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

**Q1 Attempt any TWO of the following**

**A ) Write a python program to plot 3D graph of the function f(x)=e-x2 in [-5,5] with green dashed points line with upward pointing triangle**

**-**

import numpy as np

import matplotlib.pyplot as plt

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

return np.exp(-x\*\*2)

x = np.linspace(-5, 5, 100)

y = np.linspace(-5, 5, 100) X, Y = np.meshgrid(x, y) Z = f(X, Y)

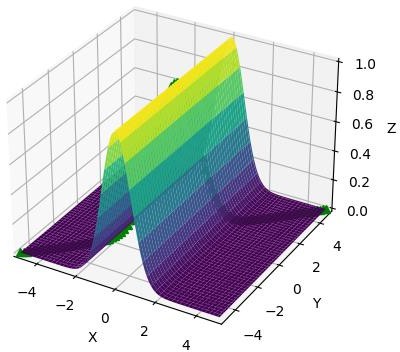
fig = plt.figure()

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

ax.plot(x, y, f(x, y), 'g--', marker='^') ax.set\_xlim([-5, 5])

ax.set\_ylim([-5, 5])

ax.set\_zlim([0, 1]) ax.set\_xlabel('X') ax.set\_ylabel('Y') ax.set\_zlabel('Z') plt.show()



**B ) Write a python program to plot the graph of the function using def() F(x)={x2+4 if -10≤x<5 , 3x+9 if 5≤x<10 }**

**-**

import matplotlib.pyplot as plt import numpy as np

def F(x):

if -10 <= x < 5:

return x\*\*2 + 4 elif 5 <= x < 10:

return 3\*x + 9

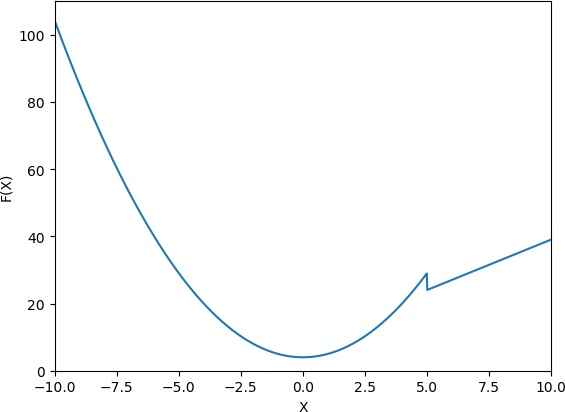
x = np.linspace(-10, 10, 1000) y = [F(i) for i in x]

plt.plot(x, y) plt.xlim([-10, 10])

plt.ylim([0, 110])

plt.xlabel('X')

plt.ylabel('F(X)') plt.show()



**C ) Write a python program to plot graph of the function f(x)=log(3x2), in[1,10] with black dashed points**

**-**

import numpy as np

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

return np.log(3\*x\*\*2)

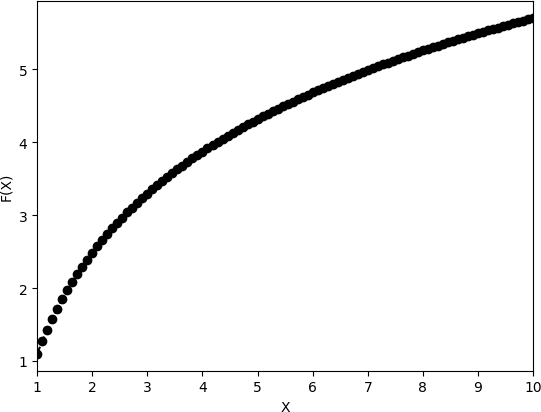
x = np.linspace(1, 10, 100)

y = f(x)

plt.plot(x, y, 'k--o') plt.xlim([1, 10])

plt.xlabel('X')

plt.ylabel('F(X)') plt.show()



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

**A )Write a python program to plot triangle with vertices[3,3],[5,6],[5,2] and its rotation about the origin by angle –π radians**

**-**

import numpy as np

import matplotlib.pyplot as plt

vertices = np.array([[3,3], [5,6], [5,2]]) theta = -np.pi

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

[np.sin(theta), np.cos(theta)]]) rotated\_vertices = np.dot(vertices, R)

plt.plot(vertices[:,0], vertices[:,1], 'b', label='Original triangle') plt.plot(rotated\_vertices[:,0], rotated\_vertices[:,1], 'r', label='Rotated triangle')

plt.xlim([-7, 7])

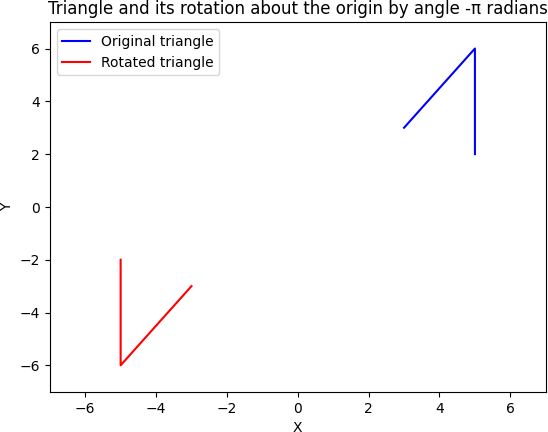
plt.ylim([-7, 7])

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Triangle and its rotation about the origin by angle -π radians') plt.legend()

plt.show()



**B ) Write a python program to generate vector x in the interval [-22,22] using numpy package with 80 subintervals**

**-**

import numpy as np num\_subintervals = 80

interval = [-22, 22]

subinterval\_size = (interval[1] - interval[0]) / num\_subintervals

x = np.arange(interval[0], interval[1] + subinterval\_size, subinterval\_size) print(x)

output :

[-2.20000000e+01 -2.14500000e+01 -2.09000000e+01 -2.03500000e+01

-1.98000000e+01 -1.92500000e+01 -1.87000000e+01 -1.81500000e+01

-1.76000000e+01 -1.70500000e+01 -1.65000000e+01 -1.59500000e+01

-1.54000000e+01 -1.48500000e+01 -1.43000000e+01 -1.37500000e+01

-1.32000000e+01 -1.26500000e+01 -1.21000000e+01 -1.15500000e+01

-1.10000000e+01 -1.04500000e+01 -9.90000000e+00 -9.35000000e+00

-8.80000000e+00 -8.25000000e+00 -7.70000000e+00 -7.15000000e+00

-6.60000000e+00 -6.05000000e+00 -5.50000000e+00 -4.95000000e+00

-4.40000000e+00 -3.85000000e+00 -3.30000000e+00 -2.75000000e+00

-2.20000000e+00 -1.65000000e+00 -1.10000000e+00 -5.50000000e-01 2.84217094e-14 5.50000000e-01 1.10000000e+00 1.65000000e+00

2.20000000e+00 2.75000000e+00 3.30000000e+00 3.85000000e+00

4.40000000e+00 4.95000000e+00 5.50000000e+00 6.05000000e+00

6.60000000e+00 7.15000000e+00 7.70000000e+00 8.25000000e+00

8.80000000e+00 9.35000000e+00 9.90000000e+00 1.04500000e+01

1.10000000e+01 1.15500000e+01 1.21000000e+01 1.26500000e+01

1.32000000e+01 1.37500000e+01 1.43000000e+01 1.48500000e+01

1.54000000e+01 1.59500000e+01 1.65000000e+01 1.70500000e+01

1.76000000e+01 1.81500000e+01 1.87000000e+01 1.92500000e+01

1.98000000e+01 2.03500000e+01 2.09000000e+01 2.14500000e+01

2.20000000e+01]

**C )Write a python program to draw a polygon with vertices (0,0),(1,0),(2,2),(1,4) also find area and perimeter of the polygon**

**-**

import matplotlib.pyplot as plt

vertices = [(0,0), (1,0), (2,2), (1,4)]

polygon = plt.Polygon(vertices, closed=True, fill=None, edgecolor='black')

fig, ax = plt.subplots() ax.add\_patch(polygon) ax.set\_xlim(-1, 3)

ax.set\_ylim(-1, 5)

perimeter = 0

for i in range(len(vertices)-1):

perimeter += ((vertices[i][0]-vertices[i+1][0])\*\*2 + (vertices[i][1]- vertices[i+1][1])\*\*2)\*\*0.5

perimeter += ((vertices[-1][0]-vertices[0][0])\*\*2 + (vertices[-1][1]- vertices[0][1])\*\*2)\*\*0.5

area = 0

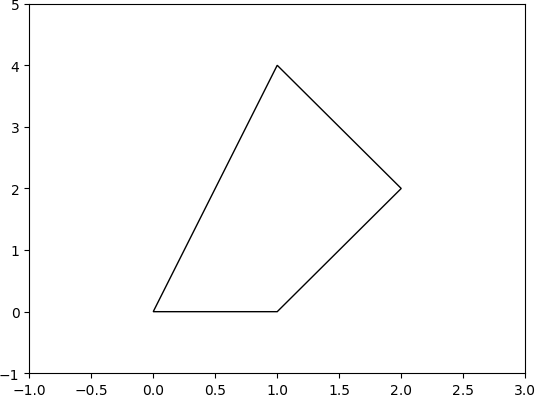
for i in range(len(vertices)-1):

area += vertices[i][0]\*vertices[i+1][1] - vertices[i+1][0]\*vertices[i][1] area += vertices[-1][0]\*vertices[0][1] - vertices[0][0]\*vertices[-1][1] area = abs(area) / 2

print('Perimeter:', perimeter) print('Area:', area) plt.show()

Perimeter: 9.595241580617241

Area: 4.0



**Q3 ) Attempt the following**

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

**I ) Write a python program to solve the following LPP : Min Z=3.5x+2y**

**Subject to x+y≥5**

**x≥4 y≥2 x,y≥0**

**-**

import numpy as np

from scipy.optimize import linprog c = np.array([3.5, 2])

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

[1, 0],

[0, 1]])

b = np.array([5, 4, 2])

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

res = linprog(c, A\_ub=A, b\_ub=b, bounds=bounds, method='simplex')

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

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

print("Z =", res.fun)

output :

Status: Optimization terminated successfully. x = 0.0

y = 0.0

Z = 0.0

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

**Subject to x≥6**

**y≥6 x+y≤11**

**x,y≥0**

**-**

import numpy as np

from scipy.optimize import linprog c = np.array([1, 1])

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

[0, -1],

[1, 1]])

b = np.array([-6, -6, 11])

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

res = linprog(c, A\_ub=A, b\_ub=b, bounds=bounds, method='simplex')

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

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

print("Z =", res.fun)

output : x = 6.0

y = 6.0

Z = 12.0

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

**I ) Apply python program in each of the following transformation on the points P[3,-1]**

**A ) reflection through X-axis**

**B ) Scaling in X-coordinate by factor 2**

**C ) Scaling in Y-coordinate by factor 1.5**

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

**-**

import numpy as np P = np.array([3, -1])

P\_reflected = np.array([P[0], -P[1]]) print("Reflection through X-axis:", P\_reflected)

P\_scaled\_x = np.array([2\*P[0], P[1]])

print("Scaling in X-coordinate by factor 2:", P\_scaled\_x)

P\_scaled\_y = np.array([P[0], 1.5\*P[1]])

print("Scaling in Y-coordinate by factor 1.5:", P\_scaled\_y)

P\_reflected\_line = np.array([P[1], P[0]])

print("Reflection through the line y=x:", P\_reflected\_line)

output :

Reflection through X-axis: [3 1]

Scaling in X-coordinate by factor 2: [ 6 -1]

Scaling in Y-coordinate by factor 1.5: [ 3. -1.5] Reflection through the line y=x: [-1 3]

**II )Find the combined transformation of the line segment between the points A[4,- 1] & B[3,0] by using python program for the following sequence of transformation**

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

**B ) Shering in Y-direction by 4.5 units**

**C ) Scaling in X-coordinate by 3 units**

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

**-**

import numpy as np A = np.array([4, -1])

B = np.array([3, 0])

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

A\_rotated = np.dot(R, A) B\_rotated = np.dot(R, B)

S = np.array([[1, 0], [4.5, 1]])

A\_sheared = np.dot(S, A\_rotated) B\_sheared = np.dot(S, B\_rotated)

Sx = np.array([[3, 0], [0, 1]]) A\_scaled = np.dot(Sx, A\_sheared) B\_scaled = np.dot(Sx, B\_sheared) R\_line = np.array([[0, 1], [1, 0]])

A\_reflected = np.dot(R\_line, A\_scaled) B\_reflected = np.dot(R\_line, B\_scaled)

print("Transformed point A:", A\_reflected) print("Transformed point B:", B\_reflected)

output :

Transformed point A: [-17. -12.]

Transformed point B: [-13.5 -9. ]