Batch: C-1 Roll No.: 16010122323

Experiment No. 3

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:

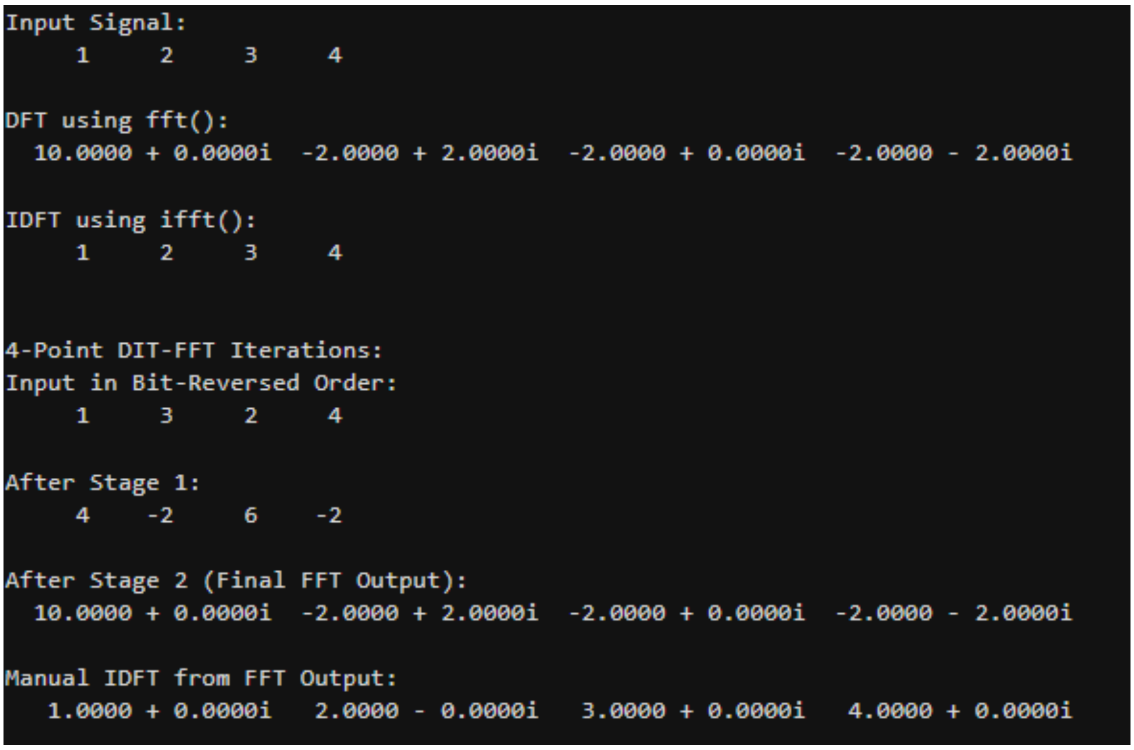
If..

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

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

Implementation steps with screenshots:

```
x = [1 2 3 4];
N = length(x);

fprintf('Input Signal:\n');
disp(x);

% --- DFT using built-in function ---
X_dft = fft(x);
fprintf('DFT using fft():\n');
disp(X_dft);

% --- IDFT using built-in function ---
x_idft = ifft(X_dft);
fprintf('IDFT using ifft():\n');
disp(x_idft);

% --- 4-Point DIT-FFT Implementation ---
fprintf('\n4-Point DIT-FFT Iterations:\n');

% Bit reversal order for N=4
x_bitrev = [x(1), x(3), x(2), x(4)];
fprintf('Input in Bit-Reversed Order:\n');
disp(x_bitrev);

% Stage 1
stage1 = zeros(1, N);
for k = 1:2:N
    stage1(k)   = x_bitrev(k) + x_bitrev(k+1);
    stage1(k+1) = x_bitrev(k) - x_bitrev(k+1);
end
fprintf('After Stage 1:\n');
disp(stage1);

% Stage 2
W = exp(-1j * 2 * pi / N);
stage2 = zeros(1, N);
stage2(1) = stage1(1) + stage1(3);
stage2(2) = stage1(2) + W^1 * stage1(4);
stage2(3) = stage1(1) - stage1(3);
stage2(4) = stage1(2) - W^1 * stage1(4);

fprintf('After Stage 2 (Final FFT Output):\n');
disp(stage2);

% --- Manual IDFT from FFT Output (using conjugation) ---
x_manual_idft = conj(fft(conj(stage2))) / N;
fprintf('Manual IDFT from FFT Output:\n');
disp(x_manual_idft);
```



```
Input Signal:
     1     2     3     4

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

IDFT using ifft():
     1     2     3     4


4-Point DIT-FFT Iterations:
Input in Bit-Reversed Order:
     1     3     2     4

After Stage 1:
     4    -2     6    -2

After Stage 2 (Final FFT Output):
  10.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 - 2.0000i

Manual IDFT from FFT Output:
   1.0000 + 0.0000i   2.0000 - 0.0000i   3.0000 + 0.0000i   4.0000 + 0.0000i
```

Conclusion:- In this experiment, we learned to compute DFT & IDFT of discrete time signals using Matlab

Date: _____________ **Signature of faculty in-charge**

Post Lab Descriptive Questions

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

Ans)

Aspect	DFT	FFT (Fast Fourier Transform)
Complexity	O(N2)O(N^2)	$O(N\log_2 N)$
Speed	Slower for large N	Much faster, especially for large N
Algorithm Type	Direct computation	Recursive divide-and-conquer (DIT/DIF)
Applications	Theoretical & small computations	Real-time systems, DSP, audio/video processing

· **DFT** requires N2N^2N2 complex multiplications and additions, which becomes expensive as N grows.

· **FFT** reduces redundancy in calculations by breaking down the DFT into smaller parts using symmetry and periodicity, making it **orders of magnitude faster**.

2. Give the properties of DFT and IDFT.

Ans)

Property	Explanation
Linearity	DFT{a·x[n] + b·y[n]} = a·X[k] + b·Y[k]
Time Shifting	$x[n - n_0] \longleftrightarrow X[k]\cdot\exp(-j\cdot 2\pi\cdot k\cdot n_0/N)$
Frequency Shifting	$x[n]\cdot\exp(j\cdot 2\pi\cdot k_0\cdot n/N) \longleftrightarrow X[(k - k_0) \bmod N]$
Conjugation	$x^*[n] \longleftrightarrow X^*[-k \bmod N]$
Parseval's Theorem	Total energy in time = Total energy in freq
Circular Convolution	$x[n] \circledast y[n] \longleftrightarrow X[k]\cdot Y[k]$
Symmetry (Real Signals)	If x[n] real, then $X[N-k] = X^*[k]$
Duality	$x[n] \longleftrightarrow X[k]$ implies $X[n] \longleftrightarrow N\cdot x[-k \bmod N]$

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

Ans)

N (Samples)	DFT Operations	FFT Operations	Observation
8	64	24	FFT is ~3× faster
1024	~1 million	~10,240	FFT is ~100× faster
10^6	10^{12} ops	~2×10^7 ops	FFT is ~50,000× faster

· As **N increases**, **DFT becomes impractical** due to its quadratic time.

· **FFT scales logarithmically**, making it ideal for real-time signal processing, radar, medical imaging, etc.

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

Ans) To **perform linear convolution using DFT**, we must **avoid time-aliasing**, so:

Formula:

N≥L+M−1N \geq L + M - 1N≥L+M−1

Where:

- L = length of first sequence
- M = length of second sequence
- N = DFT size (zero-padded if necessary)

Then:

1. **Zero-pad** both sequences to length N
2. Compute DFT → multiply → IDFT

Example:

- x[n]x[n]x[n] of length 5
- h[n]h[n]h[n] of length 4
  Then:
  $N \geq 5+4-1=8$

So pad both to length 8 before DFT.