Scan or click here for more resources

Introduction to MATLAB

MATLAB stands for Matrix Laboratory. It is a high-performance language that is used for technical computing. It was developed by Cleve Molar of the company MathWorks.Inc in the year 1984.It is written in C, C++, Java. It allows matrix manipulations, plotting of functions, implementation of algorithms and creation of user interfaces.

Getting started with MATLAB:
It is both a programming language as well as a programming environment. It allows the computation of statements in the command window itself.

- **Command Window:**
  In this window one must type and immediately execute the statements, as it requires quick prototyping. These statements cannot be saved. Thus, this is can be used for small, easily executable programs.
- **Editor (Script):**
  In this window one can execute larger programs with multiple statements, and complex functions These can be saved and are done with the file extension '.m '
- **Workspace:**
  In this window the values of the variables that are created in the course of the program (in the editor) are displayed.
- <="" b="" style="box-sizing: border-box;">
  This window displays the exact location(path) of the program file being created.

MATLAB Library comes with a set of many inbuilt functions. These functions mostly perform mathematical operations like sine, cosine and tangent. They perform more complex functions too like finding the inverse and determinant of a matrix, cross product and dot product

Although MATLAB is encoded in C, C++ and Java, it is a lot easier to implement than these three languages. For example, unlike the other three, no header files need to be initialised in the beginning of the document and for declaring a variable, the data type need not be provided. It provides an

easier alternative for vector operations. They can be performed using one command instead of multiple statements in a for or while loop.

Some of the basic functions in MATLAB and their uses are listed below:

Function	Description
disp()	The values or the text printed within single quotes is displayed on the output screen
clear	To clear all variables
close all	To close all graphics window
clc	To clear the command window
exp(x)	To compute the exponential value of x to the base e
abs(x)	To compute the absolute value of x
sqrt(x)	To compute the square root of x
log(x)	To compute the logarithmic value of x to the base e
log10(x)	To compute the logarithmic value of x to the base 10
rem(x, y)	To compute the remainder of x/y
sin(x)	To compute the sine of x
cos(x)	To compute the cosine of x
tan(x)	To compute the tangent of x
atan2(x, y)	To compute the arctangent or inverse of y/x

Writing a MATLAB program:

1. **Using Command Window:**

   Only one statement can be typed and executed at a time. It executes the statement when the enter key is pressed. This is mostly used for simple calculations.

Note: ans is a default variable created by MATLAB that stores the output of the given computation.

2. **Using Editor:**

   Multiple lines of code can be written here and only after pressing the run button (or F5) will the code be executed. It is always a good practice to write clc, clear and close all in the beginning of the program.

Note: Statements ending with a semicolon will not be displayed in the command window, however, their values will be displayed in the workspace.

Any statement followed by % in MATLAB is considered as a comment

3. **Vector Operations:**

   Operations such as addition, subtraction, multiplication and division can be done using a single command instead of multiple loops

We can also extract separate rows and columns by using the colon(:) operator. Consider a matrix A of size 3X3. The following commands can be used to extract rows and columns from Matrix A

Command	Description
A(:, n)	To extract the elements of all rows in column n of the matrix
A(m, :)	To extract the elements of all columns in row m of the matrix
A(:, m:n)	To extract the elements of all rows between columns m and n of the matrix
A(m:n, :)	To extract the elements of all columns between rows m and n of the matrix
A(p:q, m:n)	To extract the elements of rows between p and q and columns between m and n of the matrix
A(m, n)	To extract the elements of row m and column n

Applications of MATLAB

MATLAB can be used as a tool for simulating various electrical networks but the recent developments in MATLAB make it a very competitive tool for Artificial Intelligence, Robotics, Image processing, Wireless communication, Machine learning, Data analytics and whatnot. Though its mostly used by circuit branches and mechanical in the engineering domain to solve a basic set of problems its application is vast. It is a tool that enables computation, programming and graphically visualizing the results.

The basic data element of MATLAB as the name suggests is the Matrix or an array. MATLAB toolboxes are professionally built and enable you to turn your imaginations into reality. MATLAB programming is quite similar to C programming and just requires a little brush up of your basic programming skills to start working with.

Below are a few applications of MATLAB –

- **Statistics and machine learning(ML)**
  This toolbox in MATLAB can be very handy for the programmers. Statistical methods such as descriptive or inferential can be easily implemented. So is the case with machine learning. Various models can be employed to solve modern-day problems. The algorithms used can also be used for big data applications.
- **Curve fitting**
  The curve fitting toolbox helps to analyze the pattern of occurrence of data. After a particular trend which can be a curve or surface is obtained, its future trends can be predicted. Further plotting, calculating integrals, derivatives, interpolation, etc can be done.
- **Control systems**
  Systems nature can be obtained. Factors such as closed-loop, open-loop, its controllability and observability, Bode plot, Nyquist plot, etc can be obtained. Various controlling techniques such as PD, PI and PID can be visualized. Analysis can be done in the time domain or frequency domain.
- **Signal Processing**
  Signals and systems and digital signal processing are taught in various engineering streams. But MATLAB provides the opportunity for proper visualization of this. Various transforms such as Laplace, Z, etc can be done on any given signal. Theorems can be validated. Analysis can be done in the time domain or frequency domain. There are multiple built-in functions that can be used.
- **Mapping**
  Mapping has multiple applications in various domains. For example, in Big data, the MapReduce tool is quite important which has multiple applications in the real world. Theft analysis or financial fraud detection, regression models, contingency analysis, predicting techniques in social media, data monitoring, etc can be done by data mapping.
- **Deep learning**
  Its a subclass of machine learning which can be used for speech recognition, financial fraud detection, medical image analysis. Tools such as time-series, Artificial neural network(ANN), Fuzzy logic or combination of such tools can be employed.
- **Financial analysis**
  An entrepreneur before starting any endeavor needs to do a proper survey and the financial analysis in order to plan the course of action. The tools needed for this are all available in

MATLAB. Elements such as profitability, solvency, liquidity, and stability can be identified. Business valuation, capital budgeting, cost of capital, etc can be evaluated.

- **Image processing**
  The most common application that we observe almost every day are bar code scanners, selfie(face beauty, blurring the background, face detection), image enhancement, etc. The digital image processing also plays quite an important role in transmitting data from far off satellites and receiving and decoding it in the same way. Algorithms to support all such applications are available.
- **Text analysis**
  Based on the text, sentiment analysis can be done. Google gives millions of search results for any text entered within a few milliseconds. All this is possible because of text analysis. Handwriting comparison in forensics can be done. No limit to the application and just one software which can do this all.
- **Electric vehicles designing**
  Used for modeling electric vehicles and analyze their performance with a change in system inputs. Speed torque comparison, designing and simulating of a vehicle, whatnot.
- **Aerospace**
  This toolbox in MATLAB is used for analyzing the navigation and to visualize flight simulator.
- **Audio toolbox**
  Provides tools for audio processing, speech analysis, and acoustic measurement. It also provides algorithms for audio and speech feature extraction and audio signal transformation.

MATLAB Commands

MATLAB is an interactive multi-programming language and numeric computing environment developed by MathWorks. MATLAB provides the Commands that will be used when the user wants to interact with any application using the command line interface.

Following are the lists of commands used in MATLAB.

Commands for Managing a Session

Command	Use
clc	Clears command window
clear	Removes the variables from memory
global	Declares variables globally
exist	Checks for the existence of a particular file or memory
help	Searches for a help topic
quit	Stops MATLAB

Command	Use
who	Lists current variables in use
lookfor	Searches help entries for a keyboard

Example 1

- *Matlab*

```
% MATLAB Code to
illustrate
% global
    function
setGlobalx(val)
   global x
```

Output:

```
x = val;
```

Example 2

- *Matlab*

```
% MATLAB Code to
illustrate who
   x = 2
   who
```

Output:

```
Variables in the current scope: x
```

Example 3

- *Matlab*

```
% MATLAB Code to
illustrate
% clear
   clear
   type x
```

Output:

error: type 'x' is undefined

Commands for Working with the System

Command	Use
cd	Changes current directory
date	Displays current date

delete	Deletes a file
dir	Lists all the files in a directory
path	Displays search path
pwd	Displays current directory
type	Displays content of a file
wklread	Reads.wk1 spreadsheet file
save	Saves workplace variables in a file

Example 1

- *Matlab*

```
% MATLAB Code to
illustrate
% pwd
    pwd
```

Output:

```
ans = /home/oo
```

Example 2

- *Matlab*

```
% MATLAB Code to
illustrate
% date
c = date
```

Output:

```
c= '18-jun-2021'
```

Input and Output Commands

Command	Use
disp	Displays content of an array or a string
fscanf	Reads formatted data from a file
format	Controls screen-display format

fprintf	Performs formatted writes to file or screen
input	Displays the prompt and waits for input
;	Suppresses screen printing

Example 1

- Matlab

```
% MATLAB Code to
illustrate
% disp

    A = [15 150];
    S = 'Hello
World.';
    disp(A)
    disp(S)
```

Output:

```
15 150


Hello World.
```

Example 2

- Matlab

```
% MATLAB Code to illustrate
%input
 age = input('how old are you: ');
     % At this point, the variable: age,
will contain
     % whatever value the user types
```

Output:

```
20
```

Vector, Matrix, and Array Commands

Command	Use
cat	Concatenates the array
find	Finds the indices of nonzero elements
max	Returns the largest element

min	Returns the smallest element
prod	Product of each column
sort	Sorts each column
rank	Computes rank of a matrix
eye	Creates an identity matrix
inv	Computes the inverse of a matrix

Example 1

- *Matlab*

```
% MATLAB Code to
illustrate
%cat
    array1 = [1,2,3]
    array2 = [4,5,6]
    cat(1,array1,array2)
```

Output:

```
ans =


1    2    3
4    5    6
```

Example 2

- *Matlab*

```
% MATLAB Code to
illustrate
%max
max(array1)
```

Output:

```
ans = 3
```

Example 3

- *Matlab*

```
%MATLAB Code to
illustrate
%min
min(array2)
```

Output:

```
ans = 4
```

- Matlab

```
%MATLAB Code to
illustrate
%eye
eye(2,2)
```

Output:

```
ans =
 Diagonal Matrix
 1    0
 0    1
```

Plotting Commands:

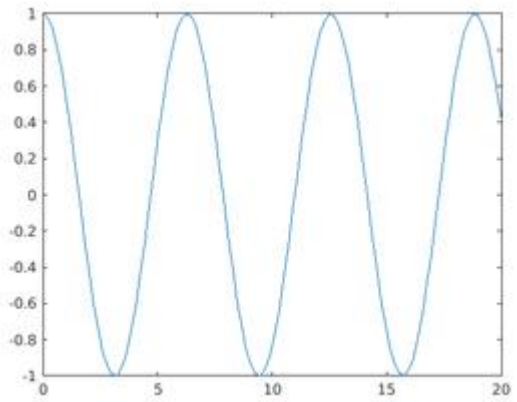
Commands	Use
axis	Sets axis limit
grid	Displays grid lines
plot	Generates xy plot
title	Puts title at top of the plot
close	Closes current plot
bar	Creates bar chart
print	Prints the plot / saves the plot
figure	Opens a new figure window
Close all	Closes all plots

Example1:

- Matlab

```
% MATLAB Code to
illustrate
%plot
    x = linspace(0,20)
    y = cos(x)
    plot(x,y)
```

Output:

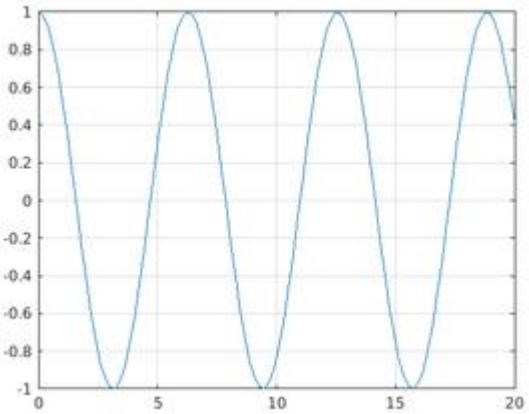


Example 2:

- Matlab

```
%MATLAB Code to
illustrate
%grid
    x =
linspace(0,20)
    y = cos(x)
    plot(x,y)
    grid on
```

Output:



Advantage of MATLAB

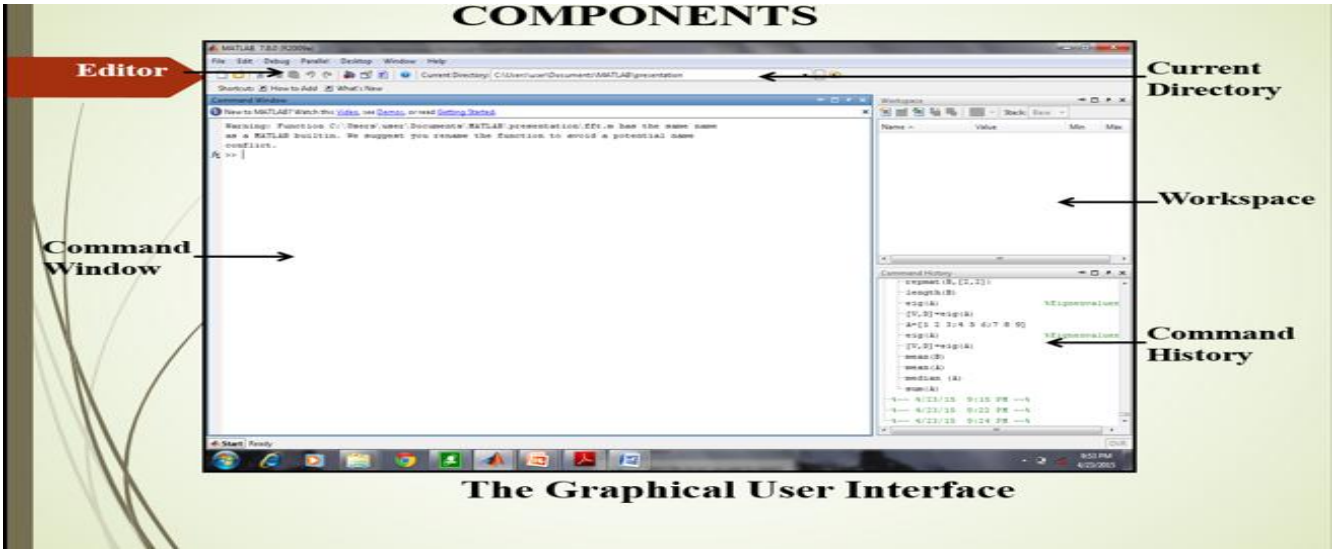
There are several advantages of MATLAB programming language:

MATLAB Compiler , Graphical User Interface,Device-Independent Plotting,Predefined Functions,Platform Independence,Ease of Use

Disadvantage of MATLAB

There is two major disadvantage of MATLAB programming language:

Cost,Interpreted language.



The desktop includes these panels:
Command Window – Issues commands for processing. This is the main area where commands can be entered at the command line. It is indicated by the command prompt (>>).
Current Directory -- GUI for directory and file manipulation. This panel allows to access the project folders and files.
Workspace -- GUI for viewing, editing, loading and saving variables. The workspace shows all the variables created and/or imported from files.
Command History – Returns history of prior commands issued in Command window. This panel shows or rerun commands that are entered at command line.
Editor – text editor is used for creating M-files

Comments

- Comments are like helping text in a program.
- They are ignored at the time of execution.
- Comments are entered by putting the % sign at the beginning of a line.
- e.g.
- % This is a comment line
- Comments can also be written at the end of a line of code.
- e.g.
- A=[1 2; 3 4]; % entering two-dimensional array
- You cannot have comments within comments.
- A block of comments can be added by using the block comment operators % { and % }.
- The MATLAB editor includes tools and context menu items to help to add, remove, or change the format of comments.

Variables:

- A variable is a tag that is assigned a value while that value remains in memory.
- The tag gives a way to reference the value in memory so that the program can read it, operate on it with other data, and save it back to memory.

Types of variables are :

Local variables

Global variables

Persistent variables

Local Variables:

- Each MATLAB function has its own local variables.
- Variables in a function are separated from those of other functions (except for nested functions), and from those of the base workspace.
- Variables defined in a function do not remain in memory from one function call to the next, unless they are defined as global or persistent.
- Scripts do not have a separate workspace. They store their variables in a workspace that is shared with the caller of the script. When called from the command line, they share the base workspace. When called from a function, they share that function's workspace.

Global Variables :

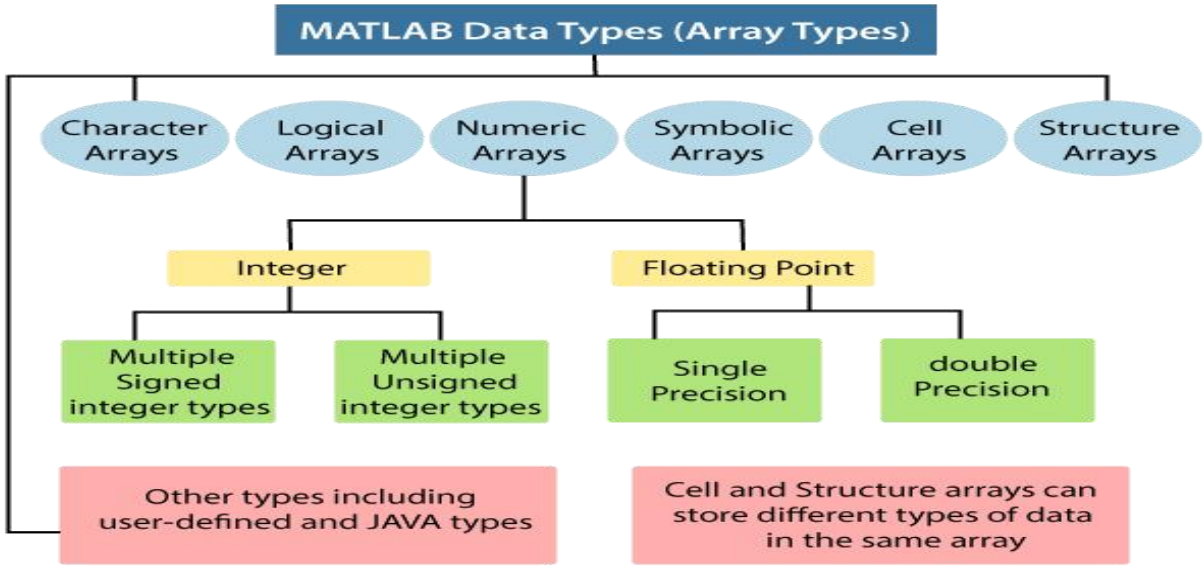
- If several functions (possibly the base workspace) declare a particular name as global, then they all share a single copy of that variable.
- Any assignment to that variable, in any function, is available to all the other functions declaring it global.
- Each function that uses a global variable must first declare the variable as global. Best way is to put global declarations toward the beginning of the function.
- If the M-file contains one or more sub-functions, then each sub-function requiring access to the global variable must declare it as global.
- To access the global variable from the MATLAB command line, variable must be declare as global at the command line.
- Global variable names are typically longer and more descriptive than local variable names, and often consist of all uppercase characters.
- e.g. global SALARY

Persistent Variables

- These variables can be declared and used within M-file functions only.
- e.g. persistent SUM
- Only the function in which the variables are declared is allowed to access these variables.
- These variables are not cleared from memory when the function exits, so their value is retained from one function call to the next.
- These variables should be declared before they are used in a function. It is usually best to put persistent declarations toward the beginning of the function.
- The function, in which a persistent variable is defined, is cleared (i.e., using clear functionname or clear all), or if the M-file for that function is edited, MATLAB clears all persistent variables used in that function.
- When a persistent variable is declared, it's value is initialized to an empty matrix, [ ].

MATLAB Data Types

The basic data type (also called a class) in MATLAB is the array or matrix. There are 15 fundamental data types in MATLAB. Each of these data types is in the build of a matrix or array. This matrix or arrays are a minimum of 0-by-0 in size and can increase to an n-dimensional array of any size.



The following table describes these data types are:

Data Type	Example	Description
int8, uint8, int16, uint16, int32, uint32, int64, uint64	uint16(65000)	The array of signed and unsigned integers. It requires less storage space than single or double. All integer data types except for int64 and uint64 can be used in mathematical operations.
single	3 * 10^38	The array of single-precision numbers. It requires less storage space than double but has less precision and a smaller range.
double	3 * 10^300 5 + 6i	Array of double-precision numbers. Two-dimensional array can be sparse. The default numeric types in MATLAB.
logical	magic(4) > 10	Array of logical value of 1 or 0 to represent true and false respectively. Two-dimensional arrays can be sparse.
char	'Hello'	Array of characters. Strings are represented as vectors of characters. For arrays including more than one string, it is good to use cell arrays.

cell array	a{1,1} = 12;
	a{1,2} = 'Red';
	a{1,3} = magic(4);
		Array of indexed cells, each capable of saving an array of a various dimension and data type.

structure	a.day = 12; a.color = 'Red';	Array of C-like structures, each structure having named fields capable of storing an array of a different dimension and data type.

	a.mat = magic(3);	
function handle	@sin	Pointer to a function. You can pass function handles to other functions.
user class	polynom([0 -2 -5])	Objects constructed from a user-defined class.
Java class	java.awt.Frame	Objects constructed from a Java class.

Numeric Types

Numeric data types in MATLAB contain signed and unsigned integers, and single- and double-precision floating-point numbers. Integer and single-precision array offer more memory efficient storage than double-precision.

All numeric types provide basic array functions, such as subscripting and reshaping. All numeric types except for int64 and uint64 can be used in numerical operations.

Integers

MATLAB has four signed and four unsigned integers data type.Signed types enable to work with a negative integer as well as positive, but cannot perform as wide a range of number as the unsigned type because one bit is used to designate positive or negative signs for the number. Unsigned types give a wider range of numbers, but these numbers can only be zero or positive.

Floating-Point Numbers

MATLAB show floating-point numbers in either double-precision or single-precision format. The default is double-precision, but we can make any number of single-precision with a simple conversion function.

Double-Precision Floating Point

MATLAB composes the double data type according to IEEE Standard 754 for double precision. Any value stored as a double-needed 64 bits, formatted as shown in the table below:

Bits	Usage
63	**Sign (0 = positive, 1 = negative)**
62 to 52	**Exponent, biased by 1023**
51 to 0	**Fraction f of the number 1.f**

Single-Precision Floating Point

MATLAB composes the single data type according to IEEE Standard 754 for single precision. Any value save as a single require 32 bits, formatted as shown in the table following:

Bits	Usage
31	Sign (0 = positive, 1 = negative)
30 to 23	Exponent, biased by 127
22 to 0	Fraction f of the number 1.f

Floating-Point Functions

Function	Description
double	It converts to double precision.
single	It converts to single precision.
class	It returns the data type of an object.
isa	It determines if the input value has the specified data type.
isfloat	It determines if the input value is a floating-point array.
isnumeric	It determines if the input value is a numeric array
eps	It returns the floating-point relative accuracy. This value is the tolerance MATLAB uses in its evaluation.
realmax	It returns the largest floating-point number your computer can represent.
realmin	It returns the smallest floating-point number our computer can represent.

Complex Numbers

Complex numbers consist of two separate part: a real part and an imaginary part. The primary imaginary unit is equal to the square root of -1. This is display in MATLAB by either of two letters: i or j.

Creating Complex Numbers

The following statement shows one method of creating a complex value in MATLAB. The variable x is assigned a complex number with a real part of 2 and an imaginary part of 3:

```
x = 2 + 3i;
```

Complex Number Functions

Function	Description
complex	It construct complex data from real and imaginary components.
i or j	It returns the imaginary unit used in constructing complex data.
real	It returns the real part of a complex number
imag	It returns the imaginary part of a complex number.
isreal	It determines if a number is real or imaginary.

MATLAB Operator

An operator is a symbol that tells the compiler to perform various numerical or logical manipulations. MATLAB is designed to operate mainly on whole matrices and arrays. Therefore, functions in MATLAB work both on scalar and non-scalar data.

MATLAB has several types of operators, symbols, and special characters to deal with variables, functions, and arithmetic operations.

MATLAB Arithmetic Operators

Arithmetic operators help in performing simple arithmetic operations like addition, subtraction, multiplication, division, and power.

Symbol	Role	Corresponding function
+	Addition	plus
+	Unary plus	uplus
-	Subtraction	minus
-	Unary minus	uminus
.*	Element-wise multiplication	times
*	Matrix multiplication	mtimes
./	Element-wise right division	rdivide
.\	Element-wise left division	ldivide
/	Matrix right division	mrdivide
\	Matrix left division	mldivide

.^	Element-wise power	power
^	Matrix power	mpower
.'	Transpose	transpose
'	Complex conjugate transpose	ctranspose

Arithmetic Operators and Arrays

Except for some matrix operators, MATLAB arithmetic operators work on corresponding functions of arrays with equal dimensions. For vectors and rectangular array, both operands must be the equivalent size unless one is a scalar. If one operand is a scalar and the other is not, MATLAB applies the scalar to every item of the other operand, this property is called scalar expansion.

This example uses scalar expansion to evaluate the product of a scalar operand and a matrix.

```
A = magic (3)
A =
     8     1     6
     3     5     7
     4     9     2
3 * A
ans=
    24     3    18
     9    15    21
    12    27     6
```

MATLAB Relational Operators

Relational operators perform value comparison operations.

Symbol	Role	Corresponding function
==	Equal to	eq
~=	Not equal to	ne
>	Greater than	gt
>=	Greater than or equal to	ge
<	Less than lt	

<=	Less than or equal to	le

Relational Operators and Arrays

The MATLAB relational operators compare corresponding components of arrays with equal dimensions. Relational operators always operate element-by-element. In this example, the resulting matrix present where the element of A is equal to the corresponding part of B.

```
A = [2 7 6; 9 0 5; 3 0.5 6];
B = [8 7 0; 3 2 5; 4 -1 7];
A == B
ans =
     0   1   0
     0   0   1
     0   0   0
```

For vectors and rectangular array, both operands must be the same size unless one is a scalar. In this case, where one operand is a scalar, and the other is not, MATLAB tests the scalar against every element of the other operand. Locations where the particular relation is true receive logical 1. Locations where the relation is false receive logical 0.

MATLAB Logical Operators

Logical operators perform logical operations and output the result in Boolean state true or false using the numbers 1 and 0, respectively.

MATLAB offer three types of logical operators and functions:

- **Element-wise:** It works on corresponding elements of logical arrays.
- **Bit-wise:** It works on corresponding bits of integer values or arrays.
- **Short-circuit:** It works on scalar, logical expressions.

The values returned by MATLAB logical operators and functions, with the exception of bit-wise functions, are of type logical and are suitable for use with logical indexing.

Element-Wise Operators and Functions

The following logical operators and functions execute element-wise logical operations on their inputs to produce a like-sized output array. The examples are shown in the following table use vector inputs A and B, where

```
A = [0 1 1 0 1];
B = [1 1 0 0 1];
```

Symbol	Role	Description	Example
&	Logical AND It returns 1 for every element location that is true (nonzero) in both arrays and 0 for all other elements.	A & B = 01001	
|	Logical OR It returns 1 for every element location that is true (nonzero) in either one or the other, or both arrays, and 0 for all other elements.	A | B = 11101	
~	Logical NOT It complements each element of the input array, A.	~A = 10010	
xor	It returns 1 for every element location that is true (nonzero) in only one array, and 0 for all other elements.	xor(A,B)=10100	

For operators and functions that take two array operands (&, |, and xor), both arrays must have the same dimensions, with each dimension being the same size. The one exception to this is where one operand is a scalar, and the other is not.

Note: MATLAB converts any finite nonzero, mathematic values used as inputs to logical expressions to logical 1, or true.

Bit-Wise Operator

The following functions execute bit-wise logical operations on nonnegative integer inputs. Inputs may be scalar or in arrays. If in arrays, these operations produce a like-sized output array.

The examples are present in the following table use scalar inputs A and B, where

```
A = 28;      % binary 11100
B = 21;      % binary 10101
```

Function	Description	Example
bitand	It returns the bit-wise AND of two nonnegative integer arguments.	bitand(A,B) = 20 (binary 10100)
bitor	It returns the bit-wise OR of two nonnegative integer arguments.	bitor(A,B) = 29 (binary 11101)

bitcmp	It returns the bit-wise complement as an n-bit number, where n is the second input argument to bitcmp.	bitcmp(A,5) = 3 (binary 00011)
bitxor	It returns the bit-wise exclusive OR of two nonnegative integer arguments.	bitxor(A,B) = 9 (binary 01001)

Short-Circuit Operators

The following operators execute AND and OR operations on logical expressions, including scalar values. They are short-circuiting operators in that they calculate their second operand only when the first operand does not fully determine the output.

Operator	Description
&&	It returns logical 1 (true) if both inputs calculate to true, and logical 0 (false) if they do not.
||	It returns logical 1 (true) if either input, or both, calculate to true, and logical 0 (false) if they do not.

MATLAB Special Characters

Special characters perform some particular tasks according to their behavior and the position where they are used.

Symbol	Symbol Name	Role
@	At symbol	○ Function manage construction and reference ○ Call super-class methods
.	Period or dot	○ Decimal point ○ Element-wise operations ○ Structure field access ○ Object property or method specifier
...	Dot dot dot or ellipsis	○ Line continuation

,	Comma	○ Separator
:	Colon	○ Vector creation ○ Indexing ○ For-loop iteration
;	Semicolon	○ Signify end of the row ○ Suppress output of code line
( )	Parentheses	○ Operator precedence ○ Function argument enclosure ○ Indexing
[ ]	Square brackets	○ Array concatenation ○ Array construction ○ Empty matrix and array element deletion ○ Multiple output argument assignment
{ }	Curly brackets	○ Cell array assignment and contents
%	Percent	○ Comment ○ Conversion specifier
%{ %}	Percent curly bracket	○ Block comments
!	Exclamation point	○ Operating system command
?	Question mark	○ Metaclass for MATLAB class
' '	Single quotes	○ Character array constructor
" "	Double quotes	○ String constructor
N/A	Space character	○ Separator

~	Tilde	○ Logical NOT ○ Argument placeholder
=	Equal sign	○ assignment

MATLAB String and Character Formatting Special Characters

There are some special characters to use only within the text of a character or string. These special characters are used to insert newlines or carriage returns, specify folder paths.

Symbol	Symbol Name	Role	Example
/ \	Forward-slash Backslash	File or folder path separation	Windows: dir([matlabroot '\toolbox\matlab\elmat\scriptview1.m']) or dir([matlabroot '/toolbox/matlab/elmat/scriptview1.m']) UNIX/Linux system: only forward slash dir([matlabroot '/toolbox/matlab/elmat/scriptview1.m'])
..	Dot dot	Parent folder	cd ..\..\example Goes up two levels and then down into the example folder
*	Asterisk	Wildcard character	dir('example_*.mat') Finds all files with names start with example and have a .mat extension
@	At symbol	Class folder indicator	\@myScriptClass\get.m
+	Plus	Package directory indicator	+mypack +mypack/scriptview1.m +mypack/@myScriptClass

MATH FUNCTIONS

- abs( ) : computes the absolute value of a variable or number
- sqrt( ) : computes the square root of a number
- exp( ) : computes exponential
- log( ) : computes natural logarithm
- rem( ) : computes the remainder of x/y such as rem(7,2)
- sign( ) : returns -1 if number< 0, 0 if number==0 and 1 if number> 0

- round( ) : round to the nearest integer
- fix( ) : round towards zero
- ceil( ) : round towards infinity
- mod( ) : modulus
- sin( ) : computes sine of number given in radians
- cos( ) : computes cosine of number given in radians
- tan( ) : computes tangent of number given in radians
- sinh( ) : computes hyperbolic sine of number given in radians
- asin( ) : computes inverse sine of number given in radians
- asinh( ) : computes inverse hyperbolic sine of number given in radians
- cosh( ) : computes hyperbolic cosine of number given in radians
- acos( ) : computes inverse cosine of number given in radians
- acosh( ) : computes inverse hyperbolic cosine of number given in radians
- tanh( ) : computes hyperbolic tangent of number given in radians
- atan( ) : computes inverse tangent of number given in radians
- atan2( ) : computes four quadrant inverse tangent
- atanh( ) : computes inverse hyperbolic tangent of number given in radians
- sec( ) : computes secant of a number given in radians
- sech( ) : computes hyperbolic secant of number given in radians
- asec( ) : computes inverse secant of number given in radians
- asech( ) : computes inverse hyperbolic secant of number given in radians
- csc( ) : computes cosecant of a number given in radians
- csch( ) : computes hyperbolic cosecant of number given in radians
- acsc( ) : computes inverse cosecant of number given in radians
- acsch( ) : computes inverse hyperbolic cosecant of number given in radians
- cot( ) : computes cotangent of a number given in radians
- coth( ) : computes hyperbolic cotangent of number given in radians
- acot( ) : computes inverse cotangent of number given in radians
- acoth( ) : computes inverse hyperbolic cotangent of number given in radians
- angle( ) : phase angle
- complex( ) : constructs complex data from real and imaginary part
- conj( ) : complex conjugate
- imag( ) : complex imaginary part
- real( ) : complex real part
- isreal( ) : true for real array