DATE:

Prac 1.

Aim: Write a program to demonstrate the following.

a) Addition of two complex numbers.

→
```
a = 4+2j.
b = 3-5j.
print ("Addition is:", a+b)
```

O/P: Addition is: (7-3j).

b) Displaying the conjugate of complex numbers

→
```
a = 2 + 3j.
print ( a.conjugate())
```

O/P: (2-3j).

one side page

DATE:

1c) Plotting a set of complex numbers.

→ import matplotlib.pyplot as plt.

```
x = 2+2j
a = [-2+4j, -1+2j, 0+2j, 1+2j, 2+2j
 ,-1+4j, 0+4j, 1+4j]
x = [x.real for x in a]
y = [x.imag for x in a]
plt.scatter(x,y, color="red")
plt.show().
```

1d) Creating a new plot by rotating given number by degree.

i) 90°

→
```
import matplotlib.pyplot as plt.
x = 2+4j
z = 1j
plt.scatter(x.real, x.imag, color='red'
c = x*z
plt.scatter(c.real, c.imag, color='blue'
plt.show().
```

one side page

1d) ii). 180°

→ import matplotlib.pyplot as plt.
x = 2+4j.
plt.scatter(x.real, x.imag, color='red')
plt.scatter(-1 * x.real, -1 * x.imag, color='green').
plt.show()

1d) iii) 270°

→ import matplotlib.pyplot as plt.
x = 2+4j
z = -1j.
plt.scatter(x.real, x.imag, color='red')
c = x * z
plt.scatter(c.real, c.imag, color='green')
plt.show()

1d) iv). Scaling by number a = 1/2, a = 1/3, a = 2

→ import matplotlib.pyplot as plt.
x = 2+4j.
scale = 0.5
scale1 = 0.33
scale2 = 2
plt.scatter(x.real, x.imag, color='red')
c = scale
d = scale1 * x
e = scale2 * x
plt.scatter(c.real, c.imag, color='green')
plt.scatter(d.real, d.imag, color='blue')
plt.show().

Practical 2

Aim: Write program to do.
- enter vector u as list.
- enter another vector b as list.
- Find the vector au + bv for different values
- find dot product of u and v.

```
import numpy as np.
u = np.array([3, 4, 5])
v = np.array([1, 2, 7])
print("Vector u", u)
print("Vector v", v)
a = int(input("Enter value for a: "))
b = int(input("Enter value for b: "))
d = a * u + b * v
p = np.dot(u, v)
print("Vector au+bv", d)
print("Dot product of u and v:", p)
```

O/P:

```
Vector u [3 4 5]
Vector v [1 2 7]
Enter the value for a: 2
Enter the value for b: 4
Vector au+bv [10 16 38]
Dot product of u and v: 46
```

Practical 3.

Aim: Basic matrix operations

a) matrix addition, :subtraction, multiplication

→ import numpy as np.

```
m1 = np.array([[1,3,4],[8,5,6]])
m2 = np.array([[8,6,9],[9,0,6]]).
print("Add matrix")
a = np.add(m1, m2)
print(a).
print("Subtract matrix")
b = np.Subtract(m2, m1)
print(b)
x = np.array([[1,7,5],[4,5,3],[3,2,1]])
y = np.array([[6,7,1],[2,3,1],[2,2,3]])
t = np.array([[0,0,0],[0,0,0],[0,0,0]]).
print("Multiplication matrix").
for i in range(len(x)):
    for j in range(len(y[0])):
        for k in range(len(y)):
            t[i][j] += x[i][k] * y[k][j]
for r in t:
    print(r)
```

3) O/P:

Add matrix.

[[9 9 13]
 [17 5 12]]

Subtract matrix.

[[7. 3. 5.]
 [1. -5. 0.]]

Multiplication matrix.

[[30 38 29]
 [40 49 42]
 [24 29 26]]

b) Check if matrix is invertible and if yes then find inverse of matrix.

→

```
import numpy as np
m = np.array([[1,2,1],[2,1,0],[3,0,2]])
print("matrix is: ", m)
c = np.linalg.det(m)
print("Determinant is:", c)
if (c!=0):
    minv = np.linalg.inv(m)
    print("inverse of matrix: " minv)
else:
    print("matrix is not invertible").
```

O/P:

```
matrix is: [[1,2,1]
            [2,1,0]
            [3 0 2]
Determinant is: -8.999999998
inverse of matrix is:
[[-0.22222 0.4444444 0.1111111111]
 [0.4444444 0.1111111111 -0.2222222]
 [0.3333333 -0.66666667 0.3333333]]
```

Practical 4.

Aim: Write program to convert matrix into its row echelon form.

a) →

```
from sympy import *
M = Matrix([[1,0,1,3],[2,3,4,7],
[-1,-3,-3,-4]])
print("Matrix: {}".format(M))
M_rref = M.rref()
print("The row echelon form of Matrix M
and the pivot columns: {}".format(M_rref
```

O/P:

```
Matrix : ([[1,0,1,3],[2,3,4,7],[-1,-3,-3,-4]
The row echelon form of Matrix M and the
pivot column
 [[1,0,1,3],
  [0,1,2/3,1/3],
  [0,0,0,0]]),(0,1))
```

b) Program to find rank of matrix.

→
```
import numpy as np
my_matrix = np.array([[1,2,1],[3,4,7],
  [3,6,3]])
print("Matrix")
for row in my_matrix:
    print(row)
    rank = np.linalg.matrix_rank(my_matrix)
print("Rank of given matrix is:", rank)
```

O/P:

```
Matrix
[1 2 1]
[3 4 7]
[3 6 3]
Rank of the given matrix is: 2
```

Practical 5.

Aim: Enter a vector B and find the projection of B orthogonal to given vector u.

→
```
import numpy as np
def opprojection (of_vec, on_vec):
    v1 = np.array (of_vec)
    v2 = np.array (on_vec)
    scal = np.dot (v1, v2)/ np.dot (v2, v2)
    vec = scal * v2
    return round (scal, 10), np.around (vec,
        decimal = 10)
print (opprojection ([4,0,4,0],[1,0,0,0]))
print (opprojection ([4,0,4,0],[8,0,2,0]))
```

O/P:

```
(np.float64(4.0), array([4., 0., 0., 0.]))
(np.float64(0.5882352941), array([4.705882
```