DISCRETE FOURIER TRANSFORM AND FAST FOURIER ALGORITHM*

V. KRISHNAN

(Department of Electrical Engineering, Indian Institute of Science, Bangalore 560012)

Received on March 5, 1974

ABSTRACT

In recent years the fast Fourier transform algorithm for computing discrete Fourier transforms has come into prominence. This development is basically due to the fact that the computer time is reduced from N^2 *to* $(N/2) \log_2 N$ *where* N *is the number of discrete samples. This paper gives an exposition of the discrete Fourier transform and the fast Fourier transform algorithm program for the calculation of FFT for pulses is also given.*

Key words: Discrete Fourier transform. Fast Fourier Algorithm.

1. INTRODUCTION

Ever since the first paper on fast Fourier transform algorithm (FFT) by Cooley and Tukey [1] momentum is gaining on the acceptance of this algorithm as the saving in the number of computations is quite large. FFT is basically an efficient method of computing the discrete Fourier transform (DFT) coefficients. A number of papers [2-7] have been motivated by Cooley and his group. Bergland [7] gives an exhaustive list of references on FFT which have appeared in recent times.

The number of computations is reduced from N^2, if a direct implementation of DFT is used to $N/2 \log_2 N$, if FFT is used. The usefulness of DFT is in its effectiveness to approximate the continuous Fourier transform (CFT). The succeeding sections give a tutorial exposition of CFT, DFT and FFT. Workable program is also appended.

---

* Based partly on a paper presented at the Conference of the Computer Society of India, February, 1973.

2. CONTINUOUS AND DISCRETE FOURIER TRANSFORMS

The continuous Fourier transform (CFT) of any signal $x(t)$ is defined by

$$X(\omega) = \int_{-\infty}^{\infty} x(t)\, e^{-j\omega t}\, dt \tag{1}$$

and the inverse Fourier transform by

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega)\, e^{j\omega t}\, d\omega. \tag{2}$$

The Fourier transform pair (1) and (2) will be denoted by a double arrow

$$x(t) \leftrightarrow X(\omega). \tag{3}$$

In a similar manner the discrete Fourier transform (DFT) of a complex sequence $x(n)$ is defined by

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n)\, e^{-2\pi jnk/N} \tag{4}$$

and the inverse discrete Fourier transform (IDFT) by

$$x(n) = \sum_{k=0}^{N-1} X(k)\, e^{2\pi jnk/N}. \tag{5}$$

Let

$$e^{2\pi jnk/N} = W_N^{nk}. \tag{6}$$

The DFT pair (4) and (5) is then given by

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n)\, W_N^{-nk} \tag{7}$$

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

or in the symbols of (3)

$$x(n) \leftrightarrow x(k). \tag{8}$$

The relationship between CFT given by (1) and (2) and DFT given by (7) will be derived. $X(\omega)$ is sampled at intervals of $k\triangle\omega$ apart ($k = 0, \pm 1, \pm 2, \ldots$). A quantity $T = 2\pi/\triangle\omega$ is defined and substitution of $\omega = 2\pi k/T$ in (1) results in

$$X(k\triangle\omega) = \int_{-\infty}^{\infty} x(t)\, e^{-2\pi jkt/T}\, dt. \tag{9}$$

The above integral can be split as follows:

$$\int_{-\infty}^{\infty} = \cdots \int_{-(l+1)T}^{-lT} + \cdots \int_{-T}^{0} + \int_{0}^{T} + \cdots \int_{lT}^{(l+1)T} + \cdots$$

The relation can be compactly expressed as

$$X(k\triangle\omega) = \sum_{l=-\infty}^{\infty} \int_{lT}^{(l+1)T} x(t)\, e^{-2\pi jkt/T}\, dt. \tag{10}$$

The above relation can be rewritten by a change of variable of integration to $t - lT$ and by utilizing the periodicity of $e^{-2\pi jkt/T}$

$$X(k\triangle\omega) = \int_{0}^{T} \left[\sum_{l=-\infty}^{\infty} x(t - lT) \right] e^{-2\pi jkt/T}\, dt. \tag{11}$$

or,

$$X(k\triangle\omega) = \int_{0}^{T} x_p(t)\, e^{-2\pi jkt/T}\, dt \tag{12}$$

where,

$$x_p(t) = \sum_{l=-\infty}^{\infty} x(t - lT) \tag{13}$$

$x_p(t)$ is called the aliassed form of the continuous signal $x(t)$. Since $x_p(t)$ is a periodic signal with period T it can be expressed as a complex Fourier series

$$x_p(t) = \sum_{k=-\infty}^{\infty} a_k e^{2\pi jkt/T}. \tag{14}$$

The Fourier coefficients a_k are given by

$$a_k = \frac{1}{T} \int_{0}^{T} x_p(t)\, e^{-2\pi jkt/T}\, dt. \tag{15}$$

Hence from (12)

$$a_k = X(k\triangle\omega)\, T$$

Substitution of this result in (14) yields

$$x_p(t) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X(k\triangle\omega)\, e^{2\pi jkn\Delta t/T}. \tag{16}$$

To consider the effects of sampling in the time domain at points $n\triangle t$, $n = 0, \pm 1, \pm 2, \ldots$, (16) is written as

$$x_p(n\triangle t) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X(k\triangle\omega)\, e^{2\pi jkt/T}, \tag{17}$$

Taking $T/\Delta t\,(=N)$ to be an integer and utilizing the fact that $e^{2\pi jkn/N}$ is a periodic function of k, (17) can be put in the form

$$x_p(n\Delta t) = \frac{1}{T}\sum_{k=0}^{N-1}\left[\sum_{l=-\infty}^{\infty} X(k\Delta\omega + lN\Delta\omega)\right]e^{2\pi jkn/N}. \tag{18}$$

Let

$$X_p(k\Delta\omega) = \sum_{l=-\infty}^{\infty} X[(k+lN)\,\Delta\omega] \tag{19}$$

where $X_p(\omega) = \sum_{l=-\infty}^{\infty} X(\omega + l\omega_s)$ is the aliassed version of $X(\omega)$, ω_s being $N\Delta\omega\,(=2\pi/\Delta t)$.

Substitution of (19) in (18) and the use of (6) result in

$$Tx_p(n\Delta t) = \sum_{k=0}^{N-1} X_p(k\Delta\omega)\,W_N{}^{nk} \tag{20}$$

(20) is exactly the same form as the DFT given by (7) if $x(n)$ and $x(k)$ are defined by

$$x(n) = Tx_p(n) \tag{21}$$

$$X(k) = X_p(k) \tag{22}$$

where Δt and $\Delta\omega$ have been dropped for convenience.

From (21) and (22) the sampled aliassed version of a continuous time signal $x(t)$ and the sampled aliassed version of its Fourier transform $X(\omega)$, form a DFT pair.

Or if

$$x(t) \leftrightarrow X(\omega)$$

then

$$Tx_p(n) \leftrightarrow X_p(k). \tag{23}$$

It is now instructive to compare $X_p(\omega)$ with $X(\omega)$. The relationship is shown in Fig. 1.

$X_p(\omega)$ is a periodic function with the period equal to the sampling frequency ω_s. There are essentially two parameters, $\omega_s\,(=2\pi/\Delta t)$, and $N\,(=T/\Delta t=\omega_s/\Delta\omega)$ to be chosen in an optimal manner. The manner in which the choice of the parameters can be made is indicated next. If $X_p(\omega)$ is to approximate $X(\omega)$ then the error in the approximation is

$$\epsilon = X_p(\omega) - X(\omega) = \sum_{l=-\infty}^{\infty} X(\omega + l\omega_s) \qquad l \neq 0. \tag{24}$$

This error can be made as small as possible by choosing ω_s as large as possible and in the limit when ω_s is infinite the error is zero. If ω_s

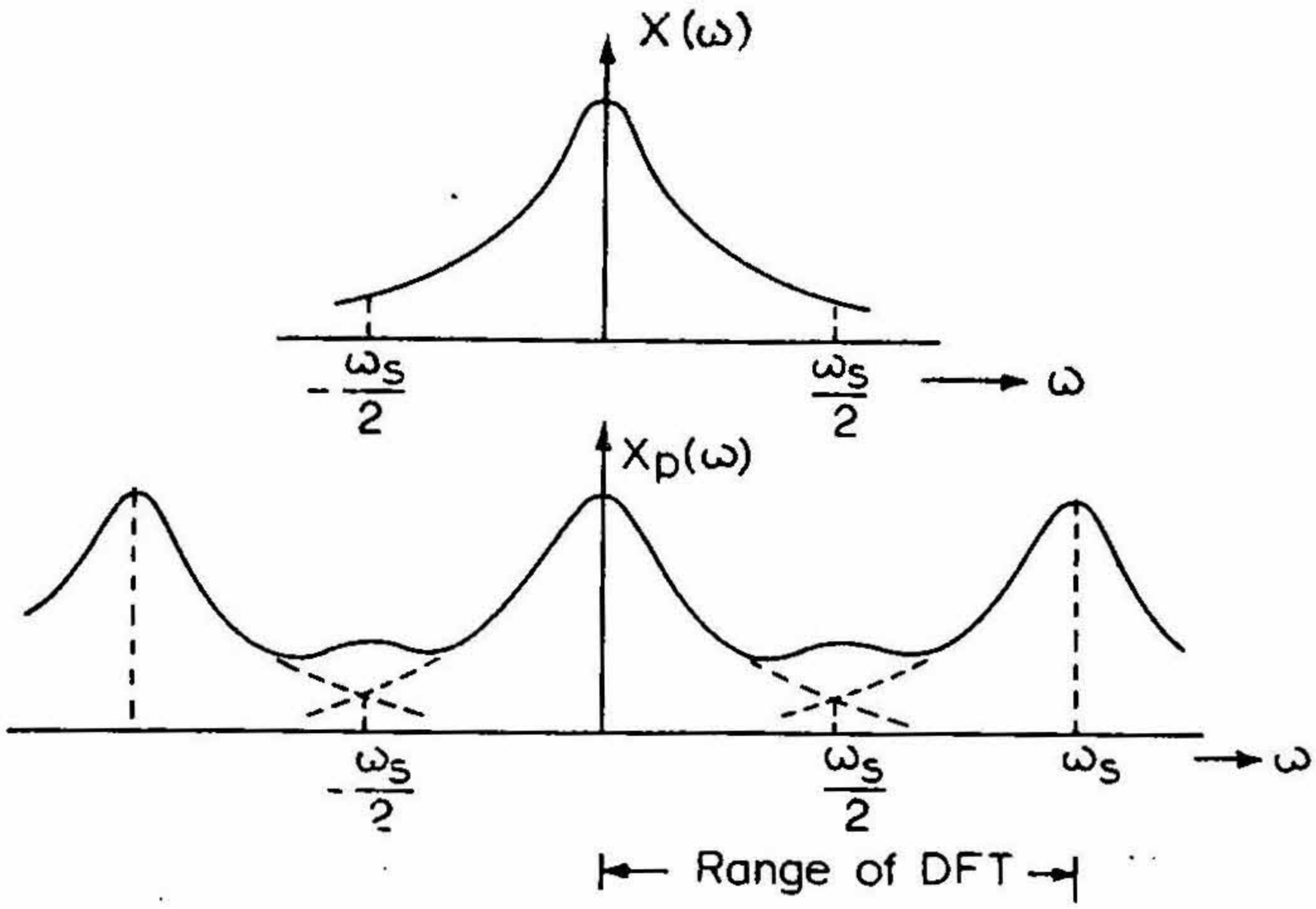


FIG. 1

is large then $\triangle t$ must be made as small as possible. However, $\triangle t$ cannot be made too small since computer round off errors will become significant. Hence a certain balance must be clearly maintained. From Fig. 1, the value of ω_s is so chosen that the contribution of $X(\omega)$ for all $|\omega| > \omega_s/2$ is very small. However, the actual form of $X(\omega)$ may not be known *a priori*. In such a situation bounds on errors can be set beforehand and $\triangle t$ is chosen such that the error between successive values of $\triangle t$ falls within the bounds prescribed. This could be programmed into the computer. In any case $\triangle t$ is chosen by some suitable method. Choice of $\triangle t$ fixes the sampling frequency ω_s.

Having chosen $\triangle t$, the next step is to choose the interval $\triangle\omega = 2\pi/N\triangle t = 2\pi/T$ at which $X_p(\omega)$ is to be sampled. For reasons to be explained in the next section it is desirable to make N as 2 raised to some integral power m. m should be large enough for a fine spacing of the frequency estimates. One criterion may be to make T such that it at least covers the non-zero part

of $x(t)$. Figure 2 makes this point clear. However in many cases this value of $N\triangle t$ will not give a fine enough spacing of the frequency estimates. This

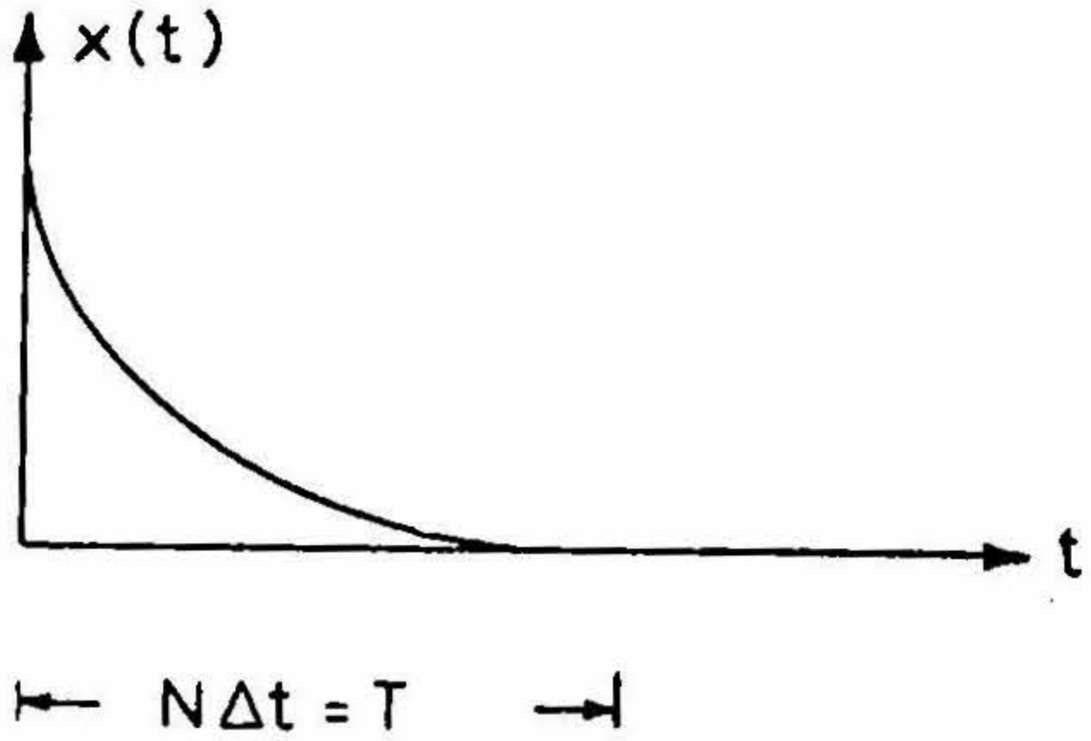


FIG. 2

is due to the fact that if the time function $x(t)$ falls rapidly to zero its Fourier transform is spread out much more. Hence certain readjustments of $N\triangle t$ have to be made in the interests of finer frequency spacings. Once m, and hence N, have been chosen the sampled aliassed time function $x_p(n\triangle t) = \Sigma\, x(n\triangle t + l\, N\triangle t)$ is formed. If the input to the FFT algorithm is the sequence $Tx_p(n\triangle t)$ the output is the DFT sequence $X_p(k\triangle\omega)$. In summary, $\triangle t$ is chosen such that ω_s is large enough to cover the region where $X(\omega)$ is significantly different from zero. However, N is chosen not necessarily to encompass the region of $x(t)$ which is significantly different from zero but to the degree of the fineness of the frequency resolution desired.

There is another matter to be made clear. Ordinarily, if $X(\omega)$ is significantly different from zero in the range $|\omega| < \omega_s/2$ then $X_p(\omega)$ will approximate $X(\omega)$ in this range

$$X(\omega) \approx X_p(\omega). \tag{25}$$

However, the DFT is viewed as the N sampled sequences in the range $[0 - \omega_s]$ of $X_p(\omega)$. As a consequence the negative ω half of $X(\omega)$ is reproduced to the right of the positive half of $X(\omega)$.

$$\begin{aligned} &X(\omega) \approx X_p(\omega) && 0 \leqslant \omega \leqslant \omega_s/2 \\ &X(\omega \approx \omega_s) - X_p(\omega) && \omega_s/2 \leqslant \omega \leqslant \omega_s. \end{aligned} \tag{26}$$

In (26) $X_p(\omega)$ does not approximate $X(\omega)$ in the interval $0 \leqslant \omega \leqslant \omega_s$ but only in the interval $0 \leqslant \omega \leqslant \omega_s/2$. This is an important point to remember.

3. Fast Fourier Transform Algorithm (Base 2)

Having shown in the previous section the connection between DFT and CFT, FFT algorithm for the computation of the IDFT given by (7)

$$x(n) = \sum_{k=0}^{N-1} X(k)\, W_N^{nk} \tag{7}$$

will be derived. From the algorithm for the IDFT the DFT given by

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

can be obtained in one of two ways.

(*a*) If the input to the algorithm is the complex sequence $x^*(n)/N$ then the output will be $X^*(k)$.

(*b*) If the input to the algorithm is $x(n)/N$ then in the structure of W_N^{nk} in the algorithm must be changed to W_N^{-nk} in which case the output will be $X(k)$.

If (7) is directly implemented in the computer N^2 multiplications and $(N-1)^2$ additions are needed. For large N the number of computations needed becomes prohibitive. The FFT algorithm will be shown to reduce the number of computations.

The fundamental principles of the algorithm will be given before going into programming considerations. The complex sequence input to the algorithm, $X(k)$, is divided into two subsequences $X_1(k)$ and $X_2(k)$, $X_1(k)$ consisting of even numbered points of $X(k)$ and $X_2(k)$ containing the odd numbered points of $X(k)$.

$$\left.\begin{aligned} X_1(k) &= X(2k) \\ X_2(k) &= X(2k+1) \end{aligned}\right\} \quad k = 0, 1, \ldots N/2 - 1. \tag{27}$$

Equation (7) can be rewritten by using (27)

$$x(n) = \sum_{k=0}^{N/2-1} X_1(k)\, W^{nk}{}_{N/2} + W_N{}^n \sum_{k=0}^{N/2-1} X_2(k)\, W_{N/2}{}^{nk} \tag{28}$$

$$n = 0, 1, 2, \ldots N-1.$$

If the index n in (28) is restricted to $0, 1, \ldots N/2 - 1$ then (28) can be written as

$$x(n) = x_1(n) + W_N^n x_2(n) \qquad n = 0, 1, \ldots N/2 - 1 \tag{29}$$

where

$$\left.\begin{array}{l} x_1(n) \leftrightarrow X_1(k) \\ x_2(n) \leftrightarrow X_2(k) \end{array}\right\} \quad n, k = 0, 1, \ldots N/2 - 1.$$

The values of $x(n)$ for $N/2 \leqslant n \leqslant N - 1$ are obtained, by utilizing the periodicity of $x_1(n)$ and $x_2(n)$, as follows.

$$x(n + N/2) = x_1(n) + W_N^{n+N/2} x_2(n). \tag{30}$$

By using the fact $W_N^{N/2} = -1$, (30) simplifies to

$$x(n + N/2) = x_1(n) - W_N^n x_2(n) \qquad n = 0, 1, \ldots N/2 - 1 \tag{31}$$

(29) and (31) gives all the samples $x(n)$ in the entire range $[0, N - 1]$. They also demonstrate that a one N point analysis is equivalent to two $N/2$ point analyses. This procedure can be continued for the resulting $N/2$ points. If N had been started with 2^m samples then m such reductions are possible. The number of computations needed for m reductions can be calculated. If CN is the number of multiplications for the N point analysis then from (29) and (31) there are two $C_N/2$ multiplications required to find $x_1(n)$ and $x_2(n)$ plus another N multiplications for finding the product $W_N^n x_2(n)$. Assuming, as stated before

$$N = 2^m$$

then

$$\begin{aligned} C_N &= 2C_{N/2} + N/2 \\ C_{N/2} &= 2C_{N/4} + N/4 \\ C_2 &= 2C_1 + 1. \end{aligned} \tag{32}$$

If (32) is solved for C_N

$$C_N = mN/2 = \tfrac{1}{2} N \log_2 N. \tag{33}$$

The FFT algorithm reduces the number of multiplications from N^2 to $N/2 \log_2 N$, a considerable amount of saving for large N. The number of additions are only a little more than the number of multiplications.

4. Example of FFT Algorithm

In this example the FFT algorithm is generated for a $N = 8$. The IDFT for this case is

$$x(n) = \sum_{k=0}^{7} X(k) W_8^{nk}. \tag{34}$$

Splitting $x(n)$ into odd and even sequences equations similar to (29) and (30) are written

$$x(n) = x_1(n) + W_8^{nk} x_2(n)$$

$$x(n+4) = x_1(n) - W_8^{n'} x_2(n). \tag{35}$$

In (35), $x_1(n)$ and $x_2(n)$ are given by

$$\left.\begin{aligned} x_1(n) &\leftrightarrow X_1(k) \\ x_2(n) &\leftrightarrow X_2(k) \end{aligned}\right\} \tag{36}$$

where

$$\left.\begin{aligned} X_1(k) &= X(2k) \\ X_2(k) &= X(2k+1). \end{aligned}\right\} \tag{37}$$

Splitting $x_1(n)$ and $x_2(n)$ into odd and even sequences results in

$$\left.\begin{aligned} x_1(n) &= x_3(n) + W_8^{2n} x_4(n) \\ x_1(n+2) &= x_3(n) - W_8^{2n} x_4(n) \end{aligned}\right\} \quad 0 \leqslant n \leqslant 1 \tag{38}$$

$$\left.\begin{aligned} x_2(n) &= x_5(n) + W_8^{2n} x_6(n) \\ x_2(n+2) &= x_5(n) - W_8^{2n} x_6(n) \end{aligned}\right\} \quad 0 \leqslant n \leqslant 1. \tag{39}$$

In (38) and (39)

$$\left.\begin{aligned} x_m(n) &= X_m(0) + W_8^{4n} X_m(1) \\ x_m(n+1) &= X_m(0) - W_8^{4n} X_m(1) \end{aligned}\right\} \quad \begin{aligned} n &= 0 \\ m &= 3, 4, 5, 6 \end{aligned} \tag{40}$$

and

$$X_m(k) = X(4k + r) \qquad \begin{aligned} k &= 0, 1 \\ (m, r) &= (3, 0), (4, 2), (5, 1), (6, 3) \end{aligned} \tag{41}$$

Substitution of (41) into (40) results in

$$\left.\begin{aligned} x_{m(0)} &= X(r) + W_8^0 X(4 + r) \\ x_{m(1)} &= X(r) - W_8^0 X(4 + r) \end{aligned}\right\} (m, r) = (3, 0), (4, 2), (5, 1), (6, 3) \tag{42}$$

Using (35), (38) and (42) the FFT algorithm as a signal flow graph is constructed.

From Fig. 3 it is seen that if the outputs $x(n)$ are in their natural order the inputs $X(k)$ are in a bit reversed order. If the input sequence $X(k)$ is stored in computer memory in the order then the intermediate results can be stored in the same memory locations as the original sequence since these are no longer required. This is possible because the intermediate results depend only upon the sequence immediately prior to it and not to

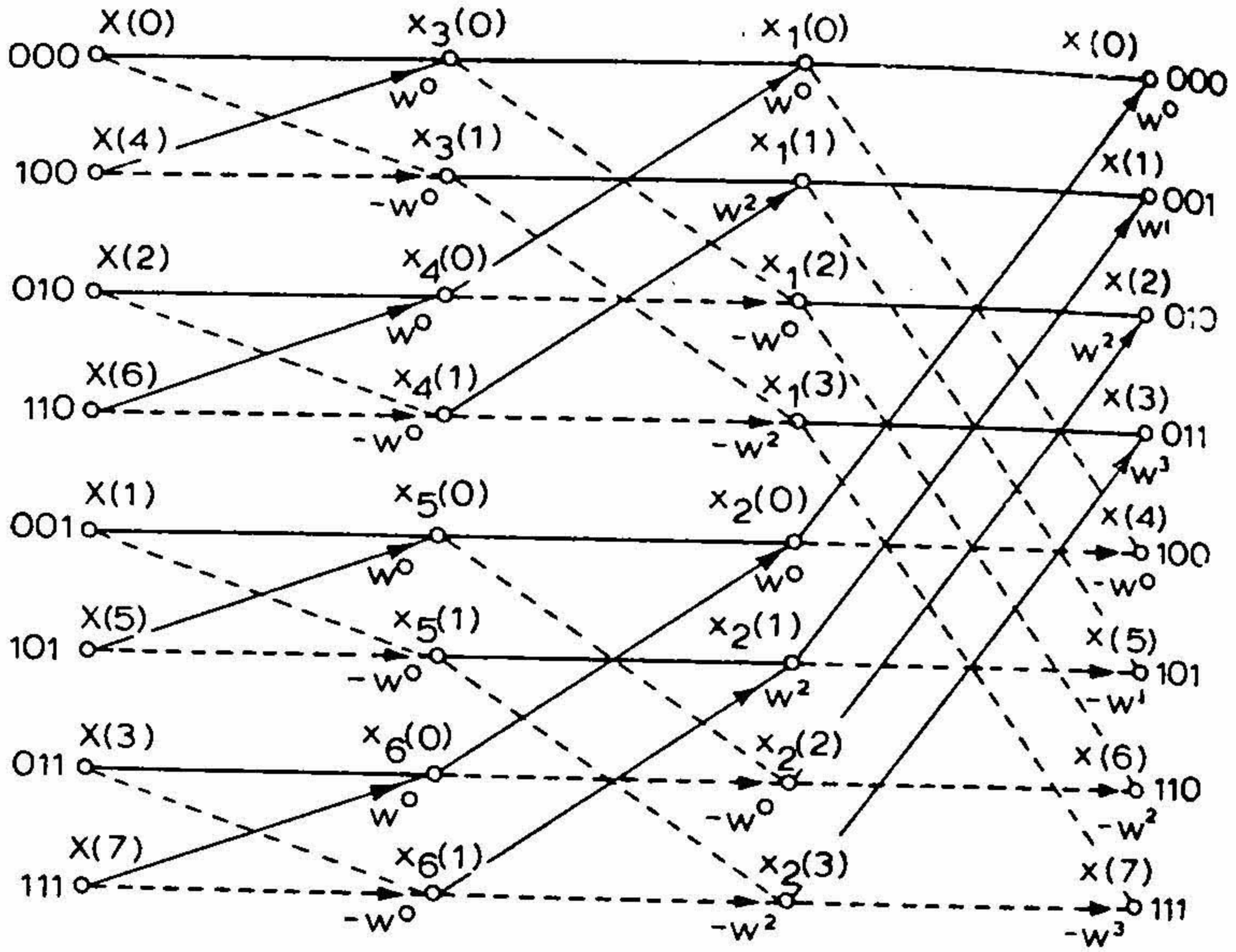

X(0)
000
X(4)
100
X(2)
010
X(6)
110
X(1)
001
X(5)
101
X(3)
011
X(7)
111
x3(0)
x3(1)
x4(0)
x4(1)
x5(0)
x5(1)
x6(0)
x6(1)
x1(0)
x1(1)
x1(2)
x1(3)
x2(0)
x2(1)
x2(2)
x2(3)
x(0)
000
x(1)
001
x(2)
010
x(3)
011
x(4)
100
x(5)
101
x(6)
110
x(7)
111
wo
-wo
w2
-w2
w1
-w1
w3
-w3

FIG. 3

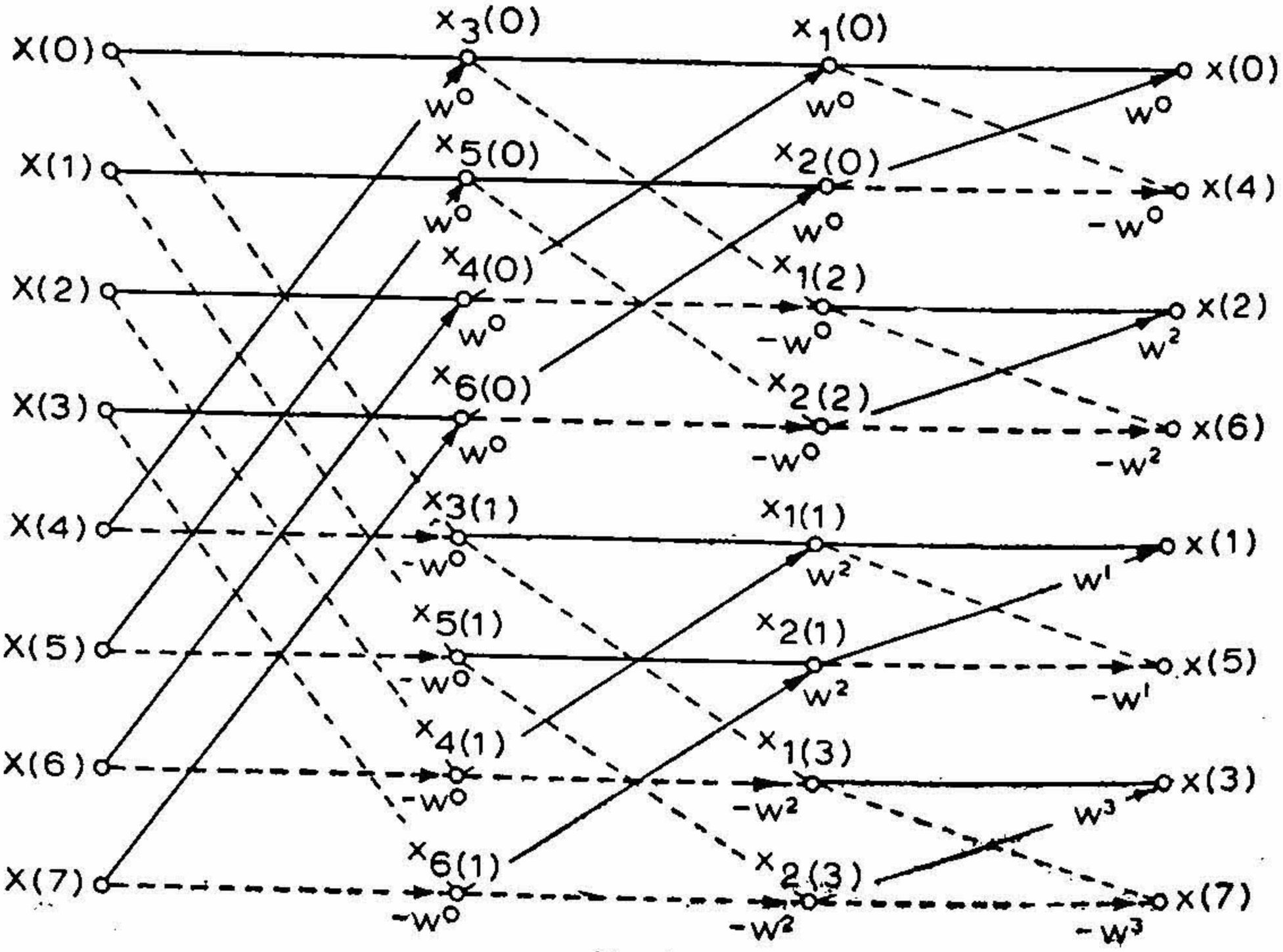

X(0)
X(1)
X(2)
X(3)
X(4)
X(5)
X(6)
X(7)
x3(0)
x5(0)
x4(0)
x6(0)
x3(1)
x5(1)
x4(1)
x6(1)
x1(0)
x2(0)
x1(2)
x2(2)
x1(1)
x2(1)
x1(3)
x2(3)
x(0)
x(4)
x(2)
x(6)
x(1)
x(5)
x(3)
x(7)
wo
-wo
w2
-w2
w1
-w1
w3
-w3

FIG. 4

any other sequences or points. The final result $x(n)$ will be in the natural order. If the input sequence $X(k)$ is stored in the computer memory in its natural order then a different variation of Fig. 3 results. In this case all the nodes at the same horizontal level as $X(4)$ are interchanged with those of $X(1)$ and all the nodes at the same horizontal level as $X(6)$ are interchanged with those of $X(3)$. In this interchanging process the arrows also go with the nodes. The resulting signal flow graph is given in Fig. 4.

5. Programming Considerations of FFT

Another derivation of the FFT algorithm is given in this section which deals in greater detail about programming and indexing and how the bit reversal comes about. In the IDFT given by

$$x(n) = \sum_{k=0}^{N-1} X(k)\, W_N^{nk} \tag{43}$$

where $N=2^m$, the indices n and k are given as functions of the m bit positions in their binary representation

$$\left.\begin{aligned} n &= n_{m-1}\,2^{m-1} + n_{m-2}\,2^{m-2} + \ldots n_1 2 + n_0 \\ k &= k_{m-1}\,2^{m-1} + k_{m-2}\,2^{m-2} + \ldots k_1 2 + k_0 \end{aligned}\right\} \tag{44}$$

where n_l, $k_l = 1$ or 0. Use of (44) in (43) yields the bit representation of $x(n)$

$$\sum_{k_0=0}^{1}\ \sum_{k_1=0}^{1}\ \cdots\ \sum_{k_{m-2}=0}^{1}\ \sum_{k_{m-1}=0}^{1} \{X(k_{m-1},\, k_{m-2}, \ldots k_1,\, k_0) \times W_N^{(n_{m-1}2^{m-1} + \ldots n_0)\,(k_{m-1}2^{m-1} + \ldots k_0)}\} \tag{45}$$

The exponent of W_N in (45) can be given by the following bilinear form

$$(n_{m-1}\,2^{m-1} + \ldots n_0)\,(k_{m-1}\,2^{m-1} + \ldots k_0)$$

$$= [n_{m-1}\, n_{m-2} \ldots n_0] \begin{bmatrix} 2^{2m-2} & 2^{2m-3} & \ldots & 2^m & 2^{m-1} \\ 2^{2m-3} & 2^{2m-4} & \ldots & 2^m & 2^{m-1} & 2^{m-2} \\ \cdot & & & \cdot & & \\ \cdot & & & \cdot & & \\ \cdot & & & \cdot & & \\ \cdot & & & \cdot & & \\ \cdot & & & \cdot & & \\ \cdot & & & \cdot & & \\ \cdot & & & \cdot & & \\ \cdot & & & \cdot & & \\ \cdot & & & \cdot & & \\ 2^m & 2^{m-1} & & 2^2 & 2 & \\ 2^{m-1} & 2^{m-2} & & 2 & 1 & \end{bmatrix} \begin{bmatrix} k_{m-1} \\ k_{m-2} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ k_1 \\ k_0 \end{bmatrix} \tag{46}$$

Since (46) is the exponent of $W_N = e^{2\pi j/N}$, and since $W_N^{2^r} = 1$ for all $r \geqslant m$, (46) effectively reduces to a lower triangular matrix

$$[n_{m-1}\, n_{m-2} \ldots n_1 n_0] \begin{bmatrix} \overline{0}\,0\,0 \ldots\ldots 0\,2^{m-1} \\ 0\,0\,0 \ldots 0\,2^{m-1}\,2^{m-2} \\ \cdot \quad\quad \cdot \\ \cdot \quad\quad \cdot \\ \cdot \quad\quad\quad \cdot \\ \cdot \quad\quad\quad \cdot \\ \cdot \\ 0\,2^{m-1} \ldots\ldots 2^2\,2 \\ \underline{2^{m-1}\,2^{m-2} \ldots 2\,1} \end{bmatrix} \begin{bmatrix} \overline{k_{m-1}} \\ k_{m-2} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ k_1 \\ \underline{k_0} \end{bmatrix} \tag{47}$$

Simplifying (47) and substituting into (45) yields

$$\sum_{k_0=0}^{1} \sum_{k_1=0}^{1} \cdots \sum_{k_{m-2}=0}^{1} \cdots \sum_{k_{m-2}=0}^{1} \sum_{k_{m-1}=0}^{1} \{X(k_{m-1}, k_{m-2}, \ldots k_1, k_0)$$
$$\times W_N^{n_0 2^{m-1} k_{m-1}} \cdot W_N^{(n_1 2^{m-1} + n_0 2^{m-2}) k_{m-2}}$$
$$\times W_N^{(n_2 2^{m-1} + n_1 2^{m-2} + n_0 2^{m-3}) k_{m-3}} \ldots$$
$$\times W_N^{(n_{l-1} 2^{m-1} + n_{l-2} 2^{m-2} + \ldots n_0 2^{m-l}) k_{m-l}} \ldots W_N^{(n_{m-1} 2^{m-1} + \ldots n_1 2 + n_0) k_0}\} \tag{48}$$

From (48) the following summations can be defined recursively.

$$\sum_{k_{m-1}=0}^{1} X(k_{m-1}, k_{m-2} \ldots k_1, k_0)\, W_N^{n_0 2^{m-1} k_{m-1}} = X_1(n_0, k_{m-2}, \ldots k_0)$$

$$\sum_{k_{m-2}=0}^{1} X_1(n_0, k_{m-2}, \ldots k_1, k_0)\, W_N^{(n_1 2^{m-1} + n_0 2^{m-2}) k_{m-2}} = X_2(n_0, n_1, k_{m-3}, \ldots k_0)$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$\sum_{k_{m-l}=0}^{1} X_{l-1}(n_0, n_1 \ldots n_{l-2}, k_{m-l}, \ldots k_0)\, W_N^{(n_{l-1} 2^{m-1} + n_{l-2} 2^{m-2} + \ldots n_0 2^{m-l}) k_{m-l}}$$
$$= X_l(n_0, n_1, \ldots n_{l-1}, k_{m-l-1} \ldots k_0)$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$\sum_{k_0=0}^{1} X_{m-1}(n_0, n_1, n_{m-2}, k_0)\, W^{(n_{m-1} 2^{m-1} + \ldots n_1 2 + n_0) k_0}$$
$$= X_m(n_0, n_1 \ldots n_{m-1}) \tag{49}$$

Substitution of (49) into (48) yields

$$x(n_{m-1}, n_{m-2} \ldots n_1, n_0) = X_m(n_0, n_1, \ldots n_{m-2}, n_{m-1}) \tag{50}$$

The storage indexing convention used is to let the m arguments (n_0, $n_1, \ldots {}_l n_{-1}, k_{m-l-1}, \ldots k_0$) of X be the binary representation of the index of its storage location. In this way each step of the algorithm given by (49) involves fetching two points from two storage locations and putting the resultant back into the same two locations. As a consequence of this indexing scheme the elements in the final step of (49), X_m are in the bit reversed order represented by (50). In order to obtain the $x(n)$'s the order of the bits have to be reversed in (50). To avoid the bit reversal an array of temporary storage locations are used and the successive arrays X_l are indexed as given below:

$$
\begin{aligned}
&X\ (k_{m-1}, k_{m-2}, \ldots k_1, k_0)\\
&X_1 (n_0, k_{m-2}, \ldots k_1, k_0)\\
&X_2 (n_1, n_0, k_{m-3}, \ldots k_1, k_0)\\
&\quad\cdot \qquad \cdot \qquad \cdot \qquad \cdot\\
&\quad\cdot \qquad \cdot \qquad \cdot \qquad \cdot\\
&\quad\cdot \qquad \cdot \qquad \cdot \qquad \cdot\\
&X_l (n_{l-1}, n_{l-2}, \ldots n_1, n_0, k_{m-l-1}, \ldots k_1, k_0)\\
&\quad\cdot \qquad \cdot \qquad \cdot \qquad \cdot\\
&\quad\cdot \qquad \cdot \qquad \cdot \qquad \cdot\\
&\quad\cdot \qquad \cdot \qquad \cdot \qquad \cdot\\
&X_m (n_{m-1}, n_{m-2}, \ldots n_1, n_0). \qquad (51)
\end{aligned}
$$

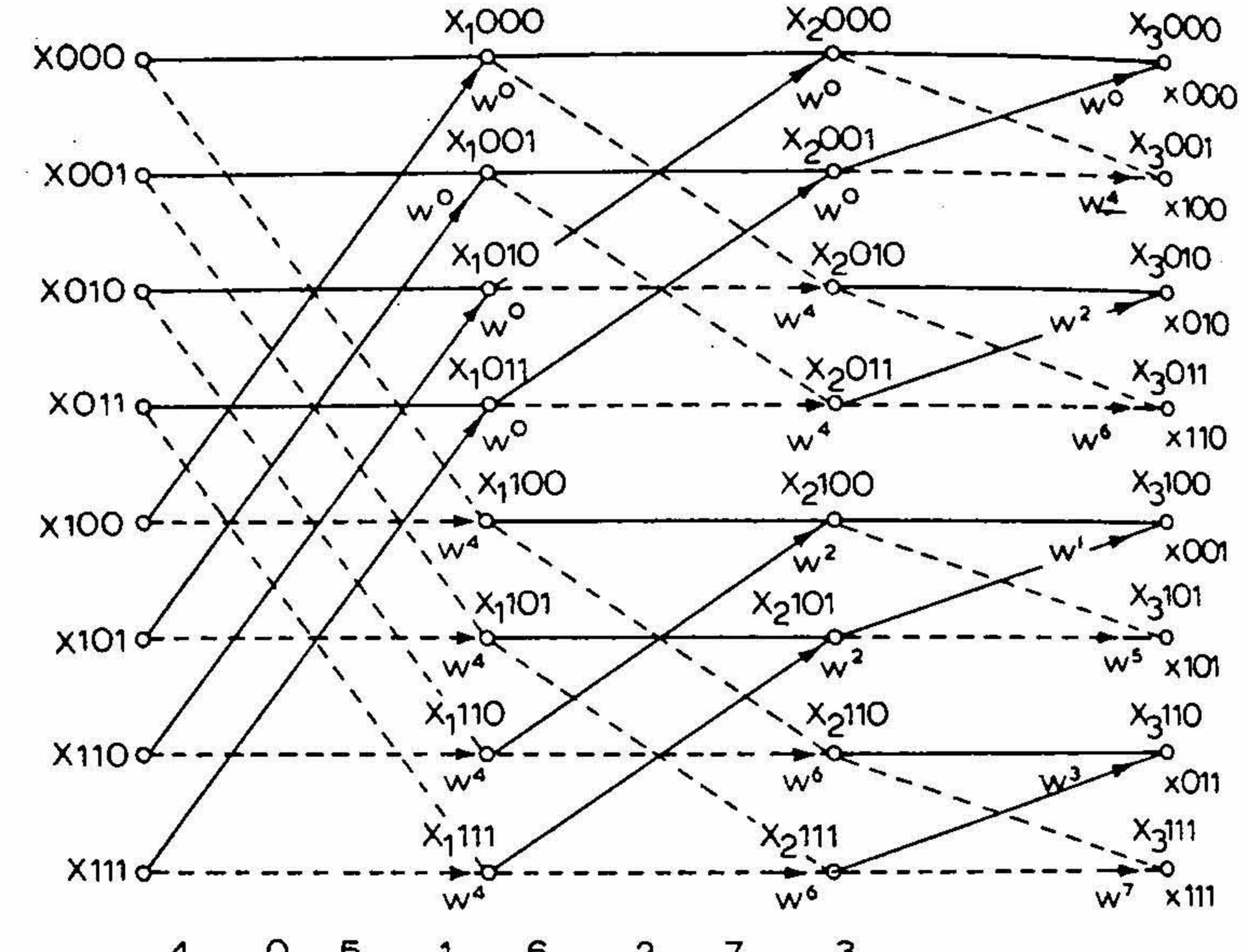
At the cost of increasing the storage locations by another N, this method has the advantage of giving the results in the natural order.

The signal flow graph of Fig. 4 can be derived from the FFT algorithm given by (49). With $N = 8$, $2^m = 8$ and $m = 3$ there will be 3 stages of reduction. These 3 stages are given below.

$$
\begin{aligned}
&\sum_{k_2=0}^{1} X(k_2, k_1, k_0)\, W_8^{n_0 2^2 k_2} = X_1 (n_0, k_1, k_0)\\
&\sum_{k_1=0}^{1} X_1 (n_0, k_1, k_0)\, W_8^{(n_1 2^2 + n_0 2) k_1} = X_2 (n_0, n_1, k_0)\\
&\sum_{k_0=0}^{1} X_2 (n_0, n_1, k_0)\, W_8^{(n_2 2^2 + n_1 2 + n_0) k_0} = X_3 (n_0, n_1, n_2) \qquad (52)
\end{aligned}
$$

This signal flow graph is slightly modified from that of Fig. 4 and is given in Fig. 5. In Fig. 5 it is to be noted that

$$
\begin{aligned}
W^4 &= -\ W^0\\
W^5 &= -\ W^1\\
W^6 &= -\ W^2\\
W^7 &= -\ W^3.
\end{aligned}
$$



Note :- $W^4 = -W^0$, $W^5 = -W^1$, $W^6 = -W^2$, $W^7 = -W^3$

FIG. 5

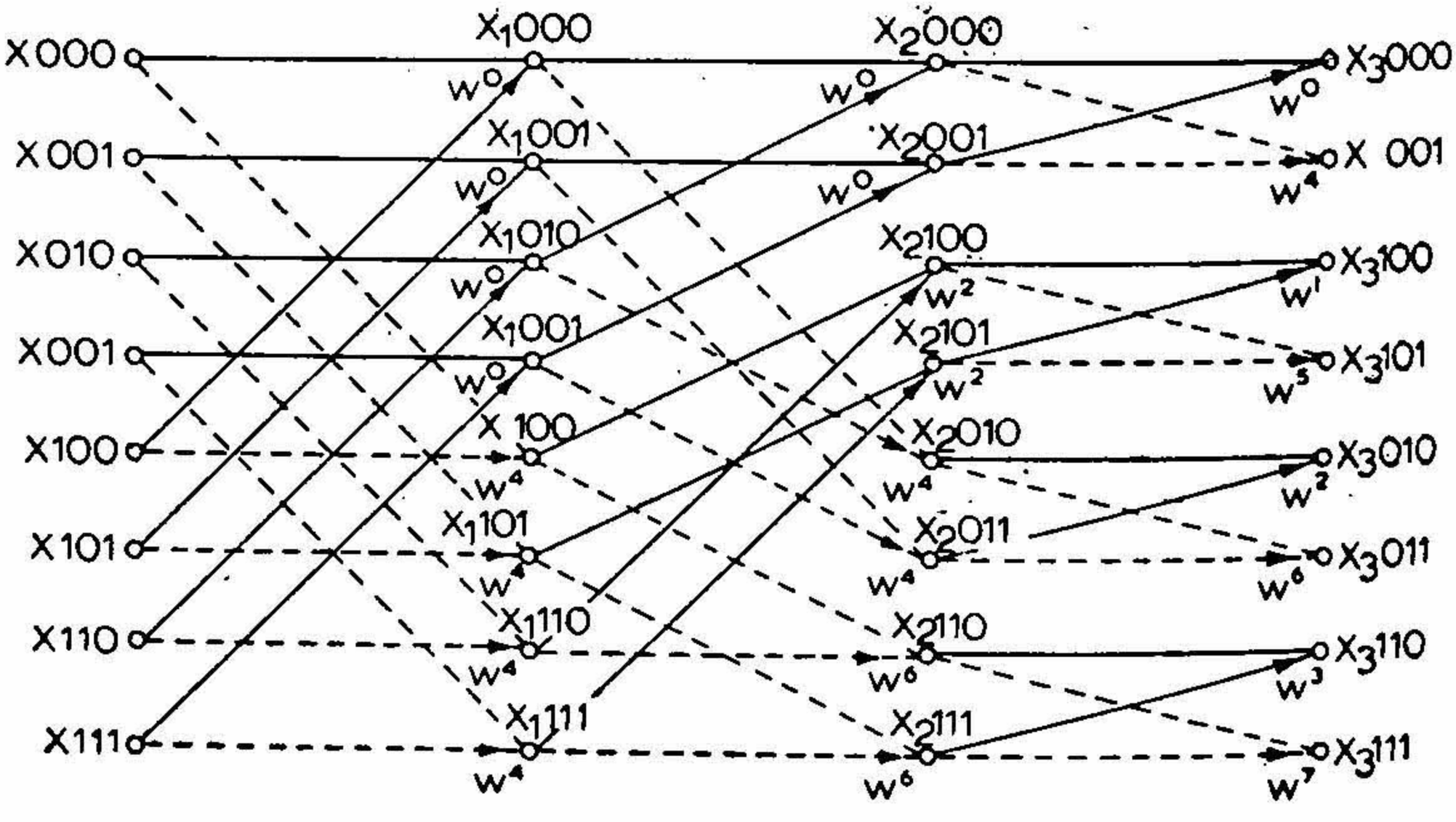


FIG. 6

In Fig. 5 the number of memory locations need not exceed the number of sample points 8. However, unshuffling the input sequence in the case of Fig. 3, and unshuffling the output sequence in the case of Figs. 4 and 5 must be done to interpret the results. This unshuffling process is accomplished by either using bit reversal tables for any N obtained by programming the computer or by doing manually. This may be a tedious process and can be avoided by using an extra *array* of memory locations as stated before, which will give both the input and the output in their natural order. The transition from the signal flow graph of Fig. 5, where the ouput sequence is in its natural order, is shown in Figs. 6 and 7. Two transitions are necessary so that the multiplying factors W_8^n's are also in their natural order. Figure 6 shows the transition where the W_8^n's in the second stage are arranged in their natural order. In Fig. 7 the W_8^n's correspond to $x(n)$ and hence

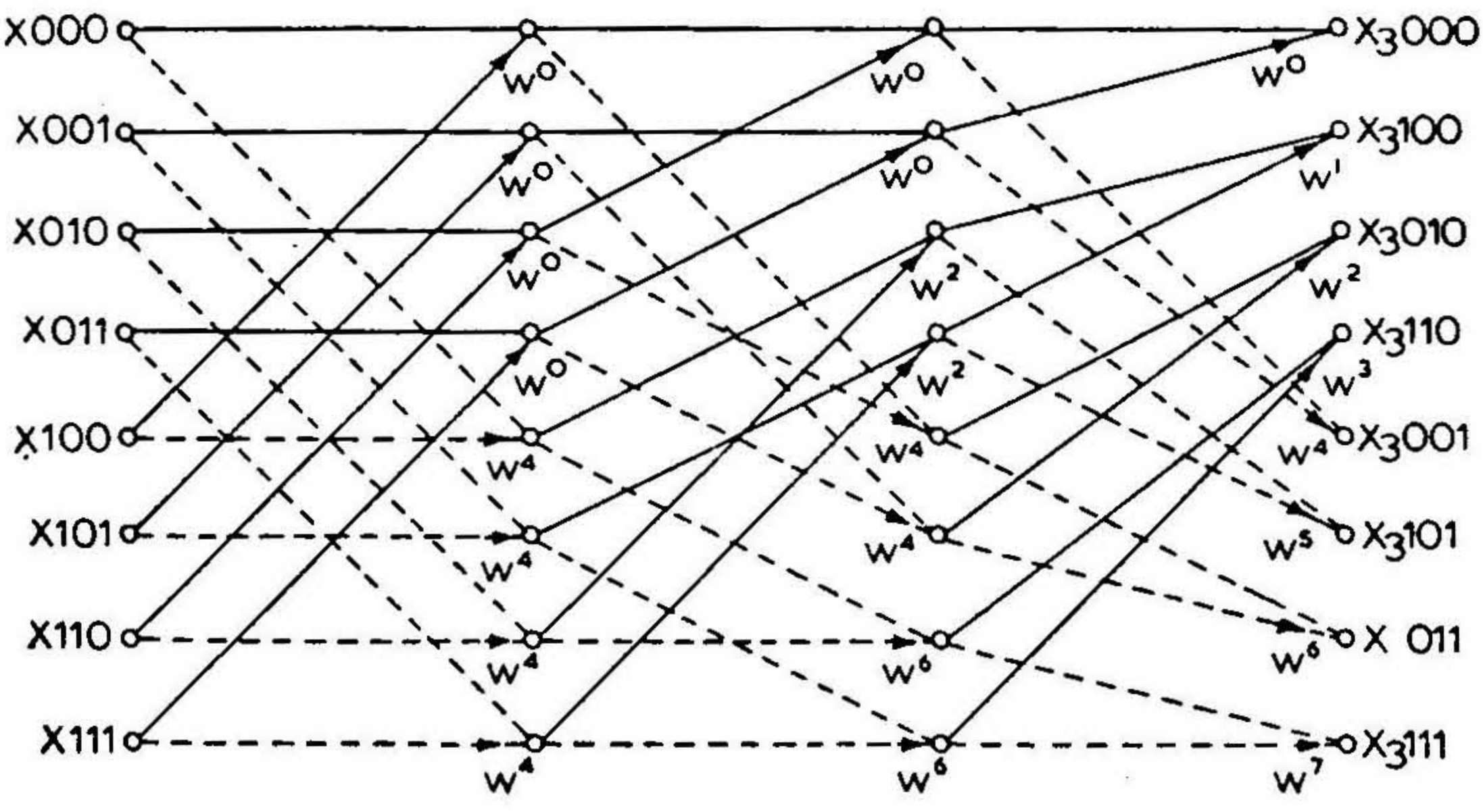


FIG. 7

$x(n)$'s are permuted to obtain the natural order. If this transition is to be effective an extra array of 8 memory locations are needed as stated before. In the case where no extra memory locations are needed there is increased computational complexity resulting in increased time whereas in the latter case where there is an extra array of memory available computational complexity is reduced. In the examples which follows extra memory locations are made use of in the IBM 360/44 system and programs are given to implement FFT algorithms. Programs are included in the appendices.

6. EXPLANATION OF THE PROGRAMS*

(*a*) *Main Program.*—The algorithm corresponding to the signal flow graph of Fig. 7 is given compactly as follows. If $N = 2^J$

$$X(R + I\,N/2^J)$$

$$= X1\left(R + I\frac{N}{2^{J-1}}\right) + W^{I\,N/2^J} \ldots X1\left(R + I\frac{N}{2^{J-1}} + \frac{N}{2^J}\right) \quad (53)$$

$$X\left(R + I\frac{N}{2^J} + \frac{N}{2}\right)$$

$$= X1\left(R + I\frac{N}{2^{J-1}}\right) - W^{I\,N/2^J} X1\left(R + I\frac{N}{2^{J-1}} + \frac{N}{2^J}\right) \quad (54)$$

(53) gives the first half of the samples and (54) the next half of the samples. In writing this algorithm use has been made of the fact $W_N^{(n+N/2)} = -W_N^n$. This algorithm as mentioned before uses an extra set, N, of memory locations, *i.e.*, a total of $2N$ memory locations. This program is called by the name SUBROUTINE FFT. This subroutine is called by the statement

CALL FFT (N, M, X, SIGN)

in the main program where

N = number of sample points = 2^M

X = Input data (IDFT or DFT)

SIGN = − 1 for DFT

+ 1 for IDFT.

The main program is given in Appendix I. The FFT subroutine is given in Appendix II.

(*b*) *Graph Plot.*—In addition to the FFT subroutine two other subroutines are employed to plot the graph by the computer. These are subroutines Graph C and Graph. The Graph C is to obtain input data to the subroutine Graph. These subroutines can be called by the statements

CALL GRAPH C (X, AR, A, N, $DELW$, $M1$)

CALL GRAPH (RR, N, $DELT$, $M1$)

where the symbols are explained below

* Acknowledge the help of Mr. K. A. Narayanan in constructing these programs.

X	Complex function to be plotted
AR, AI	Real and imaginary parts of X which are to be dimensioned in the main program
N	Number of samples
$DELT$, $DELW$	Spacing along abscissa $= T/N = \omega_s/N$
MI	Integer representing the position of Y-axis
RR	Real function to be plotted.

The program is given in Appendix III.

7. Examples

In the example the Fourier transform of a rectangular pulse is programmed. The pulse is defined by

$$x(t) = 1 \quad |t| < 1$$

$$= 0 \text{ otherwise.}$$

The period of repetition of $x_p(t)$ is chosen to be 4 seconds. To exhibit aliassing errors two different numbers of samples were taken, *viz.*, $N = 32$ and 64. Correspondingly, $\triangle t = 1/8$ and $1/16$, $\omega_s = 16\pi$ and 32π and $\triangle\omega = \pi/2$.

The sampled aliassed pulse in a period is given in Fig. 8. The figure is illustrated for the case $N = 32$. Two cases were considered for each N.

(*a*) $x_p(+)|_{t=1} = 1$; $\quad x_p(t)|_{t=3} = 0$

(*b*) $x_p(t)|_{t=1} = 0{\cdot}5$; $\quad x_p(t)|_{t=3} = 0{\cdot}5$.

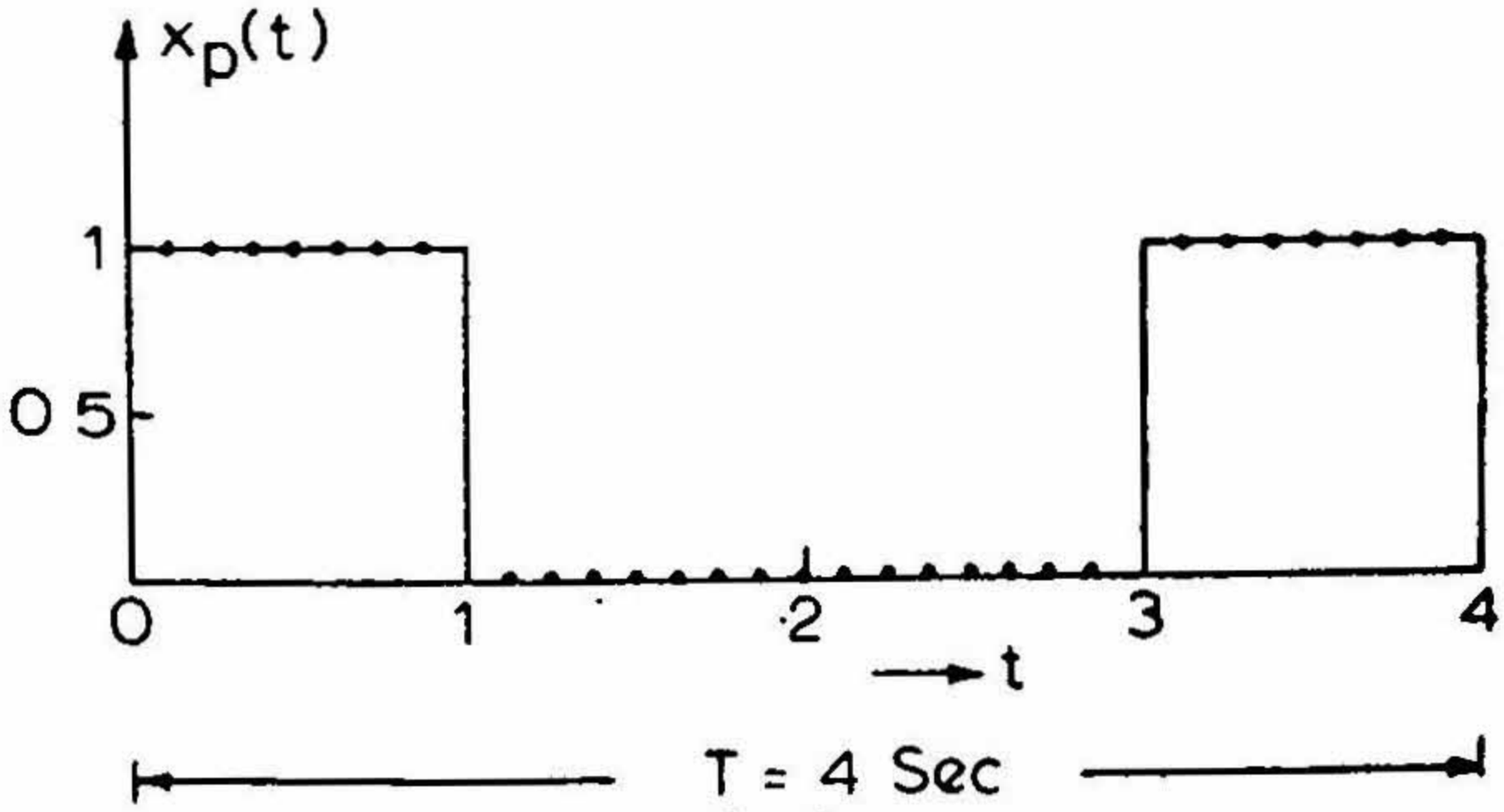
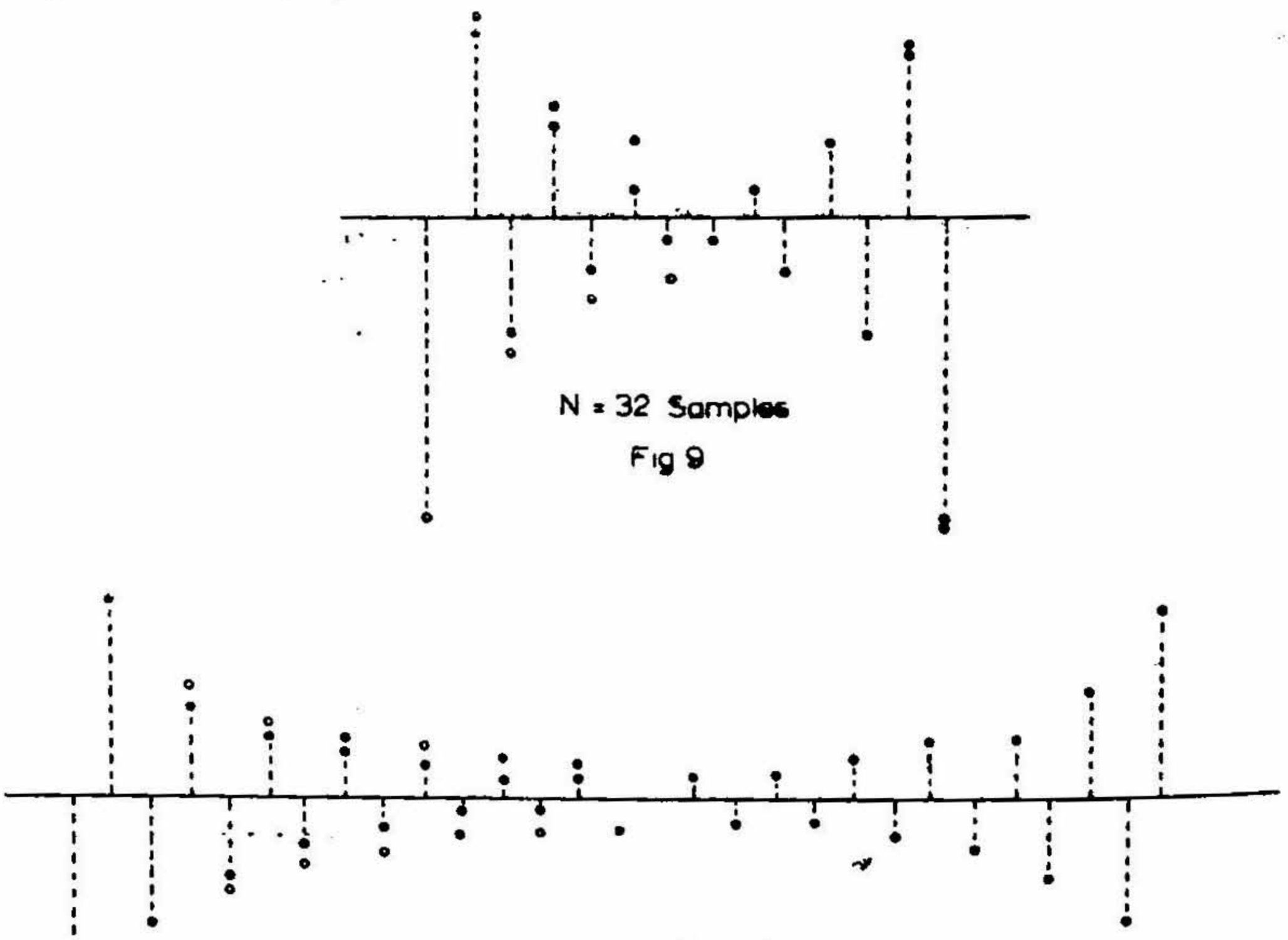


Fig. 8

Since the original pulse is an even function the sampling sequence chosen should be an even sequence. Case (*a*) above yields only a real sequence without its being even and as a result a spurious imaginary part will be obtained. On the other hand case (*b*) yields an even sequence and hence gives the imaginary part as zero. In other words the sequence in case (*a*) being real yields complex conjugate sequence as the DFT and the sequence in case (*b*) being real and even yields real and even sequence as the DFT. In either case the real part in both cases were the same. The curves plotted by the computer are appended. In these curves the actual continuous Fourier transform at the discrete values of the DFT are plotted alongside to illustrate the amount of aliassing error involved. It can be very clearly seen that these errors are much lower when $N = 64$ (Figs. 9, 10).



N = 32 Samples

Fig 9

N = 64 Samples

• - Calculated values

• - Computed values

Fig 10

Conclusions

The FFT is a very efficient method of computing the DFT of a complex time series. There are drawbacks because of the errors and spurious responses, but by careful analyses these can be minimised to the desired degree of accuracy.

Acknowledgements

I am grateful to K. A. Narayanan, graduate student of Elec. Com. Engg., for the FFT programmes given in the Appendices. I am also grateful to Dr. V. G. Tikekar of the Dept. of Applied Mathematics for helpful discussions.

References

[1] Cooley, J. W. and Tukey, J. W. — An algorithm for the machine calculation of complex Fourier series. *Math. of Comput.*, April 1965.

[2] Cooley, J. W., Lewis, P. A. W. and Welch, P. D. — The fast Fourier transform and its applications. *IEEE Trans. Education*, March 1969.

[3] Cooley, J. W., Lewis, P. A. W. and Welch. P. D. — Applications of the fast fourier transform to computation of Fourier integrals, Fourier series and convolution integrals. *IEET Trans. Audio and Electro. acoustics*, June 1967.

[4] Cooley, J. W., Lewis, P. A. W. and Welch, P. D. — The fast Fourier transform algorithm. *Jour. Sound Vib.*, 1970.

[5] Cochran, W. T. *et al.* — What is the fast Fourier transform?. *Proc. IEEE*, October 1967.

[6] Cooley, J. W., Lewis, P. A. W. and Welch, P. D. — The fast Fourier transform algorithm and its applications. *IBM Research Report RC* 1743, February 1967.

[7] Bergland, G. D. .. A guided tour of the fast Fourier transform. *IEEE Spectrum*, July 1969.