If several commands are typed in the same line, the output from any of the commands will not be displayed if a semicolon instead of a comma is typed between the commands.

Typing %:

When the symbol % (percent) is typed at the beginning of a line, the line is designated as a comment. This means that when the **Enter** key is pressed the line is not executed. The % character followed by text (comment) can also be typed after a command (in the same line). This has no effect on the execution of the command.

Usually there is no need for comments in the Command Window. Comments, however, are frequently used in a program to add descriptions or to explain the program (see Chapters 4 and 6).

The `clc` command:

The `clc` command (type `clc` and press **Enter**) clears the Command Window. After typing in the Command Window for a while, the display may become very long. Once the `clc` command is executed, a clear window is displayed. The command does not change anything that was done before. For example, if some variables were defined previously (see Section 1.6), they still exist and can be used. The up-arrow key can also be used to recall commands that were typed before.

The Command History Window:

The Command History Window lists the commands that have been entered in the Command Window. This includes commands from previous sessions. A command in the Command History Window can be used again in the Command Window. By double-clicking on the command, the command is reentered in the Command Window and executed. It is also possible to drag the command to the Command Window, make changes if needed, and then execute it. The list in the Command History Window can be cleared by selecting the lines to be deleted and

1.3.2 Using MATLAB as a Calculator

The simplest way to use MATLAB is as a calculator. This is done in the Command Window by typing a mathematical expression and pressing the **Enter** key. MATLAB calculates the expression and responds by displaying `ans =` followed by the numerical result of the expression in the next line. This is demonstrated in Tutorial 1-1.

Tutorial 1-1: Using MATLAB as a calculator.

```
>> 7+8/2
ans =
    11
>> (7+8)/2
ans =
    7.5000
>> 4+5/3+2
ans =
    7.6667
>> 5^3/2
ans =
   62.5000
>> 27^(1/3)+32^0.2
ans =
     5
>> 27^1/3+32^0.2
ans =
    11

>> 0.7854-(0.7854)^3/(1*2*3)+0.785^5/(1*2*3*4*5)...
-(0.785)^7/(1*2*3*4*5*6*7)

ans =
    0.7071
>>
```

Type and press **Enter**.

8/2 is executed first.

Type and press **Enter**.

7+8 is executed first.

5/3 is executed first.

5^3 is executed first, /2 is executed next.

1/3 is executed first, 27^(1/3) and 32^0.2 are executed next, and + is executed last.

27^1 and 32^0.2 are executed first, /3 is executed next, and + is executed last.

Type three periods ... (and press **Enter**) to continue the expression on the next line.

The last expression is the first four terms of the Taylor series for sin(π/4).

1.4 ELEMENTARY MATH BUILT-IN FUNCTIONS

In addition to basic arithmetic operations, expressions in MATLAB can include functions. MATLAB has a very large library of built-in functions. A function has a name and an argument in parentheses. For example, the function that calculates the square root of a number is `sqrt(x)`. Its name is `sqrt`, and the argument is `x`. When the function is used, the argument can be a number, a variable that has been assigned a numerical value (explained in Section 1.6), or a computable expression that can be made up of numbers and/or variables. Functions can also be included in arguments, as well as in expressions. Tutorial 1-2 shows examples of using the function `sqrt(x)` when MATLAB is used as a calculator with scalars.

Tutorial 1-2: Using the `sqrt` built-in function.

```
>> sqrt(64)                          Argument is a number.
ans =
     8
>> sqrt(50+14*3)                     Argument is an expression.
ans =
    9.5917
>> sqrt(54+9*sqrt(100))              Argument includes a function.
ans =
    12
>> (15+600/4)/sqrt(121)              Function is included in an expression.
ans =
    15
>>
```

Some commonly used elementary MATLAB mathematical built-in functions are given in Tables 1-3 through 1-5. A complete list of functions organized by category can be found in the Help Window.

Table 1-3: Elementary math functions

Function	Description	Example

`sqrt(x)`	Square root.	`>> sqrt(81)`
		`ans =`
		`9`

`exp(x)`	Exponential (e^x).	`>> exp(5)`
		`ans =`
		`148.4132`

Table 1-3: Elementary math functions (Continued)

Function	Description	Example

`abs(x)`	Absolute value.	`>> abs(-24)`
		`ans =`
		`24`

`log(x)`	Natural logarithm.
	Base *e* logarithm (ln).
		`>> log(1000)`
		`ans =`
		`6.9078`

`log10(x)`	Base 10 logarithm.	`>> log10(1000)`
		`ans =`
		`3.0000`

`factorial(x)`	The factorial function *x*!
	(*x* must be a positive integer.)
		`>> factorial(5)`
		`ans =`
		`120`

Table 1-4: Trigonometric math functions

Function	Description	Example

`sin(x)`
`sind(x)`
	Sine of angle *x* (*x* in radians).
	Sine of angle *x* (*x* in degrees).
		`>> sin(pi/6)`
		`ans =`
		`0.5000`

`cos(x)`
`cosd(x)`
	Cosine of angle *x* (*x* in radians).
	Cosine of angle *x* (*x* in degrees).
		`>> cosd(30)`
		`ans =`
		`0.8660`

`tan(x)`
`tand(x)`
	Tangent of angle *x* (*x* in radians).
	Tangent of angle *x* (*x* in degrees).
		`>> tan(pi/6)`
		`ans =`
		`0.5774`

`cot(x)`
`cotd(x)`
	Cotangent of angle *x* (*x* in radians).
	Cotangent of angle *x* (*x* in degrees).
		`>> cotd(30)`
		`ans =`
		`1.7321`

The inverse trigonometric functions are `asin(x)`, `acos(x)`, `atan(x)`, `acot(x)` for the angle in radians; and `asind(x)`, `acosd(x)`, `atand(x)`, `acotd(x)` for the angle in degrees. The hyperbolic trigonometric functions are `sinh(x)`, `cosh(x)`, `tanh(x)`, and `coth(x)`. Table 1-4 uses `pi`, which is equal to π (see Section 1.6.3).

Table 1-5: Rounding functions

Function	Description	Example

`round(x)`	Round to the nearest integer.	`>> round(17/5)`
		`ans =`
		`3`

`fix(x)`	Round toward zero.	`>> fix(13/5)`
		`ans =`
		`2`

Table 1-5: Rounding functions (Continued)

Function	Description	Example

`ceil(x)`	Round toward postive infinity.	`>> ceil(11/5)`
		`ans =`
		`3`

`floor(x)`	Round toward minus infinity.	`>> floor(-9/4)`
		`ans =`
		`-3`

`rem(x,y)`	Returns the remainder after x is divided by y.	`>> rem(13,5)`
		`ans =`
		`3`

`sign(x)`	Signum function. Returns 1 if $x > 0$, -1 if $x < 0$, and 0 $x = 0$.	`>> sign(5)`
		`ans =`
		`1`

1.6 DEFINING SCALAR VARIABLES

A variable is a name made of a letter or a combination of several letters (and digits) that is assigned a numerical value. Once a variable is assigned a numerical value, it can be used in mathematical expressions, in functions, and in any MATLAB statements and commands. A variable is actually a name of a memory location. When a new variable is defined, MATLAB allocates an appropriate memory space where the variable's assignment is stored. When the variable is used the stored data is used. If the variable is assigned a new value the content of the memory location is replaced. (In Chapter 1 we consider only variables that are assigned numerical values that are scalars. Assigning and addressing variables that are arrays is discussed in Chapter 2.)

1.6.1 The Assignment Operator

In MATLAB the = sign is called the assignment operator. The assignment operator assigns a value to a variable.

```
Variable_name = A numerical value, or a computable expression
```

- The left-hand side of the assignment operator can include only one variable name. The right-hand side can be a number, or a computable expression that can include numbers and/or variables that were previously assigned numerical values. When the **Enter** key is pressed the numerical value of the right-hand side is assigned to the variable, and MATLAB displays the variable and its assigned value in the next two lines.

  The following shows how the assignment operator works.

```
>> x=15
x =
     15

>> x=3*x-12
x =
     33
>>
```

The number 15 is assigned to the variable x.

MATLAB displays the variable name and its assigned value.

A new value is assigned to x. The new value is 3 times the previous value of x minus 12.

The last statement ($x = 3x - 12$) illustrates the difference between the assignment operator and the equal sign. If in this statement the = sign meant equal, the value of x would be 6 (solving the equation for x).

The use of previously defined variables to define a new variable is demonstrated next.

```
>> a=12
a =
     12
>> B=4
B =
      4
>> C=(a-B)+40-a/B*10
C =
     18
```

Assign 12 to a.

Assign 4 to B.

Assign the value of the expression on the right-hand side to the variable C.

- If a semicolon is typed at the end of the command, then when the **Enter** key is pressed, MATLAB does not display the variable with its assigned value (the variable still exists and is stored in memory).
- If a variable already exists, typing the variable's name and pressing the **Enter** key will display the variable and its value in the next two lines.

As an example, the last demonstration is repeated below using semicolons.

```
>> a=12;
>> B=4;
>> C=(a-B)+40-a/B*10;
>> C
C =
     18
```

The variables a, B, and C are defined but are not displayed, since a semicolon is typed at the end of each statement.

The value of the variable C is displayed by typing the name of the variable.

- Several assignments can be typed in the same line. The assignments must be separated with a comma (spaces can be added after the comma). When the **Enter** key is pressed, the assignments are executed from left to right and the variables and

their assignments are displayed. A variable is not displayed if a semicolon is typed instead of a comma. For example, the assignments of the variables `a`, `B`, and `C` above can all be done in the same line.

```
>> a=12, B=4; C=(a-B)+40-a/B*10
a =
    12
C =
    18
```

The variable `B` is not displayed because a semicolon is typed at the end of the assignment.

- A variable that already exists can be reassigned a new value. For example:

```
>> ABB=72;
>> ABB=9;
>> ABB
ABB =
     9
>>
```

A value of 72 is assigned to the variable `ABB`.

A new value of 9 is assigned to the variable `ABB`.

The current value of the variable is displayed when the name of the variable is typed and the **Enter** key is pressed.

- Once a variable is defined it can be used as an argument in functions. For example:

```
>> x=0.75;
>> E=sin(x)^2+cos(x)^2
E =
     1
>>
```

1.6.2 Rules About Variable Names

A variable can be named according to the following rules:

- Must begin with a letter.
- Can be up to 63 characters long.
- Can contain letters, digits, and the underscore character.
- Cannot contain punctuation characters (e.g., period, comma, semicolon).
- MATLAB is case-sensitive: it distinguishes between uppercase and lowercase letters. For example, `AA`, `Aa`, `aA`, and `aa` are the names of four different variables.
- No spaces are allowed between characters (use the underscore where a space is desired).
- Avoid using the name of a built-in function for a variable (i.e., avoid using `cos`, `sin`, `exp`, `sqrt`, etc.). Once a function name is used to for a variable name, the function cannot be used.

1.6.3 Predefined Variables and Keywords

There are 20 words, called keywords, that are reserved by MATLAB for various purposes and cannot be used as variable names. These words are:

```
break   case   catch  classdef  continue   else   elseif
end   for   function   global   if   otherwise   parfor
persistent   return   spmd   switch   try   while
```

When typed, these words appear in blue. An error message is displayed if the user tries to use a keyword as a variable name. (The keywords can be displayed by typing the command `iskeyword`.)

A number of frequently used variables are already defined when MATLAB is started. Some of the predefined variables are:

`ans` A variable that has the value of the last expression that was not assigned to a specific variable (see Tutorial 1-1). If the user does not assign the value of an expression to a variable, MATLAB automatically stores the result in `ans`.

`pi` The number π.

`eps` The smallest difference between two numbers. Equal to 2^(–52), which is approximately 2.2204e–016.

`inf` Used for infinity.

`i` Defined as $\sqrt{-1}$, which is: 0 + 1.0000i.

`j` Same as i.

`NaN` Stands for Not-a-Number. Used when MATLAB cannot determine a valid numeric value. Example: 0/0.

The predefined variables can be redefined to have any other value. The variables `pi`, `eps`, and `inf`, are usually not redefined since they are frequently used in many applications. Other predefined variables, such as `i` and `j`, are sometime redefined (commonly in association with loops) when complex numbers are not involved in the application.

1.7 USEFUL COMMANDS FOR MANAGING VARIABLES

The following are commands that can be used to eliminate variables or to obtain information about variables that have been created. When these commands are typed in the Command Window and the **Enter** key is pressed, either they provide information, or they perform a task as specified below.

Command	Outcome
`clear`	Removes all variables from the memory.

Command	Outcome
`clear x y z`	Removes only variables `x`, `y`, and `z` from the memory.
`who`	Displays a list of the variables currently in the memory.
`whos`	Displays a list of the variables currently in the memory and their sizes together with information about their bytes and class (see Section 4.1).