Computer Application (MATLAB)

تطبيقات الحاسبة (ماتلاب)
2025-2024

Lecture 8

by
Dr Murtada Dohan
murtada.dohan@uomus.edu.iq

Learning Objectives

- Understand Function Basics.
- Identify and describe built-in, user-defined, and anonymous functions.
- Write user-defined functions with input and output arguments
- Understand how to return and use multiple values from a function.

What Are Functions?

- Definition: Functions in MATLAB are reusable blocks of code designed to perform a specific task.
- **Benefits**: Code reusability.
  - Better organisation and readability.
  - Easier debugging and maintenance.
- Example

```
function output = myFunction(input)
    output = input^2;
end
```

Types of MATLAB Functions

- Built-in Functions:
  - Functions provided by MATLAB, e.g., sum, mean, plot.
- User-defined Functions:
  - Functions created by users for specific tasks.
- Anonymous Functions:
  - One-line functions defined in the command window or script.

Anatomy of a MATLAB Function

- **Structure of a User-defined Function:**

```
function [output1, output2] = functionName(input1, input2)
    % Function Description
    % Code
end
```

- **Components**:
  - function: Keyword to define a function.
  - [output1, output2]: Output arguments (**optional**).
  - functionName: Name of the function (same as the **filename**).
  - (input1, input2): Input arguments.

Creating and Using a Function

- Steps to Create a Function:
  - Create a new file with .m extension.
  - Define the function using the function keyword.
  - Save the file with the same name as the function.
- Example:

```
function area = calculateArea(radius)
    % Calculate the area of a circle
    area = pi * radius^2;
end
```

- Usage:

```
r = 5;
A = calculateArea(r);
disp(A);
```

Built-in vs User-defined Functions

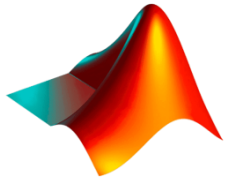
Feature	Built-in Functions	User-defined Functions
Availability	Always available	Created by the user
Examples	sin, cos, sum	Custom calculations
Modification	Not modifiable	Fully customizable

Anonymous Functions

- Definition: One-line functions defined without a .m file.
- Syntax:

```
f = @(x) x^2;
result = f(4); % Output: 16
```

- Use Cases:
  - Quick calculations.
  - Inline expressions for plotting.



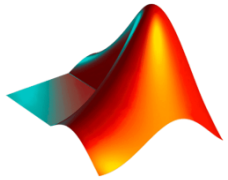
Input and Output Arguments

- Input Arguments:
  - Variables passed into the function.
- Example:

```
function greet(name)
    disp(['Hello, ', name]);
end
```

- Output Arguments:
  - Variables returned by the function.
- Example:

```
function c = add(a, b)
    c = a + b;
end
```



Functions with Multiple Outputs

- **Example:**

```
function [sumVal, prodVal] = calculate(a, b)
    sumVal = a + b;
    prodVal = a * b;
end
```

- **Usage:**

```
[s, p] = calculate(3, 4);
disp(s); % 7
disp(p); % 12
```

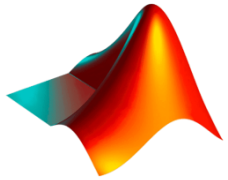
MATLAB®

Practical Example

- **Task:** Create a function to calculate the factorial of a number.
- **Code:**

```
function f = factorial(n)
    if n == 0
        f = 1;
    else
        f = n * factorial(n - 1);
    end
end
```

- **Usage:**
  - result = factorial(5); % Output: 120



Common Errors and Debugging Tips

- Errors:
  - Mismatched input/output arguments.
  - File name and function name mismatch.
- Debugging Tips:
  - Use the **disp** commands.
  - Check the workspace for variable values.

Summary

- Functions improve modularity and reusability.
- MATLAB supports different types of functions:
  - Built-in, user-defined, anonymous.
- Practice by creating and testing simple function

Introduction to MATLAB Plotting

- MATLAB provides extensive facilities for displaying vectors and matrices as graphs.
- Important functions include plot, xlabel, ylabel, title, and legend.
- Example Applications: Signal processing, data visualization, and engineering design.

Types of MATLAB Plots

R2024b

There are various functions that you can use to plot data in MATLAB®. This table classifies and illustrates the common graphics functions.

Line Plots	Scatter and Bubble Charts	Data Distribution Plots	Discrete Data Plots	Geographic Plots	Polar Plots	Contour Plots	Vector Fields	Surface and Mesh Plots	Volume Visualization	Animation	Images
plot	scatter	histogram	bar	geoplot	polarplot	contour	quiver	surf	streamline	animatedline	image
plot3	scatter3	histogram2	barh	geoscatter	polarhistogram	contourf	quiver3	surfc	streamslice	comet	imagesc
stairs	bubblechart	scatterhistogram	bar3	geobubble	polarscatter	contour3	feather	surfl	streamparticles	comet3	
errorbar	bubblechart3	boxchart	bar3h		polarbubblechart	contourslice		ribbon	streamribbon		
area	swarmchart	swarmchart	pareto		compassplot	fcontour		pcolor	streamtube		
stackedplot	swarmchart3	swarmchart3	stem		fpolarplot			fsurf	coneplot		
loglog	spy	piechart	stem3					fimplicit3	slice		
semilogx		donutchart	stairs					mesh			
semilogy		wordcloud						meshc			
fplot		bubblecloud						meshz			
fplot3		heatmap						waterfall			
fimplicit		parallelplot						fmesh			
		plotmatrix									

Related Topics

- Create 2-D Line Plot
- MATLAB Plot Gallery

Link: https://uk.mathworks.com/help/matlab/creating_plots/types-of-matlab-plots.html

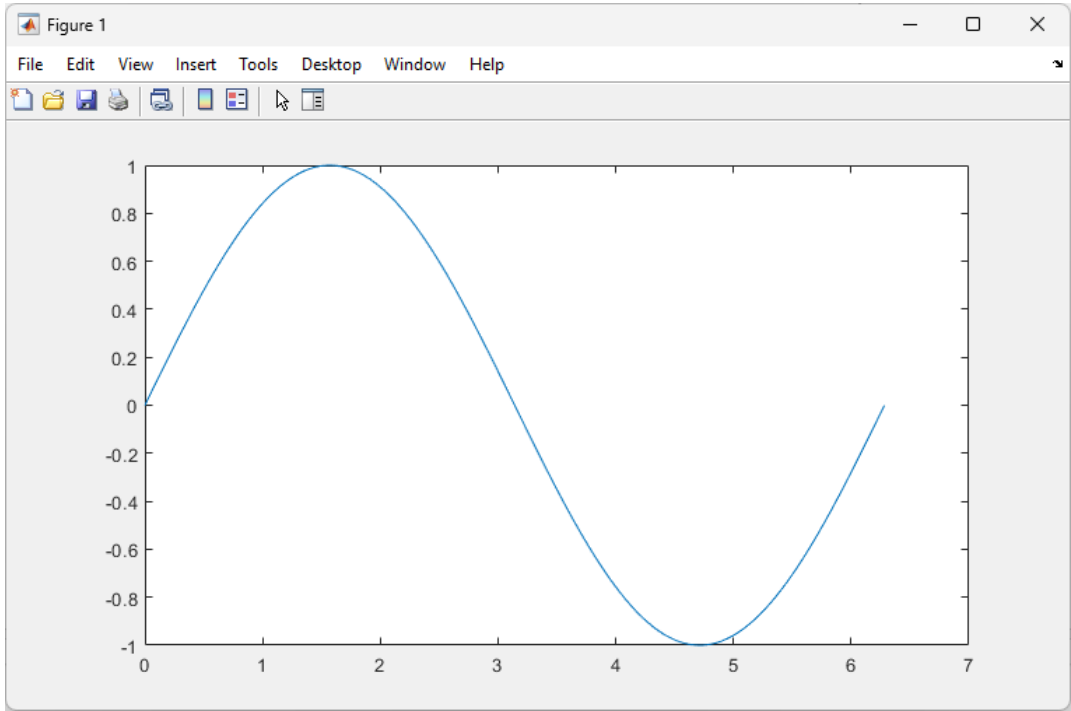
MATLAB®

Creating a Basic Plot

- Create a Plot
  - plot(y): Creates a piecewise linear graph of vector y vs. its index.
  - plot(x, y): Creates a graph of y vs. x.

- Example:

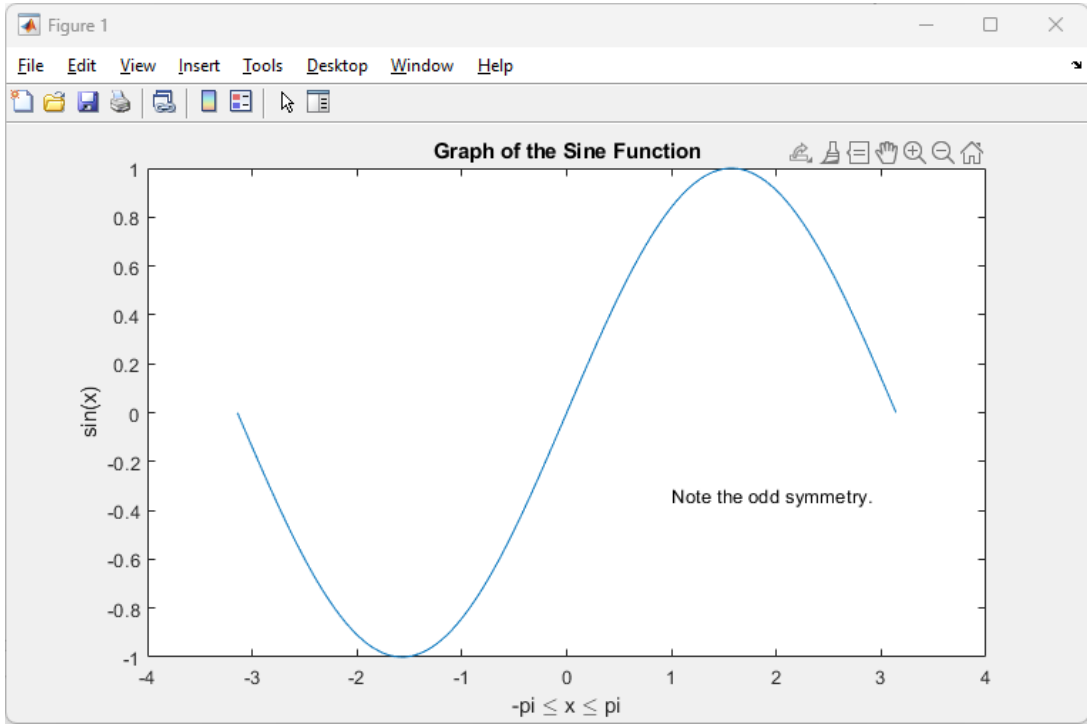
```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x, y);
```



Axis Labels and Titles

- Tools:
  - xlabel, ylabel, zlabel: Add labels to the x, y, and z axes.
  - title: Adds a title to the plot.
  - text: Inserts text anywhere in the figure.
- Example:

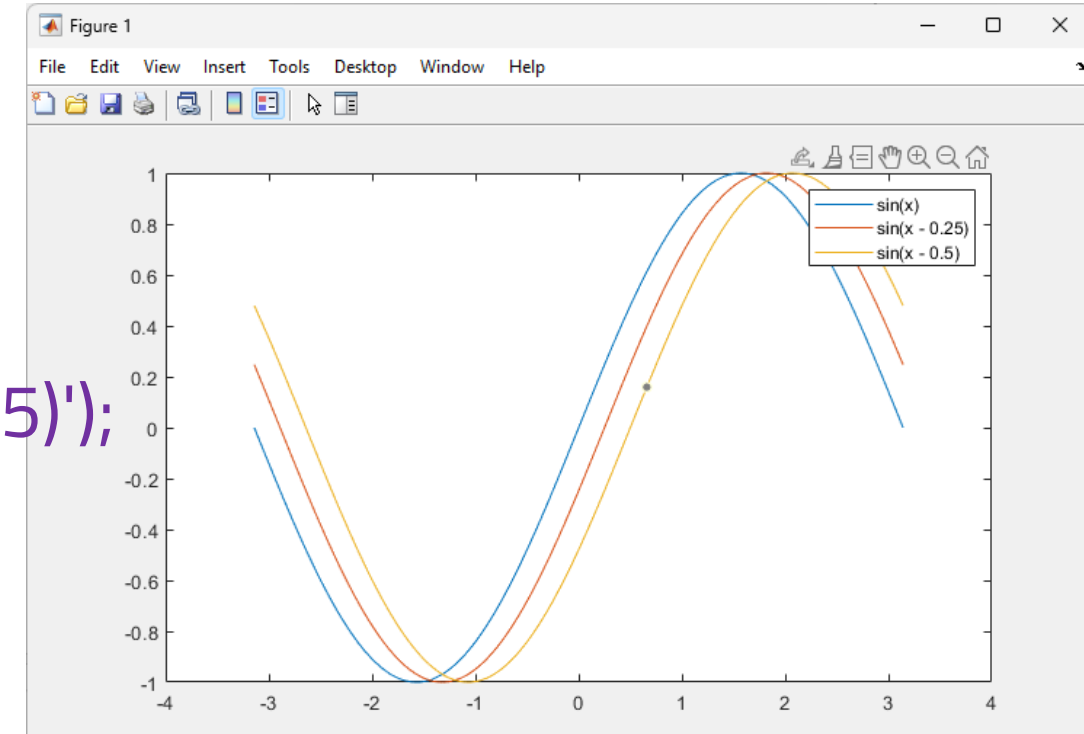
```
x = -pi:pi/100:pi;
y = sin(x);
plot(x, y);
xlabel('-pi \leq x \leq pi');
ylabel('sin(x)');
title('Graph of the Sine Function');
text(1, -1/3, 'Note the odd symmetry.');
```



Multiple Data Sets in One Graph

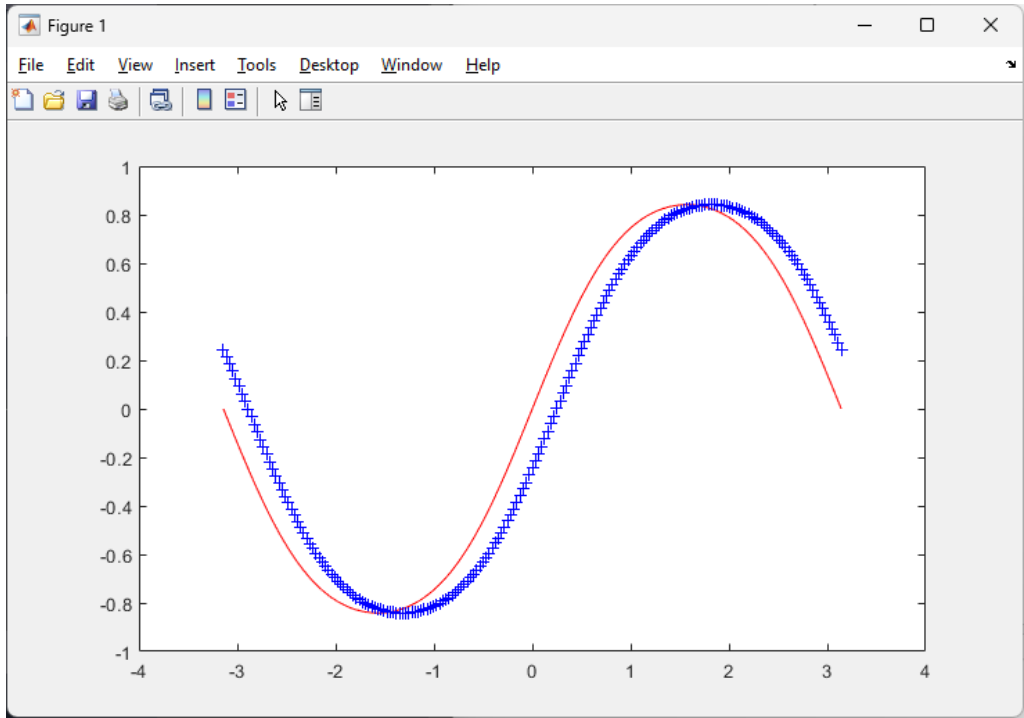
- Plotting Multiple Curves:
  - Use multiple x-y pairs in the plot function.
  - MATLAB automatically cycles through colours.
- Example:

```
y2 = sin(x - 0.25);
y3 = sin(x - 0.5);
plot(x, y, x, y2, x, y3);
legend('sin(x)', 'sin(x - 0.25)', 'sin(x - 0.5)');
```



Specifying Line Styles and Colors

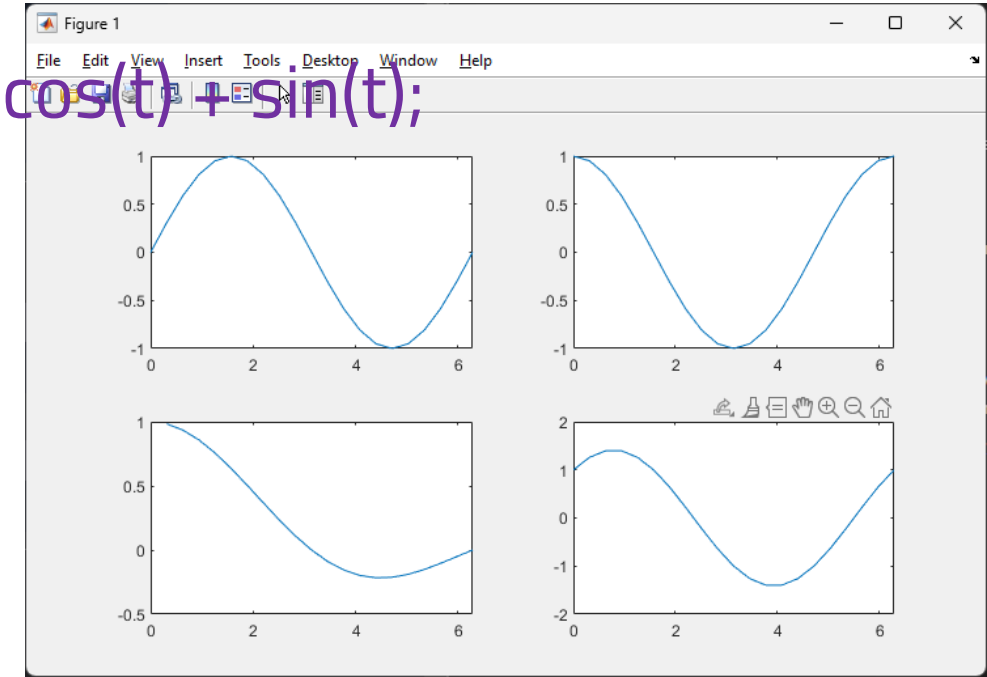
- Syntax:
  - plot(x, y, 'color_style_marker’);
- Options:
  - Colors: 'c', 'm', 'y', 'r', 'g', 'b', 'w', 'k'.
  - Line Styles: '-', '--', ':', '-.'.
  - Markers: '+', 'o', '*', 'x', 's', 'd', '^', 'v', '>’.
- Example:
  - plot(x, sin(y), ‘r-’, x, sin(y2), 'r+');



Multiple Plots in One Figure

- Using subplot:
  - subplot(m, n, p): Divides the figure into an **m-by-n** grid and selects the **p** subplot.
- Example:

```
t = 0:pi/10:2*pi;
y1 = sin(t); y2 = cos(t); y3 = sin(t)./t; y4 = cos(t) + sin(t);
subplot(2, 2, 1); plot(t, y1);
subplot(2, 2, 2); plot(t, y2);
subplot(2, 2, 3); plot(t, y3);
subplot(2, 2, 4); plot(t, y4);
```



Setting Axis Limits

- Syntax:
  - axis([xmin xmax ymin ymax]);
- Example:

```
axis([-pi pi -1 1]);
```

Summary

- Key Points:
  - Basic plotting in MATLAB using plot, xlabel, ylabel, title.
  - Plotting multiple datasets and customizing line styles and colors.
  - Using subplot for multiple plots in one figure.