Sr.no.	Date	Title	Page no.	Signature
1		Program to find the rank of a matrix.		
2		Test the consistency of the following system of equations.		
3		Solution of system of Linear equations by Gauss seidel method.		
4		Program to find the largest eigen value and the corresponding eigen vector of the matrix.		
5		2D plots for cartesian and polar curves.		
6		Program to find the angle between two polar curves and radius of curvature.		
7		Program to finding partial derivatives and Jacobians of functions.		
8		Program to find the maxima and minima of a function.		
9		Program to find the solution of first order differential equation.		
10		Python program to solve double integration and triple integration.		

Program-1(a):
#Program to find Rank of a matrix

$$A = \begin{matrix} 0 & 2 & 3 & 4 \\ 2 & 3 & 5 & 4 \\ 4 & 8 & 13 & 12 \end{matrix}$$

```
import numpy as np
A= np.array([[0,2,3,4],[2,3,5,4],[4,8,13,12]])
print("A=" ,A)
rank=np.linalg.matrix_rank(A)
print(" The rank of the given matrix =",rank)
```

Output:

```
A  =  [ [0     2  3     4]
        [2     3   5    4]
        [4   8   13  12] ]
The rank of the given matrix = 2
```

1(b)

$$B = \begin{matrix} 1 & 2 & 4 & 3 \\ 2 & 4 & 6 & 8 \\ 4 & 8 & 12 & 16 \\ 1 & 2 & 3 & 4 \end{matrix}$$

```
import numpy as np
B= np.array([[1,2,4,3],[2,4,6,8],[4,8,12,16],[1,2,3,4]])
print("B=" ,B)
rank=np.linalg.matrix_rank(B)
print(" The rank of the given matrix =",rank)
```

Output:

```
B=  [[ 1   2   4   3]
     [ 2  4   6   8]
     [ 4  8  12  16]
     [ 1  2   3   4]]
 The rank of the given matrix = 2
```

Program-2(a):

```
#Examine the consistency of the following system of equations and solve for
x+2y-z=1 , #2x+y+4z=2 , 3x+3y+4z=1

import numpy as np
A=np.matrix([[1,2,-1],[2,1,4], [3,3,4]])
B=np.matrix([[1],[2],[1]])
AB=np.concatenate ((A,B), axis=1)
rA=np.linalg. matrix_rank(A)
rAB=np.linalg. matrix_rank (AB)
n=A.shape[1]
if(rA= =rAB):
    print("The System is consistent")
    if(rA= =n):
        print("The system has unique solution")
        print (np. linalg.solve(A,B))
    else:
        print("The System has infinitely many solutions")
else:
    print("The system of equations is inconsistent")
```

Output:

```
The System is consistent
The system has unique solution
[[ 7]
[-4]
[-2]]
```

2(b):

```
#5x+y+3z=20 , 2x+5y+2z=18 , 3x+2y+z=14

import numpy as np
A=np.matrix([[5,1,3], [2,5,2], [3,2,1]])
B=np.matrix([[20], [18], [14]])
AB=np.concatenate ((A,B),axis=1)
rA=np.linalg.matrix_rank(A)
rAB=np.linalg.matrix_rank(AB)
n=A.shape[1]
if(rA= =rAB):
    print("The System is consistent")
    if(rA= =n):
        print("The system has unique solution")
        print (np. linalg.solve(A,B))
    else:
        print("The System has infinitely many solutions")
else:
    print("The system of equations is inconsistent")
```

Output:

```
The System is consistent
The system has unique solution
[[3.]
[2.]
[1.]]
```

Program-3(a):

```
#Solution of system of Linear equations by Gauss seidel method 5x+2y+z=12 ,
#x+4y+2z=15 , x+2y+5z=20

f1=lambda x, y, z: (12-2*y-z)/5
f2=lambda x, y, z: (15-x-2*z)/4
f3=lambda x, y, z: (20-x-2*y)/5
x0=0
y0=0
z0=0
count=1
e=float (input ('Enter tolerable error:'))
print(‘\ncount\tx\ty\tz\n')
condition=True
while condition:
    x1=f1(x0, y0, z0)
    y1=f2(x1, y0, z0)
    z1=f3(x1, y1, z0)
    print (‘%d\t%0.4f\t%0.4f\t%0.4f\n' % (count, x1, y1, z1))
    e1=abs (x0-x1);
    e2=abs (y0-y1);
    e3=abs (z0-z1);
    count += 1
    x0 = x1
    y0= y1
    z0 = z1
    condition=e1>e and e2>e and e3>e
print ('\n solution: x=%0.3f, y=%0.3f and z=%0.3f\n’ % (x1, y1, z1))
```

Output:

```
Enter tolerable error:0.01
count   x        y        z
  1     2.4000   3.1500   2.2600
  2     0.6880   2.4480   2.8832
  3     0.8442   2.0974   2.9922
  4     0.9626   2.0132   3.0022
solution: x=0.963, y=2.013 and z=3.002
```

3(b) :

```
#20x+y-2z=17 , 3x+20y-z=-18 , 2x-3y+20z=25

f1=lambda x,y, z: (17 -y+2*z) /20
f2=lambda x,y, z:(-18-3*x+z)/20
f3=lambda x,y, z: (25-2*x+3*y) /20
x0=0
y0=0
z0=0
count=1
e=float (input ('Enter tolerable error:'))
print(‘\ncount\tx\ty\tz\n')
condition=True
while condition:
        x1=f1(x0, y0, z0)
        y1=f2(x1, y0, z0)
        z1=f3(x1, y1, z0)
        print (‘%d\t%0.4f\t%0.4f\t%0.4f\n' % (count, x1, y1, z1))
        e1=abs (x0-x1);
        e2=abs (y0-y1);
        e3=abs (z0-z1);
        count += 1
        x0 = x1
        y0= y1
        z0 = z1
        condition=e1>e and e2>e and e3>e
print ('\n solution: x=%0.3f, y=%0.3f and z=%0.3f\n’ % (x1, y1, z1))
```

Output:

```
Enter tolerable error:0.01
count   x         y         z
  1     0.8500    -1.0275   1.0109
  2     1.0025    -0.9998   0.9998
  3     1.0000    -1.0000   1.0000
solution: x=1.0000, y=-1.0000 and z=1.0000
```

Program-4(a):

#Write the Python program to Find the largest eigen value and the corresponding

#eigen vector of the matrix A by the power method given that A= $\begin{matrix} 6 & -2 & 2 \\ -2 & 3 & -1 \\ 2 & -1 & 3 \end{matrix}$

#taking the initial Eigen vector as $[1, 1, 1]^T$.

```
import numpy as np
x=np. array ([1,1, 1])
a=np. array ([[6, -2,2], [-2, 3, -1], [2, -1, 3]])
def normalize(x):
        f=abs (x) .max ()
        xn=x/x .max ()
        return f, xn
for i in range (40) :
        x=np.dot (a, x)
        lambda1, x= normalize (x)
print('eigenvalue:' ,lambda1)
print('eigenvector:', x)
```

Output:

eigenvalue: 8.8

eigenvector: [1. -0.5 0.5]

4(b): A = $\begin{matrix} 2 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 2 \end{matrix}$ taking the initial eigen vector as $[1 ,0 ,0]^T$

```
import numpy as np
x=np. array ([1,0,0])
a=np. array ([[2,0,1], [0,2,0], [1,0,2]])
def normalize(x):
        f=abs (x) .max ()
        xn=x/x .max ()
        return f, xn
for i in range (40) :
        x=np.dot (a, x)
        lambda1, x= normalize (x)
print('eigenvalue:' ,lambda1)
print('eigenvector:', x)
```

Output:

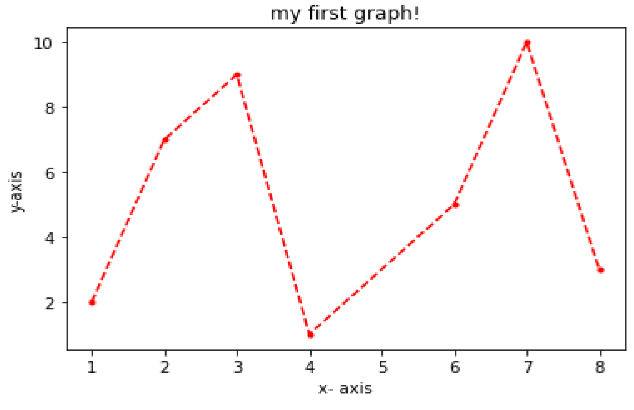
eigenvalue: 3.0

eigenvector: [1. 0. 1]

Program-5 (a) :
#Plotting a Line(line plot)

```
import numpy as np
import matplotlib. pyplot as plt
x=[1,2,3, 4, 6, 7, 8]
y=[2,7,9, 1,5, 10, 3)
plt.plot (x, y, 'r--')
plt.xlabel('x- axis')
plt.ylabel('y-axis')
plt.title('My first graph!")
plt. show ()
```

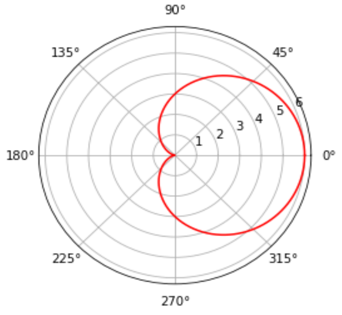
Output:



5 (b) :
#Plot cardioid r1=3*(1+cos theta)

```
from pylab import*
theta=linspace(0, 2*np . pi, 1000)
r1=3+3*cos (theta)
polar (theta, r1, 'r')
show()
```

Output:



Program6 (a):

```
#Find the angle between two polar curves, curvature and radius of curvature
# Find the angle between the curves r=4(1 + cos t) and r= 5(1 -cos t)

from sympy import*
r,t=symbols ('r, t')
r1 = 4*(1+cos (t))
r2 = 5*(1-cos (t))
dr1 = diff (r1,t)
dr2 = diff (r2,t)
t1=r1/dr1
t2=r2/dr2
q = solve (r1-r2,t)
w1=t1.subs ({t:float (q[1] )})
w2=t2.subs ({t:float (q[1] )} )
y1 = atan (w1)
y2 = atan (w2)
W= abs (y1-y2)
print(' Angle between curves in radian is %0.3f '%(w) )
```

Output:

Angle between curves in radian is 1.571

6(b):

```
# Find the radius of curvature for r = asin(nt) at t = pi/2 and n=1

from sympy import*
t,r,a,n = symbols ('t r a n')
r= a*sin(n*t)
r1 = Derivative (r,t).doit ()
r2 = Derivative (r1,t) . doit ()
rho = (r**2+r1**2)**1.5/(r**2+2*r1**2-r*r2) ;
rho1 = rho.subs (t,pi/2)
rho1 = rho1.subs (n,1)
print ("The radius of curvature is")
display (simplify (rho1))
```

Output:

The radius of curvature is

$(a^2)^{1.5}/2a^2$

Program:7(a)
#Finding partial derivatives and Jacobians of functions
#Prove that if u=e^x(x*cos (y) -y*sin (y)) then u_{xx}+u_{yy}=0

```
from sympy import*
x,y=symbols('x,y')
u=exp(x) *(x*cos (y)-y*sin (y) )
display (u)
dux=diff (u, x)
duy=diff (u,y)
uxx=diff (dux, x)
uyy=diff (duy, y)
w=uxx+uyy
w1=simplify (w)
print(' uxx+uyy:',float (w1))
```

Output:
(x cos (y) - y sin (y)) e^x
u_{xx}+u_{yy} :0.0

7(b)

```
#If u=x*y/z, v=y*z/x, w=z*x/y then find the Jacobian of ( u, v, w) w.r.t (x, y ,z)

from sympy import*
x,y, z=symbols (' x,y, z')
u=x*y/z
v=y*z/x
w=z*x/y
dux=diff (u, x)
duy=diff (u, y)
duz=diff (u, z)
dvx=diff(v, x)
dvy=diff (v,y)
dvz=diff (v, z)
dwx=diff (w, x)
dwy=diff (w, y)
dwz=diff (w, z)
J=Matrix([[dux, duy, duz ], [dvx, dvy, dvz], [dwx, dwy, dwz]])
print ("The jacobian matrix is\n")
display (j)
Jac=det (j).doit ()
print(‘ J=',Jac)
```

Output:

The jacobian matrix is

$$\begin{matrix} \frac{y}{z} & \frac{x}{z} & -\frac{zy}{z^2} \\ -\frac{yz}{x^2} & \frac{z}{x} & \frac{y}{x} \\ \frac{z}{y} & -\frac{xz}{y^2} & \frac{x}{y} \end{matrix}$$

J=4

Program:8(a)
#Find the maxima and minima of f(x, y) =x^2+y^2+3x-3y+4

```
from sympy import*
x,y=symbols ('x,y')
f=x** 2+y**2+3*x-3*y+4
d1=diff(f, x)
d2=diff(f, y)
sp1=solve (d1)
sp2=solve (d2)
A=diff(f, x, 2)
C=diff(f, y, 2)
B=diff(diff(f, y), x)
print(' The function value delta is :')
q1=A.subs ({x:sp1, y:sp2}).evalf()
q2=C.subs ({x:sp1, y:sp2}).evalf()
q3=B.subs ({x:sp1, y:sp2}).evalf()
delta=A*C-B**2
print (‘delta:’,delta)
print(‘q1:’,q1)
if(delta>0 and A<0):
	print ("f takes maximum")
elif(delta>0 and A>0):
	print ("f takes minimum”)
if(delta<0):
	print ("the point is a saddle point")
if(delta==0) :
	print ("further tests required")
```

Output:

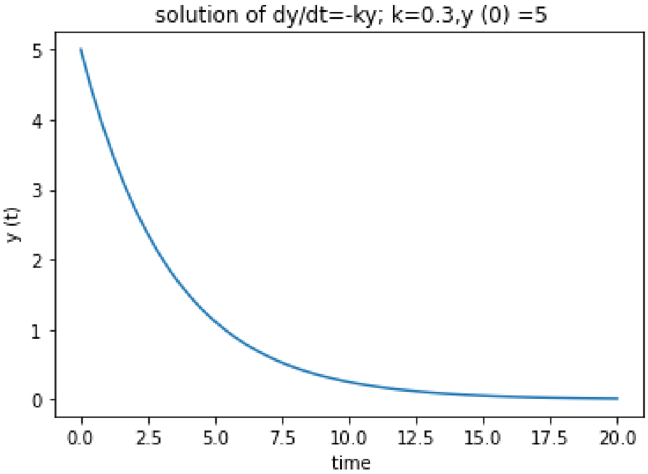
```
function value is
delta: 4
q1: 2.0000000000000
f takes minimum
```

Program:9

```
#Solution of first order differential equation
#Solve dy/dt=-ky with parameter k=0.3 and y(0) =5

import numpy as np
from scipy.integrate import odeint
import matplotlib. pyplot as plt
def model (y, t):
        k=0.3
        return -k*y
 y0=5
 t=np.linspace (0,20)
 y=odeint (model, y0, t)
plt.plot (t,y)
plt.title(' solution of dy/dt=-ky; k=0.3,y (0) =5' )
plt. xlabel (' time')
plt. ylabel (' y (t)’)
plt. show
```

Output:



Program:10(a)

Python program to find $\int_0^1 \int_x^{\sqrt{x}} (x^2 + y^2)$ dy dx

```
from sympy import*
x,y=sy.symbols('x,y')
g=x**2+y**2
integrate(integrate(g,(y,x,sqrt(x))),(x,0,1)).simplify()
```

Output:

3/35

#10(b):

#Python program to find $\int_0^1 \int_0^{\sqrt{1-x^2}} \int_0^{\sqrt{1-x^2y^2}} xyz$ dz dy dx

```
from sympy import*
x,y,z=sy.symbols('x,y,z')
g=x*y*z
integrate(integrate(integrate(g,(z,0,sqrt(1-x**2-y**2))),(y,0,sqrt(1-
x**2))),(x,0,1)).simplify()
```

Output:

1/48