K. J. Somaiya College of Engineering, Mumbai-77
Somaiya Vidyavihar University

Batch: A-4	**Roll No.:** 16010122151
Experiment No. 4	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Compute DFT & IDFT of discrete time signals using Matlab.

Objective: To learn & understand the Fourier transform operations on discrete time signals.

Expected Outcome of Experiment:

CO	Outcome
CO3	Analyze signals in frequency domain through various image transforms

Books/ Journals/ Websites referred:

1. http://www.mathworks.com/support/
2. www.math.mtu.edu/~msgocken/intro/intro.html
3. www.mccormick.northwestern.edu/docs/efirst/matlab.pdf
4. A.Nagoor Kani “Digital Signal Processing”, 2nd Edition, TMH Education.

Pre Lab/ Prior Concepts:

Implementation details along with screenshots:

Given a sequence of N samples $f(n)$, indexed by $n = 0..N-1$, the Discrete Fourier Transform (DFT) is defined as $F(k)$, where $k=0..N-1$:

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

$F(k)$ are often called the 'Fourier Coefficients' or 'Harmonics'.

The sequence $f(n)$ can be calculated from $F(k)$ using the Inverse Discrete Fourier Transform (IDFT):

$$f(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F(k) e^{+j2\pi nk/N}$$

In general, both $f(n)$ and $F(k)$ are complex.

Annex A shows that the IDFT defined above really is an *inverse* DFT.

Conventionally, the sequences $f(n)$ and $F(k)$ is referred to as 'time domain' data and 'frequency domain' data respectively. Of course there is no reason why the samples in $f(n)$ need be samples of a time dependent signal. For example, they could be spatial image samples (though in such cases a 2 dimensional set would be more common).

Although we have stated that both n and k range over $0..N-1$, the definitions above have a periodicity of N:

$$F(k + N) = F(k) \qquad f(n + N) = f(n)$$

So both $f(n)$ and $F(k)$ are defined for all (integral) n and k respectively, but we only need to calculate values in the range $0..N-1$. Any other points can be obtained using the above periodicity property.

For the sake of simplicity, when considering various Fast Fourier Transform (FFT) algorithms, we shall ignore the scaling factors and simply define the FFT and Inverse FFT (IFFT) like this:

$$FFT_N(k,f) = \sum_{n=0}^{N-1} f(n)\, e^{-j2\pi kn/N} = \sqrt{N} F(k)$$

$$IFFT_N(n,F) = \sum_{k=0}^{N-1} F(k)\, e^{+j2\pi nk/N} = \sqrt{N} f(n)$$

In fact, we shall only consider the FFT algorithms in detail. The inverse FFT (IFFT) is easily obtained from the FFT.

Here are some simple DFT's expressed as matrix multiplications.

1 point DFT:

$$[F(0)] = [1][f(0)]$$

2 point DFT:

$$\begin{bmatrix} F(0) \\ F(1) \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \end{bmatrix}$$

4 point DFT:

$$\begin{bmatrix} F(0) \\ F(1) \\ F(2) \\ F(3) \end{bmatrix} = \frac{1}{\sqrt{4}} \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -j & -1 & +j \\ +1 & -1 & +1 & -1 \\ +1 & +j & -1 & -j \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{bmatrix}$$

3 point DFT:

$$\begin{bmatrix} F(0) \\ F(1) \\ F(2) \end{bmatrix} = \frac{1}{\sqrt{3}} \begin{bmatrix} +1 & +1 & +1 \\ +1 & X & X^* \\ +1 & X^* & X \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \end{bmatrix}$$

$$where \quad X = e^{-j2\pi/3} = COS(2\pi/3) - jSIN(2\pi/3) = -\left(\frac{1 + j\sqrt{3}}{2}\right)$$

Note that each of the matrix multipliers can be inverted by conjugating the elements. This what we would expect, given that the only difference between the DFT and IDFT is the sign of the complex exponential argument.

Here's another couple of useful transforms:

If..

$$f(n) = \delta(n - n_0) \qquad n_0 = 0..N - 1$$

$$= 1 \qquad if \ (n \, MOD \, N) = n_0$$

$$= 0 \qquad if \ (n \, MOD \, N) \neq n_0$$

This is the 'Delta Function'. The usual implied periodicity has been made explicit by using *MOD N*. The DFT is therefore:

$$F(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \delta(n - n_0) e^{-j2\pi kn/N} = \frac{e^{-j2\pi kn_0/N}}{\sqrt{N}}$$

This gives us the DFT of a unit impulse at $n=n_0$. Less obvious is this DFT:

If..

$$f(n) = e^{+j2\pi k_0 n/N} \qquad k_0 = 0..N - 1$$

$$F(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} e^{+j2\pi k_0 n/N} e^{-j2\pi kn/N} = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} e^{-j2\pi (k-k_0)n/N} = \sqrt{N}\delta(k - k_0)$$

Implementation steps with screenshots for DFT:

```
x = input('Enter the input signal (in square
brackets, e.g., [1 2 3 4 5 6 7 8]): ');

N = length(x);

N_fft = 2^nextpow2(N);

x_padded = [x, zeros(1, N_fft - N)];

bit_rev_indices = bitrevorder(0:N_fft-1);

x_bit_rev = x_padded(bit_rev_indices + 1);
x_bit_rev = complex(x_bit_rev);


disp('Bit-reversed input signal:');
disp(x_bit_rev);

n = 0:N_fft-1;

k = 0:N_fft-1;

WN = exp(-1i*2*pi/N_fft);

W = WN.^(n'*k);

X = x_bit_rev * W; disp(' ');

disp('Twiddle factors:');

disp(W);

disp(' ');

disp('DFT:'); disp(X);
```

```
Command Window
>> untitled2
Enter the input signal (in square brackets, e.g., [1 2 3 4 5 6 7 8]): [2,1,2,1,1,2,1,2]
Bit-reversed input signal:
  Columns 1 through 4

   2.0000 + 0.0000i   1.0000 + 0.0000i   2.0000 + 0.0000i   1.0000 + 0.0000i

  Columns 5 through 8

   1.0000 + 0.0000i   2.0000 + 0.0000i   1.0000 + 0.0000i   2.0000 + 0.0000i


Twiddle factors:
  Columns 1 through 4

   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.7071 - 0.7071i   0.0000 - 1.0000i  -0.7071 - 0.7071i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
   1.0000 + 0.0000i  -0.7071 - 0.7071i  -0.0000 + 1.0000i   0.7071 - 0.7071i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -0.7071 + 0.7071i   0.0000 - 1.0000i   0.7071 + 0.7071i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i
   1.0000 + 0.0000i   0.7071 + 0.7071i  -0.0000 + 1.0000i  -0.7071 + 0.7071i

  Columns 5 through 8

   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
  -1.0000 - 0.0000i  -0.7071 + 0.7071i  -0.0000 + 1.0000i   0.7071 + 0.7071i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
  -1.0000 - 0.0000i   0.7071 + 0.7071i   0.0000 - 1.0000i  -0.7071 + 0.7071i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
  -1.0000 - 0.0000i   0.7071 - 0.7071i  -0.0000 + 1.0000i  -0.7071 - 0.7071i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i
  -1.0000 - 0.0000i  -0.7071 - 0.7071i   0.0000 - 1.0000i   0.7071 - 0.7071i
```

```
DFT:
  Columns 1 through 4

  12.0000 + 0.0000i   1.0000 + 0.4142i  -0.0000 - 0.0000i   1.0000 + 2.4142i

  Columns 5 through 8

   0.0000 - 0.0000i   1.0000 - 2.4142i   0.0000 - 0.0000i   1.0000 - 0.4142i
```

Implementation steps with screenshots for IDFT.

```
function idft_test()
function x_reconstructed = idft_8pt(X)

  N = length(X);
  n = 0:N-1;
  k = 0:N-1;

 WN_IDFT = exp(1i*2*pi/N);

 W_IDFT = WN_IDFT.^(n'*k);

x_reconstructed = (X * W_IDFT) / N; end

disp('Enter the type of input:');

disp('1. Real numbers');

disp('2. Complex numbers');

   choice = input('Choice: ');

switch choice

case 1


x_real = input('Enter the real part of the input signal
(in square brackets, e.g., [1 2 3 4]): ');

N_real = length(x_real);


bit_rev_indices_real = bitrevorder(0:N_real-1);

x_bit_rev_real = x_real(bit_rev_indices_real + 1);
x_reconstructed_real = idft_8pt(x_bit_rev_real);

 disp('Reconstructed signal (real part):');

 disp(x_reconstructed_real);

  case 2
  disp('Enter the complex part of the input signal in the
  format [real1 imag1; real2 imag2; ...]:');

   x_complex = input('Input: '); N_complex =

   size(x_complex, 1);

    bit_rev_indices_complex = bitrevorder(0:N_complex-1);
```

```
x_bit_rev_complex = (x_complex(bit_rev_indices_complex +
1, 1) + 1i * x_complex(bit_rev_indices_complex + 1,
2)).';

x_reconstructed_complex = idft_8pt(x_bit_rev_complex);

disp('Reconstructed signal (complex part):');

disp(x_reconstructed_complex);

otherwise

disp('Invalid choice');

end

end
```

```
Command Window
>> untitled2
Enter the type of input:
1. Real numbers
2. Complex numbers
Choice: 1
Enter the real part of the input signal (in square brackets, e.g., [1 2 3 4]): [12,14,24,10]
Reconstructed signal (real part):
  15.0000 + 0.0000i  -0.5000 + 3.5000i  -2.0000 + 0.0000i  -0.5000 - 3.5000i

>> untitled2
Enter the type of input:
1. Real numbers
2. Complex numbers
Choice: 2
Enter the complex part of the input signal in the format [real1 imag1; real2 imag2; ...]:
Input: [10,2,25,4,144,45]
Reconstructed signal (complex part):
  10.0000 + 2.0000i
```

Conclusion:- From this experiment, we implemented the concept of Discrete Fourier Transform(DFT) and Inverse DFT using MATLAB.

Date: 02/04/2025 **Signature of faculty in-charge**

Post Lab Descriptive Questions

1. **Compare and discuss the computational efficiency of DFT and FFT**

The Discrete Fourier Transform (DFT) is mathematically defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi/N)kn}$$

where X[k] is the DFT of the sequence x[n], and N is the number of samples.

Computational Complexity of DFT:

- The direct computation of DFT requires N summations and N multiplications for each output frequency component k
- Since we need to compute N values, the total complexity is **O(N^2)**.

FFT (Fast Fourier Transform)

The FFT is an optimized algorithm for computing the DFT efficiently by exploiting symmetry and periodicity properties of the exponential terms.

Computational Complexity of FFT:

- The most common FFT algorithm, Cooley-Tukey, recursively divides the DFT computation into smaller parts using the divide-and-conquer approach.

- This reduces the computational complexity to O(NlogN), which is significantly faster than the O(N^2) complexity of the DFT, especially for large N.

2. **Give the properties of DFT and IDFT.**

DFT (Discrete Fourier Transform):

- Linearity: DFT is a linear operation.
- Periodicity: The DFT of a periodic sequence is also periodic.
- Time Shifting: Shifting a signal in the time domain results in phase modulation in the frequency domain.
- Frequency Shifting: Shifting a signal in the frequency domain results in modulation in the time domain.

IDFT (Inverse Discrete Fourier Transform):

- Linearity: IDFT is also a linear operation.
- Conjugate Symmetry: The input and output sequences of the IDFT are conjugate symmetric.
- Time Reversal: Reversing the time sequence of a signal in the time domain results in complex conjugation in the frequency domain.
- Circular Convolution: Circular convolution in the time domain corresponds to multiplication in the frequency domain.

3. **Discuss the impact on computation time & efficiency when the number of samples N increases.**

As the number of samples N increases, the computational time for both DFT and FFT also increases. However, the increase in time for DFT is quadratic (O(N^2)), while for FFT, it grows logarithmically (O(N log N)). Therefore, as N increases,

the computational efficiency of FFT becomes significantly better compared to DFT. This is because FFT exploits the periodicity and symmetry properties of the DFT to reduce the number of computations required, leading to faster computation times for larger datasets.

4. **How to compute maximum length N for a circular convolution using DFT and IDFT?**

To compute the maximum length N for circular convolution using DFT and IDFT, the length of the input sequences being convolved must be considered. The maximum length N for circular convolution is equal to the sum of the lengths of the input sequences minus one. Mathematically, N = M + L - 1, where M is the length of the first sequence and L is the length of the second sequence. This ensures that the circular convolution does not suffer from aliasing or truncation effects. By computing the DFT of both sequences, performing point-wise multiplication in the frequency domain, and then taking the inverse DFT of the result, circular convolution can be efficiently computed for signals of any length up to N.