NST 2013-14: Part IA Scientific Computing

Practical Session Two (Lectures 3 & 4)
Minimisation, Fitting and Statistics

October 2013

1 Example Sheet

In this second of three practical sessions accompanying your IA Scientific Computing course, you will get to practice on the concepts introduced in Lectures 3 and 4, i.e., on Minimising and fitting, and on Statistics.

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.

You will see that the beginning is much more directed than the last tasks, for you to get used to find out for yourself. Every task is exemplified in the course book, but we encourage you to explore the MATLAB and on-line documentation first, (but avoid being stuck on a problem for too long). Consider the tasks proposed here as recommendations, and select the ones you do depending on how long it takes you. Make sure, however, that you have a substantial amount of time to face section 1.4 (it seems shorter, but it is not).

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 Minimising a function

Many situations in science require the finding of minima of a function. We will explore here ways of doing that with MATLAB. First, how MATLAB allows you to define functions (useful for many other purposes), and then how to find their minima. For that we will be using `fminsearch`, which uses a so-called simplex algorithm to search for minima.

There are other ways to face this problem, `fminsearch` being a general and useful tool but not the only one.

Although many commands in MATLAB are very robust, there may be situations in which they break (e.g. giving errors) which do not need to be a fault of MATLAB but of the suitability of a specific algorithm for a given question, or of the question itself. Would you complain if you find a funny result or message after asking MATLAB to minimise the function $f(x) = 3x - 7$?

1.1.1 Defining a function

Let us define the quartic function

$$f(x) = \frac{1}{2}ax^2 + \frac{1}{4}bx^4$$

For that, open MATLAB and get to the directory (folder) in which you want to work. The command `pwd` tells you on what directory you are now,

```
>> pwd

ans =

/Users/emilio/work/teaching/11-12/IA_SciComp/Practical2
```

(in my case). The command `cd` allows you to change directory, `cd ..` moves you one level up, and `cd subdir` would move you to the subdirectory called "subdir".

Yo may also use the Current Folder panel, typically on the left of the main one. There you will see the files and subdirectories of the present one.

Now do

```
>> edit quartic1.m
```

It will possibly ask whether you want to generate a new file, and you do. It will then open the script editor of MATLAB. Type

Information about what the different entries mean can be found in page 45 of the course book. The empty line is important.

```
function [w] = quartic1(x)
% Computes a quartic function

a = 1;
b = 1;
w = (1/2)*a*x^2 + (1/4)*b*x^4;
```

and save it (by clicking File > Save, or by clicking on the save icon, a diskette). It is now saved in your working directory. Now

(a) Go back to the command window and do

```
>> help quartic1
```

(b) Try

```
>> quartic1(3)
```

(c) Generate a x array for values between -2 and 2.

(d) Try to generate the corresponding array of `quartic1` values by doing

```
>> y = quartic1(x);
```

It should come back with nasty red-coloured error messages (welcome to the world of bugs in scripts). The function was fine when evaluated for a scalar, but does not swallow an array. That is because it is trying to obtain the square of a rectangular *matrix*.

(e) Modify the function script introducing a dot before each power raising operator as follows:

```
function [w] = quartic1(x)
% Computes a quartic function

a = 1;
b = 1;
w = (1/2)*a*x.^2 + (1/4)*b*x.^4;
```

and try again. The dotted operator tells MATLAB to raise the array to the given power taking the array as a list of numbers instead of as a matrix.

(f) Do a quick plot of the `quartic1` function using the `plot` command.

MATLAB found the function because it is in your present working directory. If you want to define functions and have them stored in a bespoke directory to be found by MATLAB, you need to set up the path for MATLAB to find it.

Details on how to set the path are in pages 46, 47 of the course book.

(a) In the command window, try writing `path` (and enter). It shows where MATLAB finds things, including its own many functions.

(b) Add a new entry to the path using

```
>> path('folder',path)
```

(where 'folder' should be the name of the new folder you want to produce) and check it.

(c) Try the GUI way of doing the same thing, going to `File > Set Path ...`

1.1.2 Minimising the function

Try

```
>> fminsearch('quartic1',0.3)
```

0.3 is the trial value. Try different trial values you may fancy within the $-2,2$ range. In view of the plot produced before, rationalise your findings. Why is it not *exactly* what you expect?

In order to understand this, try now

```
>> fminsearch('quartic1',0.001)
```

How can you understand this now? We have tricked the minimisation algorithm. It depends on some tolerance parameter settings, which can be adjusted. This particular algorithm checks the values of neighbouring points with respect to the original one, and if they are not lower than a given tolerance it stops. You do not need to understand the details now, just know that these things happen, irrespective of the tools used (MATLAB, a specific computer etc). See `help fminsearch`.

As an alternatively way to resort to a function, you can do

```
>> fminsearch(@quartic1,0.2)
```

which does the same, but the function is invoked with a so-called handle.

The handle convention, multiple parameters and several variables in this context are described in the course book, pages 48-50. The concept of the handle is more general than described, and will be used later. (If interested, simply google 'MATLAB handle'!)

The function can depend on several variables and on arrays of variables.

Generate now this other function

```
function [w] = quartic2(x,a,b)
% Computes a quartic function

w = (1/2)*a*x.^2 + (1/4)*b*x.^4;
```

(just do the changes in the editor and then use "Save as ...")

In the command window, specify values for the a and b variables, and minimise with respect to x by doing

```
>> a = 1; b = 1;
>> fminsearch(@(x) quartic2(x,a,b), 0.2)
```

which should give the same results you were obtaining before. Try now

```
>> a = -1; b = 1;
>> fminsearch(@(x) quartic2(x,a,b), 0.2)
```

and rationalise your results by plotting the corresponding function $-(1/2)x^2 + (1/4)x^4$.

Consider now a similar function in a 2D plane, i.e., a function of 2 variables.

```
function [w] = quartic2D(x,a,b)
% Two-dimensional quartic function

w = -(1/2)*a*(x(1)^2+x(2)^2) + ...
     (1/4)*b*(x(1)^4+x(2)^4);
```

which corresponds to

$$w(x_1, x_2) = -\frac{1}{2}a(x_1^2 + x_2^2) + \frac{1}{4}b(x_1^4 + x_2^4)$$

Give initial values to a and b, define a trial 2D array for x_0 (e.g. » `x0 = [1,2];`), and obtain the function's minima by doing

```
>> x = fminsearch(@(x) quartic2D(x,a,b), x0)
```

1.2 Fitting

Fitting a curve consists of finding a set of parameters that define a function in terms of a given functional form in such a way such that the function reproduces a set of data as closely as possible. A typical example is the situation when the data should approximately follow a straight line: You have a set of n data pairs (x_i, y_i) , for $i = 1...n$, and try to find the parameters a and b such that the relation $y = ax + b$ lies closest to the data.

This process can be seen as a minimisation process as well: minimise the deviation between the curve and the data, as a function of the fitting parameters. The course book expands this in detail (pages 51-54). Here we will use the special commands of MATLAB that automatise the process.

1.2.1 Polynomial fitting

The `rand` *command is described in more detail later, in chapter 4 of the course notes, but you can see it in this context in p. 36)*

Generate a set of plausible data. For instance,

```
>> x = (1:.5:10);
>> y = x + 3.0* (rand(1,19)-0.5);
```

gives you a set of (x, y) values that approximately follow the straight line $y = x$. We have used `rand(n,m)` for generating an array of random numbers between 0 and 1 with equal probability (a 1×19 array since there are 19 values in the x array).

Generate a similar set of your liking. Generate a plot with symbols at data points.

You can also plot them with error bars indicating the uncertainty (related to the amplitude of the range for the random number generator in this case). See listing 3.6 in page 53 of the course book.

You can explore fitting possibilities now using the fitting GUI, accessed `Basic Fitting` under `Tools` on the window menu of the graph. Try a linear fitting first, and then explore.

You can make larger data sets (with more data points than the 19 proposed above) and see how the fitted straight line gets closer to $y = x$.

The `polyfit` command gets you the same information from the command line:

```
>> polyfit(x,y,1)
```

gives you the two coefficients of the linear fit (in a 1×2 array). Changing the last digit to a higher integer changes the order of the fit. You can now plot the fit from that information.

Here is the script doing a similar exercise in the course book:

```
x = [0:10];
y = 2 + 3*x;
y = y + randn(1,11);
sigma = ones(1,11);
a = polyfit(x,y,1);
xfit = [0:0.01:10];
yfit = polyval(a,xfit);
errorbar(x,y,sigma,'ko','MarkerSize',6,'MarkerFaceColor','k');
hold on
plot(xfit,yfit,'k-')
xlabel('Uniform x variable','fontsize',16)
ylabel('Semi-random data','fontsize,16)
set(gca,'fontsize',16)
set(gca,'TickLength',[0.02,0.0]) xlim([0,10])
hold off
```

See whether you can understand what it does, and modify it to get a fit to a higher order polynomial.

Different procedures for fitting are described in the course book (e.g. resolving linear systems of simultaneous equations with the back-slash operator). These are described in pages 56-59.

1.3 Interpolation

A similar but different problem to fitting: here we want a function that exactly goes through the data points.

Going back to the randomised data set you had prepared before, plot it again and choose `Spline interpolant` from the options in the `Basic fitting` tools in the graph window. Splines are piece-wise smooth functions that go through the data. You can also do

```
x=[0:10];
y=exp(-x/4);
xx=[0:0.1:10];
yy=spline(x,y,xx);
```

and you get samples of the exponential function in eleven points, reproduced in a much finer grid by the splines. Plot (x, y) with symbols and (xx, yy) with a line to check it.

You can see also the exercises proposed at the end of Chapter 3, p 69.

1.4 Statistics and plotting statistical data

In this section the explanations are much less detailed, so that you have to find out how to do things by yourself. To do all exercises in the section requires more time than you will probably have during the last part of the session. Try out some of the proposed exercises during the session, but you can leave it unfinished.

Starting with the statistics in Chapter 4, first get data: the files Peterborough-climate.dat, SanSebastian-climate.dat and Madrid-climate.dat have small, manageable sets of data to start with.

Details about how to do all this are in Chapter 4 of the course book, in pp 70-77.

(a) Get them from Camtools, in “Course Resources" in the “Data" folder. Download them and move them to your working folder.

(b) Do

```
>> edit Madrid-climate.dat
```

and you will see the data in the editor window (it is a text file, as are the script files).

(c) Load the data into suitable arrays, by either dragging each file onto the MATLAB main window, double clicking on the file name in the Current Folder window (and then using the Import Wizard that pops up), or by using `dlmread` (google 'matlab dlmread', details in the course book). Depending on the procedure you use you may encounter problems related to the presence of header lines in the data files. If that is the case, and no option of the procedure allows you to skip them, you can always remove them using the editor.

(d) Draw bar charts with the `bar` command (as in page 73), comparing the three cities for any of the quantities expressed in the data, as a function of month.

(e) Use an area graph (as in Figure 4.3, `area` command, pp 75-76), to show the monthly-averaged temperature range (between min and max) in one of the places.

(f) Draw now a histogram for one of the quantities. A histogram presents the number of occasions in which data points fall within ranges of values (bins). Use the `hist` command for the plot.

(g) Obtain the average (mean), standard deviation (std1), minimum and maximum values of one of the given data sets.

1.4.1 Random numbers

We have presented random numbers above. The commands `rand(n,m)` and `randn(n,m)` generate $n \times m$ arrays for a uniform or a normal distribution, respectively.

See details in pp 81-83.

(a) Plot a histogram of a normal distribution (gaussian shape) using the `randn` command to generate a reasonable amount of random data normally distributed. Observe and comment on the behaviour of the histogram when enlarging the data set.

2 Assessed Task #2

2.1 Data and fit

You can download a data file containing historical measurements of the temperature of Cambridge (cambridge.dat). This file contains a number of header lines, followed by rows of data. Each row contains data for a given month. The information given in each row is, in order, the year, month, maximum temperature, minimum temperature, number of days with an air frost (af), total rainfall, and total sunshine hours.

Load the data into Matlab. You first need to convert the year (y) and month (m) into a single time variable: something like $t = y + (m-1)/12$ would be sensible, placing the time onto a year scale. Then you need to obtain an average temperature for each month; taking the average of the minimum and maximum temperature for each month is a reasonable estimate.

Plot the average temperature against time and observe a sloping sinusoidal function with a period of one year.

Fit a sloping sinusoidal function $a_0 + a_1(t-1961) + a_2 \sin(\frac{2\pi}{\tau} t + \delta)$ to the data for average temperature, using a period (τ) of one year and treating the phase δ as a parameter to be minimised together with a_0, a_1 and a_2. You can use `fminsearch` for the fitting (minimising the squares of the differences between the data and the function, as in the example in p 68). For the sloping background we have defined the origin in the year 1961.

The background function $a_0 + a_1(t-1961)$ (with a_0 and a_1 as obtained from the fit above) resembles the annual average temperature (given that the sine function oscillates around zero), which allows you to compute the rise in average temperature in Cambridge over the past 50 years.

2.2 Statistical analysis

(a) Show the fitted temperature data and report the fitted parameters, specifically giving the temperature rise over the past 50 years deduced from the fitted parameters.

(b) Create a histogram, using the bar method, of the monthly rainfall data, in bins of size 10 mm of rainfall.

(c) Ensure that both charts are of expected quality.

For the assessment, you are required to submit a very brief report in pdf format which should contain

(a) Your name, college, and CRSid.

(b) An appropriate title.

(c) A copy of the two figures you have generated.

(d) Appropriate figure captions.

(e) Your list of values from fitting to temperature, together with the calculated rise in temperature over the past 50 years in Cambridge.

(f) A copy of the `.m` program script used to generate the charts, the fit to the data, and the calculation of the temperature rise in Cambridge over the past 50 years. Remember, if you want to generate a second figure, you can use the `figure` command to start plotting to a new window rather than overwriting your first figure.

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(s) you have generated.
4. Appropriate figure caption(s).
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.