Mathematics for the Life Sciences (MATH1134) - Tutorial Sheet 1

The main aims of this tutorial are to

- develop familiarity with the computing environment at the University;
- provide a gentle introduction to Matlab and the Matlab programming environment;
- get students to begin to appreciate some of the issues which arise when using a computer to do calculations.

Matlab (and Python) will be used as tools in this course to help you understand the more theoretical material. Often we will use in-built functions, however you will also be exposed to the very important and marketable skill of programming. At the same time, it should be emphasised that programming is not the main focus of the course; it is only a tool (albeit quite useful) and the course is currently designed to minimise the programming requirements. I anticipate that we will write and regularly use about 3-4 main programs in the course, and that writing will be done fairly collaboratively. In additon, the good news is that it is actually quite easy to learn how to program well in Matlab (or Python) as long as you approach learning in a systematic manner and pay attention to the suggestions in the tutorial and supplementary Matlab lecture notes ***(in the "Matlab Files" folder under the Lecture 2 section of the MATH1134 Moodle page)*** *covering essentially Lecture 2 of the second year course MATH1106.*

This tutorial gives a gentle introduction to Matlabt and also exposes you to Octave and Python. Please ask as many questions as you wish to ensure that the material here is clear to you.

1. The computers in the King William computer teaching labs on this campus should have Matlab and Python (Anaconda) already installed. If you are at a University computer at which Matlab is not already installed, please use the Microsoft **Software Center** to install it. The **Software Center** can usually be found in the main Start menu in Windows. Otherwise, you can find it by typing "software center" into the search box which appears at the bottom when you click on that Start menu *(Windows 7)* OR when you click on the search icon at the bottom of the screen *(Windows 10)*. If needed, more information on how to access and use the **Software Center** may be found by clicking here (if reading a paper copy of this tutorial, the URL is

   http://www.gre.ac.uk/__data/assets/pdf_file/0019/1004905/Installing_Additional_Software_from_the_Microsoft_Software_Center.pdf)

   It is also possible to use the **virtual desktop** to access Matlab and Python (Anaconda) remotely from a University computer or any other computer with an internet connection - including from your home.

   - Instructions for logging into the virtual desktop from a University computer (and thus access the Windows lab software available on campus, such as Matlab) can be found at http://ach-support.gre.ac.uk/labdesktop/
   - You can also install the VMWare View client on your laptop or home computer, thus allowing you to log into the virtual desktop (and access the Windows lab software available on campus) from outside of the University. Instructions for installing the client for different operating systems may again be found at

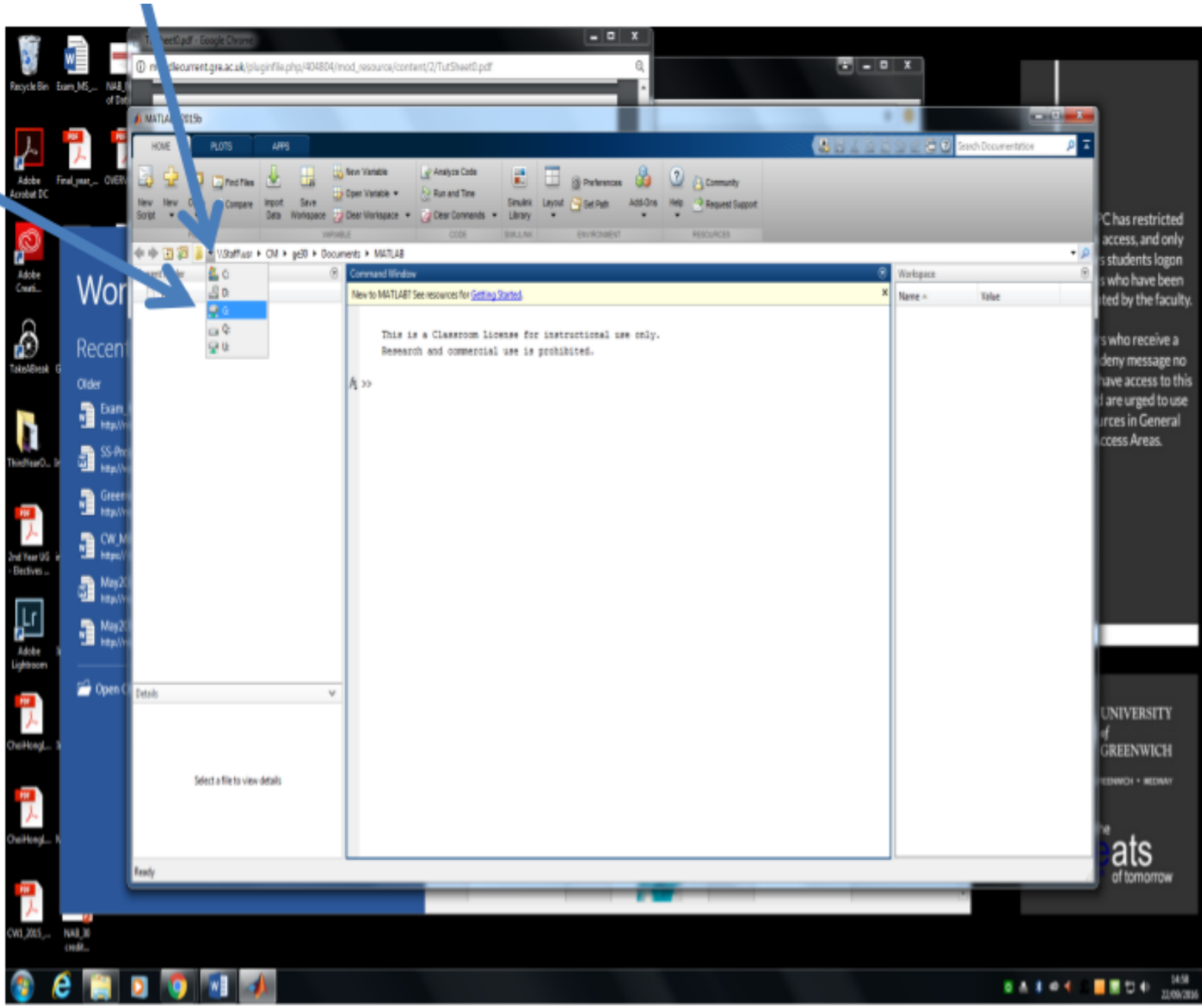
     http://ach-support.gre.ac.uk/labdesktop/

   ↪ An alternative source of information on the *Virtual Lab Desktop* is the Moodle course *MSC Mathematical Sciences* for the current academic year *(in which you all should be enrolled)*; information is provided on that Moodle page in a section called **Virtual Lab Desktop**.

- ≻ **NOTE at the time of writing of this tutorial, the computers in the King William building labs ran *Windows 10* while the Virtual Lab Desktop ran *Windows 7*, so some of the instructions in this tutorial reflect the slight differences in the two operating systems. The Virtual Lab Desktop should now also be running Windows 10**.

2. First, navigate in Windows to your G drive. This can usually be accessed by clicking on the "START" button and selecting "Computer" then double-clicking on the G drive *(Windows 7)* OR by clicking on the Windows Explorer (folder) icon in the menu at the bottom of the screen and selecting "This PC" from the left menu *(Windows 10)*. NOTE the G drive can have a long offical name, so it might sometimes be necessary to let the mouse pointer hover over the drive name to see the full name and the letter "G" at the end. The advantage of storing things in this drive is that it is accessible from all University computers and from the virtual desktop (hence from any device with an internet connection). Once in your G drive, create a folder where you will put all of your MATH1134 work - call it something like MATH1134. Go to this MATH1134 folder. You will want to create subfolders here for the different tutorials, coursework questions, etc. where you will store files related to that tutorial/coursework etc. For today, you only need one subfolder called "Tutorial1" or something similar. Go to that subfolder. We will likely put only one file in that "Tutorial1" subfolder today, but getting used to this level of organisation is good. Being this organised will make life much easier when you are doing the coursework since it will ensure all relevant "program" files are in the same place and are easily accessible.

3. Next, start Matlab by clicking on the search icon on the bottom of the screen to open up a search box *(this search box is accessible via the "START" button if using Windows 7)* and typing `matlab` in that search box and then hitting "ENTER" (there are other ways to open Matlab such as using the "START" button and selecting "Programs" then "Maths Applications" then "Matlab"). NOTE if Matlab is not yet installed on the computer you are using, see the instructions above for installing it using the **Software Center**. A Matlab *Desktop Window* should (eventually) pop up. The main feature of this *Desktop* should be a *Command Window*, with a *Current Folder* section to the left from which you can access the files you are working on currently, and a *Workspace* section to the right where you can see information about the variables defined in the current Matlab session. Use the menu at the top of the *Current Folder* section to navigate to the G drive and then select your MATH1134 folder and then your Tutorial1 subfolder (where we will later put one file) ⇝ see the following image for some guidance on how to navigate to the relevant subfolder.



In general, unless you plan to use Matlab only as a calculator and not save anything, it is a good idea to get into the habit of always first navigating to the folder you wish to work/save files in as soon as you open Matlab.

4. You can use the Matlab command window like a (very sophisticated) calculator. In doing so, most of the mathematical operators and function names are what you would expect. For example,

Operator/Function call	Matlab version
addition	$+$
subtraction	$-$
multiplication	$*$
division	/
sine of x	$\sin(x)$
cosine of x	$\cos(x)$
tangent of x	$\tan(x)$
square-root of x	$\text{sqrt}(x)$

The following operators/function calls are maybe a little unusual in Matlab, **so learn them!**

Operator/Function call	Matlab version
Raising something to a power - for example, 2^3	^ - for example 2^3
e^x	$\exp(x)$
$\ln(x)$	$\log(x)$
the absolute value of x	$\text{abs}(x)$

In addition, there are several pre-defined mathematical constants in Matlab with the names you might expect. Of these, probably the most important for you is *pi* which is, of course, π.

5. Perform the following calculations in the Matlab command window.

   (a) $\ln(0.178)$ *(Ans:* -1.72597172869*).*

   (b) $e^{\sin\left(\frac{\pi}{2}\right)}$ *(Ans:* 2.71828182845905*)*

   (c) $\pi \times e$ *(Ans:* 8.53973422267*).*

   (d) $\cos^5(3.1)$. *(Ans:* -0.99568322455*).*

   (e) $45 \times \frac{\sqrt{99}}{4}$ *(Ans:* 111.93608667449.

   (f) $(6+8)/2 \times 7$; $(6+8)/(2 \times 7)$; $6+8/2 \times 7$. *Ans:* 49, 1, 34.

SOLUTIONS

Calculation	Matlab version	Answer
(a) $\ln(0.178)$	$\log(0.178)$	**-1.72597172869**
(b) $e^{\sin\left(\frac{\pi}{2}\right)}$	$exp(sin(pi/2))$	**2.71828182845905**
(c) $\pi \times e$	$pi * exp(1)$	**8.53973422267**
(d) $\cos^5(3.1)$	$(cos(3.1))$**^5**	**-0.99568322455**
(e) $45\frac{\sqrt{99}}{4}$	$45 * sqrt(99)/4$	**111.93608667449**
(f) $(6+8)/2 \times 7$**;** $(6+8)/(2 \times 7)$**;** $6+8/2 \times 7$**.**	$(6+8)/2 * 7$**,** $(6+8)/(2 * 7)$**,** $6+8/2 * 7$	49, 1, 34

6. Type

```
format long
```

in the Matlab command window and then redo the previous exercise. What do you notice?

HINT: You don't have to re-type the commands. Just use the UP (↑) key on your computer (as many times as is necessary) to go back to old commands that you have typed. When using the UP key, you can also use the DOWN (↓) key to move forward to more recent commands that you have typed. These keys are typically located near the bottom right-hand side of your keyboard, to the left of the number pad.

SOLUTIONS

You get the same answers but with more significant digits.

7. Number 5(f) emphasises the importance of *knowing in what order a computer will perform a sequence of mathematical operations* (operator precedence). The rule is **BEDMAS**:

   Brackets → **E**xponents → **D**ivision/**M**ultiplication → **A**ddition/**S**ubtraction.

   ↪ And with operations at the same level (e.g., division and multiplication), perform operations in **Left to Right** order (the order in which you read).

   ⇝ So *USE BRACKETS IF YOU ARE UNCERTAIN.*

   I cannot stress enough how important it is to pay close attention to operator precedence when writing out formulae. Getting this wrong is often the source of very hard-to-find bugs in programs, so ALWAYS keep this BEDMAS rule in mind when entering any formula in a program (or a calculator).

8. Variables: In programming, a variable is a symbol or name that stands for a value - often a number or word(s). They are used alot in programming since they allow programs to be flexible - meaning that one program can do many different calculations depending on the values of the variables. For example, if I define a variable a to be the coefficient of the squared term in a quadratic equation, the variable b to be the coefficient of the linear term, and the variable c to be the constant term, then a general program can easily be written to solve the quadratic equation $ax^2 + bx + c = 0$ for any value of a, b, and c.

   In Matlab, you should not use a variable name in any calculation until you have assigned it a value using the "=" operator.

   For example, in the Matlab command window, type

   ```
   x = 2
   ```

   Then type

   ```
   x^2,    5*x,    10-x,    etc.
   ```

   You can just as easily change the value stored in the variable named x. For example, typing the following in the Matlab command window

   ```
   y = 5
   x = y+1
   ```

   will result in two variables called x and y storing the values 6 and 5 respectively.

   You can then try any calculation involving x and y - for example, $(x * y - 5)/2$.

9. Storing Commands - Script M-files: If you wish to store a sequence of Matlab commands so that you can re-use them, you can use an M-file *(so called because they end in the extension ".m").* There are two main kinds we will consider in this course - Function M-files and Script M-files. We will focus on Script M-files here (see the supplementary Matlab lecture for more on M-files).

   It is easy to use Matlab's in-built editor to write and save M-files. We will create a script M-file called tutorial1.m in which we will store some of the Matlab statements and commands we use in this tutorial. Recall, you should ideally be in your Tutorial1 subfolder in the G drive so that you always know where to find tutorial1.m.

   To open up a new script M-file, you can either

   - Click the <New Script> button on the Home tab, OR

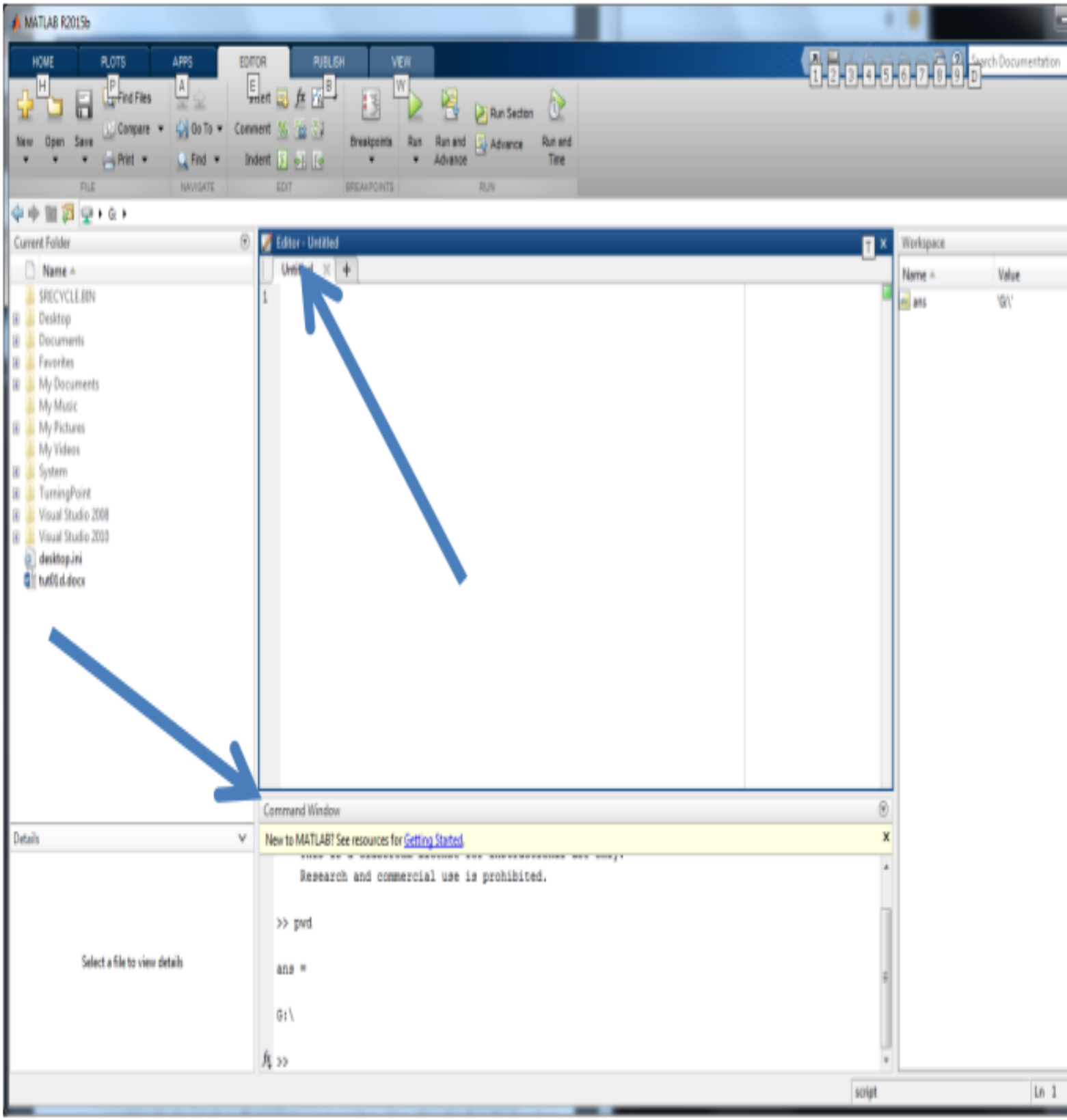
- type

  `edit tutorial1.m`

  in the Matlab *Command Window*.

→ Note: see the offical Matlab documentation on creating scripts for other ways of doing this. The two methods described above are sufficient for our needs.

- A new editor for your script M-file should now appear above your *Command Window* as shown in the following image

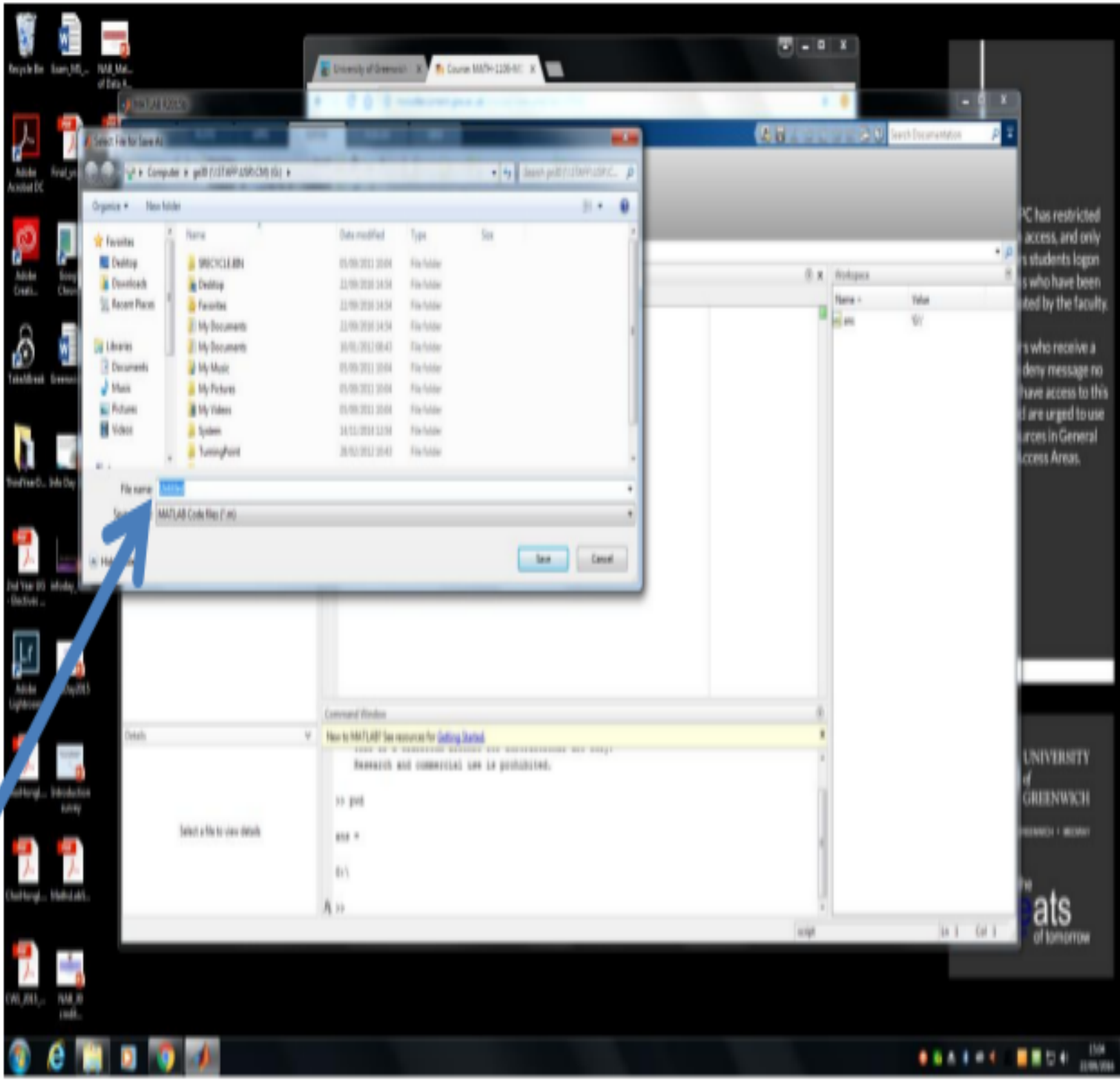


- If you opened the script M file by typing `edit tutorial1.m` in the *Command Window*, the script M-file will already have the desired name and you can edit it as you like and save your changes by clicking on the floppy disk symbol above the <Save> option while the editor is highlighted.
- Otherwise, if you opened the new script using the <New Script> button, the default name will be something like `Untitled1.m`. To rename it to `tutorial1.m`, select <Save As> from the drop-down

<Save> menu, change the name of the file to `tutorial1` on the little window that pops up, and then click <Save> on that pop up window. You can now make changes to and save this file `tutorial1` in the same way as in the preceding bullet point.

NOTE you should not add the ".m" when renaming the file - it is added automatically by Matlab.

See the image below for some indication of how to rename your script M-file.



10. Put your Matlab statements from Number 8 above into `tutorial1.m`. At the end, the contents of your script M-file should look something like this:

```
x = 2
x^2,    5*x,    10-x
y = 5
x = y+1
```

NOTE you can put several statements on one line separated by commas (or semicolons if you do not want to see the output of that statement) as is done in the second line above, but when writing programs it is often clearer to have each statement/command on a separate line.

- To run the script M-file `tutorial1.m`, simply type its name `tutorial1` at the command prompt *(i.e., the >> symbol)* in the Matlab *Command Window* then hit <Enter>, and all of the commands in `tutorial1.m` will run (assuming no bugs). NOTE you should not type the ".m" when running a script M-file this way. NOTE also that one of the advantages of this approach is that you can easily run the script M-file again by using the UP ↑ key to go back to the script running command in the *Command Window*.
- Alternatively, to run the script M-file `tutorial1.m` you can click the ▷ symbol in the top menu of the Matlab editor window containing the file `tutorial1.m`.
- However you do it, running the script M-file `tutorial1.m` will cause the results of the various statements in that file to be displayed, in order, in the *Command Window*. For example, first you will see the statement "`x = 2`" echoed back, then the answers to the three calculations involving x will be displayed with the name `ans`, then the statement "`y = 5`", then "`x = 6`".
- This output is a bit messy so I will show you how to modify it to make more sense by using the *disp()* command, which is a simple command for displaying information in Matlab, and also by using the semicolon at the end of some Matlab statements to supress their output. Compare the following to the above and see if you understand the role of the semicolon and *disp()*, as well as how to use them. Note also the introduction of three new variables to store the results of the initial set of calculations involving the variable x.

```
x = 2;
disp('Welcome to my program.  The value of x is')
disp(x)
xsquared = x^2;    fivex = 5*x;    tenminusx = 10-x;
disp('Squaring x, multiplying x by 5, subtracting x from 10 gives, respectively:')
disp(xsquared)
disp(fivex)
disp(tenminusx)
y = 5;
disp('The value of the variable y is ')
disp(y)
x = y+1;
disp('Updating the value of x by adding y to it gives x = ')
disp(x)
```

- When you run your program (script M-file) now, the output should make it clearer than before what your program is actually doing. *disp()* is quite basic and there is a more advanced/sophisticated

function for displaying information in Matlab called *fprintf()*. This is discussed in the supplementary Matlab lecture notes.

11. When naming Matlab files and variables:

DO

- Use names that make sense. For example, for a quadratic equation let the variable name a be the coefficient of x^2, b the coefficient of x and c the constant term. As another example, if you wrote a script M-file which implements the Newton-Raphson method, you could call it `NewtonRaphson.m`. See also the names chosen in Number 10 above for the three new variables used to store the results of the initial set of calculations involving the variable x.
- Pay close attention to uppercase versus lowercase *(i.e, capital letters versus common letters)* since Matlab is case-sensitive - meaning, for example, it would treat `MyVariable`, `myvariable`, and `myVariable` as three DIFFERENT variable names.

DO NOT

- Put any spaces within the name. For example, the variable name `my_name` is fine, but not `my name`. In general, where you want to put a space, use maybe an under-bar instead (as done in the `my_name` example).
- Use a name or symbol that is already an in-built Matlab function or operator. For example, do not include the operators $*$, $'$, $+$, etc. within the names of variables or files, and do not name a variable cos, sin, etc.

12. One of the key benefits of programming is that it allows you to do tedious computations quite easily. For example, to do a sum like

$$\sum_{k=1}^{10} k \quad = \quad (1+2+3+4+5+6+7+8+9+10 = 55)$$

one would typically use a *for loop*. See if you can understand how the following *for loop* works and how it computes the above sum. You can add these Matlab statements to the end of *tutorial1.m* or, if you prefer, you can create a new, appropriately named, script M-file and put these statements in it.

```
n = 10;

sum = 0;

for k = 1:n

     sum = sum + k

end

disp('The sum is ')

disp(sum)
```

Run this and read backwards in the *Command Window* to see if you can understand how the variable `sum` is being updated in the *for loop*. When you think you do understand, you can put a semicolon at the end of the line

```
sum = sum + k
```

so that running the program just prints out the final sum without all of the intermediate steps in computing it.

- NOTE the flexible way in which this program is written so that the upper limit of summation can be easily changed. For example, just by redefining

  `n=20`

  then running the program again, you get the sum $\sum_{k=1}^{20} k$ (which is actually 210).

13. Try to write a "program", incorporating the use of a *for loop*, which computes and then prints out the following sum (again you may put this at the end of one of your existing M files - such as *tutorial1.m*, or if you prefer you can put it into a separate M file):

$$\sum_{k=1}^{5} 2k^3.$$

HINT: You can copy the above program for computing $\sum_{k=1}^{10} k$ *and modify it slightly to compute* $\sum_{k=1}^{5} 2k^3$ *instead.*
This is idea of using a previously-written program and just modifying it to do something different, is a recurring theme in programming.

SOLUTIONS

```
n = 5;
sum = 0;
for k = 1:n
     sum = sum + 2*k^3
end
disp('The sum is ')
disp(sum)
```

→ **The actual sum is** 450 ←

---

14. HOMEWORK Begin looking at the supplementary lecture notes on Matlab and start trying some of the examples. Also, try accessing Matlab from a home/laptop computer using the virtual desktop - as described in Number 1 above. Email me at `E.George@gre.ac.uk` if you have any difficulty with this. I also recommend looking at the tutorial and introductory Matlab material at

    `http://uk.mathworks.com/academia/students.html?s_tid=acmain_sp_gw_bod`

    or the interactive introduction (pdf file and youtube videos) at

    `http://www.eng.ed.ac.uk/teaching/courses/matlab/`

    or the first lecture at

`http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-094-introduction-to-matlab-january-iap-2010/lecture-notes/`

    (and the later lectures for more advanced material)

Closing Remarks on Octave and Python

Octave

This is a free *Matlab* clone which you can read about, download, and install from **https://www.gnu.org/software/octave/**. Being a "clone" means that you can use *Octave* almost exactly like you can *Matlab*. For example, M-files written in *Octave* will also work in *Matlab*, and vice versa (with a few exceptions). Even the interface of *Octave*, with a command window and internal editor, is very similar to that of *Matlab*. While you can always access the University's version of *Matlab* via the **Virtual Lab Desktop**, having *Octave* installed on your laptop (or home computer) will allow you to write and enjoy Matlab programs *even if you do not have access to the internet*. What joy!!!

Python

This is a powerful general-purpose programming language which is consistenly ranked as one of the leading and most popular programming languages in the world. It is **NOT** the language we will be using in this course but it is likely that it will be used in this course in future years. **For students who are already comfortable with Matlab**, it might be worth trying to learn Python as well during this course. For such students, I would recommend forming a group and working together to learn the language. There will be no formal classes on Python but I will be putting up some Python solutions to tutorial programming exercises in a Python folder on the MATH1134 Moodle pages. You can ignore that Python folder if you plan only to use Matlab. There are a few things to note:

- I will also accept Python programs for any coursework question which asks for a Matlab program.
- Those wanting to learn Python should focus on the versions 3.x.y strand of Python, not the 2.x.y versions.
- Python is free and is typically already installed on computers (such as these lab computers). HOWEVER, I have specifically asked for the **Anaconda** distribution of Python to also be installed in these lab computers and on the **Virtual Lab Desktop**, and if you want a personal copy of Python I would also strongly recommend that you download and install that particular distribution of Python **(for free!)** on your personal computer. This is primarily because the **Anaconda** distribution of Python comes pre-installed with several very useful libraries which we would otherwise have to install on the University's default Python installation (which is not a straightforward exercise). Specifically, the *Numpy*, *Scipy*, *Matplotlib*, and *Sympy* libraries come with the **Anaconda** distribution of Python, and these libraries are very useful when programming the type of approximation methods seen in MATH1134. For more information on **Anaconda** and for links on downloading it, go to https://www.anaconda.com/what-is-anaconda/
- You can open **Anaconda** in Windows by searching for the ``Anaconda Navigator''. From the navigator, click on *spyder*, which stands for *Scientific Python Development Environment*. The development environment which opens up is sufficiently similar to the Matlab development environment that I will leave it to your intuition to figure out how to use it.
- General Python tutorials and help for learning Python programming are all over the internet. A good place to start is https://docs.python.org/3/
- Specific information on installing **Anaconda** can be found at https://docs.anaconda.com/anaconda/install.html and information on using **Anaconda** can be found at https://docs.anaconda.com/anaconda/ (starting with the *Anaconda Cheat Sheet* linked on that page).