كلية المستقبل الجامعة

قسم الفيزياء الطبية

MATLAB

Lec - 2

اعداد

م.م علا علي عبيد

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 × 3. You type it at the prompt command (>>) as follows,

```
>> 1+2*3

ans =

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
```

will result in x being given the value 1 + 2 × 3 = 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.

4

Table 1.1: Basic arithmetic operators :

Symbol Operation Example

+ Addition 2 + 3

− Subtraction 2 − 3

∗ Multiplication 2 ∗ 3

/ Division 2/3

Quitting MATLAB

To end your MATLAB session, type quit in the Command Window, or select File −→ Exit

MATLAB in the desktop main menu.

Getting started

After learning the minimum MATLAB session, we will now learn to use some additional

operations.

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. On other words, expression can involve:

- manual entry
- built-in functions
- user-defined functions

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
```

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.
```

Making corrections :

To make corrections, we can, of course 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.

Controlling the hierarchy of operations or precedence :

Let's consider the previous arithmetic operation, but now we will include parentheses. For

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

>> (1+2)*3

ans =

9

and, from previous example>> 1+2*3

ans =

7

By adding parentheses, these two expressions give different results: 9 and 7.

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. However, the standard order of

precedence of arithmetic operations can be changed by inserting parentheses. For example,

the result of 1+2×3 is quite different than the similar expression with parentheses (1+2)×3.

The results are 7 and 9 respectively. Parentheses can always be used to overrule priority,

and their use is recommended in some complex expressions to avoid ambiguity.

Therefore, to make the evaluation of expressions unambiguous, MATLAB has established a series of rules. The order in which the arithmetic operations are evaluated is given

in Table 1.2. MATLAB arithmetic operators obey the same precedence rules as those in

Table 1.2: Hierarchy of arithmetic operations

Precedence Mathematical operations

First The contents of all parentheses are evaluated first, starting

from the innermost parentheses and working outward.

Second All exponentials are evaluated, working from left to right

Third All multiplications and divisions are evaluated, working

from left to right

Fourth All additions and subtractions are evaluated, starting

from left to right

most computer programs. For operators of equal precedence, evaluation is from left to right.

Now, consider another example:

2 + 32

+

4

5

×

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
```

So here what we get: two different results. Therefore, we want to emphasize the importance

of precedence rule in order to avoid ambiguity.

1.4.6 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. For more details, see the MATLAB documentation,

or type help format.

Note - Up to now, we have let MATLAB repeat everything that we enter at the

prompt (>>). Sometimes this is not quite useful, in particular when the output is pages en

length. To prevent MATLAB from echoing what we type, simply enter a semicolon (;) at

the end of the command. For example,

```
>> x=-163.6667;
```

and then ask about the value of x by typing,

```
>> x

x =

-163.6667
```

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, it is a good idea to issue a clear command at the start of each new independent calculation.