CHAPTER-3

Chapter-3

FAST FOURIER TRANSFORM

Introduction:

Fast Fourier Transform (FFT) is a particular method of performing a series of computations that can compute the discrete Fourier transform much more rapidly than other available algorithm.

(3.1) Matrix Formulation

Consider the discrete Fourier transform.

$$X(n) = \sum_{k=0}^{N-1} x_o(k)\, e^{-i2\pi nk/N}$$

$$n = 0,\ 1,\ \ldots\ N-1. \qquad \text{-------(3.1.1)}$$

Where k, n stands for kT & n/NT resp. This is system of N equations in N unknowns.

For example consider N = 4 & let $W = e^{-i2\pi/N}$ then (3.1.1) gives.

$$\left.\begin{aligned} X(0) &= x_o(0)W^o + x_o(1)W^o + x_o(2)W^o + x_o(3)W^o \\ X(1) &= x_o(0)w^o + x_o(1)W^1 + x_o(2)W^2 + x_o(3)W^3 \\ X(2) &= x_o(0)W^o + x_o(1)W^2 + x_o(2)W^4 + x_o(3)W^6 \\ X(3) &= x_o(0)W^o + x_o(1)W^3 + x_o(2)W^6 + x_o(3)W^9 \end{aligned}\right\} \text{-------(3.1.2)}$$

In Matrix form it can be written as

$$\begin{pmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{pmatrix} = \begin{pmatrix} W^o & W^o & W^o & W^o \\ W^o & W^1 & W^2 & W^3 \\ W^o & W^2 & W^4 & W^6 \\ W^o & W^3 & W^6 & W^9 \end{pmatrix} \begin{pmatrix} X_o(0) \\ X_o(1) \\ X_o(2) \\ X_o(3) \end{pmatrix} \text{-------(3.1.3)}$$

or $X(n) = W^{nk} x_o(k)$ -------(3.1.4)

To find the solution of (3.1.2) it require N^2 complex multiplications and N(N-1) complex additions. Therefore we are interested to develop a algorithm which will reduce the number of complex multiplications and complex additions.

Since $W = e^{-i2\pi/N}$ we can write (3.1.3) as

$$\begin{pmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & W^2 & W^3 \\ 1 & W^2 & W^0 & W^2 \\ 1 & W^3 & W^2 & W^1 \end{pmatrix} \begin{pmatrix} X_o(0) \\ x_o(1) \\ x_o(2) \\ x_o(3) \end{pmatrix} \text{-------(3.1.5)}$$

By factorization of coefficient matrix we get,

$$\begin{pmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{pmatrix} = \begin{pmatrix} 1 & W^o & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{pmatrix} \begin{pmatrix} 1 & 0 & W^o & 0 \\ 0 & 1 & 0 & W^o \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{pmatrix} \begin{pmatrix} X_o(o) \\ x_o(1) \\ x_o(2) \\ X_o(3) \end{pmatrix}$$

-------(3.1.6)

But simplification on the R.H.S. gives the vector X(n) with row interchanged as shown. Let us denote it by

$$\bar{X}(n) = \begin{pmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{pmatrix}$$

Let us consider

$$\begin{pmatrix} x_1(0) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{pmatrix} = \begin{pmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^0 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^2 \end{pmatrix} \begin{pmatrix} x_o(0) \\ x_o(1) \\ x_o(2) \\ x_o(3) \end{pmatrix} \text{-------(3.1.7)}$$

Multiplication on the r.h.s. gives

$$x_1(0) = x_o(0) + W^0 x_o(2) \text{-------(3.1.8)}$$

which includes only one complex multiplication & a addition.

Similarly,

$$x_1(1) = x_o(1) + W^0 x_0(3) \text{------(3.1.9)}$$

$$x_1(2) = x_o(0) + W^2 x_o(2) = x_0(0) - W^0 x_o(2)$$

$$(\because W^2 = -W^0)$$

i.e. $x_1(2) = x_0(0) - W^0 x_o(2)$ ------(3.1.10)

and $x_1(3) = x_o(1) + W^2 x_o(3) = x_0(1) - W^0 x_o(3)$

Thus, $x_1(3) = x_o(1) - W^0 x_o(3)$ ------(3.1.11)

The complex multiplication in (3.1.10) is already done in equation (3.1.8) hence it includes only one complex addition and the complex multiplication in (3.1.11) is done in equation (3.1.8) hence it includes only one complex addition.

Thus the intermediate vector $x_1(k)$ is then determined by only four complex additions and two complex multiplications. Next consider,

$$\begin{pmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{pmatrix} = \begin{pmatrix} x_2(0) \\ x_2(1) \\ x_2(2) \\ x_2(3) \end{pmatrix} = \begin{pmatrix} 1 & W^0 & 0 & 0 \\ 1 & W^2 & 0 & 0 \\ 0 & 0 & 1 & W^1 \\ 0 & 0 & 1 & W^3 \end{pmatrix} \begin{pmatrix} x_1(o) \\ x_1(1) \\ x_1(2) \\ x_1(3) \end{pmatrix}$$

------(3.1.12)

i.e. $x_2(0) = x_1(0) + W^0 x_1(1)$

$x_2(1) = x_1(0) + W^2 x_1(1) = x_1(0) - W^0 x_1(1)$

$x_2(2) = x_1(2) + W^1 x_1(3)$

$x_2(3) = x_1(2) + W^3 x_1(3) = x_1(2) - W^1 x_1(3)$

Thus vector $x_2(k)$ is determined by only two complex multiplications and four complex additions.

Computation of $\bar{X}(n)$ thus requires a total of four complex multiplications and eight complex additions. But computation of X(n) from (3.1.3) requires sixteen complex multiplications and twelve complex additions. Factorization of a matrix reduces the number of multiplications and additions. Since computation time is proportional to the number of calculations factorization increases the efficiency of FFT algorithm.

If $N = 2^\gamma$ then FFT algorithm factorize N×N matrix into γ matrices each of order N×N. It reduces the computation time as

the number of complex multiplications and additions are $N\gamma/2$ and $N\gamma$ resp. The approximate ratio of direct to FFT computing time is given by

$$\frac{N^2}{N\gamma/2} = \frac{2N}{\gamma}$$

The matrix factoring gives

$$\bar{X}(n) = \begin{pmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{pmatrix} \qquad \text{Instead of} \qquad X(n) = \begin{pmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{pmatrix}$$

To obtain the required vector X(n) we rewrite $\bar{X}(n)$ by replacing arguments with its binary equivalent.

$$\begin{pmatrix} X(0) \\ X(2) \\ X(1) \\ X(3) \end{pmatrix} = \begin{pmatrix} X(00) \\ X(10) \\ X(01) \\ X(11) \end{pmatrix}$$

If we reverse the binary arguments in 2nd and 3rd row we get.

$$X(n) = \begin{pmatrix} X(00) \\ X(10) \\ X(01) \\ X(11) \end{pmatrix} \text{ Flips to } \begin{pmatrix} X(00) \\ X(01) \\ X(10) \\ X(11) \end{pmatrix}$$

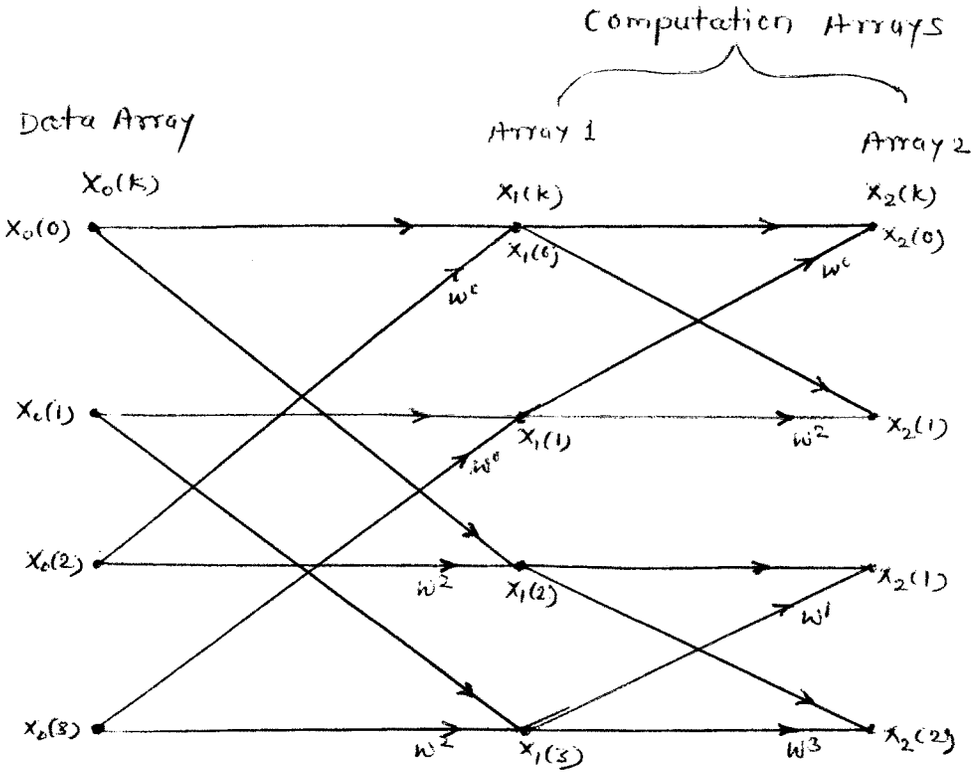
This is useful in development of FFT. For large value of N matrix factorization process is cumbersome so that graphical formulation is used to develop a flow graph for a computer program.

Signal Flow Graph:-

We convert equation (3.1.3) into signal flow graph as shown. We represent the data vectors $x_0(k)$ by a vertical column of nodes on the left of the graph. The 2^{nd} vertical array of nodes is the vector $x_1(k)$ computed in equation (3.1.7) and the next vertical column corresponds to the vector $x_2(k) = \overline{X}(n)$ as computed in equation (3.1.12) The no. of vertical arrays equals the number of matrices obtained by factorization i.e. γ.

The interpretation of flow graph is as follows: The two sclid lines incident in each node are called the transmission paths from previous node. A path brings a quantity from a previous node, multiplies the quantity by W^p and inputs the result into the node in the next array. W^p appears below the arrowhead and if $W^p = 1$ no need to write it. Results entering a node from the two transmission paths are combined additively.

Thus signal flow graph is a brief method for representing the computations in FFT algorithm. This method is easy to utilize.



(FFT signal flow graph, $N=4$)

(figure 3.1) (a)

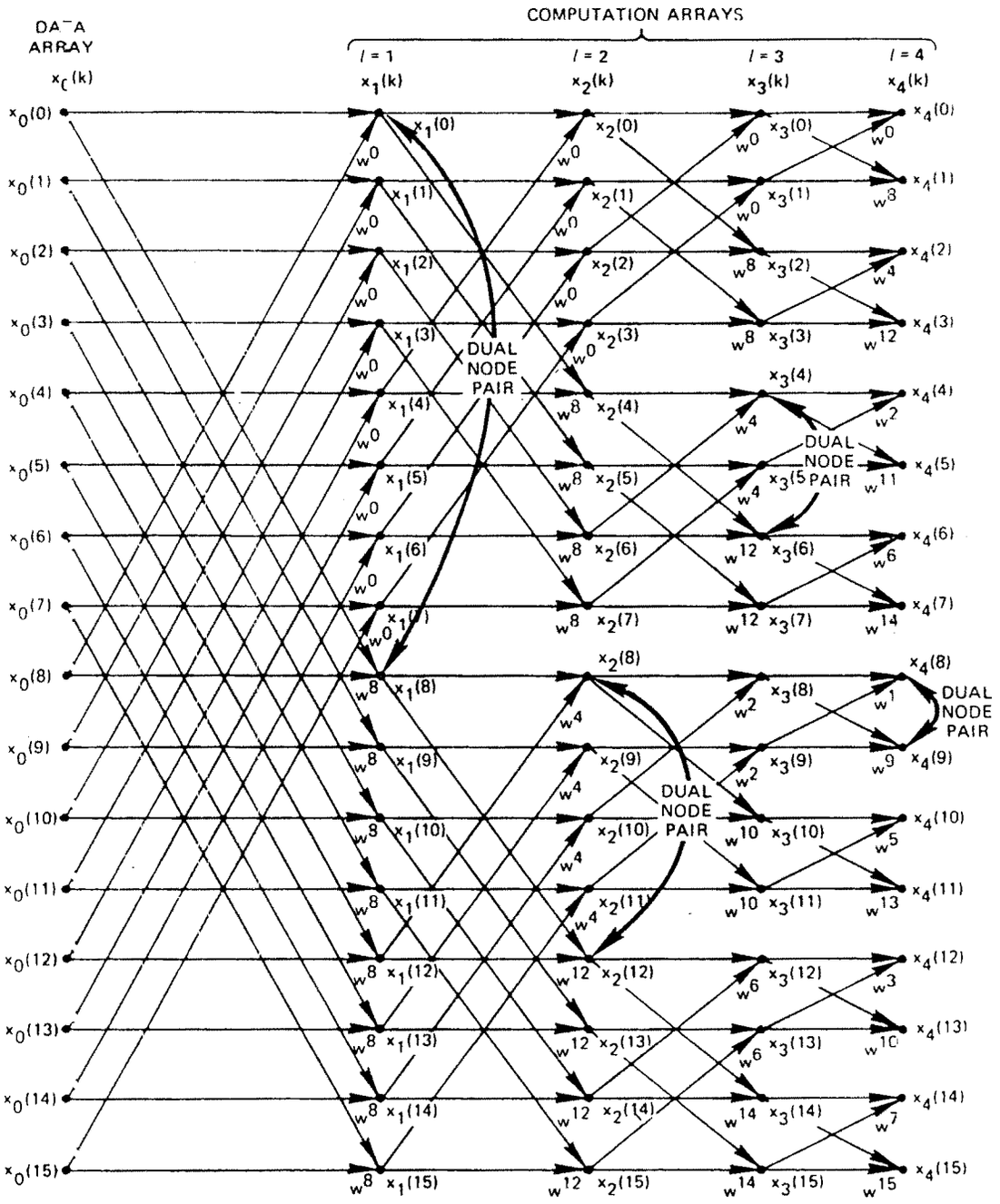

DATA ARRAY
$x_0(k)$
COMPUTATION ARRAYS
$l = 1$
$l = 2$
$l = 3$
$l = 4$
$x_1(k)$
$x_2(k)$
$x_3(k)$
$x_4(k)$
DUAL NODE PAIR
DUAL NODE PAIR
DUAL NODE PAIR
DUAL NODE PAIR

Figure 3.1 (b)

Dual Nodes: In every array the pair of nodes corresponds to the transmission paths which leaves the same pair of nodes in the previous array. The pair leaving the nodes is said to be dual node pair.

For example consider signal flow graph for N = 16 (fig.3.1)(b) the nodes $x_1(1)$ & $x_1(9)$ are computed in terms of $x_C(1)$& $x_o(9)$. But $x_0(1)$ & $x_0(9)$ does not take part in any other computation. Hence it is possible to compute $x_1(1)$ & $x_1(9)$ simultaneously and stored in the previous storage locations of $x_0(1)$ & $x_0(9)$. In above figure (fig.3.1)(b) in array ℓ=1, dual node pair say $x_1(0)$; $x_1(8)$ is separated by k=8, $N/2^{\ell} = N/2$ In array ℓ=2, a dual node pair, say $x_2(8)$; $x_2(12)$ is separated by $k=4 = N/2\ell = N/2^2$, Similarly a dual node pair $x_3(4)$; $x_3(6)$ in array ℓ=3 is separated by $k=2 = N/2^{\ell} = N/2^3$ etc. Generalizing we get the spacing between dual nodes in array ℓ is $N/2^{\ell}$. Thus dual node of $x_1(k)$ is $x_1(k + N/2^{\ell})$.

Dual Node Computation:

If the weighting factor at one node is W^p, then the weighting factor at the dual node is $W^{p+N/2}$ and since $W^p = -W^{p+N/2}$ only one multiplication is required in the computation of a dual node pair. It is given by

$$\left.\begin{aligned} x_\ell(k) &= x_{\ell-1}(k) + W^p\, x_{\ell-1}(k+N/2^\ell) \\ x_\ell(k+N/2^\ell) &= x_{\ell-1}(k) - W^p x_{\ell-1}(k+N/2^\ell) \end{aligned}\right\} \quad ---(3.1.14)$$

As above the dual node in the ℓ^{th} array is always down $N/2^\ell$ in the array. Since the spacing is $N/2^\ell$, we must skip after every $N/2^\ell$ node. In general we will compute equation (3.1.14) for the first $N/2^\ell$ nodes, skip the next $N/2^\ell$ and so on. We will stop the skipping if node index greater than (N-1).

Determination of W^p

The value of p is determined by -

i) Writing the index k in binary form with γ bits.

ii) Scaling the binary number by $(\gamma - 1)$ bits to the right and filling zeros to the left.

iii) Reversing the order of the bits. This bit-reversed number is the P.

For example consider $x_2(8)$. Since $\gamma=4$, $k=8$, & $\ell=2$, then k in binary is 1000. Scale this number by $\gamma - \ell = 4-2 = 2$ places to the right and fill in zeros. i.e. 0010 After reversing the order it gives 0100 i.e. 4.

$\therefore$ P = 4.

In the final step in computing the F.F.T. we came to know that while reversing the binary number if we proceed down the

array, interchanging x(k) with suitable x(i) we may have node previously interchanged. To skip a node that has previously been interchanged, we check the condition

if $i < k$ (i is the integer obtained by bit-reversing k.)

If condition is satisfied this implies node has been interchanged by a previous operation.

FFT Algorithms for Real data:

If we consider the time function in the complex form, that is, $h(k) = h_r(k) + h_i(k)$ then the discrete transform is generally in the complex form as

$$H(n) = (1/N) \sum_{k=0}^{N-1} \left[h_r(k) + i\, h_i(k)\right] e^{-i2\pi nk/N} \qquad \text{-------(3.1.15)}$$

and the alternate inversion formula is given by

$$h(k) = (1/N) \left[\sum_{n=0}^{N-1} \left[H_r(n) + iH_i(n)\right]^*\right]^* \qquad \text{------(3.1.16)}$$

Since equation (3.1.15) & (3.1.16) contain the common factor $e^{-i2\pi nk/N}$ both the values can be calculated in a single computer program. If given time function is real function then its discrete Fourier transform is computed as follows :

Let h(k) & g(k) be two real time functions, consider

$y(k) = h(k) + i\, g(k)$.

from linear property

$$Y(n) = H(n) + iG(n)$$

$$= \left[H_r(n) + iH_i(n)\right] + i\left[G_r(n) + iG_i(n)\right]$$

$$= \left[H_r(n) - G_i(n)\right] + i\left[H_i(n) + G_r(n)\right]$$

$$= R(n) + iI(n) \qquad \text{-------(3.1.17)}$$

Decompose R(n) & I(n) in even & odd component

$$\therefore \quad Y(n) = \left[\frac{R(n) + R(N-n)}{2}\right] + \left[\frac{(R(n) - R(N-n)}{2}\right]$$

$$+i\left[\frac{(I(n)+I(N-n)}{2}\right] + i\left[\frac{I(n) - I(N-n)}{2}\right]$$

$$\text{-------(3.1.18)}$$

$$\therefore \quad H(n) = R_e(n) + i\, I_o(n)$$

$$\left[\frac{R(n)+R(N-n)}{2}\right] + i\left[\frac{I(n) - I(N-n)}{2}\right]$$

$$\text{-------(3.1.19)}$$

Similarly,

$$iG(n) = R_o(n) + iI_e(n) \quad \text{or} \quad G(n) = I_e(n) - iR_o(n)$$

$$\therefore \quad G(n) = \left[\frac{I(n)+I(N-n)}{2}\right] - i\left[\frac{R(n) - R(N-n)}{2}\right]$$

$$\text{-------(3.1.20)}$$

Thus (3.1.19) & (3.1.20) gives the discrete Fourier transform of two simultaneous functions.

(3.2) Transform of 2N Samples with an N sample Transform:

Consider a function x(k) with 2N samples. To compute the discrete Fourier transform with N sample transform we break the 2N point function x(k) into N sample function as

$$h(k) = x(2k)$$

$$g(k) = x(2k+1), \quad k=0, 1, \ldots N-1. \qquad \text{-------(3.2.1)}$$

$$\therefore \quad X(n) = \sum_{k=0}^{2N-1} x(k)\, e^{-i2\pi nk/2N}$$

$$= \sum_{k=0}^{N-1} x(2k)\ e^{-i2\pi n(2k)/2N} + \sum_{k=0}^{N-1} x(2k+1) e^{-i2\pi n(2k+1)/2N}$$

$$= \sum_{K=0}^{N-1} h(k) e^{-i2\pi nk/N} + e^{-i\pi n/N} \sum_{K=0}^{N-1} g(k)\ e^{-i2\pi nk/N}$$

$$= H(n) + e^{-i\pi n/N}\ G(n) \qquad \text{-------(3.2.2)}$$

As before H(n) & G(n) are computed,

$$\therefore X(n) = \left[R_e(n) + iI_o(n)\right] + e^{-i\pi n/N}\left[I_e(n) - iR_o(n)\right]$$

$$= X_r(n) + iX_i(n) \qquad \text{-------(3.2.3)}$$

Where, $X_r(n) = \left(\frac{R(n) + R(N-n)}{2}\right) + \cos(\pi n/N)\left(\frac{I(n) + I(N-n)}{2}\right)$

$$- \sin\left(\frac{\pi n}{N}\right)\left(\frac{R(n) - R(N-n)}{2}\right)$$

$$\& \ X_i(n) = \left(\frac{I(n)}{2} - \frac{I(N-n)}{2}\right) - \sin(\pi n/N)\left(\frac{I(n)}{2} + \frac{I(N-n)}{2}\right)$$

$$-\cos(\pi n/N)\left(\frac{R(n)}{2} - \frac{R(N-n)}{2}\right)$$

Thus we have the computation of 2N samples Fourier transform by means of N sample transform.

(3.3) FFT Convolution And Correlation

Due to increase in computational speed which can be achieved using the FFT, it is advantageous to use frequency domain analysis. We develop the techniques for using the FFT for high speed convolution and correlation.

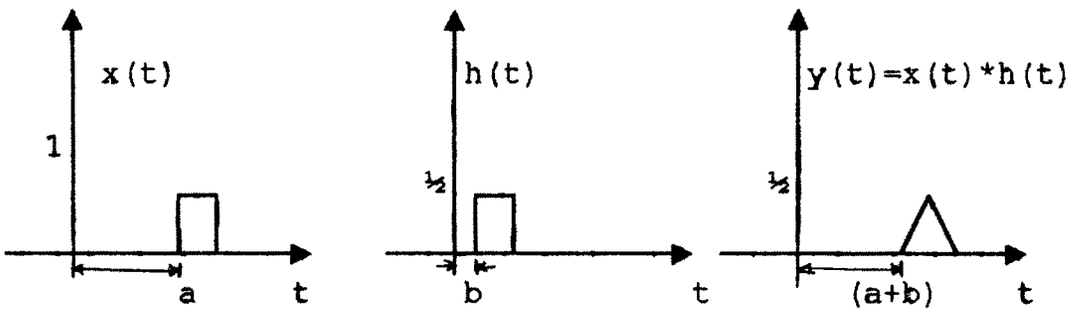
* **Finite Duration Waveforms** :-

We know, if x(k) & h(k) are periodic functions with period N then their convolution is given by

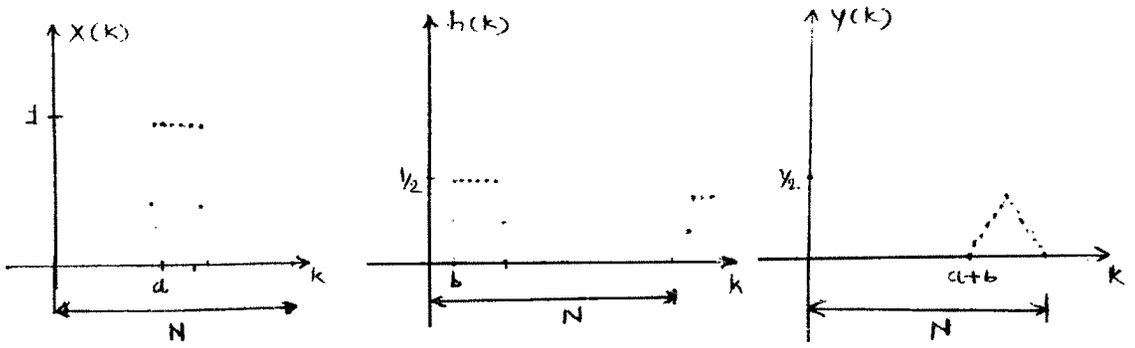
$$y(k) = \sum_{i=0}^{N-1} x(i)\, h(k-i) \qquad \text{--------(3.3.0)}$$

and it gives more correct values if it is performed correctly.

Consider the convolution of periodic functions x(t) and h(t).



(Continuous Convolution)



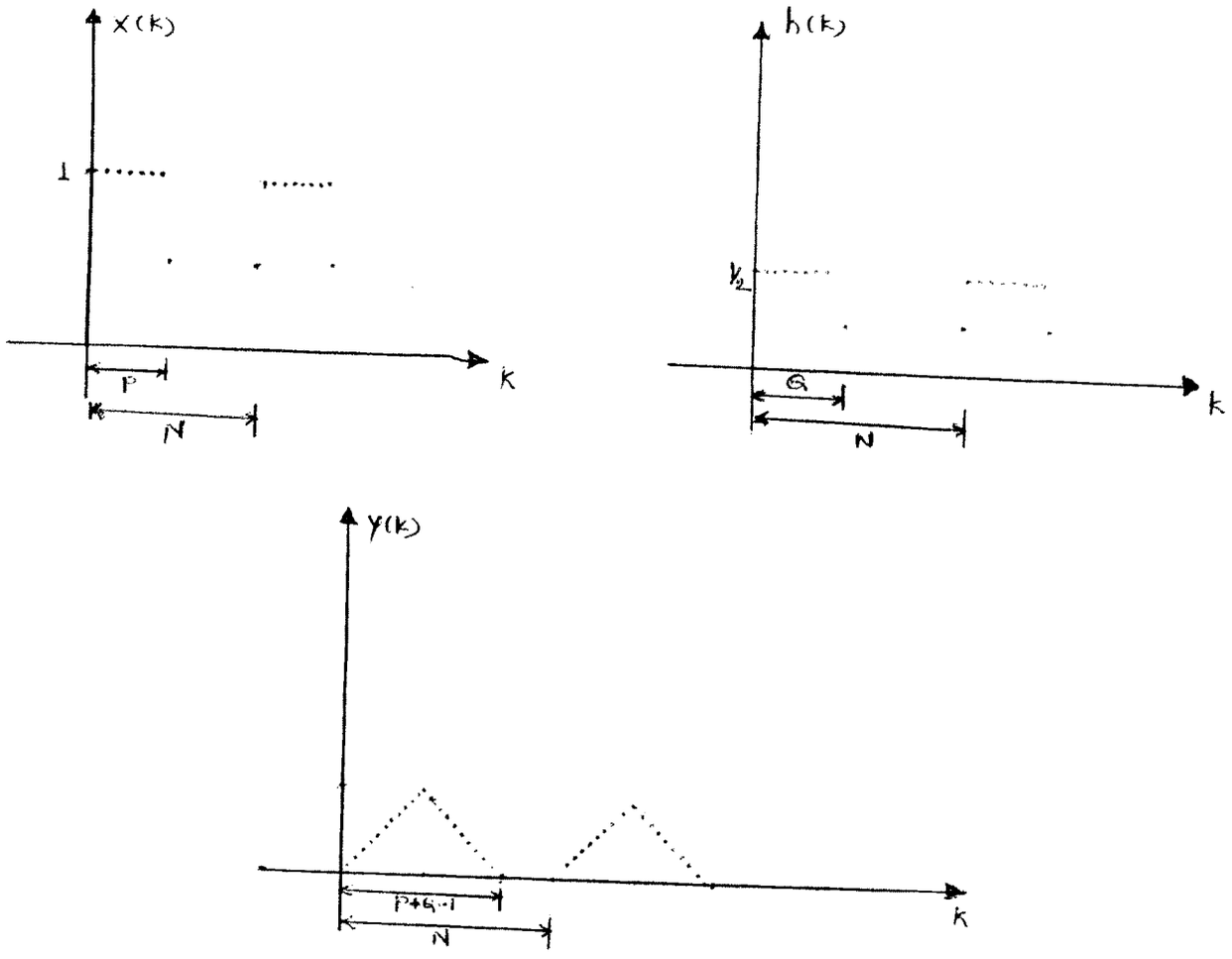
(Discrete convolution)

Since x(t) & h(t) are shifted from the origin, large N is required to avoid the overlapping or end effect.

But this results inefficient convolution as shown in following figure(fig(3.3)(c)) To compute the waveform identical with continuous using FFT we first shift both x(t) & h(t) to the origin by a & b respectively choose N >P+Q-1, $N=2^{\gamma}$, γ integer. The resulting sampled periodic functions are given by.

$$x(k) = x(kT+a),\ k=0,\ 1,\ldots P-1$$
$$= 0 \quad k=P,\ P+1,\ \ldots\ldots N-1$$

$$\& \quad h(k) = h(kT+b),\ k=0,1,\ldots\ldots\ldots\ Q-1$$
$$= 0 \quad ,k = Q,\ Q+1,\ldots\ldots..N-1 \qquad \text{-------(3.3.1)}$$



(figure (3.3)(c))

The discrete Fourier transform of x(k) & h(k) are given by

$$X(n) = \sum_{k=0}^{N-1} x(k)\ e^{-i2n\pi k/N} \qquad \text{------(3.3.2)}$$

$$H(n) = \sum_{k=0}^{N-1} h(k) e^{-i2n\pi k/N} \qquad \text{------(3.3.3)}$$

$$\therefore\ Y(n) = X(n).\ H(n) \qquad \text{------(3.3.4)}$$

Finally, the discrete convolution is given by

$$y(k) = (1/N) \sum_{n=0}^{N-1} Y(n)\ e^{i2n\pi k/N} \qquad \text{------(3.3.5)}$$

Thus equations (3.3.2),(3.3.3),(3.3.4),(3.3.5) together compute the convolution of the functions. Due to computing speed of FFT algorithm, these four equations gives a shortcut by the long way around.

Computing Speed of FFT convolution:

Time required for the computation of y(k) from equation (3.3.0) is proportional to N^2 (number of multiplications). But the time taken by (3.3.2),(3.3.3),(3.3.4) proportional to $3N \cdot \log_2(N)$ and computation time of equation (3.3.5) is proportional to N. Thus the computation of convolution using FFT is more faster than the direct computation.

FFT Correlation:

FFT correlation is very similar to FFT convolution. Consider the discrete correlation.

$$Z(k) = \sum_{i=0}^{N-1} h(i)x(k+i) \qquad \text{-------(3.3.6)}$$

Where h(k) & x(k) are the periodic functions with period N. To apply the FFT to the computation of (3.3.6) our period N should satisfy

$N \geq P + Q - 1, \qquad N=2^{\gamma}$, γ integer valued

and $x(k) = 0,\ k= 0,\ 1,\ldots(N - P)$ -------(3.3.7)

$x(k) = x(kT + a)$, $k = N - P + 1,\ N - P + 2,\ldots N -1$

After shifting & sampling x(t), i.e. P samples of x(k) are shifted to the right of the N samples defining a period. Function h(t) is shifted and sampled according to the relations

$h(k) = h(kT + b), \quad k= 0,\ 1,\ldots Q - 1$

$= 0 \qquad ,k= Q,\ Q+1,\ \ldots N - 1$ -------(3.3.8)

and then

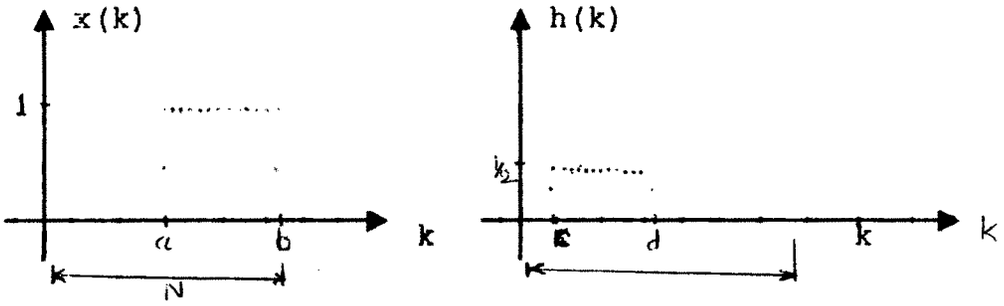
$$X(n) = \sum_{k=0}^{N-1} x(k)\ e^{-i2\pi nk/N}$$

$$H(n) = \sum_{k=0}^{N-1} h(k)e^{-i2\pi nk/N}$$

$$\therefore \quad Z(n) = X(n)H^{*}(n)$$

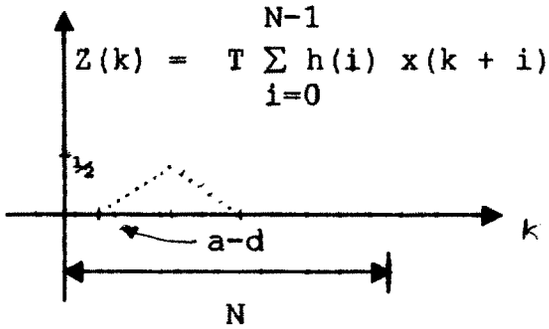
$$\therefore \quad Z(k) = (1/N) \sum_{n=0}^{N-1} Z(n)e^{-i2\pi nk/N} \qquad \text{-------(3.3.9)}$$

Ex : Consider x(k) & h(k) as shown in figure.



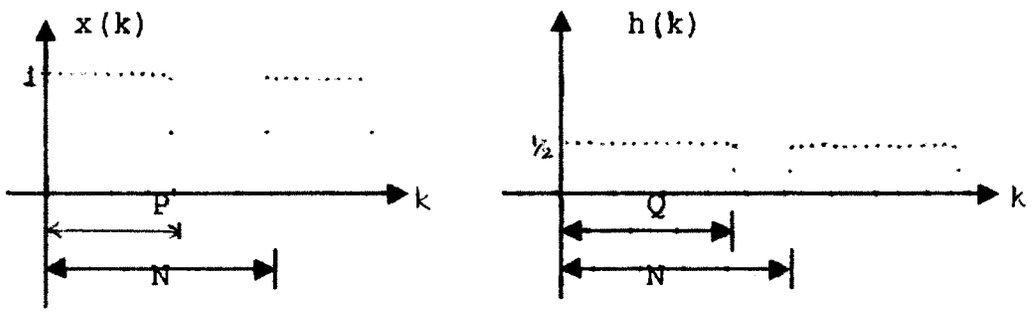
By definition correlation is shown by the following figure:

$$Z(k) = T \sum_{i=0}^{N-1} h(i)\, x(k + i)$$



Because of the choice of N correlation results in a inefficient result as the large number of zeros produced in the interval (0, a-d)

If we shift both the functions to the origin as shown in figure (3.3)(d) then the resulting correlation is given by



Figure(3.3)(d)

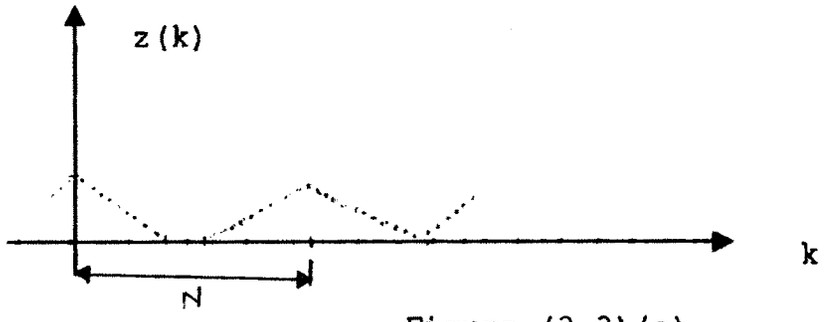


Figure (3.3)(e)

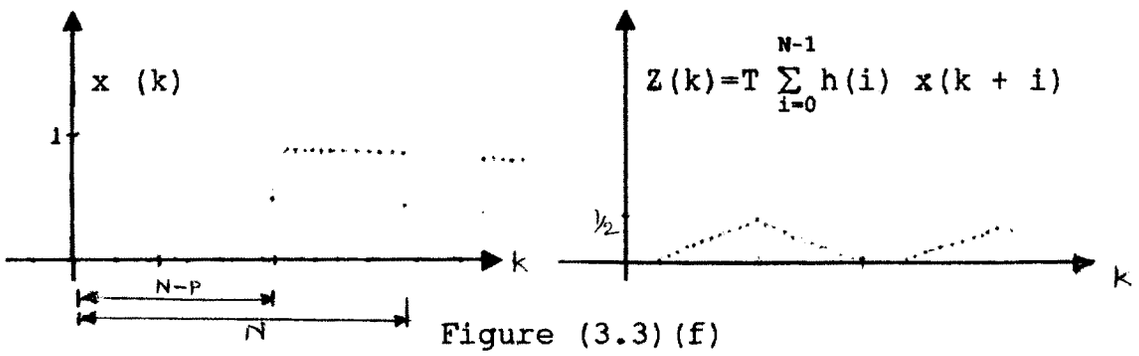


Figure (3.3)(f)

Algorithm to compute convolution of two functions:

1. Let x(t) and h(t) be finite duration waveforms shifted from origin by a and b respectively.
2. Let x(t) and h(t) be represented by P & Q samples respectively.
3. Choose N such that $N \geq P+Q-1$, $N=2^{\gamma}$, γ integer valued.
4. Define x(k) & h(k) as follows :

$$x(k) = 0, \quad k = 0,1 \ldots N-P,$$

$$= x(kT + a); \quad k=N-P+1, \quad N-P+2, \ldots N-1$$

$$h(k) = h(kT+b), \quad k = 0,1 \ldots Q-1$$

$$= 0, \quad k = Q, Q+1, \text{---} N-1$$

5. Compute discrete transform of x(k) & h(k)

$$X(n) = \sum_{k=0}^{N-1} x(k)\ e^{-i2\pi nk/N}$$

$$H(n) = \sum_{k=0}^{N-1} h(k)\ e^{-i2\pi nk/N}$$

6. Compute the product

Z(n) = x(n) H*(n), (* indicates complex conjugate)

7. Compute the value of convolution by

$$z(k) = \sum_{n=0}^{N-1} (1/N)\ Z^*(n) e^{-i2\pi nk/N}$$

(3.4) Theoretical Development of the Base 2 FFT Algorithm:

Consider the Fourier transform of $x_0(k)$

$$X(n) = \sum_{k=0}^{N-1} x_0(k) e^{-i2\pi nk/N} = \sum_{k=0}^{N-1} x_0(k)\ W^{nk}$$

$n = 0, 1, \ldots N-1.$ -------(3.4.1)

Let $N = 4 = 2^2 \quad \Rightarrow \gamma = 2$

$\therefore$ Representing n & k in binary numbers.

k	Binary (k_1, k_0)	n	Binary (n_1, n_0)
0	00	0	00
1	01	1	01
2	10	2	10
3	11	3	11

Then we can write

$$k = 2k_1 + k_0 \quad \& \quad n = 2n_1 + n_0$$

Therefore equation (3.4.1) becomes

$$X(n_1, n_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} x_0(k_1, k_0)\, W^{(2n_1 + n_0)(2k_1 + k_0)} \qquad \text{-------(3.4.2)}$$

$$= \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} x_0(k_1, k_0) W^{(2n_1 + n_0)2k_1 + (2n_1 + n_0)k_0}$$

$$= \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} x_0(k_1, k_0) W^{2n_0 k_1 + (2n_1 + n_0)k_0}$$

$$(\because W^{4n_1 k_1} = 1)$$

$$= \sum_{k_0=0}^{1} \left[\sum_{k_1=0}^{1} x_0(k_1, k_0) W^{2n_0 k_1} \right] W^{(2n_1 + n_0)K_0} \qquad \text{-------(3.4.3)}$$

First write the bracketed part as

$$x_1(n_0, k_0) = \sum_{k_1=0}^{1} x_0(k_1, k_0) W^{2n_0 k_1} \qquad \text{-------(3.4.4)}$$

Thus equation (3.4.4) in matrix form gives the first of the factored matrices or array ℓ=1 of signal flow graph similarly consider

$$x_2(n_0, n_1) = \sum_{k_0=0}^{1} x_1(n_0, k_0) W^{(2n_1 + n_o)k_o} \qquad \text{-------(3.4.5)}$$

This determines the second of the factored matrices from (3.4.3), (3.4.4), & (3.4.5)

$$X(n_1, n_0) = x_2(n_0, n_1) \qquad \text{-------(3.4.6)}$$

Equations (3.4.4), (3.4.5) & (3.4.6) represents the original cooley-Tukey formulation of the FFT algorithm for N=4.

Now, consider $N=8 = 2^3$ hence $\gamma = 3$

Therefore, Binary representation of n & k is

n, k	Binary equivalent (n_2, n_1, n_0)
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Therefore n & k in terms of binary equivalent are represented as

$$n = 4n_2 + 2n_1 + n_0; \quad k = 4k_2 + 2k_1 + k_0$$

-------(3.4.7)

Equation (3.4.1) becomes

$$X(n_2,n_1,n_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} x_0(k_2,k_1,k_0)\, W^{(4n_2+2n_1+n_o)(4k_2+2k_1+k_o)}$$

-------(3.4.8)

Simplification gives

$$X(n_2,n_1,n_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} [x_0(k_2,k_1,k_0) W^{4n_o k_2}\, W^{(2n_1+n_o)2k_1} \times W^{(4n_2+2n_1+n_o)k_o}] \quad -------(3.4.9)$$

Now let

$$x_1(n_0,k_1,k_0) = \sum_{k_2=0}^{1} x_0(k_2,k_1,k_0)\,W^{4n_0k_2}$$

$$x_2(n_0,n_1,k_0) = \sum_{k_1=0}^{1} x_1(n_0,k_1,k_0)\,W^{(2n_1+n_0)2k_1}$$

$$x_3(n_0,n_1,n_2) = \sum_{k_0=0}^{1} x_2(n_0,n_1,k_0)\,W^{(4n_2+2n_1+n_0)k_0}$$

$$X(n_2,n_1,n_0) = X_3(n_0,n_1,n_2) \qquad \text{-------(3.4.10)}$$

Thus (3.4.10) determines the matrix factorization or FFT signal flow graph for N=8.

Now to develop the general result consider $N=2^\gamma$, γ integer. Then n & k can be represented in binary form as

$$\left.\begin{aligned} n &= 2^{\gamma-1}\,n_{\gamma-1} + 2^{\gamma-2}\,n_{\gamma-2} + \ldots + n_0 \\ k &= 2^{\gamma-1}\,k_{\gamma-1} + 2^{\gamma-2}\,k_{\gamma-2} + \ldots + k_0 \end{aligned}\right\} \qquad \text{------(3.4.11)}$$

$\therefore$ Equation (3.4.1) becomes

$$x(n_{\gamma-1},n_{\gamma-2},\ldots n_0) = \sum_{k_0=0}^{1}\sum_{k_1=0}^{1}\cdots\sum_{k_{\gamma-1}=0}^{1} x(k_{\gamma-1},k_{\gamma-2},\ldots k_0)\,W^{P} \qquad \text{------(3.4.12)}$$

Where $P = (2^{\gamma-1}\,n_{\gamma-1} + 2^{\gamma-2}\,n_{\gamma-2} + \ldots + n_0)(2^{\gamma-1}\,k_{\gamma-1} + 2^{\gamma-2}\,k_{\gamma-2} + \ldots + k_0)$ ------(3.4.13)

$$= (2^{\gamma-1}\,n_{\gamma-1} + 2^{\gamma-2}\,n_{\gamma-2} + \ldots + n_0)\;2^{\gamma-1}\,k_{\gamma-1}$$

$$+ (2^{\gamma-1}\,n_{\gamma-1} + 2^{\gamma-2}\,n_{\gamma-2} + \ldots + n_0)\;2^{\gamma-2}\,k_{\gamma-2} +$$

$$
\begin{aligned}
&\quad . \; . \; .+(2^{\gamma-1}\, n_{\gamma-1} + 2^{\gamma-2}\, n_{\gamma-2} + \ldots + n_o)\, k_o \\
&= (2^{2\gamma-2}\, n_{\gamma-1}\, k_{\gamma-1} + 2^{2\gamma-3} n_{\gamma-2}\, k_{\gamma-1} + \ldots + 2^{\gamma-1}\, n_0\, k_{\gamma-1}) + \\
&+ (2^{2\gamma-3}\, n_{\gamma-1} k_{\gamma-2} + 2^{2\gamma-4}\, n_{\gamma-2}\, k_{\gamma-2} + \ldots + 2n_1\, 2^{\gamma-2}\, k_{\gamma-2.} + n_0\, 2^{\gamma-2}\, k_{\gamma-2}) \\
&\quad + \ldots + (2^{\gamma-1}\, n_{\gamma-1}\, k_0 + 2^{\gamma-2}\, n_{\gamma-2}\, k_0 + \ldots + n_o k_o) \\
&= \left[2^{\gamma}\, (2^{\gamma-2}\, n_{\gamma-1}\, k_{\gamma-1}) + 2^{\gamma}(2^{\gamma-3}\, n_{\gamma-2}\, k_{\gamma-1}) + \ldots + 2^{\gamma-1}\, n_o\, k_{\gamma-1} \right] + \\
&\quad + \left[2^{\gamma}\, (2^{\gamma-3}\, n_{\gamma-1}\, k_{\gamma-2}) + 2^{\gamma}\, (2^{\gamma-4}\, n_{\gamma-2}\, k_{\gamma-2}) + \ldots + (2n_1 + n_0)\, 2^{\gamma-2}\, k_{\gamma-2} \right] + \\
&\quad \ldots + \left[2^{\gamma-1}\, n_{\gamma-1} k_0 + 2^{\gamma-2}\, n_{\gamma-2}\, k_0 + \ldots + n_0 k_0 \right]
\end{aligned}
$$

$$
\text{since } W^{2^{\gamma}} = W^{N} = (e^{-i2\pi/N})^{N} = e^{-i2\pi} = 1
$$

$$
W^{p} = W^{2^{\gamma-1} n_0 k_{\gamma-1} + (2n_1+n_0)\, 2^{\gamma-2}\, k_{\gamma-2} + \ldots + (2^{\gamma-1} n_{\gamma-1} + 2^{\gamma-2} n_{\gamma-2} + \ldots + n_0) k_0}
$$

$$
= W^{2^{\gamma-1} n_0 k_{\gamma-1}}\; W^{(2n_1+n_0)\, 2^{\gamma-2}\, k_{\gamma-2}} \ldots W^{(2^{\gamma-1} n_{\gamma-1} + 2^{\gamma-2} n_{\gamma-2} + \ldots + n_0) k_0}
$$

------(3.4.14)

From equation (3.4.12) gives

$$
X(n_{\gamma-1}, n_{\gamma-2}, \ldots, n_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} \cdots \sum_{k_{\gamma-1}=0}^{1} x(k_{\gamma-1}, k_{\gamma-2}, \ldots, k_0) \times W^{2^{\gamma-1} n_0 k_{\gamma-1}} \times
$$

$$
\times W^{(2n_1+n_0)\, 2^{\gamma-2}\, k_{\gamma-2}} \ldots W^{(2^{\gamma-1} n_{\gamma-1} + 2^{\gamma-2} n_{\gamma-2} + \ldots + n_0) k_0}
$$

------(3.4.15)

Performing each of the summation separately and labeling the intermediate results, we obtain

$$x_1(n_0,k_{\gamma-2},..k_0)=\sum_{k_{\gamma-1}=0}^{1}x_0(k_{\gamma-1},k_{\gamma-2},...,k_0)\times W^{2^{\gamma-1}n_0k_{\gamma-1}})$$

$$x_2(n_0,n_1,k_{\gamma-3},..k_0)=\sum_{k_{\gamma-2}=0}^{1}x_1(n_0,k_{\gamma-2},...,k_0)\times W^{(2n_1+n_0)2^{\gamma-2}k_{\gamma-2}}$$

.

.

.

$$x_\gamma(n_0,n_1,...,n_{\gamma-1})=\sum_{k_0=0}^{1}x_{\gamma-1}(n_0,n_1,...,k_0)\times W^{(2^{\gamma-1}n_{\gamma-1}+2^{\gamma-2}n_{\gamma-2}+...+n_0)k_0} \qquad \text{------(3.4.16)}$$

$$X(n_{\gamma-1},n_{\gamma-2},...,n_0) = x_\gamma(n_0,n_1,\ ...,\ n_{\gamma-1})$$

The set of equation (3.4.16) represents the cooley-Tukey formulation of the FFT for $N=2^\gamma$. For the computation of these relationships only $N\gamma/2$ complex multiplications and $N\gamma$ complex additions are required.

Cooley-Tuley Algorithm

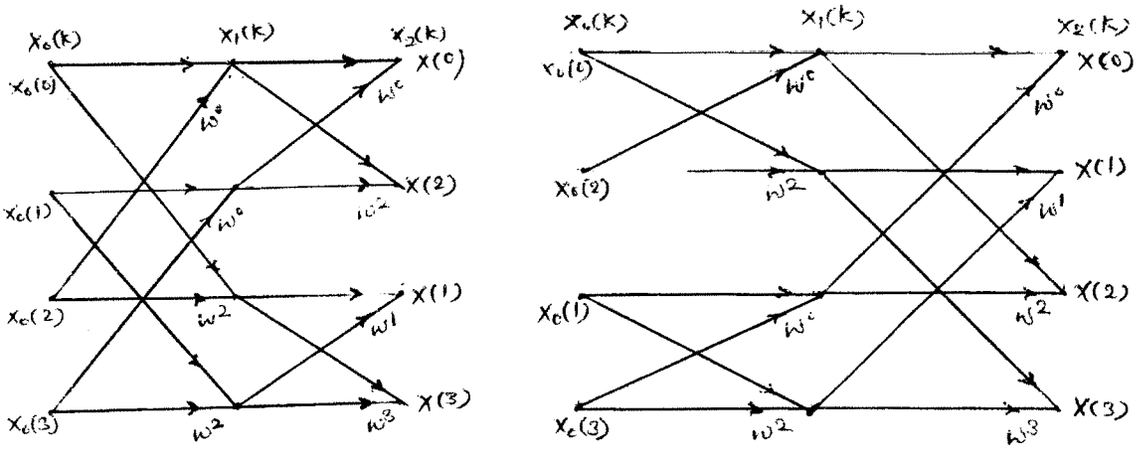
For N = 4, We have the FFT algorithm

$$x_1(n_0,\ k_0) = \sum_{k_1=0}^{1} x_0(k_0,k_1)W^{2n_0k_1}$$

$$X_2(n_0,\ n_1\ ) = \sum_{k_0=0}^{1} x_1(n_0,k_0)\ W^{(2n_1+n_0)k_0}$$

In this algorithm the input data is in natural order and the output data is in scrambled order. To determine p = nk, we require k to be bit-reversed but we want k in the natural order. It is possible to rearrange the signal flow graph in order to transform the input data in scrambled order

and output data in natural order. This is obtained by only interchanging (01) & (10) in each array, carrying the inputs with that node to each node. The result signal flow graph is shown in figure (3.4)(b).



(figure 3.4)

Sande - Tukey Algorithm:

We have for N = 4, the FFT is,

$$X(n_1, n_0) = \sum_{k_0=0}^{1} \sum_{k_1=0}^{1} x_0(k_1, k_0) W^{(2n_1 + n_0)(2k_1 + k_0)}$$

------(3.4.17)

Instead of k we separate the components of n.

$$W^{(2n_1 + n_0)} \cdot W^{(2k_1 + k_0)} = W^{4n_1k_1} \cdot W^{2n_1k_0} \cdot W^{n_0(2k_1+k_0)}$$

$$= W^{2n_1k_0} \cdot W^{n_0(2k_1+k_0)} \quad (\because W^4 = 1)$$

Thus equation (3.4.1) can be written as

$$X(n_1, n_0) = \sum_{k_0=0}^{1} \left[\sum_{k_1=0}^{1} x_0(k_1, k_0) W^{2n_0k_1} W^{n_0k_0} \right] W^{2n_1k_0}$$

------(3.4.18)

Define

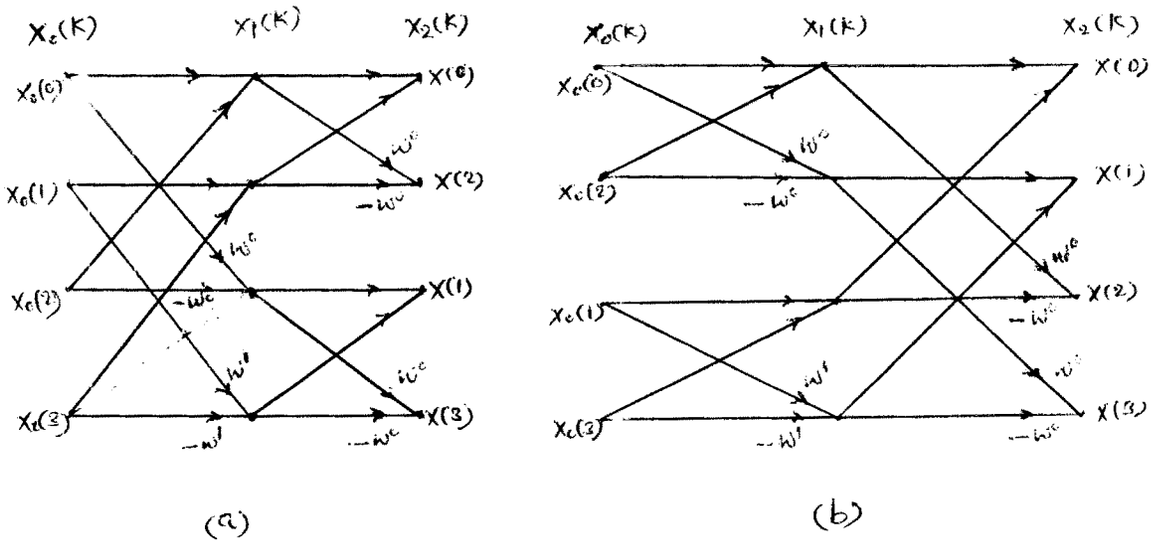
$$x_1(n_0, k_0) = \sum_{k_1=0}^{1} x_0(k_1, k_0)\ W^{2n_0k_1}\ W^{n_0k_0}$$

$$x_2(n_0, n_1) = \sum_{k_0=0}^{1} x_1(n_0, k_0)\ W^{2n_1k_0} \qquad \text{------(3.4.19)}$$

&

$$X(n_1, n_0) = x_2(n_0, n_1)$$

Using equations (3.4.19) the signal flow graph is shown as below



In this algorithm the input data is in natural order but output data is in scrambled order and the powers of W occur in natural order. To obtain natural flow signal graph result we proceed as in the cooley - Tukey algorithm. It is also possible to develop an algorithm which require natural order to occur in natural order. But it requires storage twice that of above.

(3.5) FFT Algorithms for Arbitrary Factors:

The condition $N=2^{\gamma}$ is restrictive in FFT algorithm, It is removed by considering $N = r_1 r_2 \ldots r_m$ where r_i is an integer. This is more significant than the previous in time saving.

First consider $N= r_1 r_2$ Where r_1, r_2 are integers. We can express n & k in terms of factors r_1 & r_2 as

$$n = r_1 n_1 + n_0 \quad \& \quad k = r_2 k_1 + k_0$$

$$\left.\begin{array}{ll} n_0 = 0,1 \ldots r_1-1 & n_1 = 0,1,\ldots r_2-1 \\ k_0 = 0,1\ldots r_2-1 & k_1 = 0,1,\ldots r_1-1 \end{array}\right\} \quad \text{--------(3.5.1)}$$

Then equation

$$x(n_1, n_0) = \sum_{k_0=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} x_0(k_1, k_0)\, W^{nk} \quad \text{becomes}$$

$$x(n_1, n_0) = \sum_{k_0=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} x_0(k_1, k_0)\, W^{(r_1 n_1 + n_0)(r_2 k_1 + k_0)}$$

$$= \sum_{k_0=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} x_0(k_1, k_0)\, W^{r_1 r_2 n_1 k_1 + (r_1 n_1 + n_0) k_0 + r_2 n_0 k_1}$$

$$\therefore x(n_1, n_0) = \sum_{k_0=0}^{r_2-1} \left[\sum_{k_1=0}^{r_1-1} x_0(k_1, k_0)\, W^{r_2 n_0 k_1} \right] W^{(r_1 n_1 + n_0) k_0}$$

$$(\because W^{r_1 r_2} = W^N = 1) \text{-------} (3.5.2)$$

If we consider

$$x_1(n_0, k_0) = \sum_{k_1=0}^{r_1-1} x_0(k_1, k_0)\, W^{n_0 k_1 r_2} \text{-------} (3.5.3)$$

and

$$x_2(n_0, n_1) = \sum_{k_0=0}^{r_2-1} x_1(n_0, k_0)\, W^{(r_1 n_1 + n_0) k_0} \text{-------} (3.5.4)$$

finally, $X(n_1, n_0) = x_2(n_0, n_1)$ -------(3.5.5)

Thus equation (3.5.3), (3.5.4), (3.5.5) defines the FFT algorithm for $N = r_1 r_2$.

Cooley Tukey Algorithm for $N = r_1 r_2 \ldots r_m$:-

If $N = r_1 r_2 \ldots r_m$, where $r_1, r_2 \ldots r_m$ are integers. Expressing n & k in terms of factors we get

$$\left.\begin{aligned} n &= n_{m-1}(r_1 r_2 \ldots r_{m-1}) + n_{m-2}(r_1 r_2 \ldots r_{m-2}) + \ldots + n_1 r_1 + n_0 \\ k &= k_{m-1}(r_2 r_3 \ldots r_m) + k_{m-2}(r_3 r_4 \ldots r_m) + \ldots + k_1 r_m + k_0 \end{aligned}\right\} \text{-(3.5.6)}$$

where $n_{i-1} = 0, 1, 2, \ldots r_{i-1} - 1 \quad , \quad 1 \le i \le m$

$$k_i = 0, 1, 2, \ldots r_{m-i} - 1 \quad , \quad 0 \le i \le m-1$$

$$\therefore X(n_{m-1}, n_{m-2}, \ldots n_1, n_0) = \sum_{k_0} \sum_{k_1} \ldots \sum_{k_{m-1}} x_0(k_{m-1}, k_{m-2}, k_0)\, W^{nk}$$

-------(3.5.7)

Using (3.5.6)

$$W^{nk} = W^{n\,k_{m-1}(r_2 \ldots r_m)}\, W^{n\left[k_{m-2}(r_3 \ldots r_m)+\ldots k_0\right]} \qquad \text{-------(3.5.8)}$$

hence (3.5.1) becomes

$$x(n_{m-1}, n_{m-2} \ldots n_1, n_0) = \sum_{k_0}\sum_{k_1}\ldots\left[\sum_{k_{m-1}} x_0(k_{m-1}, k_{m-2}, \ldots k_0) \times W^{n_0 k_{m-1}(r_2 r_3 \ldots r_m)}\right] \times W^{n(k_{m-2}(r_3 \ldots r_m)+\ldots+k_0)} \qquad \text{-------(3.5.9)}$$

We define

$$X_1(n_0, k_{m-2}, \ldots, k_0) = \sum_{k_{m-1}} x_0(k_{m-1}, \ldots k_0)\; W^{n_0 K_{m-1}(r_2 \ldots r_m)}$$

∴ Equation (3.5.9) becomes

$$X(n_{m-1}, n_{m-2} \ldots n_1, n_0) = \left[\sum_{k_0}\sum_{k_1}\ldots\sum_{k_{m-2}} x_1(n_0, k_{m-2}, \ldots, k_0)\; W^{n\left[k_{m-2}(r_3 \ldots r_m)+\ldots+k_0\right]}\right]$$

Define

$$x_2(n_0, n_1, k_{m-3}, \ldots, k_0) = \sum_{k_{m-2}} x_1(n_0, k_{m-2}, \ldots, k_0) W^{n[K_{m-3}(r_4 r_5 \ldots r_m)+\ldots+k_0]}$$

Continuing in this manner, we have

$$x_i(n_0, n_1 \ldots n_{i-1}, k_{m-i-1}, \ldots, k_0) = \sum_{k_{m-1}} x_{i-1}(n_0, n_1, \ldots n_{i-2}, k_{m-i} \ldots k_0) \times W^{[n_{i-1}(r_1 r_2 \ldots r_{i-1})+\ldots+n_0]}\; k_{m-i}(r_{i+1} \ldots r_m)$$

$i = 1, 2, \ldots m$

This is valid if we consider $(r_{i+1} \ldots r_m) = 1$ for $i > m-1$

and $k_1 = 0$

This gives finally

$$X(n_{m-1}, \ldots, n_0) = x_m(n_0, \ldots, n_{m-1})$$