K. J. Somaiya College of Engineering, Mumbai-77
(Autonomous College Affiliated to University of Mumbai)

Batch:___A1____	**Roll No.:__1911012 ___**
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:

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

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

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..

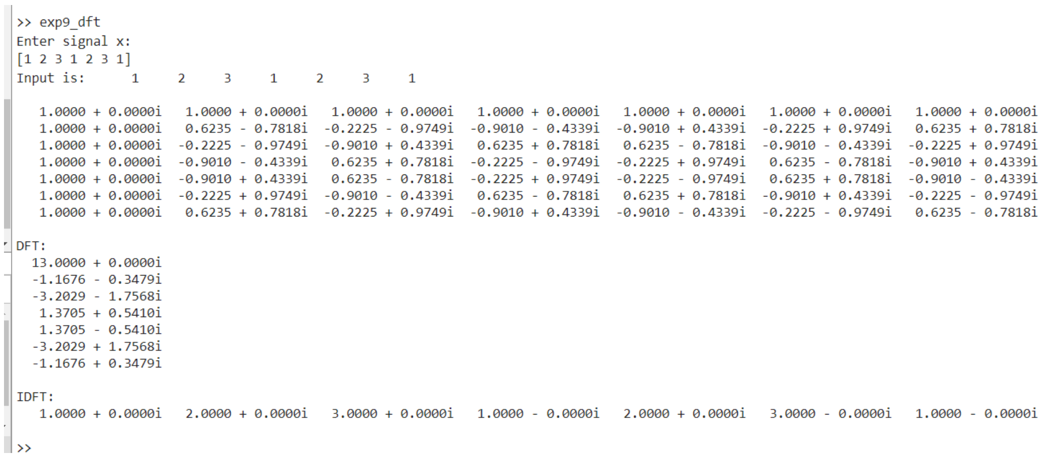
$$f(n) = e^{+j2\pi k_0 n/N} \qquad k_0 = 0..N - 1$$

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

Implementation steps with screenshots for DFT & IDFT:

```
x = input("Enter signal x: ");
n = length(x);
fprintf("Input is: ");
disp(x);
Wn = zeros(n,n);
for k = 0:1:n-1
    for l = 0:1:n-1
        Wn(k+1,l+1) = exp((-1j*2*pi*k*l/n));
    end
end

disp(Wn);
y = Wn * x'; fprintf("DFT: \n");
disp(y);

original = y' * Wn/n;
fprintf("IDFT: \n");
disp(original);
```

Output:

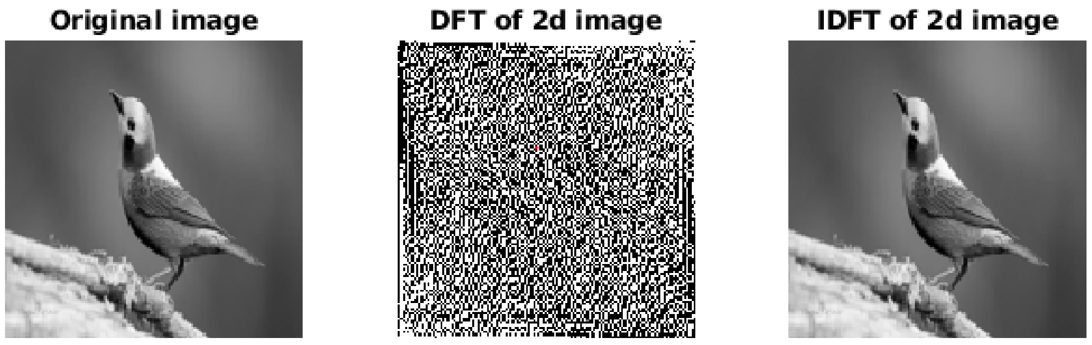


```
>> exp9_dft
Enter signal x:
[1 2 3 1 2 3 1]
Input is:     1     2     3     1     2     3     1

   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.6235 - 0.7818i  -0.2225 - 0.9749i  -0.9010 - 0.4339i  -0.9010 + 0.4339i  -0.2225 + 0.9749i   0.6235 + 0.7818i
   1.0000 + 0.0000i  -0.2225 - 0.9749i  -0.9010 + 0.4339i   0.6235 + 0.7818i   0.6235 - 0.7818i  -0.9010 - 0.4339i  -0.2225 + 0.9749i
   1.0000 + 0.0000i  -0.9010 - 0.4339i   0.6235 + 0.7818i  -0.2225 - 0.9749i  -0.2225 + 0.9749i   0.6235 - 0.7818i  -0.9010 + 0.4339i
   1.0000 + 0.0000i  -0.9010 + 0.4339i   0.6235 - 0.7818i  -0.2225 + 0.9749i  -0.2225 - 0.9749i   0.6235 + 0.7818i  -0.9010 - 0.4339i
   1.0000 + 0.0000i  -0.2225 + 0.9749i  -0.9010 - 0.4339i   0.6235 - 0.7818i   0.6235 + 0.7818i  -0.9010 + 0.4339i  -0.2225 - 0.9749i
   1.0000 + 0.0000i   0.6235 + 0.7818i  -0.2225 + 0.9749i  -0.9010 + 0.4339i  -0.9010 - 0.4339i  -0.2225 - 0.9749i   0.6235 - 0.7818i

DFT:
  13.0000 + 0.0000i
  -1.1676 - 0.3479i
  -3.2029 - 1.7568i
   1.3705 + 0.5410i
   1.3705 - 0.5410i
  -3.2029 + 1.7568i
  -1.1676 + 0.3479i

IDFT:
   1.0000 + 0.0000i   2.0000 + 0.0000i   3.0000 + 0.0000i   1.0000 - 0.0000i   2.0000 + 0.0000i   3.0000 - 0.0000i   1.0000 - 0.0000i

>>
```

2D Image:

```
a = imread('bird.jpg');
img = rgb2gray(a);
img = imresize(img,[128,128]);
s = size(img);
subplot(1,3,1)
imshow(img)
title('Original image')
tw = zeros(128,128);
for j=0:127
 for h=0:127
 tw(j+1,h+1) = exp((-1i * 2 * pi * j * h)/128);
 end
end
dftimg = tw * double(img) * tw';
subplot(1,3,2)
imshow(real(dftimg))
title('DFT image')
%idft
idftimg = (1/(128*128)) * (tw' * double(dftimg) * tw);
subplot(1,3,3)
imshow(idftimg,[])
title('IDFT image')
```

OUTPUT:



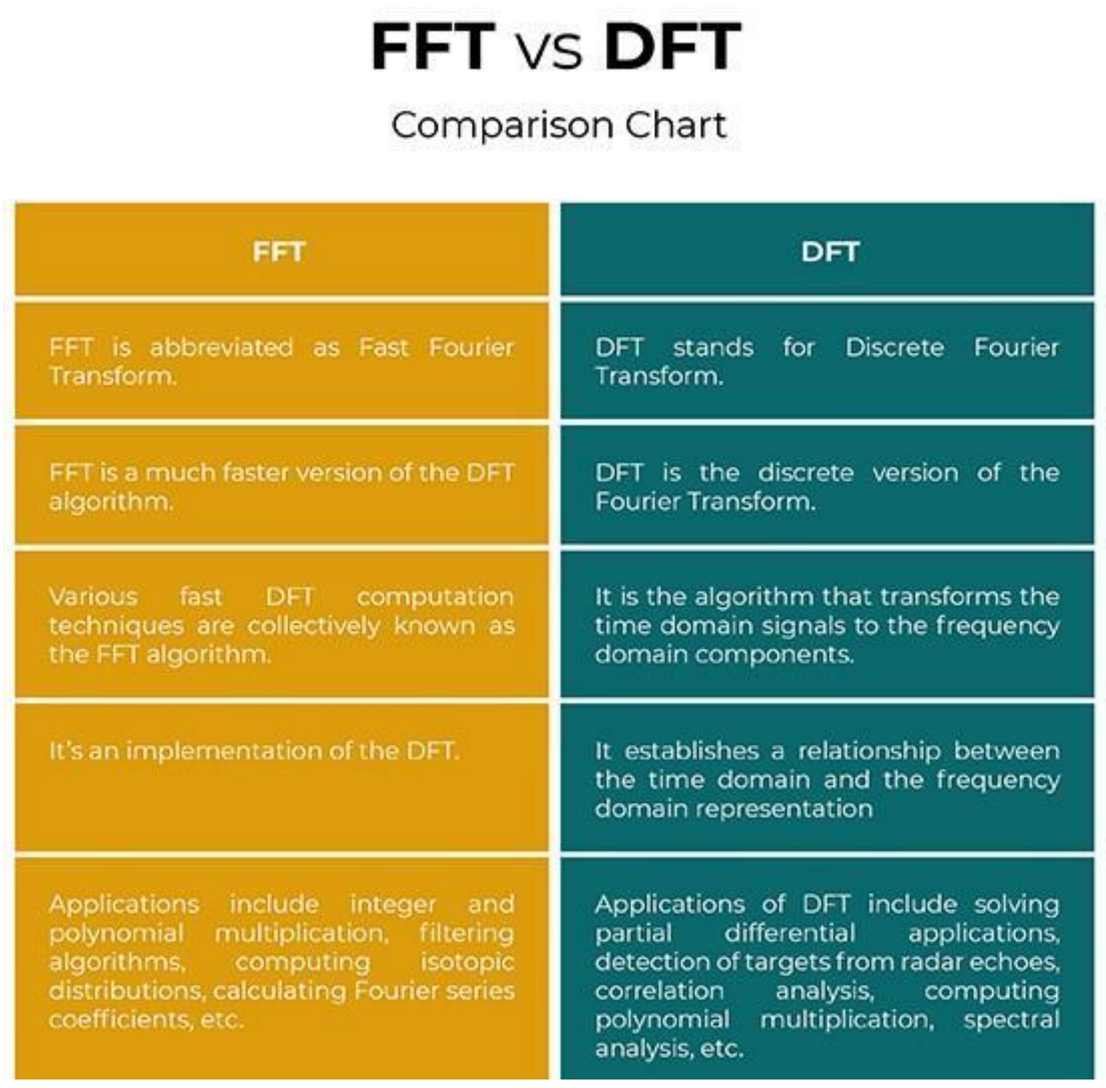
Conclusion:- :- Hence, we learn and understand the Fourier transform operations successfully.

Date: ___18/04/2022___ **Signature of faculty in-charge**

Post Lab Descriptive Questions

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

Ans:



FFT vs DFT

Comparison Chart

FFT	DFT
FFT is abbreviated as Fast Fourier Transform.	DFT stands for Discrete Fourier Transform.
FFT is a much faster version of the DFT algorithm.	DFT is the discrete version of the Fourier Transform.
Various fast DFT computation techniques are collectively known as the FFT algorithm.	It is the algorithm that transforms the time domain signals to the frequency domain components.
It's an implementation of the DFT.	It establishes a relationship between the time domain and the frequency domain representation
Applications include integer and polynomial multiplication, filtering algorithms, computing isotopic distributions, calculating Fourier series coefficients, etc.	Applications of DFT include solving partial differential applications, detection of targets from radar echoes, correlation analysis, computing polynomial multiplication, spectral analysis, etc.

2. Give the properties of DFT and IDFT.

Ans:

Property	Mathematical Representation
Linearity	$a_1x_1(n)+a_2x_2(n) \xleftrightarrow{DFT} a_1X_1(k) + a_2X_2(k)$

Periodicity	if x(n+N) = x(n) for all n then x(k+N) = X(k) for all k
Time reversal	$x(N-n) \xleftrightarrow{DFT} X(N-k)$
Duality	$x(n) \xleftrightarrow{DFT} Nx[((-k))_N]$
Circular convolution	$x_1(n) \circledcirc x_2(n) \xrightarrow{DFT} X_1(k).X_2(k)$
Circular correlation	For x(n) and y(n), circular correlation $r_{xy}(l)$ is $r_{xy}(l) \xleftrightarrow{DFT} R_{xy}(k) = X(k).Y^*(k)$
Circular frequency shift	$x(n)e^{2\pi jln/N} \xleftrightarrow{DFT} X(k+l)$ $x(n)e^{-2\pi jln/N} \xleftrightarrow{DFT} X(k-l)$
Circular time shift	$x((n-l))N = x(n-l) \xleftrightarrow{DFT} X(k)e^{-2\pi jlk/N}$ or $X(k)W^{kl}_N$ where W is the twiddle factor.
Circular symmetries of a sequence	If the circular shift is in • anti-clockwise direction (positive): Delayed discrete-time signal • clockwise direction (negative): Advanced discrete-time signal • Time reversal: Obtained by reversing samples of the discrete-time sequence about zero axis/locating x(n) in a clockwise direction.
Multiplication	$x_1(n) \;.\; x_2(n) \xrightarrow{DFT} \frac{1}{N}[X_1(k) \circledcirc X_2(k)]$
Complex conjugate	$x^*(n) \xleftrightarrow{DFT} X^*(N-k)$
Symmetry	For even sequences: $X(k) = \sum_{n=0}^{N-1} x(n)Cos(2\pi nk/N)$ For odd sequences:

	$X(k) = \sum_{n=0}^{N-1} x(n)\mathrm{Sin}(2\pi nk/N)$
Parseval's theorem	$\sum_{n=0}^{N-1} x(n).y^*(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k).Y^*(k)$

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

Ans:

The length N of the DFT is the number of frequency points that will result in the DFT output. Zero padding will result in more frequency samples, however this does not increase frequency resolution, it just interpolates samples in the DTFT. The frequency resolution is given by 1/T where T is the time length of your data (regardless of sampling rate). So if you want to increase the actual frequency resolution, you need to increase the number of samples at a given sampling rate, or decrease the sampling rate which would increase the time length for the number of samples you have.

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

Ans:

The DFT provides a convenient way to perform circular convolution. Specifically, if x1(n) is M points long and x2(n) is L points long, the circular convolution may be computed as follows:

1. Zero padding is performed to the sequence which is having lesser length, so that the lengths of both the sequences is N = max(L,M)

2. Find the N -point DFTs of x1(n) and x2(n)

3. Multiply the DFTs to form the product Y (k) = X1(k)X2(k) .

4. Find the inverse DFT of Y (k).