**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 21 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)=x4 in [0,5] with red dashed line with circle markers**

**-**

import numpy as np

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

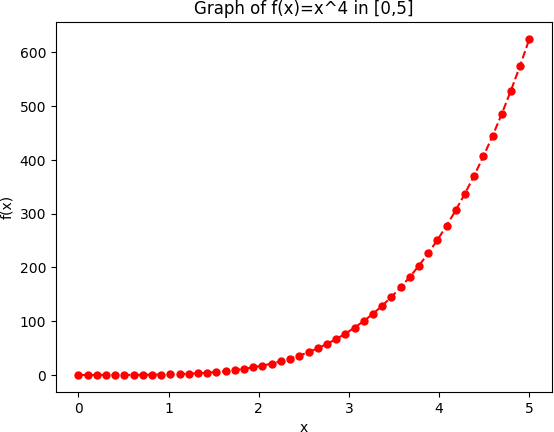
return x\*\*4

x = np.linspace(0, 5) y = f(x)

plt.plot(x, y, 'o-', color='red', linestyle='dashed', markersize=5) plt.xlabel('x')

plt.ylabel('f(x)')

plt.title('Graph of f(x)=x^4 in [0,5]') plt.show()



**B ) Write a python program to generate 3D plot of the function z=x2+y2 in -6<x,y<6**

**-**

import matplotlib.pyplot as plt import numpy as np

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

return x\*\*2 + y\*\*2

x = np.linspace(-6, 6, 100)

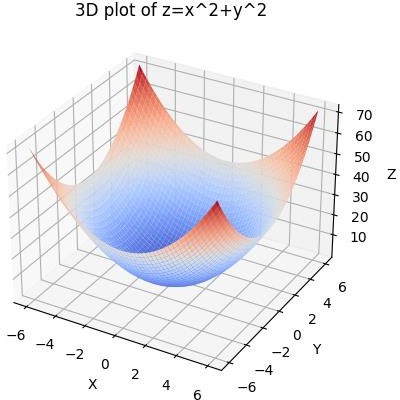
y = np.linspace(-6, 6, 100) X, Y = np.meshgrid(x, y) Z = z\_func(X, Y)

fig = plt.figure()

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

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

ax.set\_title('3D plot of z=x^2+y^2') plt.show()



**C ) Write a python program to plot 3D graph of the function f(x)=ex2+y2 for x,y E[0,2π] using wireframe**

**-**

import numpy as np

import matplotlib.pyplot as plt

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

return np.exp(x\*\*2 + y\*\*2)

x = np.linspace(0, 2\*np.pi, 100) y = np.linspace(0, 2\*np.pi, 100) X, Y = np.meshgrid(x, y)

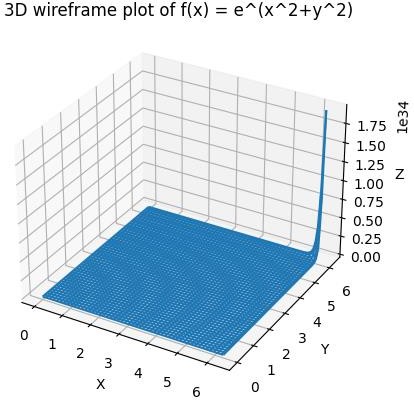
Z = f(X, Y)

fig = plt.figure()

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

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

ax.set\_title('3D wireframe plot of f(x) = e^(x^2+y^2)') plt.show()



**Q 2 ) Attempt any TWO of the following**

**A ) if the line segment joining the points A[2,5],B[4,-13] is transformed to the line segment A’B’ by the transformation matrix [T]=**𝟐 𝟑 **, then using python find the**

𝟒 𝟏

**slop and midpoint of the transformed line**

**-**

import numpy as np

T = np.array([[2, 3], [4, 1]])

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

B = np.array([[4], [-13]])

A\_prime = T @ A B\_prime = T @ B

slope = (B\_prime[1, 0] - A\_prime[1, 0]) / (B\_prime[0, 0] - A\_prime[0, 0]) midpoint = (A\_prime + B\_prime) / 2

print("Slope of the transformed line segment A'B':", slope) print("Midpoint of the transformed line segment A'B':", midpoint)

output :

Slope of the transformed line segment A'B': 0.2 Midpoint of the transformed line segment A'B': [[-6.] [ 8.]]

B **)write a python program to plot the square with vertices at [4,4],[2,4],[2,2],[4,2] and find its uniform expansion by factor 3, uniform reduction by factor 0.4**

**-**

import matplotlib.pyplot as plt import numpy as np

fig, ax = plt.subplots()

square = np.array([[4,4], [2,4], [2,2], [4,2], [4,4]])

ax.plot(square[:,0], square[:,1], color='blue')

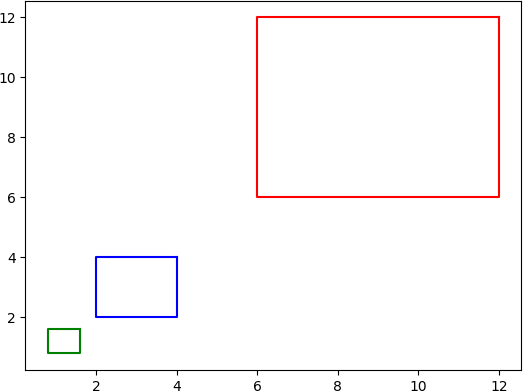
expansion\_matrix = np.array([[3,0], [0,3]]) expanded\_square = np.dot(square, expansion\_matrix)

ax.plot(expanded\_square[:,0], expanded\_square[:,1], color='red')

reduction\_matrix = np.array([[0.4,0], [0,0.4]]) reduced\_square = np.dot(square, reduction\_matrix)

ax.plot(reduced\_square[:,0], reduced\_square[:,1], color='green')

plt.show()



**C ) Write a python program to find the quation of the transformed line if shering is applied on the line 2x+y=3 in x and y direction by 2 and -3 units respectively**

**-**

from scipy.linalg import inv a, b, c = 2, 1, -3

T = [[1, 2], [-3, 1]]

T\_inv = inv(T)

coeffs = [a\*T\_inv[0][0] + b\*T\_inv[1][0], a\*T\_inv[0][1] + b\*T\_inv[1][1],

c]

print(f"The equation of the transformed line is {coeffs[0]:.0f}x + {coeffs[1]:.0f}y +

{coeffs[2]:.0f} = 0")

output :

The equation of the transformed line is 1x + -0y + -3 = 0

**Q 3 ) Attempt the following**

**I ) Write a python program to solve the following LPP : Min Z=4x+2y**

**Subject to x+y≤3**

**x-y≥2 x,y≥0**

**-**

from scipy.optimize import linprog obj\_func\_coeffs = [4, 2]

lhs\_ineq\_coeffs = [[1, 1], [-1, 1]]

rhs\_ineq\_values = [3, 2]

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

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

print('Optimal value:', round(result.fun, 2))

print('Optimal point:', (round(result.x[0], 2), round(result.x[1], 2)))

output :

Optimal value: 0.0

Optimal point: (0.0, 0.0)

I**I ) Write a python program to solve the following LPP : Max Z=2x+4y**

**Subject to 2x+y≤18**

**2x+2y≥30 X+2y=26**

**X,y≥0**

**-**

from scipy.optimize import linprog obj = [-2, -4]

lhs\_ineq = [[2, 1], [-2, -2]]

rhs\_ineq = [18, -30]

lhs\_eq = [[1, 2]]

rhs\_eq = [26]

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

result = linprog(c=obj, A\_ub=lhs\_ineq, b\_ub=rhs\_ineq, A\_eq=lhs\_eq, b\_eq=rhs\_eq, bounds=bounds, method="simplex")

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

print("y =", round(result.x[1], 2))

output :

Optimal value: -52.0

x = 3.33

y = 11.33

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

**I ) Apply the following transformation on the point P[-2,4]**

**A ) Reflection through line 3x+4y=5**

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

**C ) Scaling in Y-coordinate by factor 4.1**

**D ) Reflection through lne y=2x+3**

**-**

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

A = np.array([[7/25, 24/25], [24/25, -7/25]])

P\_A = np.dot(A, P)

print("A) Point after reflection through line 3x + 4y = 5:", P\_A)

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

P\_B = np.dot(B, P)

print("B) Point after scaling in X-coordinate by factor 6:", P\_B)

C = np.array([[1, 0], [0, 4.1]])

P\_C = np.dot(C, P)

print("C) Point after scaling in Y-coordinate by factor 4.1:", P\_C)

D = np.array([[4/5, -2/5], [-2/5, -4/5]]) P\_D = np.dot(D, P)

print("D) Point after reflection through line y = 2x + 3:", P\_D)

output :

1. Point after reflection through line 3x + 4y = 5: [ 3.28 -3.04]
2. Point after scaling in X-coordinate by factor 6: [-12 4]
3. Point after scaling in Y-coordinate by factor 4.1: [-2. 16.4]
4. Point after reflection through line y = 2x + 3: [-3.2 -2.4]

**II ) Apply the following transformation on the point P[-2,4]**

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

**B ) Scaling in both X and Y direction by 4 and 7 units respectively**

**C ) Rotation about origin by an angle 48 degree**

**D ) Reflection through line y=x**

**-**

import numpy as np import math

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

A = np.array([[1, 0], [7, 1]])

P = np.dot(A, P)

B = np.array([[4, 0], [0, 7]])

P = np.dot(B, P)

theta = math.radians(48)

C = np.array([[math.cos(theta), -math.sin(theta)], [math.sin(theta), math.cos(theta)]]) P = np.dot(C, P)

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

P = np.dot(D, P)

print("Final point after all transformations:", P)

output :

Final point after all transformations: [-52.78430105 46.66709293]