

 Marwadi University	Marwari University Faculty of Technology Department of Information and Communication Technology	
Subject: Digital Signal and Image Processing(01CT0513)	Aim: Perform FFT and IFFT on Discrete Time Signal.	
Experiment No: 06	Date: 09-09-2024	Enrollment No: 92200133030

Aim: Perform FFT and IFFT on Discrete Time Signal.

Theory:-

- The Fast Fourier Transform (FFT) is an algorithm used to efficiently compute the Discrete Fourier Transform (DFT) of a sequence or signal. It converts a time-domain signal into its frequency-domain representation, revealing the signal's frequency components.
- The Inverse Fast Fourier Transform (IFFT) is the reverse process of the FFT. It transforms the frequency-domain representation back into the time-domain signal.
- The FFT and IFFT are widely used in various applications, including signal processing, image processing, audio processing, and communication systems.

Programm:-

```
import matplotlib.pyplot as plt
import numpy as np


def plot_signal_and_spectrum(signal, spectrum):
    # Create time axis for plotting
    time_axis = np.arange(len(signal))

    # Plot the original signal
    plt.subplot(2, 1, 1)
    plt.plot(time_axis, signal)
    plt.title('Original Signal')
    plt.xlabel('Time')
    plt.ylabel('Amplitude')

    # Plot the magnitude spectrum
    plt.subplot(2, 1, 2)
    plt.plot(time_axis, spectrum)
    plt.title('Magnitude Spectrum')
    plt.xlabel('Frequency')
    plt.ylabel('Magnitude')

    plt.tight_layout()
    plt.show()

# Define the discrete-time signal
time = np.linspace(0, 1, 500)
frequency1 = 5 # Frequency of the first sinusoidal component
frequency2 = 20 # Frequency of the second sinusoidal component
```

 Marwadi University	Marwari University Faculty of Technology Department of Information and Communication Technology	
Subject: Digital Signal and Image Processing(01CT0513)	Aim: Perform FFT and IFFT on Discrete Time Signal.	
Experiment No: 06	Date: 09-09-2024	Enrollment No: 92200133030

```

amplitude1 = 1 # Amplitude of the first sinusoidal component
amplitude2 = 0.5 # Amplitude of the second sinusoidal component
signal = amplitude1 * np.sin(2 * np.pi * frequency1 * time) + \
    amplitude2 * np.sin(2 * np.pi * frequency2 * time)
# Compute the FFT of the signal
fft_result = np.fft.fft(signal)
# Compute the magnitude spectrum of the FFT result
magnitude_spectrum = np.abs(fft_result)
# Compute the IFFT of the FFT result
reconstructed_signal = np.fft.ifft(fft_result)
# Display the original signal, the magnitude spectrum, and the reconstructed
# signal
plot_signal_and_spectrum(signal, magnitude_spectrum)
# Save the magnitude spectrum plot (optional)
spectrum_path = 'Magnitude Spectrum.png'
plt.plot(magnitude_spectrum)
plt.title('Magnitude Spectrum')
plt.xlabel('Frequency')
plt.ylabel('Magnitude')
plt.savefig(spectrum_path)
print(f"Magnitude spectrum plot saved at: {spectrum_path}")
plt.show()

```

Output :-

