**Aim:** Simulate Cross-Correlation and Autocorrelation on Discrete Time Signals.

**Theory:-**

* Cross-correlation and autocorrelation are mathematical operations used to measure the similarity or correlation between two signals. They are widely used in various applications, such as signal processing, image processing, and pattern recognition.
* Cross-correlation measures the similarity between two signals at different time shifts. It computes the dot product of one signal with a time-shifted version of the other signal. The resulting cross-correlation signal
* indicates the similarity between the two signals at different time lags.
* Autocorrelation, on the other hand, measures the similarity of a signal with a time-shifted version of itself. It computes the cross-correlation of a signal with itself. The autocorrelation signal shows how the signal is correlated with itself at different time Signals.

**Programm:-**

import matplotlib.pyplot as plt

import numpy as np

def cross\_correlation(signal1, signal2):

# Compute the cross-correlation

cross\_corr = np.correlate(signal1, signal2, mode='full')

return cross\_corr

def autocorrelation(signal):

# Compute the autocorrelation

auto\_corr = np.correlate(signal, signal, mode='full')

return auto\_corr

# Define the discrete-time signals

signal1 = np.array([1, 2, 3, 4, 5])

signal2 = np.array([2, 4, 6, 8, 10])

# Compute the cross-correlation

cross\_corr = cross\_correlation(signal1, signal2)

# Compute the autocorrelation

auto\_corr = autocorrelation(signal1)

# Create the time lags for plotting

lags\_cross = np.arange(-len(signal1) + 1, len(signal2))

lags\_auto = np.arange(-len(signal1) + 1, len(signal1))

# Plot the cross-correlation and autocorrelation signals

plt.figure(figsize=(10, 6))

plt.subplot(2, 1, 1)

plt.stem(lags\_cross, cross\_corr)

plt.title('Cross-correlation')

plt.xlabel('Time Lag')

plt.ylabel('Magnitude')

plt.subplot(2, 1, 2)

plt.stem(lags\_auto, auto\_corr)

plt.title('Autocorrelation')

plt.xlabel('Time Lag')

plt.ylabel('Magnitude')

plt.tight\_layout()

plt.savefig("./Corelation.png")

plt.show()

**Output :-**

