Khulna University of Engineering & Technology, Khulna

Department of Biomedical Engineering

Course No: **BME 2151**

Name of the Experiment: **MATLAB Fundamentals**

Name: **Md. Fuad Hasan Hamim**

Roll : **2215029**

Year : **2nd**

Term : **1st**

Remarks:

Date of performance: 11 December 2024

Date of submission: 18 December 2024

Objectives:

The main objectives of the experiment are-

1. To be familiar with MATLAB software
2. To learn about the basic features of MATLAB software
3. To know about the commands and functions of MATLAB
4. To know how to operate MATLAB

Introduction:

The full form of MATLAB is MATrix LABoratory. It is a multi multi-paradigm programming language and numeric computing environment developed by MathWorks. MATLAB can perform matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

The MATLAB interface includes-

Editor Window:

The editor window helps us to write and execute the full code. In this window we have to use semicolon (;) at the end of every statements.

Command Window:

MATLAB command window is used to execute commands, write MATLAB scripts and functions, view the output of the commands etc. The window has a command prompt which starts with >>. We can execute our desired commands from this command prompt. It executes program line by line. We don't need to use semicolon (;) at the end of the statement in command window.

Workspace:

Every variable executed in the command window are stored and displayed in the workspace window. MATLAB is an inferred typed language because variables can be assigned without declaring their type, except if they are to treat as symbolic objects and that their type can change. Values can come from constants, from computation involving other variables or from input or output of a function. It (variables stored in workspace) can be used for further uses. For example-

```
>> a = 2

b = 3
```

c = a + b

a =

2

b =

3

c =

5

There are some basic functions using MATLAB software-

Array:

An array is a data structure containing a number of data values (all of which are of same type). It is also mentioned as vector. For instance-

>> x = [2 3 5]

x =

2 3 5

Creating and Manipulating Arrays:

we don't need to mention the size of array rather We can put values of an array in the 'x' variable in the amidst of 2nd [] bracket with a space between one after one entry. Thus, we can create a row vector. For instance,

>> a = [1 2 3 4 5]

a =

1 2 3 4 5

Similarly, we can create a column vector by putting a semicolon (;) one after one entry. For example-

>> a = [1;5]

a =

1

5

We can also make a matrix with the help of creating an array with multiple entries. Here, for the row elements of the matrix we have to put a comma (,) or a space after one element. When total row is completed, we have to put a semicolon (;) to jump to the next row. Thus we can make a matrix of any size we want e.g. –

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

a =

1 2 3

4 5 6

7 8 9

Zeros():

Zero Vector is a special type where all elements are zero either along the row or along the column. To make a zero vector we can command zeros() function mentioning the size of the matrix. If we put a single number in the bracket, then a square matrix of that number will be created. If we want to create a matrix of different number of row and column, then we need to mention the row and column size in the bracket. For instance,

>> zeros(3)

ans =

0 0 0

0 0 0

0 0 0

>> zeros(3,1)

ans =

```
   0
   0
   0
```

```
>> zeros(1,3)
ans =
   0    0    0
```

Ones()

ones() function works Similar way to the zeros() function. For example,

```
>> ones(3)
ans =
   1    1    1
   1    1    1
   1    1    1
```

```
>> ones(3,1)
ans =
   1
   1
   1
```

```
>> ones(1,3)
ans =
   1    1    1
```

Identity Matrix:

Identity matrix is a square matrix in which all the elements of the principal diagonals are one and all other elements are zeros. In MATLAB, identity matrix can be created by eye() functions.

```
>> eye(3)

ans =

   1   0   0

   0   1   0

   0   0   1


>> eye(3,1)

ans =

   1

   0

   0
```

Matrix and Array Operations:

Transpose Matrix: The transpose of a matrix is found by interchanging its rows into columns or columns into rows. In MATLAB, a matrix can be transpose easily by putting an apostrophe (‘) in the variable we mentioned previously e.g.-

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

a =

   1   2   3

   4   5   6

   7   8   9


>> a'

ans =

   1   4   7
```

```
   2    5    8
   3    6    9
```

Inverse Matrix: The inverse of a Matrix is the matrix that on multiplying with the original matrix results in an identity matrix. For any square matrix A, its inverse is denoted as A-1.

In MATLAB, to make an inverse matrix there is a function of inv() e.g.-

```
>> x = [4,3,8;6,2,5;1,5,9]

x =

   4    3    8
   6    2    5
   1    5    9


>> inv(x)

ans =

  -0.1429    0.2653   -0.0204
  -1.0000    0.5714    0.5714
   0.5714   -0.3469   -0.2041
```

Operator:

Multiplication of elements by element e.g.

```
> 2*5

ans =

   10
```

Power of any number (.^) e.g. –

```
>> 2.^4

ans =

   16
```

Multiplication of any number with elements of array (.*)

Division by any number with elements of array (./)

Concatenation of Array:

In MATLAB, concatenation of arrays involves combining arrays either horizontally or vertically.

Horizontal:

In horizontal concatenation, arrays are combined side-by-side along the columns. The number of rows in the arrays must be the same.

```
>> a = [1,3,5;2,4,6;7,8,10]

a =

   1    3    5

   2    4    6

   7    8   10

>> A= [a,a]


A =

   1    3    5    1    3    5

   2    4    6    2    4    6

   7    8   10    7    8   10
```

Vertical:

In vertical concatenation, arrays are combined top-to-bottom along the rows. The number of columns in the arrays must be the same.

```
>> A=[a;a]

A =

   1    3    5

   2    4    6
```

```
    7    8   10
    1    3    5
    2    4    6
    7    8   10
```

Complex Number:

Combination of both the real number and imaginary number is a complex number. MATLAB provides built-in complex number properties e.g.-

```
>> z = 3 + 4i

z =

  3.0000 + 4.0000i

>> i

ans =

  0.0000 + 1.0000i
```

Accessing data in Array:

We can access any entry from a matrix by mentioning the number of entry in the function. For example, there is a 3x3 matrix named A. We want to access the 4^{th} element of A. So, the command would be A(4).

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

A =

    1    2    3
    4    5    6
    7    8    9


>> A(4)

ans =

    2
```

There is another one method to access any entry from the matrix that is by mentioning the row and column number between the brackets.

```
>> A(3,2)

ans =

    8
```

Colon Operator:

The colon operator (:) is one of MATLAB's most versatile tools. It is used for creating ranges, slicing arrays, and specifying all elements in a dimension. This operator enhances code readability and eliminates the need for explicit loops in many cases. Example:

```
>> A = [1,2,3,4;5,6,7,8;9,10,11,12;13,14,15,16]


A =

    1    2    3    4

    5    6    7    8

    9   10   11   12

   13   14   15   16
```

Separating entries from a specific row and specific range of columns e.g.-

```
>> A(1,2:3)

ans =

    2    3
```

Separating entries from any specific row. Here elements of all columns will be considered e.g.-

```
>> A(3,:)

ans =
```

9 10 11 12

Separating entries from any specific column. Here elements of all rows will be considered e.g.-

```
>> A(:,2)

ans =

   2

   6

  10

  14
```

Accessing entries of specific numbers of elements from any specific row e.g. -

```
>> A(3,[2,4])

ans =

  10   12
```

Accessing entries of specific ranges of elements from any specific row e.g. -

```
>> A(3,2:4)

ans =

  10   11   12
```

Regularly spaced Vector:

Start: step: stop

```
>> B = 0:10:100

B =

   0   10   20   30   40   50   60   70   80   90  100

>> C= 0:10

C =

   0   1   2   3   4   5   6   7   8   9   10
```

Conclusion:

In this sessional, we came to know about the basic features, commands and functions of MATLAB. Acctually, we learned to write an array in MATLAB, various manipulations such as finding transpose, inverse, multiplications, concatenation and accessing any data from array or matrix. We got clear conception of using fundamentals of MATLAB which will help us in future sessionals.

References:

1. MATLAB. *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/MATLAB. [Accessed: Dec. 18, 2024].
2. "Array," *Google Search*. [Online]. Available: https://www.google.com/search?q=array&oq=array&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIGCAEQLhhA0gEIMTEzNWowajGoAgCwAgA&sourceid=chrome&ie=UTF-8#vhid=VXWa7lMaEOyNbM&vssid=l. [Accessed: Dec. 18, 2024].
3. GeeksforGeeks, "Inverse of Matrix," *GeeksforGeeks*. [Online]. Available: https://www.geeksforgeeks.org/inverse-of-matrix/. [Accessed: Dec. 18, 2024].
4. BYJU'S, "Types of Matrices," *BYJU'S*. [Online]. Available: https://byjus.com/jee/types-of-matrices/. [Accessed: Dec. 18, 2024].