Lab-4

MATLAB

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. These factors make MATLAB an excellent tool for teaching and research.

1.2 Starting MATLAB

After logging into your account, you can enter MATLAB by double-clicking on the MATLAB shortcut icon (MATLAB 7.0.4) 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

When MATLAB is started for the First time, the screen looks like the one that shown in the Figure 1.1. This illustration also shows the default configuration of the MATLAB desktop. You can customize the arrangement of tools and documents to suit your needs.

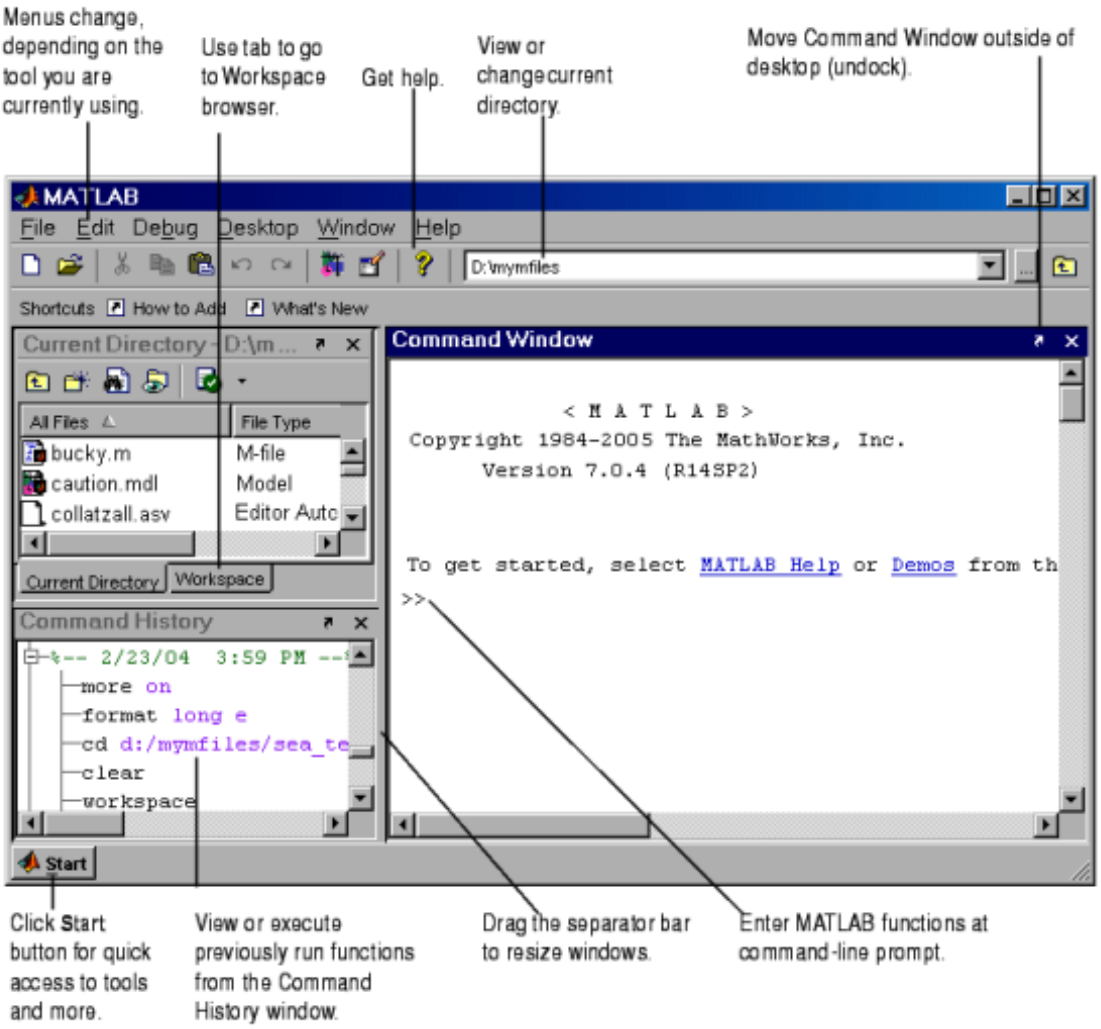

Figure 1-1: The graphical interface to the MATLAB workspace

1.3 Using MATLAB as a calculator

As an example of a simple interactive calculation, just type the expression you want to evaluate. Let's start at the very beginning. For example, let's suppose you want to calculate the expression, 1 + 2 x 3. You type it at the prompt command (>>) as follows,

```
>> x = 1+2*3
x =
7
```

You will have noticed that if you do not specify an output variable, MATLAB uses a default variable ans, short for answer, to store the results of the current calculation. Note that the variable ans is created (or overwritten, if it is already existed). To avoid this, you may assign a value to a variable or output argument name. For example,

```
>> x = 1+2*3
x =
7
```

1.4 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. Then the sequence of commands looks like this:

```
>> t = 5;
>> t = t+1
t =
6
```

1.5 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.

1.6 Getting help

To view the online documentation, select MATLAB Help from Help menu or MATLAB Help directly in the Command Window. The preferred method is to use the Help Browser. The Help Browser can be started by selecting the? icon from the desktop toolbar. On the other hand, information about any command is available by typing

```
>> help Command
```

1.7 Mathematical functions

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

Table 1-1: Elementary functions

cos(x)	**Cosine**	**sqrt(x)**	**Square root**
sin(x)	**Sine**	**max(x)**	**Maximum value**
tan(x)	**Tangent**	**min(x)**	**Minimum value**
abs(x)	**Absolute value**	**log(x)**	**Natural logarithm**

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 1-2.

Table 1-2: Predefined constant values

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

1.7.1 Examples

$$y = e^{-a}Sin(x) + 10\sqrt{y}\ for\ a = 5, x = 2\ and\ y = 8$$

```
>> 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
```

1.8 Basic plotting

```
>> x = [1 2 3 4 5 6];
>> y = [3 -1 2 4 5 1];
>> plot(x,y)
```

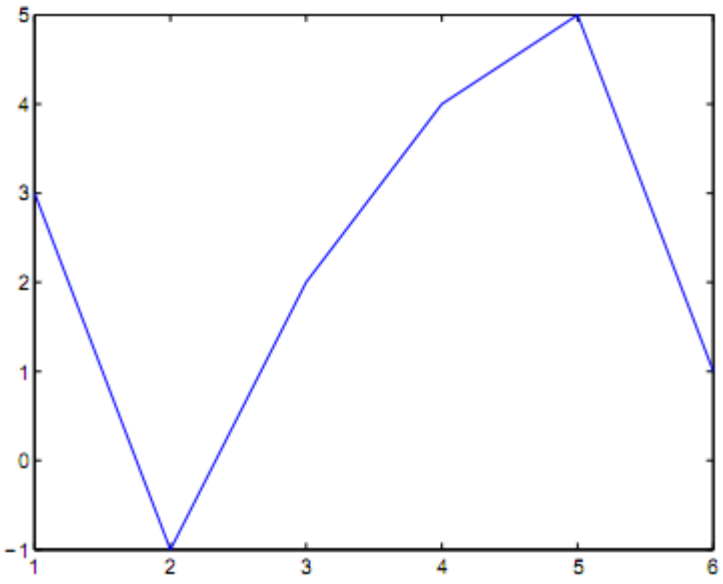

Figure 1-2: Plot for the vectors x and y

```
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> plot(x,y)
```

1.9 Adding titles, axis labels, and annotations

MATLAB enables you to add axis labels and titles. For example, using the graph from the previous example, add an x- and y-axis labels.

Now label the axes and add a title. The character \pi creates the symbol π. An example of 2D plot is shown in Figure 2.2.

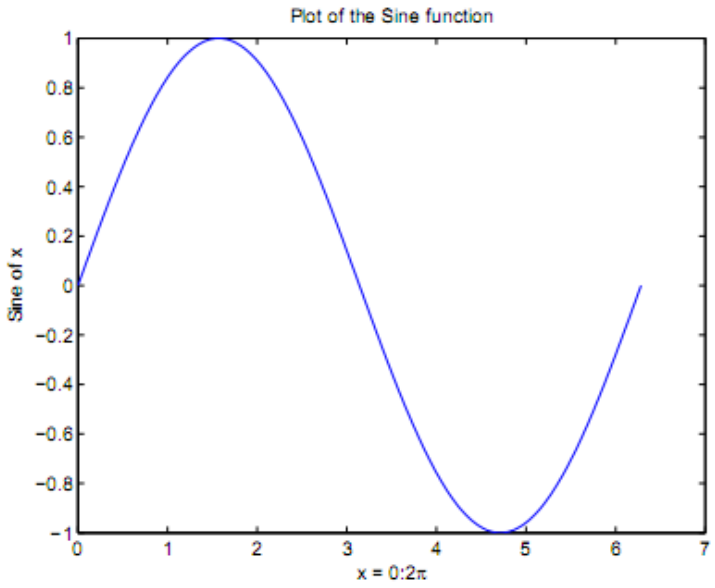

Figure 1-3: Plot of the Sine function

```
>> xlabel('x = 0:2\pi')
>> ylabel('Sine of x')
>> title('Plot of the Sine function')
```

The color of a single curve is, by default, blue, but other colors are possible. The desired color is indicated by a third argument. For example, red is selected by plot(x,y,'r'). Note the single quotes, ' ', around r.

1.10 Multiple data sets in one plot

Multiple (x; y) pairs arguments create multiple graphs with a single call to plot. For example, these statements plot three related functions of x: $y_1 = 2\cos(x)$, $y_2 = \cos(x)$, and $y_3 = 0.5 * \cos(x)$, in the interval $0 \leq x \leq 2\pi$.

```
>> x = 0:pi/100:2*pi;
>> y1 = 2*cos(x);
>> y2 = cos(x);
>> y3 = 0.5*cos(x);
>> plot (x,y1,'--',x,y2,'-',x,y3,':')
>> xlabel ('0 \leq x \leq 2\pi')
>> ylabel ('Cosine functions')
>> legend ('2*cos(x)','cos(x)','0.5*cos(x)')
>> title('Typical example of multiple plots')
>> axis([0 2*pi -3 3])
```

1.11 Entering a vector

```
>> v = [1 4 7 10 13]
v = 1 4 7 10 13
>> w = [1;4;7;10;13]
w =
1
4
7
10
13
>> w = v'
w =
1
4
7
10
```

Thus, v(1) is the first element of vector v, v(2) its second element, and so forth. Furthermore, to access blocks of elements, we use MATLAB's colon notation (:). For example, to access the first three elements of v, we write

```
>> v(1:3)
ans =
1 4 7
```

Or, all elements from the third through the last elements,

```
>> v(3,end)
ans =
7 10 13
```

where end signiFies the last element in the vector. If v is a vector, writing

```
>> v(:)
```

produces a column vector, whereas writing

```
>> v(1:end)
```

produces a row vector.

1.12 Entering a matrix

A matrix is an array of numbers. To type a matrix into MATLAB you must

- begin with a square bracket, [
- separate elements in a row with spaces or commas (,)
- use a semicolon (;) to separate rows
- end the matrix with another square bracket,].

Here is a typical example. To enter a matrix A, such as,

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

type,

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

Note that the use of semicolons (;) here is different from their use mentioned earlier to suppress output or to write multiple commands in a single line.

Once we have entered the matrix, it is automatically stored and remembered in the Workspace. We can refer to it simply as matrix A. We can then view a particular element in a matrix by specifying its location. We write,

```
>> A(2,1)
ans =
4
```

A(2,1) is an element located in the second row and first column. Its value is 4.

1.13 Matrix indexing

We select elements in a matrix just as we did for vectors, but now we need two indices. The element of row i and column j of the matrix A is denoted by A(i,j). Thus, A(i,j) in MATLAB refers to the element A_{ij} of matrix A. The first index is the row number and the second index is the column number. For example, A(1,3) is an element of first row and third column. Here, A(1,3)=3.

Correcting any entry is easy through indexing. Here we substitute A(3,3)=9 by A(3,3)=0. The result is

```
>> A(3,3) = 0
A =
 1 2 3
 4 5 6
 7 8 0
```

1.14 Dimension

To determine the dimensions of a matrix or vector, use the command size. For example,

```
>> size(A)
 ans =
 3 3
```

1.15 Solving linear equations

A^{-1}= inv(A)