Exercise-1

Programs on Arithmetic Operations, Suppressing Output, Built-in Functions, Variables

Arithmetic Operations:

Operator	Command	Details
Addition	+	Addition
	sum	Sum of array elements
	cumsum	Cumulative sum
	movsum	Moving sum
Subtraction	-	Subtraction
	diff	Differences and approximate derivatives
Multiplication	.*	Multiplication
	*	Matrix multiplication
	prod	Product of array elements
	cumprod	Cumulative product
Division	./	Right array division
	.\	Left array division
	/	Solve systems of linear equations xA = B for x
	\	Solve systems of linear equations Ax = B for x
Powers	.^	Element-wise power
	^	Matrix power

Built-in Functions:

(i) Modulo Division and Rounding

mod	Remainder after division (modulo operation)
rem	Remainder after division
ceil	Round toward positive infinity
floor	Round toward negative infinity
round	Round to nearest decimal or integer

(ii) Trigonometry

sin	Sine of argument in radians	tan	Tangent of argument in radians
sind	Sine of argument in degrees	tand	Tangent of argument in degrees
sinpi	Compute sin(X*pi) accurately	atan	Inverse tangent in radians
asin	Inverse sine in radians	atand	Inverse tangent in degrees
asind	Inverse sine in degrees	tanh	Hyperbolic tangent
sinh	Hyperbolic sine	atanh	Inverse hyperbolic tangent
asinh	Inverse hyperbolic sine	csc	Cosecant of input angle in radians
cos	Cosine of argument in radians	cscd	Cosecant of argument in degrees
cosd	Cosine of argument in degrees	acsc	Inverse cosecant in radians
cospi	Compute cos(X*pi) accurately	acscd	Inverse cosecant in degrees

acos	Inverse cosine in radians	csch	Hyperbolic cosecant
acosd	Inverse cosine in degrees	acsch	Inverse hyperbolic cosecant
cosh	Hyperbolic cosine	cot	Cotangent of angle in radians
acosh	Inverse hyperbolic cosine	cotd	Cotangent of argument in degrees
sec	Secant of angle in radians	acot	Inverse cotangent in radians
secd	Secant of argument in degrees	acotd	Inverse cotangent in degrees
asec	Inverse secant in radians	coth	Hyperbolic cotangent
asecd	Inverse secant in degrees	acoth	Inverse hyperbolic cotangent
sech	Hyperbolic secant		
asech	Inverse hyperbolic secant		
hypot	Square root of sum of squares (hypotenuse)		

(ii)Functions

exp	Exponential
log	Natural logarithm
log10	Common logarithm (base 10)
log2	Base 2 logarithm and floating-point number dissection
sqrt	Square root

MATLAB Programs

1	Calculate
	(i) $\frac{\sin(0.2\pi)}{\cos(\pi/6)} + \tan 72^0$ (ii) $(\tan 64^0 \cos 15^0)^2 + \frac{\sin^2 37^0}{\cos^2 20^0}$
	(i)
	MATLAB Code:
	`sinpi(0.2)/cospi(1/6)+tan(72)`
	Output:
	ans =
	0.4163
	(ii)
	MATLAB Code:
	`(tan(64)*cos(15))^2+(sin(37))^2/(cos(20))^2`
	Output:
	ans =
	5.6682

2	Define the variable z as z= 4.5, then evaluate:
	(i) $0.4z^4 + 3.1z^2 - 162.3z - 80.7$ (ii) $(z^3 - 23)/\left(\sqrt[3]{z^2 + 17.5}\right)$
	(i) MATLAB Code:
	`z=4.5;`
	`0.4*z^4+3.1*z^2-162.3*z-80.7`
	Output:
	ans =
	-584.2500
	(ii) MATLAB Code:
	`z=4.5;`
	`(z^3-23)/(z^2+17.5)^(1/3)`
	Output:
	ans =
	20.3080

3	Define the variable t as t=3.2, the evaluate
	(i) $\frac{1}{2}e^{2t} - 3.81t^3$ (ii) $\frac{6t^2 + 6t - 2}{t^2 - 1}$
	(i) MATLAB Code:
	`t=3.2;`
	`exp(2*t)/2-3.81*t^3`
	Output:
	ans =
	176.0764
	(ii) MATLAB Code:
	`t=3.2;`
	`(6*t^2+6*t-2)/(t^2-1)`
	Output:
	ans =
	8.5108

4	Define the variables x and y as x= 6.5 and y= 3.8, then evaluate
	(i) $\left(x^2+y^2\right)^{2/3}+\dfrac{xy}{y-x}$ (ii) $\dfrac{\sqrt{x+y}}{(x-y)^2}+2x^2-xy^2$
	(i) MATLAB Code:
	`x= 6.5;`
	`y= 3.8;`
	`(x^2+y^2)^(2/3)+x*y/(y-x)`
	Output:
	ans =
	5.6091
	(ii) MATLAB Code:
	`x= 6.5;`
	`y= 3.8;`
	`sqrt(x+y)/(x-y)^2+2*x^2-x*y^2`
	Output:
	ans =
	-8.9198

5	Define the variables a, b, c and d as c =4.6, d=1.7, a =cd^2, and $b=\dfrac{c+a}{c-d}$, then evaluate
	(i) $e^{d-b}+\sqrt[3]{c+a}-(ca)^d$ (ii) $\dfrac{d}{c}+\left(\dfrac{ct}{b}\right)^2-c^d-\dfrac{a}{b}$
	(i) MATLAB Code:
	`c =4.6;`
	`d=1.7;`
	`a =c*d^2;`
	`b=(c+a)/(c-d);`
	`exp(d-b)+(c+a)^(1/3)-(c*a)^d`
	Output:
	ans =
	-1.0861e+03
	(ii) MATLAB Code:
	`c =4.6;`
	`d=1.7;`
	`a =c*d^2;`
	`b=(c+a)/(c-d);`
	`d/c+(c*t/b)^2-c^d-a/b`
	Output:
	ans =
	-9.4810

6	Write the MATLAB code to find the simple interest using the formula $I=\dfrac{PNR}{100}$ when P= 1500, N=2.3, R= 3%. Also find total amount.
	MATLAB Code:
	`clc;clear all;`
	`%simple interest`
	`P=input('enter the value of P\n P=');`
	`N=input('enter the value of N\n N=');`
	`R=input('enter the value of R\n R=');`
	`i=(P*N*R)/100;`

	`fprintf('The simple interest I=%0.2f\n',i)` **Output:** enter the value of P P=1500 enter the value of N N=2.3 enter the value of R R=3 The simple interest I=103.50

7	Write the MATLAB code to find the Compound interest using the formula
	$A = P\left(1+\frac{r}{100}\right)^n$ when P = 2500, n = 3.5, r = 7%. Also find total interest.
	MATLAB Code:
	`clc;clear;`
	`%Compound interest`
	`P=input('enter the value of P\n P=');`
	`N=input('enter the value of N\n N=');`
	`R=input('enter the value of R\n R=');`
	`A=P*(1+R/100)^N;`
	`i=A-P;`
	`fprintf('The Compound interest I=%0.2f\n',i)`
	Output:
	enter the value of P
	P=2500
	enter the value of N
	N=7
	enter the value of R
	R=3.5
	The Compound interest I=680.70

8	Write the MATLAB code to find the Area of the circle with radius $r = \pi^{1/3} - 1$ r=pi^(1/3)-1; area=pi*r^2 **Output:** area = 0.6781
9	Write the MATLAB code to find the slope of the straight line at the point (1, 2) and horizontal intercept c = 3. **MATLAB Code:** x=1; y=2;c=3; slope=(y-c)/x **Output:** slope = -1

10	The monthly payment M of a Mortgage P for n years with fixed annual interest rate r can be calculated by the formula: $$M = P\frac{\frac{r}{12}\left(1+\frac{r}{12}\right)^{12n}}{\left(1+\frac{r}{12}\right)^{12n}-1}$$ Write the MATLAB code to find Determine the monthly payment of 30 years, Rs. 4, 50,000 mortgage with interest rate of 4.2%. Define the variables P, r and n, then use them in the formula to calculate M. **MATLAB Code:** P=450000; r= 4.2;n=30; M=P*(r/12)*(1+r/12)^(12*n)/((1+r/12)^(12*n)-1) **Output:** M = 1.5750e+05 **Output:** M = 1.5750e+05

Outcome:*<Write in your own words on learning experience, "what you learn/doing after completion of this exercise".>*

Exercise-2

Programs on Vectors , Matrices, Symbolic Mathematics

Vectors :A vector is a one-dimensional array of numbers. MATLAB allows creating two types of vectors: Row vectors, Column vectors.

Creating Vectors: Row vectors are created by enclosing the set of elements in square brackets, using space or comma to delimit the elements. Column vectors are created by enclosing the set of elements in square brackets, using semicolon to delimit the elements.

Matrix: A matrix is a two-dimensional array of numbers. In MATLAB, you create a matrix by entering elements in each row using comma or space delimited numbers and using semicolons to mark the end of each row.

The following table describes its use for this purpose (let us have a matrix A):

Format	Purpose
A(:,j)	is the j^{th} column of A.
A(i,:)	is the i^{th} row of A.
A(:,:)	is the equivalent two-dimensional array. For matrices this is the same as A.
A(j:k)	is A(j), A(j+1),...,A(k).
A(:,j:k)	is A(:,j), A(:,j+1),...,A(:,k).
A(:,:,k)	is the k^{th} page of three-dimensional array A.
A(i,j,k,:)	is a vector in four-dimensional array A. The vector includes A(i,j,k,1), A(i,j,k,2), A(i,j,k,3), and so on.
A(:)	is all the elements of A, regarded as a single column. On the left side of an assignment statement, A(:) fills A, preserving its shape from before. In this case, the right side must contain the same number of elements as A.

Vector, Matrix, and Array Commands: The following table shows various commands used for working with arrays, matrices and vectors:

Command	Purpose	Command	Purpose
cat	Concatenates arrays.	eye	Creates an identity matrix.
find	Finds indices of nonzero elements.	ones	Creates an array of ones.
length	Computes number of elements.	zeros	Creates an array of zeros.
max	Returns largest element.	cross	Computes matrix cross products.
min	Returns smallest element.	dot	Computes matrix dot products.
prod	Product of each column.	det	Computes determinant of an array.
size	Computes array size.	inv	Computes inverse of a matrix.
sort	Sorts each column.	rank	Computes rank of a matrix.
sum	Sums each column.		

MATLAB Programs

1	Create a row vector V = [1 3 5 7 -9 10]; and a column vector $W = \begin{bmatrix} 3 \\ -7 \\ 9 \\ 10 \\ 12 \end{bmatrix}$ Find the vectors 2V, W^2, V^2+2V **MATLAB CODE:** V=[1 3 7 -9 10] **Output:** V = 1 3 7 -9 10 W=[3;-7;9;10;12] **Output:** **W =** **3** **-7** **9** **10** **12** 2*V **Output:** ans = 2 6 14 -18 20 W.^2 **Output:** ans = 9 49 81 100 144 V.^2+2*V **Output:** ans = 3 15 63 63 120
2	Create a vector with eleven elements from 0 to 10. Take r = 0.5 and create another vector x = [1, r, r^2, r^3,...r^n]. Find the geometric sum s=1+r+r^2+r^3+...+r^{11}. Now calculate the limit 1/(1-r) and compare with 's'. Repeat taking n from 0 to 50 and then from 0 to 100. **MATLAB Code:** i=0:10; r=0.5; x= r.^i **output:** x = Columns 1 through 9 1.0000 0.5000 0.2500 0.1250 0.0625 0.0313 0.0156 0.0078 0.0039 Columns 10 through 11

```
    0.0020    0.0010

i=0:11; r=0.5;
x= r.^i;
s=sum(x)
output:
s =
    1.9995
d=s-1/(1-r)

d =
  -4.8828e-04
```

3

(i) Create a matrix $A=\begin{bmatrix} -2 & 1 & 4 & 3 & 2 \\ 0 & 1 & 5 & 6 & 7 \\ 3 & 4 & 9 & 10 & -5 \end{bmatrix}$

(ii) Create a sub-matrices $B=\begin{bmatrix} 1 & 5 & 6 \\ 4 & 9 & 10 \end{bmatrix}$ and $C=\begin{bmatrix} 0 & 7 \\ 3 & -5 \end{bmatrix}$

(iii) Using MATLAB commands delete 2nd row of the above matrix A and insert the same row again.

(iv)Using MATLAB commands delete 3rd and 4th columns of A. Interchange 1st and 2nd rows of A.

(i) Create a matrix $A=\begin{bmatrix} -2 & 1 & 4 & 3 & 2 \\ 0 & 1 & 5 & 6 & 7 \\ 3 & 4 & 9 & 10 & -5 \end{bmatrix}$

MATLAB CODE:

```
A=[-2 1 4 3 2;0 1 5 6 7; 3 4 9 10 -5];
```

OUTPUT:

```
A
A =
   -2    1    4    3    2
    0    1    5    6    7
    3    4    9   10   -5
```

(ii) Create a sub-matrices $B=\begin{bmatrix} 1 & 5 & 6 \\ 4 & 9 & 10 \end{bmatrix}$ **and** $C=\begin{bmatrix} 0 & 7 \\ 3 & -5 \end{bmatrix}$

MATLAB CODE:

```
B=A(2:3,2:4)
```

OUTPUT:

```
B =
    1    5    6
    4    9   10
```

MATLAB CODE:

```
C=A(2:3,[1 5])
```

OUTPUT:

```
C =
    0    7
    3      -5
```

Using MATLAB commands delete 2nd row of the above matrix A and insert the same row again.

MATLAB CODE:

```
A(2,:)=[]
A =
   -2    1    4    3    2
    3    4    9   10   -5
x=[ 0 1 5 6 7 ];
A=[A(1,:);x;A(2,:)]
```

Output:

```
A =
   -2    1    4    3    2
    3      1    5    6    7
    3    4    9   10   -5
```

(iv)Using MATLAB commands delete 3rd and 4th columns of A. Interchange 1st and 2nd rows of A.

```
A(:,3:4)=[]
```

Output:

```
A =
   -2    1    2
    0    1    7
    3    4   -5
A=[A(2,:); A(1,:); A(3,:)]
```

Output:

```
A =
    0    1    7
   -2    1    2
    3    4   -5
```

4

(i) Create a following matrix

$$A = \begin{bmatrix} 2 & 1 & 4 & 4 & 1 & 16 \\ -1 & 6 & 7 & 1 & 36 & 49 \\ 3 & 5 & 2 & 9 & 25 & 4 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 1 & 4 & 4 & 1 & 16 \\ 3 & 6 & 7 & 1 & 36 & 49 \\ 3 & 5 & 2 & 9 & 25 & 4 \\ 3 & 0 & 0 & 1 & 0 & 0 \\ 0 & 5 & 0 & 10 & 1 & 0 \\ 0 & 0 & 0 & 20 & 20 & 1 \end{bmatrix}$$

(i) Create 3X 3 matrices A and B and find (i) A+B (ii) A-B (iii)A*B (iv) A.*B. Observe the difference between the operations * and .* . Also find A^2 and A.^2; A./B, B.\A

MATLAB Code:

```
clc
P=[2,1,4,4,1,16;-1,6,7,1,36,49;3,5,2,9,25,4];
B=[1,1,4,4,1,16;3,6,7,1,36,49;3,5,2,9,25,4;3,0,0,1,0,0;0,5
,0,10,1,0;0,0,0,20,20,1];
A=[P;[zeros(3),eye(3)]];
disp("A+B=")
disp(A+B)
```

```
disp("A-B=")
disp(A-B)
disp("A*B=")
disp(A*B)
disp("A.*B=")
disp(A.*B)
disp("A^2=")
disp(A^2)
disp("A.^2=")
disp(A.^2)
disp("A./B=")
disp(A./B)
disp("B.\A=")
disp(B.\A)
```

Output:

```
A+B=
   3    2    8    8    2   32
   2   12   14    2   72   98
   6   10    4   18   50    8
   3    0    0    2    0    0
   0    5    0   10    2    0
   0    0    0   20   20    2

A-B=
   1    0    0    0    0    0
  -4    0    0    0    0    0
   0    0    0    0    0    0
  -3    0    0    0    0    0
   0   -5    0  -10    0    0
   0    0    0  -20  -20    0

A*B=
     29     33     23    379    459    113
     41    250     52   1406   1406    355
     51    168     51    374    338    305
      3      0      0      1      0      0
      0      5      0     10      1      0
      0      0      0     20     20      1

A.*B=
      2      1     16     16      1    256
     -3     36     49      1   1296   2401
      9     25      4     81    625     16
      0      0      0      1      0      0
      0      0      0      0      1      0
      0      0      0      0      0      1

A^2=
  15   28   23   49  139  113
  13   70   52   66  426  355
   7   43   51   44  258  305
   0    0    0    1    0    0
```

	0 0 0 0 1 0 0 0 0 0 0 1 A.^2= 4 1 16 16 1 256 1 36 49 1 1296 2401 9 25 4 81 625 16 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 A./B= 2.0000 1.0000 1.0000 1.0000 1.0000 1.0000 -0.3333 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0 NaN NaN 1.0000 NaN NaN NaN 0 NaN 0 1.0000 NaN NaN NaN NaN 0 0 1.0000 B.\A= 2.0000 1.0000 1.0000 1.0000 1.0000 1.0000 -0.3333 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 0 NaN NaN 1.0000 NaN NaN NaN 0 NaN 0 1.0000 NaN NaN NaN NaN 0 0 1.0000
5	If $A = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \\ 2 & 1 & 1 \end{pmatrix}$ Using MATLAB commands find dimension of A, determinant of A, inverse of A, Transpose of A, eigen values and eigen vectors of A. **MATLAB Code:** A=[1 2 1; 1 1 2; 2 1 1]; >> A=[1 2 1; 1 1 2; 2 1 1]; >> det(A) ans = 4 >> A' ans = 1 1 2 2 1 1 1 2 1 >> inv(A) ans = -0.2500 -0.2500 0.7500 0.7500 -0.2500 -0.2500 -0.2500 0.7500 -0.2500

```
>> [V D]=eig(A)
V =
   0.5774 + 0.0000i  -0.2887 - 0.5000i  -0.2887 + 0.5000i
   0.5774 + 0.0000i   0.5774 + 0.0000i   0.5774 + 0.0000i
   0.5774 + 0.0000i  -0.2887 + 0.5000i  -0.2887 - 0.5000i

D =
   4.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i  -0.5000 + 0.8660i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.5000 - 0.8660i
```

6 Create the following two row vectors: d=[6 -1 4 0 -2 5] and e=[7 5 9 0 1 3].

i. Use the two vectors in a MATLAB command to create a matrix such that the first row consists of elements 2 through 4 of vector d, the second row consists of elements 3 through 5 of vector e, and the third row consists of elements 4 through 6 of vector d.

ii. Use the two vectors in a MATLAB command to create a matrix such that the first column consists of elements 2 through 5 of vector d, and the second column consists of elements 3 through 6 of vector e.

(i) MATLAB Code:

```
d=[6 -1 4 0 -2 5] ;
e=[7 5 9 0 1 3];
a=[d(2:4);e(3:5);d(4:6)]
```

Output:

```
a =

   -1    4    0
    9    0    1
    0   -2    5
```

(ii) MATLAB Code:

```
d=[6 -1 4 0 -2 5] ;
e=[7 5 9 0 1 3];
b=[d(2:5);e(3:6)]
```

Output:

```
b =

   -1    4    0   -2
    9    0    1    3
```

7 Create the following vector: v=[5 0 -3 7 6 -1 2 8 4 9]. Write what will be displayed if the following commands are executed by MATLAB. Check your answers by executing the commands with MATLAB.

(a) a = v([4 5:7 10]) (b) b = v([9, 1, 6:-2:2])' (c) c = [b' a']

MATLAB Code:

```
v=[5 0 -3 7 6 -1 2 8 4 9];
a = v([4 5:7 10])
b = v([9, 1, 6:-2:2])
c = [b' a']
```

Output:

```
a =
    7    6   -1    2    9
b =
    4    5   -1    7    0
```

```
c =
    4     7
    5     6
   -1    -1
    7     2
    0     9
```

8 For the function $y = x^4 e^{-x}$, calculate the value of y for the following values of x using element-by-element operations: 1.5, 2, 2.5, 3, 3.5, 4.

MATLAB:

```
x=1.5:0.5:4;
y=x.^4.*exp(-x)
```

Output:

```
y =
   1.1296    2.1654    3.2064    4.0328    4.5315    4.6888
```

9 Use the eye, ones and zeros commands to create the following matrices

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Using A,B,C create

$$D = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

MATLAB Code:

```
A=eye(2)
B=ones(1,2)
C=zeros(3)
D=[C [A;B]]
```

Output:

```
A =

    1     0
    0     1

B =

    1     1

C =

    0     0     0
    0     0     0
    0     0     0

D =

    0     0     0     1     0
    0     0     0     0     1
    0     0     0     1     1
```

10	Create a Matrix $\begin{pmatrix} 6 & 9 & 12 & 15 & 18 & 21 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 2 & 1 & 0 & -1 & -2 & -3 \\ -6 & -4 & -2 & 0 & 2 & 4 \end{pmatrix}$. Evaluate the following expressions (a) A=M([1,3],[2,4]) (b) B = M(:,[1,4:6]) (c) C= M([2,3], :) **MATLAB Code:** M=[6,9,12,15,18,21;4.*ones(1,6);2,1,0,-1,-2,-3;-6,-4,-2,0,2,4] A=M([1,3],[2,4]) B = M(:,[1,4:6]) C= M([2,3], :) **Output:** M = 6 9 12 15 18 21 4 4 4 4 4 4 2 1 0 -1 -2 -3 -6 -4 -2 0 2 4 A = 9 15 1 -1 B = 6 15 18 21 4 4 4 4 2 -1 -2 -3 -6 0 2 4 C = 4 4 4 4 4 4 2 1 0 -1 -2 -3

Outcome: ***<Write in your own words on learning experience, "what you learn/doing after completion of this exercise".>***

Exercise -3

Writing scripts on MATLAB basics and Programs using functions

Creating and Running Script File: To create scripts files, you need to use a text editor. You can open the MATLAB editor in two ways:

- Using the command prompt Using the IDE
- If you are using the command prompt, type edit in the command prompt. This will open the editor. You can directly type edit and then the filename (with .m extension)

edit

Or

edit<filename>

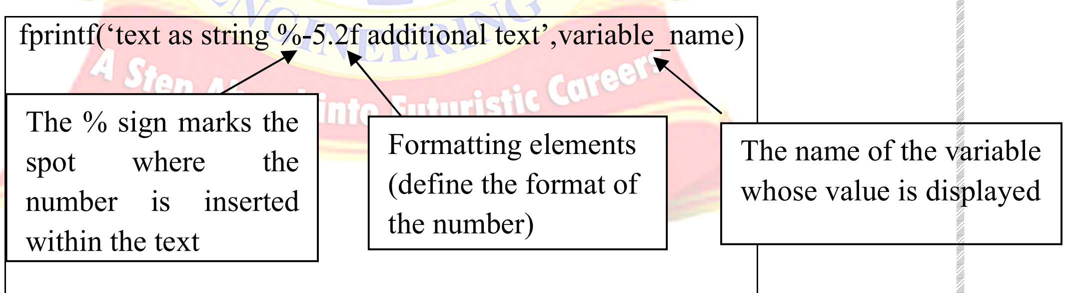
The above command will create the file in default MATLAB directory. If you want to store all program files in a specific folder, then you will have to provide the entire path. Let us create a folder named progs. Type the following commands at the command prompt(>>):

Input Command: The input() is used to read the element of a variable.

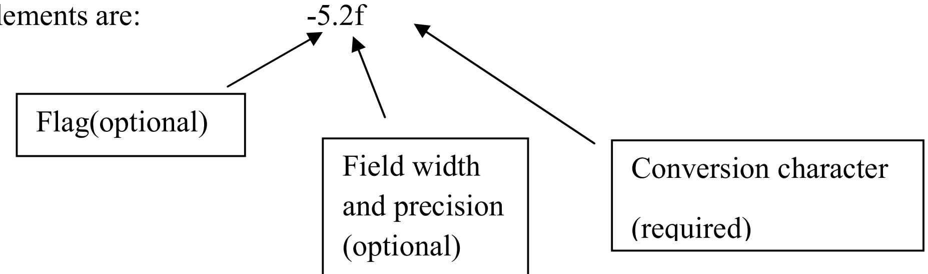
Output Commands:

(i)The disp Command: The disp command is used to display the elements of a variable without displaying the name of the variable, and to display text. The format of the disp command is: disp(name of a variable) or disp('text as string'). Every time the disp command is executed, the display it generates appears in a new line.

(ii) The fprintf**Command:**The fprintf command can be used to display output (text and data) on the screen or to save it to a file. With this command (unlike with the disp command) the output can be formatted. For example, text and numerical values of variables can be intermixed and displayed in the same line. In addition, the format of the numbers can be controlled. With many available options, the fprintf command can be long and complicated. To avoid confusion, the command is presented gradually. First, this section shows how to use the command to display text messages, then how to mix numerical data and text, next how to format the display of numbers, and finally how to save the output to a file.



The formatting elements are: -5.2f



The flag, which is optional, can be done of the following three characters:

Functions: 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

1. **if... end Statement:** An if ... end statement consists of an if statement and a boolean expression followed by one or more statements. It is delimited by the end statement.

Syntax: The syntax of an if statement in MATLAB is:

```
if <expression>
% statement(s) will execute if the boolean expression is true
<statements>
end
```

If the expression evaluates to true, then the block of code inside the if statement will be executed. If the expression evaluates to false, then the first set of code after the end statement will be executed.

2. **if...else...end Statement:** An if statement can be followed by an optional else statement, which executes when the expression is false.

Syntax: The syntax of an if...else statement in MATLAB is:

```
if <expression>
% statement(s) will execute if the boolean expression is true
<statement(s)
> else
<statement(s)>
%statement(s)will execute if the Boolean expression is false end
```

If the boolean expression evaluates to true, then the if block of code will be executed, otherwise else block of code will be executed.

3. **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
```

4. while Loop: A while loop is a repetition control structure that allows you to efficiently write a loop that needs to execute until the condition is true.
Syntax: The syntax of a for loop in MATLAB is:

```
for index = values
<program statements>

        ...
end
```

Loop Control Statements: Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed. MATLAB supports the following control statements.

Control Statement	Description
break statement	Terminates the loop statement and transfers execution to the statement immediately following the loop.
continue statement	Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

MATLAB Programs

1 Create a function file to convert temperature in degrees Fahrenheit to degree Centigrade. (C=(5/9)(F-32)). Use this function to convert 68^0F to degree centigrade.

MATLAB Code:

```
function c=f2d(F)
c=(5/9)*(F-32);
```

Output:

```
f2d(68)
ans =
   20
```

2 Create a function to obtain polar coordinates (r,θ) when Cartesian coordinates (x,y) are given. The relationship between them is given by the equations: $x = r * \cos(\theta)$, $y = r * \sin(\theta)$.

MATLAB Code:

```
function  f=p2c(r,t)
x=r*cos(t);
y=r*sin(t);
f=[x,y];
```

Output:

```
p2c(3,30)
ans =
   0.4628   -2.9641
```

3 Create a function file to find factorial of a given positive number.

MATLAB Code:

```
function f=fct(n)
x=n:-1:1;
f=prod(x);
end
```

Output:

```
fct(6)
ans =
   720
```

4

Write a function file(name it chp4one) for the function $f(x)=\frac{x\sqrt[4]{3x+5}}{(x^2+1)^2}$. The input to the function is x and the output is f(x). Write the function such that x can be a vector. Use the function to calculate: (i) f(x) for x=6. (ii) f(x) for x =1,3,5,7,9 and 11.

MATLAB Code:

```
function f=chap4one(x)
f=(x.*(3.*x+4).^(1./4))./(x.^2+1).^2;
end
```

Output:

```
chap4one(6)
ans =
   0.0095
chap4one(1:2:11)
ans =
   0.4066    0.0570    0.0154    0.0063    0.0032    0.0018
```

5

Write a function file(name it chp4three) for the function $f(x,y)=x^2-4xy+y^2$. The input to the function is x and the output is f(x,y). Use the function to calculate: (i) f(2,3) (ii) f(x,y) for x =1, y=9

MATLAB Code:

```
function f=chp4three(x,y)
f=x^2-4*x*y+y^2;
end
```

Output:

```
chp4three(2,3)
ans =
  -11
chp4three(1,9)
ans =
   46
```

6

Write a script file to find the minimum of five numbers

MATLAB Code:

```
%a script file to find the minimum of five numbers
clc; clear all;
a=input('enter the vector of length 5');
if a(1)>a(2)
    a(1)=a(2);
end
if a(1)>a(3)
    a(1)=a(3);
end
if a(1)>a(4)
    a(1)=a(4);
end
if a(1)>a(5)
    a(1)=a(5);
end
fprintf('the number is %0.0f\n',a(1))
```

Output:

```
enter the vector of length 5 [ 12 32 43 2 22]
the number is 2
```

7 Create a script file to read a number and test whether it is even or not

MATLAB Code:

```
clc
n = input('enter the value of n: ');
x=mod(n,2);
if x == 1
   fprintf('The given number %d is not even \n', n)
else
   fprintf('The given number %d is  even \n', n)
end
```

Output:

```
enter the value of n: 34
 The given number 34 is  even
```

8 Write a script file to find roots of a given quadratic equation and also displays nature of roots (Ex: real, equal; real, unequal; imaginary)

MATLAB Code:

```
clc;
clear all;
%The nature and roots of the quadratic equation
%ax^2+bx+c=0
a=input('enter the coefficient x^2 a=');
if a==0
    fprintf('invalid a\n')
else
b=input('enter the coefficient x b=');
c=input('enter the constant c=');
d=b^2-4*a*c;
if d==0
    fprintf('The roots of the given equation real and equal\n')
else if d>0
    fprintf('The roots of the given equation real and unequal\n')
    else
    fprintf('The roots of the given equation imaginary\n')
    end
end
x1=(-b+sqrt(d))/2*a;
x2=(-b-sqrt(d))/2*a;
disp(x1);
disp(x2);
end
```

Output:

```
enter the coefficient x^2 a= 1
enter the coefficient x b=5
enter the constant c=3
The roots of the given equation real and unequal
  -0.6972
  -4.3028
```

```
enter the coefficient x^2 a= 1
enter the coefficient x b=4
enter the constant c=4
The roots of the given equation real and equal
    -2

    -2

enter the coefficient x^2 a= 1
enter the coefficient x b=1
enter the constant c=1
The roots of the given equation imaginary
  -0.5000 + 0.8660i

  -0.5000 - 0.8660i
```

9 Write a script file to create a function

$$f(x)=\begin{cases} 1, \mathit{if}\ \ x<-1 \\ x^2, \mathit{if}\ -1\le x\le 2 \\ 0, \mathit{otherwise} \end{cases}$$

MATLAB Code:

```
x=input('enter the value of x);
if x<-1
    f=1;
else if x>=-1&&x<=2
        f=x^2;
    else
        f=0;
    end
end
```

Output:

```
enter the value of x 23
>> fun4
enter the value of x 23
     0

>> fun4
enter the value of x -1.5
     1
```

10 Write a script file to calculate area of a circle by taking radius as input and using '**if-else statement**'. (Note: If negative radius is given, print an error message 'Invalid Radius, give a non-negative radius')

MATLAB Code:

```
clc;clear all;
%Area of the circle
r=input('Enter the  positive radius of a circle R=');
if r<0
    fprintf('Invalid raidus\n')
else
    a=22*r^2/7;
fprintf('Area of the circle is A=%0.4f\n',a);
end
```

	Output: Enter the positive radius of a circle R=-1 Invalid raidus Enter the positive radius of a circle R= 2.3 Area of the circle is A=16.6257

11	Write a script file to find the sum of the series $\frac{1}{1^3}+\frac{1}{2^3}+\frac{1}{3^3}+\cdots$ up to n^{th} term where n=100, 500.
	MATLAB Code:
	`clc; clear all;`
	`n=input('Enter no of terms in the sequence n =')`
	`sum=0;`
	`for i=1:n`
	`     sum=sum+(1/i^3);`
	`end`
	`fprintf('The sum of the %d terms is %f\n',n,sum)`
	Output:
	Enter no of terms in the sequence n =100
	The sum of the 100 terms is 1.202007
	
	Enter no of terms in the sequence n =500
	The sum of the 500 terms is 1.202055

12	Write a script file to generate Fibonacci sequence $F_n=F_{n-1}+F_{n-2}$ using for loop. Find (i) 5th term (ii) 8th term (iii) sum upto 10th term (iv) product upto 20th term of the sequence.
	MATLAB Code:
	`clear all; clc`
	`n= input('enter the value of n: ');`
	`fibo = [1,1];`
	`for i=3:n`
	`     fibo(i) = fibo(i-1)+fibo(i-2);`
	`end`
	`fprintf('The Fibonacci sequence  is: ')`
	`disp(fibo)`
	`fprintf('enter the nth term of the Fibonacci sequence less than %d: ',n);`
	`k=input('k=');`
	`fprintf('\n The Fibonacci sequence  is %d: ',fibo(k))`
	`fprintf('\n The sum of the %d terms of Fibonacci sequence  is %d:',n,sum(fibo))`
	`fprintf('\n The product of the %d terms Fibonacci sequence  is %d\n: ',n, prod(fibo))`
	Output:
	
	enter the value of n: 20
	The Fibonacci sequence is: Columns 1 through 7
	
	1 1 2 3 5 8 13
	
	Columns 8 through 14
	
	21 34 55 89 144 233 377
	
	Columns 15 through 20

	610 987 1597 2584 4181 6765 enter the nth term of the Fibonacci sequence less than 20: k=5 The Fibonacci sequence is 5: The sum of the 20 terms of Fibonacci sequence is 17710: The product of the 20 terms Fibonacci sequence is 9.692987e+36

13	An object thrown vertically with a speed v0 reaches a height h at time t, where $h = v_0 t - \frac{1}{2} g t^2$. Write and test a function that computes the time t required to reach a specified height h, for a given value of v0. The function's inputs should be h, v0, and g. Test your function for the case where h = 100 m, v0 = 50 m/s, and g = 9.81 m/s^2. Interpret both answers.
	MATLAB Code:
	`t=5;`
	`x0=10;`
	`v0=15;`
	`a=-9.81;`
	`x=x0+v0*t+a*(t^2)/2`
	Output:
	x =
	-37.6250

14	A Model for exponential growth or decay of a quantity is given by $A(t) = A_0 e^{kt}$, where A(t) and A_0 are the quantity at time t and time 0, respectively, and k is constant unique to the specific application. Write the user-defined function A(t) and calculate the population of a village in the year 2000 when the population of a village was 67,000 in the year 1980 and 79,000 in the year 1986.
	MATLAB Code:
	`%Model for exponential growth or decay of a quantity`
	`clc;clear;`
	`n0=input('enter the initial year, n0=');`
	`A0=input('enter the intial population, A0=');`
	`n=input('enter the time when the populations is observed after n0, n=');`
	`A1=input('enter the  population when t=n, A1=');`
	`syms k0`
	`k=solve(A1-A0*exp(k0*n));`
	`t=input('enter the time to predict the population t=');`
	`Anew=A0*exp(k*t);`
	`fprintf('the population in the year %d is %0.0f\n',n0+t,double(Anew));`
	
	Output:
	enter the initial year, n0=1980
	enter the intial population, A0=67000
	enter the time when the populations is observed after n0, n=6
	enter the population when t=n, A1=79000
	enter the time to predict the population t=20
	the population in the year 2000 is 116033

15	Create a function called cone that computes the volume V of a cone whose height is h and whose radius is r. (Do not forget to check if a file already exists by that name!) The volume is given by $V=\pi r^2 h/3$ Test case: h = 30, r = 5, V = 785.3982
	MATLAB Code:
	`function v=vol(r,h)`
	`v=pi*r^2*h/3;`
	`end`
	
	Output:
	vol(5,30)
	ans =
	785.3982

Outcome: ***<Write in your own words on "what you learn/doing after completion of this exercise".>***

Exercise -4

Topic: Symbolic Mathematics

The symbolic Mathematics: For using symbolic Mathematics tools, first write the command "syms" to variable declaration.

Solving Basic Algebraic Equations in MATLAB: The solve() function is used for solving algebraic equations. In its simplest form, the 'solve' function takes the equation enclosed in quotes as an argument.

Solving Basic Algebraic Equations: The roots() is used for solving algebraic equations

Expanding and Collecting Equations in MATLAB: The expand() and the collect() commands expands and collects an equation respectively. The following example demonstrates the concepts: When you work with many symbolic functions, you should declare that your variables are symbolic.

Factorization and Simplification of Algebraic Expressions: The factor() function factorizes an expression and the simplify() function simplifies an expression.

Differentiation: The diff function, when applied to a symbolic expression, provides a symbolic derivative. diff(E) differentiates a symbolic expression E with respect to its free variable as determined by find sym.

- diff(E,v) Differentiates E with respect to symbolic variable v.
- diff(E,n) Differentiates E n times for positive integer n.
- diff(E,v,n) Differentiates E n times with respect to symbolic variable v.

Integration: The int function, when applied to a symbolic expression, provides a symbolic integration. int(E) gives indefinite integral of symbolic expression E with respect to its symbolic variable as defined by findsym. If E is a constant, the integral is with respect to x. int(E,v) gives indefinite integral of E with respect to scalar symbolic variable v. int(E,a,b) gives definite integral of E with respect to its symbolic variable from a to b, where a and b are each double or symbolic scalars. int(E,v,a,b) gives definite integral of E with respect to v from a to b.

MATLAB Programs

1 Solve the following linear system of equations: x + 3y -2z = 5, 3x + 5y + 6z = 7, 2x + 4y + 3z = 8.

MATLAB Code:

```
clc
syms x y z
e=solve(x + 3*y -2*z == 5,  3*x + 5*y + 6*z == 7, 2*x + 4*y + 3*z ==
8);
fprintf('x=%f,y=%f,z=%f\n',double(e.x),double(e.y),double(e.z))
```

Output:

x=-15.000000,y=8.000000, z=2.000000

2 Use suitable MATLAB command to expand the expression (x+1.4)(x-0.4)x(x+0.6)(x-1.4)

MATLAB Code:

```
clc
syms x
expand((x+1.4)*(x-0.4)*x*(x+0.6)*(x-1.4))
```

Output:

ans =

	x^5 + x^4/5 - (11*x^3)/5 - (49*x^2)/125 + (294*x)/625

3	Use MATLAB command to multiply the polynomials $2x^2+3$, $x^3+3.5x^2+5x-16$
	MATLAB Code:
	`clc`
	`syms x`
	`expand((2*x^2+3)*(x^3+3.5*x^2+5*x-16))`
	Output:
	ans =
	2*x^5 + 7*x^4 + 13*x^3 - (43*x^2)/2 + 15*x - 48

4	Factorise the following polynomials using MATLAB command
	(a) x^2-x-6 (b) x^3+2x^2+3x+2 (c) $(x^4 -2x^3 +3x^2 -4x +5)^3$
	MATLAB Code:
	`clc`
	`syms x`
	`factor(x^2-x-6)`
	`factor(x^3+2*x^2+3*x+2)`
	`factor((x^4 -2*x^3 +3*x^2 -4*x +5)^3)`
	Output:
	ans =
	
	[x + 2, x - 3]
	
	ans =
	
	[x + 1, x^2 + x + 2]
	
	ans =
	[x^4 - 2*x^3 + 3*x^2 - 4*x + 5, x^4 - 2*x^3 + 3*x^2 - 4*x + 5, x^4 - 2*x^3 + 3*x^2 - 4*x + 5]

5	Find the roots of the following equations:
	(a) $x^4 + x^3 - 43x^2 + 23x + 210 = 0$ (b) $2x^4 = 5$ (c) x^{10}-x-1 = 0
	MATLAB Code:
	`clc; clear;`
	`syms x`
	`solve(x^4 + x^3 - 43*x^2 + 23*x + 210==0)`
	`solve(2*x^4 - 5==0)`
	`solve(x^10-x-1==0)`
	Output:
	ans =
	-7
	-2
	3
	5
	
	ans =
	-(2^(3/4)*5^(1/4))/2
	(2^(3/4)*5^(1/4))/2
	-(2^(3/4)*5^(1/4)*1i)/2
	(2^(3/4)*5^(1/4)*1i)/2

ans =
  root(z^10 - z - 1, z, 1)
 root(z^10 - z - 1, z, 2)
 root(z^10 - z - 1, z, 3)
 root(z^10 - z - 1, z, 4)
 root(z^10 - z - 1, z, 5)
 root(z^10 - z - 1, z, 6)
 root(z^10 - z - 1, z, 7)
 root(z^10 - z - 1, z, 8)
 root(z^10 - z - 1, z, 9)
root(z^10 - z - 1, z, 10)

6 Find the derivatives of the following functions using MATLAB commands

(a) $3t^2 + 2/t^2$ (b) $(x+2)(x^2+3)$ (c) $(\sin^2 x)(\cos 2x)$ (d) $(x^2-4) / (x+5)(2x^3+5)$

MATLAB Code:

```
clc; clear;
syms x t
a=diff(3*t^2 + 2/t^2)%the derivate of (a)
b=diff((x+2)*(x^2+3))%the derivate of (b)
c=diff(sin(2*x)*cos(2*x))%the derivate of (c)
d=diff((x^2-4)/((x+5)*(2*x^3+5)))%the derivate of (d)
```

Output:

a =
 6*t - 4/t^3
 b =
2*x*(x + 2) + x^2 + 3
c =
 2*cos(2*x)^2 - 2*sin(2*x)^2
 d =
(2*x)/((2*x^3 + 5)*(x + 5)) - (x^2 - 4)/((2*x^3 + 5)*(x + 5)^2) - (6*x^2*(x^2 - 4))/((2*x^3 + 5)^2*(x + 5))

7 Evaluate the following integrals using MATLAB command

(a) $\int \frac{x^3}{\sqrt{1-x^2}}dx$ (b) $\int x^2 \cos x dx$ (c) $\int e^{-2x} \sin 5x dx$ (d) $\int \frac{x^2+4}{(x^2-3)(x+7)}dx$

MATLAB Code:

```
clc
syms x
a=int(x^3/sqrt(1-x^2))%integral of (a)
b=int(x^2*cos(x))%integral of (b)
c=int(exp(-2*x)*sin(5*x))%integral of (c)
d=int((x^2+4)/((x^2-3)*(x+7)))%integral of (d)
```

Output:

a =
-((1 - x^2)^(1/2)*(x^2 + 2))/3
 b =
sin(x)*(x^2 - 2) + 2*x*cos(x)
c =
-(exp(-2*x)*(5*cos(5*x) + 2*sin(5*x)))/29
d =
(53*log(x + 7))/46 - log(x + 3^(1/2))*((49*3^(1/2))/276 + 7/92) + log(x - 3^(1/2))*((49*3^(1/2))/276 - 7/92)

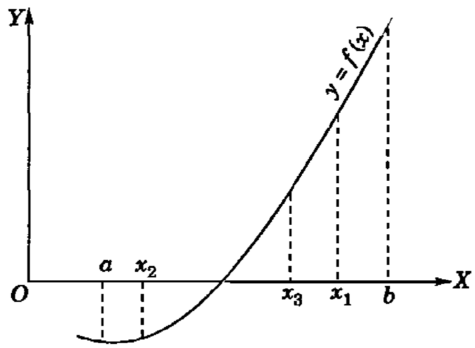
Outcome:*<Write in your own words on learning experience, "what you learn/doing after completion of this exercise".>*

Exercise -5

Topic: Programs on Root Finding:

(1) Bisection method:

This method is useful to find the root of an equation f (x) = 0 which lies between a and b. If f(x) is continuous between a and b, and f(a) and f(b) are of opposite signs then the root lies in between a and b. Let f(a) be negative and f(b) be positive. Then the first approximation to the root is $x_1 = \frac{(a+b)}{2}$. If $f(x_1) = 0$, then x_1 is a root of f(x) = 0. Otherwise, the root lies between a and x_1 or x_1 and b according as $f(x_1)$ is positive or negative. Then we bisect the interval as before and continue the process until the root is found to desired accuracy.



If $f(x_1)$ is +ve, so that the root lies between a and x_1. Then the second approximation to the root is $x_2 = \frac{(a+x_1)}{2}$. If $f(x_2)$ is - ve, the root lies between x_1 and x_2. Then the third approximation to the root is $x_3 = \frac{(x_1+x_2)}{2}$ and so on.

(2) Method of false position or Regula-falsi method:

Let us consider an equation of the form f (x) = 0 and we choose two points x_0 and x_1 such that $f(x_0)$ and $f(x_1)$ are of opposite signs i.e., the graph of y = f (x) crosses the x-axis between these points. This indicates that a root lies between x_0 and x_1 consequently $f(x_0)\, f(x_1) < 0$.

$$x = \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)}$$

The next approximation root is $x_2 = \frac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)}$

Now if $f(x_0)$ and $f(x_2)$ are of opposite signs, then the root lies between x_0 and x_2, replacing x_1 by x_2. The next approximation is x_3. This procedure is repeated till the root is found to desired accuracy.

1. Write a MATLAB Code to find the locations of the roots of the given equations.

MATLAB Code:

```
%A script file to the find the location of the roots
clc; clear;
syms x
f=input('enter the equation f(x)=0,f(x)=');
x_min=input('enter the lower limit, xmin=');
x_max=input('enter the upper limit, xmax=');
for k=x_min:x_max
    if subs(f,x,k)*subs(f,x,k-1)<0
        x1=k-1;
        x2=k;
        break;
    end

end
fprintf('the requird root lies between x1=%d and x2=%d\n',x1,x2);
```

Output:

enter the equation f(x)=0,f(x)=x^3-5*x+5
enter the lower limit, xmin=-10
enter the upper limit, xmax=10
the requird root lies between x1=-3 and x2=-2

2. Write a MATLAB code to find the root of the equation by using Bisection method and Test Case find the root of the equation x^3-3x +5=0.

MATLAB CODE USING WHILE:

```
%A script file for Bisecton Method
clc; clear;
syms x
f=input('enter the equation f(x)=0,f(x)=');
x_min=input('enter the lower limit, xmin=');
x_max=input('enter the upper limit, xmax=');
for k=x_min:x_max
    if subs(f,x,k)*subs(f,x,k-1)<0
        x1=k-1;
        x2=k;
        break;
    end

end
fprintf('the requird root lies between x1=%d and
x2=%d\n',double(x1),double(x2));
xnew=1;
i=0;
fprintf('x1\t      x2\t        f(x)\t\n');
while abs(subs(f,x,xnew))>0.0001
    xnew=(x1+x2)/2; %bisection formula
    fprintf('%0.4f\t  %0.4f\t
%0.4f\t\n',double(x1),double(x2),double(subs(f,x,xnew)));
    if subs(f,x,x1)*subs(f,x,xnew)<0
        x2=xnew;
    else
        x1=xnew;
    end
  i=i+1;

end
fprintf('the number of iterations is i=%d\n',i);
fprintf('the requird root is x=%0.4f\n',xnew);
```

Output:

```
enter the equation f(x)=0,f(x)=x^3-3*x+5
enter the lower limit, xmin=-10
enter the upper limit, xmax=10
the requird root lies between x1=-3 and x2=-2
x1        x2          f(x)
-3.0000       -2.0000          -3.1250
-2.5000       -2.0000          0.3594
-2.5000       -2.2500          -1.2715
-2.3750       -2.2500          -0.4290
-2.3125       -2.2500          -0.0281
-2.2813       -2.2500          0.1673
-2.2813       -2.2656          0.0700
-2.2813       -2.2734          0.0211
-2.2813       -2.2773          -0.0035
-2.2793       -2.2773          0.0088
-2.2793       -2.2783          0.0026
-2.2793       -2.2788          -0.0004
-2.2791       -2.2788          0.0011
-2.2791       -2.2789          0.0003
-2.2791       -2.2790          -0.0000
the number of iterations is i=15
the requird root is x=-2.2790
```

MATLAB Code using for loop:

```
%A script file for finding the location of the roots
clear all; clc;
syms x
f=input('Enter the f(x) of the equation f(x)=0\n f(x)=');
%xmax=input('Enter the maximum value');
y(1)=subs(f,x,-10);
k=2;
for i=-10:1:10
   k=k+1;
   y(k)=subs(f,x,i);
   if y(k)*y(k-1)<0
      fprintf('The required root lies between x1=%d and x2=%d\n',i-1,i);
      x1=i-1;x2=i;
      break
   end
end
fprintf('i     \tx1  \t    x2      \t f\n')
 for i = 1: 100
   xh =(x1+x2)/2; % bisection
   if subs(f,x,x1)*subs(f,x,xh) < 0
      x2 = xh;
   else
```

```
        x1 = xh;
    end
    fprintf('%d \t %-1.4f \t %-1.4f  \t %-1.4f\n',i,x1,x2,subs(f,x,xh))
    if abs(subs(f,x,xh)) < 0.0001
        break
    end
end
fprintf('The root: %-1.4f\nThe number of Iterations: %d\n',x1,i)
```

Output

```
Enter the f(x) of the equation f(x)=0
 f(x)=x^3-3*x+5
The required root lies between x1=-3 and x2=-2
i        x1          x2          f
1        -2.5000     -2.0000     -3.1250
2        -2.5000     -2.2500     0.3594
3        -2.3750     -2.2500     -1.2715
4        -2.3125     -2.2500     -0.4290
5        -2.2813     -2.2500     -0.0281
6        -2.2813     -2.2656     0.1673
7        -2.2813     -2.2734     0.0700
8        -2.2813     -2.2773     0.0211
9        -2.2793     -2.2773     -0.0035
10       -2.2793     -2.2783     0.0088
11       -2.2793     -2.2788     0.0026
12       -2.2791     -2.2788     -0.0004
13       -2.2791     -2.2789     0.0011
14       -2.2791     -2.2790     0.0003
15       -2.2790     -2.2790     -0.0000
The root: -2.2790
    The number of Iterations: 15
```

3. Write a MATLAB code to find real root of the equation by False position method and Test Case find the root of the equation $x^3-3x+5=0$.

MATLAB Code:

```
%A script file for Regula falsi Method
clc; clear;
syms x
f=input('enter the equation f(x)=0,f(x)=');
x_min=input('enter the lower limit, xmin=');
x_max=input('enter the upper limit, xmax=');
for k=x_min:x_max
    if subs(f,x,k)*subs(f,x,k-1)<0
        x1=k-1;
        x2=k;
        break;
    end

end
fprintf('the requird root lies between x1=%d and
x2=%d\n',double(x1),double(x2));
xnew=1;
i=0;
fprintf('x1\t      x2\t        f(x)\t\n');
while abs(subs(f,x,xnew))>0.0001
    xnew=(x1*subs(f,x,x2)-x2*subs(f,x,x1))/(subs(f,x,x2)-subs(f,x,x1));
%Regula Falsi formula
    fprintf('%0.4f\t  %0.4f\t
%0.4f\t\n',double(x1),double(x2),double(subs(f,x,xnew)));
    if subs(f,x,x1)*subs(f,x,xnew)<0
        x2=xnew;
    else
        x1=xnew;
    end
  i=i+1;

end
fprintf('the number of iterations is i=%d\n',i);
fprintf('the requird root is x=%0.4f\n',double(xnew));
```

Output:

enter the equation f(x)=0,f(x)=x^3-3*x+5
enter the lower limit, xmin=-10
enter the upper limit, xmax=10
the requird root lies between x1=-3 and x2=-2

x1	x2	f(x)
-3.0000	-2.0000	1.0950
-3.0000	-2.1875	0.3518
-3.0000	-2.2506	0.1084
-3.0000	-2.2704	0.0329
-3.0000	-2.2764	0.0100
-3.0000	-2.2782	0.0030
-3.0000	-2.2788	0.0009
-3.0000	-2.2789	0.0003
-3.0000	-2.2790	0.0001

the number of iterations is i=9
the requird root is x=-2.2790

Outcome:<*Write in your own words on learning experience, "what you learn/doing after completion of this exercise".*>

Exercise -6

Topic: Programs on Root Finding:

(1) Iteration method:

Let us consider an equation f(x) = 0

Express f(x) as x = ø(x) such that $|\emptyset^I(X)| < 1$

Let the initial solution be x_0 for f(x)

The next approximations are $x_1 = ø(x_0)$

$x_2 = ø(x_1)$

$x_3 = ø(x_2)$

.

.

$x_{n+1} = ø(x_n)$, where n = 0, 1 , 2 ,…

The above formula is called as Iteration formula.

Note: The iteration formula can be applied only if ø(x) is converges i.e, $|\emptyset^I(X)| < 1$.

(2) Newton - Raphson method:

Let us consider an equation of the form f (x) = 0 and x_0 be an approximate root of the equation f(x) = 0.

∴ The first approximation is $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$

The second approximation is $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$

In general, $x_{n+1} = x_1 - \frac{f(x_n)}{f'(x_n)}$

This is known as the Newton-Raphson formula or Newton's iteration formula.

1. Write a MATLAB code to find a real root of the equation by Iteration Method and Test Case find the root of the equation x^3-5x +5=0

   MATLAB Code:

```
% A Script file for Iterative Method
clear all; clc;
syms x
f=input('Enter the f(x) of the equation f(x)=0 where\n f(x)=');
%xnew=input('Enter the intial approximation');
y(1)=subs(f,x,-10);
k=2;
for i=-10:1:10
   k=k+1;
   y(k)=subs(f,x,i);
   if y(k)*y(k-1)<0
      fprintf('The required root lies between x1=%d and x2=%d\n',i-1,i);
      x1=i-1;x2=i;
      break
   end
end
f=input('Enter the f(x) of the equation x=f(x)where\n f(x)=');
max_itr=input('Enter the maximum number of iterations');
%tol=input('Enter the tolerance');
tol=0.0001;
xnew=x1;
df=diff(f,x);
```

```
fprintf('The iterations are:\n')
while abs(subs(df,x,xnew))>=1
    fprintf('The given function does not converge\n')
  xnew=input('Enter another intial approximation');
end
for i=1:max_itr
    xold=xnew;
    xnew=subs(f,x,xnew);
    err=abs(xold-xnew);
    if (err>tol)
        fprintf('%d->x%d=%f\n',i,i,xnew)
    else
        break
    end

end
fprintf('The required root is x =%10.6f\n',xnew)
```

Output

```
Enter the f(x) of the equation f(x)=0 where
 f(x)=x^3-3*x-5
The required root lies between x1=2 and x2=3
Enter the f(x) of the equation x=f(x)where
 f(x)=(3*x+5)^(1/3)
Enter the maximum number of iterations 10
The iterations are:
1->x1=2.223980
2->x2=2.268372
3->x3=2.276967
4->x4=2.278624
5->x5=2.278943
The required root is x =   2.279004
The root is 2.094544
```

2. Write a MATLAB code to find real root of the equation by Newton-Raphson method and Test Case find the root of the equation x sin x + cos x=0.

   MATLAB Code:

Program:

```
%A script file for finding a real roots using newton raphson method
clear all;clc;
syms x
f=input('enter the function f(x) corresponding to the equation f(x)=0\n f(x)=');
k=1;
x_min=-10;%input('enter the starting value');
x_max=10;%input('enter the final value');
y(1)=subs(f,x,x_min);
for i=x_min:1:x_max
    k=k+1;
    y(k)=subs(f,x,i);
    if y(k)*y(k-1)<0
```

```
        fprintf('The required root lies between\n x1=%d, x2=%d\n',i,i-1)
        xnew=i;
        break
    end
end
df=diff(f,x);
for i=1:100
  xold=xnew;
  xnew=xnew-subs(f,x,xnew)/subs(df,x,xnew);
  if (abs(xold-xnew)>0.0001)
    fprintf('%d->x%d=%f\n',i,i,xnew)
  else
      break
  end

 end
fprintf('the required root is %f \n ',xnew)
```

Output

```
enter the function f(x) corresponding to the equation f(x)=0
 f(x)=x^3-3*x+5
The required root lies between
 x1=-2, x2=-3
1->x1=-2.333333
2->x2=-2.280556
3->x3=-2.279020
the required root is -2.279019
```

Outcome:

<Write in your own words on
"what you learn/doing after completion of this exercise".>

Exercise -7

Topic: Program on Interpolation:

(1) **Newton's Forward Interpolation Formula:** Let $y_0, y_1, \ldots, y_n$ be the values of f(x) corresponding to the arguments $x_0, x_0+h, x_0+2h, \ldots, x_0+nh$, with equally spaced, then the Newton's forward interpolation polynomial y (x) is $y(x) = y_0 + p\,\Delta y_0 + \frac{p(p-1)}{2}\Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!}\Delta^3 y_0 + \frac{p(p-1)(p-2)(p-3)}{4!}\Delta^4 y_0 + \ldots,$ Where $p = \frac{x-x_0}{h}$

(2) **Newton's backward interpolation formula:** Let $y_0, y_1, \ldots, y_n$ be the values of f(x) corresponding to the arguments $x_0, x_1, \ldots, x_n$ with equally spaced, then the Newton's backward interpolation polynomial y (x) is $y(x) = y_n + p\,\nabla y_n + \frac{p(p+1)}{2}\nabla^2 y_n + \frac{p(p+1)(p+2)}{3!}\nabla^3 y_n + \frac{p(p+1)(p+2)(p+3)}{4!}\nabla^4 y_n + \ldots,$ where $p = \frac{x-x_n}{h}$

(3) **Lagrange's interpolation formula (interpolation with unevenly spaces points):** Let $y_0, y_1, \ldots, y_n$ be the values of f(x) corresponding to the arguments $x_0, x_1, \ldots, x_n$ with not necessarily equally spaced, then the Lagrange's interpolation polynomial is

$$y(x) = \frac{(x-x_1)(x-x_2)(x-x_3)\ldots(x-x_n)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)\ldots(x_0-x_n)} f(x_0) + \frac{(x-x_0)(x-x_2)(x-x_3)\ldots(x-x_n)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)\ldots(x_1-x_n)} f(x_1) + \frac{(x-x_0)(x-x_1)(x-x_3)\ldots(x-x_n)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)\ldots(x_2-x_n)} f(x_2) + \ldots + \frac{(x-x_0)(x-x_1)(x-x_2)\ldots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)(x_n-x_2)\ldots(x_n-x_{n-1})} f(x_n)$$

1 Write a MATLAB code to find the unknown value for the given data by using Newton's forward difference formula and test case: find y(1.5) for the data

x	0	1	2	3	4
y	3	6	11	18	27

MATLAB Code:

Program:

```
clear all; clc;
% Script for Newton's Forward Interpolation formula.
% x and y are two Row Matrices and p is point of interpolation
% Example
% >> x=[10,20,30,40,50]
% >> y=[-9,-41,-189,9,523]
% here h=10;
x=input('enter the row vector x');
y=input('enter the row vector y');
p=input('enter the unknown x');
h=input('enter the length');
n = length(x);
a(1) = y(1);
for k = 1 : n - 1
  d(k, 1) = (y(k+1) - y(k));
end
for j = 2 : n - 1
  for k = 1 : n - j
    d(k, j) = (d(k+1, j - 1) - d(k, j - 1));
  end
end
d
for j = 2 : n
```

```
  a(j) = d(1, j-1);
end
Df(1) = 1;
c(1) = a(1);
for j = 2 : n
  Df(j)=(p - x(j-1)) .* Df(j-1);
  c(j) = a(j) .* Df(j)/(factorial(j-1)*h^(j-1));
end
fp=sum(c);
fprintf('The required y value at x=%f is %f',p,fp);
```

Output:

```
enter the row vector x0:4
enter the row vector y[3 6 11 18 27]
enter the unknown x1.5
enter the length1

d =

    3    2    0    0
    5    2    0    0
    7    2    0    0
    9    0    0    0

The required y value at x=1.500000 is 8.250000
```

2. Write a MATLAB code to find the unknown value for the given data by using Newton's backward difference formula and test case to the following problem. The population of a town in the decimal census was given below. Estimate the population for the year 1925 using Newton.s Backward Interpolation formula

Year x	1891	1901	1911	1921	1931
Population y (thousands)	46	66	81	93	101

MATLAB Code:

Program:

```
clear all; clc;
% Script for Newton's Forward Interpolation formula.
% x and y are two Row Matrices and p is point of interpolation
% Example
% x=100:50:400;
% y=[10.63,13.03,15.04,16.81,18.42,19.90,21.27];
% h=50;
% p=410;
% here h=10;
x=input('enter the row vector x');
y=input('enter the row vector y');
p=input('enter the unknown x');
h=input('enter the length');
```

```
n = length(x);
a(1) = y(n);
for k = 1 : n - 1
  d(k, 1) = (y(k+1) - y(k));
end
for j = 2 : n - 1
  for k = 1 : n - j
    d(k, j) = (d(k+1, j - 1) - d(k, j - 1));
  end
end
d
for j = 2 : n-1
  a(j) = d(n-j+1, j-1);% delta y0, delta^2 y0,...
end
Df(1) = 1;
c(1) = a(1);
for j = 2 : n-1
  Df(j)=(p - x(n+2-j)) .* Df(j-1);
  c(j) = a(j) .* Df(j)/(factorial(j-1)*h^(j-1));
end
fp=sum(c);

fprintf('The required y value at x=%f is %f\n',p,fp);
```

Output:

```
enter the row vector x 1891:10:1931
enter the row vector y [46,66,81,93,101]
enter the unknown x 1925
enter the length10

d =

   20   -5    2   -3
   15   -3   -1    0
   12   -4    0    0
    8    0    0    0

The required y value at x=1925.000000 is 96.736000
```

3. Write a MATLAB code to find the unknown value for the given data by using using Lagrange's interpolation formula and test case to evaluate f(10) given f(x) = 168, 192, 336 at x = 1, 7, 15 respectively using Lagrange's interpolation formula.

MATLAB Code:

Program:

```
clear all; clc;

x=input('enter the column matrix x');
y=input('enter the column matrix  y with the same dimension of x');
a=input('enter the unknown');;
% Coefficients of the Lagrange interpolating polynomial.
n=length(x);
p=0;
for k=1:n
      b(k)=1;
     d(k)=1;
     for j=1:n
          if j~= k
             b(k)=b(k)*(x(k)-x(j));
            d(k)=d(k)*(a-x(j));
          end
     end
    c(k)=y(k)/b(k);
    p=p+c(k)*d(k);
end
fprinitf('The coefficients of Lagrange's Polynomial are ', c)
fprintf('\n p(a)= %10.6f',p)
fprintf('\n')
```

Output:

```
enter the column matrix x
[1;5;7]
enter the column matrix  y with the same dimension of x
[168;192;336]
enter the unknown
10
c =
     7   -24    28
 p(a)= 717.000000
```

Outcome:

Output:

Outcome:*<Write in your own words on learning experience, "what you learn/doing after completion of this exercise".>*

Exercise -8

Topic: Program on Numerical Integration:

<u>Numerical integration:</u> The process of evaluating a definite integral from a set of tabulated values of the integrand f(x) is called numerical integration. This process when applied to a function of a single variable is known as quadrature.

Trapezoidal Rule:

$\int_a^b f(x)\text{dx} = \frac{h}{2}[(y_0 + y_n) + 2\,(y_1 + y_2 + y_3 + y_4 + .. + y_{n-1})]$

Simpson's One-Third Rule:

$\int_a^b f(x)\text{dx} = \frac{h}{3}\,[(y_0 + y_n) + 4\,(y_1 + y_3 + .. + y_{n-1}) + 2(y_2 + y_4 + .. + y_{n-2})]$

Simpson's Three-Eighth Rule:

$\int_a^b f(x)\text{dx} = \frac{3h}{8}\,[(y_0 + y_n) + 3\,(y_1 + y_2 + y_4 + .. + y_{n-1}) + 2(y_3 + y_6 + .. + y_{n-3})]$

(1) Write a MATLAB code to evaluate definite integral using Trapezoidal rule and test case: evaluate $\int_0^1 \frac{dx}{1+x}$.

MATLAB Code:

Program:

```
clear all; clc;
%Trapezoidal rule
f=input('Enter the function f(x)=@(x)1/(1+x)\nf(x)=');
a = input('\nEnter the lower limit of the integral\na=');
b = input('\nEnter the upper limit of the integral\nb=');
n = input('\nEnter the number of interval \n n=');
h = (b - a)/n;
s = f(a)+f(b);
for i = 1:n-1
  s = s + 2*f(a+i*h);
end
I = h/2 * s;
fprintf('\n The requried integral value is%10.4f\n',I)
----------------------
```

Output:

```
Enter the function f(x)=@(x)1/(1+x)
f(x)=@(x)1/(1+x)
Enter the lower limit of the integral
a=0
Enter the upper limit of the integral
b=1
Enter the number of interval
 n=6
 The requried integral value is    0.6949
```

(2) Write a MATLAB code to evaluate definite integral using Simpson's 1/3rd rule and test case: evaluate $\int_0^1 \sqrt{1+x^3}\,dx$ by using Simpson's 1/3 rule.

MATLAB Code:

Program:

```
clear all; clc;
% A script file for Simpson's1/3rd rule
f=input('Enter the function f(x)=@(x)1/(1+x)\nf(x)=');
a = input('\nEnter the lower limit of the integral\na=');
b = input('\nEnter the upper limit of the integral\nb=');
n = input('\nEnter the number of interval \n n=');
h = (b - a)/n;
s = f(a)+f(b);
for i = 1:2:n-1
  s = s + 4*f(a+i*h);
end
for i = 2:2:n-2
  s = s + 2*f(a+i*h);
end
I = h/3 * s;
fprintf('\n The requried integral value is I=%10.4f\n',I)
```

Output:

```
Enter the function f(x)=@(x)1/(1+x)
f(x)=@(x)sqrt(1+x^3)

Enter the lower limit of the integral
a=0

Enter the upper limit of the integral
b=1

Enter the number of interval
 n=10

 The requried integral value is I=    1.1114
```

(3) Write a MATLAB code to evaluate definite integral using Simpson's 3/8th rule and test case: evaluate $\int_0^1 \sqrt{1+x^4}\,dx$.

MATLAB Code:

Program:

```
clear all; clc;
% A script file for Simpson's 3/8 th rule
f=input('Enter the function f(x)=@(x)1/(1+x)\nf(x)=');
a = input('\nEnter the lower limit of the integral\na=');
b = input('\nEnter the upper limit of the integral\nb=');
n = input('\nEnter the number of interval \n n=');
h = (b - a)/n;
s = f(a)+f(b);
for i = 1:3:n-1
  s = s + 3*f(a+i*h);
end
for i = 2:3:n-1
  s = s + 3*f(a+i*h);
end
for i = 3:3:n-1
  s = s + 2*f(a+i*h);
end
I = 3*h/8 * s;
fprintf('\n The requried integral value is I=%10.4f\n',I)
```

Output:

```
Enter the function f(x)=@(x)1/(1+x)
f(x)=@(x)sqrt(1+x^4)
Enter the lower limit of the integral
a=0
Enter the upper limit of the integral
b=1
Enter the number of interval
 n=6
 The requried integral value is I=    1.0894
```

Outcome:

<Write in your own words on learning experience, "what you learn/doing after completion of this exercise".>

Exercise -9

Topic: Program on Numerical Solutions of Ordinary Differential Equations:

Euler's method:

The numerical solution of the differential equation

$\frac{dy}{dx}$ = f(x,y), given the initial condition y (x_0) = y_0

The Euler's formula is $y_{n+1} = y_n + h\,f(x_n + y_n), n = 0, 1, 2, \ldots..$

Modified Euler's method:

Consider the numerical solution of the differential equation

$\frac{dy}{dx}$ = f(x,y), given the initial condition y (x_0) = y_0

To find $y(x_1) = y_1$: put $x = x_1 = x_0 + h$

$$y_1^{(0)} = y_0 + hf(x_0, y_0)$$

$$y_1^{(1)} = y_0 + \frac{h}{2}\left[f(x_0, y_0) + f\left(x_1, y_1^{(0)}\right)\right]$$

$$y_1^{(2)} = y_0 + \frac{h}{2}\left[f(x_0, y_0) + f\left(x_1, y_1^{(1)}\right)\right]$$

$$y_1^{(3)} = y_0 + \frac{h}{2}\left[f(x_0, y_0) + f\left(x_1, y_1^{(2)}\right)\right]$$

$$y_1^{(k+1)} = y_0 + \frac{h}{2}\left[f(x_0, y_0) + f\left(x_1, y_1^{(k)}\right)\right]$$

If any two successive values of $y_1^{(k)}, y_1^{(k+1)}$ are sufficiently close to one another, we will take the common value as y_1.

To find $y(x_2) = y_2$: put $x = x_2 = x_1 + h$

We use the above procedure again.

Runge – Kutta methods (RK methods):

Second order RK method: $y_1 = y_0 + \frac{1}{2}(k_1 + k_2)$

Where $k_1 = hf(x_0, y_0)$

$k_2 = hf(x_0 + h, y_0 + k_1)$

Third order RK method: $y_1 = y_0 + \frac{1}{6}(k_1 + 4k_2 + k_3)$

Where $k_1 = hf(x_0, y_0)$

$k_2 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right)$

$k_3 = hf(x_0 + h, y_0 + 2k_2 - k_1)$

Fourth order RK method: $y_1 = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

Where $k_1 = hf(x_0, y_0)$

$k_2 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}\right)$

$k_3 = hf\left(x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}\right)$

$k_4 = hf(x_0 + h, y_0 + k_3)$

1. Write a MATLAB code to find the value of y at particular value of x for solving ordinary differential equation using Euler's method and test case find y at x = 1.2 taking step size h = 0.3 for the given IVP $y' = x^2 - y^2$, $y(0) = 1$.

MATLAB Code:

```
%A script file for Eulers method
clear all; clc
f=input('enter the function @(x,y)@(x,y)(x^3+x*y^2)*exp(-x)\nf(x,y)=');
x=input('\nenter the intial value of x\n x0=');
y=input('\nenter the intial value of y\n y0=');
h=input('\nenter the step length h\n h=');
xn=input('\nenter unknown value of x\n xn=');
n=(xn-x)/h;
for i=1:n
    y=y+h*f(x,y);
    fprintf('The value of y(%0.1f)= %0.4f\n',x+h,y);
    x=x+h;
end
----------------------
```

```
Output:
enter the function @(x,y)@(x,y)(x^3+x*y^2)*exp(-x)
f(x,y)=@(x,y)x^2-y^2
enter the intial value of x
 x0=0
enter the intial value of y
 y0=1
enter the step length h
 h=0.3
enter unknown value of x
 xn=1.2
The value of y(0.3)= 0.7000
The value of y(0.6)= 0.5800
The value of y(0.9)= 0.5871
The value of y(1.2)= 0.7267
```

2. Write a MATLAB code to find the value of y at particular value of x for solving ordinary differential equation using Modified Euler's method and test case find the value of y(0.2), y(0.4) for the given IVP $y' = y + e^x$, $y(0) = 0$,

MATLAB Code:

Program:

```
%A script file for Modified Eulers method
clear all; clc
f=input('enter the function @(x,y)@(x,y)(x^3+x*y^2)*exp(-x)\nf(x,y)=');
x=input('\n enter the intial value of x \n x0=');
y=input('\nenter the intial value of y\n y0=');
h=input('\nenter the step length h\n h=');
xn=input('\n enter unknown value of x\n xn=');
n=(xn-x)/h;
for i=1:n
   fprintf('Step=%d \n',i)
      %Eulers method
   fprintf('Eulers method..........\n')
   y1=y+h*f(x,y);
   fprintf('The value using Eulers method of y(%0.1f)= %0.4f\n',x+h,y);
   x1=x+h;
   disp('Modified Eulers Method.............')
   fprintf( 'S.No.    x           \ty    \n')
   y1_old=10000;j=0;
   while(abs(y1_old-y1)>=0.0001)
      j=j+1;
      y1_old=y1;
     y1=y+(h/2)*(f(x,y)+f(x1,y1));
    fprintf(' %d \t\t%0.1f \t\ty(%0.1f)= %0.4f\n',j,x1,x1,y1);
   end
   x=x1;y=y1;
   disp('---------------------------------------')
end
----------------------------------------------------------------
```

output:

```
enter the function @(x,y)@(x,y)(x^3+x*y^2)*exp(-x)
f(x,y)=@(x,y)x^2-y^2
 enter the intial value of x
 x0=0
enter the intial value of y
 y0=1
enter the step length h
 h=0.3
 enter unknown value of x
 xn=1.2
Step=1
Eulers method..........
```

The value using Eulers method of y(0.3)= 1.0000
Modified Eulers Method............

S.No.	x	y
1	0.3	y(0.3)= 0.7900
2	0.3	y(0.3)= 0.7699
3	0.3	y(0.3)= 0.7746
4	0.3	y(0.3)= 0.7735
5	0.3	y(0.3)= 0.7738
6	0.3	y(0.3)= 0.7737

Step=2
Eulers method...........
The value using Eulers method of y(0.6)= 0.7737
Modified Eulers Method............

S.No.	x	y
1	0.6	y(0.6)= 0.6935
2	0.6	y(0.6)= 0.6793
3	0.6	y(0.6)= 0.6822
4	0.6	y(0.6)= 0.6816
5	0.6	y(0.6)= 0.6817
6	0.6	y(0.6)= 0.6817

Step=3
Eulers method...........
The value using Eulers method of y(0.9)= 0.6817
Modified Eulers Method............

S.No.	x	y
1	0.9	y(0.9)= 0.7241
2	0.9	y(0.9)= 0.7088
3	0.9	y(0.9)= 0.7121
4	0.9	y(0.9)= 0.7114
5	0.9	y(0.9)= 0.7116
6	0.9	y(0.9)= 0.7115

Step=4
Eulers method...........
The value using Eulers method of y(1.2)= 0.7115
Modified Eulers Method............

S.No.	x	y
1	1.2	y(1.2)= 0.8765
2	1.2	y(1.2)= 0.8579
3	1.2	y(1.2)= 0.8627
4	1.2	y(1.2)= 0.8615
5	1.2	y(1.2)= 0.8618
6	1.2	y(1.2)= 0.8617

----

3. Write a MATLAB code to find the value of y at particular value of x for solving ordinary differential equation using Runge – Kutta method fourth order and test obtain the values of y at x = 0.1, 0.2 for the differential equation y' =x+y, y(0) = 1 when h=0.1.

MATLAB Code:

Program:

```
% RK-4th order Method for solving first order differential equation
clear all; clc;
%dydx=@(x,y)(x+y);
dydx=input('enter the inline function eg @(x,y)(x+y):---->');
x0=input('Enter x0=');
y0=input('Enter y0=');
h=input('Enter h=');
x_new=input('Enter the unknown x=');
n=(x_new-x0)/h;
x1=x0;
y1=y0;
fprintf('x        y\n ')
for i=1:n
k1=h*dydx(x1,y1);
k2=h*dydx(x1+h/2,y1+k1/2);
k3=h*dydx(x1+h/2,y1+k2/2);
k4=h*dydx(x1+h,y1+k2);
k=(k1+2*k2+2*k3+k4)/6;
y1=y1+k;
x1=x1+h;
fprintf('%0.1f    %0.4f\n',x1,y1)
end
```

Output:

```
Enter x0=0
Enter y0=1
Enter h=0.1
Enter the unknown x=0.4
x        y
 0.1    1.1103
0.2    1.2428
0.3    1.3997
0.4    1.5836
```

Outcome:<***Write in your own words on learning experience, "what you learn/doing after completion of this exercise".***>

Exercise -10

Topic: MATLAB Solvers for differential equations.

The MATLAB command dsolve computes symbolic solutions to ordinary differential equations.

Syntax

dsolve('eq1','eq2',...,'cond1','cond2',...,'v')

Description

dsolve('eq1','eq2',...,'cond1','cond2',...,'v') symbolically solves the ordinary differential equations eq1, eq2,... using v as the independent variable. Here cond1,cond2,... specify boundary or initial conditions or both. You also can use the following syntax:
dsolve('eq1, eq2',...,'cond1,cond2',...,'v').
The default independent variable is t.

1. **Write the MATLAB Code for solving** $\frac{dy}{dx}+(y-1)\cos x = e^{-\sin x}.\cos^2 x$

   MATLAB Code:

```
syms y(t)

ode = diff(y,t)+(y-1)*cos(t) == exp(-sin(t))*(cos(t))^2

ySol(t) = dsolve(ode)
```

   Output:

```
ode(t) =

 diff(y(t), t) + cos(t)*(y(t) - 1) == exp(-sin(t))*cos(t)^2

 ySol(t) =

 exp(-sin(t))*(t/2 + sin(2*t)/4 + exp(sin(t))) + C1*exp(-sin(t))
```

2. **Write the MATLAB Code for solving** $1+(x\tan y-\sec y)\frac{dy}{dx}=0$**.**

   Rewriting the given differential equation, then we get

   $$\frac{dx}{dy}+(x\tan y-\sec y)=0$$

   MATLAB Code:

```
syms y(t)

ode2 = diff(y,t) +(y*tan(t)-sec(t)) == 0

dsolve(ode2)
```

   Output:

```
 ans =
 cos(t)*tan(t) + C1*cos(t)
```

3. **Write the MATLAB Code for solving Solve $\frac{d^4y}{dx^4} + 8\,\frac{d^2y}{dx^2} + 16y = 0.$**

MATLAB Code:

```
syms y(t)
ode3 = diff(y,t,4) + 8*diff(y,t,2)+16*y == 0
dsolve(ode3
```

Output:

ans =

C1*cos(2*t) - C3*sin(2*t) + C2*t*cos(2*t) - C4*t*sin(2*t)

4. **Write the MATLAB Code for solving Solve $(D+2)(D-1)^2y = 2\sin hx$.**
$(D^3 - 3D + 2)y = 2\sin hx$.

MATLAB Code:

```
syms y(x)
ode3 = diff(y,x,3) - 3*diff(y,x)+2*y == 2*sinh(x)
dsolve(ode3)
```

Output:

ans =

exp(x)/27 - exp(-x)/4 + (x^2*exp(x))/6 + C1*exp(x) - (x*exp(x))/9 + C3*exp(-2*x) + C2*x*exp(x)

5. **Write the MATLAB Code for solving Solve $y''' - 6\,y'' + 11y' - 6y = 0$, where $y(0) = 0, y'(0) = 0,\ y''(0) = 2.$**

MATLAB Code:

```
syms y(x)
ode5 = diff(y,x,3) - 6*diff(y,x,2)+11*diff(y,x)-6*y == 0;
Dy=diff(y,x);D2y=diff(y,x,2);
cond=[y(0)==0,Dy(0)==0,D2y(0)==2];
dsolve(ode5,cond)
```

Output:

ans =

exp(3*x) - 2*exp(2*x) + exp(x)

Outcome:*<Write in your own words on learning experience, "what you learn/doing after completion of this exercise".>*

Exercise -11

Topic: MATLAB Code for solving engineering problems

MATLAB CODES

1. Write the MATLAB Code for the following problem: The initial value problem governing the current i flowing in series R, L circuit when voltage v(t) = t is applied is given by $Ri + L\frac{di}{dt} = t\ , \quad t \geq 0$. Find the current i(t)at time t.

   MATLAB Code:

```
syms i(t) R L
lrc= R*i+L*diff(i,t)==t;
sol=dsolve(lrc,i(0)=0)
```

   Output:

```
sol =

t/R - (L - L*exp(-(R*t)/L))/R^2
```

2. Write the MATLAB Code for the following problem: An emf E = 200 e^{-5t} is applied to a series circuit consisting of 20 ohms resistor and 0.01 farad capacitor . Find the charge and current at any time if there is no initial charge on capacitor.

   $R\frac{dq}{dt} + \frac{q}{C} = E$,q(0)= 0

   MATLAB Code:

```
clear all; clc

syms q(t) R L

C=0.01; R=20;

rc= R*diff(q,t)+q/C==200*exp(-5*t);

sol=dsolve(rc,q(0)==0)
```

   Output:

```
sol =

 10*t*exp(-5*t)
```

3. Write the MATLAB Code for the following problem: If the air is maintained at 15^0C and the temperature of the body drops from 70^0C to 40^0C in 10 minutes. What will be its temperature after 30 minutes

MATLAB Code:

```
syms T(t) k
nlc1=diff(T,t)==k*(T-15);
cond=T(0)==70;
sol(t)=dsolve(nlc1,cond)
k=solve(sol(10)==40)
nlc1=diff(T,t)==k1*(T-15);
cond=T(0)==70;
sol(t)=dsolve(nlc1,cond)
```

Output:

```
sol(t) =
                    55*exp((t*log(5/11))/10) + 15
```

Outcome:

<Write in your own words on learning experience,"what you learn/doing after completion of this exercise".>

Exercise -12

Topic:Two-Dimensional Plots.

PLOTTING

Steps to plot the graph of a function

1. Define **x**, by specifying the **range of values** for the variable **x**, for which the function is to be plotted
2. Define the function, **y = f(x)**
3. Call the **plot** command, as **plot(x, y)**

Adding Title, Labels, Grid Lines, and Scaling on the Graph:

- MATLAB allows you to add title, labels along the x-axis and y-axis, grid lines and also to adjust the axes to spruce up the graph.
- The **xlabel** and **ylabel** commands generate labels along x-axis and y-axis. The **title** command allows you to put a title on the graph.
- The **grid on** command allows you to put the grid lines on the graph.
- The **axis equal** command allows generating the plot with the same scale factors and the spaces on both axes.
- The **axis square** command generates a square plot.

Setting Colors on Graph:

MATLAB provides eight basic color options for drawing graphs. The following table shows the colors and their codes:

Code	Color
w	White
k	Black
b	Blue
r	Red
c	Cyan
g	Green
m	Magenta
y	Yellow

Setting Axis Scales:

The **axis** command allows you to set the axis scales. You can provide minimum and maximum values for x and y axes using the axis command in the following way:

```
axis ( [xmin xmax ymin ymax] )
```

Generating Sub-Plots:

When you create an array of plots in the same figure, each of these plots is called a subplot. The **subplot** command is used for creating subplots.

Syntax for the command is:

```
subplot(m, n, p)
```

where, *m* and *n* are the number of rows and columns of the plot array and *p* specifies where to put a particularplot.

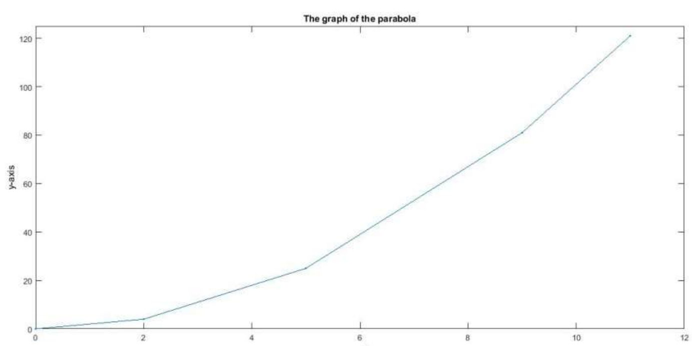
1. Write a MATLAB code to create Two-Dimensional Plots and customize the plot.

MATLAB Code:

```
%A script file to plot the graph of the function y=x^2
clc; clear;
x=input('Enter the values of x in vector form');
a=input('Minimum value on x-axis, a=');
b=input('Maximum value on x-axis b=');
c=input('Minimum value on y-axis c=');
d=input('Maximum value on y-axis d=');
y=x.^2;
plot(x,y,'.-')
title('The graph of the parabola'), xlabel('x-axis'), ylabel('y-
axis'),axis([a b c d])
```

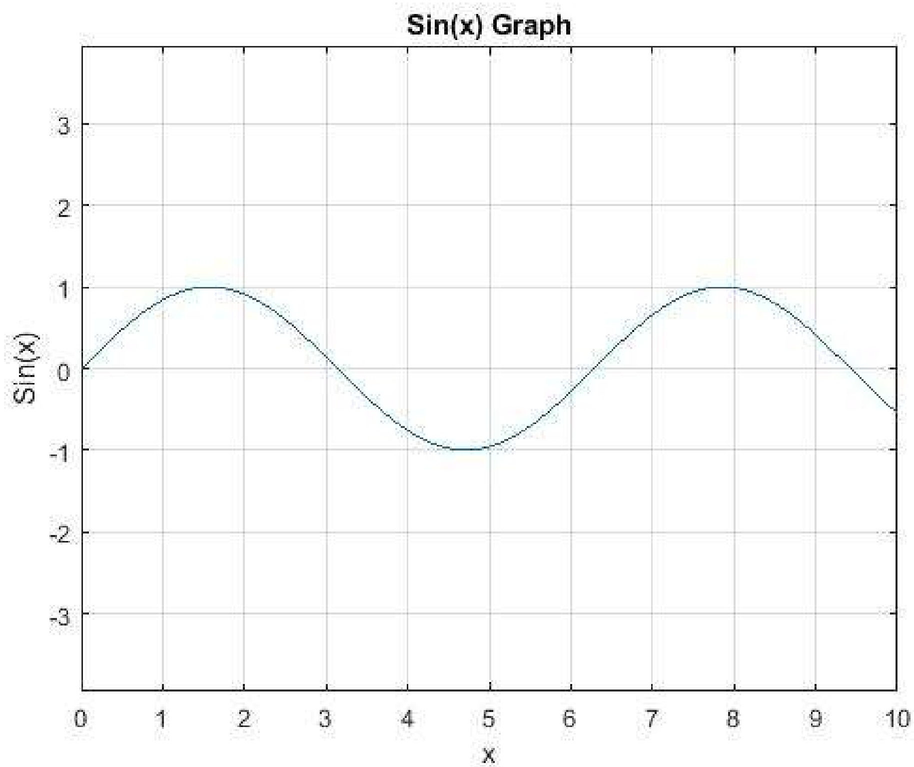
Output:

Enter the values of x in vector form [0, 2, 5, 9, 11]
Minimum value on x-axis, a=0
Maximum value on x-axis b=12
Minimum value on y-axis c=0
Maximum value on y-axis d=125



2. Write a MATLAB code to create Two-Dimensional Plots and customize the plot.
   MATLAB Code:

```
x = [0:0.01:10];
y = sin(x);
plot(x, y), xlabel('x'), ylabel('Sin(x)'), title('Sin(x)
Graph'),
grid on, axis equal
```

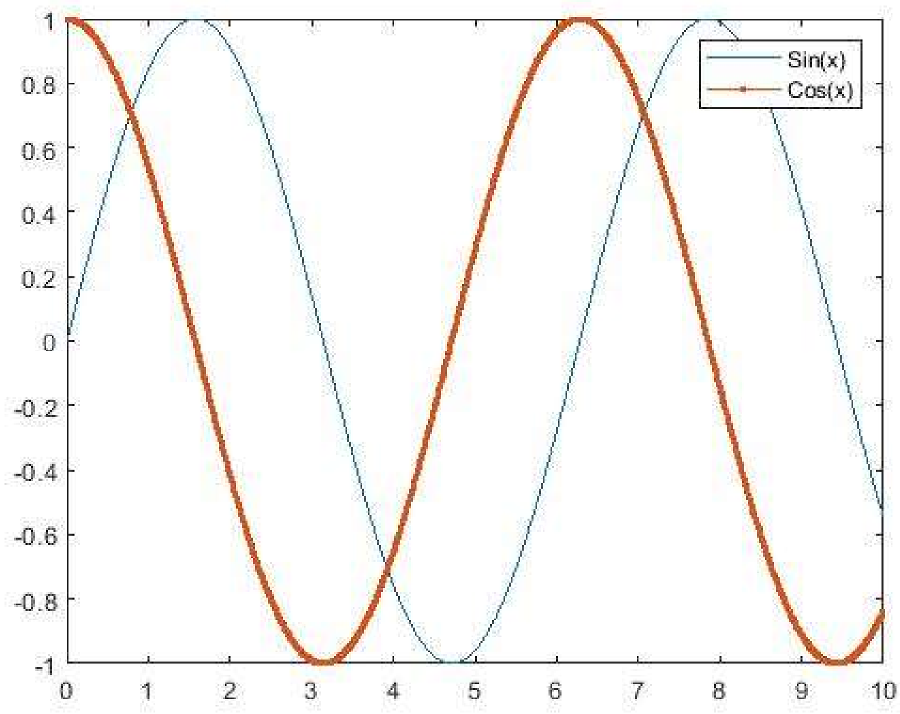


3. Write a MATLAB code to create Multiple Functions on the Same Graph and customize the plot.

MATLAB CODE:

```
x = [0 : 0.01: 10];
y =sin(x);
g =cos(x);
plot(x, y, x, g, '.-'), legend('Sin(x)', 'Cos(x)')
```

Output:



Outcome:

<Write in your own words on learning experience, "what you learn/doing after completion of this exercise".>