Chapter 4: DISCRETE FOURIER TRANSFORM (DFT)

INTRODUCTION:

Modern signal processing using computer is discrete processing moreover digital processing. So, it is necessary to represent either time domain or frequency domain signal by their samples rather than continuous function.

Frequency analysis of discrete time signal x[n] invokes Fourier transform resulting the continuous function of w i.e. $X(e^{jw})$. A difficulty encounter with the direct application of Fourier transform to a discrete time signal is that the resulting representation $X(e^{jw})$ becomes continuous function of frequency. Hence, it is unsuitable for digital processing.

The simple solution is the representation of a discrete time signal x[n] by samples of its spectrum $X(e^{jw})$. Such a frequency domain sampling leads to the Discrete Fourier Transform (DFT), powerful computational tool for performing frequency analysis of discrete time signal. A second transform domain representation that is applicable only to a finite length sequences is the DFT. The DFT is itself a sequence rather than a function of a continuous variable, and it corresponds to samples, equally spaced in frequency, of the Fourier transform of the signal. Not only its importance as a Fourier representation of sequences, the DFT plays a central role in the variety of DSP algorithms. Because there are variety of efficient algorithms for DFT computation.

Fourier analysis of different signals:

S. N.	Signal types	Fourier Analysis	Synthesis Equation	Analysis Equation
1	Continuous time periodic signal	CTFS	$x(t) = \sum_{k=-\infty}^{\infty} C_k e^{jk2\pi F_0 t}$	$C_k = \frac{1}{T}\int_{T_P} x(t)\, e^{-jk2\pi F_0 t} dt$
2	Discrete-time periodic signal	DTFS	$x[n] = \sum_{k=0}^{N-1} C_k e^{j2\pi kn/N}$	$C_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$

3	Continuous time aperiodic signal	CTFT	$x(t) = \int_{-\infty}^{\infty} X(F)\, e^{j2\pi Ft} dF$	$X(F) = \int_{-\infty}^{\infty} x(t)\, e^{-j2\pi Ft} dt$
4	Discrete-time aperiodic signal	DTFT	$x[n] = \frac{1}{2\pi}\int_{0}^{2\pi} X(\omega)\, e^{j\omega n} d\omega$	$X(\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$

Difference between DTFT and DFT

DFT	**DTFT**
1. Obtained by performing sampling in both the time & frequency domains.	1. Sampling is performed only in time domain
2. Discrete frequency spectrum	2. Continuous function of ω
3. The DFT can be applied only to finite length sequences.	3. The DTFT is applicable to any arbitrary sequences.

Frequency Domain Sampling & Reconstruction of Discrete-Time Signal

Let us consider a discrete-time signal (aperiodic) x[n]. It's Fourier transform is given by

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}$$

i

Here X($e^{j\omega}$) is continuous. Suppose we sample X(ω) periodically in frequency at a spacing of δω radians between successive samples. Since X(ω) is periodic with period 2π, only samples in the fundamental frequency range are necessary. We take N equidistance samples in the interval 0 ≤ ω ≤ 2π with spacing δω = 2π/N as shown in figure.

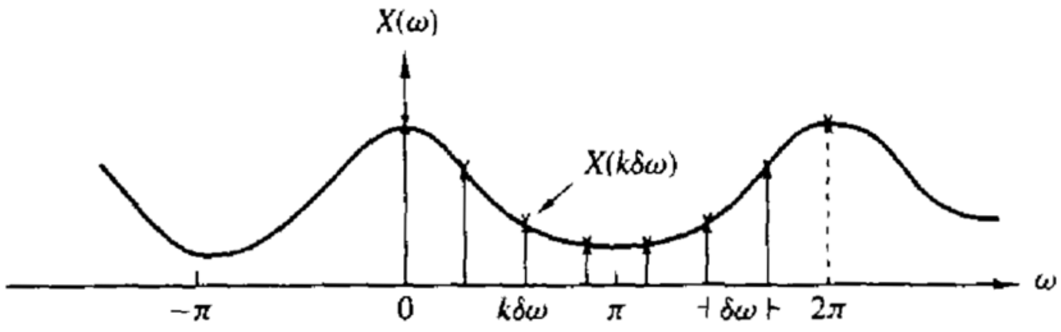


Figure 1: Frequency-domain sampling of the Fourier transform.

If we evaluate Eqn (i), at ω = 2πk/N

$$X\left(\frac{2\pi}{N}k\right) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\frac{2\pi kn}{N}}, k = 0,1,2,..,N-1 \qquad \textbf{\textit{ii}}$$

It can be subdivided into an infinite number of summations, where each sum contains N terms. Thus,

$$X\left(\frac{2\pi}{N}k\right) = \cdots + \sum_{n=-N}^{-1} x[n]e^{-j\frac{2\pi kn}{N}} + \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi kn}{N}} + \sum_{n=N}^{2N-1} x[n]e^{-j\frac{2\pi kn}{N}} + \cdots \qquad \textbf{\textit{iii}}$$

$$= \sum_{l=-\infty}^{\infty} \sum_{n=lN}^{lN+N-1} x[n]e^{-j\frac{2\pi kn}{N}}$$

Changing the index in the inner summation from n to n-lN and by interchanging the order of summation

$$X\left(\frac{2\pi}{N}k\right) = \sum_{n=0}^{N-1}\left[\sum_{l=-\infty}^{\infty} x[n-lN]\right]e^{-j\frac{2\pi kn}{N}}, for\ k = 0,1,2,\ldots,N-1 \qquad \textbf{\textit{iv}}$$

The obtained signal in the bracket of Eqn (iv) is the periodic repeatition of x[n] every N samples. Now it can be expanded in a Fourier series as,

$$x_p[n] = \sum_{k=0}^{N-1} C_k e^{j\frac{2\pi kn}{N}}, \qquad n = 0,1,2,\dots,N-1 \qquad \textbf{\textit{v}}$$

With Fourier coefficients

$$C_k = \frac{1}{N}\sum_{n=0}^{N-1} x_p[n] e^{-j\frac{2\pi kn}{N}}, \qquad k = 0,1,2,\dots,N-1 \qquad \textbf{\textit{vi}}$$

Upon comparing Eqns (iv) & (vi)

$$C_k = \frac{1}{N} X\left(\frac{2\pi}{N}k\right), \qquad k = 0,1,2,\dots,N-1 \qquad \textbf{\textit{vii}}$$

Therefore,

$$x_p[n] = \frac{1}{N}\sum_{k=0}^{N-1} X\left(\frac{2\pi}{N}k\right) e^{j\frac{2\pi kn}{N}}, \qquad n = 0,1,2,\dots,N-1 \qquad \textbf{\textit{viii}}$$

Which provides the reconstruction of the periodic signal $x_p[n]$ from the samples of the spectrum $X(\omega)$.

For $0 \le n \le L-1$ & $N \ge L$, $x[n] = x_p[n] \qquad 0 \le n \le N-1$

If $N < L$, it is not possible to recover $x[n]$ from its periodic extension due to time-domain aliasing. Thus we conclude that the spectrum of an aperiodic discrete-time signal with finite duration L can be exactly recovered from its samples at frequencies $\omega_k = 2\pi k/N$.

The Discrete Fourier Transform (DFT)

A finite duration sequence x[n] of length L has a Fourier transform

$$X(\omega) = \sum_{n=0}^{L-1} x[n] e^{-j\omega n} \quad 0 \le \omega \le 2\pi$$

Where the upper and lower indices in the summation reflect the fact that $x[n] = 0$ outside the range $0 \le n \le L-1$. When we sample $X(\omega)$ at equally spaced frequencies $\omega_k = 2\pi k/N$, $k = 0, 1, 2, \dots, N-1$, where $N \ge L$ the resultant samples are,

$$X(k) = X\left(\frac{2\pi}{N}k\right) = \sum_{n=0}^{L-1} x[n]e^{-j\frac{2\pi kn}{N}} = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi kn}{N}}, \qquad k = 0,1,2,\dots,N-1$$

Since, x[n] = 0 for n ≥ L. The relation in above equation is called the discrete Fourier transform of x[n] as it is obtained by sampling the Fourier transform X(ω) at a set of N (equally spaced) discrete frequencies.

The inverse DFT, which allows us to recover the sequence x[n] from the frequency samples is given by

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j\frac{2\pi kn}{N}}, \qquad n = 0,1,2,\dots,N-1$$

Clearly, when x[n] has length L < N, the N-point IDFT yields x[n] = 0 for L ≤ n ≤ N-1.

$$DFT: X(k) = \sum_{n=0}^{N-1} x[n]\, W_N^{kn}, \qquad k = 0,1,2,\dots,N-1$$

$$IDFT: x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-kn}, \qquad n = 0,1,2,\dots,N-1$$

Where by definition,

$$W_N = e^{-j\frac{2\pi}{N}}$$

Which is an Nth root of unity.

Properties of DFT

The notation to denote the N-point DFT pair x[n] and X(k) is

$$x[n] \underset{N}{\overset{DFT}{\longleftrightarrow}} X(k)$$

The properties of the DFT are useful in the practical techniques for processing signals. The various properties are given below:

1. **Periodicity:**

   If X(k) is an N-point DFT of x[n], then

   x[n+N] = x[n] for all n

   X(k+N) = X(k) for all k

These periodicities in x[n] and X(k) follow immediately from above formulae for the DFT and IDFT respectively.

We had not viewed the DFT X(k) as a periodic sequence. In some applications it is advantageous to do this.

2. Linearity:

If $X_1(k)$ and $X_2(k)$ are the N-point DFTs of $x_1[n]$ and $x_2[n]$ and a & b are arbitrary constants (real or complex valued), then,

$$ax_1[n] + bx_2[n] \underset{N}{\overset{DFT}{\leftrightarrow}} aX_1(k) + bX_2(k)$$

3. Shifting property:

Let $x_p[n]$ is a periodic sequence with period N which is obtained by extending x[n] periodically, ie

$$x_p[n] = \sum_{l=-\infty}^{\infty} x[n - lN]$$

Now, shift the sequence $x_p[n]$ by k units to the right which is given by,

$$x'_p[n] = x_p[n-k] = \sum_{l=-\infty}^{\infty} x[n - k - lN]$$

The finite duration sequence,

$$x'[n] = \begin{cases} x'_p[n], & 0 \le n \le N-1 \\ 0, & otherwise \end{cases}$$

Can be obtained from x[n] by a circular shift.

The circular shift of a sequence can be represented as the index modulo N,

$$x'[n] = x(n-k, modulo\ N) = x((n-k))_N$$

# If k = 2 and N = 4, we have

$x'[n] = x((n-2))_4$ which implies

$$x'(0) = x((-2))_4 = x(2)$$

$$x'(1) = x((1-2))_4 = x(3)$$

$$x'(2) = x((2-2))_4 = x(0)$$

$$x'(3) = x((3-2))_4 = x(1)$$

An N-point sequence is called circularly even if it is symmetric about the point zero on the circle, taking counterclockwise as positive direction.

This implies that,

$x[N - n] = x[n] \qquad 1 \le n \le N-1$

An N-point sequence is called circularly odd if it is antisymmetric about the point zero on the circle. This implies that

$x[N - n] = - x[n] \qquad 1 \le n \le N-1$

4. **Time reversal of a sequence**

The time reversal of an N-point sequence is attained by reversing its samples about the point zero on the circle. Thus the sequence $x((-n))_N$ is simply given as

$$x((-n))_N = x(N - n), 0 \le n \le N - 1$$

For periodic sequence $x_p[n]$,

$$\text{Even: } x_p[n] = x_p[-n] = x_p[N - n]$$

$$\text{Odd: } x_p[n] = - x_p[-n] = - x_p[N - n]$$

If $x_p[n]$ is complex valued

$$\text{Conjugate even: } x_p[n] = x^*_p[N - n]$$

$$\text{Conjugate odd: } x_p[n] = - x^*_p[N - n]$$

Thus, $x_p[n] = x_{pe}[n] + x_{po}[n]$

Where, $x_{pe}[n] = ½ (x_p[n] + x^*_p[N - n])$ and $x_{po}[n] = ½ (x_p[n] - x^*_p[N - n])$

5. **Symmetry Properties of the DFTs**

For complex valued x[n] (& its DFT X(k))

$x[n] = x_R[n] + jx_I[n] \qquad 0 \le n \le N - 1$

$X(k) = X_R(k) + jX_I(k) \qquad 0 \le k \le N - 1$

Now,

$$X(k) = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi kn}{N}}, \qquad k = 0,1,2,\dots,N-1$$

$$X_R(k) + jX_I(k) = \sum_{n=0}^{N-1} [x_R[n] + jx_I[n]] \left[cos\frac{2\pi kn}{N} - jsin\frac{2\pi kn}{N}\right]$$

$$= \sum_{n=0}^{N-1} \left[x_R[n]cos\frac{2\pi kn}{N} + x_I[n]sin\frac{2\pi kn}{N}\right] - j\sum_{n=0}^{N-1} \left[x_R[n]sin\frac{2\pi kn}{N} - x_I[n]cos\frac{2\pi kn}{N}\right]$$

Similarly of IDFT,

$$x_R[n] + jx_I[n] = \frac{1}{N}\sum_{k=0}^{N-1}\left[X_R(k)cos\frac{2\pi kn}{N} - X_I(k)sin\frac{2\pi kn}{N}\right] - j\frac{1}{N}\sum_{k=0}^{N-1}\left[X_R(k)sin\frac{2\pi kn}{N} + X_I(k)cos\frac{2\pi kn}{N}\right]$$

If x[n] is

i. Real-valued Sequences:

   X(N – k) = X*(k) = X(-k) {from definition of DFT}

ii. Real and even sequences:

   x[n] = x_R[n] and x[n] = x[N – n], 0 ≤ n ≤ N-1 then X_I(k) = 0. Hence DFT reduces to

$$X(k) = \sum_{n=0}^{N-1} x[n]cos\frac{2\pi kn}{N}, \qquad 0 \le k \le N-1$$

   Which is itself real valued and even. The IDFT reduces to

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(k)cos\frac{2\pi kn}{N}, \qquad 0 \le n \le N-1$$

iii. Real and odd sequences:

   x[n] = x_R[n] and x[n] = -x[N – n], 0 ≤ n ≤ N-1 then X_R(k) = 0. Hence DFT reduces to

$$X(k) = -j\sum_{n=0}^{N-1} x[n]sin\frac{2\pi kn}{N}, \qquad 0 \le k \le N-1$$

   Which is purely imaginary and odd. The IDFT reduces to

$$x[n] = j\frac{1}{N}\sum_{k=0}^{N-1} X(k)sin\frac{2\pi kn}{N}, \qquad 0 \le n \le N-1$$

iv. Purely imaginary sequences:

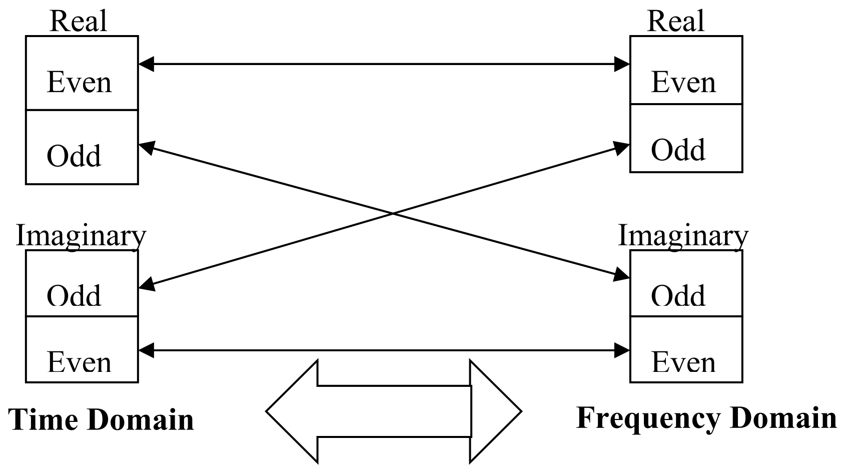
   x[n] = jx_I[n], then

$$X_R(k) = \sum_{n=0}^{N-1} x_I[n]sin\frac{2\pi kn}{N} \;\; \& \;\; X_I(k) = \sum_{n=0}^{N-1} x_I[n]cos\frac{2\pi kn}{N}$$

   We observe that X_R(k) is odd and X_I(k) is even.

   If x_I[n] is odd, then X_I(k) = 0 and hence X(k) is purely real. On the other hand if, x_I[n] is even, then X_R(k) = 0 and hence X(k) is purely imaginary.

The symmetry properties given above may be summarized in block diagram as follows:



6. Multiplication of two DFTs and circular convolution:

If $x_1[n] \overset{DFT}{\underset{N}{\leftrightarrow}} X_1(k)$ & $x_2[n] \overset{DFT}{\underset{N}{\leftrightarrow}} X_2(k)$

Let us determine the relationship between $x_3[n]$ and the sequences $x_1[n]$ and $x_2[n]$ if we multiply the two DFTs together to obtain $X_3(k)$.

We have, $X_3(k) = X_1(k)X_2(k)$, $k = 0, 1, 2, \ldots, N-1$

The IDFT of $X_3(k)$ is

$$x_3[m] = \frac{1}{N}\sum_{k=0}^{N-1} X_3(k)W_N^{-kn} = \frac{1}{N}\sum_{k=0}^{N-1} X_1(k)X_2(k)W_N^{-kn}, \;\; m = 0,1,2,\ldots,N-1$$

$$= \frac{1}{N}\sum_{k=0}^{N-1}\left[\sum_{n=0}^{N-1} x_1[n]W_N^{kn}\right]\left[\sum_{l=0}^{N-1} x_2[l]W_N^{kl}\right]W_N^{-kn}$$

Rearranging the order of summation,

$$x_3[m] = \frac{1}{N}\sum_{n=0}^{N-1} x_1[n]\sum_{l=0}^{N-1} x_2[l]\left[\sum_{k=0}^{N-1} W_N^{k(n+l-m)}\right] ---(A)$$

{We have to obtain the form of $x_3[n] = x_1[n]$ operator $x_2[n]$}

The inner sum in bracket has the form,

$$\sum_{k=0}^{N-1} a^k = \begin{cases} N, \; if\ a = 1 \\ \dfrac{1-a^N}{1-a}, \; if\ a \neq 1 \end{cases} ---(B)$$

Where a is defined as,

$$a = W_N^{(n+l-m)} = e^{-j\frac{2\pi}{N}(n+l-m)}$$

As $e^{-j2\pi p} = 1$, where p is an integer (positive or negative)

a = 1, if p = (n + l – m)/N

or, n + l – m = pN, i.e. n + l – m is a multiple integer of N.

On the other hand, $a^N = e^{-j2\pi(n+l-m)} = 1$ for any value of a ≠ 0 (as n, l, m are integers)

Consequently (B) reduces to

$$\sum_{k=0}^{N-1} a^k = \begin{cases} N, & if\ l = m - n + pN = ((m-n))_N, p\ is\ integer \\ 0, & otherwise \end{cases} \quad ---(C)$$

If we substitute (C) into (A),

$$x_3[m] = \frac{1}{N}\sum_{n=0}^{N-1} x_1[n]\left[x_2\big((m-n)\big)_N . N\right]$$

$$or, x_3[m] = \sum_{n=0}^{N-1} x_1[n]\, x_2\big((m-n)\big)_N, \quad m = 0,1,2,---,N-1$$

The expression above has the form of a convolution sum. However, it is not the ordinary linear convolution that we discussed/encountered earlier. Instead, the above convolution sum involves the index $((m\text{-}n))_N$, circular shift instead of linear shift, and is called **circular convolution.**

Thus we conclude that multiplication of the DFTs of two sequences is equivalent to the circular convolution of the two sequences in the time domain.

The circular convolution is denoted by $x_3[n] = x_1[n]$ (N) $x_2[n]$

# Find circular convolution of following signals:

$x_1[n] = \{2, 1, 2, 1\}$ & $x_2[n] = \{1, 2, 3, 4\}$ (Ans: $x_3[n] = \{14, 16, 14, 16\}$)

Calculation of circular convolution:

There are several methods to calculate circular convolution of two sequences. Three of them are discussed below:

a. Graphical Method:

In this method the sequences are plot in graph in the form of circles. The procedure or steps are given below:

i. Graph each sequences as points on a circle, taking counterclockwise (CCW) as positive direction.

ii. **Folding:** Fold any one sequence on a circle. The folded sequence is simply graphed in a clockwise direction. Eg: fold $x_2[n]$ to obtain $x_2((-n))_N$.

iii. **Shifting:** shift the folded sequence i.e. $x_2((-n))_N$ by l to obtain $x_2((l - n))_N$, take l = 0, 1, 2, 3, … , N-1. $x_2((l - n))_N$ is simply the sequence $x_2((-n))_N$ rotated CCW by l units in time.

iv. **Multiplication:** Multiply $x_1[n]$ and $x_2((l - n))_N$ to yield the product sequence.

v. **Summation:** Sum the values in the product sequence to obtain $x_3[m]$ at m = l.

vi. **Repetition:** Repeat step (iii) to (v) to obtain $x_3[m]$ for all values of n.

The basic difference between linear and circular method is that, in circular convolution, the folding and shifting operation are performed in a circular fashion by computing the index of one of the sequences modulo N. In linear convolution, there is not modulo N operation.

b. DFT and IDFT method:

In this method, the DFTs of the sequences are calculated, the two DFTs are multiplied together and finally the IDFT of multiplied sequences is calculated to obtain the circular convolution.

c. Matrix Multiplication method:

In this method, the circular convolution of two sequences $x_1[n]$ and $x_2[n]$ can be obtained by representing the sequences in the matrix form as shown below:

$$\begin{bmatrix} x_2(0) & x_2(N-1) & x_2(N-2) & & x_2(2) & x_2(1) \\ x_2(1) & x_2(0) & x_2(N-1) & \cdots & x_2(3) & x_2(2) \\ x_2(2) & x_2(1) & x_2(0) & & x_2(4) & x_2(3) \\ & \vdots & & \ddots & \vdots & \\ x_2(N-2) & x_2(N-3) & x_2(N-4) & \cdots & x_2(0) & x_2(N-1) \\ x_2(N-1) & x_2(N-2) & x_2(N-3) & & x_2(1) & x_2(0) \end{bmatrix} \begin{bmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ \vdots \\ x_1(N-2) \\ x_1(N-1) \end{bmatrix} = \overrightarrow{x_3}[n]$$

The sequence $x_2[n]$ is repeated via circular shift of samples and represented in N×N matrix form. The sequence $x_1[n]$ is represented as column matrix. The multiplication of these two matrices gives the sequence $x_3[n]$.

7. Time reversal of a sequence:

If $x[n] \underset{N}{\overset{DFT}{\leftrightarrow}} X(k)$, then

$$x((-n))_N = x[N-n] \underset{N}{\overset{DFT}{\leftrightarrow}} X((-k))_N = X(N-k)$$

8. Circular time shift of a sequence:

If $x[n] \underset{N}{\overset{DFT}{\leftrightarrow}} X(k)$, then

$$x\big((n-l)\big)_N \underset{N}{\overset{DFT}{\leftrightarrow}} X(k)e^{-j\frac{2\pi kl}{N}}$$

9. Circular frequency shift:

If $x[n] \underset{N}{\overset{DFT}{\leftrightarrow}} X(k)$, then

$$x[n]e^{j\frac{2\pi ln}{N}} \underset{N}{\overset{DFT}{\leftrightarrow}} X((k-l))_N$$

This is the dual to the circular time shifting property.

10. Complex-conjugate properties:

If $x[n] \underset{N}{\overset{DFT}{\leftrightarrow}} X(k)$, then

$$x^*[n] \underset{N}{\overset{DFT}{\leftrightarrow}} X^*\big((-k)\big)_N = X^*(N-k)$$

And the IDFT of $X^*(k)$ is

$$\frac{1}{N}\sum_{k=0}^{N-1} X^*(k)e^{j\frac{2\pi kn}{N}} = \left[\frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j\frac{2\pi k(N-n)}{N}}\right]^*$$

Therefore,

$$x^*((-n))_N = x^*(N-n) \underset{N}{\overset{DFT}{\leftrightarrow}} X^*(k)$$

11. Circular Correlation:

In general, for complex valued sequences x[n] and y[n], if

$$x[n] \underset{N}{\overset{DFT}{\leftrightarrow}} X(k) \;\&\; y[n] \underset{N}{\overset{DFT}{\leftrightarrow}} Y(k)$$

Then,

$$\tilde{\gamma}_{xy}(l) \underset{N}{\overset{DFT}{\leftrightarrow}} \tilde{R}_{xy} = X(k)Y^*(k)$$

Where$\tilde{\gamma}_{xy}$, is the (un-normalized) circular cross-correlation sequence defined as,

$$\tilde{\gamma}_{xy}(l) = \sum_{n=0}^{N-1} x[n]y^*((n-l))_N$$

Proof:

$$\tilde{\gamma}_{xy} = x(l)(N)y^*(-l)$$

$$Then, \qquad \tilde{R}_{xy} = X(k)Y^*(k)$$

In the special case when y[n] = x[n], we have the corresponding expression for the circular auto-correlation of x[n] is

$$\tilde{\gamma}_{xx}(l) \underset{N}{\overset{DFT}{\leftrightarrow}} \tilde{R}_{xx} = |X(k)|^2$$

12. Multiplication of two sequences,

$$x_1[n]x_2[n] \underset{N}{\overset{DFT}{\leftrightarrow}} \frac{1}{N} X_1(k)(N)X_2(k)$$

13. Parseval's Theorem:

For complex valued sequences x[n] & y[n], in general, if

$$x[n] \underset{N}{\overset{DFT}{\leftrightarrow}} X(k) \;\&\; y[n] \underset{N}{\overset{DFT}{\leftrightarrow}} Y(k)$$

$$then, \sum_{n=0}^{N-1} x[n]y^*[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(k)Y^*(k)$$

Proof:

From Circular Correlation

$$\tilde{\gamma}_{xy}(l) = \sum_{n=0}^{N-1} x[n]y^*\big((n-l)\big)_N$$

$$for\ l = 0, \qquad \tilde{\gamma}_{xy}(0) = \sum_{n=0}^{N-1} x[n]y^*\big((n)\big)_N = \sum_{n=0}^{N-1} x[n]y^*[n]$$

$$and, \qquad \tilde{\gamma}_{xy}(l) = \frac{1}{N}\sum_{k=0}^{N-1} \tilde{R}_{xy}(k)e^{j\frac{2\pi kl}{N}}$$

$$or, \qquad \tilde{\gamma}_{xy}(l) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)Y^*(k)e^{j\frac{2\pi kl}{N}}$$

$$for\ l = 0, \qquad \tilde{\gamma}_{xy}(0) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)Y^*(k)$$

When y[n] = x[n],

$$then, \qquad \sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N}\sum_{k=0}^{N-1} |X(k)|^2$$

This expression relates the energy in the finite duration sequence x[n] to the power in the frequency components X(k).

Applications of DFT

The DFT has seen wide usage across a large number of fields:

- Spectral analysis
- Data compression
- Partial differential equations
- Multiplication of large integers
- Outline of DFT polynomial multiplication algorithm.

Fast Fourier Transform (FFT)

Efficient computation of the DFT Algorithms

The FFT is simply an algorithm to speed up the DFT calculation by reducing the number of multiplications and additions required.

Applications of FFT algorithms

Though discrete Fourier transforms can convert time domain data into frequency domain accurately, it has an unavoidable drawback. It is the inefficiency of the algorithm which makes the processing much time consuming. As a solution to this problem, FFT is invented. FFT is much faster & efficient than DFT.

DFT plays an important role in discrete-time signal analysis and processing where it can perform lots of operations like spectral analysis, correlation analysis, linear convolution of two sequences which is a key digital filtering operation and lots more.

The FFT is simply a fast (computationally efficient) way to calculate DFT. The idea behind the FFT algorithms is the "Divide & Conquer" approach, to split the DFT into a series of lower DFT and exploiting the symmetry & periodicity properties of complex exponential. Less computation is required to evaluate and combine the lower order DFTs than to evaluate original N-point DFT.

Computational complexity of the DFT

There are many ways to measure the complexity and efficient of an implementation or algorithms, and a final assessment depends on both the available technology and the intended application. We will use the number of arithmetic multiplications and additions as a measure of computational complexity. This measure is simple to apply, and the number of multiplications and additions is directly related to the computational speed when algorithms are implemented on general purpose digital computers or special purpose microprocessors.

The N-point DFT of sequence x[n] is given by

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi kn}{N}}, \qquad k = 0,1,2,---,N-1$$

The direct computation of n-point DFT requires N^2 complex multiplication and N(N-1) additions.

For a complex-valued x[n]

$$X_R(k) = \sum_{n=0}^{N-1} [x_R[n] cos\frac{2\pi kn}{N} + x_I[n] sin\frac{2\pi kn}{N}]$$

$$X_I(k) = -\sum_{n=0}^{N-1} [x_R[n] sin\frac{2\pi kn}{N} - x_I[n] cos\frac{2\pi kn}{N}]$$

The direct computation of above DFT requires,

1. 2N^2 evaluations of trigonometric functions.

2. $4N^2$ real multiplications.
3. $4N(N-1)$ real additions.
4. A number of indexing & addressing operations.

Divide & Conquer Approach:

This approach is based on the decomposition of an N-point DFT into successively smaller DFTs.

Let us consider the computation of an N-point DFT, where N can be factored as a product of two integers

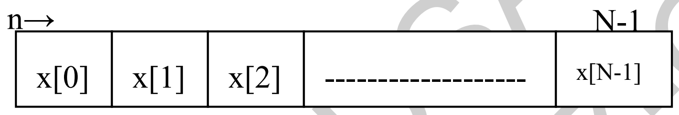
i.e. $N = LM$

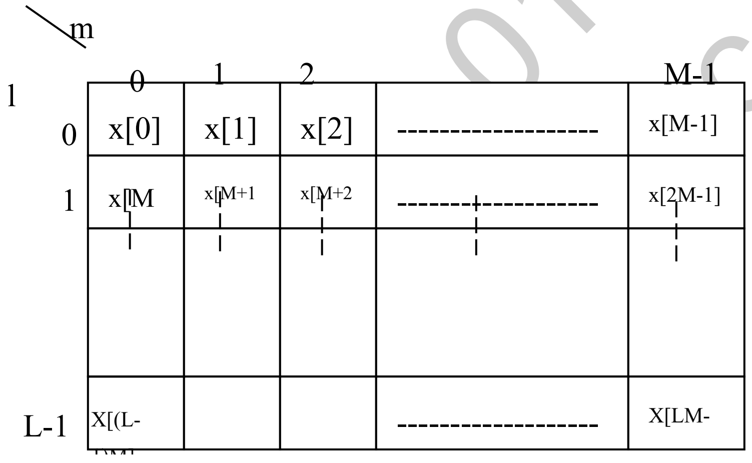
if N is prime number, then padding with zeros is done. Now the sequence $x[n]$, $0 \le n \le N-1$, can be stored in either a one-dimensional array indexed by n or as a two-dimensional array indexed by l & m, where $0 \le l \le L-1$ and $0 \le m \le M-1$. Note that l is the row index and m is the column index. Thus sequence $x[n]$ can be mapped in a rectangular array which depends on indexes (l, m).

Suppose we select the mapping

$$n = Ml + m$$

This leads to an arrangement in which the first row consists of the first M elements of $x[n]$, the second row consists of the next M elements of $x[n]$ and so on as shown in figure below:





Row-wise mapping $n = Ml + m$

On the other hand, the mapping

$$n = l + mL$$

stores the first L elements of $x[n]$ in the first column, the next L elements in the second column, and so on.

A similar arrangement can be used to store the computed DFT values

K = PQ

k = Mp + q → row-wise mapping

k = qL + p → column-wise mapping

Now suppose that x[n] is mapped into the rectangular array x[l, m] and X(k) is mapped into a corresponding rectangular array X(p, q). Then the DFT can be expressed as a double sum over the elements of the rectangular array multiplied by the corresponding phase factors.

To be specific, Let us adopt a column-wise mapping for x[n] and the row-wise mapping for the DFT, then

$$X(p,q) = \sum_{m=0}^{M-1}\sum_{l=0}^{L-1} x(l,m) W_N^{(Mp+q)(mL+l)}$$

But $W_N^{(Mp+q)(mL+l)} = W_N^{MLmp} W_N^{Mpl} W_N^{Lmq} W_N^{ql}$

However,

$W_N^{Nmp} = 1$, $W_N^{mqL} = W_{N/L}^{mq} = W_M^{mq}$

and $W_N^{mpL} = W_{N/M}^{pl} = W_L^{pl}$

$$\therefore X(p,q) = \sum_{l=0}^{L-1} \{W_N^{lq} [\sum_{m=0}^{M-1} x(l,m) W_M^{mq}]\} W_L^{lp}$$

Above expression involves the computation of DFTs of length M and length L

1. First, we compute the M-point DFTs

$$F(l,q) = \sum_{m=0}^{M-1} x(l,m) W_M^{mq}, \quad 0 \le q \le M-1$$

   For each rows l = 0, 1, 2, ----, L-1
2. Second, we compute a new rectangular array G(l, q)

$$G(l,q) = W_N^{lq} F(l,q), \quad 0 \le l \le L-1, \quad 0 \le q \le M-1$$

3. Finally, we compute the L-point DFTs.

$$X(p,q) = \sum_{l=0}^{L-1} G(l,q) W_L^{lp}, \quad for\ each\ column, 0 \le q \le M-1\ of\ the\ array\ G(l,q)$$

Computational Improvement of Divide and Conquer Approach:

Computational Complexity	Divide and Conquer	Direct DFT
Complex Multiplication	N(M + L + 1)	N^2

Complex Additions	N(M + L – 2)	N(N-1)
Eg: N = 10,000 we select L = 10 and M = 1000	N(M + L + 1) = 1,01,10,000 N(M + L – 2) = 1,00,80,000	N^2 = 10,00,00,000 N(N-1) = 9,99,90,000

When N is highly composite number,

N = $r_1r_2r_3$---r_v (products of prime numbers)

Radix-2 FFT:

If N is factored as N = $r_1r_2r_3$---r_L, where $r_1 = r_2 = r_3$ = ---- = r_L = r then N = r^L. Therefore, the DFT will be of size 'r' where the number 'r' is called the radix of the FFT algorithm.

If r = 2, then it is known as radix-2 FFT which is most widely used algorithm.

Decimation-in-Time Algorithm (DTA)

In this case, we assume that x[n] represents a sequence of N values, where N is an integer power of 2 i.e. N = 2^L. The given sequence is decimated (broken) into two N/2 point sequences which consists of the even and odd numbered values of x[n].

The N-point DFT of sequence x[n] is expressed as

$$X(k) = \sum_{n=0}^{N-1} x[n]W_N^{nk}, \qquad 0 \le k \le N-1$$

Splitting x[n] into its even & odd numbered values, we get

$$X(k) = \sum_{n=0,n-even}^{N-1} x[n]W_N^{nk} + \sum_{n=0,n-odd}^{N-1} x[n]W_N^{nk}, \qquad 0 \le k \le N-1$$

Now, putting n = 2r for even n and n = 2r+1 for odd n, we get

$$X(k) = \sum_{r=0}^{N/2-1} x[2r]W_N^{2rk} + \sum_{r=0}^{N/2-1} x[2r+1]W_N^{(2r+1)k}, \qquad 0 \le k \le N-1$$

$$Since\ W_N^2 = e^{-j\frac{2\pi}{N}\times 2} = e^{-j\frac{2\pi}{(N/2)}} = W_{N/2}$$

$$X(k) = \sum_{r=0}^{N/2-1} x[2r]W_{N/2}^{rk} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1]W_{N/2}^{rk}, \qquad 0 \le k \le N-1$$

$$or, X(k) = G(k) + W_N^k H(k), \qquad k = 0,1,2,---,\frac{N}{2}-1$$

Here, G(k) and H(k) are the N/2-points DFTs of the even and odd number sequences repectively. Here, we have computed each of the sums for $0 \le k \le N/2-1$ because G(k) and H(k) are considered periodic with period N/2. For k, greater than N/2 for X(k), using periodic property,

$X(k) = G(k) + W_N^k H(k), \qquad 0 \le k \le N/2-1$

& $X(k + N/2) = G(k + N/2) + W_N^{(k+\frac{N}{2})} H(k + N/2)$

Making the use of the symmetry property of W_N, G(k) & H(k)

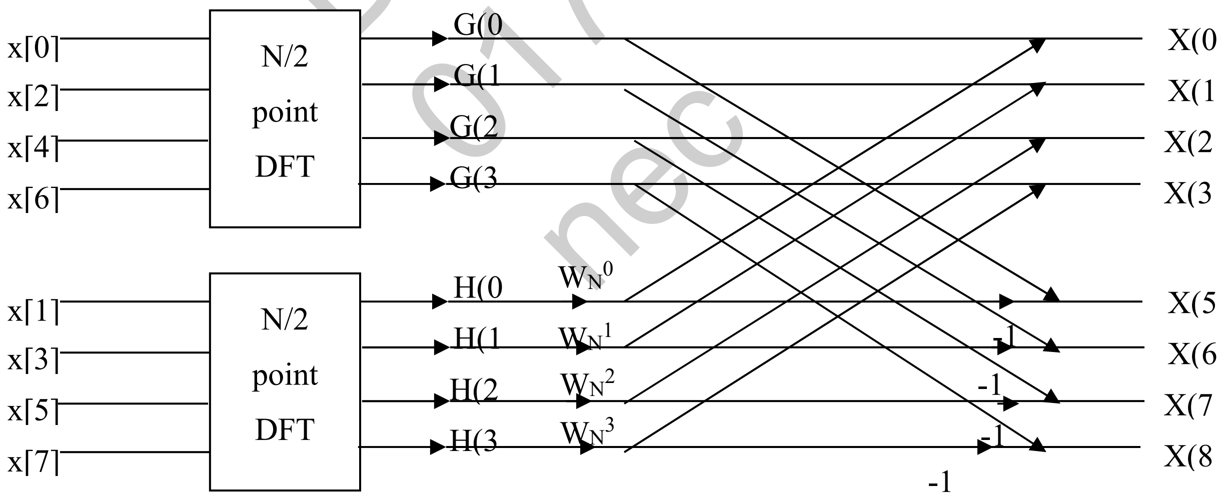
i.e. $W_N^{k+\frac{N}{2}} = -W_N^k, G\left(k+\frac{N}{2}\right) = G(k) \& H\left(k+\frac{N}{2}\right) = H(k)$

we obtain,

$X(k) = G(k) + W_N^k H(k), \qquad 0 \le k \le N/2-1$

& $X(k + N/2) = G(k) - W_N^k H(k)$

The flow graph of the decimation-in-time decomposition of an 8-Point (N=8) DFT computation into two 4-points DFT computations is shown in figure (a).



Fig(a): Illustration of flow graph of the first stage Decimation in time FFT algorithm for N=8

$$Again, \qquad G(k) = \sum_{r=0}^{N/2-1} g[r] W_{N/2}^{rk}, \qquad 0 \le k \le N/2-1$$

Or, $G(k) = \sum_{l=0}^{N/4-1} g[2l]W_{\frac{N}{2}}^{2lk} + \sum_{l=0}^{N/4-1} g[2l+1]W_{\frac{N}{2}}^{(2l+1)k},\ 0 \le k \le \frac{N}{2} - 1$

$$or, G(k) = \sum_{l=0}^{N/4-1} a[l]W_{\frac{N}{4}}^{lk} + W_{N/2}^{k} \sum_{l=0}^{N/4-1} b[l]W_{N/4}^{lk},$$

Or, G(k) = A(k) +W_N^{2k}B(k), $0 \le k \le N/4-1$

Here, A(k) and B(k) are N/4-point DFTs, while G(k) is N/2-point DFT.

For k, greater than N/4 the relation for G(k) is

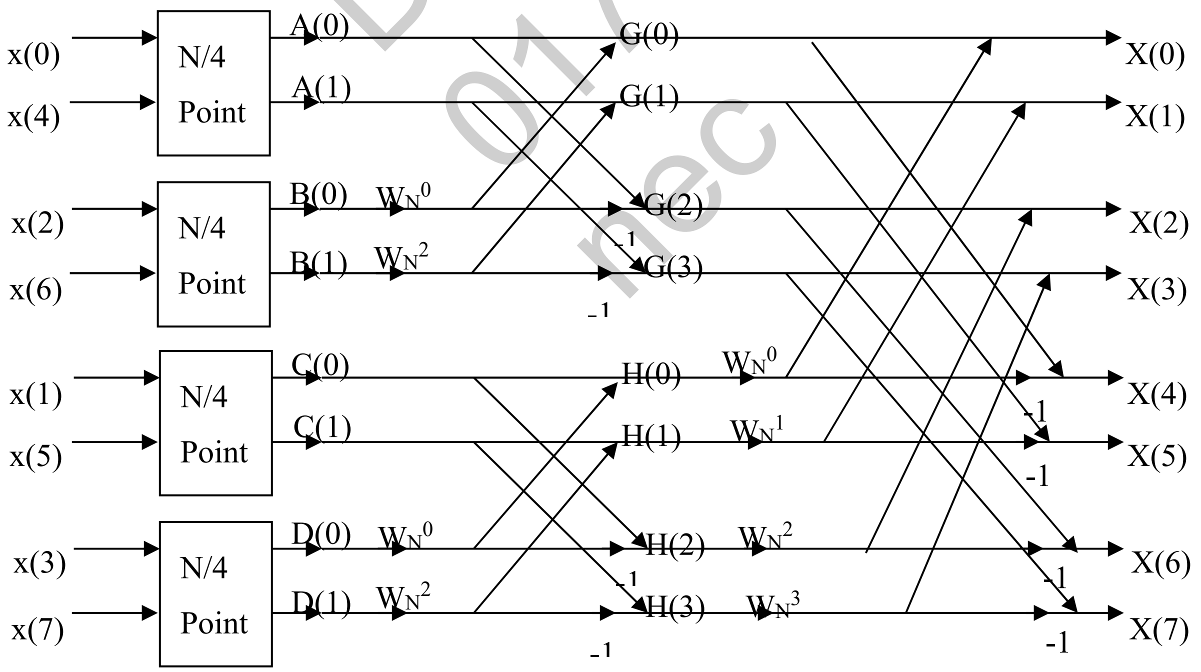
G(k + N/2) = A(k + N/4) +$W_N^{2(k+\frac{N}{4})}B$(k + N/4), $0 \le k \le N/4 - 1$

= A(k) – W_N^{2k}B(k)

In a similar manner,

H(k) = C(k) + W_N^{2k}D(k), $0 \le k \le N/4 - 1$

H(k + N/4) = C(k) - W_N^{2k}D(k),



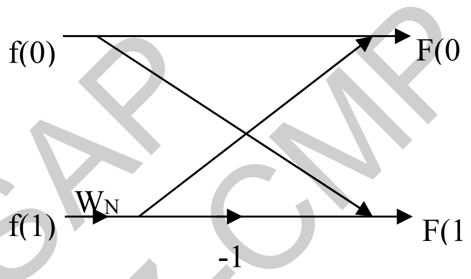
Fig(b): Illustration of flow graph of second stage decimation in time FFT algorithm for N=8

This process of reducing an L-point DFT (L is a power of 2) to an L/2 points DFTs, may be continued until we are left with 2-point DFTs, or there are L (= $\log_2 N$) stages to be evaluated.

A 2-points DFTs, F(k), k = 0, 1 can be evaluated as under

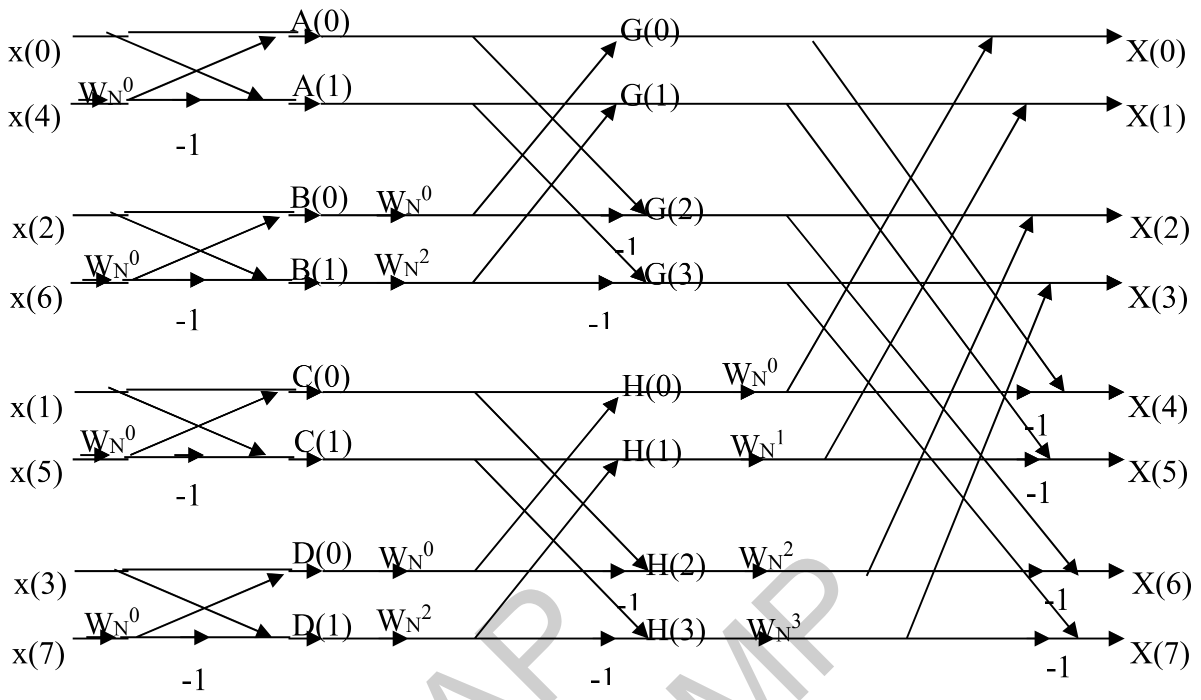
$$\begin{aligned} F(k) &= \sum_{n=0}^{N-1} f[n] W_N^{nk}, \qquad k = 0,1 \\ &= f(0)W_N^0 + f(1)W_N^k \\ \therefore F(0) &= f(0) + W_N^0 f(1) \\ \& F(1) &= f(0) + W_N^1 f(1) \\ &= f(0) - W_N^0 f(1) \end{aligned}$$

Therefore in flow graph, the 2-point DFT can be calculated as



Fig(c): FFT Butterfly for 2-point DFT.

The complete flow graph of the decimation in time FFT algorithms for N=8, which is having of three stages is shown in figure(d). The first stage compute the four 2-points DFTs, the second stage computes the two 4-points DFTs, whereas the third stage computes the desired 8-point DFT.



Fig(d): Final flow graph of a Decimation in time FFT algorithm for N=8.

In order to understand the basic concepts of the FFT and its derivation, note that the DFT expansion can be greatly simplified by taking advantage of the symmetry and periodicity of the twiddle factors (W_N^k). If the equations are rearranged and factored, the result is the Fast Fourier Transform (FFT) which requires only (N/2) $\log_2(N)$ complex multiplications. The computational efficiency of the FFT versus the DFT becomes highly significant when the FFT point size increases to several thousand. However, notice that the FFT computes *all* the output frequency components (either all or none!). If only a few spectral points need to be calculated, the DFT may actually be more efficient. Calculation of a single spectral output using the DFT requires only N complex multiplications.

The radix-2 FFT algorithm breaks the entire DFT calculation down into a number of 2-point DFTs. Each 2-point DFT consists of a multiply-and-accumulate operation called a *butterfly*, as shown in Figure (c).

The 8-point decimation-in-time (DIT) FFT algorithm computes the final output in three stages as shown in Figure (D). The eight input time samples are first divided (or *decimated*) into four groups of 2-point DFTs. The four 2-point DFTs are then combined into two 4-point DFTs. The two 4-point DFTs are then combined to produce the final output X(k). Note that the basic two-point DFT butterfly operation forms the basis for all computation. The computation is done in three stages. After the first stage computation is complete, there is no need to store any previous results. The first stage outputs can be stored in the same registers which originally held the time samples x(n). Similarly, when the second stage computation is completed, the results of the first stage computation can be deleted. In this way, *in-place* computation proceeds to the final stage.

Note that in order for the algorithm to work properly, the order of the input time samples, x(n), must be properly re-ordered using a *bit reversal* algorithm.

The decimal index, n, is converted to its binary equivalent. The binary bits are then placed in reverse order, and converted back to a decimal number. Bit reversing is often performed in DSP hardware in the data address generator (DAG), thereby simplifying the software, reducing overhead, and speeding up the computations.

The computation of the FFT using *decimation-in-frequency* (DIF) is shown in Figures (F). This method requires that the bit reversal algorithm be applied to the output X(k). Note that the butterfly for the DIF algorithm differs slightly from the decimation-in-time butterfly as shown in Figure (E). The use of decimation-in-time versus decimation-in-frequency algorithms is largely a matter of preference, as either yields the same result. System constraints may make one of the two a more optimal solution.

It should be noted that the algorithms required to compute the inverse FFT are nearly identical to those required to compute the FFT, assuming complex FFTs are used. In fact, a useful method for verifying a complex FFT algorithm consists of first taking the FFT of the x(n) time samples and then taking the inverse FFT of the X(k). At the end of this process, the original time samples, Re x(n), should be obtained and the imaginary part, Im x(n), should be zero (within the limits of the mathematical round off errors).

Computational Efficiency of an N-Point FFT:

The process to calculate N-point spectrum from the even N/2 point spectrum and the odd N/2 point spectrum can be progressively applied until it reaches the stage to calculate 2-point spectra. The 2nd stage calculate N/2 point spectra from N/4 point spectra. The 3rd stage does N/4 point spectra from N/8 point spectra, and so forth. Thus, it is already obvious that this process requires a less number of calculations, because the number of stages for N point FFT calculation is p, when $N = 2^p$, in terms of N itself, the number of the stages in $\log_2 N$. The total number of N calculations involving one addition & one multiplication must be carried out for each stage. Therefore, the total number of calculations for N-point FFT algorithm is $N\log_2 N$. The conventional method, on the other hand, requires N^2 calculations.

The total number of complex multiplications is reduced to $(N/2) \log_2 N$ and the number of complex additions is $N\log_2 N$.

Table 1: Computational Improvement of Radix-2 FFT Algorithm.

DFT: N^2 Complex Multiplications		**FFT: $(N/2) \log_2(N)$ Complex Multiplications**	
N	**DFT Multiplications**	**FFT Multiplications**	**FFT Efficiency**
256	**65,536**	**1,024**	**64 : 1**
512	**262,144**	**2,304**	**114 : 1**
1,024	**1,048,576**	**5,120**	**205 : 1**
2,048	**4,194,304**	**11,264**	**372 : 1**
4,096	**16,777,216**	**24,576**	**683 : 1**

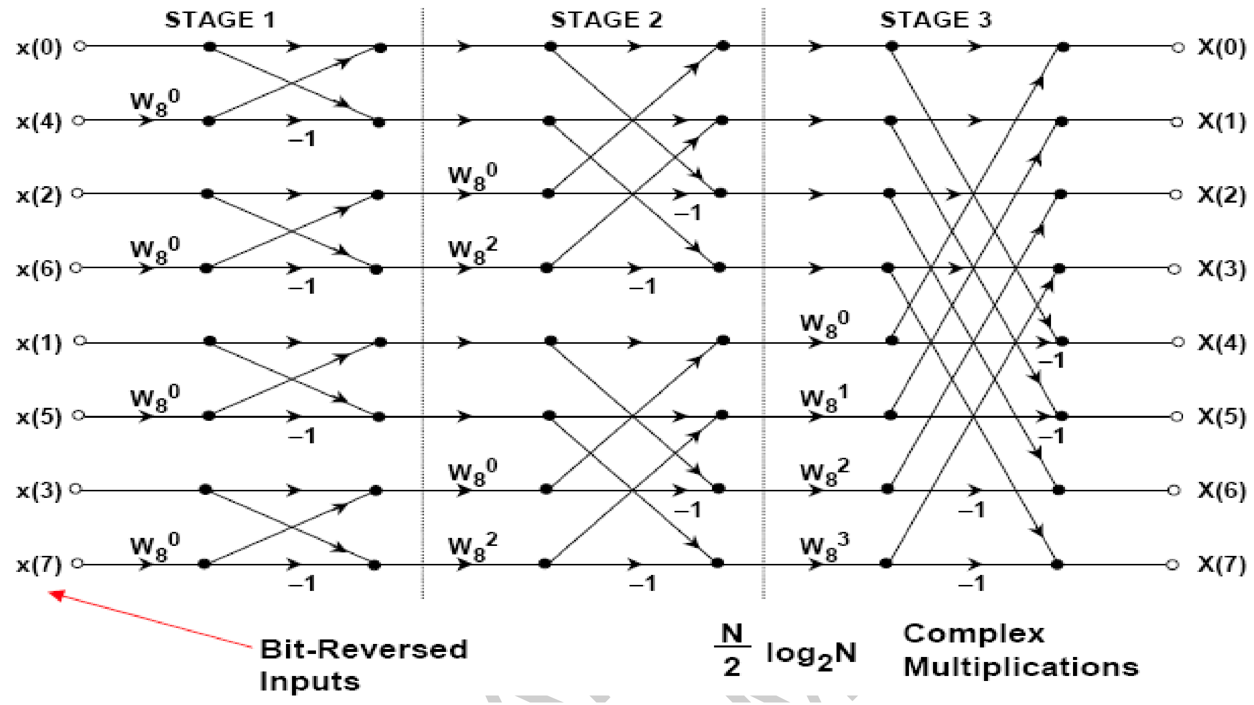
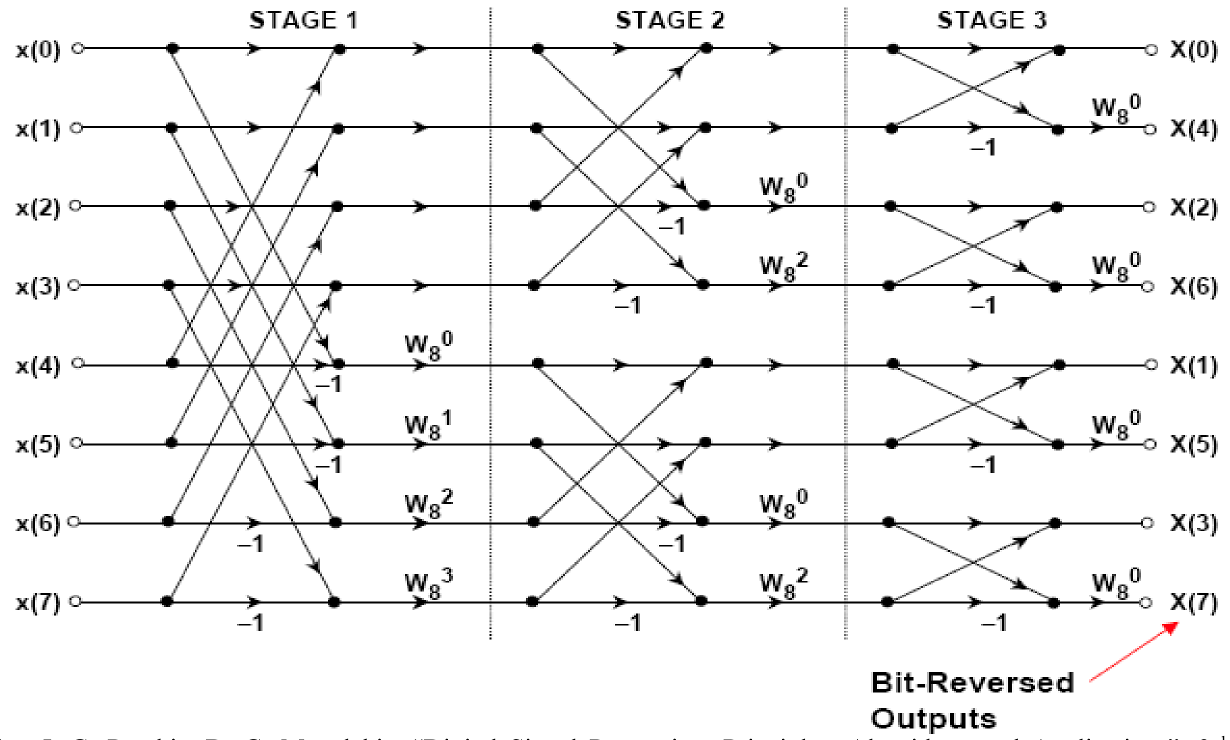


Figure: 8- point Decimation in time(above) and Decimation in Frequency(below) FFT algorithm



1. J. G. Proakis, D. G. Manolakis, “Digital Signal Processing, Principles, Algorithms and Applications”, 3rd Edition, Prentice-hall, 2000. Chapter 7 & 8.