Computer Lab 1 – Week 2

Aims:

Upon successfully completing these tutorial exercises, students should be able to:

- Perform arithmetic operations in the command window.
- Assign values to variables and use variables for arithmetic operations.
- Use built-in functions/variables for arithmetic operations.
- Write a basic script which asks the user for some information and returns an answer.

Today we will be using the command window in MATLAB. Open MATLAB and identify the command window. When the command window is ready to receive a new command, you should see the symbols >>. In the command window, any code you have written will be executed only when Enter is pressed. If the >> does not appear, it means MATLAB is either busy or is waiting for some more code before an execution is possible.

Handy hints: If at any time MATLAB is stuck trying to solve a problem that is taking too long (maybe you have accidently created an infinite loop), just press CTRL+C. Also, if you want to write the same line of code again in the command window, just press the up arrow.

1 Perform arithmetic operations in the command window.

The use of addition, subtraction and multiplication is quite obvious within MATLAB. These operations are performed using the +,- and * symbols respectively. Try the following in the command window:

```
3-2 <Press the ENTER button after each line>
5+7
4*5
```

To perform power operations, for example to find 5^3, use the ^ symbol:

```
5^3
```

When performing division, it is better to think of the division operation as an invert and then multiply. For example:

$$4 \div 2 = 4 \times 2^{-1} = 4 \times 0.5 = 2$$

The invert and multiply operation is performed by a forward slash / or back slash \. A forward slash means inverter the second number and then multiply. A backslash means invert to first number and then multiply. Try and predict the answers to the following and then try them in the command window:

```
4/2
4\2
```

The standard order of operations is used in MATLAB. Parentheses are calculation first, next power, division/multiplication and finally addition/subtraction. Try to predict the answers of the following and then test in the command window:

```
4+5*3
5+24/12-6^0+1
(8*(4+2))\24/2/2
```

2 Assign values to variables and use variables for arithmetic operations.

A variable is used to store information. At any time, the value stored in the variable may be changed and used in an operation. A variable can be given any name that contains no spaces or special characters. A variable may contain numbers, but must not start with a number. Generally variables contain letters, numbers and underscores as these characters do not usually create any conflicts. Variables are often given names with some significance of what is being stored. For example:

```
miles_to_km=1.61;
```

This will store the value 1.61 into a variable called miles_to_km. The function of the equals sign = is very important to understand (as it does not mean the same thing in mathematics). The equals sign = will store whatever is on the right hand side of the = into the variable on the left hand side.

Handy hint: If you finish a line of code with a semicolon ; then the answer to the line of code will not be displayed after the code is executed. Compare:

```
3+4;
3+4
```

Now that the miles_to_km has been given a value, the value can be used in arithmetic operations. If we want to find out how many kilometres there are in 15 miles:

```
miles_to_km*15
```

Let us introduce two new variables, a and b. To assign 2 and 3 into a and b respectively:

```
a=2;
b=3;
```

Note that values stored in variables can be overwritten:

```
a=b+1;
```

Now a contains the value 4, the old value 2 has been overwritten. To check what is stored in a variable, simply type the name of the variable and press enter.

```
a
```

It is also possible to use a variable in the calculation of the updated value of the same variable. For example, you might want to add 1 to b:

```
b=b+1;
```

This will overwrite the old value of 3 and now 4 will be stored in b.

If you want to clear all variables, use the function clear

```
clear
```

Now the variables a and b will no longer exist.

3 Use built-in functions/variables for arithmetic operations.

There are many built-in functions and variables in MATLAB that will always exist even after you use the clear function. The most common built-in variable is pi. Built-in functions include: sqrt(), sin(), cos(), tan() and exp(). Note that all functions require use of parentheses (you will get an error if you don't use them). Functions take in one or more input, perform some operation and return an answer. In the cases of the built-in functions already mentioned, all require only one input which should be written inside the parenthesis. For example:

```
x=4;
y=sqrt(x)
sin(pi)
```

```
cos(pi)
exp(sqrt(x))
exp(y)
```

Built-in variables and functions can be overwritten. If they are overwritten, the old function/value will no longer exist until the clear function is used (or MATLAB is restarted).

Try:

```
pi
pi=4
pi
clear
pi
```

4 Write a basic script which asks the user for some information and returns an answer.

Programs generally require interaction with a human to gain some data. Here, we will ask the user their age and return the year in which they were born. In order to do this, we will need to:

- write a message to the user requesting the data.
- store the data in a variable.
- calculate the year in which they were born.
- finally, write a message in the command window informing the user of the year of birth.

Because there are multiple steps, we will require multiple lines of code. The command window is useful for finding quick responses to individual lines of code, but is not useful where multiple lines are used, or require repeating, or need to be saved. Hence, we will create a script. Under the HOME tab, click New Script. You should now see an Untitled script in the Editor. This is where the code for your script shall go. Inside the script, code will not be run until the Run button is pushed, or until the name of the script is entered in the command window. Save the script as YOB (for year of birth).

In the script, write the code:

```
age=input('Enter your age: ');
birth_year=2017-age;
disp(['Your year of birth was: ', num2str(birth_year)]);
```

There is a lot going on in this code. First, let's discuss the use of the inverted comma ' character. In MATLAB, the inverted comma is used to define a string. A string is an array of characters which are used for text purposes (they are not appropriate for mathematical operations). For example, a='48'is different to a=48. The first will store the characters 4 and 8 into the variable a, whereas the second will store the number 48 into a.

Secondly, let's look at the square bracket [. In MATLAB, square brackets are used to concatenate numbers or strings, meaning to join them together. Concatenating strings with numbers will give you unexpected results. Try:

```
a=[4,70]
b=[1, 3, 5]
c=[a, b]
d=['join this string with ', 'this string']
e=['this will be strange', a]
f=['but this will work ', num2str(a)]
```

There are three in-built functions used in this code:

input() displays the string input and returns what text the user inputs in the command window as a number type.
disp() displays the string input only.
num2str() converts a number type variable to a string type variable.

Handy hint: if you are unsure of what a function does or how it works, in the command window type help, put a space and write the name of the function. Information about that function should appear. Try:

```
help input
help num2str
help disp
```

5 For you to do:

Using this example, create another script that will ask the user their height and weight and return their body mass index (BMI). BMI=mass/height^2. Height is in m and mass is in kg.