UNIT 4

More on MATLAB: Input/output, simple statistics and some programming...

4.1 Objectives

After working through this unit you should be able to use MATLAB to

1. import text and other files into your MATLAB workspace and process it;
2. save data in an appropriate format;
3. do some basic statistical operations, like computing mean values and drawing 'best fit' regression;
4. write simple programs for performing repetitive tasks within a script or a function.

4.2 Importing text files

For the exercise, you need to download file called **dscal87.dat** (choose to save the target file to your home directory) from the Blackboard (Course Documents > Learning Units > 4-Matlab > dscal87.dat).

The file contains data relating to rainfall in a subcatchment of the Senegal river in West Africa. The 1st column gives the date, the 2nd column gives the mean raingauge reading from about 8 raingauges within the catchment and the remaining 3 columns show the duration of 'cold' cloud (cloud with top below a specified temperature threshold) as seen by satellite. The units of cold cloud duration are half-hours. The temperature threshold is given at the top of the column.

Supposedly there is a relationship between cold cloud duration (CCD) and raingauge measurements. We can use MATLAB to investigate this.

First you need to import the data into the MATLAB workspace. To do this, load MATLAB and create a new script that we are going to call `read_dscal87.m`. One method of importing the data is to use the import 'wizard' by typing `S = uiimport('dscal87.dat')` : you will be guided through the wizard in which you will be asked for more details about the data. More specifically, you will have to indicate the number of **header lines** in the file. In this example, there are 8 header lines to describe the content of the file. Finally, the variable `S` will contain all the data. The variable S is a *structure* and is a very flexible tool to store data of different kinds associated to the same entity. The actual data will be the *fields* of the *structure*.

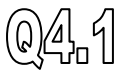
The header lines are automatically stored as strings in the field textdata, that can be accessed through `S.textdata`. You can display the headers by typing

`disp(S.textdata)`. The rest of the file is stored as a matrix in `S.data`. To make notations easier, we are going to transform this variable simply into a regular array by typing `data = S.data.` The 5 columns of data contain the date, raingauge reading and the three CCD's, respectively. We can easily access all of them individually by defining the following variables:

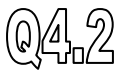
```
day    = data(:,1);
gge    = data(:,2);
CCD_40 = data(:,3);
CCD_50 = data(:,4);
CCD_60 = data(:,5);
```

4.3 Processing data

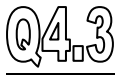
Now that we have imported the data, we can process it to extract some useful quantities like the mean values or some trends. Before doing that we should first generate a plot with time evolution of the rain-gauge and CCD measurements.



In your script `read_dscal87.m` add some instructions to create such a plot. It should have axis labels with the appropriate units, a title and a legend. What observation can you make about the relationship between CCD and raingauge estimates?

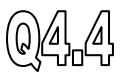


Plot an x-y graph of raingauge estimates versus CCD at -40^0C. Read through the MATLAB example (Help > Examples > Data Analysis > Basic fitting GUI) on basic fitting and see how you could fit a theoretical line to the graphed data. Add the equation of the line to the graph.



Use MATLAB functions to obtain the following quantities from the raingauge data:

- Mean daily rain for August
- Standard deviation of daily rain for August



Use the linear regression equation to calculate rainfall estimates from the CCD values in the data file. Calculate the mean and standard deviation for these data. How do they compare with your answer to Question Q4.3?

4.4 Input/output

We already saw in Unit 3 that the easiest way to import some data is to type `S = load('data_file')`. This will import the content of the file `data_file` and save it as a matrix in the variable `S`. However, this function can only be used if the data is already stored as a matrix and if the file does not contain any headers. If the data file is more complex, the import wizard `uuimport` should be used. It's also possible to read all the elements of a file one by one by using the functions `fopen` and `fscanf`. These allow much more flexibility to select only the pieces of data one is interested in and store them in a specified format. More details can be found in "MATLAB_ImportExport.pdf".

To export some results from a script/function into a text file, several options are also available. When the data to export has the form of an array, you can use the function `save`. For instance to export an array `A`, the `save` function can be used as follows:

```
save my_data.out A -ASCII
```

More flexibility can be achieved by using the function `dlmwrite`. An even more flexible approach is to use the function `fprintf`. This function can be used with both alphabetic and numeric data with any kind of delimiters. The functions `fopen` and `fclose` must be used to open/close the file before/after writing, respectively. Let us just give an example which shows how to write a file called `sine.txt` containing a short table of the sine function:

```
x = 0 : pi/8 : pi;
y = [x; sin(x)];
fid = fopen('sine.txt', 'wt');
fprintf(fid,'%10s %10s\n','x','sin x');
fprintf(fid,'%10.9f  %10.9f\n', y);
fclose(fid);
```

The function `fopen` is used to open a text file in write mode (`wt`). The `fprintf` function is then used to write two 10 characters long strings (`%10s`) on the first line of the file. The content of `y` is written on the remaining lines. Each value in `y` is treated as a 10 characters long fixed-point number with 9 digits precision (`%10.9f`). Finally the function `fclose` is used to close the file. Try it.

Q4.5 Go back to the script you wrote for questions Q3.11 and Q3.12 and add some instructions to export your results in a text file. The output files should contain two columns: height and temperature or pressure as well as the relevant headers.

4.5 Programming with MATLAB

Q4.6 Write a program that prints out the successive values of the square of all the integers between 1 and 10.

Q4.7 Write a function that takes an integer *n* as input and produces a *n-by-n* matrix *A* whose elements are given by: $A(i,j) = i+j$.

Q4.8 Write a function that takes as input the 3 coefficients *a*, *b* and *c* of the second order polynomial $p(x)=ax^2 + bx + c$ and computes its roots. Test it for several combinations of values for *a*, *b* and *c*. Keep in mind that the general formula for the roots is:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Is there any requirement for the argument of the square root?

4.14 Some more exercises

Q4.9 Go back to the script you wrote for Q3.12 and modify it such that it reads two sets of data (Albuquerque.dat and Camborne.dat, available on the Blackboard). It also should perform the same tasks required for Q3.12 for each of the soundings. Avoid repeating the same code using functions and/or loops.

Q4.10 The Saturation Vapour Pressure e_s can be calculated from the following theoretical formula (*e.g.* see MTMG02 lecture notes).

$$e_s(T) = e_s(T_0)\exp\left(-\frac{L}{R_v}\left(\frac{1}{T}-\frac{1}{T_0}\right)\right) \quad (1)$$

$e_s(T)$ is saturation vapour pressure at temperature T and

T_0 = 273.15 K $e_s(T_0)$= 6.11 hPa

$L = 2.48\text{x}10^6$ J kg^{-1} R_v = 461.5 J kg^{-1}K^{-1}

Alternatively e_s can be derived from an empirical formula:

$$e_s(T) = e_s(\mathrm{T}_0)\exp\left(\frac{A(T-T_0)}{(T-T_1)}\right) \quad (2)$$

where A = 17.27 T_0 =273.15 K T_1 = 36 K

a) Write a script to compare the two calculations with the measured values given in the file **vapour_pressure.dat** available from the Blackboard (Course Documents > Learning Units > 4-Matlab > vapour_pressure.dat). The first column of the data contains the measured vapour pressure (in kPa) and the second the corresponding temperature in K. Tabulate the data clearly and produce one graph showing e_s v T for all 3 vapour pressure data sets (measured and estimated with Eq 1 and 2) in the temperature range -5°C to $+40^{\circ}$C.

b) There is an error in the data file. Use your graph to identify the error and correct it.

c) All three graph lines should be very close together. Try to think of a way of presenting the data so as to make clear the differences between the two formulae and the measurements so that you could easily see which formula is more accurate in a given temperature range.

d) Re-derive the theoretical formula from the Clausius-Clapeyron equation (*e.g.* see MTMG02 notes) assuming L to be a linear function of temperature. Calculate a new set of values for e_s using the re-derived formula and compare this also with the measured values.