Computer Application (MATLAB)

تطبيقات الحاسبة (ماتلاب)
2025-2024

Lecture 3

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

Learning Objectives

- Understand how to write and execute basic MATLAB expressions.
- Learn how to create and manipulate matrices in MATLAB.
- Work with variables and apply assignment statements.
- Familiarize yourself with MATLAB syntax and the use of operators.
- Use comments effectively to document your code.
- Learn commands to manage the workspace, including clc, clear, and clear all

Comments in MATLAB

- Comments are lines of text in your code that MATLAB ignores during execution.
- They are used to explain code, make it more readable, and provide context for yourself and others.
- In MATLAB, comments are written using the percent sign **%**
- Example:

```
% This is a comment
x = 5; % This is another comment
```

Understanding MATLAB Syntax

- MATLAB Syntax refers to the set of rules that define the structure of valid MATLAB commands.
- Key Components of MATLAB Syntax:
  - Commands and functions.
  - Variables and operators.
  - MATLAB is case-sensitive (e.g., A is different from a)

Basic Structure of MATLAB Expressions

- A MATLAB expression typically consists of variables, operators, and functions.
- General Structure:

```
result = expression;
```

- Examples:

```
x = 5 + 3;
y = sqrt(16);
```

- The equals sign (=) is used for assignment, where the result of the expression on the right is stored in the variable on the left.

Common Operators in MATLAB

- Arithmetic Operators.

  + Addition, **-** Subtraction, ***** Multiplication, **/** Division, **^** power.

- Relational Operators.

  == Equal to, **~=** Not equal to, **>** Greater than, **<** Less than.

- Logical Operators.

  && Logical AND, **||** Logical OR, **~** Logical NOT.

Common Operators in MATLAB

- Example:

```
x = 3 + 4;      % Arithmetic
y = x > 5;      % Relational (True/False)
z = x && y;     % Logical (True/False)
```

Variables in MATLAB

- Definition: A variable is a named location in memory that stores data.
- Rules for Variable Names:
  - Must start with a letter.
  - Can include letters, numbers, and underscores (_).
  - MATLAB is case-sensitive (e.g., myVar and myvar are **different**).

Variables in MATLAB

- Examples of valid variables:

```
x = 5;
speed_of_light = 3e8;
temperature1 = 298;
```

- Invalid variables:
  - Numbers or special characters at the start (e.g., 1stVar or @value).

Variable Assignment

- Assignment Statement Format:

```
variable_name = expression;
```

- Examples:

```
a = 10;
b = 25 + 7;
c = sqrt(a)
```

- Reassigning Values:
- You can update the value of a variable at any time:

```
a = 10;
a = a + 5;
```

Basic Arithmetic Operators

- MATLAB supports basic arithmetic operators:
  - + : Addition
  - - : Subtraction
  - * : Multiplication
  - / : Division
  - ^ : Power
- Examples:

```
x = 3 + 5;
y = 10 - 2;
z = 4 * 7;
w = 8 / 2;
p = 3^2;
```

Operator Precedence in MATLAB

- Order of Operations:
  - MATLAB follows the PEMDAS rule:
  - Parentheses
  - Exponents (Power ^)
  - Multiplication and Division (*, /)
  - Addition and Subtraction (+, -)
- Examples:

```
result1 = 3 + 5 * 2;
result2 = (3 + 5) * 2;
result3 = 5^2 - 2 * 3;
```

Evaluating Expressions in MATLAB

- Examples:

```
a = 5;
b = 3;
result = a + b * 2;
```

- Combining Variables and Functions:

```
result = sqrt(a^2 + b^2);
```

- Note: MATLAB evaluates from left to right, adhering to the order of precedence.

Using Parentheses in MATLAB Expressions

- Purpose of Parentheses:
  - To control the order of operations in complex expressions.
  - Example:

```
result = (5 + 3) * (10 - 2); % Forces addition and subtraction first
```

  - Without Parentheses:

```
result = 5 + 3 * 10 - 2;  % MATLAB uses its default precedence rules
```

Common Syntax Errors and How to Avoid Them

- Missing or Extra Parentheses:

```
result = (5 + 3 * 2;  % Missing closing parentheses
```

- Incorrect Use of Operators:

```
result = 5 + * 3; % Multiplication operator misplaced
```

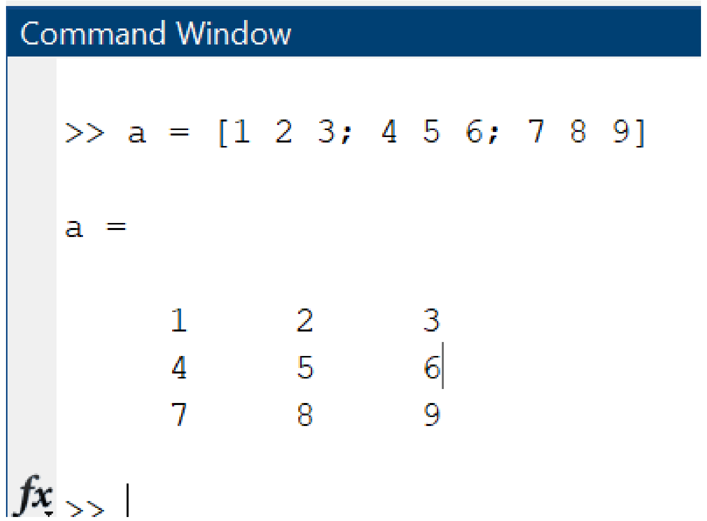
- Case Sensitivity::

```
a = 5;
A = 10;  % 'a' and 'A' are different variables
```

Entering Matrices in MATLAB

- Steps to type a matrix into MATLAB:
  - Begin with a square bracket **[**.
  - Separate elements in a row with **spaces** or **commas**.
  - Use a semicolon **;** to separate rows.
  - End with a square bracket **]**.

- Example: a = [1 2 3; 4 5 6; 7 8 9]



```
>> a = [1 2 3; 4 5 6; 7 8 9]

a =

     1     2     3
     4     5     6
     7     8     9

>> 
```

Generating Matrices with MATLAB

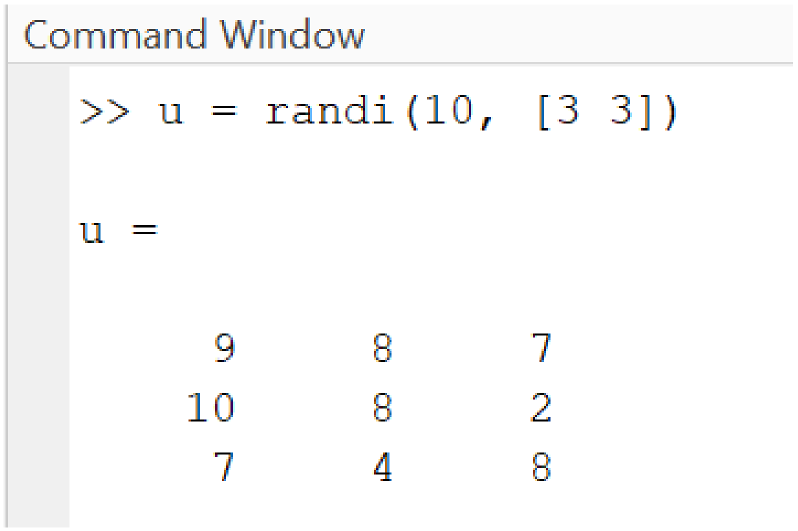
- MATLAB offers functions for generating specific types of matrices:
  - zeros(m, n): Generates a matrix filled with zeros.
  - ones(m, n): Generates a matrix filled with ones.
  - randi(max_val, [m, n]): Generates a matrix with random integers.
  - eye(n): Generates an identity matrix.

Generating Matrices with MATLAB

- Examples:

  u = randi(10, [2 2])

  u = 7 2
  9 4



```
Command Window
>> u = randi(10, [3 3])

u =

     9     8     7
    10     8     2
     7     4     8
```

Try on your machine

- Z = zeros(3, 3);
- O = ones(2, 4);
- U = randi(5, [3, 3]);
- I = eye(4);

Try on your machine

- Z = zeros(3, 3);

```
>> Z = zeros(3, 3)

Z =

     0     0     0
     0     0     0
     0     0     0
```

```
>> Z = zeros(1,4)

Z =

     0     0     0     0
```

```
>> Z = zeros(4,1)

Z =

     0
     0
     0
     0
```

Try on your machine

- O = ones(2, 4);

```
>> O = ones(2, 4)

O =

     1     1     1     1
     1     1     1     1
```

```
>> O = ones( 4,2)

O =

     1     1
     1     1
     1     1
     1     1
```

```
>> O = ones(5)

O =

     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```

Try on your machine

- U = randi(5, [3, 3]);

```
>> U = randi(5, [3, 3])

U =

   5    5    1
   1    3    3
   5    5    5
```

```
>> U = randi(1000, [3, 3])

U =

  793    36   679
  960   850   758
  656   934   744
```

```
>> U = randi(15, [4])

U =

   10   15   12   14
    3    6    4   15
    2    9    8    9
    8    4   11    3
```

Try on your machine

- I = eye(4);

```
>> I = eye(4)

I =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
>> I = eye(5,3)

I =

     1     0     0
     0     1     0
     0     0     1
     0     0     0
     0     0     0
```

```
>> I = eye(5,1)

I =

     1
     0
     0
     0
     0
```

The clear all Command

- Definition: clear all removes all variables, functions, and MEX files from the workspace.
- Purpose:
  - To completely reset the workspace.
  - Useful when starting a fresh session or avoiding conflicts.
- Usage:

```
clear all;
```

- Note: It’s more comprehensive than clear since it also clears functions and variables.

The clc Command

- Definition: clc clears the Command Window, removing all previous output.
- Purpose:
  - To clean up the Command Window when starting a new calculation or experiment.
- Usage:

```
clc
```

- Example:

```
x = 10;
disp(x);
```

- After:

```
clc
```

Commands Review

- clc:
  - Clears the Command Window.
  - Does not affect variables or the workspace.
- clear:
  - Removes specific variables or all variables if no argument is given.
  - Does not affect functions or the Command Window.
- clear all:
  - Clears everything (variables, functions, MEX files).
  - Resets the entire workspace.

Review of Key Concepts

- Basic MATLAB Expressions.
- Matrix Creation.
- Variables and Assignment.
- MATLAB Syntax & Operators.
- Comments %.
- clc, clear and clear all.

Practice Exercise 1

- Assign the variable x a value of 15 and y a value of 5.
- Calculate the result of (x^2 + y^2) and store it in a variable called result.
- Use the disp function to display the value of result.

Practice Exercise 2

- Assign values to variables a, b, and c.
- Compute the quadratic equation a*x^2 + b*x + c = 0 for x = 3.
- Use the disp function to show the result.
- Add comments to explain each step.

Practice Exercise 3

- Create a 3x3 matrix with random integers between 1 and 20.
- Create a 3x3 matrix with values from 1 to 9.
- Use addition operation to sum the arrays.
- Clear all variables and use clc to clear the Command Window.

Exercises Submission

- All exercises need to be submitted by Monday 28 Oct 23:59.

- Submit your answers via: https://forms.gle/XFW53HAUiEtKuHRK9

Let’s try MATLAB

Launch MATLAB and work towards the exercises