Computer Application (MATLAB)

تطبيقات الحاسبة (ماتلاب)
2025-2024

Week 2

by
Dr Murtada Dohan

Learning Objectives

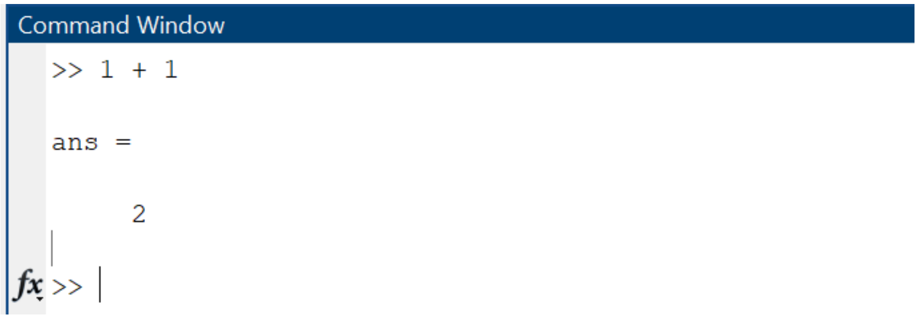
- Get familiar with basic MATLAB operations and expressions.
- Learn how to create matrices.
- Understand variable assignments.
- Use matrix generators to create common matrices with zeros, ones, random numbers, or identity matrices.
- Apply comments to improve code clarity and use clc and clear commands to manage the workspace efficiently.

First Steps with MATLAB (1)

- To get MATLAB to work out basic operations, simply type at the command prompt:

  1 + 1

- MATLAB responds with: ans = 2

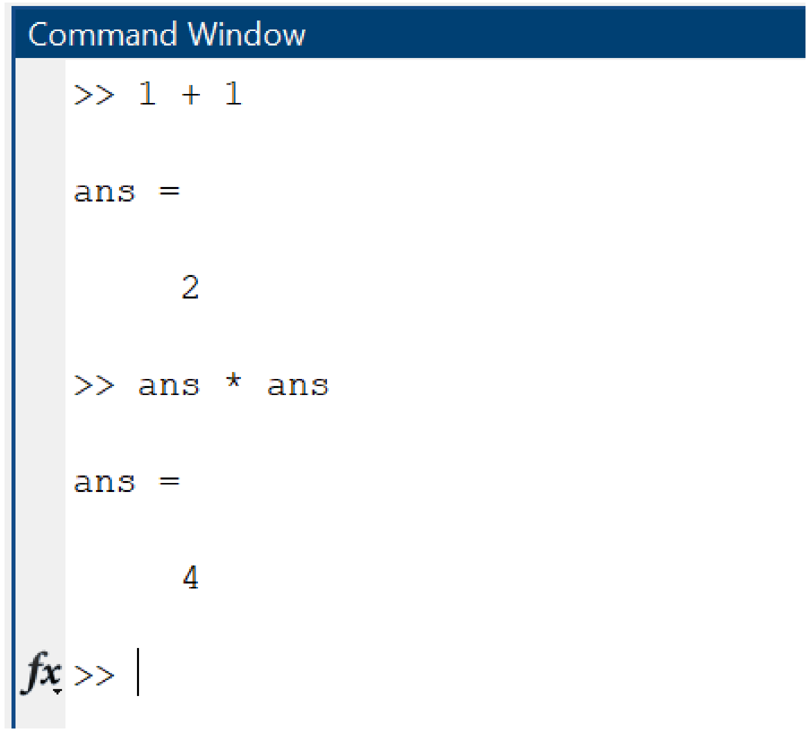


Command Window

```
>> 1 + 1

ans =

     2

>> 
```

First Steps with MATLAB (2)

- MATLAB stores the result in the variable ans, which you can reuse.
- Ex: ans * ans
- MATLAB responds with: ans = 4



```
>> 1 + 1

ans =

     2

>> ans * ans

ans =

     4

>> 
```

Spacing in MATLAB Expression

- The spacing of operators doesn’t affect the result.

```
1 + 3 * 2 - 1 / 2 * 4
```

- Gives the same answer as:

```
1+3*2-1/2*4
```

- Clearer formatting improves **readability**. Use parentheses for clarity:

```
1 + 3*2 - (1/2)*4
```

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;
```

Variables and Assignment in MATLAB

- Variables are memory locations used to store data.
- Variable names can include letters and digits but must start with a letter.
- MATLAB does not require variable declarations, but this can sometimes lead to errors.
- Assignment Example:

```
a = 6;
name = 'Mark';
```

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;
```

Displaying Variables

- Use the disp function to display variable contents.
- Example:

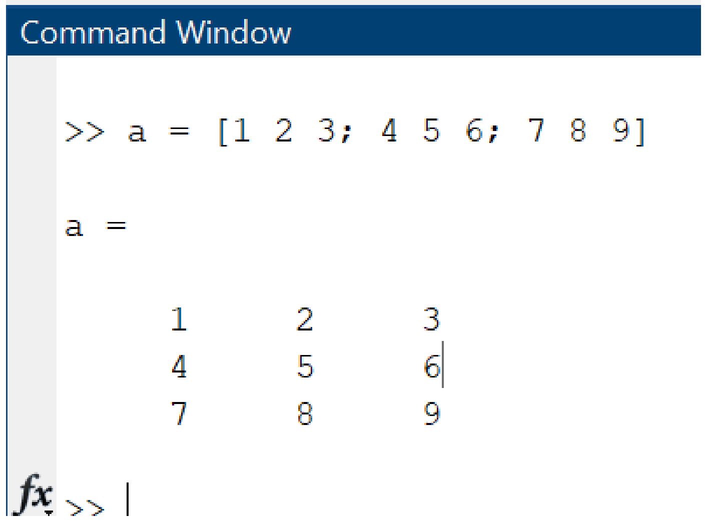
```
disp(a);
disp(name);
```

- Alternatively, typing the variable name at the command prompt will display its value.

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

>> 
```

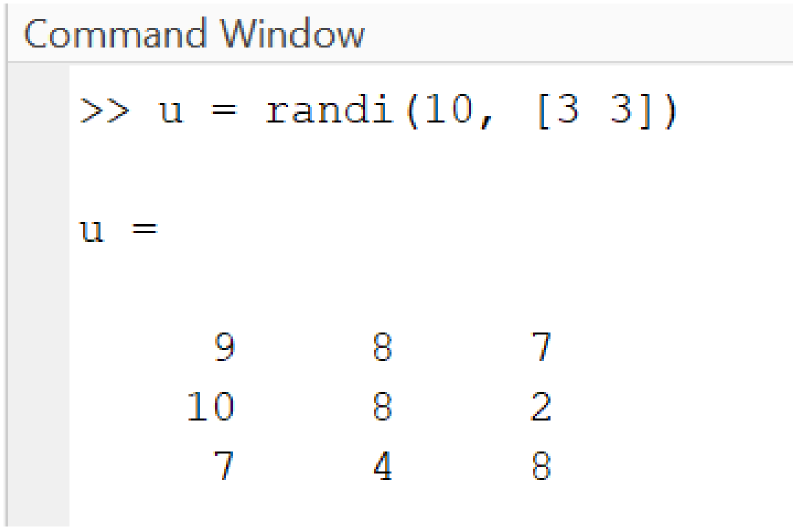
Generating Matrices with MATLAB

- 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);

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

- Recap:
  - How to enter expressions and work with the ans variable.
  - Properly spacing operations for readability.
  - Creating matrices manually and using matrix generators (zeros, ones, randi, eye).
  - Variable assignment and displaying results using disp

Practice Exercise 1

- Create a 5x5 matrix of random integers between 1 and 10.
- Create a 3x3 identity matrix and a 4x4 matrix filled with ones.

Practice Exercise 2

- Assign values to two variables and compute their sum, product, and difference.
- Display the result using disp.

Practice Exercise 3

- Task: Define variables x=22, y=3,z=5.
- Apply the following equations:
  - Resutls1=x+4*y+10
  - Results2=(x*2)+z/3
  - Results3=y^z+(x*2)/3

Exercises Submission

- All exercises need to be submitted by Monday 21st Oct 23:59.
- Submit your answers via: https://forms.gle/VTybujzDdq1r9gULA

Let’s try MATLAB

Launch MATLAB and work towards the exercises