

| **Title: C**ompute 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:

equation

*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):

equation

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*:

equation

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:

equation

equation

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

**Implementation steps with screenshots for DFT and IDFT.**

**CODE:**

clc;

clear;

close all;

N = input('Enter the size of DFT matrix (must be a power of 2): ');

if mod(log2(N),1) ~= 0

error('Error: Size must be a power of 2.');

end

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

n = 0:N-1;

k = 0:N-1;

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

disp('DFT Matrix:');

disp(W);

img\_name = input('Enter the image name with extension (e.g., img.png): ', 's');

img = imread(img\_name);

if size(img,3)==3

img\_gray = rgb2gray(img);

else

img\_gray = img;

end

img\_double = double(img\_gray);

[m, n\_img] = size(img\_double);

if m~=N || n\_img~=N

prompt = 'Image dimensions do not match the specified DFT size. Resize image to match? Y/N: ';

resizeChoice = input(prompt, 's');

if lower(resizeChoice)=='y'

img\_double = imresize(img\_double, [N N]);

else

error('Error: Image dimensions must match the size of the DFT matrix.');

end

end

DFT\_img = W \* img\_double \* W';

W\_inv = conj(W) / N;

IDFT\_img = real(W\_inv \* DFT\_img \* W\_inv');

figure;

subplot(1,3,1);

imshow(uint8(img\_double));

title('Original Image');

subplot(1,3,2);

imshow(log(1+abs(DFT\_img)), []);

title('DFT of the Image');

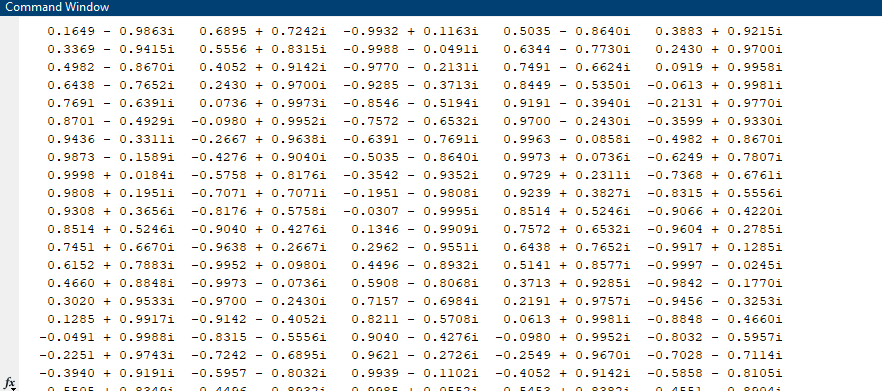
subplot(1,3,3);

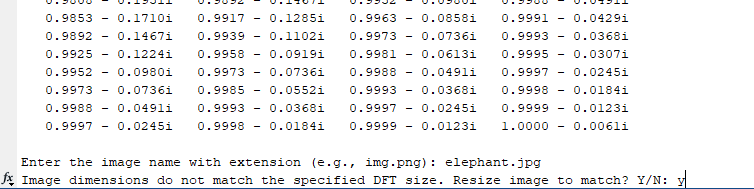
imshow(uint8(IDFT\_img));

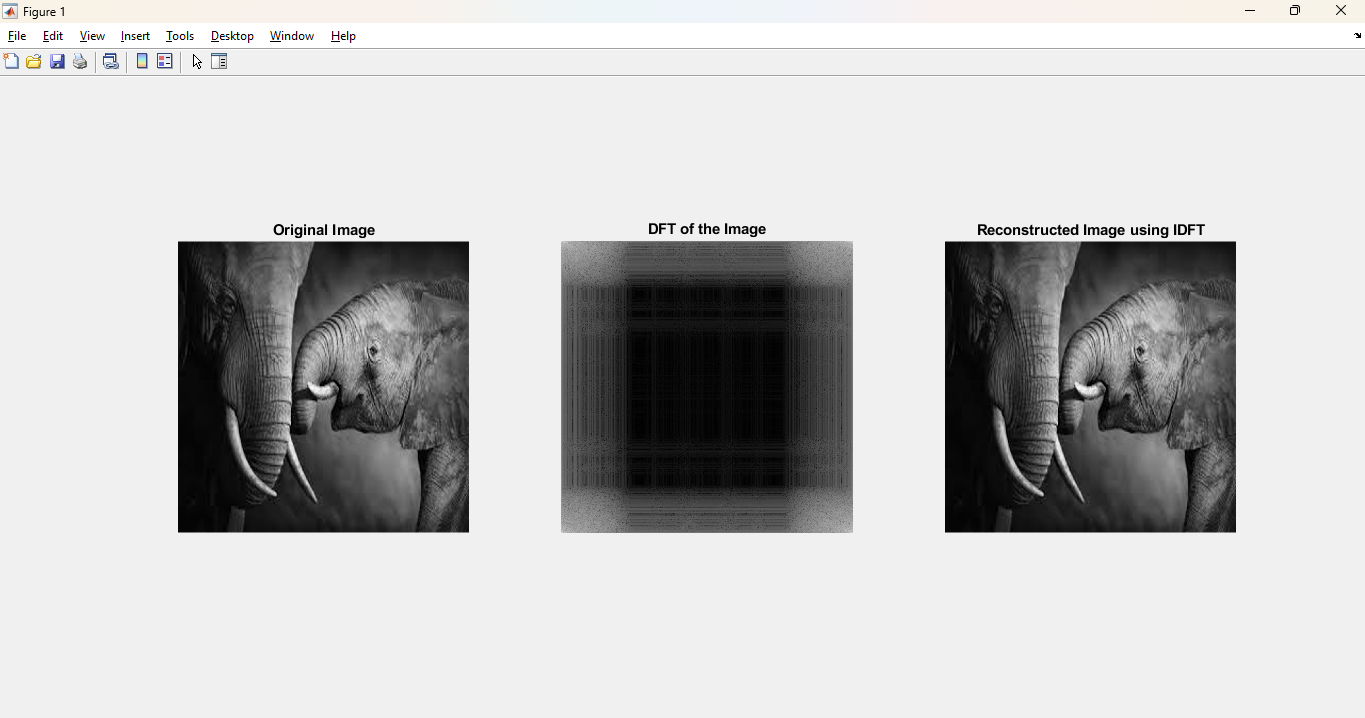
title('Reconstructed Image using IDFT');

**OUTPUT:**

****

****

****

****

**Conclusion:-**

The DFT and IDFT successfully transform and reconstruct the image, demonstrating frequency domain analysis and verifying accurate image reconstruction.

**Date: 28/03/25 Signature of faculty in-charge**

**Post Lab Descriptive Questions**

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

| **Feature** | **DFT** | **FFT** |
| --- | --- | --- |
| **Algorithm Type** | Matrix Multiplication | Divide-and-Conquer (Radix-2) |
| **Time Complexity** | O(N2) | O(Nlog⁡N) |
| **Efficiency** | Slow for large inputs | Faster for large inputs |
| **Memory Usage** | High | Low |
| **Application** | Theoretical analysis | Real-time processing |

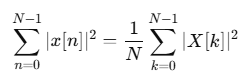
**DFT:** Direct computation involves N2 operations, making it inefficient for large data.

**FFT:** Radix-2 FFT reduces operations to Nlog⁡N, significantly improving speed, especially for power-of-2 lengths.

1. Give the properties of DFT and IDFT.

#### DFT Properties:

1. **Linearity:** 
2. **Shift Theorem:** 
3. **Convolution Theorem:  
   **
4. **Symmetry:**
   * Real input: Symmetric magnitude, antisymmetric phase.
5. **Parseval’s Theorem:**

****

#### IDFT Properties:

1. **Inverse of DFT:**
2. **Linearity:** Same as DFT.
3. **Shift Property:** Reverses frequency-domain shift.
4. **Circular Convolution Property:** Recovers original sequence after convolution in frequency domain.

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

**DFT:**

* As NNN increases, complexity grows as O(N2)
* Becomes computationally expensive for large signals.

**FFT:**

* With O(Nlog⁡N) complexity, FFT remains efficient even for large input sizes.
* Suitable for real-time and high-sample-rate applications.

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

**Circular Convolution Length:**

* For two sequences of length LLL and MMM, the maximum length NNN should be:**N≥L+M−1**

**Steps:**

* Zero-pad both sequences to length NNN.
* Apply DFT, multiply in the frequency domain.
* Apply IDFT to obtain circular convolution.