A STUDY OF FFT PRUNING AND ITS APPLICATIONS

by

M. A. RAMAKRISHNAIAH

B. E., 1970, University of Mysore, Karnataka, India

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1975

Approved by:

Nasir Ahmed

Major Professor

TABLE OF CONTENTS

ILLEGIBLE

THE FOLLOWING DOCUMENT (S) IS ILLEGIBLE DUE TO THE PRINTING ON THE ORIGINAL BEING CUT OFF

ILLEGIBLE

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Since the development of the fast fourier transform (FFT) by Cooley and Tukey [5], considerable attention has been devoted to its modification to secure increased speed for computational purposes. Basically there are four modifications to increase the computational efficiency. These are: (1) innerloop nesting, (2) change in radix, (3) data shuffling and unscrambling when the input data is real, and (4) eliminating operations on zeros when the number of nonzero input data points is considerably smaller than the desired number of output points; or the desired number of transform points is considerably smaller than the number of input points.

The first modification is used in decimation in frequency and decimation in time algorithms, some aspects of which are discussed in Chapter II. The second and third modifications are discussed in [3], [8] and [16] respectively.

This report is primarily concerned with the fourth modification which is referred to as FFT pruning. FFT pruning eliminates operations that do not contribute to the final output. It can be applied to both discrete time and frequency domains, and saves considerable time. Applications of FFT pruning include speech processing, estimation of autocorrelation functions, and computing narrow band Fourier spectra with increased frequency resolution.

FFT pruning concepts are introduced in Chapter III, while experimental results pertaining to some applications are considered in Chapter IV. Conclusions and recommendations for future work are presented in Chapter V.

CHAPTER II

DECIMATION IN TIME AND FREQUENCY

2.1 Discrete Fourier Transform

The Fourier transform pair for continuous signals can be written in the form

$$F_x(f) = \int_{-\infty}^{\infty} x(t)\ e^{-i2\pi ft}dt$$

$$x(t) = \int_{-\infty}^{\infty} F_x(f)\ e^{i2\pi ft}df$$

for $-\infty<f<\infty$, $-\infty<t<\infty$, and $i = \sqrt{-1}$. $F_x(f)$ represents the frequency domain function corresponding to the time domain function $x(t)$. Analogous to the Fourier transform, the discrete Fourier transform (DFT) is a transform that is used for the Fourier analysis of data sequences. Thus, if $\{X(m)\}$ denotes a sequence $X(m)$, $m=0, 1, \ldots,(N-1)$ of N finite valued real or complex numbers, then its DFT is defined as

$$C_x(k) = \frac{1}{N} \sum_{m=o}^{N-1} X(m)\ W^{km},\ k = 0,\ 1,\ \ldots\ldots,(N-1) \qquad (2.1)$$

where $W = e^{\frac{-i2\pi}{N}}$, $i = \sqrt{-1}$.

Again, the corresponding inverse discrete Fourier transform (IDFT) is defined as

$$X(m) = \sum_{k=o}^{N-1} C_x\ (k)\ W^{-km},\ m = 0,\ 1,\ \ldots\ldots\ (N-1) \qquad (2.2)$$

Equations (2.1) and (2.2) constitute the DFT pair.

2.2 Fast Fourier Transform

The fast Fourier Transform (FFT) is an algorithm which is used to compute the DFT. Direct evaluation of Eq. (2.1) requires N^2 multiplications and additions. In contrast, the FFT requires only N $\log_2 N$ complex number additions and multiplications. The FFT can be interpreted in terms of combining the DFT's of the individual data samples such that the occurence times of these samples are taken into account sequentially, and applied to the DFT's of progressively larger, mutually exclusive subgroups of data samples, which are combined to ultimately produce the DFT of the complete series of data samples [4].

There are two classes of FFT algorithms. These are: (i) decimation in time, and (ii) decimation in frequency. Within each class there are several modifications, each of which is most efficient when the number of data points is an integer power of two. However, some fast algorithms have been developed for cases where the number of points in the data sequence is an integer power of a radix other than two [5, 8, 16].

2.3 Decimation in Time

This form of algorithm was used by Cooley and Tukey [5]. Before discussing the algorithm it is instructive to illustrate a factorization property which is common to both decimation in time as well as decimation in frequency algorithm.

Consider the case when the number of data points N, is of the form N = A x B. Then Eq. (2.1) can be written as

$$Cx\ (c + dA) = \sum_{b=o}^{B-1} \sum_{a=o}^{A-1} X\ (b+ab)\ W^{(b+aB)(c+dA)} \tag{2.3}$$

where m = b+aB is the time index, k = c+dA is the frequency index, and a, c = 0, 1,.....(A-1); b, d = 0,1,.....(B-1). In Eq. (2.3), the quantity $W^{(b+aB)(c+dA)}$ can be simplified as follows

$$W^{(b+aB)}\,W^{(c+dA)} = W^{bc}\,W^{bdA}\,W^{acB}\,W^{adAB}$$

$$= W^{bc}\,W^{bdA}\,W^{acB} \text{ since } W^{adAB} = W^{adN} = 1 \qquad \text{.....(2.4)}$$

Substituting Eq. (2.4) in Eq. (2.3) and rearranging terms, we obtain

$$C_x(c+dA) = \sum_{b=0}^{B-1} W^{bdA} \sum_{a=o}^{A-1} X(b+aB)\,W^{acB}\,W^{bc} \qquad (2.5)$$

$$c = 0,\ 1, \ldots\ldots,\ (A-1);\ d = 0,\ 1, \ldots,\ (B-1).$$

This can be recognized as a sequence of two Fourier transforms applied to data sequences of length A and B respectively. It is observed that factoring of the exponential W^{bdA} from the inner sum in Eq. (2.5) reduces the total number of multiplications required to compute the $C_x(c+dA)$ coefficients. This technique is used in both types of algorithms (i.e. decimation in time and decimation in frequency).

Suppose the given data sequence has N samples. It is convenient to divide it into two subsequences $\{Y_1(m)\}$ and $\{Y_2(m)\}$, each of which has N/2 points. $\{Y_1(m)\}$ is composed of even numbered points, while $\{Y_2(m)\}$ is composed of odd numbered points, as illustrated in Fig. 2.1. It follows that the elements of $\{Y_1(m)\}$ and $\{Y_2(m)\}$ can be expressed as

$$\left.\begin{aligned} Y_1(m) &= X(2m) \\ Y_2(m) &= X(2m+1) \end{aligned}\right\} \text{ for } m = 0,\ 1,\ 2, \ldots (N/2-1) \qquad (2.6)$$

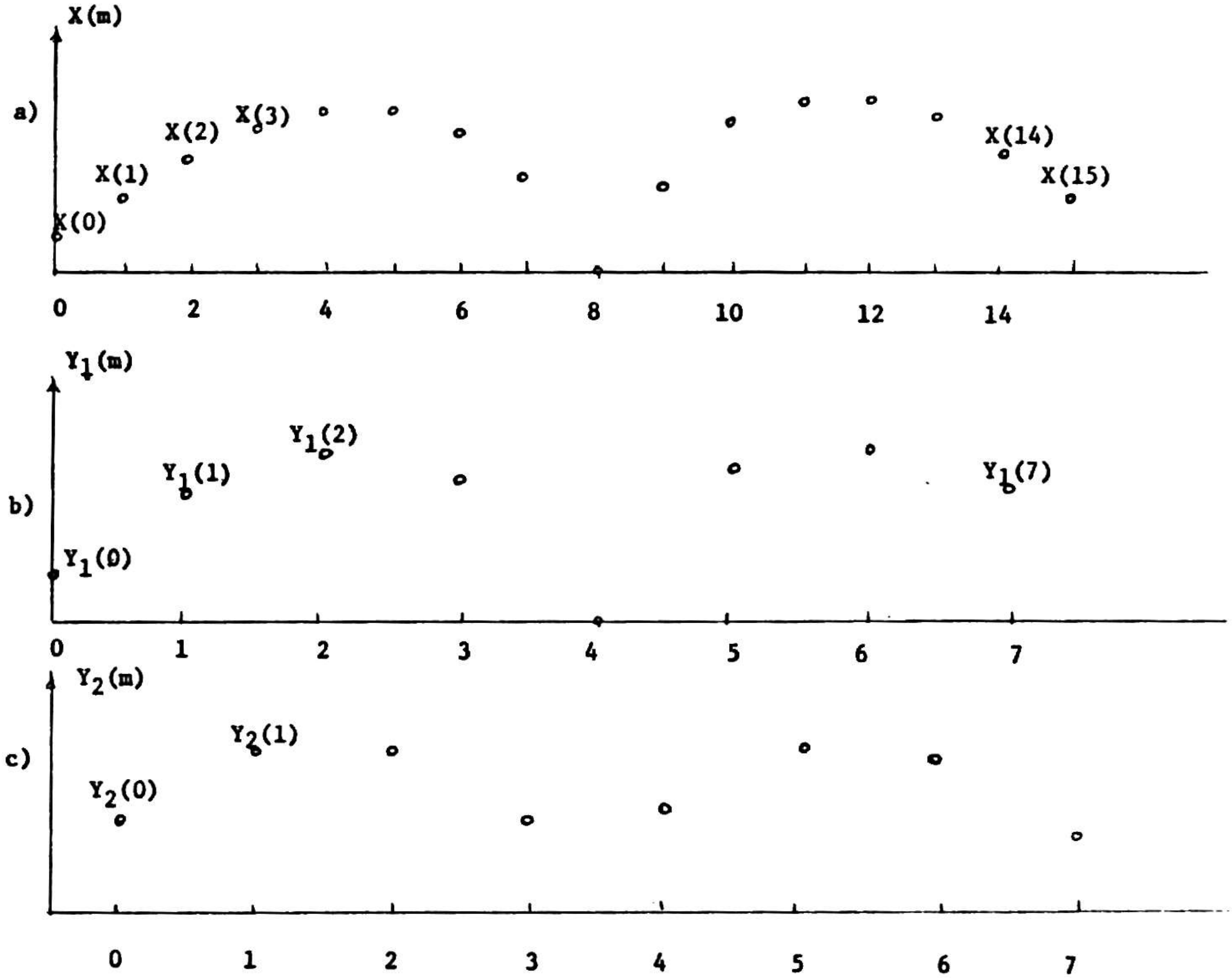


Fig. 2.1. (a) sequence $\{X(m)\}$, (b) sequence $\{Y_1(m)\}$, and (c) sequence $\{Y_2(m)\}$.

Since the data sequence $\{X(m)\}$ is considered to be periodic with period N, the subsequences $\{Y_1(m)\}$ and $\{Y_2(m)\}$ can be regarded to be periodic with period N/2. Thus the DFT's of these sequences are given by

$$Cy_1(k) = \sum_{m=0}^{N/2-1} Y_1(m)(W^2)^{mk}$$

$$k = 0,\ 1,\ \ldots..(N/2-1)$$

$$Cy_2(k) = \sum_{m=0}^{N/2-1} Y_2(m)(W^2)^{mk} \tag{2.7}$$

Now, the desired DFT of $\{X(m)\}$ can be expressed in terms of $\{Y_1(m)\}$ and $\{Y_2(m)\}$ to obtain

$$C_x(k) = \sum_{m=0}^{N/2-1} [Y_1(m)W^{2mk} + Y_2(m)W^{(2m+1)k}],\ k = 0,\ 1 \quad (N/2-1)$$

$$C_x(k) = \sum_{m=0}^{N/2-1} [Y_1(m)W^{2mk}] + W^k \sum_{m=0}^{N/2-1} Y_2(m)W^{2mk} \tag{2.8}$$

which yields

$$C_x(k) = Cy_1(k) + W^k\ Cy_2(k) \tag{2.9}$$

In Eq. (2.9), the index k takes the values 0, 1,....,(N-1). However, since $Cy_1(k)$ and $Cy_2(k)$ are periods with period N/2, they need be computed only for k = 0, 1,..., (N/2-1). Thus, $C_x(k)$ for $N/2 \leq k \leq (N-1)$ can be computed using the relation

$$C_x(k) = Cy_1(k-N/2) + W^k\ Cy_2(k-N/2) \tag{2.10}$$

The computational implication of Eqs. (2.9) and (2.10) is illustrated in Fig. (2.2). In Fig. (2.2) it is seen that an 8-point DFT is reduced to two 4-point

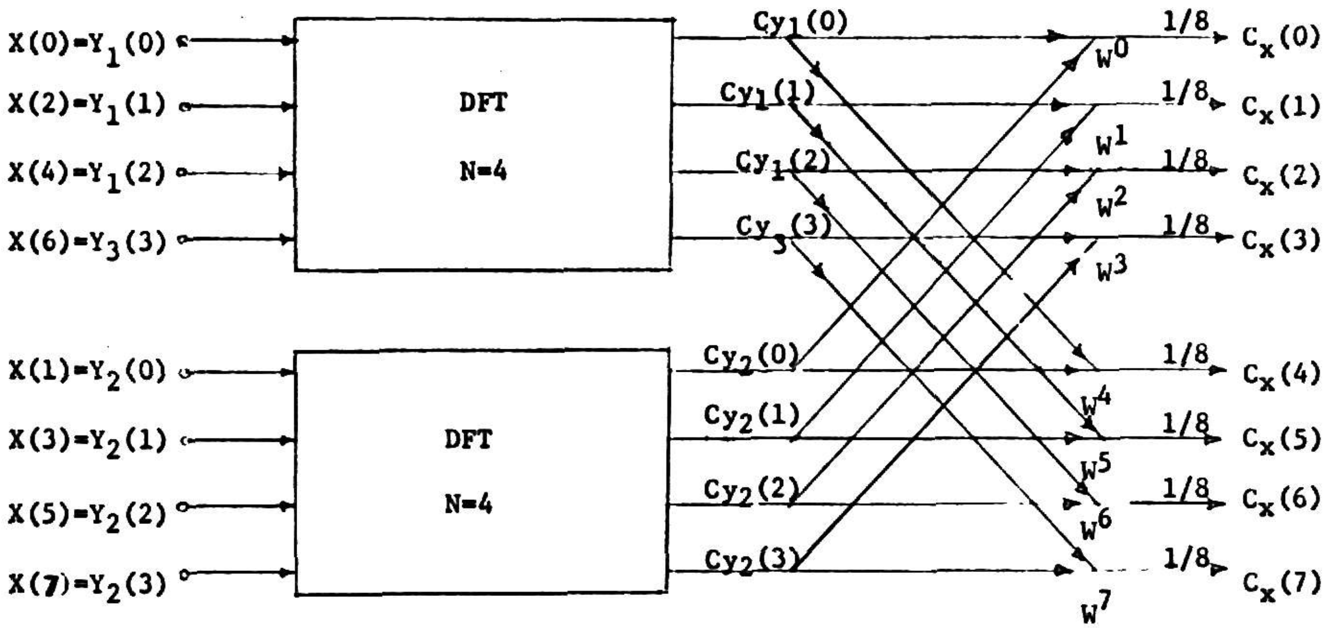


Fig. 2.2. 8-point DFT reduced to two 4-point DFT's by decimation in time.

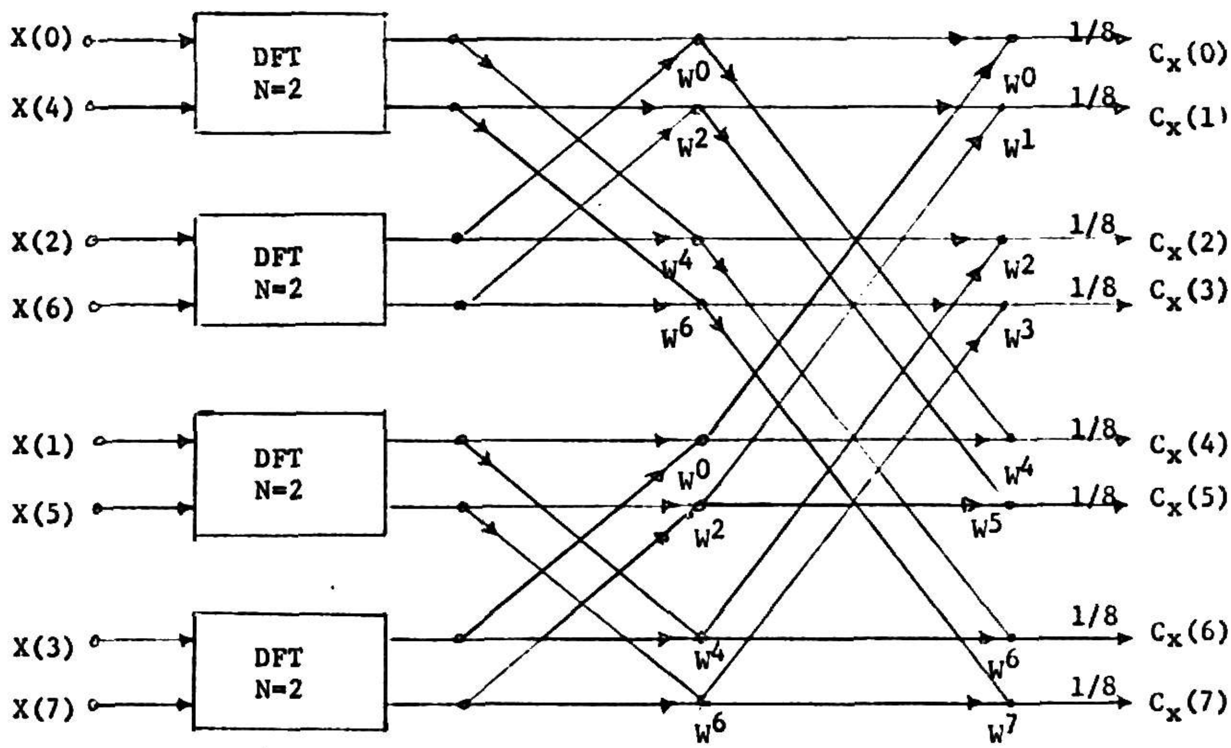


Fig. 2.3. 8-point DFT reduced to four 2-point DFT's.

DFT's. Similarly the computation of $Cy_1(k)$ and $Cy_2(k)$ can be reduced to the computation of four 2-point sequences. These reductions can be carried out as long as each function has a number of samples that is divisible by two, as illustrated in Fig. 2.3. The two 4-point DFTS in Fig. 2.3 have each been reduced to two 2-point DFTS. Finally in Fig. 2.4, the 2-point DFTS have been reduced to 1-point DFTS. Thus in general, if $N = 2^n$ we can make n such reductions by applying Eqs. (2.3), (2.9) and (2.10), first for N, then for N/2 and so on, followed by a 1-point DFT.

In Fig. 2.4 it is observed for N = 8, there are 8 x 3 nodes, 2 x 8 x 3 arrows corresponding to $N \log_2 N$ additions and $2 N \log_2 N$ multiplications. Half of the multiplications can be eliminated since the transmissions indicated by the arrows are unity. Half of the remaining multiplications are also eliminated utilizing the fact that $W^{N/2} = -1$. In all, $N \log_2 N$ additions and at most $\frac{1}{2} N \log_2 N$ multiplications are required for computing the DFT of an N-point sequence, where N is a power of two. Further, if the input data has been stored in the order X(0), X(4), X(2), X(6), X(1), X(5), X(3), X(7) as in Fig. 2.4 then the computation may be done "in place" storing all intermediate and final output data in the same storage locations as the original data sequence. Thus the number of storage locations required is approximately N.

The signal flow graph shown in Fig. 2.4 can be manipulated to yield different versions of the decimation in time algorithm. One such rearrangement is shown in Fig. 2.5, where the input data is in natural order while shuffling is necessary at the output. A relatively complicated rearrangement of Fig. 2.4 yields the signal flow graph in Fig. 2.6. In this case, both the input and the output are in natural order. However, this needs additional storage and the computation is not done 'in place' as was the case in Fig. 2.4.

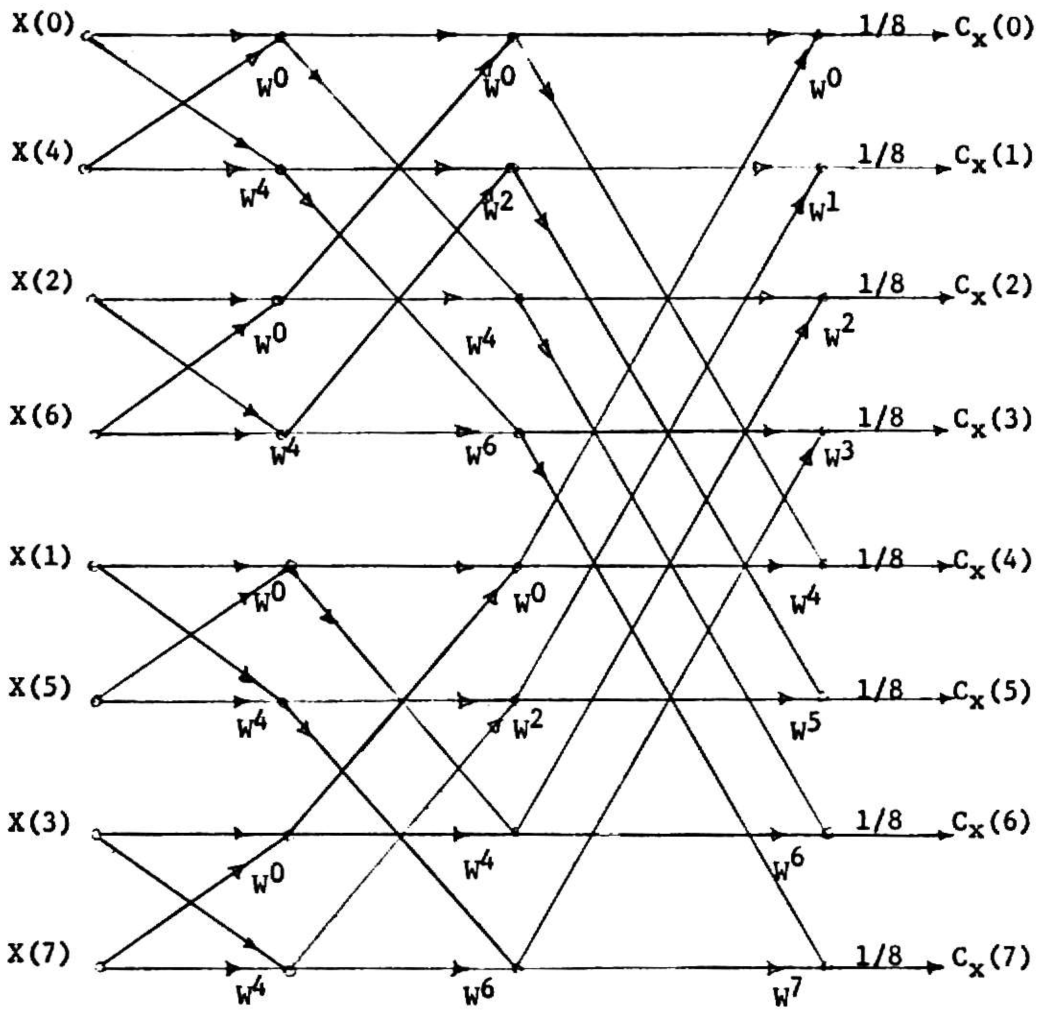


Fig. 2.4. 8-point DFT using decimation in time.

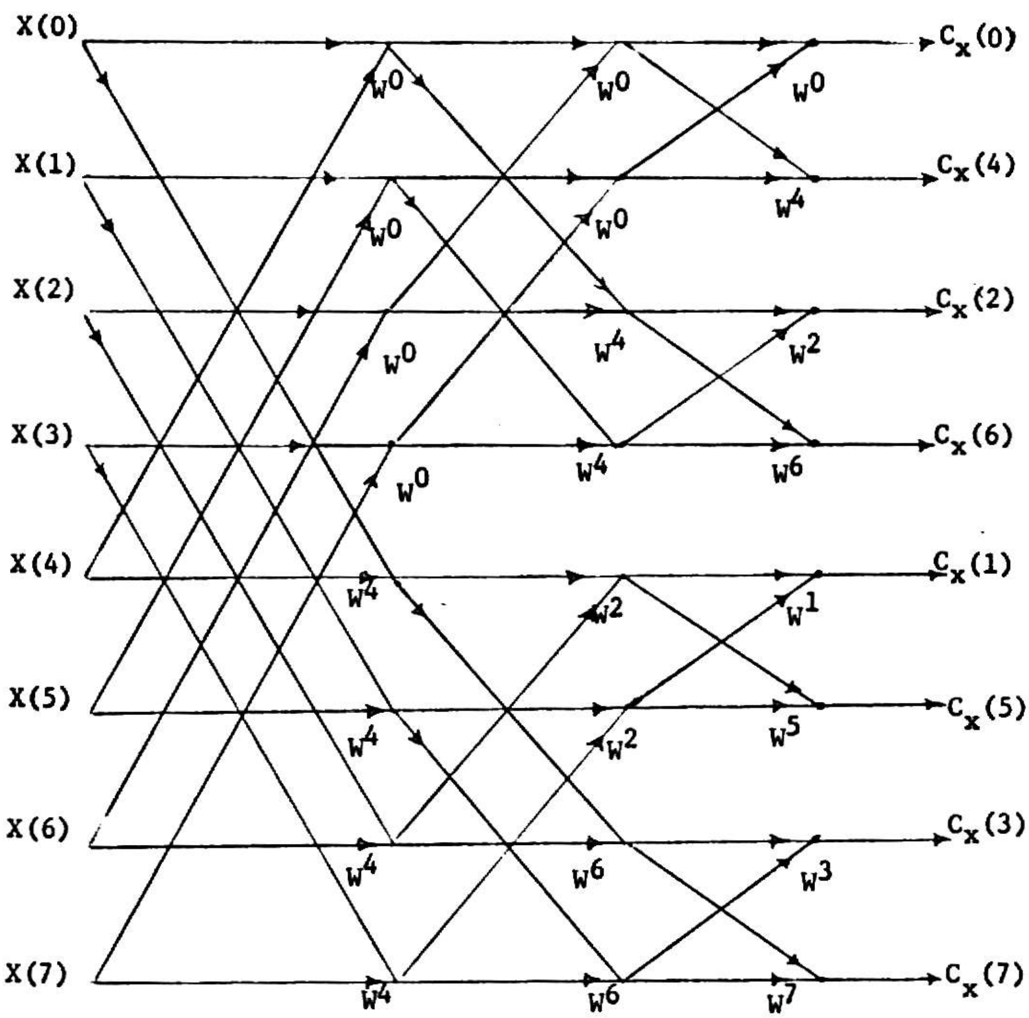


Fig. 2.5. 8-point DFT with input in natural order, using decimation in time.

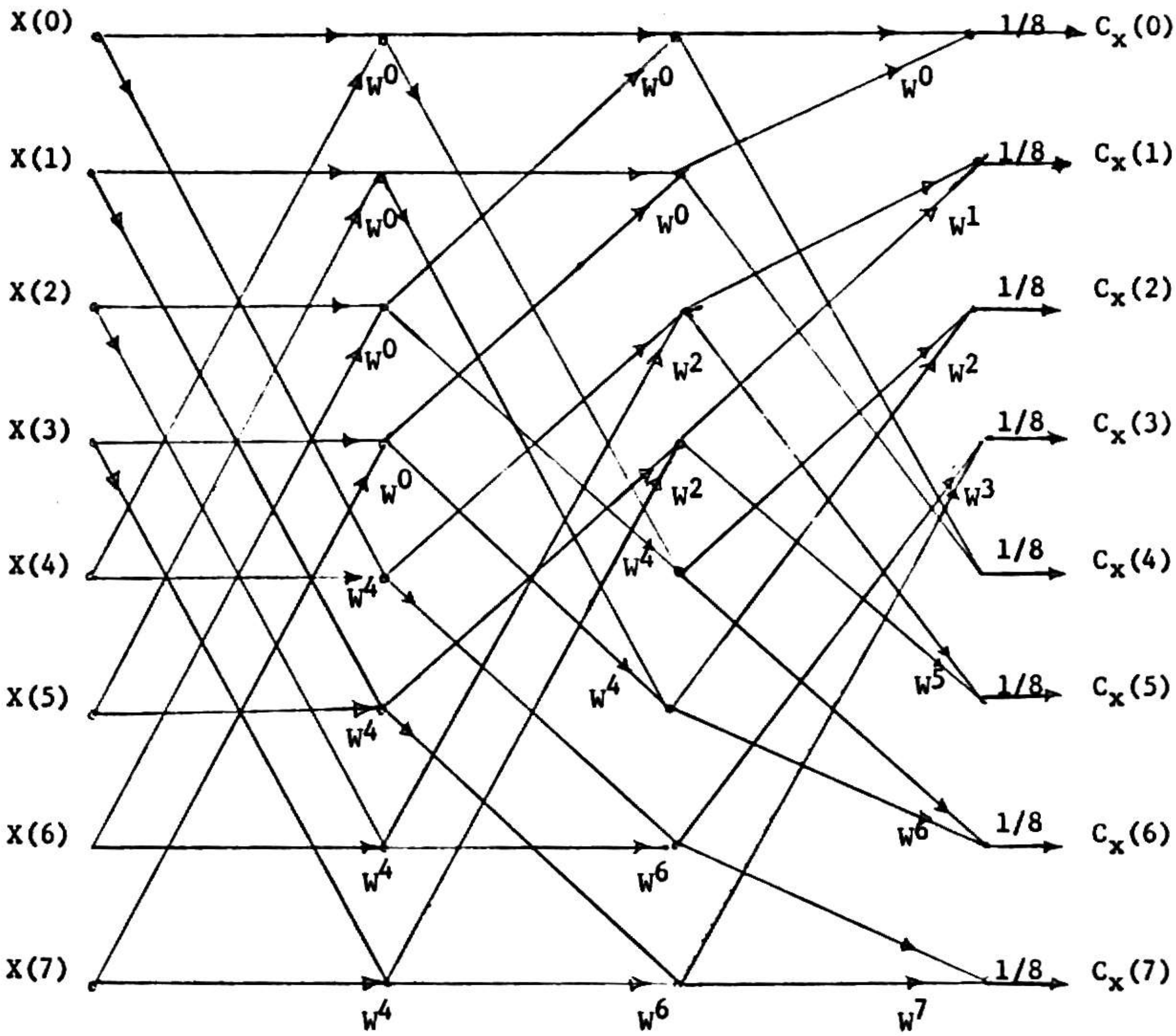


Fig. 2.6. 8-point DFT with input and output in natural order.

Thus the basic procedure for carrying out decimation in time is to form subsequences from the sequence to be transformed. Each subsequence is composed of only every n^{th} point of the original sequence. It is as though these subsequences are formed by sampling the time function at a lower rate.

A common example of the decimation in time technique is the Cooley-Tukey algorithm, a brief discussion of which follows.

Cooley-Tukey Algorithm [5] From Eq. (2.1), we have

$$C_x(k) = \frac{1}{N} \sum_{m=0}^{N-1} X(m)\, W^{km}, \; k = 0,\, 1,\, \ldots\ldots (N-1) \qquad (2.11)$$

where $W = e^{\frac{-i2\pi}{N}}$, $i = \sqrt{-1}$

suppose $N = 8$. It is convenient to represent both m and k in binary form; that is

$$m = 4m_2 + 2m_1 + m_0 \qquad (2.12)$$

$$k = 4k_2 + 2k_1 + k_0 \qquad (2.13)$$

where m_2, m_1, m_0 and k_2, k_1, k_0 are binary digits. Let

$$C_x(k) = C_x(k_2,\, k_1,\, k_0) \qquad (2.14)$$

$$x(m) = x(m_2,\, m_1,\, m_0) \qquad (2.15)$$

Substituting in Eq. (2.11), we obtain

$$C_x(k_2,\, k_1,\, k_0) = \frac{1}{N} \sum_{m_0=0}^{1} \sum_{m_1=0}^{1} \sum_{m_2=0}^{1} X(m_2,\, m_1,\, m_0)\, W^{(4k_2+2k_1+k_0)(4m_2+2m_1+m_0)} \qquad \ldots\ldots \; (2.16)$$

Noting that $W^{m+n} = W^m W^n$, we have

$$W^{(4k_2+2k_1+k_0)(4m_2+2m_1+m_0)}$$

$$= W^{(4k_2+2k_1+k_0)\ 4m_2}\ W^{(4k_2+2k_1+k_0)\ 2m_1}\ \ W^{(4k_2+2k_1+k_0)\ m_0} \qquad \ldots\ldots\ (2.17)$$

Consider each of the factors on the right hand side of the Eq. (2.17) in turn as follows:

$$W^{(4k_2+2k_1+k_0)\ 4m_2} = [W^{8(2k_2+k_1)m_2}]\ W^{4k_0m_2} \qquad (2.18)$$

$$W^{(4k_2+2k_1+k_0)\ 2m_1} = [W^{8k_2m_1}]\ W^{(2k_1+k_0)2m_1} \qquad (2.19)$$

$$W^{(4k_2+2k_1+k_0)\ m_0} = W^{(4k_2+2k_1+k_0)\ m_0} \qquad (2.20)$$

Using the property

$$W^8 = [e^{\frac{2\pi i}{8}}]^8 = e^{2\pi i} = 1 \qquad (2.21)$$

the bracketed portions of Eqs. (2.18) and (2.19) can be replaced by unity. Then Eq. (2.16) can be written in the form

$$C_x(k_2,\ k_1,\ k_0)$$

$$= \frac{1}{8} \sum_{m_0=0}^{1} \sum_{m_1=0}^{1} [\ \sum_{m_2=0}^{1} X(m_2,m_1,m_0)\ W^{4k_0m_2}]\ W^{(2k_1+k_0)\ 2m_1}\ \ W^{(4k_2+2k_1+k_0)m_0} \qquad \ldots\ldots\ (2.22)$$

The innermost summation is performed over m_2 for the two values 0 and 1. Thus the bracketed quantity in Eq. (2.22) is a function of m_1, m_0, k_0, and may be written as

$$X_1(k_0,m_1,m_0) = X\ (m_2,m_1,m_0)\ W^{4k_0m_2} \qquad (2.23)$$

Substituting Eq. (2.23) in Eq. (2.22) we get

$$C_x(k_2,k_1,k_0) = \frac{1}{8} \sum_{m_0=0}^{1} [\sum_{m_1=0}^{1} X_1(k_0,m_1,m_0)\ W^{(2k_1+k_0)\ 2m_1}]\ W^{(4k_2+2k_1+k_0)\ m_0} \tag{2.24}$$

The innermost summation is performed over m_1 for the two values 0 and 1. Again, the bracketed quantity in Eq. (2.24) is a function of k_1, k_0 and m_0. Thus it follows that

$$X_2\ (k_0,k_1,m_0) = X_1\ (k_0,m_1,m_2)\ W^{(2k_1+k_0)\ 2m_1} \tag{2.25}$$

Substituting Eq. (2.25) in Eq. (2.24) we get

$$C_x\ (k_2,k_1,k_0) = \frac{1}{8} [\sum_{m_0=0}^{1} X_2(k_0,k_1,m_0)\ W^{(4k_2+2k_1+k_0)\ m_0} \tag{2.26}$$

The summation is performed over m_0 for the two values 0 and 1 to obtain a function of k_0, k_1, k_2. The bracketed quantity may then be written as

$$X_3\ (k_0,k_1,k_2) = \sum_{m_0=0}^{1} X_2\ (k_0,k_1,m_0)\ W^{(4k_2+2k_1+k_0)\ m_0} \tag{2.27}$$

substituting Eq. (2.27) in (2.26) we obtain

$$C_x(k_2,k_1,k_0) = \frac{1}{8} X_3\ (k_0,k_1,k_2) \tag{2.28}$$

Eq. (2.28) gives the desired DFT. The signal flowgraph corresponding to the above development is shown in Fig. 2.7. This is the flowgraph for the Cooley-Tukey algorithm for N = 8.

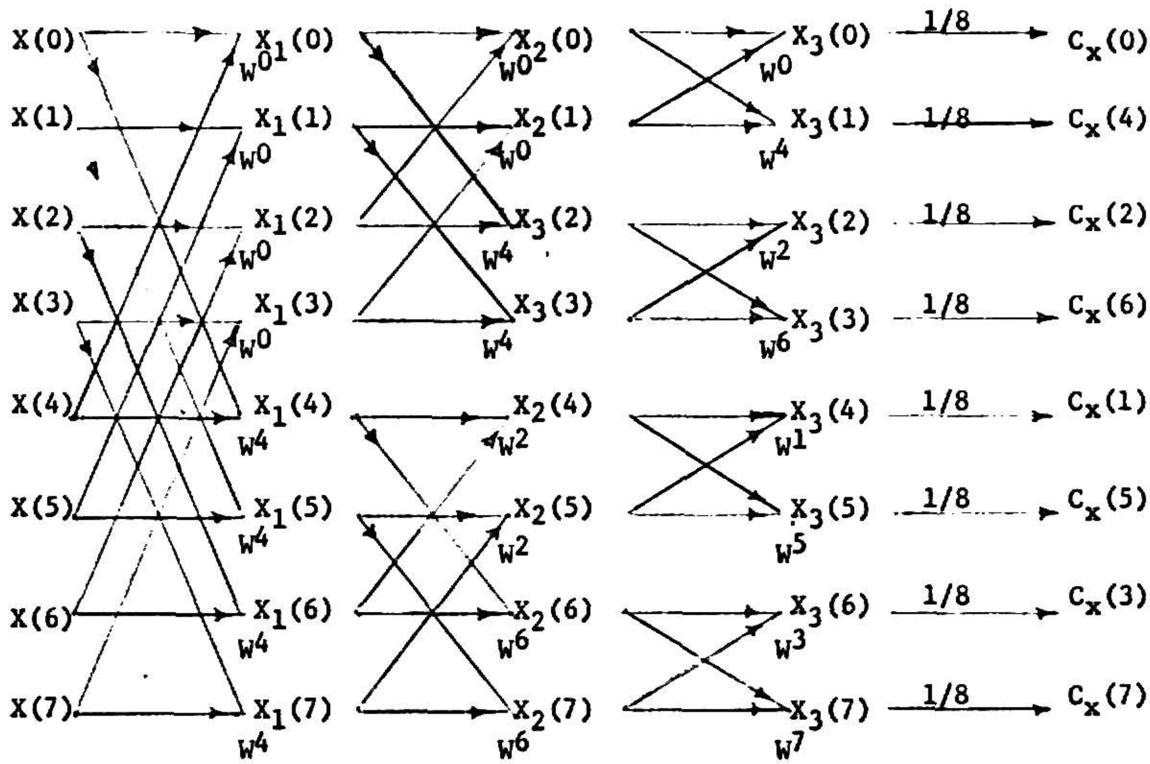


Fig. 2.7. Cooley-Tukey Algorithm for N=8.

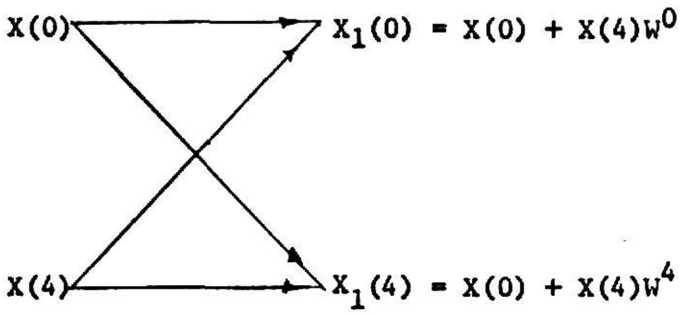


Fig. 2.8. A butterfly.

In Fig. 2.7 each iteration consists of a number of butterflies. A butterfly is represented in Fig. 2.8. Each butterfly consists of a complex addition or substraction and a multiplication. For each iteration there are four butterflies. In general, there are N/2 butterflies per iteration.

2.4 Decimation in Frequency

This form of algorithm was found independently by Sande, Cooley and Stockham.

Let $\{X(m)\}$ denote a data sequence $X(m)$, $m = 0, 1, \ldots, (N-1)$ which is obtained by sampling a band limited signal $x(t)$. As before, let $C_x(k)$, $k = 0, 1, \ldots, (N-1)$ denote the DFT coefficients of $\{X(m)\}$. We divide $\{X(m)\}$ into two sequences of $\{Y_1(m)\}$ and $\{Y_2(m)\}$ points each, as follows:

$$Y_1(m) = X(m)$$

$$Y_2(m) = X(m+N/2), \quad m = 0, 1, \ldots, (N/2-1) \tag{2.29}$$

The DFT of $\{X(m)\}$ can be expressed in terms of $\{Y_1(m)\}$ and $\{Y_2(m)\}$ to obtain

$$C_x(k) = \sum_{m=0}^{N/2-1} \{Y_1(m)\ W^{km} + Y_2(m)\ W^{k(m+N/2)}\}$$

$$= \sum_{m=0}^{N/2-1} \{Y_1(m) + W^{kN/2}\ Y_2^{(m)}\}\ W^{km} \tag{2.30}$$

Decimation in frequency is realized via Eq. (2.30) by replacing k by 2k for even $C_x(k)$ and by (2k+1) for odd $C_x(k)$. This results in

$$C_x(2k) = \sum_{m=0}^{N/2-1} \{Y_1(m)+Y_2(m)\}W^{2mk}, \quad k = 0, 1, \ldots, (N/2-1)$$

$$\ldots\ldots \tag{2.31}$$

$$C_x\ (2k+1) = \sum_{m=0}^{N/2-1} \{Y_1(m)-Y_2(m)\}\ W^{(2k+1)N/2}\}\ W^{(2k+1)m}$$

$$= \sum\{Y_1(m)-Y_2(m)\}\ W^{2km}\ W^m \qquad (2.32)$$

The signal flow graph corresponding to Eqs. (2.31) and (2.32) for N = 8 is developed as shown in Figs. 2.9, 2.10 and 2.11 In Fig. 2.9, an 8-point DFT has been reduced to two 4-point DFT's. In Figs. 2.10 and 2.11 successive reductions on smaller DFT's are carried out as long as the number of points in the subsequences is divisible by two. Fig 2.10 shows the final flow graph which involves compex additions and multiplications. In general the number of complex number computations is proportional to $N \log_2 N$.

Comparing Fig. 2.5 and Fig. 2.11, we make the following observations: (1) In decimation in time algorithms, the data sequence is shuffled while the DFT's are computed in natural order. (2) In decimation in frequency algorithms, the data sequence is not shuffled while the DFT's are in shuffled order.

By rearranging the signal flow graph of Fig. 2.11, we can obtain a flow-graph for shuffled input and natural ordered output as shown in Fig. 2.12. As it was shown for decimation in time (see Fig. 2.6), a flowgraph where the input and output are both in natural order can also be developed for decimation in frequency. This type of flow graph is shown in Fig. 2.13. An example of the decimation in time technique is the Sande-Tukey algorithm, which is discussed next.

<u>Sande-Tukey Algorithm</u> [8]:

Let N = 8 and $n = \log_2 N = 3$. Then from Eq. (2.16) we get

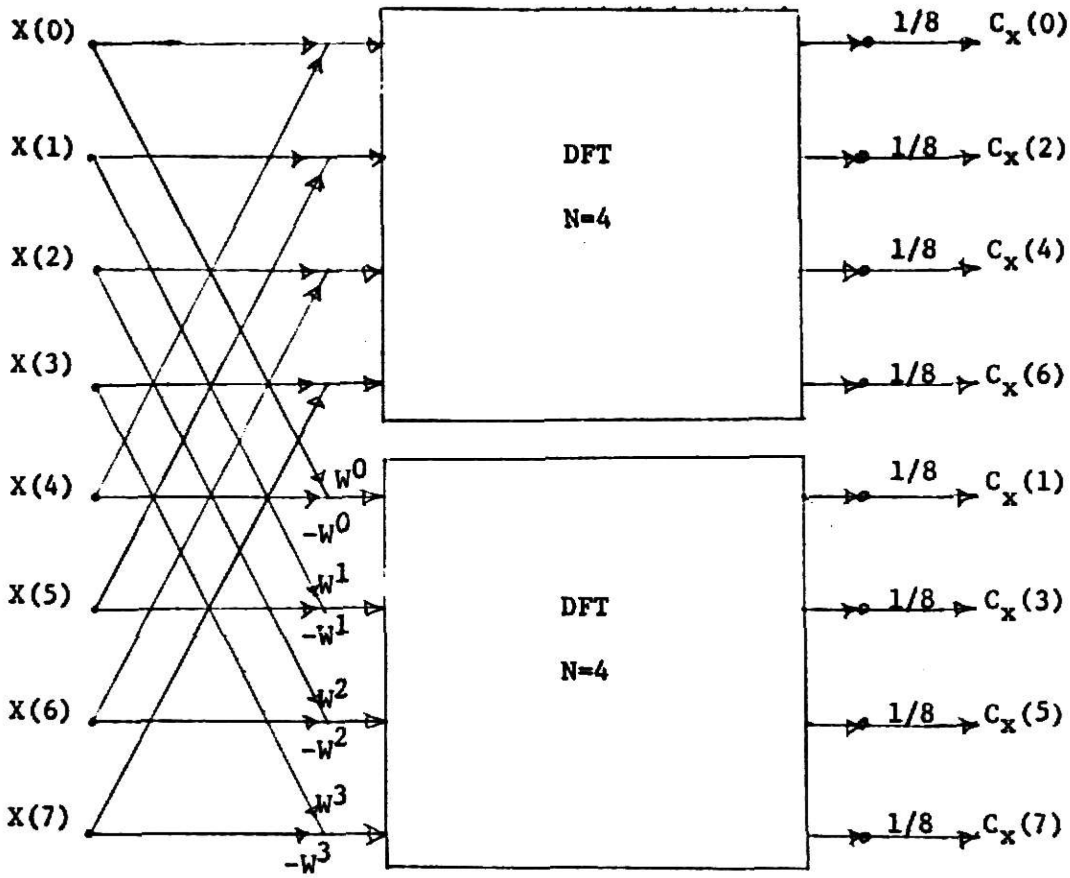
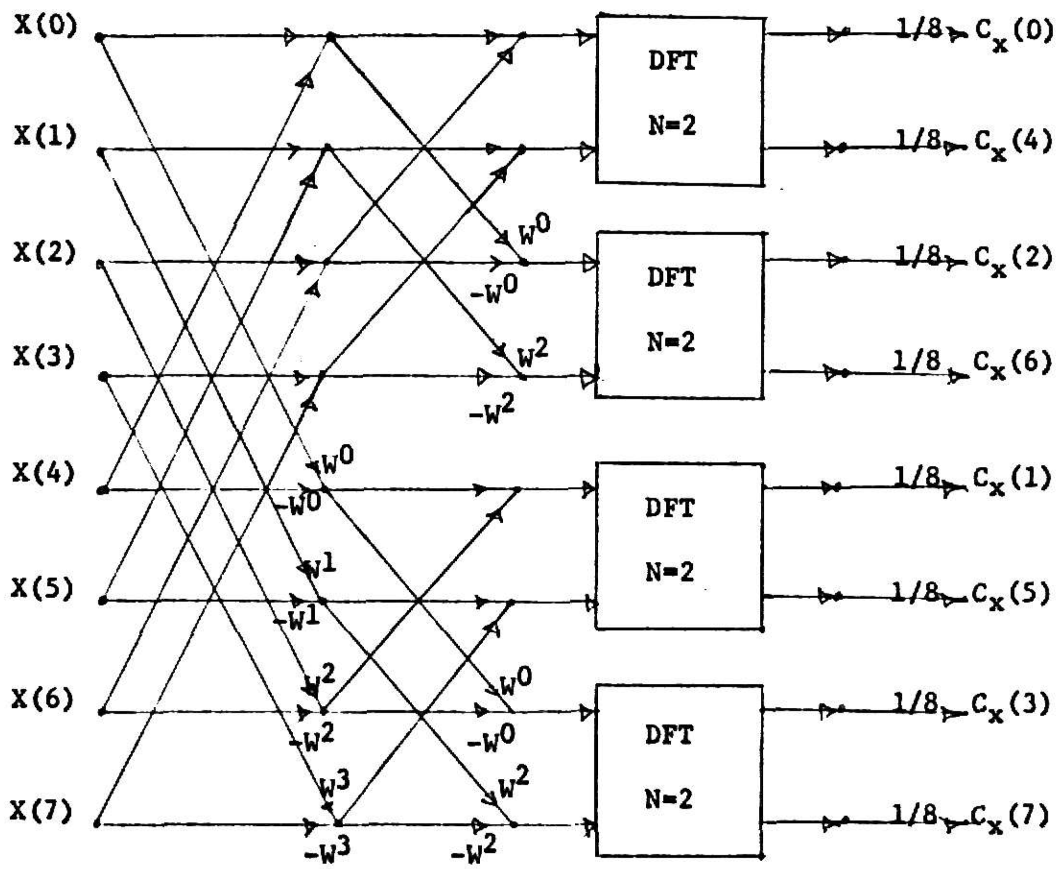


Fig. 2.9. 8-point DFT reduced to two 4-point DFT's by decimation in frequency.



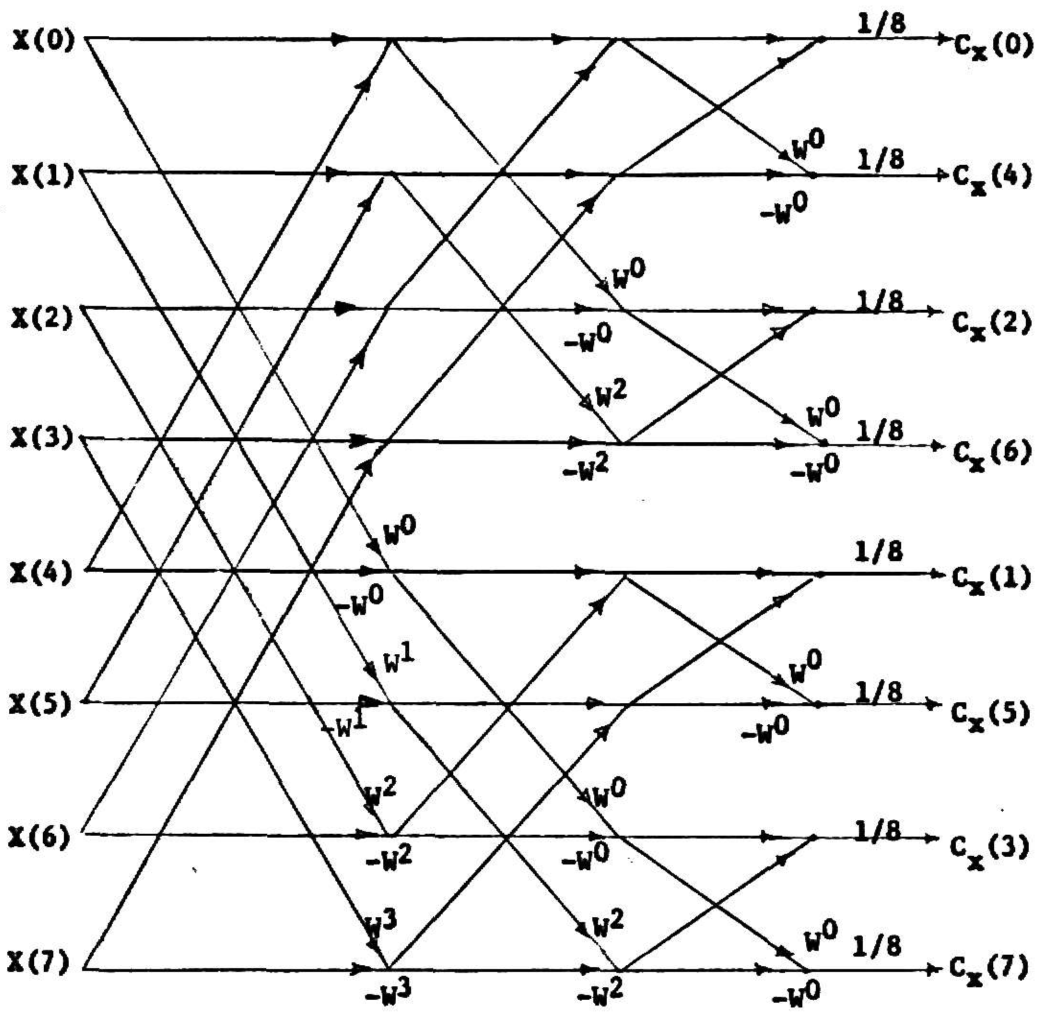


Fig. 2.11. 8-point DFT using decimation in frequency.

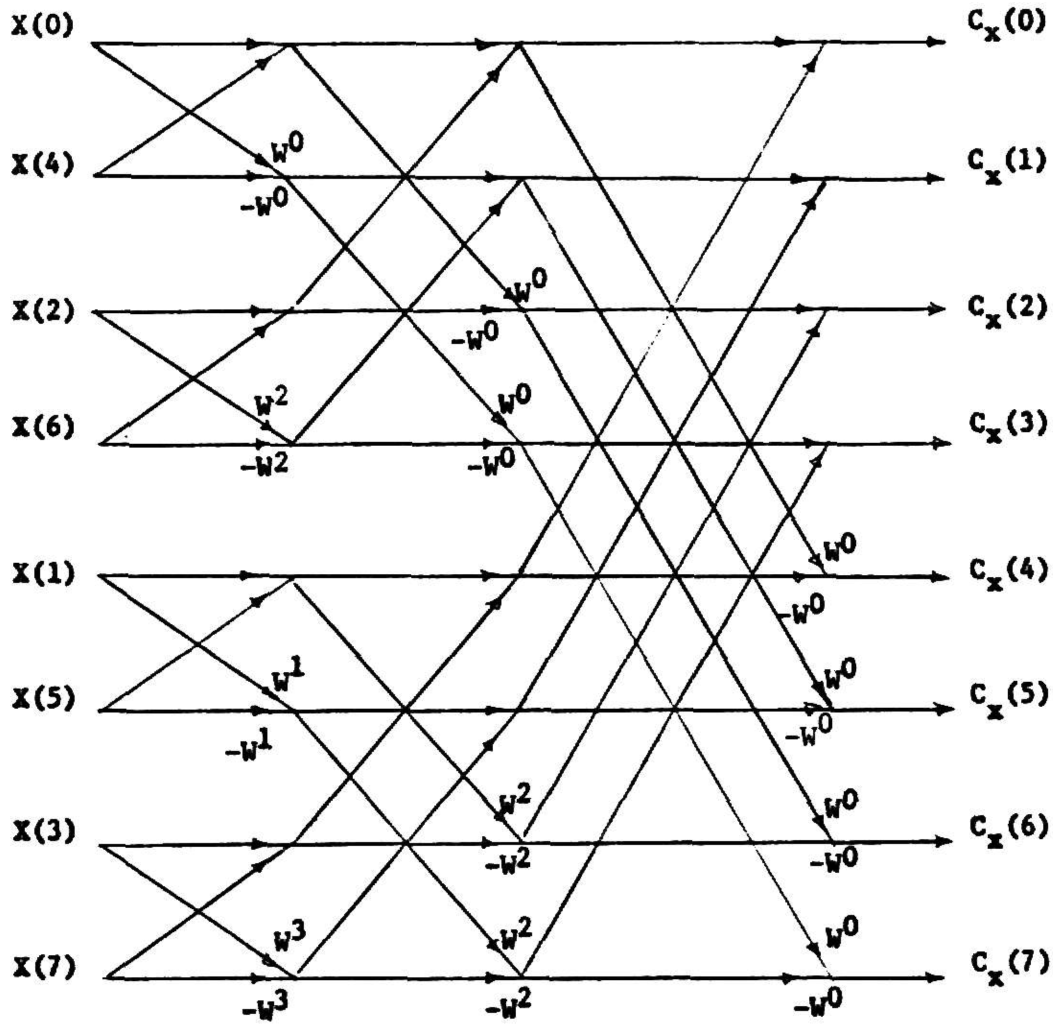


Fig. 2.12. 8-point DFT with shuffled output.

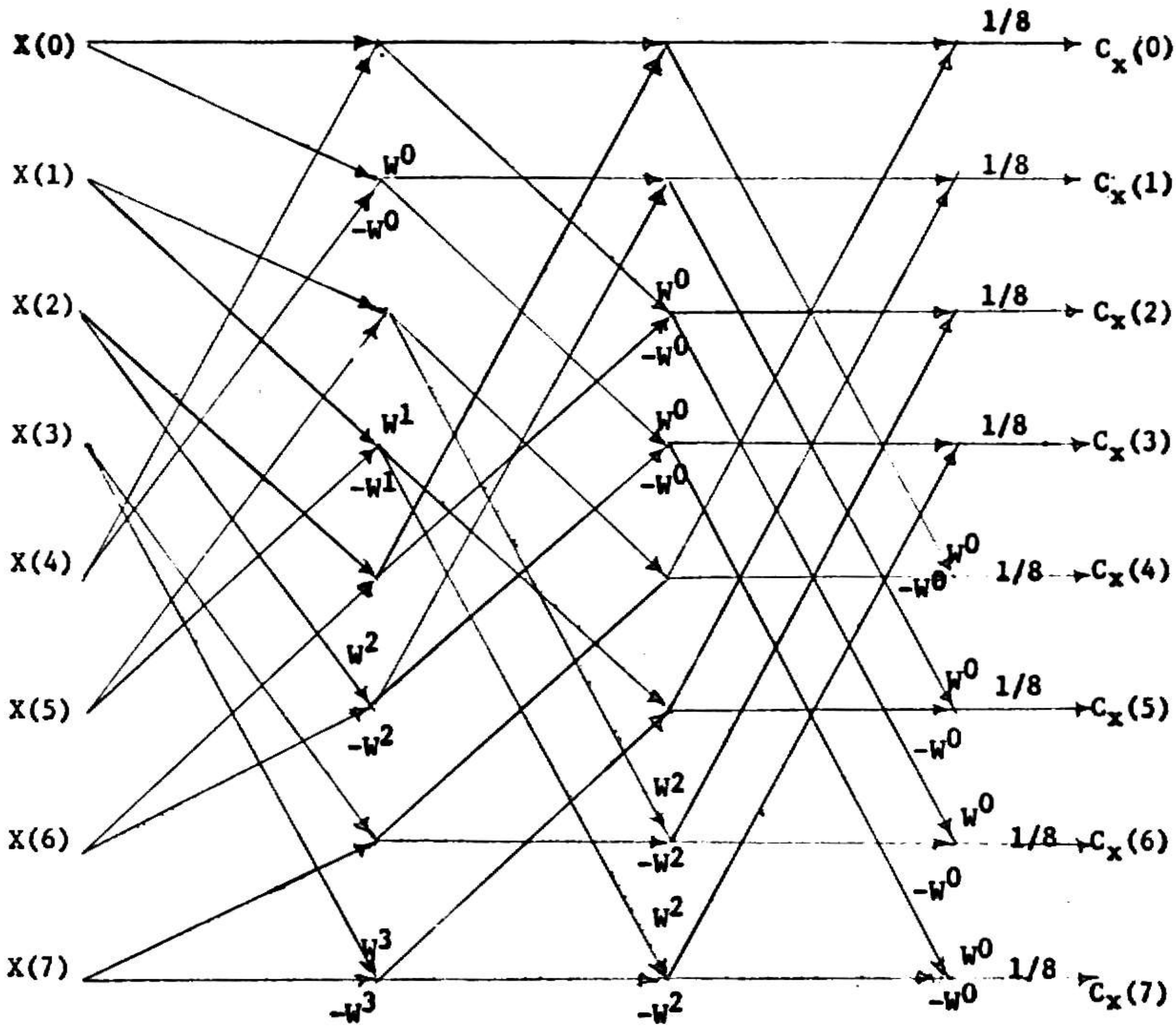


Fig. 2.13. 8-point DFT with input and output in natural order.

$$C_x(k_2,k_1,k_0) = \frac{1}{N} \sum_{m_0=0}^{1} \sum_{m_1=0}^{1} \sum_{m_2=0}^{1} X(m_2,m_1,m_0)\, W^{(4k_2+2k_1+k_0)(4m_2+2m_1+m_0)}$$

..... (2.33)

We can obtain the Sande-Tukey algorithm by separating the components of k instead of m; that is

$$W^{(4k_2+2k_1+k_0)(4m_2+2m_1+m_0)}$$

$$= W^{(4m_2+2m_1+m_0)\,k_0}\; W^{(4m_2+2m_1+m_0)\,2k_1}\; W^{(4m_2+2m_1+m_0)\,4k_2}$$

..... (2.34)

Consider each of the factors on the right hand side of the Eq. (2.34) in turn as follows:

$$W^{(4m_2+2m_1+m_0)\,k_0} = W^{(4m_2+2m_1+m_0)\,k_0} \qquad (2.35)$$

$$W^{(4m_2+2m_1+m_0)\,2k_1} = [W^{8(m_2k_1)}]\; W^{(2m_1+m_0)\,2k_1} \qquad (2.36)$$

$$W^{(4m_2+2m_1+m_0)\,4k_2} = [W^{8(2m_2+k_1)}]\; W^{4m_0k_2} \qquad (2.37)$$

Since the bracketed portions of Eqs. (2.37) and (2.38) can be replaced by unity, Eq. (2.33) can be written in the form

$$C_x(k_2,k_1,k_0)$$

$$= \frac{1}{8} \sum_{m_0=0}^{1} \sum_{m_1=0}^{1} \Big[\sum_{m_2=0}^{1} X(m_2,m_1,m_0)\, W^{(4m_2+2m_1+m_0)} \Big]\; W^{(2m_1+m_0)\,2k_1}\; W^{4m_0k_2}$$

..... (2.38)

The counterparts of Eqs. (2.23), (2.25) and (2.27) for the Sande-Tukey algorithm are as follows:

$$X_1\ (k_0,m_1,m_0) = \sum_{m_2=0}^{1} X(m_2,m_1,m_0)\ W^{(4m_2+2m_1+m_0)} \tag{2.39}$$

$$X_2\ (k_0,k_1,m_0) = \sum_{m_1=0}^{1} X_1(k_0,m_1,m_0)\ W^{(2m_1+m_0)\ 2k_1} \tag{2.40}$$

$$X_3\ (k_0,k_1,k_2) = \sum_{m_1=0}^{1} X_2(k_0,k_1,m_0)\ W^{4m_0k_2} \tag{2.41}$$

Substituting Eqs. (2.39, (2.40) and (2.41) in Eq. (2.33), it follows

$$C_x\ (k_2,k_1,k_0) = \frac{1}{8} X_3(k_0,k_1,k_2) \tag{2.42}$$

Eq. (2.42) gives the desired DFT. The signal flow graph that results from Eqs. (2.39), (2.40) and (2.41) is shown in Fig. 2.14. This flow graph represents the Sande-Tukey algorithm for N = 8.

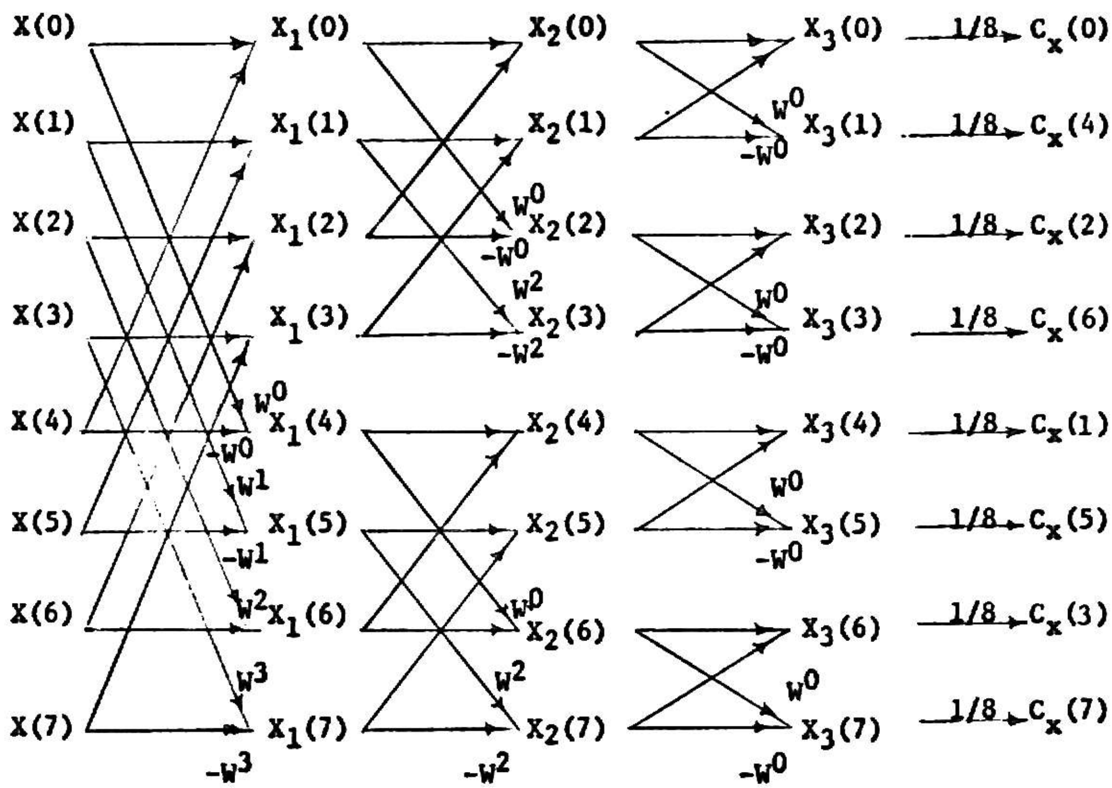


Fig. 2.14. Sande -Tukey Algorithm for N=8.

CHAPTER III

FFT PRUNING

3.1 Time Domain FFT Pruning

If we have 2^L nonzero data points out of 2^M data points, where $M > L$, then the corresponding FFT can be computed with considerable time savings by means of the pruned FFT. The zero values generally append a given data sequence. The reason for doing so is to realize an increase in frequency resolution. Operations involving zeros are either eliminated or reduced. According to Markel [10], the time saved is approximately equal to $M [L + 2 (1 - 2^{(M-L)}]^{-1}$ for radix 2. Pruning can be done for radices other than 2. In this study we restrict our attention to radix 2.

The flow graph given in Fig. 2.14 for Sande's algorithm is modified as shown in Fig. 3.1. The term $W^{(\ )}$ is sometimes referred to as a 'twiddle factor'.

FFT pruning corresponds to eliminating operations that do not contribute to the output. A flow graph for time domain FFT pruning is shown in Fig. 3.2 for $L = 1$, $M = 3$. There are two non-zero data points and three stages. Pruning is applied to first two stages. But third stage cannot be pruned. When pruning is applicable, we compute only partial butterflies instead of entire butterflies, as illustrated in Figs. 3.3 and 3.4. In general, if there are 2^L non-zero data points in a set of 2^M data points, then the number of stages where pruning can be applied equals (M-L); conversely the number of stage(s) where pruning cannot be employed equals L. For non-zero data points located at some arbitrary location in the time domain, the discrete Fourier shifting theorm can be applied before using the pruned FFT. A FORTRAN program implementation in the time domain FFT pruning is listed in Appendix I.

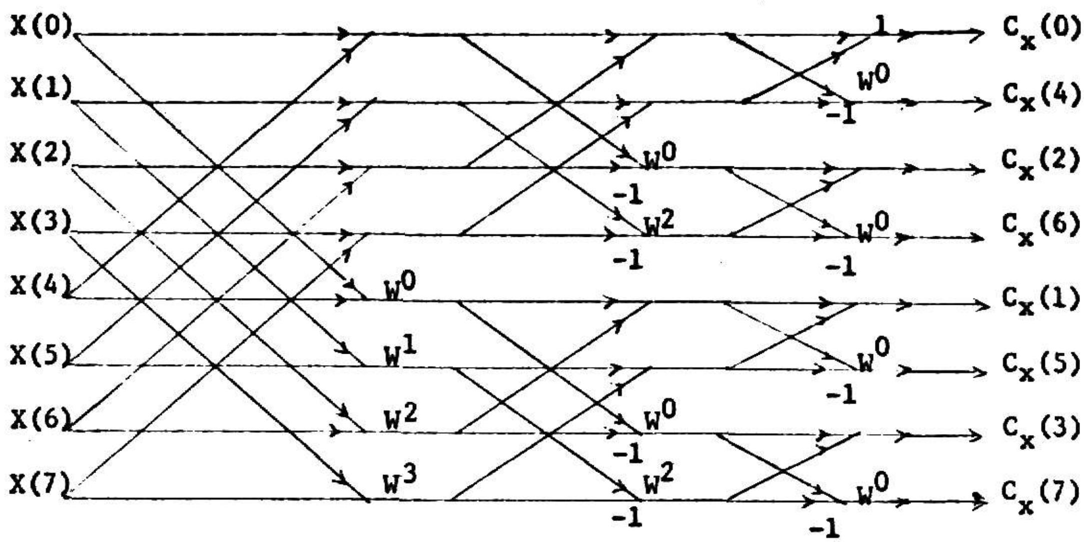


Fig. 3.1. 8-point decimation in frequency FFT.

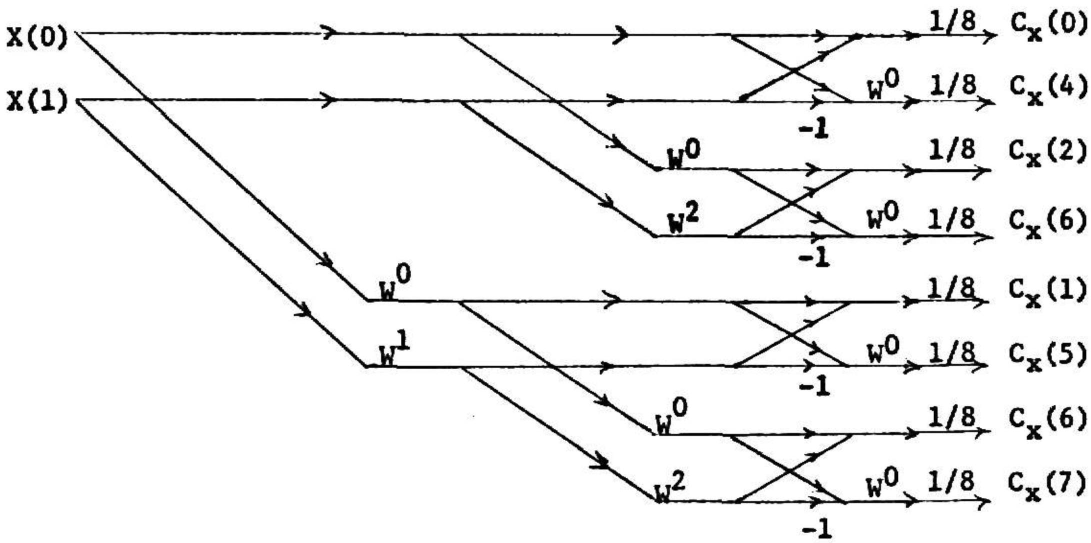


Fig. 3.2. 8-point time pruned FFT

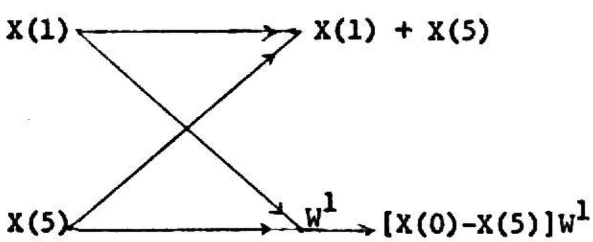


Fig. 3.3. A butterfly.

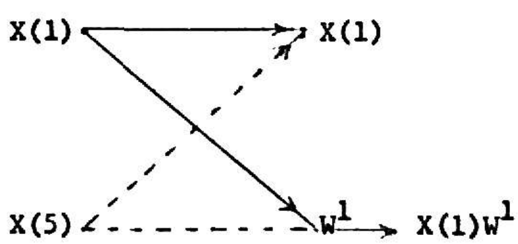


Fig. 3.4. A partial butterfly.

FFT pruning can also be extended to process two-dimensional data. The DFT of a two-dimensional array $[X(m_1,m_2)]$ consisting of N_1 rows and N_2 columns is given by

$$C_x\ (k_1,k_2) = \sum_{m_1=0}^{N_1-1} \{\sum_{m_2=0}^{N_2-1} X\ (m_1,m_2)\ W_1^{k_2 m_2}\}\ W_2^{k_1 m_1}\ ,$$

$$k_i = 0,\ 1,\ldots,\ (N_i-1),\ i = 1.2. \qquad \ldots\ldots(3.1)$$

where

$$W_1 = e^{\frac{-i2\pi}{N_1}}\ ,\ W_2 = e^{\frac{-i2\pi}{N_2}}$$

The innermost sum in Eq. (3.1) can be written as

$$Y_k\ (m_1,k_2) = \sum_{m_2=0}^{N_2-1} X\ (m_1,m_2)\ W^{k_2 m_2} \qquad (3.2)$$

Combining Eqs. (3.1) and (3.2) one obtains

$$C_x\ (k_1,k_2) = \sum_{m_1=0}^{N_1-1} Y_k\ (m_1,k_2)\ W^{k_1 m_1} \qquad (3.3)$$

From Eq. (3.2) it follows that the $(N_1 \times N_2)$ array $[Y_k(m_1,k_2)]$ is computing the FFT of each column of $[X(m_1,m_2)]$. Again Eq. (3.3) implies that the $C_x(k_1,k_2)$ are then obtained by computing the FFT of each row of $[Y_k(m_1,m_2)]$. Hence a two-dimensional FFT can be computed by N_1N_2 applications of a one-dimensional FFT. Consequently the advantages of FFT pruning carry over to the case of two-dimensional processing also.

3.2 Frequency Domain FFT Pruning

This is a converse of time domain pruning. In certain applications 2^M input data points are given, and 2^L output points are desired where $L < M$. Pruning can be efficiently employed to the decimation in time algorithm by applying Sande's innerloop nesting procedure to obtain pruning in the frequency domain.

Consider the flow graph shown Fig. 2.4, which can be modified using the relations

$$W^{N/2} = -W^0$$

and

$$W^k = -W^{(k-N/2)}, \text{ for } N > k > N/2$$

where $W = \exp(-i2\pi/N)$.

The modified signal flow graph that results is shown in Fig. 3.5. To prune in the frequency domain, we eliminate the operations which do not contribute to the required output. The flow graph for the frequency domain FFT pruning which is shown in Fig. 3.6 is self explanatory. Partial butterflies are computed rather than complete butterflies. As in the case of time pruning, there are $(M - L)$ stages that can be pruned. The flow graph in Fig. 3.6 corresponds the case $M = 3$, and $L = 1$. In general, if 2^L frequency samples are calculated at some arbitrary location in the frequency domain, the discrete Fourier shift theorem is applied to the input data prior to using the pruned FFT.

FFT pruning in the frequency domain can be used effectively in narrow band spectral analysis where a small number of DFT coefficients are required relative to a data sequence consisting of N data points (i.e., $L \ll M$). A

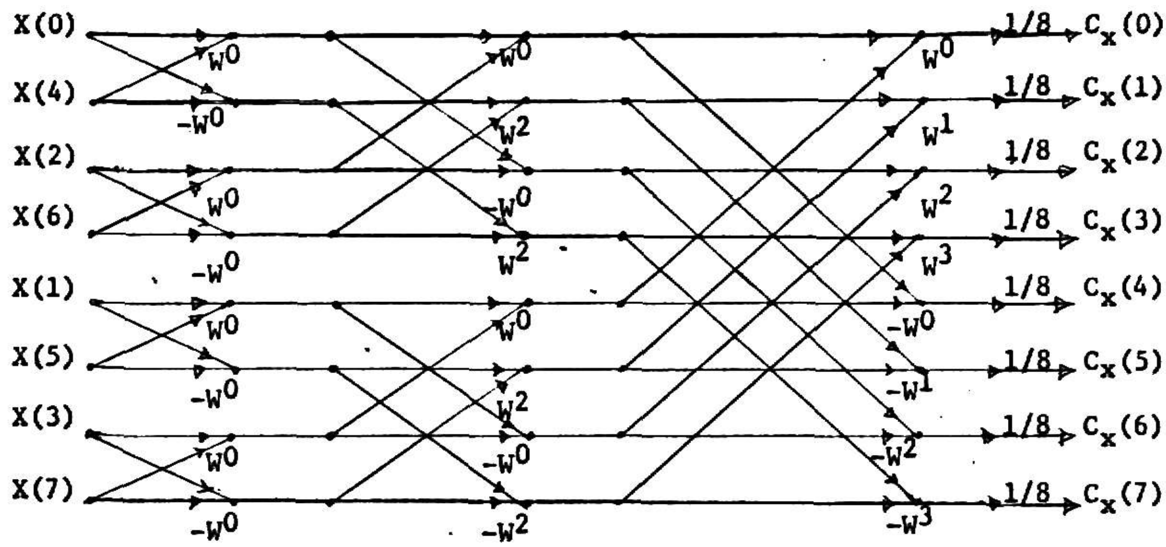


Fig. 3.5. 8-point decimation-in-time FFT.

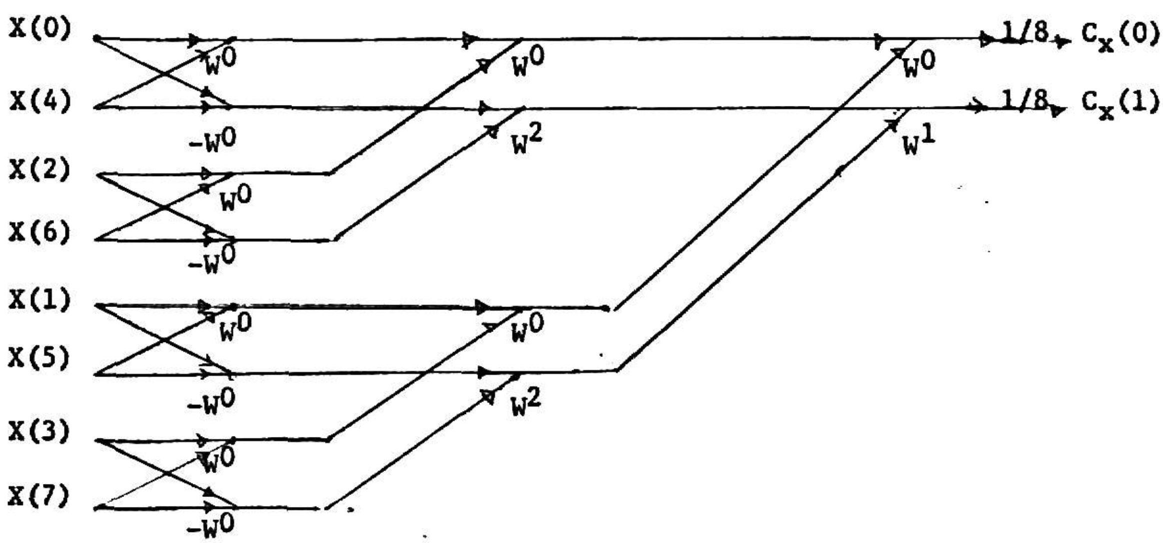


Fig. 3.6. 8-point frequency pruned FFT.

FORTRAN program listing for the frequency domain FFT pruning is given in Appendix II.

As in the case of time domain FFT pruning, pruning in the frequency domain can also be extended to two-dimensional applications.

An important application of the pruned FFT is narrowband spectral analysis. Such analyses can also be achieved via the chirp Z-transform, a brief discussion of which follows. The performances of the chirp Z-trnasform and the pruned FFT will the be compared with respect to narrow band analysis.

3.3 Chirp Z-Transform

The Chirp Z-transform (CZT) algorithm was developed by Rabiner, Shafer, and Rader [12]. The CZT algorithm can be used to compute the DFT. Let $\{X(m)\} = \{X_0, X_1, \ldots X_{N-1}\}$ denote a data sequence, which is obtained by sampling a band limited signal x(t), at the rate of 1/T samples/second. Then the CZT of $\{X(m)\}$ is defined as

$$X_k = X(Z_k) = \sum_{n=0}^{N-1} X_n Z_k^{-n}, \; k = 0, 1, \ldots (M-1) \tag{3.4}$$

where

$$Z_k = (A_0 e^{i2\pi\theta_0})(W_0 e^{i2\pi\phi_0})^{-k}, \tag{3.5}$$

Equation (3.5) describes a set of M points equally spaced on logarithmic spiral. Since the transformation from the Z-plane to the S-plane is given by

$$S = (1/T) \ln Z,$$

it follows that

$$S_k = (1/T)(\ln A_0 + i2\pi\theta_0) - (K/T)(\ln W_0 + i2\pi\phi_0) \tag{3.6}$$

The geometrical interpretation of Eq. (3.6) is illustrated in Fig. 3.7 for a contour of 8 points. In Fig. 3.7 we observe that the Z-plane contour maps into a straight line of arbitrary length and orientation in the S-plane.

As we are interested only in real values of frequency (i.e., operating on the imaginary axis), Eq. (3.6) can be simplified by substituting $A_0 = W_0 = 1$. That is

$$S_k = \frac{i}{T} 2\pi\theta_0 - \frac{i2\pi\phi_0}{T} \tag{3.7}$$

Equation (3.7) implies that the points on the imaginary axis in the S-plane are mapped on to a unit circle in the Z-plane. Equations (3.4) and (3.5) yield

$$X_k = \sum_{n=0}^{N-1} x_n A^{-n} W^{kn}, \; k = 0, 1, \ldots, \quad (M-1) \tag{3.8}$$

Where $A = e^{i2\pi\theta_0}$

$W = e^{i2\pi\phi_0}$

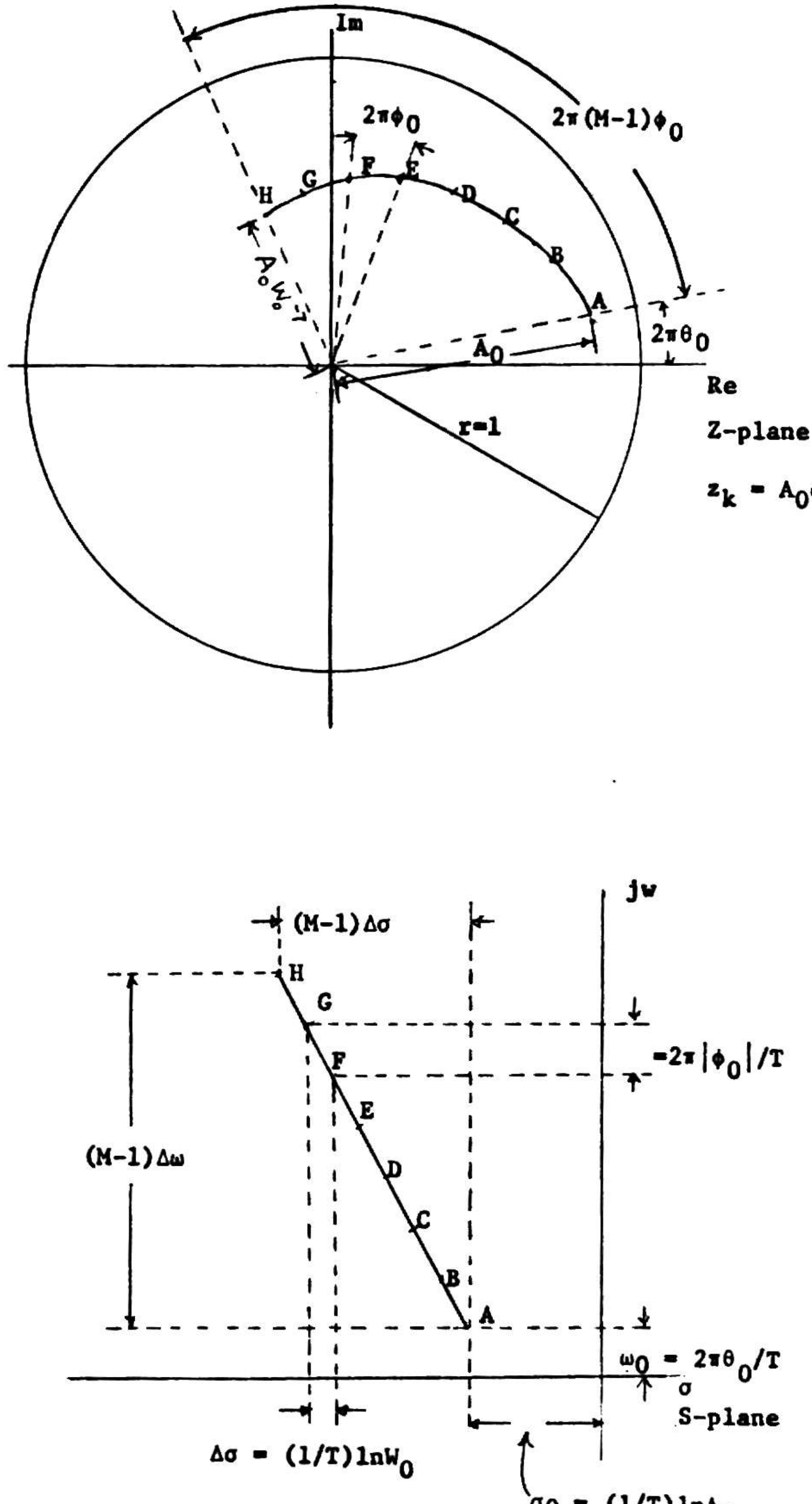
Equation (3.8) is referred to as the "modified" CZT (MCZT).

Consider the case when $\theta_0 = 0$, $M = N$ and $\phi_0 = -1/N$. Then Eq. (3.8) yields

$$X_k = \sum_{n=0}^{N-1} x_n W^{kn}, \; k = 0, 1, \ldots . (N-1) \tag{3.9}$$

where $W = e^{-i2\pi/N}$. From this we conclude that the DFT is a special case of MCZT.

<u>DFT computation using the MCZT</u>:- The algorithm is summerized as follows [14]:
(1) Choose L, the smallest integer which is a power of 2 and is greater than or equal to (M + N - 1).



$$z_k = A_0 e^{i2\pi\theta_0} \left(W_0 e^{i2\pi\phi_0}\right)^{-k}$$

$$k = 0,1,..N-1.$$

$$S_k = \left(\frac{1}{T}\ln A_0 + \frac{i2\pi\theta_0}{T}\right) - k\left(\frac{1}{T}\ln W_0 + \frac{i2\pi\phi_0}{T}\right)$$

Fig. 3.7. A general CZT contour with M=8.

(2) Form an L point sequence y_n from by weighting X_n according to

$$y_n = \begin{cases} A_n^{-n} W^{n^2/2} X_n , & n = 0, 1, \ldots, (N-1) \\ 0 , & n = N, (N+1), \ldots, (L-1) \end{cases} \quad (3.10)$$

(3) Compute the DFT of y_n using an efficient FFT algorithm, and denote it by Y_r, $r = 0, 1, \ldots, (L-1)$.

(4) Form an L point sequence v_n according to

$$V_n \quad \begin{cases} W^{-n^2/2} & , \ 0 < n < (M-1) \\ 0 & , \ (M-1) < n < (L-n+1), \text{ if } L > (M+N-1) \\ W^{-(L-n)^2/2} & (L-N+1) < n < (L-1) \end{cases} \quad (3.11)$$

From Eq. (3.11) it is clear that if $L = M-N+1$, there are no terms in V_n which equal zero.

(5) Compute the DFT of v_n and denote it by V_r, $r = 0, 1, \ldots (L-1)$.

(6) Compute the product sequence

$$G_r = Y_r V_r , \ r = 0, 1, \ldots, (L-1) \quad (3.12)$$

(7) Compute the IDFT of G_r and denote it by g_k, $k = 0, 1, \ldots, (L-1)$.

(8) Compute $X_k = A^{-k^2/2} g_k$, $k = 0, 1, \ldots, (M-1)$. Then X_k, $k = 0, 1, \ldots,$ $(M-1)$ are the desired MCZT coefficients.

Further Computational Considerations: Consider the situation when the data sequence X_n, $n = 0, 1, \ldots, (N-1)$ extremely long and we desire M, DFT coefficients where M<<N. Then using the MCZT algorithm, 3 FFT's of length L have to be computed, where L is the smallest integer power of 2 that is greater

than or equal to (M+N-1). However, it is plausible that L may be so large that storage requirements prohibit computation of the MCZT. In such cases, the sum in Eq. (3.8) can be broken in to R sums over the N points. That is, the original data sequence is divided into R partitions and hence Eq. (3.8) can be written as follows:

$$X_k = \sum_{r=0}^{R-1} A^{-r\hat{N}} W^{kr\hat{N}} \left[\sum_{n=0}^{\hat{N}-1} x_{n+r\hat{N}} A^{-n} W^{nk} \right], \quad k = 0, 1, \ldots, (M-1) \qquad \ldots\ldots (3.13)$$

Each of the R sums in square parenthesis of Eq. (3.13) can then be evaluated using the MCZT algorithm. Equation (3.13) is sometimes referred to as the partitioned MCZT and abbreviated as PAM-CZT [6]. This procedure would require storage of the order of $3(\hat{N}+M-1)$ locations.

REMARKS:

The MCZT algorithm has greater flexibility in that neither N or M need be integer power of 2. Again, more storage locations (3L) are required, and the FFT and IFFT are used twice and once respectively. The MCZT can be used for high resolution narrow band spectral analysis. An example with a FORTRAN program implementation is given in Appendix III. Alternately FFT with pruning can be used effectively for narrow band analysis. Execution times for the MCZT and the FFT with pruning are compared in Fig. 3.8 for data sequence lengths up to 64 to achieve a 4:1 increase in resolution in the 2 Hz to 3 Hz range. From Fig. 3.8 it follows that the FFT pruning is substantially faster than the MCZT. There are also some limitations in using the FFT with pruning. First, the increase in resolution is restricted to the form 2^k:1. The lower frequency of the desired bandwidth must be of the form $(\ell + \frac{1}{2} k)$, where k and ℓ are integers.

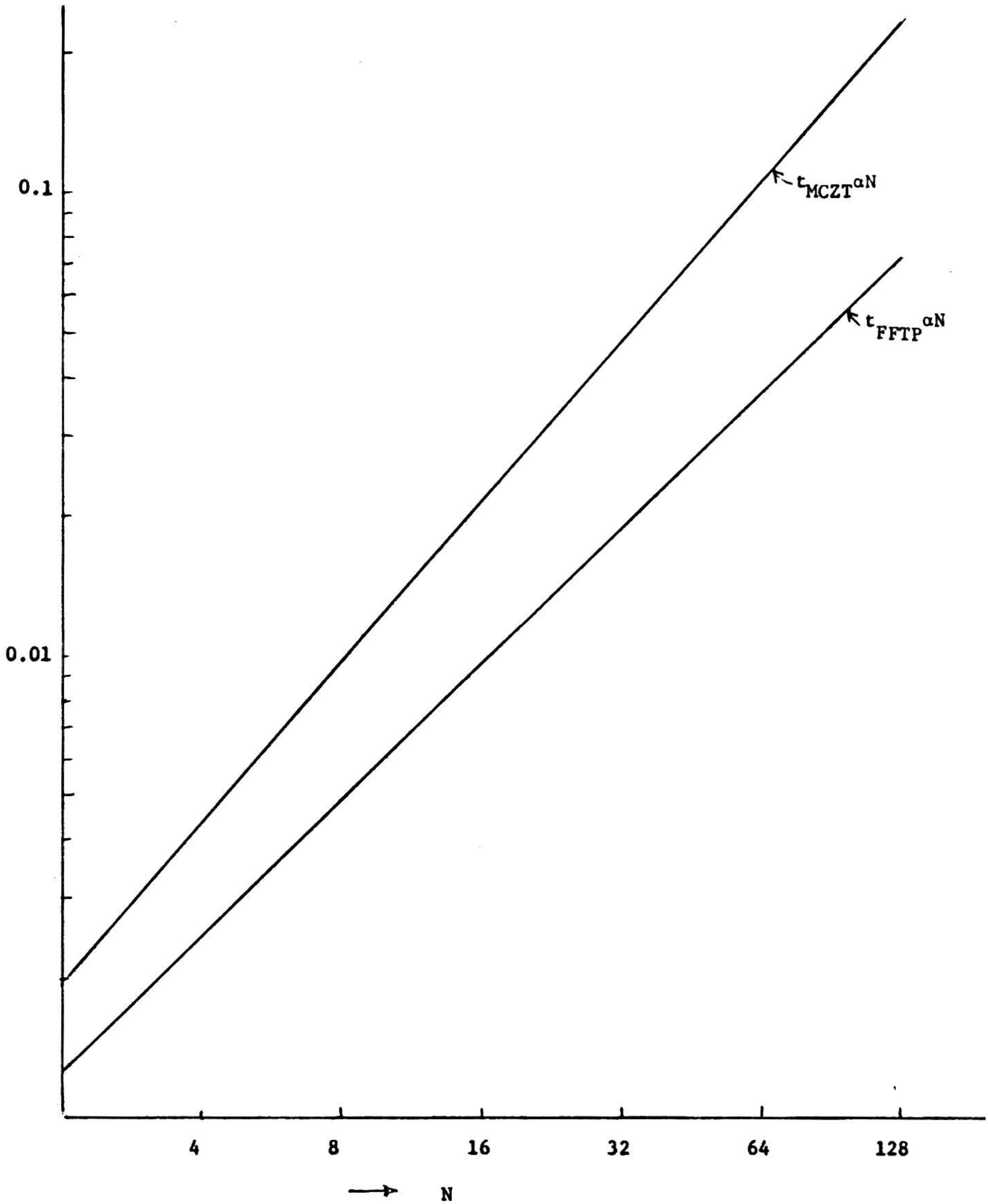


Fig. 3.8. Narrow band analysis comparison for the pruned FFT and MCZT.

CHAPTER IV

APPLICATION CONSIDERATIONS

4.1 Formant Analysis of Speech

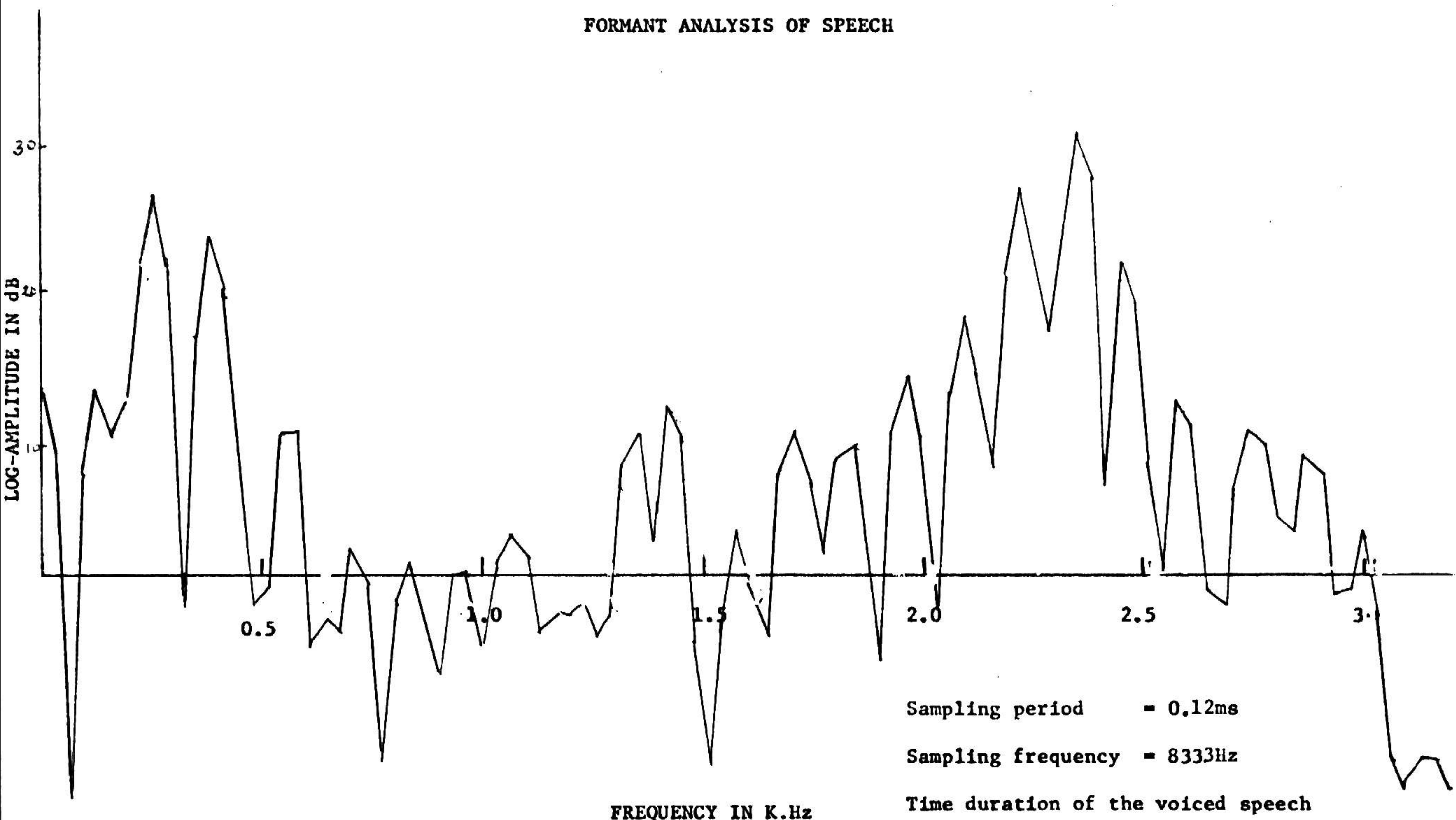
Formants of the voiced speech can be analysed using the FFT with pruning. The log-magnitude of the Fourier transform of a segment of voiced speech is shown in Fig. 4.1. The log-magnitude spectrum of a voiced speech is composed of two components: (1) a rapidly varying periodic component associated with the vocal cord excitation, and (2) a slowly varying component associated with the formant frequencies. The slowly varying component has to be separated to estimate the parameter values of the formant frequencies. The standard approach to this problem is linear filtering. One technique for achieving this filtering is through the "cepstrum". The cepstrum is defined as the inverse Fourier transform of the log-magnitude spectrum.

The cepstrum corresponding to the log-magnitude spectrum in Fig. 4.1, is shown in Fig. 4.2.* In Fig. 4.2 we observe that the rapidly varying component corresponds to the cepstral peak which occurs at about 7.560 ms, while the slowly varying component correspond to the low-time portion of the cepstrum. The slowly varying component can be extracted from the cepstrum by truncating the cepstrum values to zero at about 3.84 ms, and then computing the inverse FFT with pruning. This results in a smoothed spectrum shown in Fig. 4.3. For the purposes of illustration, the smoothed spectrum is superimposed on the corresponding log-magnitude spectrum. In Fig. 4.3 the formant frequencies correspond to the peaks in the smoothened spectrum.

---

*The speech data used is part of that collected for a joint study (of deaf speech) between the Departments of Speech and Electrical Engineering.

FORMANT ANALYSIS OF SPEECH



Sampling period = 0.12ms

Sampling frequency = 8333Hz

Time duration of the voiced speech

= 30.72ms

Number of samples = 256

Fig. 4.1. Log-magnitude Spectrum.

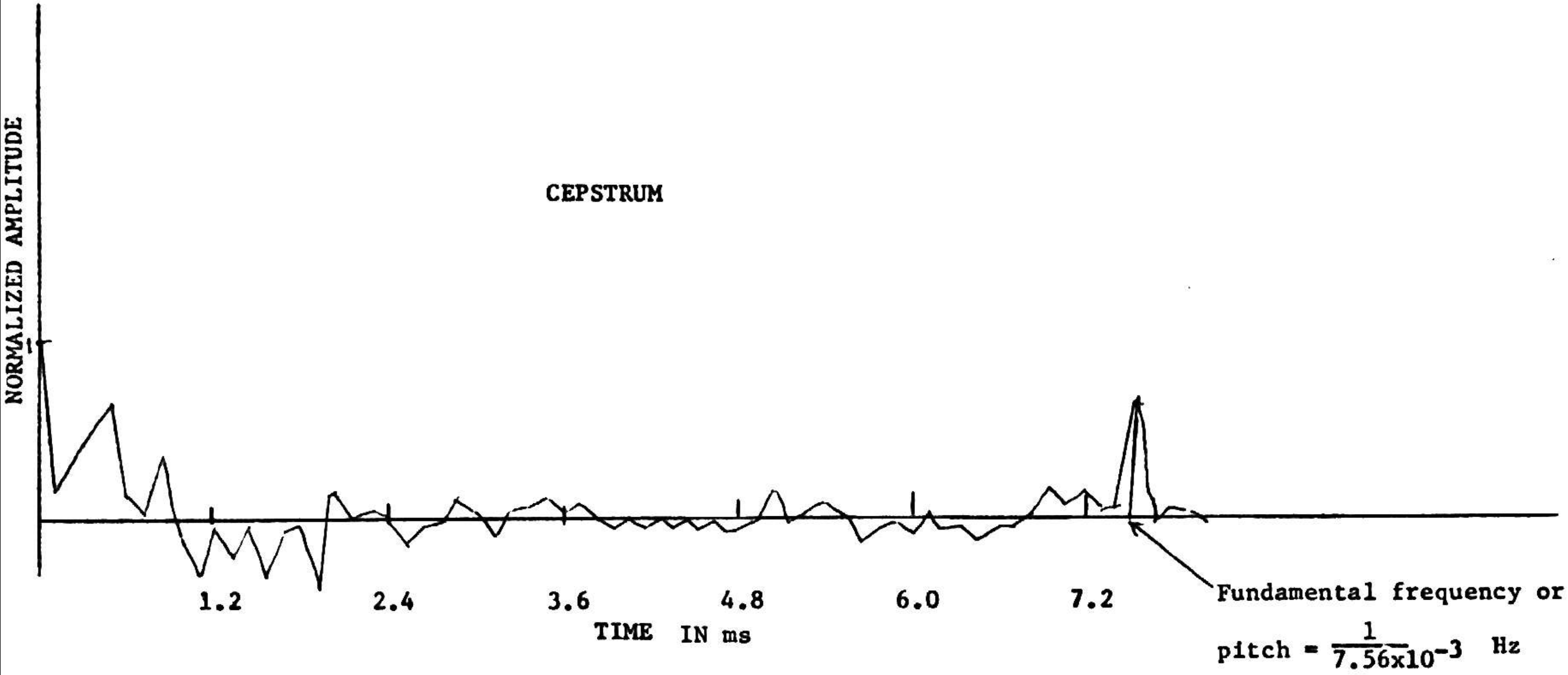


Fig. 4.2. Cepstrum corresponding to the log-magnitude spectrum in Fig. 4.1.

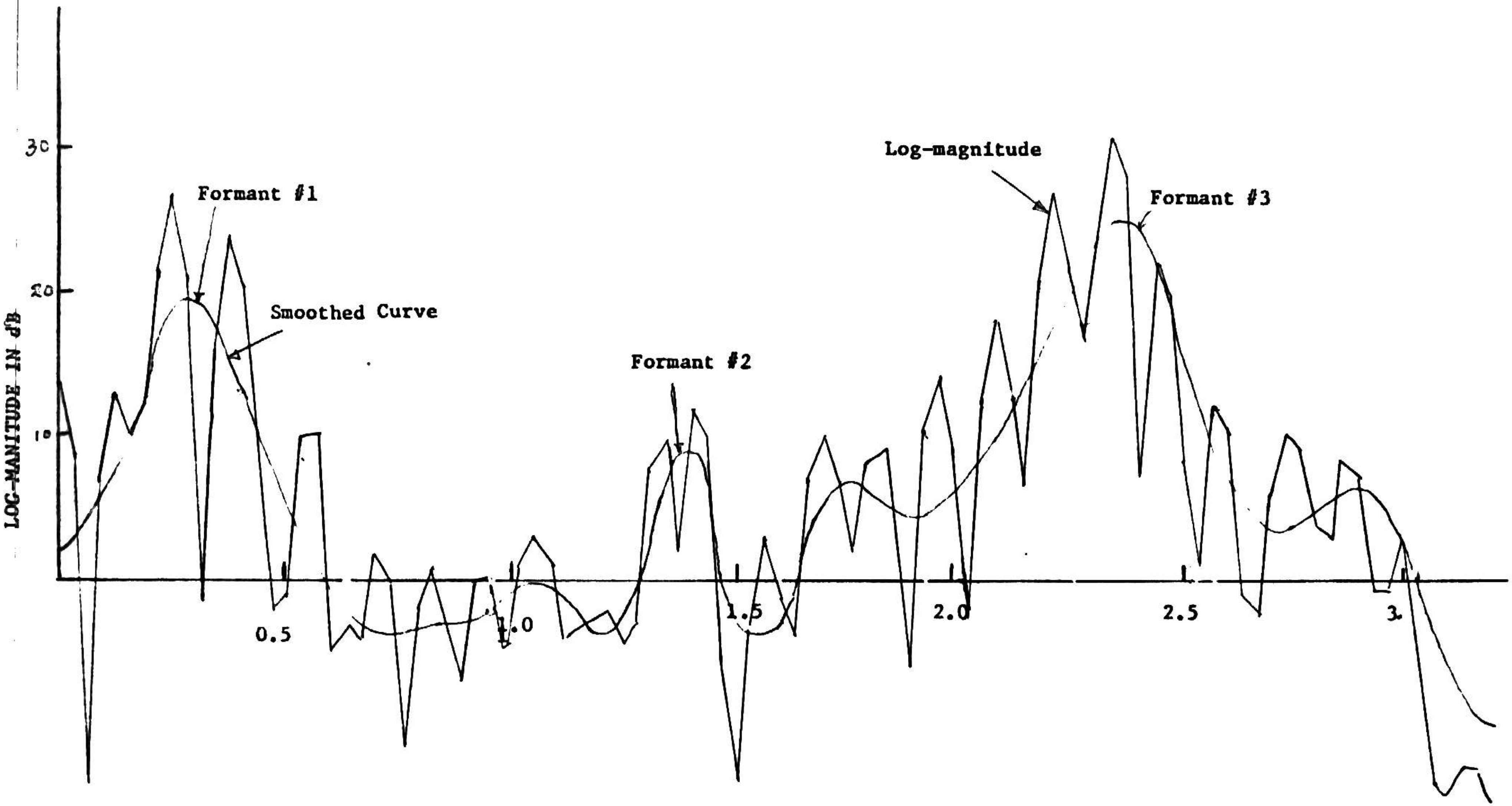


Fig. 4.3. Smoothed Log-mag Spectrum.

Let $\{Y(m)\}$ denote a data sequence $Y(m)$, $m = 0, 1, \ldots, (N-1)$ obtained from a short segment of a speech signal $y(t)$. The steps involved to secure analysis can be summarized as follows:

(1) Multiply $\{Y(m)\}$ by a window sequence $\{W(m)\}$ in order to minimize the undesirable effects introduced as a consequence of the Fourier analysis of a finite length data sequence. That is

$$X(m) = \sum_{m=0}^{N-1} Y(m)\, W(m), \quad m = 0, 1, \ldots, (N-1)$$

where

$$W(m) = \tfrac{1}{2}\left[1 - \cos\left(\frac{2\pi m}{N}\right)\right],$$

which is usually referred to as the Hanning window.

(2) Compute the DFT of $\{X(m)\}$ and denote it by $C_x(k)$, $k = 0, 1, \ldots, (N-1)$.

(3) Compute the log-magnitude of $C_x(k)$ and call it $L_x(k)$, $k=0, 1, \ldots, (N-1)$; (see Fig. 4.1).

(4) Compute IDFT of $\{L_x(k)\}$ and call it $C_L(k)$, $k=0, \ldots, (N-1)$. This is the cepstrum of a segment of speech; (see Fig. 4.2).

(5) Low-time filter the cepstral values, that is,

$$C_L(k) = \begin{cases} C_L(k) = C_L(0) & , k = 0 \\ C_L(k) = 2C_L(k) & , k = 1, \ldots, (N' - 1) \\ C_L(k) = 0 \quad , & N' < k < (N-1) \end{cases}$$

Where N' is an integer, which is less than or equal to the first half of the cepstral values between two peaks.

(6) Compute the DFT of $\{C_L(k)\}$ using the FFT with pruning. The real values of the DFT are the required values to obtain the smoothed log-magnitude spectrum.

The preceding steps are summarized in the block diagram of Fig. 4.4. Pruning is used in the low time filtering stage since the sequence C_L (k), k = 0, 1, ..., (N-1) generally consists of a large number of zeros; [see step (5) above].

4.2 High Speed Autocorrelation

FFT with pruning can also be used to compute the autocorrelation function of a data sequence. Some aspects of this are discussed in this section, using an on-line method which was recently proposed by Rader [13].

Let x(n), n = 0, 1, 2, ... be the given data sequence which may be of indifinite length. We will follow the convention that n is a time index, k is a frequency index, and m is a lag index. The upper case letters W, X, Z denote DFT's of w, x, and z respectively.

The desired auto correlation function is defined as

$$R_x(m) = \frac{1}{N} \sum_{n=0} x^*(n)\, x(n+m), \quad m = 0, 1, \ldots, M \qquad (4.1)$$

where $x^*(n)$ denotes the complex conjugate of x(n).

The given sequence is divided into blocks as illustrated in Fig. 4.5. Each block consists of M/2 data points. This process results in a set of subsequences. If the i^{th} subsequence is denoted by $\{x_i(n)\}$, then it is constructed as follows [See Fig. 4.5]:

$$x_i(n) = \begin{cases} x\,(n + i\, M/2) & 0 \leq n\, M/2 \\ 0 & M/2 \leq n \leq M, \\ & i = 0, 1, 2, \ldots. \end{cases} \qquad (4.2)$$

Let us form a second series of sequence, $\{Y_i(n)\}$, (See Fig. 4.5).

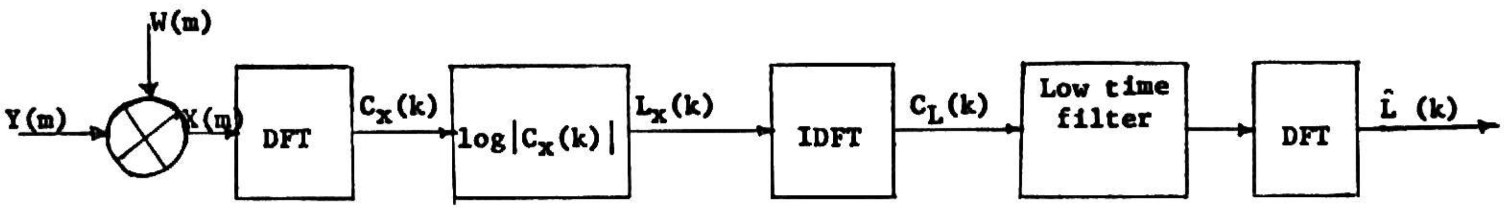


Fig. 4.4. Block Diagram.

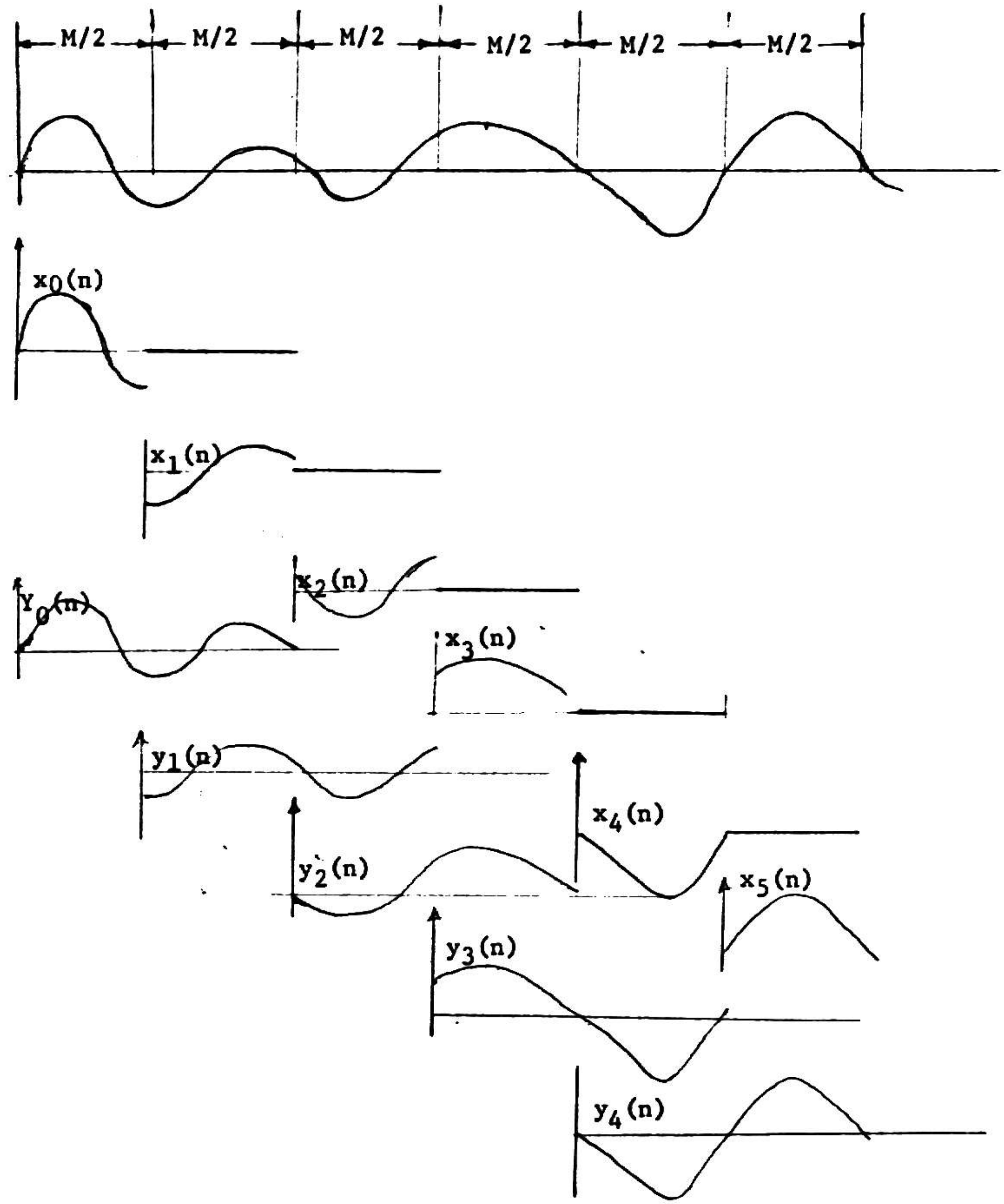


Fig. 4.5. Segmentation of given signal into blocks.

such that

$$y_i(n) = \begin{cases} x(n + i\,M/2) & 0 \leq n \leq M \\ & i = 0,\ 1,\ 2 \ldots\ldots \end{cases} \tag{4.3}$$

The y_i sequences are formed only as a pedagogical device.

The DFT's of $\{x_i(n)\}$ and $\{y_i(n)\}$ are given by

$$X_i(k) = \text{DFT}\ \{x_i(n)\} \tag{4.4}$$

$$Y_i(k) = \text{DFT}\ \{y_i(n)\} \tag{4.5}$$

Equations (4.4) and (4.5) are used to form the product

$$W_i(k) = X_i^*(k)\ Y_i(k) \tag{4.6}$$

The DFT's of the sequence $\{w_i(n)\}$ is denoted by $\{W_i(k)\}$, where the sequence $\{w_i(n)\}$ is given by

$$w_i(m) = \sum_{n=0}^{(M/2-1)} x^*\ (n+iM/2)\ x(n+iM/2+m),\ m = 0,\ 1,\ \ldots,\ M/2 \qquad \ldots\ldots(4.7).$$

Except for the factor $1/N$, we can obtain correlation by summing $w_i(m)$. Let us define $Z_i(m)$ such that

$$z_i(m) = \sum_{j=0}^{i} w_j(m)$$

$$= \sum_{j=0}^{i} \sum_{n=0}^{(M/2-1)} x^*(n+jM/2)\ x(n+jM/2+m)$$

then

$$Z_i(m) = \sum_{n=0}^{(i+1)M/2-1} x^*(n)\ x(n+m) \tag{4.8}$$

When $i = (2N/M) - 1$, $z_i(m)$ is $N \times R_x(m)$.

The sum in Eq. (4.8) can be carried out in the frequency domain. It follows

$$Z_i(k) = \sum_{j=0}^{i} W_j(k) = Z_{i-1}(k) + W_i(k) \tag{4.9}$$

The computation of $Y_i(k)$ can be made without use of $\{y_i(n)\}$. It can be shown that [13]

$$Y_i(k) = X_i(k) + (-1)^k X_i + 1^{(k)} \tag{4.10}$$

The on-line computational procedure may be summarized as follows:

(1) Form the sequence $\{x_0(n)\}$ and compute $X_0(k)$ using the FFT with pruning. Clear out $Z_0(k)$.

(2) For $i = 0, 1, \ldots, (2\,N/M) - 2$,

a) form $\{x_{i+1}(n)\}$ and compute $X_{i+1}(k)$ using FFT with pruning;

b) compute

$$Z_{i+1}(k) = Z_i(k) + X_i{}^*(k)\ [X_i(k) + (-1)^k X_{i+1}(k)];$$

(3) Compute

$$R_x(m) = (\tfrac{1}{N}) \text{ IFFT } \{Z_{(2N/M)-1}(k)\}$$

keep only the first $(M/2+1)$ values.

The step (3) gives the desired autocorrelation. The FFT with pruning which is used in steps (1) and (2) saves considerable amount of computational time. As M increases, the compution of the autocorrelation function in Eq. (4.1) using the FFT with pruning becomes more economical. This is illustrated in Fig. 4.6, where t_{FFT} and t_{FFTP} denote the execution times associated with regular FFT and FFT with pruning respectively.

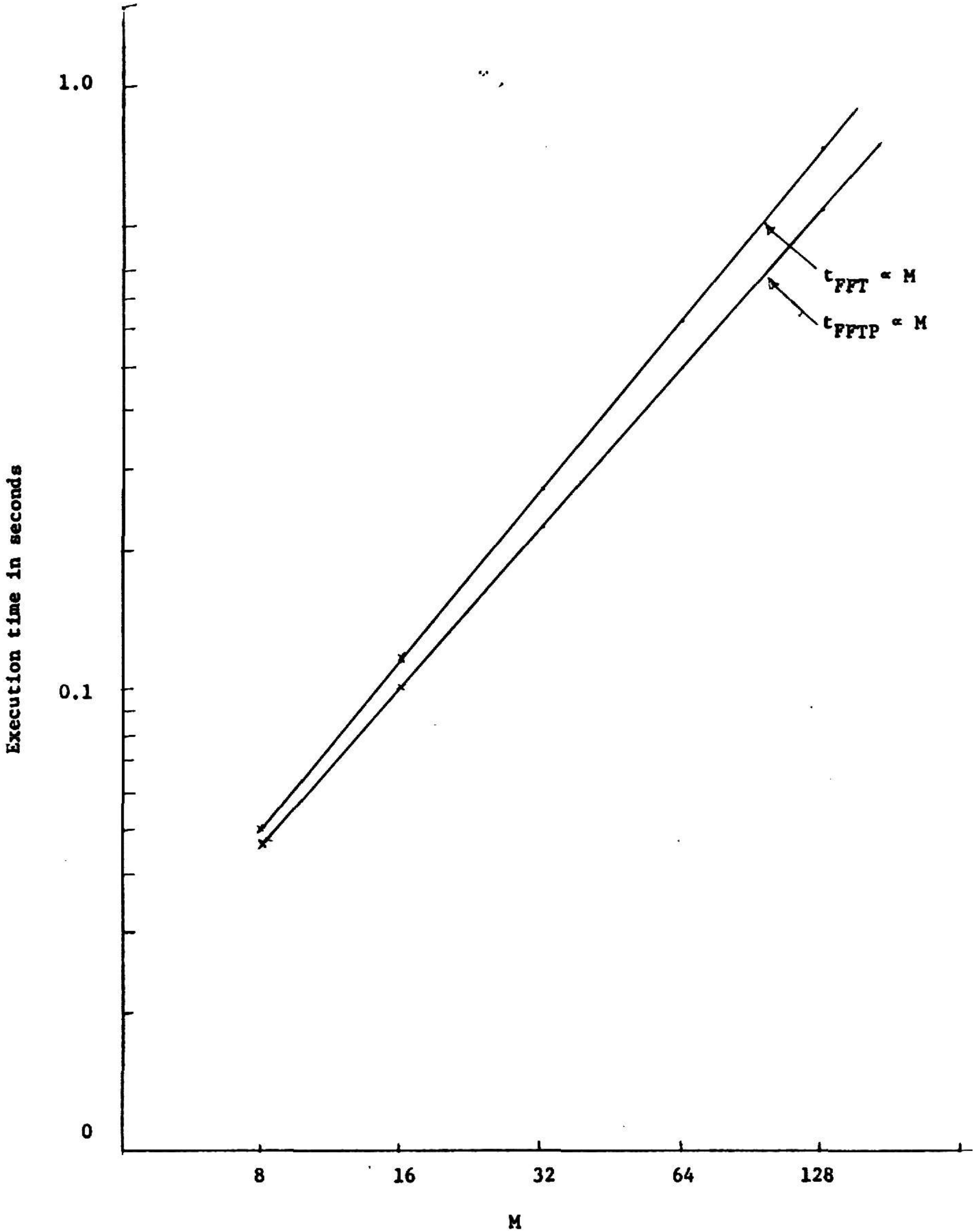


Fig. 4.6. Comparison of regular FFT and FFT pruning execution times.

CHAPTER V

CONCLUSIONS

From the results presented in Chapters III and IV, it is apparent that FFT with pruning can be used effectively to save computational time in the following areas:

1. Narrow band frequency analysis.
2. Formant analysis of speech in cases where smoothed log-magnitude spectra are desired.
3. On-line computation of autocorrelation functions.

It is recommended that the computer programs developed in connection with this study be used to analyze the speech of deaf speakers, samples of which have been collected and digitized.*

---

*This data was collected in connection with a joint study between the departments of Electrical Engineering and Speech of Kansas State University.

REFERENCES

1. Ahmed, N., Orthogonal Transforms for Digital Signal Processing, Springer-Verlog: New York/Berlin/Heidelberg, (in press).

2. Ahmed, N., Unpublished notes, Dept. of Elec. Engg., Kansas State University, Manhattan, Kansas, August 1972.

3. Bergland, G. D., "A radix-eight fast Fourier transform subroutine for real-valued series," IEEE Trans. Audio Electroacoust., Vol. AU-17, pp. 138-144, June 1969.

4. Cochran, W. T., et al., "What is the Fast Fourier Transform?," IEEE Trans. Audio Electroacoust., Vol. AU-15, No 2, pp. 45-55, June 1967.

5. Cooley, J. W. and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Math. of Comput., Vol. 19, pp. 297-301, April 1965.

6. Ferrie, J. F., C. W. Nawrocki, and G. C. Carter, "Implementation and Results of the Modified Chirp Z-transform," Published as Navy Underwater Systems Center (NUSC) Report, TM No. TC-191-72, 1972.

7. Flanagen, J. L., et al., "Synthetic Voices for Computers," IEEE Spectrum, Vol. 7, No. 10, pp. 22-44, October 1970.

8. Gentleman, W. M. and G. Sande, "Fast Fourier transforms for fun and profit," in 1966 Fall Joint Comput. Conf., AFIPS Conf. Proc., Vol. 29, Washington, D. C., Spartan, 1966, pp. 563-578.

9. Gold and Rader, "Digital Processing of Signals," McGraw-Hill, Inc., 1969.

10. Markel, J. D., "FFT Pruning", IEE Trans. Audio Electroacoust., Vol. Au-19, December 1971, pp. 305-311.

11. Otnes, R. K. and L. Enochson, "Digital Time Series Analysis." John Wiley and Sons, 1972.

12. Rabiner, L. R., R. W. Shafer, and C. M. Rader, "The Chirp Z-transform Algorithm and its applications," The Bell System Journal, Vol. 48, No. 5, May-June 1969, pp. 1249-1292.

13. Rader, C. S., "An Improved Algorithm for High Speed Autocorrelation with Applications to Spectral Estimation," IEE Trans. Audio Electroacoust., Vol. AU-18, No. 4, December 1970, pp. 439-442.

14. Shilling, S., "A Study of the Chirp Z-transform and its Applications," A Master's Degree Report, Kansas State University, Manhattan, Kansas, 1972.

15. Sid-Ahmed, M. A., "Analysis and Synthesis of One and Two Dimensional Recursive Digital Filters," Appendices B and C, Ph.D. Dissertation, Elect. Engg. Dept., University of Windsor, Canada, 1974.

16. R. C. Singleton, "An algorithm for computing the mixed radix fast Fourier transform," IEEE Trans. Audio Electroacoust., Vol. AU-117, June 1969, pp. 93-103.

APPENDICES

APPENDIX I

This appendix provides a listing of a FORTRAN program implementation of a time domain FFT pruning algorithm [See Fig. 3.2].

```
FORTRAN IV G LEVEL  21                  FFTP                 DATE = 74345             22/24/54

              SUBROUTINE FFTP(X,M,L)
        C***************************************************************************
        C                                                                          *
        C     THIS SUBROUTINE IS AN IMPLEMENTATION OF A TIME PRUNED FFT,          *
        C     USING THE DECIMATION IN FREQUENCY ALGORITHM.NUMBER OF OUTPUT        *
        C     SAMPLES=2**M,WHERE M IS GREATER THAN OR EQUAL TO L.                 *
        C                                                                          *
        C***************************************************************************
              COMPLEX CMPLX,W,X(512),T
              K=M-L
              N=2**M
              L2=2**L
              DO 1 LO=1,M
              LMX=2**(M-LO)
              LIX=2*LMX
              SCL=6.283185/LIX
              IF (LO-K)2,2,3
            2 DO 4 LM=1,L2
              ARG=(LM-1)*SCL
              W=CMPLX(COS(ARG),-SIN(ARG))
              DO 4 LI=LIX,N,LIX
              J1=LI-LIX+LM
              J2=J1+LMX
            4 X(J2)=W*X(J1)
              GO TO 1
            3 DO 5 LM=1,LMX
              ARG=(LM-1)*SCL
              W=CMPLX(COS(ARG),-SIN(ARG))
              DO 5 LI=LIX,N,LIX
              J1=LI-LIX+LM
              J2=J1+LMX
              T=X(J1)-X(J2)
              X(J1)=X(J1)+X(J2)
            5 X(J2)=W*T
            1 CONTINUE
              NV2=N/2
              NM1=N-1
              J=1
              DO 7 I=1,NM1
              IF(I.GE.J) GO TO 6
              T=X(J)
              X(J)=X(I)
              X(I)=T
            6 K=NV2
            8 IF(K.GE.J) GO TO 7
              J=J-K
              K=K/2
              GO TO 8
            7 J=J+K
              RETURN
              END
```

APPENDIX II

This appendix provides a listing of a FORTRAN program implementation of a frequency FFT pruning algorithm [See Fig. 3.6].

```
      SUBROUTINE FFTP(X,M,L)
C***********************************************************************
C                                                                      *
C     THIS SUBROUTINE IS AN IMPLEMENTATION OF A FREQUENCY PRUNED      *
C     FFT,USING THE DECIMATION IN TIME ALGORITHM.NUMBER OF INPUT       *
C     SAMPLES=2**M.NUMBER OF OUTPUT SAMPLES=2**L,WHERE M IS GREATER    *
C     THAN OR EQUAL TO L.                                              *
C                                                                      *
C***********************************************************************
      COMPLEX X(512),U,W,T,CMPLX
      PI=3.141592
      N=2**M
      N1=2**L
      NV2=N/2
      NM1=N-1
      J=1
      DO 7 I=1,NM1
      IF(I.GE.J) GO TO 5
      T=X(J)
      X(J)=X(I)
      X(I)=T
    5 K=NV2
    6 IF(K.GE.J) GO TO 7
      J=J-K
      K=K/2
      GO TO 6
    7 J=J+K
      DO 40 LO=1,M
      LE=2**LO
      LE1=LE/2
      U=CMPLX(1.0,0.0)
      W=CMPLX(COS(PI/LE1),-SIN(PI/LE1))
      IF(LO-L) 20,20,30
   20 DO 11 J=1,LE1
      DO 9 I=J,N,LE
      IP=I+LE1
      T=X(IP)*U
      X(IP)=X(I)-T
    9 X(I)=X(I)+T
   11 U=U*W
      GO TO 40
   30 DO 12 J=1,N1
      DO 10 I=J,N,LE
      IP=I+LE1
   10 X(I)=X(I)+X(IP)*U
   12 U=U*W
   40 CONTINUE
      RETURN
      END
```

APPENDIX III

This appendix consists of a listing of the computer program used to implement the MCZT and the PAM-CST. The parameters used in the program are the following:

M = # of MCZT pts

$N = NN$ = # of data points in the data sequence

NPAR = # of partitions. If it is MCZT, then NPAR = 1.

$\hat{N}$ = SPAR = Size of each partition; i.e. $\hat{N} = N/NPAR$

L = the smallest integer power of 2, which $\geq (M + \hat{N} - 1)$

DF = frequency interval which is related to the specified resolution

FU = specified upper frequency in Hz.

FL = specified lower frequency in Hz.

T = sampling interval in seconds

The desired parameters L, RAPH, and RWPH are computed using the following relations:

$$(M - 1) = \frac{FU - FL}{DF}$$

$$RWPH = \phi_0 = -(DF)T$$

$$RAPH = \theta_0 = FL(T) \qquad (A3.1)$$

<u>Illustrative example</u>. We consider the case when

$$\{X(m)\} = X_k\ ,\ k = 0, 1, \ldots, 31$$

where

$$X_k = \frac{e^{5/7}}{\sin(50\pi/32)}\, e^{-k/7} \sin(10\pi k/32),\ k = 0, 1, \ldots, 31$$

$$\ldots\ldots\ldots \quad (A3.2)$$

Equation (A3.1) represents the sampled values of a 5 Hz damped sinusoid, assumed to be sampled at 32 samples/sec.

The spectrum of X_k using the MCZT and PAM-CZT is desired such that the DFT resolution is increased by a factor of 4 in the 3 Hz to 5.5 Hz region.

Solution: The desired spectra are shown in Fig. A3.1. In the case of MCZT, the PAM-CZT algorithm was used with 8-artitions. The parameters for which [See Eq. (A3.1)] are as follows:

$\hat{N} = 1$ for MCZT, 4 for PAM-CZT

$M = 11$

$L = 16$

RWPH = - 1/128

RAPH = 3/32

DF = 0.25 Hz

The program is listed in what follows.

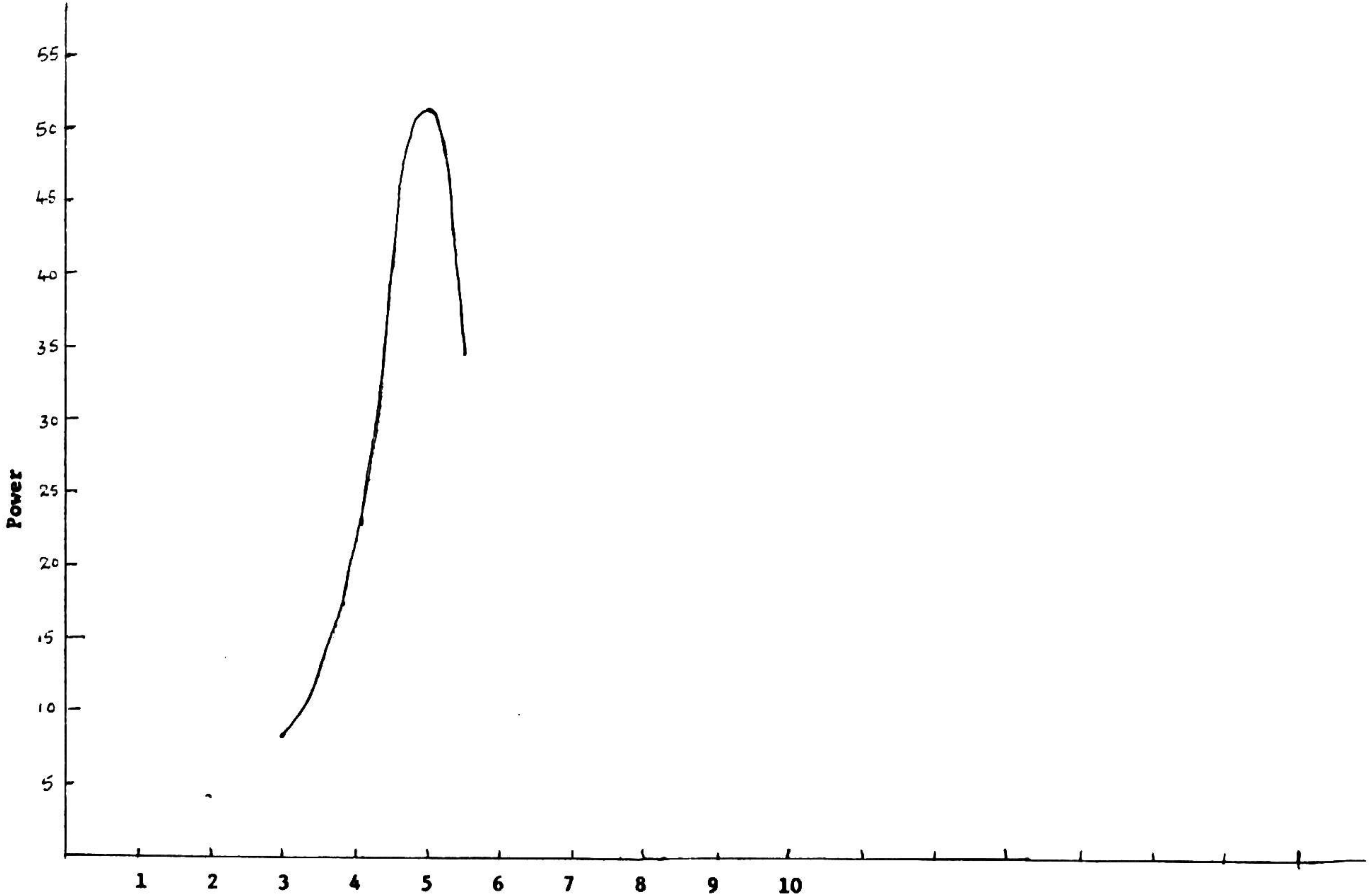


Fig. A3.1. Narrow Band Power Spectrum

```
C*********************************************************************
C                                                                    *
C     THIS PROGRAM IMPLEMENTS THE MCZT ALGORITHM AS WELL AS THE       *
C     PAM-CZT ALGORITHM. IT CAN BE USED TO COMPUTE THE DFT COEFFS     *
C     AND THE POWER SPECTRUM.                                        *
C                                                                    *
C*********************************************************************
      COMPLEX XDC(512),Y(512),Z(512),YY(512),B1,CMPLX
      DIMENSION XX(512),AMP(512)
      INTEGER SPAR
      INCARD=5
      IPRNTR=6
C
C     READ CONTROAL PARAMETERS
C
    5 FORMAT('0',5X,'NARROW BAND SPECTRA USING MCZT')
   10 READ(INCARD,20) NN,M,L,RAPH,RWPH,NPAR
   20 FORMAT(3I10,2F10.8,I10)
      IF (NN.EQ.0) GO TO 900
      PRINT 5
      WRITE(IPRNTR,30) NN,M,L,RAPH,RWPH,NPAR
   30 FORMAT('0',///10X,'NN=',I4,5X,'M=',I4,5X,'L=',I4,5X,'RAPH=',F1
     X5.8,5X,'RWPH=',F15.8,5X,'NPAR=',I5,///)
C
C     COMPUTE CONSTANTS
C
      SPAR=NN/NPAR
      R2PI=6.283185
      DNN1=R2PI*RAPH*FLOAT(SPAR)
      DNN2=R2PI*RWPH*FLOAT(SPAR)
C
C     GENERATE THE SIGNAL
C
      PI=3.141592
      C=EXP(5./7.)/(SIN(50.*PI/NN))
      DO 40 J=1,NN
      AK=(J-1)
   40 XX(J)=C*EXP(-AK/7.)*SIN(10.*PI*AK/NN)
      PRINT 50
   50 FORMAT('0',9X,'N',7X,'INPUT SIGNAL')
      DO 60 J=1,NN
      I=J-1
      WRITE(IPRNTR,55) I,XX(J)
   55 FORMAT(' ',8X,I2,4X,F10.5)
   60 CONTINUE
      DO 70 I=1,M
   70 Z(I)=CMPLX(0.0,0.0)
      DO 110 K=1,NPAR
      DO 80 J=1,SPAR
      ISPAR1=SPAR*(K-1)
      XDC(J)=CMPLX(XX(J+ISPAR1),0.0)
   80 CONTINUE
      CALL CHIRPZ(XDC,SPAR,M,L,RAPH,RWPH,Y)
      DO 90 I=1,M
      FIM1=FLOAT(I-1)
      BETA1=(DNN2*FIM1-DNN1)*(K-1)
      B1=CMPLX(COS(BETA1),SIN(BETA1))
      Y(I)=B1*Y(I)
```

```
   90 CONTINUE
      DO 110 I=1,M
  100 Z(I)=Z(I)+Y(I)
  110 CONTINUE
      PRINT 120
  120 FORMAT('0',4X,'FREQENCY',12X,'DFT COEFFS',24X,'POWER SPECTRUM')
      PRINT 130
  130 FORMAT('0',5X,'IN HERTZ',12X,'REAL',12X,'IMAG')
      FR=2.75
      DO 150 J=1,M
      FR=FR+0.25
      PS=CABS(Z(J))
      PS=PS*PS
      WRITE(IPRNTR,140) FR,Z(J),PS
  140 FORMAT('0',6X,F5.2,9X,2F13.6,11X,F12.8)
  150 CONTINUE
      PRINT 160
  160 FORMAT('1',5X,'NARROW BAND SPECTRA USING PAM-CZT')
      GO TO 10
  900 STOP
      END
```

```
FORTRAN IV G LEVEL  21              CHIRPZ             DATE = 74345          22/24/52

            SUBROUTINE CHIRPZ(X,NN,M,L,RAPH,RWPH,GK)
            COMPLEX X(512),Y(512),V(512),GK(512),C1,C2,CMPLX
            R2PI=6.283185
            DN1=R2PI*RAPH
            DN2=R2PI*RWPH
            DO 100 I=1,NN
            FIM1=FLOAT(I-1)
            THETA1=FIM1*(DN2*FIM1/2.0-DN1)
            C1=CMPLX(COS(THETA1),SIN(THETA1))
            Y(I)=X(I)*C1
        100 CONTINUE
            NNP1=NN+1
            DO 110 I=NNP1,L
            Y(I)=CMPLX(0.0,0.0)
        110 CONTINUE
      C
      C     COMPUTE THE DFT OF Y
      C
            II=0
            CALL FFT(L,Y,II)
      C
      C     FORM THE L POINT SEQUENCE
      C
            DO 120 I=1,M
            FIM1=FLOAT(I-1)
            THETA2=DN2*FIM1**2/2.0
            V(I)=CMPLX(COS(THETA2),-SIN(THETA2))
        120 CONTINUE
            IF (L.EQ.M+NN-1) GO TO 140
            LMNN1=L-NN+1
            M1=M+1
            DO 130 I=M1,LMNN1
            V(I)=CMPLX(0.0,0.0)
        130 CONTINUE
        140 CONTINUE
            LMNP2=L-NN+2
            DO 150 I=LMNP2,L
            FIM1=FLOAT(L-I+1)
            THETA2=DN2*FIM1**2/2.0
            V(I)=CMPLX(COS(THETA2),-SIN(THETA2))
        150 CONTINUE
      C
      C     COMPUTE THE L POINT DFT OF V
      C
            CALL FFT(L,V,II)
            DO 160 I=1,L
      C
      C     MULTIPLY THE SEQUENCE Y AND V TO OBTAIN GK
      C
            GK(I)=Y(I)*V(I)
        160 CONTINUE
            II=1
      C
      C     COMPUTE THE DFT OF GK
      C
            CALL FFT(L,GK,II)
            DO 170 I=1,M
            FIM1=FLOAT(I-1)
```

```
      THETA3=DN2*FIM1**2/2.0
      C2=CMPLX(COS(THETA3),SIN(THETA3))
      GK(I)=C2*GK(I)
  170 CONTINUE
      RETURN
      END
```

```
FORTRAN IV G LEVEL  21                FFT                 DATE = 74345           22/24/52

                SUBROUTINE FFT(N,X,II)
          C*********************************************************************
          C     THIS PROGRAM IMPLEMENTS THE FFT ALGORITHM TO COMPUTE THE DISCRETE
          C     FOURIER COEFFICIENTS OF A DATA SEQUENCE OF N POINTS
          C     CALLING SEQUENCE FROM THE MAIN PROGRAM:
          C     CALL FFT(N,X,II)
          C           N: NUMBER OF DATA POINTS (MAX.512)
          C           X: COMPLEX ARRAY CONTAINING THE DATA SEQUENCE. IN THE END DFT
          C              COEFFS. ARE RETURNED IN THE ARRAY. MAIN PROGRAM SHOULD
          C              DECLARE IT AS--    COMPLEX  X(512)
          C          II: FLAG FOR INVERSE
          C              II=0  FOR FORWARD TRANSFORM
          C              II=1  FOR INVERSE TRANSFORM
          C
          C*********************************************************************
                COMPLEX X(512),CMPLX,A,T,ALPHA
          C     CALCULATE THE # OF ITERATIONS (LOG. N TO THE BASE 2)
                ITER=0
                IREM=N
             10 IREM=IREM/2
                IF (IREM.EQ.0) GO TO 20
                ITER=ITER+1
                GO TO 10
             20 CONTINUE
                SIGN=-1
                IF (II.EQ.1) SIGN=1
                ARG=2.*3.141592/FLOAT(N)
                B1=COS(ARG)
                B2=SIN(ARG)
                A=CMPLX(B1,SIGN*B2)
                IP=1
                IGK=N
                DO 50 K=1,ITER
          C
          C     COMPUTATION FOR EACH ITERATION
          C
                IGK=IGK/2
                ILAST=IGK
                LA=-IP
                DO 40 I=1,ILAST
                LA=LA+IP
                IGK2=IGK+IGK
                MX=-IGK2
          C
          C     CALCULATE THE MULTIPLIER
          C
                ALPHA=A**LA
          C     IP: # OF PARTITIONS IN THE ITERATION
                DO 30 M=1,IP
          C
          C     COMPUTATION FOR EACH PARTITION
          C
                MX=MX+IGK2
                IPMX=I+MX
                IPMM=IPMX+IGK
                T=X(IPMX)-X(IPMM)
                X(IPMX)=X(IPMX)+X(IPMM)
                X(IPMM)=ALPHA*T
```

```
0034          30 CONTINUE
0035          40 CONTINUE
0036             IP=IP+IP
0037          50 CONTINUE
0038             IF (II.EQ.0) GO TO 58
0039             DO 55 I=1,N
0040          55 X(I)=X(I)/FLOAT(N)
0041          58 CONTINUE
           C
           C     UNSCRAMBLE THE BIT REVERSED DFT COEFFS
           C
0042             N2=N/2
0043             N1=N-1
0044             J=1
0045             DO 65 I=1,N1
0046             IF (I.GE.J) GO TO 59
0047             T=X(J)
0048             X(J)=X(I)
0049             X(I)=T
0050          59 K=N2
0051          60 IF (K.GE.J) GO TO 65
0052             J=J-K
0053             K=K/2
0054             GO TO 60
0055          65 J=J+K
0056             RETURN
0057             END
```

NARROW BAND SPECTRA USING MCZT

NN= 32 M= 11 L= 64 RAPH= 0.09375000 RWPH= -0.00781250 NPAR= 1

N	INPUT SIGNAL
0	0.0
1	-1.50120
2	-1.44600
3	-0.26470
4	0.83167
5	1.00000
6	0.33824
7	-0.42567
8	-0.66420
9	-0.31989
10	0.19100
11	0.42437
12	0.26523
13	-0.06343
14	-0.26041
15	-0.20317
16	-0.00000
17	0.15267
18	0.14706
19	0.02692
20	-0.08458
21	-0.10170
22	-0.03440
23	0.04329
24	0.06755
25	0.03253
26	-0.01943
27	-0.04316
28	-0.02697
29	0.00645
30	0.02648
31	0.02066

FREQENCY IN HERTZ	DFT COEFFS REAL	IMAG	POWER SPECTRUM
3.00	-2.793992	0.782343	8.41845131
3.25	-3.055629	0.984810	10.30671406
3.50	-3.338512	1.347240	12.96071625
3.75	-3.574569	1.829725	16.12542725
4.00	-3.834489	2.448029	20.69613647

4.25	-4.069427	3.445261	28.43006897
4.50	-3.816283	4.956975	39.13560486
4.75	-2.532948	6.503665	48.71347046
5.00	-0.341800	7.173411	51.57464600
5.25	1.841603	6.500021	45.64175415
5.50	3.116057	4.964082	34.35189819

NAPROW BAND SPECTRA USING PAM-CZT

NARROW BAND SPECTRA USING MCZT

NN= 32 M= 11 L= 16 RAPH= 0.09375000 RWPH= -0.00781250 NPAR= 8

N	INPUT SIGNAL
0	0.0
1	-1.50120
2	-1.44600
3	-0.26470
4	0.83167
5	1.00000
6	0.33824
7	-0.42567
8	-0.66420
9	-0.31989
10	0.19100
11	0.42437
12	0.26523
13	-0.06343
14	-0.26041
15	-0.20317
16	-0.00000
17	0.15267
18	0.14706
19	0.02692
20	-0.08458
21	-0.10170
22	-0.03440
23	0.04329
24	0.06755
25	0.03253
26	-0.01943
27	-0.04316
28	-0.02697
29	0.00645
30	0.02648
31	0.02066

FREQENCY	DFT COEFFS		POWER SPECTRUM
IN HERTZ	REAL	IMAG	
3.00	-2.793987	0.782366	8.41845131
3.25	-3.055624	0.984815	10.30669022
3.50	-3.338517	1.347249	12.96077156
3.75	-3.574569	1.829732	16.12544250
4.00	-3.834494	2.448036	20.69619751

4.25	-4.069436	3.445270	28.43016052
4.50	-3.816284	4.956997	39.13581848
4.75	-2.532944	6.503680	48.71365356
5.00	-0.341770	7.173414	51.57466125
5.25	1.841622	6.500010	45.64170837
5.50	3.116057	4.964058	34.35166931

ACKNOWLEDGEMENTS

The author wishes to thank his Major Professor, Dr. N. Ahmed for his helpful remarks and valuable guidance throughout the preparation of this report. The author also thanks Mr. T. Natarajan for some of his valuable suggestions.

A STUDY OF FFT PRUNING AND ITS APPLICATIONS

by

M. A. RAMAKRISHNAIAH

B. E., 1970, University of Mysore, Karnataka, India

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1975

FFT pruning corresponds to eliminating arithmetic operations that do not contribute to the output in the computation of DFT coefficients. It is shown that FFT pruning is faster than other FFT algorithms, when (i) number of nonzero input data points is considerably smaller than the desired number of output points, or, (ii) the desired number of transform coefficients is considerably smaller than the number of input points. It can be used effectively in the frequency domain as well as the time domain.

Applications of FFT pruning that are considered in this report are narrow-band spectral analysis, formant analysis of speech and high speed autocorrelation.