Lab 5

1. Calculate the Kronecker product of the matrices *C* and *D*:

$$C = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix} \quad \text{and} \quad D = \begin{bmatrix} 5 & 0 & 4 \\ 4 & 5 & 0 \\ 0 & 4 & 5 \end{bmatrix}$$

```
import numpy as np
from scipy.linalg import kron

C = np.array([[1, 2, 3], [3, 1, 2], [2, 3, 1]])
D = np.array([[5, 0, 4], [4, 5, 0], [0, 4, 5]])

kron_product_C_D = kron(C, D)
print("Kronecker Product of C and D:\n", kron_product_C_D)
```

```
Kronecker Product of C and D:
 [[ 5  0  4 10  0  8 15  0 12]
 [ 4  5  0  8 10  0 12 15  0]
 [ 0  4  5  0  8 10  0 12 15]
 [15  0 12  5  0  4 10  0  8]
 [12 15  0  4  5  0  8 10  0]
 [ 0 12 15  0  4  5  0  8 10]
 [10  0  8 15  0 12  5  0  4]
 [ 8 10  0 12 15  0  4  5  0]
 [ 0  8 10  0 12 15  0  4  5]]
```

2. Calculate the Kronecker Product of the matrices *U* and *W*:

$$U = \begin{bmatrix} 5 & 2 \\ 3 & 9 \end{bmatrix} \quad \text{and} \quad W = \begin{bmatrix} 5 & 0 & 4 & 9 \\ 4 & 5 & 0 & 9 \\ 0 & 4 & 5 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

```
U = np.array([[5, 2], [3, 9]])
W = np.array([[5, 0, 4, 9], [4, 5, 0, 9], [0, 4, 5, 0], [1, 2, 3, 4]])

kron_product_U_W = kron(U, W)
print("\nKronecker Product of U and W:\n", kron_product_U_W)
```

```

Kronecker Product of U and W:
 [[25  0 20 45 10  0  8 18]
 [20 25  0 45  8 10  0 18]
 [ 0 20 25  0  0  8 10  0]
 [ 5 10 15 20  2  4  6  8]
 [15  0 12 27 45  0 36 81]
 [12 15  0 27 36 45  0 81]
 [ 0 12 15  0  0 36 45  0]
 [ 3  6  9 12  9 18 27 36]]
```

3. Write a Python function `kron_add(A, B)` that takes as input two 2-D NumPy array, `A` and `B`, and calculates their Kronecker sum. Use this function to calculate the Kronecker sum of the matrices *C* and *D* from question 1.

```
def kron_add(A, B):
    A_size = A.shape[0]
    B_size = B.shape[0]
    A_kron_I = np.kron(A, np.eye(B_size))
    I_kron_B = np.kron(np.eye(A_size), B)
    return A_kron_I + I_kron_B

kron_sum_C_D = kron_add(C, D)
print("\nKronecker Sum of C and D:\n", kron_sum_C_D)
```

```
Kronecker Sum of C and D:
 [[6. 0. 4. 2. 0. 0. 3. 0. 0.]
 [4. 6. 0. 0. 2. 0. 0. 3. 0.]
 [0. 4. 6. 0. 0. 2. 0. 0. 3.]
 [3. 0. 0. 6. 0. 4. 2. 0. 0.]
 [0. 3. 0. 4. 6. 0. 0. 2. 0.]
 [0. 0. 3. 0. 4. 6. 0. 0. 2.]
 [2. 0. 0. 3. 0. 0. 6. 0. 4.]
 [0. 2. 0. 0. 3. 0. 4. 6. 0.]
 [0. 0. 2. 0. 0. 3. 0. 4. 6.]]
```

4. **Find the distance matrix for the 3 points in the datamatrix**

$$X = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

```
X = np.array([[0, 0], [1, 1], [2, 2]])
dist_matrix = np.linalg.norm(X[:, np.newaxis] - X, axis=2)
print("\nDistance Matrix for X:\n", dist_matrix)
```

```
Distance Matrix for X:
 [[0.         1.41421356 2.82842712]
 [1.41421356 0.         1.41421356]
 [2.82842712 1.41421356 0.        ]]
```

5. **Use spectral decomposition to find one datamatrix X that is compatible with the distance matrix**

$$D = \begin{bmatrix} 0 & 2 & 8 \\ 2 & 0 & 2 \\ 8 & 2 & 0 \end{bmatrix}$$

```
D = np.array([[0, 2, 8], [2, 0, 2], [8, 2, 0]])
eigenvalues, eigenvectors = np.linalg.eigh(D)
Q = eigenvectors
Lambda = np.diag(eigenvalues)

print("Data Matrix D:", D, "\n\n")
print("Q=", Q, "\n\n", "Λ=", Lambda, "\n\n", "Q.T=", Q.T)
D1 = Q @ Lambda @ Q.T
print("\nData Matrix D (from spectral decomposition):\n", np.round(D1))
```

```
Data Matrix D: [[0 2 8]
 [2 0 2]
 [8 2 0]]


Q= [[ 0.70710678  0.2141865  -0.67388734]
 [ 0.         -0.95302061 -0.30290545]
 [-0.70710678  0.2141865  -0.67388734]]

 Λ= [[-8.          0.          0.        ]
 [ 0.         -0.89897949  0.        ]
 [ 0.          0.          8.89897949]]

 Q.T= [[ 0.70710678  0.         -0.70710678]
 [ 0.2141865  -0.95302061  0.2141865 ]
```

```
 [-0.67388734 -0.30290545 -0.67388734]]

Data Matrix D (from spectral decomposition):
 [[-0.  2.  8.]
 [ 2.  0.  2.]
 [ 8.  2. -0.]]
```

Start coding or generate with AI.