NST 2013-14: Part IA Scientific Computing

Practical Session One (Lectures 1 & 2)
Introduction to MATLAB and Publication Quality Graphics

October 2013

1 Example Sheet

This is the first of three practical sessions which accompany your IA Scientific Computing course.

The session is intended to get you started in learning MATLAB. You should work through the tasks given below, in conjunction with your lecture notes and reading of the course book. We actively encourage you to work with your colleagues and discuss the material. If you get stuck, you can ask the demonstrators for help, but you should make sure you have thought about the problem yourself, first.

If you get on quickly, you can work through the **Assessed Task** (section 2 of this document) during the practical session, but this **must be done on your own**. Instructions for submitting your report and are given in Appendix A at the end of this document.

1.1 Opening & Navigating MATLAB

Start MATLAB in the same way as you would any other program (it is a bit slow to load under Microsoft Windows, usually due to antivirus software checking lots of files as it loads, but it is fast once it has loaded).

Information about installing MATLAB on your own computer, using the Cambridge University student site licence is in the course book preface.

Identify the following windows on the desktop:

(a) command window (where you type commands)

(b) editor (where you write a program of commands)

(c) current folder

(d) command history

(e) workspace

Make yourself a directory in your 'My Documents' folder or in another convenient place (depending on what computer you are using), where you can store any files you create. Change MATLAB to that directory, either by using the "Current Folder:" buttons at the top of the desktop, or typing `cd my_directory` in the command window.

1.2 Using MATLAB as a simple calculator

For more information about mathematical operators and constants that are built into MATLAB, see pages 6 to 8 of the course book.

The simplest way to use MATLAB is by typing commands into the command window. For example:

```
>> 2+2
ans =
     4

>> 3.3^2
ans =
     10.8900

>> sin(pi/2)
ans =
     1
```

Tasks:

(a) Calculate 478*0.26378 [answer: 126.0868]

(b) Calculate 2^{24} [answer: 16777216]

(c) Calculate the base 10 logarithm of 1200 [answer: 3.079]

(d) Calculate the tangent of 60 degrees [answer: 1.7321]

(e) Calculate the square root of -20 [answer: 0 + 4.4721i]

(f) Use the up-arrow key to find a previous command and re-run it with different values.

(g) Cut and paste one of your commands from the command history window back into the immediate window.

(h) Select a command in the command history and press F9 to re-run it directly.

(i) Change the display precision by typing **`format long`**, then repeat some of the above calculations to see the effect. You can change it back again by typing **`format short.`** Note that MATLAB always works internally to the high precision.

1.3 Using Variables for Calculations

For information about variables, see pages 7 to 10 in the course book.

Usually, it is easier to assign values to a variable, in order to use them in a calculation:

```
>> a = 1
a = 1

>> b = 2
b = 2

>> c = a + b
c = 3
```

Tasks:

(a) Set the value of the variable x to a some value, then calculate $x^3 + 2x^2 + 3x + 4$.

(b) Repeat the above using a few different values of x (e.g. by using the up-arrow key).

(c) Put both the assignment and calculation on the same line, by separating them with a comma.

(d) Keep both commands on the same line, but now also surpress the output from the assignment of x, by separating them with a semicolon instead.

1.4 Working with Lists and Arrays

As described in the lectures and course-book, MATLAB is powerful as you can organise data into lists, matrices and arrays of various shapes. You can then operate on entire lists or matrices in one go. For example:

For more information about lists, arrays and matrices, see pages 10-14 in the course-book.

```
>> a = 3;
>> b = [1 2 3 4];
>> c = [1:1:10]
c =
     1   2   3   4   5   6   7   8   9   10

>> d = [1 2 ; 3 4]
d =
     1   2
     3   4

>> a + b
ans =
     4  5  6  7

>> e = 2*c
e =
     2    4    6    8   10   12   14   16   18   20

>> f = d.^2
f =
     1     4
     9    16
```

Note that the “dot” operator (e.g. “`.*`” or “`.^`”) operates elementwise on the list, rather than doing “matrix multiplication” operations (this will be taught later in the IA Mathematics course).

It is easy to select part of an existing matrix using MATLAB, which is useful if you want to analyse only part of a larger piece of data. For example:

```
>> e(3)

ans = 6

>> e(1:3)

ans =
     2     4     6

>> e(5:end)

ans =
    10    12    14    16    18    20

>> f(:,2)

ans =
     4
    16
```

Tasks:

(a) Set the variable y to a list of numbers from 1 to 100, in steps of 5

(b) Take the square root of this list of numbers using the **`sqrt()`** command. Repeat the same operation, but this time using the **`^`** operator. Check the behavior with a couple of other functions, such as **`exp`** and **`log`**.

(c) Encode the matrix

$$z = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 5 & 6 & 7 \end{bmatrix}$$

(d) Use the **`inv()`** function to find the inverse of z (you will learn more about other matrix operations later in your Mathematics course).

(e) Use two different ways to make a column list containing the same numbers as the row list c in the example above. *Hint: try using the transpose operator.*

(f) Make a variable $z2$, by selecting the top left 2×2 part of z.

(g) Make variable $z3$ by selecting all the elements in the 3rd column of z.

1.5 Plotting Simple Graphs

Plotting graphs is accomplished in MATLAB by using the plot command. For example,

```
>> x = [0:1:10];
>> y = x.^2;
>> plot(x,y);
>> xlabel('This is the abscissa (x axis) label')
>> ylabel('This is the ordinate (y axis) label')
>> title('Title of my graph')
>> hold on
>> plot(x,2*y,'bo')
>> hold off
```

For more information about simple plotting, see pages 14 to 16 in the course book.

Tasks:

(a) Create a row-list (also called a row-vector) of numbers from -10 to 10 in steps of 0.5 and assign it to the variable x.

(b) Use your variable x to plot a graph of $x^3 + 2x^2 + 3x + 4$

(c) Label the axes and give the graph a title.

(d) Type **`help plot`** and read the notes. Use the information to repeat the plot, but by plotting diamond markers in red. Notice how the new plot replaces the old plot.

(e) Use the plot command, in conjuction with the **`hold on`** and **`hold off`** commands to repeat the cubic plot above, but this time make 3 line plots, in different colours, to show the effect of varying the quadratic coefficient between 1 and 3.

1.6 Using Online Help

MATLAB is a huge program and it is impossible to remember the details of all the commands. Using the built in help is an important part of working with MATLAB, and it is important to get used to finding out how to learn about new features yourself.

- The `help` command provides basic information in the command window.
- The `doc` command provides more comprehensive information in the normal documentation browser. You can also start the help browser from the help menu, or by pressing F1.
- In MATLAB, functions are deliberately named "sensibly", so you can often guess the name of a command from what you want it to do.
- There are also lots of examples available online, and specifically at the MATLAB File Exchange (at http://www.mathworks.co.uk/matlabcentral).

Tasks:

(a) Type `help tan` to get basic help about the tangent function (obviously this is just a convenient example!).

(b) Type `doc tan` and compare the amount information that is provided.

(c) Use the help browser to find out about the other trigonometric functions in MATLAB. What is the difference between `tan` and `tand` ?

(d) Find the 'Getting Started' section of the MATLAB help. This section contains some alternative tutorials, that might be helpful if you have some free time.

1.7 Using the Editor to Make Scripts

Normally MATLAB is used by putting together scripts of commands in the editor. M-files are text files (normally named with a `.m` extension) which list a series of commands for MATLAB. You can edit these in the MATLAB editor, or any other text-editor program.

Tasks:

(a) Copy the commands you have just used to plot the three cubic lines, from the command history into the editor window to make a script. Make sure no errors are introduced.

(b) Save the script to a `.m` file, and give it a name (e.g. `firstplot.m`)

(c) Run the file, either by typing its name into the command window, by pressing the "run" button (or the F5 key) in the editor window.

1.8 Working with Files

MATLAB can load up many types of files containing scientific data. The simplest forms are text files, which just contain a series of characters. As described in the lectures, CSV files are a frequently used form of text files where the data element in each row of the file are separated by commas.

For more information about loading in different types of files, see pages 17 to 22 of the course-book.

For example

```
>> data = dlmread('file.txt',',',1,0);
```

Tasks:

(a) Locate the file **`test.csv`** which is available on CamTools and put a copy of it in your working directory. Open it in notepad (or a similar text editor) and examine the contents (but not Excel, which might want to open it by default!). You can also load it into the MATLAB editor, but you will need to rename it to **`test.txt`** first, otherwise MATLAB will try and import the data immediately. Which are the header lines, and which lines contain data? Think about how you can could tell MATLAB to distinguish the two (there are several ways).

(b) Load the file into MATLAB using the graphical tools; either drag and drop the file onto the MATLAB window, or choose File Menu > Import Data... from the menus. Plot the first column of numbers against the second column of numbers. Save the plot as a MATLAB figure file (.fig extension), as we may want to use it again.

(c) Use the **`dlmread`** command to load the same file in, without using the graphical window. Use the built in help to work out what the last three arguments mean

(d) Whenever you are working with MATLAB, you should now be thinking about making a script of commands. Write a script containing the above dlmread command, to read in the file automatically, then make a line plot of the 2nd column (on the y-axis) against the 3rd column (on the x-axis).

(e) Modify the same script to load in and plot the data stored in **`iqdata.csv`** (also available on CamTools). You should plot the second and third columns (on the y-axis) against the first column (on the x-axis).

(f) MATLAB has many different commands for reading and writing. Start the online help and do a search on “File Formats”. If you needed to, what would be the best command to try and use to load data from (i) an Excel spreadsheet file, (ii) a Jpeg image file?

1.9 Publication quality graphics

For more information about plotting and refining graphics, see pages 27 to 40 of the course-book.

Within MATLAB, you can both use interactive “point and click” tools, or command line commands to draw and refine plots, then export your figure for use elsewhere. Commands are usually preferable, as it is then easy to redraw and refine plots later by re-running the script.

```
>> plot(x,y,'--k','LineWidth',2)
>> xlim([0 10])
>> ylim([-1 1])
>> axis([0 10 -1 1])
>> legend('This is line \alpha')
>> print -depsc2 'myfigure.eps'
>> print -dpng -r300 'myfigure.png'
```

For more detail about using get and set, see pages 29 to 31 of the course-book.

The **`get`** and **`set`** commands can be used to modify the properties of plots (as well as other objects in MATLAB). In each case, the first parameter is a “handle” which refers to the object you want to modify. The **`gca`** command provides the handle to the current axis (it stands for “get current axis”). You can also set parameters at the same time as you make a plot. For example:

```
>> get(gca)
[ list of settable options appears]
>> set(gca,'fontsize',12,'linewidth',2)

>> p = plot(x,y,'--k','LineWidth',2)
>> set(p, 'Linewidth',2,'Marker','o','LineStyle','--')
```

Tasks:

(a) Take the script you wrote in the last section and re-run it to generate a plot of the data stored in **`iqdata.csv`**. Use MATLAB's graphical 'PlotTools' to improve the plot (search the built-in help for "plot tools" if you need to). You should (i) adjust the x and y range of the plot, (ii) give axis labels, and adjust the font size, (iii) change the marker type and colour. Save the figure as a MATLAB .fig file, and check you can load it up again. It should hopefully be obvious that this procedure is quite convenient for one-off plots, but is quite laborious if you have to process many files.

(b) Repeat the same procedure, but this time use only the command line. You will need the **`xlim`**, **`ylim`** or **`axis`** commands, as well as **`set`**. Put all the commands in a single script (.m file), and save it, in case you need to use it later.

See pages 27-29 of the course book.

(c) Add a legend to describe the line using the **`legend`** command.

(d) Often it is useful to use Greek characters in figure labelling. MATLAB allows you to use simple LaTeX syntax to add mathematical terms to your plots (e.g. \pi, \Theta, etc.). Use this facility to add some greek letters to labelling of your figure.

For more information about LaTeX, see the course book appendix.

(e) Export your figure as (i) an EPS file, (ii) a PNG file and (iii) a JPG file. Load your exported files into *Adobe Photoshop* or a similar program and zoom in on the image. Note the pixelation of the bitmapped files, compared to the vector format. Can you see "lossy compression" artefacts in the JPEG image, as discussed in the lectures?

See pages 34-35 of the course book.

(f) Logarithmic plots are widely used in scientific computing. In MATLAB, these can be easily made by using **`semilogx`**, **`semilogy`** or **`loglog`** instead of the usual **`plot`** command. Make a logarithmic plot of the function $y = \exp(-x)$, so that it appears as a straight line.

See pages 37-40.

(g) Plots with error-bars can be made using the **`errorbar`** command. Read the built-in help information about making errorbar plots. Now, take the three columns of data in **`test.csv`** as describing x, y and the error in y, respectively, and make an errorbar plot of the data.

See page 37.

(h) Find the 'Graphics' section of the MATLAB users guide. What commands would you need to use, if at some later date you needed to make (i) pie charts, (ii) histograms, (iii) polar plots?

NB: you can also examine any object in MATLAB using the graphical object inspector, which is useful for finding out which parameters you can modify. For example, type **`inspect(gca)`** to inspect the current axis.

1.10 Some more sophisticated operations

If you progress well, you might like to try the following:

(a) In the CamTools site you will find some command (.m) files that do not work (from 'Resources', select the 'Scripts' folder, and locate the files **`faulty1.m`** and **`faulty2.m`**). Run these, and locate and then correct the errors. Debugging, as this process is called, is one of the essential skills of programmers.

(b) Look up the **`sum`** function, and observe the outcome of the operation **`sum(a)`** when **`a`** is either a single list of numbers or a two-dimensional array (matrix). How can you use this function to perform the sum of all elements of a two-dimensional array?

(c) Find out how to calculate the factorial of a number. One function that you may meet in science is $\log N!$. Show graphically that the function $N \log N - N$ is a good approximation to $\log N!$.

2 Assessed Task #1

The iPod was launched on October 23rd 2001 at a price of $399. Suppose that instead of buying an iPod on that date, you had invested the retail price in shares of either Apple, IBM or Microsoft. Write a program that will plot a single graph containing three lines, which compare the total value of your three possible investments since then. (For simplicity, you should ignore the fact that shares are usually bought in whole number incremements.)

Figure 1: The original iPod, launched on October 23rd 2001 and costing $399.

You can download tables of monthly share prices for the three computer companies, IBM, Microsoft and Apple, as CSV files from the CamTools course site (click on 'Resouces' in the left-hand sidebar, and locate the three .csv files within the 'Data for Practicals' directory). Share prices are given in US $, and the time data are given as a year value that is expressed as a decimal (for example, July 2003 would be expressed as 2003.5). Note that the launch data corresponds to the data in row 144 in each data file (when the Apple share price was $10.65), where the header row corresponds to row 0.

Use the techniques discussed in the lectures and course-book to present your plot clearly and write a very brief report, as detailed in the assessed task notes. The key issues to address are

(a) Ensure that the chart is both clear and legible, with appropriate font sizes for the axes.

(b) Add appropriate axis labels.

(c) Refine the range of data over which the graph is plotted, commensurate with the the purpose of the plot.

(d) Give today's value of your $399 investment in shares in the three companies.

(e) Give a conclusion concerning whether your investments would have been wise or not.

Submit your work as described in the assessed task notes, at the end of this document.

A Appendix: Assessed Task Notes

A.1 Assignments

There are three assessed assignments, to be submitted following lectures 2, 4 and 6. The assignments typically involve writing a small MATLAB program to manipulate some data in order to generate a plot or chart and to give a specific number.

You will be expected to submit two files. The first is a report that should contain a copy of the visualisation, any other piece of information you have been asked to generate within the assignment, and a copy of the program you wrote as an appendix. The second should be the .m program file that you created during the assignment.

A.2 Grading scheme

The primary purpose of the exercise is to generate a visualisation and a small piece of information using a MATLAB script, and all marks reflect this. Each assignment has a maximum mark of 2, to be awarded for a visualisation that matches the quality described in the lectures, a working MATLAB script, and a correct piece of extracted information. A mark of 1 will be given for a partly-completed task. 1.5 will be given for a piece of work that is broadly correct but falls short in some regard, and 0.5 will be given for a poor attempt. 0 will be given for non-submission or submission that indicates little effort and no success. We expect most students will achieve the maximum grade.

Your grade will be returned to you via CamTools.

A.3 Requesting help

The purpose of the assessed assignments is to encourage you to put the required work into the course (and not many months later) so that you gain experience in the use of MATLAB and learn good practices in scientific computing. The goal is not assessment per se, but the successful accomplishment of the tasks. For this reason we encourage seeking help if you are stuck, and to this end we run practical sessions to facilitate discussion. We do not encourage wholesale exchange of program scripts of course, because if someone else effectively does the assignment for you, you will have lost a learning opportunity, and this won't be helpful in the long run.

A.4 Generating the report and program file

You are asked to submit a short report for each assignment, which typically should be around one or two pages maximum length. We would like to discourage you from writing extended prose because the marking scheme doesn't have the scope to reward prose – all marks are only to be assigned for the quality of the visualisation, the plausibility of the Matlab script, and the correct generation of the required information.

The report should be in PDF format only; other formats such as Microsoft Word will not be acceptable. Both Microsoft Office 2010 and OpenOffice/LibreOffice can be used to generate PDF files directly, or you can install a PDF generating package such as CutePDF or Acrobat Distiller (which is also provided on every Desktop Services / PWF computer). On a Mac, you can print to a PDF file through the Print menu. If you are using LaTeX instead, by using pdfLaTeX you will automatically generate the PDF file.

Your MATLAB program script (a .m file) will be run by the person marking the work, to ensure that it gives the result and graph reproduced in the report.

A.5 Submission of work

You need to submit your work through the course CamTools site. Click on the "Assignments" link on the left. You will see links to the various Assessed Tasks as they are released. The important information against any task is the "Due" date.

To submit a task report, click on the "Submit as Student" link. This will bring up a page with more information. Click on the "Add Attachments" button. This will bring up a page that will ask you to upload a local file. To do so, you click the "Choose file" button, and browse to find the required file, click to select it, and press the “Choose” button. You need to do this for both the report and the file containing the program script.

When you are sure you have uploaded all documents, press the “Submit” button on the assignment page. Until you press this button, you will not have actually submitted your work, so it is essential that you ensure that your work has been submitted. CamTools will not automatically submit your uploaded work, because it wants to give you chance to replace your submission if you need to. **On pressing the “Submit” button you will be shown a confirmation page giving a submission ID number and will be sent an email containing the same information. If you do not receive this email, it is likely that you have not actually submitted your work.** You can also use CamTools to check the status of your assignments, and this should also tell you whether you have properly submitted your work.

Remember not to submit your assignment work until you have uploaded both the report and the program script file.

A.6 Submission Checklist

For each assessment, you are required to submit, through CamTools before the due date (given in CamTools in the Assignments section):

(a) A very brief report (a single document in PDF format) which should contain:

1. Your name, college, and CRSid.
2. An appropriate title.
3. A copy of the figure you have generated.
4. An appropriate figure caption.
5. The answers to any questions asked in the assignment.
6. An appendix containing your MATLAB code (cut and pasted).

(b) A copy of the .m program script used (e.g. to plot figures, perform analysis, etc.)

A.7 Deadlines

The deadlines for submission of each assessment are:

Assignment 1: 23:55 on Tuesday 5th November 2013.

Assignment 2: 23:55 on Tuesday 19th November 2013.

Assignment 3: 23:55 on Sunday 19th January 2014.