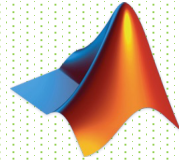




Introduction to MATLAB



Anil Kumar

A-408

Department of Mathematics

anilpundir@goa.bits-pilani.ac.in

What is MATLAB?

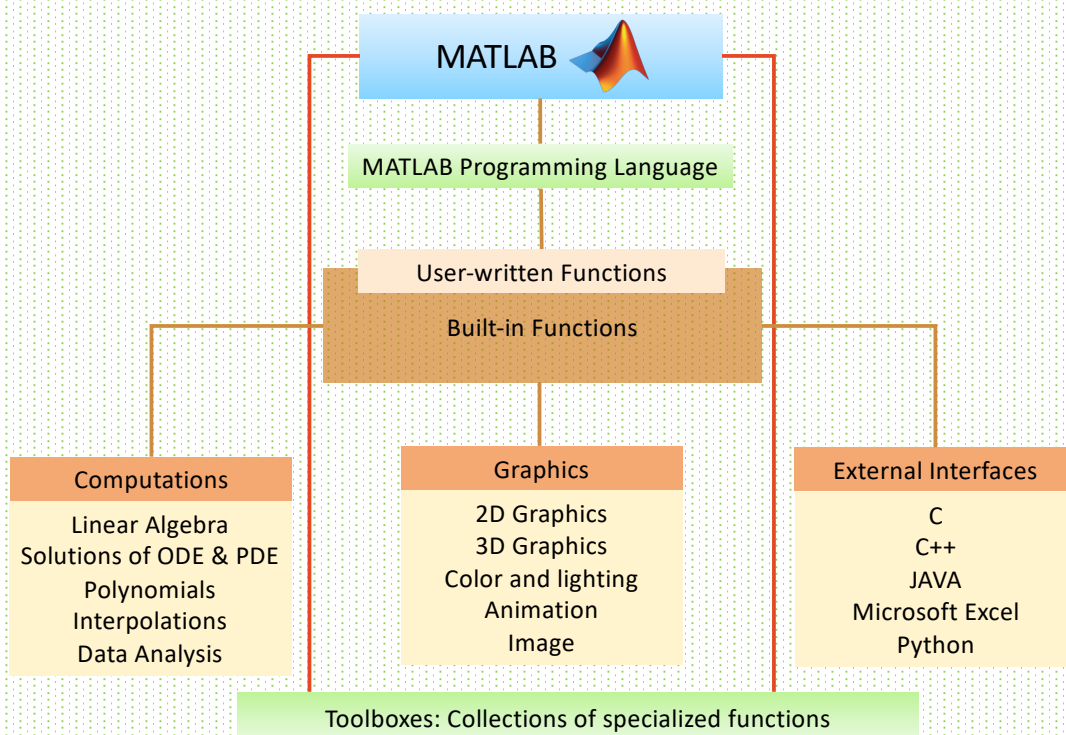


- A high-performance language for technical computing (Mathworks, 1984).
- A program for doing numerical computation. It was initially designed for solving linear algebra-type problems using matrices. The name is derived from **MAT**rix **LAB**oratory.
- MATLAB has since been expanded and now has built-in functions for solving problems requiring data analysis, signal processing, optimization, and other scientific computations. It also contains functions for 2-D and 3-D graphics and animation.
- Not a computer language, though it does most of the work of a computer language.

Why MATLAB?



- Easy to learn.
- Does not require in-depth knowledge of computer programming, such as compiling and linking.
- Interactive software, used in various areas of engineering and scientific applications.
- a **high-level language** that has many specialized toolboxes for making things easier for us.
- Provides an extensible programming/visualization environment.
- Provides professional-looking graphs.



Starting and Quitting MATLAB



- **Starting MATLAB**

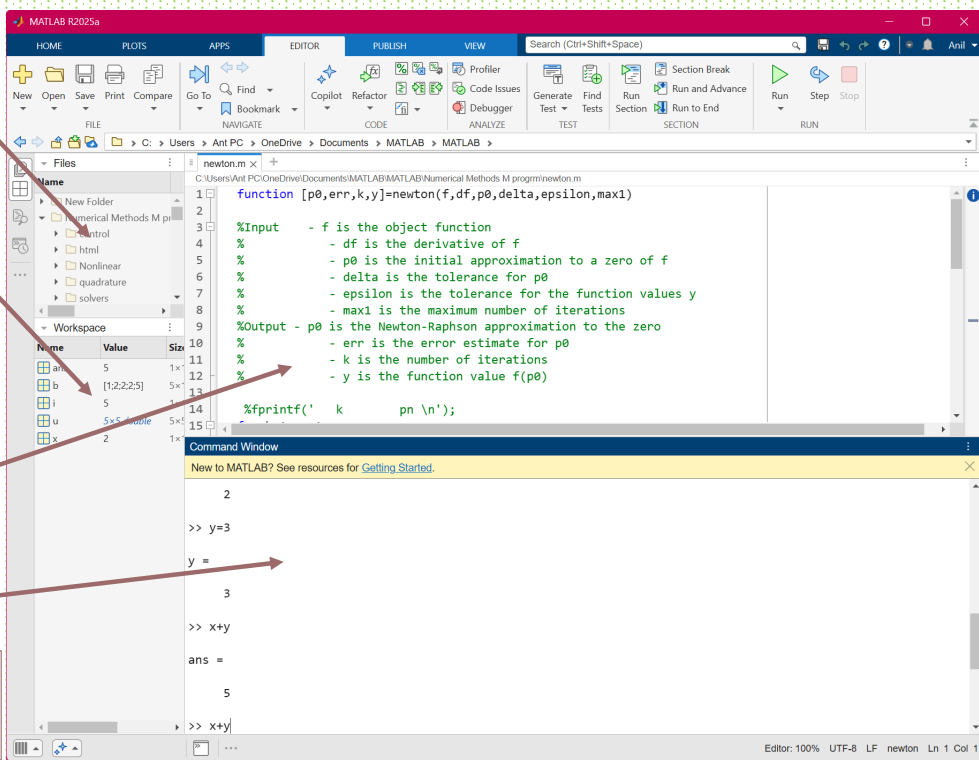
On **Windows** platforms, start MATLAB by double-clicking the MATLAB shortcut icon  on your Windows desktop.

On **UNIX** platforms, start MATLAB by typing MATLAB at the operating system prompt.

- **Quitting MATLAB**

Select **File** → **Exit MATLAB** in the desktop, or type quit in the Command Window.

MATLAB Screen



The image shows the MATLAB R2025a interface with several components annotated by red arrows and text boxes:

- Current Directory**: View folders and m-files. (Points to the File Explorer on the left)
- Workspace**: Contains variables, created or imported into MATLAB from data files or other programs. (Points to the Workspace pane on the left)
- Editor**: Create and debug M-files, which are programs you write to run MATLAB functions. (Points to the main code editor area)
- Command window**: Type commands. (Points to the Command Window at the bottom)
- Command History**: Previously run statements, copy & execute selected statements. (Points to the Command History pane at the bottom)

The MATLAB interface displays a function file named `newton.m` in the Editor. The function is defined as:

```
function [p0,err,k,y]=newton(f,df,p0,delta,epsilon,max1)
```

The function includes comments explaining the inputs and outputs, and a loop for the Newton-Raphson method. The Command Window shows the execution of the function with the following output:

```
>> y=3
y =
3
>> x+y
ans =
5
>> x+y
```

MATLAB help

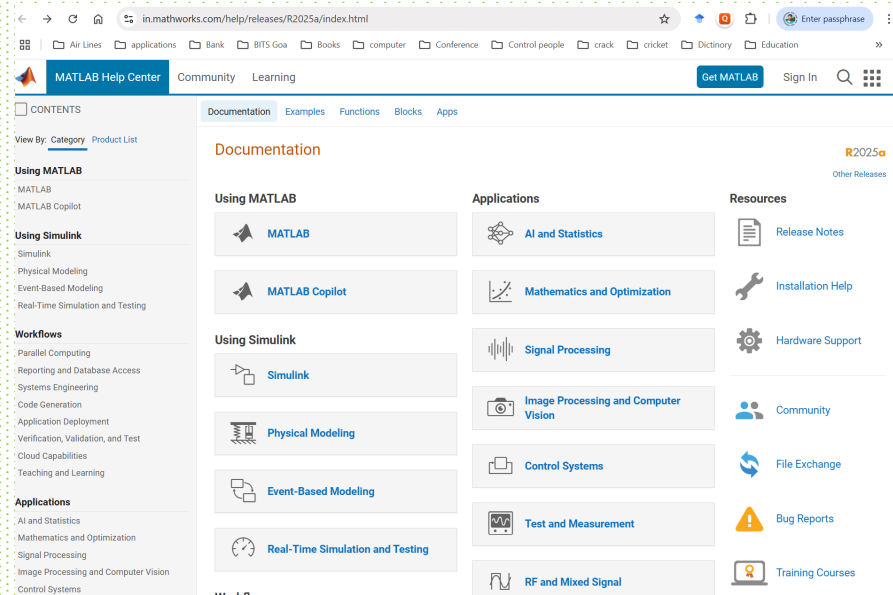
In command window:

```
>> help function_name
```

Example:

```
>>help sqrt
```

<https://in.mathworks.com/help/releases/R2025a/index.html>



Anil Kumar, Mathematics, BITS Goa

8

Some useful commands

- **what** List all the m-files in the current directory
- **dir/ls** List all files in the current directory
- **type temp** Display temp.m in the command window
- **delete temp** Delete temp.m
- **cd/chdir** Change directory
- **pwd** Show current directory
- **which temp** Display the directory path to the 'closest' temp.m
- **who** List the known variables
- **whos** List known variables plus their size
- **clear** Clear variables from the workspace
- **clc** Clear the command window

Anil Kumar, Mathematics, BITS Goa

9



Variables in MATLAB

- Begin with an alphabetic character: a
- Case sensitive: a, A
- Variable names can contain up to 2048 characters (MATLAB 2025). Use **namelengthmax** command to verify it.
- No data typing: a = 10; a = 'OK'; a = 2.5
- Default output variable: ans
- Built-in constants: **pi, i, j, inf, eps, realmax, realmin**
- Special characters: **[], (), { }, ;, %, ., :, =, ., ..., @**

Symbols...



>>	prompt
...	continue statement on next line
,	separate statements and data
%	start comment which ends at the end of the line
;	suppress output or use as a row separator in a matrix
:	specify range

Relational Operators



- MATLAB supports six relational operators:

Less Than	<	
Less Than or Equal	<=	
Greater Than	>	
Greater Than or Equal	>=	
Equal To	==	(The = character is for assignment, whereas the == character is for comparing the elements in two arrays.)
Not Equal To	~=	

Math operators



Power	\wedge or $\text{.}\wedge$	a^b or $a.^b$
Multiplication	$*$ or $\text{.}\ast$	$a*b$ or $a.*b$
Division	$/$ or ./	a/b or $a./b$
or	\backslash or $\text{.}\backslash$	$b\backslash a$ or $b.\backslash a$
Addition	$+$	$a + b$
Subtraction	$-$	$a - b$
Assignment	$=$	$a = b$ (assign b to a)

Mathematical Functions



- sqrt, sin, cos, sinh, asin, acos, exp, log, etc. (Example: [b1.m](#))
- Can handle Specialized mathematical functions, e.g., Bessel, Legendre, beta function, etc.

NUMBER DISPLAY FORMATS

By default,

- if a result is an integer, it is displayed as an integer.
- if a real number, four digits to the right of the decimal place are displayed.
- Can change this default behavior.

(Example: [b2.m](#))

Vectors and Matrices in MATLAB



- MATLAB treats all variables as matrices. For our purposes, a matrix can be thought of as an array; in fact, that is how it is stored.
- Vectors are special forms of matrices and contain only one row OR one column.
- Scalars are matrices with only one row and one column.

Vectors and Matrices in MATLAB



Row vectors:

- Start with a left bracket, enter the desired values separated by spaces (or commas), and then close it using a right bracket.

1. Colon notation

$x = \text{first} : \text{last}$

$x = \text{first} : \text{increment} : \text{last}$

2. linspace command:

$x = \text{linspace}(a, b)$: row vector x of 100 points linearly spaced between and including a and b .

$x = \text{linspace}(a, b, n)$: generates a row vector x of n points linearly spaced between and including a and b .

(Example: [b3.m](#))

Vectors and Matrices in MATLAB



• Column vectors

Specify it element by element and separate values with semicolons.

Note: Separating elements by spaces or commas specifies elements in different columns, whereas separating elements by semicolons specifies elements in different rows.

• Transpose of a vector

$b = a'$ transposes

Vectors and Matrices in MATLAB



Formation of matrices

- ✓ Commas or spaces are used to separate elements in a specific row, and semicolons are used to separate individual rows.
- ✓ *Alternately*, pressing the return or enter key while entering a matrix also tells MATLAB to start a new row.

Matrix Operations:

- Addition, subtraction, multiplication(\cdot), division by a scalar(\cdot) applied to all elements in the array/ matrix.
- $\text{size}(A)$ gives the order of the matrix.
- A' gives the transpose of A .
- $A(:, 3)$ gives the third column of A .
- $A(1, :)$ gives the first row of A .

Vectors and Matrices in MATLAB



- $A(1,:) + A(3,:)$ adds the first and third rows of A .
- $C=A+B$, $C=A-B$, $C=A*B$
- Inverse of A : $\text{inv}(A)$
- determinant of matrix A : $\text{det}(A)$
- rank of matrix A : $\text{rank}(A)$
- Identity matrix of order n : $\text{eye}(n)$
- $\text{trace}(A)$ gives the summation of the diagonal elements of a square matrix A .
- an $n \times m$ zero matrix : $\text{zeros}(n,m)$
- an ' $n \times m$ ' matrix having random entries: $\text{rand}(n,m)$

(Example: [b3.m](#))



Exercise:

1. Assume that a, b, c and d are defined as follows.

$$a = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} -1 & 2 \\ 0 & 1 \end{bmatrix}, \quad c = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \quad d = 5.$$

2. What is the result each of the following expressions?

$$a + b, \quad a.*b, \quad a.*d \quad \text{and} \quad a*d.$$

Vectors and Matrices in MATLAB



Solution of linear equations: $Ax = y$

Assuming the system has a unique solution, we can obtain x as:

- $x = \text{inv}(A)*y$ (computes inverse explicitly)
- $x = A \backslash y$

(Example: [b4.m](#))



Exercise:

- Solve the following linear systems

$$\begin{array}{ll}
 \text{(1). } \begin{array}{rcl} \frac{3}{2}x_1 & + & 3x_3 = 15 \\ -x_1 + 7x_2 - 9x_3 & = & -45 \\ 2x_1 & + & 5x_3 = 22 \end{array} &
 \text{(2). } \begin{array}{rcl} 3x_1 + 6x_2 & - & 3x_4 = 3 \\ x_1 + 3x_2 - x_3 - 4x_4 & = & -12 \\ x_1 - x_2 + x_3 + 2x_4 & = & 8 \\ 2x_1 + 3x_2 & = & 8 \end{array} &
 \text{(3). } \begin{array}{rcl} x_1 + x_2 + 2x_3 + 6x_4 & = & 11 \\ 2x_1 + 3x_2 + 6x_3 + 19x_4 & = & 36 \\ 3x_2 + 4x_3 + 15x_4 & = & 28 \\ x_1 - x_2 - x_3 - 6x_4 & = & -12 \end{array}
 \end{array}$$

Visualization



- Simple plotting: `plot(x, y)` (Example: [b5.m](#))
- Plotting with titles, labels, and grid lines (Example: [b6.m](#))
- Plotting multiple data (Example: [b7.m](#))
- Line color, Line style, Marker style, and Legends (Example: [b8.m](#))
- Subplot can be used for plotting more than one set of axes in a single figure:
`subplot(m,n,p)` (Example [b9.m](#))

This creates m x n subplots in the current figure, arranged in m rows and n columns, and selects the subplot p Grid position for new axes.

- For 3D plots: `plot3` command, `mesh(x,y,z)`, `surf(x,y,z)` and `contour(x,y,z)`.
(Example: [b10.m](#))



Exercise:

- Plot the following data

1. $y = e^{-0.2t}(\cos(t) + \sin(t)), t \in [0, 2\pi]$.
2. $y = \tan(\sin(x)) - \sin(\tan(x)), x \in [-\pi, \pi]$.
3. Write a MATLAB statements required to plot $\sin(x)$ versus $\cos(2x)$ from 0 to 2π in steps of $\pi/10$.



Loops and Logical statements

- for loop

```
for i=1:100
    a(i,i)=2*i;
end
```

- Multiple for loops are also possible.

```
for i=1:100
    for j=1:50
        for k=1:50
            a(i,j)=b(i,k)*c(k,j)+a(i,j);
        end
    end
end
```

Loops and Logical statements



- While

```
while condition  
statements  
end
```

- if, elseif, else, end

```
if condition #1  
statement #1  
elseif condition #2  
statement #2  
else  
statement #3  
end.
```

(Example: [b11.m](#))

Anil Kumar, Mathematics, BITS Goa

26

Exercise:



1. Write a MATLAB program to evaluate a function $f(x, y)$ for any two user specified values x and y . The function $f(x, y)$ is defined as follows.

$$f(x, y) = \begin{cases} x + y, & \text{if } x \geq 0 \text{ and } y \geq 0, \\ x + y^2, & \text{if } x \geq 0 \text{ and } y < 0, \\ x^2 + y, & \text{if } x < 0 \text{ and } y \geq 0, \\ x^2 + y^2, & \text{if } x < 0 \text{ and } y < 0. \end{cases}$$

2. Consider the iterative scheme $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ and write a MATLAB code to find the real root of the following equations correct to four decimal points.

(a) $\tan x = 4x$, (b) $x^4 + x^2 - 80 = 0$, (c) $\exp(x) - 3x^2 = 0$, (d) Cube root of 31.

Anil Kumar, Mathematics, BITS Goa

27

ODE solver



- Many problems that arise in science and engineering require a knowledge of a function $y = y(t)$ that satisfies the *first-order differential equation*

$$y' = f(t, y)$$

and the initial condition $y(a) = y_0$

where a and y_0 are given real numbers and f is a bivariate function that satisfies certain smoothness conditions.

ODE solver



- MATLAB has several functions for computing a numerical solution of the initial value problems for the ODEs.
- Some are listed in the following table:

Function	Application	Method used
ode23	Nonstiff ODEs	Explicit Runge-Kutta (2, 3) formula
ode45	Nonstiff ODEs	Explicit Runge-Kutta (4, 5) formula
ode113	Nonstiff ODEs	Adams-Bashforth-Moulton solver
ode15s	Stiff ODEs	Solver based on the numerical differentiation formulas
ode23s	Stiff ODEs	Solver based on a modified Rosenbrock formula of order 2

ODE solver



- A simplest form of the syntax for the MATLAB ODE solvers is

$$[t, y] = \text{solver}(\text{fun}, \text{tspan}, y0)$$

where **fun** is a string containing the name of the ODE m-file that describes the differential equation, **tspan** is the interval of integration, and **y0** is the vector holding the initial value(s).

- If **tspan** has more than two elements, then the solver returns computed values of **y** at these points.
- The output parameters **t** and **y** are the vectors holding the points of evaluation and the computed values of **y** at these points.

(Example: [Q1_1.m](#))

Exercise



- Find the numerical solution for the following initial value problems (IVPs)

1. $\frac{dy}{dt} = \frac{1}{t + y + 1}, y(0) = 0, 0 \leq t \leq 1.$

2. $\frac{dy}{dt} = -\left(e^{-10^4 t} + 1\right)(y - 1), y(0) = 2, 0 \leq t \leq 1.$

3. $\frac{dy}{dt} = -2yt, y(0) = 1, 0 \leq t \leq 1.$

Polynomials



- If $p(x) = ax^2 + bx + c$, then we use to enter this polynomial into MATLAB:

$$p = [a \ b \ c]$$

- *polyval* (*p*, *x*) evaluates a polynomial of degree *n* at *x*.
- *roots*(*p*) computes the zeros of *p*(*x*).

(Example: [b13.m](#))

Write a MATLAB program to find the roots of a quadratic equation $ax^2 + 2bx + c = 0$, whether they are real or complex roots. Use this program to find out the roots of the following equations.

(a) $x^2 + 5x + 6 = 0$, (b) $x^2 + 4x + 4 = 0$, (c) $x^2 + 2x + 5 = 0$, (d) $x^2 + x - 30 = 0$.

Curve fitting into the data



- Polynomial Curve Fitting: $p = \text{polyfit}(x, y, n)$
where *x* and *y* are vectors containing the *x* and *y* data to fit, and *n* is the order of the polynomial to return.
- *fitobject* = *fit*(*x*, *y*, *fitType*): creates the fit to the data in *x* and *y* with the model specified by *fitType*.
- *fitobject* = *fit*([*x*, *y*], *z*, *fitType*): creates a surface fit to the data in vectors *x*, *y*, and *z*.

(Example: [b14.m](#), [b15.m](#))

How to install MATLAB?



- Go to the Mathworks website.
- Create a login using the BITS email.
- Download and follow the instructions to install MATLAB using the following link: <https://in.mathworks.com/downloads/>
- In case of any problem/assistance, contact **Mr. Pushparaj M Paradkar**, Embedded Systems Lab, Department of Electrical & Electronics Engineering:

Email: pushparaj@goa.bits-pilani.ac.in

Phone: 0832-2580-230

References



- <https://in.mathworks.com/help/releases/R2025a/index.html>
- Biran, Adrian & Breiner, Moshe, **MATLAB for engineers**, Wokingham: Addison-Wesley, 1995.
- Gilat, Amos, **MATLAB: an introduction with applications**, Singapore: John Wiley, 2004.
- Chen, Ke, Giblin, Peter & Irving, Alan, **Mathematical explorations with MATLAB**, Cambridge: Cambridge University Press, 1999.
- Chapman, Stephen J., **MATLAB Programming for Engineers**, Thomson Brooks/Cole U.K., 2002



THANK
YOU