EXERCISE – 1

Introduction & Working with the MATLAB user interface

- **Program 1**

Aim: A simple program that sends a simple text message to the screen

Apparatus: MATLAB software

Code:

```
clc
fprintf (1, 'Hello World! \n');
```

Result:

```
Hello World!
```

- **Program 2**

Aim: MATLAB program to swap two variables using a temporary variable

Apparatus: MATLAB software

Code:

```
clc
fprintf (1,'MATLAB program to swap two variables using a temporary variable \n');
a = input ('enter a ');
b = input ('enter b ');
fprintf (1, 'Before swapping a= %f b = %f \n', a, b);
temp = a;
a=b;
b=temp;
fprintf (1,'After swapping a= %f b = %f \n', a, b);
```

Result:

```
enter a 28

enter b 29

Before swapping a= 28.000000 b = 29.000000

After swapping a= 29.000000 b = 28.000000
```

- **Program 3**

Aim: MATLAB program to swap two variables without using a temporary variable

Apparatus: MATLAB software

Code:

```
clc

fprintf (1, 'MATLAB program to swap two variables without using a temporary variable \n');

a = input ('enter a ');

b = input ('enter b ');

fprintf (1, 'Before swapping a= %f b = %f \n', a, b);

a = a + b;

b = a - b;

a = a - b;

fprintf (1, 'After swapping a= %f b = %f \n', a, b);
```

Result:

```
enter a 44

enter b 58

Before swapping a= 44.000000 b = 58.000000

After swapping a= 58.000000 b = 44.000000
```

- **Program 4**

Aim: MATLAB program to compute the area of a circle

Apparatus: MATLAB software

Code:

```
clc
fprintf (1,'Matlab program to compute the area of a circle\n');
radius = input ('enter radius ');
area = pi*radius^2;
fprintf (1,'Area of circle = %f \n', area);
```

Result:

```
enter radius 44
Area of circle = 6082.123377
```

- **Program 5**

Aim: MATLAB program to convert temperature from Celsius to Fahrenheit

Apparatus: MATLAB software

Code:

```
clc
fprintf (1, 'Matlab program to convert temperature from Celsius to Fahrenheit\n');
tempC = input ('enter temperature in degrees Celsius ');
tempF = 1.8*tempC+32;
fprintf (1, 'Temperature in degrees Fahrenheit = %f \n', tempF);
```

Result:

```
enter temperature in degrees Celsius 45
Temperature in degrees Fahrenheit = 113.000000
```

- **Program 6**

Aim: MATLAB program to convert temperature from Fahrenheit to Celsius

Apparatus: MATLAB software

Code:

```
clc

fprintf (1,'MATLAB program to convert temperature from Fahrenheit to Celsius \n');

tempF = input ('enter temperature in degrees Fahrenheit ');

tempC = (tempF-32) /1.8;

fprintf (1,'Temperature in degrees Celsius = %f \n', tempC);
```

Result:

```
enter temperature in degrees Fahrenheit 113

Temperature in degrees Celsius = 45.000000
```

EXERCISE – 2

Exercise on basic Mathematics (Mathematical and logical operations & solving arithmetic equations, Matrix Operations & Trigonometric functions)

- **Program 7**

Aim: MATLAB program to compute the sum and average of three numbers

Apparatus: MATLAB software

Code:

```
clc

fprintf(1, 'MATLAB program to compute maximum and minimum of three numbers \n') ;

a = input ('enter a ');

b = input ('enter b ');

c = input ('enter c ');

sum = a+b+c;
```

```
avg = sum/3;
fprintf (1,'sum is %f\n', sum);
fprintf(1, 'average is %f\n', avg) ;
```

Result:

```
enter a 22
enter b 23
enter c 28
sum is 73.000000
average is 24.333333
```

- **Program 8**

Aim: MATLAB program to compute simple interest

Apparatus: MATLAB software

Code:

```
clc
fprintf (1,'MATLAB program to compute simple interest\n');
P = input ('enter Principal amount: ');
N = input ('enter number of years: ');
R = input ('enter rate of interest: ');
simple = P*N*R/100;
fprintf (1,'Simple interest is %f\n', simple);
```

Result:

```
enter Principal amount: 5000
enter number of years: 2
enter rate of interest: 5
Simple interest is 500.000000
```

- **Program 9**

Aim: MATLAB program to identify if the given number is odd or even

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1, 'MATLAB program to identify if the given number is odd or even\n');
n = input ('enter a number: ');
if (rem (n, 2) == 0)
   fprintf (1, 'the number is even\n');
else
   fprintf (1, 'the number is odd\n');
end
```

Result:

```
enter a number: 56
the number is even
```

- **Program 10**

Aim: MATLAB program to identify if the given number is positive,negative or zero

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1, 'MATLAB program to identify if the given number is positive,
negative or zero\n');
n = input ('enter a number: ');
if (n>0)
```

```
    fprintf (1, 'the number is positive\n');
elseif (n<0)
    fprintf (1,'the number is negative\n');
else
    fprintf (1, 'the number is zero\n');
end
```

Result:

```
enter a number: -98
the number is negative
```

- **Program 11**

Aim: MATLAB program to compute maximum and minimum of three numbers

Apparatus: MATLAB software

Code:

```
clc
fprintf (1, 'Program to compute maximum and minimum of three numbers \n' ) ;
a = input ('enter a ');
b = input ('enter b ');
c = input ('enter c ');
fprintf(1,'maximum is %f\n', max (max(a,b), c) );
fprintf(1,'minimum is %f\n', min(min(a,b),c) ) ;
```

Result:

```
enter a 55
enter b 66
enter c 77
maximum is 77.000000
```

```
minimum is 55.000000
```

- **Program 12**

Aim: MATLAB program to identify if the given year is a leap year or not

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1,'MATLAB program to identify if the given year is a leap year or not\n');
n = input ('enter a year: ');
if(rem(n,100) == 0)
   if(rem(n, 400) == 0)
      fprintf (1,'%d is a leap year\n', n);
   else
      fprintf (1,'%d is NOT a leap year\n', n);
   end
else
   if(rem(n, 4) == 0)
      fprintf (1, '%d is a leap year\n', n);
   else
      fprintf (1, '%d is NOT a leap year\n', n);
   end
end
```

Result:

```
enter a year: 2098
2098 is NOT a leap year
```

- **Program 13**

Aim: MATLAB program to compute the average marks of 3 subjects and print the grade

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1, 'MATLAB program to compute the average marks of 3 subjects and print the grade\n');
m1 = input ('enter marks of subject1 : ');
m2 = input ('enter marks of subject2 : ');
m3 = input ('enter marks of subject3 : ');
average = (m1+m2+m3) /3;
fprintf (1,'average is %f\n' , average) ;
if (average>=90 && average <= 100)
   fprintf(1,'Grade is A+\n') ;
elseif (average>=80 && average <= 90)
   fprintf (1, 'Grade is A\n');
elseif (average>=70 && average <= 80)
   fprintf (1, 'Grade is B\n');
elseif (average>=60 && average <= 70)
   fprintf (1, 'Grade is C\n');
else
   fprintf (1, 'Grade is FAIL\n') ;
end
```

Result:

```
enter marks of subject1 : 80
enter marks of subject2 : 95
enter marks of subject3 : 90
```

```
average is 88.333333
Grade is A
```

- **Program 14**

Aim: MATLAB program to print the grade using switch

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1, 'MATLAB program to print the grade using switch\n');
m1 = input ('enter marks of subject1: ');
m2 = input ('enter marks of subject2: ');
m3 = input ('enter marks of subject3: ');
average = (m1+m2+m3) /3;
fprintf(1,'average = %f\n', average);
switch(floor(average/10))
   case {10,9}
      fprintf (1,'Grade = A+\n');
      break;
   case 8
      fprintf (1, 'Grade = A\n');
      break;
   case 7
      fprintf (1, 'Grade = B\n');
      break;
   case 6
      fprintf (1, 'Grade = C\n');
```

```
        break;
    otherwise
        fprintf (1,'Grade = FAIL\n');
end
```

Result:

```
 enter marks of subject1: 67
enter marks of subject2: 89
enter marks of subject3: 44
average = 66.66666
Grade = C
```

- **Program 15**

Aim: MATLAB program to display if an entered letter is vowel or not using switch

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1, 'MATLAB program to display if an entered letter is vowel or not using switch\n');
ch = input ('enter an alphabet: ','s');
switch (ch)
    case {'a', 'A' , 'e' , 'E' , 'i' , 'I' , 'o', 'O', 'u', 'U'}
        fprintf (1,'The letter is a vowel \n');
        break;
    otherwise
        fprintf (1,'The entered letter is a consonant\n');
end
```

Result:

```
enter an alphabet: X
The entered letter is a consonant
```

- **Program 16**

Aim: MATLAB program to generate Fibonacci series

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1, 'MATLAB program to generate Fibonacci series\n');
f1 =1;
f2=1;
n = input ('Enter the number of terms in the series: ');
if (n>0)
   if(n == 1)
      fprintf (1, '%3d', f1);
   elseif (n == 2)
      fprintf(1,'%3d %3d ', f1, f2);
   else
      fprintf (1, '%3d %3d ', f1, f2);
      for i=3:1:n
         f=f1+f2;
         fprintf (1, '%3d ',f);
         f1=f2;
         f2=f;
      end
```

```
        fprintf (1, '\n' ) ;
    end
end
```

RESULT:

```
 Enter the number of terms in the series: 9
  1   1   2   3   5   8  13  21  34
```

- **Program 17**

Aim: MATLAB program to generate factorial of a given integer

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1,'MATLAB program to compute factorial of a given integer\n');
n = input ('Enter n ');
if(n == 0 ||n == 1)
    fprintf (1,'Factorial of %d is 1\n', n);
else
    fact = 1;
    for i=2:n
        fact = fact*i;
    end
    fprintf (1, 'Factorial of %d is %d\n' , n, fact) ;
end
```

Result:

```
Enter n 9
```

```
Factorial of 9 is 362880
```

- **Program 18**

Aim: MATLAB program to check if a given number is palindrome or not

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1, 'MATLAB program to check if a given number is palindrome or not
\n');
n = input ('Enter n ');
m=n;
rev = 0;
while (n>0)
   r= rem (n, 10);
   rev=(rev*10) +r;
   n=floor(n/10);
end
fprintf (1, 'Reverse of the number %d is %d\n' , m, rev) ;
if (m == rev)
   fprintf (1,'%d is a palindrome\n',m) ;
else
   fprintf (1,'%d is NOT a palindrome\n', m);
end
```

Result:

```
Enter n 9898
Reverse of the number 9898 is 8989
9898 is NOT a palindrome
```

- **Program 19**

Aim: MATLAB program to print the SINE as the sum of series

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1, 'MATLAB program to print the SINE as the sum of series\n');
n = input ('Enter number of terms in the SINE series ') ;
x = input ('Enter the angle in degrees\n') ;
x = x* (pi/180);
sum=x;
term=x;
for i=1:n
   term = term*(-x^2)/((2*i)*(2*i+1));
   sum = sum+term;
end
fprintf (1,'SINE value of given angle is %f\n', sum) ;
```

Result:

```
Enter number of terms in the SINE series 5
Enter the angle in degrees
45
SINE value of given angle is 0.707107
```

- **Program 20**

Aim: MATLAB program to print the COSINE as the sum of series

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1, 'MATLAB program to print the COSINE as the sum of series \n');
n = input ('Enter number of terms in the COSINE series ') ;
x = input ('Enter the angle in degrees\n') ;
x = x* (pi/180);
sum=1;
term=1;
for i=1:n
   term = term*(-x^2)/((2*i)*(2*i-1));
   sum = sum+term;
end
fprintf (1,'COSINE value of given angle is %f\n', sum) ;
```

Result:

```
Enter number of terms in the COSINE series 5
Enter the angle in degrees
45
COSINE value of given angle is 0.707107
```

- **Program 21**

Aim: MATLAB program to compute e^x

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1,'MATLAB program to compute e^x\n' );
n = input ('Enter n: ');
x = input ('Enter x: ');
```

```
term = 1;
sum = 1;
for i=1:n
   term = term* (x/i);
   sum = sum+term;
end
fprintf(1,'exp(%f) is %f\n' ,x, sum) ;
fprintf(1,'using Matlab function exp(%f) is %f\n' , x, exp(x));
```

Result:

```
Enter n: 4
Enter x: 5
exp(5.000000) is 65.375000
using Matlab function exp(5.000000) is 148.413159
```

EXERCISE – 3

Exercise on Loops, Conditional Statements & Functions

- **Program 22**

Aim: MATLAB program to all prime numbers between 1 and n

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1,'MATLAB program to all prime numbers between 1 and n\n') ;
n = input ('Enter n: ');
fprintf (1, 'list of prime numbers between 1 and %d\n', n) ;
for i=1:n
   flag = 1;
%check if i is a prime or not
```

```
for j=2: floor (i/2)
   if (rem(i, j) == 0) %if j is a factor of i, then i is not a prime
      flag =0;
   end
end
if(flag == 1)
   fprintf (1,'%4d' , i) ;
end
end
fprintf (1,'\n');
```

Result:

```
Enter n: 6
list of prime numbers between 1 and 6
   1   2   3   5
```

- **Program 23**

Aim: MATLAB program to print Pascal's triangle

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1,'MATLAB program to Pascal''s triangle\n');
n = input ('Enter number of lines: ');
for i=1:n
   for j=1:n-i
      fprintf (1,' ');
   end
```

```
    for j=0:i
        val = factorial(i)/(factorial(j)*factorial(i-j));
        fprintf (1, '%3d ', val);
    end
    fprintf (1,' \n');
end
```

Result:

```
Enter number of lines: 6
       1   1
      1   2   1
    1   3   3   1
   1   4   6   4   1
  1   5  10  10   5   1
 1   6  15  20  15   6   1
```

- **Program 24**

Aim: MATLAB program to all prime numbers between 1 and n

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1,'MATLAB program to compute all prime numbers between 1 and n\n'
) ;
n = input ('Enter n: ');
fprintf (1, 'list of prime numbers between 1 and %d\n', n) ;
for i=1:n
    flag = 1;
%check if i is a prime or not
```

```
for j=2: floor(i/2)
   if (rem(i, j) == 0) %if j is a factor of i, then i is not a prime
      flag =0;
   end
end
if (flag == 1)
   fprintf (1,'%4d' , i) ;
end
end
fprintf (1, '\n');
```

Result:

```
Enter n: 6
list of prime numbers between 1 and 6
   1   2   3   5
```

- **Program 25**

Aim: MATLAB program to find the first integer n whose factorial is a m digit number

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1, 'MATLAB program to find the first integer n whose factorial is a m digit number\n' ) ;
m = input ('How many digits ');
n = 1;
nFactorial = 1;
while (nFactorial < 10^(m-1))
```

```
   n = n + 1;
   nFactorial = nFactorial * n;
end
fprintf (1, 'factorial of %d is %1d (a %d digit number)\n', n, nFactorial, m) ;
```

Result:

```
How many digits 7
factorial of 10 is 3628800 (a 7 digit number)
```

- **Program 26**

Aim: MATLAB program to print the COSINE as the sum of series using while loop

Apparatus: MATLAB software

Code:

```
clc;
fprintf (1, 'MATLAB program to print the COSINE as the sum of series using while loop\n') ;
x = input ('Enter the angle in degrees\n');
x = x* (pi/180);
sum=1;
term=1;
i=0;
while (abs(term)>1.0e-15)
   i=i+1;
   term = term*(-x^2)/((2*i)*(2*i-1));
   sum = sum+term;
end
fprintf (1,'COSINE value of given angle is %.16f\n', sum);
```

Result:

```
Enter the angle in degrees

98

COSINE value of given angle is -0.139173100960065 4
```

- **Program 27**

Aim: MATLAB program to print the SINE as the sum of series using while Loop

Apparatus: MATLAB software

Code:

```
clc;

fprintf (1, 'MATLAB program to print the SINE as the sum of series using while loop\n') ;

x = input ('Enter the angle in degrees\n') ;

x = x* (pi/180) ;

sum=x;

term=x;

i=0;

while(abs(term) >1.0e-20)

   i=i+1;

   term = term*(-x^2)/((2*i)*(2*i+1));

   sum = sum+term;

end

fprintf (1, 'SINE value of given angle is %.15f\n', sum) ;
```

Result:

```
Enter the angle in degrees

66

SINE value of given angle is 0.913545457642601
```

EXERCISE – 4

Exercise on plotting graphs (Plot labelling, curve labelling and editing, 2D&3D Plots)

- **Program 1**

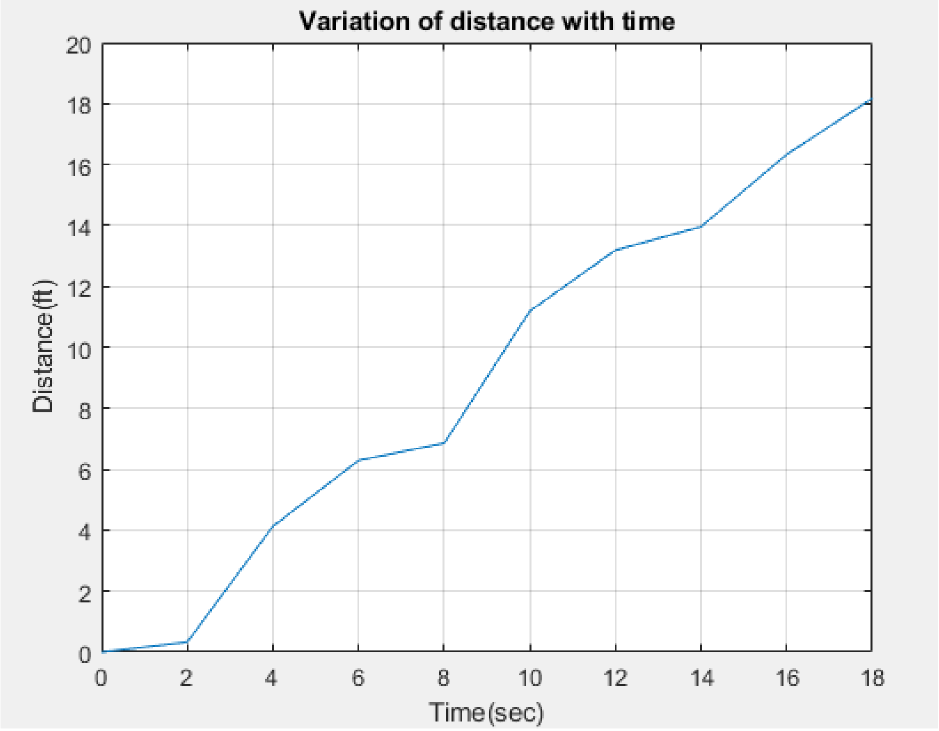
Aim: MATLAB program to create a simple two dimensional plot

Apparatus: MATLAB software

Code:

```
x = [0:2:18];
y = [0, 0.33, 4.13, 6.29, 6.85, 11.19, 13.19, 13.96, 16.33,18.17];
plot(x,y);
xlabel('Time(sec)')
ylabel('Distance(ft)')
title('Variation of distance with time')
grid
```

Result:



- **Program 2**

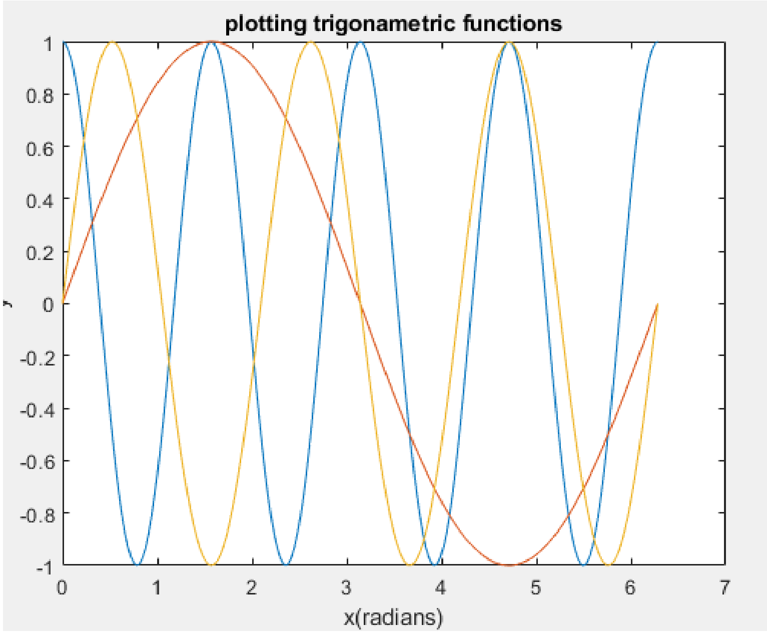
Aim: MATLAB program to create a two dimensional plot with more than one line

Apparatus: MATLAB software

Code:

```
x = 0:pi/100:2*pi;

y1 = cos(4*x);

y2 = sin(x);

y3 = sin(3*x);

% plot(x,y1)

% hold on;

% plot(x, y2)

% hold on

% plot(x,y3)

%alternately

Y = [y1;y2;y3];

plot(x,Y)

title('plotting trigonametric functions');

xlabel('x(radians)');

ylabel('y');
```

Result:



- **Program 3**

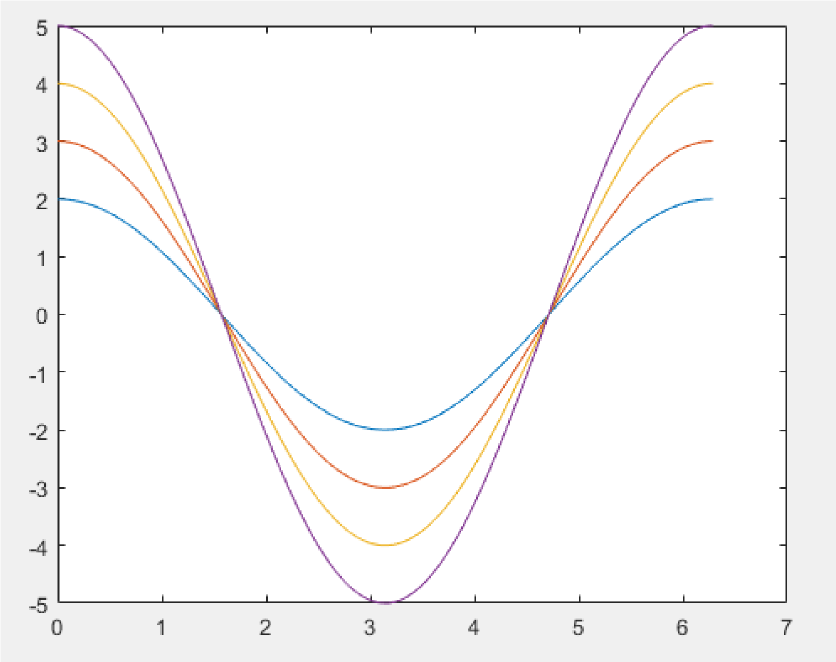
Aim: MATLAB program to create a plot with mutiple lines

Apparatus: MATLAB software

Code:

```
X = 0:pi/100:2*pi;

Y1 = cos(X)*2;

Y2 = cos(X)*3;

Y3 = cos(X)*4;

Y4 = cos(X)*5;

Y = [Y1; Y2; Y3; Y4];

plot(X, Y1, X, Y2, X, Y3, X, Y4)

% plot(X,Y)
```

Result:



- **Program 4**

Aim: MATLAB program to create a 2-D plot with line,mark and color options

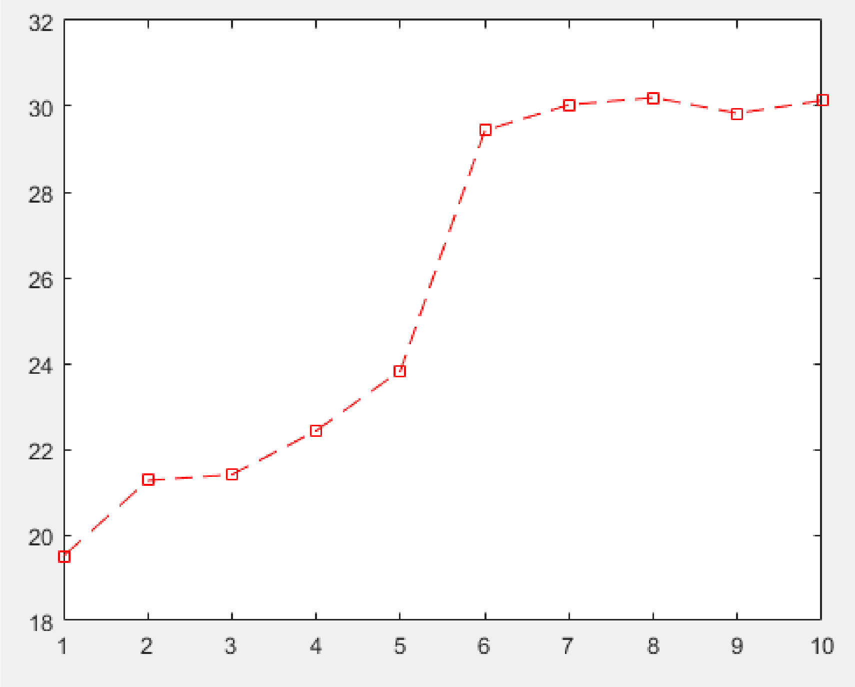
Apparatus: MATLAB software

Code:

```
x = [1:10];
y = [58.5, 63.8, 64.2, 67.3, 71.5, 88.3, 90.1, 90.6,89.5,90.4];

%LINE TYPE : - Solid   : Dotted  -. Dash-dot -- Dashed
%MARKER: . point  o circle  x cross  + plus  * star  s square d diamond
%LINE COLOR: blue    'b'   green  'g'  red 'r'  cyan 'c'
%           magenta 'm'   yellow 'y'  black 'k'  white 'w'
% plot(x,y,':ok',x,2*y,'--xr',x,y/2,'-sb',x,y/3,'-.*g')
 figure(1)
plot(x,y,'-ok')  %line : SOLID   marker: CIRCLE   color: BLACK
figure(2)
```

```
plot(x,2*y,':xb')  %line : DOTTED   marker: CROSS   color: BLUE
%
figure(3)
plot(x,y/2,'-.*g')  %line : DASH-DOT   marker: STAR   color: GREEN
%
figure(4)
plot(x,y/3,'--sr')  %line : DASHED  marker: SQUARE   color: RED
```

Result:



- **Program 5**

Aim: Exercise problem

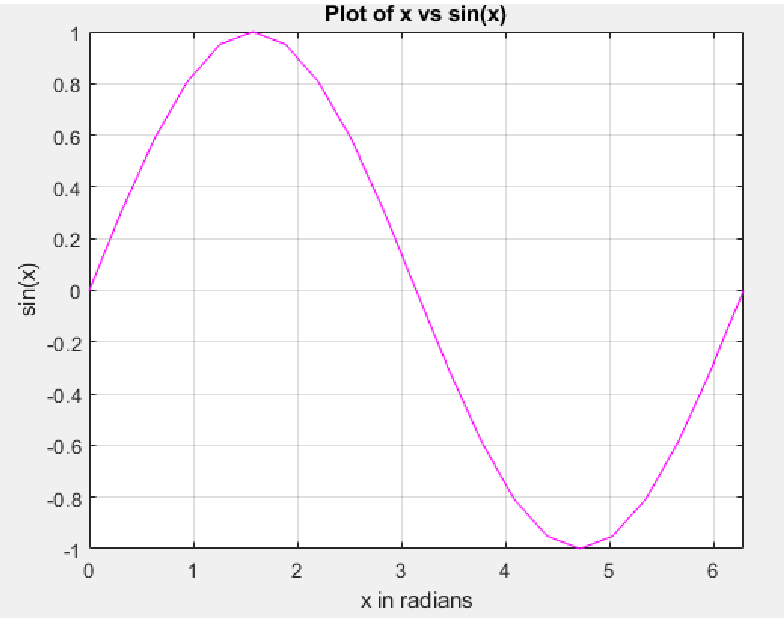
Apparatus: MATLAB software

Code:

```
%Plot x vs y for y=sin(x). Let x vary from zero to 2pi in steps of 0.1pi
%Add a title and labels to your plot
x = 0:0.1*pi:2*pi;
```

```
y = sin(x);
plot(x,y,'m')
title('Plot of x vs sin(x)');
xlabel('x in radians');
ylabel('sin(x)');
axis([0,2*pi,-1,1]);
grid
```

Result:



- **Program 6**

Aim: Exercise problem

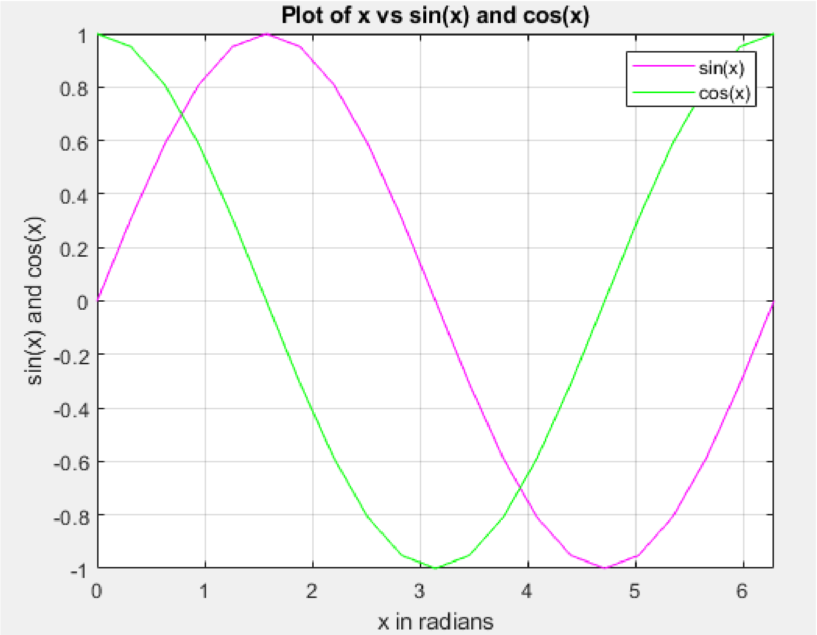
Apparatus: MATLAB software

Code:

```
% Plot x versus y1 and y2 for y1=sin(x) and y2=cos(x). Let x vary from
% 0 to 2*pi in increments of 0.1p. Add a title and labels to your plot.
%Plot x vs y for y=sin(x). Let x vary from zero to 2pi in steps of 0.1pi
%Add a title and labels to your plot
```

```
x = 0:0.1*pi:2*pi;
y1 = sin(x);
y2 = cos(x);
plot(x,y1,'m',x,y2,'g')
title('Plot of x vs sin(x) and cos(x)');
xlabel('x in radians');
ylabel('sin(x) and cos(x)');
axis([0,2*pi,-1,1]);
legend('sin(x)','cos(x)');
grid
 % make the sin (x) line dashed and red.
% Make the cos(x) line dotted and green
% plot(x,y1,':r',x,y2,'.g')
% gtext('This is a figure created by XXXXXXX');
```

Result:



- **Program 7**

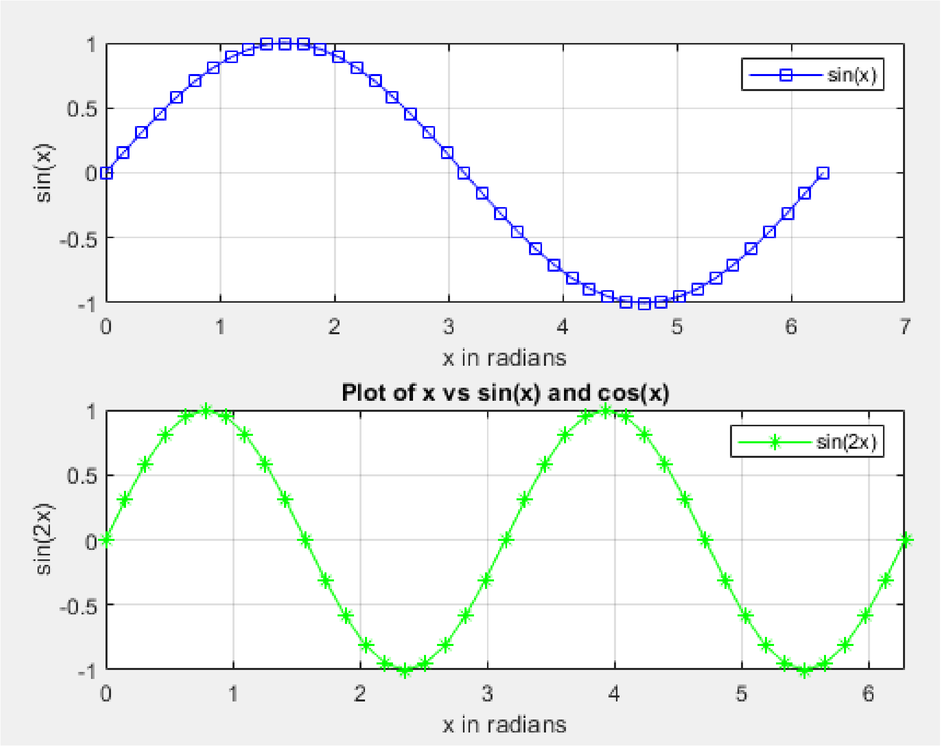
Aim: Making subplots

Apparatus: MATLAB software

Code:

```
% Plot x versus y1 and y2 for y1=sin(x) and y2=cos(x). Let x vary from
% 0 to 2*pi in increments of 0.1p. Add a title and labels to your plot.
%Plot x vs y for y=sin(x). Let x vary from zero to 2pi in steps of 0.1pi
%Add a title and labels to your plot
x = 0:pi/20:2*pi;
subplot(2,1,1)
plot(x,sin(x),'-sb')
legend('sin(x)');
xlabel('x in radians');
ylabel('sin(x)');
grid
subplot(2,1,2)
plot(x,sin(2*x),'-*g')
xlabel('x in radians');
ylabel('sin(2x)');
legend('sin(2x)');
title('Plot of x vs sin(x) and cos(x)');
axis([0,2*pi,-1,1]);
grid
```

Result:



- **Program 8A**

Aim: Making subplots

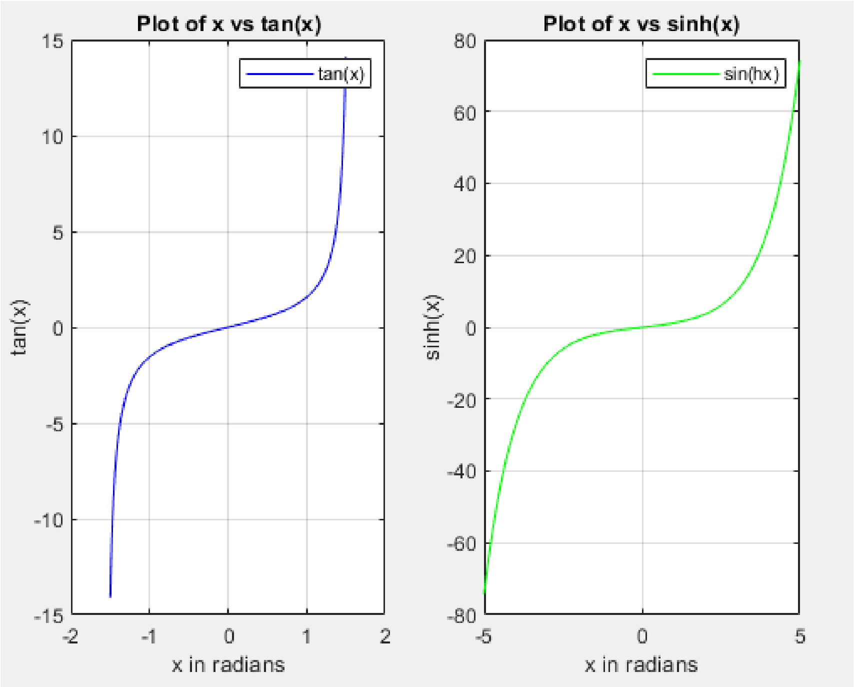
Apparatus: MATLAB software

Code:

```
% Plot x versus y1 and y2 for y1=sin(x) and y2=cos(x). Let x vary from
% 0 to 2*pi in increments of 0.1p. Add a title and labels to your plot.
%Plot x vs y for y=sin(x). Let x vary from zero to 2pi in steps of 0.1pi
%Add a title and labels to your plot
x = -1.5:0.01:1.5;
subplot(1,2,1)
plot(x,tan(x),'b')
title('Plot of x vs tan(x)');
legend('tan(x)');
xlabel('x in radians');
```

```
ylabel('tan(x)');
grid
subplot(1,2,2)
x = -5:0.01:5;
plot(x,sinh(x),'g')
xlabel('x in radians');
ylabel('sinh(x)');
legend('sin(hx)');
title('Plot of x vs sinh(x)');
grid
```

Result:



- **Program 8B**

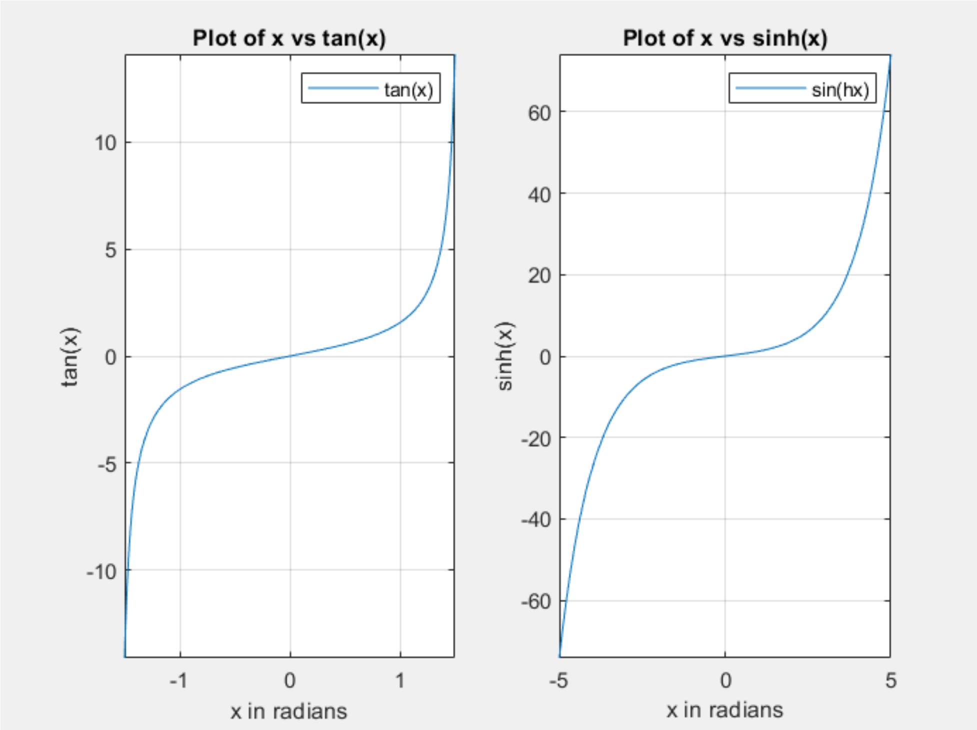
Aim: Making function plots

Apparatus: MATLAB software

Code:

```
% Plot x versus y1 and y2 for y1=sin(x) and y2=cos(x). Let x vary from
% 0 to 2*pi in increments of 0.1p. Add a title and labels to your plot.
%Plot x vs y for y=sin(x). Let x vary from zero to 2pi in steps of 0.1pi
%Add a title and labels to your plot
subplot(1,2,1)
fplot('tan(x)',[-1.5,1.5])
title('Plot of x vs tan(x)');
legend('tan(x)');
xlabel('x in radians');
ylabel('tan(x)');
grid
subplot(1,2,2)
fplot('sinh(x)',[-5,5])
xlabel('x in radians');
ylabel('sinh(x)');
legend('sin(hx)');
title('Plot of x vs sinh(x)');
grid
```

Result:



- **Program 9A**

Aim: Making polar plots

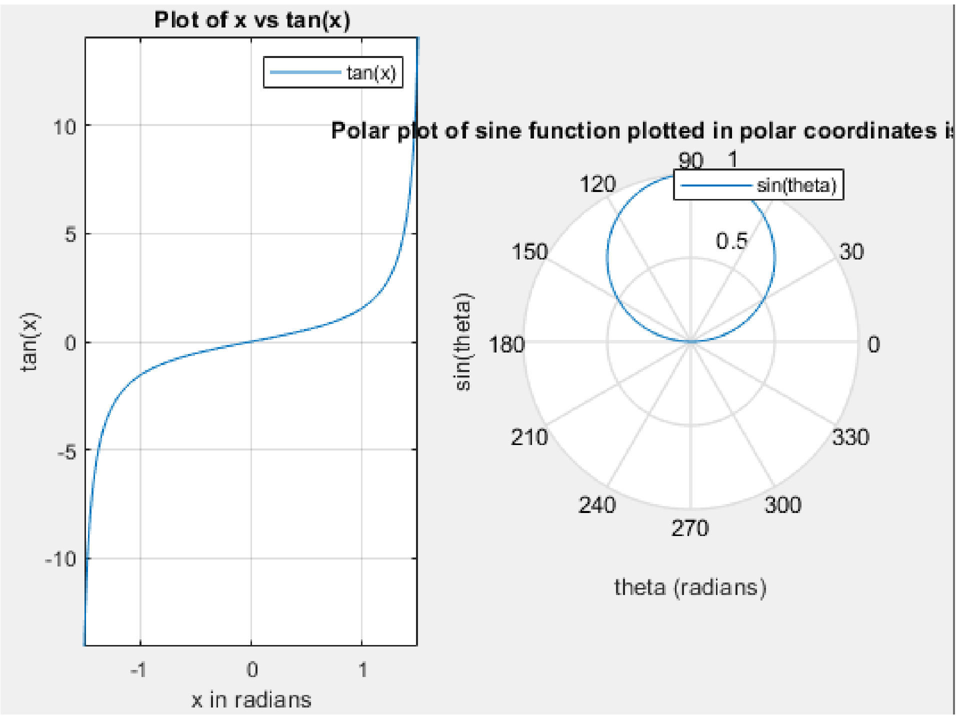
Apparatus: MATLAB software

Code:

```
%-----------------------------------------------------------------------------------
% NOTE: MATLAB provides plotting capability with polar coordinates:
% polar(theta, r) generates a polar plot of angle theta (in radians) and radial
distance r .
%-----------------------------------------------------------------------------------
%Make a polar plot of sin(x)
theta = 0:pi/100:2*pi;
y = sin(theta);
polar(theta,y);
xlabel('theta (radians)');
ylabel('sin(theta)');
```

```
legend('sin(theta)','NorthOutside');
title('Polar plot of sine function plotted in polar coordinates is a circle');
grid
```

Result:



- **Program 9B**

Aim: Making polar plots

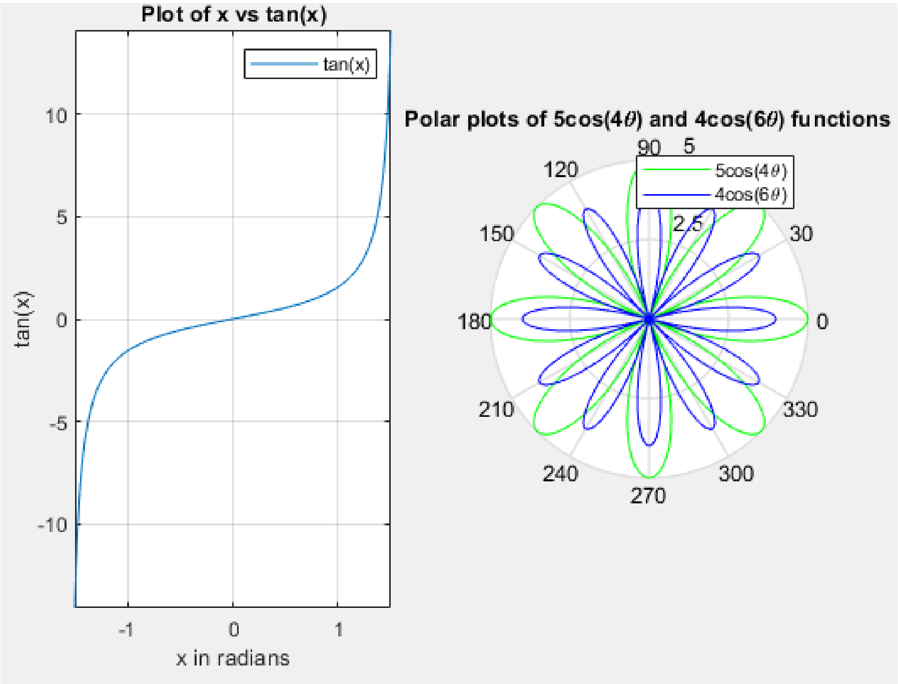
Apparatus: MATLAB software

Code:

```
%---------------------------------------------------------------------------
% NOTE: MATLAB provides plotting capability with polar coordinates:
% polar(theta, r) generates a polar plot of angle theta (in radians) and radial distance r .
%---------------------------------------------------------------------------
%Make a polar plot of sin(x)
theta = 0:pi/100:2*pi;
```

```
y = 5*cos(4*theta);
polar(theta,y,'g');
title('Polar plots of 5cos(4\theta) and 4cos(6\theta) functions');
grid
hold;
y = 4*cos(6*theta);
polar(theta,y,'b');
legend('5cos(4\theta)','4cos(6\theta)');
hold off
```

Result:



- **Program 9C**

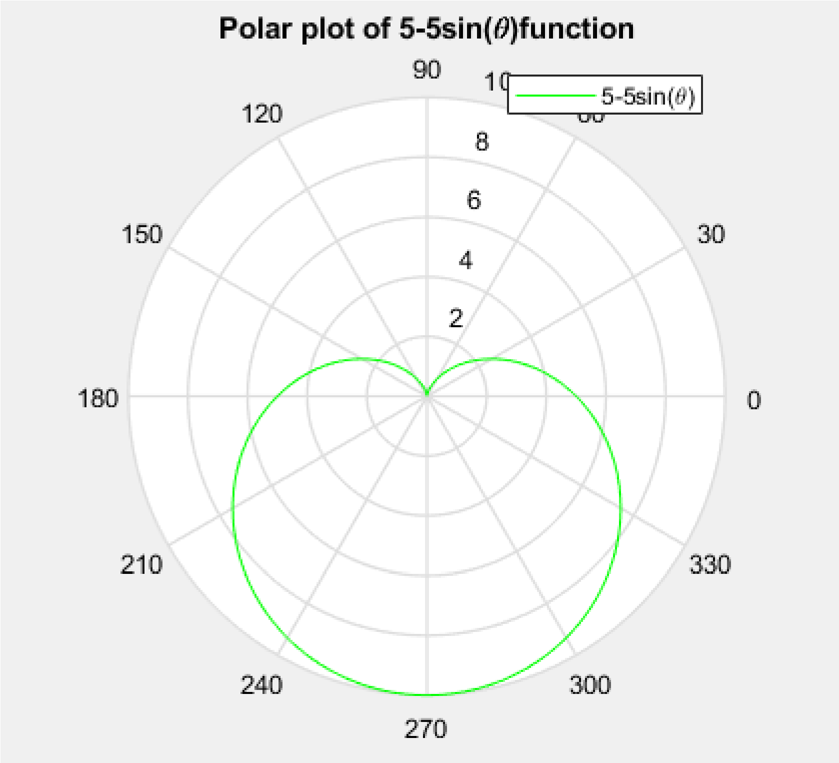
Aim: Making polar plots

Apparatus: MATLAB software

Code:

```
%-------------------------------------------------------------------------------------
% NOTE: MATLAB provides plotting capability with polar coordinates:
% polar(theta, r) generates a polar plot of angle theta (in radians) and radial distance r .
%-------------------------------------------------------------------------------------
%Make a polar plot of 5-5sin(theta)
theta = 0:pi/100:2*pi;
y = 5-5*sin(theta);
polar(theta,y,'g');
title('Polar plot of 5-5sin(\theta)function');
grid
legend('5-5sin(\theta)');
```

Result:



- **Program 9D**

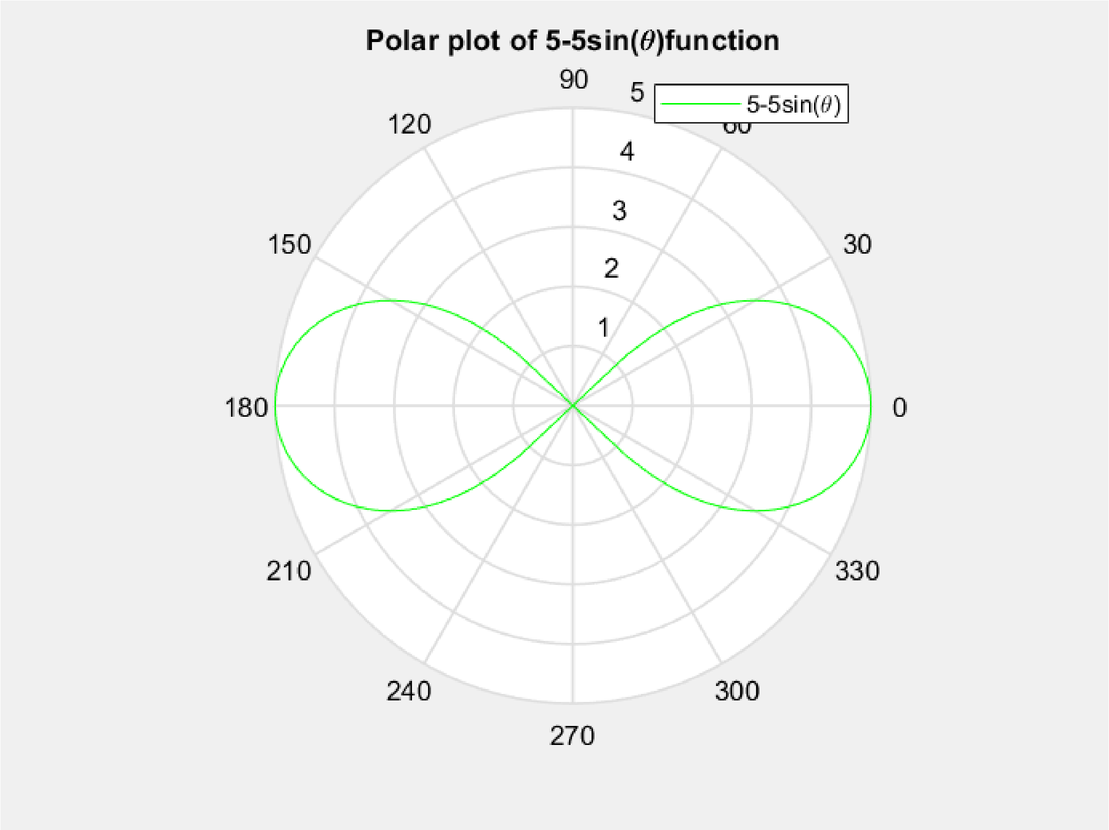
Aim: Making polar plots

Apparatus: MATLAB software

Code:

```
%------------------------------------------------------------------------------------
% NOTE: MATLAB provides plotting capability with polar coordinates:
% polar(theta, r) generates a polar plot of angle theta (in radians) and radial distance r .
%------------------------------------------------------------------------------------
%Make a polar plot of 5-5sin(theta)
theta = 0:pi/100:2*pi;
y=sqrt(5^2*cos(2*theta));
polar(theta,y,'g');
title('Polar plot of 5-5sin(\theta)function');
grid
legend('5-5sin(\theta)');
```

Result:



- **Program 10**

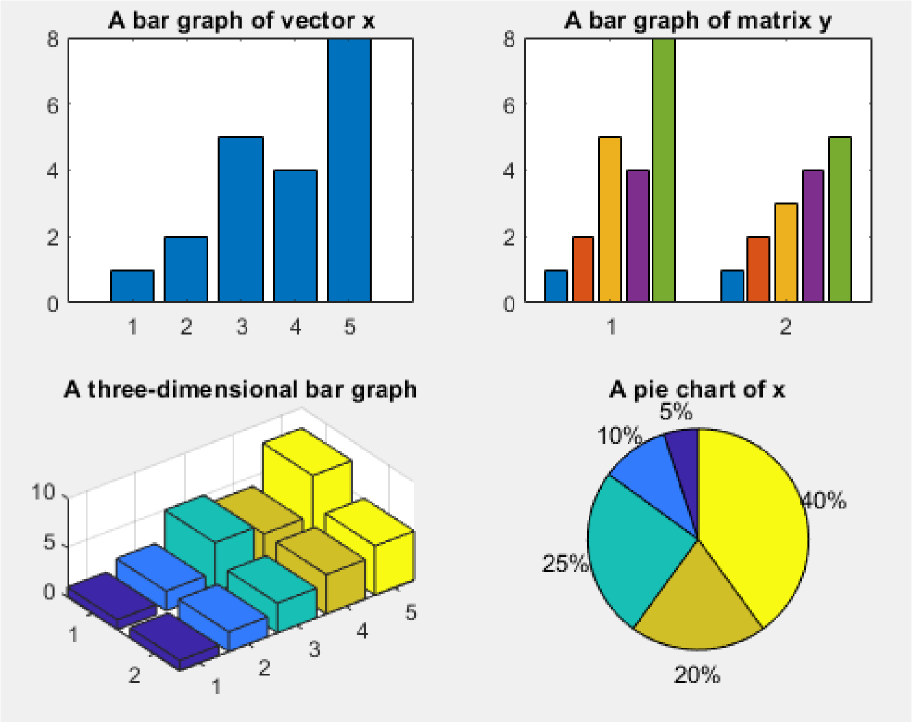
Aim: Different Graphs

Apparatus: MATLAB software

Code:

```
clear, clc
x = linspace(0,10*pi,1000);
y = cos(x);
z = sin(x);
% plot3(x,y,z)
comet3(x,y,z)
grid
xlabel('angle'), ylabel('cos(x)'), zlabel('sin(x)'), title('A Spring')
```

Result:



- **Program 11**

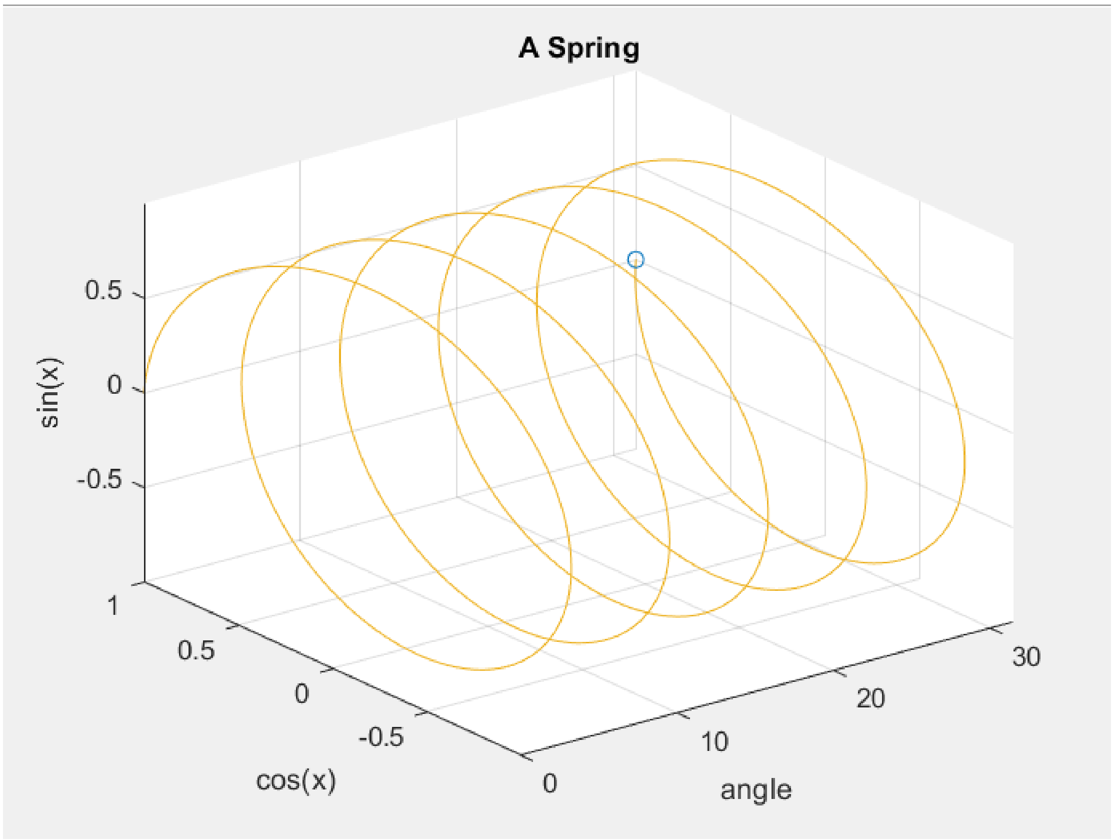
Aim: Creating three dimensional plots

Apparatus: MATLAB software

Code:

```
clear, clc

x = linspace(0,10*pi,1000);

y = cos(x);

z = sin(x);

% plot3(x,y,z)

comet3(x,y,z)

grid

xlabel('angle'), ylabel('cos(x)'), zlabel('sin(x)'), title('A Spring')
```

Result:



- **Program 12**

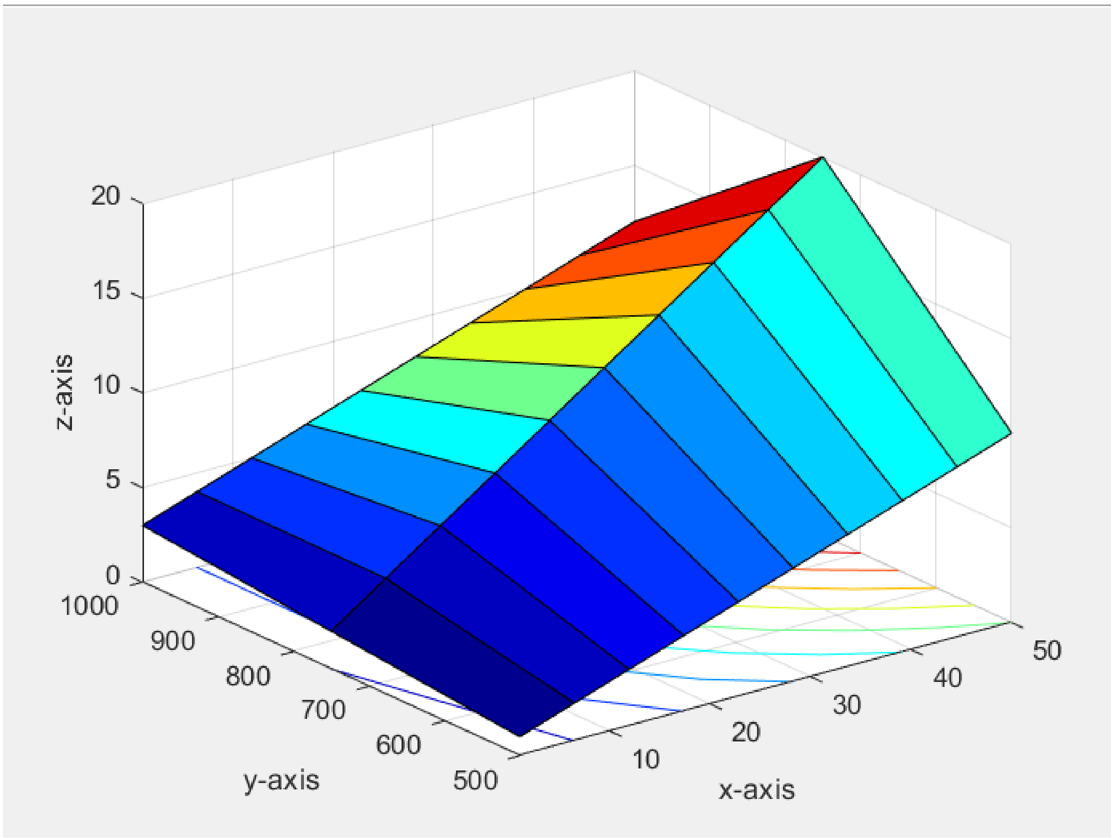
Aim: Creating three dimensional plots

Apparatus: MATLAB software

Code:

```
clear, clc
x = linspace(1,50,10);   %10 elements
y = linspace(500,1000,3); %3 elements
z = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10;
    2, 4, 6, 8, 10, 12, 14, 16, 18, 20;
    3, 4, 5, 6, 7, 8, 9, 10, 11, 12]; %3 rows and 10 columns
% IMPORTANT NOTE:
%  The x vector must have the same number of elements as the number of columns
% in the z vector and the y vector must have the same number of elements as
% the number of rows in the z vector
mesh(x,y,z)%Creates a three-dimensional line plot
% surf(x,y,z)%Creates a surface plot; similar to the mesh function
% contour(x,y,z) %Generates a contour plot
surfc(x,y,z)%Creates a combined surface plot and contour plot


xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
%setting color scheme using colormap
%following is the list of colormaps
% autumn spring summer winter
% hot    cool   bone   copper
% colorcube hsv  prism  flag
% pink white jet (default)
%
%
colormap('jet')
```

Result:



- **Program 13**

Aim: Creating three dimensional plots

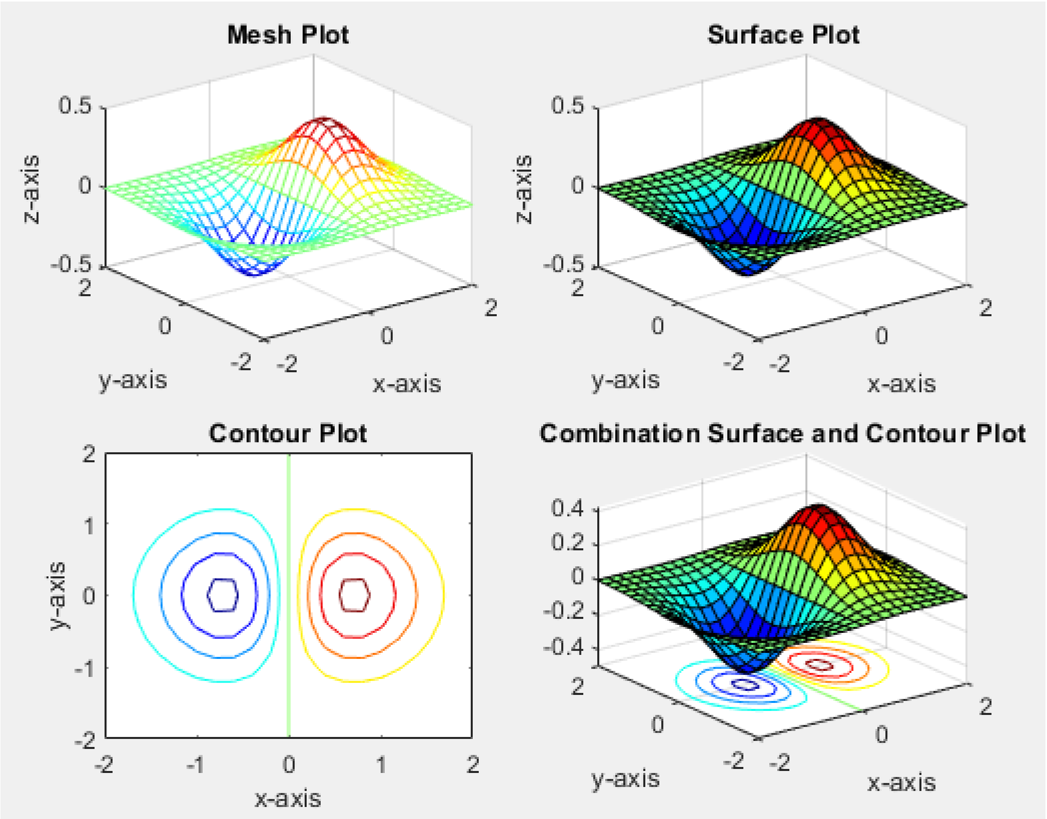
Apparatus: MATLAB software

Code:

```
clear, clc

x= [-2:0.2:2];

y= [-2:0.2:2];

[X,Y] = meshgrid(x,y);

Z = X.*exp(-X.^2 - Y.^2);

xlabel('x-axis')

ylabel('y-axis')

zlabel('z-axis')

subplot(2,2,1)

mesh(X,Y,Z)
```

```
title('Mesh Plot'), xlabel('x-axis'), ylabel('y-axis'),
zlabel('z-axis')
subplot(2,2,2)
surf(X,Y,Z)
title('Surface Plot'), xlabel('x-axis'), ylabel('y-axis'),
zlabel('z-axis')
subplot(2,2,3)
contour(X,Y,Z)
xlabel('x-axis'), ylabel('y-axis'), title('Contour Plot')
subplot(2,2,4)
surfc(X,Y,Z)
xlabel('x-axis'), ylabel('y-axis')
title('Combination Surface and Contour Plot')
```

Result:



EXERCISE – 5

Exercise on Differentiation & Integration problem

- **Program: Differentiation**

Aim: Differentiation

Apparatus: MATLAB software

Code:

```
syms x

f = sin(5*x);

diff(f)


g = exp(x)*cos(x);

y = diff(g)

%To find the derivative of g for a given value of x, substitute x for the value using subs and return a numerical value using vpa. Find the derivative of g at x = 2.

vpa(subs(y,x,2))

%To take the second derivative of g,

diff(g,2)

%or

diff(diff(g))

%to take the derivative of a constant, you must first define the constant as a symbolic expression

c = sym('5');

diff(c)

%To differentiate an expression that contains more than one symbolic variable, specify the variable that you want to differentiate with respect to. The diff command then calculates the partial derivative of the expression with respect to that variable.

syms s t

f = sin(s*t);

diff(f,t)

diff(f,s)

%The diff function can also take a symbolic matrix as its input. In this case, the differentiation is done element-by-element.
```

```
syms a x

A = [cos(a*x),sin(a*x);-sin(a*x),cos(a*x)]

diff(A)

%To calculate the Jacobian matrix, J, of this transformation, use the jacobian function. The mathematical notation for J is

syms r l f

x = r*cos(l)*cos(f);

y = r*cos(l)*sin(f);

z = r*sin(l);

J = jacobian([x; y; z], [r l f])

detJ = simplify(det(J))
```

Result:

```
ans =


5*cos(5*x)




y =


exp(x)*cos(x) - exp(x)*sin(x)




ans =


-9.7937820180676088383807818261614




ans =


-2*exp(x)*sin(x)
```

```
ans =

-2*exp(x)*sin(x)


ans =

0


ans =

s*cos(s*t)


ans =

t*cos(s*t)


A =

[  cos(a*x), sin(a*x)]
[ -sin(a*x), cos(a*x)]


ans =

[ -a*sin(a*x),  a*cos(a*x)]
[ -a*cos(a*x), -a*sin(a*x)]



J =


[ cos(f)*cos(l), -r*cos(f)*sin(l), -r*cos(l)*sin(f)]
[ cos(l)*sin(f), -r*sin(f)*sin(l),  r*cos(f)*cos(l)]
[       sin(l),         r*cos(l),                0]



detJ =


-r^2*cos(l)
```

- **Program: Integration**

Aim: Integration

Apparatus: MATLAB software

Code:

```
syms x n
f = x^n;
int(f)
syms y
f = y^(-1);
int(f)
syms x n
f = n^x;
int(f)
syms a b theta
f = sin(a*theta+b);
```

```
int(f)
syms u
f = 1/(1+u^2);
int(f)
syms x
f = exp(-a*x^2);
int(f, x, -inf, inf)
syms a x
f = 1/(a^2 + x^2);
F = int(f, x, -inf, inf)
```

Result:

```
ans =


piecewise(n == -1, log(x), n ~= -1, x^(n + 1)/(n + 1))



ans =


log(y)



ans =


n^x/log(n)



ans =


-cos(b + a*theta)/a
```

```
ans =


atan(u)



ans =


piecewise(a < 0, Inf, 0 <= real(a) | (angle(a) in Dom::Interval(-pi/2, pi/2) | angle(a) in {-pi/2, pi/2}) & a
~= 0, pi^(1/2)/a^(1/2), real(a) < 0 & ~angle(a) in Dom::Interval(-pi/2, pi/2) & ~angle(a) in {-pi/2, pi/2}
& ~a < 0, int(exp(-x^2*a), x, -Inf, Inf))



F =


(pi*signIm(1i/a))/a


>>
```

EXERCISE – 6

Exercise on Differential Equations

- **Program: Differential Equations**

Aim: Solve this differential equation.

Apparatus: MATLAB software

Code:

```
%dy/dt = ty
%First, represent y by using syms to create the symbolic function y(t).
syms y(t)
%Define the equation using == and represent differentiation using the diff function
ode = diff(y,t) == t*y
%Solve the equation using dsolve.
ySol(t) = dsolve(ode)
%Solve Differential Equation with Condition
cond = y(0) == 2;
ySol(t) = dsolve(ode,cond)
%Nonlinear Differential Equation with Initial Condition
%((dy/dt)+y)^2=1
%y(0)=0
syms y(t)
ode = (diff(y,t)+y)^2 == 1;
cond = y(0) == 0;
ySol(t) = dsolve(ode,cond)
%Second-Order ODE with Initial Conditions
%d^2y/dx^2 = cos(2x)-y
%y(0)=1
%y'(0)=1
syms y(x)
Dy = diff(y);
```

```
ode = diff(y,x,2) == cos(2*x)-y;

cond1 = y(0) == 1;

cond2 = Dy(0) == 0;

conds = [cond1 cond2];

ySol(x) = dsolve(ode,conds);

ySol = simplify(ySol)


%example

%dy/dt+4y(t)=e^(-t)

%y(0)=1

syms y(t)

ode = diff(y)+4*y == exp(-t);

cond = y(0) == 1;

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

Result:

```
ode(t) =


diff(y(t), t) == t*y(t)



ySol(t) =


C4*exp(t^2/2)



ySol(t) =


2*exp(t^2/2)
```

```
ySol(t) =


 exp(-t) - 1

 1 - exp(-t)



ySol(x) =


1 - (8*sin(x/2)^4)/3



ySol(t) =


exp(-t)/3 + (2*exp(-4*t))/3
```

- **Program: First order ordinary Differential Equations**

Aim: First-Order Linear ODE with Initial Condition

Apparatus:

Code:

```
% First, represent y by using syms to create the symbolic function y(t).
syms y(t)
ode = diff(y,t) == t*y
% Solve the equation using dsolve.
ySol(t) = dsolve(ode)
%solve with initial conditions
cond = y(0) == 2;
ySol(t) = dsolve(ode,cond)
```

Result:

```
ode(t) =

diff(y(t), t) == t*y(t)


ySol(t) =

C4*exp(t^2/2)


ySol(t) =

2*exp(t^2/2)
```

EXERCISE – 7

Exercise on Gauss Elimination Method

- **Program: Gauss Elimination Technique**

Aim: Gauss elimination Technique

Apparatus: MATLAB software

Code:

```
C = [1 2 -1; 2 1 -2; -3 1 1]

b= [3 3 -6]'

A = [C b];                              %Augmented Matrix

n= size(A,1);                             %number of eqns/variables

x = zeros(n,1);                           %variable matrix [x1 x2 ... xn] coulmn

for i=1:n-1

   for j=i+1:n

      m = A(j,i)/A(i,i)

      A(j,:) = A(j,:) - m*A(i,:)

   end

end

x(n) = A(n,n+1)/A(n,n)

for i=n-1:-1:1

   summ = 0

   for j=i+1:n

      summ = summ + A(i,j)*x(j,:)

      x(i,:) = (A(i,n+1) - summ)/A(i,i)

   end

end
```

Result:

gauss_elimination_technique

```
C =

     1     2    -1
     2     1    -2
    -3     1     1


b =

     3
     3
    -6


m =

     2


A =

     1     2    -1     3
     0    -3     0    -3
    -3     1     1    -6


m =

    -3
```

```
A =

     1     2    -1     3
     0    -3     0    -3
     0     7    -2     3


m =

   -2.3333


A =

     1     2    -1     3
     0    -3     0    -3
     0     0    -2    -4


x =

     0
     0
     2


summm =

     0
```

```
summ =

     0


x =

     0
     1
     2


summ =

     0


summ =

     2


x =

     1
     1
     2


summ =
     0



x =


     3
     1
     2
```

EXERCISE-8

EXERCISE ON SOLVING OF POLYNOMIAL EQUATIONS(BISECTION,NEWTON-RAPHSON METHOD ETC)

AIM:

ROOTS AND POLYVALUE OF POLYNOMIAL

ADDITION & MULTIPLICATION POLYNOMIAL

PARTIAL FRACTION EXPANSION

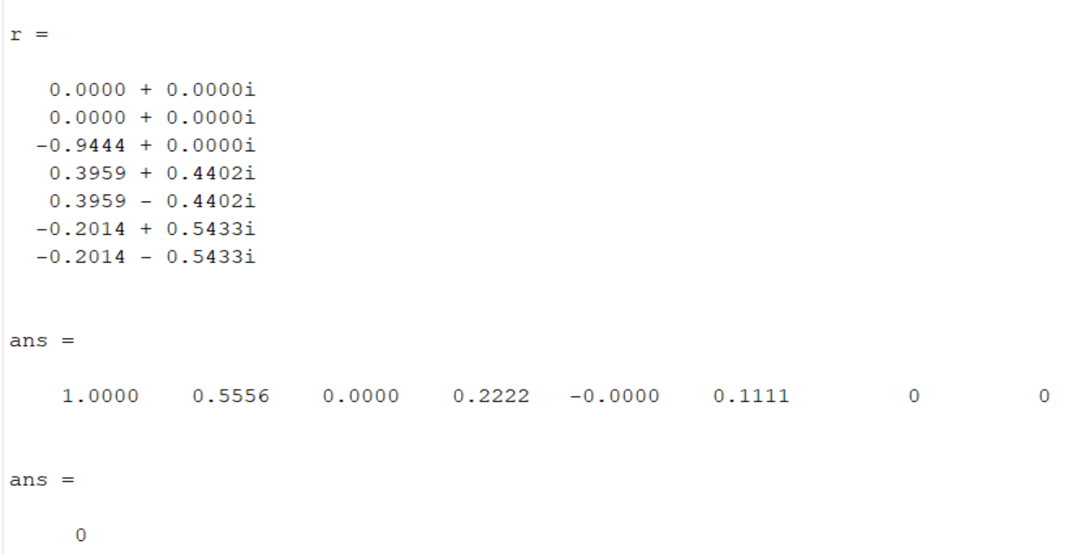
APPARATUS: using matlab programming software

CODE:

ROOTS AND POLYVALUE OF POLYNOMIAL:

```
p=[9 5 0 2 0 1 0 0];
r=roots(p)
poly(r)
polyval(p,0)
```

RESULT:

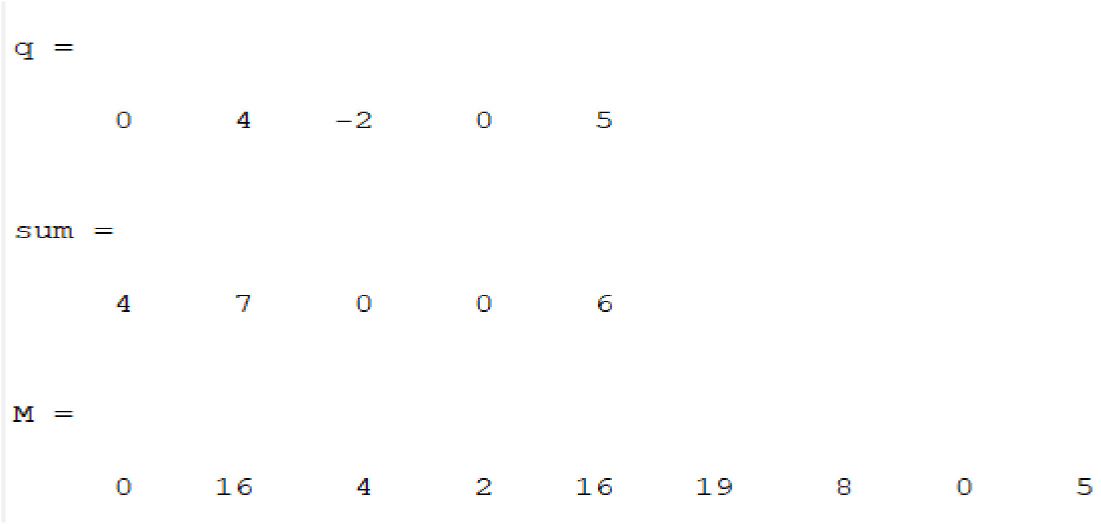


```
r =

   0.0000 + 0.0000i
   0.0000 + 0.0000i
  -0.9444 + 0.0000i
   0.3959 + 0.4402i
   0.3959 - 0.4402i
  -0.2014 + 0.5433i
  -0.2014 - 0.5433i


ans =

    1.0000    0.5556    0.0000    0.2222   -0.0000    0.1111         0         0


ans =

     0
```

ADDITION & MULTIPLICATION POLYNOMIAL:

```
p=[4 3 2 0 1 ];
q=[0 4 -2 0 5]
sum=p+q
M=conv(p,q)
```

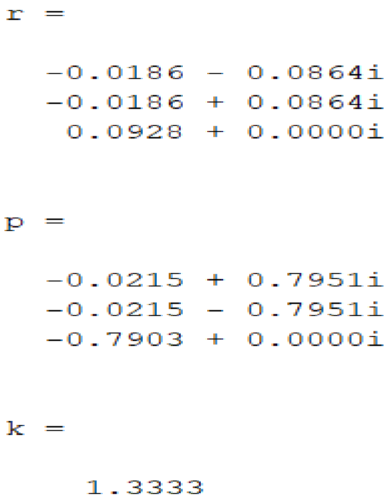
RESULT:



```
q =

     0     4    -2     0     5


sum =

     4     7     0     0     6


M =

     0    16     4     2    16    19     8     0     5
```

PARTIAL FRACTION EXPANSION:

```
P=[8 7 6 5];
Q=[6 5 4 3];
[r,p,k]=residue(P,Q)
RESULT:
```



```
r =

  -0.0186 - 0.0864i
  -0.0186 + 0.0864i
   0.0928 + 0.0000i


p =

  -0.0215 + 0.7951i
  -0.0215 - 0.7951i
  -0.7903 + 0.0000i


k =

    1.3333
```

EXERCISE-9

PROGRAM ON STIFFNESS MATRIX CALCULATIONS FOR FINITE ELEMENTS(1D-BAR,BEAM&OTHERS)INTRODUCTION TO FINITE DIFFRENCE METHOD FOR A BEAM PROBLEM

AIM:

FINITE DIFFERENCE METHOD

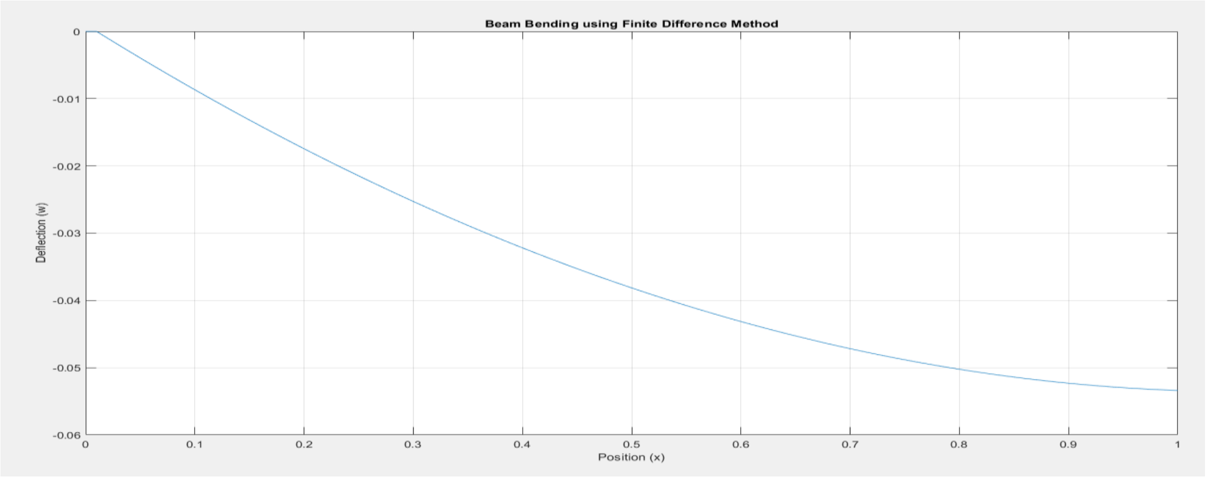
APPARATUS: **using matlab programming software**

Code:

FINITE DIFFERENCE METHOD:

```
function fdm_beam_bending()
    % Parameters
    L = 1;          % Length of the beam
    E = 1e7;        % Young's modulus
    I = 1e-4;       % Moment of inertia
    q0 = -1e4;      % Distributed load

    % Spatial discretization
    nx = 100;       % Number of spatial points
    dx = L / (nx - 1);
    x = linspace(0, L, nx)';

    % Allocate memory for deflection at each spatial point
    w = zeros(nx, 1);

    % Apply boundary conditions
    w(1) = 0;       % w(0) = 0 (fixed boundary)

    % Finite Difference Method
    for i = 2:nx-1
        % Fourth-order accurate central difference for second derivative
        w_xx = (w(i+1) - 2*w(i) + w(i-1)) / dx^2;

        % Calculate the distributed load q(x) at the current spatial point
        q_x = q0 * (L - x(i));

        % Update deflection using FDM equation
        w(i+1) = w(i) + dx^2 / (E * I) * (q_x - w_xx);
    end

    % Plot the deflection profile
    plot(x, w);
    xlabel('Position (x)');
    ylabel('Deflection (w)');
    title('Beam Bending using Finite Difference Method');
    grid on;
end
```

RESULTS:



```
function fdm_plate_bending()
    % Parameters
    Lx = 1;          % Length of the plate in x-direction
    Ly = 1;          % Length of the plate in y-direction
    Nx = 5;         % Number of spatial points in x-direction
    Ny = 5;         % Number of spatial points in y-direction
    Dx = Lx / (Nx - 1);
    Dy = Ly / (Ny - 1);
    D = 1e-5;        % Flexural rigidity
    q0 = -1e-4;      % Magnitude of distributed load

    % Create spatial grid
    x = linspace(0, Lx, Nx);
    y = linspace(0, Ly, Ny);
    [X, Y] = meshgrid(x, y);

    % Allocate memory for deflection at each spatial point
    w = zeros(Nx, Ny);

    % Apply boundary conditions (all sides simply supported)
    w(:, 1) = 0;          % w(x, 0) = 0
    w(:, Ny) = 0;         % w(x, Ly) = 0
    w(1, :) = 0;          % w(0, y) = 0
    w(Nx, :) = 0;         % w(Lx, y) = 0

    % Finite Difference Method
    for i = 2:Nx-1
        for j = 2:Ny-1
            % Fourth-order accurate central differences for second
derivatives
            w_xx = (w(i+1, j) - 2*w(i, j) + w(i-1, j)) / Dx^2;
            w_yy = (w(i, j+1) - 2*w(i, j) + w(i, j-1)) / Dy^2;
            w_xxyy = (w(i+1, j+1) - w(i-1, j+1) - w(i+1, j-1) + w(i-1, j-
1)) / (4*Dx*Dy);

            % Calculate the distributed load q(x, y) at the current spatial
point
            q_xy = q0 * (Lx - x(i)) * (Ly - y(j));

            % Update deflection using FDM equation
            w(i, j) = (q_xy - D * (w_xx + 2*w_xxyy + w_yy)) / (2*D);
        end
    end
    % Plot the deflection profile
    figure;
    surf(X, Y, w);
    xlabel('x');
    ylabel('y');
    zlabel('Deflection (w)');
    title('Plate Bending using Finite Difference Method');
    colormap('jet');
    colorbar;
    grid on;
end
```

RESULTS:

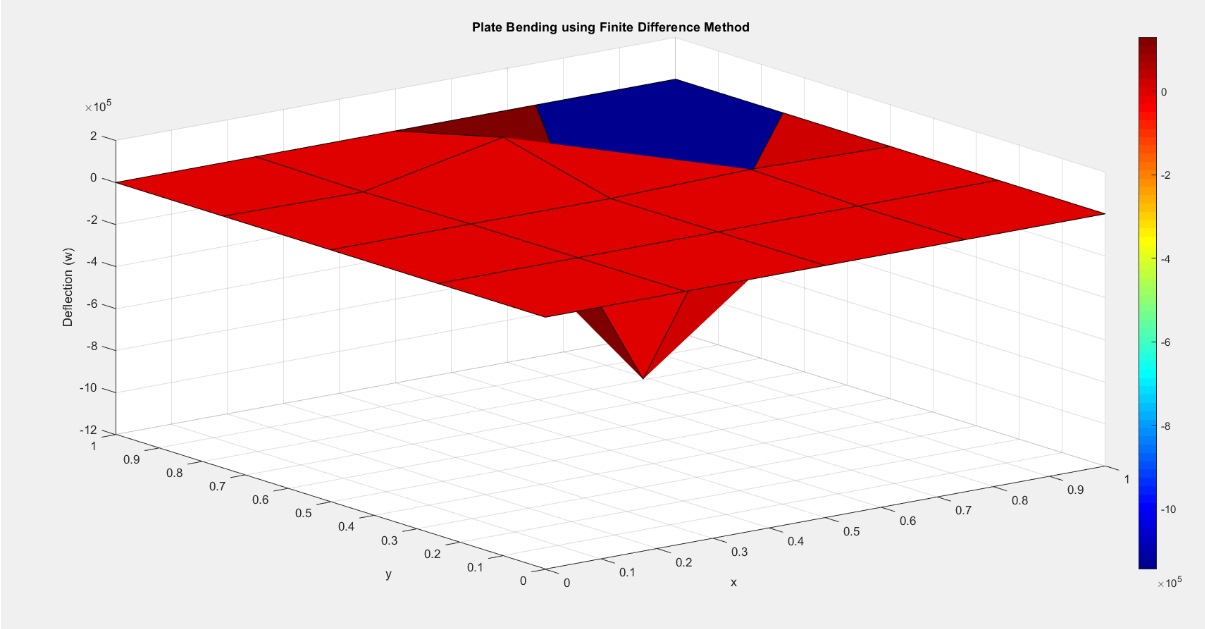


Plate Bending using Finite Difference Method
×10^5
Deflection (w)
y
x
×10^5