Sahakar Maharshi Bhausaheb Santuji Thorat

College Sangamner

DEPARTMENT OF COMPUTER SCIENCE

Sub : Mathematics

| Remark |
| --- |
| Demonstrator's Signature |
| Date:- / /20 |

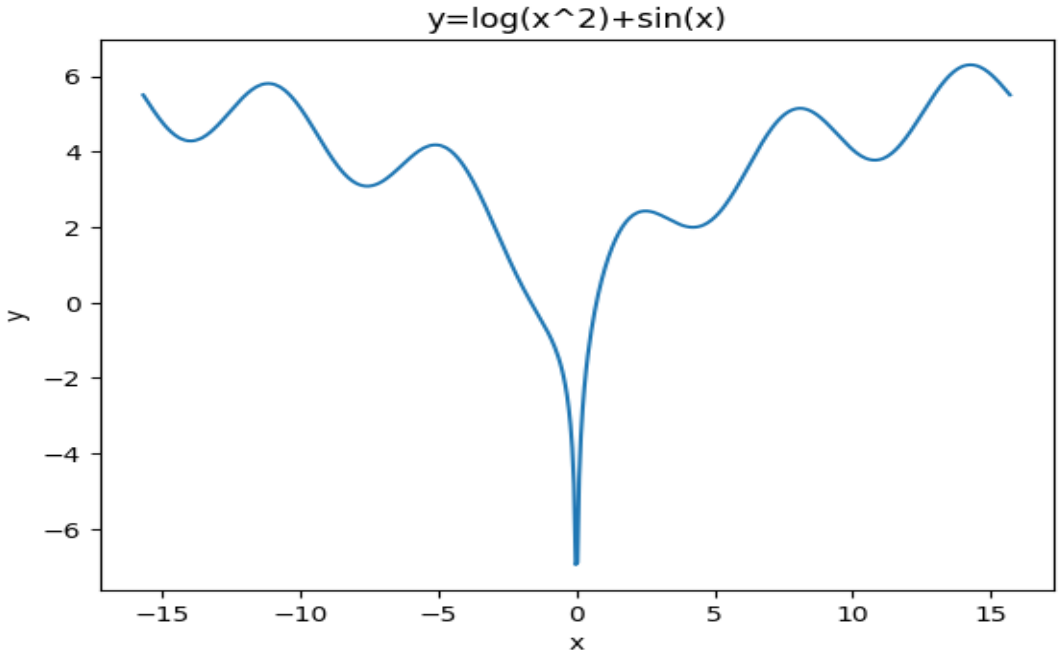
Name:-_Gorde Yash Somnath_____ Roll.No:-__21_Date:-_____
Title of the expt:-__Slip no 22_____Page.no:-_____Class:-____BCS_____

## Q1. Attempt any TWO of the following

**A ) Write a python program to draw 2D plot y=log($x^2$)+sin(x) with suitable label in the x axis , y axis and a title in [-5π,5π]**

-→

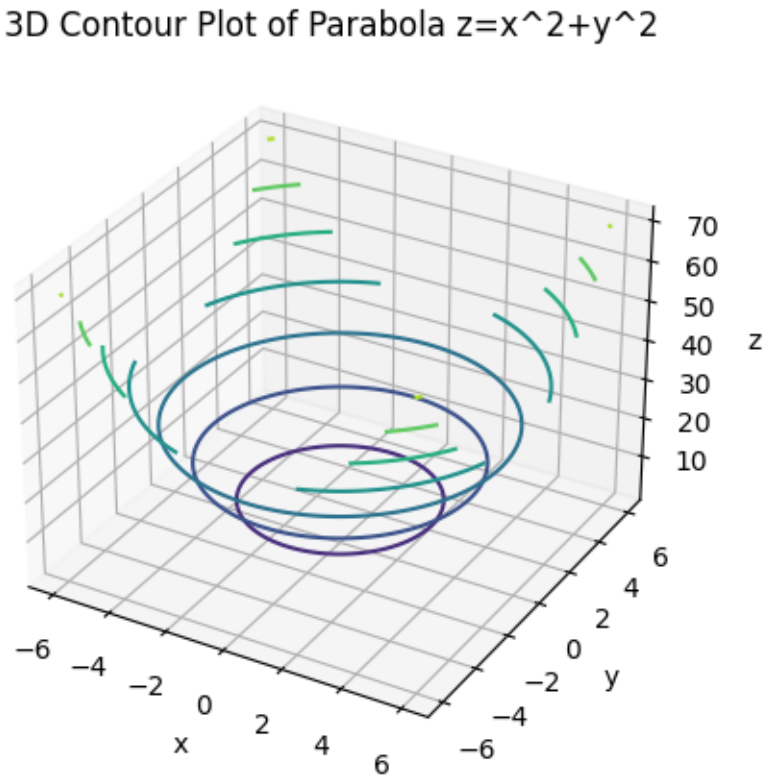
```
import numpy as np
import matplotlib.pyplot as plt
def f(x):
    return np.log(x**2) + np.sin(x)

x = np.linspace(-5*np.pi, 5*np.pi, 500)
y = f(x)

plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('y=log(x^2)+sin(x)')
plt.show()
```



**B ) Write a python program to plot 3D dimrentional contour plot of parabola $z=x^2+y^2$ -6<x,y<6**

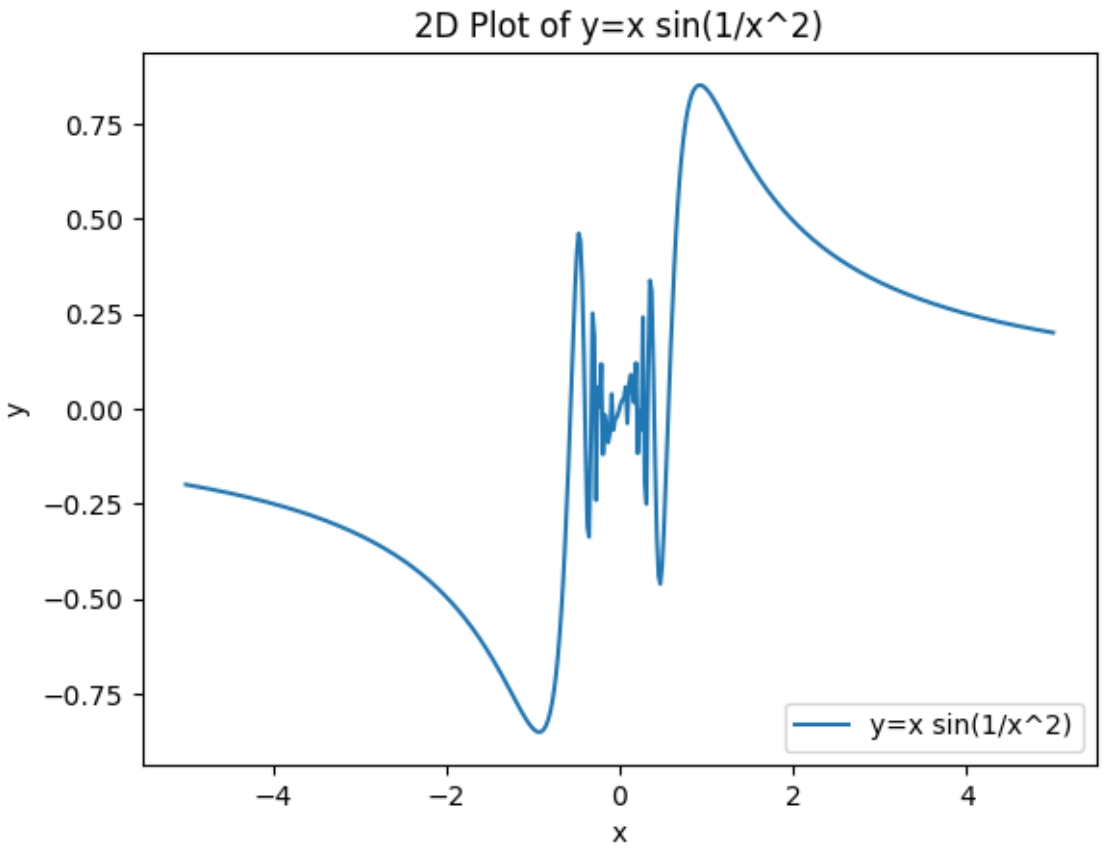
**-→**

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
def f(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 = f(X, Y)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.contour(X, Y, Z)

ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
plt.title('3D Contour Plot of Parabola z=x^2+y^2')
plt.show()
```



**C ) Write a python program to draw 2D plot y=x sin($\frac{1}{x2}$) in [-5,5] with suitable label in the x axis , y axis , a title and location of legend to lower right corner -→**

```
import numpy as np
import matplotlib.pyplot as plt
def f(x):
    return x * np.sin(1/x**2)

x = np.linspace(-5, 5, 500)
y = f(x)

plt.plot(x, y, label='y=x sin(1/x^2)')
plt.xlabel('x')
plt.ylabel('y')
plt.title('2D Plot of y=x sin(1/x^2)')
plt.legend(loc='lower right')
plt.show()
```



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

**A ) Write a python program to find the angle at each vertices of the triangle ABC where A[0,0],B[2,2], and C[0,2]**
**-→**

```
import numpy as np
A = np.array([0, 0])
B = np.array([2, 2])
C = np.array([0, 2])

a = np.linalg.norm(B - C)  # side BC
b = np.linalg.norm(A - C)  # side AC
c = np.linalg.norm(A - B)  # side AB

alpha = np.arccos((b**2 + c**2 - a**2) / (2 * b * c)) * 180 / np.pi
beta = np.arccos((a**2 + c**2 - b**2) / (2 * a * c)) * 180 / np.pi
gamma = np.arccos((a**2 + b**2 - c**2) / (2 * a * b)) * 180 / np.pi

print(f"The angles at vertices A, B, and C are {alpha:.2f}°, {beta:.2f}°, and
{gamma:.2f}°, respectively.")
```

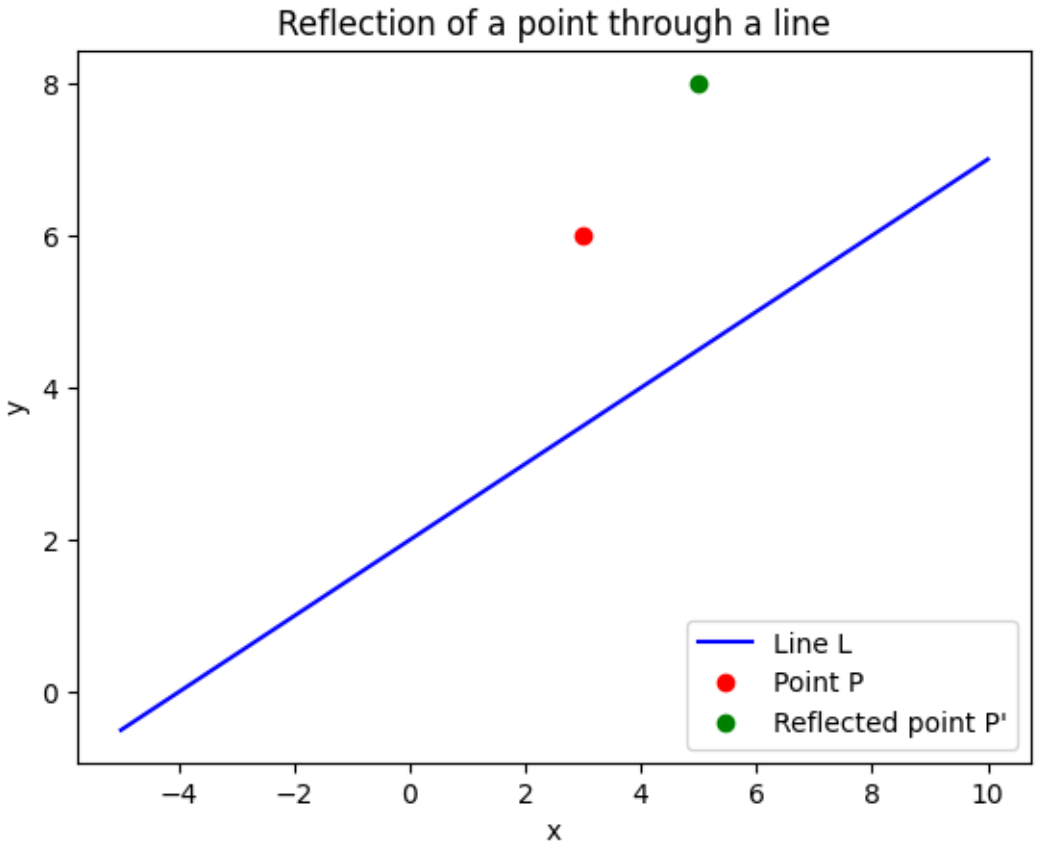
output :

The angles at vertices A, B, and C are 45.00°, 45.00°, and 90.00°, respectively.

**B ) Write a python program to reflect the point P[3,6] through the line x-2y+4=0**
**-→**

```
import matplotlib.pyplot as plt
import numpy as np
P = [3, 6]
a, b, c = 1, -2, 4

x = np.linspace(-5, 10, 100)
y = (a * x + c) / -b
x_prime = P[0] - 2 * (a * P[0] + b * P[1] + c) / (a**2 + b**2)
y_prime = P[1] - 2 * (a * P[0] + b * P[1] + c) / (a**2 + b**2)

fig, ax = plt.subplots()
ax.plot(x, y, 'b-', label='Line L')
ax.plot(P[0], P[1], 'ro', label='Point P')
ax.plot(x_prime, y_prime, 'go', label="Reflected point P'")
ax.set_xlabel('x')
ax.set_ylabel('y')
```

```
ax.set_title('Reflection of a point through a line')
ax.legend(loc='lower right')
plt.show()
```




**C ) Write a python program to find area and perimeter of the triangle ABC where A[0,0],B[5,0],C[3,3]**

**-→**

```
import math
A = [0, 0]
B = [5, 0]
C = [3, 3]

AB = math.dist(A, B)
BC = math.dist(B, C)
CA = math.dist(C, A)

s = (AB + BC + CA) / 2
area = math.sqrt(s * (s - AB) * (s - BC) * (s - CA))
perimeter = AB + BC + CA
```

```
print(f"The area of the triangle ABC is {area:.2f} square units")
print(f"The perimeter of the triangle ABC is {perimeter:.2f} units")
```

output :

```
The area of the triangle ABC is 7.50 square units
The perimeter of the triangle ABC is 12.85 units
```

**Q3 ) Attempt the following**

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

**I ) 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
c = [-4, -1, -3, -5]
A = [[-4, -6, 5, 4],
    [3, 2, -4, -1],
    [8, 3, -3, -2]]

b = [-20, 10, 20]
x_bounds = (0, None)
y_bounds = (0, None)
z_bounds = (0, None)
w_bounds = (0, None)
res = linprog(c=c, A_ub=A, b_ub=b, bounds=[x_bounds, y_bounds, z_bounds,
w_bounds], method='simplex')

print("Status:", res.message)
print(f"Optimal values: x={res.x[0]:.2f}, y={res.x[1]:.2f}, z={res.x[2]:.2f},
w={res.x[3]:.2f}")
print(f"Optimal objective function value: {res.fun:.2f}")
```

output :

Status: Optimization failed. The problem appears to be unbounded.
Optimal values: x=1.67, y=3.33, z=0.00, w=1.67
Optimal objective function value: -18.33

**II )Write a python program to solve the following LPP :**

**Min Z=x+y**

**Subject to x+y≤11**

**x≥6**

**y≥6**

**x,y≥0**

**-→**

```
from scipy.optimize import linprog
c = [1, 1]
A = [[1, 1],
    [-1, 0],
    [0, -1]]

b = [11, -6, -6]
x_bounds = (0, None)
y_bounds = (0, None)
res = linprog(c=c, A_ub=A, b_ub=b, bounds=[x_bounds, y_bounds], method='simplex')

print("Status:", res.message)
print(f"Optimal values: x={res.x[0]:.2f}, y={res.x[1]:.2f}")
print(f"Optimal objective function value: {res.fun:.2f}")
```

output :

Optimal values: x=6.00, y=6.00
Optimal objective function value: 12.00

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

**I ) Write a python program for each of the following**

**A ) Rotate the point (1,1) about (1,4) through angle π/2**

**B ) Find Distance between two points (0,0) and (1,0)**

**C ) Find the shering of the point (3,4) in X direction by 3 units**

**D ) Represent two dimensional points using point function (-2,5)**

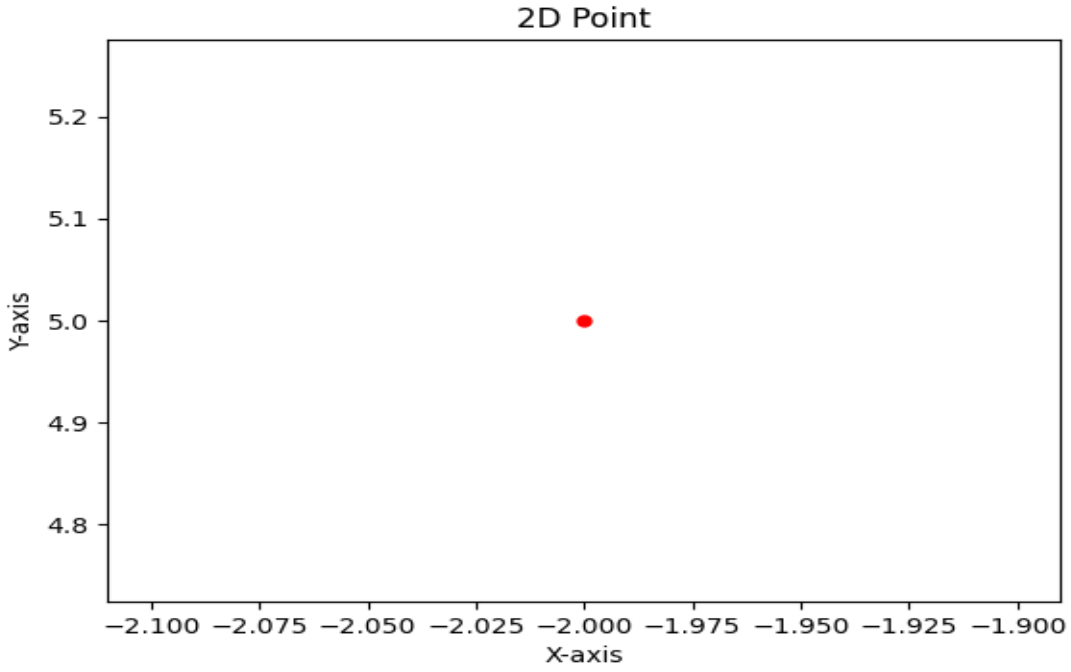
-→

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import linprog

theta = np.pi/2
P = np.array([[1], [1]])
T = np.array([[1, 0], [0, 1]])
R = np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])
Q = np.array([[1], [4]])
P_new = T.dot(Q) + R.dot(P - Q)
print("Rotated point:", P_new.T)

p1 = np.array([0, 0])
p2 = np.array([1, 0])
dist = np.linalg.norm(p2-p1)
print("Distance between two points:", dist)

P = np.array([[3], [4]])
T = np.array([[1, 0], [0, 1]])
S = np.array([[1, 0], [0, 0]])
P_new = T.dot(S.dot(P))
print("Sheared point:", P_new.T)

x, y = -2, 5
plt.plot(x, y, marker='o', markersize=5, color="red")
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('2D Point')
plt.show()

c = [-4, -1, -3, -5]
A = [[4, 6, -5, -4], [-3, -2, 4, 1], [-8, -3, 3, 2]]
b = [-20, 10, 20]
bounds = [(0, None), (0, None), (0, None), (0, None)]
res = linprog(c=c, A_ub=A, b_ub=b, bounds=bounds, method='simplex')
print("LPP Solution:")
print("Optimal value:", round(res.fun, 2))
print("Optimal point:", [round(x, 2) for x in res.x])
```

output :

```
Rotated point: [[4. 4.]]
Distance between two points: 1.0
```

Sheared point: [[3 0]]




LPP Solution:
Optimal value: -810.0
Optimal point: [0.0, 100.0, 20.0, 130.0]

**II ) A company has 3 production facility S1,S2 and S3 with production capacity of 7,9 and 18 units (in 100's ) per week of a product respectively.These units are to be shpped to 4 warchouses D1,D2,D3 and D4 with requirement of 5,6,7 and 14 units (in 100's) per week, respectively. The transpotation costs(in rupees) per unit between factories to warechouses are given in the table below**

| | **D1** | **D2** | **D3** | **D4** | **supply** |
|---|---|---|---|---|---|
| **S1** | **19** | **30** | **50** | **10** | **7** |
| **S2** | **70** | **30** | **40** | **60** | **9** |
| **S3** | **40** | **8** | **70** | **20** | **18** |
| **Demand** | **5** | **8** | **7** | **14** | **34** |

**Write a python program to solve transpotation problem for minimize the costs of whole operation**

**-→**

```
import numpy as np
from scipy.optimize import linprog
costs = np.array([[19, 30, 50, 10],
                  [70, 30, 40, 60],
                  [40, 8, 70, 20]])
supply = np.array([7, 9, 18])
demand = np.array([5, 8, 7, 14])
```

```
n_supply = len(supply)
n_demand = len(demand)
c = costs.flatten()
A_eq = np.zeros((n_supply + n_demand, n_supply * n_demand))
for i in range(n_supply):
    A_eq[i, i*n_demand:(i+1)*n_demand] = 1
for j in range(n_demand):
    A_eq[n_supply+j, j::n_demand] = 1
b_eq = np.concatenate((supply, demand))
bounds = [(0, None)] * len(c)
res = linprog(c, A_eq=A_eq, b_eq=b_eq, bounds=bounds, method='simplex')
solution = res.x.reshape((n_supply, n_demand))

print("Optimal solution:")
print(solution)
print("Total cost:", res.fun)
```

output :

```
Optimal solution:
[[ 5.  0.  0.  2.]
 [ 0.  2.  7.  0.]
 [ 0.  6.  0. 12.]]
Total cost: 743.0
```