Department of Electrical Engineering
Indian Institute of Technology Kharagpur

Digital Signal Processing Laboratory (EE39203)
Autumn, 2022-23

Experiment 6: Discrete Fourier Transform
Slot: **X** Date: **13/10/2023**

Student Name: **P. Manoj Kumar** Roll No.: **21IE10027**

Grading Rubric

	Tick the best applicable per row			Points
	Below Expectation	Lacking in Some	Meets all Expectation	
Completeness of the report				
Organization of the report (5 pts) *With cover sheet, answers are in the same order as questions in the lab, copies of the questions are included in report, prepared in LaTeX*				
Quality of figures (5 pts) *Correctly labelled with title, x-axis, y-axis, and name(s)*				
Understanding the effects of truncating the signal on its DTFT (30 pts) *Magnitude and phase plots, hamming/rect windows, questions*				
Implementation of DFT and inverse DFT (30 pts) *Matlab codes, frequency and time-domain plots, analytical expressions*				
Implementation of DFT and IDFT using matrix multiplication (20 pts) *Matrices A,B,C, Matlab codes, plots, questions*				
Computation time comparison (10 pts) *Runtimes, questions*				
			TOTAL (100 pts)	

Total Points (100): TA Name: TA Initials:

Digital Signal Processing Laboratory (EE39203)

P Manoj Kumar (21IE10027)

Experiment 6 - Discrete Fourier Transform

1 Introduction

1.1 Discrete-Time Fourier Transform (DTFT)

The Discrete-Time Fourier Transform (DTFT) of a discrete signal $x(n)$ is defined as:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-jn\omega} \quad (1)$$

The inverse DTFT is given by:

$$x(n) = \frac{1}{2\pi}\int X(e^{j\omega})e^{j\omega n}d\omega \quad (2)$$

1.2 Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) is a sampled version of the DTFT, making it better suited for numerical evaluation on computers. It is defined as:

$$X_N(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} \quad (3)$$

The inverse DFT is given by:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X_N(k)e^{j2\pi kn/N} \quad (4)$$

Where $X_N(k)$ is an N-point DFT of $x(n)$.

1.3 Deriving the DFT from the DTFT : Truncating the Time-domain Signal

To compute the DTFT, we often use a truncated version of the signal due to computational constraints. Let $w(n)$ be a rectangular window of length N:

$$w(n) = \begin{cases} 1 & 0 \le n \le N-1 \\ 0 & \text{elsewhere} \end{cases} \quad (5)$$

The truncated signal $x_{\text{tr}}(n)$ is then defined as:

$$x_{\text{tr}}(n) = w(n)x(n) \quad (6)$$

The DTFT of $x_{\text{tr}}(n)$ is given by:

$$X_{\text{tr}}(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x_{\text{tr}}(n)e^{-jn\omega} \quad (7)$$

However, it's important to note that $X(e^{j\omega})$ and $X_{\text{tr}}(e^{j\omega})$ are generally not equal.

1.4 Relation between DTFT and Truncated DTFT

The relation between the true DTFT, $X(e^{j\omega})$, and the truncated DTFT, $X_{\text{tr}}(e^{j\omega})$, is given by:

$$X_{\text{tr}}(e^{j\omega}) = \frac{1}{2\pi}\int X(e^{j\sigma})W(e^{j(\omega-\sigma)})d\sigma \quad (9)$$

Where $W(e^{j\omega})$ is the DTFT of $w(n)$. For $\omega \neq 0, \pm 2\pi, \ldots$, $W(e^{j\omega})$ is given by:

$$W(e^{j\omega}) = e^{-j\omega(N-1)/2}\frac{\sin(\omega N/2)}{\sin(\omega/2)} \quad (11)$$

Notice that the magnitude of this function is similar to $\text{sinc}\left(\frac{\omega N}{2}\right)$, except that it is periodic in ω with period 2π.

1.5 Frequency Sampling

The relationship between the Discrete-Time Fourier Transform (DTFT) and the Discrete Fourier Transform (DFT) can be expressed using the formula:

$$\omega = \frac{2\pi k}{N}$$

For $k = 0, 1, \ldots, (N-1)$, we find that:

$$X_{\text{tr}}(e^{j\omega})\big|_{\omega=\frac{2\pi k}{N}} = \sum_{n=0}^{N-1} x(n)e^{-j\omega n} = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N} = X_N(k)$$

In short, the Discrete Fourier Transform (DFT) values result from sampling the DTFT of the truncated signal.

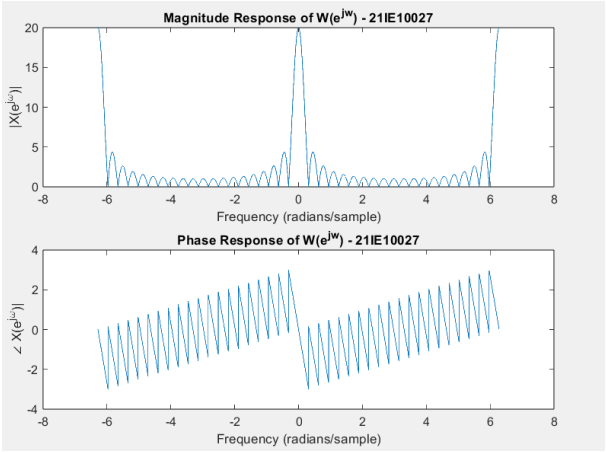
$$X_N(k) = X_{\text{tr}}(e^{j2\pi k/N})$$

2 Windowing Effects

2.1 Magnitude and Phase Response for $|W(e^{j\omega})|$

The MATLAB code and Frequency Response of the Window function is given below.

```
N = 20;                          % Length of signal
n = 0:N-1;                       % Discrete time indices
x_n = cos(pi .* n / 4);          % Generate input signal x[n]
w_i = -2 * pi : 0.001 : 2 * pi; % Define frequency range
% Calculate frequency response using sinc interpolation
W = exp((-1j*w_i)*(N - 1)/2) .* (sin(w_i*N/2)./sin(w_i/2));
W(w_i == 0 | w_i == 2*pi | w_i == -2*pi) = N;
% Plot magnitude and phase of the frequency response
figure(1)
subplot(2,1,1);
plot(w_i,abs(W));
title('Magnitude Response of W(e^{jw}) - 21IE10027');
xlabel('Frequency (radians/sample)');
ylabel('|X(e^{j\omega})|');
subplot(2,1,2);
plot(w_i,angle(W));
title('Phase Response of W(e^{jw}) - 21IE10027');
xlabel('Frequency (radians/sample)');
ylabel('\angle X(e^{j\omega})|');
```



2.2 Analytical Expression for $|X(e^{j\omega})|$

The analytical expression for $X(e^{j\omega})$ for the given signal $x[n] = \cos\left(\frac{\pi n}{4}\right)$ can be derived by taking the Discrete-Time Fourier Transform (DTFT) of $x[n]$. The DTFT of $x[n]$ is given by:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] \cdot e^{-j\omega n}$$

For $x[n] = \cos\left(\frac{\pi n}{4}\right)$, the DTFT can be calculated as follows:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \cos\left(\frac{\pi n}{4}\right) \cdot e^{-j\omega n}$$

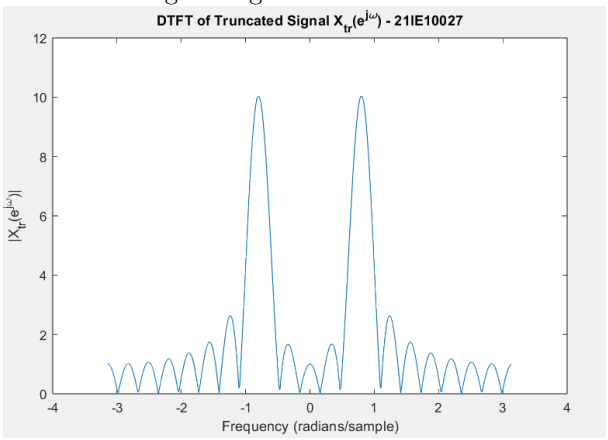
The analytical expression for $X(e^{j\omega})$ is:

$$X(e^{j\omega}) = \frac{1}{2}\left[\delta\left(\omega - \frac{\pi}{4}\right) + \delta\left(\omega + \frac{\pi}{4}\right)\right]$$

2.3 Magnitude Plot of $|X_{tr}(e^{j\omega})|$

The MATLAB code and Magnitude Response of the Truncated Signal is given below.

```
% Generate rectangular window
win = ones(1, N);
% Apply window to truncate signal
x_trunc = x_n .* win;
% Compute DTFT of truncated signal
figure(2)
[X, w] = DTFT(x_trunc, 512);
plot(w, abs(X));
title('DTFT of Truncated Signal X_{tr}(e^{j\omega}) - 21IE10027');
xlabel('Frequency (radians/sample)');
ylabel('|X_{tr}(e^{j\omega})|');
```



2.4 Difference between $|X_{\mathbf{tr}}(e^{j\omega})|$ and $|X(e^{j\omega})|$:

- $|X_{\mathrm{tr}}(e^{j\omega})|$ represents the magnitude spectrum of the signal after it has been truncated with a rectangular window. It will exhibit a sinc-like main lobe with sidelobes due to the convolution of the ideal spectrum with the Fourier transform of the rectangular window function.
- $|X(e^{j\omega})|$ represents the magnitude spectrum of the non-truncated signal ($x[n] = \cos\left(\frac{\pi n}{4}\right)$). This will have a single peak at $\omega = \frac{\pi}{4}$ due to the original cosine function.

  The main reason for the difference is the effect of the windowing operation. The rectangular window (truncation) in the time domain corresponds to a sinc function in the frequency domain. This sinc function smears the original spectrum, causing the main lobe to spread out and introduce sidelobes.

2.5 Effects of Using a Different Window Function:

If you were to use a Hamming window instead of the rectangular window:

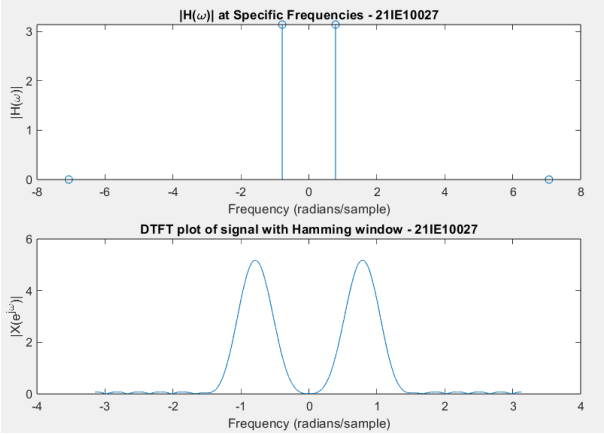
- The Hamming window will introduce a smoother transition from the main lobe to the sidelobes. It has lower sidelobes compared to the rectangular window.
- The main lobe width will be narrower compared to the rectangular window.
- The sidelobes will be attenuated.
- The spectral leakage due to the truncation effect will still be present, but it will be less severe compared to the rectangular window.

In summary, using a Hamming window will result in a better trade-off between main lobe width and sidelobe levels compared to a rectangular window. The spectral leakage will still be present, but it will be less pronounced.

2.6 DTFT of Signal with Hamming Window

The MATLAB code and Magnitude Response of the Signal with Hamming window is given below.

```
% Plot magnitude of DTFT at specific frequencies
figure(3)
subplot(2,1,1);
n_dtft = [-pi/4 - 2*pi, -pi/4, pi/4, pi/4 + 2*pi];
dtft = (n_dtft == -pi/4) + (n_dtft == pi/4);
stem(n_dtft, pi .* dtft)
title('|H(\omega)| at Specific Frequencies - 21IE10027')
xlabel('Frequency (radians/sample)');
ylabel('|H({\omega})|');
% Apply Hamming window to signal
subplot(2,1,2);
hamming_win = hamming(N);
x_ham = x_n' .* hamming_win;
[X3, w3] = DTFT(x_ham, 512);
plot(w3, abs(X3))
title('DTFT plot of signal with Hamming window - 21IE10027')
xlabel('Frequency (radians/sample)');
ylabel('|X(e^{j\omega})|');
```



3 Computation of DFT and IDFT

3.1 MATLAB Code for DFTSum and IDFTSum

```
function [X] = DFTsum(x)
    j=sqrt(-1);
    N=length(x);
    X=zeros(1,N); % Initialize the output array with zeros
    for k=1:N
        for n=1:N
            % Perform the DFT summation formula
            X(k)=X(k)+x(n)*exp(-j*2*pi*(k-1)*(n-1)/N);
        end
    end
end
```

```
function [x] = IDFTsum(X)
    j=sqrt(-1);
    N=length(X);
    x=zeros(1,N); % Initialize the output array with zeros
    for n=1:N
        for k=1:N
            % Perform the IDFT summation formula
            x(n)=x(n)+(1/N)*X(k)*exp(j*2*pi*(k-1)*(n-1)/N);
        end
    end
end
```

3.2 Analytical Expressions of DFT

- $x[n] = \delta[n]$ for $N = 10$.
- $x[n] = 1$ for $N = 10$.
- $x[n] = e^{j\frac{2\pi n}{10}}$ for $N = 10$.
- $x[n] = \cos\left(\frac{2\pi n}{10}\right)$ for $N = 10$.

The DFT $X_1(k)$ is given by:

$$X_1(k) = \sum_{n=0}^{N-1} x_1[n]e^{-j\frac{2\pi}{N}kn} = e^{-j\frac{2\pi}{N}k\cdot 0} = 1$$

The DFT $X_2(k)$ is given by:

$$X_2(k) = \sum_{n=0}^{N-1} x_2[n]e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} e^{-j\frac{2\pi}{N}kn} = N\delta[k]$$

The DFT $X_3(k)$ is given by:

$$X_3(k) = \sum_{n=0}^{N-1} x_3[n]e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} e^{j\frac{2\pi}{10}n}e^{-j\frac{2\pi}{N}kn} = 10\delta[k-1]$$

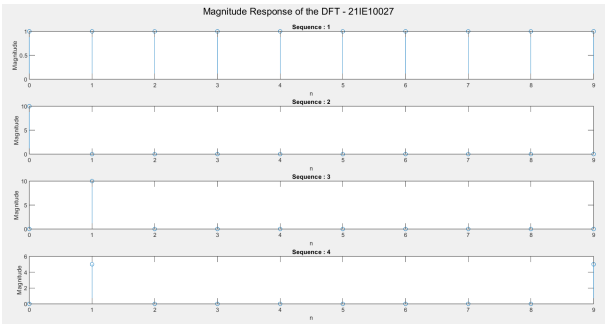
The DFT $X_4(k)$ is given by:

$$X_4(k) = \sum_{n=0}^{N-1} x_4[n]e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} \cos\left(\frac{2\pi}{10}n\right)e^{-j\frac{2\pi}{N}kn}$$

3.3 Magnitude Plot of DFT

The MATLAB code for the DFT of the following functions and the magnitude plots were given below.

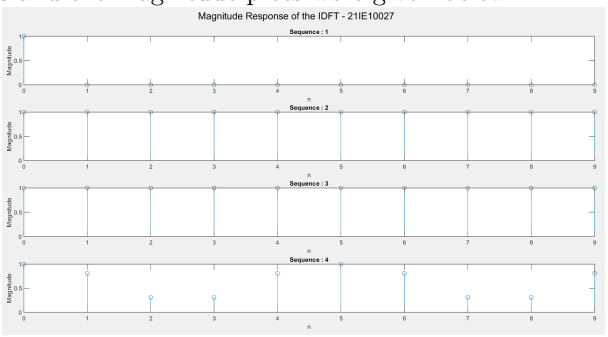
```
N = 10;       % Length of the sequence
n = 0:N-1;    % Discrete time indices
% Define different input sequences
x_n = { (n == 0), ones(1, length(n)), exp(1j * 2 * pi .* n ./ 10)...
    ,cos(2 * pi .* n ./ 10) };
% Calculate DFT for each input sequence and plot magnitude
figure;
for i = 1:length(x_n)
    X = DFTsum(x_n{i});
    subplot(4,1,i)
    stem(n, abs(X))
    xlabel('n');
    ylabel('Magnitude');
    title(['Sequence : ', num2str(i)])
end
sgtitle('Magnitude Response of the DFT - 21IE10027')
```



3.4 Magnitude Plot of IDFT

The MATLAB code for the IDFT time-domain plots and the magnitude plots were given below.

```
% Calculate and plot IDFT for each DFT result
figure;
for i = 1:length(x_n)
    X = DFTsum(x_n{i});
    x = IDFTsum(X);
    subplot(4,1,i)
    stem(n, abs(x))
    xlabel('n');
    ylabel('Magnitude');
    title(['Sequence : ', num2str(i)])
end
sgtitle('Magnitude Response of the IDFT - 21IE10027')
```



4 Matrix Representation of the DFT and IDFT

4.1 MATLAB Code for DFTMatrix and IDFTMatrix

```
function A = DFTmatrix(N)
    A = zeros(N); % Initialize DFT matrix A
    for k = 1:N
        for n = 1:N
            % Calculate the matrix elements using the DFT formula
            A(k,n) = exp(-1j * 2*pi * (k-1) * (n-1) / N);
        end
    end
end
```

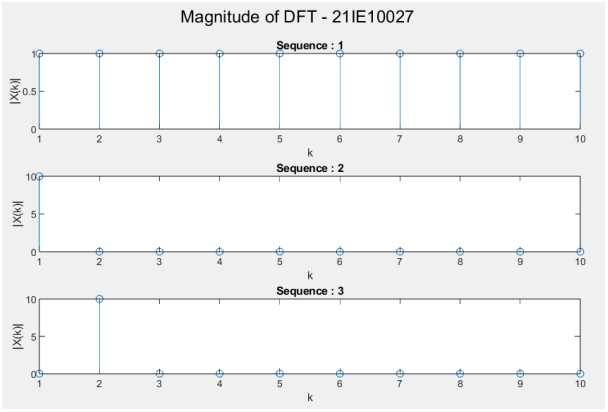
```
function B = IDFTmatrix(N)
    B = zeros(N); % Initialize a matrix of size N by N with zeros
    for k = 1:N
        for n = 1:N
            % Calculate the matrix elements using the IDFT formula
            B(k,n) = (1/N) * exp(1j * 2 * pi * (k-1) * (n-1) / N);
        end
    end
end
```

4.2 Magnitude Plot of DFT

The MATLAB code for computing DFT using DFTMatrix Multiplication with the below three functions is given below with the magnitude plots.

$$x[n] = \delta[n], \quad x[n] = 1, \quad x[n] = e^{j\frac{2\pi n}{N}} \quad \text{for } N = 10.$$

```matlab
% Define different input sequences
N = 10;       % Length of the sequence
n = 0:N-1;    % Discrete time indices
x_n = { (n == 0), ones(1, length(n)), exp(1j * 2 * pi .* n ./ N)};
% Calculate DFT for each input sequence
for i = 1:length(x_n)
    X = DFTmatrix(N) * x_n{i}.';
    % Plot magnitude
    subplot(3,1,i)
    stem(abs(X));
    xlabel('k');
    ylabel('|X(k)|');
    title(['Sequence : ', num2str(i)]);
end
sgtitle('Magnitude of DFT - 21IE10027');
```



4.3 Computation of Matrix A, B and BA for N=5

```matlab
N = 5;
A = DFTmatrix(N);
disp('Matrix A for N = 5:');
disp(A);
B = IDFTmatrix(N);
disp('Matrix B for N = 5:');
disp(B);
C = IDFTmatrix(N) * DFTmatrix(N);
disp('Matrix C (C = BA) for N = 5:');
disp(C);
```

```
Matrix A for N = 5:
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.3090 - 0.9511i  -0.8090 - 0.5878i  -0.8090 + 0.5878i   0.3090 + 0.9511i
   1.0000 + 0.0000i  -0.8090 - 0.5878i   0.3090 + 0.9511i   0.3090 - 0.9511i  -0.8090 + 0.5878i
   1.0000 + 0.0000i  -0.8090 + 0.5878i   0.3090 - 0.9511i   0.3090 + 0.9511i  -0.8090 - 0.5878i
   1.0000 + 0.0000i   0.3090 + 0.9511i  -0.8090 + 0.5878i  -0.8090 - 0.5878i   0.3090 - 0.9511i

Matrix B for N = 5:
   0.2000 + 0.0000i   0.2000 + 0.0000i   0.2000 + 0.0000i   0.2000 + 0.0000i   0.2000 + 0.0000i
   0.2000 + 0.0000i   0.0618 + 0.1902i  -0.1618 + 0.1176i  -0.1618 - 0.1176i   0.0618 - 0.1902i
   0.2000 + 0.0000i  -0.1618 + 0.1176i   0.0618 - 0.1902i   0.0618 + 0.1902i  -0.1618 - 0.1176i
   0.2000 + 0.0000i  -0.1618 - 0.1176i   0.0618 + 0.1902i   0.0618 - 0.1902i  -0.1618 + 0.1176i
   0.2000 + 0.0000i   0.0618 - 0.1902i  -0.1618 - 0.1176i  -0.1618 + 0.1176i   0.0618 + 0.1902i

Matrix C (C = BA) for N = 5:
   1.0000 + 0.0000i  -0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 - 0.0000i   0.0000 - 0.0000i
  -0.0000 + 0.0000i   1.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 - 0.0000i   0.0000 - 0.0000i
  -0.0000 + 0.0000i  -0.0000 + 0.0000i   1.0000 + 0.0000i  -0.0000 - 0.0000i  -0.0000 - 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   1.0000 - 0.0000i  -0.0000 - 0.0000i
   0.0000 + 0.0000i  -0.0000 + 0.0000i  -0.0000 + 0.0000i  -0.0000 - 0.0000i   1.0000 + 0.0000i
```

4.4 Discussions

- **How many multiplies are required to compute an N-point DFT using the matrix method?**

  For an N-point DFT using the matrix method, the number of complex multiplications is N^2, because each element of the output vector is computed using a sum of N products, and there are N elements in the output.

- **Analytical expression for the elements of the inverse DFT matrix B:**

  The elements of the inverse DFT matrix B are given by:

$$B_{kn} = \frac{1}{N} e^{j\frac{2\pi}{N}(k-1)(n-1)}, \quad 1 \leq k, n \leq N$$

  This expression is derived from the formula (16) with a negative sign and a normalization factor of $1/N$.

- **What form does $C = BA$ have? Why does it have this form?**

  The matrix $C = BA$ should be an identity matrix of size $N \times N$ (up to numerical precision errors). This is because the product of the DFT matrix A and the inverse DFT matrix B should result in the identity matrix. This is a fundamental property of the Discrete Fourier Transform.

5 Computation Time Comparison

5.1 MATLAB Code for Computing Time in DFTSum and Matrix Multiplication

```
N = 512; % Define the signal length N
% Generate the signal x[n] = cos(2πn/10) for N = 512
n = 0:N-1;
x = cos(2 * pi * n / 10);
% Compute the matrix A for N = 512
A = zeros(N, N);
for k = 1:N
    for n = 1:N
        A(k, n) = exp(-1i * 2 * pi * (k-1) * (n-1) / N);
    end
end
% Time the DFT summation implementation
tic;
X_dft = DFTsum(x);
t_1 = toc;
% Time the matrix multiplication implementation
tic;
X_matrix = A * x.';
t_2 = toc;
% Print the computation times
fprintf('Computation time for DFT summation: %.6f seconds\n', t_1);
fprintf('Computation time for matrix multiplication: %.6f seconds\n', t_2);
```

5.2 Observations

```
>> DSP_30
Computation time for DFT summation: 0.161723 seconds
Computation time for matrix multiplication: 0.021066 seconds
>> DSP_30
Computation time for DFT summation: 0.120204 seconds
Computation time for matrix multiplication: 0.026240 seconds
DSP_30
Computation time for DFT summation: 0.064200 seconds
Computation time for matrix multiplication: 0.003399 seconds
>> DSP_30
Computation time for DFT summation: 0.059730 seconds
Computation time for matrix multiplication: 0.000425 seconds
>> DSP_30
Computation time for DFT summation: 0.059654 seconds
Computation time for matrix multiplication: 0.000593 seconds
```

5.3 Computation Times

5.3.1 For DFT Summation Method:

Trial Number	Computation Time (s)
1	0.161723
2	0.120204
3	0.064200
4	0.059730
5	0.059654
Average Time	0.093702 seconds

5.3.2 For Matrix Multiplication Method:

Trial Number	Computation Time (s)
1	0.021066
2	0.026240
3	0.003399
4	0.000425
5	0.000593
Average Time	0.010144 seconds

5.4 Conclusion

The average computation time for the DFT summation method is approximately 0.093702 seconds, while the average computation time for the matrix multiplication method is approximately 0.010144 seconds.

The Matrix Multiplication Method is faster compared to the DFT Summation Method. This is evident from the significantly lower cputime values for the Matrix Multiplication Method.

The Matrix Multiplication Method typically requires less storage. This is because it performs a single matrix-vector multiplication, which involves fewer memory operations compared to the summation approach. The DFT Summation Method involves multiple summations and multiplications, potentially requiring more temporary storage.

Overall, the Matrix Multiplication Method not only offers better computational efficiency but also tends to be more memory-efficient. This makes it a preferred choice for computing the Discrete Fourier Transform in practice.