MCA 105 Programming with MATLAB

By the end of this paper, you should:

1. Know the **basic operations** in MATLAB.
2. Understand the **MATLAB environment** (workspace, editor, command window).
3. Perform **simple calculations** and math functions.
4. Carry out **numerical computations & analysis** (like solving equations, differentiation, integration).
5. Learn to build **applications and GUI (Graphical User Interface)**.
6. Get a **beginner-friendly overview** of MATLAB toolboxes and features.

🎓 Outcomes (What *you* will be able to do)

When you finish:

- Use the MATLAB **desktop & GUI** confidently.
- Write **scripts & functions**.
- Solve math/science problems using MATLAB.
- Visualize results with **2D & 3D plots**.
- Understand how to use **toolboxes** (like linear algebra, signal processing, etc.).

🔹 Unit I – Introduction & Basics

- Features of MATLAB.
- Workspace & Desktop layout.
- Writing and running scripts (M-files).
- Using MATLAB as a **calculator** (basic arithmetic, variables, comments).
- Working with **complex numbers**, floating point, built-in math functions.
- Logical & relational operations.
- Small applications like solving math problems.

👉 This is like **getting started with MATLAB**.

🔹 Unit II – Arrays, Matrices & Strings

- Creating vectors (1D arrays).
- Indexing, slicing, reshaping arrays.
- Sorting arrays, building multi-dimensional arrays.
- Matrix operations (addition, multiplication, transpose, inverse).
- String operations (joining, searching, manipulating text).

👉 Focus here is **linear algebra and data handling**.

🔹 Unit III – Control Flow & Functions

- if, if-else, switch-case, for, while, nested loops.
- try-catch (error handling).
- Writing **functions** (with input/output arguments).
- Sub-functions, nested functions.
- Function handles & anonymous functions (@(x) x^2+1).

👉 This teaches **programming logic in MATLAB**.

🔹 Unit IV – Graphics & Visualization

- **2D plots**: plot(), multiple graphs, style options.
- **Specialized 2D plots**: bar chart, stem plot, histogram.
- **3D plots**: mesh, surface, contour, line plots.
- Changing viewpoints & adding colors.

👉 This is about **data visualization**.

🔹 Unit V – Applications

- **Linear Algebra**: solving linear equations, eigenvalues, eigenvectors.
- **Polynomials**: roots, addition, multiplication, division, curve fitting.
- **Data Analysis**: reading data, processing, statistical functions.
- **Calculus**: differentiation, integration.
- **Differential Equations**: solving ODEs with MATLAB.

👉 This is **real-world application** of MATLAB.

UNIT 1 – Introduction to MATLAB (Exam Notes)

1. Features of MATLAB

- High-level programming language.
- Combines **computation + visualization + programming**.
- Easy-to-use environment.
- Supports **matrix & vector** operations natively.
- Built-in **mathematical & engineering functions**.
- Supports **2D & 3D graphics**.
- Extensible via **toolboxes** (e.g., Image Processing, Signal Processing).
- Provides **Graphical User Interface (GUI) development**.

2. MATLAB Workspace

- **Workspace** → Area where variables created during a session are stored.
- Commands to manage workspace:
  - who → lists variables.
  - whos → lists variables with details.
  - clear → removes variables.
  - clc → clears command window.
  - close all → closes all figures.

3. MATLAB Desktop

Main components:

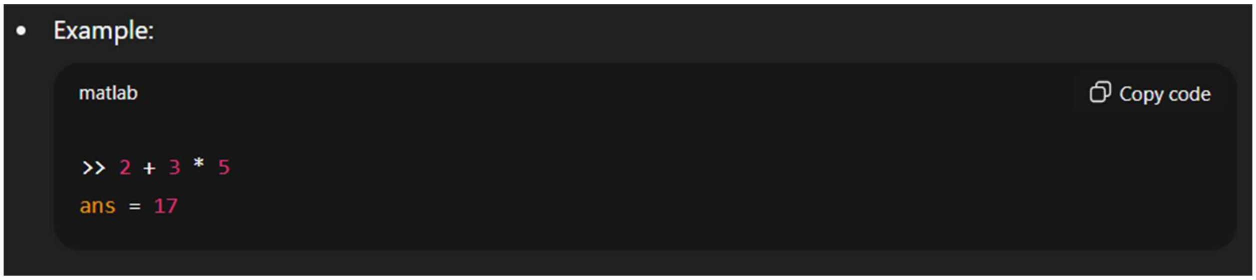
1. **Command Window** – where commands are executed.
2. **Editor (M-file Editor)** – to write & run scripts/functions.
3. **Workspace Browser** – shows variables in memory.
4. **Current Folder** – shows accessible files.
5. **Command History** – previously entered commands.

4. M-Files (Scripts)

- **M-file** → A file with extension .m.
- Two types:
  - **Script** – sequence of commands, operates on workspace variables.
  - **Function** – has input/output arguments, works independently.
- **Creating M-file**:
  - Open editor → write commands → save as filename.m.
  - Run using filename.

5. MATLAB as a Calculator

- Can perform basic arithmetic operations:
  - Addition: +
  - Subtraction: -
  - Multiplication: *
  - Division: / (right division), \ (left division)
  - Power: ^

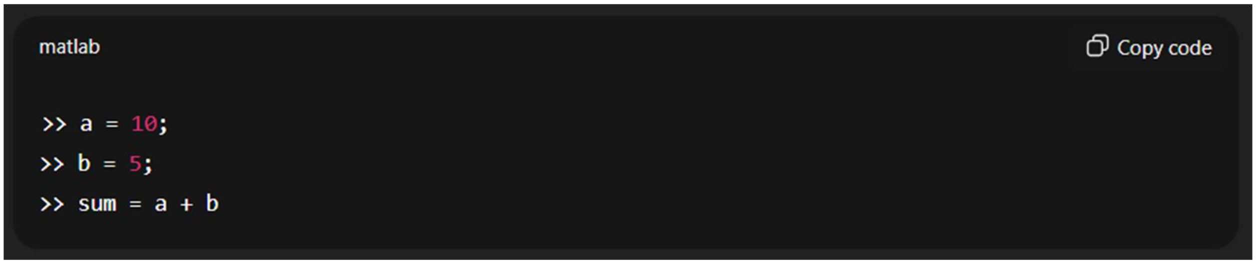


- Example:

```matlab
>> 2 + 3 * 5
ans = 17
```

6. Variables

- No declaration required, created when a value is assigned.
- Naming rules:
  - Start with a letter.
  - Case sensitive.
  - Can contain letters, digits, underscore.
- Example:



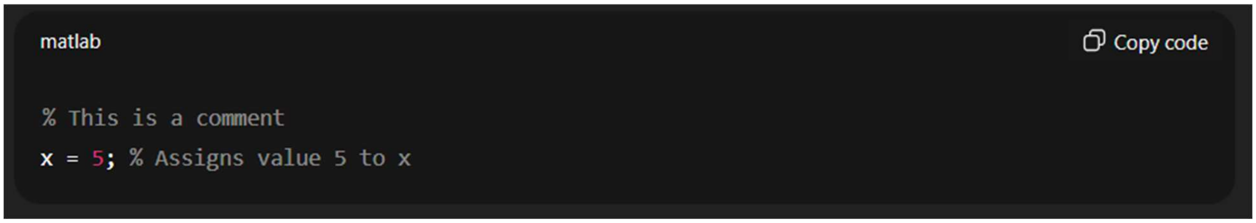
```matlab
>> a = 10;
>> b = 5;
>> sum = a + b
```

7. Comments

- Used to explain code, ignored by MATLAB.

- Symbol: %

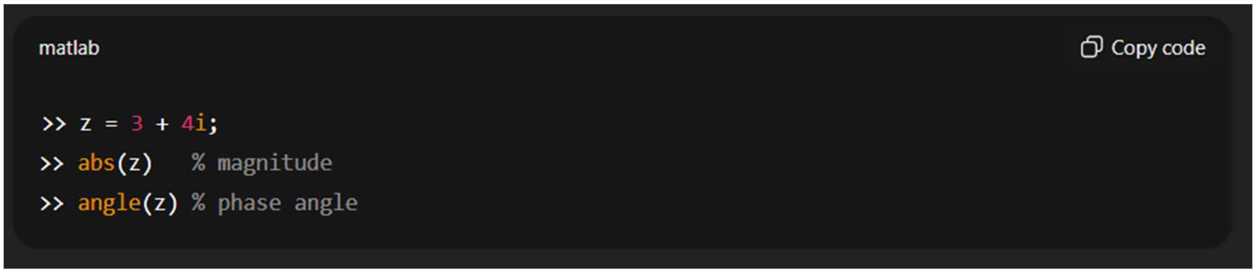
- Example:



```matlab
% This is a comment
x = 5; % Assigns value 5 to x
```

8. Complex Numbers

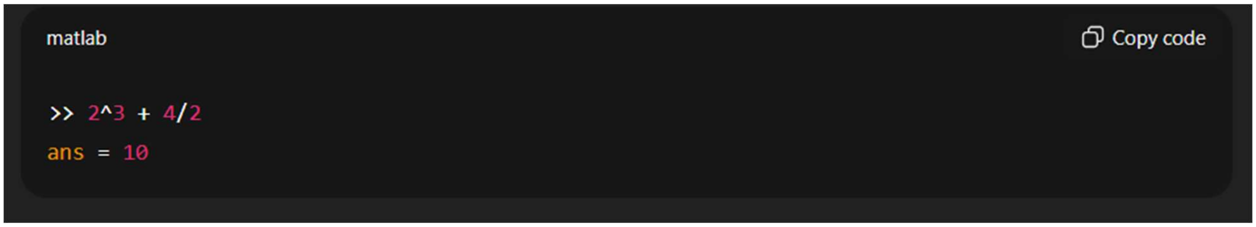
- Represented as a + bi or a + bj.
- Example:



```
>> z = 3 + 4i;
>> abs(z)   % magnitude
>> angle(z) % phase angle
```

9. Arithmetic with Scalars

- Scalar operations apply to single numbers.
- Operators follow **precedence rules** (BODMAS).
- Example:



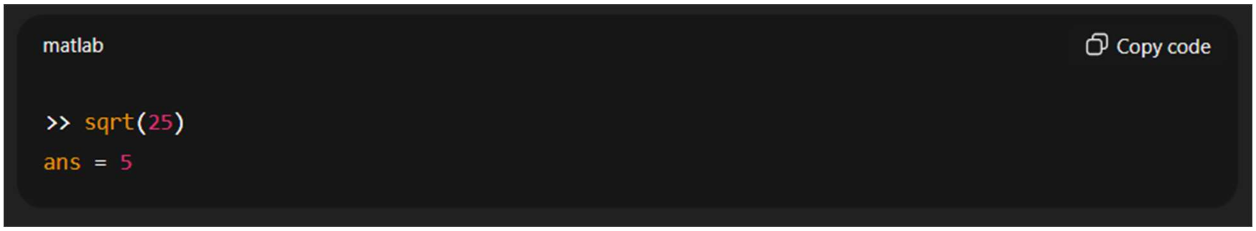
```
>> 2^3 + 4/2
ans = 10
```

10. Floating-Point Arithmetic

- MATLAB stores numbers as **double precision (64-bit)** by default.
- Special values:
  - Inf → infinity (1/0).
  - NaN → Not-a-Number (0/0).

11. Mathematical Functions

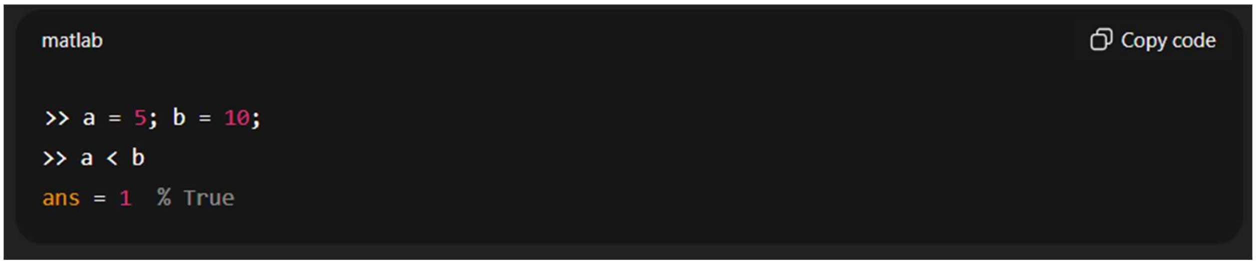
- Built-in math functions available:
  - sqrt(x), log(x), exp(x), sin(x), cos(x), tan(x).
  - abs(x) → absolute value.
- Example:



```matlab
>> sqrt(25)
ans = 5
```

12. Relational & Logical Operations

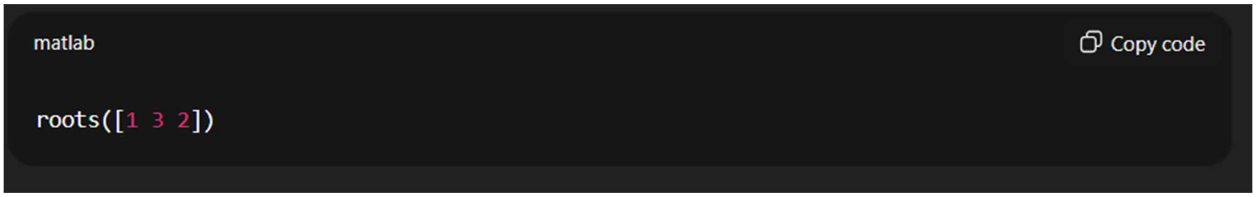
- Relational: <, >, <=, >=, ==, ~=.
- Logical: & (AND), | (OR), ~ (NOT).
- Example:



```matlab
>> a = 5; b = 10;
>> a < b
ans = 1  % True
```

13. Applications in Problem Solving

- Solving equations, numerical computations, and quick testing.
- Example: Solving quadratic equation x^2 + 3x + 2 = 0:



```matlab
roots([1 3 2])
```

☑ Quick Revision Points

- MATLAB = Matrix Laboratory.
- Workspace → stores variables.
- .m files → Scripts & Functions.
- % for comments.
- Complex numbers with i or j.
- Functions: sqrt, exp, log, sin, cos, abs.
- Special values: Inf, NaN.
- Relational ops: <, >, ==, ~=.

UNIT II – Arrays and Matrices

1. Arrays in MATLAB

- **Definition:** Array = ordered collection of numbers/characters arranged in rows & columns.
- **Types:**
    1. **One-dimensional array (Vector):**
        - Row vector → a = [1 2 3 4]
        - Column vector → b = [1; 2; 3; 4]
    2. **Two-dimensional array (Matrix):**
        - A = [1 2 3; 4 5 6; 7 8 9]
    3. **Multi-dimensional arrays:**
        - Arrays with >2 dimensions (e.g., images).
        - Example: B = rand(3,3,4) → 3×3×4 array.

---

2. Array Addressing & Indexing

- MATLAB uses **1-based indexing** (not 0-based like C).
- Access elements:
    - A(2,3) → element at row 2, column 3.
    - A(2,:) → entire 2nd row.
    - A(:,3) → entire 3rd column.
    - A(end,:) → last row.
- Colon operator : is powerful for slicing.

3. Array Manipulation

- **Transpose:** A' (matrix transpose).
- **Concatenation:**
  - Row-wise: [A B]
  - Column-wise: [A; B]
- **Reshape:** reshape(A, m, n) → changes shape without changing data.
- **Flipping:**
  - flipud(A) → flips up-down.
  - fliplr(A) → flips left-right.

---

4. Array Sorting

- **sort(A)** → sorts each column ascending.
- **sort(A,2)** → sorts each row.
- **sortrows(A)** → sorts rows of a matrix.

---

5. Multi-dimensional Arrays

- Arrays with 3 or more indices.
- Example: A = zeros(3,3,3) → 3D array of zeros.
- Access: A(2,3,1) → element at (2,3) in first 2D slice.

6. Built-in Functions for Handling Arrays

- length(A) → length of vector.
- size(A) → dimensions of array.
- numel(A) → number of elements.
- zeros(m,n) → m×n matrix of 0's.
- ones(m,n) → m×n matrix of 1's.
- eye(n) → identity matrix of size n.
- rand(m,n) → random numbers (uniform distribution).
- diag(A) → extracts diagonal elements.

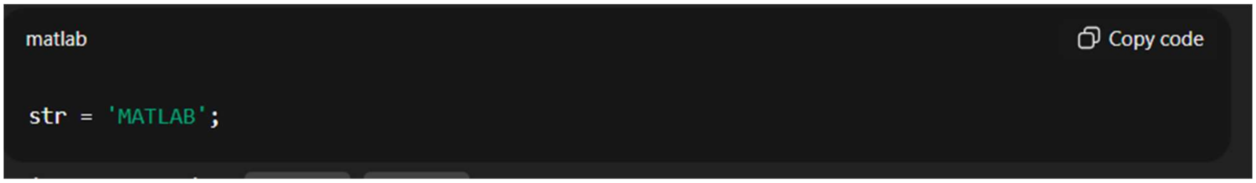
---

7. Matrix Manipulation & Operations

- **Addition/Subtraction:** C = A + B
- **Multiplication:**
  - Element-wise: A .* B
  - Matrix product: A * B
- **Division:**
  - Right division: A / B
  - Left division: A \ B
- **Power:**
  - Element-wise: A.^2
  - Matrix power: A^2
- **Inverse:** inv(A)
- **Determinant:** det(A)
- **Rank:** rank(A)

8. Character Strings in MATLAB

- Strings are stored as arrays of characters.
- Example:



```matlab
str = 'MATLAB';
```

- **String construction:** 'Hello', "World"
- **Concatenation:** strcat('Hello', 'World') → HelloWorld.
- **Useful functions:**
  - length(str) → length of string.
  - lower(str) → convert to lowercase.
  - upper(str) → convert to uppercase.
  - strcmp(str1,str2) → compares strings.
  - strfind(str,'at') → finds substring position.
  - strrep(str,'old','new') → replace substring.

📌 Quick Revision Points

- MATLAB indexing starts from **1**.
- : (colon operator) = slicing tool.
- size(), length(), numel() → array properties.
- **Element-wise operations:** .*, ./, .^
- **Matrix operations:** *, /, ^
- Strings are arrays of characters with many built-in functions.

☑ Most Expected Exam Questions (Unit II):

1. Explain array addressing and indexing in MATLAB with examples.
2. Write short notes on array manipulation functions.
3. Differentiate between element-wise operations and matrix operations.
4. Explain multi-dimensional arrays in MATLAB.
5. What are the different string manipulation functions in MATLAB?

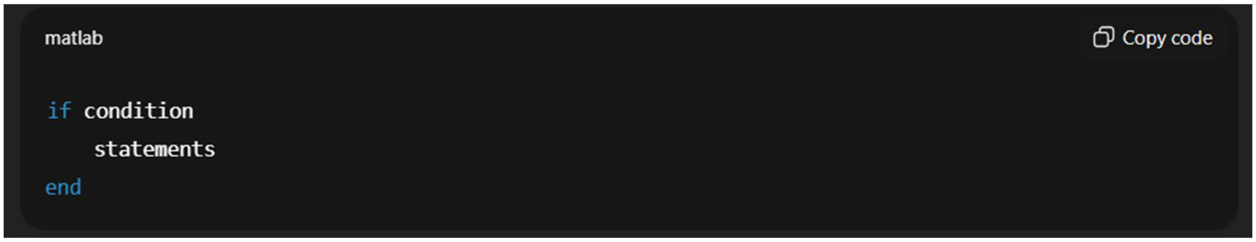
UNIT III – Control Flow & Functions

1. Control Flow in MATLAB

Control flow statements let you **direct the execution path** of your program.

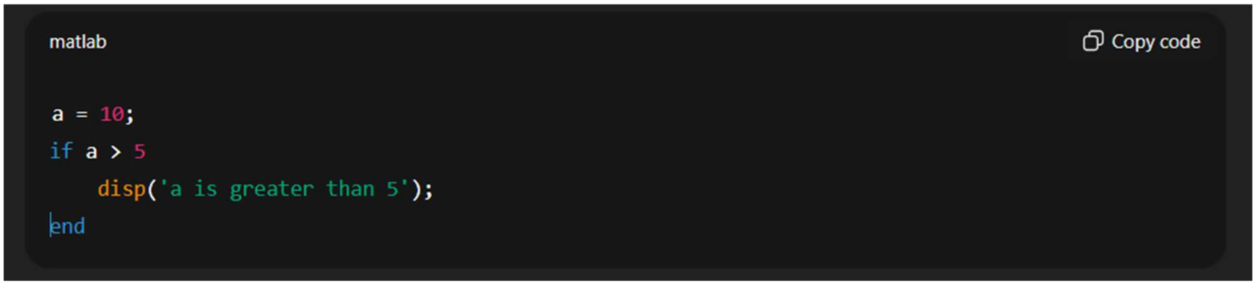
A. if Statement

- **Syntax:**



```matlab
if condition
    statements
end
```

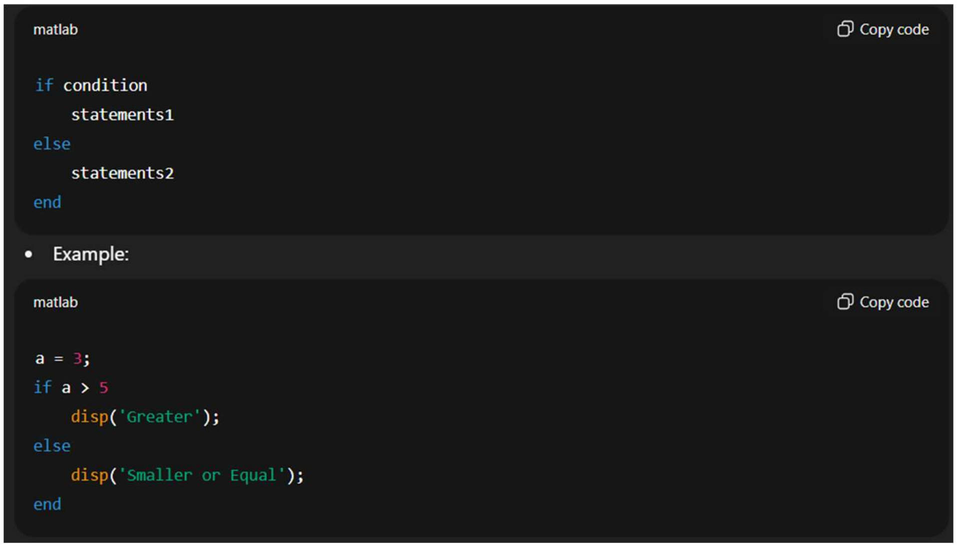
- Executes statements **only if condition is true**.
- Example:



```matlab
a = 10;
if a > 5
    disp('a is greater than 5');
end
```

B. if-else Statement

- Executes **one block if true, another if false**.
- **Syntax:**



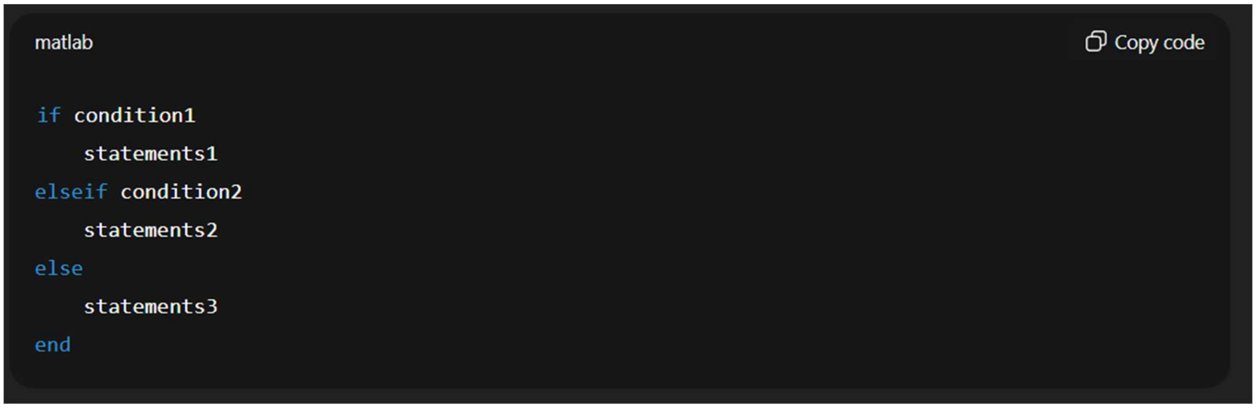
```matlab
if condition
    statements1
else
    statements2
end
```

- Example:

```matlab
a = 3;
if a > 5
    disp('Greater');
else
    disp('Smaller or Equal');
end
```

C. if-elseif-else Statement

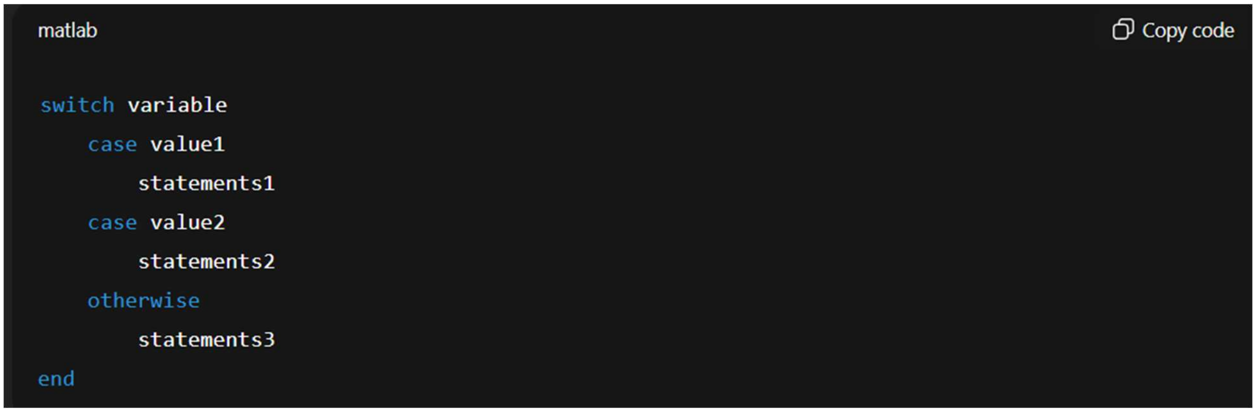
- Used for **multiple conditions**.
- Syntax:



```matlab
if condition1
    statements1
elseif condition2
    statements2
else
    statements3
end
```

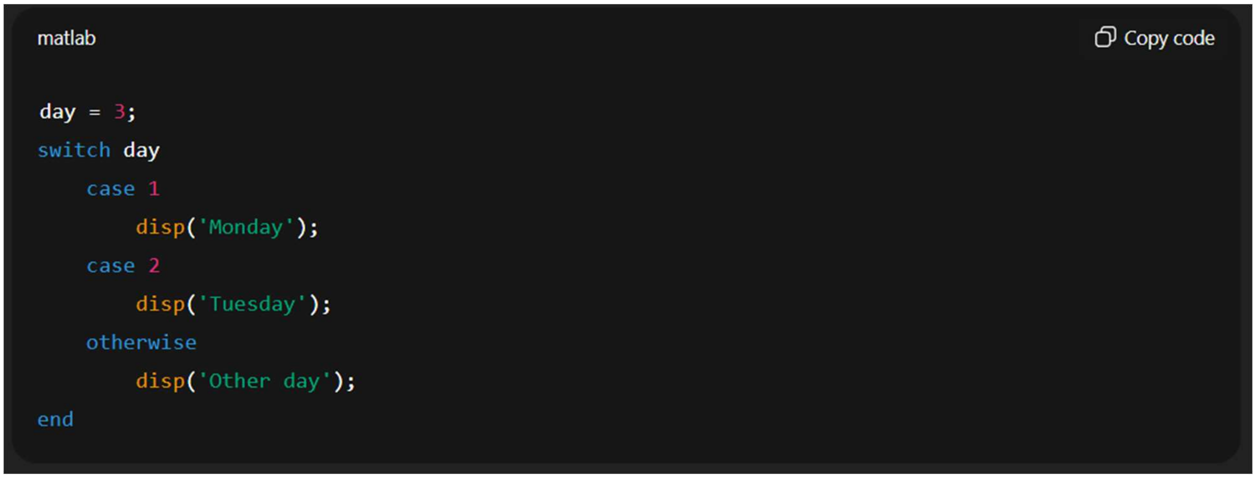
D. switch-case Statement

- Used to **choose one among many options**.
- Syntax:



```matlab
switch variable
    case value1
        statements1
    case value2
        statements2
    otherwise
        statements3
end
```

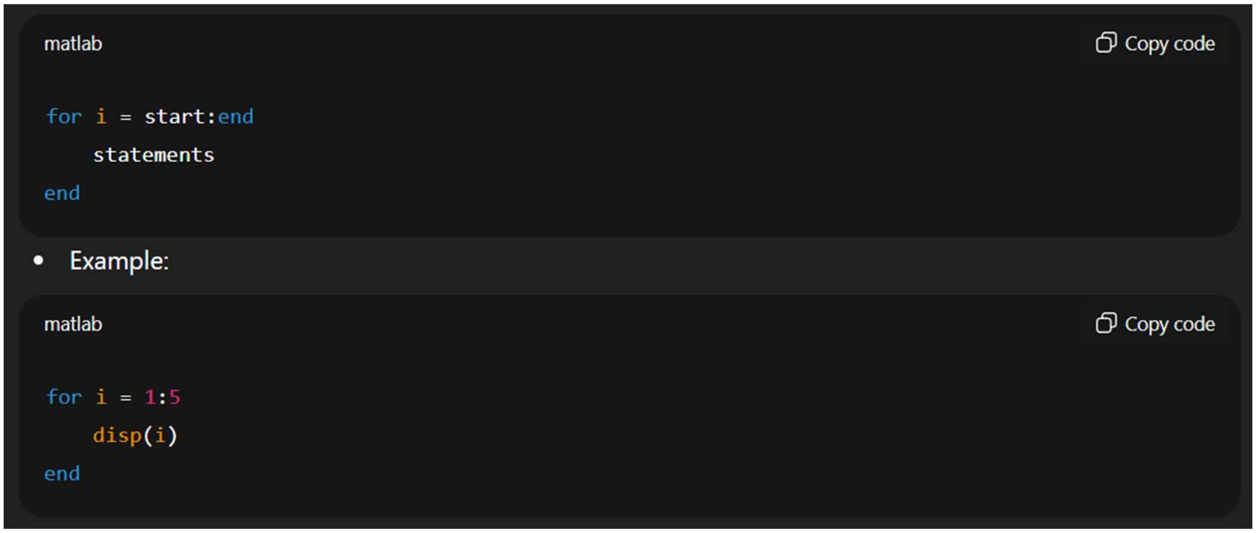
Example:



```matlab
day = 3;
switch day
    case 1
        disp('Monday');
    case 2
        disp('Tuesday');
    otherwise
        disp('Other day');
end
```

E. for Loop

- Repeats a block **fixed number of times**.
- Syntax:



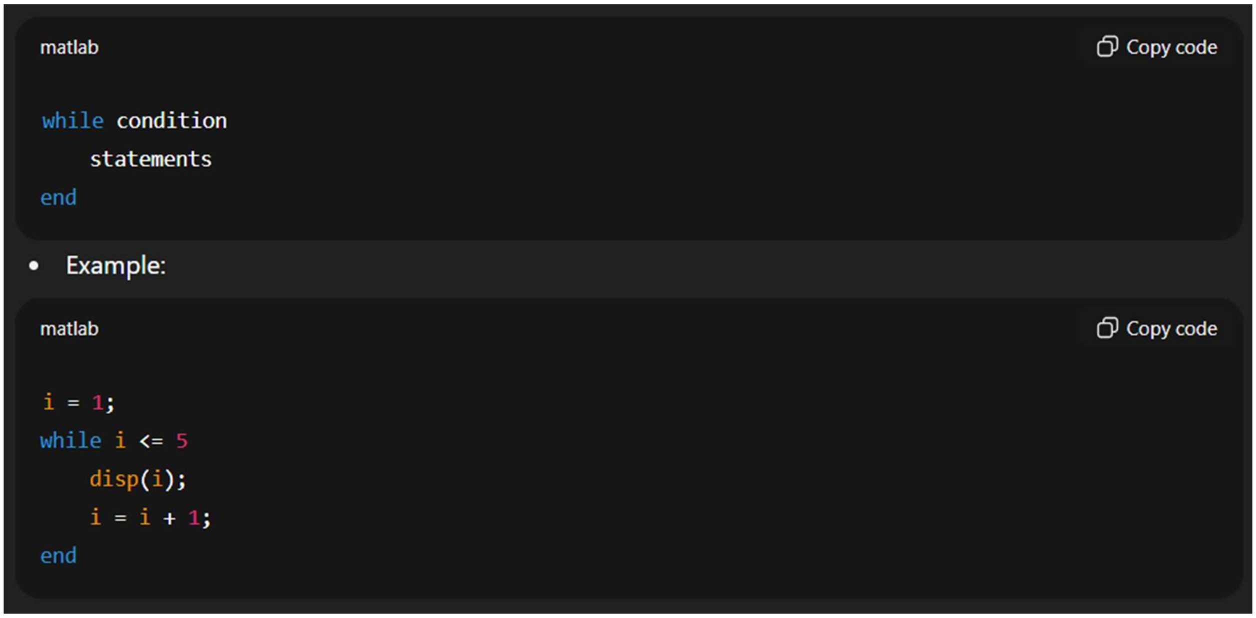
```matlab
for i = start:end
    statements
end
```

- Example:

```matlab
for i = 1:5
    disp(i)
end
```

F. while Loop

- Repeats **until a condition becomes false**.
- Syntax:



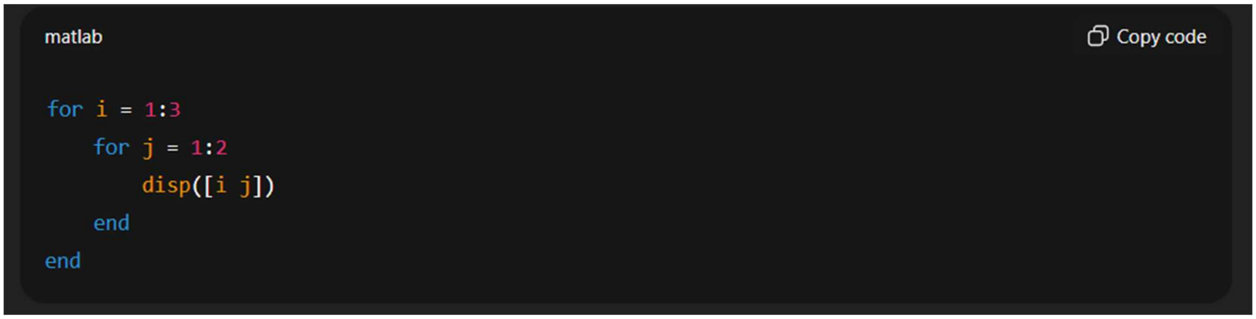
```matlab
while condition
    statements
end
```

- Example:

```matlab
i = 1;
while i <= 5
    disp(i);
    i = i + 1;
end
```

G. Nested Loops

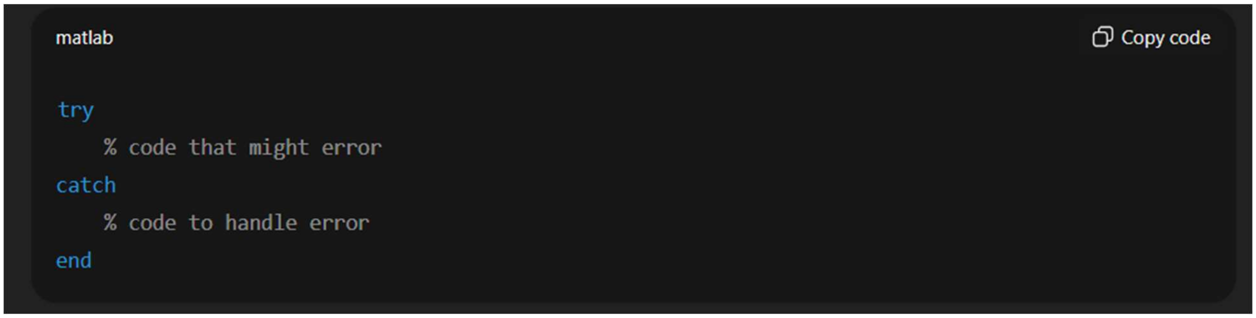
- A loop inside another loop.



```matlab
for i = 1:3
    for j = 1:2
        disp([i j])
    end
end
```

H. try-catch Block

- Handles **errors gracefully**.
- Syntax:



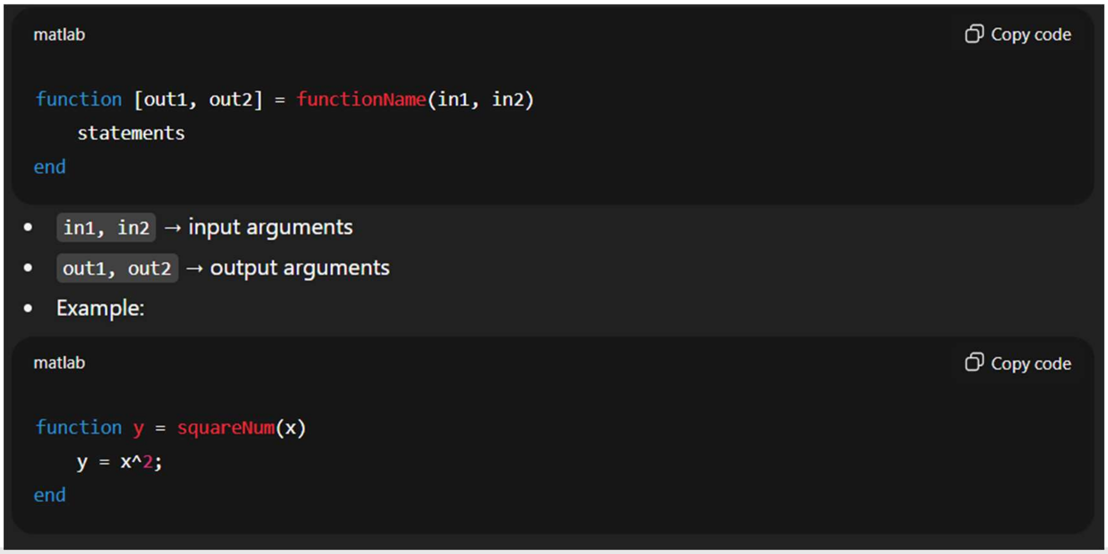
```matlab
try
    % code that might error
catch
    % code to handle error
end
```

2. Functions in MATLAB

Functions let you **group code into reusable blocks**.

A. Function Construction

- **Syntax:**



```matlab
function [out1, out2] = functionName(in1, in2)
    statements
end
```

- `in1, in2` → input arguments
- `out1, out2` → output arguments
- Example:

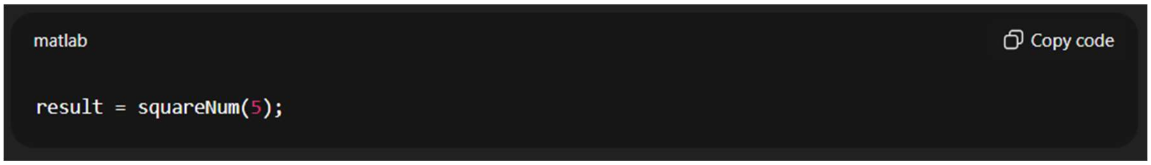
```matlab
function y = squareNum(x)
    y = x^2;
end
```

B. Rules for Constructing Functions

1. Function **name must match file name**.
2. First line contains function keyword.
3. Input/output arguments **optional**.

C. Executing Functions

- Call function from **command line or script**:



```matlab
result = squareNum(5);
```

D. Sub-functions

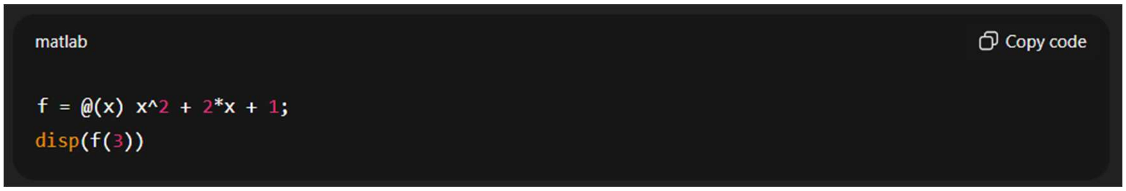
- Functions **inside a file** after the main function.
- Only **accessible within that file**.

E. Nested Functions

- Functions defined **inside another function**.
- Can **access parent function variables**.

F. Function Handles & Anonymous Functions

- **Anonymous function:** single-line function.



```matlab
f = @(x) x^2 + 2*x + 1;
disp(f(3))
```

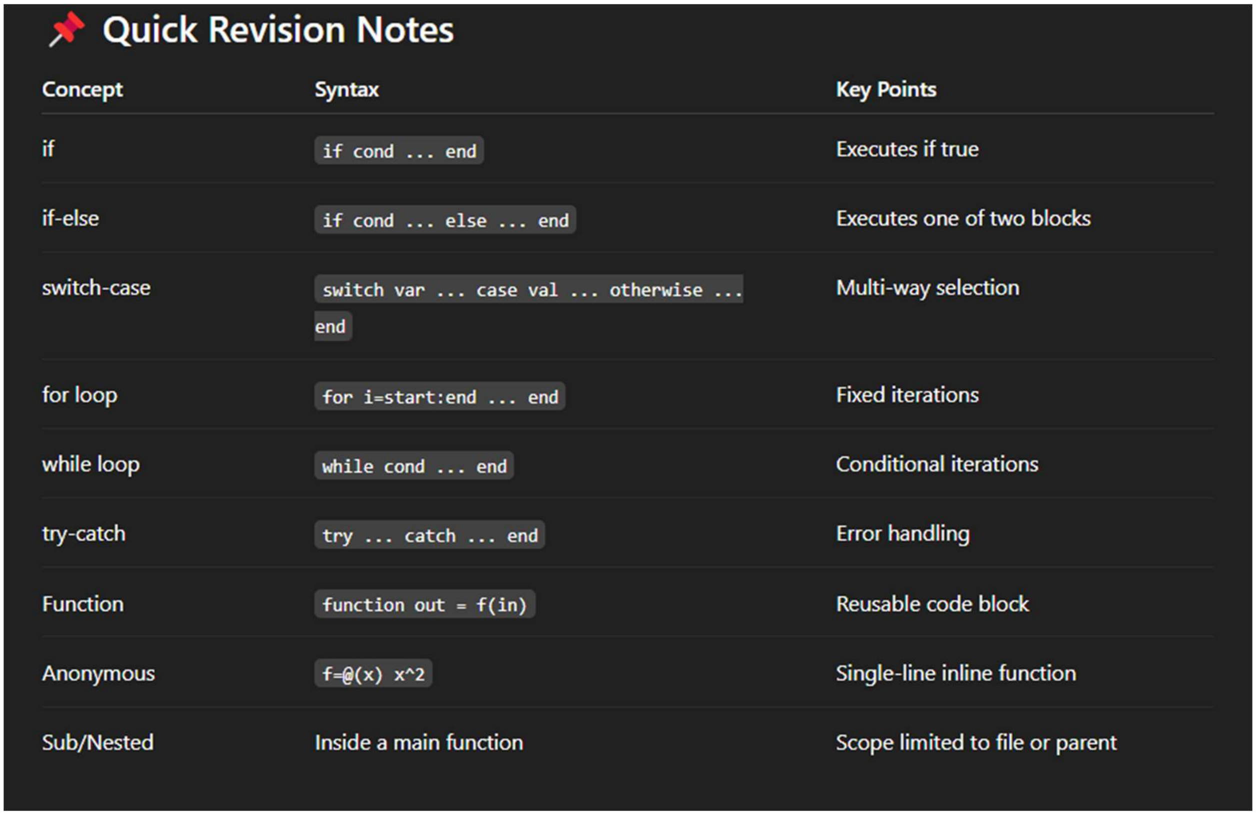
- **Function handle:** allows passing functions as variables.

G. Command Line Functions

- MATLAB provides **predefined functions** you can use directly.
- Examples: sum(), mean(), max(), min(), sqrt(), abs()

H. Using a Function File

- Save function in .m file.
- Call it from **scripts or command window**.



📌 Quick Revision Notes

Concept	Syntax	Key Points
if	`if cond ... end`	Executes if true
if-else	`if cond ... else ... end`	Executes one of two blocks
switch-case	`switch var ... case val ... otherwise ... end`	Multi-way selection
for loop	`for i=start:end ... end`	Fixed iterations
while loop	`while cond ... end`	Conditional iterations
try-catch	`try ... catch ... end`	Error handling
Function	`function out = f(in)`	Reusable code block
Anonymous	`f=@(x) x^2`	Single-line inline function
Sub/Nested	Inside a main function	Scope limited to file or parent

☑ Expected Exam Questions (Unit III):

1. Explain different **control flow statements** in MATLAB.
2. Differentiate for and while loops with examples.
3. Explain try-catch block with example.
4. Write a function in MATLAB to calculate factorial.
5. Explain **anonymous and nested functions**.

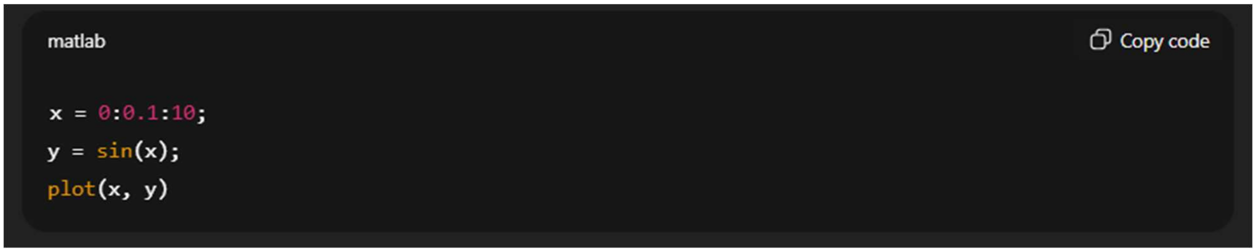
UNIT IV – MATLAB Graphics

MATLAB provides powerful tools to **visualize data** in 2D and 3D, which helps in analysis and presentation.

1. Two-Dimensional (2D) Graphics

A. Basic Plot

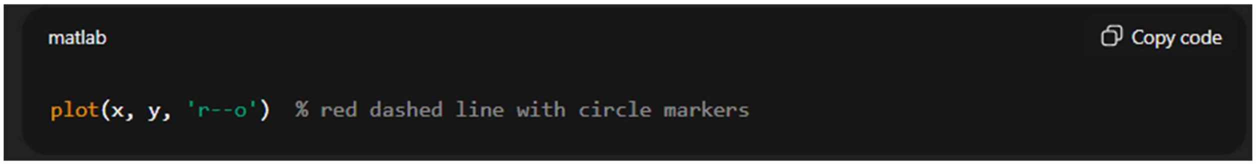
- **Function:** plot(x, y)
- Plots **y vs x** in a 2D graph.
- Example:



```matlab
x = 0:0.1:10;
y = sin(x);
plot(x, y)
```

B. Style Options

- Customize **color, line style, and marker**.
- Syntax:



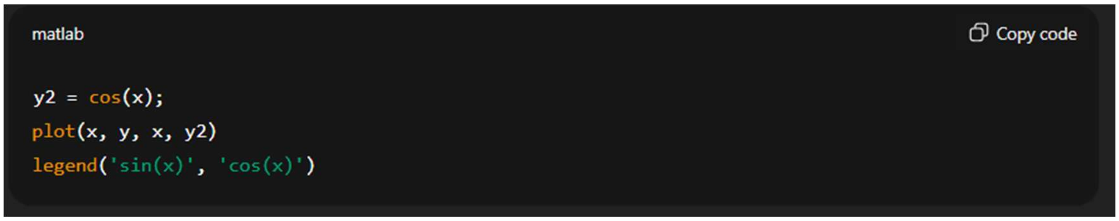
```matlab
plot(x, y, 'r--o')  % red dashed line with circle markers
```

Common codes:

- Colors: 'r'=red, 'g'=green, 'b'=blue, 'k'=black
- Line styles: '-'=solid, '--'=dashed, ':'=dotted
- Markers: 'o', '+', '*', 's'=square, '^'=triangle

C. Multiple Plots

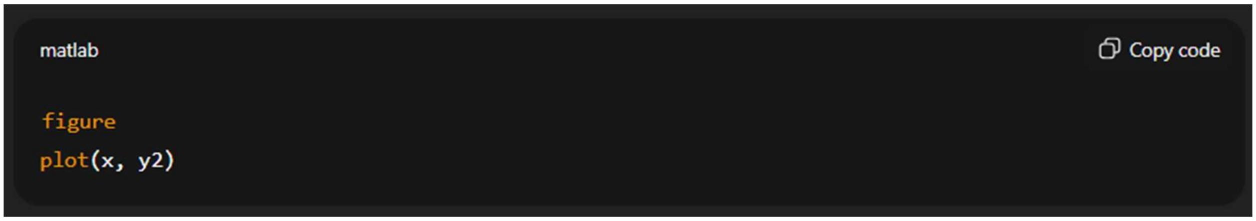
- Plot multiple datasets in **one figure**.



```matlab
y2 = cos(x);
plot(x, y, x, y2)
legend('sin(x)', 'cos(x)')
```

D. Multiple Figures

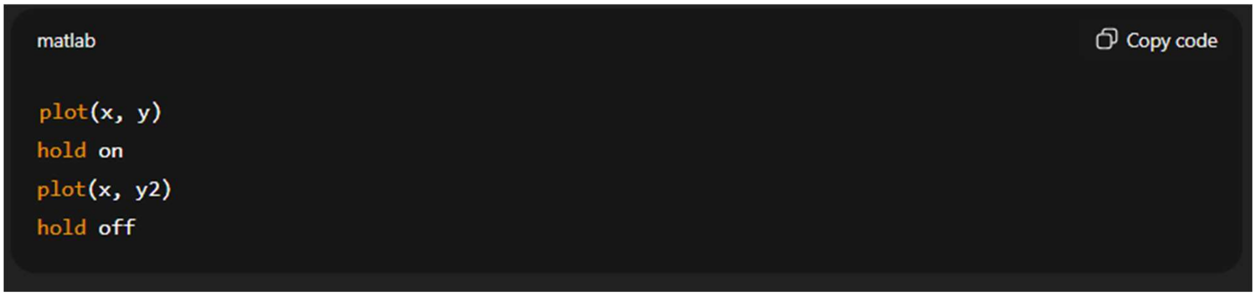
- Use figure to create **new plotting windows**.



```matlab
figure
plot(x, y2)
```

E. Overlay Plots

- Add plots **without replacing** existing plots using hold on.



```matlab
plot(x, y)
hold on
plot(x, y2)
hold off
```

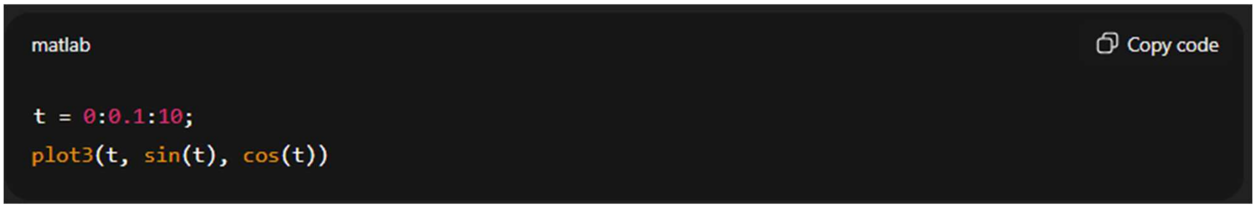
F. Specialized 2-D Plots

- **Bar chart:** bar(x, y)
- **Stem plot:** stem(x, y)
- **Pie chart:** pie(values)
- **Area plot:** area(x, y)

2. Three-Dimensional (3D) Graphics

A. Line Plots

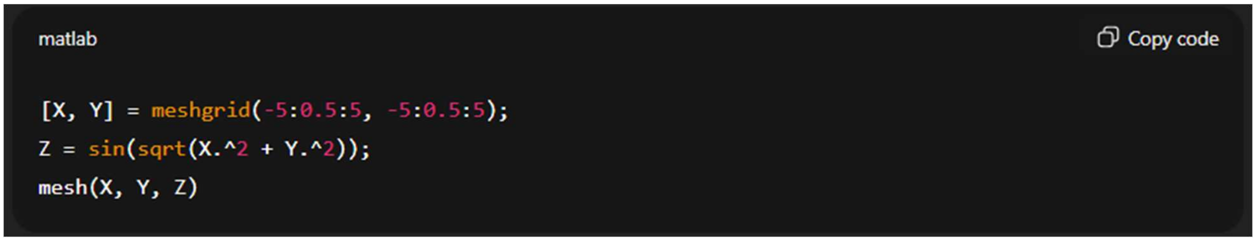
- 3D line plot using plot3(x, y, z)



```matlab
t = 0:0.1:10;
plot3(t, sin(t), cos(t))
```

B. Mesh Plots

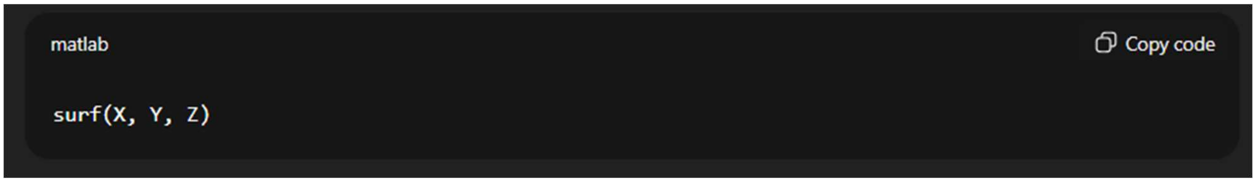
- Displays a **wireframe surface**.



```matlab
[X, Y] = meshgrid(-5:0.5:5, -5:0.5:5);
Z = sin(sqrt(X.^2 + Y.^2));
mesh(X, Y, Z)
```

C. Surface Plots

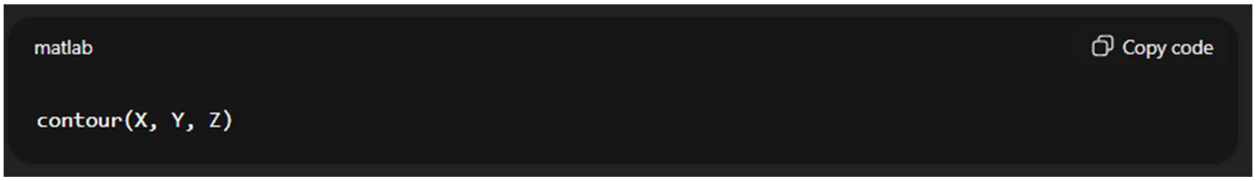
- Colored 3D surface. Syntax: surf(X, Y, Z)



```matlab
surf(X, Y, Z)
```

D. Contour Plots

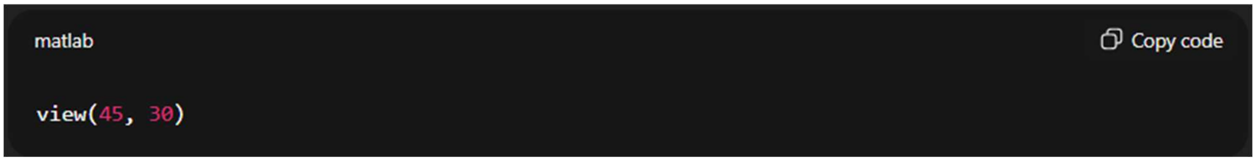
- 2D representation of 3D data using **contour lines**.
- Syntax: contour(X, Y, Z)



```matlab
contour(X, Y, Z)
```

E. Changing Viewpoints

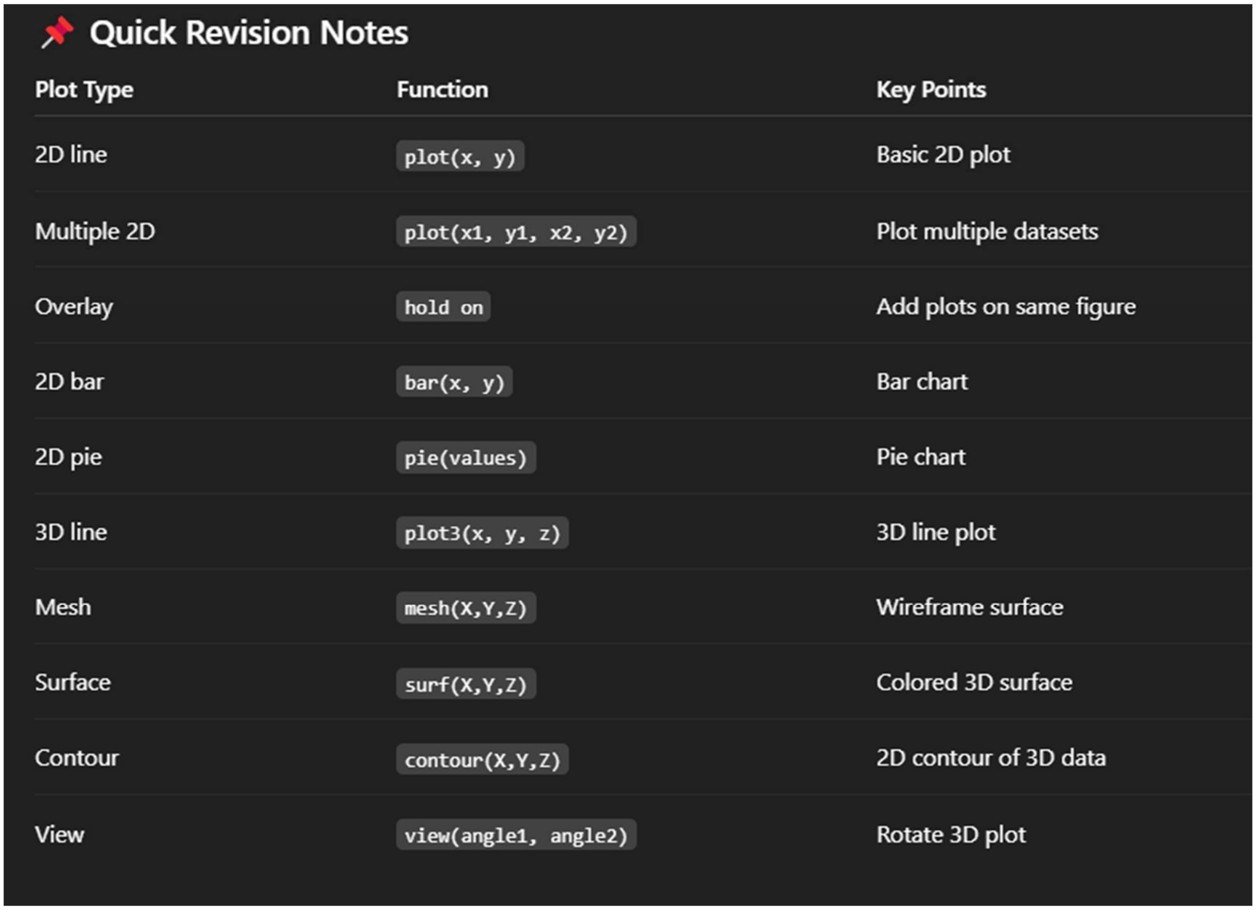
- Use view(angle1, angle2) to rotate the 3D view.



```matlab
view(45, 30)
```

F. Specialized 3-D Plots

- **3D bar plot:** bar3(Y)
- **3D stem plot:** stem3(X, Y, Z)
- **3D mesh with color:** meshc(X, Y, Z)



📌 Quick Revision Notes

Plot Type	Function	Key Points
2D line	`plot(x, y)`	Basic 2D plot
Multiple 2D	`plot(x1, y1, x2, y2)`	Plot multiple datasets
Overlay	`hold on`	Add plots on same figure
2D bar	`bar(x, y)`	Bar chart
2D pie	`pie(values)`	Pie chart
3D line	`plot3(x, y, z)`	3D line plot
Mesh	`mesh(X,Y,Z)`	Wireframe surface
Surface	`surf(X,Y,Z)`	Colored 3D surface
Contour	`contour(X,Y,Z)`	2D contour of 3D data
View	`view(angle1, angle2)`	Rotate 3D plot

☑ **Expected Exam Questions (Unit IV):**

1. Explain basic 2D and 3D plotting in MATLAB.
2. Differentiate between mesh and surf plots with examples.
3. How to overlay multiple plots in MATLAB?
4. Write MATLAB code for a contour plot of a 3D function.
5. Explain the use of hold on and figure.

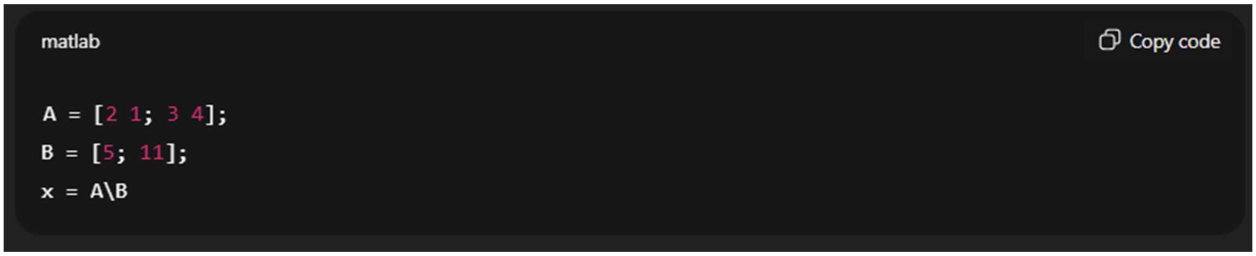
UNIT V – MATLAB Applications

MATLAB is widely used for **scientific computing, engineering analysis, and numerical problem solving**. This unit focuses on its **applications in linear algebra, polynomials, data analysis, calculus, and differential equations**.

1. Linear Algebra

A. Solving Linear Systems

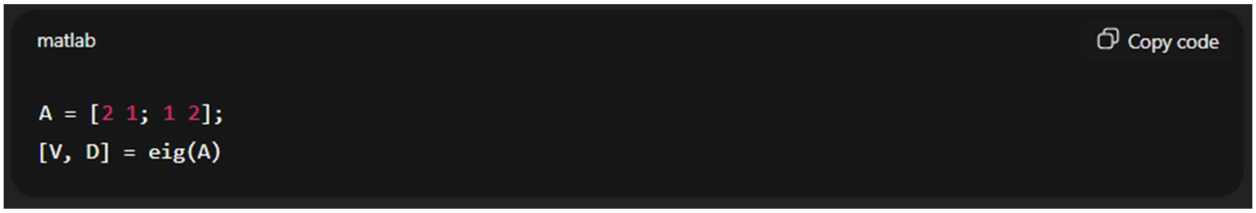
- Solve equations of the form **Ax = B**
- Function: x = A\B or x = linsolve(A, B)
- Example:



```matlab
A = [2 1; 3 4];
B = [5; 11];
x = A\B
```

B. Eigenvalues and Eigenvectors

- **Eigenvalues** (eig(A))
- **Eigenvectors** ([V, D] = eig(A))
- Example:

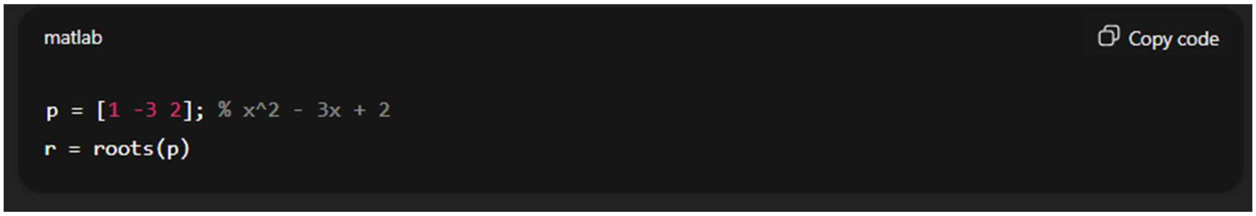


```matlab
A = [2 1; 1 2];
[V, D] = eig(A)
```

2. Polynomials

A. Roots

- Find roots of a polynomial p(x) using roots(p)
- Example:



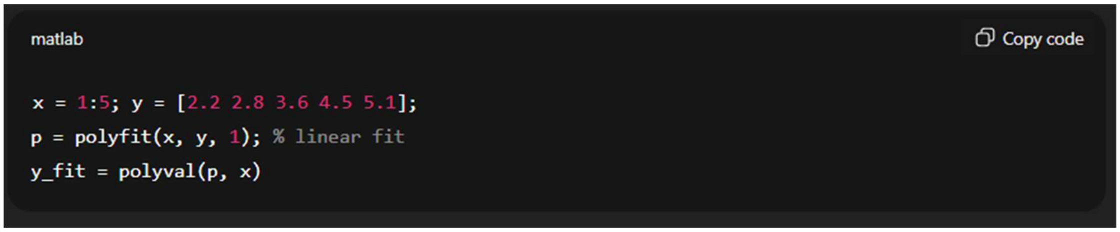
```matlab
p = [1 -3 2]; % x^2 - 3x + 2
r = roots(p)
```

B. Addition, Multiplication, Division

- Polynomials represented as vectors: [coefficients]
- conv(p1, p2) → multiply
- deconv(p1, p2) → divide
- polyadd → addition (custom function or use + on vectors)

C. Curve Fitting

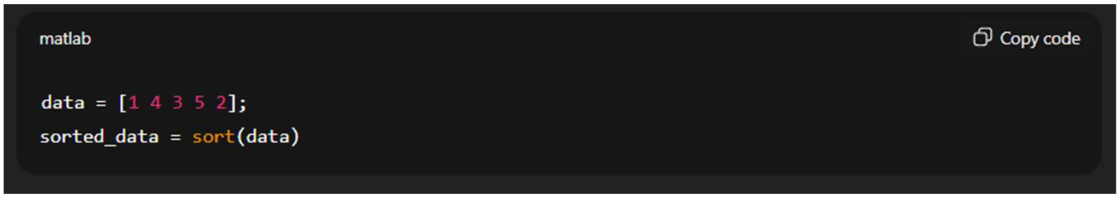
- Fit data points with polynomial using polyfit(x, y, n)
- polyval(p, x) → evaluate polynomial
- Example:



```matlab
x = 1:5; y = [2.2 2.8 3.6 4.5 5.1];
p = polyfit(x, y, 1); % linear fit
y_fit = polyval(p, x)
```

3. Data Analysis

- **Statistical functions:** mean(x), median(x), std(x), sum(x)
- **Sorting & indexing:** sort(x), max(x), min(x)
- Example:



```matlab
data = [1 4 3 5 2];
sorted_data = sort(data)
```

4. Calculus

A. Differentiation

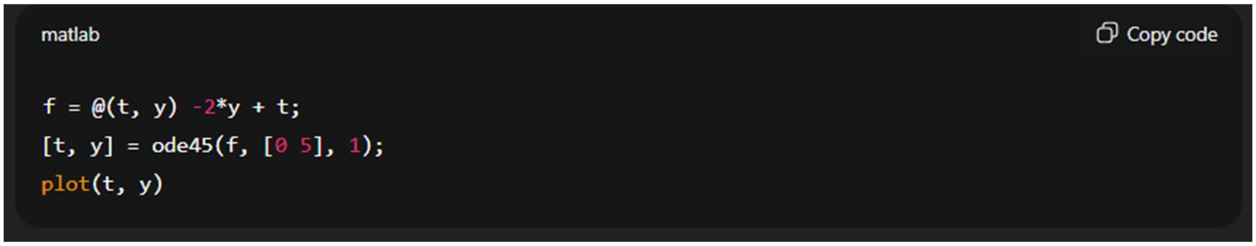
- Numerical derivative using diff(y)./diff(x)
- Symbolic differentiation using syms x; diff(f, x)

B. Integration

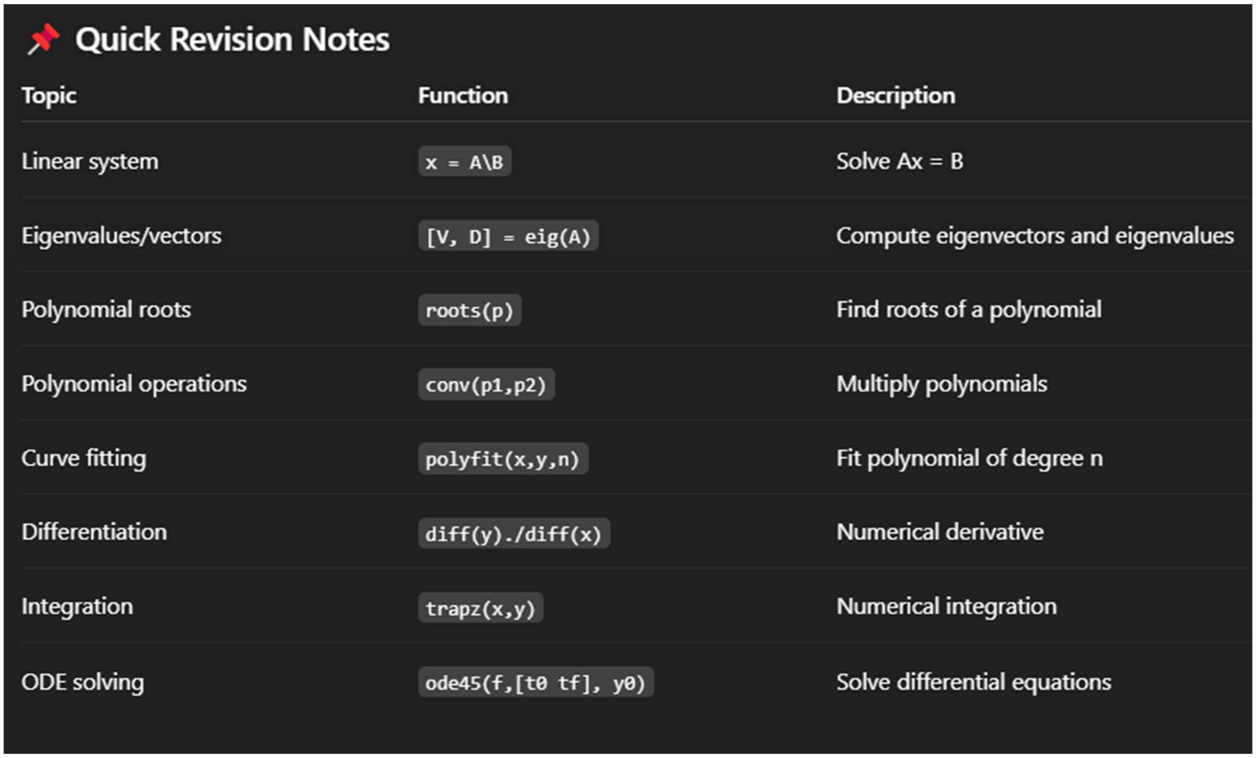
- Numerical integration: trapz(x, y)
- Symbolic integration: int(f, x)

5. Differential Equations

- **Ordinary Differential Equations (ODEs)** solved using ode45, ode23
- Example:



```matlab
f = @(t, y) -2*y + t;
[t, y] = ode45(f, [0 5], 1);
plot(t, y)
```



📌 Quick Revision Notes

Topic	Function	Description
Linear system	`x = A\B`	Solve Ax = B
Eigenvalues/vectors	`[V, D] = eig(A)`	Compute eigenvectors and eigenvalues
Polynomial roots	`roots(p)`	Find roots of a polynomial
Polynomial operations	`conv(p1,p2)`	Multiply polynomials
Curve fitting	`polyfit(x,y,n)`	Fit polynomial of degree n
Differentiation	`diff(y)./diff(x)`	Numerical derivative
Integration	`trapz(x,y)`	Numerical integration
ODE solving	`ode45(f,[t0 tf], y0)`	Solve differential equations

☑ Expected Exam Questions (Unit V):

1. Solve a linear system using MATLAB.
2. Find eigenvalues and eigenvectors of a matrix.
3. Find the roots of a polynomial and perform addition/multiplication.
4. Fit a polynomial to a set of data points.
5. Solve a first-order differential equation using MATLAB.
6. Compute the derivative and integral of a function numerically.