**Sahakar Maharshi Bhausaheb Santuji Thorat College Sangamner**

**Remark Demonstrator’s Signature**

**Date:- / /20**

**DEPARTMENT OF COMPUTER SCIENCE**

**Sub : Mathematics**

**Name:-\_Gorde Yash Somnath Roll.No:-\_21 Date:- Title of the expt:- Slip no 13 Page.no:- Class:- BCS**

**Q1. Attempt any two of the following**

**A ) Write a python program to plot 2D graph of the function f(x)=x2 and g(x)=x3 in [-1,1]**

**-**

import matplotlib.pyplot as plt import numpy as np

# Define the x values

x = np.linspace(-1, 1, 100)

# Define the functions

f = x\*\*2

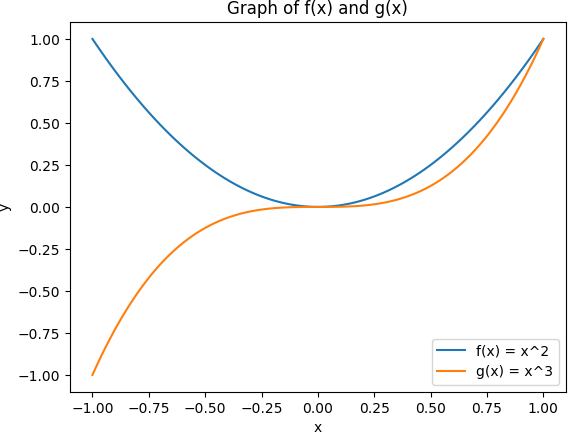
g = x\*\*3

plt.plot(x, f, label='f(x) = x^2') plt.plot(x, g, label='g(x) = x^3') plt.xlabel('x')

plt.ylabel('y')

plt.title('Graph of f(x) and g(x)') plt.legend()

plt.show()



**B ) Using Python plot the surface plot of parabola z=x2+y2 in -6 <x,y<6**

**-**

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D x = np.linspace(-6, 6, 100)

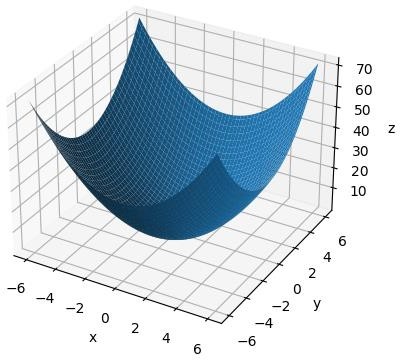
y = np.linspace(-6, 6, 100) X, Y = np.meshgrid(x, y)

Z = X\*\*2 + Y\*\*2

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d') ax.plot\_surface(X, Y, Z)

ax.set\_xlabel('x') ax.set\_ylabel('y') ax.set\_zlabel('z') plt.show()



# C ) Write a python program to plot the 3D line graph whose parametric equation is (cos(2x),sin(2x),x) for 10 ≤x≤20 (in red color) with title to the graph

**-**

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D x = np.linspace(10, 20, 1000)

y = np.cos(2\*x)

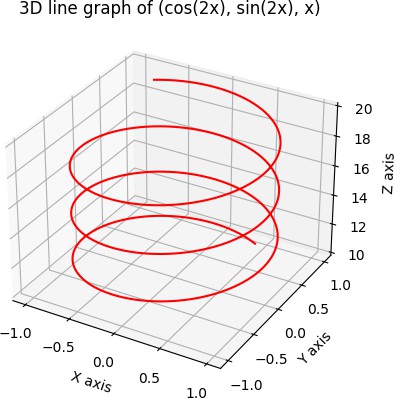
z = np.sin(2\*x) w = x

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d') ax.plot(y, z, w, color='red') ax.set\_xlabel('X axis')

ax.set\_ylabel('Y axis') ax.set\_zlabel('Z axis')

plt.title('3D line graph of (cos(2x), sin(2x), x)') plt.show()



# Q2 ) Attempt any TWO of the following

1. **Write a python program to reflect the ∆ABC through the line y=3 where A(1,0),B(2,-1),C(- 1,3)**

**-**

import numpy as np

import matplotlib.pyplot as plt

A = np.array([1, 0])

B = np.array([2, -1])

C = np.array([-1, 3])

x = np.linspace(-5, 5, 100) y = np.full\_like(x, 3)

plt.plot(A[0], A[1], 'o', label='A')

plt.plot(B[0], B[1], 'o', label='B')

plt.plot(C[0], C[1], 'o', label='C') plt.plot(x, y, label='y=3')

plt.axis('equal') plt.grid() plt.legend()

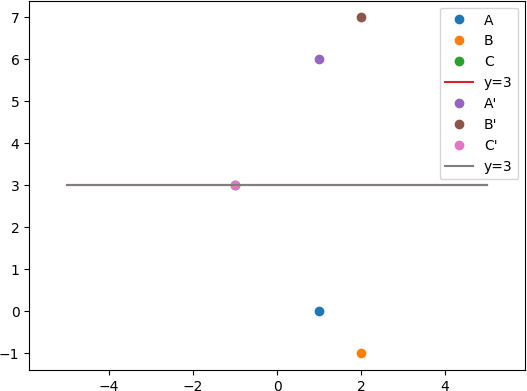
# Reflect the coordinates of the triangle across the line y=3 A\_reflect = np.array([A[0], 2\*3 - A[1]])

B\_reflect = np.array([B[0], 2\*3 - B[1]])

C\_reflect = np.array([C[0], 2\*3 - C[1]])

plt.plot(A\_reflect[0], A\_reflect[1], 'o', label='A\'') plt.plot(B\_reflect[0], B\_reflect[1], 'o', label='B\'') plt.plot(C\_reflect[0], C\_reflect[1], 'o', label='C\'') plt.plot(x, y, label='y=3')

plt.axis('equal') plt.grid() plt.legend() plt.show()



# Find the area and perimeter of the ∆ABC where A[0,0],B[5,0],C[3,3]

**-**

import numpy as np

A = np.array([0, 0])

B = np.array([5, 0])

C = np.array([3, 3])

dist\_AB = np.linalg.norm(B - A)

dist\_BC = np.linalg.norm(C - B) dist\_CA = np.linalg.norm(A - C)

perimeter = dist\_AB + dist\_BC + dist\_CA

area = 0.5 \* abs((A[0]\*B[1] + B[0]\*C[1] + C[0]\*A[1]) - (A[1]\*B[0] + B[1]\*C[0] + C[1]\*A[0]))

print("Perimeter =", perimeter) print("Area =", area)

Output :

Perimeter = 12.848191962583275

Area = 7.5

1. **Using sympy declare the points P(5,2),Q(5,-2),R(5,0) ,check whether these points are colliner.Declare the ray passing through the points P and Q , find the length of this ray between P and Q, Also find slop of this ray**

**-**

from sympy import Point, Segment, Line

P = Point(5, 2)

Q = Point(5, -2)

R = Point(5, 0)

area = abs((Q.x - P.x)\*(R.y - P.y) - (R.x - P.x)\*(Q.y - P.y))/2 if area < 1e-6:

print("The points are collinear") else:

print("The points are not collinear")

ray\_PQ = Segment(P, Q).direction length\_PQ = Segment(P, Q).length slope\_PQ = Line(P, Q).slope

print("Length of ray PQ:", length\_PQ) print("Slope of ray PQ:", slope\_PQ)

output :

The points are collinear Length of ray PQ: 4 Slope of ray PQ: oo

**Q3 ) Attempt the following**

**A ) Attempt any ONE of the following**

**I ) Write a python program to solve the following LPP : Max Z=5x+3y**

**Subject to X+Y≤7**

**2x+5y≤1 X,y≥0**

**-**

import numpy as np

from scipy.optimize import linprog c = np.array([5, 3])

A = np.array([[1, 1], [2, 5]])

b = np.array([7, 1]) x\_bounds = (0, None) y\_bounds = (0, None)

result = linprog(c, A\_ub=A, b\_ub=b, bounds=[x\_bounds, y\_bounds], method='simplex')

print("Status:", result.message) print("x =", result.x[0])

print("y =", result.x[1])

print("Optimal value of Z =", result.fun)

output :

Status: Optimization terminated successfully. x = 0.0

y = 0.0

Optimal value of Z = 0.0

**II ) write a python program to display the following LPP by using pulp module and simplex method.Find its optimal solution if exist**

**Max Z=3x+2y+5z Subject to x+2y+z≤430**

**3x+4z≤460 X+4y≤120**

**X,y,z≥0**

**-**

from pulp import \*

prob = LpProblem("LPP", LpMaximize) x = LpVariable("x", lowBound=0)

y = LpVariable("y", lowBound=0)

z = LpVariable("z", lowBound=0)

prob += 3\*x + 2\*y + 5\*z prob += x + 2\*y + z <= 430 prob += 3\*x + 4\*z <= 460 prob += x + 4\*y <= 120 prob.solve()

print("Status:", LpStatus[prob.status]) print("x =", value(x))

print("y =", value(y))

print("z =", value(z))

print("Optimal value of Z =", value(prob.objective))

**B ) Attempt any ONE of the following**

**I ) Write a python program to apply the following transformation on the point (-2,4)**

**:**

**A ) Shering in Y direction by 7 units**

**B ) Scaling in X and Y direction by** 𝟕 **and 7 units respectively**

𝟐

**C ) Shering in X and Y direction by 4 and 7 units respectively**

**D ) Rotation about origin by an angle 60⸰**

**-**

import numpy as np

p = np.array([-2, 4, 1]).T

T\_A = np.array([[1, 0, 0], [7, 1, 0], [0, 0, 1]])

T\_B = np.array([[7/2, 0, 0], [0, 7, 0], [0, 0, 1]])

T\_C = np.array([[1, 4, 0], [7, 1, 0], [0, 0, 1]])

theta = np.radians(60)

T\_D = np.array([[np.cos(theta), -np.sin(theta), 0], [np.sin(theta), np.cos(theta), 0], [0, 0, 1]])

p\_A = np.dot(T\_A, p) p\_B = np.dot(T\_B, p) p\_C = np.dot(T\_C, p) p\_D = np.dot(T\_D, p)

print("A) Shearing in Y direction by 7 units:", p\_A)

print("B) Scaling in X and Y direction by 7/2 and 7 units respectively:", p\_B) print("C) Shearing in X and Y direction by 4 and 7 units respectively:", p\_C) print("D) Rotation about origin by an angle 60⸰:", p\_D)

Output :

1. Shearing in Y direction by 7 units: [ -2 -10 1]
2. Scaling in X and Y direction by 7/2 and 7 units respectively: [-7. 28. 1.]
3. Shearing in X and Y direction by 4 and 7 units respectively: [ 14 -10 1]
4. Rotation about origin by an angle 60⸰: [-4.46410162 0.26794919 1. ]

**II ) Write a python program to plot 2D x-axis and y-axis in black colour.in the same diagram plot**

**A ) Green tringle with vertices [5,4],[7,4],[6,6]**

**B )Blue rectangle with vertices[2,2],[10,2],[10,8],[2,8]**

**C )Red polygon with vertices[6,2],[10,4],[8,7],[4,8],[2,4]**

**D )Isosceles triangle with vertices [0,0],[4,0],[2,4]**

**-**

import matplotlib.pyplot as plt plt.axhline(y=0, color='black') plt.axvline(x=0, color='black') plt.plot([5, 7, 6], [4, 4, 6], 'g')

plt.plot([2, 10, 10, 2, 2], [2, 2, 8, 8, 2], 'b')

plt.plot([6, 10, 8, 4, 2], [2, 4, 7, 8, 4], 'r')

plt.plot([0, 4, 2, 0], [0, 0, 4, 0], 'm')

plt.xlim(-2, 12)

plt.ylim(-2, 12)

plt.title('Shapes') plt.xlabel('x-axis') plt.ylabel('y-axis') plt.show()

