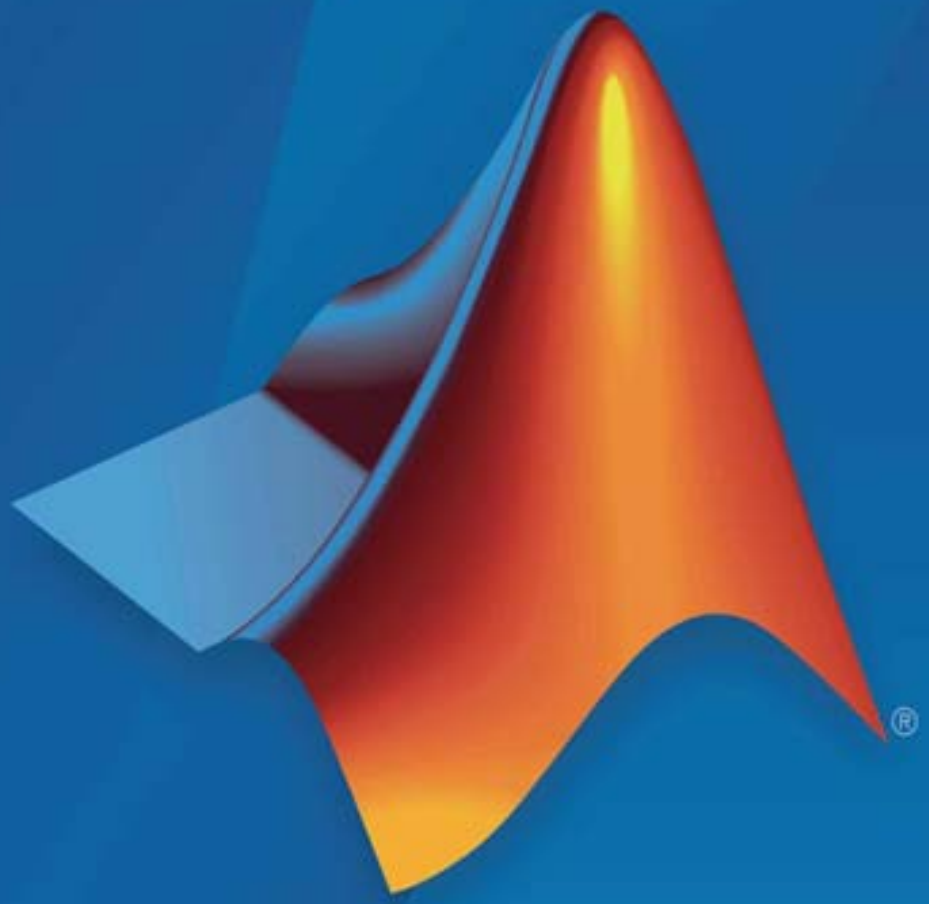
الجامعة المستنصرية
كلية الهندسة
قسم هندسة الموارد المائية



Lecture 9: Managing Files in MATLAB

المحاضرة التاسعة: ادارة الملفات في ماتلاب

المنهاج الدراسي لمادة البرمجه والتطبيقات (ماتلاب)
الكورس الدراسي الاول / المرحلة الثانية

من اعداد د. صباح حسن لفته

Introduction

Data are stored in many different formats, depending on the devices and programs that created the data and on the application. For example, sound might be stored in a .wav file, and an image might be stored in a .jpg file. Many applications store data in Excel spreadsheets (***.xls files***). The most generic of these files is the ***ASCII file***, usually stored as a **.dat** or a **.txt** file. You may want to import these data into MATLAB to analyze in a MATLAB program, or you might want to save your data in one of these formats to make the file easier to export to another application.

a- Importing Data

There are many ways to import data to MATLAB code which depends on the kind of data that MATLAB allows to import. The following table represents a list of some data types that recognised by MATLAB.

File type	Extension	Note
Text	.mat	MATLAB workspace
	.dat	ASCII data
	.txt	ASCII data
	.csv	Comma-separated values ASCII data
Spreadsheet data	.xls, xlxx	Excel spreadsheet

One of these ways to import data to MATLAB code is using ***load command*** which loads all the variables from the MAT-file (matlab.mat), if it exists, and returns an error if it does not exist. The simplest form of load command is:

>> load ('filename')

This will load all the variables from filename given a full pathname or MATLAB path relative partial path name. the imported file must be in the same folder that MATLAB exist inside it.

Another form that can be used to import only specific variables inside MAT file such as:

>> Var=load ('filename', 'X', 'Y', 'Z')

In which, x, y, and z are variables saved inside the MATLAB filename which is imported MATLAB.

In some cases, dealing with large data sets are a big challenge in MATLAB codes, so what we really need is an easy way to read data from a file and use it in a MATLAB program. The ***textread function*** serves that purpose. The textread function reads ASCII files that are formatted into columns of data, where each column can be of a different type, and stores the contents of each column in a separate output array. This function is *very* useful for importing large amounts of data printed out by other applications. The form of the textread function is:

>> [a,b,c,...] = textread(filename , format ,n)

where filename is the name of the file to open, format is a string containing a description of the type of data in each column, and n is the number of lines to read. (If n is missing, the function reads to the end of the file.) The format string contains the same types of format descriptors as ***function fprintf***. Note that the number of output arguments must match the number of columns that you are reading.

Let take the following example, suppose that file test_input.dat contains the following data:

```
James     Jones    O+    3.51   22    Yes
Sally     Smith    A+    3.28   23     No
```

The first three columns in this file contain character data, the next two contain numbers, and the last column contains character data. This data could be read into a series of arrays with the following function:

```
>> [first,last,blood,gpa,age,answer] = textread('test_input.dat','%s %s %s %f %d …
                                                %s')
```

Note the string descriptors %s for the columns where there is string data, and the numeric descriptors %f and %d for the columns where there is floating point and integer data.String data is returned in a cell array, and numeric data is always returned in a double array. When this command is executed, the results are:

```
>> first =
        'James'
        'Sally'
>> last =
        'Jones'
        'Smith'
>> blood =
        'O+'
        'A+'
>> gpa =
        3.5100
        3.2800
>> age =
       42
       28
```

```
>> answer=
        'Yes'
         'No'
```

This function can also skip selected columns by adding an asterisk to the corresponding format descriptor (for example, %*s).

On the other hand, if we need to import data from Excel Microsoft sheet files, ***xlsread function*** is a recommended command for this purpose. The forms of this function are:

>> Var=xlsread(ˈfilenameˈ)

This will read the first worksheet in the Microsoft Excel spreadsheet named filename and returns the numeric data in a matrix.

>> Var=xlsread(ˈfilenameˈ, ˈsheetˈ)

This will read the specified worksheet.

>> Var=xlsread(ˈfilenameˈ, ˈsheetˈ,ˈ xlRangeˈ)

This will read the specified worksheet and range.

Hint:

The input variables of xlsread command must be enter as strings (′ ′) so that MATLAB can execute the statement with no error.

Let say we have Excel worksheet contains three columns, each with 10 numbers, the first one is the counting numbers, second represents age values and the third contains

temperature readings. The file is stored under the name ‘Example_1.xlsx’. To import the data from this file:

>> Var=xlsread(´Example_1.xlsx´,´Sheet1´,´A2:C12´)

The result will be as follow;

```
Var =
     1    23    35
     2    14    36
     3    45    37
     4    67    38
     5    10    36
     6    32    37
     7    55    39
     8    21    35
     9    37    34
    10    44    38
```

The ***input function*** is another method to enter data in MATLAB program which displays a prompt string in the Command Window and then waits for the user to type in a response. For example, consider the following statement:

>> Inp_Var = input('Enter an input value:');

When this statement is executed, MATLAB prints out the string 'Enter an input value: ' and then waits for the user to respond. If the user enters a single number, it may just be typed in. If the user enters an array, it must be enclosed in brackets. In

either case, whatever is typed will be stored in variable **Inp_Var** when the return key is entered. If only the return key is entered, then an empty matrix will be created and stored in the variable.

b- Exporting Data

Writing results of MATLAB program to a file is a very important process. This procedure can be achieved using different functions. MATLAB provides many of these functions in its library. We will focus only on the most used functions to write data to a file.

The **save** command one of the popular functions that used in MATLAB to save data from the current MATLAB workspace into a disk file. The most common form of this command is:

>> save filename var1 var2 var3

where **filename** is the name of the file where the variables are saved, and **var1**, **var2**, and so forth are the variables to be saved in the file. By default, the file name will be given the extension "mat", and such data files are called MAT-files. If no variables are specified, then the entire contents of the workspace are saved.

MATLAB saves MAT-files in a special compact format which preserves many details, including the name and type of each variable, the size of each array, and all data values. Unfortunately, the MAT-file is in a format that cannot be read by other programs. If data must be shared with other programs, then the ***-ascii*** option should be specified, and the data values will be written to the file as ASCII character strings separated by spaces. However, the special information such as variable names and types are lost when the data is saved in ASCII format, and the resulting data file will be much larger.

For example, suppose the array x is defined as

>> x =[1.23 3.14 6.28; -5.1 7.00 0];

Then the command **"save x.dat x –ascii"** will produce a file named ***x.dat*** containing the following data:

1.2300000e+000 3.1400000e+000 6.2800000e+000

-5.1000000e+000 7.0000000e+000 0.0000000e+000

This data is in a format that can be read by spreadsheets or by programs written in other computer languages, so it makes easy to share data between MATLAB programs and other applications.

If in case we need to save data in Excel spreadsheet, then ***xlswrite command*** will be the best choice for that. The form of this function is;

>> **xlswrite('filename.xls', Var)**

where Var is the array, we want to store in the Excel spreadsheet.

There is another format for this function as follow;

>> xlswrite(**'filename.xls',Var, 'sheet', 'xlRange')**

This will write data to the specified worksheet and range.

Hint:

MATLAB provides many functions that can be used effectively to call and save data which can not be covered in this course, for more information you can search the help of the program and figure out its documentation about how to use these functions in your MATLAB codes.

There is a useful function that can be used with functions that use to load and save data in MATLAB called ***change directory (cd)***. This command allows MATLAB to change the current directory to the one that want to bring files from or save data to files inside it. The format of this function is:

>> New_folder=cd(**'full address of the requested folder')**

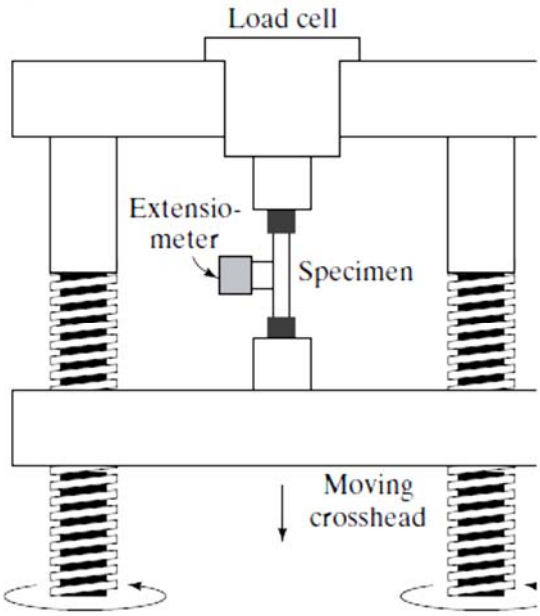
for example, let say we want to call file under the name ***my_example.xlsx*** to MATLAB program and the file exists in folder with name ***my folder***. We can write the code statements to change the current directory and import data as follow:

```
>> New_folder=cd('C:\Users\engsa\Desktop\my folder  ');
>> Var=xlsread('my_example_1.xlsx', 'Sheet1', 'A2:C12')
>> Current_folder=cd('C:\Program Files\MATLAB\R2016b\bin ');
```

After execution of these statements, MATLAB change the current folder to the new folder that contains the file required to bring data from it. Then will back to the folder of the MATLAB program because without this, there will an error display in the command widow. The results as follow:

```
>> Var =
     1    23    35
     2    14    36
     3    45    37
     4    67    38
     5    10    36
     6    32    37
     7    55    39
     8    21    35
     9    37    34
    10    44    38
```

Example (1): A tensile testing machine such as the one shown below is used to determine the behavior of materials as they are deformed. In the typical test, a specimen is stretched at a steady rate. The force (load) required to deform the material is measured, as is the resulting deformation. An example set of data measured in one such test is shown in Table listed below and recorded in Excel file (Tension.xlsx).



Load, kN	Length, cm
0	5.0800
9	5.0861
18	5.0919
27	5.0978
33	5.1039
36	5.1125
38	5.1265
40	5.1582
42	5.2070
44	5.2705

These data can be used to calculate the applied stress and the resulting strain with the following equations.

$$\sigma = \frac{F}{A} \qquad \text{and} \qquad \varepsilon = \frac{l - l_o}{l_o}$$

where:

σ = stress in (kN/m²) or (MPa).

F = applied force in (kN).

A = sample cross-sectional area in (m²).

ε = strain in (cm/cm).

l = sample length in (cm).

l_o = original sample length in (cm).

Required; Write MATLAB program to do the following:

1- Loaded the data of test that recorded in file Tension.

2- Calculate the stress and strain using the above equations and the measured values. The diameter of sample is 1.3 cm.

3- Create an *x–y* plot with strain on the *x* -axis and stress on the *y*-axis. Connect the data points with a solid red line and use circles to mark each data point. Add title to the plot and x-label and y-label, scaling both axes according to the results. add a text to define the yield point inside the plot.

4- Save input and resulted data (length, force, stress, strain) to excel spreadsheet file under appropriate name.

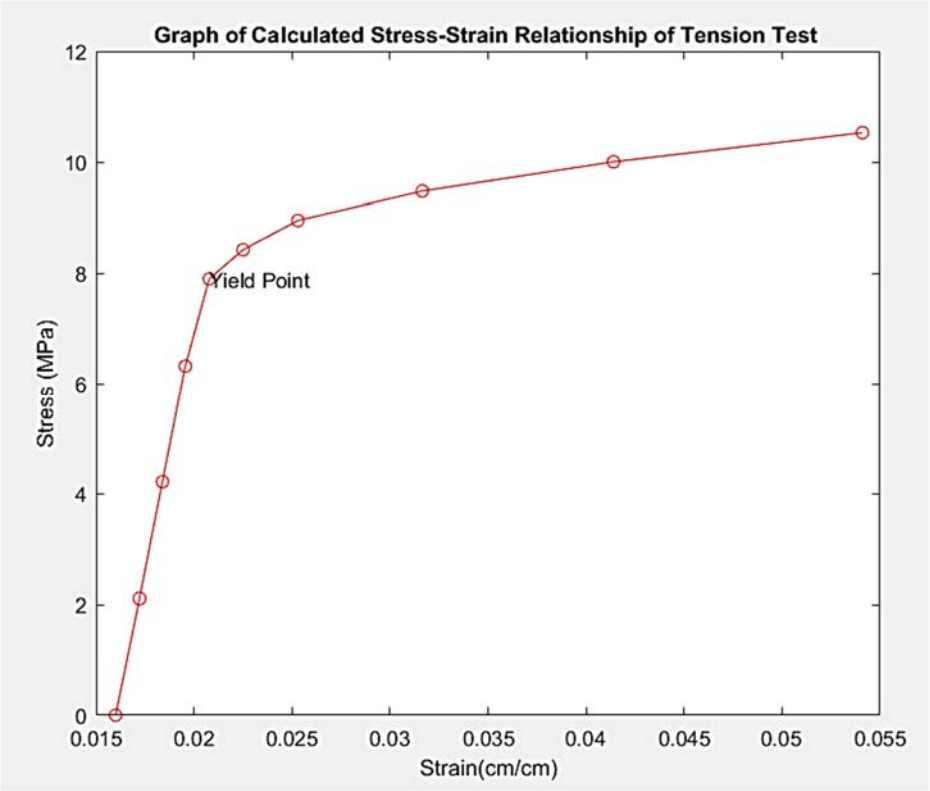
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% MATLAB Program designed for Analysis Tension Test Results %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Program prepared by: Dr. Sabah Hassan and Lect. Shymaa Khalid%%%%%
```

```
%%% Date: Wednesday 17/2/2021
%%%%=========================================%%%%%%
%% The code is prepared to process tension test results
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Definition of Variables %%%%%%%%%%%%%%%%%%%%%%
%% Ten: main input data
%% Diam: diameter of sample (cm)
%% Len: original length of sample
%% Fc: force vector (kN)
%% Ls: sample length vector (cm)
%% Rad: radius of sample (cm)
%% Ar: cross-sectional area of sample (cm^2)
%% M: main resulted matrix
%%%%=============================================%%%%%
>> clear all;
>> clc;
>> close all;
%% Section 1: Enter known variables
>> New_fold=cd('C:\Users\engsa\Desktop\my folder ');
>> Ten=xlsread('Tension.xlsx', 'Sheet1', 'A2:B11');
>> Curr_fold= cd('C:\Program Files\MATLAB\R2016b\bin ');
>> Diam= input('Hi Student Tell Me What is the Diameter=')
>> Len=input('Hi Student Again Where is the Sample Length=')
%% Section 2: Calculating required variables stress and strain
>> Fc= Ten(:,1);    %% force vector
>> Ls=Ten(:,2);     %% sample length
>> Vl=length(Fc); %% vector length for loop
```

```
>> Rad= (Diam/2)/100; %% radius of the sample (m)
>> Ar=((Rad^2)/pi); %% cross-sectional area of the sample
>> Strain=zeros(Vl,1);
>> Stress=zeros(Vl,1);
>> for i=1:Vl
>> Stress(i)=(Fc(i)/Ar);              %% stress in kPa units
>> Strain(i)=(Ls(i)-Len)/Len;    %% strain in cm/cm
>> end
%% Section3: Plotting the results
>> plot(Strain,Stress/1000, '-or')
>> xlabel('Strain(cm/cm)')
>> ylabel('Stress (MPa)')
>> title('Graph of Calculated Stress-Strain Relationship of Tension Test ')
>> text(Strain(5),Stress(5)/1000,'Yield Point')
>> axis([0.015,0.055,0,12])
%% Section4: Saving the input and output data
>> cd('C:\Users\engsa\Desktop\my folder ');
>> M=[Fc,Stress/1000,Ls,Strain];
>> xlswrite('My_results.xlsx', M)
>> cd('C:\Program Files\MATLAB\R2016b\bin ');
```

The results after the execution of the code;

Plotting the stress-strain curve



Saving the input and output data