Chapter 1

1.1 Introduction

The name MATLAB stands for MATrix LABoratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming.

MATLAB is an interactive system whose basic data element is an *array* that does not require dimensioning.

It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

1.2 A minimum MATLAB session

1.2.1 Starting MATLAB

After logging into your account, you can enter MATLAB by double-clicking on the MATLAB shortcut icon on your Windows desktop. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains other windows. The major tools within or accessible from the desktop are:

- The Command Window
- The Command History
- The Workspace
- The Current Directory
- The Help Browser
- The Start button

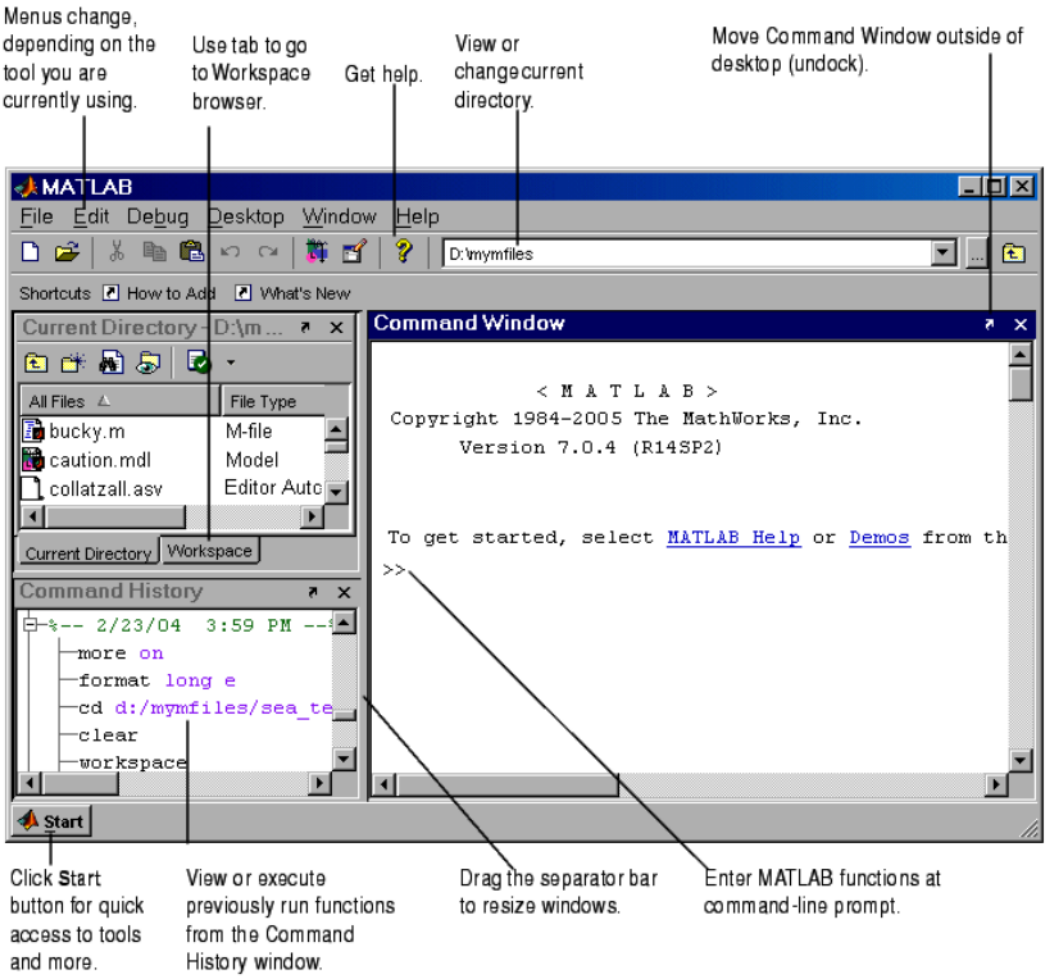

Figure 1.1: The graphical interface to the MATLAB workspace

The Figure 1.1 shows the default configuration of the MATLAB desktop. You can customize the arrangement of tools and documents to suit your needs.

The MATLAB desktop contains the prompt (>>) in the Command Window.

1.2.2 Using MATLAB as a calculator

As an example of a simple calculation, just type the expression: 1 + 2 × 3.

type it at the prompt command (>>) as follows,

*>> 1+2*3*

ans =

7

- If we do not specify an output variable, MATLAB uses a default variable *ans*, short for answer, to store the results of the current calculation. To avoid this, you may assign a value to a variable or output argument name. For example,

*>> x = 1+2*3*

x =

7

This variable name can always be used to refer to the results of the previous computations. Therefore, computing 4x will result in

*>> 4*x*

ans =

28.0000

Before we conclude this minimum session, Table 1.1 gives the partial list of arithmetic operators.

SYMBOL	OPERATION	EXAMPLE
+	Addition	$2+3$
−	Subtraction	$2-3$
*	Multiplication	$2*3$
/	Division	$2/3$

Table 1.1: Basic arithmetic operators

1.2.3 Quitting MATLAB

To end a MATLAB session, we type quit in the Command Window, or select File −→ Exit MATLAB in the desktop main menu.

1.3 Getting started

1.3.1 Creating MATLAB variables

MATLAB variables are created with an assignment statement. The syntax of variable assignment is *variable name = a value (or an expression)*

For example,

>> x = expression

Where expression is a combination of numerical values, mathematical operators, variables, and function calls.

1.3.2 Overwriting variable

Once a variable has been created, it can be reassigned. In addition, if you do not wish to see the intermediate results, you can suppress the numerical output by putting a semicolon *(;)* at the end of the line.

>> t = 5;

>> t = t+1

t =

6

1.3.3 Error messages

If we enter an expression incorrectly, MATLAB will return an error message. For example, in the following, we left out the multiplication sign, *, in the following expression

>> x = 10;

>> 5x

??? 5x

|

Error: Unexpected MATLAB expression.

1.3.4 Making corrections

To make corrections, we can, retype the expressions. But if the expression is lengthy, we make more mistakes by typing a second time. A previously typed command can be recalled with the up-arrow key ↑. When the command is displayed at the command prompt, it can be modified if needed and executed.

1.3.5 Controlling the hierarchy of operations or precedence

Let's consider the previous arithmetic operation, but now including parentheses.

For example, 1 + 2 × 3 will become (1 + 2) × 3

*>> (1+2)*3*

ans =

9

and, from previous example

6

*>> 1+2*3*

ans =

7

Parentheses can always be used to overrule priority, and their use is recommended in some complex expressions to avoid ambiguity.

Now, consider the example:

$$\frac{1}{2+3^2}+\frac{4}{5}\times\frac{6}{7}$$

In MATLAB, it becomes

*>> 1/(2+3^2)+4/5*6/7*

ans =

0.7766

or, if parentheses are missing,

*>> 1/2+3^2+4/5*6/7*

ans =

10.1857

1.3.6 Controlling the appearance of floating point number

MATLAB by default displays only 4 decimals in the result of the calculations, for example

−163.6667, as shown in above examples. However, MATLAB does numerical calculations in double precision, which is 15 digits. The command format controls how the results of computations are displayed. Here are some examples of the different formats together with the resulting outputs.

>> format short

>> x=-163.6667

If we want to see all 15 digits, we use the command *format long*

>> format long

>> x= -1.636666666666667e+002

To return to the standard format, enter *format short*, or simply *format*.

There are several other formats in the MATLAB documentation, (or in the help format).

1.3.7 Managing the workspace

The contents of the workspace persist between the executions of separate commands. Therefore, it is possible for the results of one problem to have an effect on the next one. To avoid this possibility, we use a *clear* command at the start of each new independent calculation.

>> clear

The command *clear* or *clear all* removes all variables from the workspace. This frees up system memory. In order to display a list of the variables currently in the memory, type

>> who

While, *whos* will give more details which include *size, space allocation*, and *class* of the variables.

1.3.8 Keeping track of work session

It is possible to keep track of everything done during a MATLAB session with the *diary* command.

>> diary

or give a name to a created file,

>> diary FileName

The function *diary* is useful if you want to save a complete MATLAB session. They save all input and output as they appear in the MATLAB window. When you want to stop the recording, enter *diary off*. If you want to start recording again, enter *diary on*.

1.3.9 Entering multiple statements per line

It is possible to enter multiple statements per line. Use commas *(,)* or semicolons *(;)* to enter more than one statement at once. Commas *(,)* allow multiple statements per line without suppressing output.

>> a=7; b=cos(a), c=cosh(a)

b =

0.6570

c =

548.3170

1.3.11 Getting help

To view the online documentation, select MATLAB Help from Help menu or MATLAB Help directly in the Command Window. On the other hand, information about any command is available by typing

>> help Command

Another way to get help is to use the *lookfor* command, which differs from the *help* command. 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.

For example, looking for a function to take the inverse of a matrix. The command *help inverse* will produce nothing. the command *lookfor inverse* will produce detailed information, which includes the function of interest, inv.

>> lookfor inverse

examples:

• Use help to request info on a *specific function*

>> help sqrt

• Use *lookfor* to find functions by *keywords*. The general form is

>> lookfor FunctionName

1.4. Mathematical functions

MATLAB offers many predefined mathematical functions for technical computing which contains a large set of mathematical functions.

Typing *help elfun* and *help specfun* calls up full lists of elementary and special functions respectively.

There is a long list of mathematical functions that are built into MATLAB. These

functions are called built-ins.

Table 1.2 lists some commonly used functions, where variables x and y can be numbers, vectors, or matrices.

`cos(x)`	Cosine	`abs(x)`	Absolute value
`sin(x)`	Sine	`sign(x)`	Signum function
`tan(x)`	Tangent	`max(x)`	Maximum value
`acos(x)`	Arc cosine	`min(x)`	Minimum value
`asin(x)`	Arc sine	`ceil(x)`	Round towards $+\infty$
`atan(x)`	Arc tangent	`floor(x)`	Round towards $-\infty$
`exp(x)`	Exponential	`round(x)`	Round to nearest integer
`sqrt(x)`	Square root	`rem(x)`	Remainder after division
`log(x)`	Natural logarithm	`angle(x)`	Phase angle
`log10(x)`	Common logarithm	`conj(x)`	Complex conjugate

Table 1.2: Elementary functions

In addition to the elementary functions, MATLAB includes a number of predefined constant values. A list of the most common values is given in Table 1.3.

`pi`	The π number, $\pi = 3.14159\ldots$
`i,j`	The imaginary unit i, $\sqrt{-1}$
`Inf`	The infinity, ∞
`NaN`	Not a number

Table 1.3: Predefined constant values

1.4.1 Examples

We illustrate here some typical examples which related to the elementary functions previously defined. As a first example, the value of the expression $y = e^{-a}\sin(x) + 10\sqrt{y}$, for a = 5, x = 2, and y = 8 is computed by

```
>> a = 5; x = 2; y = 8;
>> y = exp(-a)*sin(x)+10*sqrt(y)
y =
28.2904
```

- The subsequent examples are

```
>> log(142)
ans =
4.9558
>> log10(142)

ans =

2.1523
```

Note the difference between the natural logarithm *log(x)* and the decimal logarithm (base 10) *log10(x).*

- To calculate sin(π/4) and e10,

```
>> sin(pi/4)
ans =
0.7071

>> exp(10)
ans =
2.2026e+004
```

Notes:

• There are some exceptions. For example, *i* and *j* are pre-assigned to √−1. However, one or both of *i* or *j* are often used as loop indices.