Tutorial 7 – Week 8

Aims:

Upon successfully completing these tutorial exercises, students should be able to:

- Input and output data using inbuilt functions.
- Input and output data using low-level file input/output.
- Create plots in MATLAB using the plot tool.

Tutorial 7: This tutorial will show you how to import and export several different file types using MATLAB functions.

1 Input and output data using inbuilt functions Part A

A .mat file may be used to save and load workspace data in Matlab. For example:

```
clear all;
a=1;
b=2;
save('a_b.mat');
clear all;
```

This code will save variables a and b in a file called a_b.mat. To retrieve the variables:

```
load('a_b.mat');
```

If you wish to store data using Excel the xlswrite function may be used:

```
A=rand(5); writematrix(A,'save_a.xlsx','Sheet',1,'Range','A3:E8');
```

In this case,

- **save_a.xlsx** is the name of the file that will be created,
- **A** is the data to be stored,
- **1** is the sheet number where the data is to be stored and
- **A3:E8** is the cells where the information will be stored.Open up the xls

file and verify that the data is in the correct place.

```
The data can then be read back into MATLAB using readmatrix
B= readmatrix('save_a.xlsx','Sheet',1,'Range','A3:E8')
```

In this case the elements in E1:F2 in sheet 1 will be saved as an array into variable B.

If you are saving data which contains text as well as numerical data:

```
C={'mike',2;'tim',3};
X=table(C);
writetable(X,'example.xlsx');
```

You will find that using:

```
Clear
D=readtable('example.xlsx');
```

For you to do:

Create a two dimensional cell array containing both text and numerical data (you can use any data you like). Save and retrieve the data into Excel.

2 Input and output data using inbuilt functions Part B

Open a new text file using Notepad. Save the file as 'example.csv' in your current directory. In the text file write the following:

```
1,2,3
5,6,8
1,0,-23
```

Save and close the text file. Notice that all elements in a row are separated by commas (hence the name comma separated file .csv). The contents of the file can be important into MATLAB using:

```
m=readmatrix('example.csv');
```

Modify m by changing the first row to zeros:

```
m(1,1:3)=0;
```

Then overwrite the old csv file with the new values for m:

```
csvwrite('example.csv',m);
```

The character/s used to separate elements with the text file are known as delimiters. Hence the delimiter of a csv is ‘,’. If a different delimiter is used you can use the functions dlmread() and dlmwrite().

Alter the example.csv file so that ‘|’ is used to separate elements instead of commas. To read the data use:

```
D=rand(5);
dlmwrite('AAA',D,'|')
```

For you to do:

Create your own text file using an unusual delimiter. A delimiter can be any character. Import your data into a new variable.

3 Input and output data using low-level file input/output.

MATLAB has convenient functions when reading from an Excel spreadsheet or a common text format such as csv. However, if the text format is unusual a low level input technique may be necessary. Consider the following text file:

```
Measurement  Data
N=3

12:00:00
01-Jan-1977
4.21  6.55  6.78  6.55
9.15  0.35  7.57  NaN
7.92  8.49  7.43  7.06
9.59  9.33  3.92  0.31
09:10:02
23-Aug-1990
2.76  6.94  4.38  1.86
0.46  3.17  NaN   4.89
0.97  9.50  7.65  4.45
8.23  0.34  7.95  6.46
15:03:40
15-Apr-2003
7.09  6.55  9.59  7.51
7.54  1.62  3.40  2.55
NaN   1.19  5.85  5.05
6.79  4.98  2.23  6.99
```

Copy and paste this text into a file called '**mymeas.dat**'. Save the file in the current directory. Notice that there is no consistent delimiter in this data file.

The functions that are used in low level input are:

- **[fID]=fopen('***file name goes here***')** when opens the file and creates a file identifier fID to be used in future functions.
- **N=fscanf(fID,***'put info on how to read the file here', put the size of the array to be created***)** which reads the file and saves the result into N.
- **fclose(fID)** which closes the file.

Examine the following code:

```
filename = 'mymeas.dat';
measrows = 4;
meascols = 4;

% open the file
fid = fopen(filename);

% read the file headers, find N (one value)
N = fscanf(fid, '%*s %*s\nN=%d\n\n', 1);

% read each set of measurements
for n = 1:N
    mtime{n} = fscanf(fid, '%s', 1);
    mdate{n} = fscanf(fid, '%s', 1);

    % fscanf fills the array in column order,
    % so transpose the results
    meas{n}  =  fscanf(fid, '%f', [measrows, meascols])';
end


% close the file
fclose(fid);
```

Consider the string inside the first **fscanf** function: `'%*s %*s\nN=%d\n\n'`

The format of this string is the method that may be used to read a file in any format.

%s may be used to read a string.

%*s may be used to skip a string.

\n signifies a new line.

%d may be used to read an integer.

Compare this string with the format of the first two lines of the data file.

Notice that the format of the data file is repetitive after the first two lines. Hence, a **for loop** may be used to prevent writing extraneous code.

- The code in the **for loop** captures the time data and measurement data into individual cell arrays.

The third input of the fscanf function signifies how many times the fscanf function should be applied. Notice that in the final fscanf that an array of values is used corresponding to the number of rows and columns in the measurement data.

For you to do:

Create a new data file and copy and paste the following text into the new file.

```
Stock: Yahoo
Date     Open     High     Low Close    Adj Close*  Volume
Jun 12 2017     53.79    54.55    52.60    53.12    53.12    57619100
Jun 09 2017     56.16    57.39    53.66    54.02    54.02    74186700
Jun 08 2017     55.35    55.94    53.49    55.71    55.71    75367300
Jun 07 2017     50.47    50.72    50.28    50.55    50.55    25993500
Jun 06 2017     50.42    50.76    50.30    50.50    50.50    27315900
Jun 05 2017     50.52    50.88    50.40    50.60    50.60    16600800
Jun 02 2017     50.53    50.75    50.31    50.60    50.60    9857200
Jun 01 2017     50.49    50.70    50.07    50.65    50.65    7776700
May 31 2017     50.60    50.65    49.97    50.32    50.32    8750800
May 30 2017     50.67    50.94    50.50    50.56    50.56    8400200
May 26 2017     50.50    50.74    50.32    50.67    50.67    6049500
May 25 2017     50.48    50.75    50.05    50.60    50.60    11961100
May 24 2017     50.18    50.59    49.90    50.32    50.32    12170900
May 23 2017     50.45    50.90    50.26    50.31    50.31    7342000
May 22 2017     50.51    51.08    50.22    50.65    50.65    9696600
May 19 2017     50.03    51.23    50.01    50.18    50.18    11182000
May 18 2017     48.68    50.03    47.74    49.63    49.63    13737500
May 17 2017     50.70    50.70    49.45    49.65    49.65    20954100
May 16 2017     50.05    51.12    50.03    50.96    50.96    19423600
May 15 2017     49.71    49.94    49.48    49.86    49.86    9129300
May 12 2017     49.75    49.75    49.44    49.65    49.65    6185800
May 11 2017     49.45    49.71    49.14    49.69    49.69    6142900
May 10 2017     49.40    49.60    49.14    49.40    49.40    8360400
May 09 2017     49.14    49.67    48.94    49.49    49.49    10279100
May 08 2017     48.71    48.96    48.51    48.85    48.85    10059600
May 05 2017     48.45    48.66    48.27    48.49    48.49    7835500
May 04 2017     48.63    48.68    48.16    48.45    48.45    6208900
May 03 2017     49.19    49.19    48.17    48.54    48.54    4702400
May 02 2017     48.72    49.30    48.72    49.09    49.09    6106300
```

- Read the text file and save the
  - stock name, month, day, year, open price, high price, low price, close price, adjusted close price and volume in **cell arrays**.

Handy hint: Try to capture only one piece of data with each fscanf where strings are involved. To skip the extraneous text, you could use:

```
Stock=fscanf(fID,'%*s %s\nDate    Open    High    Low Close   Adj Close*  Volume\n',1);
```

PLOTS

4 Create plots in MATLAB using the plot tool

The **plot()** function is a very useful inbuilt function in MATLAB that may be used for 2D plots. To create a 2D plot:

- First, create an array for the x axis.
- Second, write the equation for the y axis.
- Finally, use the plot function plot(x,y).

For example, to plot $y=x^2$ between -10 and +10 in increments of 0.1:

```
x=-10:0.1:10;
y=x.^2;
plot(x,y);
```

Recall that the **.^** operator is needed for an element-wise power operation. Whenever performing an element wise multiplication, division or power, the normal operand must be preceded with a full stop.

It is possible to plot multiple functions at once and add features to the plot, such as a title, legend and axis labels:

```
x=-10:0.1:10;
y1=x.^2;
y2=x.^3;
plot(x,y1,x,y2);
title('Functions');
xlabel('x');
ylabel('y');
legend('y=x.^2','y=x.^3');
```

It is also possible to change line colours, types and thicknesses. For a full list of options, type 'help plot' in the command window. For example:

```
x=-10:0.1:10;
y=x.^2;
plot(x,y,'k:','LineWidth',3);
```

4.1 For you to do:

Create a plot containing three different functions of your choosing. Customise the colours, line thicknesses and line types. Include labels and a legend.

MATLAB is also capable of creating 3D plots. The functions **meshgrid()**, **mesh()** and **surf()** may be used for 3D plotting. To create a 3D plot:

- Use meshgrid to create the x and y arrays.
- Compute z.
- Create the plot using either mesh(z) or surf(z).

For example:

```
[x,y]=meshgrid(-1:0.01:1);
z=sin(x)-cos(y);
mesh(z);
```