Lecture - 10

Efficient Computation of DFT

2.1 Introduction

We studied Discrete Fourier Transform (DFT) earlier. The DFT is used in large number of applications of DSP such as filtering, correlation analysis, spectrum analysis etc. But the direct computation of DFT involves large number of computations. Hence the processor remain busy. Special algorithms have been developed to compute DFT quickly. These algorithms exploit the periodicity and symmetry properties of twiddle factors (phase factors). Hence DFT is computed fast using such algorithms compared to direct computation. These algorithms are collectively called as Fast Fourier Transform (FFT) algorithms. These algorithms are very efficient in terms of computations. As the value of 'N' increases, the computational efficiency of FFT algorithms increases.

2.2 Efficient Computation of the DFT

2.2.1 Direct Computation of DFT

Before directly starting study of FFT algorithms, let us see the complexity of computations in the direct computation of DFT. By definition, the DFT is given as,

$$X(k) = \sum_{n=0}^{N-1} x(n)\, W_N^{kn}, \qquad k = 0, 1, \ldots . N-1$$

Here $x(n)$ is the input sequence which can be real or complex. And W_N is the twiddle factor or phase factor which is complex number. Thus computation of $X(k)$ involves the multiplications and summations of complex numbers. Fig. 2.2.1 shows the expansion of summation of the above equation.

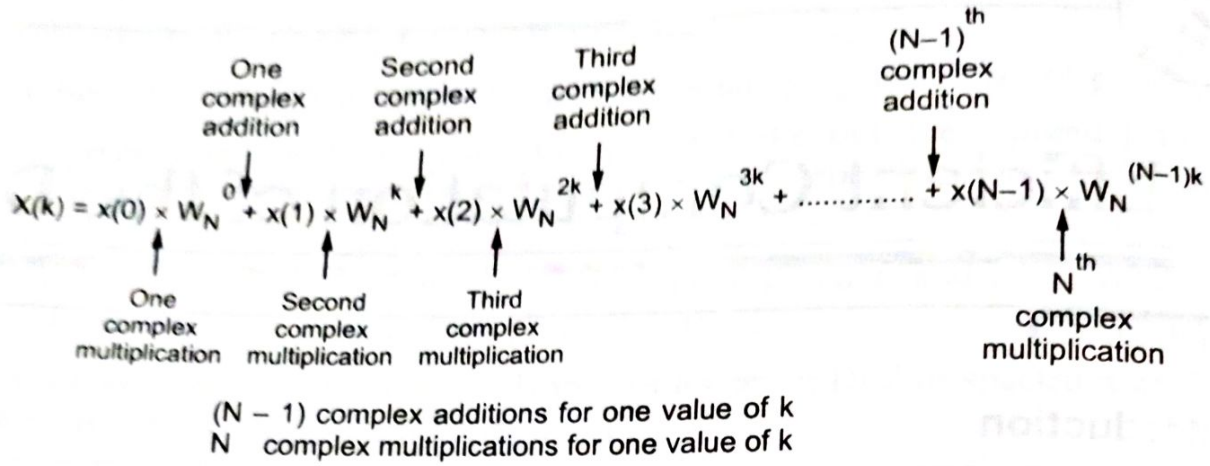

Fig. 2.2.1 Number of complex multiplications and additions in direct computation of DFT

For any value of 'k' in this equation, observe the multiplications and additions involved. 'N' number of complex multiplications and '$(N-1)$' number of complex additions are required to calculate $X(k)$ for one value of k. We know that there are such $k = 0, 1, \ldots N-1$; i.e. 'N' number of values of $X(k)$.

Hence,

$$\left.\begin{array}{l}\text{Number of complex multiplications required for calculating X (k) for } k = 0, 1, \ldots, N-1\end{array}\right\} = N \times N = N^2 \quad \ldots (2.2.1)$$

And,

$$\left.\begin{array}{l}\text{Number of complex additions required for calculating X (k) for } k = 0, 1, \ldots N-1\end{array}\right\} = (N-1) \times N = N^2 - N \quad \ldots (2.2.2)$$

Thus if we want to evaluate the 1024 point DFT of the sequence, then N = 1024,

Complex multiplications = $N^2 = (1024)^2 \approx 1 \times 10^6$

Complex additions = $N^2 - N = (1024)^2 - 1024 \approx 1 \times 10^6$

Here let us assume that the processor executes one complex multiplication in 1 microsecond. Let the one complex addition is also executed in 1 microsecond. Then the time required for computations will be,

$$\begin{aligned}\text{Time} &= (\text{complex multiplications}) \times (\text{time for one multiplication}) \\ &\quad + (\text{complex additions}) \times (\text{time for one addition}) \\ &= \left(1 \times 10^6 \times 1 \times 10^{-6}\right) + \left(1 \times 10^6 \times 1 \times 10^{-6}\right) \\ &= 1 + 1 = 2 \text{ seconds.}\end{aligned}$$

Thus two seconds of the time is required for computations of 1024 point DFT. In terms of processors, this is large time. This is because processors has to do lot of other work such as fetching and storing data in the memory, handling data inputs and outputs, displays etc. Hence real time computation of DFT for large values of 'N' becomes practically impossible by direct computation. We will see further that FFT algorithms are extremely fast and their computation speed increases as 'N' increases. Hence FFT algorithms are used always to compute DFT.

Computational Complexity with Direct Computation

For direct computation :

According to the definition of DFT we have,

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \quad k = 0, 1, 2, ... N-1 \qquad ...(1)$$

Equation (1) indicates that we have to take multiplication of $x(n)$ and twiddle factor. Then we have to add all the terms. Since twiddle factor is complex; we need to perform complex multiplications and complex additions.

Complex multiplications :

As given by Equation (1), for one value of 'k' multiplication should be performed for all values of 'n'. The range of 'n' is from 0 to $N-1$. So for one value of 'k'; N complex multiplications are required. Now the range of k is also from $k = 0$ to $k = N-1$. The total complex multiplications are,

$$\text{Complex multiplications} = N \times N = N^2 \qquad ..(2)$$

Complex additions :

According to Equation (1), for each value of K we need to add the product terms of $x(n) W_N^{kn}$. For example, let us say $N = 4$.

$$\text{For } k = 0 \Rightarrow X(0) = \sum_{n=0}^{3} x(n) W_4^{0 \times n} = \sum_{n=0}^{3} x(n) W_4^0$$

$$\therefore \quad X(0) = x(0) W_4^0 + x(1) W_4^0 + x(2) W_4^0 + x(3) W_4^0 \qquad ...(3)$$

In Equation (3); four complex multiplications are required and three complex additions are required. Here we have considered $N = 4$. Thus for each value of 'k'; N complex multiplications are required and 'N – 1' complex additions are required. Now the total values of k and 'N'.

$$\therefore \text{ Complex Additions} = N(N-1) = N^2 - N$$

1.1 FFT Introduction :

We have studied how to obtain DFT of a sequence by using direct computation. Basically, the direct computation of DFT requires large number of computations. So more processing time is required.

For the computation of N-point DFT, N^2 complex multiplications and $N^2 - N$ complex additions are required. If the value of N is large then the number of computations will go into lakhs. This proves inefficiency of direct DFT computation.

In 1965, Cooley and Tukey developed very efficient algorithm to implement the DFT. This algorithm is called as Fast Fourier Transform (FFT). These FFT algorithms are very efficient in terms of computations. By using these algorithms, number of arithmetic operations involved in the computation of DFT are greatly reduced.

Different FFT algorithms are available : out of which Radix-2 FFT algorithm is most important FFT algorithm.

1.2 Radix-2 FFT Algorithm :

While calculating DFT; we have discussed that we always calculate 'N'-point DFT. The number N can be factored as,

$$N = r_1 , r_2 , r_3 \ldots r_v \quad \ldots(1)$$

Here every 'r' is a prime.

Now if $r_1 = r_2 = r_3 = ... = r_v = r$

then we can write,

$$N = r^v \quad \ldots(2)$$

Here 'r' is called as radix (base) of FFT algorithm and

'v' indicates number of stages in FFT algorithm.

Now radix means base and if its value is '2' then it is called as radix-2 FFT algorithm. Thus when r = 2; Equation (2) becomes,

$$N = 2^v \quad \ldots(3)$$

Thus if we are computing 8 point DFT then N = 8

$$\therefore \quad 8 = 2^v$$

$$\therefore \quad v = 3 \quad \ldots(4)$$

So for 8 point DFT, there are three stages of FFT algorithm. While computing FFT, divide number of input samples by 2, till you reach minimum two samples. Based on this division there are two algorithms as follows :

(1) Radix-2 Decimation in Time (DIT) algorithm.

(2) Radix-2 Decimation in Frequency (DIF) algorithm.

Before studying these algorithms we will derive some important properties of twiddle factor W_N. The twiddle factor W_N is given by,

$$W_N = e^{-\frac{j2\pi}{N}} \qquad ...(5)$$

1. $\mathbf{W_N^k = W_N^{k+N}}$

Using Equation (5) we can write,

$$W_N^{k+N} = \left[e^{-\frac{j2\pi}{N}} \right]^{k+N} = e^{-\frac{j2\pi k}{N}} \cdot e^{-j2\pi}$$

But $e^{-j2\pi} = \cos 2\pi - j \sin 2\pi = 1 - j0 = 1$

$$\therefore \quad W_N^{k+N} = e^{-j\frac{2\pi}{N}k}$$

$$\therefore \quad W_N^{k+N} = \left(e^{-j\frac{2\pi}{N}} \right)^k \qquad ...(6)$$

In Equation (6), the bracket term is W_N

$$\therefore \quad W_N^{k+N} = W_N^k \qquad ...(7)$$

Equation (7) indicates that twiddle factor is periodic.

2. $\mathbf{W_N^{k+\frac{N}{2}} = -W_N^k}$

Using Equation (6) we can write,

$$W_N^{k+\frac{N}{2}} = \left[e^{-\frac{j2\pi}{N}} \right]^{\left(k+\frac{N}{2}\right)} = e^{-\frac{j2\pi k}{N}} \cdot e^{-\frac{j2\pi}{N}\cdot\frac{N}{2}}$$

$$\therefore \quad W_N^{k+\frac{N}{2}} = e^{-\frac{j2\pi k}{N}} \cdot e^{-j\pi} \qquad ...(8)$$

But $e^{-j\pi} = \cos \pi - j \sin \pi = -1 - 0 = -1$

$$\therefore \quad W_N^{k+\frac{N}{2}} = -e^{-j\frac{2\pi k}{N}}$$

$$\therefore \quad W_N^{k+\frac{N}{2}} = -\left(e^{-\frac{j2\pi}{N}} \right)^k \qquad ...(9)$$

In Equation (9), the bracket term is, W_N

$$\therefore \quad W_N^{k+\frac{N}{2}} = -W_N^k \qquad ...(10)$$

Equation (10) indicates that twiddle factor is symmetric.

3. $W_N^2 = W_{N/2}$

From Equation (5) we can write,

$$W_{N/2} = e^{-\frac{j2\pi}{N/2}} = e^{-\frac{j2\pi}{N}\cdot 2}$$

$$\therefore \quad W_{N/2} = \left[e^{-\frac{j2\pi}{N}} \right]^2$$

$$\therefore W_{N/2} = W_N^2$$