About the Tutorial

MATLAB is a programming language developed by MathWorks. It started out as a matrix programming language where linear algebra programming was simple. It can be run both under interactive sessions and as a batch job.

This tutorial gives you aggressively a gentle introduction of MATLAB programming language. It is designed to give students fluency in MATLAB programming language. Problem-based MATLAB examples have been given in simple and easy way to make your learning fast and effective.

Audience

This tutorial has been prepared for the beginners to help them understand basic to advanced functionality of MATLAB. After completing this tutorial you will find yourself at a moderate level of expertise in using MATLAB from where you can take yourself to next levels.

Prerequisites

We assume you have a little knowledge of any computer programming and understand concepts like variables, constants, expressions, statements, etc. If you have done programming in any other high-level language like C, C++ or Java, then it will be very much beneficial and learning MATLAB will be like a fun for you.

Copyright & Disclaimer Notice

Table of Contents

1. OVERVIEW

MATLAB (matrix laboratory) is a fourth-generation high-level programming language and interactive environment for numerical computation, visualization and programming.

MATLAB is developed by MathWorks.

It allows matrix manipulations; plotting of functions and data; implementation of algorithms; creation of user interfaces; interfacing with programs written in other languages, including C, C++, Java, and FORTRAN; analyze data; develop algorithms; and create models and applications.

It has numerous built-in commands and math functions that help you in mathematical calculations, generating plots, and performing numerical methods.

MATLAB's Power of Computational Mathematics

MATLAB is used in every facet of computational mathematics. Following are some commonly used mathematical calculations where it is used most commonly:

- Dealing with Matrices and Arrays
- 2-D and 3-D Plotting and graphics
- Linear Algebra
- Algebraic Equations
- Non-linear Functions
- Statistics
- Data Analysis
- Calculus and Differential Equations
- Numerical Calculations
- Integration
- Transforms
- Curve Fitting
- Various other special functions

Features of MATLAB

Following are the basic features of MATLAB:

- It is a high-level language for numerical computation, visualization and application development.
- It also provides an interactive environment for iterative exploration, design and problem solving.
- It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
- It provides built-in graphics for visualizing data and tools for creating custom plots.
- MATLAB's programming interface gives development tools for improving code quality, maintainability, and maximizing performance.
- It provides tools for building applications with custom graphical interfaces.
- It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

Uses of MATLAB

MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including:

- signal processing and Communications
- image and video Processing
- control systems
- test and measurement
- computational finance
- computational biology

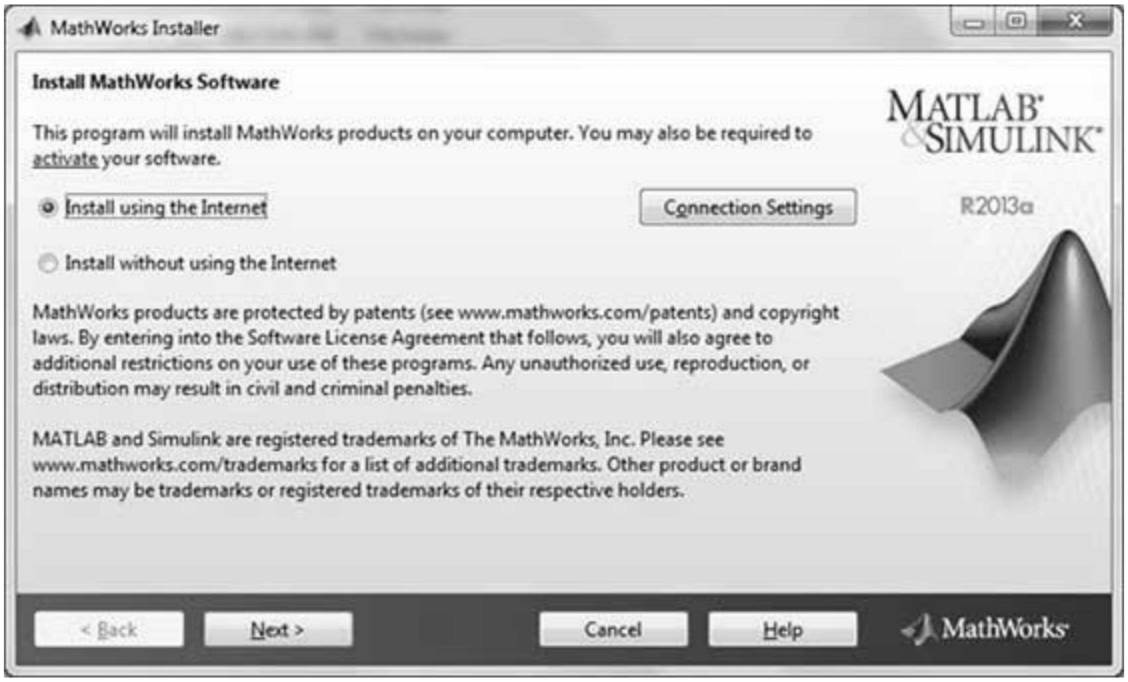
2. ENVIRONMENT

Local Environment Setup

Setting up MATLAB environment is a matter of few clicks. The installer can be downloaded from http://in.mathworks.com/downloads/web_downloads:

MathWorks provides the licensed product, a trial version and a student version as well. You need to log into the site and wait a little for their approval.

After downloading the installer the software can be installed through few clicks.



Install MathWorks Software

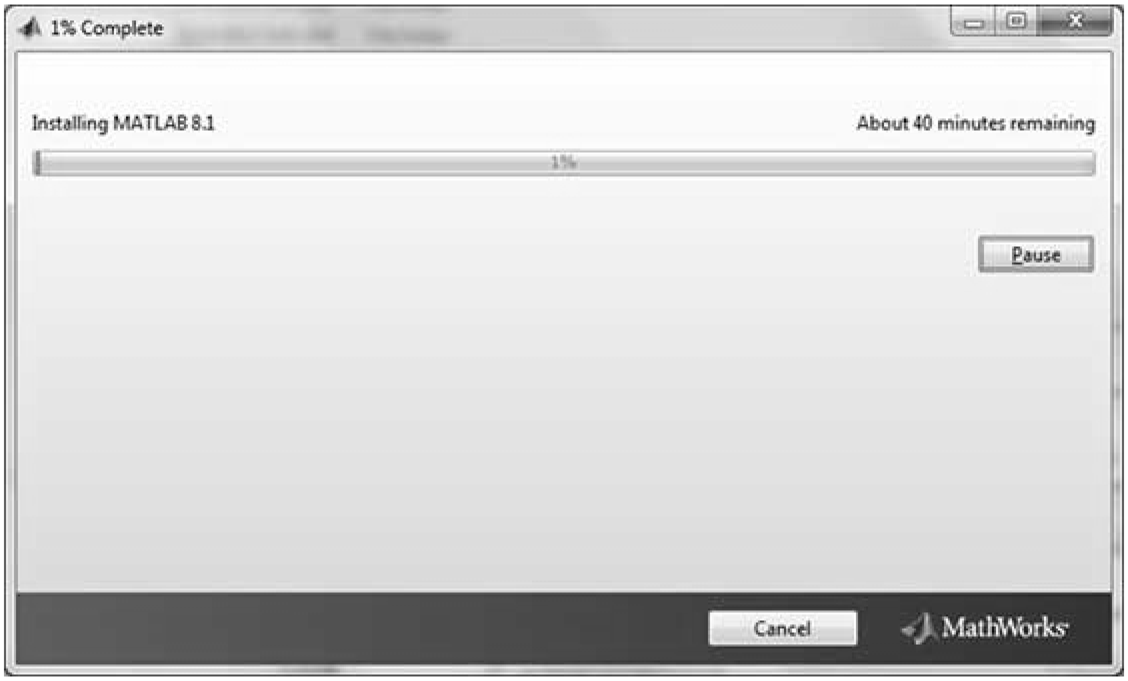
This program will install MathWorks products on your computer. You may also be required to activate your software.

(●) Install using the Internet

( ) Install without using the Internet

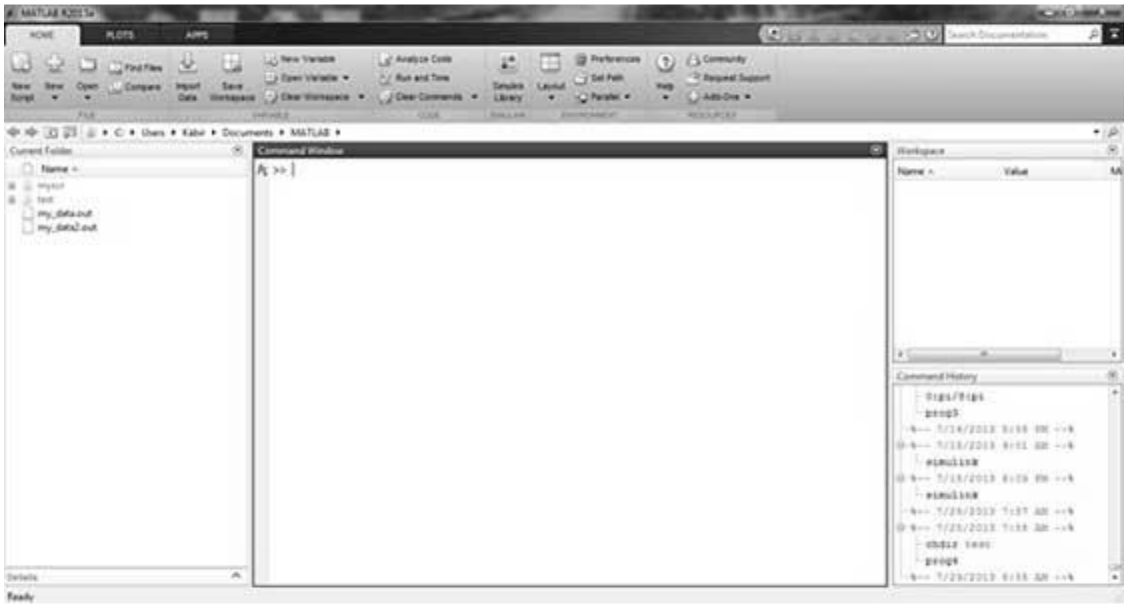
MathWorks products are protected by patents (see www.mathworks.com/patents) and copyright laws. By entering into the Software License Agreement that follows, you will also agree to additional restrictions on your use of these programs. Any unauthorized use, reproduction, or distribution may result in civil and criminal penalties.

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. Please see www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.



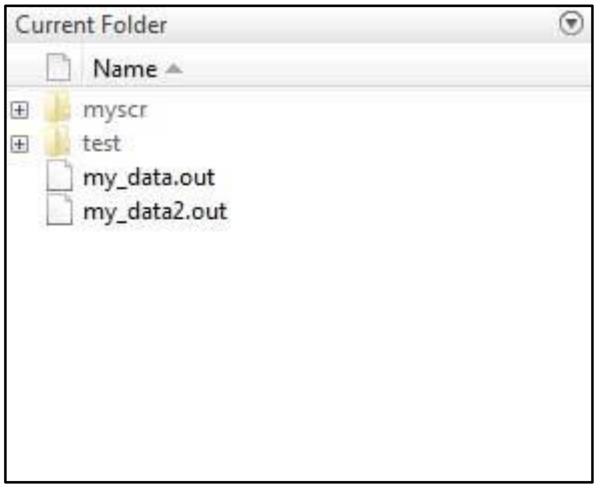
Understanding the MATLAB Environment

MATLAB development IDE can be launched from the icon created on the desktop. The main working window in MATLAB is called the desktop. When MATLAB is started, the desktop appears in its default layout:

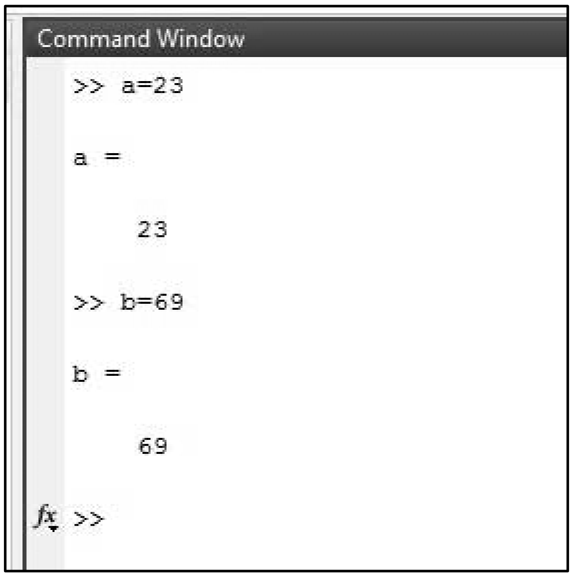


The desktop has the following panels:

Current Folder - This panel allows you to access the project folders and files.

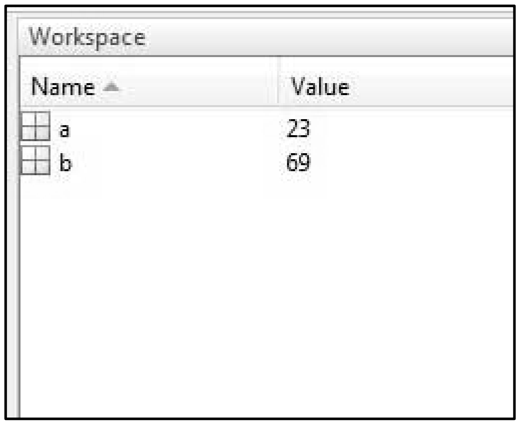


Command Window - This is the main area where commands can be entered at the command line. It is indicated by the command prompt (>>).



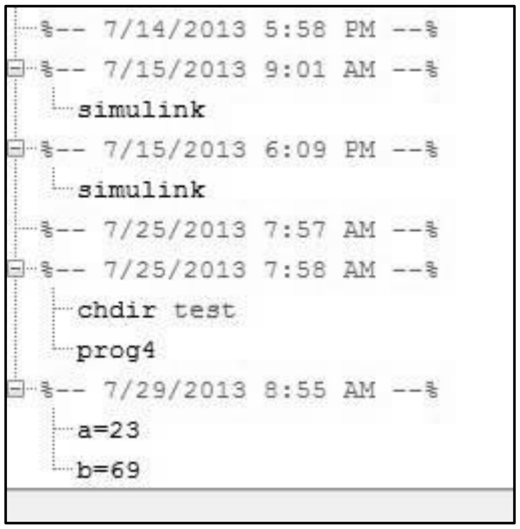
```
>> a=23

a =

    23

>> b=69

b =

    69

>>
```

Workspace - The workspace shows all the variables created and/or imported from files.



Name	Value
a	23
b	69

Command History - This panel shows or rerun commands that are entered at the command line.



```
%-- 7/14/2013 5:58 PM --%
%-- 7/15/2013 9:01 AM --%
  simulink
%-- 7/15/2013 6:09 PM --%
  simulink
%-- 7/25/2013 7:57 AM --%
%-- 7/25/2013 7:58 AM --%
  chdir test
  prog4
%-- 7/29/2013 8:55 AM --%
  a=23
  b=69
```

Set up GNU Octave

If you are willing to use Octave on your machine (Linux, BSD, OS X or Windows), then kindly download latest version from http://www.gnu.org/software/octave/download.html. You can check the given installation instructions for your machine

3. BASIC SYNTAX

MATLAB environment behaves like a super-complex calculator. You can enter commands at the >> command prompt.

MATLAB is an interpreted environment. In other words, you give a command and MATLAB executes it right away.

Hands on Practice

Type a valid expression, for example,

```
5 + 5
```

And press ENTER

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is:

```
ans = 10
```

Let us take up few more examples:

```
3 ^ 2        % 3 raised to the power of 2
```

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is:

```
ans = 9
```

Another example,

```
sin(pi /2)    % sine of angle 90o
```

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is:

```
ans = 1
```

Another example,

```
7/0             % Divide by zero
```

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is:

```
ans = Inf

warning: division by zero
```

Another example,

```
732 * 20.3
```

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is:

```
ans =  1.4860e+04
```

MATLAB provides some special expressions for some mathematical symbols, like pi for π, Inf for ∞, i (and j) for √-1 etc. **Nan** stands for 'not a number'.

Use of Semicolon (;) in MATLAB

Semicolon (;) indicates end of statement. However, if you want to suppress and hide the MATLAB output for an expression, add a semicolon after the expression.

For example,

```
x = 3;

y = x + 5
```

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is:

```
y =  8
```

Adding Comments

The percent symbol (%) is used for indicating a comment line. For example,

```
x = 9     % assign the value 9 to x
```

You can also write a block of comments using the block comment operators % { and % }.

The MATLAB editor includes tools and context menu items to help you add, remove, or change the format of comments.

Commonly used Operators and Special Characters

MATLAB supports the following commonly used operators and special characters:

Operator	**Purpose**
+	Plus; addition operator.
-	Minus; subtraction operator.
*****	Scalar and matrix multiplication operator.
.*	Array multiplication operator.
^	Scalar and matrix exponentiation operator.
.^	Array exponentiation operator.
\	Left-division operator.
/	Right-division operator.

.	Array left-division operator.
./	Array right-division operator.
:	Colon; generates regularly spaced elements and represents an entire row or column.
( )	Parentheses; encloses function arguments and array indices; overrides precedence.
[ ]	Brackets; enclosures array elements.
.	Decimal point.
…	Ellipsis; line-continuation operator
,	Comma; separates statements and elements in a row
;	Semicolon; separates columns and suppresses display.
%	Percent sign; designates a comment and specifies formatting.
_	Quote sign and transpose operator.
._	Non-conjugated transpose operator.
=	Assignment operator.

Special Variables and Constants

MATLAB supports the following special variables and constants:

Name	Meaning
ans	Most recent answer.
eps	Accuracy of floating-point precision.
i,j	The imaginary unit √-1.
Inf	Infinity.
NaN	Undefined numerical result (not a number).
pi	The number π

Naming Variables

Variable names consist of a letter followed by any number of letters, digits or underscore.

MATLAB is **case-sensitive**.

Variable names can be of any length, however, MATLAB uses only first N characters, where N is given by the function **namelengthmax**.

Saving Your Work

The **save** command is used for saving all the variables in the workspace, as a file with .mat extension, in the current directory.

For example,

```
save myfile
```

You can reload the file anytime later using the **load** command.

```
load myfile
```

4. VARIABLES

In MATLAB environment, every variable is an array or matrix.

You can assign variables in a simple way. For example,

```
x = 3       % defining x and initializing it with a value
```

MATLAB will execute the above statement and return the following result:

```
x =

     3
```

It creates a 1-by-1 matrix named *x* and stores the value 3 in its element. Let us check another example,

```
x = sqrt(16)       % defining x and initializing it with an expression
```

MATLAB will execute the above statement and return the following result:

```
x =

     4
```

Please note that:

Once a variable is entered into the system, you can refer to it later.

Variables must have values before they are used.

When an expression returns a result that is not assigned to any variable, the system assigns it to a variable named ans, which can be used later.

For example,

```
sqrt(78)
```

MATLAB will execute the above statement and return the following result:

```
ans =

    8.8318
```

You can use this variable ***ans***:

```
9876/ans
```

MATLAB will execute the above statement and return the following result:

```
ans =

   1.1182e+03
```

Let's look at another example:

```
x = 7 * 8;

y = x * 7.89
```

MATLAB will execute the above statement and return the following result:

```
y =

  441.8400
```

Multiple Assignments

You can have multiple assignments on the same line. For example,

```
a = 2; b = 7; c = a * b;
```

MATLAB will execute the above statement and return the following result:

```
c =

    14
```

I have forgotten the Variables!

The **who** command displays all the variable names you have used.

```
who
```

MATLAB will execute the above statement and return the following result:

```
Your variables are:

a    ans  b    c    x    y
```

The **whos** command displays little more about the variables:

Variables currently in memory

Type of each variables

Memory allocated to each variable

Whether they are complex variables or not

```
whos
```

MATLAB will execute the above statement and return the following result:

```
  Name      Size            Bytes  Class     Attributes



  a         1x1                 8  double

  ans       1x1                 8  double

  b         1x1                 8  double

  c         1x1                 8  double

  x         1x1                 8  double

  y         1x1                 8  double
```

The **clear** command deletes all (or the specified) variable(s) from the memory.

```
clear x     % it will delete x, won't display anything

clear      % it will delete all variables in the workspace
```

```
             %  peacefully and unobtrusively
```

Long Assignments

Long assignments can be extended to another line by using an ellipses (...). For example,

```
initial_velocity = 0;

acceleration = 9.8;

time = 20;

final_velocity = initial_velocity ...

    + acceleration * time
```

MATLAB will execute the above statement and return the following result:

```
final_velocity =

   196
```

The format Command

By default, MATLAB displays numbers with four decimal place values. This is known as ***short format***.

However, if you want more precision, you need to use the **format** command.

The **format long** command displays 16 digits after decimal.

For example:

```
format long

x = 7 + 10/3 + 5 ^ 1.2
```

MATLAB will execute the above statement and return the following result:

```
x =

  17.231981640639408
```

Another example,

```
format short

x = 7 + 10/3 + 5 ^ 1.2
```

MATLAB will execute the above statement and return the following result:

```
x =

   17.2320
```

The **format bank** command rounds numbers to two decimal places. For example,

```
format bank

daily_wage = 177.45;

weekly_wage = daily_wage * 6
```

MATLAB will execute the above statement and return the following result:

```
weekly_wage =

       1064.70
```

MATLAB displays large numbers using exponential notation.

The **format short e** command allows displaying in exponential form with four decimal places plus the exponent.

For example,

```
format short e

4.678 * 4.9
```

MATLAB will execute the above statement and return the following result:

```
ans =

   2.2922e+01
```

The **format long e** command allows displaying in exponential form with four decimal places plus the exponent. For example,

```
format long e

x = pi
```

MATLAB will execute the above statement and return the following result:

```
x =

    3.141592653589793e+00
```

The **format rat** command gives the closest rational expression resulting from a calculation. For example,

```
format rat

4.678 * 4.9
```

MATLAB will execute the above statement and return the following result:

```
ans =

   2063/90
```

Creating Vectors

A vector is a one-dimensional array of numbers. MATLAB allows creating two types of vectors:

Row vectors

Column vectors

Row vectors are created by enclosing the set of elements in square brackets, using space or comma to delimit the elements.

For example,

```
r = [7 8 9 10 11]
```

MATLAB will execute the above statement and return the following result:

```
r =

  Columns 1 through 4

       7              8              9             10

  Column 5

      11
```

Another example,

```
r = [7 8 9 10 11];

t = [2, 3, 4, 5, 6];

res = r + t
```

MATLAB will execute the above statement and return the following result:

```
res =

  Columns 1 through 4

       9             11             13             15

  Column 5

      17
```

Column vectors are created by enclosing the set of elements in square brackets, using semicolon (;) to delimit the elements.

```
c = [7;  8;  9;  10; 11]
```

MATLAB will execute the above statement and return the following result:

```
c =

       7
```

```
      8

      9

     10

     11
```

Creating Matrices

A matrix is a two-dimensional array of numbers.

In MATLAB, a matrix is created by entering each row as a sequence of space or comma separated elements, and end of a row is demarcated by a semicolon. For example, let us create a 3-by-3 matrix as:

```
m = [1 2 3; 4 5 6; 7 8 9]
```

MATLAB will execute the above statement and return the following result:

```
m =

      1          2          3

      4          5          6

      7          8          9
```

5. COMMANDS

MATLAB is an interactive program for numerical computation and data visualization. You can enter a command by typing it at the MATLAB prompt '>>' on the **Command Window**.

In this section, we will provide lists of commonly used general MATLAB commands.

Commands for Managing a Session

MATLAB provides various commands for managing a session. The following table provides all such commands:

Command	**Purpose**
clc	Clears command window.
clear	Removes variables from memory.
exist	Checks for existence of file or variable.
global	Declares variables to be global.
help	Searches for a help topic.
lookfor	Searches help entries for a keyword.
quit	Stops MATLAB.
who	Lists current variables.
whos	Lists current variables (long display).

Commands for Working with the System

MATLAB provides various useful commands for working with the system, like saving the current work in the workspace as a file and loading the file later.

It also provides various commands for other system-related activities like, displaying date, listing files in the directory, displaying current directory, etc.

The following table displays some commonly used system-related commands:

Command	**Purpose**
cd	Changes current directory.
date	Displays current date.
delete	Deletes a file.
diary	Switches on/off diary file recording.
dir	Lists all files in current directory.
load	Loads workspace variables from a file.
path	Displays search path.
pwd	Displays current directory.
save	Saves workspace variables in a file.
type	Displays contents of a file.
what	Lists all MATLAB files in the current directory.
wklread	Reads .wk1 spreadsheet file.

Input and Output Commands

MATLAB provides the following input and output related commands:

Command	**Purpose**
disp	Displays contents of an array or string.
fscanf	Read formatted data from a file.
format	Controls screen-display format.
fprintf	Performs formatted writes to screen or file.
input	Displays prompts and waits for input.
;	Suppresses screen printing.

The **fscanf** and **fprintf** commands behave like C scanf and printf functions. They support the following format codes:

Format Code	**Purpose**
%s	Format as a string.
%d	Format as an integer.
%f	Format as a floating point value.

%e	Format as a floating point value in scientific notation.
%g	Format in the most compact form: %f or %e.
\n	Insert a new line in the output string.
\t	Insert a tab in the output string.

The format function has the following forms used for numeric display:

Format Function	**Display up to**
format short	Four decimal digits (default).
format long	16 decimal digits.
format short e	Five digits plus exponent.
format long e	16 digits plus exponents.
format bank	Two decimal digits.
format +	Positive, negative, or zero.
format rat	Rational approximation.
format compact	Suppresses some line feeds.
format loose	Resets to less compact display mode.

Vector, Matrix, and Array Commands

The following table shows various commands used for working with arrays, matrices and vectors:

Command	**Purpose**
cat	Concatenates arrays.
find	Finds indices of nonzero elements.
length	Computes number of elements.
linspace	Creates regularly spaced vector.
logspace	Creates logarithmically spaced vector.
max	Returns largest element.
min	Returns smallest element.
prod	Product of each column.
reshape	Changes size.
size	Computes array size.
sort	Sorts each column.
sum	Sums each column.
eye	Creates an identity matrix.
ones	Creates an array of ones.

zeros	Creates an array of zeros.
cross	Computes matrix cross products.
dot	Computes matrix dot products.
det	Computes determinant of an array.
inv	Computes inverse of a matrix.
pinv	Computes pseudoinverse of a matrix.
rank	Computes rank of a matrix.
rref	Computes reduced row echelon form.
cell	Creates cell array.
celldisp	Displays cell array.
cellplot	Displays graphical representation of cell array.
num2cell	Converts numeric array to cell array.
deal	Matches input and output lists.
iscell	Identifies cell array.

Plotting Commands

MATLAB provides numerous commands for plotting graphs. The following table shows some of the commonly used commands for plotting:

Command	**Purpose**
axis	Sets axis limits.
fplot	Intelligent plotting of functions.
grid	Displays gridlines.
plot	Generates xy plot.
print	Prints plot or saves plot to a file.
title	Puts text at top of plot.
xlabel	Adds text label to x-axis.
ylabel	Adds text label to y-axis.
axes	Creates axes objects.
close	Closes the current plot.
close all	Closes all plots.
figure	Opens a new figure window.
gtext	Enables label placement by mouse.
hold	Freezes current plot.
legend	Legend placement by mouse.
refresh	Redraws current figure window.

set	Specifies properties of objects such as axes.
subplot	Creates plots in sub windows.
text	Places string in figure.
bar	Creates bar chart.
loglog	Creates log-log plot.
polar	Creates polar plot.
semilogx	Creates semi log plot. (logarithmic abscissa).
semilogy	Creates semi log plot. (logarithmic ordinate).
stairs	Creates stairs plot.
stem	Creates stem plot.

6. M-FILES

So far, we have used MATLAB environment as a calculator. However, MATLAB is also a powerful programming language, as well as an interactive computational environment.

In previous chapters, you have learned how to enter commands from the MATLAB command prompt. MATLAB also allows you to write series of commands into a file and execute the file as complete unit, like writing a function and calling it.

The M Files

MATLAB allows writing two kinds of program files:

Scripts - script files are program files with **.m extension**. In these files, you write series of commands, which you want to execute together. Scripts do not accept inputs and do not return any outputs. They operate on data in the workspace.

Functions - functions files are also program files with **.m extension**. Functions can accept inputs and return outputs. Internal variables are local to the function.

You can use the MATLAB editor or any other text editor to create your **.m** files. In this section, we will discuss the script files. A script file contains multiple sequential lines of MATLAB commands and function calls. You can run a script by typing its name at the command line.

Creating and Running Script File

To create scripts files, you need to use a text editor. You can open the MATLAB editor in two ways:

Using the command prompt

Using the IDE

If you are using the command prompt, type **edit** in the command prompt. This will open the editor. You can directly type **edit** and then the filename (with .m extension)

```
edit

Or
```

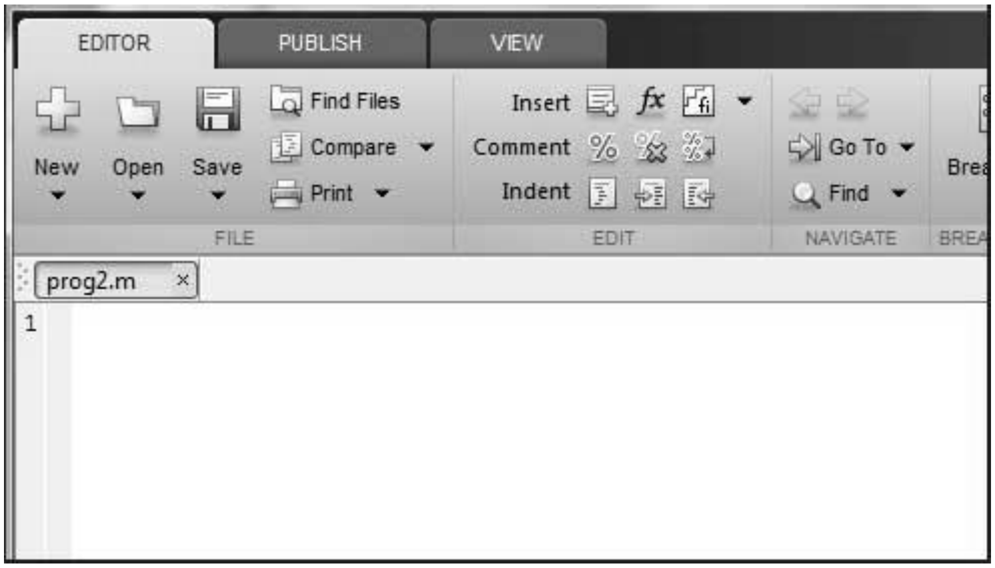
```
edit <filename>
```

The above command will create the file in default MATLAB directory. If you want to store all program files in a specific folder, then you will have to provide the entire path.

Let us create a folder named progs. Type the following commands at the command prompt(>>):

```
mkdir progs    % create directory progs under default directory

chdir progs    % changing the current directory to progs

edit  prog1.m  % creating an m file named prog1.m
```

If you are creating the file for first time, MATLAB prompts you to confirm it. Click Yes.



Alternatively, if you are using the IDE, choose NEW -> Script. This also opens the editor and creates a file named Untitled. You can name and save the file after typing the code.

Type the following code in the editor:

```
NoOfStudents = 6000;

TeachingStaff = 150;

NonTeachingStaff = 20;
```

```
Total = NoOfStudents + TeachingStaff ...

   + NonTeachingStaff;

disp(Total);
```

After creating and saving the file, you can run it in two ways:

Clicking the **Run** button on the editor window or

Just typing the filename (without extension) in the command prompt: >> prog1

The command window prompt displays the result:

```
6170
```

Example

Create a script file, and type the following code:

```
a = 5; b = 7;

c = a + b

d = c + sin(b)

e = 5 * d

f = exp(-d)
```

When the above code is compiled and executed, it produces the following result:

```
c =

   12

d =

  12.6570

e =

  63.2849

f =
   3.1852e-06
```

7. DATA TYPES

MATLAB does not require any type declaration or dimension statements. Whenever MATLAB encounters a new variable name, it creates the variable and allocates appropriate memory space.

If the variable already exists, then MATLAB replaces the original content with new content and allocates new storage space, where necessary.

For example,

```
Total = 42
```

The above statement creates a 1-by-1 matrix named 'Total' and stores the value 42 in it.

Data Types Available in MATLAB

MATLAB provides 15 fundamental data types. Every data type stores data that is in the form of a matrix or array. The size of this matrix or array is a minimum of 0-by-0 and this can grow up to a matrix or array of any size.

The following table shows the most commonly used data types in MATLAB:

Data Type	Description
int8	8-bit signed integer
uint8	8-bit unsigned integer
int16	16-bit signed integer
uint16	16-bit unsigned integer
int32	32-bit signed integer

uint32	32-bit unsigned integer
int64	64-bit signed integer
uint64	64-bit unsigned integer
single	single precision numerical data
double	double precision numerical data
logical	logical values of 1 or 0, represent true and false respectively
char	character data (strings are stored as vector of characters)
cell array	array of indexed cells, each capable of storing an array of a different dimension and data type
structure	C-like structures, each structure having named fields capable of storing an array of a different dimension and data type
function handle	pointer to a function
user classes	objects constructed from a user-defined class
java classes	objects constructed from a Java class

Example

Create a script file with the following code:

```
str = 'Hello World!'

n = 2345

d = double(n)
```

```
un = uint32(789.50)

rn = 5678.92347

c = int32(rn)
```

When the above code is compiled and executed, it produces the following result:

```
str =

Hello World!

n =

   2345

d =

   2345

un =

   790

rn =

   5.6789e+03

c =

   5679
```

Data Type Conversion

MATLAB provides various functions for converting a value from one data type to another. The following table shows the data type conversion functions:

Function	**Purpose**

Char	Convert to character array (string)
int2str	Convert integer data to string
mat2str	Convert matrix to string
num2str	Convert number to string
str2double	Convert string to double-precision value
str2num	Convert string to number
native2unicode	Convert numeric bytes to Unicode characters
unicode2native	Convert Unicode characters to numeric bytes
base2dec	Convert base N number string to decimal number
bin2dec	Convert binary number string to decimal number
dec2base	Convert decimal to base N number in string
dec2bin	Convert decimal to binary number in string
dec2hex	Convert decimal to hexadecimal number in string
hex2dec	Convert hexadecimal number string to decimal number
hex2num	Convert hexadecimal number string to double-precision number
num2hex	Convert singles and doubles to IEEE hexadecimal strings

cell2mat	Convert cell array to numeric array
cell2struct	Convert cell array to structure array
cellstr	Create cell array of strings from character array
mat2cell	Convert array to cell array with potentially different sized cells
num2cell	Convert array to cell array with consistently sized cells
struct2cell	Convert structure to cell array

Determination of Data Types

MATLAB provides various functions for identifying data type of a variable.

Following table provides the functions for determining the data type of a variable:

Function	**Purpose**
is	Detect state
isa	Determine if input is object of specified class
iscell	Determine whether input is cell array
iscellstr	Determine whether input is cell array of strings
ischar	Determine whether item is character array
isfield	Determine whether input is structure array field
isfloat	Determine if input is floating-point array

ishghandle	True for Handle Graphics object handles
isinteger	Determine if input is integer array
isjava	Determine if input is Java object
islogical	Determine if input is logical array
isnumeric	Determine if input is numeric array
isobject	Determine if input is MATLAB object
isreal	Check if input is real array
isscalar	Determine whether input is scalar
isstr	Determine whether input is character array
isstruct	Determine whether input is structure array
isvector	Determine whether input is vector
class	Determine class of object
validateattributes	Check validity of array
whos	List variables in workspace, with sizes and types

Example

Create a script file with the following code:

```
x = 3

isinteger(x)
```

```
isfloat(x)

isvector(x)

isscalar(x)

isnumeric(x)


x = 23.54

isinteger(x)

isfloat(x)

isvector(x)

isscalar(x)

isnumeric(x)


x = [1 2 3]
```