MATLAB and SIMULINK (A Basic Understanding for Engineers)

MATLAB and SIMULINK (A Basic Understanding for Engineers)

By

Pooja Mohindru and Pankaj Mohindru

Cambridge
Scholars
Publishing

MATLAB and SIMULINK (A Basic Understanding for Engineers)

By Pooja Mohindru and Pankaj Mohindru

This book first published 2020

Cambridge Scholars Publishing

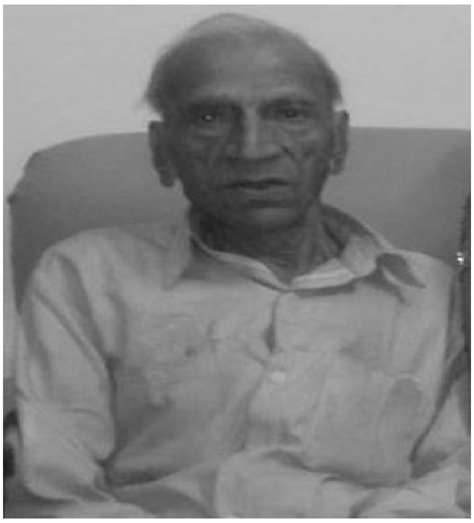
Lady Stephenson Library, Newcastle upon Tyne, NE6 2PA, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN (10): 1-5275-5005-2
ISBN (13): 978-1-5275-5005-6

In fond memory of

Smt. Hardevi Mehta



Sh. Lal Chand Mehta

TABLE OF CONTENTS

ABSTRACT

MATLAB (the **Matrix Laboratory)** is a computer system based on dealing with matrices. It was originally designed to assist in scientific and engineering problem solving by doing numerical computations with matrices and vectors. SIMULINK (Simulation and Link) is integrated with MATLAB and a graphical extension of MATLAB for generating and simulating the block-based models of the desired systems and doing their analysis. The purpose of this book is to make the students learn MATLAB and SIMULINK easily and quickly. MATLAB can be used for a range of applications which includes signal processing, analog and digital communications, image and video processing, control systems, computational finance, computational biology, etc. More than a million engineers, researchers, scientists in industry and academia use MATLAB as the language of technical computing. It is a high level language and an interactive environment for numerical computation, visualization, and programming. It helps in analyzing data, developing several algorithms and creating models and applications. The language, tools and built-in math functions available in MATLAB enable the users to explore multiple approaches and reach a solution faster than they can solve using spreadsheets or traditional programming languages. On the other hand, SIMULINK is a block diagram environment used for performing multi-domain simulation and analyzing model based design of various systems. It supports a system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems.

The basic MATLAB data element is a matrix. Its commands are expressed in notations which are almost similar to the form that are used in mathematics and engineering. For instance, the mathematical expression b = y x is written as `b = y*x in MATLAB`, where y and x can be any values or matrices entered by the user. In order to solve for x in terms of y and b, the command is written as `x = b/y or y\b`. There is no need to program matrix operations explicitly like multiplication or inversion which makes MATLAB much quicker in solving problems than any other high-level language. There are hundreds of built-in functions that come with MATLAB along with many optional toolboxes/blocksets that can be used for specific purposes such as machine learning, image processing, control systems, signal processing, communication, bioinformatics, etc.

MATLAB is the most widely used software today in academics and industrial community. It provides simple ways for representing the data or information along with a large number of plotting routines to assist the users in presenting the outcomes of their research in various fields easily and effortlessly. It is very good in manipulating large sets of numbers and doing matrix operations and that's why named MATLAB. Symbolic computations can also be accomplished easily using it. MATLAB can also communicate with an external hardware and be programmed with the hardware.

The book provides all the basic knowledge required for using the MATLAB features and commands in doing computations. MATLAB is a fundamental and leading programming language that deals mainly with matrices/arrays. A scalar is a 1x1 matrix and a row vector of length say 5, is a 1x5 matrix. It is an environment for performing different types of calculations and simulations quickly. MATLAB was developed in the late 70s and early 80s by Cleve Moler, a professor of mathematics and computer science. In the earliest version of MATLAB, there were about 80 commands available for doing matrix/array calculations swiftly and with a minimal programming. Now, more than thousands of commands are available in the latest version in order to perform several types of calculations easily and effectively.

SIMULINK is an add-on feature in MATLAB that allows the users (engineers, science graduates, etc.) to simulate various systems by combining blocks of several types. To assist the user in simulating and analyzing the performance of various systems using block diagrams in SIMULINK, the book provides an introduction to using SIMULINK in various engineering and science applications. It covers a comprehensive explanation of modeling and simulation of dynamic systems using SIMULINK. It is integrated in MATLAB to offer a model based simulation and analysis of a dynamic system within its own Graphical User Interface (GUI) environment. SIMULINK offers simple click-and-drag mouse operations which helps in constructing block-based models. It also includes a set of block Sub-libraries and toolboxes for both linear and nonlinear analysis. Being an integral part of MATLAB, it is easy to switch between the two environments during an analysis process so that the user can take full advantage of their features. The book contains all the related information for using the MATLAB and SIMULINK tools in doing scientific calculations, solving numeric problems and simulating systems.

The topics covered in the book are explained in a very simple language and lucid manner to catch the interest of the readers. The book is written by authors who are articulate and dogged persons. The efforts put in by the authors while writing the book definitely proves to be very fruitful to the users who are the beginners and never used the software before.

MATLAB tool is a must learn skill for all the professionals who want to develop a bright career in engineering, science or any related field. An engineer must possess excellent **programming skills** with MATLAB in order to build a highly progressive career in any discipline. The book aims to improve graduates/postgraduates employability by providing them with the key hunted skills in computer simulation by using the MATLAB and SIMULINK programming environments. MATLAB and SIMULINK are used worldwide in industries as well as research institutes for developing up-to-date or technologically advanced solutions to many engineering, science and research problems. Graduates/Postgraduates with the requisite skills in MATLAB and SIMULINK can get employment and gain lucrative salary in the electronics/electrical industry, automotive industry, aerospace industry and power industry. Today, graduates/postgraduates with good programming skills in MATLAB and SIMULINK are also required in other industrial sectors such as biomedical, utilities & energy, earth & ocean Sciences as well as finance.

ACKNOWLEDGEMENTS

First and foremost, the authors thank the almighty for providing strength, ability, knowledge and opportunity to write the book. They express gratitude to the God for blessing them with an intellectual insight, temperament, energy and passion to pour their heart and soul into writing the book.

The authors deeply acknowledge the assistance received from Dr. Manjit Singh Patterh, Director IQAC, and Dr Manjeet Singh Bhamrah, Head of the E.C.E. department. They are extremely thankful to Dr. S.S. Bhatia and Dr. Rajesh Khanna, Thapar University, Patiala for enlightening them and always boosting their enthusiasm. They are deeply thankful to Dr. Ashok Goel for his guidance and motivation. They also want to thank all colleagues for giving immense support.

Last but not the least, with great pleasure the authors want to express a sincere recognition to all the family members especially the little ones for being alongside all throughout the long journey of the book and providing untiring support, love and encouragement. A big thank you to all those who inspire the authors to keep going on a path to success and always ensure them that a good time keeps flowing.

The author Dr. Pooja takes pride in dedicating the book to her incredibly hard working and affectionate Grandparents, **Sh. Lal Chand Mehta** and **Smt. Hardevi Mehta.** The author is deeply indebted to them for showering their blessings from the heaven.

CHAPTER 1

MATLAB OVERVIEW FOR BEGINNERS

1.1 Introduction

MATLAB is a fourth generation high-level programming language and software environment that is widely used by science students, engineers, industry professionals and research scientists for doing numerical or symbolic computations, performing data analysis, writing long codes or complicated algorithms, etc. MATLAB is essentially a highly sophisticated calculator at a basic level. It is an interactive system which assists the user in solving many technical and scientific problems quite easily. The basic data elements to operate on in MATLAB are arrays which can be scalars, vectors or matrices. Therefore, it is known as the matrix-based language which allows several operations like manipulating arrays, plotting graphs for various data sets or functions, implementing algorithms (sometimes lengthy as well as difficult), designing user interfaces, interfacing with programs written in other languages and modeling real-world systems. MATLAB is easier to start and provides several built-in functions, toolboxes and blocksets so that it can be extensively used in industries and academics to perform various computational tasks. MATLAB supports an array without the need to declare its dimensions. It is not required to perform low-level tasks like declaring variables, specifying data types and allocating memory in MATLAB which makes it easier to use.

The essential components of MATLAB are as follows:

(A) Computing Environment

MATLAB provides an interactive working environment called the Desktop which further contains the Command Window, Workspace, Editor and Debugger window, Help Browsers, etc.

(B) Library of Math Built-in Functions

MATLAB has an extensive collection of mathematical functions ranging from basic (like sum, difference, multiplication, division, trigonometric functions, complex or polynomial arithmetic, etc.) to more sophisticated functions (like matrix Inverse, Eigen values, Bessel functions, fast Fourier transforms, Laplace transforms, Symbolic, etc.), and therefore is the most preferred system for technical and scientific computing.

(C) Array Programming Environment

It is a matrix-based programming language which makes linear algebra programming quite simple. It is equipped with the control flow statements, loops, functions and data structures having the exact syntaxes. It also includes input/output and object-oriented programming features.

(D) Graphical Display and Handling Features

MATLAB provides a wide-ranging set of advanced graphics display features to handle and display vectors and matrices in a graphical form. In addition, it also includes various high-level functions for data (two-dimensional or three-dimensional) visualization, image processing and animation. It also provides several tools and facilities which can customize the appearance of the generated graph as well as generate a graphical user interface for any MATLAB application.

(E) External Interfacing MATLAB supports an interfacing with external hardware applications.

(F) Toolboxes and Blocksets

Many toolboxes and blocksets are installed in it to work with different kinds of problems.

1.1.1 Usefulness of MATLAB

MATLAB is a powerful computing system which is designed for performing array computations quite easily. The name MATLAB stands for Matrix Laboratory. It uses matrix or vector notations which makes complex mathematical expressions or equations quite simple and the user can easily handle the scientific and engineering problems. A matrix is a rectangular array of real or complex numbers. A single element matrix i.e. a matrix with only one element having dimensions of one row-by-one column is called a scalar. Matrices with only one row or one column are called as row and column vectors, respectively. MATLAB works with a complete matrix simultaneously, whereas the other high-level programming languages work with only one number at a time. This feature removes the necessity of having unnecessary loops and eliminates repetition of the same statements in MATLAB which further makes a code to the point, brief and easily understandable. MATLAB contains an online help system so that the user can obtain all the needed information just by clicking the Help button on the Desktop toolbar or the Help menu in any tool.

MATLAB is used in:

- Dealing with Matrices and Vectors
- 2-D and 3-D Graph Plotting
- Linear Algebra and Algebraic Equations
- Non-linear Functions
- Statistics and Data Analysis
- Calculus and Differential Equations
- Numerical and Symbolic Integration/Differentiation
- Working with Integral Transforms
- Dealing with Complex-valued Functions

MATLAB is an extensively used computational tool in the fields of physics, chemistry, math and all engineering streams. MATLAB applications include a wide range as follows:

- Signal Processing and Communications
- Image and Video Processing
- Control Systems, Robotics and Mechanical Engineering
- Measurements and Instrumentation
- Finance Industries
- Biology and Biomedical Field
- Earth and Ocean Sciences

Thus, MATLAB is used in every element of data computing and visualizing.

1.2 To Start with MATLAB

After installing MATLAB, the user can begin working with it by double clicking with left mouse button on the MATLAB-shortcut icon which can be seen on the computers desktop. Alternatively, the users can also begin with MATLAB by selecting the **Start** menu on the computers desktop and then go to the **All Programs** -> **MATLAB** options. After a while, the MATLAB Desktop (the Main) window appears on the screen which contains a set of menus or tools that forms the base for doing computations with MATLAB both interactively (like a calculator) and non-interactively (as a programming language). The MATLAB **Desktop** consists of the four basic windows: Command Window, Workspace, Current Directory and Command History. It has two more windows known as the Figure and Editor window that can be invoked by the user when needed. The Figure window is used for plotting functions graphically and pops up only whenever we give the plot related commands and the Editor window is used for writing and editing program files (the M-files or Script files). In an older version of MATLAB, the Editor window can be opened from the set of menus/tools by clicking on the pull-down **File** menu and selecting the **New** -> **M-file** options. Whereas, the Editor window is invoked in the latest version by selecting the **New Script** menu icon (under the HOME tab) as shown in Fig. 1.1 below.

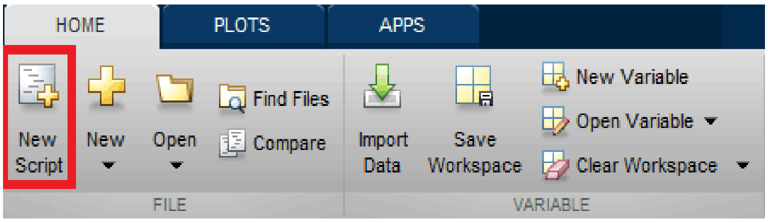
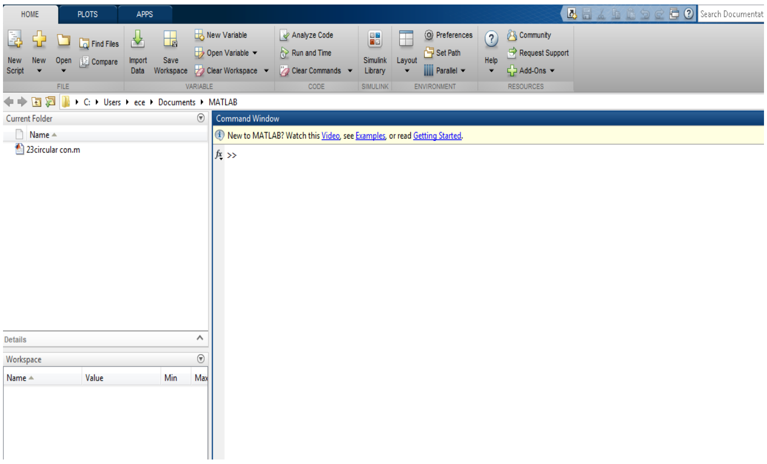


Figure 1.1: Appearance of a MATLAB Toolstrip

(**Note**: The M-file or Script file is a simple text file where we can write a code and add a series of commands. When the M-file is run, all the statements of a code are executed sequentially in the same way as they are executed at the command-line prompt and the results are printed in the Command Window)

To begin working with MATLAB, install the latest version of MATLAB and start it in order to see its GUI (Graphical User Interface) on the computer screen as shown below in Fig. 1.2. The user should always keep in mind that various drag and drop windows can be positioned at different locations in different versions of MATLAB. The toolbar icons can also be arranged differently in MATLAB, and therefore the versions may look different from each other on the screen. Thus, a slightly different GUI appears on the screen for an older version of MATLAB.

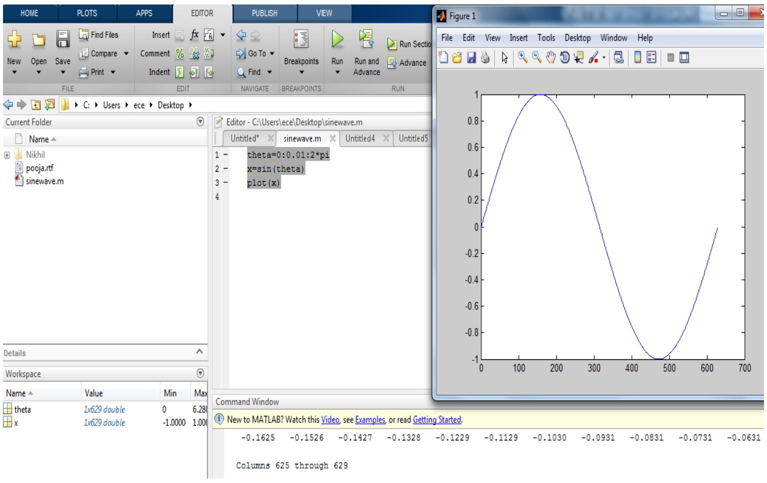


New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
fx >>
```

Figure 1.2: Appearance and Organization of the Desktop

The main screen window includes the three basic window panes: Command Window (the panel for writing instructions), Workspace (the memory window pane) and Current Folder (the browser window pane) as can be seen in Fig. 1.2 above. Whereas, the Editor as well as Figure windows (as shown in Fig. 1.3 below) appear on the screen only when invoked by the user.



```
theta=0:0.01:2*pi
x=sin(theta)
plot(x)
```

Name	Value	Min	Max
theta	1x629 double	0	6.28
x	1x629 double	-1.0000	1.00

New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
   -0.1625   -0.1526   -0.1427   -0.1328   -0.1229   -0.1129   -0.1030   -0.0931   -0.0831   -0.0731   -0.0631

  Columns 625 through 629
```

Figure 1.3: MATLAB GUI Depicting its Five Windows

The toolstrip available in the latest version of MATLAB includes a series of tabs (sets of tools/menus) to perform many functions. These tabs are further divided into the sections that contain a series of buttons, drop-down menus and other user interface elements which are used to execute and control different operations.

A brief description of the tabs and their subgroups is as follows:

The **FILE** section within the HOME tab is used to work with program files. It mainly provides us the control tools which includes the **New Script** option (to create a new dot m file with an extension .m for programming in MATLAB), the **New** option (a drop-down menu to create new Script files, graph plotting files, function files, SIMULINK models, etc.) and the **Open** option (to open already existing projects or files for re-execution). The section allows all the file related operations such as creating new scripts, opening already existing files, comparing two files and many more.

The HOME tab with the **VARIABLE** section is used to handle variables. It makes working with variables easy by allowing several options for defining new variables, assigning values to them, importing variables from other program files and saving current working variables into a program file for future use. The third section is the **CODE** which is used to operate on the generated codes. It allows the user to handle various commands, run the generated codes and analyze them. The **SIMULINK** is the fourth subgroup which allows the model generation (the block diagram based representation) and simulation of various systems. It provides the **Simulink Library Browser** icon to browse and search the relevant blocks' Sub-libraries for building models to solve various problems. The fifth section is the **ENVIRONMENT** which allows us to modify a current working environment and path. Here, the Desktop layout can be changed by clicking the **Layout** drop-down menu and setting preferences (such as fonts, keyboard shortcuts, initial working folder, etc.) to optimize the basic layout. The last one is the **RESOURCES** section which is used to obtain the information related to using the tools and accessing the MathWorks worldwide community.

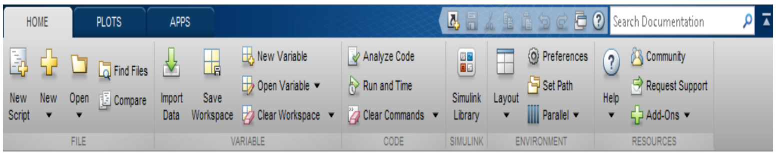
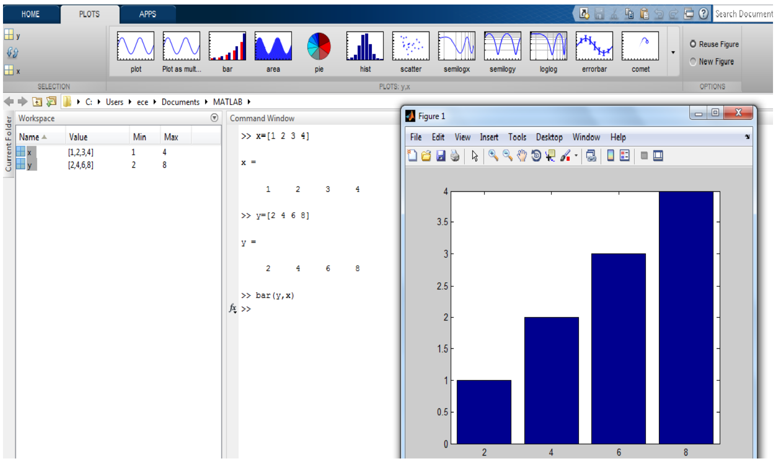


Figure 1.4(a): The HOME Tab

The three tabs namely the HOME tab, PLOTS tab and APPS tab (as can be seen in Fig. 1.2 and Fig. 1.4(a)) are called global tabs as they are always present on the screen once the MATLAB Desktop is open. They are not affected by what the user is doing in MATLAB. All the common operations like creating program files, importing data, managing the Workspace and setting the Desktop layout can be performed in the HOME tab.

The PLOTS tab is used to create graphical plots in MATLAB as shown below in Fig. 1.4(b). Whenever we click on the PLOTS tab, it displays a gallery of several plots (line, bar, pie, loglog, etc.) available in MATLAB and toolboxes that are installed by the user. By clicking on a downward facing arrow (black triangle) on the far right, a full size of the plot gallery can be made available with many more choices.



Name	Value	Min	Max
x	[1,2,3,4]	1	4
y	[2,4,6,8]	2	8

```
>> x=[1 2 3 4]

x =

     1     2     3     4

>> y=[2 4 6 8]

y =

     2     4     6     8

>> bar(y,x)
fx >>
```

Figure 1.4(b): The PLOTS Tab

In order to create a plot by choosing any option from the gallery, first select the entered variables from the Workspace window that we want to plot. After that, choose a type of visualization that best suits to represent the selected data in an effective and presentable way. In order to select more than one variable from the Workspace, click a left mouse button on the second variable while pressing a keyboard **Ctrl** key. All the selected variables look like as shown in Fig. 1.4(b). The plotting command is also displayed at the prompt along with the generated plot in the Figure window.

The last of the global tabs is the APPS tab shown below in Fig. 1.5, where interactive MATLAB applications are executed.

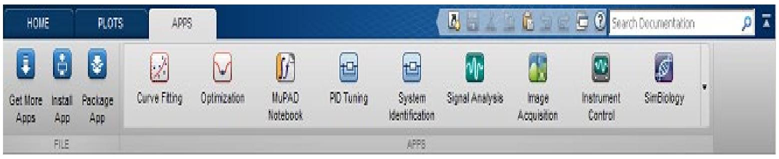


Figure 1.5: The APPS Tab

The Figure window (where graphical plots appear) in MATLAB as can be seen in Fig. 1.3, contains many useful actions in its menus/tools like the Zooming in and out, Rotating 3D axes, Copying & pasting, Plot Edit Mode, Plot tools (an interactive plotting), Figure Palette, Data Cursor Mode, etc.

It can be stated that MATLAB is equally a programming language and a software environment which allows the user to execute the statements/commands from the command-line prompt or generate lengthy programs/codes and user-defined functions in the Editor window using the M-file (a plain text file with the .m extension and containing a series of commands).

1.3 Creating Variables and Assigning Data

A variable can be created in MATLAB and a required data (or an information) can be assigned to it either by typing the assignment statement at the prompt (>>) in the Command Window or into the M-file in the Editor window. The Workspace window can also be used to enter data values in the variables using a spreadsheet format. The MATLAB variable is any name given by the user during a session to refer to a definite location in the memory where MATLAB stores the assigned expression/information on the right side of the statement (if containing no tasks to be performed) or its computed value when an evaluation is needed so that the saved value can be accessed by typing the variable name associated with it. In MATLAB, a variable may be used by assigning an expression which may contain a numeric value, multiple arithmetic operations with numeric values, a numeric array, text string scalars/arrays or other data types and is interpreted as a single data element. The MATLAB Workspace consists of all the variables entered by the user. Therefore, a variable name must come before the assigned data so that it can be stored in the Workspace.

There are some guidelines which must be followed while using MATLAB for computing as discussed below.

• **How to name variables:** The MATLAB variables are case-sensitive that must start with a letter. They can be named using any combination of letters, numbers and underscore with maximum allowed characters equal to 31 and no spacing within them. Punctuation marks must never be used while naming the variables. Also, there is a library of reserved keywords or built-in functions in that cannot be used for naming the MATLAB variables. All the MATLAB built-in functions appear in a different color (blue) when typed in program files (see Section 2.9). It is important to note that in MATLAB there is no need to declare the type of a variable before assigning a data to it. MATLAB itself sets a variable to have the same type as is associated with the data assigned to it.

• **Assigning different data types to variables in the Command Window:**

MATLAB accepts a data in the form of an expression that generally contains an array. An array usually contains a number, a list of numbers (arranged into multiple rows, multiple columns or both), a character and a list of many characters or words. In MATLAB, an array is assigned to a valid variable name. For instance, if we type the x=6 in MATLAB, the character x represents a variable name whereas the value 6 is a number assigned to the x and an equal sign is the assignment operator which assigns the scalar number 6 to the variable x.

Few examples of the MATLAB statements (variable = expression) are shown below.

Type in MATLAB:

```
A=3;      % Assigns the scalar value 3 (1x1 array containing just one number) to the variable A in MATLAB

A=[1 2 3 4];      % Assigns the numbers 1, 2, 3 and 4 to a row vector (1x4 array) with the variable name A
```

(**Note**: Spaces or commas within square brackets separate the entries column-wise on the same row)

```
A=[1; 2];      % Generates the column vector A (2x1 array)

A=[1 2 3; 4 5 6];   % Generates the matrix with the variable name A (2x3 array)
```

The MATLAB expression assigned to a named variable may also contain a single operation or multiple operations on numeric/character values, the MATLAB built-in functions or user-defined functions along with input arguments, etc.

For example:

```
A=1+5+6-2;      % Returns the scalar number 10 to the variable A
```

In the MATLAB assignment statement (used for storing the result of a computation), an expression which is entered on the right hand side of an equal sign can be a numeric value or character, any valid combination of scalars (numerical values) or characters, arrays (numeric or character or string), variables, parameters, arithmetic operators and function calls.

(**Remember**: A semicolon within square brackets starts a fresh row)

(**Note**: The percent symbol (%) is used to denote a comment which means that the text written after the % sign is always ignored by MATLAB)

Thus, in order to generate the assignment statement in MATLAB, the user needs to write a variable name on the left side of an equal sign and enter an expression to be evaluated on the right side. An equal sign (the assignment operator) assigns the computed value to the variable name when the generated statement is run.

(**Note**: To save the window area, two or more MATLAB statements can be placed on one line by separating the statements with **commas** so that each statement is evaluated from left to right. In order to suppress the display of output of the multiple statements entered on the same line, separate the statements with semicolons instead of commas)

- **Execution of the statements in the Command Window:**

To print an assigned value as the output or compute the result of an assigned expression, press an Enter key (the execute/run button) in the Command Window. A variable name along with the assigned or computed value is exhibited as the output only if the user does not put a semicolon operator at the end of any statement. In other words, whenever an Enter/Return key is pressed to execute the assignment statement, a variable name along with its value appears at the Command Window in an indented form. While executing the statement in the Command Window, a semicolon should be placed at the end whenever it is not necessary to make the output result visible on the screen.

(**Remember**: When the MATLAB statement is executed with a semicolon operator at the end of the statement, the operator suppresses the display of the output but a typed or computed expression is assigned to a variable)

- **Valid syntax for assigning a data to variables**:

In MATLAB, a data (or information) is transferred to a variable through the assignment statement using an equal sign by mentioning a variable name on the left side and a data on the right side of an equal sign. When the assignment statement is executed, MATLAB evaluates an expression (at the right hand side of an equal sign) containing a data and assigns the result into a named variable (at the left hand side of an equal sign).

For example, to write the assignment statement in MATLAB, type as follows:

```
a=3+2 % Does not assume that the variable a is equal to 3+2, instead interprets as the value of 5 is assigned to the a
```

gives the result:

```
a=

5
```

But, the reverse assignment statement which is true in algebra is meaningless and not allowed in MATLAB. For example, typing as below:

```
3+2 = a
```

gives us an error message:

```
??? 3+2 = a

|

Error: Missing operator, comma, or semicolon.
```

(**Note**: The numeric value 3+2 = 5 is a fixed value, and therefore it cannot be assigned another value)

A variable entered in MATLAB always gets stored as an array in the Workspace window and can be accessed by having a look at the current Workspace. The entered variable on the left side keeps on holding its assigned value (on the right side) until another value or expression is assigned to it or its current value is erased from the system memory (the Workspace).

For example:

• **Assignment of a fresh value to an existing variable**:

(i) In MATLAB, the user can assign a new value to the same variable by typing the statements on the separate command-lines as follows:

```
x = 1;     % The value 1 is assigned to the x before using the x on the right side in the statement on the next line
x = x + 3;     % In Math it is meaningless, but in MATLAB it is substantial and makes complete sense
```

(**Note**: A variable must be assigned a fixed value before using it on the right side of an equal sign so that the MATLAB statement containing the variable itself in an expression can have a computable value)

(**Remember**: While writing the assignment statement, the user can put a space character before and after an equal sign to improve the readability as MATLAB does not answer for spaces)

The assignment operator (an equal sign) instructs MATLAB to act upon the expressions/instructions given on the right hand side and change the states of the variable x accordingly. In Mathematics, the second line is an assertive statement which carries no meaning, whereas MATLAB calculates the expression x + 3 (to the right of an equal sign) on the second line at x=1 and assigns the fresh value to the variable x by overwriting the old value.

Thus, MATLAB executes the above two statements on the separate command-lines in the following meaningful manner:

a). Firstly, it checks out whether any variable (or a memory cell) is already defined by the user with the name x. If not, MATLAB assigns the value 1 to the variable x.

b). Then, it takes out the value 1 stored in the variable x and add it to the value 3 and saves the resulting answer 4 in the same variable named x, discarding its previous value.

(**Note**: The statements, 1 = x or x+3 = x, are not allowed in MATLAB. An algebraic equation of type say, x+1=10, can only be solved for x in MATLAB by first defining the x as the symbolic variable. Also in MATLAB, double equal signs are used to represent an equation instead of an equal sign)

(ii) Alternatively, a new value can be assigned to the variable x by first clearing its previous value from the memory using the command clear x. The MATLAB Workspace can be cleared by typing the keyword **clear all** which erases all the defined variables and their stored values from the system memory. This way a fresh value can be assigned to the same variable.

(**Note**: The **clear** command is different from the **clc** command. The clc command only clears the Command Window. It cannot clear the defined variables and their values from the memory)

(**Remember**: Whenever there is a vast amount of data printing out to the screen while a program is running, printing of the most wanted results slow down which may confuse the user. To avoid this, a semicolon is placed at the end of the MATLAB statements to suppress the display of their outputs)

Thus, by default the Command Window enables the user to enter individual statements (**one-statement-per-line**) at the command-line prompt (>>) and view the generated results after executing the entered MATLAB statement on the single line. The MATLAB statement can be continued to the next line by using an **ellipsis** (…) so that a long statement that expands on multiple lines can be executed as a single statement.

The commands typed at the prompt appear in Courier font. The Command Window allows both entering variables in the command-line and plotting a data in the Figure window. When a variable is double clicked on in the Workspace, its value gets displayed in a spreadsheet in the new window. The user can write a code (a series of statements) in program files (the M-files or Function M-files with the arguments) using the Editor window in order to view the computed outputs of the statements at the Command Window prompt and plot graphs in the Figure window by using the plot related commands.

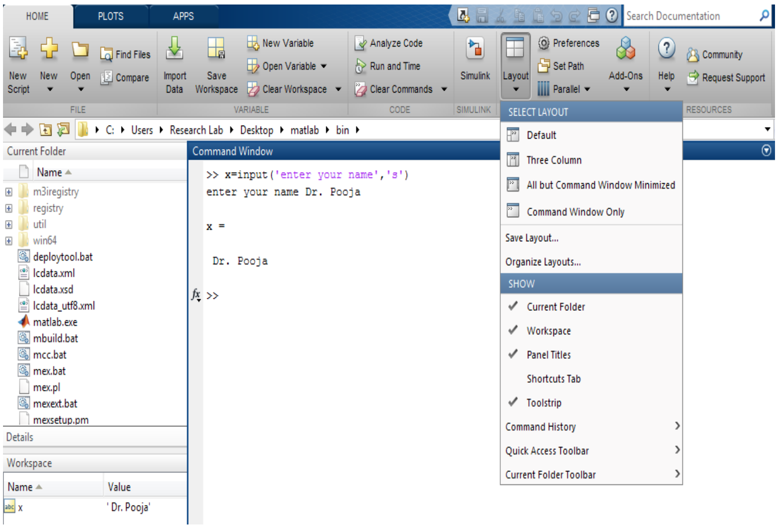
Word of warning: The symbol >> is the command-line prompt which simply indicates that MATLAB is ready to accept an input and waiting for the single line statement to be entered by the user. Therefore, a symbol >> must not be **typed** by the user anywhere within the MATLAB statements.

(**Note**: Remember to press an Enter/Return key after each line for executing the MATLAB statement typed on the single command-line as well as retrieving the command prompt for entering the fresh statement/instruction in the next command-line)

If the Command Window is not open, we can access it by selecting the Command Window from the Desktop menu (Desktop -> Desktop Layout) in an older version. To restore the Command Window to the default location or add the Command History window pane in the latest version, go to the **HOME** tab -> **ENVIRONMENT** section, then click the **Layout** drop-down menu as shown below in Fig. 1.6:

Figure 1.6: The Desktop Layout Icon

The Layout contents/sub-menus appears as shown in Fig. 1.7 below. Here, we can select one of the layout options under the **SELECT LAYOUT** option in order to change the Desktop layout as desired.



```
>> x=input('enter your name','s')
enter your name Dr. Pooja

x =

 Dr. Pooja

>>
```

Figure 1.7: The Desktop Layout Options

Under the **SHOW** option, we can select or clear the MATLAB Desktop window panes/tools that we want to show or hide, respectively.

1.4 The MATLAB Environment

After beginning MATLAB, multiple panels appear on the screen (the main window) which includes the Command Window, Workspace, Current Directory, Command History, etc. However, the other windows such as the Editor/Debugger, Figure window, Help browser pane, etc. can be invoked as desired. MATLAB contains wide-ranging sets of built-in functions, additional toolboxes as well as blocksets to work with more specialized topics like control system, fuzzy logic, neural networks, signal processing, image processing, etc. It can be used as a programming language or an interactive calculator. In a calculator mode, the built-in and toolbox functions are used for performing computations. Whereas, in a programming mode MATLAB provides an editor, debugger and profiler that enables the user to write, save and edit the Scripts (M-files) and Function program files. An expression (on the right side) assigned to a variable (on the left side) includes the MATLAB operators and built-in functions and are executed at the command-line prompt >> in the Command Window. The Command Window is useful both as a scientific calculator (to do numeric computations) and a graphing tool (to plot graphs). In order to write longer programs, it is more convenient to write codes in the M-files using a separate window (the Editor window) and then run the files to see the results in the Command Window. All the named variables are stored and visible in the Workspace window pane. The related information about any function or toolbox is made available by typing the command-line help function or using the Help browser.

1.5 Nomenclature for the Variables

Every programmer should follow the MATLAB naming conventions (a set of certain rules) while using variables at the command prompt or in program files to avoid the MATLAB error messages. The variable names entered by the user/programmer have to to be purposeful and should enhance clarity for the other users. It is a widely acknowledged practice to define the variable names in a mixed-case style (mixture of uppercase and lowercase characters) but they must start with a lowercase letter. The names cannot be started with a number or digit. They must not contain spaces or special characters. Thus, it can be concluded that the variable names distinguish between uppercase and lowercase letters, start with a lowercase letter followed by mixed-case letters, digits or underscore (to split parts of compound variable names). The MATLAB special keywords such as the end, for, if, case, switch, etc. must not be used to name any variable.

1.6 Summary of Useful and Important Features

i. The default layout of the **MATLAB Desktop** contains the **Command Window** at its right hand side, where the user can enter interactive commands at the command prompt (>>) so that the commands may be executed on the spot. The Command Window can be used for computation purposes as like a calculator is used. When MATLAB is ready to accept a new instruction, it shows the prompt >> on the screen. The user has to press an Enter (or return) key after finishing a statement/command containing multiple instructions in one line (same row) in order to see the executed output of the single statement. While executing the statement, MATLAB does not bother whether a cursor or graphic pointer (in the shape of a vertical bar) is at the end of the line statement or not. By default, the statement on the single line (same row) is executed. Thus, always remember to press enter in order to run the statement and view the assigned data or computed result of the single statement (without a semicolon at the end) as an output.

The **Command History** window displays a list of the commands which are currently or most recently entered in the Command Window and keep those commands in the list until they are deleted by the user. The **Command History** window also lists the time and date of each MATLAB session in a short date format at the top of the history of the statements entered in that session. The commands that were entered in the past can be re-executed by double clicking on them or dragging them from the Command History window. The previously entered commands can also be browsed through and executed again by scrolling arrow keys up and down.

The **Editor** window is used to create either a new program file or modify the existing ones. A new program file (the M-file or Function M-file) can be created in an older version by selecting the File -> New -> M-file options from the Desktop menu. In the latest version, a new program file can be created by clicking on the **New Script** icon present within the **HOME** tab on the Desktop toolbar as shown in Fig. 1.1 above. After clicking on the **New Script** icon, the **EDITOR** tab also becomes visible on the top of the Desktop menu as shown below in Fig. 1.8 below. An already existing program file can be opened in MATLAB by selecting the File -> Open options from the Desktop menu (in an older version) or by clicking on the **Open** icon under the EDITOR tab in the FILE pane (in the latest version).

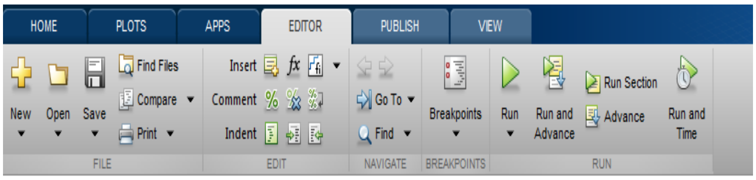


Figure 1.8: The EDITOR Tab (control tools/buttons)

The **Workspace** shows a list of all the variables that are currently defined and also specifies the type and size of each variable (i.e. scalar, vector, etc.). To clear the Workspace, use the command **clear all**. From the Workspace, a selected variable can also be plotted quickly and easily by right clicking on the variable name and choosing a plot type in the PLOTS tab available on the MATLAB Desktop. We can use a Ctrl key plus click in order to select multiple variables for plotting.

ii. One of the extremely useful functions, the **help**: MATLAB has many built-in functions to perform both elementary mathematical operations and complex scientific calculations. The availability of wide-ranging sets of functions in MATLAB makes it advantageous over other programming languages. By typing the help command at the command-line prompt, we can see a list of different categories that MATLAB commands fall into such as general, elementary matrix operations, elementary math functions, graphics, etc.

MATLAB provides two especially useful commands, the **lookfor topic** (in order to find exact names of the available built-in functions linked to a specific topic) and the **help function_name** (to get valid syntaxes for using the built-in functions).

If we type:

```
>> help function_name
```

The above command displays exact information about how to use a predefined function in MATLAB along with its functional description. Thus, typing as **help function_name** in the Command Window describes a valid **syntax** (structure) for using the specific predefined function. The user needs to type like help stem, help sin, help sqrt, etc., where the stem, sin and sqrt are the names of the MATLAB built-in functions. If the user cannot distinguish the function name which suits a desired topic, the same can be obtained by using the other useful command, the lookfor.

iii. The **lookfor** command, the second extremely useful function: To see a list of correct function names that are spelled with a certain word in MATLAB, use the lookfor command and type like lookfor discrete, lookfor sine, etc. In other words, to identify the built-in function name (along with its purpose) related to a picky operation, type as **lookfor topic**, where the **topic** is some name linked with the desired operation. For example, if the user needs to compute an inverse of a matrix, a function named inverse cannot be used because MATLAB does not support any function by this name. Also, typing the **help inverse** command produces nothing. The desired function name we are looking for can be obtained by typing the lookfor inverse command. Because the **lookfor** command displays a list of all the related functions, it takes a minute or so to look at the large number of files and the user has to wait for viewing the results.

There are few other relevant help commands in MATLAB such as the info, what and which. Their complete details can be obtained by typing the command help function_name. MATLAB also contains a variety of demos that can be invoked with the demo command.

iv. The percent symbol (%): It is used to add or denote a comment which ensures that a generated MATLAB code is self-explanatory and user friendly so that it can be easily understood. Anything following the % symbol is always ignored by MATLAB. The MATLAB comments denote a text in a program file that has no effect on the program itself. As good coding is meant to be human readable, comments are preferably given in a generated code at the same time they are solving the purpose and adding relevant information to the code. The comments can be inserted with a purpose to explain the usage of a particular expression, provide the reference information, justify the decisions if needed, describe the limitations and bring up the needed improvements.

v. Inserting spaces to avoid confusion and provide better visuals of the generated statements: While generating the assignment statement at the prompt or in the M-file (plain texts file containing a series of statements), the user can add blank spaces (optional) to make the entered statement easily readable. Thus, both the statements, y = tan(4 * pi) / 5 and y=tan(4*pi)/5, are interpreted in a similar way in MATLAB. Inserting a white space character carefully around the assignment operator improves the readability of the statements by making individual components in them easily noticeable to the user. Addition of spaces before and after an equal sign in the MATLAB statement provides a good visual indication for distinguishing the left and right hand sides of the statement. Thus, all the MATLAB operators such as =, &, |, etc. used in different statements should better be surrounded by spaces for more clarification.

(**Word of Warning**: The user should be extremely careful while using this flexibility and must ensure that MATLAB stores the correct values in order to output the desired result. For example, the statement 2 * cos (pi) * 3 gives a much different result from the result that is obtained in MATLAB by typing the 2*cos(pi)*3 (see the syntax of using the built-in functions)

vi. A set of three dots/full stops or an ellipsis (...), the line continuation operator: By default, MATLAB executes one statement per command-line. The MATLAB statement may contain a rich number of instructions/operations that may extend beyond the single line. If a command of instructions to be entered in MATLAB is excessively long and cannot be entered in one row (the same command-line), the remaining instructions of the single statement can be continued to a next row (the next command-line) by using an ellipsis so that instructions on the multiple lines are executed as the single MATLAB statement.

For example:

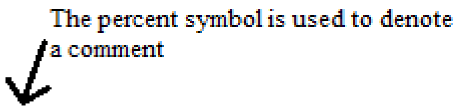
```
 x = (1 + 2 + 3 + 4 + 5 ...

+ 6 + 7)
```

The above MATLAB statement is sustained on the next line by putting three dots at the end of the first line (the line to be continued) and results in the scalar output equal to 28.

The user should always press an Enter key for starting the new command-line (on the fresh prompt) as well as instructing MATLAB to run the current command. Also, an ellipsis cannot be placed within single right quotes in order to continue a long string to the next line.

Also, we can put multiple statements on one command-line by using a comma between every statement on the same line as shown below:



The percent symbol is used to denote a comment

```
x=1+4+6, x, y=x+1  % The three statements on the single command-line are executed from left to right
```

MATLAB interprets them as three different statements and executes in an order from left to right.

vii. The semicolon operator (;): If after the statement, i.e. at the end, a semicolon is added then MATLAB suppresses the display of its output and no result appears in the Command Window for that statement. The use of a semicolon at the end of the statement does allow the computation of an expression in the statement but suppresses printing the calculated result in the Command Window. However, if it is useful to have a look at the output, the operator should not be used after an expression (or at the end of the statement) so that the result can be printed and becomes visible on the screen.

(**Note**: In MATLAB, a semicolon within square brackets is also used to start a fresh row of a matrix)

viii. Use of keyboard arrows in MATLAB: To recall instructions from the Command History panel/window without retyping them, an up arrow key ↑ should be used in the Command Window. Whereas, down arrow key on a computer keyboard should be used to go forward in the statements entered already.

ix. MATLAB as a **case-sensitive** language**:** It is a case-sensitive as it interprets each uppercase and lowercase letter differently while using it for the MATLAB variables and built-in functions. Thus, the variable names like PROFIT, Profit and profit are three different entities in MATLAB. Similarly, MATLAB understands the function name cos( ) but does not recognize the Cos( ) as all the built-in functions are named with an exact word in a lower case.

(**Note**: MATLAB is not **completely** case-sensitive)

x. MATLAB indices: All MATLAB arrays have their **Index** or **Subscript** begin **at 1** (an integer 1 at the subscript position represents the 1st element of an array). The particular elements of an array can be referred by means of **subscripts/indices**. For example, if a vector named B contains elements in row fashion, the user can extract or access the first element from the B by typing as B(1), where the scalar value 1 denotes a subscript.

In order to have a zero-index (means a zero at the subscript position represents the 1st element of an array) or minus-numbered index in MATLAB, the user needs to define another variable just for storing the indices/locations of the elements of an array and write a whole code carefully keeping in mind the defined index variable and implement the changes accordingly.

xi. The variables names versus MATLAB keywords/built-in functions: The variable names must not resemble the MATLAB built-in function names or reserved keywords (e.g. for, while, if, etc.). A list of the reserved keywords can be generated by typing the iskeyword in MATLAB. The user should go for variable names which are meaningful and must start them with a letter only. The letters (31), digits (0-9) and underscore (_) are allowed while naming variables in MATLAB.

xii. A colon operator (:) in MATLAB: To create sequences of numbers (discrete signals generation) as well as to select a specified range of values from a discrete signal (a segment selection), a colon is used. Whenever there is a lot of data in an array and it is desired to extract a portion of the data for some analysis, we use the colon operator to address certain elements of the array.

xiii. The **clc** command: It clears the screen by erasing all the text from the Command Window. After running the **clc**, the previously displayed texts get cleared and cannot be scrolled with a scroll bar in the Command Window. It is always better to use the clc command in program files as the first line of a code so that the output of the current execution can be displayed in the same starting position (erasing the previous outputs) on the MATLAB Desktop screen in the Command Window.

xiv. The **clear** command: It can remove the variables (selected or all) from the current Workspace, and thereby erases them from the system memory. The command **clear all** removes all the current variables from the MATLAB's memory, whereas the command **clear x** removes the specific variable x from the memory.

xv. MATLAB always attempts to display integers (whole numbers) exactly if they are not large enough. Numbers lying in the range, $0.001 < x \leq 1000$, can be represented in MATLAB in a usual decimal form (a fixed-point). Otherwise, a number is represented in a scientific (floating point) notation. If an integer is too large, it is displayed in a scientific notation with five significant digits, e.g. the number 1 234 567 890 is displayed in MATLAB as 1.2346e+09 which should be interpreted as the 1.2346×10^9. Numbers with decimal parts are displayed with four significant decimal digits, e.g. the decimal number 1000.1 is represented as 1.0001e+003 in MATLAB.

xvi. MATLAB has 14 fundamental data types (or classes): The default data type of a number is double-precision and all MATLAB computations are done in a double-precision format. In MATLAB, double-precision floating point numbers represent real numbers, but not to an arbitrary precision. For example, the value of a trigonometric function $\sin(\pi) \neq 0$ in MATLAB because the software package only stores an approximate value of the math function π.

xvii. The default variable name **ans**: MATLAB utilizes the name **ans** (by default) to return the answer of an expression that is not assigned to any variable name and written without using an equal sign on its left hand side.

xviii. Help and Search tools in MATLAB: For the beginners, various informative manuals are available which can be browsed by clicking on the question-mark pushbutton ? that is present on the toolstrip near the top right of the MATLAB screen as shown below in Fig. 1.9. In the latest version, the Help browser tool is a drop-down menu having the sub-menus as shown in Fig. 1.10 below.

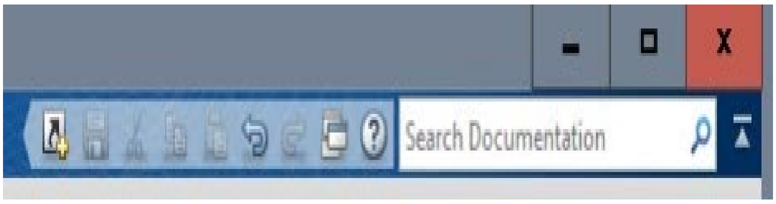


Figure 1.9: The Help and Search Tools

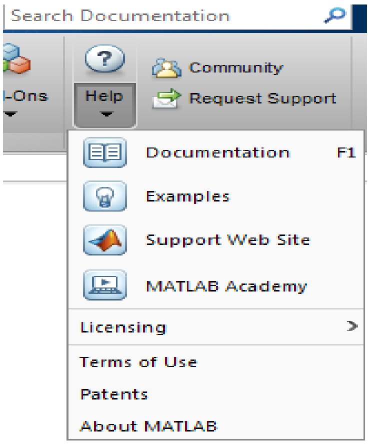


Figure 1.10: The Help Tool Contents

The online manuals on MATLAB can also be accessed through Google search. The user can also enter the function names in the Search Documentation text box (see Fig. 1.9) to have the information about how to use them in MATLAB.

xix. Error messages: If any command or operation entered by the user in a program file is invalid and not as per the rigorous set of rules followed by MATLAB, the software displays a warning or error message in the Command Window and points to the exact line in which there is some mistake. A mistake in the form of an error message appears in red in the Command Window along with a beep which must be read and corrected carefully. It is difficult to write a code without errors (bugs), but MATLAB has a powerful debugging functionality to find bugs (errors) in a generated code that are responsible for not getting the expected results.

xx. The symbol **%{ ……. %}**, for the multi-line comment: If a set of comments to be entered by the user extend beyond one line, the user should enclose the set within the symbol %{**…….** %}. To take a continuous group of lines as a comment, at first (before writing the first text **line)** type the symbol %{ alone on the separate line and then in the end (after the last text **line**) type the symbol %} alone on the separate line. For example:

```
%{
  A Collection of lines to be used as comments
%}
```

xxi. Storing the generated codes: The Workspace gets cleared after quitting MATLAB. In the latest version, the **Save** icon from the Editor tab on the Desktop toolbar is used for storing all the current variables in the Workspace. All program files are saved with the .m extension in MATLAB to be interpreted and re-used during a current session or during another session. The M-file should not be stored by a name that already exists as the name of the MATLAB built-in function.

xxii. The assignment operator (=), an equal sign: It **assigns** a data (on the right side of an equal sign) to a variable (on the left side of an equal sign) or **saves** the value of a computed expression (on the right side of an equal sign) in a variable. Thus, the assignment operator transfers an answer of the MATLAB assignment statement (after evaluating an expression on the right side) into a variable (mentioned on the left side of an equal sign).

xxiii. The double equality symbol (==): When double equality signs are used as an operator in MATLAB, it tests for the equivalent numeric values or finds the occurrence of a specific character and returns a logical value.

(**Note**: The double equal symbol is used to represent an equation while solving it using the MATLAB symbolic math)

xxiv. No declaration section: In MATLAB, variables are created by the user when their names are typed on the left hand side of an equal sign and there is no need to declare the variables before using them. By default, numeric variables are defined as type double. When any real or complex number is entered as the value of a variable, MATLAB **automatically** sets the variable to be real or complex, respectively and eliminates the need of having a declaration section before the naming section. There are three ways to input a data in MATLAB:

- ❖ Assign a data to a variable through the assignment statement (see Section 2.1).
- ❖ Input a data from a computer keyboard by using the **input** built-in function. The input function displays a text string (a focused message typed as the argument to the input function) in the Command Window at the appointed time and then waits for the user to type in a desired response to the printed message. The argument (a text string within round brackets) given to the input function must be enclosed in apostrophes (single quotes). For details, refer to Section 2.3.1(xv).
- ❖ Read a set of data from program files (the M-files) that is already stored in the computer's memory.

xxv. Working with numeric and symbolic variables on the right side of an equal sign: The assignment statement, b = 2*x, if executed directly in the Command Window without assigning any value to the variable x (on the right side of an equal sign), results in an error message. This is because MATLAB tries to evaluate an expression given on the right hand side and assigns the computed value to the variable b, but here the evaluation fails. Therefore, a value must be formerly allocated to each variable before using the variables on the right side of an equal sign in the MATLAB assignment statement. When the user first assigns a value to the variable x and then executes the assignment statement by typing as b = 2*x, MATLAB stores the result in the variable b and remembers the stored value only. It immediately forgets the relation between b and x so that if a value of the variable x is changed later, the value of the variable b does not change. Whereas, if the user gives another assignment statement by typing as b = x^2, MATLAB discards previously calculated value of the variable b, and thereby substitutes a fresh value equal to the square of the variable x in the variable b. Thus, it can be stated that MATLAB always stores the computed value of the latest expression assigned to a given variable and forgets the previous expression allocated to the same, thereby overrides the previously computed value. This can be avoided by writing a code in the M-file.

Symbolic Variables: MATLAB do accept symbolic inputs but only when we first create the symbolic numbers, variables or expressions by using either the command **syms** or **sym**. The equation x+5=0 can only be solved using the MATLAB Symbolic Math toolbox.

xxvi. An Enter key as the execute button: In MATLAB, an Enter key is pressed to execute the typed instructions. Hitting it is must to activate an instruction.

xxvii. When MATLAB starts doing something strange or just stops performing anything, it is better to restart it.

CHAPTER 2

MATLAB BASICS

2.1 Basics of MATLAB

The chapter introduces MATLAB basics and facilitates learning of necessary skills that are required in order to work with the software efficiently and meaningfully. MATLAB allows the user to manipulate (visualize, compute, plot, import or export, analyze, modify, etc.) required data/information. It employs arrays as the basic representation of a given data or information and expresses the scientific problems and solutions in mathematical notations. The prerequisite to using MATLAB is that the user should be familiar in advance with the applicable statement structures, operator symbols, built-in function names (along with valid syntaxes for using them) and special keywords that are executable in the software.

To start MATLAB, click on the windows **Start** symbol at the bottom left of the computer screen, then click on the **All Programs** in its menu/toolbar and after that click on the **MATLAB** option (available only when the software is already installed on the computer). This opens the MATLAB GUI which consists of the Command Window where the software can perform all mathematical computations like a calculator by accepting an input from the user. When MATLAB is ready to accept the single line statement (having an instruction or multiple instructions in an expression on the right side), it shows the command prompt >> in the Command Window. After typing in the statement per line, the user can hit an **Enter** key in order to evaluate it and print the result (when typed without a semicolon). It does not matter to MATLAB whether a computer cursor is at the end of the line statement or not. MATLAB is easy to work with as it does not involve declaration of variables before naming them and using in a program file. When a variable is used for the first time, the software allocates space for it. To assign any value to the MATLAB variable, we use an equal sign. For example, in order to set the numeric variable x equal to 2, we type in MATLAB as follows:

```
>> x = 2        % The single assignment statement on the single line (same row)
```

results in:

```
x =

  2
```

In the Command Window, instructions given to MATLAB can be edited only before they are run with an Enter keystroke.

(**Note**: It is necessary to hit an **Enter** key to instruct MATLAB to execute the single line statement)

2.1.1 Working with Arrays

One of the basic data structure most frequently used in MATLAB is an array. An array variable can be a scalar (number), vector (equivalent to one-dimensional array) or matrix (two-dimensional array). The user should learn how to think in terms of arrays (either vectors or matrices) in order to work with MATLAB quickly and effectively. Arrays can be of more than two dimensions and can contain a data other than numbers.

(i) Writing the MATLAB Assignment Statement:

The syntax for writing the assignment statement (i.e. assigning values or expressions to variables) in MATLAB is as follows:

```
   variable_name=an_expression   % The syntax for writing the assignment statement
```

In MATLAB, variables can be defined by simply assigning expressions to their names and there is no need to declare the type of a variable. When an expression (containing a scalar or an array) of a specific data type such as numeric constant (integer/fractional number, real constant and complex number) or character constant (single and string constant) is assigned to a named variable, MATLAB automatically sets the type of the named variable to be the same as that of an array assigned to it. Any assigned variable gets saved and may be displayed by typing the name of the variable afterwards. An expression containing element entries along with the suitable operators, built-in or user-defined functions can also be assigned to a named variable. When the assignment statement is executed, an expression to the right of an equal sign is first evaluated or computed and the output value obtained is assigned to a variable mentioned on the left of an equal sign.

(**Note**: We can add spaces (optional) while writing the MATLAB assignment statements so as to make them easily readable and understandable)

Things to Remember:

While creating variables and using them in MATLAB, the following tips should be followed religiously:

It is not required in MATLAB to declare in advance any numeric variable or constant at the start of a program file. A variable name is a user-created entity that permits saving an information data assigned to the variable and recalling the saved data in order to access it throughout the current MATLAB session. MATLAB allows the programmer/user to access an output variable in a program file by changing the information (data values) carried by the input variables while keeping operations on the variables the same as before. A variable is generally named either by using individual characters (like a, A, x, etc.) or strings of characters (like var1, new_value, etc.) in MATLAB. The MATLAB variables are named according to a certain set of guidelines, the most important of which is that they must begin with a letter. Blank spaces cannot be included in the MATLAB variable names. The Workspace consists of complete information about all the currently active variables which can be made visible in the Command Window by typing the whos command.

Last but not least, the MATLAB assignment statement which is **terminated** with a **semicolon** (;) gets executed but the result of an execution does not appear in the Command Window. This means that an expression that is assigned to a variable in the given statement is definitely evaluated even though the display of output is suppressed by a semicolon.

(ii) Creating Scalars i.e. 1x1 Arrays

The simplest form of an array is a scalar (special case of a matrix) which is a number with dimensions equal to 1x1 i.e. one row-by-one column. A scalar is a single value or element (e.g. 2, 9, etc.) which can be entered in MATLAB as shown below.

Type as follows:

```
  x=9     % Represents the scalar x (1x1 array)
```

MATLAB replies with:

```
x =

    9
```

A scalar (constant) does not need any square brackets. If the above statement is ended with a semicolon, it does not repeat the variable x with the assigned value of 2 in an indented form on the display screen. The variable x can be easily accessed by typing as x at the command prompt followed by hitting enter in order to recall the value assigned to it. To see that the variable x is a 1x1 array, type and execute the **whos** command.

Working with Vectors:

A **vector** is a one-dimensional array that consists of numbers or elements either in row shape (called as row vectors) or column shape (called as column vectors). A list which contains numbers/elements that are enclosed within square brackets and separated by commas/spaces or semicolons is considered as a vector in MATLAB. Therefore, row and column vectors are defined in MATLAB as shown below.

(iii) Defining Row Vectors:

A row vector is a (1xn) matrix which contains n number of columns and a single (or one) row only. For defining a row vector in MATLAB, all the elements of a row are placed inside square brackets [ ] and the n different columns (containing single values) within a row must be separated by using either spaces or commas.

Row vectors can be entered in the following ways:

- A variable name is typed first, followed by an equal sign, and then the desired element values of a row vector are assigned to the variable.

For example:

```
w=[1 2 1]  % Blank spaces are used to split the elements (placed between square brackets) within a row
```

results in the row vector w containing three elements 1, 2 and 1 as shown below:

```
w =

  1 2 1
```

- Alternatively, we can define the row vector w by typing as follows:

```
w=[1,2,1];   % Within square brackets, the values of a row are separated by commas in the vector w
```

In order to make the above command more understandable, blank spaces can also be added along with commas, so we can also type:

```
w=[1, 2, 1];    % Remember, adding spaces along with commas is optional
```

A **semicolon** at the end of any statement is used to suppress the **display** of the output of that statement i.e. it stops the output from appearing in the Command Window. It is not required to print the output of each statement on the screen as displaying results of a large number of longer statements in a lengthy code may confuse the user and take more time to print the desired result that is actually needed to be viewed.

(iv) Creating Column Vectors:

A column vector is a mx1 matrix that has m number of rows but a single (or one) column only. For defining a column vector in MATLAB, all the elements of a column are placed inside square brackets [ ] and m different rows (containing single values) must be separated by using the semicolon operator (;). The operator is also called the row separator and is used to generate entries within a column.

Column vectors can be entered in MATLAB using the following ways:

- For defining a column vector, a semicolon is used after each element inside square brackets to end a row (or start a fresh new row). For example, typing as follows:

```
y=[11; 12; 13] % Semicolons are put to end rows (each having a single value) of the column vector y
```

results in the column vector y having three rows and a single column as shown below:

```
y =
  11
 12
 13
```

Thus, the variable y is a column vector which has only one element in each row and the elements of a same row are separated from the elements of another by a semicolon.

- A fresh new row can also be entered by typing its elements on the newline (obtained by pressing an Enter key) in order to separate rows to generate a column vector. For example:

```
>> x=[1
2
3]
x =
  1
  2
  3
```

Since column vectors are transpose of row vectors, column vectors can also be created by using the transpose operator (') on row vectors as shown below.

(v) Transpose of Vectors:

A special operator named prime or apostrophe (') evaluates transpose of an array having real elements. The transpose of a scalar returns a scalar itself. On the other hand, transposing a real vector (or real matrix) swaps rows and columns while retaining their order as shown in an example below.

The row vector w is defined by typing as:

```
w=[1 2 1]
```

Now, if we type as follows:

```
W = w'    % Swaps rows and columns
```

results in the column vector W:

```
W =
  1
  2
  1
```

Thus, a row vector can be transformed into a column vector by applying the transpose operator.

However, the transpose (an apostrophe) operator computes conjugate transpose of complex vectors and therefore also negates their imaginary parts. But, if we place a dot before the operator, i.e. the operator (.') transposes a complex vector (or complex matrix) without conjugation which implies that it does not change signs of the imaginary parts of complex elements in the vector.

(vi) Generating Sequences of Numbers (ranges of values) as Vectors:

MATLAB utilizes the colon operator (:) to create vectors of equally spaced elements using a beginning element value, ending (maximum) element value and step size (an increment) in between the element values. To create larger vectors or sequences, say up to 50, we need to type a lot of numbers in the row vector. The use of colon operator allows us an easier way to create the vectors having huge number of elements.

For example, a sequence of numbers as a vector can be generated (initialized) in MATLAB with the colon operator by using the syntax begin:incremental_step:end as shown below.

If we type:

```
X = 1:0.5:5
```

it results in:

```
X =
   Columns 1 through 7
   1.0000  1.5000  2.0000  2.5000  3.0000  3.5000  4.0000
   Columns 8 through 9
   4.5000  5.0000
```

(The elements values are 1, 1.5, ….., 5 in increments of 0.5. Thus, in the assignment statement when colons separate the three argument values, the middle value is regarded as an incremental value)

Whereas, typing:

```
X=1:3
```

returns:

```
X =

   1  2  3
```

This means that the row vector X which starts at 1 and ends at 3 in incremental steps of 1 (by default) is created. Thus, it can be stated that if the user does not specify an increment size and omits the middle argument step, a default incremental value of 1 is used. Therefore, an array a:c is same as an array a:1:c in MATLAB.

To conclude, the command [b:s:e] specifically creates a vector with the element values b, b+s, b+2*s, ... up to e. If a step size is not specified, MATLAB assumes it to be 1 by default. Therefore, the command [0:2:10] creates a vector of even integers with the starting value 0, an incremental value of 2 and the last value 10, i.e. [0 2 4 6 8 10]. On the other hand, typing as x = 8:-1:1 generates elements in the variable x starting from 8, 7, …, up to 1 (since an increment size is a negative integer value).

A colon is also used in MATLAB indexing to address multiple elements of an array by using the form start:end so that a specified range of values can be selected or modified in the array.

(vii) Defining Matrices:

A matrix is a two-dimensional (2D) rectangular data set or collection of numbers arranged into fixed number of rows and columns. Matrices are entered in MATLAB by combining both the techniques used to define row and column vectors.

• To define a matrix in MATLAB, the user can enter multiple rows as well as columns by treating it as columns of row vectors. That is, the elements in each row of a matrix are entered as a row vector by separating the entries within the same row using either commas or spaces or both, whereas a row (containing multiple values) is separated from the other by using semicolons to generate entries within columns as shown below.

For example, typing:

```
M = [1 2 3; 3 3 1]     % Generates the 2x2 matrix with the variable name M
```

results in:

```
M =
  1 2 3
 3 3 1
```

In the above command, all column values of the same row are separated by spaces and the two rows are distinguished by putting a semicolon.

```
>> p=[1 2;3 4;5 6]     % Generates the 3x2 matrix p
p =
  1   2
  3   4
  5   6
```

Thus, to generate a matrix, place all numeric values or elements between square brackets, use either spaces or commas or both to separate the elements within same rows and use semicolons to separate the matrix rows.

• A fresh matrix row can also be started by inserting a new line within square brackets and using spaces or commas to separate the elements within a same row as shown below:

```
M=[1 2 3        % Hit enter for the line break in order to separate the rows
3 3 1]
```

gives:

```
M =
  1 2 3
 3 3 1
```

One more example:

```
>> h=[2 4 6
3 2 1]
h =
  2   4   6
  3   2   1
```

MATLAB also provides special functions to create convenient matrices and vectors without having the need to type or read in each of the elements as shown below.