**SEMINAR REPORT**

**ON**

**INTRODUCTION**

**TO**

**MATLAB**

**SUBMITTED BY: - DIVYA GUPTA|CSA 3rd SEM |ROLL NO 48**

**SUBMITTED TO: -MR.AMIT GUPTA**

# Graphic_Era_Hill_University_LogoCertificate

# This is to certify that SEMINAR REPORT on INTRODUCTION TO MATLAB which is submitted by DIVYA GUPTA in partial fulfilment of the requirement for the award of degree B.Tech. in Computer Science Engineering to Graphic Era Hill University, Dehradun is a record of the candidate’s own work carried out by her under my/our supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

# Date:                                                        Supervisor:

# Graphic_Era_Hill_University_Logo

# Acknowledgement

# I thank my teacher Mr Amit Gupta for his continued support and encouragement. I offer my sincere appreciation for the learning opportunities provided by my committee.

# My completion of this project could not have been accomplished without the support of my classmates.

# Finally, to my parents , my heartfelt thanks.

# 

# Graphic_Era_Hill_University_Logo INDEX

|  |  |
| --- | --- |
| No | TOPIC |
| 1. | INTRODUCTION |
| 2. | HISTORY |
| 3. | KEY FEATURES |
| 4. | NUMERIC COMPUTATION |
| 5. | THE MATLAB LANGUAGE |
| 6. | DEVELOPMENT TOOLS |
| 7. | DATA ANALYSIS AND VISUALIZATION |
| 8. | WHY MATLAB ? |
| 9. | ADVANTAGES |
| 10. | OPERATOR PRECEDENCE |
| 11. | VARIABLES |
| 12. | MATRICES IN MATLAB |
| 13. | MATLAB DEBUGGER |
| 14. | CONCLUSION |

|  |  |
| --- | --- |
|  |  |

INTRODUCTION

**MATLAB** (**mat**rix **lab**oratory) is a multi-paradigm numerical computing environment and fourth-generation programming language. A proprietary programming language developed by MathWorks. It was designed by Cleve Moller . MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, FORTRAN and Python.

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

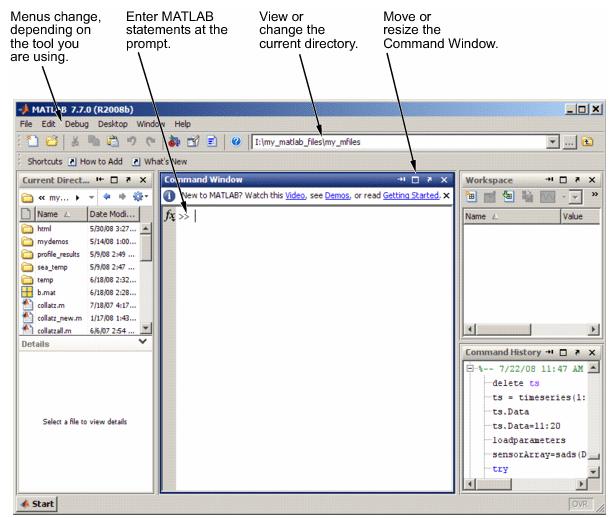
In 2004, MATLAB had around one million users across industry and academia. MATLAB users come from various backgrounds of engineering, science, and economics.

Developed by Mathworks. Its initial release was 31 years ago in 1984.Development status is still active. It is written in C, C++, Java, and MATLAB.

It can work on Microsoft platforms, OS Linux. It was mainly built for computing numerical problems. It is proprietary commercial software.

The MATLAB high-performance language for technical computing integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include

* Math and computation
* Algorithm development
* Data acquisition
* Modeling, simulation, and prototyping
* Data analysis, exploration, and visualization
* Scientific and engineering graphics
* Application development, including graphical user interface building
* When you start MATLAB, the desktop appears, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB.

The following illustration shows the default desktop..

**HISTORY**

Cleve Moller, the chairman of the computer science department at the University of New Mexico, started developing MATLAB in the late 1970s. He designed it to give his students access to LINPACK and EISPACK without them having to learn Fortran. It soon spread to other universities and found a strong audience within the applied mathematics community. Jack Little, an engineer, was exposed to it during a visit Moler made to Stanford University in 1983. Recognizing its commercial potential, he joined with Moler and Steve Bangert. They rewrote MATLAB in C and founded MathWorks in 1984 to continue its development. These rewritten libraries were known as JACKPAC. In 2000, MATLAB was rewritten to use a newer set of libraries for matrix manipulation, LAPACK.

MATLAB was first adopted by researchers and practitioners in control engineering, Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra, numerical analysis, and is popular amongst scientists involved in image processing.

**KEY FEATURES**

* High-level language for [numerical computation](http://in.mathworks.com/products/matlab/features.html#numeric_computation), [visualization](http://in.mathworks.com/products/matlab/features.html#data_analysis), and [application development](http://in.mathworks.com/products/matlab/features.html#application_development)
* Interactive environment for iterative exploration, design, and problem solving
* Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations
* Built-in graphics for visualizing data and tools for creating custom plots
* Development tools for improving code quality and maintainability and maximizing performance
* Tools for building applications with custom graphical interfaces
* Functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET, and Microsoft Excel.

**Numeric Computation**

MATLAB provides a range of numerical computation methods for analyzing data, developing algorithms, and creating models. The MATLAB language includes mathematical functions that support common engineering and[science operations](http://in.mathworks.com/discovery/scientific-computing.html). Core math functions use processor-optimized libraries to provide fast execution of vector and matrix calculations.

Available methods include:

* Interpolation and regression
* Differentiation and integration
* Linear systems of equations
* Fourier analysis
* Eigenvalues and singular values
* Ordinary differential equations (ODEs)
* Sparse matrices

MATLAB add-on products provide functions in specialized areas such as statistics, optimization, signal analysis, and machine learning.

### The MATLAB Language

The MATLAB language provides native support for the vector and matrix operations that are fundamental to solving engineering and scientific problems, enabling fast development and execution.

With the MATLAB language, you can write programs and develop algorithms faster than with traditional languages because you do not need to perform low-level administrative tasks such as declaring variables, specifying data types, and allocating memory. In many cases, the support for vector and matrix operations eliminates the need for for-loops. As a result, one line of MATLAB code can often replace several lines of C or C++ code.

MATLAB provides features of traditional programming languages, including flow control, error handling, and object-oriented programming (OOP). You can use fundamental data types or advanced data structures, or you can define custom data types.

You can produce immediate results by interactively executing commands one at a time. This approach lets you quickly explore multiple options and iterate to an optimal solution. You can capture interactive steps as scripts and functions to reuse and automate your work.

MATLAB add-on products provide built-in algorithms for signal processing and communications, image and video processing, control systems, and many other domains. By combining these algorithms with your own, you can build complex programs and applications.

Development Tools

MATLAB includes a variety of tools for efficient algorithm development, including:

* **Command Window** - Lets you interactively enter data, execute commands and programs, and display results
* **MATLAB Editor** - Provides editing and debugging features, such as setting breakpoints and stepping through individual lines of code
* **Code Analyzer** - Automatically checks code for problems and recommends modifications to maximize performance and maintainability
* **MATLAB Profiler** - Measures performance of MATLAB programs and identifies areas of code to modify for improvement

Additional tools compare code and data files, and provide reports showing file dependencies, annotated reminders, and code coverage.

## Data Analysis and Visualization

MATLAB provides tools to acquire, analyze, and visualize data, enabling you to gain insight into your data in a fraction of the time it would take using spreadsheets or traditional programming languages. You can also document and share your results through plots and reports or as published MATLAB code.

**Acquiring Data**

MATLAB lets you access data from files, other applications, databases, and external devices. You can read data from popular file formats such as Microsoft Excel; text or binary files; image, sound, and video files; and scientific files such as netCDF and HDF. File I/O functions let you work with data files in any format.

Using MATLAB with add-on products, you can acquire data from hardware devices, such as your computer's serial port or sound card, as well as stream live, measured data directly into MATLAB for analysis and visualization. You can also communicate with instruments such as oscilloscopes, function generators, and signal analyzers.

**Analyzing Data**

MATLAB lets you manage, filter, and preprocess your data. You can perform exploratory data analysis to uncover trends, test assumptions, and build descriptive models. MATLAB provides functions for filtering and smoothing, interpolation, convolution, and fast Fourier transforms (FFTs). Add-on products provide capabilities for curve and surface fitting, multivariate statistics, spectral analysis, image analysis, system identification, and other analysis tasks.

**Visualizing Data**

MATLAB provides built-in 2-D and 3-D plotting functions, as well as volume visualization functions. You can use these functions to visualize and understand data and communicate results. Plots can be customized either interactively or programmatically.

The MATLAB plot gallery provides examples of many ways to display data graphically in MATLAB.

**WHY MATLAB?**

MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide. It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

**ADVANTAGES**

MATLAB has several advantages over other methods or languages:

* Its basic data element is the matrix. A simple integer is considered an matrix of one row and one column.  Several mathematical operations that work on arrays or matrices are built-in to the Matlab environment. For example, cross-products, dot-products, determinants, inverse matrices.
* Vectorized operations. Adding two arrays together needs only one command, instead of a for or while loop.
* The graphical output is optimized for interaction. You can plot your data very easily, and then change colors, sizes, scales, etc, by using the graphical interactive tools.
* Matlab’s functionality can be greatly expanded by the addition of toolboxes. These are sets of specific functions that provided more specialized functionality. Ex: Excel link allows data to be written in a format recognized by Excel, Statistics Toolbox allows more specialized statistical manipulation of data (Anova, Basic Fits, etc)

**OPERATOR PRECEDENCE**

You can build expressions that use any combination of arithmetic, relational, and logical operators. Precedence levels determine the order in which MATLAB evaluates an expression. Within each precedence level, operators have equal precedence and are evaluated from left to right. The precedence rules for MATLAB operators are shown in this list, ordered from highest precedence level to lowest precedence level:

1. Parentheses ()
2. Transpose (.'), power (.^), complex conjugate transpose ('), matrix power (^)
3. Unary plus (+), unary minus (-), logical negation (~)
4. Multiplication (.\*), right division (./), left division (.\), matrix multiplication (\*), matrix right division (/), matrix left division (\)
5. Addition (+), subtraction (-)
6. Colon operator (:)
7. Less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), equal to (==), not equal to (~=)
8. Element-wise AND (&)
9. Element-wise OR (|)
10. Short-circuit AND (&&)
11. Short-circuit OR (||)

## The Colon Operator

The colon, :, is one of the most important MATLAB operators.It is considered as a horse ride in MATLAB. It occurs in several different forms.

The expression 1:10 is a row vector containing the integers from 1 to 10:

1 2 3 4 5 6 7 8 9 10

To obtain non unit spacing, specify an increment. For example,

100:-7:50

Is:

100 93 86 79 72 65 58 51

(ranging from 100 to 50 subtracting 7 at each time.)

## Variables

Like most other programming languages, the MATLAB language provides mathematical expressions, but unlike most programming languages, these expressions involve entire matrices.

MATLAB does not require any type declarations or dimension statements. When MATLAB encounters a new variable name, it automatically creates the variable and allocates the appropriate amount of storage. If the variable already exists, MATLAB changes its contents and, if necessary, allocates new storage. For example,num\_students = 25 creates a 1-by-1 matrix named num\_students and stores the value 25 in its single element. To view the matrix assigned to any variable, simply enter the variable name.

Variable names consist of a letter, followed by any number of letters, digits, or underscores. MATLAB is case sensitive; it distinguishes between uppercase and lowercase letters. A and a are not the same variable.

**MATRICES IN MATLAB**

You can enter matrices into MATLAB in several different ways:

* Enter an explicit list of elements.
* Load matrices from external data files.
* Generate matrices using built-in functions.
* Create matrices with your own functions in M-files.

Use these conventions to create a Matrix:

* Separate the elements of a row with blanks or commas.
* Use a semicolon, ; , to indicate the end of each row.
* Surround the entire list of elements with square brackets, [ ]

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

This is what MATLAB displays after you hit <enter>

A =

    16     3     2    13

     5    10    11     8

     9     6     7    12

     4    15    14     1

Let’s prove it is a magic square. Let’s get the sum of all columns by typing sum(A)

The answer (ans) is:

ans =

    34    34    34    34

This result is a row vector. Each column in A adds up to 34. That’s magic!

How about the row sums? MATLAB has a preference for working with the columns of a matrix, so the easiest way to get the row sums is to transpose the matrix, compute the column sums of the transpose, and then transpose the result. The transpose operation is denoted by an apostrophe or single quote, '. It flips a matrix about its main diagonal and it turns a row vector into a column vector.

A'

ans =

    16     5     9     4

     3    10     6    15

     2    11     7    14

    13     8    12     1

Now:      **sum(A')'**    produces a column vector containing the row sums

ans =

    34

    34

    34

    34

The sum of the elements on the main diagonal is easily obtained with the help of the diag function, which picks off that diagonal.

**diag(A)**

ans =

    16

    10

     7

     1

**sum(diag(A))**

ans =

    34

The other diagonal, the so-called anti-diagonal, is not so important mathematically, so MATLAB does not have a ready-made function for it. But a function originally intended for use in graphics, fliplr, flips a matrix from left to right.

**sum(diag(fliplr(A))**

ans =

    34

A(1,4) + A(2,4) + A(3,4) + A(4,4)

ans =

     34

The most effective way to perform this operation is using the ‘:’ operator, one of Matlab’s workhorses :

**sum(A(:,4))**

ans =

     34

It reads as “add every element in column 4”

If you want to see these elements, simply type

A(:,4)

**MATLAB DEBUGGER**

Programs can run very quickly, and examining variables as they fly by during real-time execution can be difficult. A more sophisticated way of examining variables during execution is to set "breakpoints" in a program. The computer will pause at these breakpoints during execution of the code to allow you to inspect the values of variables.

MATLAB's editor allows you to set such breakpoints, as shown below, by clicking the mouse to the left of the line of interest. They are indicated by a red dot at the left side of the window.

When MATLAB is in debugging mode, the command line prompt will change from '>>' to 'K>>'. When the program has paused at a breakpoint, a green arrow appears in the editor window indicating the breakpoint location, and by extention, where execution will continue when the user so commands.

During a breakpoint pause, the values held in variables will appear superimposed on the Editor window when the cursor is passed over the variable.

The MATLAB editor draws a distinction between *running* a function and *continuing* it. The continue option will resume execution of a function that has paused at a breakpoint.

Most debugging commands can be activated through either the editor or the command line.

You also have the option of stepping through the program line by line.

More debugging features are available via the command line (see 'help debug' for information.) A subset:

|  |  |
| --- | --- |
| Dbstop | Set a debugging beakpoint at a certain line. |
| dbclear | Remove a breakpoint. |
| Dbcont | Resume execution from most recent breakpoint. |
| Dbstack | List which function called which. |

You can also make the breakpoint *conditional*, such as having the function break if a function produces a MATLAB warning.

**CONCLUSION**

Matlab is not only a programming language, but a programming environment as well.

You can perform operations from the command line, as a sophisticated calculator.

Or you can create programs and functions that perform repetitive tasks, just as any other computer language.

According to me, I felt it somewhat similar like shell programming in various ways:

\* It has a command window just as we have our shell prompt in linux/unix.

\* It has an editor to type functions or to make files as we have in VI editor.

The difference is just that it is specially designed for matrix manipulations.

Also, linux/unix is an operating system while it is just a programming language.

There are also disadvantages:

* It uses a large amount of memory and on slow computers it is very hard to use.
* It sits “on top” of Windows, getting as much CPU time as Windows allows it to have. This makes real-time applications very complicated.

 You can use MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology.