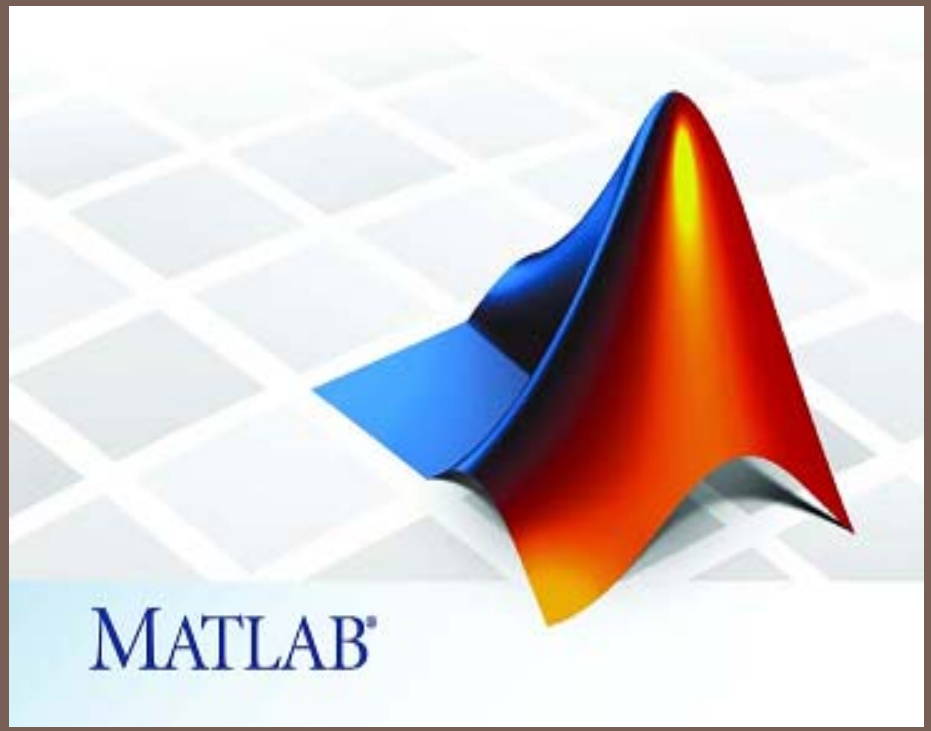
INTRODUCTION TO MATLAB



Principles & Basic Features of MATLAB

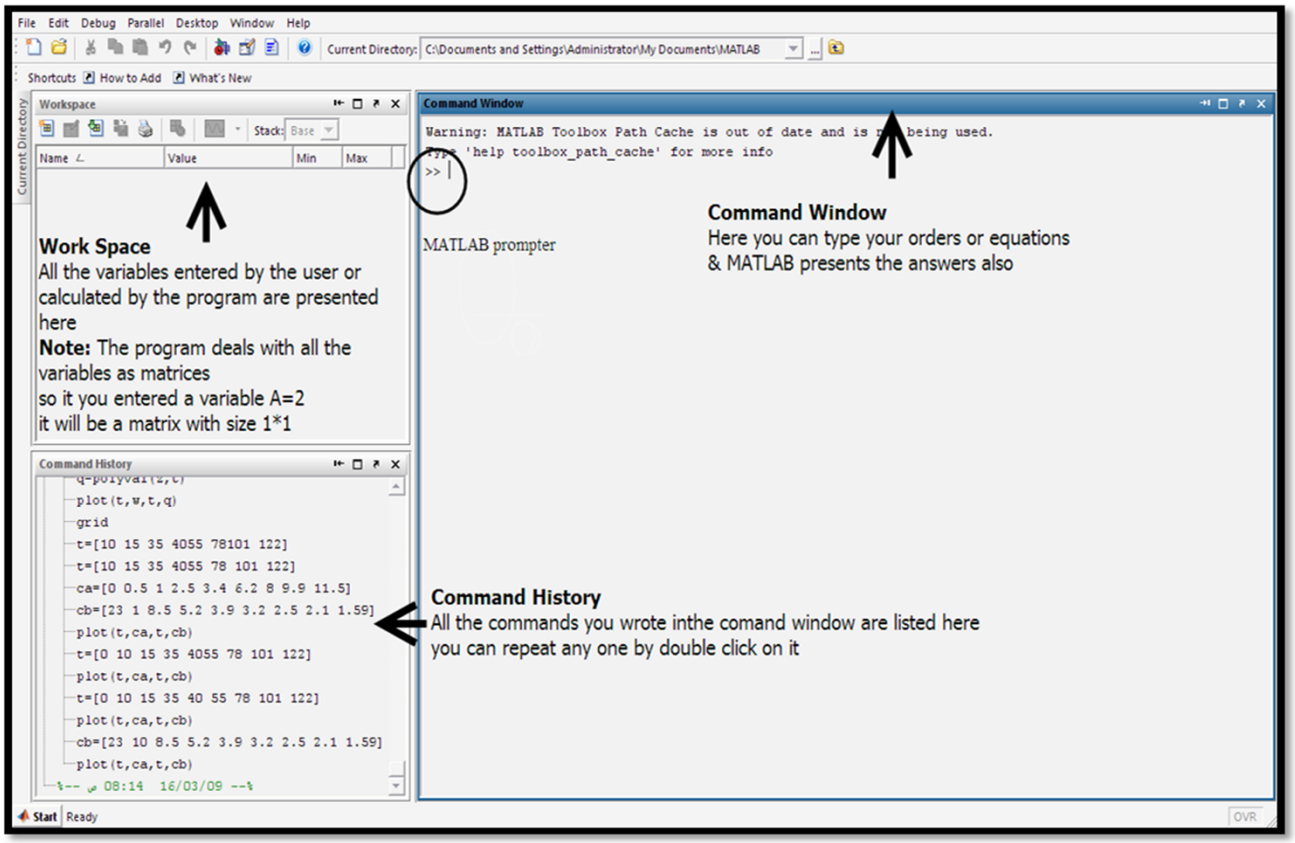
MATLAB is a software package that lets you do mathematics and computation, analyze data, develop algorithms, do simulation and modeling, and produce graphical displays and graphical user interfaces.

Key Features of MATLAB

- High-level language for numerical computation, visualization, and application development
- Interactive environment for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations
- Built-in graphics for visualizing data and tools for creating custom plots
- Tools for building applications with custom graphical interfaces
- Functions for integrating MATLAB based algorithms with C, Java, .NET, and Microsoft® Excel®

Principles & Basic Features of MATLAB

- MATLAB is started by clicking the START button, then clicking on the MATLAB icon from the PROGRAMS group. You can double-click the short-cut icon of MATLAB if it exists on your desktop.



Principles & Basic Features of MATLAB

- You get MATLAB to do things for you by typing in command window.
  - MATLAB prompts you with two greater-than signs (>>) when it is ready to accept a command from you.

Simple Mathematics & Fundamental Expressions

- Working in the MATLAB environment is straightforward because most commands are entered as you would write them mathematically.
- MATLAB is an expression language; the expressions you type are interpreted and evaluated.
- MATLAB statements are usually of the form: **variable = expression**, or simply expression

Simple Mathematics & Fundamental Expressions

If the variable name and = sign are omitted, a variable ***ans*** (for answer) is automatically created to which the result is assigned.

For example, entering the following simple expression

```
>> a = 4/3
```

yields the MATLAB response

```
a =

    1.3333
```

and assigns to variable **a** the value of 4 divided by 3.

Simple Mathematics & Fundamental Expressions

Variables Names:

- The first character in a variable name must be a letter.
- Contains no signs, e.g. * % / ? + , ……etc.
- Contains no spaces like: ***Gas pressure***.
  - You can type it ***Gas_pressure***
- Not an order or a function name, e.g. ***help, sin, ode45*** …
- In general, it is a good idea to use appropriate and memorable names for variables.
- MATLAB recognizes the first 19 characters of a variable name.
- **MATLAB is case sensitive**, so, for example, **a** and **A** are two different variables.

Simple Mathematics & Fundamental Expressions

- If you don't care to create a new variable but want to know the value of an expression, you can type the expression by itself, example:

```
>> 4/3
```

which yields

```
ans =

    1.3333
```

- where ***ans*** is a MATLAB-generated dummy variable that stands for “answer”.

Simple Mathematics & Fundamental Expressions

- Before you write any expression, all the variables in the right hand side from the equation must be pre-defined. For example if you wrote:

```
>>  3*5/n
```

returns:

??? Undefined function or variable 'n'.

Simple Mathematics & Fundamental Expressions

- If the last character of a statement is a *semicolon* (**;**), the printing is suppressed, but the assignment is carried out.
- This is essential in suppressing unwanted printing of intermediate results.
- For example:

```
>> b = 4+7;
```

will create a new variable **b** whose value is 11, but MATLAB will not display the value of **b**. If you wrote:

```
>> c = b+7
```

returns:

```
c =
     18
```

Simple Mathematics & Fundamental Expressions

- Moreover, you can check the value of a variable at any time by entering the variable name at the prompt as follows:

```
>> b
b =
     11
```

- This is an extremely useful feature, especially when you are debugging a sequence of expressions.

Simple Mathematics & Fundamental Expressions

- Expressions, as opposed to variables, can be made up of sequences of numbers, operators, functions and variables that have been previously defined. For example, since **a** and **b** have already been defined, we can do the following:

```
>> d = a*(b-1)
d =
        13.3333
```

Simple Mathematics & Fundamental Expressions

- Note that blank spaces can be used to improve readability.
- If you are typing in an expression that doesn't fit on one line, use ***ellipsis dots*** (...) at the end of the line and continue typing on the next line, e.g.

```
>> p = 1 + 2 + ...
          3 + 4 + 6

p =

    16
```

Arithmetic operators in MATLAB

- Arithmetic operators are the same as those commonly used, except that * represents multiplication, / performs left division and ^ is the power operator. For example typing

```
>>  p^2
ans    =
          256
```

Arithmetic operators in MATLAB

- **Note:** Left division is defined like right division except that the order of dividend and divisor is swapped, i.e., a\b = (a) $^{-1}$ b. For *a* and *b* scalars, a\b = b/a; in other words, the division is carried from left to right; for example:

```
>> y = 6\3
```

returns

```
y =

    0.5000
```

and not 2.000, as if we were dividing 6 by 3!

Arithmetic operators in MATLAB

- MATLAB performs operations in the following order:

1. ( ) Parentheses
2. ^ Power operator
3. * Multiplication and Division (/ and \)
4. + Addition and Subtraction -

- Precedence of like operators proceeds from left to right. But parentheses can be used to affect the order of operation.

Arithmetic operators in MATLAB

- The following three examples illustrate these precedence rules:

```
>> 1+2^3/4*2
ans =
     5
>> 1+2^3/ (4*2)
ans =
     2
>> (1+2)^3/(4*2)
ans =
      3.3750
```

Arithmetic operators in MATLAB

Parentheses must be written in the following cases:

- For the expression if it contains more than one parameter, e.g.: **A/(B+C)** or **4/(2-3)**
- For the expression if there is a summation or subtraction, e.g.: **(x+y)/4**
- For the power if it was fraction as , e.g.: **5^(1/7)**
- For any function, e.g.: **sin(x)**

Predefined variables in MATLAB

- These include **i** and **j**, both of which denotes . Using **i** or **j** to generate complex numbers can be very convenient.

```
>> i
ans =
     0 + 1.0000i
>> j
ans =
     0 + 1.0000i
```

Predefined variables in MATLAB

- However, predefined variables can be overwritten, so be careful using the variable names **i** and **j** to mean other than . For example, many people use *i* and *j* as indices for vectors and matrices. As a result, they often use **i** or **j** this way in MATLAB, e.g., by entering **i=5** to reassign **i** to be equal to 5 instead of i.

Predefined variables in MATLAB

Other predefined variables are:

pi, which stands for π

Inf, which stands for ∞

NaN, which stands for not a number (e.g. 0/0)

MATLAB will return ∞ or **NaN** when you divide by zero. When this happens, execution of your calculation will not be terminated:

```
>> d = 4/0
d =
   Inf
```

Predefined variables in MATLAB

- Similarly, MATLAB returns **NaN** when you attempt to perform certain undefined calculations e.g.,

```
>> dd =  Inf/Inf
dd  =
                    NaN
```

- In general, you should try to avoid generating **Inf** & **NaN** because they can lead meaningless results in your calculations.
- It is also important to know, however, that the permanent variable **eps** (epsilon) gives the machine precision about 10^{-16} on most machines. It is useful in determining tolerances for convergence of iterative processes.

MATLAB workspace & saving sessions

- MATLAB allows you to clear the command window.
  - ***clc*** command: clears the command window, and give you a fresh **>>** prompt.
  - ***clf*** command: clears the graphics window and leaves it blank. (We will talk about that later on).
  - ***clear*** command: removes all the variables in the work space.
  - If the **clear** command is followed by some variables, these variables are erased. If you type

    ```
    >> clear a b c
    ```

    only the variables a, b and c will be cleared.

MATLAB workspace & saving sessions

- The **who** command displays the names of all your variables available in the MATLAB workspace.
- The **whos** command gives you the names of all the variables, along with information about each variable's size, number of elements, number of bytes, density, and whether the variable is complex.

The MATLAB workspace & saving sessions

- While all computations in MATLAB are performed in double precision, the format of the displayed output can be controlled by the following commands:

>> ***format short***	fixed point with 4 decimal places (the default)
>> ***format***	Default. (Same as **short**).
>> ***format long***	fixed point with 14 decimal place
>> ***format short e***	scientific notation with 4 decimal places
>> ***format long e***	scientific notation with 15 decimal places
>> ***format bank***	Fixed format for currency (money).
>> ***format compact***	Suppress extra line-feeds.
>> ***format loose***	Puts the extra line-feeds back in.
>> ***format rat***	Fraction

The MATLAB workspace & saving sessions

- Once invoked, the chosen format remains in effect until changed. To see more types of format try:

```
>> help format
```

- For example, to see a five-digit floating point display (scientific notation) enter

```
>> c= 13.3330;
>> format short e
```

The new display will look as follows:

```
>> c
c   =
      1.3333e+01
```

Smart Recall

- Repeated use of the ↑ key recalls earlier commands.
- If you type the first few characters of a previous command and then press the ↑ key MATLAB will recall the last command that began with those characters.
- Subsequent use of ↑ will recall earlier commands that began with those characters.

MATHEMATICAL FUNCTIONS

Mathematical Functions

- MATLAB includes many standard functions that are easily incorporated into expressions.
  - All of the standard functions such as ***sin***, ***cos***, ***log***, ***exp***, ***sqrt***, as well as many others
- Each function is a block of code that accomplishes a specific task.
- A pair of parentheses follows each function, which includes the so-called ***argument(s) of the function***. For example:

```
>> sin(pi/4)
     ans =
     0.7071
```

pi/4 is the **argument** of the **sin** function.

Mathematical Functions

- The simplest functions has one input argument and one output argument
- Example: **sqrt** function, which returns the square root of the input argument.
- The input argument goes inside a pair of parentheses.

```
>> y = sqrt(1+4*i)
y =
   1.6005 + 1.2496 i
```

- Notice the way in which the complex number $(1 + 4i)$ was displayed.

Elementary & Trigonometric Functions

Elementary Functions		Trigonometric Functions	
abs ( )	absolute value	**sin ( )**	sine
sqrt ( )	square root	**cos ( )**	cosine
exp ( )	exponential base e	**tan ( )**	tangent
log ( )	natural logarithm	**asin ( )**	arcsine = $\sin^{-1}$
log10 ( )	log base 10	**acos ( )**	arccosine = $\cos^{-1}$
		atan ( )	arctangent = $\tan^{-1}$
real ( )	real part	**sinh ( )**	hyperbolic sine
imag ( )	imaginary part	**cosh ( )**	hyperbolic cosine
conj ( )	complex conjugate	**tanh ( )**	hyperbolic tangent
round ( )	round to nearest integer	**asinh ( )**	hyperbolic arcsine
fix ( )	round toward zero	**acosh ( )**	hyperbolic arccosine
ceil ( )	round towards plus infinity	**atanh ( )**	hyperbolic arctangent
floor ( )	round towards minus infinity		

Angles in MATLAB

- MATLAB deals with **angles in radian not in degrees**
  - you should convert the angle first from degree to radian by multiplying by pi/180.
- For the inverse trigonometric functions which returns the angle itself you should multiply the angle by 180/pi.

```
>> sin(90*pi/180)
     ans =
          1
```

```
>> asin(1)*180/pi
     ans =
          90
```

Scientific Notation

- If you want to write a constant in the scientific notation format e, K= 6.67×10^{-8} (K is Boltzmann's constant) you can write it:

```
>> K= 6.67*10^-8
```

- But it's easier to write this:

```
>> K= 6.67e-8
```

```
K =
   6.6700e-008
```

Exercise

- Compute the following:
  - $\frac{\sqrt[6]{10}}{2^3-1}$

```
>> 10^(1/6)/(2^3-1)
```

  - $e^5.\tan(30°)$

```
>> exp(5)*tan(30*pi/180)
```

  - $\ln(\sin^2(7\pi))$

```
>> log(sin(7*pi)^2)
```

  - $\log(2x10^5)$

```
>> log10(2e5)
```