Computer II (MATLAB)

الحاسوب 2

2025-2024

Lecture 7

by

Dr. Ahmed Hasan Al-Janabi

Ahmed.janabi@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

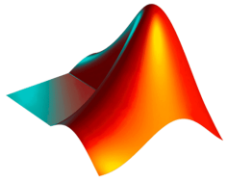
Feature	Built-in Functions	User-defined Functions
Availability	Always available	Created by the user
Examples	sin, cos, sum	Custom calculations
Modification	Not modifiable	Fully customizable

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

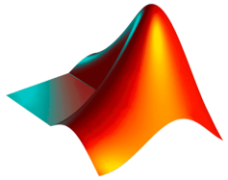
- Input Arguments:
  - Variables passed into the function.
- Example:

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

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

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

A = add(2,4);
disp(A);
```



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
```

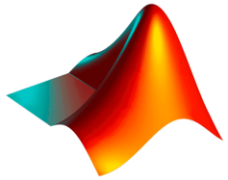
MATLAB®

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



MATLAB®

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