Sahakar Maharshi Bhausaheb Santuji Thorat

College Sangamner

DEPARTMENT OF COMPUTER SCIENCE

Sub : Mathematics

Remark
Demonstrator's Signature
Date:- / /20

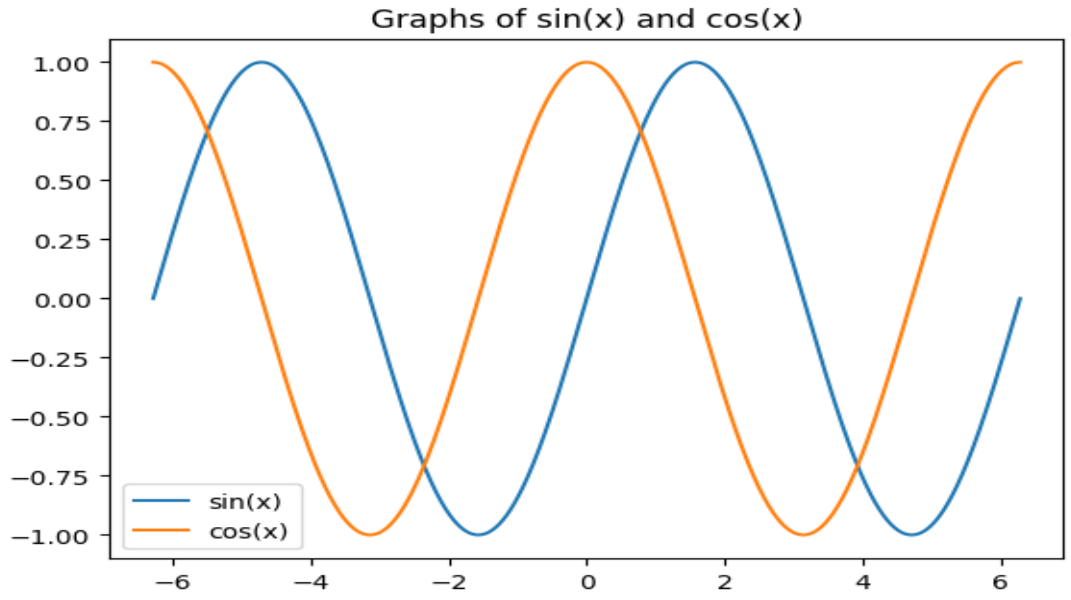
Name:-_Gorde Yash Somnath_____ Roll.No:-_21__Date:-_____
Title of the expt:-__Slip no 20_____Page.no:-_____Class:-_____BCS_____

Q1. Attempt any TWO of the following

A) Write a python program to plot 2D graph of the function f(x)=sin x and g(x)=cos x in [-2π,2π]

-→

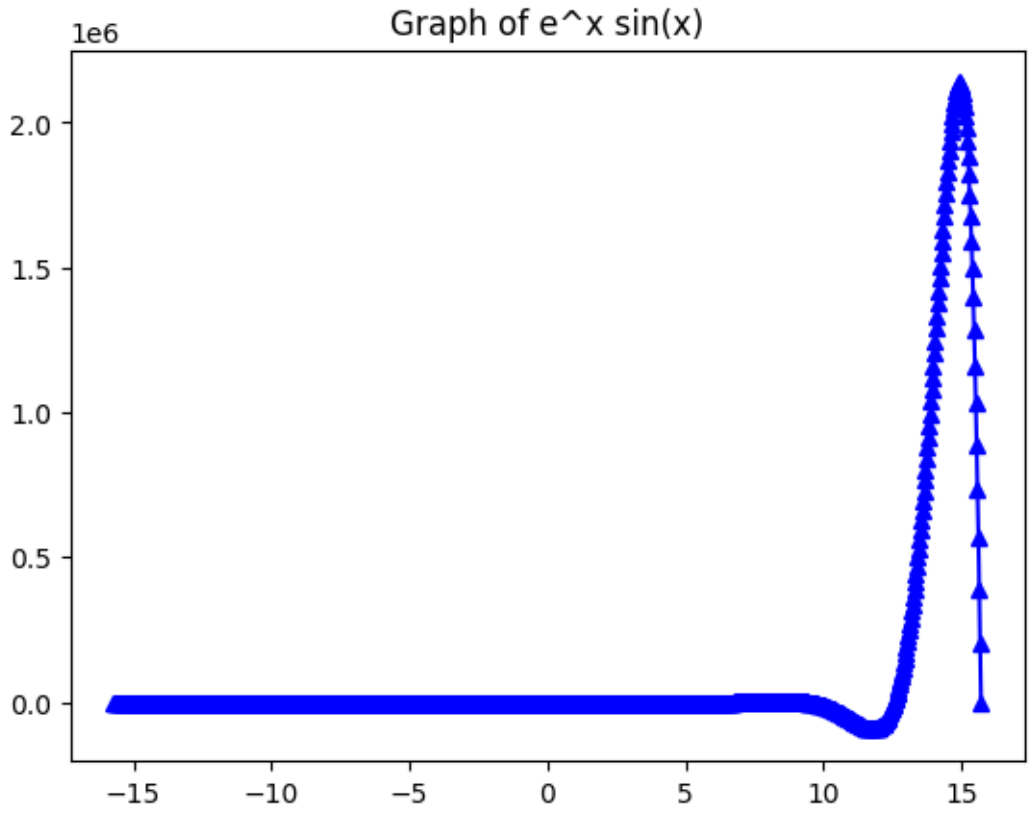
```
import numpy as np
import matplotlib.pyplot as plt
def f(x):
    return np.sin(x)
def g(x):
    return np.cos(x)

x = np.linspace(-2*np.pi, 2*np.pi, 1000)
plt.plot(x, f(x), label='sin(x)')
plt.plot(x, g(x), label='cos(x)')
plt.legend()
plt.title('Graphs of sin(x) and cos(x)')
plt.show()
```



B) Write a python program to plot the 2D graph of the function f(x)=e^x sin x in [-5π,5π] with blue points line with upward poininting triangle

-→

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

x = np.linspace(-5*np.pi, 5*np.pi, 1000)

y = f(x)
plt.plot(x, y, 'b^-')
plt.title('Graph of e^x sin(x)')
plt.show()
```

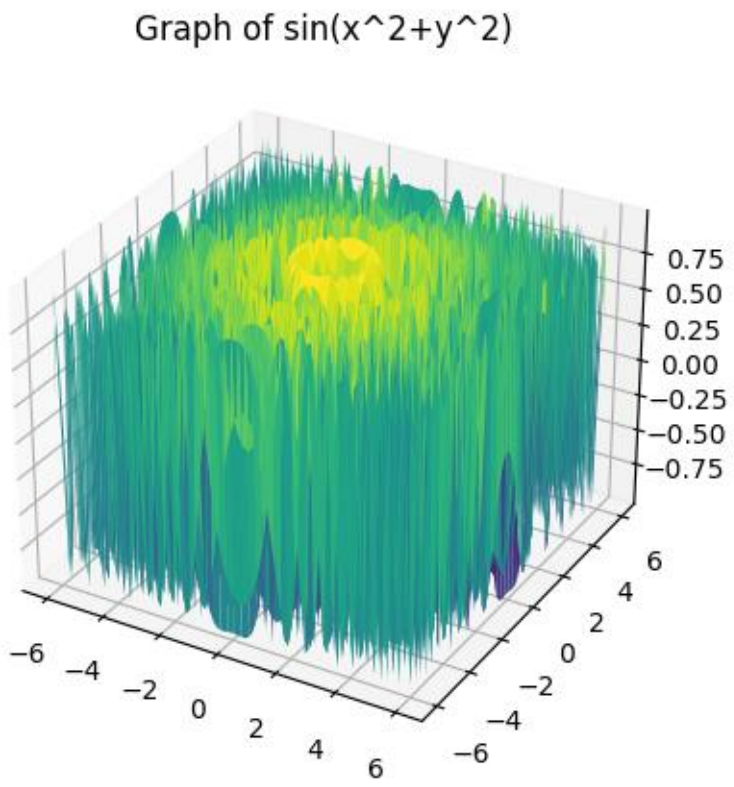


C)Write a python program to plot the 3D graph of the function f(x)=sin(x^2+y^2) -6<x,y<6

-→

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
def f(x, y):
```

```
    return np.sin(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 = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis')
ax.set_title('Graph of sin(x^2+y^2)')
plt.show()
```



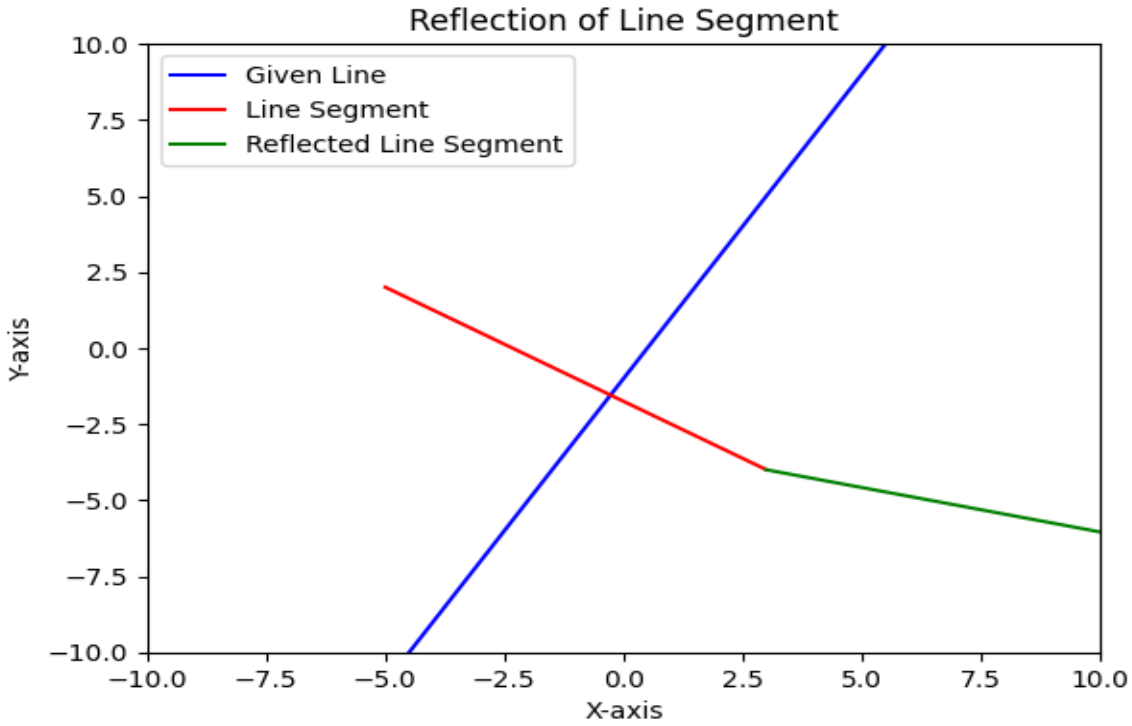
Q2) Attempt any TWO of the following

A) Write a python program to reflect the line segment joining the points A[-5,2],B[3,-4] through the line y=2x-1

-→

```
import numpy as np
import matplotlib.pyplot as plt
def given_line(x):
    return 2*x - 1
```

```
A = np.array([-5, 2])
B = np.array([3, -4])

m = 2
c = -1

x = np.linspace(-10, 10, 100)
plt.plot(x, given_line(x), 'b-', label='Given Line')
plt.plot([A[0], B[0]], [A[1], B[1]], 'r-', label='Line Segment')

v = np.array([B[0]-A[0], B[1]-A[1]])
w = np.array([m, -1])
projection = (np.dot(v, w) / np.dot(w, w)) * w
reflection = 2*projection - v
C = B + reflection

plt.plot([B[0], C[0]], [B[1], C[1]], 'g-', label='Reflected Line Segment')
plt.legend()

plt.xlim(-10, 10)
plt.ylim(-10, 10)

plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Reflection of Line Segment')
plt.show()
```



B) Write a python program to find the area and perimeter of a polygon with vertices (0,0),(-2,0),(5,5),(1,-3)

-→

```
from shapely.geometry import Polygon

vertices = [(0, 0), (-2, 0), (5, 5), (1, -3)]
polygon = Polygon(vertices)
area = polygon.area
perimeter = polygon.length
print(f"Area: {area:.2f}")
print(f"Perimeter: {perimeter:.2f}")
```

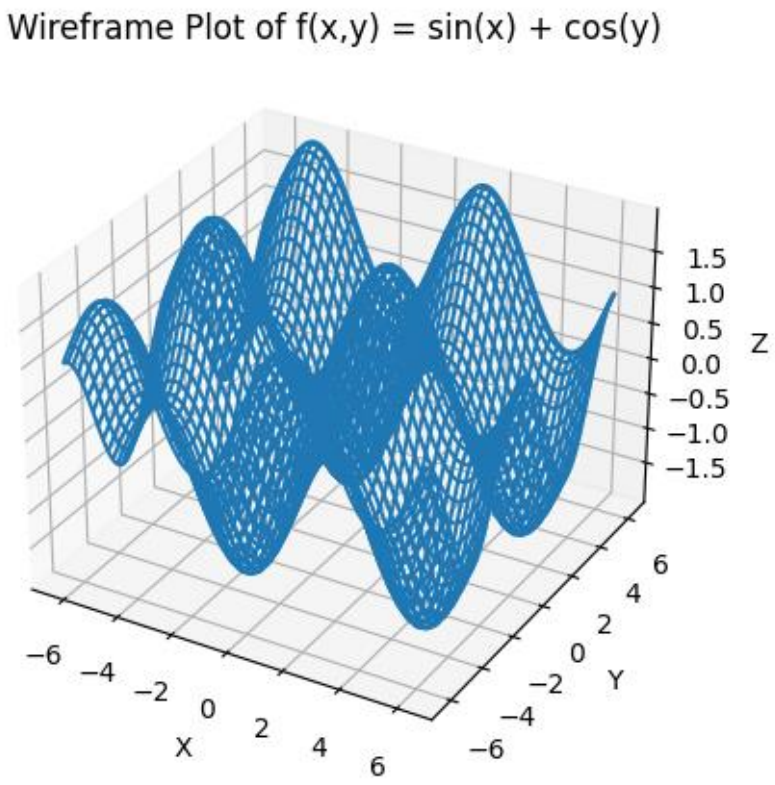
output :

```
Area: 15.00
Perimeter: 22.71
```

C)Write a python program to plot the 3D graph of the function f(x,y)=sin x +cos y ,x,y E[-2π,2π] using wireframe plot

-→

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
def f(x, y):
    return np.sin(x) + np.cos(y)

x = np.linspace(-2*np.pi, 2*np.pi, 100)
y = np.linspace(-2*np.pi, 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('Wireframe Plot of f(x,y) = sin(x) + cos(y)')
plt.show()
```



Q3) Attempt the following

A) Attempt any ONE of the following

I) Write a python program to solve the following LPP :

Max Z=x+y

Subject to x-y≥1

X+y≥2

X,y≥0

-→

```
import numpy as np
from scipy.optimize import linprog
c = np.array([1, 1])
A = np.array([[-1, 1], [1, 1]])
b = np.array([1, 2])
bounds = [(0, None), (0, None)]
res = linprog(-c, A_ub=A, b_ub=b, bounds=bounds)

print(f"Optimal value of Z: {-res.fun:.2f}")
print(f"x: {res.x[0]:.2f}")
print(f"y: {res.x[1]:.2f}")
```

output :

Optimal value of Z: 2.00
x: 2.00
y: 0.00

II) 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 = [(4, None), (None, 2)]
res = linprog(c, A_ub=A, b_ub=b, bounds=bounds)

print(f"Optimal value of Z: {res.fun:.2f}")
print(f"x: {res.x[0]:.2f}")
print(f"y: {res.x[1]:.2f}")
```

output :

Optimal value of Z: 18.00
x: 4.00
y: 2.00

B) Attempt any ONE of the following

I) Apply the following transformation on the point P[3,-2]

A) Scaling in Y direction by 4 units
B) Reflection through Y axis
C) Rotation about origin by an angle 45◦
D)Reflection through the line y=x

-→

```
import numpy as np
P = np.array([3, -2])

A = np.array([[1, 0], [0, 4]])
P_A = A.dot(P)
print("After scaling in Y direction by 4 units:", P_A)

B = np.array([[-1, 0], [0, 1]])
P_B = B.dot(P)
print("After reflection through Y axis:", P_B)

theta = np.deg2rad(45)
C = np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])
P_C = C.dot(P)
print("After rotation about origin by 45 degrees:", P_C)

D = np.array([[0, 1], [1, 0]])
P_D = D.dot(P)
print("After reflection through the line y=x:", P_D)
```

output :

```
After scaling in Y direction by 4 units: [ 3 -8]
After reflection through Y axis: [-3 -2]
After rotation about origin by 45 degrees: [3.53553391 0.70710678]
After reflection through the line y=x: [-2  3]
```

II) Apply the following transformation on the point P[3,-2]

A) Shering in x direction by -2 units

B) Scaling in x and y direction by -3 and 2 units respectively

C) reflection through x axis

D) Reflection through the line y=-x

-→

```
import numpy as np
P = np.array([3, -2])

A = np.array([[1, -2], [0, 1]])
P_A = A.dot(P)
print("After shering in x direction by -2 units:", P_A)
```

```
B = np.array([[-3, 0], [0, 2]])
P_B = B.dot(P)
print("After scaling in x and y direction by -3 and 2 units respectively:", P_B)

C = np.array([[1, 0], [0, -1]])
P_C = C.dot(P)
print("After reflection through x axis:", P_C)

D = np.array([[0, -1], [-1, 0]])
P_D = D.dot(P)
print("After reflection through the line y=-x:", P_D)
```

output :

```
After shering in x direction by -2 units: [ 7 -2]
After scaling in x and y direction by -3 and 2 units respectively: [-9 -4]
After reflection through x axis: [3 2]
After reflection through the line y=-x: [ 2 -3]
```