DIGITAL SIGNAL PROCESSING

(DSC – 14)

Practical File

Submitted By

Goldi Kumari
Roll No.: 23294917143 (Batch: ECE B– B1)
B. Tech Electronics and Communication Engineering
(Semester V)

To

Dr. Ajay Kumar Gupta
(Assistant Professor)
Department of Electronics and Communication Engineering

FACULTY OF TECHNOLOGY
UNIVERSITY OF DELHI
NEW DELHI – 110007
(2025 – 2026)

Discrete-Time Fourier Transform of a Sequence

Aim

To compute and plot the Discrete-Time Fourier Transform (DTFT) of the discrete-time sequence

$$x[n] = \cos(0.1\pi n)$$

using the matrix multiplication approach in MATLAB.

Theory

According to Proakis and Manolakis, the Discrete-Time Fourier Transform (DTFT) of a discrete-time signal $x[n]$ is defined as

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}, \quad -\pi \le \omega \le \pi.$$

The DTFT provides a frequency-domain representation of a sequence, establishing a relationship between the discrete-time sequence $x[n]$ and its spectrum $X(e^{j\omega})$, which is a periodic and continuous function of ω with period 2π.

For an aperiodic finite-energy signal, the DTFT generally has a continuous spectrum. When the sequence is finite in length, such as $x[n]$ defined for $0 \le n \le N-1$, the DTFT can be evaluated at a finite number of frequency samples. In practice, we choose a dense grid of ω values and compute

$$X(\omega_k) = \sum_{n=0}^{N-1} x[n]e^{-j\omega_k n}, \quad k = 0, 1, \ldots, M-1.$$

This computation can be expressed compactly using a matrix formulation. If

$$E_{k,n} = e^{-j\omega_k n},$$

then the spectrum samples are obtained as

$$X = E \cdot x.$$

For the sequence

$$x[n] = \cos(0.1\pi n), \quad 0 \le n \le N-1,$$

the DTFT spectrum will consist of impulses or sharp peaks at $\omega = \pm 0.1\pi$, corresponding to the positive and negative frequency components of the cosine signal.

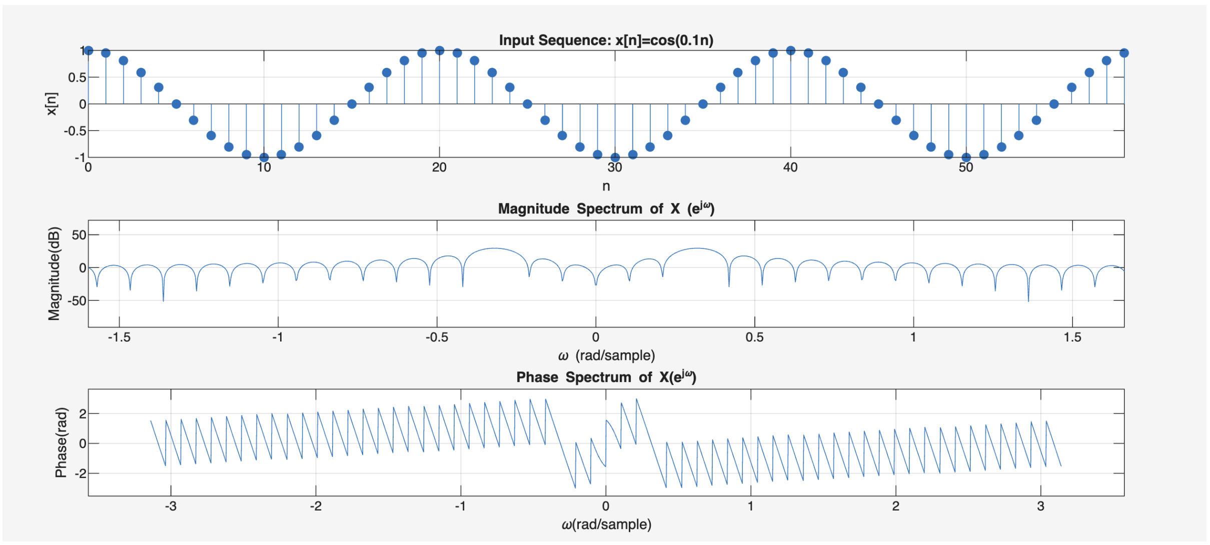
MATLAB Code

Listing 1: DTFT using matrix multiplication

```
clc;
clear;
```

```
close all;

N = 60;
n = 0:(N-1);
x = cos(0.1*pi*n);

w = linspace(-pi,pi,2048);

E = exp(-1j*(w.')*n);
X = (E*x.');

subplot (3,1,1);
stem (n,x,'filled');
grid on;
xlabel ('n');
ylabel ('x[n]');
title ('Input Sequence: x[n]=cos(0.1n)');
xlim ([0 N-1]) ;

subplot (3,1,2);
plot(w,20*log10(abs(X)));
grid on;
xlabel ('\omega (rad/sample)');
ylabel ('Magnitude(dB)');
title ('Magnitude Spectrum of X (e^{j\omega})');

subplot (3,1,3);
plot(w,angle(X));
grid on;
xlabel ('\omega(rad/sample)');
ylabel ('Phase(rad)');
title ('Phase Spectrum of X(e^{j\omega})');
```

Results



1. The input sequence $x[n]$ in the time domain.
2. The magnitude spectrum $|X(e^{j\omega})|$ in dB.
3. The phase spectrum $\angle X(e^{j\omega})$.

- The input sequence $x[n] = \cos(0.1\pi n)$ was generated for $N = 60$ samples.
- The DTFT was computed using the matrix multiplication method.
- The magnitude spectrum shows distinct peaks at $\omega = \pm 0.1\pi$, as expected for a cosine signal.
- The phase spectrum corresponds to the phase shifts of the sinusoidal components.

Linear Convolution of Two Sequences

Aim

To compute the linear convolution of two discrete-time sequences

$$x[n] = [1, 2, 3, 2], \quad y[n] = [1, 2, 2]$$

using a matrix-based approach in MATLAB and compare the result with MATLAB's built-in `conv` function.

Theory

The linear convolution of two discrete-time sequences $x[n]$ and $y[n]$ is defined as

$$r[n] = (x * y)[n] = \sum_{k=0}^{m-1} x[k]y[n-k],$$

where m and n are the lengths of $x[n]$ and $y[n]$, and $r[n]$ is defined for $0 \le n \le m+n-2$.

The convolution can also be represented using a **matrix approach**. For sequences x of length m and y of length n, we construct a convolution matrix X of size $l \times n$, where $l = m + n - 1$. Each column of X is a shifted version of $x[n]$, padded with zeros. Then the convolution can be computed as

$$r = X \cdot y_{\text{column}}.$$

This method provides a clear numerical procedure for computing convolution without directly using loops for the summation.

MATLAB Code

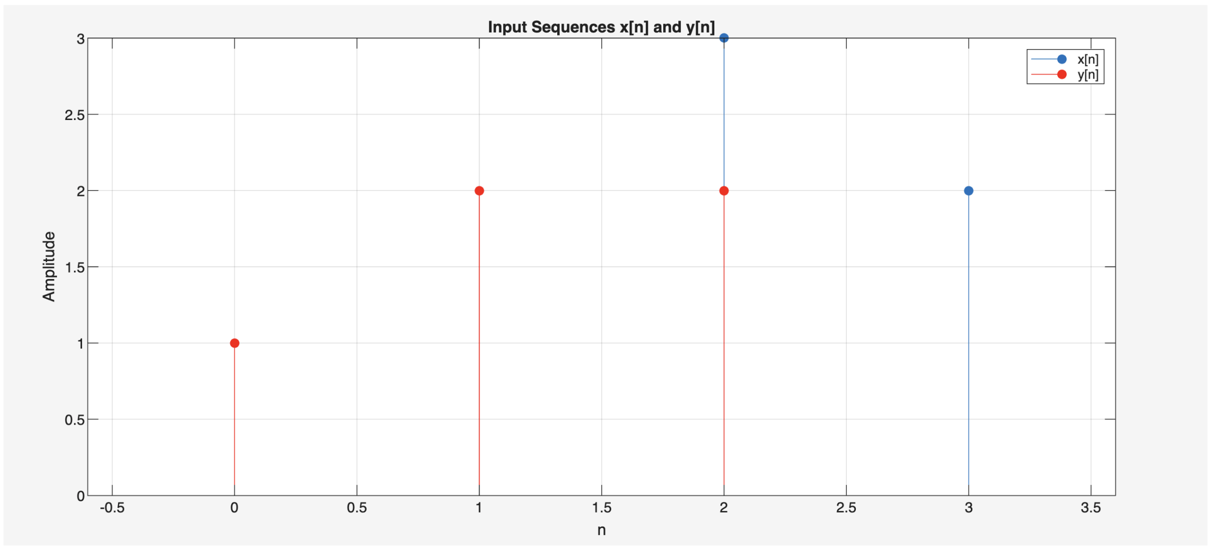
Listing 2: Matrix-based convolution of two sequences

```
clc;
clear;
close all;

x = [1, 2, 3, 2];
y = [1, 2, 2];

m = length(x);
n = length(y);
l = m + n - 1;

x_n = zeros(l, n);
for col = 1:n
    for row = col : col + m - 1
        x_n(row, col) = x(row - col + 1);
    end
end

disp('Convolution matrix X:');
disp(x_n);

% Column vector of y
y_m = y(:);

result = x_n * y_m;
disp('Matrix-based convolution result:');
disp(result);

conv_result = conv(x,y);
disp('conv(x,y) result:');
disp(conv_result);

figure;
```

```
stem(0:m-1, x, 'filled', 'DisplayName','x[n]');
hold on;
stem(0:n-1, y, 'r', 'filled', 'DisplayName','y[n]');
hold off;
grid on;
xlabel('n');
ylabel('Amplitude');
title('Input Sequences x[n] and y[n]');
legend;

figure;
subplot(1,2,1)
stem(0:l-1, result, 'filled');
title('Matrix-based Convolution');
xlabel('n'); ylabel('Amplitude'); grid on;

subplot(1,2,2)
stem(0:l-1, conv_result, 'filled');
title('MATLAB conv(x,y)');
xlabel('n'); ylabel('Amplitude'); grid on;
```

Results

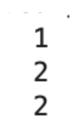
The input sequences ::



- The convolution matrix X generated is:

```
Convolution matrix X:
     1     0     0
     2     1     0
     3     2     1
     2     3     2
     0     2     3
     0     0     2
```

- The column vector for y is

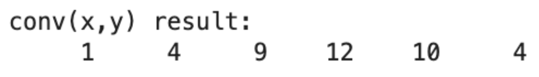


```
1
2
2
```

- Multiplying the matrix and column vector yields the convolution:

$$r[n] = [1\ 4\ 9\ 12\ 10\ 4]$$

- This matches exactly with MATLAB's built-in `conv(x,y)` function.



```
conv(x,y) result:
     1     4     9    12    10     4
```

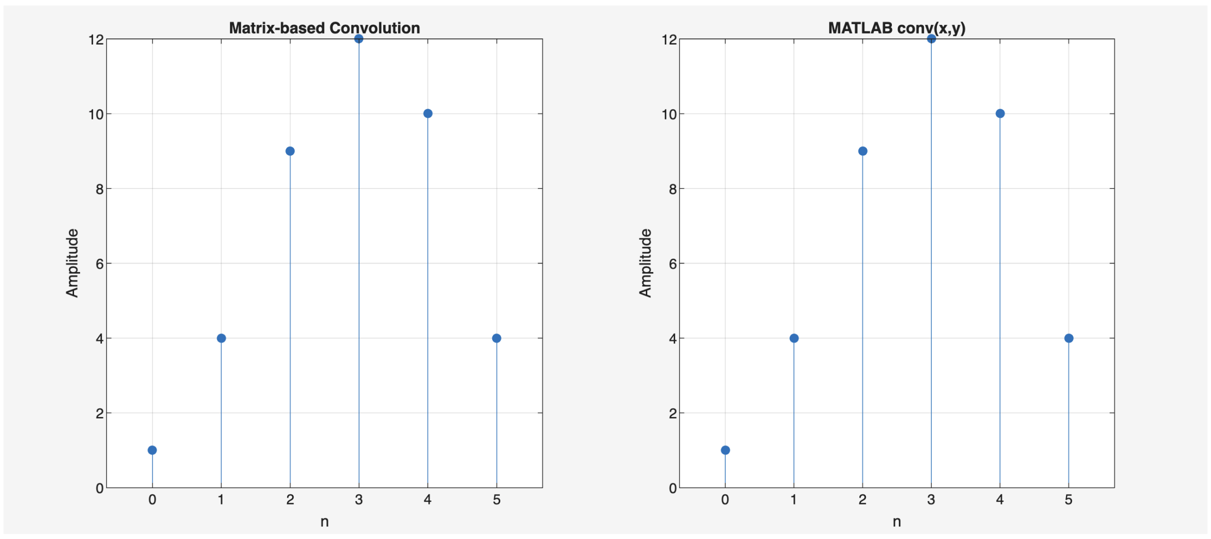


Figure 1: Convolution result

Circular Convolution of Two Sequences Using Matrix Multiplication

Aim

To compute the circular convolution of two discrete-time sequences

$$x[n] = [1, 3, 2, 5], \quad h[n] = [2, 1, 3]$$

using the matrix multiplication approach in MATLAB and verify the result with MATLAB's built-in `cconv` function.

Theory

Circular convolution of two sequences $x[n]$ and $h[n]$ of length K is defined as

$$y_c[n] = \sum_{m=0}^{K-1} x[m]h[(n-m) \bmod K], \quad n = 0, 1, \ldots, K-1.$$

Using a **matrix multiplication method**, we construct a circular convolution matrix X of size $K \times K$ from $x[n]$, where each row is a circularly shifted version of $x[n]$. Then, the circular convolution is computed as

$$y_c = X \cdot h_{\text{column}},$$

where h_{column} is the column vector of the second sequence.

This method allows us to compute the circular convolution without directly using the summation formula.

MATLAB Code

Listing 3: Function for circular convolution

```
function result = cconv_g(x,h)
    M = length(x);
    N = length(h);
    K = max(M,N);
    X = zeros(K, K);
    for i = 1:K
        for j = 1:K
            if(i-j+1 > 0)
                X(i,j) = x(i-j+1);
            elseif(i < j)
                X(i,j) = x(i-j+1+K);
            else
                X(i,j) = 0;
            end
        end
    end

    Y = zeros(K,1);

    for i = 1:N
        Y(i) = h(i);
    end

    result = X * Y;
end
```

Listing 4: Verifying convolution result

```
x = [1 3 2 5];
```

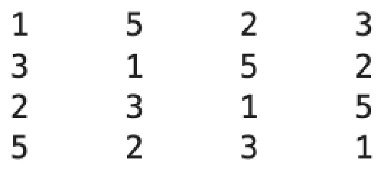
```
h = [2 1 3];

result = cconv_g(x,h);
disp("Circular convolution result")
disp(result)

disp("Circular convolution result using cconv function")
disp(cconv(x,h,K))
```

Listing 5: Linear convolution from circular convolution

```
x = [1 3 2 5];
h = [2 1 3];
M = length(x);
N = length(h);
K = M+N-1;

X = [x, zeros(1,K-M)];
Y = [h, zeros(1,K-N)];

result = cconv_g(X,Y);
disp('Circular convolution result:');
disp(result);

linear_conv = conv(x,h);
disp('Linear convolution result:');
disp(linear_conv);
```

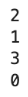
Results

- The circular convolution matrix X generated from $x[n]$ is of size $K \times K$ with circularly shifted rows.



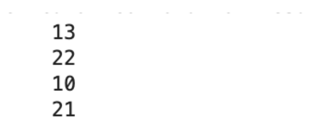
```
1    5    2    3
3    1    5    2
2    3    1    5
5    2    3    1
```

- The column vector for $h[n]$ is



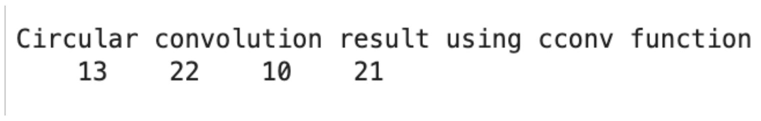
```
2
1
3
0
```

- Multiplying the matrix and column vector yields the circular convolution result



```
13
22
10
21
```

- This matches exactly with MATLAB’s built-in `cconv(x,h,K)` function.



```
Circular convolution result using cconv function
    13    22    10    21
```

N-Point Discrete Fourier Transform (DFT) and Inverse DFT

Aim

a) WAP to compute n-point DFT X(K) of a given sequence x[n] and verify result using fft function b) Obtain results of IDFT of computed DFT to get back x[n] c) Verify the parseval's theorem d)Verify convolution theorem

Theory

The **Discrete Fourier Transform (DFT)** of an N-point sequence $x[n]$ is defined as

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}, \quad k = 0, 1, \ldots, N-1.$$

The **Inverse DFT (IDFT)** is defined as

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N}, \quad n = 0, 1, \ldots, N-1.$$

Parseval's Theorem relates energy in time and frequency domains:

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

Convolution Theorem states that the convolution of two sequences in the time domain is equivalent to multiplication in the frequency domain:

$$x_1[n] * x_2[n] \longleftrightarrow X_1[k] \cdot X_2[k]$$

MATLAB Code

Listing 6: Compute N-point DFT, IDFT, Parseval's theorem, and Convolution theorem

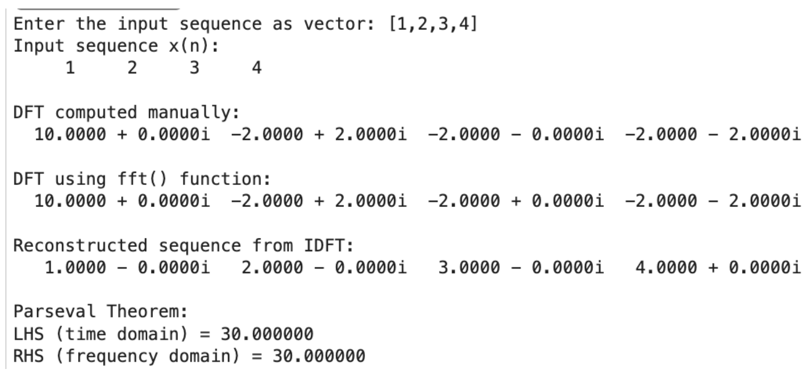
```
clc;
clear; close all;

% Input sequence
x = input('Enter the input sequence as vector: ');
N = length(x);
disp('Input sequence x(n):'); disp(x);

% (a) Compute N-point DFT manually
X_m = zeros(1,N);
for k = 0:N-1
    for n = 0:N-1
```

```
        X_m(k+1) = X_m(k+1) + x(n+1)*exp(-1j*2*pi*k*n/N);
    end
end

% Using MATLAB fft function
X_fft = fft(x,N);

disp('DFT computed manually:'); disp(X_m);
disp('DFT using fft() function:'); disp(X_fft);

% (b) Compute IDFT
x_r = zeros(1,N);
for n = 0:N-1
    for k = 0:N-1
        x_r(n+1) = x_r(n+1) + (1/N)*X_m(k+1)*exp(1j*2*pi*k*n/N);
    end
end
disp('Reconstructed sequence from IDFT:'); disp(x_r);

% (c) Verify Parseval's theorem
lhs = sum(abs(x).^2);
rhs = (1/N)*sum(abs(X_m).^2);
disp('Parseval Theorem:');
fprintf('LHS (time domain) = %f\n', lhs);
fprintf('RHS (frequency domain) = %f\n', rhs);
if abs(lhs-rhs) < 1e-6
    disp('Parseval theorem verified!');
else
    disp('Parseval theorem not verified!');
end

% (d) Verify Convolution Theorem
x1 = input('Enter first sequence for convolution: ');
x2 = input('Enter second sequence for convolution: ');
Nconv = length(x1) + length(x2) - 1;

y_time = conv(x1, x2);

X1 = fft(x1, Nconv);
X2 = fft(x2, Nconv);
Y_freq = X1 .* X2;
y_freq = ifft(Y_freq);

disp('Convolution in time domain:'); disp(y_time);
disp('Convolution using DFT (IDFT of product):'); disp(y_freq);

if max(abs(y_time - y_freq)) < 1e-6
    disp('Convolution theorem verified!');
```

```
else
    disp('Convolution theorem not verified!');
end

% Plots
figure;
subplot(3,1,1);
stem(0:N-1, x, 'filled'); title('INPUT SEQUENCE x(n)');
xlabel('n'); ylabel('Amplitude'); grid on;

subplot(3,1,2);
stem(0:N-1, abs(X_fft), 'filled'); title('MAGNITUDE |X(k)|');
xlabel('Frequency index k'); ylabel('Magnitude'); grid on;

subplot(3,1,3);
stem(0:N-1, angle(X_fft), 'filled'); title('PHASE \angle X(k)');
xlabel('Frequency index k'); ylabel('Phase (radians)'); grid on;
```

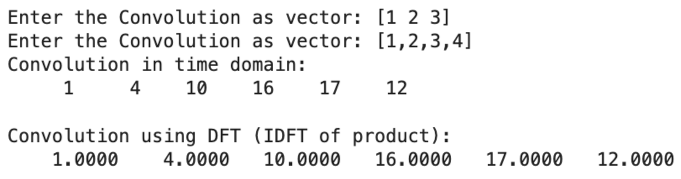
Results

- The input sequence $x[n]$ is displayed.
- The N-point DFT computed manually and using `fft` are shown and verified to be equal.
- The IDFT of the computed DFT reconstructs the original sequence $x[n]$.
- Parseval's theorem is verified:



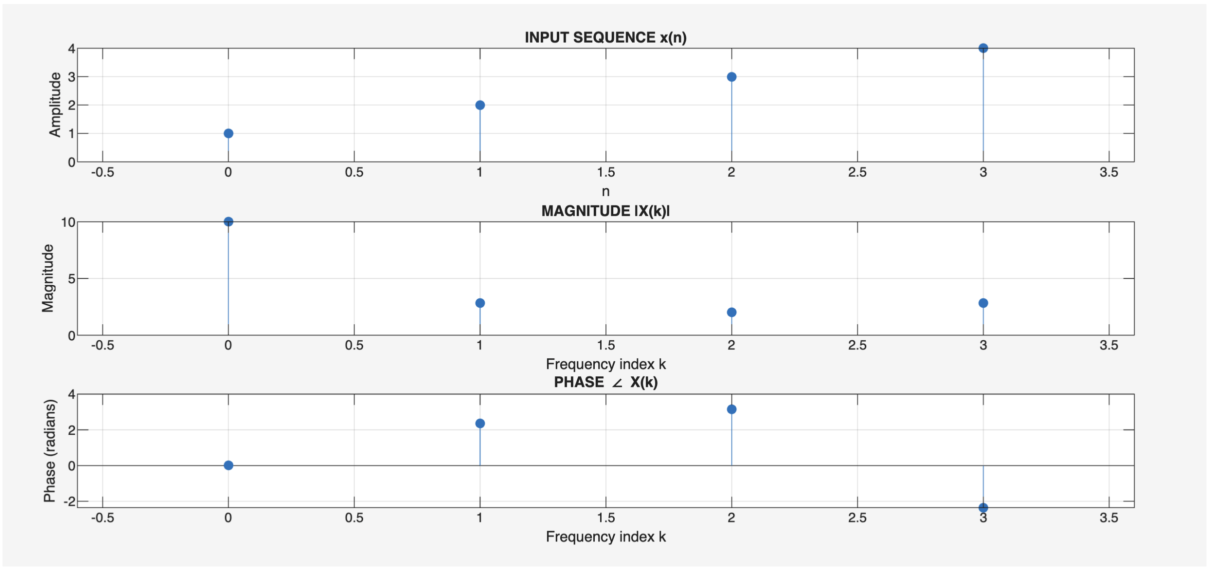
```
Enter the input sequence as vector: [1,2,3,4]
Input sequence x(n):
     1     2     3     4

DFT computed manually:
  10.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 - 0.0000i  -2.0000 - 2.0000i

DFT using fft() function:
  10.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 - 2.0000i

Reconstructed sequence from IDFT:
   1.0000 - 0.0000i   2.0000 - 0.0000i   3.0000 - 0.0000i   4.0000 + 0.0000i

Parseval Theorem:
LHS (time domain) = 30.000000
RHS (frequency domain) = 30.000000
```

- Convolution theorem is verified by comparing linear convolution with IDFT of the product of DFTs:



```
Enter the Convolution as vector: [1 2 3]
Enter the Convolution as vector: [1,2,3,4]
Convolution in time domain:
     1     4    10    16    17    12

Convolution using DFT (IDFT of product):
    1.0000    4.0000   10.0000   16.0000   17.0000   12.0000
```

- Plots of input sequence, magnitude spectrum, and phase spectrum are shown:



Experiment 5: Properties of Discrete Fourier Transform (DFT)

Aim

To prove the following properties of the Discrete Fourier Transform (DFT):

1. Circular Time Shift
2. Circular Time Reversal
3. Circular Frequency Shift
4. Duality

Apparatus Required

MATLAB Software

Theory

The Discrete Fourier Transform (DFT) is a fundamental tool for analyzing discrete-time signals in the frequency domain. If $x(n)$ is a discrete signal of length N, its DFT is given by:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}$$

and the Inverse DFT (IDFT) is:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j2\pi kn/N}$$

Properties:

1. **Circular Time Shift:** If $x(n) \leftrightarrow X(k)$, then $x((n-m))_N \leftrightarrow e^{-j2\pi km/N}X(k)$
2. **Circular Time Reversal:** If $x(n) \leftrightarrow X(k)$, then $x((-n))_N \leftrightarrow X((-k))_N$
3. **Circular Frequency Shift:** If $x(n) \leftrightarrow X(k)$, then $x(n)e^{j2\pi mn/N} \leftrightarrow X((k-m))_N$
4. **Duality:** If $x(n) \leftrightarrow X(k)$, then $X(n) \leftrightarrow Nx(-k)$

MATLAB Code

```
clc;
clear all;
close all;

N = 8;                           % Number of points
n = 0:N-1;
x = [1 2 3 4 0 0 0 0];          % Input sequence
X = fft(x, N);                  % DFT of input

%% 1. Circular Time Shift
m = 2;
x_shift = circshift(x, [0, m]);
X_shift = fft(x_shift, N);

figure;
subplot(2,1,1);
stem(n, abs(X)); title('Original Spectrum');
subplot(2,1,2);
stem(n, abs(X_shift)); title('Circular Time Shift Spectrum');
saveas(gcf,'circular_time_shift.png');

%% 2. Circular Time Reversal
x_rev = fliplr(x);
X_rev = fft(x_rev, N);

figure;
subplot(2,1,1);
stem(n, abs(X)); title('Original Spectrum');
subplot(2,1,2);
stem(n, abs(X_rev)); title('Circular Time Reversal Spectrum');
saveas(gcf,'circular_time_reversal.png');

%% 3. Circular Frequency Shift
m = 2;
x_freq_shift = x .* exp(1j*2*pi*m*n/N);
X_freq_shift = fft(x_freq_shift, N);

figure;
subplot(2,1,1);
stem(n, abs(X)); title('Original Spectrum');
subplot(2,1,2);
stem(n, abs(X_freq_shift)); title('Circular Frequency Shift Spectrum');
saveas(gcf,'circular_frequency_shift.png');

%% 4. Duality
X_dual = fft(X, N);
```

```
x_dual = N * ifft(X_dual);
figure;
subplot(2,1,1);
stem(n, abs(x)); title('Original Signal');
subplot(2,1,2);
stem(n, abs(x_dual)); title('Duality Property');
saveas(gcf,'duality_property.png');
```

Result

The properties of Discrete Fourier Transform, namely Circular Time Shift, Circular Time Reversal, Circular Frequency Shift, and Duality, were successfully verified using MATLAB.

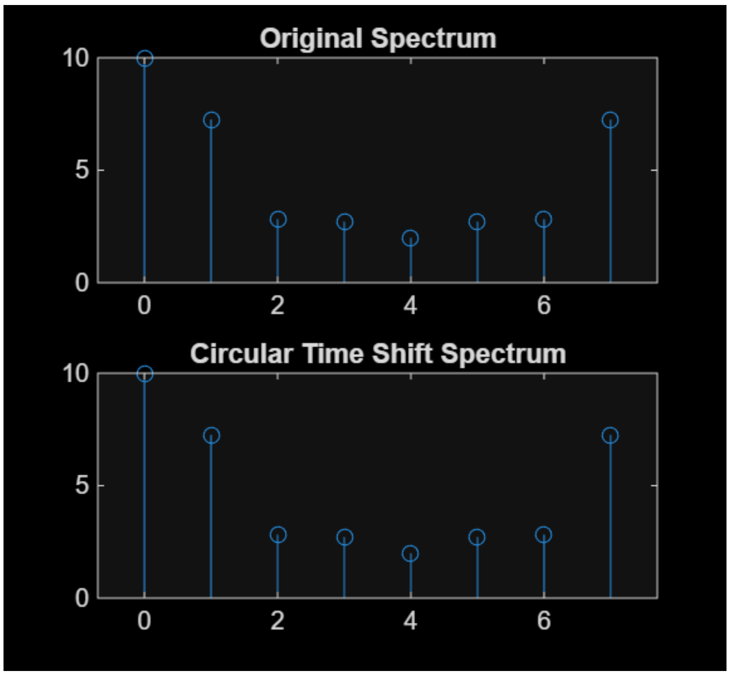


Figure 2: Circular Time Shift Property

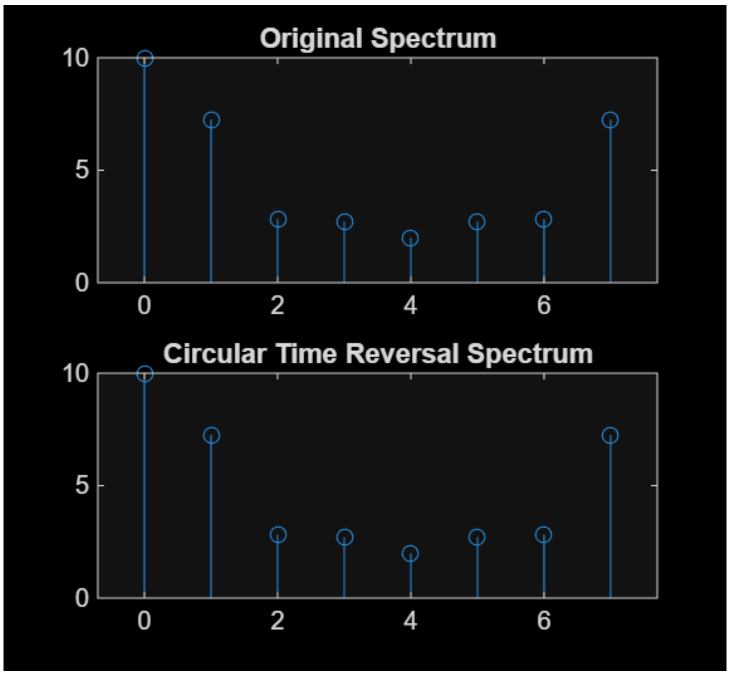


Figure 3: Circular Time Reversal Property

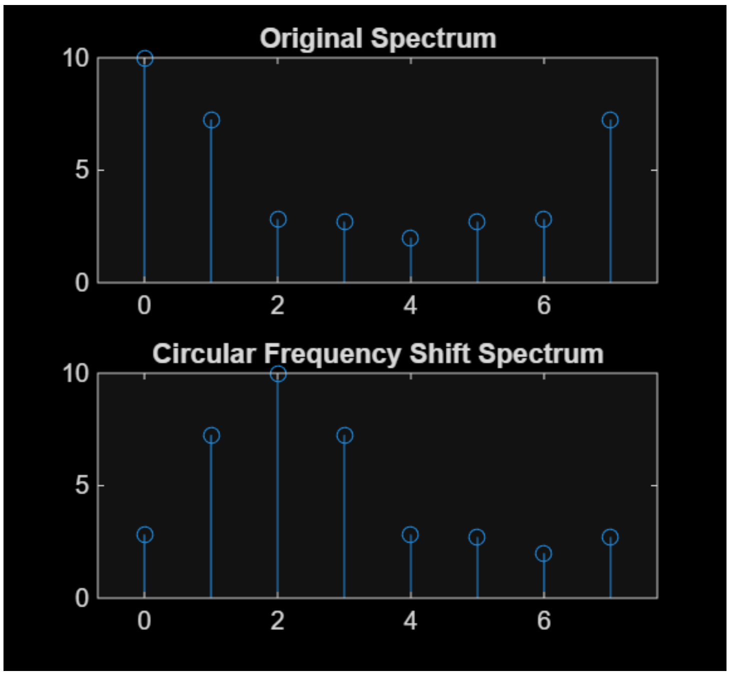


Figure 4: Circular Frequency Shift Property

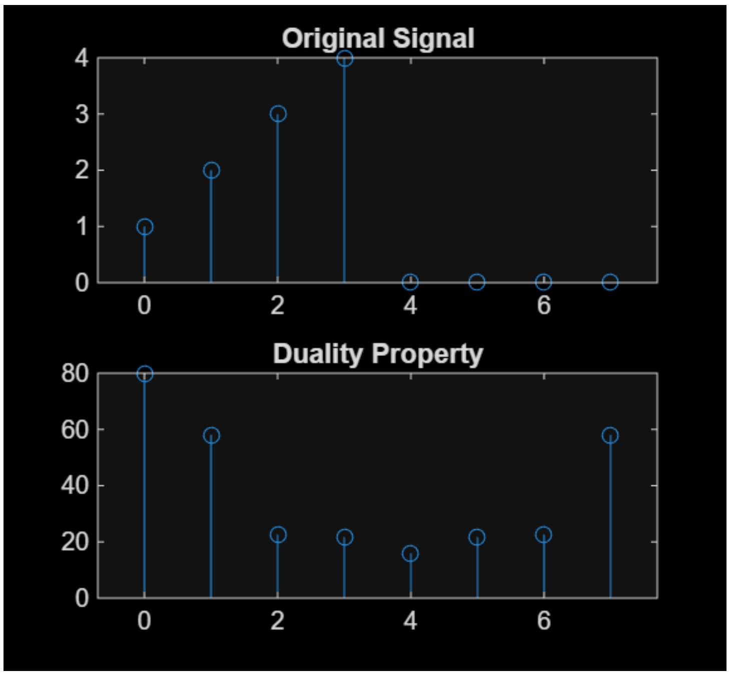


Figure 5: Duality Property

Conclusion

All four fundamental properties of the Discrete Fourier Transform were successfully demonstrated and verified using MATLAB plots.

Viva Questions

1. **What is the difference between DFT and FFT?**
   The Discrete Fourier Transform (DFT) is a mathematical operation that converts a finite-duration discrete-time signal into its frequency-domain representation. The Fast Fourier Transform (FFT) is an efficient algorithm used to compute the DFT with significantly reduced computational complexity, typically from $O(N^2)$ to $O(N \log N)$.

2. **What is meant by circular convolution?**
   Circular convolution is a type of convolution where the sequences are assumed to be periodic. It represents the convolution of two finite-length discrete-time sequences under modulo-N arithmetic. It is defined as:
   $$y[n] = \sum_{k=0}^{N-1} x[k]\, h[(n-k) \bmod N]$$
   Circular convolution is equivalent to multiplication in the DFT domain.

3. **Explain the significance of circular time shift.**
   A circular time shift of a sequence corresponds to multiplying its DFT by a complex exponential term. If $x[n]$ has a DFT $X[k]$, then a circular shift by m samples gives:
   $$x[(n-m) \bmod N] \overset{\text{DFT}}{\longleftrightarrow} X[k]e^{-j\frac{2\pi km}{N}}$$

This property is useful in understanding how time-domain shifts affect the frequency domain phase.

4. **What does the duality property signify in DFT?**
   The duality property of DFT states that if $x[n]$ has a DFT $X[k]$, then $X[n]$ has a DFT equal to N times the time-reversed sequence $x[-k]$. Mathematically,

$$x\big((n-m) \bmod N\big) \overset{\text{DFT}}{\longleftrightarrow} X[k]\, e^{-j\frac{2\pi km}{N}}$$

   This shows a symmetric relationship between the time and frequency domains.

5. **Why is DFT periodic in both time and frequency domains?**
   The DFT assumes the input sequence is periodic with period N. Hence, both the time-domain sequence and its DFT repeat every N samples. This periodicity arises from the discrete and finite-length nature of the DFT, which implies sampling in one domain corresponds to periodic extension in the other.

Experiment 6: FIR Window Functions

Aim

To study and prove the following properties related to FIR window functions:

1. Compare the magnitude response of various window functions for a given length N.
2. Compare the magnitude response of a specific window function for different values of N.
3. Observe the linear phase characteristic of a window of length N.
4. Modify some values of $w(n)$ from part (c) to observe the non-linear phase response of $w'(n)$ due to violation of symmetry, i.e., $w'(n) \neq w'(N-n-1)$.

Apparatus Required

MATLAB Software

Theory

Finite Impulse Response (FIR) filters are implemented by truncating the ideal infinite impulse response using a window function. A window function $w(n)$ is multiplied with the ideal impulse response $h_d(n)$ to obtain a realizable FIR filter:

$$h(n) = h_d(n) \cdot w(n)$$

Common window functions include:

$$\begin{aligned}
\text{Rectangular: } & w(n) = 1, \quad 0 \leq n \leq N-1 \\
\text{Hamming: } & w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \\
\text{Hanning: } & w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) \\
\text{Blackman: } & w(n) = 0.42 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right)
\end{aligned}$$

The choice of window affects the main-lobe width and side-lobe attenuation of the resulting frequency response.

MATLAB Code

```
clc;
clear all;
close all;
```

```
%% Part A: Compare various window functions for fixed N
N = 51;
w_rect = rectwin(N);
w_ham = hamming(N);
w_han = hann(N);
w_black = blackman(N);

figure;
freqz(w_rect,1,512);
title('Rectangular Window');
saveas(gcf,'rectangular_window.png');

figure;
freqz(w_ham,1,512);
title('Hamming Window');
saveas(gcf,'hamming_window.png');

figure;
freqz(w_han,1,512);
title('Hanning Window');
saveas(gcf,'hanning_window.png');

figure;
freqz(w_black,1,512);
title('Blackman Window');
saveas(gcf,'blackman_window.png');

%% Part B: Compare one window (Hamming) for different N
N1 = 21; N2 = 51; N3 = 101;
w1 = hamming(N1);
w2 = hamming(N2);
w3 = hamming(N3);

figure;
freqz(w1,1,512);
title('Hamming Window (N=21)');
saveas(gcf,'hamming_N21.png');

figure;
freqz(w2,1,512);
title('Hamming Window (N=51)');
saveas(gcf,'hamming_N51.png');

figure;
freqz(w3,1,512);
title('Hamming Window (N=101)');
saveas(gcf,'hamming_N101.png');
```

```
%% Part C: Linear phase characteristic of Hamming window
N = 51;
w = hamming(N);
[H, f] = freqz(w, 1, 512);
figure;
subplot(2,1,1);
plot(f/pi, abs(H));
title('Magnitude Response of Hamming Window');
subplot(2,1,2);
plot(f/pi, unwrap(angle(H)));
title('Phase Response (Linear)');
saveas(gcf,'linear_phase.png');

%% Part D: Modify symmetry to observe non-linear phase
w_dash = w;
w_dash(10) = w_dash(10) + 0.3;   % breaking symmetry
[H_dash, f_dash] = freqz(w_dash, 1, 512);
figure;
subplot(2,1,1);
plot(f_dash/pi, abs(H_dash));
title('Magnitude Response (Non-Symmetric Window)');
subplot(2,1,2);
plot(f_dash/pi, unwrap(angle(H_dash)));
title('Phase Response (Non-Linear)');
saveas(gcf,'nonlinear_phase.png');
```

Result

The magnitude and phase responses for various FIR window functions were plotted and analyzed. All corresponding figures are attached below.

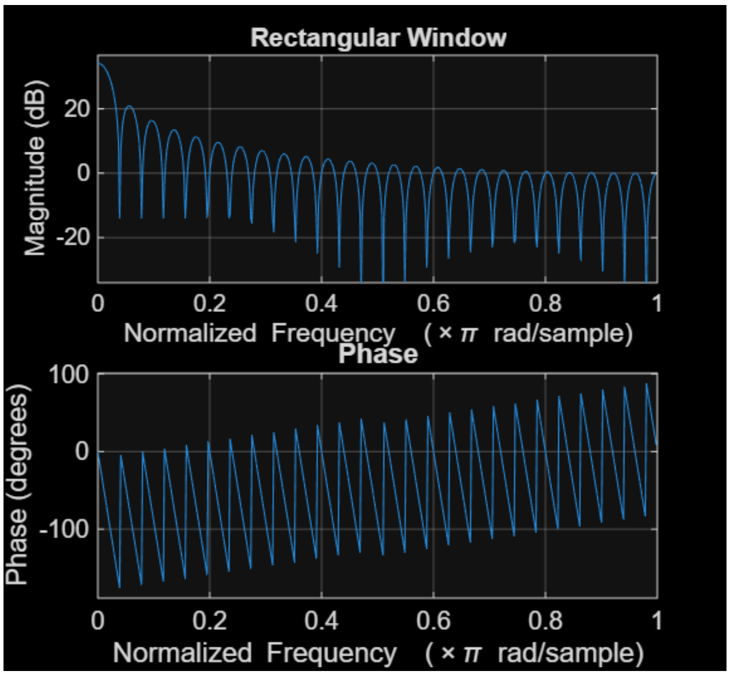


Figure 6: Rectangular Window

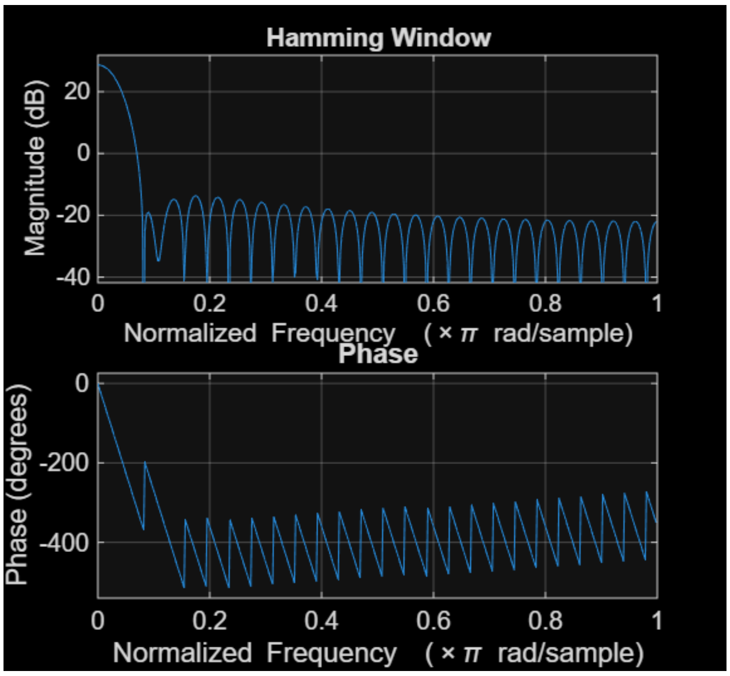


Figure 7: Hamming Window

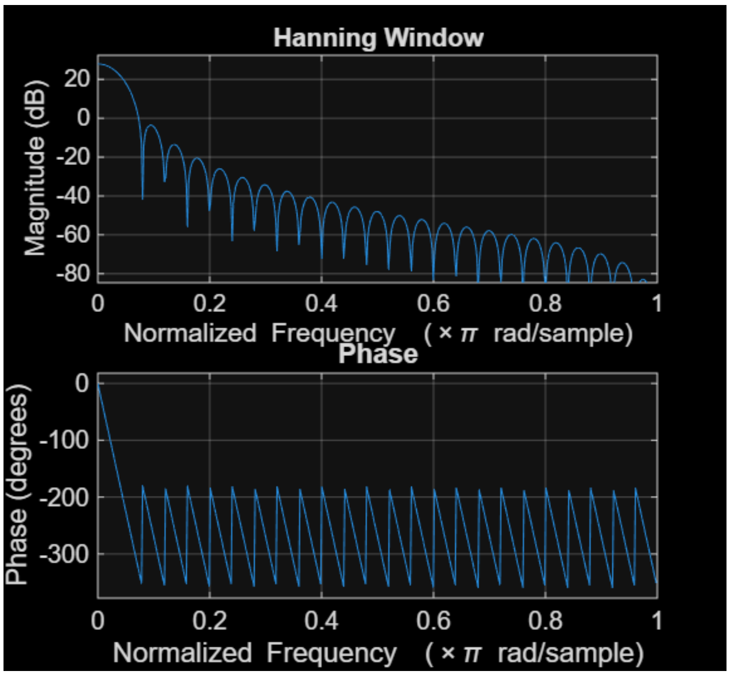


Figure 8: Hanning Window

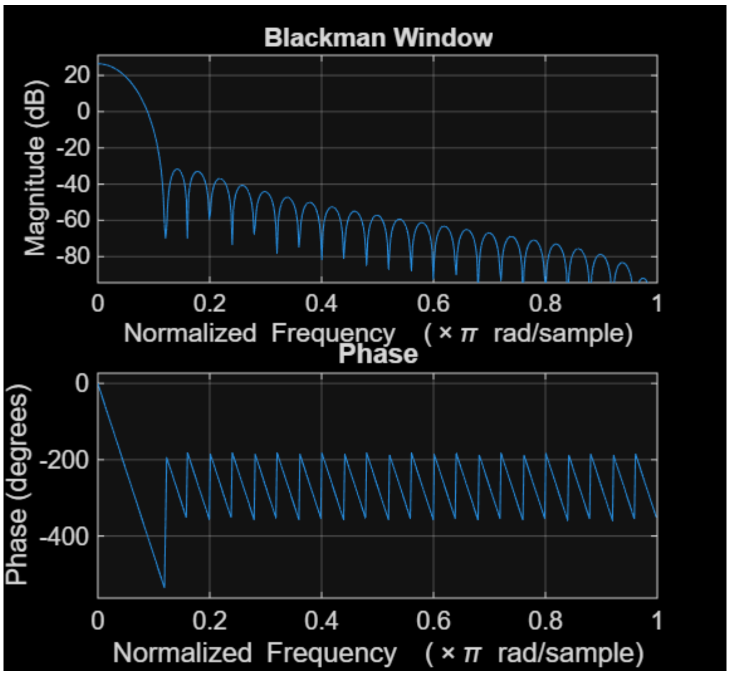


Figure 9: Blackman Window

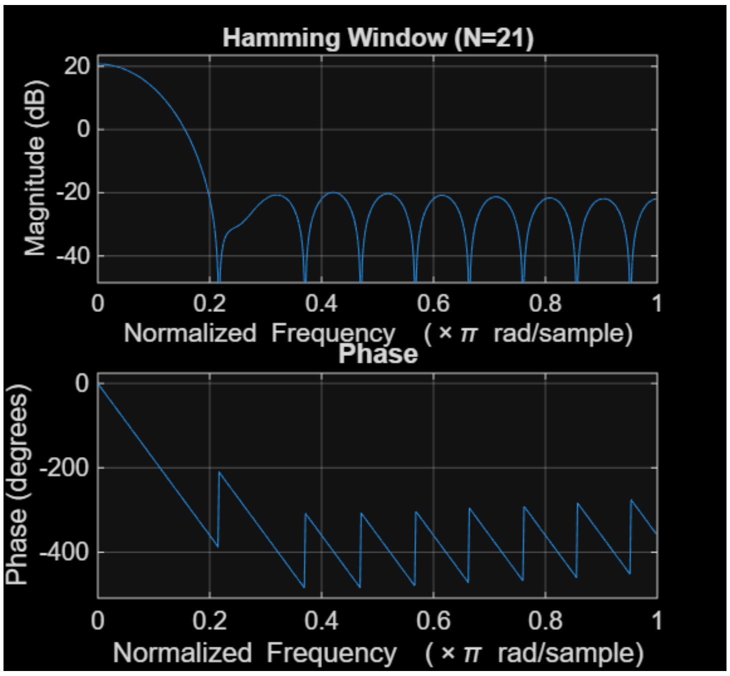


Figure 10: Hamming Window for N = 21

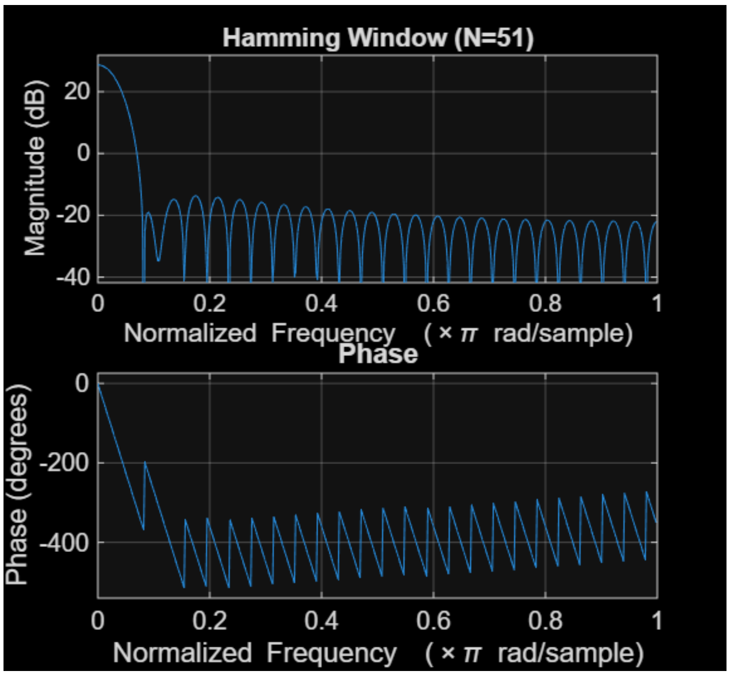


Figure 11: Hamming Window for N = 51

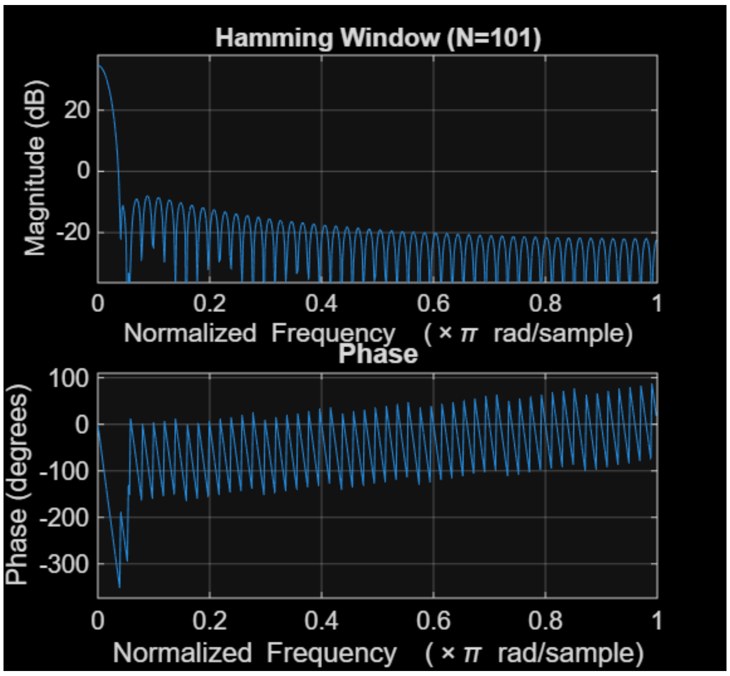


Figure 12: Hamming Window for N = 101

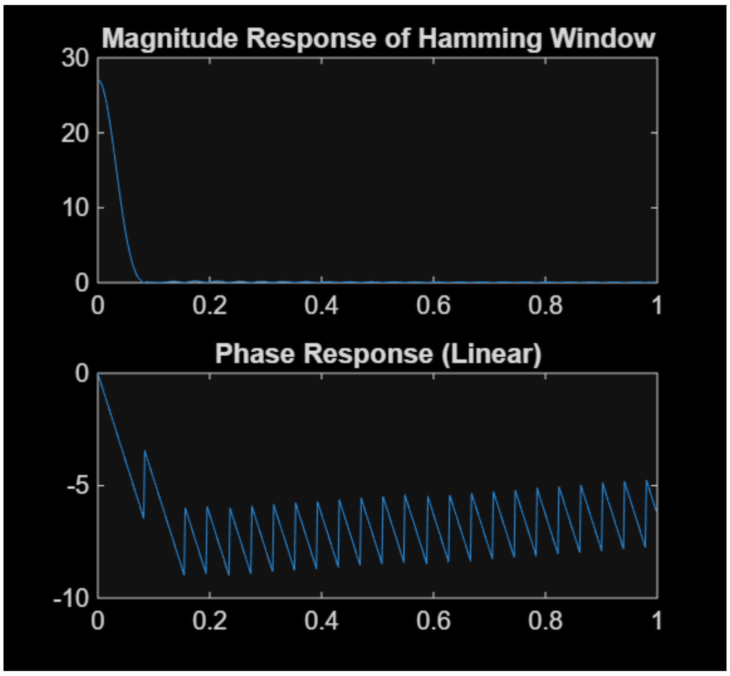


Figure 13: Linear Phase Response of Hamming Window

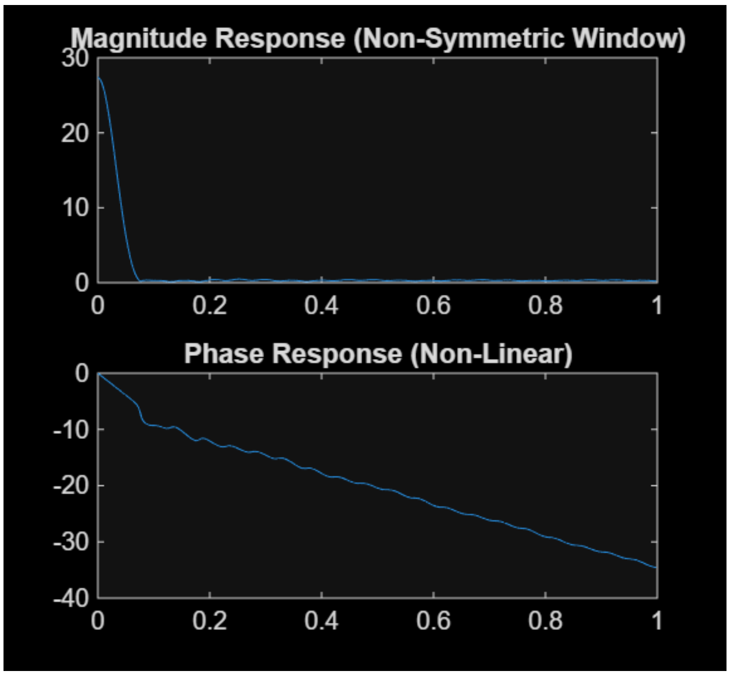


Figure 14: Non-Linear Phase Response after Modifying Symmetry

Conclusion

The properties of FIR window functions were successfully verified. Different windows were compared based on their magnitude and phase characteristics. The linear phase property was observed for symmetric windows, while breaking symmetry resulted in a non-linear phase response.

Viva Questions and Answers

1. **What is the importance of window functions in FIR filter design?**
   Window functions are used to truncate the infinite impulse response obtained from the ideal filter design to a finite length. They help control the spectral leakage and determine the trade-off between the main-lobe width (transition bandwidth) and side-lobe levels (stopband attenuation).

2. **How do the main-lobe width and side-lobe attenuation affect filter performance?**
   The main-lobe width determines the transition bandwidth — narrower main lobes result in sharper transitions. The side-lobe attenuation controls the amount of ripple or leakage in the stopband. A wider main lobe gives smoother transitions but better stopband suppression.

3. **Why does a symmetric window yield a linear phase response?**
   When the impulse response of an FIR filter is symmetric (i.e., $h[n] = h[M-1-n]$), the resulting filter has a linear phase characteristic. This ensures that all frequency components experience the same time delay, preventing phase distortion in the output signal.

4. **What happens when window symmetry is broken?**
   Breaking the symmetry of the window results in a non-linear phase response, causing different frequency components to experience different phase delays. This can lead to waveform distortion in the filtered signal.

5. **Compare Hamming, Hanning, and Blackman windows in terms of side-lobe attenuation.**
   The Hamming window provides side-lobe attenuation of approximately 41 dB, the Hanning window about 44 dB, and the Blackman window about 58 dB. Thus, the Blackman window offers the highest side-lobe suppression at the cost of a wider main-lobe (poorer frequency resolution).

Experiment 7: Design of FIR Filter

Aim

To design an FIR filter.

Apparatus Required

- MATLAB Software
- Computer System

Theory

Finite Impulse Response (FIR) filters are a class of digital filters with a finite number of coefficients. They are always stable and can be designed to have a linear phase response. The design process generally involves determining the filter order and coefficients based on desired frequency specifications.

In this experiment:

1. **Part A:** Calculate the length of the filter N using the Kaiser formula based on given specifications.
2. **Part B:** Obtain and plot the impulse response $h(n)$ of a low-pass filter using the Blackman window for N.
3. **Part C:** Plot the magnitude and phase spectra of $h(n)$.
4. **Part D:** Modify the coefficients to obtain $h'(n)$ and observe the new magnitude and phase spectra.

Given specifications:

$$f_s = 20 \text{ kHz}, \quad r_p = 0.07, \quad r_s = 0.04, \quad f_p = 2 \text{ kHz}, \quad f_s = 4 \text{ kHz}$$

MATLAB Code

```
clear; clc; close all;

% Given Specifications
fs = 20000;      % Sampling frequency in Hz
rp = 0.07;       % Passband ripple
rs = 0.04;       % Stopband ripple
fp = 2000;       % Passband frequency in Hz
fs1 = 4000;      % Stopband frequency in Hz

% Normalized frequencies
wp = 2*fp/fs;
```

```
ws = 2*fs1/fs;

% Calculate delta values
delta_p = rp;
delta_s = rs;

% Kaiser formula for filter order
A = -20*log10(min(delta_p, delta_s));
if A < 21
    alpha = 0;
elseif A <= 50
    alpha = 0.5842*(A-21)^0.4 + 0.07886*(A-21);
else
    alpha = 0.1102*(A-8.7);
end

delta_f = abs(ws - wp);
N = ceil((A - 8) / (2.285 * delta_f * pi));
if mod(N,2)~=0
    N = N+1; % make N even
end

fprintf('Calculated Filter Length N = %d\n', N);

% Part B: Design Low-Pass Filter using Blackman Window
wc = (wp + ws)/2;
b = fir1(N-1, wc, 'low', blackman(N));

figure(1);
stem(b);
title('Impulse Response h(n)');
xlabel('n'); ylabel('Amplitude');
saveas(gcf,'fig7_ImpulseResponse.png');

% Part C: Magnitude and Phase Response
[H, f] = freqz(b,1,1024,fs);

figure(2);
subplot(2,1,1);
plot(f, abs(H));
title('Magnitude Spectrum of h(n)');
xlabel('Frequency (Hz)'); ylabel('|H(f)|');

subplot(2,1,2);
plot(f, angle(H));
title('Phase Spectrum of h(n)');
xlabel('Frequency (Hz)'); ylabel('Phase (radians)');
saveas(gcf,'fig7_MagPhaseResponse.png');
```

```
% Part D: Modify coefficients
b_dash = b;
b_dash(10) = b_dash(10) + 0.2; % small change for asymmetry

[H_dash, f_dash] = freqz(b_dash,1,1024,fs);

figure(3);
subplot(2,1,1);
plot(f_dash, abs(H_dash));
title('Magnitude Spectrum of h''(n)');
xlabel('Frequency (Hz)'); ylabel('|H''(f)|');

subplot(2,1,2);
plot(f_dash, angle(H_dash));
title('Phase Spectrum of h''(n)');
xlabel('Frequency (Hz)'); ylabel('Phase (radians)');
saveas(gcf,'fig7_ModifiedMagPhase.png');
```

Result

- The FIR filter was designed successfully using the Blackman window.
- The calculated filter length was found to be appropriate for the given specifications.
- The plots obtained are as follows:

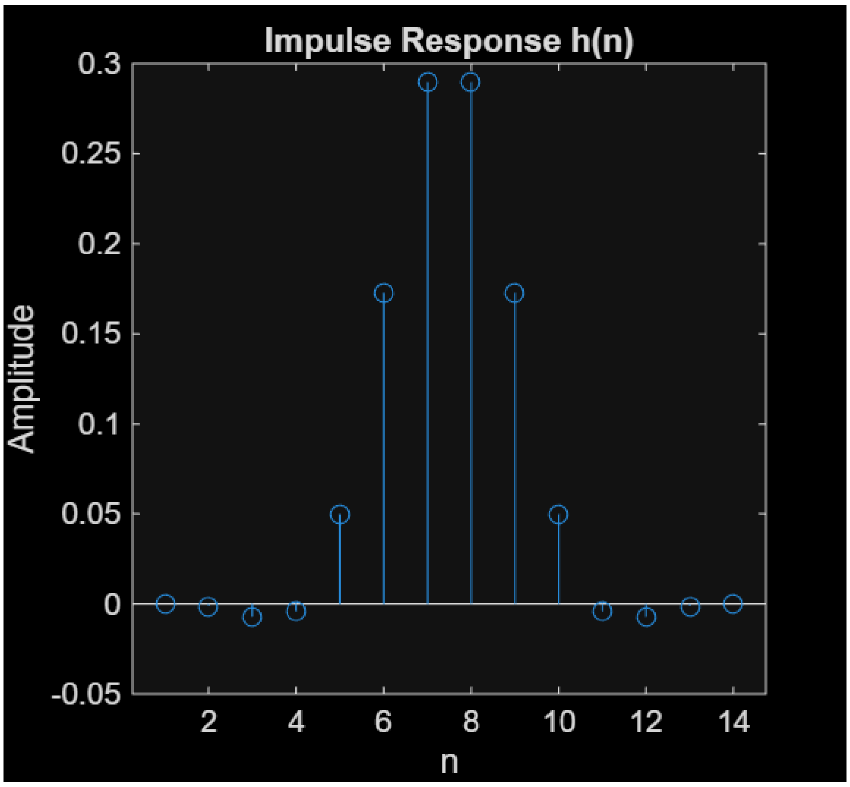


Figure 15: Impulse Response $h(n)$

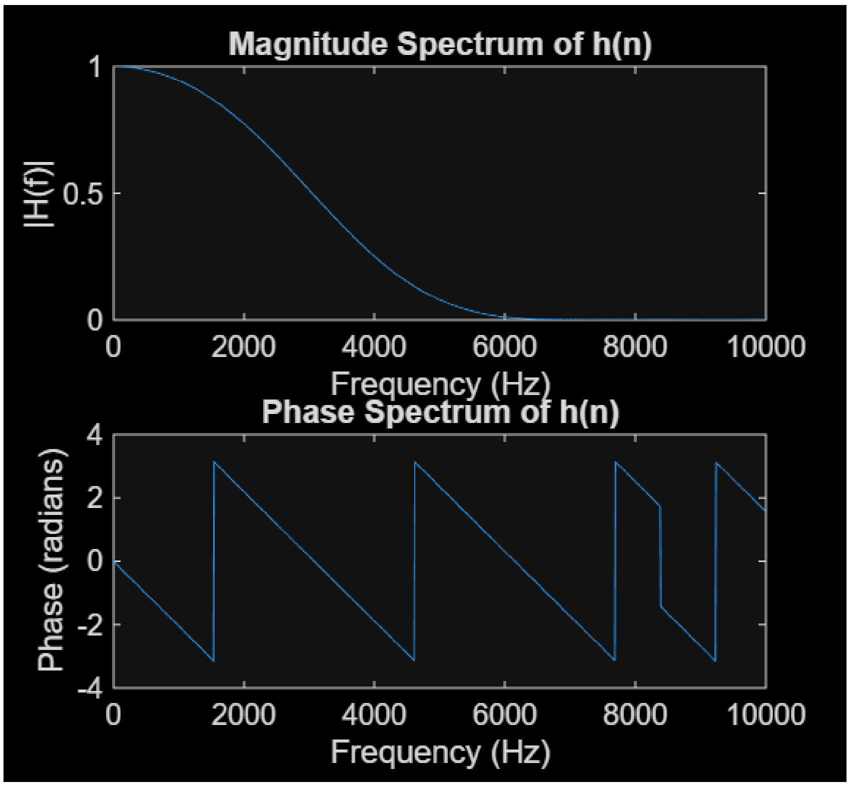


Figure 16: Magnitude and Phase Spectrum of $h(n)$

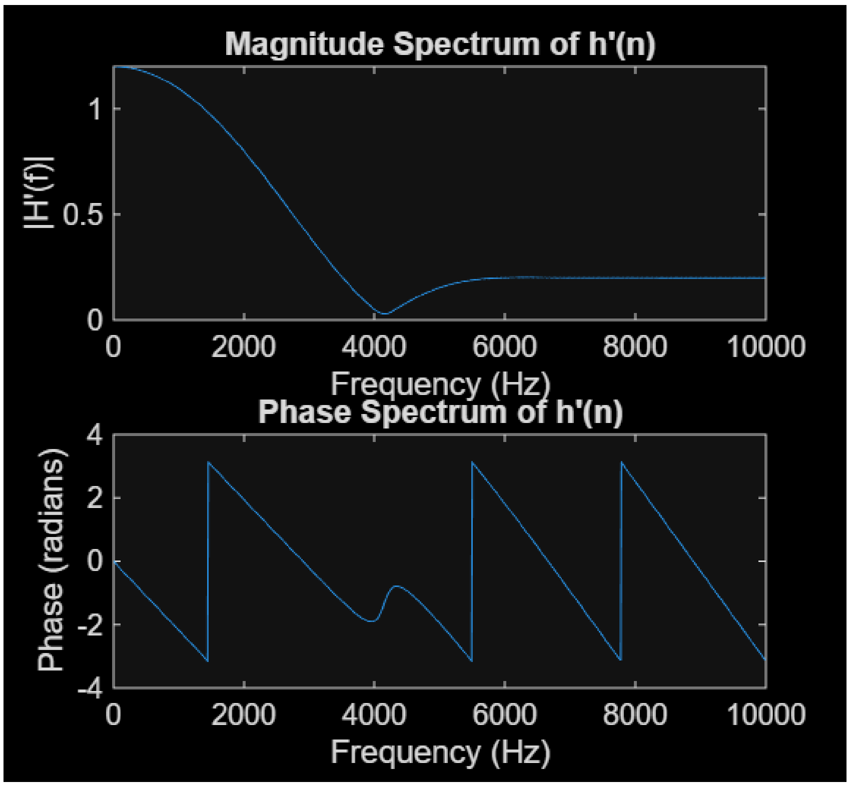


Figure 17: Magnitude and Phase Spectrum of Modified $h'(n)$

Conclusion

The FIR filter was designed using the Blackman window based on the given specifications. The impulse response and magnitude-phase spectra confirm a low-pass characteristic with linear phase. Modification of the coefficients introduced asymmetry, leading to a non-linear phase response.

Viva Questions and Answers

1. **What is the main difference between FIR and IIR filters?**
   Finite Impulse Response (FIR) filters have a finite-duration impulse response because they do not use feedback. Infinite Impulse Response (IIR) filters, on the other hand, use feedback and have an impulse response that theoretically extends to infinity. FIR filters are inherently stable, while IIR filters can become unstable if their poles lie outside the unit circle.

2. **Why are FIR filters always stable?**
   FIR filters are always stable because they consist only of feedforward terms. Their output depends solely on a finite number of past input samples and not on previous outputs. Since there are no feedback paths, the poles of the FIR system lie at the origin, ensuring BIBO (Bounded Input Bounded Output) stability.

3. **What is the advantage of using window functions in FIR design?**
   Window functions are used to truncate the ideal infinite-duration impulse response into a finite-length sequence. They control the trade-off between main-lobe width and side-lobe level in the frequency response, thereby influencing the filter's transition bandwidth and stopband attenuation.

4. **What causes a linear phase response in FIR filters?**
   A linear phase response in FIR filters is achieved when the impulse response is symmetric or anti-symmetric, i.e., $h[n] = \pm h[M-1-n]$. This symmetry ensures a constant group delay for all frequency components, preventing phase distortion in the output signal.

5. **How does modifying the coefficients affect the filter's phase response?**
   Any change that disturbs the symmetry of the filter coefficients results in a non-linear phase response. The loss of symmetry introduces phase distortion, causing different frequency components to experience unequal time delays.

Experiment 8: Design of FIR Hilbert Transformer

Aim:
To design an FIR Hilbert Transformer using the Hamming window method and analyze its impulse, magnitude, and phase responses, and verify the phase-shifting property by passing a cosine input through it.

Apparatus Required:

- MATLAB Software
- Computer System

Theory:
A Hilbert Transformer is a type of FIR filter that introduces a -90° phase shift for all positive frequency components and a $+90^\circ$ shift for all negative frequencies of a signal. It is widely used in modulation, demodulation, and analytic signal generation.
The ideal impulse response of a Hilbert transformer is:

$$h_d(n) = \begin{cases} \dfrac{2}{\pi(n - \frac{M-1}{2})}, & \text{for } n - \frac{M-1}{2} \text{ odd} \\ 0, & \text{for } n - \frac{M-1}{2} \text{ even} \end{cases}$$

A practical FIR version is obtained by multiplying $h_d(n)$ with a window function such as the Hamming window:

$$h(n) = h_d(n) \times w(n)$$

where $w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{M-1}\right)$.

MATLAB Code:

```
% EXPERIMENT 8: FIR Hilbert Transformer using Hamming Window


clc; clear; close all;

% Part (a): Design FIR Hilbert Transformer
M = 15;                  % Filter length
alpha = (M-1)/2;         % Center index
n = 0:M-1;               % Sample index

% Ideal impulse response of Hilbert Transformer
hd = zeros(1,M);
for i = 1:M
    if (n(i) - alpha) == 0
```

```
        hd(i) = 0;
    elseif mod(n(i)-alpha,2) ~= 0
        hd(i) = 2/(pi*(n(i)-alpha));
    else
        hd(i) = 0;
    end
end

% Apply Hamming window
w = hamming(M)';
h = hd .* w;

% Plot impulse response
figure;
stem(n, h, 'filled');
xlabel('n'); ylabel('h(n)');
title('Impulse Response of FIR Hilbert Transformer');
grid on;
saveas(gcf, 'exp8_impulse_response.png');

%% Part (b): Magnitude and Phase Spectrum
[H, w1] = freqz(h, 1, 512);

figure;
subplot(2,1,1);
plot(w1/pi, abs(H));
xlabel('\omega/\pi'); ylabel('|H(\omega)|');
title('Magnitude Spectrum of Hilbert Transformer');
grid on;

subplot(2,1,2);
plot(w1/pi, angle(H));
xlabel('\omega/\pi'); ylabel('H(\omega)');
title('Phase Spectrum of Hilbert Transformer');
grid on;
saveas(gcf, 'exp8_magnitude_phase.png');

%% Part (c): Input signal and filtering
n2 = 0:63;
x = cos(pi*n2/8);           % Input cosine signal
y = filter(h, 1, x);        % Output signal through Hilbert Transformer

%% Part (d): Plot input and output signals
figure;
subplot(2,1,1);
plot(n2, x, 'LineWidth', 1.2);
xlabel('n'); ylabel('x(n)');
title('Input Signal: x(n) = cos(\pi n / 8)');
```

```
grid on;

subplot(2,1,2);
plot(n2, y, 'r', 'LineWidth', 1.2);
xlabel('n'); ylabel('y(n)');
title('Output Signal after Hilbert Transformer');
grid on;
saveas(gcf, 'exp8_input_output.png');
```

Result:

- The FIR Hilbert Transformer of length $M = 15$ was successfully designed using the Hamming window.
- The magnitude and phase responses confirm that the filter introduces a constant phase shift of approximately 90°.
- The output $y(n)$ is a sine wave phase-shifted by 90° with respect to the input cosine signal.

Plots:

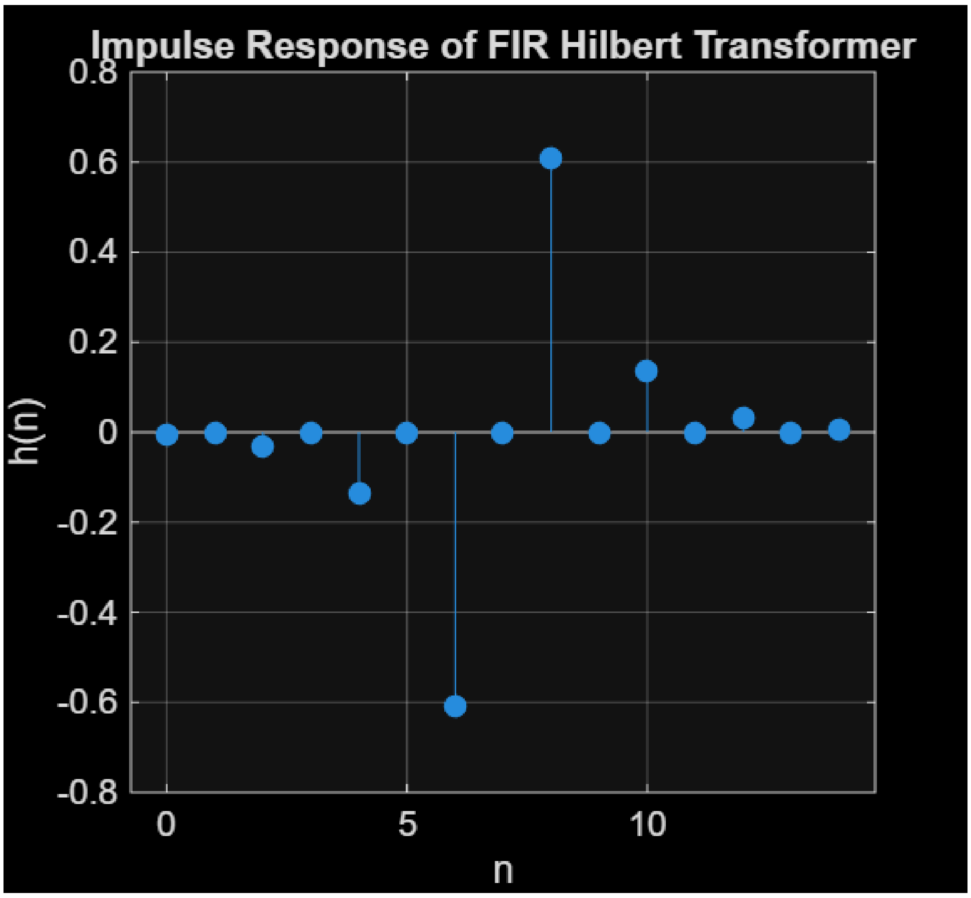


Figure 18: Impulse Response of FIR Hilbert Transformer

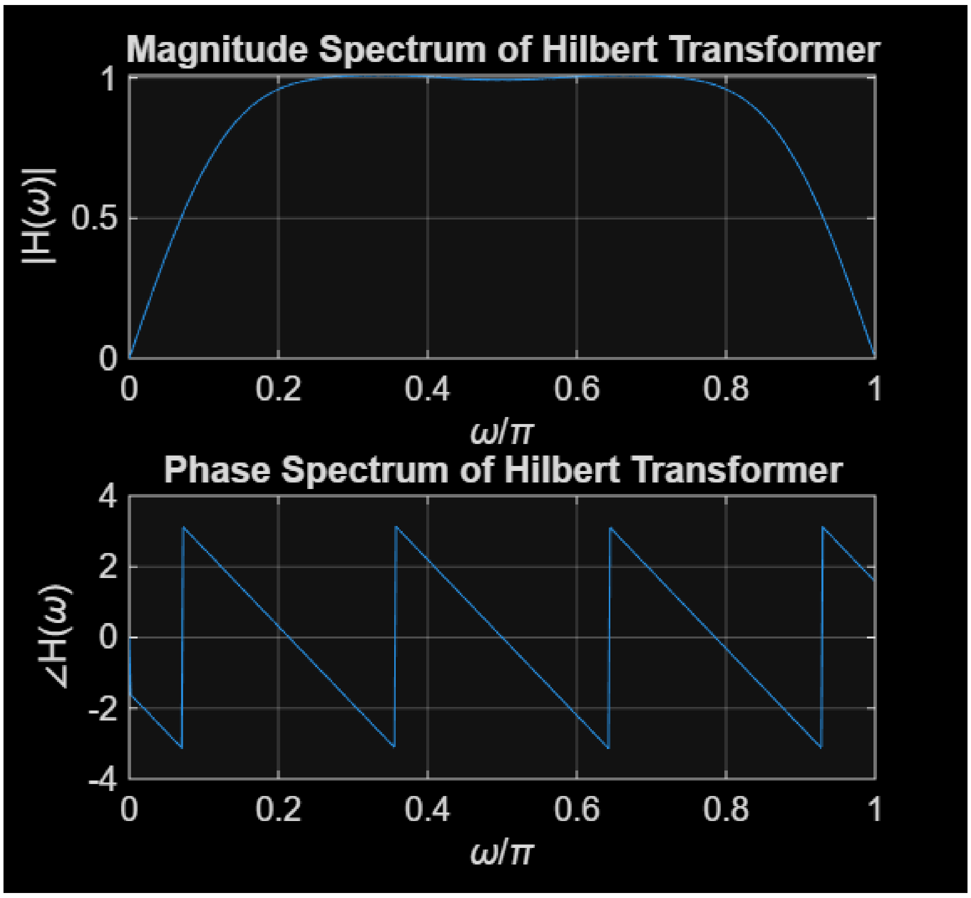


Figure 19: Magnitude and Phase Spectrum of Hilbert Transformer

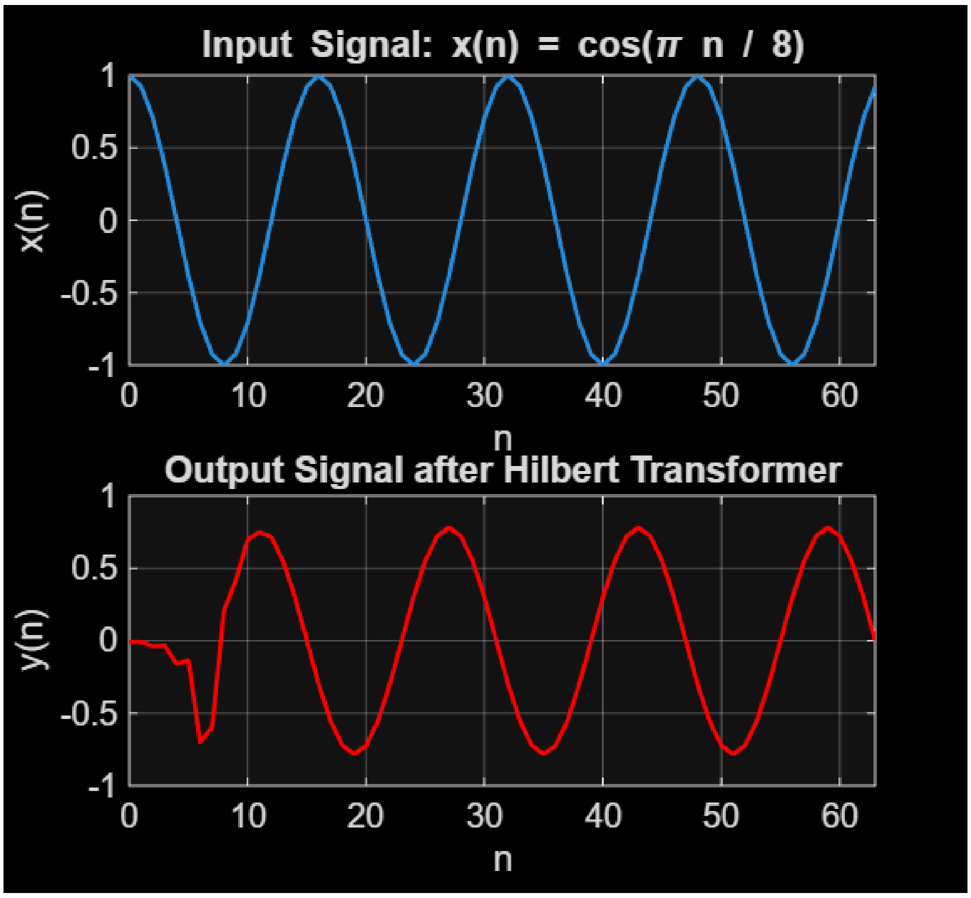


Figure 20: Input and Output Waveforms Showing Phase Shift

Conclusion:
The FIR Hilbert Transformer designed using the Hamming window successfully shifts the phase of the input cosine signal by 90°. This verifies its application in systems requiring quadrature signal generation such as modulation, demodulation, and analytic signal generation.

Viva Questions and Answers:

1. **What is the function of a Hilbert Transformer?**
   A Hilbert Transformer produces a phase shift of 90° for all frequency components of an input signal without altering their magnitudes. It effectively converts a real signal into a version that is phase-shifted by $\pi/2$, thereby generating the quadrature component of the input signal.

2. **Why is the Hilbert Transformer considered a phase-shifting filter?**
   The Hilbert Transformer has a frequency response with a constant magnitude and a linear phase shift of ±90° across the entire frequency band. Since it changes only the phase and not the amplitude of the spectral components, it is classified as a phase-shifting filter.

3. **What is the purpose of using a window function in FIR filter design?**
   Window functions are used to truncate the ideal, infinite-duration impulse response of a desired filter to a finite length. They control the trade-off between transition

bandwidth and stopband attenuation, reducing Gibbs oscillations and improving the frequency response smoothness.

4. **What is the difference between Hamming and Hanning windows?**
The Hamming window offers side-lobe attenuation of approximately 41 dB, while the Hanning (Hann) window provides about 44 dB. The Hanning window exhibits slightly lower side-lobe amplitude but a broader main lobe, resulting in a smoother transition band compared to the Hamming window.

5. **How can the Hilbert Transformer be used to generate an analytic signal?**
An analytic signal can be formed by combining a real signal $x(t)$ with its Hilbert-transformed version $\hat{x}(t)$ as:

$$z(t) = x(t) + j\hat{x}(t)$$

This analytic signal has all its energy concentrated in the positive frequency spectrum, which is useful in modulation, envelope detection, and signal analysis.