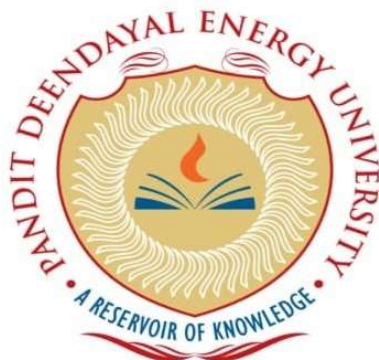


PANDIT DEENDAYAL ENERGY UNIVERSITY

SCHOOL OF TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING



LAB MANUAL

INDUSTRY 4.0 (20IF201T)

B. TECH - SEMESTER V

NAME : Ravi Jamanbhai Makwana

ROLL NO: 21BCP418

BATCH : Div-6, G-12

BRANCH: Computer Engineering

CERTIFICATE

This is to certify that roll no 21BCP418 of B.Tech 3rd year
Computer Engineering branch has completed the laboratory sessions
for the subjects of Industry 4.0 (20IF201T) satisfactorily.

Subject Coordinator

INDEX

Sr No	Experiment	Date of submission	Marks obtained	Sign
1				
2				
3	Introduction to MATLAB programming and SIMULINK			
4	Design of smart meter for recording the electricity consumption			
5	Design of Ultrasonic proximity sensors using Arduino			
6				
7				
8				
9				
10				

Experiment No: 03

Introduction to MATLAB programming and SIMULINK

Aim: To understand the basics about MATLAB software and learn basic programming and simulation.

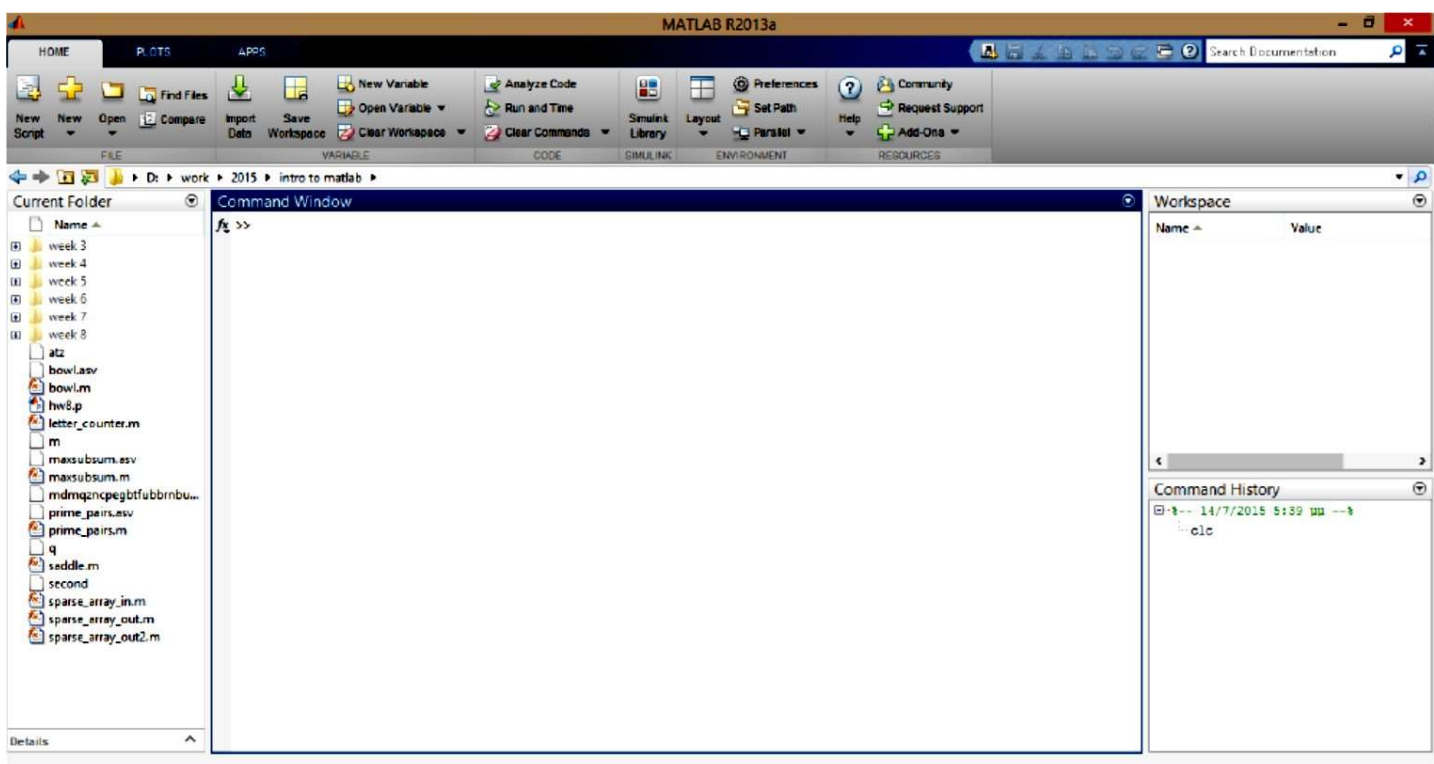
❖ Introduction

"MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation." - from Mathworks MATLAB is available on all three major operation systems.

SIMULINK is a powerful simulator for modeling and simulating dynamic systems.

One of the biggest advantages of using MATLAB is one does not need to declare ahead of time, the type of variable being used. The base version of MATLAB/SIMULINK is aptly supported by a variety of toolboxes.

The window is divided into three main parts.



- The Command Window is the main window where the commands are input.
- The Current Directory shows the directory from which MatLab runs the files and functions we have created.
- Command History shows the history of all the commands that we input into the Command Window.

- The main menu has three different tabs, HOME, PLOTS and APPS. From the Home tab the appearance and layout of MatLab can be changed.

❖ **Script File**

A script file in MATLAB is any set of MATLAB commands which are executed in that sequence. Script files have ".m" extension. It is recommended to use a meaningful name for the filename without including "space".

A script file can be run by simply typing the filename without its extension.

The scope of the variables used in script files is the general workspace. They can be accessed even after the execution.

A script file can contain functions which can be used within that script.

MATLAB executes the script file by interpreting every line, which makes it somewhat slow at times.

A script file can be put on path so that it can be called from any directory.

Note: Do not assign a script file name which is already a MATLAB in built function.

❖ **Matrix Algebra**

Create a row vector $A = [1 \ 3 \ 6 \ 8]$

Create a column vector $A = [1;3;6;8]$

Transpose of matrix $A = [1 \ 3 \ 6 \ 8]'$ Or use **transpose(A)**

Create a matrix $A = [1 \ 3; \ 6 \ 8]$

Assign each element to matrix $A(1, 1) = 1; A(2, 1) = 6; A(1, 2) = 3; A(2, 2) = 8;$

- Both matrices and vectors are enclosed in square brackets
- Elements are accessed using parentheses

❖ **For creating uniformly spaced vector** $B = (0 : 1 : 99)$

B has 100 elements from 0 to 99 with spacing of 1.

Access first two and last two elements of B: $B = ([1 \ 2 \ \text{end}-1 \ \text{end}])$

Take every third element of B and store in C: $C = B(1:3:\text{end})$

❖ **Special Matrices/Array**

Matlab allows to create a variety of special matrices that appear in several problems.

- ❖ **Create an Identity matrix** $\text{Imat} = \text{eye}(3; 3)$
- ❖ **Create a matrix of ones and zeros** $A = \text{ones}(3; 3)$ $B = \text{zeros}(3; 3)$
- ❖ **Hadamard Matrix** $H = \text{hadamard}(2)$
- ❖ **Magic Matrix** $M = \text{magic}(3)$
- ❖ **Toeplitz Matrix** $T = \text{toeplitz}([3 \ 4]; [3 \ 2])$
- ❖ **Simple checks on matrices** Check if the matrix is empty
 $A = []$; $\text{isempty}(A)$
- ❖ **Check if two matrices are equal**
 $A = \text{exp}([1 \ 2; \ 3 \ 4])$

```
B=expm([1 2; 3 4])
isequal(A,B)
```

- ❖ Check if a matrix contains real elements

```
A = [1 2; 3 4]
B = ones(2, 2)
C = A + j * B
isreal(C)
```
- ❖ **Check if matrix elements are NaN (Not a Number)**

```
A = [1 2; 3 inf]
isnan(A)
```
- ❖ **Mathematical Operations on Matrices**
Transpose of a matrix $A = [1 \ 8 \ 3; 5 \ 6 \ 0]$ A'
- ❖ **Extract triangular part**

```
tril(A)
```
- ❖ **Find indices of elements**

```
find(A) find(A >= 4)
```
- ❖ **Matrix Multiplication**

```
A = [1 8; -5 0; 9 2] B = [2 6; 8 3; -1 5]
```
- ❖ **Element wise multiplication:** $A.*B$
- ❖ **Product of A and B:** $A*B$
- ❖ **Maximum and minimum of A**

```
max(A) min(abs(A))
```
- ❖ **Inverse of matrix A:** $\text{inv}(A)$
- ❖ **Determinant of matrix A:** $\text{det}(A)$
- ❖ **Matrix division B**

```
B *inv(A)
```
- ❖ **Element wise division:** $B./A$
- ❖ **Eigen values of matrix A:** $\text{eig}(A)$
- ❖ **Rank of matrix A:** $\text{rank}(A)$
- ❖ **Characteristics equation**

```
eqA = poly([1 2; 3 4])
roots(eqA),
```

Note: compare this with eigen values of A
- ❖ **LU factorization** $[L,U] = \text{lu}([1 \ 2; 3 \ 4])$
- ❖ **Orthogonalization** $Q = \text{orth}([1 \ 2; 3 \ 4])$
- ❖ **Special Numbers and variables**

pi: The value of pi
inf: Infinity (or a very very large number)
eps: Floating point relative accuracy
i or j: Imaginary number,
NaN: Not-a-Number
ans: The most recent answer
end: The last element of a vector; OR to indicate end of a loop or a conditional statement
all: Used with clear command to clear all variables
- ❖ **Elementary functions**

Trigonometric: sin, sinh, cos, atan, sec
Exponential: exp, log, log2, pow2, sqr
Complex: abs, imag, conj, unwrap,
angle Rounding: fix, floor, ceil, mod,
rem, sign Specialized: bessell, beta, erf,
dot, gamma Number theoretic: factor,
primes, factorial, gcd

❖ **Create 100 samples of sine and cosine**

$x = \sin(2 * \pi * 0:2 * (0 : 99)) ;$ $y = \cos(2 * \pi * 0:2 * (0 : 99))$

PART A: MATLAB Programming

(1) Sum of series $1 + 1/2 + 1/4 + 1/6 + \dots$ for 100 terms

```
clc
% clear all
% sum of series  $1 + 1/2 + 1/4 + 1/6 + \dots$  for 100 terms
x=1;    %initialising the first term
n=input('The number of terms in series to be summed:')
for i=1:n
    y=1/(2*i);

    x=x+y;
end
display(x)
```

Output)

```
The number of terms in series to be summed:
10

x =

    2.4645
```

(2) Square roots of odd positive integers

```
clc
%clear all
% square roots of odd positive integers:

n=input('The total number of odd positive terms whose roots
are to be displayed:')

for i=1:n
    y=(2*i)-1;
    x(i)=sqrt(y);
end

display(x')
```

Output)

```
The total number of odd positive terms whose roots are to be displayed:
5
    1.0000
    1.7321
    2.2361
    2.6458
    3.0000
```

(3) Find the smallest 'k' so that $1 + 2 + 3 + \dots + k \geq 30$

```
clc
% smallest 'k' so that  $1 + 2 + 3 + \dots + k \geq 30$ 
for k=1:100
```

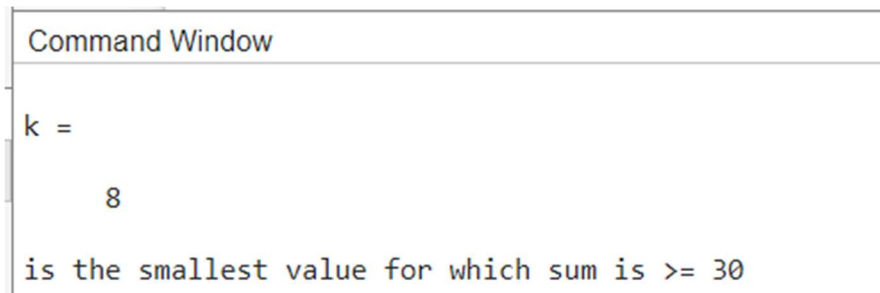


```
    sum = k*(k+1)/2;
    if sum>=30
display(k)

display('is the smallest value for which sum is >= 30')
    break

    end
end
```

Output)



The screenshot shows a MATLAB Command Window with the following output:

```
Command Window

k =

     8

is the smallest value for which sum is >= 30
```

(4) Create a function file to input the radius & area of circle is returned as output:

```
% Input the radius & output area of circle:

function ar=cir_area(r)
r=input('Enter the radius of the circle: ')
ar=pi*r*r;

disp('Area of the circle is:')
```

Output)

Command Window

```
>> lab3_4
Enter the radius of the circle:
4

r =

    4

Area of the circle is:

ans =

    50.2655
```

(5) To check whether the number is odd or even

```
clc
% To check whether the number is odd or even

x=input('Enter any number: ')
z=rem(x,2)

if z==0
    display('The entered number is an even number.')
else
    display('The entered number is an odd number.')
end
```

Output)

```
Enter any number:
23

x =

    23

The entered number is an odd number.
```

(6) To display all prime numbers from 50-150

```
clc
clear all
% display all primes from 50 to 150
x1=input('Enter the lower limit of range') % 50
x2=input('Enter the upper limit of range') % 150
y=primes(x2)

for k=1:length(y)
    if y(k)>50
        disp(y(k))
    end
end
```

Output)

Command Window

Enter the lower limit of range

50

x1 =

50

Enter the upper limit of range

150

x2 =

150

53

59

61

67

71

73

79

83

89

97

101

103

107

109

113

127

131

137

139

149

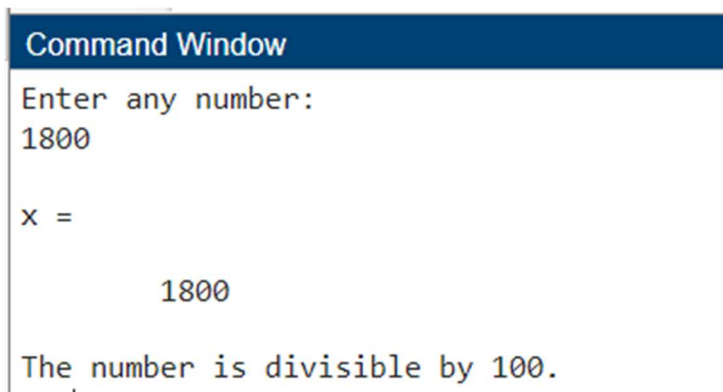
(7) Display if number is divisible by 100

```
clc
% Display if number is divisible by 100

x=input('Enter any number: ')
z=rem(x,100)

if z==0
    disp('The number is divisible by 100.')
else
    disp('The number is not divisible by 100.')
end
```

Output)



The screenshot shows the MATLAB Command Window with a dark blue title bar. The text inside shows the prompt 'Enter any number:' followed by the user input '1800'. Below this, it shows 'x =' followed by the value '1800'. At the bottom, it displays the output message 'The number is divisible by 100.'.

```
Command Window
Enter any number:
1800

x =

    1800

The number is divisible by 100.
```

(8) Swap the values of given numbers

```
clc
% swap the values of given numbers
```

```
x1=input('Enter any number:')
x2=input('Enter any number:')

a=x2;
b=x1;

x1=a
x2=b

sprintf('X1 is swapped by X2..!')
```

Output)

Command Window

```
Enter any number:
18

x1 =

    18

Enter any number:
20

x2 =

    20

x1 =

    20

x2 =

    18

ans =

    'X1 is swapped by X2..!'
```

