Numerical Methods
Lab 0

08 February 2018

Instructions

- Read all the instructions carefully.
- MATLAB has a help file for every function if you get stuck.
- There are also numerous sources available on the internet, Google is your friend!

SECTION 1
Introduction

Since you are now third years, you are expected to source a large quantity of information by yourself. However, when required - I will provide you with useful MATLAB syntax and explanations. The first lab serves as an introduction to MATLAB. Although some of you have used it before, many have not.

MATLAB'S command window works similar to that of the interpretor of PYTHON. As such it will read line by line. Just like PYTHON, we also write scripts in MATLAB. This file has the extension .m - It is always good practice to write code in a script and never the command window.

Play around with the command line. You will find that many things you're used to in PYTHON are the same in MATLAB.
A major difference however, is how PYTHON is value specific. Secretly - MATLAB controls this for you behind the scenes which allows for sloppy coding. PYTHON does not. A **value** is one of the basic things a program works with, such as a letter or a number - for example: 1,2 and ‘Hello, World!’.
These values belong to different **types**: 2 is an integer, and ‘Hello, World!’ is a string, since it contains a “string" of letters. These are differentiated by you and the command line since they are enclosed in quotation marks.

These different types are extremely important in PYTHON, less so in MATLAB. You can easily find out the type a value has by asking the interpretor with `type()` in PYTHON and `class()` in MATLAB.

```
>>> type('Hello,World!') # In Python
<type 'str'> # In Python
```

```
>> class(4) % In Matlab
ans =

double
>> class('Hello,World!') % In Matlab
ans =

char
>> isfloat(4)

ans =

1
```

MATLAB usefully returns the class type of the input in question. In addition you can also ask it more detailed classifications, using the `is*`, where * may be replaced with a variety of options.
Now explore the different operations that you can perform with the above types.

SECTION 2

Suppressed Output and Functions

It is always good programming practice to write your code in functions. A function is like the exact built in functions that you should be used to. It's use is to take some input and produce a output.

It is very important when writing functions to suppressed your code - especially in this course as any additional outputs will be deemed incorrect.

A sample function would look like this:

```
1 function y = square(x)
2 y = x*x;
3 end
```

Here, the function takes some input, performs some operations and returns the result `y`. Notice the semi-colon on line 2. This suppress the line from being printing to the command line. Unlike PYTHON, MATLAB will attempt to print every line unless suppress.
A small extension of the above code would be if you wanted multiple outputs. Modifying the code to find the cube as well as the square we have:

```
function [y1 y2] = squareCube(x)
y1 = x*x;
y2 = x*x*x;
end
```

When calling this function, you must use two variables - if you do not, you will simply get `y1`. A sample call would be `[y1 y2] = square(4)`. Play around with the above until you are comfortable with how functions operate.

NB! - Remember! All functions in MATLAB must be finished with an `end` statement.

SECTION 3

Prompting User Input

Often it is necessary to await user input for various programs. This was done using `raw_input` with PYTHON and `inline` in MATLAB. This function however, will soon be removed from the language. The function `input` will now be the equivalent statement. An example of this would be:

```
prompt = 'What_is_the_original_value?';
result = input(prompt)
X = result*10
```

Make sure that you have a strong understanding on the above as each week will require you to make use of them.

SECTION 4

Warm-up Exercises

The following exercises are not related to the numerical methods content but rather serve as a means to practice your MATLAB skills.

4.1 Exercise

Write a program that prompts a user for their name and then welcomes them. A sample input/output would be:

```
Enter your name:  Matthew
Hello Matthew
```

4.2 Exercise

Write a program to prompt the user for hours and rate per hour to compute gross pay. A sample input/output would be:

```
Enter Hours:  35
Enter Rate:  2.75
Pay:  96.25
```

SECTION 5

Main Exercises

5.1 Exercise 1

Absolute Error - Write a function that will calculate the absolute error. The first line of your function should look like this:

```
function error = AbsoluteError(approxVal, trueVal)
```

A sample input would be:

```
 error = AbsoluteError(65.02, 65)
```

With the approximate value read in first and the correct value second. While the output would be:

```
The absolute error is:  0.02
```

5.2 Exercise 2

Repeat the process above but instead of the absolute error, compute the relative error. Again the first line of your function should look like this:

```
function error = RelativeError(approxVal, trueVal)
```

The sample case above should return:

```
The relative error is:  0.000307692307692
```

5.3 Exercise 3

If we list all the natural numbers below 15 that are multiples of 3, we get 3, 6, 9, 12. The sum of these numbers is 30. Write a matlab function that takes in as input a number n, and which returns as output the sum of the multiples of 3 below n .

5.4 Exercise 4

If we list all the natural numbers below 15 that are multiples of 3 and 5, we get 3, 5, 6, 9, 10, 12. The sum of these numbers is 45. Write a matlab function that takes in as input a number n, and which returns as output the sum of the multiples of 3 and 5 below n.