Chapter 3

Read, display and save data in MATLAB

1. Introduction

Reading, displaying and saving data in MATLAB are essential operations for data manipulation and analysis. Firstly, MATLAB environment offers several ways to read data stored in different file formats. After reading and obtaining data, the displaying operations are crucial for analysis and visualization. In this context, the MATLAB software provides a straightforward way to output data to the Command Window. Finally, saving data is an important task for their use in future works. Therefore, in MATLAB environment we can save data and preserve their structures and variables.

In this chapter, our objective is to investigate different functions that enable the effective management of the outlined operations (reading, displaying, and saving data). We will not only understand the mechanisms of these functions but also reinforce our understanding through simple and comprehensive examples.

2. Reading and displaying data in MATLAB

We introduce in these subsections the main functions of reading and displaying data: `input()`,`disp()` and `fprintf()`.

2.1. Input function

This function allows reading data structure entered by users. The syntax of this function is given as follows:

```
>> Num = input('Please enter a number: ')
Please enter a number: 5

Num =

     5
```

In this example, the user should enter a number, and this value is stored in the variable ‘`Num`’. If the user needs to enter a text instead of a number, it is necessary to use quotation marks around the string. In a more appropriate way, we must add ‘`s`’ argument in the use of the input function. The two possible ways are indicated below:

```
>> Name1 = input('Please enter a name: ')
Please enter a name: 'Jijel'

Name1 =

    'Jijel'

>> Name2 = input('Please enter a name: ', 's')
Please enter a name: Algeria

Name2 =

    'Algeria'
```

Displaying data in MATLAB is an operation generally performed through using functions to show information or results to the user. MATLAB offers various methods for displaying data. The next points present the common ways to display data in MATLAB.

2.2. Output functions

Two main output statements are provided by MATLAB to display results and expressions. We discuss in details in theses subsections, how to use `disp()` and `fprintf()` functions.

The function `disp( )`

The `disp()` function is employed in MATLAB to display different kinds of data structure, such as numbers and text. However, `disp()` does not allow formatting. The next command window shows the use of `disp()` function.

```
>> A = 5; B = 10;
>> % Display values
>> disp(A)
     5

>> disp(A + B)
    15

>> % Display text
>> disp('Hellow world')
Hellow world
>> msg = 'Good morning';
>> disp(msg)
Good morning
```

The function `fprintf ()`

The `fprintf()` function is used for displaying different data structures, where the main characteristic of this function is controlling and determining the format and layout of the displayed outputs or results. The basic syntax of `fprinf()` is given as follows:

`fprintf(formatSpec,A1,...,An)`, where:

`formatSpec` is a set of characters that determine the format of the output and can include combinations of '`%`' character and a conversion character such as `d, i, o, u, x, f, e, g, c,` or `s`.

`A1,...,An` presents the variables or the data structure needed to include the formatted output.

The next points clarify the most used formatting options:

a) **The `%s` format**: used to format and display string data, such as words, sentences, or text messages. If we display only one character, we can use **`%c`** format. A practical example is showcased in what follows:

```
>> % The %s format for string data
>> Name = 'Ahmed';
>> fprintf('My name is %s\n', Name)
My name is Ahmed
```

In this example, we use `%s` to insert the value of the Name variable into the formatted output. The `\n` argument is used to move down to the next line.

b) **The `%d` format**: used to format and display integer values in a specified format as indicated in the next example.

```
>> % The %d format integer values
>> Age = 20;
>> fprintf('Ahmed is %d years old.', Age)
Ahmed is 20 years old.
```

In this example, we use `%d` to format and display the variable `Age`. The result of this program displays a message such as "`Ahmed age is: 20`", where `%d` is replaced by the value of the `Age` variable.

c) **The `%f format`**: This format is used to insert, control and display the format of floating-numbers. In other manner, the `%f` format helps displaying the real numbers with decimal points in the results. The use of this format is indicated below:

```
>> % The %f format for floating numbers
>> fprintf('The value of pi is: %f\n', pi)
The value of pi is: 3.141593
```

In this example, we use the `%f` to determine the format of floating-numbers. The results of this program show a message such as `"The value of pi is: 3.141593"`, where `%f` is replaced by the value of the pi constant.

In addition to formatting options, we can add the ***width field*** before the formatting character to indicate the number of characters used in printing. For example, `%`**`3`**`d` and `%`**`8`**`s` indicate printing the variable with 3 numbers and 8 characters respectively. In the case of the float variables, we can specify the number of decimal places. For example, to print a float number in field of 5 of width where 2 places are reserved for decimals after the decimal point, we write %**5.2**f. The next example, clarifies the use of the width field in the formatting procedure.

```
>> L = 5.371;
>> W = 2;
>> fprintf('The length of the rectangle is : %3.2f \n', L)
The length of the rectangle is : 5.37
>> fprintf('The length of the rectangle is : %.2f \n', L)
The length of the rectangle is : 5.37
>> fprintf('The width of the rectangle is : %.1f \n', W)
The width of the rectangle is : 2.0
>> fprintf('The area of a rectangle with length %.2f and width %.1f
is : %.3f \n', L, W, L*W)
The area of a rectangle with length 5.37 and width 2.0 is : 10.742
```

3. Saving and loading data in MATLAB

In MATLAB, saving data presents a crucial operation of maintaining the content of different data structures to a file for future use. Saving data in MATLAB helps to preserve the results of programs and share them with others or follow computations in subsequent analysis.
There are various methods or manners used to save data structure in MATLAB regarding saving workspace variable into MAT file, saving data to text or spreadsheet files.

3.1. Saving data in binary file format

To save the variables of the workspace in binary format (.mat file), we use the following functions:

`save ('file-name.mat')` or `save file-name.mat` : saves all variables located in the workplace;

`save ('file-name', 'variable1', 'variable2')` or `save file-name, variable1, variable2`: saves only the workspace variables you list after the filename: `'variable1'` and `'variable2'`.

`save('file-name','variables','-append')` adds variables to the existing file-name. If a variable already exists in the selected file, the save function overwrites it with the new value in the workspace.

The following command window shows an example of using the `save()` command.

```
>> Name = 'Ahmed';

>> Age = 20;

>> save('mydata.mat') % Save all workspace variables to a .mat file

>> Father = 'Mohammed';

>> save('fdata.mat','Father') % Save Father variable to fdata.mat file

>> save('mydata', 'Father', '-append'); % Add variable Father to mydata.mat
```

In addition, we can save the data into file by using the menu bar of MATLAB or contextual menu in the workspace part, as presented in Figure 1:

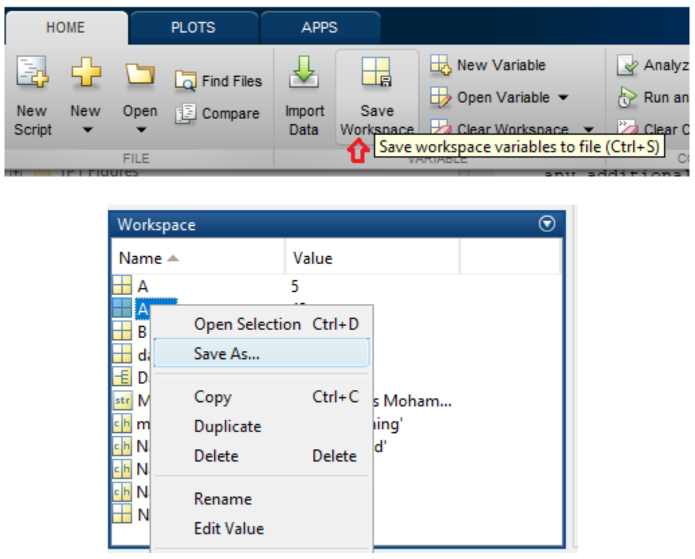


Figure 1: Save data using MATLAB menu

3.2. Loading data from file in MATLAB

In MATLAB, it is possible to load the different data structures from several sources and files. The next lines present the most used function to load data.

This function `load()` is used to read data structure from a file, which can be a MATLAB file (.mat file), text file, Excel file or others and store it in the workspace or in a defined user variable. The basic syntax of this function is indicated below:

`load('file-name')`: reads data from the indicated file `'file-name'`.
`Data =load('file-name')`: reads data and save it in variable `Data`.
`load('file-name', 'variables')`: reads specified variables from a .mat file.

Next command lines display the use of the load function.

1. Load the contents of the `mydata.mat` file into the workspace:

```
>> load('mydata.mat')
>> whos
  Name          Size                Bytes  Class     Attributes

  Age           1x1                     8  double
  Father        1x8                    16  char
  Name          1x5                    10  char
```

2. Load the contents of a `.mat` file into a variable in the workspace:

```
>> Data = load('mydata.mat')

Data =

  struct with fields:

       Age: 20
      Name: 'Ahmed'
    Father: 'Mohammed'

>> whos
  Name        Size                Bytes  Class     Attributes
  Data       1x1                  538  struct
```

3. Load a variable from a `.mat` file into the workspace:

```
>> load('mydata.mat', 'Age')
>> whos
  Name       Size                Bytes  Class     Attributes

  Age        1x1                     8  double
```

In the first case, the `load()` function reads the data structure located in `'mydata.mat'` and stores it into the workspace. However, the acquired data are saved into `Data` variable in the second case. The last example presents the use of `load()` function to load a specific variable from a .mat file.

In addition, we can read the data located in external file using the MATLAB menu as indicated in Figure.2.

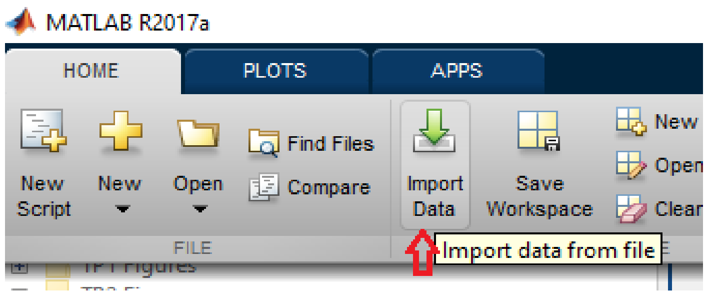
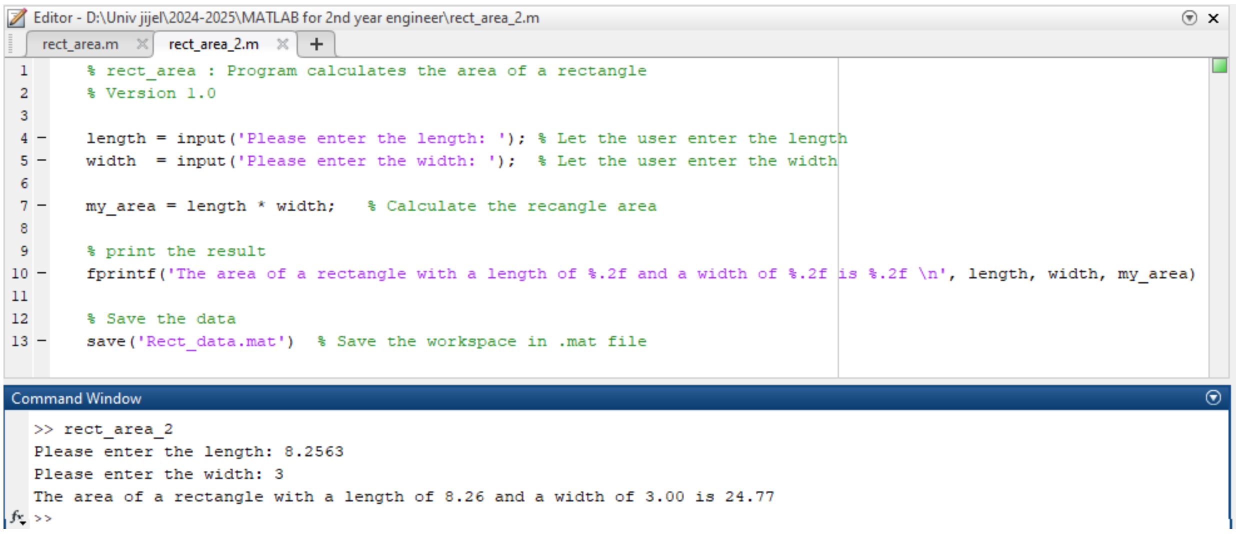


Figure 2: Loading data using MATLAB menu

4. Script with input, output and save functions

In our first script, we have created a program that calculates the area of a rectangle. Now, we improve our program by using the input, output and save functions to let the user enter the value of the length and the width, display the calculated area in the screen and save the workspace into a .mat file. Figure.3 shows such modifications and the execution of the script.


Editor - D:\Univ jijel\2024-2025\MATLAB for 2nd year engineer\rect_area_2.m
rect_area.m
rect_area_2.m
Command Window

```
% rect_area : Program calculates the area of a rectangle
% Version 1.0

length = input('Please enter the length: '); % Let the user enter the length
width  = input('Please enter the width: ');  % Let the user enter the width

my_area = length * width;   % Calculate the recangle area

% print the result
fprintf('The area of a rectangle with a length of %.2f and a width of %.2f is %.2f \n', length, width, my_area)

% Save the data
save('Rect_data.mat')  % Save the workspace in .mat file
```

```
>> rect_area_2
Please enter the length: 8.2563
Please enter the width: 3
The area of a rectangle with a length of 8.26 and a width of 3.00 is 24.77
>>
```

Figure. 3. Rectangle area program with input and output functions

5. Practical work 3

1. Give the output of the following commands and check the results using the MATLAB Command window.

```
>> disp('I am an Engineer')
>> D = 'Engineer'; disp('I am an  %s', D)
>> a = 5; fprintf('%d  %f   %.2f   %s \n', a , a, a, 'a+a')
>> A = 5.231; B = 0.1; fprintf('The sum of A and B is %.2f \n', A+B)
>> A = 5.231; B = 0.1; fprintf('%.2f + %.3f = %.5f \n', A, B, A+B)
```

2. Write a script file that computes the area of a circle (using the input and the output functions).

3. Write the script that computes the following formulas (using the input and the output functions):

   $R1 = 3\pi/2 \times a, \; R2 = cosinus\ (6\pi/b)$

   a. Calculate *R1* and *R2* for *a = 5* and *b = 3.*

   b. Save all variables located in the workspace in a .mat file and name it 'myData'.

   c. Calculate *R1* and *R2* for *a = 10* and *b = 12.*

   d. Include the obtained results to the created .mat file.

4. Based on the previous script:

a. Load the saved files;
b. Display the entered variables;
c. Display the obtained results.

5. Write a script to calculate the surface of a cylinder given by:

   $Cylinder\ surface = 2\pi r^2 + 2\pi rh$ using the input and the output functions.
   where r and h are the radius and the height of the cylinder respectively.

   a. Using the developed script calculate the surface of the indicated cylinders below:
      1. Cylinder1: $r = 5.53,\ h = 3.8$.
      2. Cylinder2: $r = 8.2,\ h = 10.45$.
      3. Cylinder3: $r = 120,\ h = 38.22$.
   b. Save the obtained results in a .mat file;
   c. Clear all data;
   d. Using a variable, load from the saved data only the surface of the two last cylinders.