**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 25 Page.no:- Class:- BCS**

**Q1 Attempt any TWO of the following**

**A ) Write a python program to generate 3D plot of the function Z=cos x+ cos y in -10 <x ,y<10**

**-**

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D def f(x, y):

return np.cos(x) + np.cos(y)

x = np.linspace(-10, 10, 100)

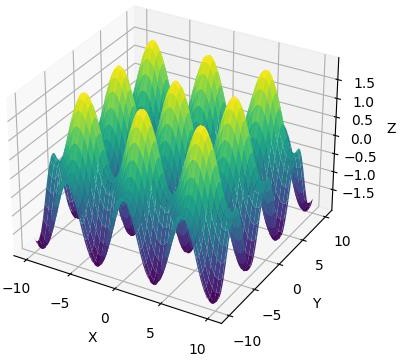
y = np.linspace(-10, 10, 100) X, Y = np.meshgrid(x, y)

Z = f(X, Y)

fig = plt.figure()

ax = fig.gca(projection='3d') ax.plot\_surface(X, Y, Z, cmap='viridis') ax.set\_xlabel('X')

ax.set\_ylabel('Y') ax.set\_zlabel('Z') plt.show()



**B ) Using python plot the graph of function f(x)=sin-1(x) on the interval [-1,1]**

**-**

import numpy as np

import matplotlib.pyplot as plt def f(x):

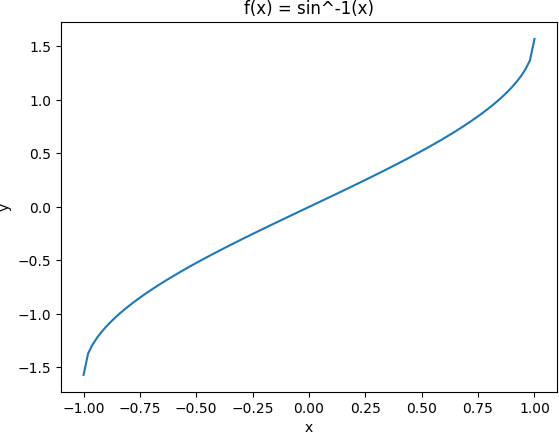
return np.arcsin(x)

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

plt.plot(x, y)

plt.title("f(x) = sin^-1(x)") plt.xlabel("x")

plt.ylabel("y") plt.show()



**C )Using python plot the surface plot of the function z=cos(x2+y2-0.5) in the interval form -1<x,y<1**

**-**

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D def f(x, y):

return np.cos(x\*\*2 + y\*\*2 - 0.5)

x = np.linspace(-1, 1, 50)

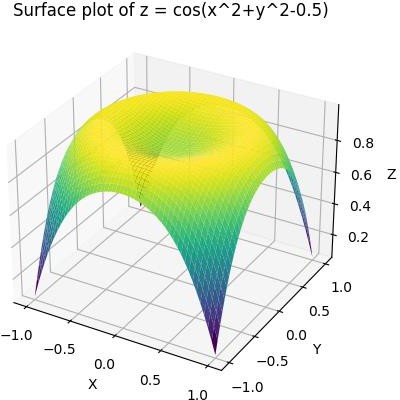
y = np.linspace(-1, 1, 50) X, Y = np.meshgrid(x, y) Z = f(X, Y)

fig = plt.figure()

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

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

plt.title('Surface plot of z = cos(x^2+y^2-0.5)') plt.show()



**Q2 ) Attempt any TWO of the following**

**A ) Rotate the line segment by 180⸰ having end points (1,0) and (2,-1)**

**-**

import numpy as np

import matplotlib.pyplot as plt x = np.array([1, 2])

y = np.array([0, -1])

T = np.array([[-1, 0],

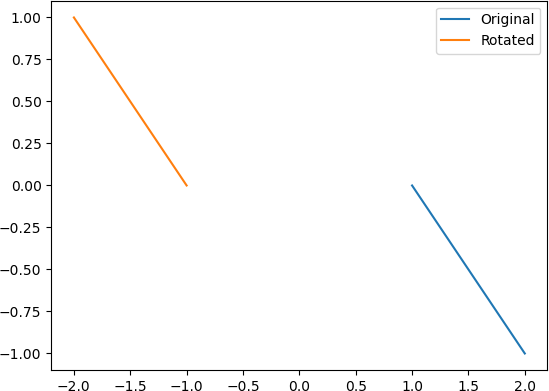
[0, -1]])

new\_coords = np.dot(T, np.vstack([x, y]))

plt.plot(x, y, label='Original')

plt.plot(new\_coords[0], new\_coords[1], label='Rotated') plt.legend()

plt.show()



**B ) Using sympy,declare the points P(5,2),Q(5,-2),R(5,0) chevk 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, Ray P = Point(5, 2)

Q = Point(5, -2)

R = Point(5, 0)

if P.is\_collinear(Q, R):

print("Points P, Q, and R are collinear") else:

print("Points P, Q, and R are not collinear") ray\_PQ = Ray(P, Q)

length\_PQ = P.distance(Q) print("Length of ray PQ:", length\_PQ)

slope\_PQ = ray\_PQ.slope print("Slope of ray PQ:", slope\_PQ)

output :

Points P, Q, and R are collinear Length of ray PQ: 4

Slope of ray PQ: oo

**C )Generate triangle with vertices (0,0),(4,0),(1,4) check whether the triangle is Scalene triangle**

**-**

import matplotlib.pyplot as plt

A = [0, 0]

B = [4, 0]

C = [1, 4]

plt.plot([A[0], B[0]], [A[1], B[1]], 'r')

plt.plot([B[0], C[0]], [B[1], C[1]], 'r')

plt.plot([C[0], A[0]], [C[1], A[1]], 'r')

AB = ((B[0]-A[0])\*\*2 + (B[1]-A[1])\*\*2)\*\*0.5

BC = ((C[0]-B[0])\*\*2 + (C[1]-B[1])\*\*2)\*\*0.5

AC = ((C[0]-A[0])\*\*2 + (C[1]-A[1])\*\*2)\*\*0.5

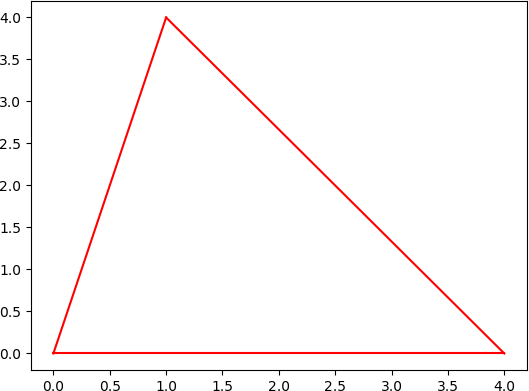
if AB != BC and AB != AC and BC != AC:

print("The triangle is a scalene triangle.") else:

print("The triangle is not a scalene triangle.")

plt.show()

The triangle is a scalene triangle



**Q3 ) Attempt the following**

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

**I ) Write a python program to solve the following LPP : Max Z=150x+75y**

**Subject to 4x+6y≤24**

**5x+3y≤15 X,y≥0**

**-**

from scipy.optimize import linprog obj = [-150, -75]

lhs = [[4, 6], [5, 3]]

rhs = [24, 15]

x\_bounds = (0, None) y\_bounds = (0, None)

result = linprog(c=obj, A\_ub=lhs, b\_ub=rhs, bounds=[x\_bounds, y\_bounds])

print('Optimal value:', round(result.fun, 2)) print('x:', round(result.x[0], 2))

print('y:', round(result.x[1], 2))

output :

Optimal value: -450.0

x: 3.0

y: 0.0

**II )Write a Python program to solve the following LPP : Max Z=4x+y+3z+5w**

**Subject to 4x+6y-5z-4w≥20**

**-3x-2y+4z+w≤10**

**-8x-3y+3z+2w≤20 X,y,z,w≥0**

**-**

from scipy.optimize import linprog obj\_func = [-4, -1, -3, -5]

lhs\_ineq = [ [4, 6, -5, -4],

[-3, -2, 4, 1],

[-8, -3, 3, 2]

]

rhs\_ineq = [20, -10, -20]

bounds = [(0, None), (0, None), (0, None), (0, None)]

res = linprog(c=obj\_func, A\_ub=lhs\_ineq, b\_ub=rhs\_ineq, bounds=bounds, method='simplex')

print("Optimal value of Z:", -res.fun) # since we used the negative of obj\_func print("Values of x, y, z, w:", res.x)

output :

Optimal value of Z: 18.333333333333332

Values of x, y, z, w: [1.66666667 3.33333333 0. 1.66666667]

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

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

**A ) reflection through X-axis**

**B ) Scaling in X-coordinate by factor 6**

**C ) Shering in x direction by 4 units**

**D ) Rotate about origin through an angle 30⸰**

**-**

import numpy as np

import matplotlib.pyplot as plt from math import sin, cos, radians

# Initial point

P = np.array([-2, 4])

T1 = np.array([[1, 0], [0, -1]])

T2 = np.array([[6, 0], [0, 1]])

T3 = np.array([[1, 0], [4, 1]])

theta = radians(30)

T4 = np.array([[cos(theta), -sin(theta)], [sin(theta), cos(theta)]])

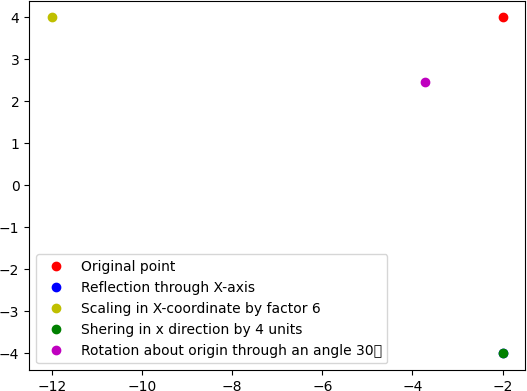
P1 = np.dot(T1, P) P2 = np.dot(T2, P) P3 = np.dot(T3, P) P4 = np.dot(T4, P)

plt.plot(P[0], P[1], 'ro', label='Original point')

plt.plot(P1[0], P1[1], 'bo', label='Reflection through X-axis') plt.plot(P2[0], P2[1], 'yo', label='Scaling in X-coordinate by factor 6') plt.plot(P3[0], P3[1], 'go', label='Shering in x direction by 4 units')

plt.plot(P4[0], P4[1], 'mo', label='Rotation about origin through an angle 30⸰')

plt.legend() plt.show()



**II ) Write a python program to find the combined transformation of the line segment between the points A[3,2] & B 2,-3] for the following sequence of transformation**

**A ) Rotation about origin through an angle** 𝝅

𝟔

**B ) Scaling in Y-coordinate by -4 units**

**C ) Uniform scaling by -6.4 units**

**D ) Shering in Y direction by 5 units**

**-**

import numpy as np

A = np.array([3, 2])

B = np.array([2, -3])

def rotation(theta):

return np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])

def scaling\_x(k):

return np.array([[k, 0],

[0, 1]])

def scaling\_y(k):

return np.array([[1, 0],

[0, k]])

def uniform\_scaling(k):

return np.array([[k, 0],

[0, k]])

def shering\_y(d):

return np.array([[1, d],

[0, 1]])

AB = B - A

AB = AB.reshape((2,1)) T1 = rotation(np.pi/6) T2 = scaling\_y(-4)

T3 = uniform\_scaling(-6.4) T4 = shering\_y(5)

T = T4 @ T3 @ T2 @ T1

AB\_transformed = T @ AB

B\_transformed = A + AB\_transformed.flatten()

print("The combined transformation of the line segment between A and B is:") print(T)

print("The transformed endpoint B' is:", B\_transformed)

output :

The combined transformation of the line segment between A and B is: [[ 58.45743742 114.05125168]

[ 12.8 22.17025034]]

The transformed endpoint B' is: [-625.71369584 -121.65125168]