**Lab no 01**

**Introduction to MATLAB**

MATLAB is a technical computing environment for high-performance numeric computation and visualization. MATLAB integrates numerical analysis, matrix computation, signal processing, and graphics into one easy-to-use compiler-less environment. Unlike traditional programming languages such as C, MATLAB is optimized for vector and matrix operations. In fact, the name MATLAB stands for matrix (MAT), laboratory (LAB).

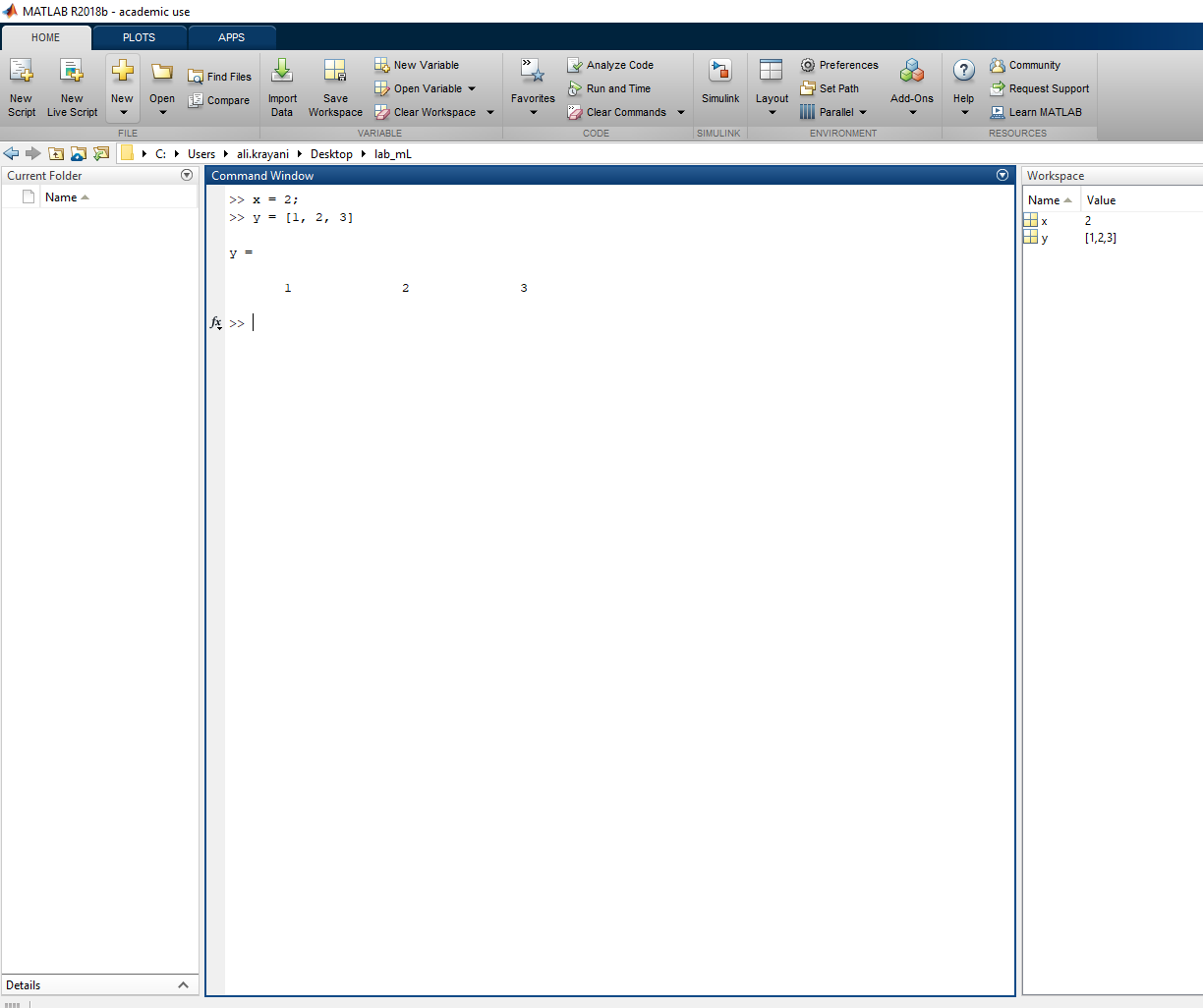
In order to understand well programming with MATLAB we will provide examples and exercises to learn and practice immediately.

In this introductory tutorial we will cover the following topics (basics of MATLAB):

1. Variables
2. Relational operators
3. Logical Operators
4. Vectors and Matrices
5. Flow Control
6. Graphics Fundamentals

Each section contains some examples, try to execute these examples for better understanding.

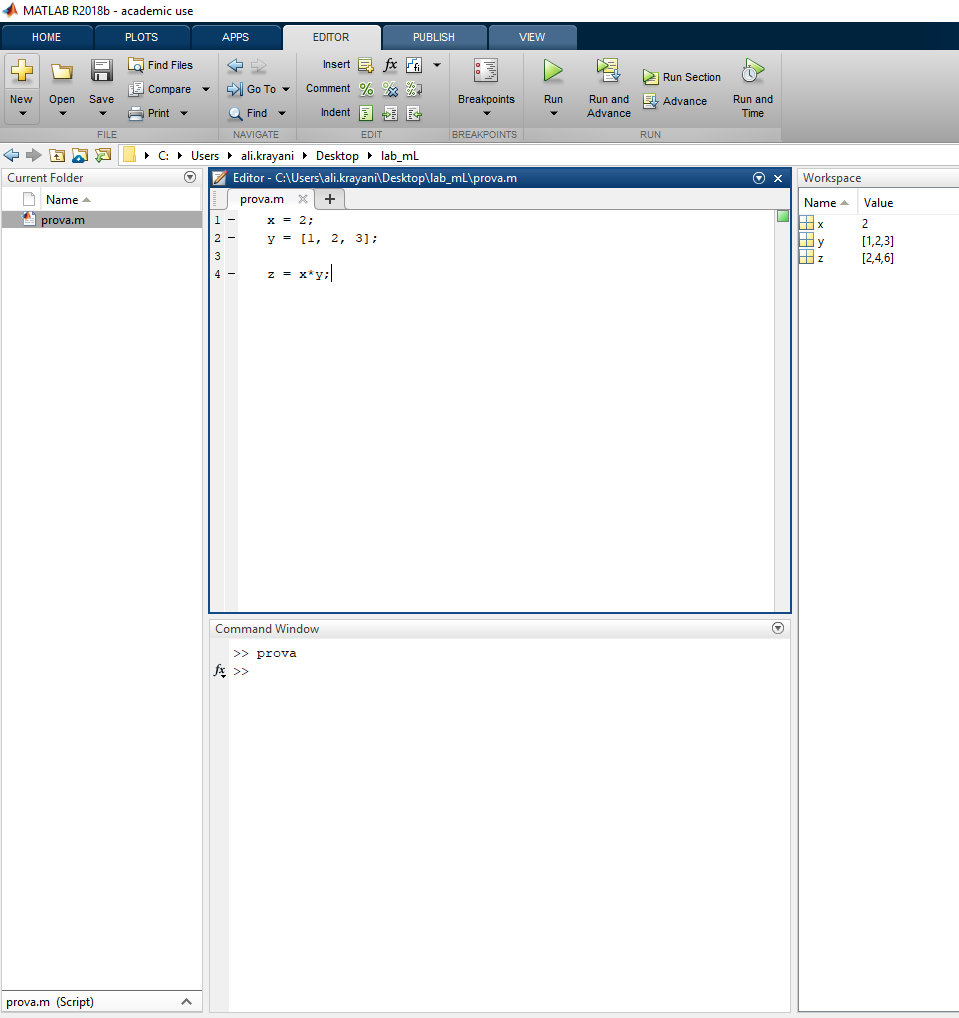
You can execute your code directly from the command prompt or by creating a new script as shown in the following images:



You can type and run your code directly in the **Command window**

In **Workspace** you can find your results and variables

OR you can create a script and run it as follows:



**Create a new script (new -> script)**

**After typing the code that you want to execute.**

**Press Run.**

**Then you can find the results in the workspace**

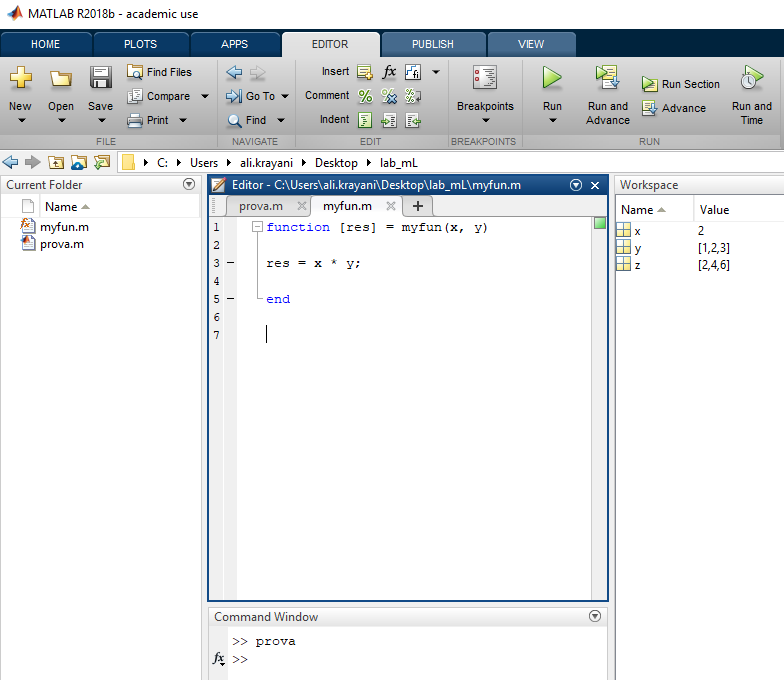
**To visualize the variable, you can just write its name in the command window**

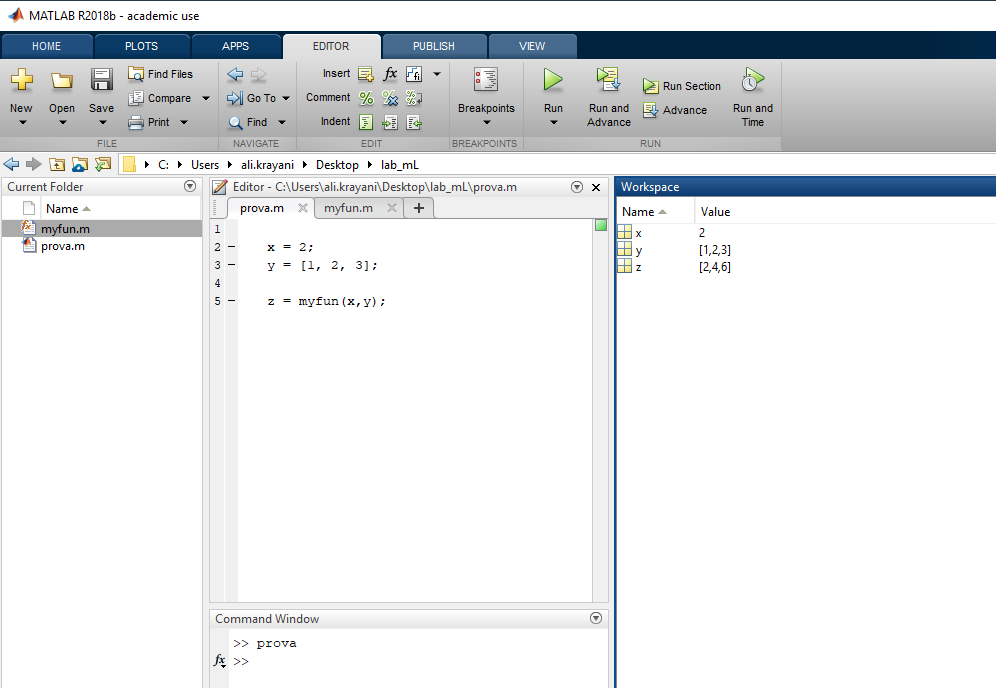
How to use function in MATLAB

Syntax of a function statement is:

function [out1, out2, ..., outN] = myfun(in1, in2, in3, ..., inN)

Ex: let’s call a function to calculate the product of two vectors





**Right click here**

**new**

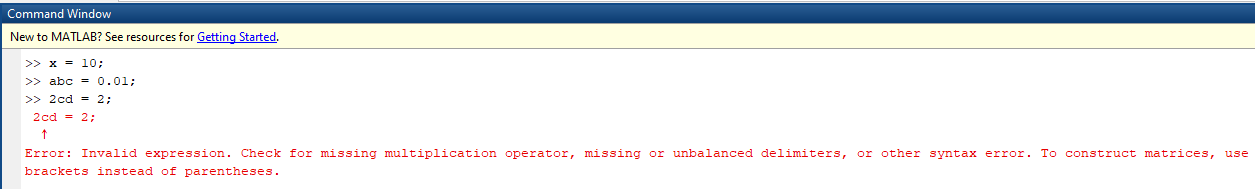
**Function**

1. **Variables**

Variable names must start with a letter and can be followed by letters, digits and underscores.

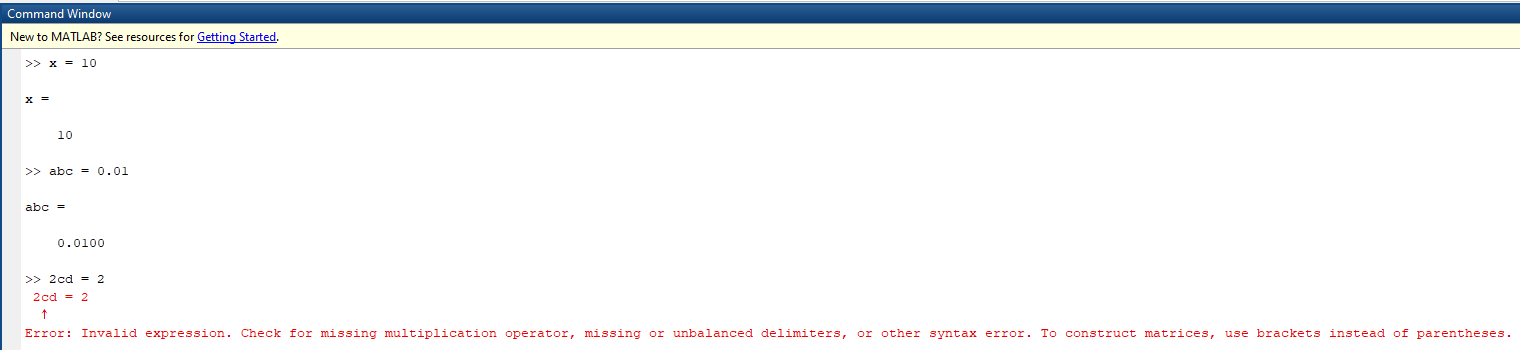
*variablename = expression*

Example :



Putting a semicolon at the end of a statement suppresses the output.

Example without semicolon:



1. **Relational Operators**

MATLAB supports six relational operators

Less than <

Less than or equal <=

Greater than >

Greater than or equal >=

Equal to ==

*Not Equal to ~=*

1. **Logical Operators**

MATLAB supports three logical operators

not ~

and &

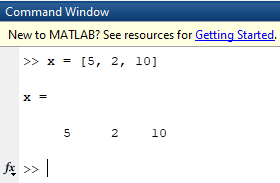
or ||

1. **Vectors and Matrices**

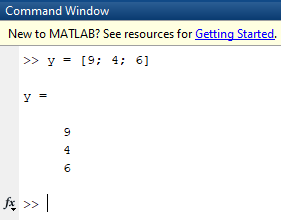
MATLAB treats all variables as matrices. A matrix can be thought of as an array, in fact, that is how it is stored.

Vectors are special forms of matrices and contain only one row OR one column.

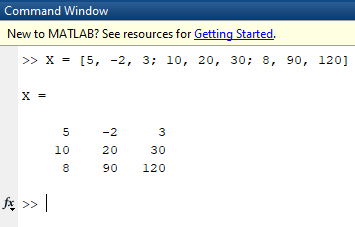
A matrix with only one row is called a row vector. A row vector can be created in MATLAB as follows (note the commas):



A matrix with only one column is called a column vector. A column vector can be created in MATLAB as follows (note the semicolons):

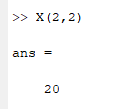


A matrix can be created in MATLAB as follows (note the commas and semicolons)

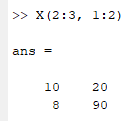


Note: MATLAB’s indices start from 1

You extract any element from the matrix by specifying the corresponding rows and columns. For example, lets extract value 20 which has (row, column) = (2,2):



Also, we can extract a sub-matrix from the original matrix. For example, lets extract elements from 2nd, 3rd row and from 1st, 2nd column



|  |
| --- |
| **Matrix Manipulation functions** |
| **zeros** : creates a matrix of all zeros |
| **ones** : creates a matrix of all ones |
| **eye** : creates an identity matrix |
| **rand** : generates uniformly distributed random numbers in [0, 1] |
| **diag** : diagonal matrices and diagonal of a matrix |
| **size** : returns matrix dimensions |
| **length** : returns length of a matrix (row or column) |
| **det** : matrix determinant |
| **inv** : matrix inverse |
| **find** : searches for the given values in an array or in a matrix |

For more information, type *help function* at command prompt. type for example *help* ***zeros***.

1. **Flow Control**

MATLAB has five flow control statements

* If
* for
* while
* switch
* break

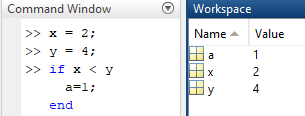
**> if** : the general form of the if statement is

if expression

…

end

Ex:



with more than one **if**

if expression

…

else if expression

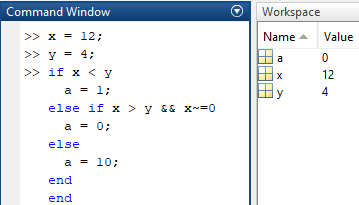
…

else

…

end

Ex



>**for**: repeat statements a specific number of times.

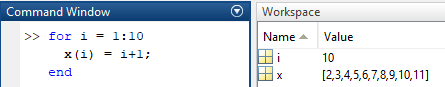
The general form of a **for** statement is

for variable = expression

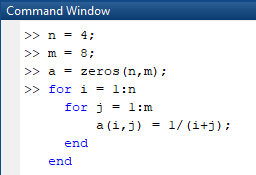
…

end

Ex1



Ex2



**>while** loop: repeat statements an indefinite number of times.

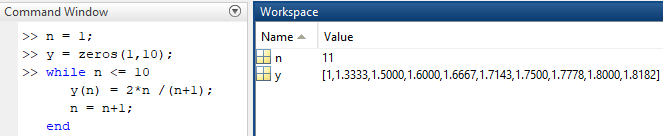
The general form of while statement is

while expression

…

end

Ex



**>switch:** Switch among several cases based on expression

The general form of the switch statement is:

switch switch\_exp

case case\_expr1

…

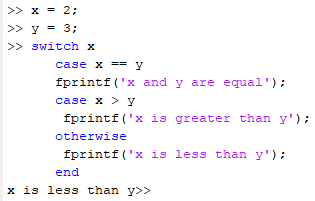
case case\_expr2

…

otherwise

…

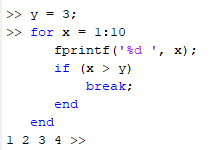
end



**>break**

it terminates the execution *for* and *while* loops

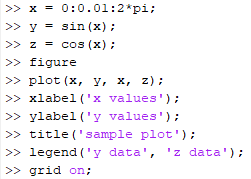
Ex



1. **Graphics Fundamentals**

2D Plotting

Ex: Plot sin(x) and cos(x) over [0,2], on the same plot with different colours





**title**

**Y label**

**legend**

**X label**

**Grid**

|  |
| --- |
| **Elementary Math functions**  **(functions already defined in matlab)** |
| **abs** : finds absolute value of all elements in the matrix |
| **sign** : signum function |
| **sin**, **cos** : trigonometric functions |
| **asin**, **acos** : inverse trigonometric functions |
| **exp** : exponential |
| **log**, **log10** : natural logarithm, logarithm (base 10) |
| **ceil**, **floor** : round towards +infinity, -infinity respectively |
| **round** : round towards nearest integer |
| **real**, **imag** : real and imaginary part of a complex matrix |
| **sort** : sort elements in ascending order |
| **sum, prod** : summation and product of elements |
| **max, min** : maximum and minimum of arrays |
| **mean, median** : average and median of arrays |
| **std, var** : standard deviation and variance |

For more information, type *help function* at command prompt. type for example *help* ***abs***.

After introducing the basics of matlab, we are going to do some practice.

For each exercise (except in Exercise 1 use the command window), create a new script to execute your code (save it as ex\_number.m)

**Exercises**

**Exercise 1**

Define the following column arrays in MATLAB: and

Then issue the following commands:

a’

a\*b

a’\*b

a.\* b

3.\*b

What do each of these three operators (‘ \* .\*) do ?

**Exercise 2:**

A general system of linear equations can be expressed in terms of a co-efficient matrix *A*, a right-hand side (column) vector b and an unknown (column) vector *x* as follows:

Or component wise, as:

The solution is obtained by:

Given:

Solve , , , , using the matrix functions?

**Exercise 3**

Write a script to print a triangle of \*’s, as follows

To print you can use fprintf(‘ ’), for new line you can use fprintf(‘\n’);

You must use a “for” loop.

**Exercise 4**

Create and plot a signal using the following commands:

**>> t = -10:0.1:10;**

**>> x0 = t.\*exp(-abs(t));**

**>> plot(t, x0)**

Create the related signals and

Plot all three signals together in one window.

**Exercise 5:**

Generate a vector consisting of random floating numbers between 0 and 1.

Use: where 1, 10 are numbers of rows and columns respectively.

Note: is different from , used to generate integer numbers.

After generating the vector , find the maximum and minimum values in the vector and the corresponding indices.

You must create two functions myMin and myMax which return two outputs and receive one input.

[minimum, index\_min] = myMin(x);

[maximum, index\_max] = myMax(x);

Don’t use the predefined functions “max” and “min” ; instead, create your own functions.

**Exercise 6:**

Create a 3 × 5 matrix with random variables. Perform each of the following two ways: using built-in functions, and also using loops (with if statements if necessary):

Find the minimum and maximum values in each column.

Find the minimum and maximum values in each row.

Find the minimum and maximum values in the entire matrix.

**You can upload your reports on our channel on Teams:**

**LAB - 2024/2025 Physical Layer for software radio Files Lab01 student reports**

**Please try to compress the MATLAB files into one file called: Lab01\_yourName**