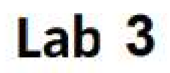
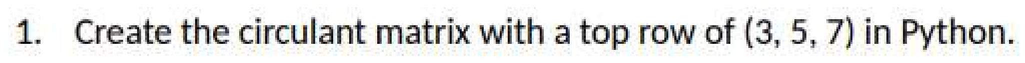


Lab 3

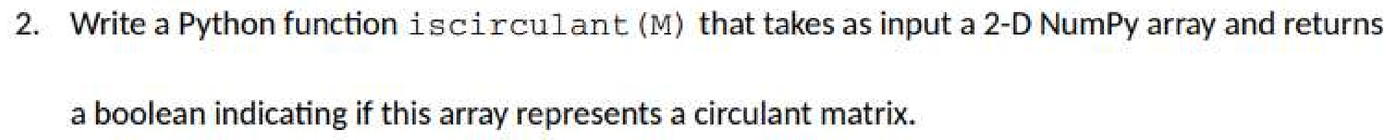


1. Create the circulant matrix with a top row of (3, 5, 7) in Python.

```
import numpy as np
from scipy.linalg import circulant, eig


top_row = np.array([3, 5, 7])
circ_matrix = circulant(top_row)
circ_matrix
```

```
array([[3, 7, 5],
       [5, 3, 7],
       [7, 5, 3]])
```

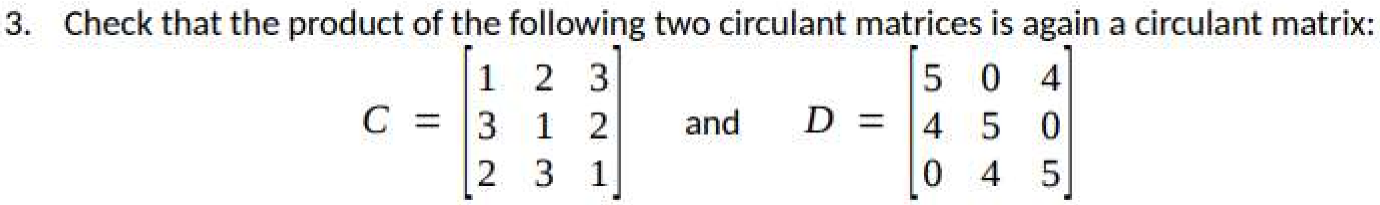


2. Write a Python function `iscirculant(M)` that takes as input a 2-D NumPy array and returns a boolean indicating if this array represents a circulant matrix.

```
def iscirculant(M):
  rows, cols = M.shape
  first_row = M[0]
  for i in range(1, rows):
    if not np.all(M[i] == np.roll(first_row, i)):
      return False
  return True

print(circ_matrix)
iscirculant(circ_matrix)
```

```
[[3 7 5]
 [5 3 7]
 [7 5 3]]
True
```



3. Check that the product of the following two circulant matrices is again a circulant matrix:

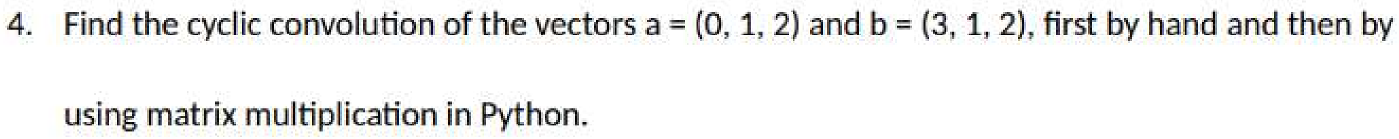
$$C = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix} \quad \text{and} \quad D = \begin{bmatrix} 5 & 0 & 4 \\ 4 & 5 & 0 \\ 0 & 4 & 5 \end{bmatrix}$$

```
C = np.array([[1, 2, 3],
              [3, 1, 2],
              [2, 3, 1]])

D = np.array([[5, 0, 4],
              [4, 5, 0],
              [0, 4, 5]])

CD = np.dot(C, D)
print(CD)
iscirculant(CD)
```

```
[[13 22 19]
 [19 13 22]
 [22 19 13]]
True
```

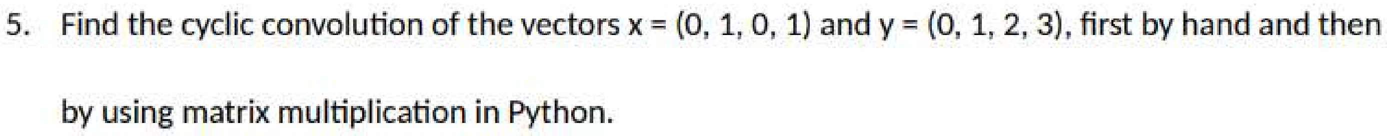


4. Find the cyclic convolution of the vectors a = (0, 1, 2) and b = (3, 1, 2), first by hand and then by using matrix multiplication in Python.

```
a = np.array([0, 1, 2])
b = np.array([3, 1, 2])

A_circ = circulant(a)
A_circ @ b
```

```
array([4, 7, 7])
```

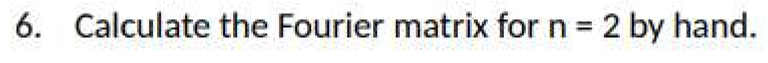


5. Find the cyclic convolution of the vectors x = (0, 1, 0, 1) and y = (0, 1, 2, 3), first by hand and then by using matrix multiplication in Python.

```
x = np.array([0, 1, 0, 1])
y = np.array([0, 1, 2, 3])

X_circ = circulant(x)
X_circ @ y
```

```
array([4, 2, 4, 2])
```



6. Calculate the Fourier matrix for n = 2 by hand.

7. Verify the last question in Python.

```
n = 2
F_2 = np.fft.fft(np.eye(n))
F_2
```

```
array([[ 1.+0.j,  1.+0.j],
       [ 1.+0.j, -1.+0.j]])
```

8. Calculate the Fourier matrix for n = 4 directly in Python.

```
n = 4
F_4 = np.fft.fft(np.eye(n))
F_4
```

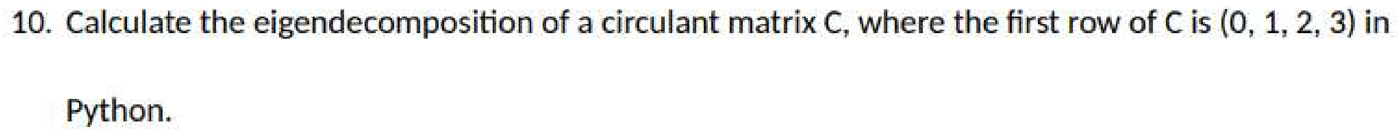
```
array([[ 1.+0.j,  1.+0.j,  1.+0.j,  1.+0.j],
       [ 1.+0.j,  0.-1.j, -1.+0.j,  0.+1.j],
       [ 1.+0.j, -1.+0.j,  1.+0.j, -1.+0.j],
       [ 1.+0.j,  0.+1.j, -1.+0.j,  0.-1.j]])
```

9. Calculate the eigendecomposition of the matrix $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ by hand.

```
A = np.array([[2, 1],
              [1, 2]])
eigvals_A, eigvecs_A = eig(A)
print(eigvals_A)
print("\n\n")
print(eigvecs_A)
```

```
[3.+0.j 1.+0.j]
```

```
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
```



10. Calculate the eigendecomposition of a circulant matrix C, where the first row of C is (0, 1, 2, 3) in Python.

```
C_circ = circulant([0, 1, 2, 3])
eigvals_C_circ, eigvecs_C_circ = eig(C_circ)
print(eigvals_C_circ)
print("\n\n")
print(eigvecs_C_circ)
```

```
[ 6.+0.j -2.+2.j -2.-2.j -2.+0.j]



[[ 5.00000000e-01+0.00000000e+00j  5.00000000e-01+0.00000000e+00j
   5.00000000e-01-0.00000000e+00j -5.00000000e-01+0.00000000e+00j]
 [ 5.00000000e-01+0.00000000e+00j  2.00001206e-17+5.00000000e-01j
   2.00001206e-17-5.00000000e-01j  5.00000000e-01+0.00000000e+00j]
 [ 5.00000000e-01+0.00000000e+00j -5.00000000e-01+1.80155628e-16j
  -5.00000000e-01-1.80155628e-16j -5.00000000e-01+0.00000000e+00j]
 [ 5.00000000e-01+0.00000000e+00j  3.23220574e-17-5.00000000e-01j
   3.23220574e-17+5.00000000e-01j  5.00000000e-01+0.00000000e+00j]]
```