Experiment 4 : Fast Fourier transform

Name	Prathamesh Mane
UID no. & Branch	**2022200078 (B1)**
Experiment No.	4

AIM:	The aim of this experiment is to implement computationally Fast Algorithms.
OBJECTIVE:	1. Develop a program to perform FFT of N point Signal. 2. Calculate FFT of a given DT signal and verify the results using mathematical formula. 3. Computational efficiency of FFT.
INPUT SPECIFICATION:	**1.** Length of first Signal N **2.** DT Signal values
PROBLEM DEFINITION:	(1) Take any four-point sequence x[n]. Find FFT of x[n] and IFFT of {X[k]}. (2) Calculate Real and Complex Additions & Multiplications involved to find X[k]

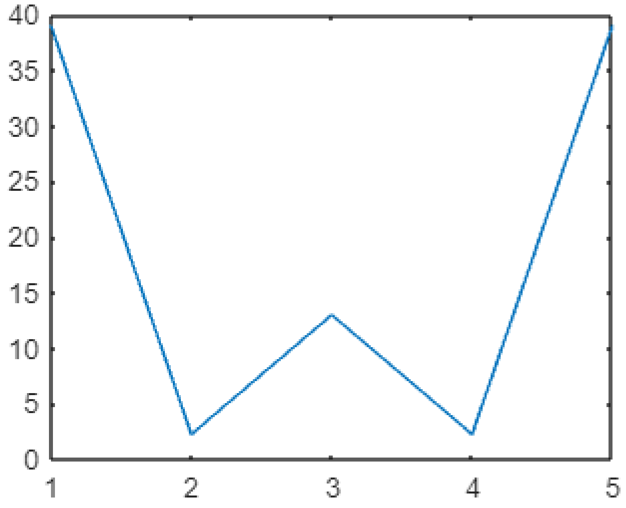
Theoretical solution

- **Case 1 : Question : A= [6,12,7,14] length L=4**

Result analysis :

A= [6,12,7,14]
N=4
X[k] = [39, -1 + 2j, -13, -1 - 2j]
Magnitude : [39 , 2.24 , 13 , 2.24]

Magnitude spectrum :



Code result :

```
Enter the length of x[n] (4 pt or 8 pt) = : 4
Enter the values of x[n]: 6 12 7 14

Input signal x[n] =   6.00    12.00    7.00    14.00

FFT results X[k] = :

  39.000  + j    0.000
  -1.000  + j    2.000
 -13.000  + j    0.000
  -1.000  + j   -2.000


Inverse FFT results x[n] = :

   6.000  + j    0.000
  12.000  + j   -0.000
   7.000  + j    0.000
  14.000  + j   -0.000
```

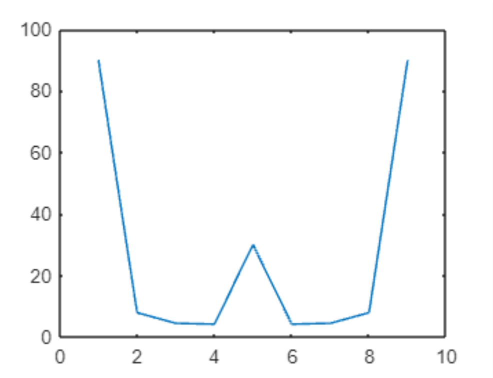
- **Case 2: Question : A= [6,12,7,14,8,16,9,18]**

Result analysis :

N=8
X[k] = [90.000 + j 0.000
-2.000 + j7.657
-2.000 + j4.000
-2.000 + j3.657
-30.000 + j0.000
-2.000 + j-3.657
-2.000 + j-4.000
-2.000 + j-7.657]

Magnitude : [90,7.91,4.47,4.1681,30,4.168,4.47,7.91]

Magnitude spectrum :



Code result :

```
Enter the length of x[n] (4 pt or 8 pt) = : 8
Enter the values of x[n]: 6 12 7 14 8 16 9 18

Input signal x[n] =   6.00    12.00    7.00    14.00    8.00    16.00    9.00    18.00

FFT results X[k] = :

  90.000  + j    0.000
  -2.000  + j    7.657
  -2.000  + j    4.000
  -2.000  + j    3.657
 -30.000  + j    0.000
  -2.000  + j   -3.657
  -2.000  + j   -4.000
  -2.000  + j   -7.657


Inverse FFT results x[n] = :

   6.000  + j    0.000
  12.000  + j    0.000
   7.000  + j    0.000
  14.000  + j   -0.000
   8.000  + j    0.000
  16.000  + j    0.000
   9.000  + j    0.000
  18.000  + j   -0.000
```

CONCLUSION:	**1. Computational Efficiency in DFT** **For N=4N :** • **Total Real Multiplications:** 4×4^2 = 64 • **Total Real Additions:** 4×4^2−2×4=64−8=56 **For N=8N :** • **Total Real Multiplications:** 4×82=256 • **Total Real Additions:** 4×8^2−2×8=256−16=240 **2. Computational Efficiency in FFT** **For N=4:** • **Total Real Multiplications:** 2×4×log2(4) =2×4×2 =16 • **Total Real Additions:** 3×4×log2(4) =3×4×2 =24 **For N=8 :** • **Total Real Multiplications:** 2×8×log2(8) =2×8×3 =48 • **Total Real Additions:** 3×8×log2(8) =3×8×3 =72 **3. FFT Performance** **The FFT produces fast results due to:** • **Less Computations:** The FFT significantly reduces the number of multiplications and additions compared to the DFT. For N=4N and N=8, the reduction in operations is substantial. The FFT dramatically reduces computational complexity compared to the DFT. For small N like 4 and 8, the difference is clear and substantial. As N increases, the efficiency of the FFT becomes even more pronounced, making it the preferred choice for larger datasets. The FFT achieves this efficiency through a reduction in the total number of computations and the potential for parallel processing, which enhances performance in practical applications.