AMERICAN INTERNATIONAL UNIVERSITY BANGLADESI

Faculty of Engineering

Laboratory Report Cover Sheet

Students must complete all details except the faculty use part.

Please submit all reports to your subject supervisor or the office of the concerned faculty.

Laboratory Title: Introduction to MATLAB **Experiment Number:** 01
Due Date: 01/02/2022 **Semester:** SPRING 2022 – 2023
Subject Code: COE 3103 **Subject Name:** Data communication **Section:** K
Course Instructor: DR. SHUVRA MONDAL **Degree Program:** BSc CSE

Declaration and Statement of Authorship:

1. I/we hold a copy of this report, which can be produced if the original is lost/ damaged.
2. This report is my/our original work and no part of it has been copied from any other student’s work or from any other source except where due acknowledgement is made.
3. No part of this report has been written for me/us by any other person except where such collaboration has been authorized by the lecturer/teacher concerned and is clearly acknowledged in the report.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the School for review and comparison, including review by external examiners.

I/we understand that

7. Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to expulsion from the University. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.
8. Enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy your work

Group Number (if applicable): **01** □ Individual Submission □ Group Submission

No.	Student Name	Student ID Number
1	SHEAKH, MOHAMMAD BIN AB. JALIL SHEAKH	20-42132-1
2	AURTHY, MOST. LILUN NAHAR	20-43997-2
3	NISHAT, TARIKUL ISLAM	21-44632-1
4	MULLICK, IFTEKHAR UDDIN	21-44649-1
5	ULLAH, MD ISMAIL JOBI	21-44747-1
6	ALANSAR, SADIAH	21-45612-3

For faculty use only: **Total Marks:________ Marks Obtained: _________**

Faculty comment:

Title:

Introduction to MATLAB

Abstract:

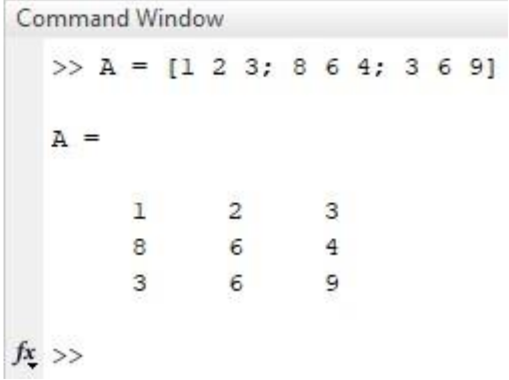
The objective of this experiment was to understand the use of MATLAB in various equations of data communication engineering problems. Using the obtained knowledge various commands, syntax and tools had to be used and implemented in obtaining the necessary results.

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 no interactive 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 features. These toolboxes are comprehensive collections of MATLAB functions, so-called M files extending the MATLAB environment to solve particular mathematical and engineering problems. MATLAB is a matrix-based programming tool. Although matrices often need not be dimensioned explicitly, the user must always look carefully for matrix dimensions. If it is not defined otherwise, the standard matrix exhibits two dimensions n x m. Column vectors and row vectors are represented consistently by n x 1 and 1 x n matrices, respectively.

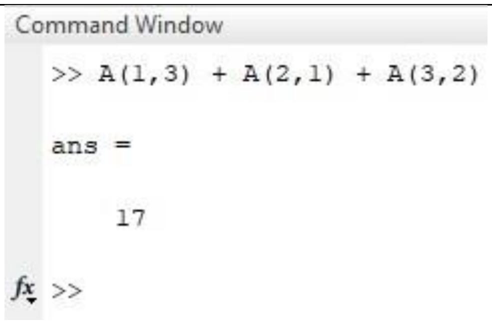
Simulation:

<u>Entering Matrices and Addressing the Elements</u>

Code: >> A = [1 2 3; 8 6 4; 3 6 9]	

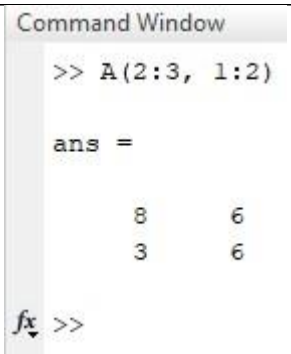
```
>> A = [1 2 3; 8 6 4; 3 6 9]

A =

     1     2     3
     8     6     4
     3     6     9

>>
```

The above code was used to create a matrix. ‘A’ represents the name of the matrix and the value within the third brackets are elements of the matrix. If there are multiple columns in the matrix a space ( ) was used to separate the elements and to end a row a semicolon (;) was used after finishing entering the elements of a row.

Code: >> A(1,3)+A(2,1)+A(3,2)	

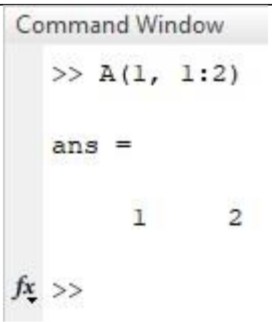
```
>> A(1,3) + A(2,1) + A(3,2)

ans =

    17

>>
```

The above code sums up the selected elements. To select the elements, the name of the matrix was used along with the position of the matrix. If the matrix is A(m,n), ‘m’ represents the row and ‘n’ represents the column.

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

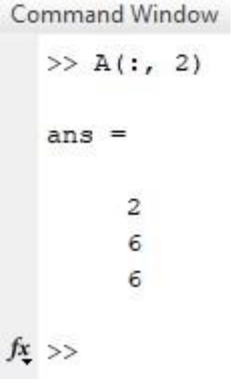
```
>> A(2:3, 1:2)

ans =

     8     6
     3     6

>>
```

The above code represents the code within a certain range. If the range of a matrix is A(m:n,k:l), the range of the row was ‘m’ to ‘n’ and the range of the column was ‘k’ to ‘l’. All the elements within this range was display on the prompt.

Code: >> A(1,1:2)	

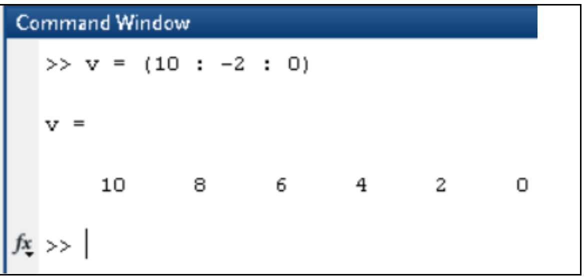
```
>> A(1, 1:2)

ans =

     1     2

>>
```

The above code displayed the elements of the first row which are within ‘1’ to ‘2’ columns. In this same way, the elements of a certain column can be represented as well.

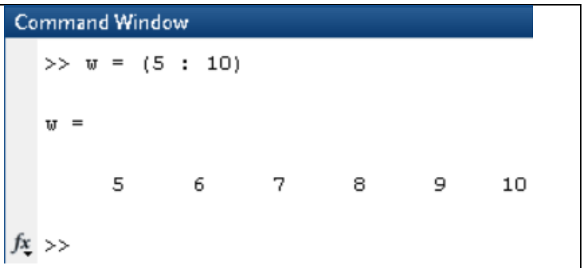
Code: >> A(:,2)	 Command Window >> A(:, 2) ans = 2 6 6 >>

In this code implementation, all the elements of a certain row or column were represented. In the above example, all the elements of the 2nd column of 'A' matrix have been represented. To represent all rows, the colon ':' sign was used.

Generating Matrices

Code: >> v = (10 : -2 : 0)	 Command Window >> v = (10 : -2 : 0) v = 10 8 6 4 2 0 >>

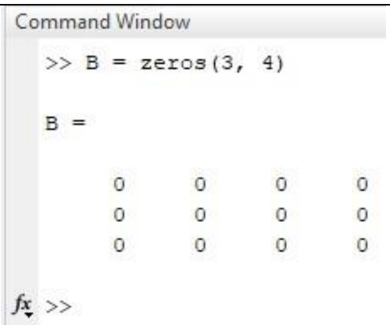
In the above implementation, a matrix was generated. The start value of the matrix was 10 and the end value was 0. The value was decreased by -2. Hence, a matrix was generated which had 1 row and 6 columns with 6 elements.

Code: >> w = (5 : 10)	 Command Window >> w = (5 : 10) w = 5 6 7 8 9 10 >>

In the above implementation, a matrix was generated which started with 5 and ended with 10. In this case, the difference between the starting and ending was not explicitly defined in the code. Hence, the value was increased by 1, which is the default incremental value.

Code:

```
>> B = zeros(3, 4)
```

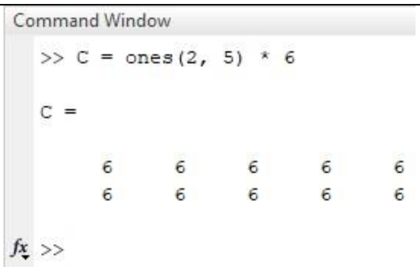


```
>> B = zeros(3, 4)

B =

     0     0     0     0
     0     0     0     0
     0     0     0     0

fx >>
```

Using the above command, a 3x4 matrix is generated which contained only zeros (0) as its elements. This reduces the time to create a matrix as this works as a macro to create a matrix.

Code:

```
>> C = ones(2,5)*6
```

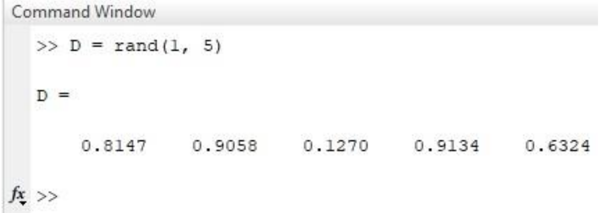


```
>> C = ones(2, 5) * 6

C =

     6     6     6     6     6
     6     6     6     6     6

fx >>
```

Using the ‘ones’ function, a (m, n) matrix is generated which contains only ‘1’ as elements of its matrix. After generating the ones matrix, 6 is multiplied by the matrix.

Code:

```
>> D = rand(1,5)
```

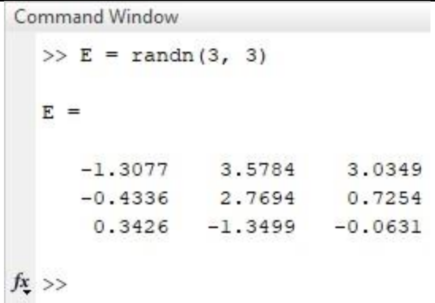


```
>> D = rand(1, 5)

D =

    0.8147    0.9058    0.1270    0.9134    0.6324

fx >>
```

To create a matrix with random values, ‘rand’ function is used. In the above code, a (1 X 5) matrix is created with random values. The ‘rand’ function uses positive values only to fill the elements of the matrix.

Code:

```
>> E = randn(3,3)
```



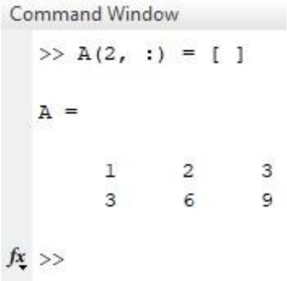
```
>> E = randn(3, 3)

E =

   -1.3077    3.5784    3.0349
   -0.4336    2.7694    0.7254
    0.3426   -1.3499   -0.0631

fx >>
```

The above function works just like ‘rand’. The difference between the two is ‘randn’ uses any real number as elements of the matrix that it is implemented on.

Deleting rows and columns

Code:

```
>> A(2,:) = [ ]
```



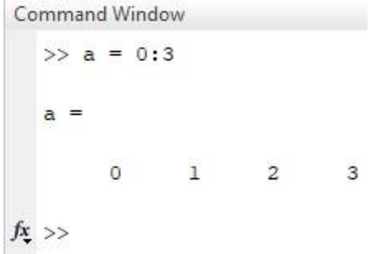
```
>> A(2, :) = [ ]

A =

     1     2     3
     3     6     9

fx >>
```

To delete a certain element, the above code was used. In the above implementation, the whole 2nd row of Matrix ‘A’ was deleted.

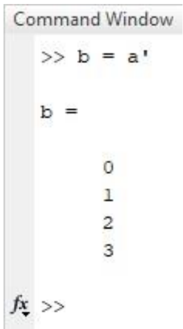
Array Orientation:

Code:

```
>> a = 0 : 3;
>> b = a’
```



```
>> a = 0:3

a =

     0     1     2     3

fx >>
```



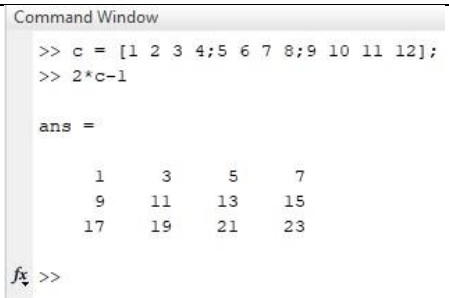
```
>> b = a'

b =

     0
     1
     2
     3

fx >>
```

In the above implantation, the orientation of a matrix was changed. All the values were inversed using the technique.

Scalar-Array Mathematics

Code:

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



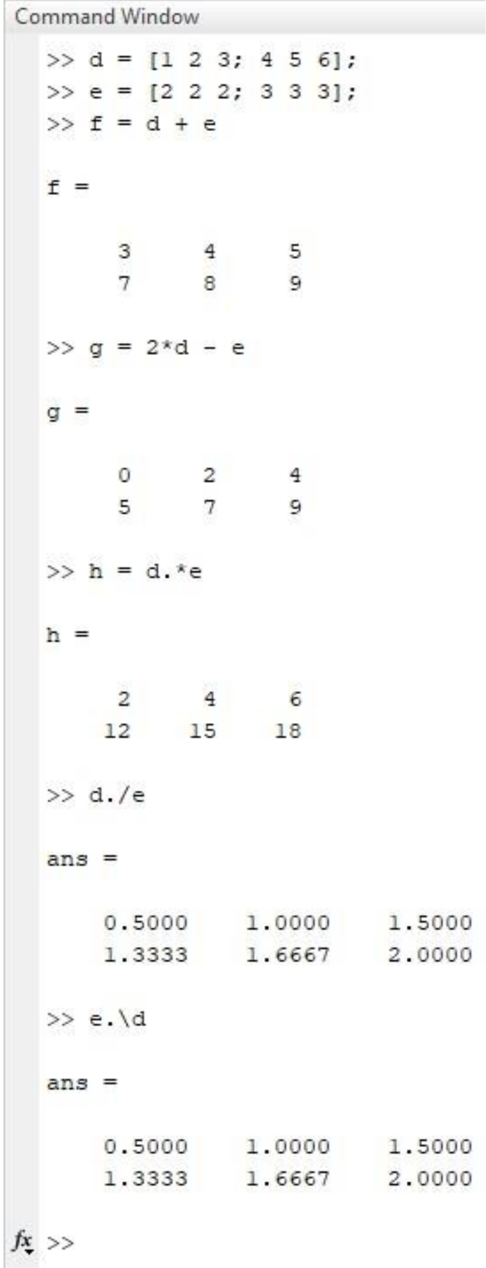
```
>> c = [1 2 3 4;5 6 7 8;9 10 11 12];
>> 2*c-1

ans =

     1     3     5     7
     9    11    13    15
    17    19    21    23

fx >>
```

The above implementation creates a matrix called ‘c’. All the elements of matrix ‘c’ have been multiplied by 2 and subtracted by 1 from each element of the result.

Array-Array Mathematics

Code:

```
>> d = [1 2 3; 4 5 6]
>> e = [2 2 2; 3 3 3]
>> f = d + e
>> g = 2*d – e
>> h = d.*e
>> d./e
>> e.\d
```

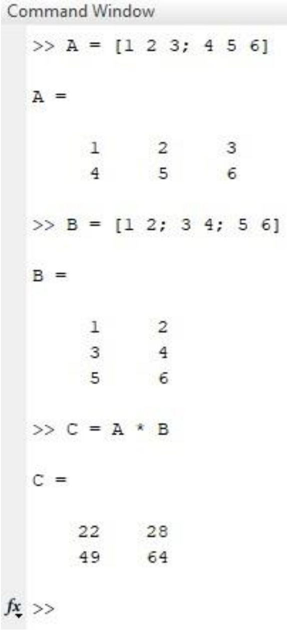


```
>> d = [1 2 3; 4 5 6];
>> e = [2 2 2; 3 3 3];
>> f = d + e

f =

     3     4     5
     7     8     9

>> g = 2*d - e

g =

     0     2     4
     5     7     9

>> h = d.*e

h =

     2     4     6
    12    15    18

>> d./e

ans =

    0.5000    1.0000    1.5000
    1.3333    1.6667    2.0000

>> e.\d

ans =

    0.5000    1.0000    1.5000
    1.3333    1.6667    2.0000

fx >>
```

The above implementation performed various mathematical operations between matrices. At first, two matrices were entered called ‘d’ and ‘e’. ‘f’ holds the value of the sum of ‘d’ and ‘e’. Then, ‘d’ was multiplied by 2 and then subtracted by ‘e’. Element-by-element multiplication using dot multiplication (.*) and array division (./) was performed.

Code:

```
>> A = [1 2 3; 4 5 6]
>> B = [1 2; 3 4; 5 6]
>> C = A * B
```



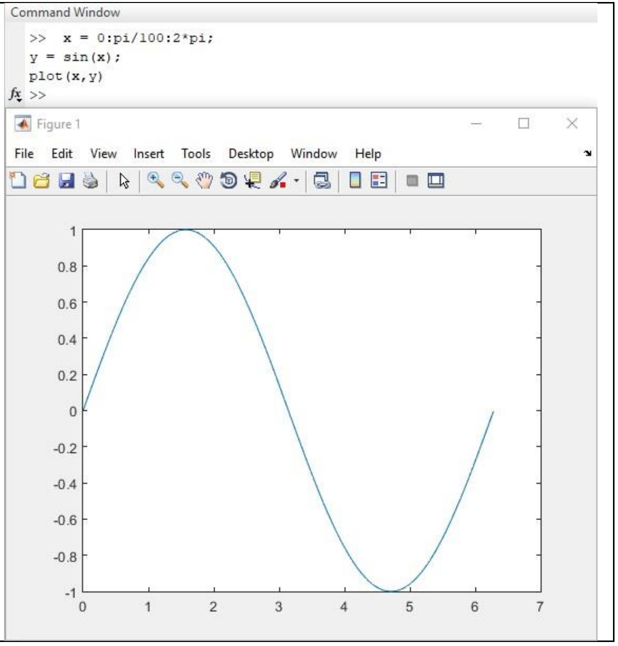
```
>> A = [1 2 3; 4 5 6]

A =

     1     2     3
     4     5     6

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

B =

     1     2
     3     4
     5     6

>> C = A * B

C =

    22    28
    49    64

fx >>
```

The above implementation performed a matrix multiplication. To perform this, ‘*’ operator was used.

Creating a plot

Code:

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

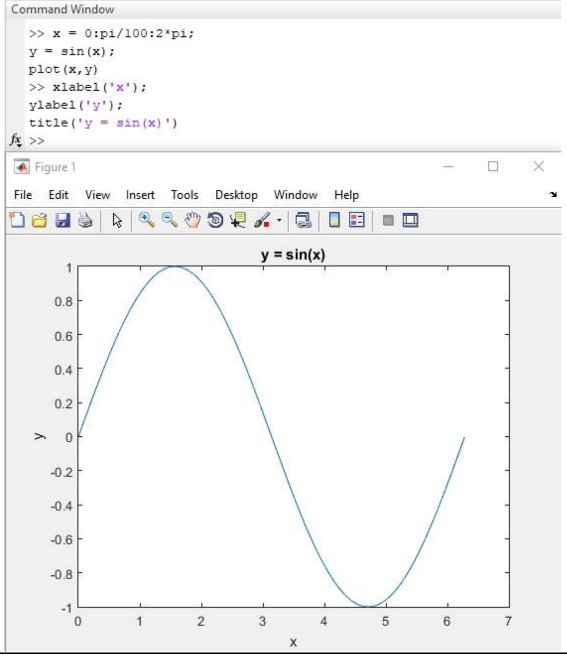


```
>>  x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
fx >>
```

The above code implementation plots a curve of a certain equation. ‘x’ represents the range of the values and the difference between each point of the range. ‘y’ represents the equation that needs to be plotted on the graph. Using the function ‘plot’, the plotted graph emerged.

Code:

```
>> x = 0 : pi/100 : 2*pi;
>> y = sin(x);
>> plot(x,y)
>> xlabel(‘x’);
>> ylabel(‘y’);
>> title(‘y = sin(x)’)
```

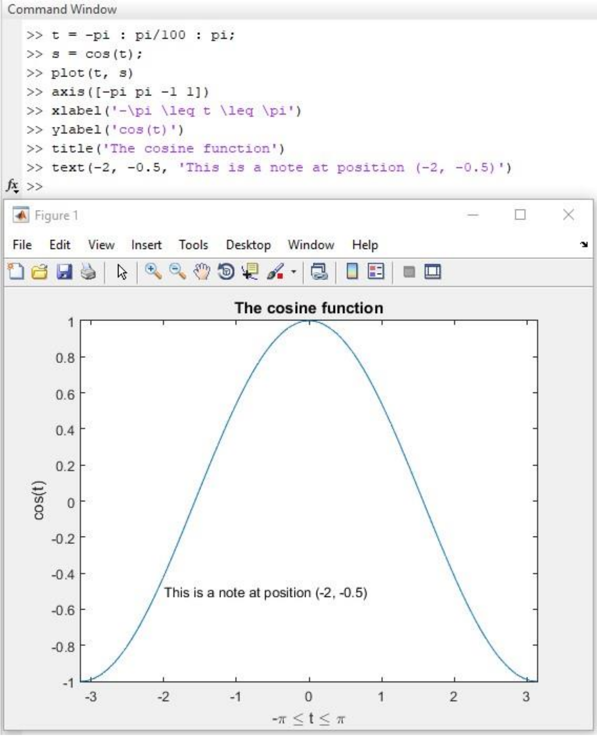


```
>> x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
>> xlabel('x');
ylabel('y');
title('y = sin(x)')
fx >>
```

Formatting a graph is another important part of graph plotting. To table the x-axis and y-axis, the function ‘xlabel’ and ‘ylabel’. ‘title’ was used to give the title of the graph.

Code:

```
>> 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)’)
```



```
>> 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)')
fx >>
```

To put a label on a certain point on the graph, the ‘text’ function was used. At first, the axis value of x, y and label text was provided.

Discussion and Conclusion:

From the above simulations, various functionalities of MATLAB were observed in hand. Various functions that were available on MATLAB were learned and observed. Using this knowledge, graphs were plotted in MATLAB software. Various formatting on the graph was learned from this experiment as well. Hence, it can be said that all the objectives of this experiment were obtained properly.

References:

- Prakash C. Gupta, “Data communications”, Prentice Hall India Pvt.
- William Stallings, "Data and Computer Communications”, Pearson
- Forouzan, B. A. "Data Communication and Networking. Tata McGraw." (2005).
- AIUB Data Communication Engineering Lab Manual, Report 01