Worksheet: An Introduction to MATLAB

The aim of this worksheet is to give you an introduction to some of the elementary features of MATLAB, including basic commands, plotting, loops and the manipulation of vectors. These commands will all come in useful when you come to write your own programs.

1. Use MATLAB to do the following simple calculations

$$(i)\quad 1+2+3 \qquad (ii)\quad \sqrt{2} \qquad (iii)\quad \cos(\pi/6)$$

N.B. The pre-defined variable `pi` is a numerical approximation of π. The result of the calculation gets saved in a variable called `ans`. To recall the last value, type `ans`.

2. For (*iii*) above, type `ans^2` to compute $\cos^2(\pi/6)$. Now subtract this from the exact answer of 3/4. You should find that the result is not zero, but is very very small $\sim 10^{-16}$. Why is this? N.B. To get an answer presented to more decimal places, type `format long`. Other options you could experiment with include: `format short e`, `format hex`, `format bank` and `format rat`.

3. Variables. Often, you may wish to save the result of a calculation for later use. To do this, you should assign the result to a variable. For example, try typing:

```
>> myvar = 3*pi*log(2);
>> myvar
```

The first line assigns the value $3\pi\ln 2$ to the variable name `myvar`. The second line just recalls the value of `myvar` and writes it to screen. Note that adding a semicolon `;` to the end of an expression suppresses the screen output.

4. Vectors. Matlab handles vectors (and lists of numbers) very naturally - this is one of its great strengths.
(a) Generate a vector of even integers from 0 to 20 by typing `x = [0:2:20]`. Do you understand this expression? Try creating some other lists of numbers.
(b) The third element of `x` can be found by typing `x(3)`. Do this. What do you think `x(7)` is? Check your answer.
(c) We can create other vectors from `x`. Try the following operations:

$$(i)\quad \texttt{sin(x)} \qquad (ii)\quad \texttt{x.\^{}2} \qquad (iii)\quad \texttt{x.\^{}(-1)}$$

In (ii) and (iii) the decimal point after `x` means that the operation is applied *pointwise* to the elements of `x`. Note that MATLAB set 1/0 to `Inf`.

Make sure you understand how *pointwise* operation works. For example, try predicting the result of entering

```
[0 : 1 : 5].^2
```

then check your prediction.

5. Simple plotting. Plotting in MATLAB is easy. To plot the components of a vector `x` type `plot(x)`. To plot the vector `y` against `x` type `plot(x,y)`. To plot this with green crosses type `plot(x,y,'g+')`. Simple!
(a) Generate the vector `x = [0:0.01:10];`. Remember, the semi-colon stops the vector being printed out – preventing a lot of messy data appearing on screen. Now construct the vectors `y = sin(pi*x)` and `z = cos(3*pi*x)` (adding semicolons at the end, as before). Plot the vector `y` against `x`. To keep this figure and add another graph over the top, type `hold on`. Do this and plot the vector `z` against `x` in green. Now type `hold off`.
(b) Plot a pretty Lissajous figure by plotting `z` against `y`.
(c) Typing `help plot` will tell you much more about the way to use the plot function, and the alternative options.

6. Loops. In implementing most numerical methods you will need to repeat a simple set of instructions over and over again. MATLAB achieves this by using a `for <statements> end` loop. Here is a an example of a loop to print the squares of the first 10 numbers:

```
for k = 1:10
k ^ 2
end
```

(a) Type in this set of instructions and check that it works.
(b) Modify the loop so that it types out the cubes of the numbers between 11 and 20.

7. Time series. Use function findpeaks to find values and locations of local maxima in a set of data. The file spots_num.mat contains the average number of sunspots observed every year from 1749 to 2012. The data are available from NASA. Find the maxima and their years of occurrence. Plot them along with the data.

```
load sunspot.dat
year = sunspot(:,1);
avSpots = sunspot(:,2);
[pks,locs] = findpeaks(avSpots);
plot(year,avSpots,year(locs),pks,'or')
xlabel('Year')
ylabel('Number')
axis tight
```

Can you estimate how many such peaks were observed per each 50-year period?

8. Images. Read an image into the workspace, using the imread command. The example reads one of the sample images included with the toolbox, an image of a young girl in a file named pout.tif , and stores it in an array named I .

```
I = imread('pout.tif');
mshow(I)
```

Check how the imread function stores the image data in the workspace, using the whos command: `whos I`

The image pout.tif is a somewhat low contrast image. To see the distribution of intensities in the image, create a histogram by calling the imhist function. (Precede the call to imhist with the figure command so that the histogram does not overwrite the display of the image I in the current figure window.) Notice how the histogram indicates that the intensity range of the image is rather narrow. The range does not cover the potential range of [0, 255], and is missing the high and low values that would result in good contrast.

```
figure, imhist(I)
```

Improve the contrast in an image, using the histeq function. Histogram equalization spreads the intensity values over the full range of the image. Display the image.

```
    I2 = histeq(I);
figure, imshow(I2)
```

Call the imhist function again to create a histogram of the equalized image I2. The histogram of I2 is more spread out over the entire range than the histogram of I .

```
figure, imhist(I2)
```

Write the newly adjusted image I2 to a disk file, using the imwrite function and '.png' format.

```
imwrite (I2, 'pout2.png');
```

Check the contents of the newly written file. The imfinfo function returns information about the image in the file, such as its format, size, width, and height.

```
imfinfo('pout2.png')
```

9. M-Files By now, it is becoming quite a chore to have to re-type all the lines of your program every time you just want to make a trivial change. Instead, you can store many lines of code (your "program") in a *script file*. These are files with the extension `.m`. Try creating a new script file called `pout.m`. To do this, right-click in the panel on the left of the Matlab window, and select `New -> M-File`. Then copy and paste the code from (8) straight into this file.
(a) Try running the file by typing its name at the command prompt: `>> pout`.
(b) Change the code to cut larger or smaller part of the image each time the program is run. To do this, replace the second line of the program with:

```
n = input("Please enter the number of pixels :  ");
```

(c) At the top of the script file, add a few comments describing the program. A comment is human-readable text that MATLAB ignores. A comment must be preceded by a %. It is good practice to comment your code thoroughly. Try adding something like:

```
% CELL: Written by <yourname> on <date>
```

Now, try entering `help cell` at the command prompt.

10. MATLAB introduction. Finally, if you have time, try the built-in introductory session by typing `intro` after the prompt. Further information on MATLAB can be obtained by typing `help` and a demonstration of what it can do can be obtained by typing `demo`.