CSE 220 Offline 4

Discrete Fourier Transfom & FFT

1 Task 1

1.1 Overview

In task 1, we will explore how the cross-correlation between two discrete signals can be efficiently calculated using the Discrete Fourier Transform (DFT). Using DFT for cross-correlation offers significant computational advantages. Instead of performing a direct time-domain correlation, which involves computationally expensive operations, we can use the frequency domain properties of the DFT.

1.2 Background

1.2.1 DFT

The Discrete Fourier Transform (DFT) is a mathematical technique used to transform a discrete-time signal from the time domain to the frequency domain. It decomposes the signal into a sum of sinusoidal components, each with a specific frequency, amplitude, and phase. The DFT provides a representation of how the signal's energy is distributed across different frequency components, which is essential for many applications such as signal analysis, filtering, and compression. The Discrete Fourier Transform (DFT) of a discrete-time signal $x(n)$ is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}kn}$$

where:

- $x(n)$ is the input signal in the time domain (with n as the sample index),

- $X(k)$ is the frequency-domain representation of the signal (with k as the frequency index),
- N is the total number of samples in the signal,
- j is the imaginary unit ($j = \sqrt{-1}$),
- $e^{-j\frac{2\pi}{N}kn}$ is the complex exponential term representing the frequency components of the signal.

Similarly, the Inverse Discrete Fourier Transform (IDFT) is used to reconstruct a signal from its frequency-domain representation:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) \cdot e^{j\frac{2\pi}{N}kn}$$

where:

- $x(n)$ is the reconstructed signal in the time domain at sample n,
- $X(k)$ is the frequency-domain representation of the signal at frequency index k,
- N is the total number of samples,
- j is the imaginary unit ($j = \sqrt{-1}$),
- $e^{j\frac{2\pi}{N}kn}$ is the complex exponential term.

1.2.2 Cross Correlation

Cross-correlation is a measure of similarity between two signals as one is shifted relative to the other. It is used to find the time-lag at which the signals best align, which can reveal time shifts, detect patterns, or analyze the relationship between signals. Mathematically, cross-correlation involves shifting one signal over another and calculating the sum of their point-wise products at each shift.

1.3 Problem Description

In this problem, we explore how to use the Discrete Fourier Transform (DFT) and its properties to calculate the cross-correlation between two discrete signals to find the time lag between them. This method is useful in signal processing to determine the alignment and similarity of signals that may be shifted in time and contaminated with noise.

Scenario

Imagine that an earthquake occurs, and two monitoring stations, Station A and Station B, record the seismic wave generated by the event:

- **Station A** records the seismic wave as Signal A.
- **Station B** records a similar signal (Signal B) that is time-shifted and contains added noise due to environmental factors and distance differences.

Objective

The task is to find the time lag between the two signals, which represents the delay between the reception of the signals at the two stations. This is accomplished using the DFT-based method to compute cross-correlation efficiently. The time lag is then used to estimate the distance between the stations, assuming a known wave velocity.

Main Tasks to Implement

1. **Signal Acquisition**:
    - Signal A and Signal B are generated to represent the seismic waves recorded at both stations. Signal B is created by introducing a random time shift to Signal A and adding noise to simulate real-world conditions. Both signals are generated and sampled from a sinusoidal signal. This part has already been given in the sample code.
2. **DFT Computation**:
    - The DFT of both signals is computed to transform them from the time domain to the frequency domain. **You cannot use any built in package for DFT and IDFT computation**
3. **IDFT Computation**:
    - The IDFT of both signals is computed to transform them from the freq domain back to the time domain. **You cannot use any built in package for DFT and IDFT computation**

4. **Cross-Correlation Calculation**:

   - Cross-correlation must be determined using DFT properties. It can be calculated by Using DFT of both signals and finally calculating IDFT of the desired DFT output. You will then need to consider only the real part of the IDFT result. . **You cannot use any package to directly calculate cross-correlation**

5. **Sample Lag Detection**:

   - You will need to determine the sample lag from the calculated cross-correlation output, and finally use this result to calculate the delay in signal reception between the two stations. The index of the maximum psoitive value in the cross-correlation output indicates the sample lag.

6. **Distance Estimation**:

   - The distance between the stations is estimated using the time lag and the velocity of the wave signal, where the wave velocity is known based on the type of seismic wave (e.g., P-wave or S-wave). For P-wave, the velocity is 8 km/s
     - Formula:

$$\text{Distance} = |\text{Sample lag}| \times \left(\frac{1}{\text{Sampling rate}}\right) \times \text{Wave velocity}$$

Plotting Requirements

As part of the analysis, students are required to create visual representations to better understand the data and the results. The following plots must be generated:

1. **Signal A and Signal B**:

   - Plot the amplitude of both signals over sample(n) to visualize how the signals vary at each station.
   - Label the axes appropriately: the x-axis should represent sample number (n), and the y-axis should represent amplitude.
   - Use different colors or markers to differentiate between Signal A and Signal B.

2. **Magnitude Spectrum of Signal A and Signal B**:

   - Plot the magnitude of amplitude of the spectrum of both the signals over the sample indices to visualize how the spectrum vary at each station.
   - Label the axes appropriately: the x-axis should represent sample indices, and the y-axis should represent amplitude.
   - Use different colors or markers to differentiate between Signal A and Signal B spectrum.

3. **Cross-Correlation Plot**:

   - Plot the cross-correlation function, showing the correlation values at different sample lags.
   - Label the x-axis as "Lag (samples)" and the y-axis as "Correlation".

Important Note

It can be observed that even when working with noisy Signal A and a shifted noisy Signal B, the cross-correlation method will still accurately estimate the sample lag in most cases. Why?**Filtering the noise is not mandatory for this assignment**

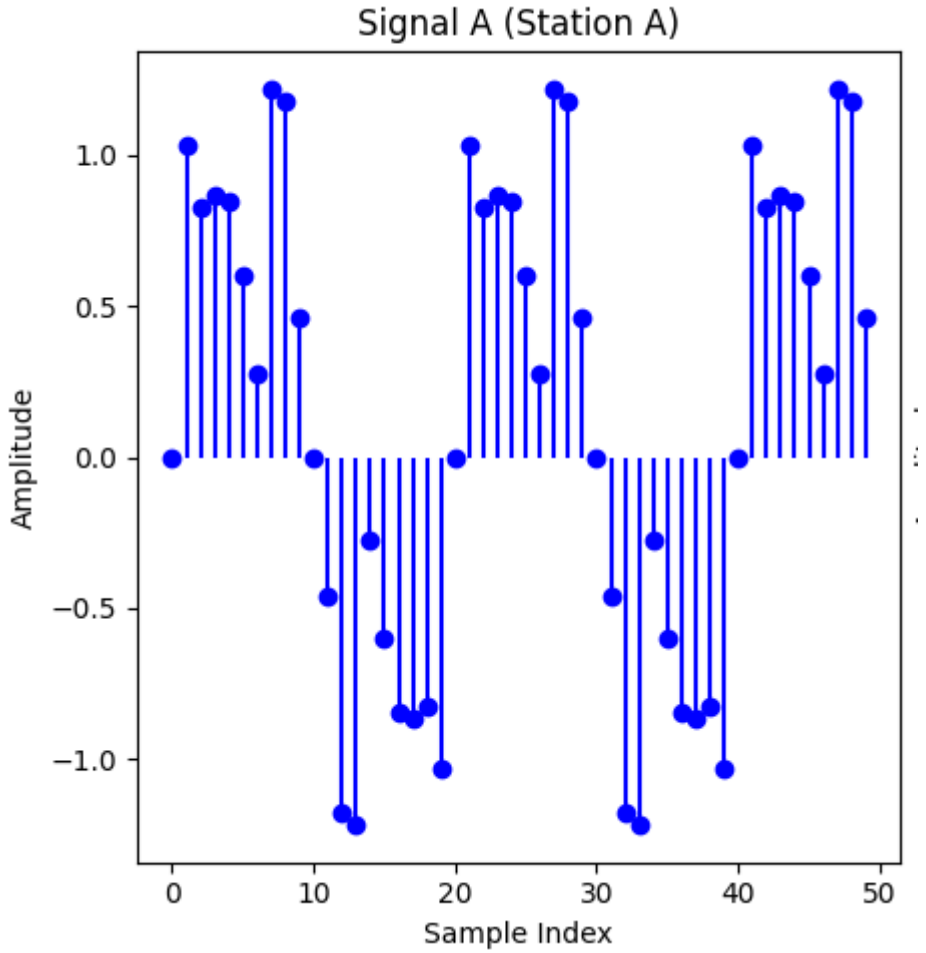
Bonus Task

Experiment with the noise frequencies, noise amplitudes and apply filtering to the noisy signal and explain the impact of filtering. Moreover, you can introduce more noise to check whether cross-correlation still works or not.

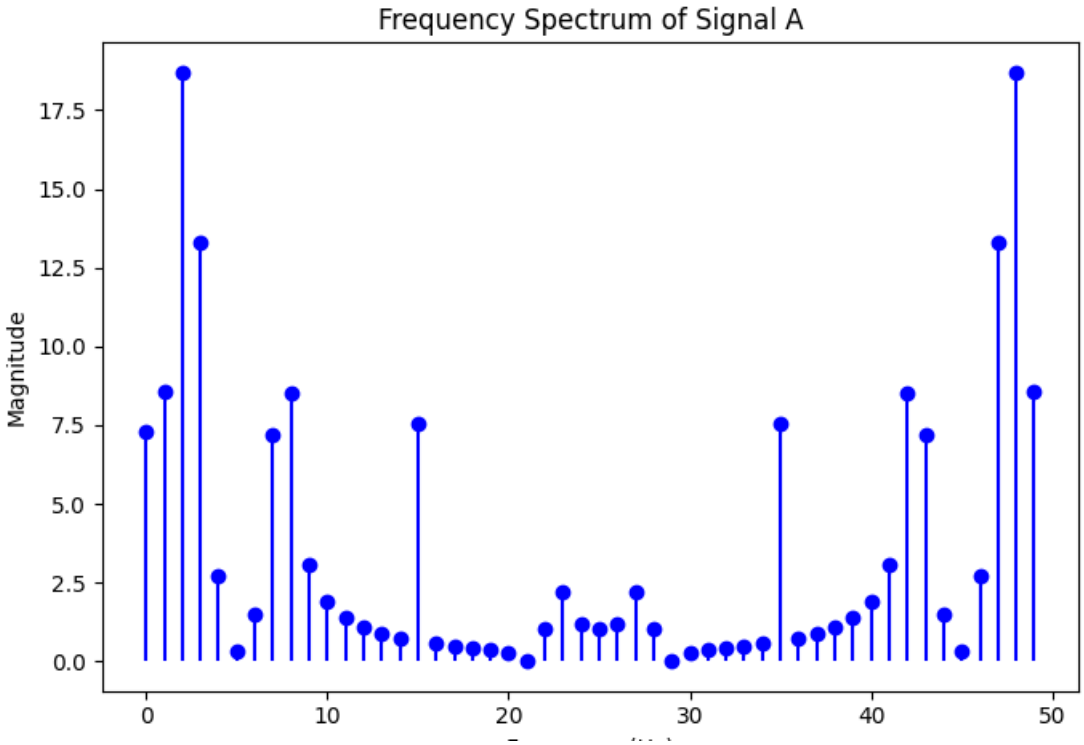
NB: Follow the template code for signal generation. Add necessary functionalities to complete the tasks. For bonus task, you are allowed to change the noise portion for experimenting and are allowed to add extra functions for filtering.

1.4 Sample Plots

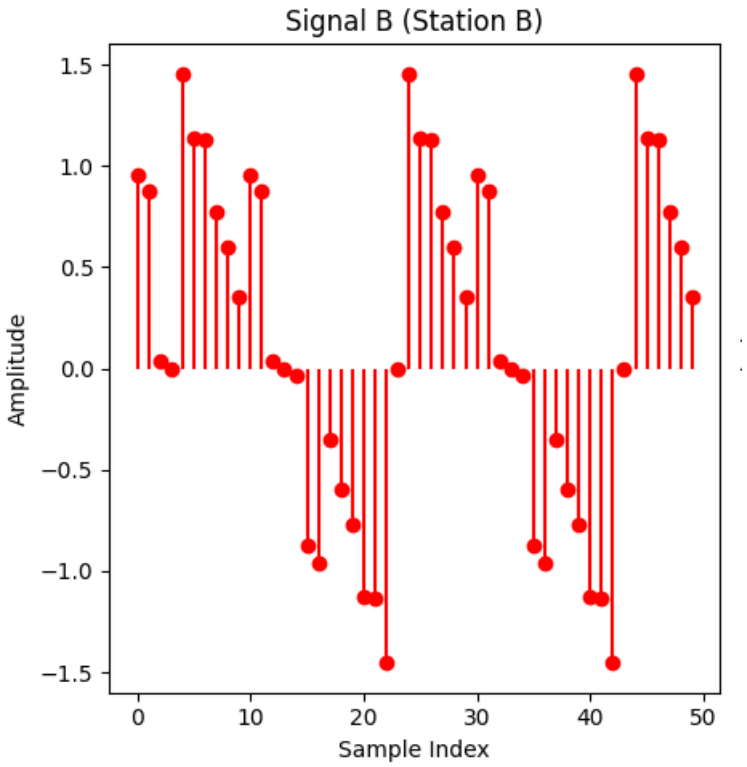
Signal A



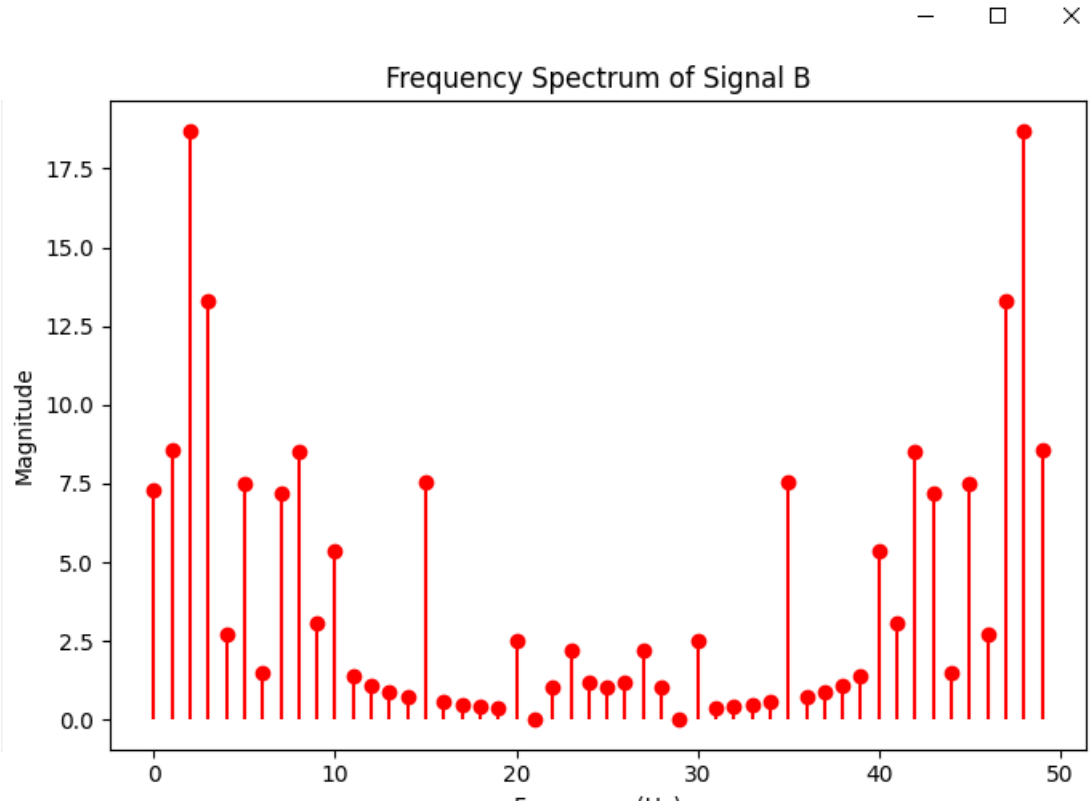
Spectrum of Signal A



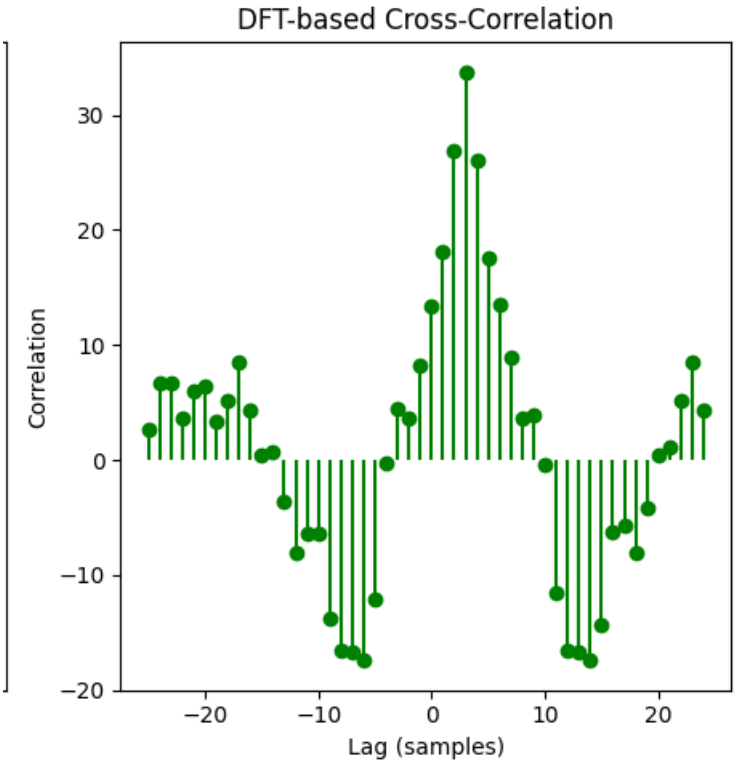
Signal B


Signal B (Station B)
Amplitude
Sample Index
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
10
20
30
40
50

Spectrum of Signal B



Cross-Corelation



2 Task 2

2.1 Overview

In task 2, we will explore FFT and DFT. The Fast Fourier Transform (FFT) is an efficient algorithm to compute the DFT. FFT reduces the time complexity of calculating the DFT from $O(n^2)$ to $O(nlogn)$ making it far more suitable for large-scale signal processing. The most widely used FFT algorithm is the Cooley-Tukey algorithm, which recursively divides the DFT into smaller DFTs, exploiting symmetries and reducing computational complexity.

2.2 Problem Description

In this task, we aim to explore and compare the computational performance of the Discrete Fourier Transform (DFT) and the Fast Fourier Transform (FFT). The objective is to implement both DFT and FFT algorithms and measure their runtime for various input sizes. The discrete signals will be generated randomly with the number of samples n ranging from 4, 8, 16, 32 ..., (powers of 2).

Goals

- **Generate random discrete signals**: Create sequences of length n (where $n = 2^k$ for $k \in \{2, 3, \ldots, \}$).
- **Implement DFT and FFT**: Develop and apply both DFT and FFT algorithms on these signals.
- **Implement IDFT and IFFT**: Compute the IDFT and IFFT to reconstruct the original signals from the frequency domain.
- **Compare runtimes**: Measure and plot the runtime of DFT/FFT and IDFT/IFFT for each input size to illustrate the efficiency difference.**For each number of samples(n), run each algortihm at least 10 times and take average of those execution times. Use the average values for plotting**

2.3 Expected Outcome

By analyzing the runtime data, you will be able to know the computational advantage of FFT over the naive DFT, particularly as the input

size n increases. This analysis will highlight the $O(N^2)$ time complexity of DFT/IDFT versus the $O(N \log N)$ time complexity of FFT/IFFT. The reconstructed signals will also verify the correctness of both transforms.

NB: No template file is given for this task

3 Submission

Submit a zip file containing the python files for the tasks. Rename it with your student id and then submit the zip file.

4 Marks Distribution

Problem Type	**Criteria**	**Marks**
Task 1	DFT implementation	15
	IDFT implementation	15
	Cross correlation calculation	10
	Distance Estimation	10
	Plotting	10
	Bonus	10
Task 2	Proper Signal Generation	5
	FFT implementation	10
	IFFT implementation	10
	Execution time Calculation	5
	Plotting	10
	Total	**110**

Table 1: Marks distribution for Assignment 4.