Aim ↬ To understand and implement the Discrete Fourier Transform (DFT) and its inverse for analyzing and synthesizing discrete-time signals using MATLAB.

Software Required ↬ MATLAB

Theory ↬

The Discrete Fourier Transform (DFT) is a powerful signal processing tool that converts a finite discrete-time signal from the time domain into the frequency domain. It is widely used in various applications, such as digital signal processing, image analysis, and communications.

The DFT transforms a discrete-time signal x[n] with N samples into its frequency-domain representation X[k]. The following formula defines the DFT:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-jk\Omega_0 n}$$

where:

- X[k] is the DFT of x[n], representing the signal in the frequency domain.
- k is the frequency index, ranging from 0 to N−1.
- $\Omega_0 = \frac{2\pi}{N}$ is the fundamental frequency.

The DFT provides a discrete frequency spectrum of the signal, which reveals the amplitude and phase information of different frequency components. It is particularly useful in analyzing periodic signals and performing operations like filtering and spectral analysis.

DFT analyzes finite-length signals and gives a discrete frequency spectrum, while DTFT handles infinite-length signals and provides a continuous frequency spectrum. DFT is practical for digital computations, whereas DTFT is more theoretical.

Properties of DFT ↴

1] Linearity ↪

$$ax_1[n] + bx_2[n] \leftrightarrow aX_1[k] + bX_2[k]$$

2] Time Shifting ↪

$$x[n - n_0] \leftrightarrow X[k] e^{-jk\Omega n_0}$$

3] Frequency Shifting ↪

$$x[n]e^{jk_0\Omega n} \leftrightarrow X[k-k_0]$$

4] Time Reversal ↪

$$x[-n] \leftrightarrow X[-k]$$

5] Convolution ↪

$$x[n] * h[n] \leftrightarrow X[k] \cdot H[k]$$

6] Multiplication in Time Domain ↪

$$x[n].h[n] \leftrightarrow \frac{1}{N}\sum_{m=0}^{N-1} X[m]H[k-m]$$

7] Conjugation ↪

$$x^*[n] \leftrightarrow X^*[-k]$$

8] Duality ↪

$$x[n] \leftrightarrow X[k] \qquad \text{implies} \qquad X[n] \leftrightarrow x[-k]$$

9] Difference in Time Domain ↪

$$x[n] - x[n-1] \leftrightarrow \left(1 - e^{-j\Omega_0 k}\right)X[k]$$

10] Summation in Time Domain ↪

$$\sum_{n=0}^{N-1} x[n] \leftrightarrow \frac{X[k]}{j\Omega_0 k}$$

11] Parseval's Power Property ↪

$$P = \frac{1}{N}\sum_{n=0}^{N-1} |x[n]|^2 = \sum_{k=0}^{N-1} |X[k]|^2$$

Inverse Discrete Fourier Transform (IDFT) ↴

The Inverse Discrete Fourier Transform (IDFT) reconstructs the original discrete-time signal from its frequency-domain representation X[k]. The IDFT is given by:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{jk\Omega_0 n}$$

The IDFT allows us to synthesize the original signal from its frequency-domain representation, demonstrating that the transformation process is reversible.

Code ↬

```
% Discrete Fourier Transform
x_n = [1, 2, 1, 2];
N = length(x_n);

stem(0:N-1, x_n, 'filled'); xlabel('n'); ylabel('x[n]'); title('Original Signal');

% DFT using built-in functions
X_k = fft(x_n);

mag_X_k = abs(X_k);
phs_X_k = angle(X_k);

figure;
subplot(2,2,1); stem(0:N-1, X_k, 'filled'); xlabel('k'); ylabel('X[k]'); title('Discrete Fourier
Transform');
subplot(2,2,2); stem(0:N-1, mag_X_k, 'filled'); xlabel('k'); ylabel('|X[k]|'); title('Magnitude
Spectrum');
subplot(2,2,3); stem(0:N-1, phs_X_k, 'filled'); xlabel('k'); ylabel('∠X[k]'); title('Phase
Spectrum');

% IDFT using built-in functions
x_n_remake = ifft(X_k);

subplot(2,2,4); stem(0:N-1, x_n_remake, 'filled'); xlabel('n'); ylabel('x[n]');
title('Reconstructed Signal');

% DFT using Formula
F_k = zeros(1,N);

for k = 0:N-1
   for n = 0:N-1
      F_k(k+1) = F_k(k+1) + x_n(n+1)*exp(-1i * 2 * pi * k *n /N);
   end
end

mag_F_k = abs(F_k);
phs_F_k = angle(F_k);

figure;
```

```
subplot(2,2,1); stem(0:N-1, F_k, 'filled'); xlabel('k'); ylabel('F[k]'); title('Discrete Fourier Transform');
subplot(2,2,2); stem(0:N-1, mag_F_k, 'filled'); xlabel('k'); ylabel('|F[k]|'); title('Magnitude Spectrum');
subplot(2,2,3); stem(0:N-1, phs_F_k, 'filled'); xlabel('k'); ylabel('∠F[k]'); title('Phase Spectrum');

% IDFT using Formula
f_n = zeros(1,N);

for n = 0:N-1
    for k = 0:N-1
        f_n(n+1) = f_n(n+1) + (1/N)*F_k(k+1)*exp(1i * 2 * pi * k *n /N);
    end
end

subplot(2,2,4); stem(0:N-1, f_n, 'filled'); xlabel('n'); ylabel('x[n]'); title('Reconstructed Signal');
```

Output ↬

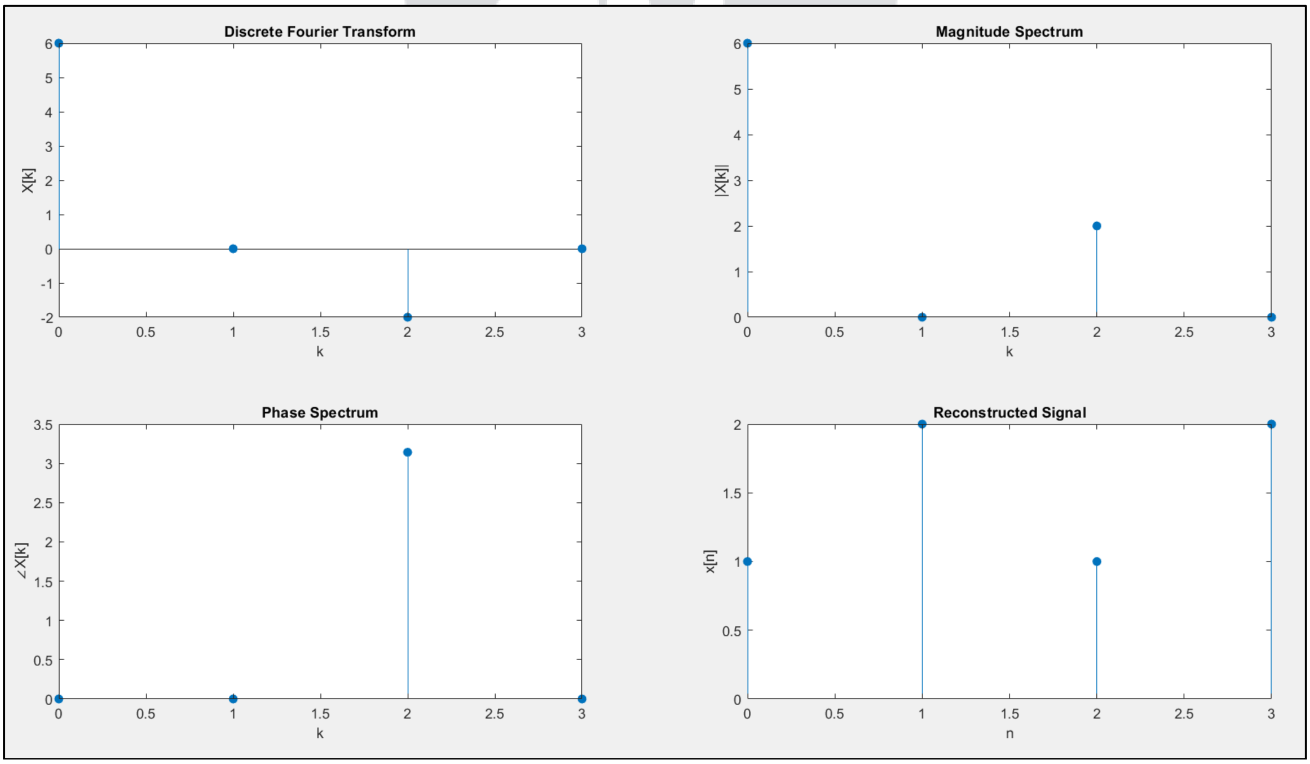

Fig. i) Plots for DFT and IDFT using built-in functions

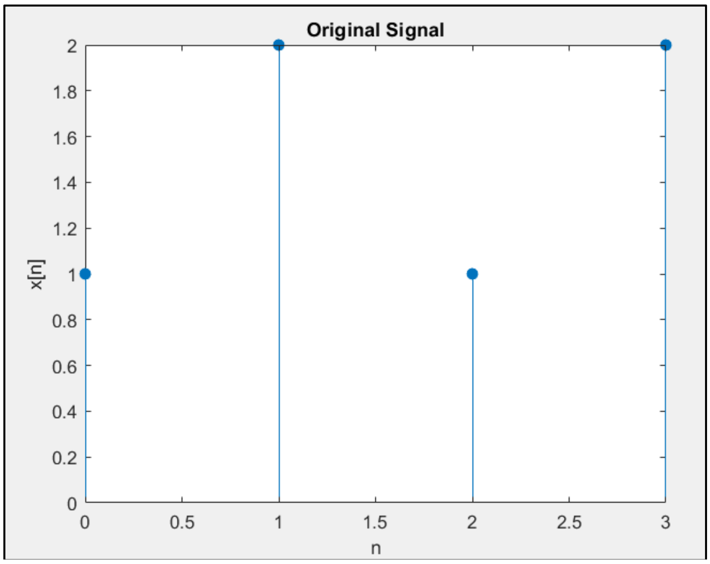

Fig. ii) Original input signal

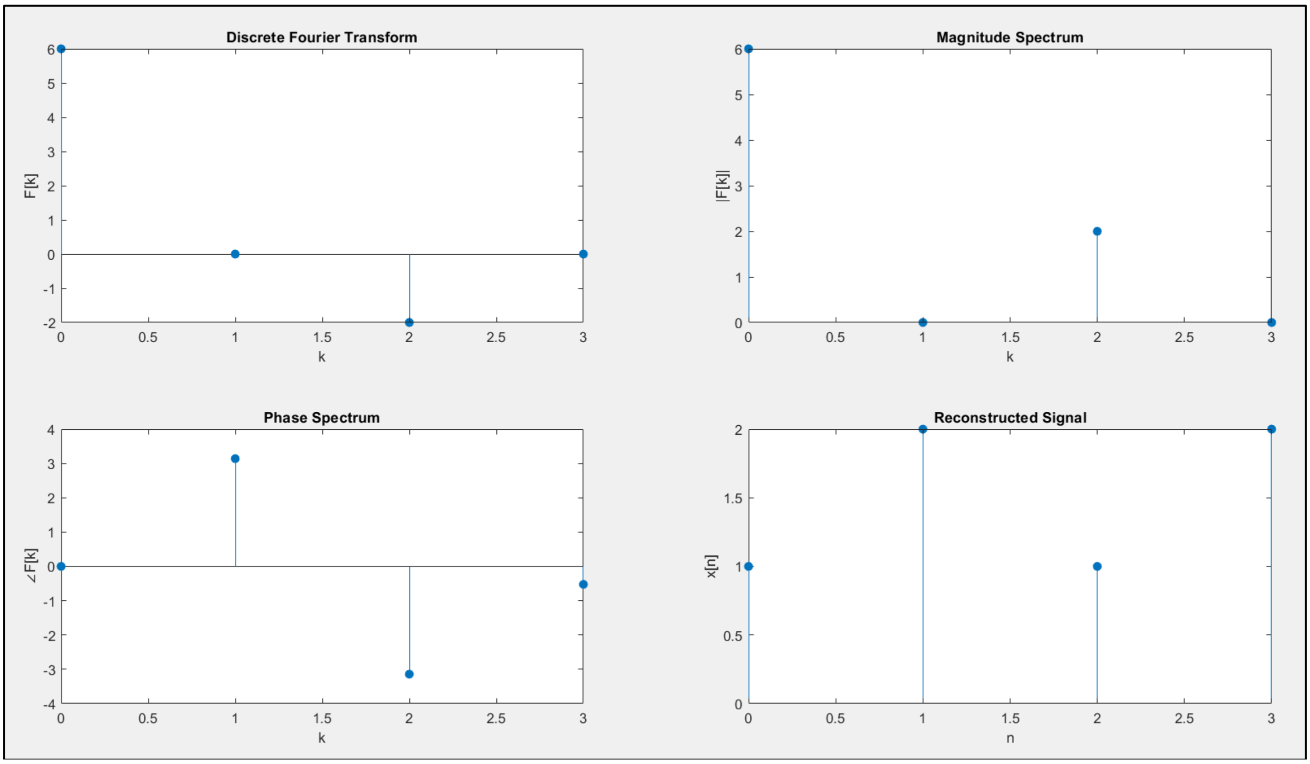

Fig. iii) Plots for DFT and IDFT using formulas

Result ↬

The Discrete Fourier Transform and its inverse were accurately implemented, allowing for effective frequency-domain analysis and signal reconstruction. The MATLAB results confirmed the proper application of the DFT in processing discrete-time signals.

Conclusion ↬

The Discrete Fourier Transform is essential for analyzing discrete-time signals in the frequency domain. It provides insights into signal frequency components and allows for efficient signal processing.

Precautions ↬

- Ensure the correct number of samples N for accurate results.
- Verify the periodicity of signals to avoid aliasing effects.
- Check the reconstruction accuracy by comparing the original and reconstructed signals.