Tutorial Sheet: MATLAB

This tutorial will help you to practice the use of MATLAB for applied mathematics. You may use the command line to test commands, but you should save your work in a script or a Live Script so that you can refer to them later.

Matlab Live Scripts are particularly helpful as they allow you to see the code, your commentary, *and* all of the output (e.g. graphs) at the same time. To do this, click "New">"Live Script", and then make sure to save it. Type your commands in the grey boxes of the script (one per line), and add comments using the percentage sign % or by clicking "Text" so that you can explain to yourself what each command does and also to label "Q2(b)" etc... To execute all of the commands in your script line-by-line, click "Run".

1. **Numeric and symbolic variables - what's the difference?**

   MATLAB makes a distinction between *numeric* variables, which are numbers stored to a high-precision decimal, and *symbolic* variables which are used for symbolic algebra (differentiation, integration, etc.) and store the numbers as "precise" such as fractions.

   (a) Type `1/3` in the command window, and press enter. What is the output?

   (b) Now, type `sym(1/3)` Do you see a different result?

   (c) Lets's store some variables. Type `x = 1/3` Is x numeric or symbolic?

   (d) Type `y = sym(1/3)` Is y numeric or symbolic?

   (e) We can obtain the value of a symbolic variable as a decimal approximation. Type `double(y)`

   (f) Type `clear` . What has this command done?

   (g) Store 2/3 in Matlab as a decimal approximation, stored as a variable called z.

   (h) Obtain the value of z as a precise fraction.

   (i) Store 4/3 in Matlab as a precise fraction, stored as a variable called w.

   (j) Obtain the value of w as a decimal approximation and store it in a variable called $w2$.

   (k) We can declare a variable as symbolic without giving it a specific value. Try typing `syms p`

2. **Calculations:**

Calculate the following using MATLAB commands. Remember that MATLAB will use the order of operations (BODMAS) when evaluating calculations, so you may need to use brackets!

You may choose whether to give your final answers as precise values (using symbolic variables) or as decimal approximations, but ensure that you know how to work with both.

(a)

$$15 + \frac{39}{2}$$

(b)

$$\frac{15+39}{2}$$

(c)

$$k = 6 \times 104 - 2^{-3}$$

(d)

$$\cos(\pi)$$

(e)

$$y = \left(\mathrm{e}^{-3} + \sqrt{19}\right)^{-2}$$

(f)

$$f(x) = \int 4x + 1 \mathrm{d}x$$

(g)

$$\int_{x=0}^{x=2\pi} \sin(3x) \mathrm{d}x$$

(h)

$$M = \int_{t=-5}^{t=3} t \cdot \cos\left(\frac{2t}{\pi}\right) \mathrm{d}t$$

(i)

$$y = \int_{x=-\pi}^{x=\pi} \left(\frac{3x^2+2x-7}{3+\sin(2x)}\right) \mathrm{d}x$$

3. **Graphs and plotting, Part 1:**

Lets's create a simple plot of $\sin(x)$ and $\sin(2x+1)$ on the same graph and customise it.

(a) To use the simpler version of curve plotting, we will need to use "symbolic variables". First, declare x as a symbolic variable.

(b) Then create our two dependent variables:

$$y1 = \sin(x) \quad \text{and} \quad y2 = \sin(2x+1)$$

(c) Use `fplot` to plot a graph of $y1$ against x.

(d) What command is needed to plot another function on this same graph?

(e) Plot $y2$ against x on the same graph.

(f) I am particularly interested in the behaviour of these functions in the range $2 < x < 6$. Change the limits of x so that this is what is in view.

(g) As you can see, it turns out that `fplot` only draws the graphs between $-5 < x < 5$ by default. To correct this, we will need to start again. First, call `hold off` so that the next plot replaces everything we have done so far.

(h) To re-plot $y1$, drawing it between $2 < x < 6$, we can add an argument to the fplot command that contains the lower and upper limits for x:

```
fplot( x , y1 , [ 2  6 ] )
```

(i) Re-plot $y2$ against x over the same range on this same plot.

(j) Label the x-axis as "x", and the y-axis as "y".

(k) Add a legend, calling $y1$ and $y2$ "Data set 1" and "Data set 2" respectively.

4. **Graphs and plotting, Part 2:**

This version of graph plotting uses discrete numeric data rather than symbolic variables. It can be a little more complicated.

A medical student decides to record my sugar intake (in kg) at the same time on ten consecutive days. This results in a set of ten data points, with an independent variable - time t (in days) - and a dependent variable - mass M of sugar (kg).

(a) Input these two variables as column vectors by typing the commands below:

```
t = [1 2 3 4 5 6 7 8 9 10];
M = [0.573    0.366    1.900    0.841    0.155
          1.910    1.187   0.040    1.703    1.411];
```

(b) Experiment to find out the significance of including the semicolon at the end of the command.

(c) Create a scatter plot of these data points.

(d) The student hypothesises that this data may fit a pattern of $M = \cos(2t) + 1$ (note that we could determine the values $a = 2$ and $b = 1$ from $M = \cos(at) + b$ using the curve fitting techniques that we have done previously!)

Create a new variable $Mpred = \cos(2t) + 1$ and plot it as a curve on the same graph.

(e) As you can see, this fitted function does not seem like a smooth cosine curve. This is because Matlab has determined the value of $Mpred$ on each of the 10 days and then `plot` has simply connected the points. To view how these data compare to a better drawing of $\cos(2t) + 1$ we need to do the following:

```
t2 = linspace( 1 , 10 , 1000 )
```

This creates a vector of 1000 evenly-spaced data points in the range $1 \leq t \leq 10$ that we are looking at.

```
Mpred2 = cos( 2 * t2 ) + 1
```

This determines the predicted values of M for each of those 1000 new values of t.

```
plot( t2, Mpred2 )
```

Assuming you already used `hold on`, this superimposes this third plot onto our image.

This should look much more accurate now!

(f) Add a legend to the graph, naming the three plots, and appropriate labels on the x-axis and y-axis.

(g) If you have further time during the tutorial, you may wish to experiment using the additional customisation options for the `plot` and `scatter` commands. You can find information on these using the comprehensive Mathworks website:

`https://uk.mathworks.com/help/matlab/ref/plot.html`

Finally, you may wish to save this customised image. Find out how to do that, choosing the size, name, and file type (PNG, JPEG, etc.) here:

`https://uk.mathworks.com/help/matlab/ref/print.html`