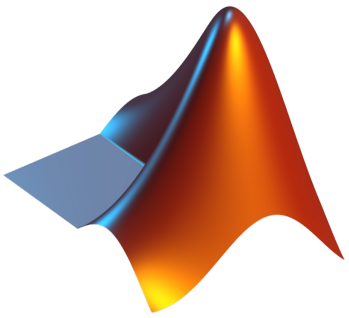
جامعة المستقبل
AL MUSTAQBAL UNIVERSITY
كلية العلوم



2nd class

2024- 2025

Numerical Analysis

Practical

MATLAB

Lecture 1

Asst. Lect. Mohammed Jabbar

Mohammed.Jabbar.Obaid@uomus.edu.iq

الرياضيات المتقدمة: المرحلة الثانية

مادة التحليل العددي عملي

ماتلاب

المحاضرة الاولى

استاذ المادة: م.م محمد جبار

Cybersecurity Department

قسم الأمن السيبراني

Contents

1 Introduction to MATLAB for Numerical Computations

MATLAB (short for MATrix LABoratory) is a high-level programming environment widely used for numerical computations, data analysis, algorithm development, and visualization. It is especially powerful for working with matrices and arrays, making it ideal for scientific and engineering applications.

1.1 Matlab Programming

A computer program is a sequence of instructions in a given language that achieves a specific task.

MATLAB programming is centered around matrix and array manipulations, offering an extensive collection of built-in functions and toolboxes for numerical computation, algorithm development, data analysis, and visualization.

1.2 Understanding the MATLAB Environment

MATLAB development IDE can be launched from the icon created on the desktop. Understanding the MATLAB environment is essential for efficiently using its features. Below are the key components of the MATLAB environment.

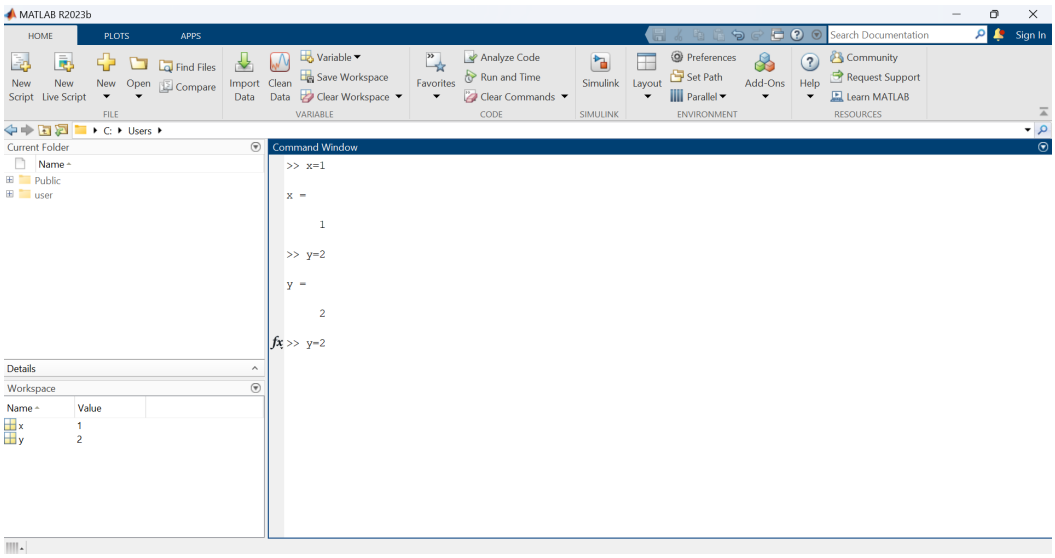


Figure 1: The main working widow in Matlab

1.2.1 MATLAB Desktop Components

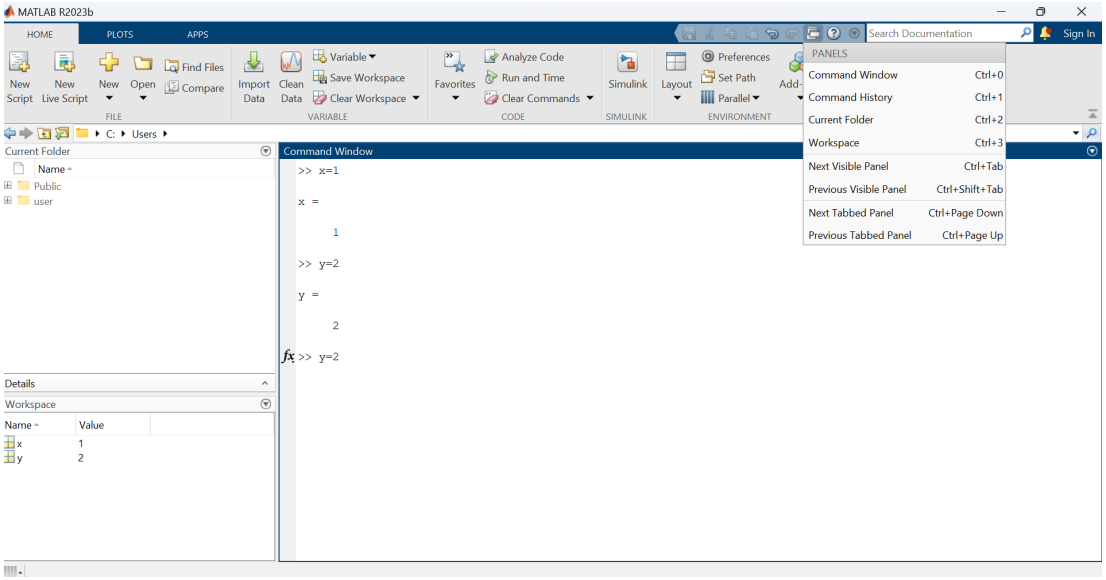


Figure 2: MATLAB Desktop Components: Switch Window

1. **Command Window:**

   - The Command Window is where you can type and execute MATLAB commands interactively. For example, entering a calculation like $3+4$ will immediately show the result.
   - You can also run scripts and functions from the Command Window by typing their names.

Figure 3: Command Window

2. **Workspace:**

   - The Workspace shows all the variables currently in memory. You can view their values and details like size and type.
   - You can double-click a variable in the Workspace to open the Variable Editor and inspect or modify its contents.

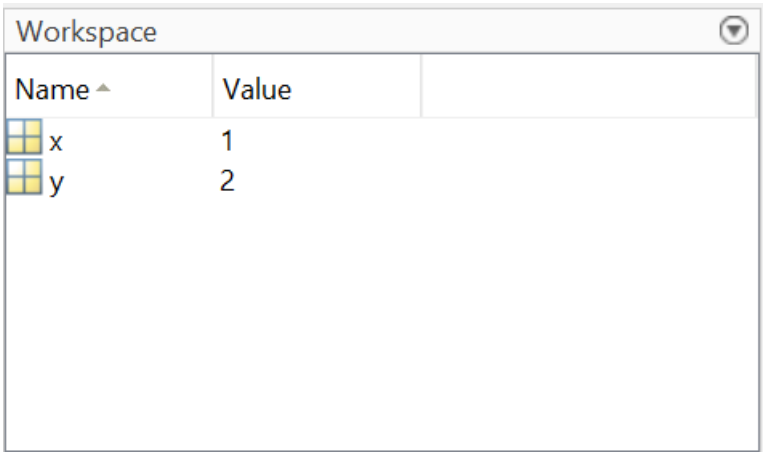


Figure 4: Workspace

3. **Command History:** This window stores all the commands you have previously entered in the Command Window. You can easily re-execute commands from here by double-clicking them.

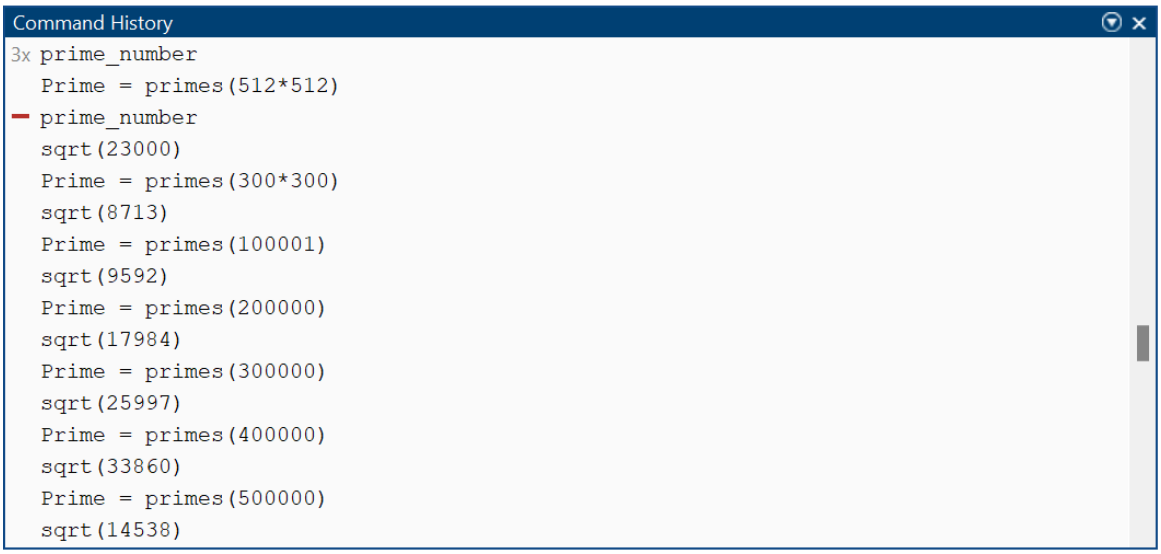


Figure 5: Command History

4. **Current Folder:** The Current Folder window displays the files and folders in your working directory. This is the default location where MATLAB will save or look for scripts, functions, and data files.

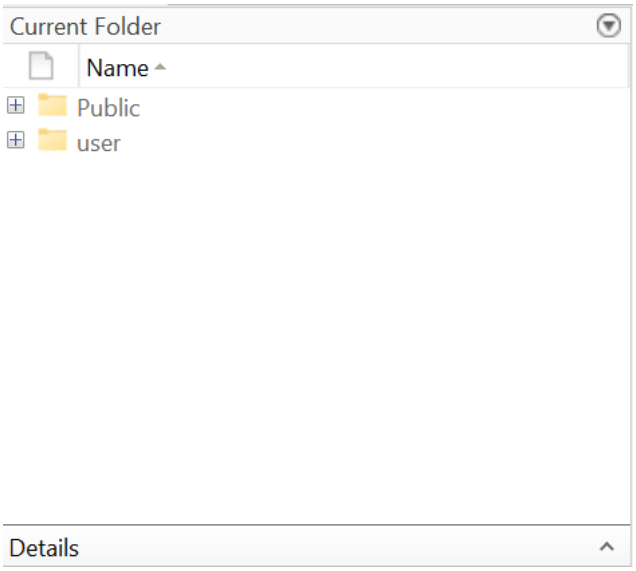


Figure 6: Current Folder

2 Basic Syntax

MATLAB has a simple and intuitive syntax designed for numerical computing and matrix manipulations. Here's an overview of the basic syntax rules and commands to get started with MATLAB.

2.1 Hands on Practice

Type a valid expression, for example,

```
5+5
%And press ENTER. When you click the Execute button, or type Ctrl+E, MATLAB ...
executes it immediately and the result returned is:
ans =
    10
```

Let us take up few more examples:

```
3^2 %3 raised to the power of 2
%Press ENTER.
ans =
     9
```

Another example,

```
sin(pi/2) %sine of angle 90 (π/2)
%Press ENTER.
ans =
     1
```

Another example,

```
7/0 %Divide by zero
%Press ENTER.
ans =
     Inf %Warning: division by zero
```

Another example,

```
732*20.3
%Press ENTER.
ans =
     1.4860e+04
```

MATLAB provides some special expressions for some mathematical symbols, like π, ∞, $\sqrt{a}$ etc. Nan stands for 'not a number'.

Function	Description	Syntax
sin(x)	Sine of x	$\sin(x)$
cos(x)	Cosine of x	$\cos(x)$
tan(x)	Tangent of x	$\tan(x)$
asin(x)	Inverse sine (arcsin) of x	$\sin^{-1}(x)$
acos(x)	Inverse cosine (arccos) of x	$\cos^{-1}(x)$
atan(x)	Inverse tangent (arctan) of x	$\tan^{-1}(x)$
exp(x)	Exponential function of x	e^x
log(x)	Natural logarithm of x	$\log(x)$
log10(x)	Base-10 logarithm of x	$\log_{10}(x)$
sqrt(x)	Square root of x	$\sqrt{x}$
abs(x)	Absolute value of x	$\|x\|$
floor(x)	Round towards negative infinity	$floor(x)$
ceil(x)	Round towards positive infinity	$ceil(x)$
round(x)	Round to nearest integer	$round(x)$
sinh(x)	Hyperbolic sine of x	$\sinh(x)$
cosh(x)	Hyperbolic cosine of x	$\cosh(x)$
tanh(x)	Hyperbolic tangent of x	$\tanh(x)$
asinh(x)	Inverse hyperbolic sine of x	$sinh^{-1}(x)$
acosh(x)	Inverse hyperbolic cosine of x	$\cosh^{-1}(x)$
atanh(x)	Inverse hyperbolic tangent of x	$\tanh^{-1}(x)$

Table 1: Table of Special Functions in MATLAB

Operator	Description	Example
+	Addition	$a + b$
-	Subtraction	$a - b$
*	Multiplication	$a * b$
/	Right Division	$\frac{a}{b}$
.+	Element-wise Addition	$A. + B$
.-	Element-wise Subtraction	$A. - B$
.*	Element-wise Multiplication	$A. * B$
./	Element-wise Right Division	$A./B$

Table 2: Table of Arithmetic Operators in MATLAB

2.2 Variables

In MATLAB environment, every variable is an array or matrix. You can assign variables in a simple way.

Variables are used to store data for manipulation and analysis. Here's a brief overview of how to create and use variables in MATLAB:

2.2.1 Creating Variables

You can create variables by simply assigning a value to a name. Variable names must start with a letter, followed by letters, digits, or underscores.

```
    x = 5 %Numeric variable
%Press ENTER
 x =

        5

 y = sin(67) %Numeric variable

%Press ENTER
 y =

        -0.8555

 name = 'MATLAB' %String variable

%Press ENTER
 name =

        'MATLAB'

 A = [1, 2, 3; 4, 5, 6] %Matrix variable

%Press ENTER
 A =

        1   2   3

        4   5   6

%Accessing Variables: You can access the value of a variable by simply typing its ...
name.
y
%Press ENTER
y =

        -0.8555
```

2.2.2 Variable Types

MATLAB supports various data types, including:

- **Numeric:** Scalars, vectors, and matrices.
- **Strings:** Character arrays and string arrays.
- **Logical:** Boolean values (true/false).
- **Cells:** Cell arrays for heterogeneous data.
- **Structures:** Data structures for complex data types.

2.2.3 Clearing Variables

You can remove variables from the workspace using the clear command.

```
    clear x
%Press ENTER
 %Removes variable x
 clear all
%Press ENTER
 %Clears all variables
```

Remark. The (clc) command in MATLAB is used to clear the Command Window. It does not delete variables from memory, but it removes all the text and results from the Command Window, giving you a fresh, clean workspace for the next set of outputs.

2.2.4 Checking Variable Information

Use the (whos) command to see the variables in the workspace along with their sizes and types.

```
whos %See all variables
```

Remark. In MATLAB, the semicolon (;) has two main purposes:

1. **Suppress Output:** When you use a semicolon at the end of a statement or command, MATLAB will execute the command but suppress the output in the Command Window. For example:

```
x = 5; %The value is assigned, but no output is displayed.
%Without the semicolon:

x =

     5
```

2. **Separate Commands on the Same Line:** You can use a semicolon to separate multiple statements or commands on the same line. For example:

```
a = 3; b = 7; c = a + b;
%This executes all three commands in sequence, a, b, and c.
```

3 Matrices and Vectors

In MATLAB, matrices and vectors are fundamental data structures for numerical computing.

3.1 Creating Vectors

```
%Row Vector:
R= [1, 2, 3, 4, 5]

R =

     1   2   3   4   5
```

```
%Column Vector:
C = [1; 2; 3; 4; 5]

C =

     1
     2
     3
     4
     5
```

3.1.1 linspace function

In MATLAB, the linspace function generates a row vector of linearly spaced elements between two specified limits. It is especially useful for creating vectors when you know the number of points you want between a start and an end value, rather than specifying the step size (as you do with the colon operator :).

$$y = \text{linspace}(a,\ b,\ n)$$

- a: Start value.
- b: End value.
- n: Number of points to generate

Default Number of Points (100 points if n is omitted):

$$y = \text{linspace}(a,\ b)$$

```
y = linspace(1,50,5)

y =

    1.00   13.25   25.50   37.75   50.00
```

3.1.2 Colon Operator

Creates a vector from a to b with a step of s

$$v = a{:}s{:}b$$

Creates a vector from 1 to 10 with a step of 2

$$v = 1{:}2{:}10 = 1\ 3\ 5\ 7\ 9$$

3.2 Creating Matrices

```
%Manually:
A= [1, 2, 3; 4, 5, 6; 7, 8, 9]
```

A =

1	2	3
4	5	6
7	8	9

```
%Zero Matrix:
B = zeros(3, 3) %3x3 matrix of zeros
```

B =

0	0	0
0	0	0
0	0	0

```
%Identity Matrix:
E = eye(3) %3x3 identity matrix
```

E =

1	0	0
0	1	0
0	0	1

%Random Matrix:

R= rand(3, 3); %3x3 matrix of random values between 0 and 1

3.3 Matrix Functions

A= [1, 2, 3; 4, 5, 6; 7, 8, 9]

A =

$$\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

%Determinant:

DA = det(A)

DA =

6.6613e-16

%Inverse:

IA = inv(A)

IA =

$$\begin{matrix} -0.4504 & 0.9007 & -0.4504 \\ 0.9007 & -1.8014 & 0.9007 \\ -0.4504 & 0.9007 & -0.4504 \end{matrix}$$

4 Functions Applied on Matrices

1. Sum of Elements

Function:

$$\text{sum}(A) \quad \text{(Sums each column of matrix } A\text{)}$$

Example:

$$A = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix} \quad \Rightarrow \quad \text{sum}(A) = \begin{matrix} 12 & 15 & 18 \end{matrix}$$

Function:

$$\text{sum}(A, 2) \quad \text{(Sums each row of matrix } A\text{)}$$

Example:

$$\text{sum}(A, 2) = \begin{matrix} 6 \\ 15 \\ 24 \end{matrix}$$

2. Product of Elements

Function:

$$\text{prod}(A) \quad \text{(Product of elements in each column)}$$

Example:

$$\text{prod}(A) = \begin{matrix} 28 & 80 & 162 \end{matrix}$$

Function:

$$\text{prod}(A, 2) \quad \text{(Product of elements in each row)}$$

Example:

$$\text{prod}(A, 2) = \begin{matrix} 6 \\ 120 \\ 504 \end{matrix}$$

3. Mean

Function:

$$\text{mean}(A) \quad \text{(Mean of each column of matrix } A\text{)}$$

Example:

$$\text{mean}(A) = \begin{matrix} 4 & 5 & 6 \end{matrix}$$

Function:

$$\text{mean}(A, 2) \quad \text{(Mean of each row of matrix } A\text{)}$$

Example:

$$\text{mean}(A, 2) = \begin{matrix} 2 \\ 5 \\ 8 \end{matrix}$$

4. Transpose

Function:

$$A^T \quad \text{(Transpose of matrix } A\text{)}$$

Example:

$$A' = \begin{matrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{matrix}$$

5 Indexing of Matrix Values

In MATLAB, indexing is a way to access specific elements of a matrix. MATLAB uses 1-based indexing, meaning that the first element of any array is indexed by 1.

1. Accessing Individual Elements

To access an individual element in a matrix A, use the syntax $A(i, j)$, where i is the row index and j is the column index.

Example:

$$A = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

$A(2, 3) = 6$ (Accessing the element in the 2nd row and 3rd column)

2. Accessing Entire Rows or Columns

To access an entire row, use $A(i, :)$, and to access an entire column, use $A(:, j)$.

Example:

- Accessing the 2nd row:

$$A(2, :) = \begin{matrix} 4 & 5 & 6 \end{matrix}$$

- Accessing the 3rd column:

$$A(:, 3) = \begin{matrix} 3 \\ 6 \\ 9 \end{matrix}$$

3. Logical Indexing

Logical indexing allows you to access elements based on conditions.

Example: To access elements greater than 5:

$$C = A(A > 5) = \begin{matrix} 6 \\ 7 \\ 8 \\ 9 \end{matrix}$$

5. Modifying Elements

You can modify elements in a matrix using the same indexing methods.

Example: To set the element in the 1st row and 2nd column to 10:

$$A(1,2) = 999 \quad \Rightarrow \quad A = \begin{matrix} 1 & 999 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$$

6 M- Files and Syntax

6.1 M- Files

An M-file in MATLAB is a script or function that is saved with the .m extension. M-files are used to store sequences of commands that can be executed together, making them essential for automating tasks, performing calculations, and organizing code.

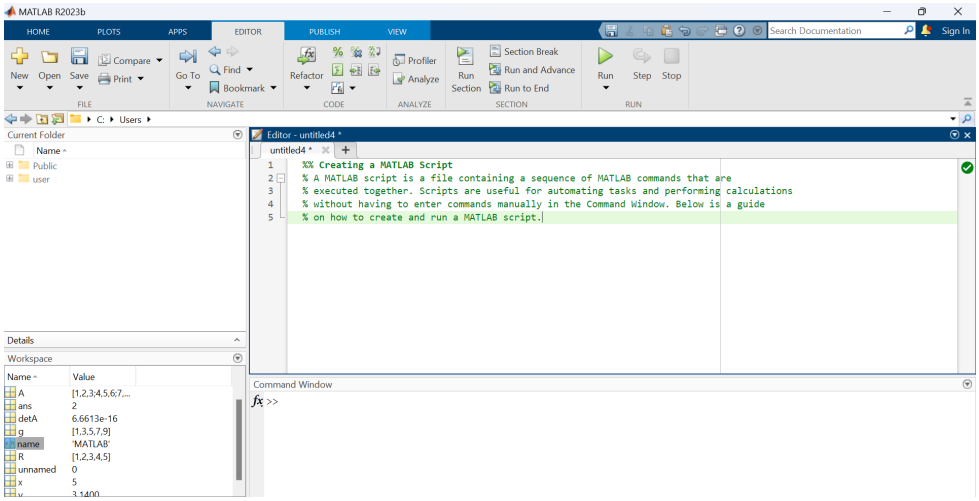


Figure 7: m- File

6.1.1 Creating a MATLAB Script

a. Open MATLAB: Launch MATLAB and navigate to the Home tab.

b. Create a New Script: Click on the New Script button in the toolbar or go to File > New > Script. This opens the MATLAB Editor.

c. Write Your Script: In the editor, write your MATLAB code. For example:

```
x = inspace (1,50,5)
    if x ≥ 50
        disp('Passed the exam');
end
```

6.2 Functions

A function is a separate file that accepts input arguments and returns outputs. Functions are also saved with a .m extension, but they start with a function keyword.

```
function z=abPlus(x,y);
z=x+y+10;
end
% abPlus(7,9)=26
```

7 Control Structures

Control structures in MATLAB are essential for managing the flow of execution in programs. They allow you to make decisions (conditional statements), repeat tasks (loops), and control the execution path based on specific conditions.

7.1 Conditional Statements

7.1.1 The if Statement

The if statement allows you to execute a block of code based on a condition.

```
x=15/4;
if x > 5
    disp('x is greater than 5');
end
```

7.1.2 The if-else Statement

The if-else statement allows you to execute one block of code if the condition is true and another if it is false.

```
x = 65/34;
if x > 5
    disp('x is greater than 5');
else
    disp('x is not greater than 5');
end
```

7.1.3 The if-elseif-else Statement

The if-elseif-else statement allows multiple conditions to be checked sequentially.

```
x = 65/54-65/45;
if x > 0
    disp('x is positive');
elseif x < 0
```

```
5       disp('x is negative');
6   else
7       disp('x is zero');
8   end
```

7.2 Loops

Loops allow you to repeat a block of code multiple times.

7.2.1 The for Loop

A for loop is used to iterate over a range of values.

```
1   for i = 1:5
2       disp(['Iteration: ', num2str(i)]);
3   end
4   % Output.
5   % Iteration: 1
6   % Iteration: 2
7   % Iteration: 3
8   % Iteration: 4
9   % Iteration: 5
```

7.2.2 The while Loop

A while loop continues to execute as long as a specified condition is true.

```
1   x = 1;
2   while x <= 5
3       disp(['x: ', num2str(x)]);
4       x = x + 1; % Increment x
5   end
```

```
% Output.
% x: 1
% x: 2
% x: 3
% x: 4
% x: 5
```

7.3 Break and Continue Statements

7.3.1 break Statement

Terminates the loop immediately.

```
for i = 1:10
    if i == 5
        break; % Exit the loop when i is 5
    end
```

7.3.2 continue Statement

Skips the rest of the loop iteration and proceeds to the next iteration.

```
for i = 1:10
    if mod(i, 2) == 0
        continue; % Skip even numbers
    end
    disp(['Odd i: ', num2str(i)]);
end
% Output.
% Odd i: 1
% Odd i: 3
% Odd i: 5
```

```
% Odd i: 7
% Odd i: 9
```

8 Plotting and Visualization

In MATLAB, plotting and visualization are essential tools for analyzing data and presenting results. MATLAB provides a wide range of functions to create different types of plots, graphs, and visualizations.

8.1 Basic Plotting

8.1.1 Simple 2D Line Plot

The plot function is used to create 2D line plots.

```
x = 0:0.1:2*pi; % Generate x values from 0 to 2
y = sin(x); % Compute the sine of each x value

plot(x, y) % Create the plot
xlabel('x'); % Label for x-axis
ylabel('sin(x)'); % Label for y-axis
title('Plot of sin(x)'); % Title of the plot
```

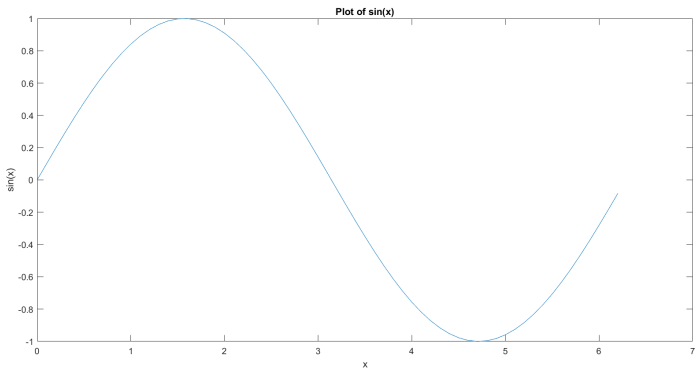


Figure 8: Plot of sin(x)

8.1.2 Multiple Lines on the Same Plot

You can plot multiple lines on the same graph by passing multiple sets of data to plot.

```
x = 0:0.1:2*pi;
y1 = sin(x);
y2 = cos(x);

plot(x, y1, '-r', x, y2, '--b'); % Plot both sin(x) and cos(x)
xlabel('x');
ylabel('y');
legend('sin(x)', 'cos(x)'); % Add a legend
title('Plot of sin(x) and cos(x)');
```

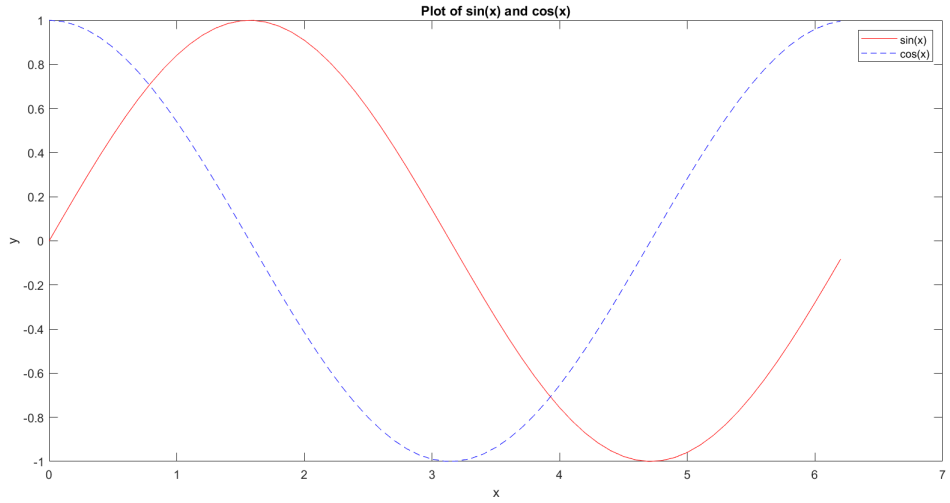


Figure 9: Plot of sin(x) and cos(x)

8.1.3 Subplots

The subplot function allows you to display multiple plots in the same figure.

```
x = 0:0.1:2*pi;
y1 = sin(x);
```

```
y2 = cos(x);

subplot(2,1,1); % 2 rows, 1 column, first subplot
plot(x, y1);
title('sin(x)');

subplot(2,1,2); % 2 rows, 1 column, second subplot
plot(x, y2);
title('cos(x)');
```

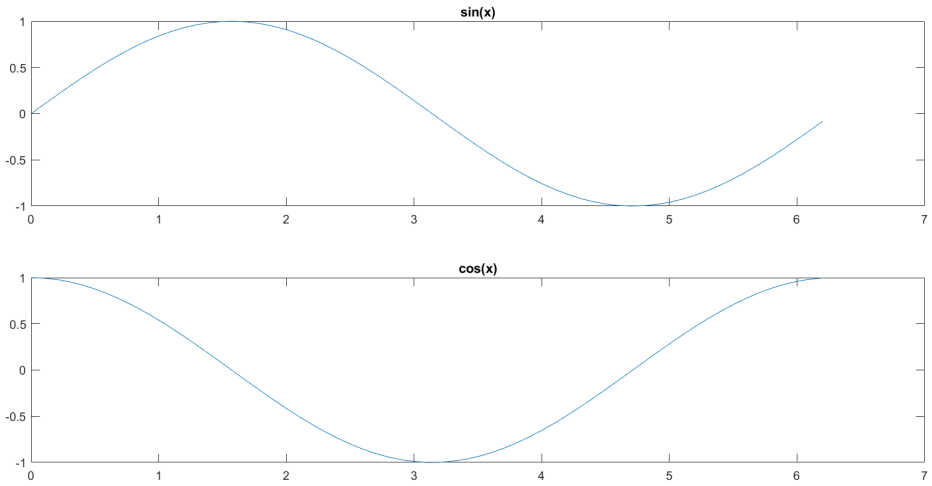


Figure 10: subplot: sin(x) and cos(x)

9 Homework

9.1 Homework of Basic Syntax

1. Calculate the equations below using command window in Matlab, where $x = 13$.

$$y = 2x + 7x^2 * 3$$
$$z = y - 0.67x + \frac{3}{7}$$
$$m = \frac{y}{5z} * 213$$
$$v = \sqrt{17} - \pi$$

2. $\sin(76) + \cos(65)$

9.2 Homework of Matrices and Vectors

1. Use linspace to create a vector consist of 7 values within $(5 - 49)$ range.

2. Create two matrices $3 * 3$ and combine between them in one matrix.

3. Create a matrix $4 * 4$ and implement some operations on created matrix:

   - Find the sum and prod values in the matrix.
   - Find the mean and Transpose values in the matrix.

9.3 Homework of Indexing of Matrix Values

1. Create matrix $5 * 5$ and find the index $(3, 2)$.

2. Create matrix $3 * 3$ and find the index $(3, :)$ and $(:, 3)$.

9.4 Homework of M- Files and Syntax

1. Create function to calculate energy based on the equation $E = mc^2$, where $c = 299792458$.

2. Create any function

9.5 Homework of Control Structures

1. Write a script to check if a number is positive, negative, or zero (if Statements).

2. Write a script that keeps asking the user to enter a positive number. The script will stop when a negative number is entered (while Loop).

9.6 Homework of Plotting and Visualization

1. Create a simple 2D line plot for the function $y = x^2$.

2. Use subplots to display the functions $y_1 = \sin(x_1)$, $y_2 = \cos(x_2)$ and $y_3 = \tan(x_3)$.