Name	Anish Gade
Branch	T.E CSE (DS)
Subject	FOSIP
Experiment No.	1
Date	10-09-2023

Aim: To study mathematical operation such as : Linear Convolution, Circular Convolution, and Linear Convolution using Circular Convolution.

Objectives:

- To Develop a function to find Linear Convolution and Circular Convolution
- To Calculate Linear convolution, Circular convolution, Linear Convolution using Circular Convolution and verify the results using mathematical formulation.
- To Conclude on aliasing effect in Circular convolution

Theory:

Introduction:

Convolution is a fundamental mathematical operation that plays a crucial role in various fields, including signal processing, image processing, and deep learning. It is a mathematical technique used to combine two functions or signals to produce a third, often representing the interaction between them. In this article, we will explore the concept of convolution, its applications, and its significance in the realms of both traditional signal processing and modern deep learning.

The Basics of Convolution:

Convolution is a mathematical operation that takes two input functions, typically denoted as f(t) and g(t) in continuous time or f[n] and g[n] in discrete time, and produces an output function, usually denoted as h(t) or h[n]. The operation is represented by an operator symbol ($*$), and its formula can be written as:

Continuous-time convolution:

$$f(x) * g(x) \quad = \quad \int_{-\infty}^{\infty} f(\tau) \cdot g(x-\tau)\, d\tau$$

Discrete-time convolution:

$$f[x] * g[x] \quad = \quad \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$

Linear and Circular Convolution:

- **Linear Convolution:**
  Linear convolution is the most common form of convolution, and it's used when the sequences or signals involved have a defined beginning and end. It's often denoted as "$\otimes$" and is computed using the convolution sum.
- **Circular Convolution:**
  Circular convolution is employed when the sequences or signals involved are periodic, meaning they repeat indefinitely without a distinct beginning or end. It is often denoted as "$\circledast$" and is computed using the circular convolution formula.

Applications of Convolution:

- **Applications in Signal Processing:**
  Convolution has a wide range of applications in signal processing. Some key uses include:

  a. **Filtering:** Convolution is used for filtering signals to remove noise or enhance specific frequency components. For example, it's used in image processing to apply various filters like blurring, sharpening, and edge detection.

  b. **Time-Domain Analysis**: Convolution is employed to analyze how a system responds to an input signal, such as in calculating the response of an electrical circuit to an input voltage.

  c. **Cross-Correlation:** Convolution is used to measure the similarity between two signals, a technique known as cross-correlation. It's used in pattern recognition, speech recognition, and more.

- **Convolution in Deep Learning:**
  In recent years, convolution has gained tremendous popularity in the field of deep learning, especially in Convolutional Neural Networks (CNNs). CNNs are designed to automatically learn and extract hierarchical features from input data, making them particularly suited for image and audio analysis. The key components of convolution in deep learning are:

  a. **Convolutional Layers:** In CNNs, convolutional layers consist of learnable filters that slide over the input data, computing the convolution operation. These filters automatically learn to detect features like edges, textures, and more.

  b. **Pooling Layers:** Pooling layers, often used alongside convolutional layers, reduce the spatial dimensions of feature maps while retaining essential information. Common pooling operations include max-pooling and average-pooling.

  c. **Strides and Padding:** Strides control the step size of the filter as it slides over the input data, while padding adds extra pixels to the input to control the output size.

d. **Multiple Channels:** In CNNs, data often consists of multiple channels (e.g., RGB channels in images). Convolution operations can handle multiple channels simultaneously, allowing for the extraction of complex features.

Linear Convolution code and output:

Code:

```
import numpy as np
def linear_convolution(x, h):
    m = len(x)
    n = len(h)
    l = m + n - 1
    y = np.zeros(l)
    for i in range(l):
        y[i] = 0
        for k in range(m):
            if i - k >= 0 and i - k < n:
                y[i] += x[k] * h[i - k]

    return y

if __name__ == '__main__':
    x = list(map(float, input('Enter the input sequence x(n): ').split()))
    h = list(map(float, input('Enter the impulse response h(n): ').split()))
    y = linear_convolution(x, h)
    print('The linear convolution of x(n) and h(n) is: ', y)
    print('The length of the linear convolution is: ', len(y))
```

Output:

```
Enter the input sequence x(n): 1 2 3
Enter the impulse response h(n): 5 6 7
The linear convolution of x(n) and h(n) is:  [ 5. 16. 34. 32. 21.]
The length of the linear convolution is:  5
```

Linear Convolution code and output:

Code:

```
import numpy as np

def circular_convolution(x, h):
    m = len(x)
    n = len(h)
```

```
    if m!=n:
        print('The length of the sequences must be equal')
        return

    y = np.zeros(m)
    for i in range(m):
        for k in range(m):
            y[i] += x[k] * h[(i - k) % m]

    return y

if __name__ == '__main__':
    x = list(map(float, input('Enter the input sequence x(n): ').split()))
    h = list(map(float, input('Enter the impulse response h(n): ').split()))
    y = circular_convolution(x, h)
    print('The circular convolution of x(n) and h(n) is: ', y)
    print('The length of the circular convolution is: ', len(y))
```

Output:

```
Enter the input sequence x(n): 1 2 3
Enter the impulse response h(n): 4 5 6
The circular convolution of x(n) and h(n) is:  [31. 31. 28.]
The length of the circular convolution is:  3
```

Audio Password Code:

Code:

```
# %% [markdown]
# # Low Pass Filter

# %% [markdown]
# Importing the required libraries

# %%
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as signal
import scipy.io.wavfile as wavfile
import sounddevice as sd
import warnings

# %% [markdown]
# Filtering Warnings
```

```
# %%
warnings.filterwarnings('ignore')

# %% [markdown]
# Loading the audio file in signal and sampling rate

# %%
fs, x = wavfile.read('./Audio/sample_with_noise.wav')
print('Sampling rate: ', fs)
print('Number of samples: ', x.shape[0])


# %% [markdown]
# Adding the noise to the signal

# %%
noise = np.random.normal(0, 1, x.shape)
x = x + noise

# %% [markdown]
# Playing the noisy signal

# %%
print("Playing the original signal with noise...")
sd.play(x, fs)
sd.wait()

# %% [markdown]
# Save the noisy signal

# %%
wavfile.write('./Audio/sample_with_noise.wav', fs, x)

# %% [markdown]
# Declaring the filter parameters

# %%
Fpass = 4000.0
Fstop = 6000.0
Fs = 44000.0
N = 101

# %% [markdown]
# Making the impulse response of the filter

# %%
h = signal.remez(N, [0, Fpass, Fstop, 0.5 * Fs], [1, 0], Hz=Fs)
```

```
# %% [markdown]
# Performing Convolution of the signal and the impulse response

# %%
y = signal.lfilter(h, 1, x)

# %% [markdown]
# Playing the filtered signal

# %%
print("Playing the filtered signal...")
sd.play(y, fs)
sd.wait()

# %% [markdown]
# Save the filtered signal

# %%
wavfile.write('./Audio/sample_filtered.wav', fs, y.astype(np.int16))
```

Output:

The Audio files have been uploaded to moodle.

Conclusion:

I learned discrete time convolution operations and their application in audio noise filtering.