Unit - 1

MATLAB

Introduction Of MATLAB

MATLAB (matrix laboratory) is a fourth-generation high-level programming language and interactive environment for numerical computation, visualization and programming.

MATLAB is developed by MathWorks.

It allows matrix manipulations; plotting of functions and data; implementation of algorithms; creation of user interfaces; interfacing with programs written in other languages, including C, C++, Java, and FORTRAN; analyze data; develop algorithms; and create models and applications.

It has numerous built-in commands and math functions that help you in mathematical calculations, generating plots, and performing numerical methods.

MATLAB's Power of Computational Mathematics

MATLAB is used in every facet of computational mathematics. Following are some commonly used mathematical calculations where it is used most commonly:

- Dealing with Matrices and Arrays
- 2-D and 3-D Plotting and graphics
- Linear Algebra
- Algebraic Equations
- Non-linear Functions
- Statistics
- Data Analysis

- Calculus and Differential Equations
- Numerical Calculations
- Integration
- Transforms
- Curve Fitting
- Various other special functions

Features of MATLAB

Following are the basic features of MATLAB:

- It is a high-level language for numerical computation, visualization and application development.
- It also provides an interactive environment for iterative exploration, design and problem solving.
- It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
- It provides built-in graphics for visualizing data and tools for creating custom plots.
- MATLAB's programming interface gives development tools for improving code quality, maintainability, and maximizing performance.
- It provides tools for building applications with custom graphical interfaces.
- It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

Uses of MATLAB

MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including:

- signal processing and Communications
- image and video Processing
- control systems
- test and measurement
- computational finance
- computational biology

Input/ Output

Input / Output is possible in various forms in MATLAB. There are numerous functions for sending data to the Standard output device (monitor) and for reading data from the standard input device (keyboard).

In MATLAB, there are three methods to assign a value to a variable:

Using the assignment (=) statement

Read the data from a file stored in the system

Take the value as an input from the user

Input() Function

To accept data from the user, one can use input() function.

```
R = input('Enter the Radius of the circle: ');
```

When this statement is executed, the string 'Enter the Radius of the circle: ' is printed on the screen and the system waits for the user to enter a value. The data accepted is stored in the variable R. Notice, the type of the variable R is not specified. R can take any value valid in MATLAB, it can be character, integer, floating point number, string, or even an empty value (if the user presses enter without entering any value).

Menu() Function

This function is used to give the user an option to select his or her preferred choice from a window. The choice of the user returns a particular scalar value to the program.

To create such a window,

```
X = menu('Title', 'choice – 1', 'choice – 2', ….., 'choice – n');
```

Here, the value of the choice made by the user will be stored in the variable x. When executed, this command will create a menu and the user shall be able to enter his or her choice by mouse or by keyboard.

Output functions

Any non-integer value in MATLAB is displayed up to four digits after decimal point. For example, if we give

```
X = 4
Y = 200.0201
```

The output will be –

```
X = 4
Y = 200.02
```

Types of graphs

Matlab can be used to create different types of graphs. Pay careful attention to the coordinate

axes and ranges that you see for each command used.

There are following types of Graphs

2D Graphs

3D Graphs

Array

All variables of all data types in MATLAB are multidimensional arrays. A vector is

a one-dimensional array and a matrix is a two-dimensional array.

Multidimensional Arrays

An array having more than two dimensions is called a multidimensional array in

MATLAB. Multidimensional arrays in MATLAB are an extension of the normal twodimensional matrix.

Generally to generate a multidimensional array, we first create a two-dimensional

array and extend it.

Function

A function is a group of statements that together perform a task. In MATLAB, functions are defined in separate files. The name of the file and of the function should be the same. Functions operate on variables within their own workspace, which is also called the local workspace, separate from the workspace you access at the MATLAB command prompt which is called the base workspace.

Functions can accept more than one input arguments and may return more than one output arguments

Syntax of a function statement is:

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

Anonymous Functions

An anonymous function is like an inline function in traditional programming languages, defined within a single MATLAB statement. It consists of a single MATLAB expression and any number of input and output arguments. You can define an anonymous function right at the MATLAB command line or within a function or script.

This way you can create simple functions without having to create a file for them.

The syntax for creating an anonymous function from an expression is

```
f = @(arglist)expression
```

Primary and Sub-Functions

Any function other than an anonymous function must be defined within a file. Each function file contains a required primary function that appears first and any number of optional sub-functions that comes after the primary function and used by it.

Nested Functions

You can define functions within the body of another function. These are called nested functions. A nested function contains any or all of the components of any other function.

Nested functions are defined within the scope of another function and they share access to the containing function's workspace.

A nested function follows the below syntax:

```
function x = A(p1, p2)
```

```
...
B(p2)
 function y = B(p3)
 ...
 end
...
end
```

Private Functions

A private function is a primary function that is visible only to a limited group of other functions. If you do not want to expose the implementation of a function(s), you can create them as private functions. Private functions reside in subfolders with the special name private. They are visible only to functions in the parent folder.

Loops

There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially. The first statement in a function is executed first, followed by the second, and so on. Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times.

MATLAB provides following types of loops to handle looping requirements. Click

the following links to check their detail:

Loop Type	**Description**
while loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
for loop	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
nested loops	You can use one or more loops inside any another loop.

The while Loop

The while loop repeatedly executes statements while condition is true.

Syntax

The syntax of a while loop in MATLAB is:

```
while <expression>
  <statements>
end
```

The while loop repeatedly executes program statement(s) as long as the expression remains true.

The for Loop

A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax

The syntax of a for loop in MATLAB is:

```
for index = values
<program statements>
 ...
end
```

The Nested Loops

MATLAB allows to use one loop inside another loop. Following section shows few examples to illustrate the concept.

Syntax

The syntax for a nested for loop statement in MATLAB is as follows:

```
for m = 1:j
 for n = 1:k
 <statements>;
 end
```

Structures

Structures are data structures that group together value that are logically related in what are called fields of the structure. An advantage of structure is that field are name which help to make it clear what value are stored in the structure . however, structure variable are not arrays. they do not have elements that are indexed. so it is not possible to loop through the value in a structure or to be use vectorized code.

Simulink

Simulink is a simulation and model-based design environment for dynamic and embedded systems, integrated with MATLAB. Simulink, also developed by MathWorks, is a data flow graphical programming language tool for modeling, simulating and analyzing multi-domain dynamic systems. It is basically a graphical block diagramming tool with customizable set of block libraries.It allows you to incorporate MATLAB algorithms into models as well as export the

simulation results into MATLAB for further analysis.

Simulink supports:

- system-level design
- simulation
- automatic code generation
- testing and verification of embedded systems

Simulink is capable of systematic verification and validation of models through modelling style checking, requirements traceability and model coverage analysis. Simulink Design Verifier allows you to identify design errors and to generate test case scenarios for model checking.

Using Simulink

To open Simulink, type in the MATLAB work space:

simulink

Unit – 2

LabVIEW

Introduction of LabVIEW :-

LabVIEW is developed and produced by National Instruments as an environment used for graphical system design.

The name LabVIEW is a shortened form of its description: Laboratory Virtual Instrument Engineering Workbench.

LabVIEW is a visual programming language: it is a system-design platform and development environment that was aimed at enabling all forms of system to be developed.

LabVIEW was developed by National Instruments as a workbench for controlling test instrumentation. However its applications have spread well beyond just test instrumentation to the whole field of system design and operation.

Creation of VIs

You can create custom template VIs to avoid adding the same components to the block diagram or the front panel each time you want to perform a similar operation.

Complete the following steps to create a new template based on an existing VI template.

- **Select File»New to display the New dialog box.**
- **From the Create New list, select a template VI from the From Template directory similar to the one you want to create.**
- **Click the OK button. The front panel and block diagram of the template VI**

appear.

.

Edit the VI.

- **Save the VI as a template.**
- **Select File»Save As to display a file dialog box.**
- **(Optional) Navigate to the `labview\templates` directory if you want the template VI to appear in the New dialog box with the template VIs LabVIEW installs. To set the block diagram preview, save an image of the block diagram as a `.png` file in the `labview\templates` directory. Append `d` to the name of the VI to create the `.png` filename. For example, if you want to create the block diagram preview of `Test.vi`, save an image of the block diagram as `Testd.png` in the `labview\templates` directory. This block diagram preview and the VI description appear in the Description section when the VI template is selected in the Create New list.**
- **(Windows) Select Template VIs from the Save as type pull-down menu. (macOS and Linux) Include the template VI file extension (`.vit`).**
- **Enter the name of the template VI and click the OK button.**

Creation of Sub-Vis

A subVI is simply a VI used in (or called by) another VI. A subVI node (comprised of icon/connector in a calling VI block diagram) is analogous to a subroutine call in a main program.

Creating a SubVI from a VI

- **Moving to the programming part, create a VI as we have done in 1st tutorial and save it for future use, as we have done in all previous tutorials.**
- **To understand the use of sub VIs, what we have to do is to first design a program (bigger one) which we want to convert into modules or sub VIs. To save our time lets use the VI we have already created in the graphs and charts tutorial. The one in which temperature is converted from Fahrenheit to Celsius scale, as shown in the figure below,**
- **From the top bar, when you click on the edit button a dropdown will appear.**
- **elect the area of the code or block diagram from the complete VI that you**

want to put inside the subVI, for reference.

Structure

A structure is defined as a graphical representation of a loop (i.e., a loop is nothing but a set of code blocks that are executed based on the condition match). In reality, structures have control over the execution flow within a Virtual Instrument (VI).

Execution structures contain sections of graphical code and control how and when the code inside is run. The most common execution structures are While Loops, For Loops and Case structures which you can use to run the same section of code multiple times or to execute a different section of code based on some condition.

There are different types of structures available within LabVIEW. We will discuss the types in detail.

Where can we find the structure in LabVIEW?

- To Access a structure, the developer will have to go through the following steps:
- To access Functions, the developer has open the Front Panel.
- Click on “View” module in the front panel and select “Functions”.
- From “functions”, select “programming” option and look for “Structure” option.
- Clicking on “Structures” will give you in detail options.

we will discuss the various structures that are available within LabVIEW. The list is as follows:

- While Loop structure
- For Loop structure
- Sequence structure

- **Flat sequence structure**
- **Stacked sequence structure**
- **Event structure**
- **Timed structure**
- **Diagram disable structure**
- **Conditional disable structure**

.

Array

A group of homogeneous elements of a specific data type is known as an ARRAY, one of the simplest data structures. Arrays hold a sequence of data elements, usually of the same size and same data type placed in contiguous memory locations that can be individually referenced. Hence arrays are essentially a way to store many values under the same name. Individual elements are accessed by their position in the array. The position is given by an index, which is also called a subscript. The index usually uses a consecutive range of integers. Some arrays are multi-dimensional, but generally, one-and two-dimensional arrays are the most common.

Arrays In Labview

In LABVIEW arrays group data elements of the same type. They are analogous to arrays in traditional languages. An array consists of elements and dimensions. Elements are the data that make up an array. A dimension is the length, height or depth of an array. An   array can have one or more dimensions and as many as (231) – 1 elements per dimension, memory permitting.

You can build arrays of numeric, Boolean, path, string and cluster data types. You cannot create arrays of arrays. However, you can use a multidimensional array or an array of clusters where each cluster contains one or more arrays. Also, you cannot create an array of subpanel controls, tab controls, .NET controls, ActiveX controls, charts or multiplot XY graphs.

Cluster

CLUSTERS group data elements of mixed types. An example of a cluster is the LABVIEW error cluster, which combines a Boolean value, a numeric value and a string. A cluster is similar to a record or a struct in text-based programming languages. The below Figure shows the error cluster control and the corresponding terminal created in the BLOCK DIAGRAM. This cluster consists of a Boolean control (status), a numeric control (code) and a string control (source).

The main cluster operations are bundle, unbundled, bundle by name and unbundle by name. Use the cluster functions to create and manipulate clusters. For example, you can perform tasks similar to the following:

- **Extract individual data elements from a cluster.**
- **Add individual data elements to a cluster.**
- **Break a cluster out into its individual data elements.**

The Bundle function assembles individual components into a single new cluster and allows you to replace elements in an existing order. The Unbundled function splits a cluster into its individual components. When it is required to operate on a few elements and not the entire cluster elements, you use the Bundle By Name function. They are referenced by names rather than by position. The Unbundle By Name function returns the cluster elements whose names are specified.

LabVIEW Graphs and Charts

Graphs and charts are one of the most essential tools to represent data, and they are making trends in different aspects. They are widely used in various sectors to convey the information visually and to make sense of data

A chart is a graphical representation of data in symbols whereas graphs portray the relationship between different data in less space

create a graph and chart program in LabVIEW.

Step 1: Launch LabVIEW platform. Open-File Menu and click on the ‘New’ option to build a new Virtual Instrument (VI). A VI has two windows; window comprising the front panel view with inputs and outputs and block diagram which contains the code.

For example, create a program for changing the temperature from Fahrenheit to Celsius and display them as a graph in LabVIEW.

Step 2: Right-click on the front panel - Navigate to Control Palette – Select Numeric type – Click on Vertical Slide Bar as shown in the image.

Step 3: Use the formula Tc=5/9(Tf-32) to convert temperature from Fahrenheit to Celsius Scale. This formula includes various operations in it.

So, connect the output of Temperature slider to the upper input of the subtraction block which is present in the function palette in the numeric section and set value 32 to another side of subtraction block.

Step 4: In the same section, select the division block and set numeric constant value 5 at the upper end whereas value 9 at the lower input.

Step 5: In multiplier block, connect the output from subtraction block and division block as shown in the below image. The multiply block gives the output of temperature value in Celsius.

Step 6: Right-click on Front Panel – Select numeric from the control palette – Click on Thermometer. Connect thermometer with the output of the multiply block and name them

Step 7: Attach a numeric indicator at the temperature at both input temperature and temperature block.

Step 8: You can change the scale of fill as follows: Select scale of fill – Right Click on Vertical Slide fill – Select Properties – Select Scale – Set Maximum value as per your need.

Step 9: Click on the Run button or Press (Ctrl+R) to run the program. Now, you have created only the bar graphs.

Step 10: Navigate to charts and graph section in LabVIEW. Select graphs from the control palette and choose Waveform Charts. Join the block diagram of the process with the result of the multiply block. Run the VI continuously and change the value of temperature in the Slide fill to see the change in variation of data in the chart .

Step 11: Move to the graph section and set the data of any type about a period of time to see the graph. Make use of ‘for’ loop to iterate the value and insert the wait block in it. Connect a wire from output of the multiply block to the edge of ‘for’ loop. In the control palette portion, select waveform graphs .

Step 12: Join the block of graphs to the edge of the loop as shown in the image. Then run the program as said above where you can see the variation in the chart and but nothing in graphs. When you stop the program of iteration gets completed you can view the data variations in the graph.

String

A string is a sequence of displayable or non-displayable ASCII characters. Strings provide a platform-independent format for information and data. Some of the more common applications of strings include the following:

Creating simple text messages.

Controlling instruments by sending text commands to the instrument and returning data values in the form of either ASCII or binary strings which you then convert to numeric values.

Storing numeric data to disk. To store numeric data in an ASCII file, you must first convert numeric data to strings before writing the data to a disk file.

Instructing or prompting the user with dialog boxes.

On the front panel, strings appear as tables, text entry boxes, and labels. LabVIEW includes built-in VIs and functions you can use to manipulate strings, including formatting strings, parsing strings, and other editing.

File I/O

A typical file I/O operation involves the following process.

- **Create or open a file. After the file opens, a unique identifier called a refnum represents the file.**
- **Use a VI or function from the File I/O palette to read from or write to the file.**
- **Close the file.**

File I/O VIs and some File I/O functions, such as the Read from Text File and Write to Text File functions, can perform all three steps for typical file I/O operations. The VIs and functions designed for multiple operations might not be as efficient as the functions that perform individual operations.