Experiment-2

AIM: Simulate linear convolution and circular convolution on discrete time signals.

Theory:

Linear convolution and circular convolution are mathematical operations used to combine two signals to obtain a third signal. They are widely used in various applications, such as signal processing, image processing, and audio processing.

Linear convolution calculates the sum of element-wise products of two signals, considering the full range of valid indices. It is typically used for finite-length signals and can produce an output signal that is longer than the input signals.

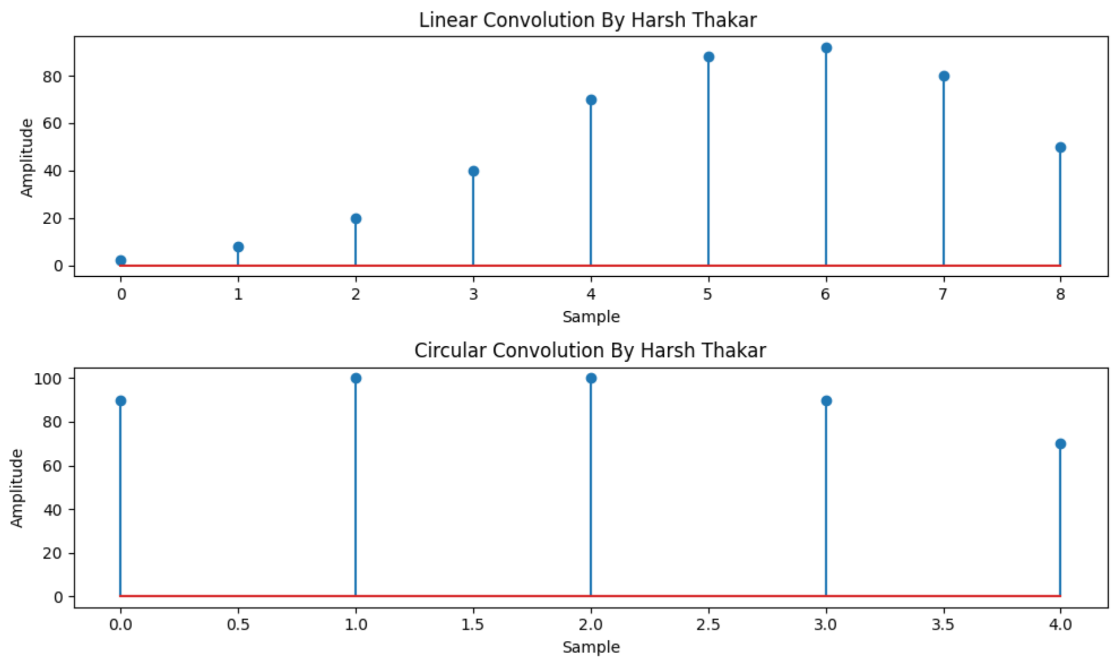
Circular convolution, on the other hand, calculates the sum of element-wise products of two signals, considering a periodic extension of the input signals. It is commonly used for periodic or infinite-length signals and produces an output signal with the same length as the input signals.

Program

```
import numpy as np

import matplotlib.pyplot as plt

def linear_convolution(signal1, signal2):

    # Compute the linear convolution

    linear_conv = np.convolve(signal1, signal2, mode='full')

    return linear_conv

def circular_convolution(signal1, signal2):

    # Compute the circular convolution

If( len(signal1) > len(signal2) ):

fft_length=len(signal1);

else:

fft_length=len(signal2);

    fft_signal1 = np.fft.fft(signal1, fft_length)

    fft_signal2 = np.fft.fft(signal2, fft_length)

circular_conv = np.fft.ifft(fft_signal1 * fft_signal2)

    return circular_conv

# Define the discrete-time signals
```

```
signal1 = np.array([1, 2, 3, 4, 5])
signal2 = np.array([2, 4, 6, 8, 10])
# Compute the linear convolution
linear_conv = linear_convolution(signal1, signal2)
# Compute the circular convolution
circular_conv = circular_convolution(signal1, signal2)
# Plot the linear and circular convolution results
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
plt.stem(linear_conv)
plt.title('Linear Convolution')
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.subplot(2, 1, 2)
plt.stem(circular_conv)
plt.title('Circular Convolution')
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.tight_layout()
plt.show()
```

Output



Conclusion

This program successfully demonstrates the difference between linear and circular convolution for discrete-time signals using Python. The linear_convolution function uses numpy.convolve to compute the full convolution without periodic wrapping, while the circular_convolution function performs convolution through the FFT–IFFT method, where both signals are zero-padded to the same length before transformation. The results are plotted using stem plots, clearly showing that linear convolution produces a longer output representing the full overlap of signals, whereas circular convolution wraps around and produces a periodic result. This experiment helps visualize how convolution works in both domains and highlights the practical use of FFT in signal processing.