# 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