Lecture Four

Programming in MATLAB

Introduction

So far in these lab sessions, all the commands were executed in the Command Window. The problem is that the commands entered in the Command Window cannot be saved and executed again for several times. Therefore, a different way of executing repeatedly commands with MATLAB is:

1. to create a file with a list of commands,
2. save the file, and
3. run the file.

If needed, corrections or changes can be made to the commands in the file. The files that are used for this purpose are called script files or scripts for short.
This section covers the following topics:

- **M-File Scripts**
- **M-File Functions**

❖ M-File Scripts

A script file is an external file that contains a sequence of MATLAB statements. Script files have a filename extension .m and are often called M-files. M-files can be scripts that simply execute a series of MATLAB statements, or they can be functions that can accept arguments and can produce one or more outputs.

Example 1

Consider the system of equations:

$$x + 2y + 3z = 1$$
$$3x + 3y + 4z = 1$$
$$2x + 3y + 3z = 2$$

Find the solution x to the system of equations.

Solution:

1. Use the MATLAB editor to create a file: File → New → M-file.
2. Enter the following statements in the file:
   a. A = [1 2 3; 3 3 4; 2 3 3];
   b. b = [1; 1; 2];
   c. x = A\b

3. Save the file, for example, example1.m.
4. Run the file, in the command line, by typing:

```
>> example1
x =
    -0.5000
     1.5000
    -0.5000
```

When execution completes, the variables (A, b, and x) remain in the workspace. To see a listing of them, enter whos at the command prompt.

Note !: The MATLAB editor is both a text editor specialized for creating M-files and a graphical MATLAB debugger. The MATLAB editor has numerous menus for tasks such as saving, viewing, and debugging.

Because it performs some simple checks and also uses color to differentiate between various elements of codes, this text editor is recommended as the tool of choice for writing and editing M-files. There is another way to open the editor:

```
>> edit
```

or

```
>> edit filename.m
```

to open filename.m.

M-File functions

As mentioned earlier, functions are programs (or routines) that accept input arguments and return output arguments. Each M-file function (or function or M-file for short) has its own area of workspace, separated from the MATLAB base workspace.

- Anatomy of a M-File function

This simple function shows the basic parts of an M-file.

```
1. function f = factorial(n)
2. % FACTORIAL(N) returns the factorial of N.
3. % Compute a factorial value.
4. f = prod(1:n);
```

The first line of a function M-file starts with the keyword function. It gives the function name and order of arguments. In the case of function factorial, there are up to one output argument and one input argument. Table 1 summarizes the M-file function.

Table 1: Anatomy of a M-File function

Part no.	M-file element	Description
(1)	Function definition line	Define the function name, and the number and order of input and output arguments
(2)	H1 line	A one line summary description of the program, displayed when you request Help
(3)	Help text	A more detailed description of the program
(4)	Function body	Program code that performs the actual computations

As an example, for n = 5, the result is,

```
>> f = factorial(5)
     f =
     120
```

Both functions and scripts can have all of these parts, except for the function definition line which applies to function only. In addition, it is important to note that function name must begin with a letter, and must be no longer than the maximum of 63 characters. Furthermore, the name of the text file that you save will consist of the function name with the extension .m. Thus, the above example file would be factorial.m. Table 2 summarizes the differences between scripts and functions.

Table 2: Difference between scripts and functions

Scripts	Function
- Do not accept input arguments or return output arguments.	- Can accept input arguments and return output arguments.
- Store variables in a workspace that is shared with other scripts	- Store variables in a workspace internal to the function.
- Are useful for automating a series of commands	- Are useful for extending the MATLAB language for your application

- Input and output arguments

As mentioned above, the input arguments are listed inside parentheses following the function name. The output arguments are listed inside the brackets on the left side. They are used to transfer the output from the function file. The general form looks like this

function [outputs] = function_name(inputs)

Function file can have none, one, or several output arguments. Table below illustrates some possible combinations of input and output arguments.

Table 3: Example of input and output arguments

function C=FtoC(F)	**One input argument and one output argument**
function area=TrapArea(a,b,h)	Three inputs and one output
function [h,d]=motion(v,angle)	Two inputs and two outputs

Input to a script file

When a script file is executed, the variables that are used in the calculations within the file must have assigned values. The assignment of a value to a variable can be done in three ways.

1. The variable is defined in the script file.
2. The variable is defined in the command prompt.
3. The variable is entered when the script is executed.

We have already seen the two first cases. Here, we will focus our attention on the third one. In this case, the variable is defined in the script file. When the file is executed, the user is prompted to assign a value to the variable in the command prompt. This is done by using the input command.

Example 2.

```
% This script file calculates the average of points
% scored in three games.
% The point from each game are assigned to a variable
% by using the `input' command.

game1 = input('Enter the points scored in the first game ');
game2 = input('Enter the points scored in the second game ');
game3 = input('Enter the points scored in the third game ');
```

```
average = (game1+game2+game3)/3
```

The following shows the command prompt when this script file (saved as example) is executed.

```
>> example
>> Enter the points scored in the first game 15
>> Enter the points scored in the second game 23
>> Enter the points scored in the third game 10
average =
          16
```

Example 3

Enter symbolic names :

```
clc;
clear;
z = input ('enter name', 's');
```

To denote the entry of string

❖ We can also use **prompt** and **menu** commands to inputting in MATLAB.

Output commands

As discussed before, MATLAB automatically generates a display when commands are executed. In addition to this automatic display, MATLAB has several commands that can be used to generate displays or outputs. Two commands that are frequently used to generate output are: **disp** and **fprintf**. The main differences between these two commands can be summarized as follows:

Table 4: disp and fprintf commands

▪ **disp**	. Simple to use. . Provide limited control over the appearance of output.
▪ **fprintf**	. Slightly more complicated than disp. . Provide total control over the appearance of output.

The display **disp** function allows the programmer to display the contents of a string or a matrix in the command window. Although the **disp** function is

adequate for many display tasks, the **fprintf** function gives the programmer considerably more control over the way results are displayed. Table 5 below illustrates the various formats supported by **fprintf**.

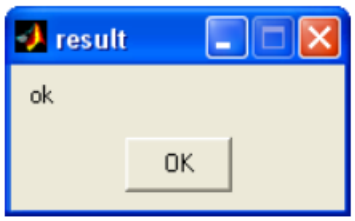
Table 5. Special character used in fprintf () statements.

Type Specifier	Printing Form: fprintf('**format string**', variables_to_be_printed,..)	Special Character	Meaning
%c	Character type	\n	New line
%s	String type	\t	Tab
%d	Decimal integer number type	\b	Backspace
%f	Floating point number type	\r	CR return
%e	Decimal exponential type	\f	Form feed
%x	Hexadecimal integer number	%%	%
%bx	Floating number in 16 hexadecimal digits(64 bits)	''	'

❖ We can also use **msgbox** command to output in MATLAB

```
>> msgbox ('ok', 'result')
```

Thing to be printed Title of Box



Example 4:

Write a function file that converts temperature in degrees Fahrenheit (F^O) to degrees Centigrade (C^O). Use input and fprintf commands to display a mix of text and numbers. Recall the conversion formulation, C = 5/9 * (F - 32).

Answer

```
function [ C ] = FtoC( F )
F = input('Enter the temperature in degrees Fahrenheit ');
C = 5/9 * (F - 32)
fprintf(' The temperature in degrees Centigrade=%d\n', C);
```

```
end
```

This M-file produces the following interaction in the command window:

```
Enter the temperature in degrees Fahrenheit 5
The temperature in degrees Centigrade = -15
```

Example 5

Write a program to swap two numbers

Answer

```
a = input("enter a value :");
b = input("enter b value :");
temp = a;
a = b;
b = temp;
fprintf('a value = %d\n', a)
fprintf('b value = %d\n', b)
```