National Institute of Technology
Calicut

Department of Electronics and Communication Engineering

Digital Signal Processing LAB

Week 5 Report

Date: 21–08–2025
Instructor: Dr. Sakthivel V

Submitted by:
Mummana Jagadeesh B231113EC
Muhammed Nihal MP B230437EC

Aim

To implement and analyze the Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) of discrete-time signals using MATLAB, verify the results using the inverse FFT (IFFT), and plot the magnitude and phase spectrum.

Formulae

- Discrete Fourier Transform (DFT) for a sequence $x[n]$ of length N:

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, \ldots, N-1$$

- Inverse Discrete Fourier Transform (IDFT) to recover $x[n]$ from $X[k]$:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\, e^{j\frac{2\pi}{N}kn}, \quad n = 0, 1, \ldots, N-1$$

- The complex rotation factor, often called the **twiddle factor**, is defined as:

$$W_N = e^{-j\frac{2\pi}{N}}$$

Theory

- **Discrete Fourier Transform (DFT):** The DFT converts a discrete-time sequence $x[n]$ of length N into its frequency domain representation. It is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, \ldots, N-1$$

- **Inverse Discrete Fourier Transform (IDFT):** The IDFT allows recovery of the original time-domain sequence from its DFT values. It is expressed as:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\, e^{j\frac{2\pi}{N}kn}, \quad n = 0, 1, \ldots, N-1$$

- **Fast Fourier Transform (FFT):** The FFT is an efficient algorithm to compute the DFT. While the direct computation of the DFT requires $O(N^2)$ operations, FFT reduces the complexity to $O(N \log N)$ by exploiting symmetry and periodicity properties of the twiddle factors:

$$W_N = e^{-j\frac{2\pi}{N}}$$

- **Decimation in Time FFT (DIT-FFT):** In the DIT-FFT approach, the input sequence $x[n]$ is divided into even and odd indexed samples. The DFT is recursively computed as:

$$X[k] = X_{\text{even}}[k] + W_N^k\, X_{\text{odd}}[k], \quad k = 0, 1, \ldots, N-1$$

- **Decimation in Frequency FFT (DIF-FFT):** In the DIF-FFT method, the decomposition is applied in the frequency domain rather than the time domain. The general recursive relation is given by:

$$X[k] = G[k] + W_N^k \, H[k], \quad k = 0, 1, \ldots, N/2 - 1$$

where $G[k]$ and $H[k]$ correspond to the partial frequency-domain results.

QUESTION 1

Given the discrete-time sequence:

$$x(n) = \left\{\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}, 0, 0, 0, 0\right\}$$

Compute the 8-point Discrete Fourier Transform (DFT) of $x(n)$ using:

1. Decimation-in-Frequency FFT (DIF-FFT) algorithm
2. Decimation-in-Time FFT (DIT-FFT) algorithm

Verify the results using the built-in `fft` command and plot both the magnitude and phase spectra.

MATLAB Code Listing

Code:

MATLAB Code

```
y = input('Enter the input sequence: ');
n = length(y);

Y_dit = myDIT(y);
Y_dif = myDIF(y);
Y_sys = fft(y);

disp('DIT FFT Result:'); disp(Y_dit.');
disp('DIF FFT Result:'); disp(Y_dif.');

disp('Inbuilt FFT Result:');
disp(Y_sys.');

k = 0:n-1;
figure;

subplot(3,2,1)
stem(k, abs(Y_dit),'filled')
title('DIT |X[k]|')
```

```
subplot(3,2,2)
stem(k, angle(Y_dit),'filled')
title('DIT  X [k]')

subplot(3,2,3)
stem(k, abs(Y_dif),'filled')
title('DIF |X[k]|')

subplot(3,2,4)
stem(k, angle(Y_dif),'filled')
title('DIF  X [k]')

subplot(3,2,5)
stem(k, abs(Y_sys),'filled')
title('Built-in |X[k]|')

subplot(3,2,6)
stem(k, angle(Y_sys),'filled')
title('Built-in  X [k]')

function Y = myDIT(seq)
    m = length(seq);
    if m == 1
        Y = seq;
    else
        e = seq(1:2:end);
        o = seq(2:2:end);
        E = myDIT(e);
        O = myDIT(o);
        W = exp(-1j*2*pi*(0:m/2-1)/m);
        Y = [E + W.*O , E - W.*O];
    end
end

function Y = myDIF(seq)
    seq = seq(:).';
    m = length(seq);
    if floor(log2(m)) ~= log2(m)
        error('Size must be power of 2')
    end
    Ytemp = difRec(seq);
    idx = 0:m-1;
    r = revBits(idx,m);
    Y = zeros(1,m);
    for p = 1:m
        Y(r(p)+1) = Ytemp(p);
```

```
    end
end

function Y = difRec(s)
    m = length(s);
    if m == 1
        Y = s;
    else
        a = s(1:m/2) + s(m/2+1:m);
        b = (s(1:m/2) - s(m/2+1:m)) .* exp(-1j*2*pi*(0:m
            /2-1)/m);
        A = difRec(a);
        B = difRec(b);
        Y = [A, B];
    end
end

function r = revBits(idx,m)
    bits = log2(m);
    r = zeros(size(idx));
    for q = 1:length(idx)
        val = idx(q); rev = 0;
        for j = 1:bits
            b = bitand(val,1);
            rev = bitor(bitshift(rev,1),b);
            val = bitshift(val,-1);
        end
        r(q) = rev;
    end
end
```

Explanation of the MATLAB Code

The given MATLAB code implements the Fast Fourier Transform (FFT) using three different approaches:

- **Decimation in Time (DIT)** FFT
- **Decimation in Frequency (DIF)** FFT
- MATLAB's built-in `fft()` function

The results of each method are compared both numerically and graphically.

Main Script

- The input sequence $y[n]$ is taken from the user.
- Its length is stored in the variable n.

- The functions `myDIT()`, `myDIF()`, and the built-in `fft()` are used to compute the Discrete Fourier Transform (DFT).
- The results are displayed in tabular form and plotted as both magnitude and phase spectra.

Decimation in Time (DIT)

The function `myDIT()` recursively divides the sequence into even and odd indexed parts:

$$X[k] = E[k] + W_N^k O[k], \quad X[k + N/2] = E[k] - W_N^k O[k]$$

where $E[k]$ and $O[k]$ are the DFTs of the even and odd subsequences respectively, and

$$W_N^k = e^{-j\frac{2\pi k}{N}}$$

is the twiddle factor. The recursion continues until the subsequences are of length 1.

Decimation in Frequency (DIF)

The function `myDIF()` uses a top-down approach, where the input sequence is split into two halves:

$$A[n] = x[n] + x[n + N/2], \quad B[n] = \big(x[n] - x[n + N/2]\big)W_N^n$$

The recursion is then applied to $A[n]$ and $B[n]$. At the end, bit-reversal ordering is applied using the function `revBits()` to arrange the outputs in the correct order.

Built-in FFT

MATLAB's built-in `fft()` function directly computes the DFT using an optimized FFT algorithm. This serves as a reference for validating the DIT and DIF implementations.

Plotting Results

Subplots are generated to compare:

- DIT FFT magnitude and phase
- DIF FFT magnitude and phase
- Built-in FFT magnitude and phase

This visual comparison demonstrates that all three methods produce identical results.

Output

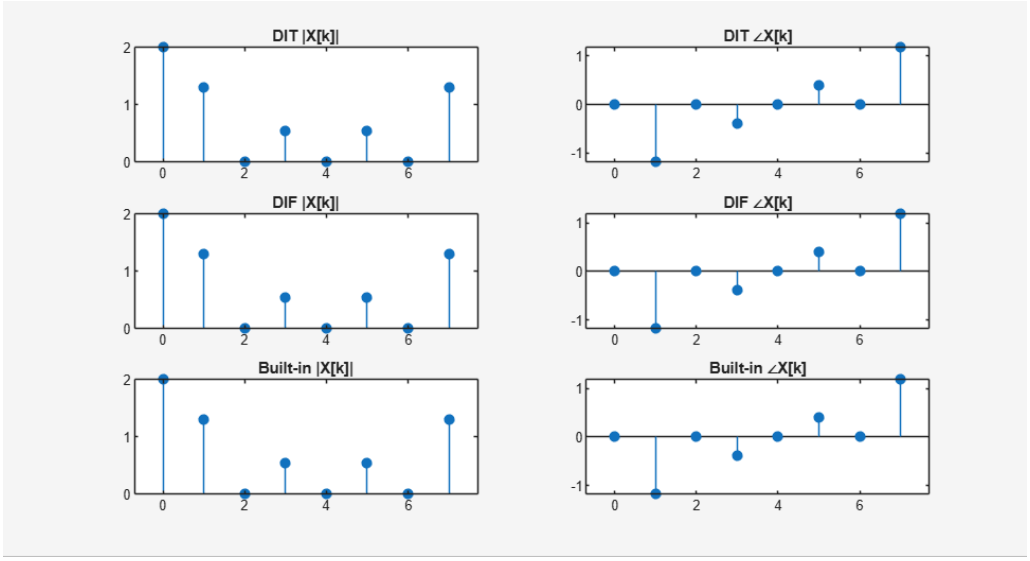
```
Enter the input sequence: [0.5 0.5 0.5 0.5 0 0 0 0]
DIT FFT Result:
   2.0000 + 0.0000i
   0.5000 - 1.2071i
   0.0000 + 0.0000i
   0.5000 - 0.2071i
   0.0000 + 0.0000i
   0.5000 + 0.2071i
   0.0000 + 0.0000i
   0.5000 + 1.2071i

DIF FFT Result:
   2.0000 + 0.0000i
   0.5000 - 1.2071i
   0.0000 + 0.0000i
   0.5000 - 0.2071i
   0.0000 + 0.0000i
   0.5000 + 0.2071i
   0.0000 + 0.0000i
   0.5000 + 1.2071i

Inbuilt FFT Result:
   2.0000 + 0.0000i
   0.5000 - 1.2071i
   0.0000 + 0.0000i
   0.5000 - 0.2071i
   0.0000 + 0.0000i
   0.5000 + 0.2071i
   0.0000 + 0.0000i
   0.5000 + 1.2071i
```

Figure 1: Command Window Output

Output Plots:



QUESTION 2

Consider the frequency-domain sequence:

$$X = \{8, 0, 0, 0, 0, 0, 0, 0\}$$

Compute the 8-point Inverse Discrete Fourier Transform (IDFT) of X using:

1. Decimation-in-Frequency IFFT (DIF-IFFT) algorithm
2. Decimation-in-Time IFFT (DIT-IFFT) algorithm

Verify the results using the built-in `ifft` command and plot the magnitude and phase spectra of the reconstructed sequence.

MATLAB Code Listing

Code:

MATLAB Code

```
X_in = input('Enter the frequency-domain sequence: ');
N = length(X_in);

x_dit = (1/N)*conj(myDIT(conj(X_in)))';
x_dif = (1/N)*conj(myDIF(conj(X_in)))';
x_sys = ifft(X_in);

m1 = abs(x_dit); p1 = angle(x_dit);
m2 = abs(x_dif); p2 = angle(x_dif);
m3 = abs(x_sys); p3 = angle(x_sys);

n = 0:N-1;
figure

subplot(3,2,1)
stem(n,m1,'filled')
title('DIT-IDFT |x[n]|')

subplot(3,2,2)
stem(n,p1,'filled')
title('DIT-IDFT  x [n]')

subplot(3,2,3)
stem(n,m2,'filled')
title('DIF-IDFT |x[n]|')

subplot(3,2,4)
stem(n,p2,'filled')
title('DIF-IDFT  x [n]')
```

```
subplot(3,2,5)
stem(n,m3,'filled')
title('IFFT |x[n]|')

subplot(3,2,6)
stem(n,p3,'filled')
title('IFFT  x [n]')

disp('IDFT using DIT:')
disp(x_dit')
disp('IDFT using DIF:')
disp(x_dif')
disp('Built-in IFFT:')
disp(x_sys)

function Y = myDIT(seq)
    L = length(seq);
    if L == 1
        Y = seq;
    else
        e = seq(1:2:end);
        o = seq(2:2:end);
        E = myDIT(e);
        O = myDIT(o);
        W = exp(-1j*2*pi*(0:L/2-1)/L);
        Y = [E+W.*O , E-W.*O];
    end
end

function Y = myDIF(seq)
    L = length(seq);
    if L == 1
        Y = seq;
    else
        a = seq(1:L/2)+seq(L/2+1:L);
        b = (seq(1:L/2)-seq(L/2+1:L)).*exp(-1j*2*pi*(0:L
            /2-1)/L);
        A = myDIF(a);
        B = myDIF(b);
        Y = [A B];
    end
end
```

Explanation of the MATLAB Code for IDFT

The given MATLAB code computes the Inverse Discrete Fourier Transform (IDFT) using three methods:

- **Decimation in Time (DIT)** approach
- **Decimation in Frequency (DIF)** approach
- MATLAB's built-in `ifft()` function

The numerical results and corresponding plots of magnitude and phase are generated for comparison.

Main Script

- The input sequence $X[k]$ in the frequency domain is provided by the user.
- Its length is stored in the variable N.
- The IDFT is computed using:

$$x[n] = \frac{1}{N} \cdot \text{conj}\left(\text{FFT}\left(\text{conj}(X[k])\right)\right), \quad n = 0, 1, \ldots, N-1$$

  This property relates the IDFT to the DFT.
- The functions `myDIT()` and `myDIF()` (same recursive algorithms used for FFT) are reused here with conjugate symmetry to calculate the IDFT.
- MATLAB's built-in `ifft()` function is also used for verification.

Mathematical Background

The Discrete Fourier Transform (DFT) is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, \ldots, N-1$$

The inverse DFT (IDFT) is given by:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\, e^{j\frac{2\pi}{N}kn}, \quad n = 0, 1, \ldots, N-1$$

Using the conjugate property, the IDFT can be computed as:

$$x[n] = \frac{1}{N} \cdot \text{conj}\left(\text{DFT}\big(\text{conj}(X[k])\big)\right)$$

DIT-IDFT

The function `myDIT()` implements a recursive Decimation in Time algorithm. The frequency-domain sequence is split into even and odd indexed parts:

$$X[k] = E[k] + W_N^k O[k], \quad X[k + N/2] = E[k] - W_N^k O[k]$$

Here, $E[k]$ and $O[k]$ represent the transforms of the even and odd samples. The same structure as FFT is used, with conjugate symmetry ensuring the inverse computation.

DIF-IDFT

The function `myDIF()` implements the Decimation in Frequency approach, where the sequence is split into two halves:

$$A[n] = x[n] + x[n + N/2], \quad B[n] = \big(x[n] - x[n + N/2]\big)W_N^n$$

Recursion is applied to $A[n]$ and $B[n]$ until sequences of length 1 are reached. This produces the IDFT through the same butterfly operations as FFT, but with conjugation.

Built-in IFFT

MATLAB's `ifft()` computes the IDFT using optimized routines. It provides a reference to validate both DIT and DIF implementations.

Plotting Results

The magnitude and phase of $x[n]$ are computed for all three methods

- DIT-based IDFT magnitude and phase
- DIF-based IDFT magnitude and phase
- Built-in IFFT magnitude and phase

The results confirm that all three approaches produce identical IDFT outputs.

Output

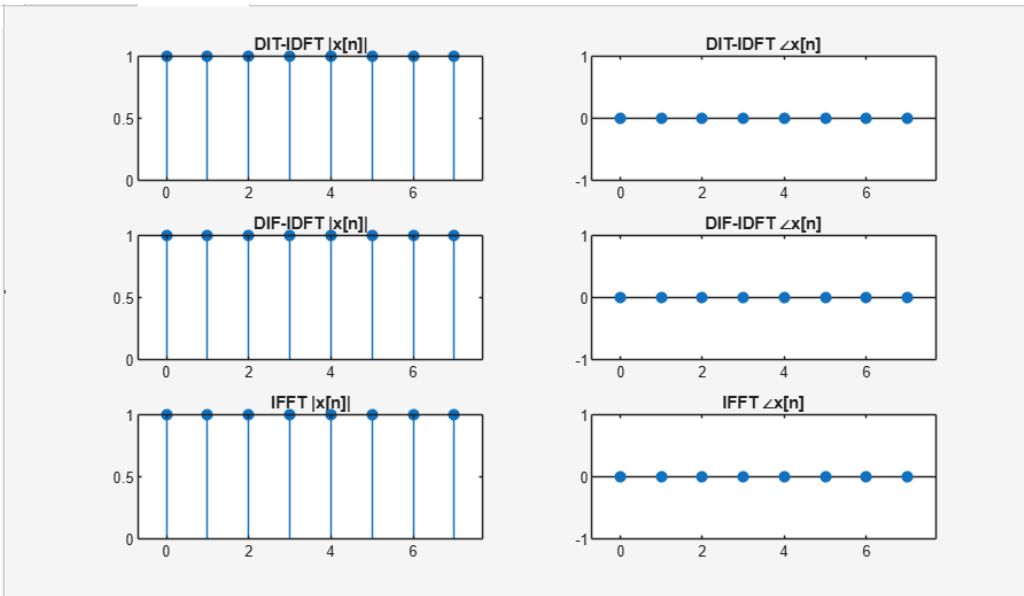
```
>> q52
Enter the frequency-domain sequence: [8 0 0 0 0 0 0 0]
IDFT using DIT:
     1     1     1     1     1     1     1     1

IDFT using DIF:
     1     1     1     1     1     1     1     1

Built-in IFFT:
     1     1     1     1     1     1     1     1
```

Figure 2: Command Window Output

Output Plots:



QUESTION 3

Elementary Signals and Their Spectra:

a) $x(t) = \delta(t)$
b) $x(t) = u(t)$
c) $x(t) = 10\sin(100\pi t)$
d) $x(t) = 2\sin(50\pi t) + 3\sin(100\pi t) + 5\sin(200\pi t) + 10\sin(500\pi t)$
e) $x(t) = e^{j120\pi t}$

For each of the above signals, sketch or plot the corresponding magnitude and phase spectra

Code:

MATLAB Code

```
expr = input('Enter the signal expression in t: ','s');

N = 512;
fs = 1000;
t = (0:N-1)/fs;

x = eval(expr);

function X = myDIT(x)
    N = length(x);
    if N == 1
        X = x;
    else
        Xe = myDIT(x(1:2:end));
        Xo = myDIT(x(2:2:end));
        k = 0:(N/2-1);
        W = exp(-2j*pi*k/N);
        X = [Xe + W.*Xo , Xe - W.*Xo];
    end
end

function Xs = shiftFFT(X)
    N = length(X);
    Xs = [X(N/2+1:end), X(1:N/2)];
end

X = myDIT(x);
Xs = shiftFFT(X);
f = (-N/2:N/2-1)*(fs/N);

figure
```

```
subplot(2,1,1)
stem(f,abs(Xs),'filled','MarkerSize',3)
title(['Magnitude Spectrum of x(t) = ' expr])
xlabel('Frequency (Hz)')
ylabel('|X(f)|')
grid on

subplot(2,1,2)
stem(f,angle(Xs),'filled','MarkerSize',3)
title(['Phase Spectrum of x(t) = ' expr])
xlabel('Frequency (Hz)')
ylabel(' X (f) (rad)')
grid on
```

Explanation of the MATLAB Code for Spectrum Analysis

This MATLAB code computes and visualizes the magnitude and phase spectrum of a given time-domain signal expression $x(t)$ using the Decimation in Time (DIT) FFT algorithm.

Main Script

- The user is prompted to enter a continuous-time signal expression, such as:

$$x(t) = \sin(2\pi 50t) + \cos(2\pi 120t)$$

- Sampling parameters are defined:

$$N = 512, \quad f_s = 1000\,\text{Hz}, \quad t = \frac{0, 1, 2, \ldots, N-1}{f_s}$$

- The signal values are evaluated using MATLAB's `eval()` function.

Decimation in Time (DIT) FFT

The function `myDIT()` recursively computes the Discrete Fourier Transform (DFT) using the radix-2 Decimation in Time approach:

$$X[k] = E[k] + W_N^k O[k], \quad X[k + N/2] = E[k] - W_N^k O[k]$$

where

$$W_N^k = e^{-j\frac{2\pi}{N}k}$$

is the twiddle factor, and $E[k], O[k]$ are the DFTs of the even and odd-indexed subsequences of $x[n]$.

Plotting Results

- The magnitude spectrum shows the distribution of frequency components in $x(t)$.
- The phase spectrum shows the phase contribution of each frequency component.

Output Plots:

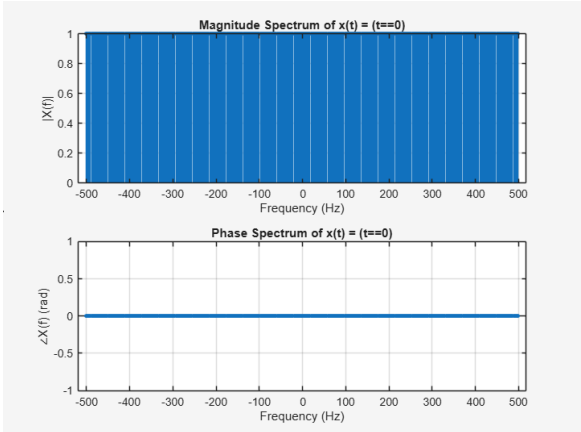


Figure 3: Delta Signal $\delta(t)$ (nonzero only at $t = 0$)

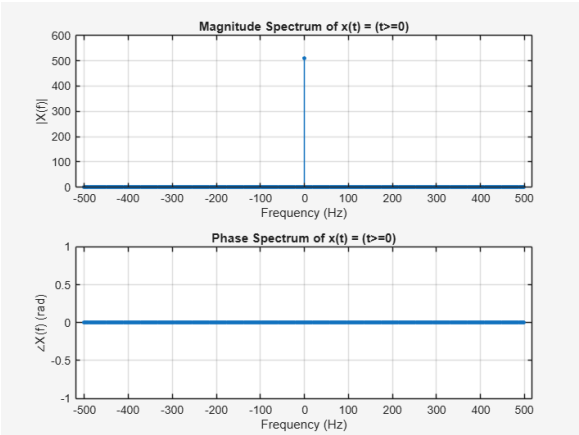


Figure 4: Unit Step Signal $u(t)$ $(t \geq 0)$

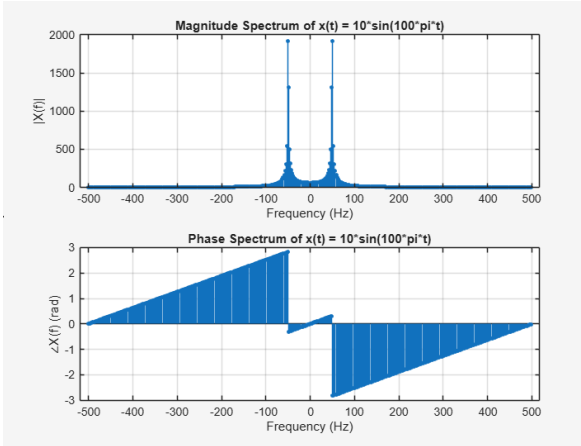


Figure 5: Sinusoidal Signal $x(t) = 10\sin(100\pi t)$

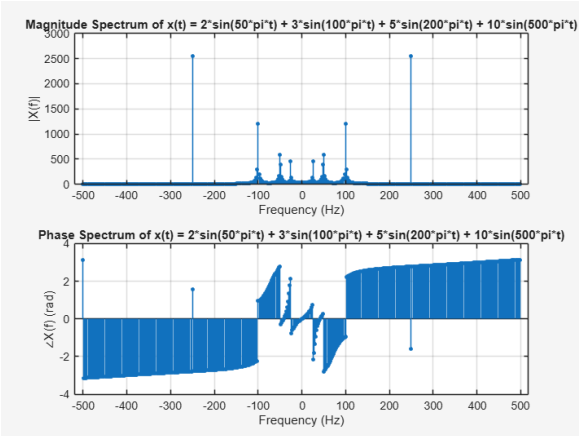


Figure 6: Sum of Sinusoids $x(t) = 2\sin(50\pi t) + 3\sin(100\pi t) + 5\sin(200\pi t) + 10\sin(500\pi t)$

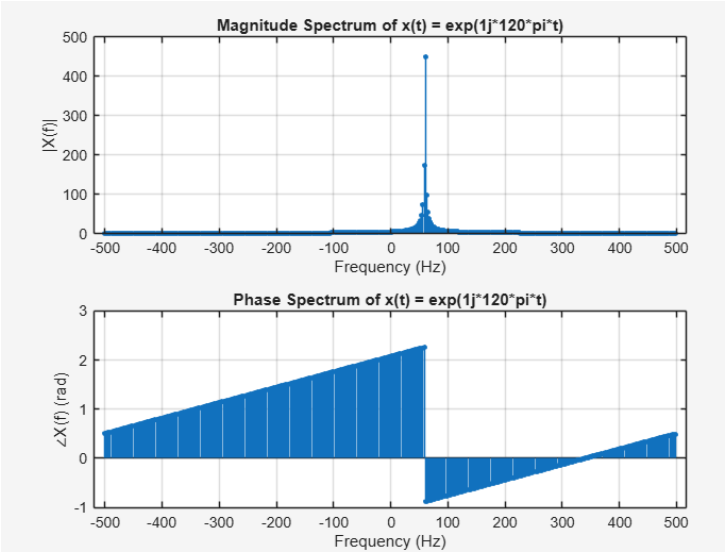


Figure 7: Complex Exponential $x(t) = e^{j120\pi t}$