American International University- Bangladesh (AIUB) Faculty of Engineering (FE)

|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | DATA COMMUNICATION | **Course Code:** | COE3103 |
| **Semester:** | Summer 2024-25 | **Section:** | D |
| **Faculty:** | MOHAMMAD ASADUZZAMAN KHAN | **Group:** | 05 |

|  |  |
| --- | --- |
| **Experiment No:** | 01 |
| **Experiment Name:** | Introduction to MATLAB |

|  |  |  |  |
| --- | --- | --- | --- |
| **Group Members** | | **Name** | **ID** |
|  | 1. | Basudeb Kundu | 23-50856-1 |
| 2. | Prohlad Chandra Das | 23-50922-1 |
| 3. | Debashis Kumar Das | 23-50953-1 |
| 4. | Indronill Dutta Nill | 23-50974-1 |
| 5. | Nafiur Rahman Nirob | 23-50991-1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Performance Date:** | 16-7-25 | **Due Date:** | 27-7-25 |

**Marking Rubrics (to be filled by Lab Instructor)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Category | Proficient [6] | Good [4] | Acceptable [2] | Unacceptable [1] | Secured Marks |
| **Theoretical Background, Methods & procedures sections** | All information, measures and variables are provided and  explained. | All Information provided is sufficient, but more explanation is  needed. | Most information is correct, but some information may be  missing or inaccurate. | Much information is missing and/or inaccurate. |  |
| **Results** | All of the criteria are met; results are described clearly and accurately; | Most criteria are met, but there may be some lack of clarity and/or incorrect information. | Experimental results don’t match exactly with the theoretical values and/or analysis  is unclear. | Experimental results are missing or incorrect; |  |
| **Discussion** | Demonstrates thorough and sophisticated understanding.  Conclusions drawn are appropriate for  analyses; | Hypotheses are clearly stated, but some concluding statements not supported by data or data not well  integrated. | Some hypotheses missing or misstated; conclusions not supported by data. | Conclusions don’t match hypotheses, not supported by data; no integration of data from different sources. |  |
| **General formatting** | Title page, placement of figures and figure captions, and other  formatting issues all correct. | Minor errors in formatting. | Major errors and/or missing information. | Not proper style in text. |  |
| **Writing & organization** | Writing is strong and easy to understand; ideas are fully elaborated and connected; effective transitions between sentences; no  typographic, spelling, or grammatical errors. | Writing is clear and easy to understand; ideas are connected; effective transitions between sentences; minor typographic, spelling, or grammatical errors. | Most of the required criteria are met, but some lack of clarity, typographic, spelling, or grammatical errors are present. | Very unclear, many errors. |  |
| Comments: |  | | | Total Marks (Out of **30**): |  |

**Title: Introduction to MATLAB**

**Abstract:**

This experiment is designed to-

1. To understand the use of MATLAB for solving communication engineering problems.

2.To develop understanding of MATLAB environment, commands and syntax.

**Introduction:**

Matlab is a high-performance language for technical computing. It integrates computation, programming and visualization in a user-friendly environment where problems and solutions are expressed in an easy-to-understand mathematical notation.

Matlab is an interactive system whose basic data element is an array that does not require dimensioning. This allows the user to solve many technical computing problems, especially those with matrix and vector operations, in less time than it would take to write a program in a scalar noninteractive language such as C or Fortran.

Matlab features a family of application-specific solutions which are called toolboxes. It is very important to most users of Matlab, that toolboxes allow to learn and apply specialized technology. These toolboxes are comprehensive collections of Matlab functions, so-called M files, that extend the Matlab environment to solve particular classes of problems.

Matlab is a matrix-based programming tool. Although matrices often need not to be dimensioned explicitly, the user has always to look carefully for matrix dimensions. If it is not defined otherwise, the standard matrix exhibits two dimensions’ n × m. Column vectors and row vectors are represented consistently by n × 1 and 1 × n matrices, respectively.

**Getting Started:**

a) Go to the start button, then programs, MATLAB and then start MATLAB. It is preferred that you have MATLAB2016a. You can then start MATLAB by double clicking on its icon on Desktop, if there is any.

b) The Prompt:

**>>**

The operator shows above is the prompt in MATLAB. MATLAB is interactive language like C, Java etc. We can write the commands over here.

c) In MATLAB we can see our previous commands and instructions by pressing the up key. Press the key once to see the previous entry, twice to see the entry before that and so on. We can also edit the text by using forward and back-word keys.

**Entering Matrices and Addressing the Elements**

The elements of a matrix must be entered one-by-one in a list where the elements of a row  
be separated with commas or blank spaces and the rows are divided by semicolons.

The whole list must be surrounded with square brackets, e.g.

|  |
| --- |
| >> A = [1 2 3; 8 6 4; 3 6 9] |

After pressing “**Enter**” Matlab displays the numbers entered in the command line

**A = A screenshot of a computer

AI-generated content may be incorrect.**

Addressing an element of a matrix is also very easy. The n-th element of the m-th column in matrix A from above is A(n,m). So typing

|  |
| --- |
| >> A(1,3) + A(2,1) + A(3,2) |

will compute the answer

**ans = A white background with black text

AI-generated content may be incorrect.**

Further examples:

|  |
| --- |
| >> A(1,1:2) |

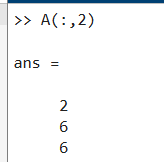
addresses the first two elements of the first row.

**ans = A screenshot of a computer

AI-generated content may be incorrect.**

|  |
| --- |
| >> A(:,2) |

addresses all elements of the second column.

**ans = **

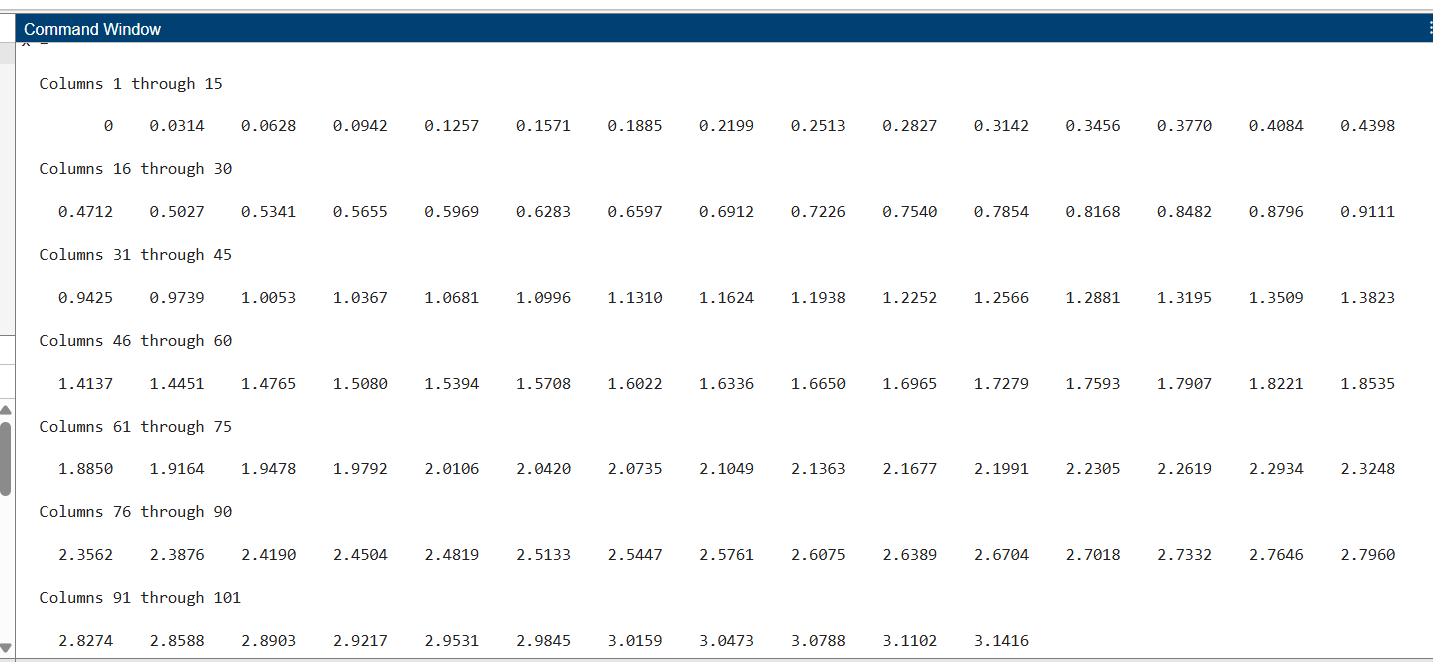
**Generating Matrices**

There are different ways to generate matrices. Assigning elements explicitly was presented in the paragraph above. To create a row vector with 101 equidistant values starting at 0 and ending by π, this method would be very tedious. So two other possibilities are shown below:

|  |
| --- |
| >> x = linspace(0,pi,101) |

Or

|  |
| --- |
| >> x = (0:0.01:1)\*pi |



In the first case, the Matlab function linspace is used to create x. The function’s arguments are described by:

**linspace (first value, last value, number of values)** with the default number of values = 100.

In the second case, the colon notation (0:0.01:1) creates an array that starts at 0, increments by 0.01 and ends at 1. Afterwards each element in this array is multiplied by π to create the desired values in x. Both of these array creation forms are common in Matlab. While the colon notation form allows to specify the increment between data elements directly, but not the number of data elements, the Matlab function linspace allows to specify the number of data elements directly, but not the increment value between these data elements. The colon notation is very often used in Matlab, therefore a closer look should be taken on it. (first value:increment:last value) creates an array starting at first value, ending at last value with an increment which can be negative as well, e.g.

|  |
| --- |
| >> v = (10:-2:0) |

**v = A white background with black numbers

AI-generated content may be incorrect.**

If the increment is 1, then its usage is optional:

|  |
| --- |
| >> w = (5:10) |

**w = A number and a circle

AI-generated content may be incorrect.**

Matlab also provides four functions that generate basic matrices: zeros, ones, rand and randn. Some examples:

|  |
| --- |
| >> B = zeros(3,4) |

**B = A number of numbers on a white background

AI-generated content may be incorrect.**

|  |
| --- |
| >> C = ones(2,5)\*6 |

**C = A number on a white background

AI-generated content may be incorrect.**

|  |
| --- |
| >> D = rand(1,5) |

generates uniformly distributed random elements

**D = A number and numbers on a white background

AI-generated content may be incorrect.**

|  |
| --- |
| >> E = randn(3,3) |

generates normally -also called Gaussian- distributed random elements

**E = A screenshot of a computer

AI-generated content may be incorrect.**

**Deleting rows and columns**

To delete rows or columns of a matrix, just use a pair of square brackets, e.g.

|  |
| --- |
| >> A(2,:) = [ ] |

deletes the second row of A.

**A =**

It is not possible to delete a single element of a matrix, because afterwards it would not still be a matrix. (Exception: vectors, since here deleting an element is the same as deleting a row/column.)

**Array Orientation**

The orientation of an array can be changed with the Matlab transpose operator’:

|  |
| --- |
| >> a = 0:3 |

**a = A screenshot of a computer

AI-generated content may be incorrect.**

|  |
| --- |
| >> b = a’ |

**b =**

**Scalar-Array Mathematics**

Addition, subtraction, multiplication and division by a scalar apply the operation to all elements of the array:

|  |
| --- |
| >> c = [1 2 3 4;5 6 7 8;9 10 11 12] |

c = A white background with numbers

AI-generated content may be incorrect.

|  |
| --- |
| >> 2\*c-1 |

multiplies each element in c by two and subtracts one from each element of the result.

**ans = A white background with black numbers

AI-generated content may be incorrect.**

**Array-Array Mathematics**

When two arrays have the same dimensions, which means that they have the same number of rows and columns, addition, subtraction, multiplication and division apply on an element-by-element basis in Matlab.

|  |
| --- |
| >> d = [1 2 3; 4 5 6] |

**d = A number and equation on a white background

AI-generated content may be incorrect.**

|  |
| --- |
| >> e = [2 2 2; 3 3 3] |

**e = A white background with black numbers

AI-generated content may be incorrect.**

|  |
| --- |
| >> f = d+e |

adds d to e on an element-by-element basis

**f = A white background with black numbers and symbols

AI-generated content may be incorrect.**

|  |
| --- |
| >> g = 2\*d-e |

multiplies d by two and subtracts e from the result

**g = A screenshot of a math problem

AI-generated content may be incorrect.**

Element-by-element multiplication and division work similarly, but the notation is slightly different:

|  |
| --- |
| >> h = d.\*e |

**h = A white background with black text

AI-generated content may be incorrect.**

The element-by-element multiplication uses the dot multiplication symbol .\*, the elementby element array division uses either ./ or .\

|  |
| --- |
| >> d./e |

**ans = A number with numbers on it

AI-generated content may be incorrect.**

|  |
| --- |
| >> e.\d |

**ans = A number with numbers on it

AI-generated content may be incorrect.**

In both cases, the elements of the array in front of the slash is divided by the elements of the array behind the slash. To compute a matrix multiplication only the asterisk \* must be used, e.g.

|  |
| --- |
| >> C = A \* B |

Therefore, the number of columns of A must equal the number of rows of B.

|  |
| --- |
| >> A = [1 2 3; 4 5 6] |

**A = A screenshot of a computer

AI-generated content may be incorrect.**

|  |
| --- |
| >> B = [1 2; 3 4; 5 6] |

**B = A white background with numbers and lines

AI-generated content may be incorrect.**

|  |
| --- |
| >> C = A \* B |

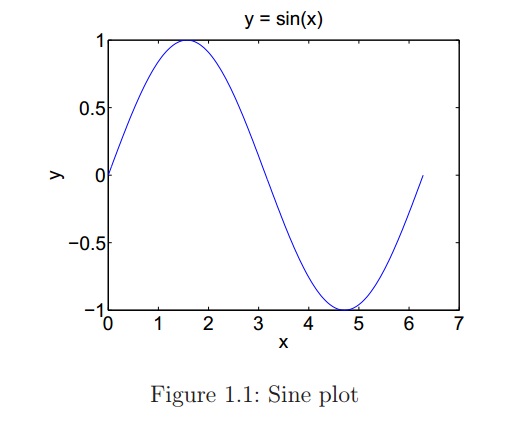
**C = A math equations and numbers

AI-generated content may be incorrect.**

**Creating a Plot**

The plot function has different forms, depending on the input arguments. If y is a vector, plot(y) produces a piecewise linear graph of the elements of y versus the index of the elements of y. If two vectors are specified as arguments, plot(x,y) produces a graph of y versus x. For example, to plot the value of the sine function from zero to 2π, use

|  |
| --- |
| >> x = 0:pi/100:2\*pi; >> y = sin(x); >> plot(x,y) |



The xlabel, ylabel and zlabel functions are useful to add x-, y- and z-axis labels. The function is only necessary for three-dimensional plots. The title function adds a title to a graph at the top of the figure and the text function inserts a text in a figure. The following commands create the final appearance of figure 1.1 .

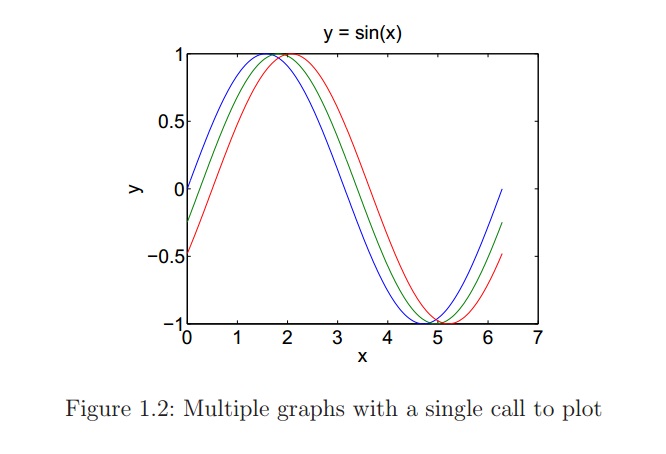
|  |
| --- |
| >> xlabel(’x’); >> ylabel(’y’); >> title(’y = sin(x)’) |

Multiple x-y pairs create multiple graphs with a single call to plot. Matlab automatically cycles through a predefined (but user settable) list of colors to distinguish between different graphs. For example, these statements plot three related functions of x1, each curve in a separate distinguishing color:

|  |
| --- |
| >> x1 = 0:pi/100:2\*pi; >> y1 = sin(x1); >> y2 = sin(x1 - 0.25); >> y3 = sin(x2 - 0.5); >> plot(x1,y1,x1,y2,x1,y3) |

The number of points of the individual graphs may be even different. It is possible to specify the color, the line style and the markers, such as plus signs or circles, with:

plot(x,y,’color style marker’)



**A color style marker is a 1-, 2-, or 3-character string. It may consist of a color type, a line style type, and a marker type:**

**Color strings are ’c’, ’m’, ’y’, ’r’, ’g’, ’b’, ’w’ and ’k’. These correspond to cyan, magenta, yellow, red, green, blue, white, and black.**

**Line style strings are ’-’ for solid, ’--’ for dashed, ’:’ for dotted, ’-.’ for dash-dotted and ’none’ for no line.**

**The most common marker types include’+’, ’o’, ’\*’ and ’x’.**

For example, the statement plot(x1,y1,’b:\*’) plots a blue dotted line and places asterisk sign markers at each data point. If only a marker type is specified but not a line style, Matlab draws only the marker.

The plot function automatically opens a figure window to plot the graphic. If there is already an existing figure window, this windows will be used for the new plot. The command figure can be used to keep an existing figure window and open a new one, which will be used for the next plot. To make an existing window the current window, type figure(n) where n is the number in the title bar of the window to be selected. The next graphic will be plotted in this selected window.

To add further plots to an existing graph, the hold command is useful. The hold on command keeps the content of the figure and plots can be added. So the above example could be done with three single plot commands and the hold on command. hold off ends the hold on status of a figure window. hold can be used to toggle between on and off.

**Controlling Axes**

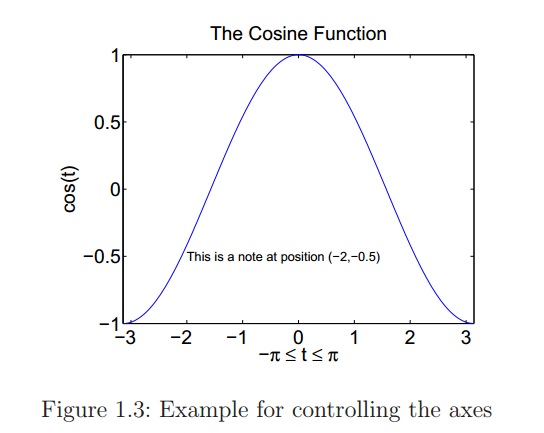
Usually, Matlab finds the maxima and minima of the data to be plotted by itself and uses them to create an appropriate plot box and axes labeling. The axis function overwrites this default by setting custom axis limits,

|  |
| --- |
| >> axis([xmin xmax ymin ymax]). |

The following example illustrates the use of the functions presented above.

|  |
| --- |
| >> t = -pi:pi/100:pi; >> s = cos(t); >> plot(t,s) >> axis([-pi pi -1 1]) >> xlabel(’-\pi \leq t \leq \pi’) >> ylabel(’cos(t)’) >> title(’The cosine function’) >> text(-2, -0.5,’This is a note at position (-2, -0.5)’) \leq is used to generate the less-equal sign. |

To take a closer look at an interesting part of a plot, the zoom command can be used. Afterwards it is possible to zoom by marking this part with the mouse. The grid command is used to turn a grid on and off.



**Pre-Lab task:**

1. Install MATLAB2016a Software in your personal Laptop/Desktop.
2. Go through the User Guide of MATLAB Software.

**Software:**

MATLAB2016a

**References:**

1. MATLAB user guide.
2. Prof. Dr.-Ing. Andreas Czylwik, “MATLAB for Communications”