



# MATLAB

MATLAB is an abbreviation meaning “Matrix Laboratory”. As such it is a programming language which provides many convenient ways for creating vectors, matrices, and multi-dimensional arrays. In the MATLAB, a vector refers to a one dimensional ( $1 \times n$ ) or ( $n \times 1$ ) matrix, commonly referred to as an array in other programming languages. On the other hand, a matrix generally referred to as a 2-dimensional array, i.e. an  $n \times m$  array where  $m$  and  $n$  are greater than or equal to 1. Arrays with more than two dimensions are referred to as multidimensional arrays. In this unit, we shall expose ourselves to the MATLAB environment beginning with its operating windows, frame, and ending with its user functionalities.

## 1. MATLAB environment

MATLAB environment is command oriented somewhat like UNIX. A prompt “>>” appears on the screen and a MATLAB statement can be entered. When the <ENTER> key is pressed, the statement is executed, and another prompt “>>” appears.

MATLAB has four main key features or windows named as:

- a) Command window
- b) Command History
- c) Workspace
- d) Current Directory

As a user of MATLAB, we often find ourselves interacting more with the command window as it is taken as the main MATLAB feature. With the command window, we are able to define our variables, enter values and run functions and M-file scripts (more about m-files later during the third unit of programming with MATLAB. In addition, through the same command window, we are able to view our results. Figure 2 displays the MATLAB main features or environment.

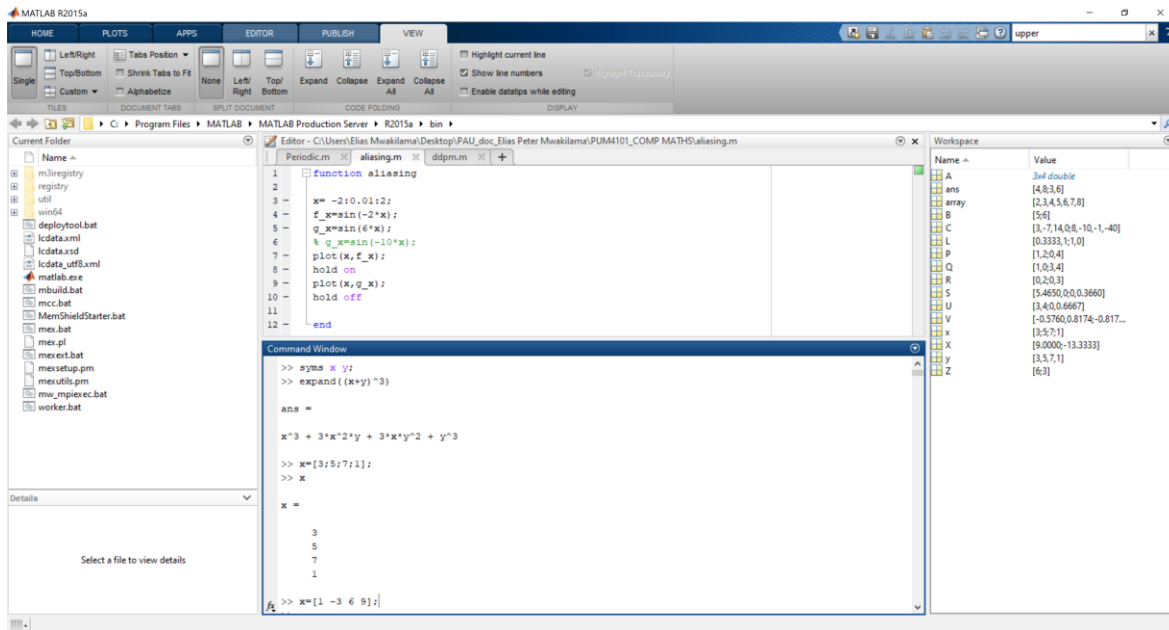


Figure 2: MATLAB Environment

We shall then discuss each of the windows separately, including the *variables* feature.

#### a) Command window

This is the main window. The command window (Figure 3) is used to enter variables and to run functions and M-file scripts. Again, it is through this same window that immediate computational results are displayed after issuing a command or an instruction to execute something.



Figure 3: MATLAB command window

This is where we type all our commands or instructions, appearing after the command prompt “>>”. For example, defining the matrix

$$A = \begin{bmatrix} 2 & 3 & -5 & 0 \\ 1 & 2 & 4 & 8 \\ 7 & 11 & 3 & 6 \end{bmatrix}$$

we type as and hit “enter” button to allow MATLAB display the result

```
>> A=[2 3 -5 0;1 2 4 8;7 11 3 6]
A =
     2     3    -5     0
     1     2     4     8
     7    11     3     6
```

Likewise, if we need to do a basic arithmetic operation such as  $2 + 5 = 5$ , we type

```
>> 2+5
ans =

     7
```

In our matrix example, we call “A” a *variable* which stores the defined matrix, and so is the *default* variable “ans” which stores the result of adding the two numbers “5” and “7”. Thus, we note that in MATLAB, it is important to define variable which is used to store the result such that it can be referred to in the next stages. Without defining or specifying such a variable, by default MATLAB considers using the variable name “ans”.

## b) Variables

In MATLAB and also in most other programming languages, variables are defined with the assignment operator “=”. As such, we say that MATLAB is dynamically typed, that is, variables can be assigned without declaring their type, and that their types can change. As such, values can come from either constants, or from computation involving values of other variables, or from the output of a given function. For example

```
>> x=17
x =
    17
>> x='hat'
x =
    hat
>> x=[3*4 pi]
x =
    12.0000    3.1416
>> y=3*sin(x)
y =
   -1.6097    0.0000
```

Unlike with many other programming languages, where the semicolon is used to terminate the commands, with MATLAB, a semicolon serves to suppress the output of the line that it concludes. Thus, if we type a semicolon (;) at the end of a command or instruction, MATLAB does not display the result or rather it does not respond. However, inside a matrix, a semicolon acts as a row separator. For example(s)

```
>> A=[2 3 -5;1 2 4;7 11 3]
A =
     2     3    -5
     1     2     4
     7    11     3

>> A=[2 3 -5;1 2 4;7 11 3];

>>
```

We must take note that *variables are case sensitive* and they must start with a letter, followed by either letters, digits or underscores. Each variable name is defined as a matrix and stored in the “workspace window”. Besides defining oneself variables, MATLAB has special variables which should be executed subject to a particular instruction. We present a few of these in Table 1.

Table 1: MATLAB special variables

| Variable | Meaning                                          |
|----------|--------------------------------------------------|
| ans      | Default variable name for results or answer      |
| pi       | Value for $\pi$                                  |
| eps      | Smallest incremental number                      |
| inf      | Infinity                                         |
| NaN      | Not a number e.g. when you have 0/0              |
| i and j  | $i=j=\text{square root of } -1$ , imaginary unit |
| realmin  | The smallest usable positive real number         |
| realmax  | The largest usable positive real number          |

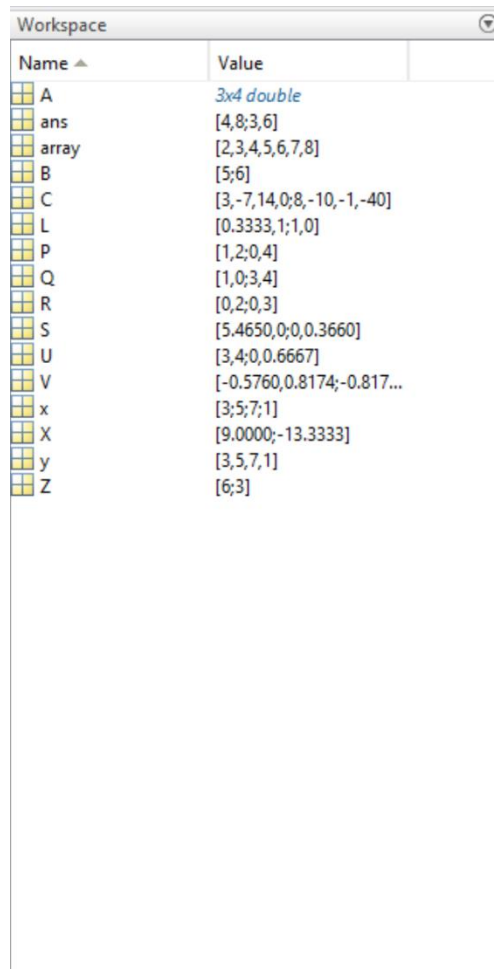
In addition to the specified roles of command prompt “>>” and semicolon “;”, other useful symbols are discussed in Table 2.

Table 2: Other MATLAB Symbols

| Symbol | Usage in MATLAB                                      |
|--------|------------------------------------------------------|
| >>     | prompt                                               |
| ...    | Continue statement on the next line                  |
| ,      | Separate statements or data                          |
| %      | Start comment which ends at the end of line          |
| ;      | Suppress output or used as row separator in a matrix |
| :      | Specify range                                        |

### c) Workspace

The workspace lists all our variables in use or used before as long as we have our MATLAB opened. Figure 4 describes such MATLAB workspace window.



The screenshot shows the MATLAB Workspace window with a table of variables. Each variable name is preceded by a small grid icon. The 'Value' column shows the data type and dimensions for each variable.

| Name ▲ | Value                     |
|--------|---------------------------|
| A      | 3x4 double                |
| ans    | [4,8;3,6]                 |
| array  | [2,3,4,5,6,7,8]           |
| B      | [5;6]                     |
| C      | [3,-7,14,0;8,-10,-1,-40]  |
| L      | [0.3333,1;1,0]            |
| P      | [1,2;0,4]                 |
| Q      | [1,0;3,4]                 |
| R      | [0,2;0,3]                 |
| S      | [5.4650,0;0,0.3660]       |
| U      | [3,4;0,0.6667]            |
| V      | [-0.5760,0.8174;-0.817... |
| x      | [3;5;7;1]                 |
| X      | [9.0000;-13.3333]         |
| y      | [3,5,7,1]                 |
| Z      | [6;3]                     |

Figure 4: MATLAB workspace window

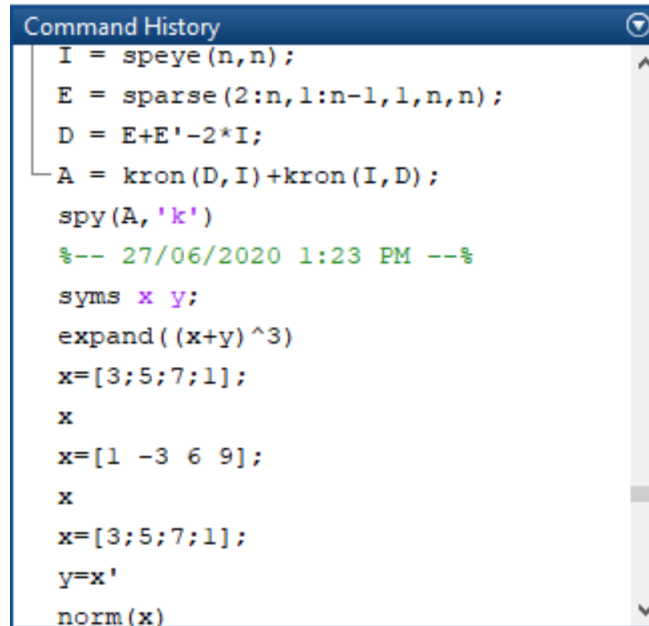
With the workspace window, we should note that by typing

- “>>who” on the command window, we can access a list of all commands being used.
- “>>whos” on the command window, we can get a list of all commands together with their current values, dimensions, etc.
- “>>clear” on the command window cleans every variable in current use and in memory.

If not cleared in memory, then all other commands in use are displayed in “command history window”.

#### d) Command History

All statements or commands we enter in the command window are stored in the Command History (see Figure 5). From the command history, we can therefore view and search for all previously run statements, as well as copy and execute selected statements. Further, we can create an M-file from the selected statements. In other words, the command history serves as a recycle bin!



```
Command History
I = speye(n,n);
E = sparse(2:n,1:n-1,1,n,n);
D = E+E'-2*I;
A = kron(D,I)+kron(I,D);
spy(A,'k')
%-- 27/06/2020 1:23 PM --%
syms x y;
expand((x+y)^3)
x=[3;5;7;1];
x
x=[1 -3 6 9];
x
x=[3;5;7;1];
y=x'
norm(x)
```

Figure 5: MATLAB command history

However, when doing programming in MATLAB, the M-files are stored and displayed in the “current directory” or more recently referred to as “current folder”.

#### e) Current Directory

The Current Directory window or recently called “The Current Folder” window lists all M-files, etc. during programming process (see Figure 6).

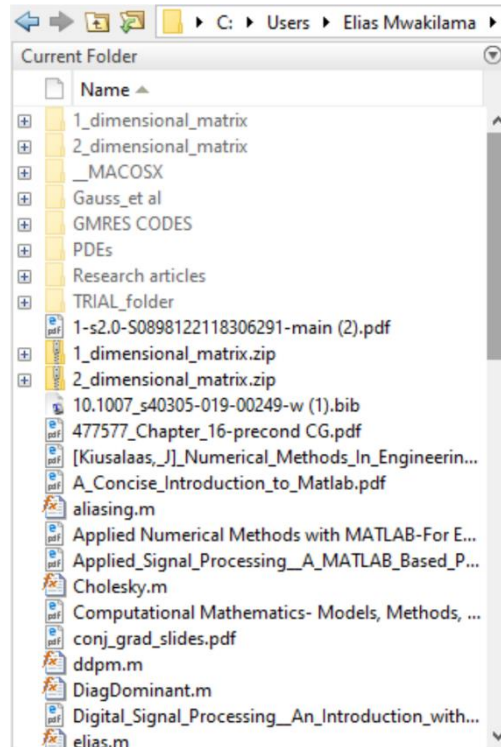


Figure 6: MATLAB Current Directory

## 2. Useful commands in MATLAB

We have to end this section with some very useful commands presented in Table X. Most important of all is the “Help” command which will help us learn more about MATLAB.

Table 3: Some useful MATLAB commands

| Command              | Description                                                                        |
|----------------------|------------------------------------------------------------------------------------|
| help                 | Gives us general MATLAB features and different areas we may explore                |
| help <b>V</b>        | Gives us help on the specified subject or item “ <b>V</b> ” in MATLAB              |
| who, whos            | Provides us with lists of all variables including their value types for the latter |
| clear                | Clears all variables in the workspace                                              |
| clear <b>V</b>       | Clears the specified variable or item “ <b>V</b> ” in the workspace                |
| what                 | Lists all M-files in the working folder                                            |
| lookfor              | Search for keyword in MATLAB m-file                                                |
| clc                  | Clear the command window                                                           |
| dir or ls            | Lists all files in the current directory or current folder                         |
| cd <b>T</b>          | Change current directory to <b>T</b>                                               |
| pwd                  | Show current directory or current folder                                           |
| type <i>Exercise</i> | Display <i>Exercise.m</i> in the command window                                    |
| delete <i>test</i>   | Delete <i>test.m</i>                                                               |

For example, if one types “*help*” on the search space, we get a MATLAB help comprehensive window as shown in Figure 7. You may also type this “*help*” command in the command window and get a list of MATLAB files from which to seek your help from.

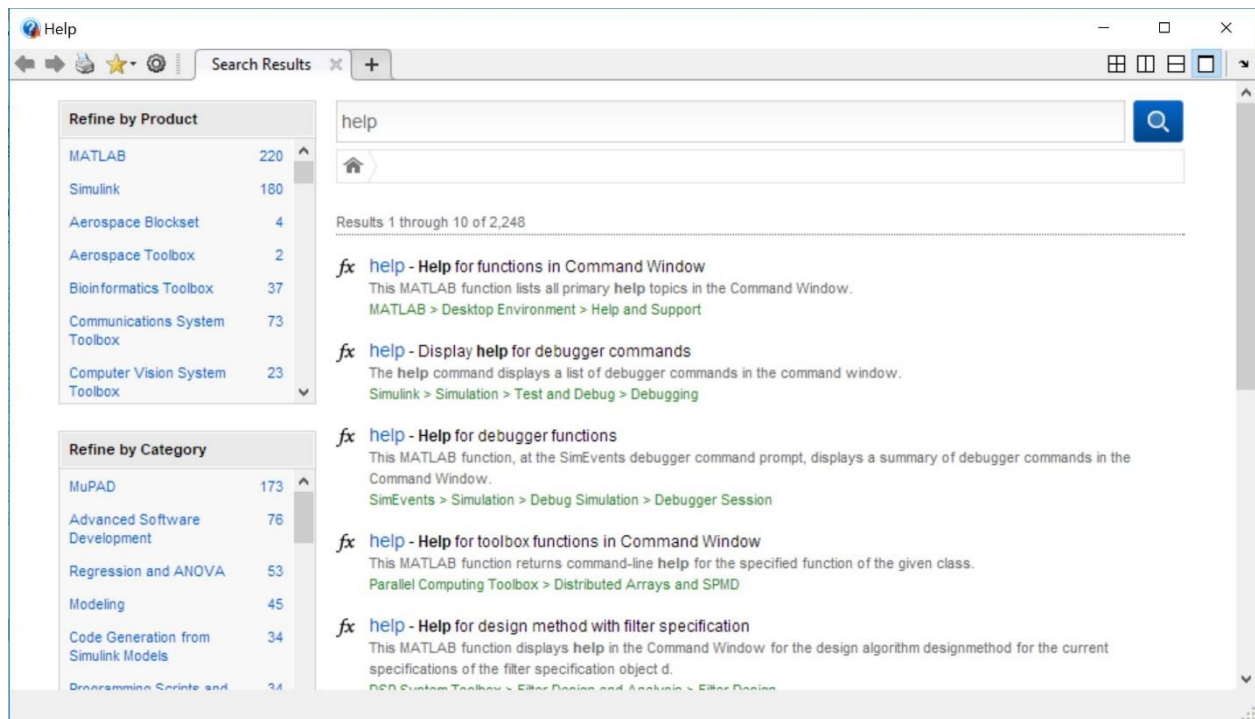


Figure 7: MATLAB Help comprehensive system window

### 3. Numerical and Algebraic expressions in MATLAB

Mostly, MATLAB operations are based on representation of every variable or element you define as a matrix. Therefore, it is of importance to know well how matrices are defined. In MATLAB, Matrices can be defined by separating the elements of a row with a blank space or comma and using a semicolon to terminate each row. The list of elements should be square brackets: []. For example, a matrix of the form

$$A = \begin{bmatrix} 2 & 3 & -5 & 0 \\ 1 & 2 & 4 & 8 \\ 7 & 11 & 3 & 6 \end{bmatrix}$$

can be represented as

```
>> A=[2 3 -5 0;1 2 4 8;7 11 3 6]
A =
     2     3    -5     0
     1     2     4     8
     7    11     3     6
```



On the other hand, parentheses: ( ) are used to access elements and subarrays. Subarrays are as well used to denote a function argument list. For example, entries “-5” and “11” can be accessed by

```
>> A(1,3)
ans =
    -5
>> A(3,2)
ans =
    11
```

Sets of matrix indices can be specified by expressions of arrays such as “1:4”, which evaluates an array [1, 2, 3, 4]. For instance, a submatrix taken from rows 2 through 3, and columns 3 through 4 may be written as:

```
>> A(2:3, 3:4)
ans =
     4     8
     3     6
```

This previous example tells us that with MATLAB, we can generate an array of sequence of numbers with a common difference by use of an *incremental function*. Thus, with a syntax “**initial:increment:terminator**” one can generate a list of number that begins with a specified “initial” and ends with “terminator” with “increment” being the common difference. This absolutely reminds us of how to define an arithmetic progression right! For example

```
>> array=2:2:8
array =
     2     4     6     8
```

creates a variable “array” which stores in a sequence of number 2 to 8, with an increment of 2. If no increment is specified, MATLAB considers a default increment to be “one” such that the new array becomes

```
>> array=2:8
array =
     2     3     4     5     6     7     8
```

In some cases, we may use same parentheses and colon to access either entire specified row or column of a given matrix. For example, given a matrix

$$A = \begin{bmatrix} 2 & 3 & -5 & 0 \\ 1 & 2 & 4 & 8 \\ 7 & 11 & 3 & 6 \end{bmatrix}$$

we can access column two of matrix A by typing the command **col2** = **A(:,2)**, e.g. in MATLAB

```
>> A=[2 3 -5 0;1 2 4 8;7 11 3 6]
A =
     2     3    -5     0
     1     2     4     8
     7    11     3     6
>> col2=A(:,2)
col2 =
     3
     2
    11
```

Alternatively, we access the entire row two by typing the command **row2** = **A(2,:)**, e.g. in MATLAB, we have

```
>> A=[2 3 -5 0;1 2 4 8;7 11 3 6]
A =
     2     3    -5     0
     1     2     4     8
     7    11     3     6
>> row2=A(2,:)
row2 =
     1     2     4     8
```

We must take note that in MATLAB, indexing is one-based, which is the usual convention for matrices in mathematics. This is typical for programming languages, whose arrays often start with zero such as in C++. Some useful math and assignment MATLAB operators are in Table 4.

Table 4: MATLAB Math and Assignment operator

| Description       | Symbolic expression                                    |
|-------------------|--------------------------------------------------------|
| Power or exponent | <sup>^</sup> or <sup>.^</sup> i.e. $a^b$ or $a.^b$     |
| Multiplication    | <sup>*</sup> or <sup>.*</sup> i.e. $a*b$ or $a.*b$     |
| Division          | <sup>/</sup> or <sup>./</sup> i.e. $a/b$ or $a./b$     |
| Assignment        | <sup>=</sup> i.e. $a=b$ ‘assign b to a’                |
| Specify range     | <sup>:</sup> i.e. $a:b$ ‘generate numbers from a to b’ |
| Comment a line    | <sup>%</sup> i.e. %mango, ‘comments word mango’        |

#### 4. Scripts and functions M-files in MATLAB

In Unit three of this module, we shall spend some time on how to write a program that executes a series of mathematical steps. Such a program is called a “Script” and is recorded as an “M-file”. M-files are text files that contain MATLAB code. We will see that we can use the MATLAB Editor or another text editor such as Notepad to create a file containing the same statements that we would type at the MATLAB command line. Then this file can be saved under a name of our choice ending in “.m”. So, just as we are able to recognize “.xls” and “.doc” as Excel and Microsoft Word files respectively in a particular computer’s folder, so do we recognize “.m” file as a MATLAB file.

##### a. Scripts

Thus, whilst in MATLAB environment (Figure 2), we can create a new m-file from the File/Home menu → New menu or the New Script button on the Tool bar.

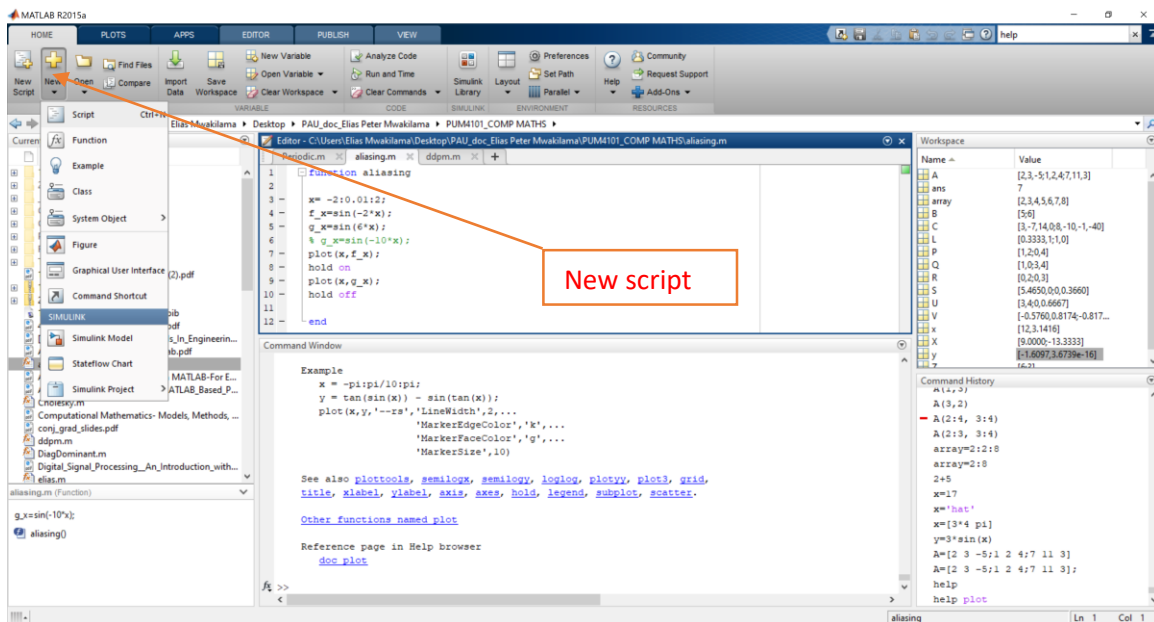


Figure 8: Creating a New Script file

By this process we get a blank script m-file as shown in Figure 9 titled New Editor.

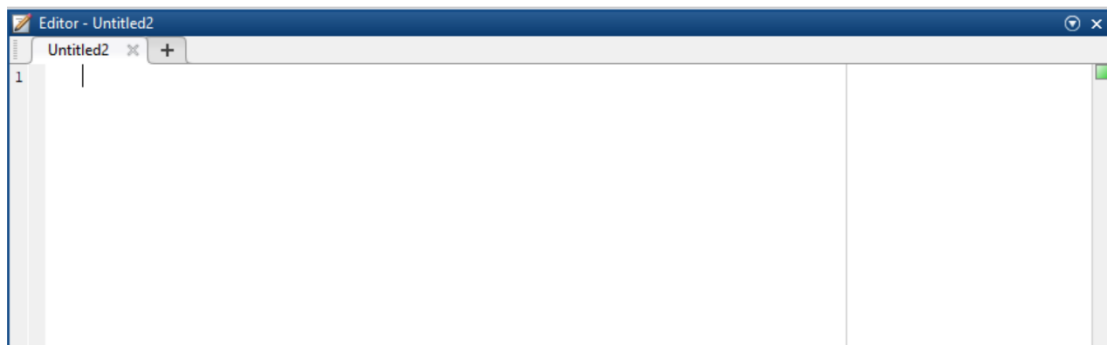


Figure 9: An example of a script file

Later, when we start writing our own MATLAB programming files, we should be able to generate and run our m-files as shown in Figure 10.

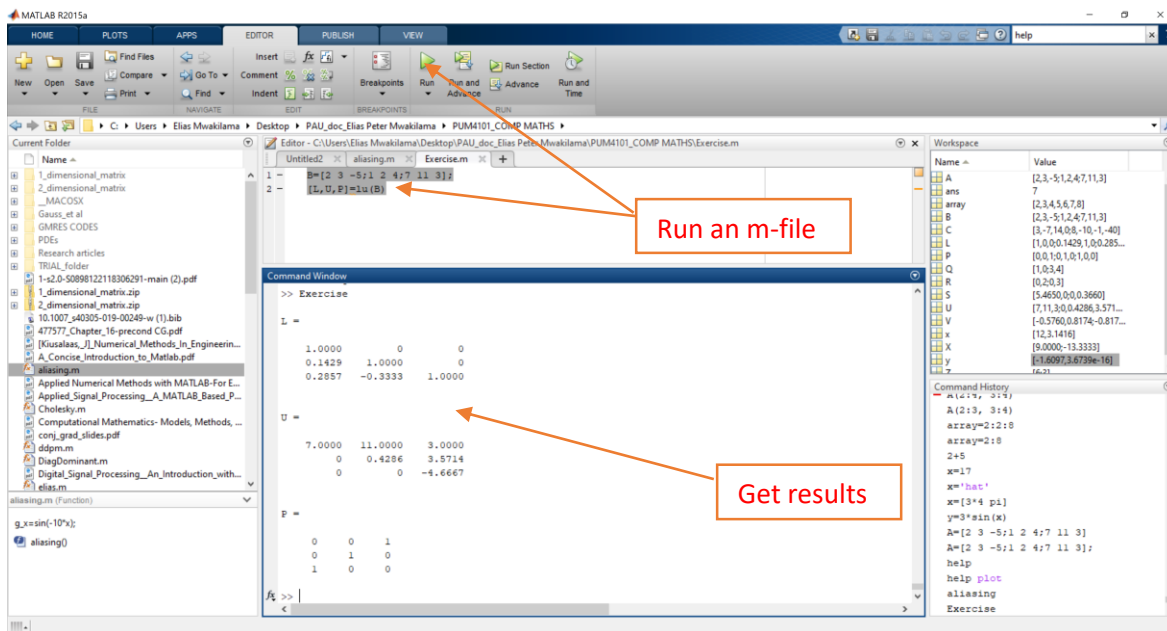


Figure 10: Running a script m-file

## b. functions M-files

Later, we shall see that we can create own functions, save, and run them as m-files (Figure 11). This is very crucial during the process of writing large programming files whose role is to execute key programs.

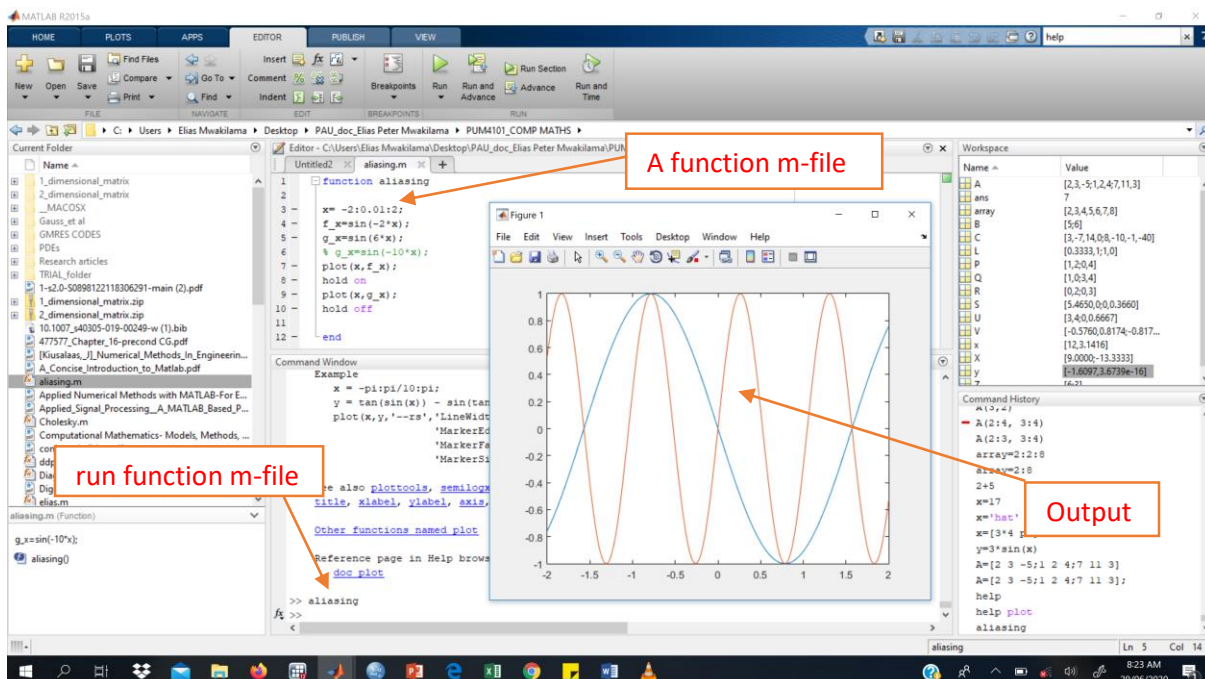


Figure 11: Creating, saving and running function m-files