

UE21MA241B
LINEAR ALGEBRA

Applications of Signal Processing using
Linear Algebra

Aneesh Rao- PES2UG21EC014
Anirudh N S-PES2UG21EC015
Anjali-PES2UG21EC016
Anoop B S-PES2UG21EC017

Linear Filtering

The process of applying a linear filter to a signal or an image to manipulate it is known as linear filtering. To create an output signal or image, a linear filter is a mathematical operation that is applied to each pixel of the input signal or image.

- 1. Convolution
- 2. Digital Filtering

Convolution

it is a way of combining two functions to produce a third function that describes how one function affects the other. In signal processing, convolution is used to calculate the output of a system in response to an input signal. It is a mathematical tool that allows us to understand how signals are transformed as they pass through a system.

The convolution of two functions $f(x)$ and $g(x)$ is defined as:

$$(f * g)(x) = \int f(t) g(x-t) dt$$

Convolution

```
In [24]: ▶ #Linear convolution
import numpy as np

def convolve(x, h):
    N = len(x)
    M = len(h)
    X = np.zeros((N+M-1, N))
    for i in range(N):
        X[i:i+M, i] = h
    y = np.dot(X, x)
    return y

x = np.array([1,2,3])
h = np.array([1,2,1,1,1])
y = convolve(x, h)
print(y)

[1.  4.  8.  9.  6.  5.  3.]
```

Convolution

Advantages of using matrices for convolution

1. Easy implementation
2. Fast computation
3. Flexibility
4. Easy Boundary Handling

Disadvantages

1. Computationally expensive
2. Boundary handling
3. Memory requirements

Digital Filtering

Digital filters are signal processing units that use digital signal processing methods to treat discrete-time signals. They are frequently used in a variety of applications, such as biological signal processing, telecommunications, control systems, audio and video processing, and audio and video processing

1. Infinite Impulse Response Filters
2. Finite Impulse Response Filters

Digital Filtering

FIR Filters

A Finite Impulse Response (FIR) filter is a type of digital filter used in signal processing applications. It operates on a discrete-time signal, which is a signal that is sampled at specific time intervals. The main function of an FIR filter is to remove unwanted frequency components from a signal while preserving the desired components.

FIR Filters

```
In [25]: ▶ import numpy as np
import matplotlib.pyplot as plt

def fir_filter_freq(x, b):
    M = len(b)
    N = len(x)
    H = np.fft.fft(b, N)
    X = np.fft.fft(x, N)
    Y = H * X
    y = np.real(np.fft.ifft(Y, N))

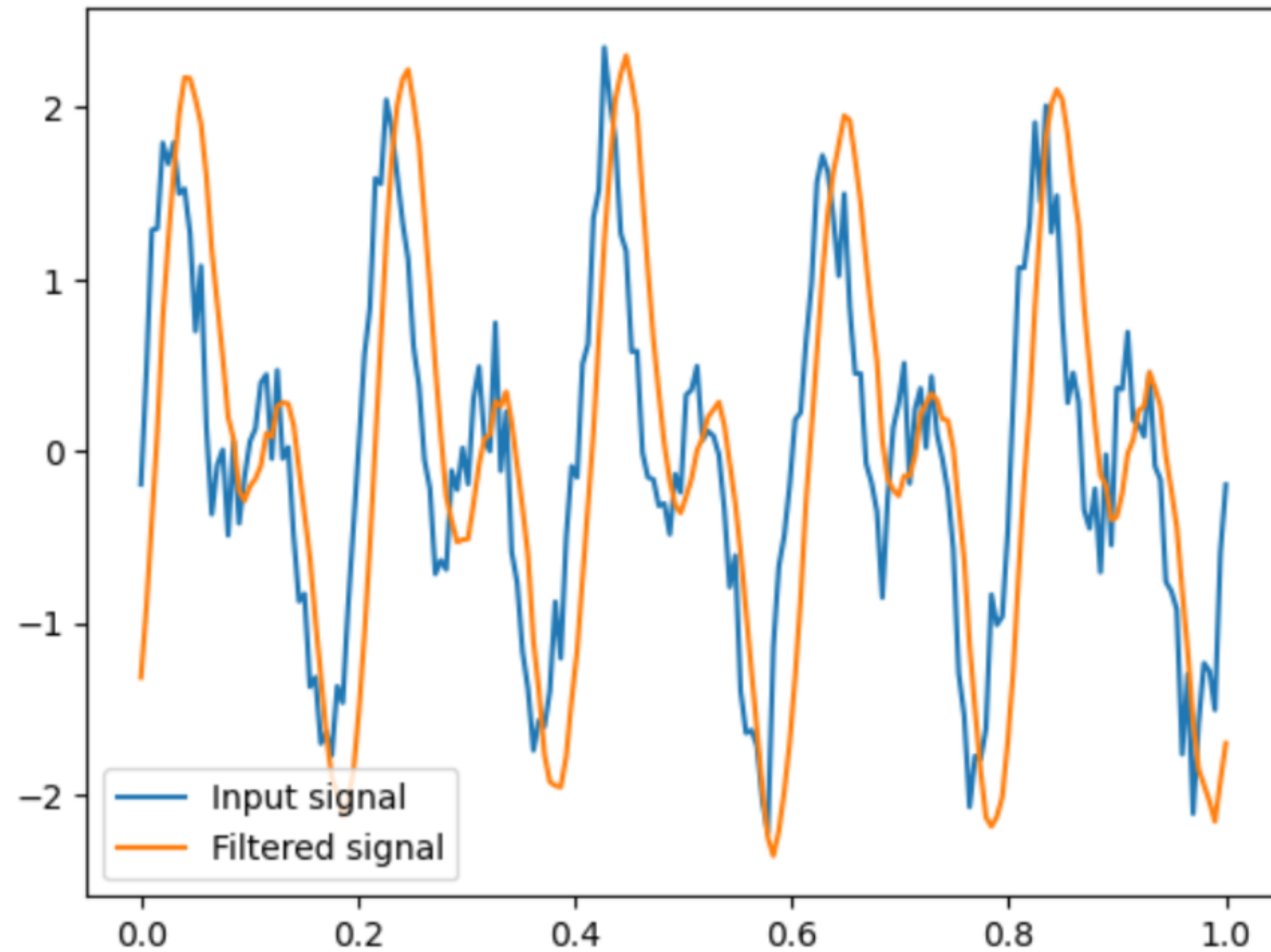
    return y[0:N]

n = 200
t = np.linspace(0, 1, n)
x = np.sin(2*np.pi*5*t) + np.sin(2*np.pi*10*t) + np.random.randn(n)*0.25
b = np.array([0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2])

y = fir_filter_freq(x, b)

plt.plot(t, x, label='Input signal')
plt.plot(t, y, label='Filtered signal')
plt.legend()
plt.show()
```


FIR Filters



FIR Filters

Advantages of linear algebra to design FIR Filters

1. Systematic Approach
2. Flexibility
3. Optimization
4. Faster Implementation

Disadvantages

1. Trade-off between system length and performance
2. Computational complexity


Fourier Analysis

The mathematical process of Fourier analysis is used to break down a complicated signal into its component frequencies. The method is named for French mathematician Joseph Fourier, who first proposed the idea of expressing a function as the sum of sinusoidal functions. Any complex signal can be represented as a sum of sinusoidal functions with various frequencies, amplitudes, and phases, according to the theory behind Fourier analysis. The Fourier series is this illustration, and it enables us to examine a signal's frequency content. The Fourier transform, a mathematical operation that transforms a time-domain signal into its frequency-domain representation, is used in practice to implement Fourier analysis.

Discrete Fourier Transform

A decorative graphic consisting of several concentric, thin circular lines in the top right corner of the slide.

The discrete Fourier transform (DFT) is a specific application of Fourier analysis. A finite series of evenly spaced samples of a signal from the time domain is converted into the frequency domain using the DFT, a mathematical operation. Similar to the Fourier transform, the DFT enables us to examine a signal's frequency content, but it is tailored exclusively for digital signals.

A decorative graphic consisting of several concentric, thin circular lines in the bottom left corner of the slide.

Discrete Fourier Transform

D^1			D^2					D^3												
		l					l													
		0	0					0	1	0	1									
k	0	1	W_2^0	k	0	1	W_4^0	W_4^1 <th rowspan="10">k</th> <th>0</th> <td>1</td> <td>2</td> <td>3</td> <td>W_8^0</td> <td>W_8^1</td> <td>W_8^2</td> <td>W_8^3</td>	k	0	1	2	3	W_8^0	W_8^1	W_8^2	W_8^3			
	0	W_2^0	W_2^0		1	1	W_4^0	W_4^1		1	1	1	1	W_8^0	W_8^1	W_8^2	W_8^3			
k	0	W_4^0	W_4^0	k	0	W_4^0	W_4^1	2		1	1	1	W_8^0	W_8^1	W_8^2	W_8^3				
	1	W_4^1	W_4^1		1	W_4^1	W_4^2	3		1	1	1	W_8^0	W_8^1	W_8^2	W_8^3				
k	0	W_8^0	W_8^0	k	0	W_8^0	W_8^1	W_8^2		0	W_8^0	W_8^1	W_8^2	W_8^3	k	0	W_8^0	W_8^1	W_8^2	W_8^3
	1	W_8^1	W_8^1		1	W_8^1	W_8^2	W_8^3		1	W_8^1	W_8^2	W_8^3	W_8^4	1 <td>W_8^1</td> <td>W_8^2</td> <td>W_8^3</td> <td>W_8^4</td>	W_8^1	W_8^2	W_8^3	W_8^4	
k	2	W_8^2	W_8^2	k	2	W_8^2	W_8^3	W_8^4		2	W_8^2	W_8^3	W_8^4	W_8^5	2 <td>W_8^2</td> <td>W_8^3</td> <td>W_8^4</td> <td>W_8^5</td>	W_8^2	W_8^3	W_8^4	W_8^5	
	3	W_8^3	W_8^3		3	W_8^3	W_8^4	W_8^5		3	W_8^3	W_8^4	W_8^5	W_8^6	3 <td>W_8^3</td> <td>W_8^4</td> <td>W_8^5</td> <td>W_8^6</td>	W_8^3	W_8^4	W_8^5	W_8^6	

S_{00}	S_{01}
S_{10}	S_{11}

S_{00}	S_{01}
S_{10}	S_{11}

Discrete Fourier Transform

```
In [83]: ▶ import numpy as np
import matplotlib.pyplot as plt

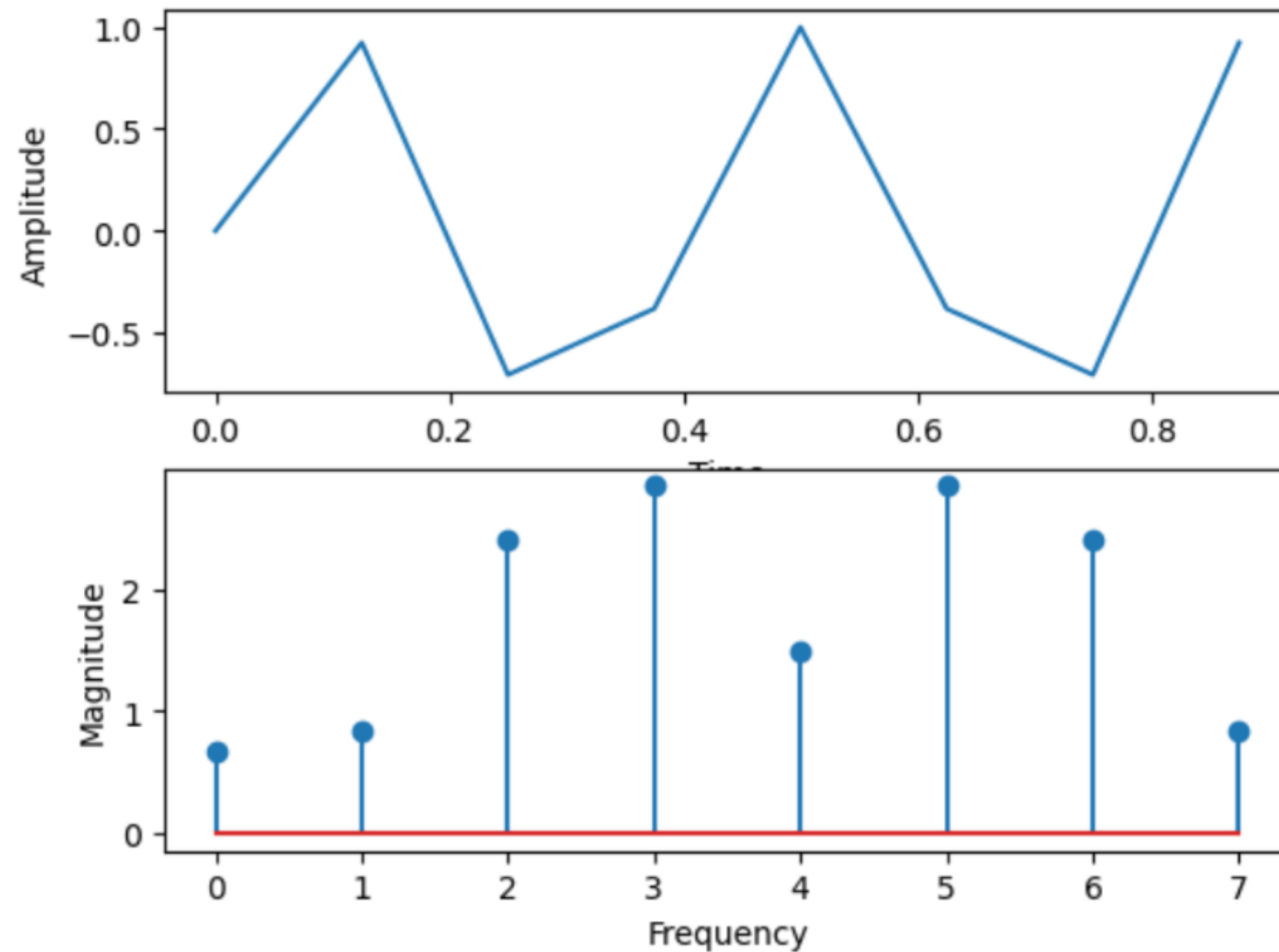
t = np.linspace(0, 1, 8, endpoint=False)
x = np.sin(5*np.pi*t)

N = len(x)
W = np.zeros((N, N), dtype=np.complex_)
for n in range(N):
    for k in range(N):
        W[n, k] = np.exp(-2j*np.pi*n*k/N)
X = np.dot(W, x)
print(X)
mag = np.abs(X)

fig, ax = plt.subplots(2, 1)
ax[0].plot(t, x)
ax[0].set_xlabel('Time')
ax[0].set_ylabel('Amplitude')
ax[1].stem(mag)
ax[1].set_xlabel('Frequency')
ax[1].set_ylabel('Magnitude')
plt.show()
```

Discrete Fourier Transform

```
[ 0.66817864+0.00000000e+00j  0.84775907-7.77156117e-16j  
 2.41421356-1.85844335e-16j -2.84775907+5.55111512e-16j  
-1.49660576-7.33125383e-16j -2.84775907-3.55271368e-15j  
 2.41421356+2.55109146e-15j  0.84775907+3.33066907e-15j]
```



Eigenvalues and Eigenvectors

Eigenvalues are an important notion in linear algebra. An eigenvalue of a square matrix A is a scalar that fulfills the following equation:

$$A v = \lambda v$$

where v is the equation-satisfying eigenvector, a nonzero vector. In other words, when the associated eigenvector v of the matrix A is multiplied, the outcome is a scalar multiple of v that is equal to λ times v .

A matrix's eigenvalues are significant because they reveal details about the matrix's characteristics. For instance, the trace of A (the total of A 's diagonal elements) is equal to the sum of A 's eigenvalues, and the determinant of A is equal to the product of its eigenvalues.

Eigenvalues and Eigenvectors

the DFT matrix's eigenvalues and eigenvectors are utilized to examine the characteristics of the DFT and the signal being transformed. For instance, the phases of the eigenvalues represent the phase shifts of the respective frequency components, and the magnitudes of the eigenvalues show how much energy is present at each frequency component of the signal.

Eigenvalues and Eigenvectors

```
In [30]: ▶ import numpy as np
N = 8
t = np.arange(N)
x = np.sin(5*np.pi*t)
W = np.zeros((N, N), dtype=np.complex64)
for i in range(N):
    for j in range(N):
        W[i,j] = np.exp(-2j * np.pi * i * j / N)

X = np.dot(W, x)
el, ev = np.linalg.eig(W)
print('Eigenvalues of DFT matrix:')
print(el)
print('Eigenvectors of DFT matrix:')
print(ev)

eigvals_mag = np.absolute(ev)
print("Energy is present at each frequency component of the signal.:\n", eigvals_mag)
```

Eigenvalues

Eigenvalues of DFT matrix:

```
[-2.8284271e+00-3.2129265e-16j -2.2204460e-15+2.8284271e+00j  
 -2.8284271e+00-1.9428903e-15j -4.4408921e-16-2.8284271e+00j  
  1.6104739e-15-2.8284271e+00j  2.8284271e+00-4.4408921e-16j  
  2.8284271e+00+3.7375329e-20j  2.8284271e+00+1.6094541e-15j]
```

Eigenvectors

Eigenvectors of DFT matrix:

```
[[ 5.68527341e-01+0.0000000e+00j -1.51035358e-16+4.9547526e-17j
  -8.21957216e-02+1.2762114e-01j  2.29768672e-16+1.2724280e-16j
  -1.60119323e-16+3.6741965e-16j  4.77433826e-09-5.7342531e-09j
   8.22628558e-01+0.0000000e+00j -5.25779203e-02-1.1012467e-03j]
[-3.10937911e-01+2.9233371e-18j -3.53553385e-01+2.1605149e-16j
  -1.15920946e-01-9.0241775e-02j -5.00000000e-01+2.4052910e-17j
  -3.53553385e-01+6.2221188e-09j -5.00000000e-01-5.1667803e-09j
   2.12358877e-01+7.3711335e-04j -2.93917626e-01-3.8934601e-04j]
[-3.10937911e-01-1.1809027e-17j  5.00000000e-01+0.0000000e+00j
   5.94216883e-01+0.0000000e+00j -3.00582048e-09-1.2075881e-08j
  -5.00000000e-01+1.2803431e-18j  1.71150968e-10-8.6189067e-10j
   2.16346949e-01-4.3178772e-04j  1.50394469e-01-2.2807566e-04j]
[-3.10937911e-01+2.9233371e-18j  3.53553385e-01-3.3798904e-17j
  -1.15920946e-01-9.0241775e-02j -5.00000000e-01+1.7077875e-08j
   3.53553385e-01+6.2221188e-09j  5.00000000e-01+0.0000000e+00j
   2.12358877e-01+7.3710218e-04j -2.93917626e-01-3.8935299e-04j]
[-3.10937911e-01-2.6925857e-17j -1.71988941e-16-2.0485792e-16j
  -4.10069674e-01-1.2762114e-01j  2.43360110e-16-2.9566429e-17j
  -2.72747918e-17+2.7639471e-16j -3.94794775e-09+1.5726777e-09j
   2.21986935e-01-2.0848557e-03j  7.78746665e-01+0.0000000e+00j]
[-3.10937911e-01-4.7419849e-17j -3.53553385e-01-2.1391284e-16j
  -1.15920946e-01-9.0241775e-02j  5.00000000e-01-1.7077875e-08j
  -3.53553385e-01-6.2221188e-09j  5.00000000e-01-3.9975778e-16j
   2.12358877e-01+7.3710218e-04j -2.93917626e-01-3.8935299e-04j]
[-3.10937911e-01-5.4935738e-17j -5.00000000e-01+2.8404683e-16j
   5.94216883e-01+9.1220757e-17j  3.00582004e-09+1.2075881e-08j
   5.00000000e-01+0.0000000e+00j  1.71151066e-10-8.6189039e-10j
   2.16346949e-01-4.3178772e-04j  1.50394469e-01-2.2807566e-04j]
[-3.10937911e-01-4.7419849e-17j  3.53553385e-01-2.5664063e-16j
  -1.15920946e-01-9.0241775e-02j  5.00000000e-01+0.0000000e+00j
   3.53553385e-01-6.2221184e-09j -5.00000000e-01-5.1667799e-09j
   2.12358877e-01+7.3711335e-04j -2.93917626e-01-3.8934601e-04j]]
```


Eigenvalues and Eigenvectors

Energy at each frequency component of the signal.:

```
[[5.6852734e-01 1.5895482e-16 1.5180017e-01 2.6264877e-16 4.0079346e-16
 7.4616331e-09 8.2262856e-01 5.2589450e-02]
 [3.1093791e-01 3.5355338e-01 1.4690556e-01 5.0000000e-01 3.5355338e-01
 5.0000000e-01 2.1236016e-01 2.9391789e-01]
 [3.1093791e-01 5.0000000e-01 5.9421688e-01 1.2444350e-08 5.0000000e-01
 8.7871965e-10 2.1634738e-01 1.5039465e-01]
 [3.1093791e-01 3.5355338e-01 1.4690556e-01 5.0000000e-01 3.5355338e-01
 5.0000000e-01 2.1236016e-01 2.9391789e-01]
 [3.1093791e-01 2.6748266e-16 4.2946979e-01 2.4514957e-16 2.7773721e-16
 4.2496597e-09 2.2199672e-01 7.7874666e-01]
 [3.1093791e-01 3.5355338e-01 1.4690556e-01 5.0000000e-01 3.5355338e-01
 5.0000000e-01 2.1236016e-01 2.9391789e-01]
 [3.1093791e-01 5.0000000e-01 5.9421688e-01 1.2444350e-08 5.0000000e-01
 8.7871938e-10 2.1634738e-01 1.5039465e-01]
 [3.1093791e-01 3.5355338e-01 1.4690556e-01 5.0000000e-01 3.5355338e-01
 5.0000000e-01 2.1236016e-01 2.9391789e-01]]
```

Eigenvalues and Eigenvectors

Advantages of using eigenvalues in Fourier Analysis

1. Efficient computation
2. Dimensionality reduction
3. Interpretability
4. Noise Resistance

THANK YOU