Computer Application (MATLAB)

تطبيقات الحاسبة (ماتلاب)
2025-2024

Lecture 4

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

Learning Objectives

- Subscripting, Colon Operator, end Keyword, Transpose, Deletion.
- Understand the basic structure and purpose of if-else statements in MATLAB.
- Be able to implement conditional logic using if, elseif, and else.
- if statements with arrays
- Logical operations on arrays
- Element-wise

Metrics Operations

```
>> myarray = ones(2,2)/2
myarray =
    0.5000    0.5000
    0.5000    0.5000
```

MATLAB®

Selecting Elements with Subscripting

- Use indices to select specific elements or submatrices.
- Examples:

```
u = [0.9, 0.7, 0.2, 0.4, 0.9];
u(3);  % Accesses the 3rd element: 0.2

a = [1 2 3; 4 5 6];
a(2,3);  % Accesses the element in the 2nd row, 3rd column: 6
a(1:2, 3);  % Selects elements in rows 1 and 2, column 3
```

MATLAB®

The Colon Operator in MATLAB

- Purpose: The colon operator (:) is versatile for creating vectors, iterating, and subscripting.
- Examples:

```
a = [1 2 3; 4 5 6; 7 8 9];
a(2, :);  % Selects the entire 2nd row: [4 5 6]
a(:, 3);  % Selects the entire 3rd column: [3 6 9]
a(:);  % Flattens matrix into a single column vector
```

MATLAB®

Accessing the Last Element with end

- Explanation: end refers to the last index in a dimension.
- Examples:

```
q = [7 8 9 10; 6 1 2 20; 5 4 3 30];
q(end, end);  % Accesses the last element: 30
q(2, end-1:end);  % Selects last two elements in the 2nd row: [2 20]
q(end-2:end, end-1:end);  % Selects submatrix of last two rows, columns
```

Transposing Matrices

- Definition: The transpose operation switches rows with columns.
- Syntax: Use ' to transpose.
- Examples:

```
>> D = [2 0 1; 9 6 8; 4 1 1]

D =

     2     0     1
     9     6     8
     4     1     1

>> D'

ans =

     2     9     4
     0     6     1
     1     8     1
```

MATLAB®

Deleting Rows or Columns

- Syntax: Set a row or column to [] to delete it.
- Examples:

```
a = [1 2 3; 4 5 6; 7 8 9];
a(:, 2) = [];  % Deletes the 2nd column
a=
     1  3
     4  6
     7  9
```

Introduction to Conditional Statement

- What are Conditional Statements?
  - Statements that execute different code based on certain conditions.
  - Enable decision-making in programs, controlling which code segments run.
- MATLAB's if-else Structure:
  - if: Runs a block of code if the condition is true.
  - elseif: Specifies additional conditions.
  - else: Runs a block of code if all previous conditions are false.

The Structure of if-else in MATLAB

- Basic syntax:

```
if condition
    % Code to execute if condition is true
elseif other_condition
    % Code to execute if other_condition is true
else
    % Code to execute if none of the conditions are true
End
```

- Note: Always close the if-else statement with end.

Example: Simple if Statement

- Example code to check if a number is positive:

```
x = 5;
if x > 0
    disp('x is positive');
end
```

- Explanation:
  - If x > 0 is true, MATLAB displays "x is positive."
  - If x were negative or zero, the code within if would not execute.

Example: if-else Statemen

- Example code to check if a number is positive or negative:

```
x = -3;
if x > 0
    disp('x is positive’);
else
    disp('x is negative’);
end
```

- Explanation:
  - MATLAB evaluates x > 0. If false, the code within else executes instead..

Example: if-elseif-else Statement

- Example to check if a number is positive, negative, or zero:

```
x = 0;
if x > 0
    disp('x is positive');
elseif x < 0
    disp('x is negative');
else
    disp('x is zero');
end
```

- Explanation:
  - MATLAB evaluates each condition in order until one is true. If none are true, else executes.

Using Logical Operators in Conditions

- Logical Operators allow combining multiple conditions:
  - **&&:** Logical AND (both conditions must be true)
  - **||:** Logical OR (at least one condition must be true)
  - **~:** Logical NOT (inverts true to false and vice versa)
- Example:

```
a = 5; b = 10;
if a > 0 && b > 0
    disp('Both a and b are positive');
end
```

MATLAB®

Nested if-else Statements

- What is Nesting?
  - Placing one if-else statement inside another for complex conditions.
- Example:

```
x = 10;
if x > 0
    if x > 5
        disp('x is positive and greater than 5');
    else
        disp('x is positive but 5 or less');
    end
end
```

Example: Grade Classification

- Write a script to classify a student's grade based on their score:
- Example:

```
score = 85;
if score >= 90
    disp('Grade: A');
elseif score >= 80
    disp('Grade: B');
elseif score >= 70
    disp('Grade: C');
elseif score >= 60
    disp('Grade: D');
else
    disp('Grade: F');
end
```

Common Mistakes to Avoid

- Forgetting to use **end** to close **if-else** blocks.
- Incorrectly using **=** instead of **==** for equality check.
- Overusing nested **if-else** statements when simpler logic would suffice.
- Mixing up logical operators (e.g., **&&** and **||**).

if with Entire Arrays

- MATLAB evaluates conditions in if statements as true if all elements of an array meet the condition.
- Example:

```
A = [1, 2, 3];
if all(A > 0)
    disp('All elements are positive');
end
```

- Note: If any element does not satisfy the condition, if will consider the entire condition as false.

Using all and any Functions with Arrays

- all(array): Returns true if all elements of array are true.
- any(array): Returns true if at least one element of array is true.
- Examples:

```
A = [1, -3, 5];
if any(A < 0)
    disp('There are negative elements');
End

if all(A > 0)
    disp('All elements are positive');
else
    disp('Not all elements are positive');
end
```

Applying Element-wise Condition

- Element-wise conditions allow applying logical tests to each element in an array.
- Syntax: Use element-wise operators with arrays (&, |, ~).
- Example:

```
A = [5, -3, 8];
B = A > 0;  % Element-wise comparison
disp(B);  % Output: [1 0 1]
```

MATLAB®

Conditional Indexing with Arrays

- You can use logical conditions to select elements from an array.
- Example:

```
A = [1, -2, 3, -4, 5];
posElements = A(A > 0);  % Select positive elements
disp(posElements);  % Output: [1 3 5]
```

MATLAB®

Combining Multiple Conditions on Arrays

- Use logical operators to combine conditions for element-wise evaluations.
- Example:

```
A = [10, 15, 20, 25, 30];
selectedElements = A(A > 10 & A < 25);  % Elements between 10 and 25
disp(selectedElements);  % Output: [15 20]
```

Using Nested if Statements with Arrays

- Use nested if statements for multi-step checks on arrays.
- Example:

```
A = [4, 9, 16, 25];
if all(A > 0)
    if any(sqrt(A) == 5)
        disp('Array contains an element whose square root is 5');
    else
        disp('No element has a square root of 5');
    end
end
```

Review of Key Concepts

- Subscripting, Colon Operator, end Keyword, Transpose, Deletion.
- Use if, elseif, and else to create conditional branches
- if with Arrays: Evaluates as true only if all elements meet the condition.
- Logical Functions: all and any for evaluating conditions across elements.
- Element-wise Conditions: Apply conditions on individual array elements.
- Conditional Indexing: Select elements that meet specific conditions.

Practice Exercise 1

- ask 1: Create an array and check if all elements are greater than zero.
- Task 2: Find and display elements that are greater than a specified threshold (e.g., 10).
- Task 3: Check if any element in the array is negative; if so, display "Contains negative values."

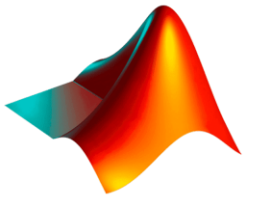
Exercises Submission

- All exercises need to be submitted by Monday 4th Nov 23:59.

- Submit your answers via: https://forms.gle/UaPR65LQ3ib9DUYn9

Let’s try MATLAB

Launch MATLAB and work towards the exercises



MATLAB®