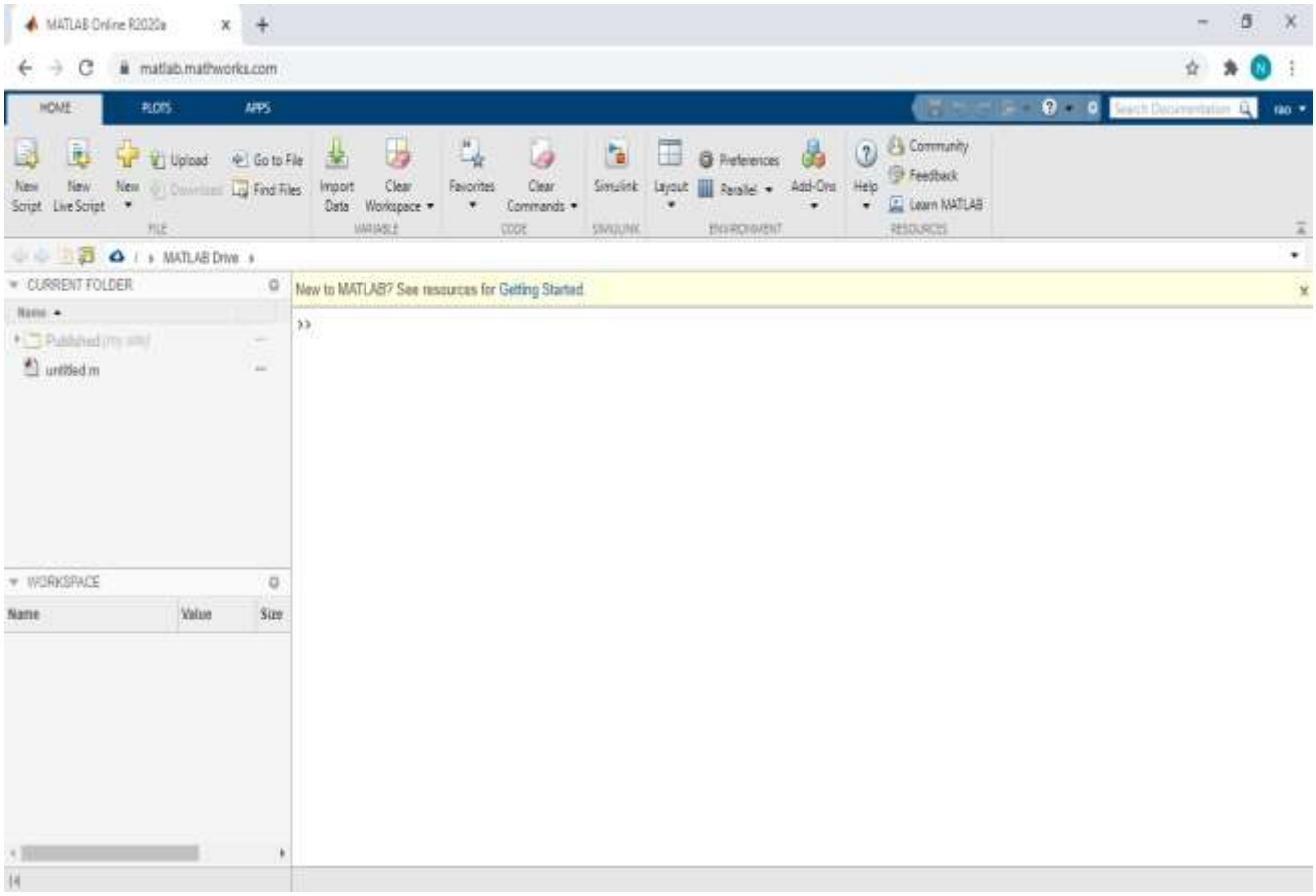
Introduction to MATLAB

The name MATLAB stands for MATrix LABoratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.

MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide. It has powerful built-in routines that enable a very wide variety of computations. It also has easy-to-use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

Desktop Basics



Basic arithmetic Operations

- A scalar is a 1×1 matrix.
- A row vector of length (say 3) is a 1×3 matrix.
- A column vector of length (say 4) is a 4×1 matrix.
- MATLAB can work as a calculator.

- The below table summarizes the arithmetic operators available in MATLAB.

Operation	Description	Operation	Description
+	Addition (a+b)	/	Right division (a/b)
-	Subtraction (a-b)	\	Left division (a\b)
*	Multiplication (a*b)	^	Exponentiation (a^b) eg. 5^3=5*5*=125
sqrt(x)	Square root of ($\sqrt{x}$)		

Basic MATLAB Commands

- The workspace contains variables that you create within or import into MATLAB from data files or other programs.
- You can view the contents of the workspace using

  >> whos (which gives the detailed information of the variables in the workspace) while >> who (the variables in short can be viewed with the command).
- >> clc (The clc function clears the Command Window)
- >> clear (To clear all the variables from the workspace, use the clear command)
- To get help on any command of MATLAB,

  >> help <Type a command>

Matrices and Arrays

- All MATLAB variables are multidimensional arrays, no matter what type of data.
- To create an array with four elements in a single row, separate the elements with either a comma (i.e., ,) or a space.

  >> a = [1,2,3,4] or >> a = [1 2 3 4]

- To create a matrix that has multiple rows, separate the rows with semicolons.

  >> a = [1,2,3;4,5,6;7,8,10] or >> a = [1 2 3;4 5 6;7 8 10]

- Some inbuilt functions, namely, ones, zeros and eye etc., may be also used.

  For example

(i) To create a 5x1 matrix of zeros,

  >> z = zeros(5, 1)

(ii) To create a 5x1 matrix of ones,

  >> z = ones(5, 1)

(iii) To create an identity matrix of order 4,

  >> z = eye(4)

Matrix Operations

Operation	Syntax	Remarks
Addition	>> C=A+B	Matrices A and B must be of same order.
Subtraction	>> D=A-B	Matrices A and B must be of same order.
Multiplication	>> C=A*B	Matrices A and B must be compatible for multiplication (A is of order mxp

		and B is pxn, then C is of order mxn.)
Power of a Matrix	>>C=A^ n	eg. A^3=A*A*A
Element-wise multiplication	>> C=A.*B	eg. A=[x, y]; B=[a, b], then A.*B=[x*a, y*b] Matrices A and B must be of same order.
Element-wise division	>> C=A./B	eg. A=[x, y]; B=[a, b], then A./B=[x/a, y/b] Matrices A and B must be of same order
Element-wise exponentiation	>> C=A.^p	eg. A=[x, y]; p=2; then A.^p=[x^2, y^2]
Matrix Transpose	>> transpose(A)	
Determinant of A	>>det(A)	
Inverse of A	>>inv(A)	
Display all the diagonal elements of A	>>diag(A)	
Order of A	>>size(A)	

Generating/Referencing Matrix Elements

Operation	Syntax	Remarks
To create an array	>> x=linspace(a, b, n)	Creates an array of n numbers from a to b.
	x=a : h: b	Creates an array from *a* to *b* with spacing h.
Length of a vector	`>> n=length(X)`	Displays the number of elements in the vector X
Size of an Array	`>>[m, n]=size(A)`	Displays the number of rows & columns of the matrix.
Referencing an element	`>> A(i, j)`	To pick out the (i, j)th element of matrix A.
Referencing a column	`>> A(:,j)`	To reference all the elements in the j-th column of A.
Referencing a row	`>> A(i,:)`	To reference all the elements in the i-th row of A.

Creating Symbolic Variables

There are two ways to create symbolic variables, syms and sym.

To create symbolic variables x and y using syms as

```
>> syms x  y
```

To create symbolic variable x as

```
>> x = sym('x')
```

With syms, you can create multiple variables in one command. Create the variables a, b and c.

```
>> syms a b c
```

Symbolic math computations

Symbolic arithmetic		
Declaring symbols	`>> syms x y`	Declaring that x and y are symbols
$y = x^3 - 3x^2 + 3x - 1$	`>>y=x^3-3*x^2+3*x-1`	Declaring symbolic functions
	`>>y(x)=x^3-3*x^2+3*x-1`	Function values can be directly obtained by calling with `y(a)`
Additional commands	`simplify(y), expand(y)`	simplifies the expression. expands the expression.
Substitution in different ways	`subs(y,3)`	Substitutes 3 in place of x for the function y=f(x)
	`subs(z,x,3)`	If z=g(x,y), and we want to substitute x=3, ie., to find g(3,y)
	`subs(z,{x,y},{2,3})`	If z=h(x,y), and we want to substitute x=2 and y=3, i.e., to find h(2,3)

Creating Symbolic Expressions

Suppose we want to study the quadratic function $ax^2 + bx + c = 0$, we first create the symbolic variables a, b, c and x.

```
>> syms a b c x
```

```
>>eqn = a*x^2 + b*x + c == 0;
```

```
>> solve(eqn, x)
```

or `>> roots([a b c])`

These results are

```
-(b + (b^2 - 4*a*c)^(1/2))/(2*a)
-(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

Transcendental functions

Operation	Syntax	Remarks
e^x	`>> exp(x)`	
$\log_e x$	`>> log(x)`	Base e
$\log_{10} x$	`>> log10(x)`	Base 10
sin *x*	`>> sin(x)`	Here *x* is in radians. When *x* is in degrees, we can use `sind(x),cosd(x),tand(x), secd(x), cscd(x).`
cos *x*	`>> cos(x)`	
tan *x*	`>> tan(x)`	
sec *x*	`>> sec(x)`	
cosec *x*	`>> csc(x)`	

Inverse trigonometric functions	`>> asin(x)`
	`>> acos(x)`
	`>> atan(x)`	

Calculus with Symbolic Math

Limits:

The following examples illustrate how to find the limit of a function $\lim_{x\to a} f(x)$.

Ex. 1. $\lim_{x\to\infty}\left(1-\frac{2}{x}\right)^x$

Type the following code in a script file and run.

```
syms x
f(x)=(1-2/x)^x;
L = limit(f(x), x, inf)
```

Output:

```
L=
exp(-2)
```

Ex. 2. $\lim_{x\to\infty}\left(\sqrt{9x^2+x}-3x\right)$

Type the following code in a script file and run.

```
syms x
f = sqrt(9*x^2 + x) - 3*x
L = limit(f, x, inf)
```

Output:

```
L=
1/6
```

Derivative of a $f(x)$ can be found directly by the syntax `diff(f(x),x)` and second derivative by `diff(f(x),x,2)`. The following code illustrates how to find derivative of $f(x)=\frac{x^2+4x+3}{\sqrt{x}}$.

```
syms x
f(x)=(x^2+4*x+3)/sqrt(x)
df_dx = diff(f(x), x)       %First derivative
d2_fx = diff(f(x), x,2)     %Second derivative
```

Output:

```
df_dx =
(2*x+4)/x^(1/2)-(x^2+4*x+3)/(2*x^(3/2))
d2_fx =
(3*(x^2+4*x+3))/(4*x^(5/2))-(2*x+4)/x^(3/2)+2/x^(1/2)
```