Lab 1_Program to compute area and volume

Area

$A = \iint\limits_D dydx$

$A = \iint\limits_D dxdy$

1. Find the area bounded by the curves $y^2 = x^3$ and $x^2 = y^3$

```
from sympy import *
x, y = symbols("x y")
f = 1
I = integrate(f, (y, x**(3/2), x**(2/3)), (x, 0, 1))
print("Area=",I)
```

```
Area= 0.200000000000000
```

Polar form

$A = \iint\limits_D rdrd\theta$

Find the area lying inside the cardioid $r = a(1 + cos\theta)$ and outside the circle $r = a$

```
from sympy import *
r,t,a = symbols("r t a")
I1=2*integrate(r,(r, a, a*(1+cos(t))),(t, 0, pi/2))
I3=simplify(I1)
pprint(I3)
```

```
 2
a ⋅(π + 8)
──────────
    4
```

Activity-1

1. Find the area of a plate in the form of a quadrant of the ellipse $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$.
2. Find the area between the parabolas $y^2 = 4ax$ and $x^2 = 4ay$.
3. Find the area lying between the parabola $y = x^2$ and the line $x + y - 2 = 0$.
4. Find the area lying between the parabola $y = 4x - x^2$ and the line $y = x$.
5. Find the area of the cardiod $r = a(1 + \cos\theta)$ in second quadrant.
6. Find the area of the cardiod $r = a(1 + \cos\theta)$.
7. Find the area of the Lemniscate $r^2 = a^2 \cos 2\theta$.
8. Find the area which is inside the circle $r = 3a\cos\theta$ and outside the cardioid $r = a(1 + cos\theta)$.

Volume

Find the volume of the tetrahedron bounded by the plane $\frac{x}{a} + \frac{y}{b} + \frac{z}{c} = 1$ and the co-ordinate planes

```
from sympy import *
x, y,z,a,b,c = symbols("x y z a b c")
f = 1
I = integrate(f, (z,0,c*(1-(x/a)-(y/b))) ,(y, 0, b*(1-(x/a))), (x, 0, a))
pprint(I)
```

```
a⋅b⋅c
─────
  6
```

Find volume of the sphere $x^2 + y^2 + z^2 = a^2$ by using triple integration

```
from sympy import *
x,y,z,a,k=symbols('x y z a k',positive=True)
f=1
I1=integrate(f,(z,0,sqrt(k**2-y**2)),(y,0,k))
I2=I1.subs(k,sqrt(a**2-x**2))
V=integrate(I2,(x,0,a))
display(8*V)
```

$$\frac{4\pi a^3}{3}$$

Activity-2

1. Find the volume of the ellipsoid $\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=1$.
2. Find the volume of the sphere $x^2+y^2+z^2=a^2$.
3. Calculate the volume of the solid bounded by the planes $x=0, y=0$,z=0$and x+y+z=a$.
4. Find the volume bounded by the cylinder $x^2+y^2=4$ and the plane $y+z=4$ and $z=0$.

Lab 2_Evaluation of improper Integrals

1. Evaluate $\int_0^\infty \frac{1}{1+x^2}dx$

```
from sympy import *
x =symbols(' x', positive=True)
integrate(1/(1+x**2),(x,0,oo))
```

$$\frac{\pi}{2}$$

2. Evaluate $\int\limits_0^\infty \frac{t^4}{4^t}dt$

```
from sympy import *
t =symbols(' t', positive=True)
integrate((t**4)/(4**t),(t,0,oo))
```

$$\frac{3}{4\log{(2)}^5}$$

3. Evaluate $\int\limits_{-\infty}^\infty \frac{1}{a^2+x^2}dx$

```
from sympy import *
x,a =symbols(' x, a', positive=True)
integrate(1/(a**2+x**2),(x,-oo,oo))
```

$$\frac{\pi}{a}$$

4. Evaluate $\Gamma\left(\frac{5}{2}\right)$

```
from sympy import *
print('Γ(5/2)=',gamma(5/2))
```

```
Γ(5/2)= 1.32934038817914
```

5. Evaluate $\beta\left(\frac{5}{2},\frac{7}{2}\right)$

```
from sympy import *
print('β(5/2,7/2)=',beta(5/2,7/2))
```

```
β(5/2,7/2)= 0.0368155389092554
```

Activity-3

1. Evaluate $\int\limits_0^\infty 3^{-4x^2}dx$.
2. Evaluate $\int\limits_0^\infty \frac{x^4}{4^x}dx$.
3. Evaluate $\int\limits_0^\infty e^{-kx}x^{p-1}dx, k>0$.

4. Evaluate $\int_0^\infty e^{-t^2} t^{2n-1} dt$.

5. Evaluate $\int_0^\infty \frac{x^{n-1}}{1+x} dx$.

6. Evaluate $\int_0^\infty \frac{1}{1+y^4} dy$.

7. Evaluate $\int_0^1 x^m (1-x^n)^p dx$.

8. Evaluate $\int_0^1 \frac{1}{\sqrt{1-x^4}} dx$

Lab 3_Finding gradient, divergent, curl and their geometrical interpretation.

Diveregence

Evaluate the diverengence of $2x^2 z\hat{i} - xy^2 z\hat{j} + 3yz^2\hat{k}$ at the point (1,1,1).

```
from sympy.physics.vector import ReferenceFrame
from sympy.physics.vector import *
from sympy import *
R = ReferenceFrame('R')
x,y,z=symbols("x,y,z")
i,j,k=R.x,R.y,R.z
x,y,z=R[0],R[1],R[2]
F=(2*x**2*z)*i-(x*y**2*z)*j+(3*y*z**2)*k
G = divergence(F, R)
f= G.subs(dict(zip(R.varlist, var('x:z'))))
f1=f.subs({x:1,y:1,z:1}).evalf()
display(f1)
```

8.0

Curl

Evaluate the curl of $xyz^2\hat{i} + yzx^2\hat{j} + zxy^2\hat{k}$ at (1,2,3)

```
from sympy.physics.vector import ReferenceFrame
from sympy.physics.vector import *
from sympy import *
R = ReferenceFrame('R')
x,y,z=symbols("x,y,z")
i,j,k=R.x,R.y,R.z
x,y,z=R[0],R[1],R[2]
F=(x*y*z**2)*i+(y*z*x**2)*j+(z*x*y**2)*k
G = curl(F, R)
f= G.subs(dict(zip(R.varlist, var('x:z'))))
f1=f.subs({x:1,y:2,z:3}).evalf()
display(f1)
```

$10.0\hat{\mathbf{r}}_{\mathbf{x}} + 3.0\hat{\mathbf{r}}_{\mathbf{z}}$

Show that the vector field $\vec{f} = 2x^2 z\hat{i} - 10xyz\hat{j} + 3xz^2\hat{k}$ is a solenoidal.

```
from sympy.physics.vector import ReferenceFrame
from sympy.physics.vector import *
from sympy import *
R = ReferenceFrame('R')
x,y,z=symbols("x,y,z")
i,j,k=R.x,R.y,R.z
x,y,z=R[0],R[1],R[2]
F=(2*(x**2)*z)*i-(10*x*y*z)*j+(3*x*(z**2))*k
G = divergence(F, R)
if G==0:
   print("The  given vector is solendial")
else:
   print("The  given vector is not solendial")
```

```
The  given vector is solendial
```

Prove that $\vec{f} = (6xy + z^3)\hat{i} + (3x^2 - z)\hat{j} + (3xz^2 - y)\hat{k}$ is irrotational

```
from sympy.physics.vector import ReferenceFrame
from sympy.physics.vector import *
from sympy import *
```

```
R = ReferenceFrame('R')
x,y,z=symbols("x,y,z")
i,j,k=R.x,R.y,R.z
x,y,z=R[0],R[1],R[2]
F=(6*x*y+z**3)*i+(3*x**2-z)*j+(3*x*z**2-y)*k
G = curl(F, R)
if G==0:
   print("The  given vector is irrotataional")
else:
   print("The  given vector is not irrotataional ")
```

```
The  given vector is irrotataional
```

Activity-4

1. Find the gradient of ϕ where ϕ is
   (i.) $x^2 + y^2 + z^2$.
   (ii.) $3x^2y - y^3z^3$ at $(1, -2, -1)$
   (iii.) $x^2y^2z^3$ at $(1, -1, 2)$
2. If $f(x, y, z) = 3x^2y - y^3z^2$, find ∇f.
3. The force in an electrostatic field given by $f = 4x^2 + 9y^2 + z^2$ has the direction of the gradient ∇f. Find this gradient at $(5, -1, -11)$.
4. The temperature of points in space is given $T(x, y, z) = x^2 + y^2 - z$. A mosquito located at $(1, 1, 2)$ desires to fly in such a direction it will get warm as soon as possible. In what direction should it move?
5. If $\vec{f} = 3x^2y\hat{i} - y^3z^2\hat{j} + xyz^2\hat{k}$, find $\nabla \cdot \vec{f}$.
6. Evaluate diveregence and curl at the point $(1, 2, 3)$ given
   (i.) $\vec{f} = x^2yz\hat{i} + xy^2z\hat{j} + xyz^2\hat{k}$.
   (ii.) $\vec{f} = 3x^2\,\hat{i} + 5xy^2\,\hat{j} + 5xyz^3\hat{k}$.
   (iii.) $\vec{f} = xyz\hat{i} + 3x^2y\hat{j} + (xz^2 - y^2z)\hat{k}$.
7. Show that the vector field $\vec{f} = 3y^4z^2\,\hat{i} + 4x^2z^2\,\hat{j} - 3x^2y^2\hat{k}$ is solenoidal.
8. Verify whether the vector field $\vec{f} = 3xy\hat{i} + 20yz^2\,\hat{j} - 15xz\hat{k}$ is solenoidal or not.
9. Verify whether the vector field $\vec{f} = (x^2 - yz)\,\hat{i} + (y^2 - zx)\,\hat{j} + (z^2 - xy)\hat{k}$ is irrotational or not.
10. Prove that $\vec{f} = (y^2 - z^2 + 3yz - 2x)\,\hat{i} + (3xz + 2xy)\,\hat{j} + (3xy - 2xz + 2z)\hat{k}$ is both solenoidal and irrotational.

Lab 4_Finding the dimension, basis of a vector space and Graphical representation of linear transformation (MCS and MES only)

Obtain dimension of the vector space spanned by the vectors $[1, 2, 3]$, $[2, 3, 1]$, $[3, 1, 2]$

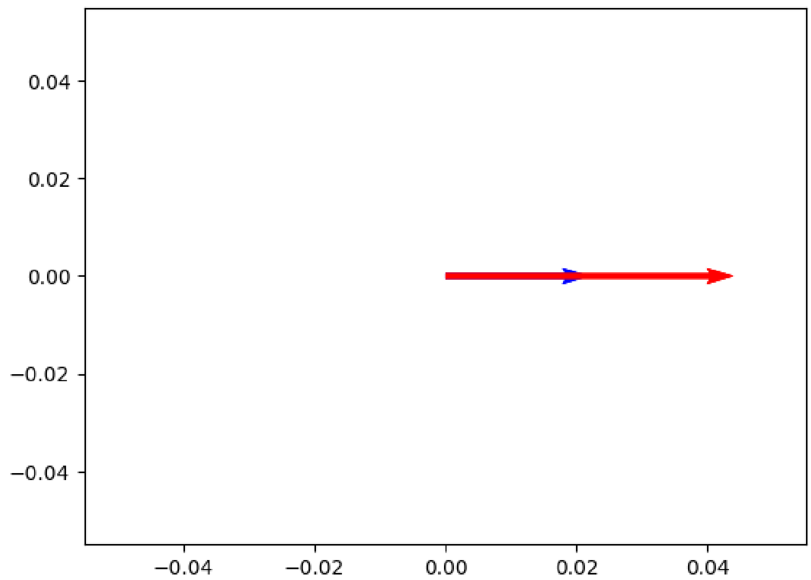
```
import numpy as np
# Define the vector space V
V = np.array([
    [1, 2, 3],
    [2, 3, 1],
    [3, 1, 2]])
# Find the dimension and basis of V
rank= np.linalg.matrix_rank(V)
print("Dimension of the vector space is",rank)
```

```
Dimension of the vector space is 3
```

Horizontal Stretch

Represent the horizontal stretch transformation $T : R^2 \to R^2$ geometrically. Find the image of vector $(10, 0)$ when it is stretched horizontally by 2 units.

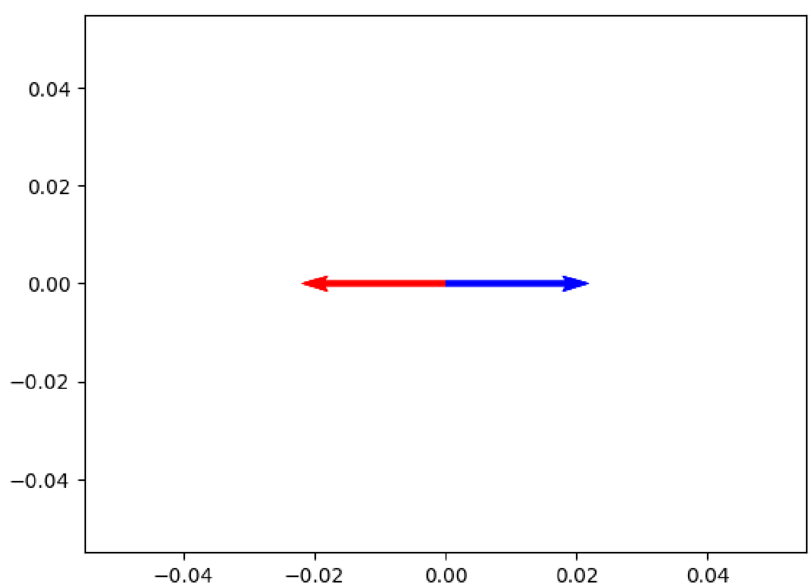
```
import numpy as np
import matplotlib . pyplot as plt
V = np. array ([[10 ,0]])
origin = np. array ([[0, 0, 0],[0, 0, 0]]) # origin point
A=np. matrix ([[2,0],[0,1]])
V1=np. matrix (V)
V2=A*np. transpose (V1)
V2=np. array (V2)
plt . quiver (*origin , V[:,0], V[:,1], color =['b'], scale =50)
plt . quiver (*origin , V2[0,:], V2[1,:], color =['r'], scale =50)
plt . show ()
```



Reflection

Represent the reflection transformation $T : R^2 \to R^2$ geometrically. Find the image of vector $(10, 0)$ when it is reflected about y-axis.

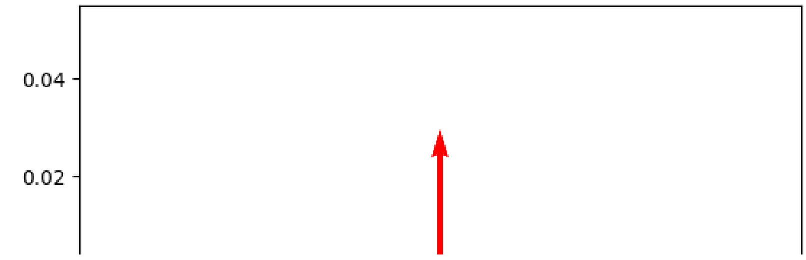
```
import numpy as np
import matplotlib . pyplot as plt
V = np. array ([[10 ,0]])
origin = np. array ([[0, 0, 0],[0, 0, 0]]) # origin point
A=np. matrix ([[-1,0],[0,1]])
V1=np. matrix (V)
V2=A*np. transpose (V1)
V2=np. array (V2)
plt . quiver (*origin , V[:,0], V[:,1], color =['b'], scale =50)
plt . quiver (*origin , V2[0,:], V2[1,:], color =['r'], scale =50)
plt . show ()
```



Rotation

Represent the rotation transformation $T : R^2 \to R^2$ geometrically. Find the image of vector $(10, 0)$ when it is rotated by $\pi/2$ radians.

```
import numpy as np
import matplotlib . pyplot as plt
V = np. array ([[10 ,0]])
origin = np. array ([[0, 0, 0],[0, 0, 0]]) # origin point
A=np. matrix ([[0,-1],[1,1]])
V1=np. matrix (V)
V2=A*np. transpose (V1)
V2=np. array (V2)
plt . quiver (*origin , V[:,0], V[:,1], color =['b'], scale =50)
plt . quiver (*origin , V2[0,:], V2[1,:], color =['r'], scale =50)
plt . show ()
```

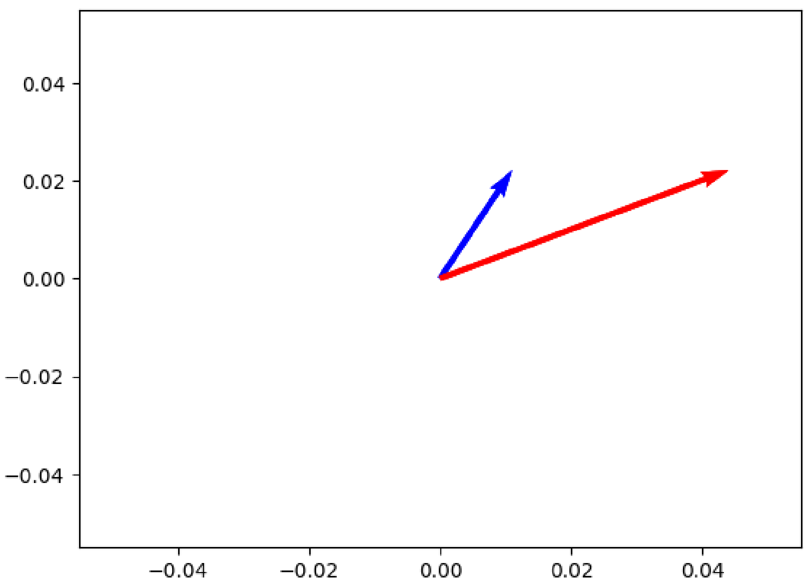


Shear Transformation

Represent the Shear transformation $T : R^2 \to R^2$ geometrically. Find the image of $(2, 3)$ under shear transformation.

```
import numpy as np
import matplotlib . pyplot as plt
V = np. array ([[2,3]])
origin = np. array ([[0, 0, 0],[0, 0, 0]]) # origin point
A=np. matrix ([[1,2],[0,1]])
V1=np. matrix (V)
V2=A*np. transpose (V1)
V2=np. array (V2)
print (" Image of given vectors is:", V2)
plt . quiver (*origin , V[:,0], V[:,1], color =['b'], scale =20)
plt . quiver (*origin , V2[0,:], V2[1,:], color =['r'], scale =20)
plt . show ()
```

```
Image of given vectors is: [[8]
[3]]
```



Activity-5

I. Find dimension of the vector space (or subspace) spanned by the following set of vectors:

1. $S = \{(1, 2), (3, 2)\}$
2. $S = \{(1, 2, 3), (0, 1, 1), (1, 3, 4)\}$
3. $S = \{(1, 0, 0), (0, 1, 0), (0, 3, -1)\}$
4. $S = \{(1, 1, 2), (0, 5, -1), (1, 16, -1)\}$
5. $S = \{(1, -1, 3, 4), (2, 1, 6, 8), (1, 1, 1, 1), (3, 2, 7, 9)\}$
6. $S = \{(2, 2, 0, 4), (1, -2, 3, 7), (1, 2, 1, -3), (4, 2, 4, 8)\}$
7. $S = \{(1, 2, 3, 4), (2, 4, 6, 8), (2, 2, 2, 2)\}$

II. Find the image of $(1, 4)$ under following $2D$ transformations:

1. Horizontal stretch
2. Reflection
3. Shear
4. Rotation

Lab 5_Verification of rank - nullity theorem (MCS and MES only)

Verify the rank-nullity theorem for the linear transformation $T : R^3 \to R^3$ defined by $T(x, y, z) = (x + 4y + 7z, 2x + 5y + 8z, 3x + 6y + 9z)$.

```
import numpy as np
from scipy.linalg import null_space

# Define a linear transformation
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Find the rank of the matrix A
rank = np.linalg.matrix_rank(A)
print("Rank of the matrix",rank)

# Find the null space of the matrix A
ns = null_space(A)
print("Null space of the matrix",ns)
# Find the dimension of the null space
nullity = ns.shape[1]
print("Null space of the matrix",nullity)
# Verify the rank-nullity theorem
if rank + nullity == A.shape[1]:
    print("Rank-nullity theorem holds.")
else:
    print("Rank-nullity theorem does not hold.")
```

```
Rank of the matrix 2
Null space of the matrix [[-0.40824829]
 [ 0.81649658]
 [-0.40824829]]
Null space of the matrix 1
Rank-nullity theorem holds.
```

Activity-6

Verify the rank-nullity theorem for the linear transformation $T : R^n \to R^m$ defined by:

1. $T(x,y) = (x+2y, 2x+y)$
2. $T(x,y) = (x-y, -x+y)$
3. $T(x,y) = (y,x)$
4. $T(x,y,z) = (x-y+2z, 2x+y-z, 3x+z)$
5. $T(x,y,z) = (x+2y+3z, 3x+2y+z, x+y+z)$
6. $T(x,y,z) = (x+y+z, x+y-z, x-y-z)$
7. $T(x,y,z) = (2x-3y+4z, 6x-9y+12z, 4x-6y+8z)$
8. $T(x_1,x_2,x_3,x_4) = (x_1+2x_2-x_3+x_4, 2x_1+x_2+x_3+3x_4)$

Lab 4_Green's Theorem (MME and MCV only)

Statement of Green's theorem in the plane:

If $P(x,y)$ and $Q(x,y)$ be two continuous functions having continuous partial derivatives in a region R of the $xy-$plane, bounded by a simple closed curve C, then

$$\oint_C (Pdx + Qdy) = \iint_R \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right) dxdy.$$

- Using Green's theorem, evaluate $\oint_R ((x+2y)dx + (x-2y)dy)$, where R is the region bounded by coordinate axes, the line $x=1$ and $y=1$.

```
from sympy import *
var ('x,y')
p=x+2*y
q=x-2*y
f= diff (q,x)- diff (p,y)
soln = integrate (f,[x,0,1],[y,0,1])
print ("I=",soln )
```

```
I= -1
```

Activit-5

Using Green's theorem, evaluate the following integrals:

1. $\oint_C ((3x+4y)dx + (2x-3y)dy)$, where C is closed curve bounded by coordinate axes, the line $x=1$ and $y=1$.
2. $\oint_C \left((xy+y^2)dx + x^2dy\right)$, where C is closed curve bounded by $y=x$ and $y=x^2$.
3. $\oint_C \left((3x-8y^2)dx + (4y-6xy)dy\right)$, where C is closed curve bounded by $x=0, y=0$ and $x+y=1$.
4. $\oint_C \left((x^2-2xy)dx + (x^2y+3)dy\right)$, where C is closed curve bounded by $y^2=8x$ and $x=2$.

Lab 5_Solution of linear partial differential equations (MME and MCV only)

Solve the PDE: $2p+3q=1$, where $p=\frac{\partial z}{\partial x}$ and $q=\frac{\partial z}{\partial y}$

```
from sympy.solvers.pde import pdsolve
from sympy import *
from sympy.abc import x,y,a
f=Function('f')
z=f(x,y)
zx=z.diff(x)
zy=z.diff(y)
eq=Eq(2*zx+3*zy,1)
display(eq)
soln=pdsolve(eq,z)
display(soln)
```

$$2\frac{\partial}{\partial x}f(x,y)+3\frac{\partial}{\partial y}f(x,y)=1$$
$$f(x,y)=\frac{2x}{13}+\frac{3y}{13}+F(3x-2y)$$

Activity-6

Solve the linear PDE's:

1. $2p+3q=1$
2. $xp+2q=x$
3. $yp+q=y$
4. $xp+yq=z$
5. $x^2p+y^2q=1$
6. $x^2p+q=z$
7. $\frac{x^2}{y}p+yq=1$
8. $x^3p+2xyq=1$

where $p=\frac{\partial z}{\partial x}$ and $q=\frac{\partial z}{\partial y}$

Lab 6_Solution of algebraic and transcendental equation by Newton-Raphson method

Find a root of the equation $3x=cosx+1$, near 1, by Newton Raphson method performing 5 iterations.

```
from sympy import *
x= Symbol ('x')
g = input ('Enter the function ') #%3x -cos(x)-1; % function
f= lambdify (x,g)
dg = diff (g);
df= lambdify (x,dg)
x0= float ( input ('Enter the intial approximation ')); # x0=1
n= int( input ('Enter the number of iterations ')); #n=5;
for i in range (1,n+1):
    x1 =(x0 - (f(x0)/df(x0)))
    print ('itration %d \t the root %0.3f \t function value %0.3f \n',(i, x1 ,f(x1))); # print all iteration value
    x0 = x1
```

```
Enter the function 3*x-cos(x)-1
Enter the intial approximation 0
Enter the number of iterations 10
itration %d      the root %0.3f          function value %0.3f
 (1, 0.6666666666666666, 0.21411273922305196)
itration %d      the root %0.3f          function value %0.3f
 (2, 0.6074928533539964, 0.0013968570539599767)
itration %d      the root %0.3f          function value %0.3f
 (3, 0.6071016657001078, 6.282984976735406e-08)
itration %d      the root %0.3f          function value %0.3f
 (4, 0.6071016481031226, -1.1102230246251565e-16)
itration %d      the root %0.3f          function value %0.3f
 (5, 0.6071016481031226, -1.1102230246251565e-16)
itration %d      the root %0.3f          function value %0.3f
 (6, 0.6071016481031226, -1.1102230246251565e-16)
itration %d      the root %0.3f          function value %0.3f
 (7, 0.6071016481031226, -1.1102230246251565e-16)
itration %d      the root %0.3f          function value %0.3f
 (8, 0.6071016481031226, -1.1102230246251565e-16)
itration %d      the root %0.3f          function value %0.3f
 (9, 0.6071016481031226, -1.1102230246251565e-16)
itration %d      the root %0.3f          function value %0.3f
 (10, 0.6071016481031226, -1.1102230246251565e-16)
```

Activity 7

Solve Algebraic and transcendental equations by Newton-Raphson method

1. $x^3 - 3x + 1$ near $x = 0.5$ Ans: 0.347 and near $x = 2$ Ans: 1.532
2. $xe^x - 2 = 0,$ near $x = 0.5$ Ans: 0.853
3. $xln(x) - 1.2 = 0$ in$(2, 3)$ correct to 4 decimal places. Ans: 2.7406
4. $3x = cos(x) + 1$ in the interval (0,1) correct to 4 decimal places Ans: 0.6071
5. $3sin(x) - 2x + 5$ near $x = 3$ Ans: 0.684
6. $xsin(x) + cos(x) = 0$ near x=π Ans: 2.7985
7. Find the fourth root of 32 using Newton-Raphson method. Ans: 2.3784

Lab 7_Interpolation/Extrapolation using Newton's forward and backward difference formula.

1. Use Newtons forward interpolation to obtain the interpolating polynomial and hence calculate $y(2)$ for the following: $x = [1, 3, 5, 7, 9]$ and $y = [6, 10, 62, 210, 502]$

```
from sympy import *
import numpy as np
n = int( input ('Enter number of data points : '))
x = np. zeros ((n))
y = np. zeros ((n,n))
# Reading data points
print ('Enter data for x and y: ')
for i in range (n):
    x[i] = float ( input ( 'x['+str(i)+']= '))
    y[i][0] = float ( input ( 'y['+str(i)+']= '))
#Generating forward difference table
for i in range (1,n):
    for j in range (0,n-i):
        y[j][i] = y[j+1][i-1] - y[j][i-1]
print ('\ nFORWARD DIFFERENCE TABLE \n');
for i in range (0,n):
    print ('%0.2f ' %(x[i]), end='')
    for j in range (0, n-i):
        print ('\t\t%0.2f ' %(y[i][j]), end='')
    print ()
# obtaining the polynomial
t= symbols ('t')
f=[] # f is a list type data

p=(t-x[0])/(x[1]-x[0])
f. append (p)
for i in range (1,n-1):
    f. append (f[i-1]*(p-i)/(i+1))
    poly =y[0][0]
for i in range (n-1):
    poly = poly +y[0][i+1]*f[i]
simp_poly = simplify ( poly )
print ('\ nTHE INTERPOLATING POLYNOMIAL IS\n');
pprint ( simp_poly )
# if you want to interpolate at some point the next session will help
inter = input ('Do you want to interpolate at a point (y/n)? ') # y
if inter =='y':
    a= float ( input ('enter the point ')) #2
    interpol = lambdify (t, simp_poly )
    result = interpol (a)
    print ('\ nThe value of the function at ' ,a,'is\n',result );
```

```
Enter number of data points : 5
Enter data for x and y:
x[0]= 1
y[0]= 6
x[1]= 3
y[1]= 10
x[2]= 5
y[2]= 62
x[3]= 7
y[3]= 210
x[4]= 9
y[4]= 502
\ nFORWARD DIFFERENCE TABLE

1.00        6.00        4.00        48.00       48.00       0.00
3.00        10.00       52.00       96.00       48.00
5.00        62.00       148.00      144.00
7.00        210.00      292.00
9.00        502.00
\ nTHE INTERPOLATING POLYNOMIAL IS

     3       2
```

```
1.0·t  - 3.0·t  + 1.0·t + 7.0
Do you want to interpolate at a point (y/n)? y
enter the point 2
\ nThe value of the function at  2.0 is
 5.0
```

2. Use Newtons backward interpolation to obtain the interpolating polynomial and hence calculate $y(8)$ for the following data: $x = [1, 3, 5, 7, 9]$ and $y = [6, 10, 62, 210, 502]$

```
from sympy import *
import numpy as np
import sys
print (" This will use Newton 's backword intepolation formula ")
# Reading number of unknowns
n = int( input ('Enter number of data points : '))
# Making numpy array of n & n x n size and initializing
# to zero for storing x and y value along with differences of y
x = np. zeros ((n))
y = np. zeros ((n,n))
# Reading data points
print ('Enter data for x and y: ')
for i in range (n):
    x[i] = float ( input ( 'x['+str(i)+']= '))
    y[i][0] = float ( input ( 'y['+str(i)+']= '))
# Generating backward difference table
for i in range (1,n):
    for j in range (n-1,i-2,-1):
        y[j][i] = y[j][i-1] - y[j-1][i-1]
print ('\ nBACKWARD DIFFERENCE TABLE \n');
for i in range (0,n):
    print ('%0.2f ' %(x[i]), end='')
    for j in range (0, i+1):
        print ('\t%0.2f ' %(y[i][j]), end='')
    print ()
#obtaining the polynomial
t= symbols ('t')
f=[]
p=(t-x[n-1])/(x[1]-x[0])
f. append (p)
for i in range (1,n-1):
      f. append (f[i-1]*(p+i)/(i+1))
poly =y[n-1][0]
print ( poly )
for i in range (n-1):
    poly = poly +y[n-1][i+1]*f[i]
    simp_poly = simplify ( poly )
print ('\ nTHE INTERPOLATING POLYNOMIAL IS\n');
pprint ( simp_poly )
# if you want to interpolate at some point the next session will help
inter = input ('Do you want to interpolate at a point (y/n)? ')
if inter =='y':
      a= float ( input ('enter the point '))
      interpol = lambdify (t, simp_poly )
      result = interpol (a)
      print ('\ nThe value of the function at ' ,a,'is\n',result );
```

```
 This will use Newton 's backword intepolation formula
Enter number of data points : 5
Enter data for x and y:
x[0]= 1
y[0]= 6
x[1]= 3
y[1]= 10
x[2]= 5
y[2]= 62
x[3]= 7
y[3]= 210
x[4]= 9
y[4]= 502
\ nBACKWARD DIFFERENCE TABLE

1.00    6.00
3.00    10.00   4.00
5.00    62.00   52.00   48.00
7.00    210.00  148.00  96.00   48.00
9.00    502.00  292.00  144.00  48.00   0.00
502.0
\ nTHE INTERPOLATING POLYNOMIAL IS

     3        2
1.0·t  - 3.0·t  + 1.0·t + 7.0
Do you want to interpolate at a point (y/n)? y
enter the point 8
\ nThe value of the function at  8.0 is
 335.0
```

Activity 7

1. Using Newton's forward formula, compute the pressure of the steam at temperature 1420 from the following steam table. Temperature: [140,150,160,170,180] , Pressure : [3.685,4.854, 6.302, 8.076,10.225]
2. Fit a polynomial of degree three which takes the following values. $x = [3, 4, 5, 6]$ and $y = [6, 24, 60, 120]$ Ans:$x^3 - 3x^2 + 2x$

3. The area of a circle (A) corresponding to diameter (D) is given below: $D = [80, 85, 90, 95, 100]$ and $A = [5026, 5674, 6362, 7088, 7854]$. Find the area corresponding to diameter 105 using an appropriate interpolation formula.
4. Given f(40) = 184, f(50) = 204, f(60) = 226, f(70) = 250, f(80) = 276, f(90) = 304, find f(38) and f(85) using suitable interpolation formulae.
5. Given $sin45^\circ = 0.7071, sin500^\circ = 0.7660, sin550^\circ = 0.8192, sin600^\circ = 0.8660$ find $sin570^\circ$ using an appropriate interpolation formula.
6. Find $f(2.5)$ by using Newton's backward interpolation formula given that $f(0) = 7.4720, f(1) = 7.5854, f(2) = 7.6922, f(3) = 7.8119, f(4) = 7.9252$
7. From the following data estimate the number of students scoring marks more than 40 but less than 45. Marks: [30-40,40-50,50-60,60-70,70-80] , No. of students: [31,42, 51, 35,31]

Lab 8_Computation of area under the curve using Trapezoidal Rule, Simpson's (1/3)rd rule, Simpson's (3/8)th rule and Weddle's rule.

Trapezoidal Rule

```
import numpy as np

a = 0
b = np.pi
n = 11
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

I_trap = (h/2)*(f[0] + 2*sum(f[1:n-1]) + f[n-1])

print(I_trap)
```

```
1.9835235375094546
```

Simpsons $1/3^{rd}$ Rule

```
import numpy as np

a = 0
b = np.pi
n = 25
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

I_trap =(h/3) * (f[0] + 2*sum(f[:n-2:2])+ 4*sum(f[1:n-1:2]) + f[n-1])

print(I_trap)
```

```
2.00000326887716
```

Simpsons $3/8^{th}$ Rule

```
import numpy as np

a = 0
b = np.pi
n = 20
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

I_trap =(3*h/8) * (f[0] + 3*sum(f[1:n-1]) - sum(f[1:n-1:3]) + f[n-1])

print(I_trap)
```

```
1.9965856909340378
```

Activity 8

1. Use Simpson's 1/3rd rule to find $\int_0^{0.6} e^{-x^2} dx$. Ans: 0.5351
2. By using Simpson's 3/8th rule with h= 0.2 find the approximate area under the curve $y = \frac{x^2-1}{x^2+1}$ between the ordinates $x = 1$ and $x = 2.8$, Ans: 0.9152
3. Evaluate $I = \int_4^{5.2} log(x)dx$ a) Simpson's 1/3rd rule, Ans: a) 1.8278472.., b) Simpson's 3/8th rule Ans: b) 1.8278470
4. Evaluate $\int_0^1 \frac{1}{x^2+1} dx$.by using Simpson's 1/3rd rule taking four equal strips and hence deduce an approximate value of π. Ans 0.7854

5. Evaluate $\int_0^1 \frac{1}{x+1}dx$ taking seven ordinates by applying Simpson's 3/8th rule. Hence deduce the value of $ln(2)$ Ans: 0.6932

6. Use Simpson's 1/3rd & 3/8th rule to evaluate $\int_1^4 e^{1/x}dx$ Ans 4.9257

Lab 9_Solution of ODE of first order and first degree by Taylor's series and Modified Euler's method.

9.1 Taylor series method to solve ODE

Solve: $\frac{dy}{dx} - 2y = 3e^x$ with$y(0) = 0$ using Taylor series method x = 0.1, 0.2, 0.3.

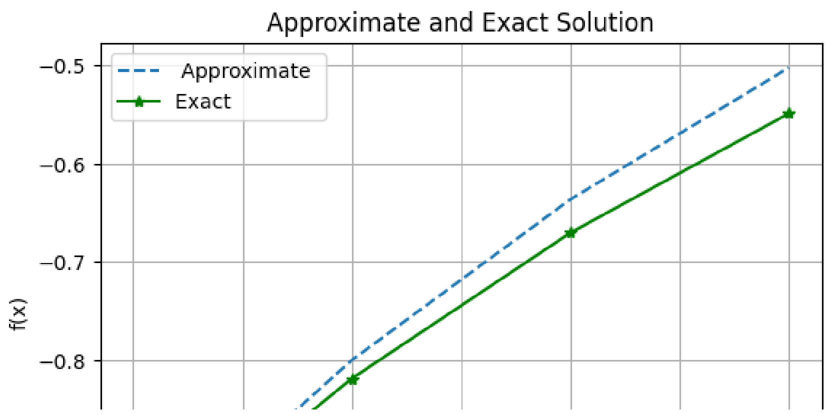
```
from numpy import *
def taylor (deriv ,x,y,xStop ,h):
    X = []
    Y = []
    X. append (x)
    Y. append (y)
    while x < xStop : # Loop over integration steps
        D = deriv (x,y) # Derivatives of y
        H = 1.0
        for j in range (3): # Build Taylor series
            H = H*h/(j + 1)
            y = y + D[j]*H # H = h^j/j!
            x = x + h
        X. append (x) # Append results to
        Y. append (y) # lists X and Y
    return array (X),array (Y) # Convert lists into arrays
 # deriv = user - supplied function that returns derivatives in the 4 x n array
def deriv (x,y):
      D = zeros ((4,1))
      D[0] = [2*y[0] + 3*exp(x)]
      D[1] = [4*y[0]+ 9*exp(x)]
      D[2] = [8*y[0]+ 21*exp(x)]
      D[3] = [16*y[0]+ 45*exp(x)]
      return D
x = 0.0 # Initial value of x
xStop = 0.3 # last value
y = array ([0.0]) # Initial values of y
h = 0.1 # Step size
X,Y = taylor (deriv ,x,y,xStop ,h)
print("The required values are :at x= %0.2f , y=%0.5f , x=%0.2f , y=%0.5f , x = %0.2f , y=%0.5f ,x = %0.2f , y=%0.5f" %(X[0],Y[0],X[1],Y[1],X[2
```

9.2 Euler's method to solve ODE:

Solve: $\frac{dy}{dx} = e^{-x}$ with$y(0) = -1$ using Taylor series method x = 0.2, 0.4, 0.6.

```
import numpy as np
import matplotlib . pyplot as plt
# Define parameters
f = lambda x, y: np.exp(-x) # ODE
h = 0.2 # Step size
y0 = -1 # Initial Condition
n=3
# Explicit Euler Method
y=np.zeros(n+1)
x=np.zeros(n++1)
y[0] = y0
x[0]=0
for i in range (0, n):
    x[i+1]=x[i]+h
    y[i + 1] = y[i] + h*f(x[i], y[i])
print ("The required values are at x= %0.2f , y=%0.5f , x=%0.2f , y=%0.5f ,x = %0.2f , y=%0.5f ,x = %0.2f , y=%0.5f"%(x[0],y[0],x[1],y[1],x[2],
print ("\n\n")
plt . plot (x, y, '--', label =' Approximate ')
plt . plot (x, -np.exp (-x), 'g*-', label ='Exact ')
plt . title (" Approximate and Exact Solution " )
plt . xlabel ('x')
plt . ylabel ('f(x)')
plt . grid ()
plt . legend ()
plt . show ()
```

```
The required values are at x= 0.00 , y=-1.00000 , x=0.20 , y=-0.80000 ,x = 0.40 , y=-0.63625 ,x = 0.60 , y=-0.50219
```



Activity 9

1. Solve the following differential equations using Euler's modified method:

a. $\frac{dy}{dx} = \log x + y$ with $y(1) = 2$ at $x = 1.2$ and $x = 1.4$. Take $h = 0.2$

b. $y^{'} = x + \sin y$ with $y(0) = 1$. Compute $y(0.2)$ and $y(0.4)$

c. $\frac{dy}{dx} = \frac{y-x}{y+x}$ with the boundary conditions $y(0) = 1$. Compute y at $x = 0.1$

2. Solve the following differential equations using Taylor's series method:

a. $y^{'} = x - y^2$ with $y(0) = 1$. Compute $y(0.1)$

b. $y^{'} = x^2 y - 1$ with $y(0) = 1, h = 0.1$. Compute $y(0.2)$

c. $y^{'} = x + y$ with $y(0) = 1, h = 0.1$. Compute $y(0.1), y(0.2), y(0.3)$

Modified Euler's Method

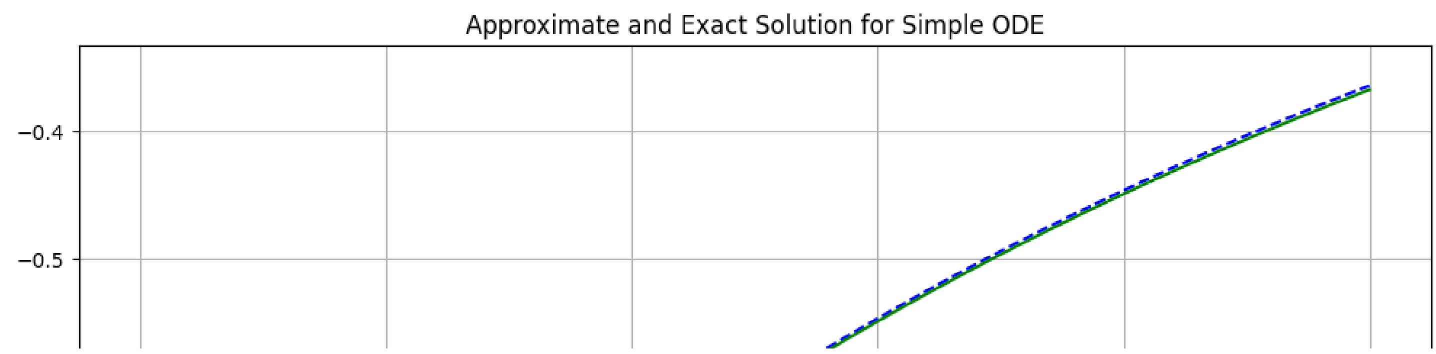
```
import numpy as np
import matplotlib.pyplot as plt

# Define parameters
f = lambda t, s: np.exp(-t) # ODE
h = 0.01 # Step size
t = np.arange(0, 1 + h, h) # Numerical grid
s0 = -1 # Initial Condition

# Explicit Euler Method
s = np.zeros(len(t))
s[0] = s0

for i in range(0, len(t) - 1):
    s[i + 1] = s[i] + h*f(t[i], s[i])

plt.figure(figsize = (12, 8))
plt.plot(t, s, 'b--', label='Approximate')
plt.plot(t, -np.exp(-t), 'g', label='Exact')
plt.title('Approximate and Exact Solution for Simple ODE')
plt.xlabel('t')
plt.ylabel('f(t)')
plt.grid()
plt.legend(loc='lower right')
plt.show()
```



Lab 10_Solution of ODE of first order and first degree by Runge-Kutta 4th order and Milne's predictor-corrector method

Runge-Kutta 4^{th} order method

Apply the Runge Kutta method to find the solution of $\frac{dy}{dx} = 1 + \frac{y}{x}$ at $y(2)$, taking $h = 0.2$. Given that $y(1) = 2$.

```
from sympy import *
import numpy as np
def RungeKutta (g,x0 ,h,y0 ,xn):
  x,y= symbols('x,y')
  f= lambdify ([x,y],g)
  xt=x0+h
  Y=[y0]
  while xt<=xn:
    k1=h*f(x0 ,y0)
    k2=h*f(x0+h/2, y0+k1/2)
    k3=h*f(x0+h/2, y0+k2/2)
    k4=h*f(x0+h, y0+k3)
    y1=y0+(1/6)*(k1+2*k2+2*k3+k4)
    Y. append (y1)
    # print ('y(%3.3f '%xt ,') is %3.3f '%y1)
    x0=xt
    y0=y1
    xt=xt+h
  return np. round (Y,2)
RungeKutta ('1+(y/x)',1,0.2,2,2)
```

```
array([2.  , 2.62, 3.27, 3.95, 4.66, 5.39])
```

Milne's predictor and corrector method

Apply Milne's predictor and corrector method to solve $dy/dx = x^2 + (y/2)$ at $y(1.4)$. Given that $y(1) = 2$, $y(1.1) = 2.2156$, $y(1.2) = 2.4649$, $y(1.3) = 2.7514$. Use corrector formula thrice.

```
# Milne 's method to solve first order DE
# Use corrector formula thrice
x0=1
y0=2
y1=2.2156
y2=2.4649
y3=2.7514
h=0.1
x1=x0+h
x2=x1+h
x3=x2+h
x4=x3+h
def f(x,y):
  return x ** 2+(y/2)
y10 =f(x0 , y0)
y11 =f(x1 ,y1)
y12 =f(x2 ,y2)
y13 =f(x3 ,y3)
y4p =y0+(4*h/3)*(2*y11-y12+2*y13)
print ('predicted value of y4 is %3.3f '%y4p)
y14 =f(x4 ,y4p );
for i in range (1,4):
  y4=y2+(h/3)*(y14 +4*y13 +y12 );
  print ('corrected value of y4 after',i,'iterations is',y4)
  y14=f(x4 ,y4);
```

```
predicted value of y4 is 3.079
corrected value of y4 after 1 iterations is 3.0793962222222224
corrected value of y4 after 2 iterations is 3.079398270370371
corrected value of y4 after 3 iterations is 3.079398304506173
```

Activity-10

1. Apply Runge-Kutta fourth order method, to find an approximate value of y when $x = 0.2$ given that $\frac{dy}{dx} = x + y$ and $y = 1$ when $x = 0$.
2. Using Runge-Kutta method of fourth order, solve $\frac{dy}{dx} = \frac{y^2-x^2}{y^2+x^2}$ with $y(0) = 1$ at $x = 0.2, 0.4$.
3. Using Runge-Kutta method of fourth order, find $y(0.2)$ given that $\frac{dy}{dx} = 3x + \frac{y}{2}$ with $y(0) = 1$ taking $h = 0.1$
4. Apply Runge-Kutta method to find an approximate value of y for $x = 0.2$ in steps of 0.1 if $\frac{dy}{dx} = x + y^2$ given that $y = 1$, where $x = 0$
5. Use Runge-kutta method of order 4, find y for $x = 0.1, 0.2, 0.3$ given that $\frac{dy}{dx} = xy + y^2$
6. Using Milne's method, find $y(4.5)$ given $5xy' + y^2 - 2 = 0$ given
   $y(4) = 1, y(4.1) = 1.0049, y(4.2) = 1.00097, y(4.3) = 1.0143, y(4.4) = 1.0187$