MATLAB Basics

1 Introduction

The purpose of this tutorial is to present basics of MATLAB. Our goal is to provide sufficient information to give you a good start. We do not assume any prior knowledge of this package. The emphasis here is "learning by doing". The name MATLAB stands for MATrix LABoratory. MATLAB is a high-level computer language for *technical computing*. It integrates *computation*, *visualization*, and *programming environment*.

2 A minimum MATLAB session

The goal of this minimum session (also called *starting* and *exiting* sessions)is to learn the first steps:

- How to **log on**
- **Invoke** MATLAB
- Do a few simple calculations
- How to **quit** MATLAB

2.1 Starting MATLAB

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 Folder (Directory)**
- The **Help Browser**

2.2 Using MATLAB as a calculator

As an example of a simple interactive calculation, just type the expression you want to evaluate in the **Command Window**. This window allows you to enter simple commands. For example, let's suppose you want to calculate the expression, $1 + 2 \cdot 3$. You type it at the prompt command (>>) in the **Command Window**,

```
>> 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 already existed). To avoid this inconvenience due to overwrite, 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 $1 + 2 \cdot 3 = 7$. This variable name can always be used to refer to the results of the previous computations. Therefore, computing $4 \cdot$ `x` will result in

```
>> 4*x
ans =
    28.0000
```

To clear the **Command Window** type `clc` and next press the **enter** or **Return** key. Before we conclude this minimum session, Table 1 gives the partial list of common arithmetic operators.

Table 1: Arithmetic operators

Operation	Algebraic Syntax	MATLAB syntax
Addition	a+b	a+b
Subtraction	a-b	a-b
Multiplication	$a \cdot b$	a*b
Division	$\frac{a}{b}$ or a÷b	a/b
Exponentiation	a^b	a∧b

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**.

3 Getting started

Our goal is to provide sufficient information to give you a good start. If you are familiar with another computer language, it is not difficult to pick up the rest as you go. If you are not, it is important that you follow the recommendations given in this handout step by step.

3.1 Creating MATLAB variables

MATLAB variables are created with an assignment statement. The syntax of variable assignment is

```
>> x = expression
```

where expression is a combination of numerical values, mathematical operators, variables, and function calls. The following naming rules apply

- All names must start with a *letter*. The names can be any length, but only the first 63 characters are used in MATLAB.
- MATLAB variables are *case sensitive*. A variable named `Pressure` is different from a variable named `pressure`.
- Some words are *reserved* and cannot be used to name a variable; for example, `plot, print, global, function`, . . . The command `iskeyword` causes MATLAB to list these reserved names.

3.2 Overwriting variables

3.2.1 Description

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.

3.2.2 Exercise 1

1. Create a variable `t` that is equal to 5. Use a semicolon to suppress the output.
2. Reassign `t` to now have the value of `t+1`. Omit the semicolon this time. The value of t should now be 6.

Remember to use a semicolon at the end of a line when you don't want to see the output of the command. This can often eliminate unnecessary information you may not need.

3.3 Error messages

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

```
>> x = 10;
>> 5x
??? 5x
     |
Error: Unexpected MATLAB expression
```

3.4 Making corrections

To make corrections, we can, of course retype the expressions. But if the expression is lengthy, we may make more mistakes by typing it 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. We can also keep pressing the up-arrow key to get to even earlier commands in the command history.

3.5 Controlling order of operations

In all mathematical calculations, it is important to understand the order in which operations are performed. MATLAB follows the standard algebraic rules for the order of operations. As an example, let's consider the previous arithmetic operation, 1 + 2 x 3, but now we will include parentheses. The expression will become

```
>> (1+2)*3
ans   =
      9
```

In this case, MATLAB first performs calculations inside the parentheses. Let's recall the previous result,

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

It is important to note that 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 any algebra course. Exponents are evaluated first, followed by multiplication and division, and finally by addition and subtraction. However, the standard order of precedence of arithmetic operations can be changed by inserting parentheses. Then the expression inside the parentheses is evaluated first according to the normal order of operations, and everything outside of the parentheses is evaluated next.

3.6 Controlling how many decimals are displayed

3.6.1 Description

MATLAB by default displays only 4 digits after decimal point, for example -163.6667. However, MATLAB does numerical calculations in double precision, which is 15 digits after decimal points. The command `format` controls how the results of computations are displayed.

3.6.2 Exercise 2

1. Assign the value -163.666666666667 to the variable x.
2. Type `format short` in the command window and hit enter. Then type `x` and hit enter.
3. If you want to see all 15 digits of x, type `format long`, hit enter and then enter `x` in the next line to see that it has all 15 digits.

To return to the standard format (4 digits), enter `format short`, or simply `format`. There are several other formats. For more details, see the MATLAB documentation, or type `help format` or `doc format`.

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, it is a good idea to issue 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. On the other hand, if you want to display a list of the variables currently in the system memory, type

```
>> who
```

while, `whos` will give more detailed information which includes size, space allocation, and class of the variables.

3.8 Keeping track of your 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
```

where `FileName` could be any arbitrary name you choose; e.g. `diary lab1`.

The function `diary` is useful if you want to save a complete MATLAB session. It saves all inputs and outputs as they appear in the MATLAB window. It may be a good idea to clear the screen first, using the command `clc`, before recording. 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 is saved in the folder that your internal path is set to (you can see/change the folder you are in by looking at the file name above the command window). The file can be opened by an editor or a word processing program and edited to remove extraneous material, or to add your comments to it. This command is useful, for example, in the process of preparing a lab submission or a homework.

3.9 Miscellaneous commands

Here are a few useful commands:

- To abort a MATLAB computation, type ctrl-c.

  This can be useful when a long script or function is running and you want to stop it.

- To continue a command from the next line, type ...

  This is helpful when you have a very long command that you are putting in.

3.10 Getting help

To view the online documentation, select MATLAB Help. 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
```

or you can type

```
>> doc command
```

where you replace the word "command" with the command you are trying to get help on. Another way to get help is to use the `lookfor` command. The `lookfor` command 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.

4 Introduction to Matrices

Matrices are the basic elements of the MATLAB environment. A matrix is a two-dimensional array consisting of m rows and n columns, i.e. A(m,n). Special cases are column vectors (n=1) and row vectors (m=1).

4.1 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 an 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] and once you 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 the first column. Its value is 4.

4.2 Exercise 3

a. Create the matrix

$$M = \begin{bmatrix} 2 & 7 & 4 \\ -1 & 0 & 5 \\ 9 & 3 & -6 \end{bmatrix}$$

b. Find the value of the element in the third row and second column in the same way as the previous example.

4.3 Matrix indexing

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 the first row and third column; i.e. A(1,3)=3. Therefore, 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

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$

Single elements of a matrix are accessed as A(i,j), where i $\geq$ 1 and j $\geq$ 1. Zero or negative subscripts are not supported in MATLAB.

4.4 Colon operator

The colon operator (:) will prove very useful and understanding how it works is the key to efficient and convenient usage of MATLAB. It occurs in several different forms. Often we must deal with matrices or vectors that are too large to enter one element at a time. For example, suppose we want to enter a vector x consisting of points (0; 0.1; 0.2; 0.3; ... ; 5). We can use the command,

```
>> x = 0:0.1:5;
```

The first number is the starting value, the second number is the increment in which the values will increase, and the third number is the ending value. The vector **x** above creates a row vector that goes from 0 to 5 with an increment of 0.1.

4.5 Exercise 4

Create a column vector that goes from 2 to 6 with increments of 0.25.

4.6 Colon operator in a matrix

The colon operator can also be used in a matrix to pick out a certain row or column. For example, the statement A(m:n,k:l) specifies rows m to n and columns k to l. Subscript expressions refer to portions of a matrix. For example,

```
>> A(2,:)
ans  =
     4  5  6
```

where ":" means that all of the columns are chosen. In the previous example, the second row and all of the columns of the matrix A are selected. The colon operator can be used in this way to select all of the rows of a matrix as well.

The colon operator can also be used to extract a sub-matrix from a matrix.

```
>> A(:,2:3)
ans  =
     2  3
     5  6
     8  0
```

The first input within the parentheses (the colon) indicates that all of the rows of the matrix A are selected and the second input (2:3) refers to the second through the third columns of the matrix. Thus, A(:,2:3) is a sub-matrix with the last two columns of A.