

1.

Introduction to MATLAB

MATLAB is an interactive, numerical computation program. It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. In some installations MATLAB will also have a Symbolic Toolbox which allows MATLAB to perform symbolic calculations as well as numerical calculations. In this chapter we will describe how MATLAB handles simple numerical expressions and mathematical formulas.

MATLAB is available on almost every computer system. Its interface is similar regardless of the system being used. We will assume that you have sufficient understanding of your computer to start up MATLAB and that you are now faced with a window on your computer which contains the MATLAB prompt¹, `>>`, and a cursor waiting for you to do something. This is called the MATLAB Command Window, and it is time to begin.

Numerical Expressions

In its most elementary use, MATLAB is an extremely powerful calculator, with many built-in functions, and a very large and easily accessible memory. Let's start at the very beginning. Suppose you want to calculate a number such as $12.3(48.5 + \frac{342}{39})$. You can accomplish this using MATLAB by entering `12.3*(48.5+342/39)`. Try it. You should get the following:

```
>> 12.3*(48.5+342/39)
ans =
    704.4115
```

Notice that what you enter into MATLAB does not differ greatly from what you would write on a piece of paper. The only changes from the algebra that you use every day are the different symbols used for the algebraic operations. These are standard in the computer world, and are made necessary by the unavailability of the standard symbols on a keyboard. Here is a partial list of symbols used in MATLAB.

+	addition	-	subtraction
*	multiplication	^	exponentiation
/	right division	\	left division

While `+` and `-` have their standard meanings, `*` is used to indicate multiplication. You will notice that division can be indicated in two ways. The fraction $\frac{2}{3}$ can be indicated in MATLAB as either `2/3` or as `3\2`. These are referred to as right division and left division, respectively.

```
>> 2/3
ans =
    0.6667
>> 3\2
ans =
    0.6667
```

¹ In the narrative that follows, readers are expected to enter the text that appears after the command prompt (`>>`). You must press the **Enter** or **Return** key to execute the command.

Exponentiation is quite different in MATLAB; it has to be, since MATLAB has no way of entering superscripts. Consequently, the power 4^3 must be entered as 4^3 .

```
>> 4^3
ans =
    64
```

The order in which MATLAB performs arithmetic operations is exactly that taught in high school algebra courses. Exponentiations are done first, followed by multiplications and divisions, and finally by additions and subtractions. The standard order of precedence of arithmetic operations can be changed by inserting parentheses. For example, the result of $12.3*(48.5+342)/39$ is quite different than the similar expression we computed earlier, as you will discover if you try it.

MATLAB allows the assignment of numerical values to variable names. For example, if you enter

```
>> x = 3
x =
    3
```

then MATLAB will remember that x stands for 3 in subsequent computations. Therefore, computing $2.5*x$ will result in

```
>> 2.5*x
ans =
    7.5000
```

You can also assign names to the results of computations. For example,

```
>> y = (x+2)^3
y =
    125
```

will result in y being given the value $(3 + 2)^3 = 125$.

You will have noticed that if you do not assign a name for a computation, MATLAB will assign the default name ans to the result. This name can always be used to refer to the results of the previous computation. For example:

```
>> 2+3
ans =
    5

>> ans/5
ans =
    1
```

MATLAB has a few preassigned variables or constants. The constant $\pi = 3.14159\dots$ is given the name pi .

```
>> pi
ans =
    3.1416
```

The square root of -1 is i .

```
>> sqrt(-1)
ans =
      0 + 1.0000i
```

Engineers and physicists frequently use i to represent current, so they prefer to use j for the square root of -1 . MATLAB is well aware of this preference.

```
>> j
ans =
      0 + 1.0000i
```

There is no symbol for e , the base of the natural logarithms, but this can be easily computed as `exp(1)`.

```
>> exp(1)
ans =
      2.7183
```

Mathematical Functions

There is a long list of mathematical functions that are built into MATLAB. Included are all of the functions that are standard in calculus courses.

Elementary Functions

<code>abs(x)</code>	The absolute value of x , i.e. $ x $.
<code>sqrt(x)</code>	The square root of x , i.e. \sqrt{x} .
<code>sign(x)</code>	The signum of x , i.e. 0 if $x = 0$, -1 if $x < 0$, and $+1$ if $x > 0$.

The Trigonometric Functions

<code>sin(x)</code>	The sine of x , i.e. $\sin(x)$.
<code>cos(x)</code>	The cosine of x , i.e. $\cos(x)$.
<code>tan(x)</code>	The tangent of x , i.e. $\tan(x)$.
<code>cot(x)</code>	The cotangent of x , i.e. $\cot(x)$.
<code>sec(x)</code>	The secant of x , i.e. $\sec(x)$.
<code>csc(x)</code>	The cosecant of x , i.e. $\csc(x)$.

The Inverse Trigonometric Functions

<code>asin(x)</code>	The inverse sine of x , i.e. $\arcsin(x)$ or $\sin^{-1}(x)$.
<code>acos(x)</code>	The inverse cosine of x , i.e. $\arccos(x)$ or $\cos^{-1}(x)$.
<code>atan(x)</code>	The inverse tangent of x , i.e. $\arctan(x)$ or $\tan^{-1}(x)$.
<code>acot(x)</code>	The inverse cotangent of x , i.e. $\text{arccot}(x)$ or $\cot^{-1}(x)$.
<code>asec(x)</code>	The inverse secant of x , i.e. $\text{arcsec}(x)$ or $\sec^{-1}(x)$.
<code>acsc(x)</code>	The inverse cosecant of x , i.e. $\text{arccsc}(x)$ or $\csc^{-1}(x)$.

The Exponential and Logarithm Functions

<code>exp(x)</code>	The exponential of x , i.e. e^x .
<code>log(x)</code>	The natural logarithm of x , i.e. $\ln(x)$.
<code>log10(x)</code>	The logarithm of x to base 10, i.e. $\log_{10}(x)$.

The Hyperbolic Functions

<code>sinh(x)</code>	The hyperbolic sine of x , i.e. $\sinh(x)$.
<code>cosh(x)</code>	The hyperbolic cosine of x , i.e. $\cosh(x)$.
<code>tanh(x)</code>	The hyperbolic tangent of x , i.e. $\tanh(x)$.
<code>coth(x)</code>	The hyperbolic cotangent of x , i.e. $\coth(x)$.
<code>sech(x)</code>	The hyperbolic secant of x , i.e. $\operatorname{sech}(x)$.
<code>csch(x)</code>	The hyperbolic cosecant of x , i.e. $\operatorname{csch}(x)$.

The Inverse Hyperbolic Functions

<code>asinh(x)</code>	The inverse hyperbolic sine of x , i.e. $\sinh^{-1}(x)$.
<code>acosh(x)</code>	The inverse hyperbolic cosine of x , i.e. $\cosh^{-1}(x)$.
<code>atanh(x)</code>	The inverse hyperbolic tangent of x , i.e. $\tanh^{-1}(x)$.
<code>acoth(x)</code>	The inverse hyperbolic cotangent of x , i.e. $\coth^{-1}(x)$.
<code>asech(x)</code>	The inverse hyperbolic secant of x , i.e. $\operatorname{sech}^{-1}(x)$.
<code>acsch(x)</code>	The inverse hyperbolic cosecant of x , i.e. $\operatorname{csch}^{-1}(x)$.

For a more extensive list of the functions available, see the MATLAB *User's Guide*, or MATLAB's online documentation.² All of these functions can be entered at the MATLAB prompt either alone or in combination. For example, to calculate $\sin(x) - \ln(\cos(x))$, where $x = 6$, we simply enter

```
>> x = 6
x =
    6

>> sin(x)-log(cos(x))
ans =
   -0.2388
```

Take special notice that $\ln(\cos(x))$ is entered as `log(cos(x))`. The function `log` is MATLAB's representation of the natural logarithm function.

Output Format

Up to now we have let MATLAB repeat everything that we enter at the prompt. Sometimes this is not useful, particularly when the output is pages in length. To prevent MATLAB from echoing what we

² MATLAB comes with extensive online help. Typing `helpdesk` at the MATLAB prompt should open MATLAB's helpdesk in a separate browser. You can also access MATLAB's standard help files by typing `help` at the MATLAB prompt. For a list of MATLAB's elementary functions, type `help elfun` at the MATLAB prompt.

tell it, simply enter a semicolon at the end of a command. For example, enter

```
>> q=7;
```

and then ask MATLAB what it thinks `q` is by entering

```
>> q
q =
    7
```

If you use MATLAB to compute $\cos(\pi)$, you get

```
>> cos(pi)
ans =
   -1
```

In this case MATLAB is smart enough to realize that the answer is an integer and it displays the answer in that form. However, $\cos(3)$ is not an integer, and MATLAB gives us -0.9900 as its value. Thus, if MATLAB is not sure that a number is an integer, it displays five significant figures in its answer. As another example, 1.57 is very close to $\pi/2$, and $\cos(\pi/2) = 0$. MATLAB gives us

```
>> cos(1.57)
ans =
 7.9633e-004
```

This is an example of MATLAB's exponential, or scientific notation. It stands for 7.9633×10^{-4} , or 0.00079633 . In this case MATLAB again displays five significant digits in its answer. All of these illustrate the default format, which is called the `short` format. It is important to realize that although MATLAB only displays five significant digits in the default format, it is computing the answer to an accuracy of sixteen significant figures.

There are several other formats. We will discuss two of them. If it is necessary or desirable to have more significant digits displayed, enter `format long` at the MATLAB prompt. MATLAB will then display about sixteen significant digits. For example,

```
>> format long
>> cos(1.57)
ans =
 7.963267107332634e-004
```

There is another output format which we will find useful. If you enter `format rat`, then all numbers will be shown as rational numbers. This is called the rational format. If the numbers are actually irrational, MATLAB will find a very close rational approximation to the number.

```
>> cos(1.57)
ans =
 47/59021
```

The rational format is most useful when you are working with numbers you know to be rational. After using a different format, you can return to the standard, short format by entering `format short`.

Complex Arithmetic

One of the nicest features of MATLAB is that it works as easily with complex numbers as it does with real numbers. The complex number $z = 2 - 3i$ is entered exactly as it is written.

```
>> z = 2-3i
z =
    2.0000 - 3.0000i
```

Then if we enter $w = 3 + 5i$, we can calculate sums, products, and quotients of these numbers in exactly the same way we do for real numbers. For example,

```
>> w = 3+5i;
>> z*w
ans =
    21.0000 + 1.0000i
```

and

```
>> z/w
ans =
   -0.2647 - 0.5588i
```

Any of the arithmetic functions listed earlier can be applied to complex numbers. For example,

```
>> y = sqrt(w)
y =
    2.1013 + 1.1897i
```

and

```
>> y*y
ans =
    3.0000 + 5.0000i
```

Since $y^2 = w$, it is a square root of the complex number w . The reader might try $\cos(w)$ and $\exp(w)$. In particular, the reader might wish to verify Euler's formula

$$e^{i\theta} = \cos(\theta) + i \sin(\theta)$$

for several values of θ , including $\theta = 2\pi, \pi, \pi/2$.

```
>> theta = pi; exp(i*theta), cos(theta) + i*sin(theta)
ans =
   -1.0000 + 0.0000i
ans =
   -1.0000 + 0.0000i
```

Note that several MATLAB commands can be placed on a single line, separated by commas, or semicolons, should you desire to suppress the output.

The ease with which MATLAB handles complex numbers has one drawback. There is at least one case where the answer is not the one we expect. Use MATLAB to calculate $(-1)^{1/3}$. Most people would expect the answer -1 , but MATLAB gives us

```
>> (-1)^(1/3)
ans =
    0.5000 + 0.8660i
```

At first glance this may seem strange, but if you cube this complex number you do get -1 . Consequently MATLAB is finding a *complex* cube root of -1 , while we would expect a real root. The situation is even worse, since in most of the cases where this will arise in this manual, it is not the complex cube root we want. We will want the cube root of -1 to be -1 .

However, this is a price we have to pay for other benefits. For MATLAB to be so flexible that it can calculate roots of arbitrary order of arbitrary complex numbers, it is necessary that it should give what seems like a strange answer for the cube root of negative numbers. In fact the same applies to any odd root of negative numbers. What we need is a way to work around the problem.

Notice that if $x < 0$, then $x = -1 \times |x|$, and we can find a negative cube root as $-1 \times |x|^{1/3}$. Here we are taking the real cube root of the positive number $|x|$, and MATLAB does that the way we want it done. But suppose the situation arises where we do not know beforehand whether x is positive or negative. What we want is

$$x^{1/3} = \begin{cases} |x|^{1/3}, & \text{if } x > 0; \\ 0, & \text{if } x = 0; \\ -1 \times |x|^{1/3}, & \text{if } x < 0. \end{cases}$$

To write this more succinctly we use the *signum* function $\text{sgn}(x)$ (in MATLAB it is denoted by `sign(x)`). This function is defined to be

$$\text{sgn}(x) = \begin{cases} 1, & \text{if } x > 0; \\ 0, & \text{if } x = 0; \\ -1, & \text{if } x < 0. \end{cases}$$

Thus, in all cases we have $x = \text{sgn}(x) |x|$, and the real cube root is

$$x^{1/3} = \text{sgn}(x) |x|^{1/3}.$$

In MATLAB, we would enter `sign(x)*abs(x)^(1/3)`.

Recording Your Work

It is frequently useful to be able to record what happens in a MATLAB session. For example, in the process of preparing a homework submission, it should not be necessary to copy all of the output from the computer screen. You ought to be able to do this automatically. The MATLAB `diary` command makes this possible.

For example, suppose you are doing your first homework assignment and you want to record what you are doing in MATLAB. To do this, choose a name, perhaps `hw1`, for the file in which you wish to record the output. Then enter `diary hw1` at the MATLAB prompt. From this point on, everything that

appears in the Command Window will also be recorded in the file `hw1`. When you want to stop recording enter `diary off`. If you want to start recording again, enter `diary on`.

The file that is created is a simple text file. It can be opened by an editor or a word processing program and edited to remove extraneous material, or to add your comments. You can use the MATLAB editor for this process. To access it, first be sure you have stopped the editing process by executing `diary off`. Next, either click on the open file in the Toolbar, or select **Open file** from the **Edit** menu. The file selection window that opens is set to display only MATLAB files, so the text diary file will not be visible. Click on the Files of type: popup menu and select All Files. Then select the name of your diary file. You will now be able to make any changes you want in the file. You will also be able to print the file to get a hard copy.

To open your diary file, or any other file you create with MATLAB, you will have to know where it is stored on your computer. This means that you will have to understand the directory structure on your computer. (We are using directory as a synonym of file folder.) Your diary file is saved in the current directory. In MATLAB 6, the current directory is displayed in a small box at the top of the command window. Next to it there is a button with three dots (an ellipsis) on it. Clicking this button will open a new window containing the directory tree of your computer. You can make any directory the current directory by selecting it from the directory tree.

Different operating systems provide different ways of handling files and directories. However, it is possible to do a lot of file handling entirely within MATLAB. For this purpose MATLAB uses a combination of UNIX and DOS commands. You can find the name of the current directory by using the command `pwd`. The response will be the full address of the current directory in the language of your operating system. You can obtain a list of the files in the current directory with the commands `ls` or `dir`. You can change directories with the commands `cd` or `chdir`. You can make a new directory with the command `mkdir`. We suggest that you use the `help` command³ to find out more, and experiment with these commands to learn what they do in your system.

Exercises

1. Use the standard procedure on your computer system to create a folder named `mywork`. In MATLAB change to this folder either by using the command `cd` at the MATLAB prompt or by clicking the ellipsis (...) button next to the Current Directory edit box on your MATLAB toolbar and browsing to the folder `mywork`. Look at the Current Directory edit box to be sure that `mywork` is the current directory. You can also use the command `pwd` at the MATLAB prompt. Clear the command window with the command `clc`, then clear your workspace of all variables with the command `clear`. Start a diary session with `diary hwk1`. Read Chapter 1 (Introduction to Matlab) of this manual again, but this time enter each of the commands in the narrative at the MATLAB prompt as you read. When you are finished, enter the command `diary off`. Open the file `hwk1` in your favorite editor or word processor (or open it in Matlab's editor by typing `edit hwk1` at the MATLAB prompt). Edit and correct any mistakes that you made. Save and print the edited file and submit the result to your instructor.

³ For example, type `help cd` to obtain help on using the command `cd`.