What is the Fast Fourier Transform?

G-AE Subcommittee on Measurement Concepts

WILLIAM T. COCHRAN
JAMES W. COOLEY
DAVID L. FAVIN, MEMBER, IEEE
HOWARD D. HELMS, MEMBER, IEEE
REGINALD A. KAENEL, SENIOR MEMBER, IEEE
WILLIAM W. LANG, SENIOR MEMBER, IEEE
GEORGE C. MALING, JR., ASSOCIATE MEMBER, IEEE
DAVID E. NELSON, MEMBER, IEEE,
CHARLES M. RADER, MEMBER, IEEE
PETER D. WELCH

Abstract—**The fast Fourier transform is a computational tool which facilitates signal analysis such as power spectrum analysis and filter simulation by means of digital computers. It is a method for efficiently computing the discrete Fourier transform of a series of data samples (referred to as a time series). In this paper, the discrete Fourier transform of a time series is defined, some of its properties are discussed, the associated fast method (fast Fourier transform) for computing this transform is derived, and some of the computational aspects of the method are presented. Examples are included to demonstrate the concepts involved.**

Introduction

An ALGORITHM for the computation of Fourier coefficients which requires much less computational effort than was required in the past was reported by Cooley and Tukey [1] in 1965. This method is now widely known as the "fast Fourier transform," and has produced major changes in computational techniques used in digital spectral analysis, filter simulation, and related fields. The technique has a long and interesting history that has been summarized by Cooley, Lewis, and Welch in this issue [2].

The fast Fourier transform (FFT) is a method for efficiently computing the discrete Fourier transform (DFT) of a time series (discrete data samples). The efficiency of this method is such that solutions to many problems can now be obtained substantially more economically than in the past. This is the reason for the very great current interest in this technique.

The discrete Fourier transform (DFT) is a transform in its own right such as the Fourier integral transform or the Fourier series transform. It is a powerful reversible mapping operation for time series. As the name implies, it has mathematical properties that are entirely analogous to those of the Fourier integral transform. In particular, it defines a spectrum of a time series; multiplication of the transform of two time series corresponds to convolving the time series.

If digital analysis techniques are to be used for analyzing a continuous waveform then it is necessary that the data be sampled (usually at equally spaced intervals of time) in order to produce a time series of discrete samples which can be fed into a digital computer. As is well known [6], such a time series completely represents the continuous waveform, provided this waveform is frequency band-limited and the samples are taken at a rate that is at least twice the highest frequency present in the waveform. When these samples are equally spaced they are known as Nyquist samples. It will be shown that the DFT of such a time series is closely related to the Fourier transform of the continuous waveform from which samples have been taken to form the time series. This makes the DFT particularly useful for power spectrum analysis and filter simulation on digital computers.

The fast Fourier transform (FFT), then, is a highly efficient procedure for computing the DFT of a time series. It takes advantage of the fact that the calculation of the coefficients of the DFT can be carried out iteratively, which results in a considerable savings of computation time. This manipulation is not intuitively obvious, perhaps explaining why this approach was overlooked for such a long time. Specifically, if the time series consists of $N=2^n$ samples, then about $2nN=2N\cdot\log_2 N$ arithmetic operations will be shown to be required to evaluate all N associated DFT coefficients. In comparison with the number of operations required for the calculation of the DFT coefficients with straightforward procedures (N^2), this number is so small when N is large as to completely change the computationally economical approach to various problems. For example, it has been reported that for $N=8192$ samples, the computations require about five seconds

Manuscript received March 10, 1967.

W. T. Cochran, D. L. Favin, and R. A. Kaenel are with Bell Telephone Laboratories, Inc., Murray Hill, N. J.

H. D. Helms is with Bell Telephone Laboratories, Inc., Whippany, N. J.

J. W. Cooley and P. D. Welch are with the IBM Research Center, Yorktown Heights, N. Y.

W. W. Lang and G. C. Maling are with the IBM Corporation, Poughkeepsie, N. Y.

C. M. Rader is with Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Mass. (Operated with support from the U. S. Air Force.)

D. E. Nelson is with the Electronics Division of the General Dynamics Corporation, Rochester, N. Y.

for the evaluation of all 8192 DFT coefficients on an IBM 7094 computer. Conventional procedures take on the order of half an hour.

The known applications where a substantial reduction in computation time has been achieved include: 1) computation of the power spectra and autocorrelation functions of sampled data [4]; 2) simulation of filters [5]; 3) pattern recognition by using a two-dimensional form of the DFT; 4) computation of bispectra, cross-covariance functions, cepstra and related functions; and 5) decomposing of convolved functions.

The Discrete Fourier Transform (DFT)

Definition of the DFT and its Inverse

Since the FFT is an efficient method for computing the DFT it is appropriate to begin by discussing the DFT and some of the properties that make it so useful a transformation. The DFT is defined by[1]

$$A_r = \sum_{k=0}^{N-1} X_k \exp(-2\pi jrk/N) \qquad r = 0, \cdots, N-1 \quad (1)$$

where A_r is the rth coefficient of the DFT and X_k denotes the kth sample of the time series which consists of N samples and $j=\sqrt{-1}$. The X_k's can be complex numbers and the A_r's are almost always complex. For notational convenience (1) is often written as

$$A_r = \sum_{k=0}^{N-1} (X_k) W^{rk} \qquad r = 0, \cdots, N-1 \quad (2)$$

where

$$W = \exp(-2\pi j/N). \quad (3)$$

Since the X_k's are often values of a function at discrete time points, the index r is sometimes called the "frequency" of the DFT. The DFT has also been called the "discrete Fourier transform" or the "discrete time, finite range Fourier transform."

There exists the usual inverse of the DFT and, because the form is very similar to that of the DFT, the FFT may be used to compute it.

The inverse of (2) is

$$X_l = (1/N) \sum_{r=0}^{N-1} A_r W^{-rl} \quad l = 0, 1, \cdots, N-1. \quad (4)$$

This relationship is called the inverse discrete Fourier transform (IDFT). It is easy to show that this inversion is valid by inserting (2) into (4)

$$X_l = \sum_{r=0}^{N-1} \sum_{k=0}^{N-1} (X_k/N) W^{r(k-l)}. \quad (5)$$

Interchanging in (5) the order of summing over the indices r and k, and using the orthogonality relation

$$\sum_{r=0}^{N-1} \exp(2\pi j(n-m)r/N) = N, \text{ if } n \equiv m \bmod N$$
$$= 0, \text{ otherwise} \quad (6)$$

establishes that the right side of (5) is in fact equal to X_k.

It is useful to extend the range of definition of A_r to all integers (positive and negative). Within this definition it follows that

$$A_r = A_{N+r} = A_{2N+r} = \cdots \quad (7)$$

Similarly,

$$X_l = X_{N+l} = N_{2N+l} = \cdots. \quad (8)$$

Relationships between the DFT and the Fourier Transform of a Continuous Waveform

An important property that makes the DFT so eminently useful is the relationship between the DFT of a sequence of Nyquist samples and the Fourier transform of a continuous waveform, that is represented by the Nyquist samples. To recognize this relationship, consider a frequency band-limited waveform $g(t)$ whose Nyquist samples, X_k, vanish outside the time interval $0 \leq t \leq NT$

$$g(t) = \sum_{k=0}^{N-1} \frac{\sin(\pi(t-kT)/T)}{(\pi(t-kT)/T)} \cdot X_k \quad (9)$$

where T is the time spacing between the samples. A periodic repetition of $g(t)$ can be constructed that has identically the same Nyquist samples in the time interval $0 \leq t \leq NT$

$$g_p(t) = \sum_l \sum_{k=0}^{N-1} X_k \cdot \frac{\sin(\pi(t-kT-lNT)/T)}{(\pi(t-kT-lNT)/T)}. \quad (10)$$

Let the Fourier transform of $g(t)$ be $G(f)$. As is well known [6], this transform is exactly specified at discrete frequencies by the complex Fourier series coefficients of $g_p(t)$. From this it follows:

$$\frac{G(n/NT)}{NT} = D_n$$
$$= (1/NT) \int_0^{NT} g_p(t) \cdot \exp(-2\pi jnt/NT) \cdot dt$$
$$= (1/NT) \sum_{k=0}^{N-1} X_k \cdot \exp(-2\pi jnkT/NT) \quad (11)$$

where $|n| \leq N/2$ due to the spectral bandwidth limitation implicitly assumed by the sampling theorem underlying the validity of Nyquist samples.

Comparing (11) and (1) it is seen that they are exactly the same except for a factor of NT and (r, n) are both unbounded. That is,

$$N \cdot A_r = D_n \text{ for } r = n \text{ and } T = 1 \text{ second.} \quad (12)$$

The bounds specified for r and n require a correspondence which depends on (7)

[1] The definition of the DFT is not uniform in the literature. Some authors use A_r/N as the DFT coefficients, others use $A_r/\sqrt{N}$, still others use a positive exponent.

$$\frac{G(n/NT)}{NT} = D_n = N \cdot A_r$$

where

$$n = r \quad \text{for } n = 0, 1, \cdots, q < N/2,$$

and

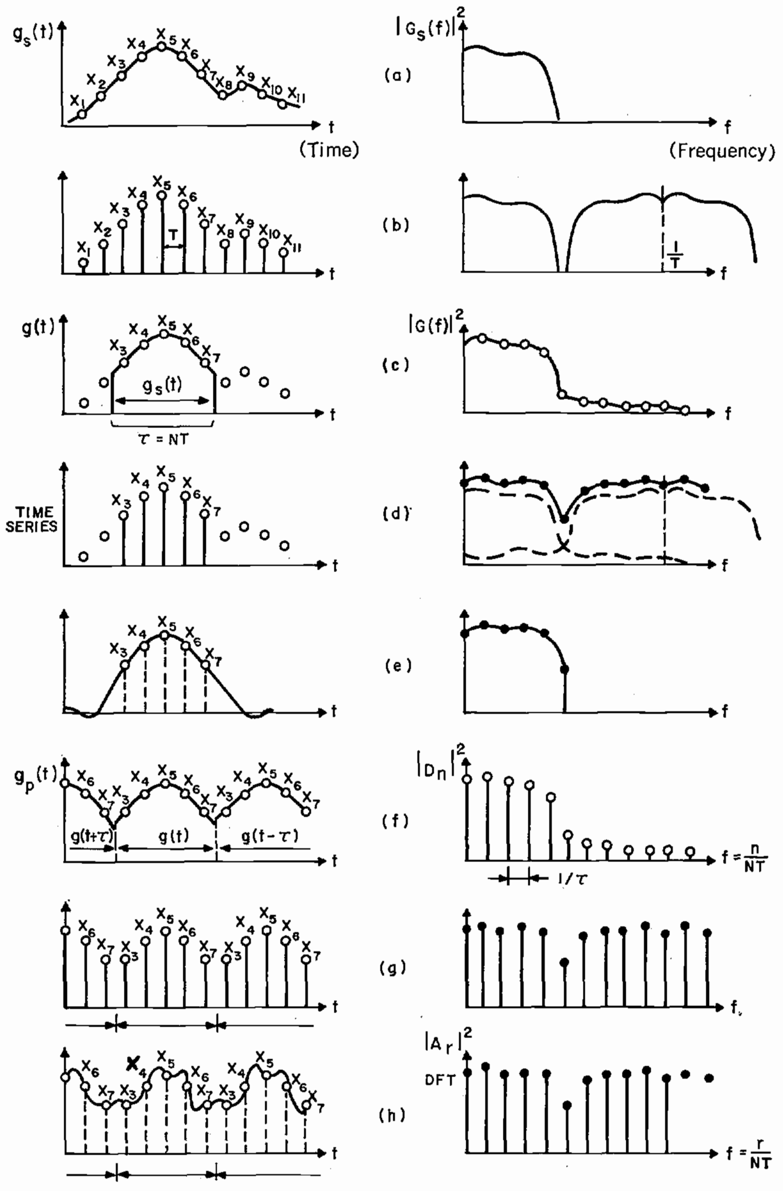
$$n = N - r \quad \text{for } n = -1, -2, \cdots, -q > -N/2 \quad (13)$$

and

$$\frac{G(n/NT)}{NT} = D_n = N \cdot A_r/2 \quad \text{for } n = N/2. \quad (14)$$

Equations (13) and (14) give a direct relationship between the DFT coefficients and the Fourier transform at discrete frequencies for the waveform stipulated by (9). A one-to-one correspondence could have been obtained if the running variable r had been bounded by $\pm N/2$. This, however, would have required distinguishing between even and odd values of N, a distinction avoided by keeping r positive.

A waveform of the type considered by (9) is shown in Fig. 1(e). It is usually obtained as an approximation of a frequency band-limited source waveform [such as the one sketched in Fig. 1(a)] by truncating the Nyquist sample series of this waveform, and reconstructing the continuous waveform corresponding to the truncated Nyquist sample series [Fig. 1(b), (d), and (e)]. Notwithstanding the identity of the Nyquist samples of this reconstructed waveform and the frequency band-limited source waveform, these waveforms differ in the truncation interval [Fig. 1(c) and (e)]. The difference is usually referred to as aliasing distortion; the mechanics of this distortion is most apparent in the frequency domain [Fig. 1(c)–(e)]. It can be made negligibly small by choosing a sufficiently large product of the frequency bandwidth of the source waveform and the duration of the truncation interval [6] (e.g., N is greater than ten).



(a) Frequency-band-limited source waveform.
(b) Nyquist samples of the frequency band-limited source waveform.
(c) Truncated source waveform.
(d) Truncated series of Nyquist samples of the source waveform.
(e) Frequency-band-limited waveform whose Nyquist samples are identical to the truncated series of Nyquist samples of the source waveform.
(f) Periodic continuation of the truncated source waveform.
(g) Periodic continuation of the truncated series of Nyquist samples of the source waveform.
(h) DFT coefficients interpreted as Fourier series coefficients producing complex waveform.

Fig. 1. Related waveforms and their corresponding spectra as defined by the Fourier transforms (integral transforms for energy-limited waveforms; series transform for periodic waveforms).

These aliasing distortions are carried over directly to the discrete spectra of the periodically repeated waveforms [Fig. 1(f) and (g)], and appear correspondingly in the DFT of the truncated series of Nyquist samples [Fig. 1(h)]. It may be of interest to observe that the waveform corresponding to the DFT coefficients interpreted as Fourier series coefficients is complex [Fig. 1(h)].

Some Useful Properties of the DFT

Another property that makes the DFT eminently useful is the convolution relationship. That is, the IDFT of the product of two DFTs is the periodic mean convolution of the two time series of the DFTs. This relationship proves very useful when computing the filter output as a result of an input waveform; it becomes especially effective when computed by the FFT. A derivation of this property is given in Appendix A.

Other properties of the DFT are in agreement with the corresponding properties of the Fourier integral transform, perhaps with slight modifications. For example, the DFT of a time series circularly shifted by h is the DFT of the time series multiplied by W^{-rh}. Furthermore, the DFT of the sum of two functions is the sum of the DFT of the two functions. These properties are readily derived using the definition of the DFT. These and other properties have been compiled by Gentleman and Sande [7].

The Fast Fourier Transform

General Description of the FFT

As mentioned in the Introduction, the FFT is an algorithm that makes possible the computation of the DFT of a time series more rapidly than do other algorithms available. The possibility of computing the DFT by such a fast algorithm makes the DFT technique important. A comparison of the computational savings that may be achieved through use of the FFT is summarized in Table I for various computations that are frequently performed. It is important to add that the computational efforts listed represent comparable upper bounds; the actual efforts depend on the number N and the programming ingenuity applied [7].

It may be useful to point out that the FFT not only reduces the computation time; it also substantially reduces round-off errors associated with these computations. In fact, both computation time and round-off error essentially are reduced by a factor of $(\log_2 N)/N$ where N is the number of data samples in the time series. For example, if $N=1024=2^{10}$, then $N\cdot\log_2 N=10\,240$ [7], [9]. Conventional methods for computing (1) for $N=1024$ would require an effort proportional to $N^2=1\,048\,576$, more than 50 times that required with the FFT.

The FFT is a clever computational technique of sequentially combining progressively larger weighted sums of data samples so as to produce the DFT coefficients as defined by (2). The technique can be interpreted in terms of combining the DFTs of the individual data samples such that the occurrence times of these samples are taken into account sequentially and applied to the DFTs of progressively larger mutually exclusive subgroups of data samples, which are combined to ultimately produce the DFT of the complete series of data samples. The explanation of the FFT algorithm adopted in this paper is believed to be particularly descriptive for programming purposes.

TABLE I
Comparison of the Number of Multiplications Required Using "Direct" and FFT Methods

Operation	Formula	Approximate Number of Multiplications (upper comparable bounds)	
		Direct	FFT
Discrete Fourier Transform (DFT)	$\sum_{k=0}^{N-1} X_k e^{-2\pi jrk/N} \quad r = 1, 2, \cdots, N-1$	N^2	$2N \log_2 N$
Filtering (Convolution)	$\sum_{k=0}^{N-1} X_k Y_{u-k} \quad u = 0, 1, \cdots, N-1$	N^2	$3N \log_2 N$
Autocorrelation Functions	$\sum_{k=0}^{N-1-r} X_k X_{r+k} \quad r = 0, 1, \cdots, N-1$	$\frac{N}{4}\left(\frac{N}{2}+3\right)$	$3N \log_2 N$
Two-Dimensional Fourier Transform (Pattern Analysis)	$\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} X_{k,l} e^{-2\pi j(kq+r/N)} \quad r, q = 0, 1, \cdots, N-1$	N^4	$4N^2 \log_2 N$
Two-Dimensional Filtering	$\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} X_{k,l} Y_{q-k,r-l} \quad q, r = 1, 2, \cdots, N-1$	N^4	$3N^2 \log_2 N$

Conventional Forms of the FFT

Decimation in Time: The DFT [as per (2)] and its inverse [see (4)] are of the same form so that a procedure, machine, or sub-routine capable of computing one can be used for computing the other by simply exchanging the roles of X_k and A_r, and making appropriate scale-factor and sign changes. The two basic forms of the FFT, each with its several modifications, are therefore equivalent. However, it is worth distinguishing between them and discussing them separately. Let us first consider the form used by Cooley and Tukey [1] which shall be called *decimation in time.* Reversing the roles of A_r and X_k gives the form called *decimation in frequency*, which will be considered afterwards.

Suppose a time series having N samples [such as X_k shown in Fig. 2(a)] is divided into two functions, Y_k and Z_k, each of which has only half as many points ($N/2$). The function Y_k is composed of the even-numbered points ($X_0, X_2, X_4 \cdots$), and Z_k is composed of the odd numbered points ($X_1, X_3, X_5 \cdots$). These functions are shown in Fig. 2(b) and (c), and we may write them formally as

$$\begin{aligned} Y_k &= X_{2k} \\ & \qquad k = 0, 1, 2, \cdots, \frac{N}{2} - 1. \qquad (15) \\ Z_k &= X_{2k+1} \end{aligned}$$

Since Y_k and Z_k are sequences of $N/2$ points each, they have discrete Fourier transforms defined by

$$\begin{aligned} B_r &= \sum_{k=0}^{(N/2)-1} Y_k \exp(-4\pi jrk/N) \\ & \qquad r = 0, 1, 2, \cdots, \frac{N}{2} - 1. \qquad (16) \\ C_r &= \sum_{k=0}^{(N/2)-1} Z_k \exp(-4\pi jrk/N) \end{aligned}$$

The discrete Fourier transform that we want is A_r, which we can write in terms of the odd- and even-numbered points

$$\begin{aligned} A_r = \sum_{k=0}^{(N/2)-1} \Big\{ & Y_k \exp(-4\pi jrk/N) \\ & + Z_k \exp\left(-\frac{2\pi jr}{N}[2k+1]\right)\Big\} \\ & \qquad r = 0, 1, 2, \cdots, N-1 \qquad (17) \end{aligned}$$

or

$$\begin{aligned} A_r = & \sum_{k=0}^{(N/2)-1} Y_k \exp(-4\pi jrk/N) \\ & + \exp(-2\pi jr/N) \sum_{k=0}^{(N/2)-1} Z_k \exp(-4\pi jrk/N) \qquad (18) \end{aligned}$$

which, using (16), may be written in the following form:

$$A_r = B_r + \exp(-2\pi jr/N) C_r \quad 0 \le r < N/2. \qquad (19)$$

For values of r greater than $N/2$, the discrete Fourier transforms B_r and C_r repeat periodically the values taken on when $r < N/2$. Therefore, substituting $r + N/2$ for r in (19), we obtain

$$\begin{aligned} A_{r+N/2} &= B_r + \exp\left(-2\pi j\left[r + \frac{N}{2}\right]/N\right) C_r \\ & \qquad 0 \le r < N/2 \\ &= B_r - \exp(-2\pi jr/N) C_r \qquad 0 \le r < N/2. \qquad (20) \end{aligned}$$

By using (3), (19) and (20) may be written as

$$\begin{aligned} A_r &= B_r + W^r C_r \qquad 0 \le r < N/2 \qquad (21) \\ A_{r+N/2} &= B_r - W^r C_r \qquad 0 \le r < N/2. \qquad (22) \end{aligned}$$

From (21) and (22), the first $N/2$ and last $N/2$ points of the discrete Fourier transform of X_k (a sequence having N samples) can be simply obtained from the DFT of Y_k and Z_k, both sequences of $N/2$ samples.

Assuming that we have a method which computes discrete Fourier transforms in a time proportional to the square of the number of samples, we can use this algorithm to compute the transforms of Y_k and Z_k, requiring a time proportional to $2(N/2)^2$, and use (21) and (22) to find A_r with additional N operations. This is illustrated in the signal flow graph of Fig. 3. The points on the left are the values of X_k (i.e., Y_k and Z_k), and the points on the right are the points of the discrete Fourier transform, A_r. For simplicity, Fig. 3 is drawn for the case where X_k is an eight-point function, and advantage is taken of the fact that $W^n = -W^{n-N/2}$, as per (3).

However, since Y_k and Z_k are to be transformed, and since we have shown that the computation of the DFT of N samples can be reduced to computing the DFTs of two sequences of $N/2$ samples each, the computation of B_k (or C_k) can be reduced to the computation of sequences of $N/4$ samples. These reductions can be carried out as long as each function has a number of samples that is divisible by 2. Thus, if $N = 2^n$ we can make n such reductions, applying (15), (21), and (22) first for N, then for $N/2, \cdots$, and finally for a two-point function. The discrete Fourier transform of a one-point function is, of course, the sample itself. The successive reduction of an eight-point discrete Fourier transform, begun in Fig. 3, is continued in Figs. 4 and 5. In Fig. 5 the operation has been completely reduced to complex multiplications and additions. From the signal flow graph there are 8 by 3 terminal nodes and 2 by 8 by 3 arrows, corresponding to 24 additions and 48 multiplications. Half of the multiplications can be omitted since the transmission indicated by the arrow is unity. Half of the remaining multiplications are also easily eliminated, as we shall see below. Thus, in general, $N \cdot \log_2 N$ complex additions and, at most, $\frac{1}{2}N \cdot \log_2 N$ complex multi-

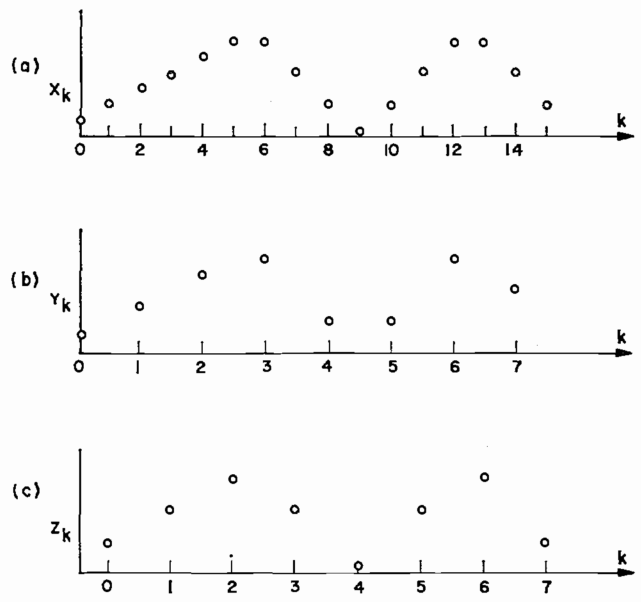


Fig. 2. Decomposition of a time series into two part-time series, each of which consists of half the samples.

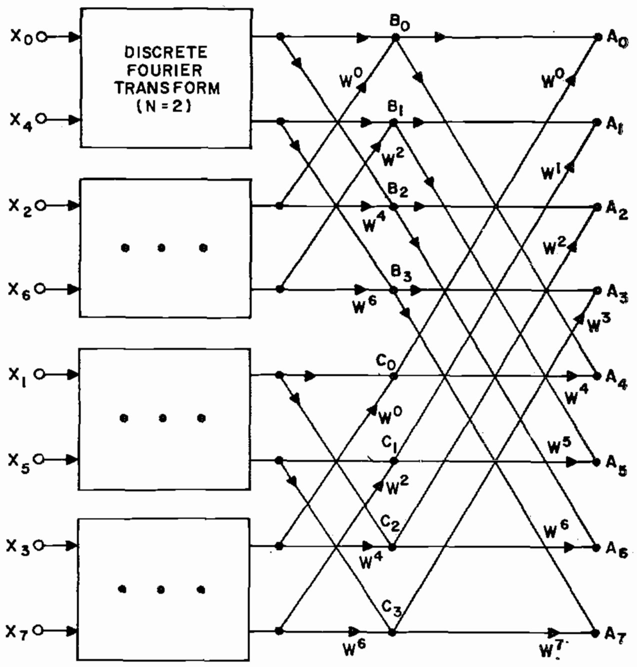

DISCRETE FOURIER TRANSFORM (N=2)
B_0 B_1 B_2 B_3 C_0 C_1 C_2 C_3
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7

Fig. 4. Signal flow graph illustrating further reduction of the DFT computation suggested by Fig. 3.

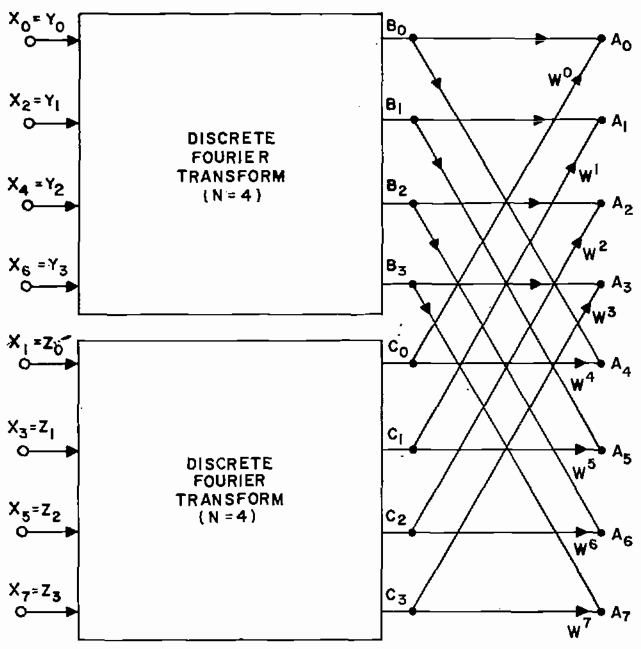

DISCRETE FOURIER TRANSFORM (N=4)
DISCRETE FOURIER TRANSFORM (N=4)

Fig. 3. Signal flow graph illustrating the reduction of endpoint DFT to two DFTs of $N/2$ points each, using decimation in time. The signal flow graph may be unfamiliar to some readers. Basically it is composed of dots (or nodes) and arrows (transmissions). Each node represents a variable, and the arrows terminating at that node originate at the nodes whose variables contribute to the value of the variable at that node. The contributions are additive, and the weight of each contribution, if other than unity, is indicated by the constant written close to the arrowhead of the transmission. Thus, in this example, the quantity A_7 at the bottom right node is equal to $B_3+W_7\times C_3$. Operations other than addition and constant multiplication must be clearly indicated by symbols other than · or ⟶.

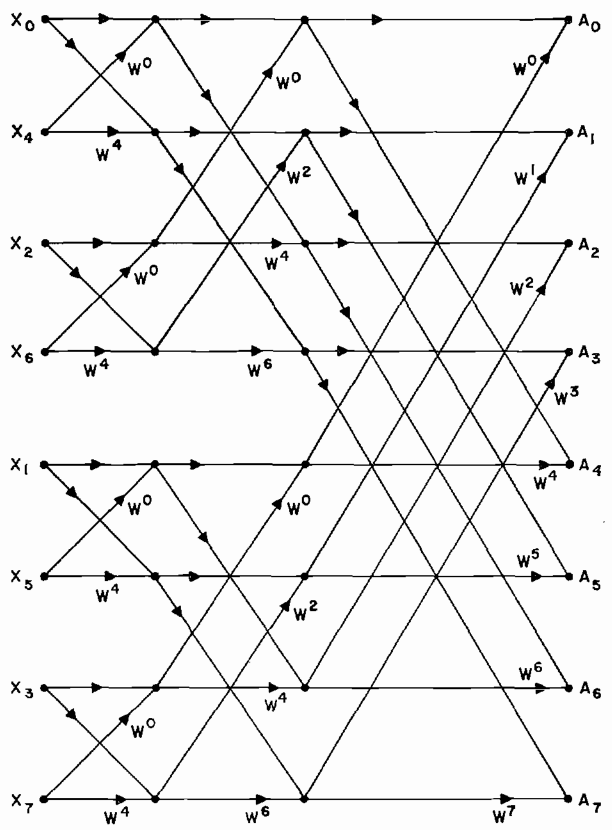


Fig. 5. Signal flow graph illustrating the computation of the DFT when the operations involved are completely reduced to multiplications and additions

plications are required for computation of the discrete Fourier transform of an N point sequence, where N is a power of 2.

When N is not a power of 2, but has a factor p, the development of equations analogous to (15) through (22) is possible by forming p different sequences, $Y_k^{(i)} = X_{pk+i}$, each having N/p samples. Each of these sequences has a DFT $B_r^{(i)}$, and the DFT of the sequence X_k can be computed from the p simpler DFTs with pN complex multiplications and additions. That is,

$$A_{r+m(N/p)} = \sum_{i=0}^{p-1} B_r^{(i)} W^{i[r+m(N/p)]} \qquad m = 0, 1, 2, \cdots, p-1 \qquad r = 0, 1, 2, \cdots, \frac{N}{p} - 1. \tag{23}$$

The computation of the DFTs can be further simplified if N has additional prime factors.

Further information about the fast Fourier transform can be extracted from Fig. 5. For example, if the input sequence X_k is stored in computer memory in the order

$$X_0,\ X_4,\ X_2,\ X_6,\ X_1,\ X_5,\ X_3,\ X_7, \tag{24}$$

as in Fig. 5, the computation of the discrete Fourier transform may be done "in place," that is, by writing all intermediate results over the original data sequence, and writing the final answer over the intermediate results. Thus, no storage is needed beyond that required for the original N complex numbers. To see this, suppose that each node corresponds to two memory registers (the quantities to be stored are complex). The eight nodes farthest to the left in Fig. 5 then represent the registers containing the shuffled order input data. The first step in the computation is to compute the contents of the registers represented by the eight nodes just to the right of the input nodes. But each pair of input nodes affects only the corresponding pair of nodes immediately to the right, and if the computation deals with two nodes at a time, the newly computed quantities may be written into the registers from which the input values were taken, since the input values are no longer needed for further computation. The second step, computation of the quantities associated with the next vertical array of nodes to the right, also involves pairs of nodes although these pairs are now two locations apart instead of one. This fact does not change the property of "in place" computation, since each pair of nodes affects only the pair of nodes immediately to the right. After a new pair of results is computed, it may be stored in the registers which held the old results that are no longer needed. In the computation for the final array of nodes, corresponding to the values of the DFT, the computation involves pairs of nodes separated by four locations, but the "in place" property still holds.

For this version of the algorithm, the initial shuffling of the data sequence, X_k, was necessary for the "in place" computation. This shuffling is due to the repeated movement of odd-numbered members of a sequence to the end of the sequence during each stage of the reduction, as shown in Figs. 3, 4, and 5. This shuffling has been called *bit reversal*[2] because the samples are stored in bit-reversed order; i.e., $X_4 = X_{(100)_2}$ is stored in position $(011)_2 = 3$, etc. Note that the initial data shuffling can also be done "in place."

Variations of Decimation in Time: If one so desires, the signal flow graph shown in Fig. 5 can be manipulated to yield different forms of the *decimation in time* version of the algorithm. If one imagines that in Fig. 5 all the nodes on the same horizontal level as A_1 are interchanged with all the nodes on the same horizontal level as A_4, and all the nodes on the level of A_3 are interchanged with the nodes on the level of A_6, *with the arrows carried along with the nodes*, then one obtains a flow graph like that of Fig. 6.

For this rearrangement one need not shuffle the original data into the bit-reversed order, but the resulting spectrum needs to be *unshuffled*. An additional disadvantage might be that the powers of W needed in the computation are in bit-reversed order. Cooley's original description of the algorithm [1] corresponds to the flow graph of Fig. 6.

A somewhat more complicated rearrangement of Fig. 5 yields the signal flow graph of Fig. 7. For this case both the input data and the resulting spectrum are in "natural" order, and the coefficients in the computation are also used in a natural order. However, the computation may no longer be done "in place." Therefore, at least one other array of registers must be provided. This signal flow graph, and a procedure corresponding to it, are due to Stockham [8].

Decimation in Frequency: Let us now consider a second, quite distinct, form of the fast Fourier transform algorithm, *decimation in frequency*. This form was found independently by Sande [7], and Cooley and Stockham [8]. Let the time series X_k have a DFT A_r. The series and the DFT both contain N terms. As before, we divide X_k into two sequences having $N/2$ points each. However, the first sequence, Y_k, is now composed of the first $N/2$ points in X_k, and the second, Z_k, is composed of the last $N/2$ points in X_k. Formally, then

$$\begin{aligned} Y_k &= X_k \\ & \qquad k = 0, 1, 2, \cdots, \frac{N}{2} - 1. \\ Z_k &= X_{k+N/2} \end{aligned} \tag{25}$$

[2] This is a special case of digit reversal where the radix of the address is 2; more general digit reversals are available for transforms with other radices.

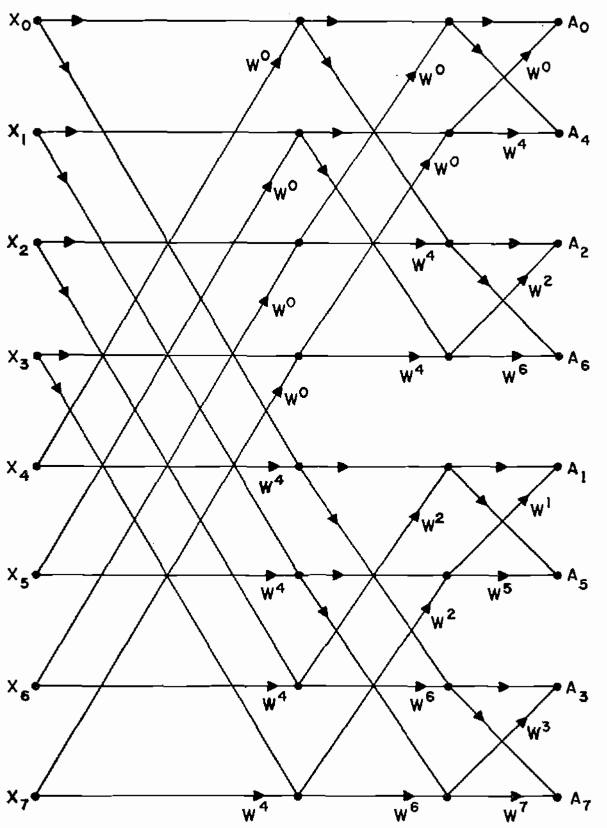


Fig. 6. Rearrangement of the flow graph of Fig. 5 illustrating the DFT computation from naturally ordered time samples.

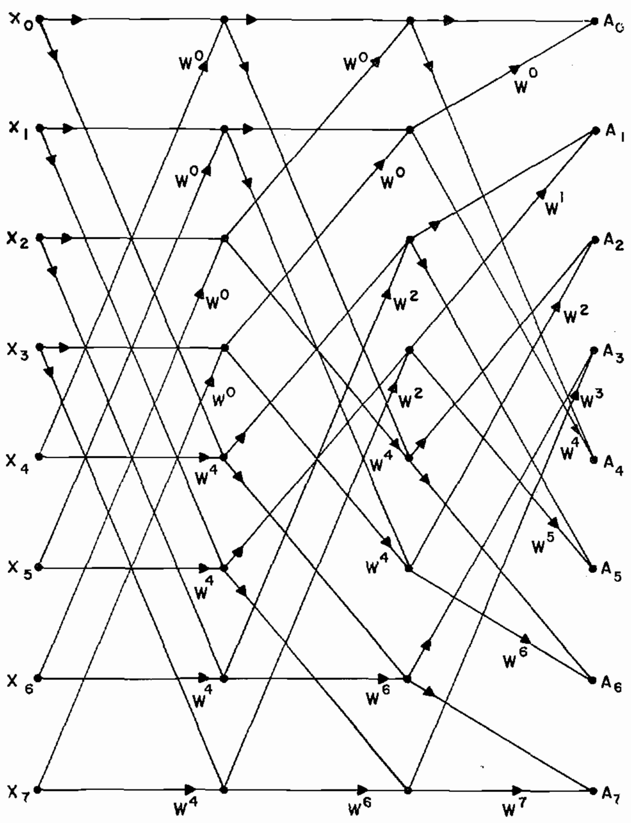


Fig. 7. Rearrangement of the flow graph of Fig. 5 illustrating the DFT computation without bit reversal.

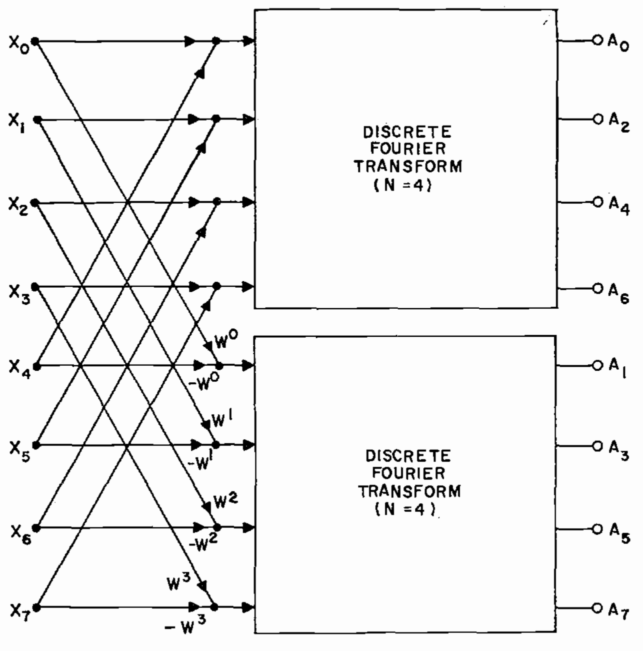


Fig. 8. Signal flow graph illustrating the reduction of endpoint DFT to two DFTs of $N/2$ points each, using decimation in frequency.

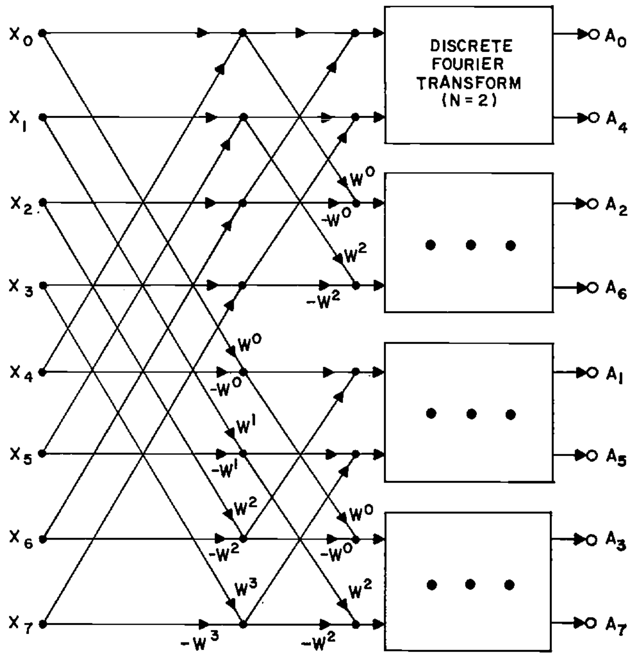


Fig. 9. Signal flow graph illustrating further reduction of the DFT computation suggested by Fig. 8.

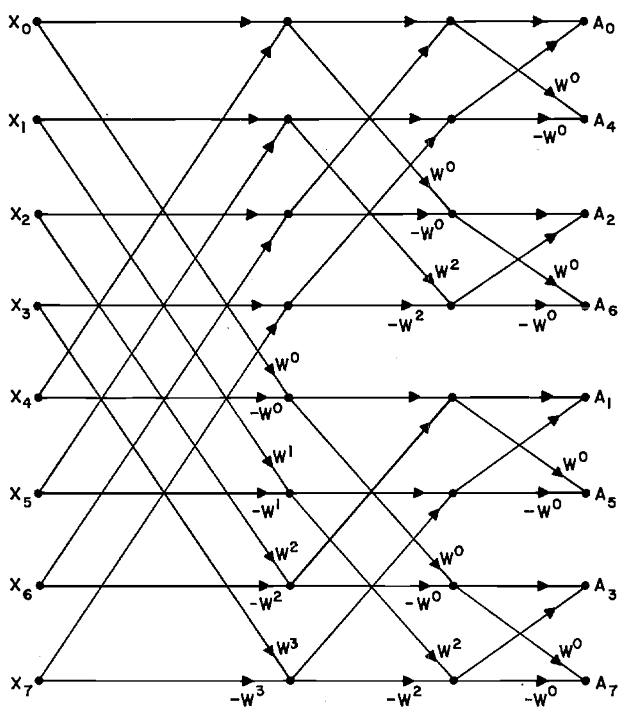


Fig. 10. Signal flow graph illustrating the computation of the DFT when the operations involved are completely reduced to multiplications and additions.

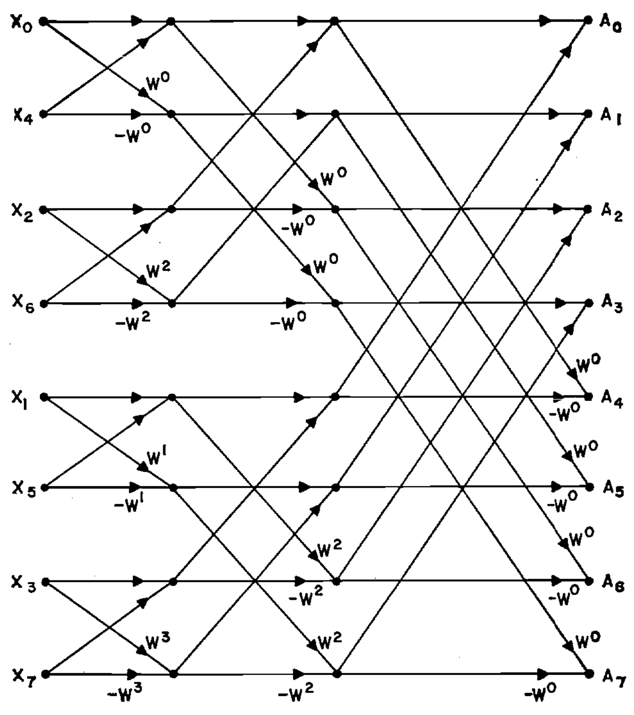


Fig. 11. Rearrangement of the flow graph of Fig. 10 illustrating the computation of the DFT to yield naturally ordered DFT coefficients.

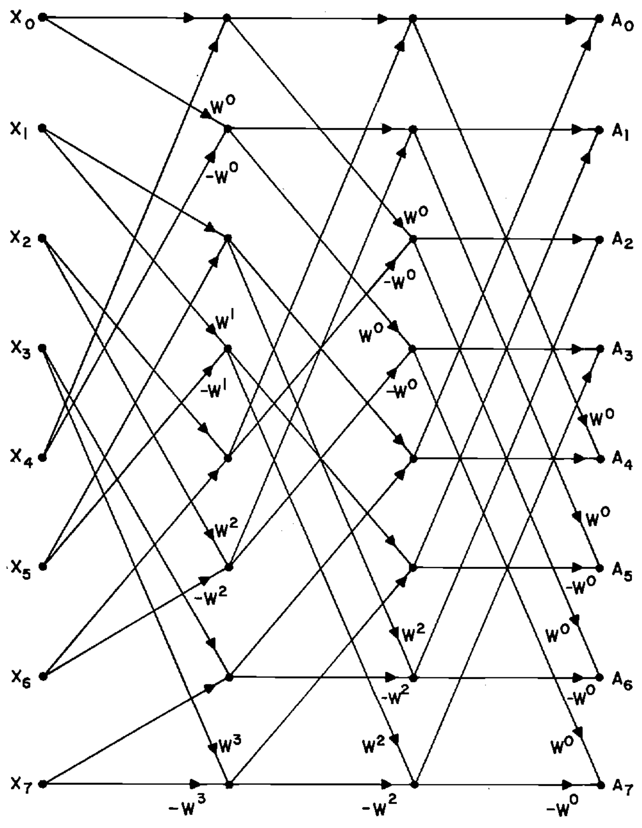


Fig. 12. Rearrangement of the flow graph of Fig. 10 illustrating the DFT computation without bit reversal.

The N point DFT of X_k may now be written in terms of Y_k and Z_k

$$A_r = \sum_{k=0}^{(N/2)-1} \left\{ Y_k \exp(-2\pi jrk/N) + Z_k \exp\left(-2\pi jr\left[k + \frac{N}{2}\right]/N\right)\right\} \tag{26}$$

$$A_r = \sum_{k=0}^{(N/2)-1} \{ Y_k + [\exp(-\pi jr)]Z_k\} \exp(-2\pi jrk/N). \tag{27}$$

Let us consider separately the even-numbered and odd-numbered points of the transform. Let the even-numbered points be R_r and the odd-numbered points be S_r, where

$$\begin{aligned} R_r &= A_{2r} \\ & \qquad 0 \leq r < N/2. \\ S_r &= A_{2r+1} \end{aligned} \tag{28}$$

It is this step that may be called *decimation in frequency*. Note that for computing the even-numbered spectrum points, (27) becomes simply

$$R_r = A_{2r} = \sum_{k=0}^{(N/2)-1} \{ Y_k + Z_k\} e^{(-2\pi jrk)/(N/2)} \tag{29}$$

which we recognize as the $N/2$ point DFT of the function (Y_k+Z_k), the sum of the first $N/2$ and the last $N/2$ time samples. Similarly, for the odd-numbered spectrum points, (27) becomes

$$\begin{aligned} S_r = A_{2r+1} &= \sum_{k=0}^{(N/2)-1} \{ Y_k + Z_k \exp(-\pi j[2r+1])\} \\ &\qquad \cdot \exp(-2\pi j[2r+1]k/N) \\ &= \sum_{k=0}^{(N/2)-1} \{ Y_k - Z_k\} e^{(-2\pi jk)/N} e^{[(-2\pi jrk)/(N/2)]} \end{aligned} \tag{30}$$

which we recognize as the $N/2$ point DFT of the function $(Y_k - Z_k) \exp(-2\pi jk/N)$.

It can be concluded from (29) and (30) that the DFT of an N-sample sequence, X_k, may be determined as follows. For even-numbered transform points, it may be computed as an $N/2$ point DFT of a simple combination of the first $N/2$ and last $N/2$ samples of X_k. For odd-numbered transform points, it may be computed as another $N/2$ point DFT of a different simple combination of the first and last $N/2$ samples of X_k. This is illustrated in the signal flow graph of Fig. 8 for an eight-point function. W has been defined in (3).

As was the case with decimation in time, we can replace each of the DFTs indicated in Fig. 8 by two 2-point DFTs, and each of the 2-point DFTs by two 1-point transforms, these last being equivalency operations. These steps are indicated in Figs. 9 and 10.

Examination of Fig. 10 gives us much information about the method of decimation in frequency, and allows us to compare it with decimation in time. Both methods require $N/2 \cdot \log N$ complex additions, complex subtractions, and complex multiplications. Both computations can be done in place. If the coefficients in the computation are to be used in a "natural" rather than "bit-reversed" order, as in Figs. 5 and 10, then the decimation in frequency method works on time samples in unshuffled order and yields frequency samples in shuffled (bit-reversed) order. Recall that Fig. 5 yielded the opposite result.

We are also able to rearrange the nodes in Fig. 10 to obtain the signal flow graph, Fig. 11, which works on shuffled time samples and yields naturally ordered frequency samples, but the coefficients are needed by the computation in bit-reversed order. The geometry of this signal flow graph is identical to the geometry of Fig. 5, just as the geometry of Fig. 10 is identical to the geometry of Fig. 6. The differences lie in the transmissions.

A somewhat more complicated rearrangement of Fig. 10 (shown in Fig. 12) yields a signal flow graph that takes unshuffled samples of the time series and produces a set of Fourier coefficients that are ***not*** in bit-reversed order. The computation cannot, however, be done "in place," and at least one other array of registers must be provided. The method is similar to that shown in Fig. 7 for decimation in time. The forms of Figs. 5, 6, 7, 10, 11, and 12 constitute a set of what we might call canonic forms of the fast Fourier transform. We may choose among these forms to find an algorithm with the properties of "in place" computation, normally ordered input, normally ordered output,·or normally ordered coefficients, but not all four at once. To achieve "in place" computation, we must deal with bit reversal, and to eliminate bit reversal we must give up "in place" computation. The two methods most effective when using homogeneous storage facilities are those providing in right order the sine and cosine coefficients needed in the computation. The other methods seem less desirable since they require wasteful tables. Still, all six methods have about equal usefulness, and the method used best will depend on the problem at hand. For example, the method shown in Fig. 10 may be used to transform from the time to the frequency domain, and the method shown in Fig. 4 may be used for the inverse transform. Any of the methods described above may be used for the inverse discrete Fourier transform if the coefficients are replaced by their complex conjugates, and if the result of the computation is multiplied by $1/N$.

The six forms mentioned are, in a sense, canonic, but one could also employ a combination of decimation in time and decimation in frequency at different stages in the reduction process, yielding a hybrid signal flow graph.

A Useful Computational Variation: It may be worth pointing out here how some programming simplicity is realized when the factors p and $q=N/p$ are relatively prime. As described by Cooley, Lewis, and Welch, [2], the "twiddle factor" W^{ir} of (23) can be eliminated by choosing subsequences of the X_k's that are different than those used before. The DFT computations are then conveniently performed in two stages.

1) Compute the q-point transforms

$$B_r^{(i)} = \sum_{k=0}^{q-1} Y_k^{(i)} \cdot W^{pkr} \qquad \begin{matrix} i = 0, 1, \cdots, p-1 \\ r = 0, 1, \cdots, q-1 \end{matrix} \tag{31}$$

of each of the p sequences

$$Y_k^{(i)} = X_{pk+qi} \qquad \begin{matrix} i = 0, 1, \cdots, p-1 \\ k = 0, 1, \cdots, q-1. \end{matrix} \tag{32}$$

2) Compute, then, the p-point transforms

$$A_s = \sum_{i=0}^{p-1} B_r^{(i)} \cdot W^{qim} \tag{33}$$

of the q sequences $B_r^{(i)}$, where

$$s = r\cdot p(p)_q^{-1} + m\cdot q(q)_p^{-1} \qquad (\text{mod } N, 0 \leq s < N) \tag{34}$$

and the notation $(p)_q^{-1}$ is meant to represent the reciprocal of p, mod q, i.e., the solution of $p(p)_q^{-1} > 1$ (mod q).

Conclusion

The integral transform method has been one of the foundations of analysis for many years because of the ease with which the transformed expressions may be manipulated, particularly in such diverse areas as acoustic wave propagation. speech transmission, linear network theory, transport phenomena, optics, and electromagnetic theory. Many problems which are particularly amenable to solution by integral transform methods have not been attacked by this method in the past because of the high cost of obtaining numerical results this way.

The fast Fourier transform has certainly modified the economics of solution by transform methods. Some new applications are presented in this special issue, and further interesting and profitable applications probably will be found during the next few years.

Appendix

As is well known, if the filter impulse response is frequency-band-limited to $1/2T$ Hz and is given by its Nyquist samples Y_h spaced T second apart, and furthermore, if the input waveform is also frequency-band-limited to $1/2T$ Hz and given by its Nyquist samples X_k spaced T second apart, then the filter output waveform is also frequency-band-limited to $1/2T$ Hz and completely specified by its Nyquist samples Z_s spaced T second apart

$$Z_s = \sum_{k=0}^{s} X_k \cdot Y_{s-k} = \sum_{l=0}^{s} X_{s-l} \cdot Y_l. \tag{35}$$

The convolution relationship facilitates computation of this equation.

To prove the convolution relationship, let the DFT of the X_k's be A_r and correspondingly the DFT of the Y_h's be B_r. The IDFT of the product of $A_r \cdot B_r$ then becomes [see (4)]

$$\begin{aligned} \left(\frac{1}{N^2}\right) \sum_{r=0}^{N-1} A_r B_r W^{-rs} &= \frac{1}{N^2} \sum_{r=0}^{N-1} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} X_k Y_l W^{r(k+l-s)} \\ &= \frac{1}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} X_k Y_l \sum_{r=0}^{N-1} \frac{W^{r(k+l-s)}}{N} \\ &= \frac{1}{N} \sum_{k=0}^{s} X_k Y_{s-k} + \frac{1}{N} \sum_{k=s+1}^{N-1} X_k Y_{N+s-k} \\ &= \left(\frac{1}{N}\right) \cdot Z_s + \text{perturbation term}. \end{aligned} \tag{36}$$

If the first $N/2$ samples of each of the two time series (X_k) and (Y_h) are assumed to be identically zero, then the perturbation term of (36) is zero so that the IDFT of the product of the two DFTs multiplied by N is equal to the convolution product Z_s of (35). Since it is always possible to select the time series to be convolved such that half of the samples are zero, the convolution relationship for the DFT can be used to compute the convolution product [see (35)] of two time series.

It is useful to point out that if $A_r = B_r$, a periodic autocorrelation function emerges.

References

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. of Comput.*, vol. 19, pp. 297–301, April 1965.
[2] J. W. Cooley, P. A. W. Lewis, and P. D. Welch, "Historical notes on the fast Fourier transform" this issue, p. 76–79.
[3] R. B. Blackman and J. W. Tukey, *The Measurement of Power Spectra.* New York: Dover, 1959.
[4] C. Bingham, M. D. Godfrey, and J. W. Tukey, "Modern techniques of power spectrum estimation," this issue, p. 56–66.
[5] T. G. Stockham, "High speed convolution and correlation," *1966 Spring Joint Computer Conf., AFIPS Proc.*, vol. 28. Washington, D. C.: Spartan, 1966, pp. 229–233.
[6] W. T. Cochran, J. J. Downing, D. L. Davin, H. D. Helms, R. A. Kaenel, W. W. Lang, and D. E. Nelson, "Burst measurements in the frequency domain," *Proc. IEEE*, vol. 54, pp. 830–841, June 1966.
[7] W. M. Gentleman and G. Sande, "Fast Fourier transforms—for fun and profit," *1966 Fall Joint Computer Conf. AFIPS Proc.*, vol. 29. Washington, D. C.: Spartan, 1966, pp. 563–578.
[8] Private communication.
[9] J. W. Cooley, "Applications of the fast Fourier transform method," *Proc. of the IBM Scientific Computing Symp.*, June 1966.