Faster Fourier Transformation: The Algorithm of S. Winograd

Shalhav Zohar

February 15, 1979

National Aeronautics and Space Administration
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California
Faster Fourier Transformation: The Algorithm of S. Winograd

Shalhav Zohar

February 15, 1979

National Aeronautics and Space Administration
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California
The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under NASA Contract No. NAS7-100.
NOTE REGARDING PUBLICATION DATE

The first version of this paper was completed in March 1977. Subsequently, following the publication of [10], a few changes were implemented. These are pointed out in the text and footnotes. The paper was finalized in its present form in May 1977. The delay in publication to February 1979 was due to factors beyond the author's control.
ABSTRACT

The new DFT algorithm of S. Winograd is developed and presented in detail. This is an algorithm which uses about \( \frac{1}{5} \) of the number of multiplications used by the Cooley-Tukey algorithm and is applicable to any order which is a product of relatively prime factors from the following list: 2, 3, 4, 5, 7, 8, 9, 16. The algorithm is presented in terms of a series of tableaus—one for each term in this list—which are convenient, compact, graphical representations of the sequence of arithmetic operations in the corresponding parts of the algorithm. Using these in conjunction with Tables 5-6 (pp. 80, 84), makes it relatively easy to apply the algorithm and evaluate its performance.

The organization of the paper allows skipping a large part of it on a first reading (see p. 3).
# TABLE OF CONTENTS

I. Introduction .................................................................................................................. 1  
II. Strategy Development ................................................................................................. 4  
III. The Basic LCT Algorithms ....................................................................................... 13  
IV. The Basic DFT Algorithms for Prime N ..................................................................... 28  
V. The Basic DFT Algorithms for N=4,9 ........................................................................ 36  
VI. The Basic DFT Algorithms for N=8,16 ..................................................................... 46  
VII. The DFT Algorithm for \( N = \prod_{k=1}^{K} N_k \) ......................................................... 63  
VIII. Speed Analysis ........................................................................................................ 79  
IX. Concluding Remarks .................................................................................................. 86  

Appendix: Polynomial Congruences .................................................................................. 88  

# FIGURES

Figure 1. Algorithm for LCT of order 2 ............................................................................ 17  
Figure 2. Algorithm for LCT of order 4 ............................................................................ 19  
Figure 3. Stages in the development of the algorithm for LCT of order 6 ..................... 25  
Figure 4. Algorithm for LCT of order 6 ............................................................................ 27  
Figure 5. Algorithm for DFT of order 3 ............................................................................ 31  
Figure 6. Algorithm for DFT of order 5 ............................................................................ 31  
Figure 7. Computation of \( \omega_i \)'s for the DFT algorithm of order 7 ......................... 34  
Figure 8. Algorithm for DFT of order 7 ............................................................................. 35  
Figure 9. Algorithm for DFT of order 2 ............................................................................. 37  
Figure 10. Algorithm for DFT of order 4 .......................................................................... 39  
Figure 11. Computation of \( \omega_i \)'s for the DFT algorithm of order 9 ......................... 42  
Figure 12. Index permutations in assembling Fig. 13 ....................................................... 43  
Figure 13. Algorithm for DFT of order 9 .......................................................................... 45  
Figure 14. Algorithm for DFT of order 8 .......................................................................... 51  
Figure 15. Intermediate tableau for the computation of \( F'_i \) for the -DFT of order 16 ........................................................................................................ 54  
Figure 16. E matrix for the computation of \( F'_i - F'_i \) for the DFT of order 16 .......... 58
FIGURES (Cont'd)

Figure 17. Algorithm for DFT of order 16 ........................................ 60
Figure 18. Schematic representation of the basic DFT tableaus ............... 65
Figure 19. Input square of the 8-th order modified tableau for - example (7.12) ......................................................... 73
Figure 20. Subdivision of the Y matrix for example (7.29) ..................... 73
Figure 21. Algorithm for DFT of order 30 (example (7.29)) ................... 75

TABLES

Table 1. Rational Factorization of $x^n - 1$ ....................................... 14
Table 2. Modular Representation of $v$ in Example (7.12); $N_1 = 8$; $N_2 = 3$; $N_3 = 5$ ......................................................... 69
Table 3. Scrambling for Example (7.12); $n$ vs. $v = (v_1, v_2, v_3)$ .......... 70
Table 4. The DFT Tableau Multipliers ($\omega_{1, N}$) ................................ 77
Table 5. Summary of Basic DFT Tableaus ........................................... 80
Table 6. Summary of DFT Algorithms ............................................... 84

Table A1. $s_n(x) \mod m(x)$ ............................................................. 90
Table A2. $\{P(x)Q(x)\} \mod m(x)$ ..................................................... 92
I. Introduction

Ever since the discovery of the FFT algorithm [1], the following question must have been phrased in many minds: "Does the FFT algorithm represent the ultimate in the fast computation of the discrete Fourier transform (DFT) or is there a still faster algorithm yet to be discovered?" One answer was provided in 1968 by Yavne [2] who showed that the number of multiplications could be halved while leaving the number of additions unchanged. More recently (1976), a significant step along this path was taken by S. Winograd [3] who developed an algorithm which reduces the number of multiplications of the radix-2 FFT algorithm [1] by a factor of about 5. This reduction is accompanied by a small increase or decrease in the number of additions. In most cases, the increase does not exceed 20%.

Our basis for comparison both here and later on is the "nominal" performance of the Cooley-Tukey (FFT) algorithm, namely, the computation of an N-th order DFT of complex data with \( M_{CT} \) real multiplications and \( A_{CT} \) real additions where

\[
M_{CT} = 2N \log_2 N; \quad A_{CT} = 1.5M_{CT}
\] 

(1.1)

We adopt (1.1) as the basis for comparison for all N.

Winograd's algorithm then performs the above task using about \( \frac{M_{CT}}{5} \) real multiplications. We devote the rest of this section to a description of the capabilities and constraints of the algorithm so that the reader could assess its suitability to his needs before delving any deeper.

At its present state of development, the algorithm is applicable to any N satisfying

\[
N = \prod_{k=1}^{K} N_k
\]

(1.2)

in which the \( N_k \)'s are relatively prime factors taken from the following list

\[
N_k = 2, 3, 4, 5, 7, 8, 9, 16
\]

(1.3)

The maximal \( N \) is therefore \( 16 \cdot 9 \cdot 7 \cdot 5 = 5040 \). All of the N values satisfying the above prescriptions are listed in the summary of the algorithm presented in Table 6 (p. 84). The actual multiplication reduction factor is listed there for each N, under the heading \( G_m \). Note that the average of \( G_m \) for all \( N > 140 \) is about 5.5. With such a large reduction in the number of multiplications, it is quite obvious that the new algorithm will run faster than the Cooley-Tukey algorithm in most systems. To make a more specific claim, we must know the basic system parameter \( \mu \).

Eqn. (1.1) is adopted as a convenient yardstick. It should be borne in mind that in addition to Yavne's algorithm, there are other FFT variants which are somewhat more efficient than (1.1).
which is the ratio of the time taken to execute one real multiplication to the time taken to execute one real addition. (The term "real" is used here as opposed to "complex"; not as opposed to "integer" in the Fortran language). For very large \( \mu \) (microprocessors, software multipliers, etc.), the speed gain approaches \( G_\infty \) asymptotically. For lower values of \( \mu \), the gain would be smaller. Denoting the speed gain by \( G \), it will be shown (pp. 85, 79) that
\[
G(\mu) = G_\infty \frac{\mu + 1.5}{\mu + R}
\]
where \( R \) is another parameter listed with \( G_\infty \) in Table 6. Obviously, it is now a trivial matter to compute the speed gain for any system and any permissible \( N \).

Since \( R > 1.5 \) for all practical \( N \) values, it is obvious that the advantage of this new algorithm diminishes with decreasing \( \mu \). Yet, it is interesting to note that even in the extreme case of \( \mu = 1 \), the new algorithm is still the faster one for all \( N \) values of Table 6.

The main disadvantage of the algorithm is its need for a large memory. We express this in terms of the parameter \( M \) of Table 6. For the processing of complex data we have to have a storage array of \( 1.5M \) real words. \( M \) varies from about \( 2N \) at the lower end of the table to about \( 4N \) at its upper end. Thus, for high \( N \) values, we require a storage array of size \( 6N \). This is \( 4N \) more than the minimal requirements of \( 2N \) for storing the input vector.

Of the total memory requirement of \( 1.5M \) real words, \( 0.5M \) are needed for the storage of precomputed constants. Since not all of these constants are distinct, it is probably feasible to reduce this part by the use of a more involved addressing scheme.

Another probable disadvantage of the new algorithm is that, in comparison with the Cooley-Tukey algorithm, it might require more bits per word to maintain a prescribed level of precision. This effect is discussed in some more detail in section IX but the whole subject merits further study.

The development of the algorithm consists of two distinct parts. In the first part, fast DFT algorithms for the low orders listed in (1.3) are developed as a set of building blocks. The second part introduces a combining algorithm which integrates groups of these building blocks into the desired final structures, namely, DFT algorithms for orders \( N \) prescribed by (1.2).

The low-order algorithms of (1.3) are derivable from algorithms of orders 2, 4, 6 of another type of transformation called LCT (Left-Circulant Transformation). Hence, the following structure of the paper: Section II is devoted to the DFT-LCT interrelationship. This is followed by the three LCT algorithms in Section III.
and the seven DFT algorithms derived from them in Sections IV, V, VI. Section VII tackles the integration of these low order algorithms into the desired algorithm for order N satisfying (1.2). Section VIII is devoted to performance evaluation and Section IX concludes with an overview and some comments regarding "in-place" transformation.

The treatment of the subject is detailed and relatively complete, aiming to provide a sound basis for further development of the subject. Naturally, this demands a substantial investment of time. It should be pointed out, however, that a reasonable grasp of the basic ideas and their application can be obtained by skipping the detailed derivations of the low-order algorithms. If this is acceptable, the following parts of the paper may prove sufficient: Section II, first part of Section III (up to the treatment of the left circulant of order 4 on p. 18), the introduction of the $\eta$ vector on p. 40, the tableau generalization in Section VI (portion bounded by eqns. (6.16), (6.17) on p. 49), Section VII, last part of Section VIII (following eqn. (8.9) on p. 83), and Section IX.

We conclude this introduction with a few words regarding the circumstances that initiated work on the present paper. Winograd's description of his algorithm [3] is very short (1½ pages) and, partly because of that, hard to follow. However, even with the necessary mathematical background and a full mastery of the paper, actual implementation would still be out of reach without a lot of additional hard work. This is partly due to the fact that the basic building blocks comprising the algorithm are only sketched in general outline—not actually constructed.\(^\text{1b}\) It turns out that, for some of these, the transition from outline to end product is far from trivial.

The work reported here was started as an effort to understand and be able to apply what appeared to be—and is in fact—a highly promising, fascinating, algorithm. It was in the course of this work, as the various difficulties were being overcome and the different pieces of the puzzle were beginning to form a coherent overall picture, that the prospect of sharing the knowledge thus acquired, began to have merit. There is no question that the absence of a fuller treatment of the algorithm constitutes a serious obstacle to its wider dissemination and use. This is where the present paper comes in. We present Winograd's algorithm in detail, we construct the required building blocks and we show how to incorporate them in the overall algorithm. In short, we provide the missing links that would allow immediate implementation of the algorithm.

\(^{1b}\) Since the submission of this paper for publication, a summary of the algorithms did appear in print [10], though still without derivation. This helped to shed light on two points of discrepancy between Winograd's stated results [3] and the first version of this paper. For further details see footnotes 8 (p. 26) and 9 (p. 41).
II. **Strategy Development**

The cornerstone of Winograd's algorithm is a theorem [4] providing the solution to a seemingly unrelated problem: Given the two polynomials \( A(x), B(x) \), what is the minimal number of multiplications required to compute

\[
\{ A(x)B(x) \} \mod C(x)
\]

where \( C(x) \) is a given monic polynomial. The connection with the DFT consists of two links: Firstly, the DFT matrix is shown to be related to another special transformation in which the transforming matrix is a left-circulant (exact definition follows later). Secondly, the evaluation of this transformation is shown to be identical with the evaluation of (2.1). Thus, the minimization of the number of multiplications in (2.1) leads via the above two links to a minimization of the number of multiplications in the computation of the DFT.

We proceed now with some required definitions. A Hankel matrix is one in which the value of element \( a_{ij} \) is a function of \( (i+j) \). In such a matrix, one encounters identical elements as one moves along any diagonal sloping down and to the left. Obviously, the matrix is completely determined by its first row and last column.

The matrix we will be concerned with here is a special case of a Hankel matrix, namely, a Hankel matrix for which (for order \( n \) and index range \( 0,1,\ldots, n-1 \))

\[
a_{\rho,n-1} = a_{0,\rho-1} \quad (1 \leq \rho \leq n-1)
\]

that is, the last column is a trivial rearrangement of the elements of the first row. Hence, this matrix is completely prescribed by its first row. Indeed, the second row is obtained from the first one by a circular left shift (element shifted out on the left, reappears on the right), the third is derived the same way from the second and so on. We call such a matrix a left-circulant\(^2\) or, equivalently,

---

1c If \( C(x) \) is of degree \( n \), it is said to be monic when the coefficient of \( x^n \) is 1.

2 This is based on the term circulant which is commonly used to describe a matrix generated from its first row by circular right shifts.
an LC matrix. Similarly, the linear transformation effected by such a matrix
will be referred to as an LCT (Left-Circulant-Transformation). Finally, a
matrix which is not LC but contains a submatrix which is, will be called a
quasi-left-circulant matrix (QLC).

We turn now to the LCT-DFT link. As will become obvious later on, we
should concern ourselves with a trivial modification of the DFT defined as follows:

\[ W = e^{-\frac{2\pi}{N}} \]

\[ F_u = \sum_{v=0}^{N-1} \Omega^{uv} f_v \quad (u = 0, 1, \ldots, N-1) \]  

(2.4)

\( \Omega \) in (2.4) is an arbitrary complex constant. When \( \Omega = 1 \), (2.4) reduces to the
standard DFT.

Let \( N \), the order of the DFT, satisfy

\[ N = p^k \quad (p \text{ prime}; k \text{ integer}) \]

\[ k \begin{cases} \infty & (p \text{ odd}) \\ 3 & (p = 2) \end{cases} \]  

(2.5)

We proceed to show now that for such \( N \), eqn. (2.4) can be brought into the form
of a QLC matrix\(^3\). The derivation follows \[5\] and is based on the number
theoretic idea of a primitive root. \( g \), the primitive root of \( N \) (satisfying (2.5))
is an integer whose integral powers (mod \( N \)) generate all integers in the interval
\((1,N)\) except multiples of \( p \).

The number of multiples of \( p \) in the above interval is

\[ \frac{N}{p} = p^{k-1} \]  

(2.6)

Therefore, the number of integers generated by \( g \) is

\[ n = p^k - p^{k-1} = (p-1)p^{k-1} \]  

(2.7)

and we may say that the sequence

\[ \{g^\rho \mod N\} \quad (\rho = 0, 1, \ldots, n-1) \]  

(2.8)

is just a permutation of those integers in the interval \((1,N)\) which are not mul-
tiples of \( p \).

\(^3\)This is true for a range wider than (2.5). However, (2.5) is sufficient
for our purpose.
We use these ideas now to relabel the indices of (2.4) as follows. All indices which are not multiples of \( p \) will be represented as in (2.8). Specifically, denoting

\[
\begin{align*}
    r &= \sigma \pmod{N} \quad (\rho, \sigma = 0, 1, \ldots, n-1) \\
    s &= \sigma \pmod{N}
\end{align*}
\]  

(2.9)

we define

\[
B_{\rho} = F_{r}; \quad b_{\sigma} = f_{s} \quad (\rho, \sigma = 0, 1, \ldots, n-1)
\]  

(2.10)

The indices which are multiples of \( p \) are now used to define

\[
B(1) = F(1); \quad b(i) = f(i) \quad (i \pmod{p} = 0)
\]  

(2.11)

With these definitions we embark now on the elimination of \( F_u \), \( f_v \) from (2.4). In doing this, we split the summation of (2.4) into two parts. In the first part \( v = mp \) (\( m \) integer), that is, \( v \pmod{p} = 0 \). In the second part \( v = s \). The result is\(^4\):

\[
B(tp) = F_{tp} = \Omega \sum_{m=0}^{N-n-1} W^{mp} b_{(mp)} + \Omega \sum_{\sigma=0}^{n-1} W^{tp} b_{\sigma} \quad (t=0, 1, \ldots, N-n-1)
\]

(2.12)

\[
B_{\rho} = \hat{B}_{\rho} + B'_{\rho} \quad (\rho = 0, 1, \ldots, n-1)
\]

(2.13)

where

\[
\hat{B}_{\rho} = \Omega \sum_{m=0}^{N-n-1} W^{mp} b_{(mp)} \quad (\rho = 0, 1, \ldots, n-1)
\]

(2.14)

\[
B'_{\rho} = \Omega \sum_{\sigma=0}^{n-1} W^{(\rho+\sigma)} b_{\sigma} \quad (\rho = 0, 1, \ldots, n-1)
\]

(2.15)

The term \((\rho+\sigma)\) identifies (2.15) as a Hankel transformation, that is, if we write down (2.15) with \( \rho \) and \( \sigma \) as the row and column indices, respectively, then the matrix transforming \( b \) into \( B' \) is a Hankel matrix. More than that, it is that special kind of a Hankel matrix referred to earlier as a left-circulant. To see this, note that in view of the LC condition (2.2), our Hankel

\[
\text{We take here advantage of the fact that (2.3) ensures } W^{(m \pmod{N})} = W^m.
\]
matrix would be an LC if
\[ g^{p+n-1} = g^{n-1} \mod N \]  \hspace{1cm} (2.16)

But since the primitive root of \( N \) always satisfies\(^5\)
\[ g^n = 1 \mod N \]  \hspace{1cm} (2.17)

It is obvious that (2.16) is indeed true and (2.15) is an LCT. We conclude that the permutations (2.9)-(2.11) will transform any DFT matrix of order \( N \) prescribed by (2.5), into a QLC matrix whose LC portion is of order \( n \).

Of particular interest is the special case in which \( N \) is prime, that is, \( k=1; \ N=p \), so that (2.7) yields
\[ n = N-1 \]  \hspace{1cm} (2.18)

and the sequence (2.8) is just a permutation of the integers 1,2,...,n. In this case, eqns. (2.12), (2.14) simplify as follows:
\[ B_0 = \hat{P}_0 + \sum_{\rho=0}^{n-1} b_{\rho} \]  \hspace{1cm} (2.19)

To illustrate these ideas, consider the case of \( N=7 \). The least positive primitive root of 7 is 3 \( [6] \). Indeed, direct computation yields

\[
\begin{array}{c|ccccc}
\rho & 0 & 1 & 2 & 3 & 4 \\
\hline
3\rho \mod 7 & 1 & 3 & 2 & 6 & 4 & 5 \\
\end{array}
\]  \hspace{1cm} (2.21)

Applying (2.9)-(2.11), we get the following form

---

\(^5\)In the standard treatment of primitive roots, (2.8) is usually replaced by
\[ \{g^\rho \mod N\} (\rho=1,2,...,n) \]  \hspace{1cm} (2.8')

Congruence (2.17) shows that the two formulations are equivalent. To prove (2.17) assume \( g^m = 1 \mod N \) for \( m<n \). It follows then that \( g^{m+1} = g^1 \mod N \) and thus two of the terms in sequence (2.8') are equal. Hence the contradiction that \( g \) is not a primitive root. Conclusion: \( m=n \).
The LC structure is quite apparent here.

So far, we have established the first link to problem (2.1) for $N$ satisfying (2.5). This constraint on $N$ will be relaxed later on. We turn now to the second link, namely, showing that evaluation of an LCT is equivalent to evaluation of (2.1). We intend to establish this equivalence and, from it, derive algorithms for some general low-order LC transformations. It should be pointed out, however, that the LC matrix we are concerned with is one obtained by permuting a DFT submatrix and, as such, is still a function of only one variable ($W$) whereas the general LC matrix of order $n$ is a function of $n$ variables. This suggests further simplifications in our case which will indeed be realized later on.

The matrix multiplication we are considering is shown in (2.23) where the LC pattern is clearly visible.
We introduce now the auxiliary polynomials. (NOTE: The polynomial subscript indicates its degree.)

\[ A_m(x) = \sum_{i=0}^{m} a_i x^i \]  

\[ B_m(x) = \sum_{i=0}^{m} b_i x^i \]  

\[ T_m(x) = \sum_{i=0}^{m} t_i x^i \]  

Consider the product polynomial

\[ V_{2m}(x) = A_m(x) B_m(x) = \sum_{i=0}^{2m} v_i x^i \]  

It is easy to see that the coefficients of this polynomial are obtainable by the matrix product (2.28).
Comparing this to (2.23), we see that interchanging the two indicated triangular sections in (2.28) will transform the matrix of (2.28) into a trivial augmentation of the matrix of (2.23). This fact can be translated to the following relationship between \( T_m(x) \) and \( V_{2m}(x) \):

\[
T_m(x) = V_{2m}(x) - (a_{m,1} + a_{m-1,2} + \ldots + a_{1,m}) (x^{m+1} - 1) \\
- (a_{m,2} + \ldots + a_{2,m}) x^{m+1} - 1 \\
. . . \\
- (a_{m,m-1} + a_{m-1,m}) x^{m-2} (x^{m+1} - 1) \\
- a_m x^{m-1} (x^{m+1} - 1)
\]

\[
\therefore T_m(x) = V_{2m}(x) - (x^{m+1} - 1) F_{m-1}(x)
\]

where \( F_{m-1}(x) \) is the indicated polynomial of degree \((m-1)\). Hence

\[
\frac{V_{2m}(x)}{x^{m+1} - 1} = F_{m-1}(x) + \frac{T_m(x)}{x^{m+1} - 1}
\]

Note that, in the quotient on the right, the denominator degree is higher than the numerator degree. This means that \( T_m(x) \) is just the remainder obtained when dividing \( V_{2m}(x) \) by \((x^{m+1} - 1)\). In other words,

\[
T_m(x) = \{A_m(x) B_m(x)\} \mod (x^{n-1})
\]

We have made use here of the fact that the order of the LC matrix in (2.23) is \( n = m+1 \)

Eqn. (2.32) is a prescription for performing the LCT of (2.23) through polynomial manipulations identical with those of (2.1). This, then, establishes the second link.

Consider now the number of multiplications required to evaluate (2.23). Straight matrix multiplication requires \( n^2 \) scalar multiplications. A much
lower value is prescribed by Winograd's theorem \([4]\). In its narrower application to the present case, the theorem states that if \(x^n - 1\) is representable as

\[
x^n - 1 = \prod_{i=1}^{k(n)} m_i(x)
\]

(2.34)

where the \(m_i(x)\) are distinct polynomials irreducible over the field of rationals, then the minimal number of multiplications is \((2n - k)\), provided multiplications by rational numbers are not counted.

The exclusion of rational multiplications merits an explanation. Suppose we have a minimal realization of \(T_m(x)\) of the following form

\[
T_m(x) = \sum_{r=1}^{R} \frac{J_r}{K_r} F_r(x)
\]

(2.35)

where \(J_r, K_r\) are integers and in which \(F_r(x)\) involves no rational multiplications. According to the theorem, the \(F_r\)'s will require a total of \((2n - k)\) multiplications and we are essentially being told that the additional \(R\) rational multiplications appearing in (2.35) do not count. To see what is involved here, let us clear fractions in (2.35). Let \(K\) be the least common denominator and let

\[
\frac{J_r}{K_r} = \frac{J'_r}{K}
\]

(2.36)

Hence,

\[
K T_m(x) = \sum_{r=1}^{R} J'_r F_r(x)
\]

(2.37)

Each of the multiplications by \(J'_r\) can be implemented as \(J'_r - 1\) additions so that \(K T_m(x)\) does not require any multiplications above the \((2n - k)\) used in
computing the $F_r$'s. Finally, we compensate for the multiplication by $K$ on the left, by prescaling the $a$ matrix, that is, replacing $a_i$, $A_m(x)$ by

$$\hat{a}_i = \frac{a_i}{K}; \quad \hat{A}_m(x) = \sum_{i=0}^{m} \hat{a}_i x^i \quad (2.38)$$

Thus, (2.32) is now replaced by

$$T_m(x) = K[\hat{A}_m(x) B_m(x)] \text{ mod } (x^n-1) \quad (2.39)$$

and we see that multiplications by rationals can always be eliminated without an increase in the number of irrational multiplications.

It should be pointed out that while this argument is theoretically sound, practically, one should be concerned with the cost in terms of the extra additions introduced to eliminate the rational multiplications. Obviously, when these extra additions take longer than the multiplications they replace, we would be better off leaving the multiplications in. This will, indeed, be the case when $n$ is large. Therefore, the algorithm taking advantage of Winograd's theorem, has to be constructed in such a way that a DFT of large order is broken down into many LC transformations of low order. As a matter of fact, all the DFT orders appearing in Table 6 ($N_{\text{max}} = 5040$) call for just three LC transformations of orders 2, 4, 6.

The factoring of $(x^n-1)$ for these three cases is shown in Table 1 in which the last column gives the minimal number of multiplications as stated by Winograd's theorem. In the next section we develop the specific algorithms which realize these minima. These three algorithms serve as a foundation for the subsequent construction of DFT algorithms for all orders listed in (1.3).
III. The Basic LCT Algorithms

Our approach here is to present the general method first in sufficient
detail so that its application to the three specific $n$ values can be subsequently
presented as a mostly self-explanatory sequence of equations.

The starting point is (2.39) in which $K$ is left indeterminate till the
very end of the derivation. The factoring of $x^n - 1$ is spelled out in Table 1
which identifies the $m_i(x)$ factors of (2.34). With the $m_i$'s available, we
evaluate $T_m(x)$ in a two-phase scheme based on the polynomial version of the Chinese
Remainder Theorem [7]. In phase 1 we compute

$$u_i(x) = \left(\hat{A}_m(x) \cdot B_m(x)\right) \mod m_i(x) \quad (3.2)$$

for all $i$ of (2.34). This is based entirely on results established in the Appendix
and summarized in Tables A1, A2 there. In phase 2 we use the polynomial version
of Garner's algorithm [7] to construct $T_m(x)$ from the $u_i$'s. This calls for
the auxiliary functions $c_i(x)$, $v_i(x)$ introduced below. Their utilization in
the construction of $T_m(x)$ is spelled out in (3.6).

$$v_1(x) = u_1(x) \quad (3.4)$$

$$v_i(x) = \left\{ \left( \left( \left( u_1(x) - v_1(x) \right) c_{1,1}(x) - v_2(x) \right) c_{2,1}(x) - v_3(x) \right) c_{3,1}(x) - \ldots, -v_{i-1}(x) \right) c_{i-1,1}(x) \right\} \mod m_i(x) \quad (3.5)$$

$$T_m(x) = K \left\{ v_1(x) + \sum_{i=2}^{k(m-1)} v_i(x) \prod_{j=1}^{i-1} m_j(x) \right\} \quad (k \text{ from Table 1}) \quad (3.6)$$

The computation of $c_{ij}(x)$ is trivial when $m_j(x)$ is of degree 1, that is,
$m_j(x) = x - x_j$. From (A.5) in the Appendix, we see that, in this case,
Table 1

Rational Factorization of $x^n - 1$

<table>
<thead>
<tr>
<th>n</th>
<th>$x^n - 1$</th>
<th>k</th>
<th>2n-k</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$(x-1)(x+1)$</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>$(x-1)(x+1)(x^2+1)$</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>$(x-1)(x+1)(x^2+x+1)(x^2-x+1)$</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>
\begin{align}
m_1(x_j)c_{ij}(x_j) &= 1 
\end{align}

Now, any \( c_{ij}(x) \) satisfying (3.3) (and hence (3.7)) would do. Choosing the lowest degree we get

\begin{align}
c_{ij}(x) &= c_{ij} = \frac{1}{m_1(x_j)} 
\end{align}

With these derivation outlines spelled out, we turn now to specific cases. To establish the evolving pattern, we follow these outlines even in the low order case (\( n=2 \)) where direct derivation could be simpler.

**Left-Circulant Transformation of Order 2 (Fig. 1)**

\[
\begin{bmatrix}
t_1 \\
t_0
\end{bmatrix} = \begin{bmatrix}
a_1 & a_0 \\
a_0 & a_1
\end{bmatrix} \begin{bmatrix}
b_0 \\
b_1
\end{bmatrix}
\]

\[
\hat{a}_1 = \frac{a_1}{\lambda_1}; \quad \hat{A}_1(x) = \sum_{i=0}^{1} \hat{a}_1 x^i; \quad B_1(x) = \sum_{i=0}^{1} b_1 x^i
\]

**Phase 1**

\[
T_1(x) = \frac{K\{\hat{A}_1(x) \ B_1(x)\}}{m_1 m_2} \mod \frac{(x-1)(x+1)}{m_1 m_2}
\]

\[
u_1(x) = \{\hat{A}_1(x) \ B_1(x)\} \mod (x-1)
\]

\[
= \frac{(\hat{a}_0+\hat{a}_1) (b_0+b_1)}{\alpha_1 \beta_1} \quad \text{(see (A.5))}
\]

\[
\delta_1 = u_1 = \alpha_1 \beta_1
\]

\[
u_2(x) = \{\hat{A}_1(x) \ B_1(x)\} \mod (x+1)
\]

\[
= \frac{(\hat{a}_0-\hat{a}_1) (b_0-b_1)}{\alpha_2 \beta_2}
\]

\[
\delta_2 = u_2 = \alpha_2 \beta_2
\]
Phase 2

\[ c_{12}(x) = \frac{1}{m_1(x_2)} = \frac{1}{m_1(-1)} = -\frac{1}{2} \]

\[ v_1 = \delta_1 \]

\[ v_2 = (\delta_2 - \delta_1)(-\frac{1}{2}) \text{ mod } (x+1) = \frac{1}{2}(\delta_1 - \delta_2) \]

\[ T_1(x) = K \left[ \delta_1 + \frac{1}{2} (\delta_1 - \delta_2)(x-1) \right] = \frac{K}{2} (\delta_1 - \delta_2) x + \frac{K}{2} (\delta_1 + \delta_2) \]

(3.11)

The obvious choice here is \( K = 2 \) so that the final result is

\[ t_0 = \delta_1 + \delta_2; \quad t_1 = \delta_1 - \delta_2 \]

The algorithm is summarized graphically in the tableau of Figure 1. The conventions adopted here are quite simple and are also the ones adopted in the more complex tableaux presented later on. We defined \( \beta_i \) as a linear combination of the \( b_j \)'s. The \( \beta_i \) row lists the (non-zero) coefficients of this linear combination. Similarly, we found that \( t_i \) is a linear combination of the \( \delta_j \)'s. The \( t_i \) column lists the coefficients of this linear combination. The left corner arrow indicates that the \( \beta_i \)'s are derived from the \( b_j \)'s and not the other way around. Usually there is no directional ambiguity so that the arrows may be omitted. The equations \( \delta_i = \beta_i \alpha_i \) appear explicitly in the tableau using the Fortran multiplication symbol.

Finally, recall that the basic eqn. (2.39) is fully symmetric with respect to \( \{\hat{a}_j\}, \{b_j\} \). We have taken advantage of this in the terminology introduced here. Thus, coupled with each \( \beta_i \) which is a specific function of \( \{b_j\} \) (say, \( \phi_i((b_j)) \)) spelled out in the tableau, there is the variable \( \alpha_i \) which is exactly the same function of \( \{\hat{a}_j\} \), that is,

\[ \alpha_i = \phi_i( \{a_{ij} \} / K) \]

This convention is adhered to throughout the paper. Practically, this means that
Fig. 1. Algorithm for LCT of order 2

\[ b_0 \quad b_1 \quad t_1 \quad t_0 \]
\[
\begin{array}{cccc}
1 & 1 & \beta_1 & \star a_1 = t_1 \\
1 & -1 & \beta_2 & \star a_2 = t_0 \\
\end{array}
\]

\[ a_1 = \phi_1 (\{a_j\}) \]
\[ \beta_1 = \phi_1 (\{b_j\}) \]

\[ n = 2 \quad (2M; 4A) \]
the left part of the tableau has to be run through twice. First, with \( \{a_j\}_K \) replacing \( \{b_j\} \) and thus yielding \( \{a_1\} \) instead of \( \{\beta_1\} \), then we run through the full tableau with \( \{b_j\} \) as input. Note, however, that in spite of the mathematical symmetry between \( \{a_j\}_K \) and \( \{b_j\} \), practically, there is an important difference between them. \( a \) is considered a constant matrix transforming a number of different data vectors \( b \). Therefore, the \( a_1 \)'s may be precomputed once and for all, their computation being ignored in accounting for the cost of transforming one data vector \( b \). With this in mind, we count only the number of explicit arithmetic operations in the tableau, arriving at 2 multiplications and 4 additions, for which we adopt the designation \( (2M; 4A) \) appearing in the figure. Note that the 2 multiplications are the minimum prescribed by Winograd's theorem (right column in Table 1).

**Left Circulant Transformation of Order 4 (Fig. 2)**

In this and all other tableaus derived in this paper, it is suggested that the reader consider the tableau as he follows its derivation, noting the graphical representation of each mathematical statement as the algorithm evolves.

\[
\begin{bmatrix}
t_3 \\
t_2 \\
t_1 \\
t_0 \\
\end{bmatrix} = \begin{bmatrix}
a_3 & a_2 & a_1 & a_0 \\
a_2 & a_1 & a_0 & a_3 \\
a_1 & a_0 & a_3 & a_2 \\
a_0 & a_3 & a_2 & a_1 \\
\end{bmatrix} \begin{bmatrix}
b_0 \\
b_1 \\
b_2 \\
b_3 \\
\end{bmatrix}
\]

\[
\hat{a}_1 = \frac{a_1}{K} \\
\hat{A}_3(x) = \sum_{i=0}^{3} \hat{a}_i x^i \\
B_3(x) = \sum_{i=0}^{3} b_i x^i
\]

\[(3.13)\]

**Phase 1**

\[
T_3(x) = K \{\hat{A}_3(x)B_3(x)\} \mod\{(x^2+1)(x+1)(x-1)\}
\]

\[
u_3(x) = \hat{A}_3(1)B_3(1) = \left(\hat{a}_0 + \hat{a}_1 + \hat{a}_2 + \hat{a}_3\right) \left(b_0 + b_1 + b_2 + b_3\right)
\]

\[
c_1 = b_0 + b_2; \quad c_2 = b_1 + b_3
\]

\[\therefore \beta_1 = c_1 + c_2\]

\[
\delta_1 = u_3 = a_1 \beta_1
\]
\begin{align*}
\beta_1 \ast a_1 &= 8_1 = 1 \\
\beta_2 \ast a_2 &= 8_2 = -1 \\
\beta_3 \ast 2a_5 &= 8_3 = 1 \\
\beta_4 \ast 2(a_4 - a_3) &= 8_4 = 1 \\
\beta_5 \ast (-2a_4) &= 8_5 = 1 \\
\end{align*}

\begin{align*}
\alpha_i &= \phi_i \left\{ \frac{a_i}{4} \right\} \\
\beta_i &= \phi_i \left\{ b_i \right\} \\
\end{align*}

\text{Fig. 2. Algorithm for LCT of order 4}
\[ u_2(x) = \lambda_3(-1)B_3(-1) = \frac{(\hat{a}_0 - \hat{a}_2 + \hat{a}_3)(b_0 - b_1 + b_2 - b_3)}{a_2 \beta_2} \]

\[ \therefore \beta_2 = c_1 - c_2 \]

\[ \delta_2 = u_2 = a_2 \beta_2 \]

\( u_1(x) \) is evaluated in two steps:

\[ \lambda_3(x) \mod (x^2 + 1) = \left( \frac{\hat{a}_1 - \hat{a}_3}{a_3} x + \frac{\hat{a}_0 - \hat{a}_2}{a_4} \right) \]  
\[ \text{(from Table A1)} \]

\[ B_3(x) \mod (x^2 + 1) = \left( \frac{b_1 - b_3}{\beta_3} x + \frac{b_0 - b_2}{\beta_4} \right) \]

Note that the tableau of Fig. 2 defines \( \beta_3, \beta_4 \) indirectly in terms of

\[ c_3 = b_1 - b_3; \quad c_4 = b_0 - b_2 \]

Thus,

\[ \beta_3 = c_3; \quad \beta_4 = c_4 \]

so that no arithmetic operations are involved in the last two equations. Whenever this is the case, we circle the relevant terms in the tableau to stress this fact.

Now we combine the two results of (3.15) using Table A2 (\( \alpha_3 = p_1; \beta_3 = q_1; \) etc.) getting

\[ u_1(x) = \left[ -\left( \alpha_4 - \alpha_3 \right) \beta_4 + \alpha_4 \left( \beta_3 + \beta_4 \right) \right] x + \left[ \alpha_4 \left( \beta_3 + \beta_4 \right) - \left( \alpha_3 + \alpha_4 \right) \beta_3 \right] \]

We introduce now

\[ \alpha_5 = \alpha_3 + \alpha_4; \quad \beta_5 = \beta_3 + \beta_4 = c_3 + c_4 \]

\[ \delta_3 = 2 \alpha_5 \beta_3; \quad \delta_4 = 2 \left( \alpha_4 - \alpha_3 \right) \beta_4; \quad \delta_5 = -2 \alpha_4 \beta_5 \]

Hence,

\[ u_1(x) = -\frac{1}{2} \left( \delta_4 + \delta_5 \right) x - \frac{1}{2} \left( \delta_3 + \delta_4 \right) \]
Phase 2

\[ c_{12} = \frac{1}{m_1(x_2)} - \frac{1}{m_1(-1)}, \quad c_{13} = \frac{1}{m_1(x_3)} - \frac{1}{m_1(1)}, \quad c_{23} = \frac{1}{m_2(x_3)} - \frac{1}{m_2(1)} \]

\[ v_1(x) = u_1(x) = -\frac{1}{2}(e_4 x + e_3) \]

\[ v_2 = \left[ (\delta_2 + \frac{1}{2} e_4 x + \frac{1}{2} e_3) \frac{1}{2} \right] \mod (x+1) = \frac{1}{4}(2\delta_2 - e_4 + e_3) \]

\[ v_3 = \left[ (\delta_1 - \frac{1}{2} e_4 x + \frac{1}{2} e_3) \frac{1}{2} - \frac{1}{4}(2\delta_2 - e_4 + e_3) \right] \mod (x-1) = \frac{1}{4}(\delta_1 + \delta_2 + e_4) \]

\[ T_3(x) = -\frac{K}{2}(e_4 x + e_3) + \frac{K}{4}(2\delta_2 - e_4 + e_3)(x^2 + 1) + \frac{K}{4}(\delta_1 - \delta_2 + e_4)(x^3 + x^2 + x + 1) \]

Adopting

\[ K = 4, \]

\[ T_3(x) = (\delta_1 - \delta_2 + e_4)x^3 + (\delta_1 + \delta_2 + e_3)x^2 + (\delta_1 - \delta_2 - e_4)x + (\delta_1 + \delta_2 - e_3) \]

Introducing now

\[ e_1 = \delta_1 - \delta_2; \quad e_2 = \delta_1 + \delta_2 \]

we get the final result

\[ T_3(x) = (e_1 + e_4)x^3 + (e_2 - e_3)x^2 + (e_1 - e_4)x + (e_2 - e_3) \quad (3.16) \]

A count of arithmetic operations in the tableau (excluding the \( a_1 \) manipulations) yields 5 multiplications and 15 additions \(^6\), again realizing the multiplication minimum of Table 1.

**Left-Circulant Transformation of order 6 (Fig. 4)**

\[
\begin{bmatrix}
 t_5 \\
 t_4 \\
 t_3 \\
 t_2 \\
 t_1 \\
 t_0
\end{bmatrix}
= \begin{bmatrix}
 a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\
 a_4 & a_3 & a_2 & a_1 & a_0 & a_5 \\
 a_3 & a_2 & a_1 & a_0 & a_5 & a_4 \\
 a_2 & a_1 & a_0 & a_5 & a_4 & a_3 \\
 a_1 & a_0 & a_5 & a_4 & a_3 & a_2 \\
 a_0 & a_5 & a_4 & a_3 & a_2 & a_1
\end{bmatrix}
= \begin{bmatrix}
 b_0 \\
 b_1 \\
 b_2 \\
 b_3 \\
 b_4 \\
 b_5
\end{bmatrix}
\]

\[ \hat{a}_1 = \frac{a_1}{K} \]

\[ \hat{b}_j(x) = \sum_{i=0}^{5} a_i x^i \]

\[ b_j(x) = \sum_{i=0}^{5} b_i x^i \quad (3.17) \]

\(^6\)The author wishes to acknowledge here the help of Dr. R. G. Lipes of JPL in reducing the number of additions from 16 to 15.
Phase 1

\[ T_5(x) = K(\hat{A}_5(x)B_5(x)) \mod \left(\frac{x^2-x+1}{m_1} \cdot \frac{x^2+x+1}{m_2} \cdot \frac{x+1}{m_3} \cdot \frac{x-1}{m_4}\right) \tag{3.18} \]

\[ u_4 = \hat{A}_5(1)B_5(1) = \left(\sum_{i=0}^{5} \beta_1 \right) \left(\sum_{i=0}^{5} b_1 \right) \]

\[ c_1 = b_3+b_0; \ c_2 = b_5+b_2; \ c_3 = b_4+b_1 \]

\[ u_4 = \left(\sum_{i=0}^{5} \alpha_i \right) \left(\frac{c_1+c_2+c_3}{\beta_1} \right) \]

\[ \delta_1 = u_4 = \alpha_1 \beta_1 \]

\[ u_3 = \hat{A}_5(-1)B_5(-1) = \left(-\delta_0+\delta_1-\delta_2+\delta_3-\delta_4+\delta_5\right) \left(-b_0+b_1-b_2+b_3-b_4+b_5\right) \]

\[ c_6 = b_3-b_0; \ c_5=b_5-b_2; \ c_4 = b_4-b_1 \]

\[ u_3 = \alpha_6 \left(\frac{c_6+c_5-c_4}{\beta_6} \right) \]

\[ \delta_6 = u_3 = \alpha_6 \beta_6 \]

\[ u_1(x), u_2(x) \text{ are evaluated in two steps} \]

\[ B_5(x) \mod (x^2+x+1) = \left[ \frac{b_4+b_1}{c_3} \right] x + \left[ \frac{b_5+b_2}{c_2} \right] \quad \text{(from Table A1)} \]

\[ = \left(\frac{c_3-c_2}{\beta_2} \right) x + \left(\frac{c_1-c_2}{\beta_3} \right) = \beta_2 x + \beta_3 \]

\[ \therefore \hat{A}_5(x) \mod (x^2+x+1) = \alpha_2 x + \alpha_3 \]
We introduce now:

\[ S_7 = S_3 - S_2 = (\alpha_3 - \alpha_2) (\beta_3 - \beta_2) \]

(3.19)  

(from Table A2)

\[ 0.7 = 0.3 - 0.2 \]

and use these to transform (3.19) into

\[ u_2(x) = (0.2 f_3^3 + 0.7 f_3^2) x + (0.7 f_3^2 + 0.3 f_3^7) \]

(3.21)  

(from Table A1)

\[ \beta_7 = \beta_3 - \beta_2 = 0.5 - 0.3 = 0.2 \]

(3.22)  

\[ \alpha_7 = \alpha_3 - \alpha_2 \]

(3.23)  

\[ u_1(x) = (\hat{A}_7(x) B_5(x)) \mod (x^2 + x + 1) = \left[ (\alpha_4 + \alpha_5) (\beta_4 + \beta_5) - \alpha_4 \beta_5 \right] x + (\alpha_4 \beta_4 - \alpha_5 \beta_5) \]

(3.24)  

(from Table A2)

\[ \hat{A}_5(x) \mod (x^2 + x + 1) = -\alpha_5 x - \alpha_4 \]

\[ B_5(x) \mod (x^2 + x + 1) = -\left[ (b_5^5 - b_5^4 + b_4^5 - b_4^4) x - \left( \frac{b_3^5 - b_3^4 + b_4^5 - b_4^4}{c_6^5 - c_6^4} \right) \right] \]

\[ = - \left( \frac{c_5^5 + c_4^5}{c_5^4 + c_4^4} x - (c_6^5 - c_6^4) \right) = -\beta_4 x \]

\[ = \hat{A}_5(x) \mod (x^2 + x + 1) = -\alpha_5 x - \alpha_4 \]

Phase 2

\[ \{ m_1(x) c_{12}(x) \} \mod m_2(x) = 1 \]

Let \( c_{12}(x) = \gamma_1 x + \gamma_0 \)

\[ m_1(x) c_{12}(x) = (x^2 + x + 1)(\gamma_1 x + \gamma_0) = \gamma_1 x^3 + (\gamma_0 - \gamma_1) x^2 + (\gamma_1 x + \gamma_0) x + \gamma_0 \]

\[ \gamma_1 - \gamma_0 + \gamma_1 + \gamma_0 = 1 \]

(3.25)  

(from Table A1)

\[ \gamma_1 = \gamma_0 = \frac{1}{2}; \quad c_{12}(x) = \frac{1}{2} (x + 1) \]

The \( e_i \)'s defined here, do not appear in the final tableau and are used only in the intermediate steps.
Collecting equal power terms yields the desired $t_i$'s. We make now the obvious choice $K=6$ and present the results in the tableau format in Fig. 3a. We note here that the coefficients of $t_0$ and $t_3$ have identical magnitudes. On the left of the bisecting line, the coefficients themselves are identical whereas to the right of it, they have opposite signs. This holds true also for the pairs $(t_1, t_4), (t_2, t_5)$. Taking advantage of these symmetries, we can reduce the number of additions as shown in Fig. 3b.

The final manipulations involve the elimination of the $e_i$'s from Fig. 3b. This is based on their definitions (3.21), (3.24) and on a judicious application of (3.20), (3.23) as follows:
Fig. 3. Stages in the development of the algorithm for LCT of order 6
This completes the derivation.

\[ g_1 = e_1 - e_2 = -2\alpha_2\beta_2 - \alpha_2(\beta_7 + \beta_2) - \alpha_3\beta_7 = \beta_2(\alpha_3 - \alpha_7) + \beta_7(\alpha_2 - \alpha_3) \]  
\( \delta_2 \quad \delta_7 \)  
\[ g_2 = e_4 - e_5 = \alpha_5(\beta_8 - \beta_5) + 2\alpha_5\beta_5 - \alpha_4\beta_8 = \beta_5(\alpha_8 + \alpha_5) + \beta_8(\alpha_4 - \alpha_5) \]  
\( \delta_5 \quad \delta_8 \)  
\[ g_3 = e_2 + e_3 = \alpha_7(\beta_3 - \beta_7) - \alpha_2\beta_3 + 2\alpha_3\beta_7 = \beta_3(\alpha_7 - \alpha_2) - \beta_7(\alpha_2 - \alpha_3) \]  
\( \delta_3 \quad \delta_7 \)  
\[ g_4 = e_5 + e_6 = \alpha_8(\beta_8 - \beta_4) + \alpha_5\beta_4 + 2\alpha_4\beta_8 = \beta_4(\alpha_8 + \alpha_5) - \beta_8(\alpha_4 - \alpha_5) \]  
\( \delta_4 \quad \delta_8 \)  
\[ g_5 = e_2 + e_3 = \alpha_7(\beta_3 - \beta_7) - \alpha_2\beta_3 + 2\alpha_3\beta_7 = \beta_3(\alpha_7 - \alpha_2) - \beta_7(\alpha_2 - \alpha_3) \]  
\( \delta_3 \quad \delta_7 \)  
\[ g_6 = e_5 + e_6 = \alpha_8(\beta_8 - \beta_4) + \alpha_5\beta_4 + 2\alpha_4\beta_8 = \beta_4(\alpha_8 + \alpha_5) - \beta_8(\alpha_4 - \alpha_5) \]  
\( \delta_4 \quad \delta_8 \)  
\[ g_7 = e_2 + e_3 = \alpha_7(\beta_3 - \beta_7) - \alpha_2\beta_3 + 2\alpha_3\beta_7 = \beta_3(\alpha_7 - \alpha_2) - \beta_7(\alpha_2 - \alpha_3) \]  
\( \delta_3 \quad \delta_7 \)  
\[ g_8 = e_5 + e_6 = \alpha_8(\beta_8 - \beta_4) + \alpha_5\beta_4 + 2\alpha_4\beta_8 = \beta_4(\alpha_8 + \alpha_5) - \beta_8(\alpha_4 - \alpha_5) \]  
\( \delta_4 \quad \delta_8 \)  

The corresponding tableau derived in the first version of this paper contained 6 extra additions. The subsequent appearance in print of the prescription of Winograd's algorithm [10], provided the clue to the specific manipulations ((3.25), etc.) utilized here to eliminate these extra additions.
\[ a_1 = \phi_1 \left( \frac{a_1}{6} \right) \]
\[ \beta_1 = \phi_1 \left( b_1 \right) \]

\[ n = 6 \ (8M; 34A) \]

Fig. 4. Algorithm for LCT of order 6
IV. The Basic DFT Algorithms for Prime N

In this section, we apply the tableaus just derived to obtain the DFT algorithms for the odd prime terms of list (1.3), namely, 3, 5, 7. As indicated in Section I, these will serve as building blocks for higher order DFT's.

We have seen in section II that with the proper relabeling, an N-th order DFT matrix displays an n-th order LC submatrix (2.7). The main part of the contribution of this submatrix to the overall transformation is spelled out in (2.15) repeated here

\[ B'_{\rho} = \sum_{\sigma=0}^{n-1} \Omega (g^{\rho+n\sigma}) b_{\sigma} \quad (\rho = 0, 1, \ldots, n-1) \]  

(4.1)

On the other hand, the LC tableaus of the last section are based on the (2.23), (2.33) formulation of the n-th order LC transformation. Therefore, in applying the LCT tableaus to the LC transformation expressed in (4.1), we must adopt the following identifications

\[ B'_{\rho} = t^{n-1-\rho} \quad (\rho = 0, 1, \ldots, n-1) \]  

(4.2)

\[ a_{\rho} = \Omega (g^{n-1-\rho}) \quad (\rho = 0, 1, \ldots, n-1) \]  

(4.3)

Note the effect of (4.3). The LCT tableaus, being general, provide only a prescription for the computation of the \( \alpha_i \)'s from the \( a_i \)'s. However, since (4.3) provides an explicit formula for the \( a_i \)'s, the \( \alpha_i \)'s may actually be computed. Specifically, the \( \alpha_i \)'s are expressible as

\[ \alpha_i = \Omega \varepsilon_i \]  

(4.4)

in which the \( \varepsilon_i \)'s are functions of \( i, N \) only and can thus be precomputed once and for all.

We copy now the remaining equations of the relabeled DFT ((2.13), (2.19), (2.20))

\[ B(0) = b(0) + \sum_{\sigma=0}^{n-1} b_{\sigma} \]  

(4.7)

Note that eqns. (4.6), (4.7) are valid only for the special case of prime N and are based on
Eqns. (4.1)-(4.5), on the other hand, are quite general and will also be applied in the next two sections where \( N \) is not prime.

Our first step in the construction of the DFT tableau for prime \( N \) is the computation of the \( \epsilon_i(N) \) constants. This is done by evaluating the \( a_p \)'s from (4.3) and then using them in the LCT tableau of order \( N-1 \) to compute the \( a_1 \)'s, and hence the \( \epsilon_i \)'s (4.4).

The next step is to copy the LCT tableau of order \( N-1 \) into the DFT tableau of order \( N \). This is equivalent to the implementation of (4.1) and yields a tableau transforming the \( b_i \)'s into the \( t_j \)'s. Now, using (2.9)-(2.11), (4.2) we replace these variables by \( f_i \)'s and \( F_j \)'s. The input and output index sequences we get at this point will usually be non-monotonic. Therefore, we now permute the rows/columns of the input and output squares to achieve index monotonicity.

It should be pointed out that the variable changes (2.9)-(2.11) were introduced to expose the LC structure and thus allow the application of (2.39). Once this has been done, however, we want the resulting DFT tableau to be expressed in terms of the original variables \( F_u, f_v \) with standard monotonic index sequences since this simplifies the integration of the tableau into the algorithm for larger \( N \).

We turn now to the implementation of (4.7). Since all three LCT tableaus satisfy

\[
\beta_1 = \sum_{i=0}^{n-1} b_i, \tag{4.9}
\]

eqn. (4.7) is equivalent to

\[
F_0 = B^\prime(0) = \Omega(b(0) + \beta_1) = \Omega(f_0 + \beta_1) \tag{4.10}
\]

Finally, in order to efficiently realize (4.5), we examine the three LCT tableaus to see whether they contain a term which appears as a component with coefficient +1 of all \( t_i \)'s. This term turns out to be \( \delta_1 \).

Hence, replacing \( \delta_1 = \alpha_1 \beta_1 \) by

\[
\delta_1 = \Omega b(0)^\prime \alpha_1 \beta_1 \tag{4.11}
\]

will convert the output from \( B^\prime_\rho \) to \( B_\rho \). Note, however, that the term \( \Omega b(0) \) is
already included in \( B(0) \) computed in (4.10). This raises the possibility of eliminating one of the two multiplications evident in (4.11). Indeed, combining (4.10) with (4.11) yields (using (4.4))

\[
\delta_1 = B(0) + (e_1 - 1)\Omega \beta_1 = F_0 + (e_1 - 1)\Omega \beta_1
\]  

(4.12)

The multiplier of \( \beta_1 \) is seen here to be the product of \( \Omega \) and a function of \( N \). This turns out to be the general pattern for all \( \beta_1 \) in all the DFT tableaus yet to be derived. Hence, we introduce now the notation

\[
\xi_1 = (\omega_1 \Omega) \beta_1
\]  

(4.13)

where \( \omega_1 \) is a function of \( i, N \) only. \( \beta_1 \) is always initially transformed as in (4.13). Hence one of our tasks in constructing the DFT tableaus is the determination of the numerical values of the \( \omega_1 \) constants. As we shall see in the course of the developments, the underlying LCT tableaus in conjunction with (4.4), (4.13), always uniquely determine the \( \omega_1 \)'s. In the case of (4.12)

\[
\omega_1 = e_1^{-1}; \quad \delta_1 = B(0) + (\omega_1 \Omega) \beta_1 = F_0 + (\omega_1 \Omega) \beta_1
\]  

(4.14)

So far, we have considered the LCT-DFT transition in general terms. We turn now to specific cases:

**DFT of order 3 (Fig. 5)**

Eqn. (4.10) is explicitly stated in the tableau. From (4.3) (with \( W = e^{-\frac{2\pi i}{3}} \)) we get

\[
\begin{bmatrix}
a_0/2 \\
a_1/2
\end{bmatrix}
= \begin{bmatrix}
\Omega/2 \\
\Omega^1/2
\end{bmatrix}
= \begin{bmatrix}
W \\
W^1
\end{bmatrix}
\]  

(4.15)

where \( \bar{W} \) is the complex conjugate of \( W \). Hence, applying the second order LCT tableau (Fig. 1), we find

\[
\begin{array}{ccc}
\Omega^{1/2} & \Omega^1/2 \\
1 & 1 & \alpha_1 \\
1 & -1 & \alpha_2
\end{array}
\]  

\[
= \begin{array}{c}
\Omega/2 \\
-\Omega/2
\end{array}; \quad e_1 = -\frac{1}{2}; \quad \omega_1 = e_1^{-1} = -\frac{3}{2}
\]  

\[
= i\sqrt{3} \Omega; \quad e_2 = i\sqrt{3}/2; \quad \omega_2 = e_2 = i\sqrt{3}/2
\]
Fig. 5. Algorithm for DFT of order 3

Fig. 6. Algorithm for DFT of order 5

\[ \beta_0 \]
\[ (f_0 + \beta_1) \ast \omega = F_0 \]
\[ \beta_1 \]
\[ (f_0 + \beta_1) \ast (-\frac{3}{2})\omega + F_0 = \delta_1, \delta_2 \]
\[ \beta_2 \]
\[ (f_0 + \beta_1) \ast (i\sqrt{3}/2)\omega = \delta_1, \delta_2 \]

\[ N = 3 \ (3M; 6A) \]

\[ N = 5 \ (6M; 17A) \]
DFT of order 5 (Fig. 6)

\[ W = e^{-i \frac{2\pi}{5}}; \quad g = 2. \]  Hence, from (4.3),

\[
\begin{bmatrix}
\frac{1}{4} a_0 \\
\frac{1}{4} a_1 \\
\frac{1}{4} a_2 \\
\frac{1}{4} a_3
\end{bmatrix}
= \begin{bmatrix}
W^3 \\
W^4 \\
W^2 \\
W^1
\end{bmatrix}
= \begin{bmatrix}
\frac{1}{4} W^2 \\
\frac{1}{4} W^1 \\
\frac{1}{4} W^2 \\
\frac{1}{4} W^1
\end{bmatrix}
\]

(4.16)

The \(a_i\)'s are determined from the following prescription of Fig. 2:

\[
\begin{array}{c|ccc}
\frac{\Omega}{4} W^2 & 1 & 1 \\
\frac{\Omega}{4} W^1 & 1 & 1 \\
\frac{\Omega}{4} W^2 & 1 & -1 \\
\frac{\Omega}{4} W^1 & 1 & -1 \\
\end{array}
\]

\[
\begin{align*}
c_1 &= -\frac{\Omega}{2} \cos 36^\circ \\
c_2 &= \frac{\Omega}{2} \cos 72^\circ \\
c_3 &= \frac{\Omega}{2} \sin 72^\circ \\
c_4 &= \frac{\Omega}{2} \sin 36^\circ \\
\end{align*}
\]

(4.17)

(4.4) and (4.17) prescribe the \(\varepsilon_i\)'s. Finally, from (4.14) and the \(\beta_i\) multipliers in Fig. 2 we get
\[
\begin{align*}
\omega_1 &= e^{i(\theta - 1)} = -\frac{5}{4} \\
\omega_2 &= e^{i2\theta} = -\frac{1}{2} (\cos 36^\circ + \cos 72^\circ) \\
\omega_3 &= 2e^{i\theta} = i (\sin 36^\circ + \sin 72^\circ) \\
\omega_4 &= 2(i e^{i\theta} - e^{-i\theta}) = i(\sin 36^\circ - \sin 72^\circ) \\
\omega_5 &= -2e^{i\theta} = -i \sin 36^\circ \\
\end{align*}
\]

DFT of order 7 (Fig. 8)

\[
W = e^{-\frac{2\pi}{7}}; \ g = 3. \ \text{Hence from (4.3)}
\]

\[
\begin{bmatrix}
ap_0 \\
ap_1 \\
ap_2 \\
ap_3 \\
ap_4 \\
ap_5 \\
\end{bmatrix}
= \begin{bmatrix}
W^0 \\
W^1 \\
W^2 \\
W^3 \\
W^4 \\
W^5 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
ap_0 \\
ap_1 \\
ap_2 \\
ap_3 \\
ap_4 \\
ap_5 \\
\end{bmatrix}
= \begin{bmatrix}
\Omega^0 \\
\Omega^1 \\
\Omega^2 \\
\Omega^3 \\
\Omega^4 \\
\Omega^5 \\
\end{bmatrix}
\]

All \(\alpha_i\)'s are expressible in terms of the angle

\[
\theta = \frac{90^\circ}{7}
\]

and its multiples. The prescription of Fig. 4 for the computation of the \(\alpha_i\)'s is shown in Fig. 7 and the final tableau based on that is shown in Fig. 8.
\[
\begin{array}{ccccccc}
\Omega \bar{W}^2 & 1 & 1 & 1 & a_1 &= -\frac{\Omega}{6} \\
\Omega \bar{W}^3 & 1 & -1 & 1 & a_2 &= -\frac{\Omega}{3} (\cos 2\theta + \cos 4\theta) \\
\Omega \bar{W}^1 & 1 & -1 & -1 & a_3 &= -\frac{\Omega}{3} (\sin \theta + \cos 4\theta) \\
\Omega \bar{W}^2 & 1 & 1 & -1 & a_4 &= -\frac{\Omega}{3} (\cos \theta - \sin 4\theta) \\
\Omega \bar{W}^3 & 1 & 1 & 1 & a_5 &= -\frac{\Omega}{3} (\sin 2\theta + \cos 4\theta) \\
\Omega \bar{W}^4 & 1 & -1 & -1 & a_6 &= -\frac{\Omega}{3} (\cos \theta - \sin 2\theta + \sin 4\theta) \\
\Omega \bar{W}^5 & 1 & -1 & 1 & a_7 &= -\frac{\Omega}{3} (\sin \theta - \cos 2\theta) \\
\Omega \bar{W}^6 & 1 & 1 & 1 & a_8 &= -\frac{\Omega}{3} (\cos \theta + \sin 2\theta) \\
\end{array}
\]

\[
\begin{align*}
c_1 &= -\frac{\Omega}{3} \sin \theta \\
c_2 &= \frac{\Omega}{3} \cos 4\theta \\
c_3 &= \frac{\Omega}{3} \cos 2\theta \\
c_4 &= \frac{\Omega}{3} \sin 2\theta \\
c_5 &= \frac{\Omega}{3} \sin 4\theta \\
c_6 &= \frac{\Omega}{3} \cos \theta \\
\end{align*}
\]

Fig. 7. Computation of \(\omega_i\)'s for the DFT algorithm of order 7
\[ (f_0 + \beta_1) \Omega = F_0 \]

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ c_1 c_2 c_3 c_4 c_5 c_6 \]

\[ \beta_1 \]

\[ \omega_1 \Omega + F_0 = \]

\[ \omega_2 \Omega = \]

\[ \omega_3 \Omega = \]

\[ \omega_4 \Omega = \]

\[ \omega_5 \Omega = \]

\[ \omega_6 \Omega = \]

\[ \omega_7 \Omega = \]

\[ \omega_8 \Omega = \]

\[ \frac{\theta}{7} \]

\[ \omega_1 = -\frac{7}{6} \]

\[ \omega_5 = -\frac{1}{3} (2 \cos \theta + \sin 2\theta - \sin 4\theta) \]

\[ \omega_2 = \frac{1}{3} (2 \sin \theta - \cos 2\theta + \cos 4\theta) \]

\[ \omega_6 = -\frac{1}{3} (\cos \theta + \sin 2\theta + 2 \cos 4\theta) \]

\[ \omega_3 = \frac{1}{3} (-\sin \theta + 2 \cos 2\theta + \cos 4\theta) \]

\[ \omega_7 = \frac{1}{3} (\sin \theta + \cos 2\theta + 2 \cos 4\theta) \]

\[ \omega_4 = -\frac{1}{3} (\cos \theta + 2 \sin 2\theta + \sin 4\theta) \]

\[ \omega_8 = -\frac{1}{3} (-\cos \theta + \sin 2\theta + 2 \sin 4\theta) \]

**Fig. 8. Algorithm for DFT of order 7**
V. The Basic DFT Algorithms for $N=4,9$

From (1.2) we see that with the tableaus of the last section, the highest realizable $N$ would be $105(=3\cdot5\cdot7)$. With this in mind, we add now the trivial algorithm for $N=2$ (Fig. 9) thus increasing the maximal $N$ to 210.

If we want still higher $N$, we may either generate new tableaus for successively higher primes (11, 13, 17,...), or devise new tableaus for $N=p^k$ (p prime). We adopt the latter course here.

DFT of order 4 (Fig. 10)

$N=2^2$, yielding (2.7),

\[ n=2 \]

The primitive root of 4 is 3. Hence,

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r = 3^\rho \mod 4$</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

The number of rows excluded from the LC pattern is 2. This number increases still further in the subsequent tableaus, reaching 8 for $N=16$. This calls for some new and some modified terminology to facilitate handling the non-LC part of the matrix.

First we extend the definition of $r$ so that (2.10) will now include (2.11)

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>0</th>
<th>1</th>
<th>(0)</th>
<th>(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Similarly, for the column indices $\sigma, s$, we now have

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>0</th>
<th>1</th>
<th>(0)</th>
<th>(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

In subsequent derivations, here and in the next section, we shall assume without explicitly so stating that the definitions of $r,s$ have been extended, as indicated here, to cover all indices. Next, we complement (2.13)-(2.15) with
Fig. 9. Algorithm for DFT of order 2
Finally, recalling that the \((r,s)\) element of the DFT matrix (2.4) is \(\sum_r^{rN} e^{j2\pi rs/N}\), we introduce the "exponent matrix" \(E\),

\[
E_{r,s} = (r,s) \mod N
\]  

which we find very convenient in accounting for the contribution of the non-LC part.

In the present case

\[
s \rightarrow 0 \quad 2 \quad 1 \quad 3 \\
\sigma \rightarrow (0) \quad (2) \quad 0 \quad 1
\]

\[
E = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 2 & 2 \\
0 & 2 & 1 & 3 \\
0 & 2 & 3 & 1 \\
\end{bmatrix}
\]

Note that we have added here both kinds of row and column indices. One simple way of obtaining \(E\) is directly from its definition (5.6). In writing it down, we make sure that the non-bracketed \(0,\sigma\) indices would follow the sequence \(0,1,2,\ldots\). This will bring out the LC structure. The sequence of the other indices is immaterial.

We observe that (5.7) displays a second order LC submatrix as was to be expected. We handle the effect of this submatrix with the LCT tableau of Fig. 1, starting with the evaluation of the \(a_1\)'s

\[
\begin{bmatrix}
a_0 \\
a_1 \\
\end{bmatrix} = \begin{bmatrix}
\frac{1}{2} & \frac{\Omega}{2} \\
\frac{\Omega}{2} & \frac{1}{2} \\
\end{bmatrix} \begin{bmatrix}
W^3 \\
W \\
\end{bmatrix} = \begin{bmatrix}
i \\
-1 \\
\end{bmatrix}
\]

Hence (From Fig. 1)

\[
a_1 = 0; \quad a_2 = i\Omega
\]  

We conclude that only the \(\beta_2\) row contributes to \(F_1\) and we copy it into the \(\beta_3\) row of Fig. 10.
Fig. 10. Algorithm for DFT of order 4

\[
\begin{array}{cccc}
\beta_0 \ast \Omega &=& \mathbf{F}_0 & 1 & 1 \\
\beta_1 \ast \Omega &=& \mathbf{F}_1 & 1 & -1 \\
\beta_2 \ast \Omega &=& \mathbf{F}_2 & 1 & 1 \\
\beta_3 \ast \Omega &=& \mathbf{F}_3 & -1 & 1 \\
\end{array}
\]

\[N = 4 \quad (4M; 8A)\]
Note that the output vector $F$ appears scrambled in Fig. 10. In section IX we point out the advantage of certain tableau structures which allow storage economies in implementing the algorithm. The tableaus generated up to this point have both this desirable structure and an unscrambled output $F$. From here on, it seems impossible to realize both these desirable features simultaneously and we opt for the more important memory conserving structure. Hence, the scrambled output. The scrambling here is quite simple but becomes complex for $N=16$. To facilitate prescribing and handling of this, we have added the vector $\eta$ to the affected tableaus. The scrambled $F$ is identical with the unscrambled $\eta$. (see Fig. 10)

We return now to the realization of the remainder of the $E$ matrix

\begin{equation}
F - F' = \Omega(f_0 - f_2) = \frac{\Omega \beta_0}{\delta_2} \delta_2
\end{equation}

We have used here the fact that for $N = 4$, $\omega^2 = -1$. Equations like (5.10) can be read off directly from the $E$ matrix. Such equations will henceforth be presented without any comment.

\begin{equation}
F_2 = \Omega((f_0 + f_2) - (f_1 + f_3)) = \frac{\Omega \beta_0}{\delta_0} - \frac{\Omega \beta_1}{\delta_1} = \delta_0 - \delta_1
\end{equation}

\begin{equation}
F_0 = \Omega((f_0 + f_2) + (f_1 + f_3)) = \frac{\Omega \beta_0}{\delta_0} + \frac{\Omega \beta_1}{\delta_1} = \delta_0 + \delta_1
\end{equation}

This completes the derivation.

**DFT of Order 9 (Fig. 13)**

$N = 3^2$. Hence (2.7),

\begin{equation}
n = 6
\end{equation}

The primitive root of 9 is the primitive root of 3, namely, 2. This leads to

\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
$\rho$ & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
$\varphi = 2^\rho \mod 9$ & 1 & 2 & 4 & 8 & 7 & 5 \\
\hline
\end{tabular}

Hence, the following $E$ matrix
With \( n = 6 \), Fig. 4 will be applicable here. We consider the \( \alpha_i \)'s first. The \( E \) matrix (5.13) clearly displays the \( a_1 \) sequence (this is also in agreement with (4.3)). Hence,

\[
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
a_4 \\
a_5 \\
\end{bmatrix} = \frac{1}{6} \begin{bmatrix}
\omega \omega \omega \\
\omega^2 \omega^2 \\
\omega^3 \omega^3 \\
\omega^4 \omega^4 \\
\omega^5 \omega^5 \\
\end{bmatrix} = \frac{1}{6} \begin{bmatrix}
\omega^2 \omega^2 \\
\omega^4 \omega^4 \\
\omega^6 \omega^6 \\
\omega^8 \omega^8 \\
\omega^{10} \omega^{10} \\
\end{bmatrix}
\]

(5.14)

The computation of the \( \alpha_i \)'s and \( \omega_i \)'s is shown in Fig. 11. Note that, in the terminology of Figs. 4, 11

\[ \alpha_1 = \alpha_6 = 0 \]

(5.15)

This means that in the realization of \( \text{F}_4' \) which is effected by copying Figs. 4, 11 into Fig. 13, the terms associated with \( \alpha_1 \), \( \alpha_6 \), should be eliminated. \(^9\) In the actual

\(^9\) Fig. 13 seems to indicate that these terms have not been eliminated. This is misleading as the terms supporting this impression are those added later on. Winograd \[3\] appears to be unaware of (5.15). This leads to 2 extra multiplications in his algorithm for \( N = 9 \).
\[
\begin{array}{cccccc}
\frac{1}{6}w^4 & 1 & 1 & 1 & \text{c}_1 & = \frac{2}{3} (\sin 10^\circ - \cos 20^\circ + \cos 40^\circ) = 0 \\
\frac{1}{6}w^2 & 1 & -1 & \text{c}_2 & = \frac{2}{3} (\cos 20^\circ + \cos 40^\circ) \\
\frac{1}{6}w & 1 & -1 & \text{c}_3 & = \frac{2}{3} (\cos 10^\circ + \cos 20^\circ - \sin 40^\circ) \\
\frac{1}{6}w^4 & 1 & 1 & \text{c}_4 & = \frac{2}{3} (\sin 10^\circ + \cos 20^\circ + \cos 40^\circ) \\
\frac{1}{6}w^2 & 1 & 1 & \text{c}_5 & = \frac{2}{3} (\sin 10^\circ - \cos 20^\circ + \sin 40^\circ) \\
\frac{1}{6}w & 1 & 1 & \text{c}_6 & = \frac{2}{3} (\sin 10^\circ + \cos 20^\circ - \sin 40^\circ) \\
\end{array}
\]

Fig. 11. Computation of \( \omega_j \)'s for the DFT algorithm of order 9
copying from these figures, we also apply some permutations in addition to the obvious ones involving the input and output variables. These permutations are spelled out in (Fig. 12). Note that the permutation \( g_i \rightarrow e_j \) is shown in two sequential steps: \( g_i \rightarrow e_j; e_j \rightarrow g_j \). (Example: \( g_1 \rightarrow e_2 \rightarrow g_2 \))

<table>
<thead>
<tr>
<th>Figs. 4, 11 index (i)</th>
<th>0 1 2 3 4 5 6 7 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>( b_i \rightarrow f_j )</td>
<td>1 2 4 8 7 5</td>
</tr>
<tr>
<td>( c_i \rightarrow e_j )</td>
<td>1 4 2 7 5 8</td>
</tr>
<tr>
<td>( \omega_i \rightarrow \omega_j ); ( \beta_i \rightarrow \beta_j ); ( \delta_i \rightarrow \delta_j )</td>
<td>4 2 7 5 9 10</td>
</tr>
<tr>
<td>( s_i \rightarrow e_j )</td>
<td>9 4 2 7 5 10</td>
</tr>
<tr>
<td>( e_i \rightarrow g_j )</td>
<td>3 4 2 7 5 6</td>
</tr>
<tr>
<td>( f_i \rightarrow g_j )</td>
<td>5 7 8 4 2 1</td>
</tr>
</tbody>
</table>

Fig. 12. Index permutations in assembling Fig. 13.

We turn now to account for the rest of the matrix using (5.13) as our guide. In doing that we encounter the following constants

\[
\Omega^3 = \Omega(-\sin 30^\circ - i \cos 30^\circ) = -\Omega\left(\frac{1}{2} + i \frac{\sqrt{3}}{2}\right)
\]

\[
\Omega^6 = \Omega^3 = -\Omega\left(\frac{1}{2} - i \frac{\sqrt{3}}{2}\right)
\]

which we use in the following.

\[
F_r - F_r' = \Omega\left(f_0 - \frac{1}{2} + i \frac{\sqrt{3}}{2}\right) f_3 - \left(\frac{1}{2} - i \frac{\sqrt{3}}{2}\right) f_6
\]

\[
= \Omega\left(\frac{f_0}{\beta_0} - \frac{1}{2}(f_0 + f_3) + \frac{i\sqrt{3}}{2}(f_6 - f_3)\right) = \Omega\beta_0 - \frac{1}{2}\Omega \beta_3 - \left(\frac{i\sqrt{3}}{2}\right) \delta_0
\]

\[
= \frac{\delta_0 - \delta_3}{\beta_0 - \beta_3} - \frac{\delta_6}{\beta_0 - \beta_3}
\]

In the case of \( g_3 \rightarrow e_2 \rightarrow g_2 \), we introduce two canceling sign changes.
We withhold implementation and proceed to the next group

\[ r = 2; 8; 5 \]

Compared to the previous case we have here interchange of \( W^3 \) with \( W^6 (=\overline{W}^3) \). Hence,

\[ F_r = F'_r + e_3 - e_6 \]

At this point we implement both groups as shown in Fig. 12.

Next we implement \( F_3, F_6 \). From (5.13),

\[
F_3 = \Omega \left\{ (f_0 + f_3 + f_6) - \frac{1}{2} + \frac{1}{2} \right\} (f_1 + f_4 + f_7) - \frac{1}{2} - \frac{1}{2} \right\} (f_2 + f_8 + f_5) \right\} + \\
\beta_0 \beta_3 \left\{ f_0 + (f_6 + f_3) - \frac{1}{2} f_1 + \frac{1}{2} f_2 + \frac{1}{2} f_3 \right\} + \\
\left\{ f_0 f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 \right\} + \\
= \frac{1}{2} \left\{ f_0 f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 \right\} + \\
= e_0 - \delta_1 - \delta_8 = e_0 - e_1 - e_8 = g_1 - g_8 \\
e_1 e_8 g_1 g_8
\]

We turn now to \( F_6 \). The only difference here is the interchange of \( W^3 \) with \( W^6 (=\overline{W}^3) \). Hence

\[ F_6 = g_1 + g_8 \]

Finally,

\[
F_0 = \Omega (f_0 + f_3 + f_6) + \Omega B_1 = \Omega \left\{ (\delta_0 + \delta_3) + (\delta_8) \right\} + 2 \delta_1 = e_0 + 2 e_1 = g_0 \\
\delta_0 \delta_3 \delta_1
\]

This completes the derivation.

With the two additional tableaus developed in this section, the maximal realizable \( N \) has been pushed to 1260 (4, 5, 7, 9). We push it still higher (5040) with the tableaus of the next section.
\[ \omega_2 = -\frac{1}{3}(2 \sin 10^\circ + \cos 20^\circ - \cos 40^\circ) \]
\[ \omega_7 = -\frac{1}{3}(2 \cos 10^\circ + \sin 20^\circ + \sin 40^\circ) \]
\[ \omega_4 = \frac{1}{3}(\sin 10^\circ + 2 \cos 20^\circ + \cos 40^\circ) \]
\[ \omega_9 = \frac{1}{3}(-\sin 10^\circ + \cos 20^\circ + 2 \cos 40^\circ) \]
\[ \omega_5 = -\frac{1}{3}(\cos 10^\circ + 2 \sin 20^\circ - \sin 40^\circ) \]
\[ \omega_{10} = -\frac{1}{3}(\cos 10^\circ + \sin 20^\circ + 2 \sin 40^\circ) \]

Fig. 13. Algorithm for DFT of order 9
VI. The Basic DFT Algorithms for \( N = 8, 16 \)

In developing the tableaus for \( N = 8, 16 \), we face a complication due to the fact that
\[
N = 2^k \quad (k > 2)
\] (6.1)

has no primitive roots. In other words, unlike the \( N = 4 \) case, there is no integer whose powers (mod \( N \)) would generate all of the odd numbers in the interval \((1, N)\).

We can, however, generate half of these with powers (mod \( N \)) of the number 3 \( [9] \).

We proceed now to modify the relabeling scheme to handle this case. First we modify (2.9) to read as follows:
\[
\begin{align*}
\rho &= 3^p \mod N \\
\sigma &= 3^s \mod N
\end{align*}
\] (6.2)

\[
\begin{align*}
\rho &= -3^p \mod N \\
\sigma &= -3^s \mod N
\end{align*}
\] (6.3)

Note that we have introduced here (6.3) a new type of index so that we now have three kinds: \((i), \bar{i}, i\). We illustrate this with the case \( N = 8 \)

\[
\begin{array}{c|cccc}
\rho & \bar{0} & \bar{1} & 0 & 1 \\
\hline
r = 3^p \mod 8 & - & - & 1 & 3 \\
r = -3^p \mod 8 & 7 & 5 & - & - \\
\end{array}
\] (6.4)

It should be stressed that the bar or parentheses have no effect on the numerical value of the index. Their only effect is on the choice of the functional relationships connecting the new entities \( b_g, b_\rho \) to the original variables \( f_s, F_r \).

Consider for example the expression \( g_{1\bar{1}} \). If we evaluate it for \( i = 1 \), its value is \( g_{1\bar{1}} = g_{13} \) (see (6.4)). If, on the other hand, we want its value for \( i = \bar{1} \) we get \( g_{1\bar{1}} = g_{13} \).

Eqns. (2.10), (2.11) now read
\[
\begin{align*}
F_r &= B_\rho; \quad f_s = b_\rho \quad (r, s \text{ odd}) \\
F_r &= B(r); \quad f_s = b(s) \quad (r, s \text{ even})
\end{align*}
\] (6.5)
so that the overall $r,p$ functional relationship can be summarized as follows

$$
\begin{array}{c|cccccccc}
   & 0 & 1 & 0 & 1 & 0 & 2 & 4 & 6 \\
---&---&---&---&---&---&---&---&---
   r & 7 & 5 & 1 & 3 & 0 & 2 & 4 & 6 \\
\end{array}
$$

(6.6)

with an identical table relating $s$ to $\sigma$.

Eqns. (6.2)-(6.5) are now applied to eliminate $F_u,F_v$ from (2.4). (See analogous treatment in section II.)

$$
B_{(2t)} = F_{2t} = \sum_{m=0,2,4,6} \left[ \frac{N-1}{2} \right] W^{2mt} b_m + \sum_{\sigma=0}^{\frac{N-1}{2}} W^{-2t3^{\sigma}} b_\sigma + \sum_{\sigma=0}^{\frac{N-1}{2}} W^{2t3^{\sigma}} b_\sigma \\
(t = 0,1,\ldots,\frac{N-1}{2}-1)
$$

(6.7)

$$
\hat{B}_p = \hat{B}_p + \hat{B}'_p; \quad \hat{F}_r = \hat{F}_p + \hat{F}'_r \\
(r \text{ odd})
$$

(6.8)

$$
\begin{array}{c}
   \hat{B}_p = \\
   \left\{ \begin{array}{l}
   \sum_{m=0,2,4,6} \left[ \frac{N-1}{2} \right] W^{-m3^\sigma} b_m \\
   \sum_{m=0,2,4,6} \left[ \frac{N-1}{2} \right] W^{m3^\sigma} b_m
   \end{array} \right. \\
   (\sigma = 0,1,\ldots,\frac{N-1}{4}-1)
\end{array}
$$

(6.9)

$$
\begin{array}{c}
   \hat{F}_r = \\
   \left\{ \begin{array}{l}
   \sum_{\sigma=0}^{\frac{N-1}{4}} W^{(3^\sigma+\sigma)} b_\sigma + \sum_{\sigma=0}^{\frac{N-1}{4}} W^{-3^\sigma}\sigma b_\sigma \\
   \sum_{\sigma=0}^{\frac{N-1}{4}} W^{-(3^\sigma+\sigma)} b_\sigma + \sum_{\sigma=0}^{\frac{N-1}{4}} W^{-(3^\sigma+\sigma)} b_\sigma
   \end{array} \right. \\
   (\sigma = 0,1,\ldots,\frac{N-1}{4}-1)
\end{array}
$$

(6.10)

It is obvious that all four matrix products appearing in (6.10) are Hankel transformations $(p+\sigma)$ of order $\frac{N}{4}$. We prove now that they are also LCT's. To do this, we must show that (see (2.2), (2.16))

$$
3^{\left(\frac{N-1}{2}\right)} = 3^{N-1} \mod N \\
(N = 2^k; k > 2)
$$

(6.11)
or equivalently,

\[ \frac{N}{3^4} = 1 \mod N \quad (N = 2^k; \ k > 2) \]  

(6.12)

We prove (6.12) by induction on \( k \). Assume it to be true for \( N = n \), then \( \frac{n}{3^4} = mn + 1 \) (m integer). Squaring yields \( 3^4 = (m^2 \frac{n}{2}) (2n) + m(2n) + 1 \) so that \( 3^4 = 1 \mod (2n) \) and (6.12) is true for \( N = 2n \). Finally, (6.12) is obviously true for \( k = 3 \).

Thus, (6.10) involves four LC matrix products. Furthermore, these four LC matrices comprise a second order compound matrix which is an LC itself. All these features stand out quite clearly in the two cases to be developed now.

**DFT of Order 8 (Fig. 14)**

Applying the permutation of (6.6), we get the following E matrix

\[
\sigma \rightarrow (0)(4)(2)(6) \quad \overline{0} \quad \overline{1} \quad 0 \quad 1
\]

\[
E = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 4 & 4 & 4 \\
0 & 0 & 4 & 4 & 6 & 2 & 2 & 6 \\
0 & 0 & 4 & 4 & 2 & 6 & 6 & 2 \\
0 & 4 & 6 & 2 & 1 & 3 & 7 & 5 \\
0 & 4 & 2 & 6 & 3 & 1 & 5 & 7 \\
0 & 4 & 2 & 6 & 7 & 5 & 1 & 3 \\
0 & 4 & 6 & 2 & 5 & 7 & 3 & 1
\end{bmatrix}
\]

(6.13)

We turn now to an explicit representation of the submatrix corresponding to the lower right quarter of E. Denoting

\[ w_k = e^{2\pi ik} \]

we get

\[
\hat{\mathbf{f}}_1 = \begin{bmatrix}
\mathbf{F}'_7 \\
\mathbf{F}'_5 \\
\mathbf{F}'_3
\end{bmatrix} = \begin{bmatrix}
\hat{\mathbf{a}}_1 & \hat{\mathbf{a}}_0
\end{bmatrix} \begin{bmatrix}
\mathbf{w}_1 & \mathbf{w}_3 \\
\mathbf{w}_7 & \mathbf{w}_5
\end{bmatrix} \begin{bmatrix}
\mathbf{f}_7 \\
\mathbf{f}_5 \\
\mathbf{f}_3
\end{bmatrix}
\]

\[
\hat{\mathbf{b}}_0 = \begin{bmatrix}
\mathbf{b}_0
\end{bmatrix}
\]

(6.15)
We see here four second-order LC submatrices, two each, of types \( \hat{a}_0, \hat{a}_1 \). Using the terminology introduced here, we write down the equivalent second-order compound matrix

\[
\begin{bmatrix}
\hat{\varepsilon}_1 \\
\hat{\varepsilon}_0
\end{bmatrix} = \begin{bmatrix}
\hat{a}_1 & \hat{a}_0 \\
\hat{a}_0 & \hat{a}_1
\end{bmatrix}
\begin{bmatrix}
\hat{\varepsilon}_0 \\
\hat{\varepsilon}_1
\end{bmatrix}
\]

(6.16)

Note that (6.16) is also an LT. We propose now to evaluate (6.16) through a direct application of the tableau of Fig. 1. Some elaboration is in order here. The tableaus in this paper have been derived with the implied assumption that the variables appearing in them are all scalars. This is not really necessary. Reviewing the derivations, one concludes that the tableaus are also valid under the following generalized interpretation:

1. The row (column) elements \((f_1, e_1, \beta_1, \delta_1, e_1, \ldots)\) are column submatrices
2. \(a_1, a_i, \Omega\) are square matrices
3. \(\omega_i\) is still a scalar constant
4. The tableau entries

\[
\beta_i a_1 = \delta_i; \quad \beta_i \omega_i \Omega = \delta_i
\]

should be interpreted as the following matrix products

\[
\alpha_i \beta_i = \delta_i; \quad \omega_i \Omega \beta_i = \delta_i
\]

We introduce this generalization here with the immediate goal of evaluating (6.16). However, its importance transcends this immediate application. It is this generalization which elevates the tableaus from the rather theoretical realm of fast algorithms for very low order DFT's into the very practical realm of high-order, high-speed DFT algorithms. The specific way in which this is done is presented in detail in section VII.

We return now to the evaluation of (6.16). Applying Fig. 1 we get

\[
\begin{bmatrix}
\hat{\varepsilon}_0 & \hat{\varepsilon}_1 \\
1 & 1 \\
1 & -1
\end{bmatrix}
\begin{bmatrix}
\hat{\varepsilon}_1 \\
\hat{\varepsilon}_0
\end{bmatrix} = \begin{bmatrix}
\hat{\varepsilon}_0 & \hat{\varepsilon}_1 \\
1 & 1 \\
1 & -1
\end{bmatrix}
\]

(6.17)

---

10. This "modified interpretation" can be avoided by using row matrices instead of column matrices. Note in this context that the \(\Omega\) matrix we will be concerned with is symmetric.
We turn now to the realization of the remaining parts of (6.13)

\[
F_r = F'_r + \Omega(\tilde{c}_0 - \tilde{c}_4) + i(\tilde{c}_6 - \tilde{c}_2) = F'_r + \Omega(c_4 + i c_6) = F'_r + \Omega \beta_4 = F'_r + \beta_4
\]

\[
F_r = F'_r + \Omega(\tilde{c}_6 - \tilde{c}_4) = F'_r + \beta_6
\]
Fig. 14. Algorithm for DFT of order 8

\[
\begin{array}{cccccccc}
  f_0 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 \\
  1 & 1 & 1 & 1 & 1 & -1 & 1 & 1 \\
  1 & 1 & c_1 & 1 & 1 & 1 & -1 & 1 \\
  1 & 1 & c_2 & 1 & 1 & 1 & -1 & 1 \\
  1 & 1 & c_4 & 1 & 1 & 1 & -1 & 1 \\
  1 & 1 & c_5 & 1 & 1 & 1 & -1 & 1 \\
  1 & 1 & c_6 & 1 & 1 & 1 & -1 & 1 \\
  1 & 1 & c_7 & 1 & 1 & 1 & -1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
  g_0 & g_1 & g_2 & g_3 & g_4 & g_5 & g_6 & g_7 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
  \beta_0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
  \Omega & (1/2)\Omega & \Omega & (i/2)\Omega \\
  1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
  s_0 & s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & s_7 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
  1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
  N=8 (8M, 26A) \\
\end{array}
\]
This concludes the derivation.

**DFT of Order 16 (Fig. 17)**

The permutation is controlled by the following table

<table>
<thead>
<tr>
<th>(\rho)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3^\rho \mod 16</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>-3^\rho \mod 16</td>
<td>15</td>
<td>13</td>
<td>7</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Applying this we get the following E matrix
In view of the complexity of the present case, we derive the tableau in two stages. \( F_1' \), the contribution of the LC submatrices, is considered separately in the intermediate tableau of Fig. 15 which is then copied into the final tableau of Fig. 17. Following the previous case, we write down explicitly the LC part
Fig. 15. Intermediate tableau for the computation of $F'_i$ for the DFT of order 16
We handle (6.22) via the tableaus (6.17), (6.18). It is obvious from (6.18) that $\hat{\alpha}_1, \hat{\alpha}_2$ will be LC matrices. Hence, it is sufficient to compute their first rows only. We express these in terms of the basic angle

$$\theta = \frac{360^0}{16} = 22.5^0$$

$$\begin{array}{c}
\text{First row of } \hat{\alpha}_1 = \Omega \begin{bmatrix} \cos \theta & \sin \theta & -\cos \theta & -\sin \theta \end{bmatrix}
\end{array}$$

$$\begin{array}{c}
\text{First row of } \hat{\alpha}_2 = i\Omega \begin{bmatrix} \sin \theta & \cos \theta & -\sin \theta & -\cos \theta \end{bmatrix}
\end{array}$$

Turning to (6.17), we get

$$\begin{bmatrix}
\hat{\beta}_1 = \\
\hat{\beta}_2 = 
\end{bmatrix}$$

$$\begin{bmatrix}
\hat{\alpha}_1 \\
\hat{\alpha}_0
\end{bmatrix} = \begin{bmatrix}
\hat{\alpha}_1 \\
\hat{\alpha}_0
\end{bmatrix}$$
The next step in (6.17) calls for the evaluation of two LCT's of order 4, namely,

\[ \delta_1 = \hat{\delta}_1 \hat{\beta}_1 \quad (i = 1, 2) \quad (6.27) \]

We adopt now the following notation for the components of \( \delta_1 \)

\[ \hat{\delta}_1 = \begin{bmatrix} \hat{\delta}_{10} \\ \hat{\delta}_{11} \\ \hat{\delta}_{12} \\ \hat{\delta}_{13} \end{bmatrix} \quad (6.28) \]

These will be determined now by applying the tableau of Fig. 2.

**Implementation of** \( \hat{\delta}_1 = \hat{\delta}_1 \hat{\beta}_1 \)

\[
\begin{array}{cccc}
-\frac{\Omega}{4} \sin \theta & 1 & 1 \\
-\frac{\Omega}{4} \cos \theta & 1 & 1 \\
\frac{\Omega}{4} \sin \theta & 1 & -1 \\
\frac{\Omega}{4} \cos \theta & 1 & -1 \\
0 & 0 & -\frac{\Omega}{2} \cos \theta & -\frac{\Omega}{2} \sin \theta \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 1 & 0 & 0 \\
1 & -1 & -\frac{\Omega}{2} \cos \theta & \frac{\Omega}{2} \sin \theta \\
1 & 1 & -\frac{\Omega}{2} \cos \theta & \frac{\Omega}{2} \sin \theta \\
1 & 1 & -\frac{\Omega}{2} \cos \theta & \frac{\Omega}{2} \sin \theta \\
\end{array}
\]

\[
\left\{ \begin{array}{l}
\mathbf{\epsilon}_5 \Omega; \omega_5 = 2\mathbf{\epsilon}_1 \Omega = -(\cos \theta + \sin \theta) \\
\mathbf{\epsilon}_7 \Omega; \omega_7 = 2(\mathbf{\epsilon}_7 - \mathbf{\epsilon}_5) = (\cos \theta - \sin \theta) \\
\end{array} \right. \quad (6.29)
\]

Note that the vanishing of the first two multipliers means that only a portion of Fig. 2 is required, namely, the portion involving the rows of \( \beta_3, \beta_4, \beta_5 \). This is copied into rows of \( \beta_5, \beta_7, \beta_16 \), respectively, in Fig. 15 (Hence the adopted \( \omega \) indices). Note further that Fig. 15 shows two alternative paths leading from
\( \delta_1 \) to \( F'_r \). The upper path should be ignored for now. The preceding discussion has derived that portion of the tableau leading to \( \delta_1 \). We turn now to \( \delta_2 \).

**Implementation of \( \delta_2 = \delta_2 \delta_2 \)**

\[
\begin{array}{c|ccc}
 & -i\Omega \cos \theta & 1 & 1 \\
\hline
-\frac{i\Omega}{4} \sin \theta & 1 & 1 \\
\frac{i\Omega}{4} \cos \theta & 1 & -1 \\
\frac{i\Omega}{4} \sin \theta & 1 & -1 \\
\hline
0 & 0 & -\frac{i\Omega}{2} \sin \theta & -\frac{i\Omega}{2} \cos \theta
\end{array}
\]

\[
\begin{array}{c|ccc}
1 & 1 & 0 \\
1 & -1 & 0 \\
\hline
\text{1} & \text{1} & -\frac{i\Omega}{2} \theta & \frac{1}{\Omega} \sin \theta
\end{array}
\]

The situation here is very similar to the \( \delta_1 \) case. The rows of \( \beta_3, \beta_4, \beta_5 \) of Fig. 2 are now copied into the rows of \( \beta_{13}, \beta_{15}, \beta_{17} \), respectively, of Fig. 15.

With \( \delta_1, \delta_2 \) now available, we apply (6.17) to obtain \( \hat{c}_{1} \hat{c}_{0} \) as shown in the lower part of Fig. 15. The transition from \( g_1 \) to \( F'_r \) is shown there requiring 8 additions. The upper part realizes the same transformation with only 4 additions and is the version copied into Fig. 17. The identity of the two paths can be easily verified by inspection. For example, the upper part prescribes \( F'_1 = g_7 + g_{15} \) but so does the lower part. Verifying such agreements for all 8 outputs, establishes the identity.

We implement now the remaining parts of (6.20). To bring out clearly the various symmetries we are exploiting here, we show an explicit form of the remaining part of the permuted DFT matrix in Fig. 16. It is shown here for convenience as the sum of two matrices and expressed in terms of the constant

\[
\gamma = \frac{1}{\sqrt{2}} 
\]
Fig. 16. E matrix for the computation of $F_i - F'_i$ for the DFT of order 16.
We define now

\[ c_{14} = f_6 f_{14}; \quad c_{12} = f_4 f_{12}; \quad c_{10} = f_2 f_{10}; \quad c_8 = f_0 f_8 \]

and proceed with the evaluation as follows:

\[ r = 1;9 \]

\[ F_r - F' = \Omega \left( \frac{c_8 + i c_{12}}{\beta_{12}} \right) - \gamma \left[ \frac{(c_{14} - c_{10}) + i(c_{14} + c_{10})}{e_{10}} \right] \]

\[ = \frac{\delta_{12} - i \gamma \Omega}{\beta_{12}} \frac{\delta_{10}}{e_{10}} = \frac{g_{12} - \delta_{10}}{g_{10}} = g_{12} g_{10} = h_{12} \]

\[ \therefore F_r = F' + h_{12} \]

\[ r = 5;13 \]

\[ F_r = F' + (g_{12} + g_{10}) = F' + h_{10} \]

\[ r = 3;11 \]

\[ F_r - F' = \Omega \left( \frac{c_8 + i c_{12}}{\beta_{12}} \right) + \gamma \left[ \frac{(c_{14} - c_{10}) - i(c_{14} + c_{10})}{e_{14}} \right] \]

\[ = \frac{\delta_{12} - i \gamma \Omega}{\beta_{12}} \frac{\delta_{14}}{e_{14}} = \frac{g_{12} - \delta_{14}}{g_{14}} = g_{12} g_{14} = h_{14} \]

\[ \therefore F_r = F' + h_{14} \]

\[ r = 7;15 \]

\[ F_r = F' + (g_{8} + g_{14}) = F' + h_{14} \]

We define now

\[ c_0 = f_0 f_8; \quad c_2 = f_2 f_{10}; \quad c_4 = f_4 f_{12}; \quad c_6 = f_6 f_{14} \]
Fig. 17. Algorithm for DFT of order 16
\[ g_4 = \delta_4 = \Omega \beta_4 = \Omega (c_0 - c_4) \]

and turn to the next group of rows \( (F_2, F_6, F_{10}, F_{14}) \)

\[
F_2 = g_4 + \Omega \left\{ -i \left( \frac{c_2 - c_6}{\beta_5} \right) + \gamma \left[ \frac{(c_1 + c_3)}{e_1} - \frac{(c_5 + c_3)}{e_3} + \frac{i(c_{15} - c_9)}{e_9} + \frac{i(c_{13} - c_{11})}{e_{11}} \right] \right\}
\]

\[
= g_4 - \frac{i\Omega_6}{\delta_6} + \gamma \Omega (e_1 - e_3) + \frac{1}{\delta_3} + \gamma \Omega (e_9 + e_{11}) = g_4 - \frac{\delta_6}{\beta_3} + \gamma \Omega_3 + \frac{1}{\delta_3} + \gamma \Omega \beta_9 = g_4 - g_6 + \delta_3 + \delta_9
\]

\[
= (g_4 - g_6) + (g_9 + g_3) = h_4 + h_3
\]

Noting which columns in Fig. 16 are associated with each of the four \( g_1 \)'s comprising \( F_2 \), we observe that \( F_6, F_{10}, F_{14} \) involve the same \( g_1 \)'s but with different sign combinations. Specifically,

\[
F_{10} = (g_4 - g_6) - (g_9 + g_3) = h_4 - h_3
\]

\[
F_6 = h_6 + h_9
\]

\[
F_{14} = (g_4 + g_6) - (g_9 - g_3) = h_6 - h_9
\]

Finally we consider the upper four rows

\[
F_0 = \Omega \left\{ \left( \frac{c_0 + c_4}{\beta_0} \right) + \left( \frac{c_2 + c_6}{\beta_2} \right) + \left( \frac{c_1 + c_3}{\beta_1} \right) + \left( \frac{c_5 + c_3}{\beta_3} \right) \right\} = \Omega \left\{ \left( \frac{e_0 + e_2}{\beta_0} \right) + \left( \frac{e_1 + e_3}{\beta_1} \right) \right\}
\]

\[
= \Omega \beta_0 + \frac{\Omega_1}{\delta_0} = \delta_0 + \delta_1 = g_0 + g_1 = h_0 + h_1
\]

\[
\therefore F_8 = h_0 - h_1
\]
\[ F_4 = \Omega \left( \frac{e_0 - e_2}{\beta_2} + i \left[ \frac{(c_{15} - c_9)(c_{13} - c_{11})}{e_9 e_{11}} \right] \right) = \frac{\Omega \beta_2}{\delta_2} + i \frac{\Omega (e_9 - e_{11})}{\beta_{11}} = \frac{\delta_2}{g_2} + i \frac{\Omega \beta_{11}}{\delta_{11}} \]

\[ = \frac{g_2 + \delta_{11}}{h_2 + g_{11}} = h_2 + h_{11} \]

\[ \therefore F_{12} = h_2 - h_{11} \]

This completes the derivation. Note that the size of this tableau imposes certain notational peculiarities. In particular, \( \omega(i) = \omega'_i \), \( \beta(i) = \beta'_i \), etc.
VII. The DFT Algorithm for

\[ N = \prod_{k=1}^{K} N_k \]

At this point, we finally are in possession of DFT tableaus for all orders listed in (1.3). Our immediate goal is their integration into a DFT algorithm of order

\[ N = \prod_{k=1}^{K} N_k \quad (N_k's \text{ relatively prime}) \quad (7.1) \]

In the subsequent derivations, we will also need the following partial products of these factors

\[ \psi_i = \prod_{k=i+1}^{K} N_k \quad (0 < i < K) \]
\[ \psi_K = 1 \]
\[ n_i = \prod_{k=1}^{K} N_k = \frac{N}{N_i} \quad (7.3) \]

Let us introduce now a set of matrices which figure prominently in the development of the algorithm. Consider a matrix whose order \( N \) satisfies (7.1). We denote by \( \Omega_i \) its upper-left submatrix of order \( \psi_i \). Note that (7.2) implies the existence of a sequence of these submatrices

\[ \Omega_0, \Omega_1, \ldots, \Omega_K \]

The matrix order decreases to the right. \( \Omega_0 \) on the left is of order \( N \) and is thus identical with the overall matrix. \( \Omega_K \) on the right is of order 1 and is thus the upper-left element of the overall matrix.

The second item we introduce here is the graphical representation of the tableaus shown in Figure 18. We describe this in terms of the generalized, compound-matrix, interpretation of the tableaus (see discussion following (6.16)). Let the tableau variables \( f_i, \beta_i, \delta_i, F_i, \ldots \) etc. represent \( m \)-dimensional vectors. Then, the tableau of order \( N \) is applicable to any matrix transformation of order \( mN \) in which the transforming matrix, \( Y \), when regarded as a compound matrix of order \( N \), is the \( N \)-th order DFT matrix (2.4). In this

---

The basic idea underlying the developments in this section is commonly attributed to I.J. Good [11]. Here, we find it more convenient to avoid Good's explicit use of the Kronecker matrix product.
case, the $\Omega$ parameter of the tableau is the $m$-th order upper-left submatrix of $Y$.

Examination of the tableaus reveals that they all consist of three parts corresponding to three distinct phases of the algorithms they describe. In phase 1, the $mN$-dimensional input vector $f$ is operated upon to yield the $m$-dimensional $\beta_i$ vectors. In phase 2, a scalar multiple of the $\Omega$ submatrix transforms $\beta_i$ into $\xi_i$ according to (4.13) $(\xi_i = (\omega_1 \Omega) \beta_i; \ i = 0, 1, \ldots, M-1)$. $M$ is the number of multiplications appearing in the tableau designation. Obviously, this is also the number of $\beta_i$ vectors generated in phase 1. Finally, in phase 3, the $m$-dimensional $\xi_i$ vectors are operated upon to yield the $mN$-dimensional output vector $F$. The three phases are represented schematically in Fig. 18. The conventions adopted here are as follows: All lines represent vectors. We may attach an integer to a line to indicate the dimensionality of the vector it represents (see Fig. 21). Phase 1 is represented by a circle, phase 3 by a square. In either case, the symbol inside designates the matrix transforming the "circle input" to the "square output". Note that the only arithmetic operations involved inside either the circle or the square, are additions and subtractions.

With these preliminaries out of the way, we are ready now to consider a set of scrambling guidelines that would allow a simple implementation of the DFT of order $N$ satisfying (7.1). Denoting the scrambled matrix by $Y$, we propose the following set of sufficient conditions:

1. $Y(=\Omega_0)$ should be a compound DFT matrix of order $N_1$. This will allow the implementation of phase 1 of the $N_1$ tableau. Phase 2 calls for the determination of $(\omega_1 \Omega_1) \beta_1$. Hence,

2. $\Omega_1$ should be a compound DFT matrix of order $N_2$. We apply phase 1 of the $N_2$ tableau and are led, in phase 2 to a transformation involving $\Omega_2$. Hence,

3. $\Omega_2$ should be a compound DFT matrix of order $N_3$ and so on down to $\Omega_{K-1}$ which should be a DFT matrix of order $N_K$.

For most $i$ values, $\xi_i = \delta_i$ of the tableaus but not for all of them. For example, the $N = 5$ tableau implies $\xi_0 = F_0$, $\delta_1 = \xi_1 + \xi_0$. 

\[\text{For most } i \text{ values, } \xi_i = \delta_i \text{ of the tableaus but not for all of them. For example, the } N = 5 \text{ tableau implies } \xi_0 = F_0, \delta_1 = \xi_1 + \xi_0.\]
Fig. 18. Schematic representation of the basic DFT tableaus
All this can be summarized as follows: A convenient relabeling scheme would be one which satisfies the following set of $\kappa$ constraints:

\[
\text{Submatrix } \Omega_{k-1} \text{ of } Y \text{ should be a compound DFT matrix of order } N_k. \text{ This should hold for all } 1 \leq k \leq \kappa
\] (7.4)

While an algorithm following the above outline would be quite convenient to implement, it is not at all clear that a scrambling scheme that would simultaneously satisfy all $\kappa$ constraints of (7.4) does, in fact, exist.

We proceed now to develop a relabeling scheme which comes very close to (7.4), adding only a minor complication to the above algorithm outline. We start with the standard DFT matrix (2.4) with $\Omega = 1$.

\[
\hat{p}_u = \sum_{v=0}^{N-1} w^{uv} \hat{p}_v \quad (u=0,1,\ldots,N-1). \tag{7.5}
\]

Let

\[
Q_m = \hat{p}_u; \quad q_n = \hat{p}_v \tag{7.6}
\]

where

\[
u = \lambda(m); \quad v = \lambda(n) \quad (m,n = 0,1,\ldots,N-1) \tag{7.7}
\]

and the function $\lambda$ is yet to be specified. This transforms (7.5) into

\[
Q_m = \sum_{n=0}^{N-1} w^{\lambda(m)} \lambda(n) q_n = \sum_{n=0}^{N-1} Y_{mn} q_n \quad (m=0,1,\ldots,N-1) \tag{7.8}
\]

The function $\lambda$ will be defined in terms of a modular representation \[7\] of $u,v$. In other words, the relabeling depends on the entities

\[
u_k = u \mod N_k \quad (k=1,2,\ldots,\kappa) \tag{7.9}
\]

According to the Chinese Remainder Theorem \[7\], these remainders uniquely determine any $0 \leq (u,v) < N$. Hence, we may adopt the following representation for $u,v$

---

12 We use here $\hat{F}_u, \hat{f}_v$ to distinguish these entities from the tableau variables $F_u, f_v$. 
The function $\lambda$ is now defined in terms of the following combination of (7.7), (7.10)

$$v = (v_1, v_2, \ldots, v_K) = \lambda(n)$$  \hspace{1cm} (7.11)

$\lambda(n)$ should be such that as $n$ follows the sequence $0, 1, \ldots, N-1$, each $v_k$ should follow a periodic repetition of the sequence $0, 1, \ldots, N_k-1$, starting with $v_k = 0$. $v_k$ should be stepped with every $v_k$-increment of $n$ (see (7.2)). This means that $v_1$ varies very slowly, $v_2$ varies faster and so on, up to $v_K$ which varies in step with $n$.

To illustrate this scrambling, consider the following example

$$K = 3; N_1 = 8; N_2 = 3; N_3 = 5; \therefore N = 120$$  \hspace{1cm} (7.12)

for which part of the index sequence would look as follows:

<table>
<thead>
<tr>
<th>$n$</th>
<th>$v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>48 = (0,0,3)</td>
</tr>
<tr>
<td>4</td>
<td>24 = (0,0,4)</td>
</tr>
<tr>
<td>5</td>
<td>40 = (0,1,0)</td>
</tr>
<tr>
<td>6</td>
<td>16 = (0,1,1)</td>
</tr>
<tr>
<td>62</td>
<td>12 = (4,0,2)</td>
</tr>
<tr>
<td>63</td>
<td>108 = (4,0,3)</td>
</tr>
<tr>
<td>64</td>
<td>84 = (4,0,4)</td>
</tr>
<tr>
<td>65</td>
<td>100 = (4,1,0)</td>
</tr>
<tr>
<td>66</td>
<td>76 = (4,1,1)</td>
</tr>
</tbody>
</table>

One could use modular arithmetic subroutines to determine this sequence on a computer. Alternatively, it could be determined in a scheme using neither computers nor computations. All that is called for is some tedious writing down of
repetitive sequences. We illustrate this method for our example (7.12) in Tables 2, 3. First we write down the sequence 0,1, ..., N-1 as shown in Table 2 under the heading v. Then we write down next to this column, under the heading v_k, a periodic repetition of the sequence 0,1, ..., N_k-1 and repeat this for all 1 ≤ k ≤ k. This yields representation (7.10) for all v's. The particular arrangement in Table 2 saves some writing and is convenient for the next step which is just a reordering of Table 2 in the desired sequence.

To simplify the process, we let Table 2 determine the order in which Table 3 is being filled in. For example, the first v values copied from Table 2 into Table 3 are 0, 40, 80, 8, 48, 88, 16, 56, etc. Table 3 then prescribes the index sequence of the scrambled input vector for (7.12), namely,

\[ \hat{f}_0, \hat{f}_{46}, \hat{f}_{72}, \ldots, \hat{f}_{104}, \hat{f}_{105}, \ldots, \hat{f}_{89}, \hat{f}_{90}, \ldots, \hat{f}_{119} \]

To facilitate the analysis of the adopted scrambling, we introduce now E, the exponent matrix of Y. Recall that (7.8) implies

\[ Y_{mn} = W^{λ(m)λ(n)} \]

Hence we define the exponent matrix E as follows:

\[ E_{mn} = λ(m)λ(n) = uv \]

Similarly, paralleling the Ω_k submatrices of Y we define \( \hat{E}_k \) as the upper-left \( v_k \)-order submatrix of E.

Let us examine now the structure of \( \hat{E}_{k-1} \), starting with the distribution of \( u_k, v_k \). The scrambling prescribes that, starting with the value zero at the upper-left corner, \( u_k \) should be increased by one every \( v_k \) rows. Similarly, \( v_k \) should increase by one every \( v_k \) columns. This, then defines a subdivision of \( \hat{E}_{k-1} \) into submatrices of order \( v_k \). \( \hat{E}_{k-1} \) can now be regarded as a compound matrix of order \( N_k \left( \frac{v_{k-1}}{v_k} \right) \), whose (r,s) "element" is characterized by

\[ u_k = r; \quad v_k = s \]

The (0,0) element of this compound matrix is obviously identical with \( \hat{E}_k \). Let us pick now an element in an arbitrary position in \( \hat{E}_k \). Its u,v will have the form

\[ u = (0, \ldots, 0, u_{k+1}, u_{k+2}, \ldots, u_k) \]
\[ v = (0, \ldots, 0, v_{k+1}, v_{k+2}, \ldots, v_k) \]
Table 2

Modular Representation of $v$ in Example (7.12)

$N_1 = 8; N_2 = 3; N_3 = 5$

<table>
<thead>
<tr>
<th>$v^1$</th>
<th>$v^2$</th>
<th>$v^3$</th>
<th>$v^1$</th>
<th>$v^2$</th>
<th>$v^3$</th>
<th>$v^1$</th>
<th>$v^2$</th>
<th>$v^3$</th>
<th>$v^1$</th>
<th>$v^2$</th>
<th>$v^3$</th>
<th>$v^1$</th>
<th>$v^2$</th>
<th>$v^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>2</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>4</td>
<td>2</td>
<td>11</td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>0</td>
<td>2</td>
<td>13</td>
<td>0</td>
<td>2</td>
<td>12</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>1</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>14</td>
<td>2</td>
<td>4</td>
<td>15</td>
<td>2</td>
<td>4</td>
<td>14</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>40</td>
<td>1</td>
<td>0</td>
<td>48</td>
<td>0</td>
<td>3</td>
<td>56</td>
<td>1</td>
<td>1</td>
<td>64</td>
<td>1</td>
<td>4</td>
<td>72</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>41</td>
<td>2</td>
<td>1</td>
<td>49</td>
<td>1</td>
<td>4</td>
<td>57</td>
<td>0</td>
<td>2</td>
<td>65</td>
<td>0</td>
<td>2</td>
<td>73</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>0</td>
<td>2</td>
<td>50</td>
<td>2</td>
<td>0</td>
<td>58</td>
<td>1</td>
<td>3</td>
<td>66</td>
<td>0</td>
<td>1</td>
<td>74</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>43</td>
<td>1</td>
<td>3</td>
<td>51</td>
<td>0</td>
<td>1</td>
<td>59</td>
<td>2</td>
<td>4</td>
<td>67</td>
<td>1</td>
<td>2</td>
<td>75</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>44</td>
<td>2</td>
<td>4</td>
<td>52</td>
<td>1</td>
<td>2</td>
<td>60</td>
<td>0</td>
<td>0</td>
<td>68</td>
<td>2</td>
<td>3</td>
<td>76</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>45</td>
<td>0</td>
<td>0</td>
<td>53</td>
<td>2</td>
<td>3</td>
<td>61</td>
<td>1</td>
<td>1</td>
<td>69</td>
<td>0</td>
<td>4</td>
<td>77</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>46</td>
<td>1</td>
<td>1</td>
<td>54</td>
<td>0</td>
<td>4</td>
<td>62</td>
<td>2</td>
<td>2</td>
<td>70</td>
<td>1</td>
<td>0</td>
<td>78</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>47</td>
<td>2</td>
<td>2</td>
<td>55</td>
<td>1</td>
<td>0</td>
<td>63</td>
<td>0</td>
<td>3</td>
<td>71</td>
<td>2</td>
<td>1</td>
<td>79</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>80</td>
<td>2</td>
<td>0</td>
<td>88</td>
<td>1</td>
<td>3</td>
<td>96</td>
<td>0</td>
<td>1</td>
<td>104</td>
<td>2</td>
<td>4</td>
<td>112</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>81</td>
<td>0</td>
<td>1</td>
<td>89</td>
<td>2</td>
<td>4</td>
<td>97</td>
<td>1</td>
<td>2</td>
<td>105</td>
<td>0</td>
<td>0</td>
<td>113</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>82</td>
<td>1</td>
<td>2</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>98</td>
<td>2</td>
<td>3</td>
<td>106</td>
<td>1</td>
<td>1</td>
<td>114</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>83</td>
<td>2</td>
<td>3</td>
<td>91</td>
<td>1</td>
<td>1</td>
<td>99</td>
<td>0</td>
<td>4</td>
<td>107</td>
<td>2</td>
<td>2</td>
<td>115</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>84</td>
<td>0</td>
<td>4</td>
<td>92</td>
<td>2</td>
<td>2</td>
<td>100</td>
<td>1</td>
<td>0</td>
<td>108</td>
<td>0</td>
<td>3</td>
<td>116</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>85</td>
<td>1</td>
<td>0</td>
<td>93</td>
<td>0</td>
<td>3</td>
<td>101</td>
<td>2</td>
<td>1</td>
<td>109</td>
<td>1</td>
<td>4</td>
<td>117</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>86</td>
<td>2</td>
<td>1</td>
<td>94</td>
<td>1</td>
<td>4</td>
<td>102</td>
<td>0</td>
<td>2</td>
<td>110</td>
<td>2</td>
<td>0</td>
<td>118</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>87</td>
<td>0</td>
<td>2</td>
<td>95</td>
<td>2</td>
<td>0</td>
<td>103</td>
<td>1</td>
<td>3</td>
<td>111</td>
<td>0</td>
<td>1</td>
<td>119</td>
<td>2</td>
</tr>
</tbody>
</table>
Table 3
Scrambling for Example (7.12)
\( n \) vs. \( v = (v_1, v_2, v_3) \)

<table>
<thead>
<tr>
<th></th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( v_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>105</td>
<td>90</td>
<td>75</td>
</tr>
<tr>
<td>96</td>
<td>81</td>
<td>66</td>
<td>51</td>
</tr>
<tr>
<td>72</td>
<td>57</td>
<td>42</td>
<td>27</td>
</tr>
<tr>
<td>48</td>
<td>33</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>24</td>
<td>9</td>
<td>114</td>
<td>99</td>
</tr>
<tr>
<td>40</td>
<td>25</td>
<td>10</td>
<td>115</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>106</td>
<td>91</td>
</tr>
<tr>
<td>112</td>
<td>97</td>
<td>82</td>
<td>67</td>
</tr>
<tr>
<td>88</td>
<td>73</td>
<td>58</td>
<td>43</td>
</tr>
<tr>
<td>64</td>
<td>49</td>
<td>34</td>
<td>19</td>
</tr>
<tr>
<td>80</td>
<td>65</td>
<td>50</td>
<td>35</td>
</tr>
<tr>
<td>56</td>
<td>41</td>
<td>26</td>
<td>11</td>
</tr>
<tr>
<td>32</td>
<td>17</td>
<td>2</td>
<td>107</td>
</tr>
<tr>
<td>8</td>
<td>113</td>
<td>98</td>
<td>83</td>
</tr>
<tr>
<td>104</td>
<td>89</td>
<td>74</td>
<td>59</td>
</tr>
</tbody>
</table>

-70-
Now, the adopted scrambling scheme guarantees that all elements of $\mathcal{E}_{k-1}$ occupying the same identical position in the other submatrices of $\mathcal{E}_{k-1}'$ have $u_1, v_1$ which differ from (7.16) only in $u_k, v_k$ and these satisfy (7.15). This means that the difference between an element in submatrix $(r,s)$ of $\mathcal{E}_{k-1}$ and the corresponding element in $\mathcal{E}_k$ (submatrix $(0,0)$) is just (7.17), (7.14)

$$\Delta E(r,s) = (0, \ldots, 0, rs \mod N_k, 0, \ldots, 0).$$  \hspace{1cm} (7.17)

We notice the important fact that the difference is independent of the specific common location of the two paired elements in their respective submatrices. Hence, the difference is constant throughout the $(r,s)$ submatrix. In other words, the $(r,s)$ submatrix of $\mathcal{E}_{k-1}$ could be generated from $\mathcal{E}_k$ simply by adding the constant $\Delta E(r,s)$ to all its elements.

We are interested in an explicit expression for this important constant. Considering its implicit formulation (7.17), we conclude that it must be some integral multiple of $n_k$ (7.3). Specifically,

$$\Delta E(r,s) = \eta(r,s)n_k$$  \hspace{1cm} (7.18)

where the integer $\eta$ satisfies

$$\eta < N_k$$  \hspace{1cm} (7.19)

$$n_k \eta - rs \equiv 0 \pmod{N_k}.$$  \hspace{1cm} (7.20)

Eqn. (7.20) is a linear congruence for the unknown $\eta$ for which there is an explicit solution [8], namely,

$$\eta = (rs') \mod N_k$$  \hspace{1cm} (7.21)

where

$$s' = (s \zeta_k') \mod N_k$$  \hspace{1cm} (7.22)

$$\zeta_k = n_k^{-1} \mod N_k.$$  \hspace{1cm} (7.23)

The result we have established for the submatrices of $\mathcal{E}_{k-1}$ translates as follows for the submatrices of $\Omega_{k-1}$: The $(r,s)$ submatrix of $\Omega_{k-1}$

\footnote{\text{$\phi(n)$ is the Euler Totient function defined as the number of integers not exceeding, and relatively prime to, $n$.}}
is just \( W^\Delta E(r,s) \Omega_k \). Applying (7.18), we find that
\[
W^\Delta E(r,s) = e^{-i \frac{2\pi}{N_k} \eta(r,s)}
\]  

(7.24)

Denoting now
\[
W_k = e^{-i \frac{2\pi}{N_k}}
\]

(7.25)

we get
\[
W^\Delta E(r,s) = W_k \eta(r,s)
\]

(7.26)

so that the \((r,s)\) "element" of the compound \(N_k\)-th order \(\Omega_{k-1}\) is
\[
W_{\Omega_k} \eta(r,s)_{\Omega_k} = W_k (rs') \mod N_k \Omega_k = W_k (rs') \mod N_k
\]

This is very similar to the \((r,s)\) element of the DFT matrix (2.4) of order \(N_k\) with \(\Omega = \Omega_k\). The only difference is that \(s'\) has now replaced \(s\). This, however, is a trivial difference involving only column permutations. To prove this, it is sufficient to show that there is a one-to-one correspondence between \(s\) and \(s'\) so that when \(s\) goes through the values \(0, 1, \ldots, N_k - 1\), \(s'\) goes through a permutation of them. This will, indeed, be the case if \(z_k\) (in (7.22)) is relatively prime to \(N_k\). But this is guaranteed by (7.23) since \(n_k\) is relatively prime to \(N_k\) (see (7.3)).

We conclude that a trivial modification of the DFT tableau of order \(N_k\), will evaluate the transformation effected by \(\Omega_{k-1}\). Specifically, we should modify the input square of the \(N_k\) tableau by permuting the \(f_i\) rows/columns so that the \(i\) sequence would be identical with the \(s'\) sequence (instead of the natural number sequence \((s)\) of the unmodified tableau). We refer to such a tableau as a "modified tableau" and use this term from now on, only with this restricted, precise, meaning.

We illustrate now the tableau modification with \(k = 1\) in our example (7.12)

\[
n_1 = N_2 N_3 = 15; \quad \phi(N_1) = \phi(8) = 4
\]

\[
z_1 = 15^4 - 1 \mod 8 = (-1)^3 \mod 8 = 7
\]

Hence
Fig. 19. Input square of the 8-th order modified tableau for example (7.12)

Fig. 20. Subdivision of the Y matrix for example (7.29)
The modified input square called for by (7.27) is shown in Fig. 19. Note that, in this case, the modification involves only sign changes.

The result we have established may be summarized as follows:

\[
\text{Submatrix } \Omega_{k-1} \text{ of } Y \text{ is a column permutation of the compound DFT matrix of order } N_k \text{ (2.4) with } \Omega = \Omega_k. \quad \text{This is valid for all } 1 \leq k \leq K.
\]

(7.28)

This differs from (7.4) in the column permutation. But, as we have just pointed out, this only means that in the algorithm for order \( N \) (7.1), the standard tableaus should be replaced by the modified tableaus. This is the minor complication referred to earlier.

We turn now to a simple example to illustrate the algorithm developed here. The example chosen has the following parameters:

\[
K = 3; \quad N_1 = 3; \quad N_2 = 2; \quad N_3 = 5; \quad \therefore N = 30 \quad (7.29)
\]

The first step is the scrambling of the input vector \( \hat{F} \) to yield the vector \( q \). This and the concurrent scrambling of the output \( \hat{F} \), transform the DFT matrix into the \( Y \) matrix of (7.13). \( Y \) is now subdivided as indicated schematically in Fig. 20 (the indicated "measurements" refer to the number of rows/columns). The next step is to regard \( Y(=\Omega_o) \) as a compound third-order matrix \( (N_1 = 3) \) and apply the third order modified DFT tableau with the tableau's \( \Omega \) identified with \( \Omega_1 \) of Fig. 20. The application of the tableau is shown in Fig. 21 which utilizes the schematic tableau representation introduced in Fig. 18. Phase 1 is shown at the top (circled \( \Omega_o \)), phase 3 is at the bottom (\( \Omega_o \) in a square) and phase 2 is all the region in between. Phase 1 requires separation of the input vector into its three "components". This is implemented in the following straightforward manner:

| \( s \) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| \( s' = (7s) \mod 8 \) | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

(7.27)
Fig. 21. Algorithm for DFT of order 30 (example (7.29))
The outputs of phase 1 are the three \( \beta_1 \) vectors of dimensionality 10. Before considering phase 2, we have to introduce the generalized notation we use there for the \( \omega_1 \)'s. \( \omega_1 \) for the DFT tableau of order \( N \) is referred to now as \( \omega_{1, N'} \) (\( \omega_{1, N'} \) also appearing in Fig. 21, will be defined later). These constants are either explicitly listed in the tableaus or are inferred from the convention that every \( \beta_1 \) is multiplied by \( (\omega_1 \Omega) \). For example, the DFT tableau of order 5 states explicitly \( \omega_{1,5} = -\frac{5}{4} \). Implicitly, we infer \( \omega_{0,5} = 1 \). All the \( \omega_{ij} \)'s are tabulated in Table 4 for convenience. The values there have been computed on a 10-digit calculator. For more precise values, one should refer to the exact expressions in the tableaus.

Returning now to Fig. 21, we find that phase 2 of the third order tableau requires the evaluation of \( \xi_1 = (\omega_{13} \Omega_1) \beta_1 \). Here we apply again result (7.28) which implies (with \( k = 2 \)) that the transformation \( (\omega_{13} \Omega_1) \beta_1 \) could be evaluated with the modified second order DFT tableau (with the tableau's \( \Omega \) identified with \( (\omega_{13} \Omega_2) \) (see Fig. 20)). Each of the three 10-dimensional \( \beta_1 \)'s is therefore shown in Fig. 21 as the input to phase 1 of a second order tableau. In each of these applications of the second order tableau, phase 1 yields a pair of 5-dimensional \( \beta \) vectors. Consider now the specific 5-dimensional \( \beta \) vector on the extreme left of Fig. 21. It has been generated by phase 1 of the tableau implementing the transformation based on the matrix \( (\omega_{03} \Omega_1) \). Therefore, phase 2 calls for its transformation by the matrix \( \omega_{02} (\omega_{03} \Omega_2) \). Fig. 21 shows, instead, the matrix \( \hat{\Omega}_{02} \Omega_2 \). We have adopted here the following somewhat unusual terminology:

\[
\hat{\omega}_{ij} = \omega_{ij} *(\text{the first } \Omega_k \text{ multiplier met in moving against the arrows in the upper half of Fig. 21})
\] (7.31)

Thus, \( \hat{\omega}_{ij} \) is only defined with respect to a specific diagram. Furthermore, the same symbol may have a different numerical value at a different location.
Table 4

The DFT Tableau Multipliers \( (\omega_{k,N}) \)

<table>
<thead>
<tr>
<th>N</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1.5</td>
<td>1</td>
<td>-1.25</td>
<td>-1.1666667E-9</td>
<td>7.0710678E-1</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>18.6602540E-1</td>
<td>-5.5901699E-1</td>
<td>5.5854267E-2</td>
<td>1</td>
<td>-1.7364818E-1</td>
<td>1</td>
<td>-1.3065630E-0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>11.5388418E-0</td>
<td>7.3430220E-1</td>
<td>1</td>
<td>9.3969262E-1</td>
<td>7.0710678E-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>-13.6327126E-1</td>
<td>-18.7484229E-1</td>
<td>1</td>
<td>1</td>
<td>-13.4202014E-1</td>
<td>1</td>
<td>-13.4202014E-1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-15.8778525E-1</td>
<td>-15.3396936E-1</td>
<td>1</td>
<td>1</td>
<td>-18.6602540E-1</td>
<td>1</td>
<td>5.4119610E-1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>14.4095855E-1</td>
<td>7.9015647E-1</td>
<td>17.0710678E-1</td>
<td>-18.6602540E-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>-13.4087293E-1</td>
<td>-13.4087293E-1</td>
<td>-16.4278761E-1</td>
<td>7.6604444E-1</td>
<td>17.0710678E-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>-11.3065630E-0</td>
<td>17.0710678E-1</td>
<td>-15.4119610E-1</td>
<td>3.8268343E-1</td>
<td>19.2387953E-1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

NOTE: Numbers shown in 3 digits or less are exact.

All other numbers are in Fortran E-format \((3.4E-1 = 3.4 \times 10^{-1})\).
in the diagram. For example, we have just seen that on the extreme left,
\[ \hat{\omega}_{02} = \omega_0 \omega_2 \omega_3. \] This symbol appears in two other places to the right. The first one equals \( \omega_0 \omega_2 \omega_3 \), the second equals \( \omega_0 \omega_2 \omega_3 \).

Returning now to the main line of the argument, each one of the \( S \)-dimensional \( \beta \) vectors should now be input to a modified DFT tableau of order \( S \). This time, the \( \beta \) "vectors" generated in phase 1 are of dimensionality 1, namely, scalars. They are to be multiplied by \( \hat{\omega}_{15} \Omega_3 = \hat{\omega}_{15} \) (\( \Omega_3 \) is the scalar 1). At this point, the multiplications are actually carried out and the numerical values of the multipliers are needed. Their determination is straightforward. Consider for example the multiplier on the extreme left of Fig. 21

\[ \hat{\omega}_{05} = \omega_0 \omega_5 \omega_2 = \omega_0 \omega_2 \omega_3. \]

Similarly, for the multiplier on the extreme right

\[ \hat{\omega}_{55} = \omega_5 \hat{\omega}_{12} = \omega_5 \omega_1 \omega_2 \omega_3. \]

and in general, the value of the multiplier at a specific node is the product of all the \( \omega_{ij} \) terms (stripped of their circumflex) which one encounters in moving from that node up the (inverted) tree structure to the stem.

The multiplications are indicated by the half-circle, half-square shapes stringed along the center line of Fig. 21. (These can be regarded as representing the combined three phases of the DFT algorithm of order 1 with \( \Omega = \hat{\omega}_{15} \)).

The 36 terms we get after performing the multiplications comprise 6 independent groups resulting from the 6 separate applications of the modified tableau of order \( S \). The terms of each of these groups are now combined as prescribed in phase 3 of the \( S \)-th order tableau to yield six \( S \)-dimensional vectors. Each of these is actually a term of the form \( \hat{\omega}_{12} \Omega_2 \beta_1 \) required to complete the computations in the three applications of the second order tableau. These computations now yield, as tableau outputs, three 10-dimensional vectors which are identical with \( \omega_{13} \hat{\omega}_{12} \Omega_2 \beta_1 \) of the third order tableau. Combining these as prescribed by phase 3 of this tableau yields the 30-dimensional \( Q \) vector which is just a scrambled version of the desired vector \( \hat{F} \).

Fig. 21, though directly applicable to example (7.29) only, is characteristic of all \( N \) values. For larger \( N \), there might be one more level of branching in each half of the diagram and the number of branches per node may be higher. Otherwise, the structure is identical with that of Fig. 21.
VIII. Speed Analysis

In sections IV-VI we constructed the basic DFT tableaus. In section VII we showed how to use them in an efficient algorithm for certain N values. Our purpose in this section is to determine just how fast the resulting algorithm is and present a summary (Table 6) of the pertinent parameters for all orders realizable with the constructed tableaus. The basis for these developments is Table 5 which presents a summary of the tableau parameters. These have been collected from the tableau designations and have been fully explained earlier. The values appearing here are in general agreement with Winograd's results\(^\text{14}\) (Table 1 of 3\(^\text{1}\)). The only difference is in the number of multiplications in the tableau of order 9. Winograd uses 13; we use 11. This means that, using this tableau in the computation of any DFT whose order is divisible by 9, will yield a 15% reduction in the number of multiplications as compared to Winograd's results (see (8.3)). In most of these cases there is also a reduction in the number of additions (see (8.9)).

Consider now the algorithm of order \(N = \prod_{k=1}^{K} N_k\) as applied to complex data\(^\text{15}\). For each \(N_k\), we read off from Table 5 the corresponding number of complex multiplications \(M_k\) and complex additions \(A_k\). We are interested in two functions of these variables, namely, the total number of real multiplications \(\mathcal{M}\) and the total number of real additions \(\mathcal{A}\) for the overall algorithm realized in the order implied in (7.1) (phase 1 of the \(N_1\) tableau realized first; phase 1 of the \(N_K\) tableau realized last). With this goal in mind, we turn now to a mathematical formulation of some of the characteristics of the algorithm structure which are quite evident in Fig. 21.

We note that the output of phase 1 of the \(N_1\) tableau is a set of \(M_1\) vectors \((\beta_1)\) of dimensionality \(v_1\). Each of these now generates (at the output of phase 1 of the \(N_2\) tableau) \(M_2\) vectors of dimensionality \(v_2\), and so on. It is obvious therefore that

\[
\text{The total output of phase 1 of all the tableaus of order } N_k \text{ consists of } \left(\prod_{i=1}^{K} M_i\right) \text{ vectors of dimensionality } v_k \quad (7.2)
\]

\(^{14}\) Our \(M\) values, representing the total number of multiplications, should be compared to the sum of Winograd's two multiplication columns.

\(^{15}\) The computation of the number of arithmetic operations for real data is more involved and will only be briefly discussed later on.
Table 5

Summary of Basic DFT Tableaus

<table>
<thead>
<tr>
<th>Tableau Order</th>
<th>Total Number of Multiplications</th>
<th>Number of Multiplications by 1 or i</th>
<th>Number of Additions</th>
<th>Tableau Figure</th>
<th>Tableau Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>9</td>
<td>37</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>10</td>
<td>39</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>1</td>
<td>17</td>
<td>6</td>
<td>31</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>1</td>
<td>36</td>
<td>8</td>
<td>35</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>6</td>
<td>26</td>
<td>14</td>
<td>51</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>1</td>
<td>44</td>
<td>13</td>
<td>45</td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td>8</td>
<td>74</td>
<td>17</td>
<td>60</td>
</tr>
</tbody>
</table>
As each one of these vectors is fed to phase 1 of an \( N_{k+1} \) tableau, this is also the number of tableaus of order \( N_{k+1} \) or, equivalently,

\[
\begin{align*}
\text{The number of tableaus of order } N_k \text{ is } & \prod_{i=1}^{k-1} M_i \\
\end{align*}
\]

(8.2)

The simplest application of these results is the determination of the number of multiplications. Let \( M_K \) be the total number of (complex) scalar multiplications in phase 2 of the algorithm realized as a cascade of \( \kappa \) stages. The variables which are multiplied are the \( 1 \)-dimensional \( \beta_i's \) generated in the last stage of the cascade \( (N_K) \). From (8.1) we know that there are \( \prod_{i=1}^{K} M_i \) such terms. Hence

\[
M_K = \prod_{i=1}^{K} M_i
\]

(8.3)

To get the total number of real multiplications in the overall cascade \( (M) \), we note that in each of the counted multiplications, only one of the two factors is complex, the other being real or imaginary (see Table 4). Therefore,

\[
M = 2M_K
\]

(8.4)

We have seen in Section VII that each of the overall multipliers for the cascade is a product of \( \kappa \) tableau multipliers, one from each tableau of the cascade. Since each tableau has, at least, one multiplier whose value is 1 or \( i \) \((m \geq 1 \text{ in Table 5})\), some of the \( M_K \) overall multipliers will be 1 or \( i \). A consideration of the structure of Fig. 21 shows that the number of such multipliers is

\[
P_K = \prod_{i=1}^{K} m_i
\]

(8.5)

\(^{16}\text{All permissible } N \text{ values are expressible as } N = H2^r \text{ (H odd; } 0 \leq r \leq 4).\)

Using this with the values of \( m_i \) listed in Table 5, yields

\[
P_K = 2r + \delta_{0,r}
\]
Therefore, if one accepts the somewhat more complex programming involved in the special handling of these $P_K$ trivial multipliers, the DFT can actually be computed with the smaller number of real multiplications

$$\hat{N} = 2 (M_K - P_K) \quad (8.6)$$

The summary in Table 6 covers both cases $(8.4), (8.6)$.\(^{17}\)

We turn now to the additions count. Let $A_K$ be the total number of (complex) scalar additions in a cascade of $K$ stages. Hence, the corresponding number of real additions is

$$A = 2A_K \quad (8.7)$$

The simplest way to determine $A_K$ is through a recursive argument. Assume that we have the result for a cascade of length $K-1$ and we add to it one more stage at position $K$. This has two effects. First, the additions in the first $K-1$ stages will now involve vectors whose dimensionalities are $N_K$ times their previous values. Hence, the previous $A_{K-1}$ additions are transformed into $N_K A_{K-1}$ additions. To this we should add the additions of the last stage. From (8.2), the number of tableaus in the last stage is $M_{K-1} = M_{K-1} \quad (8.3)$. Each such tableau requires $A_K$ additions of scalars. Hence, the number of scalar additions in the last stage is $M_{K-1} A_K$ and the total is

---

\(^{17}\)One could argue that in a binary machine, multiplication by $\frac{1}{2}$ is also trivial and should be excluded from the multiplication count. If this attitude is adopted, then the 9th order tableau will have 3 trivial multiplications ($m = 3$). The effect of this will be quite pronounced for $N = 144$, reducing $\hat{N}$ from 380 (Table 6) to 348.
We have added here a recursive rephrasing of (8.4) as well as the initial conditions to provide a complete prescription for the simultaneous computation of both $\mathcal{A}_k$ and $\mathcal{M}_k$. (8.8) also yields explicit formulae for $\mathcal{A}_k$. For example

$$\mathcal{A}_4 = A_1 N_2 N_3 N_4 + M_1 A_2 N_3 N_4 + M_1 M_2 A_3 N_4 + M_1 M_2 M_3 A_4 \tag{8.9}$$

An important fact clearly indicated by (8.9) is that $\mathcal{A}_k$, unlike $\mathcal{M}_k$, is also a function of the order of the $N_k$'s comprising $N$. Thus, it is important to find the cascade order minimizing $\mathcal{A}_k$.

With eqns. (8.3)-(8.8) and Table 5 at our disposal, we can now compute $\mathcal{A}, \mathcal{M}$ for any $N$ satisfying (7.1). Table 6 presents the results of such computations implemented by a simple computer program which also provides information for the selection of the most efficient cascade ordering. This $N_k$ sequence appears in the last columns of Table 6. We provide here room for two sequences since, in some cases, the same values of $\mathcal{M}, \mathcal{A}$ are obtained with two different $N_k$ sequences. In this case, the choice could be governed by arguments other than efficiency.

Each $N_k$ value appears with a bracketed number to its right. This is the $\zeta_k$ of (7.22). Thus, Table 6 provides both the sequence of tableaus to be realized and the permutation required in the input square of each tableau (see discussion preceding (7.27).

Note that adoption of the order prescribed in Table 6 is quite important. For example, with $N = 240$, Table 6 states $\mathcal{A} = 5016$ with the prescribed order 3; 16; 5. If, instead, we adopt the order 5; 16; 3, the number of real additions jumps to 5592—an increase of 11%.

We turn now to the two remaining columns of Table 6, namely, $C_0$, $R$. These are the two parameters mentioned in section I and are required to determine the speed advantage in a specific system.
Table 6

Summary of DFT Algorithm

\( \mu = \) time for one real multiplication

<table>
<thead>
<tr>
<th>Order of DFT</th>
<th>Multiplications by 1 and i, included in count</th>
<th>Speed-Gain Function Parameters</th>
<th>Number of Real Multiplications</th>
<th>Multiplications by 1 and i, excluded from count</th>
<th>Speed-Gain Function Parameters</th>
<th>Number of Real Multiplications</th>
<th>Number of Real Additions</th>
<th>Tableau Realization Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>( G(\mu) = 1 )</td>
<td>( R )</td>
<td>( A )</td>
<td>( G(\mu) = 1 )</td>
<td>( R )</td>
<td>( A )</td>
<td>( \mu )</td>
<td>( R )</td>
</tr>
<tr>
<td>2</td>
<td>1.000</td>
<td>1.000</td>
<td>0</td>
<td>2(1)</td>
<td>2.377</td>
<td>3.000</td>
<td>4</td>
<td>2(1)</td>
</tr>
<tr>
<td>3</td>
<td>1.585</td>
<td>2.000</td>
<td>6</td>
<td>3(1)</td>
<td>3.877</td>
<td>4.500</td>
<td>8</td>
<td>3(2)</td>
</tr>
<tr>
<td>4</td>
<td>2.000</td>
<td>2.000</td>
<td>12</td>
<td>4(1)</td>
<td>5.047</td>
<td>6.000</td>
<td>16</td>
<td>4(2)</td>
</tr>
<tr>
<td>5</td>
<td>2.673</td>
<td>2.833</td>
<td>22</td>
<td>5(3)</td>
<td>6.631</td>
<td>7.500</td>
<td>32</td>
<td>5(3)</td>
</tr>
<tr>
<td>6</td>
<td>3.250</td>
<td>3.000</td>
<td>36</td>
<td>6(1)</td>
<td>8.289</td>
<td>9.111</td>
<td>64</td>
<td>6(2)</td>
</tr>
<tr>
<td>7</td>
<td>3.924</td>
<td>4.000</td>
<td>56</td>
<td>7(1)</td>
<td>10.000</td>
<td>11.000</td>
<td>96</td>
<td>7(2)</td>
</tr>
<tr>
<td>8</td>
<td>4.608</td>
<td>4.333</td>
<td>84</td>
<td>8(1)</td>
<td>11.889</td>
<td>13.000</td>
<td>144</td>
<td>8(2)</td>
</tr>
<tr>
<td>9</td>
<td>5.300</td>
<td>5.000</td>
<td>112</td>
<td>9(1)</td>
<td>13.811</td>
<td>15.000</td>
<td>192</td>
<td>9(3)</td>
</tr>
<tr>
<td>10</td>
<td>6.000</td>
<td>6.000</td>
<td>144</td>
<td>10(2)</td>
<td>15.831</td>
<td>17.000</td>
<td>240</td>
<td>10(3)</td>
</tr>
<tr>
<td>11</td>
<td>6.924</td>
<td>6.333</td>
<td>180</td>
<td>11(1)</td>
<td>17.911</td>
<td>19.000</td>
<td>288</td>
<td>11(4)</td>
</tr>
<tr>
<td>12</td>
<td>8.000</td>
<td>7.500</td>
<td>226</td>
<td>12(1)</td>
<td>20.000</td>
<td>22.000</td>
<td>336</td>
<td>12(2)</td>
</tr>
<tr>
<td>13</td>
<td>9.200</td>
<td>8.625</td>
<td>280</td>
<td>13(2)</td>
<td>22.225</td>
<td>25.000</td>
<td>400</td>
<td>13(4)</td>
</tr>
<tr>
<td>14</td>
<td>12.000</td>
<td>11.000</td>
<td>360</td>
<td>16(1)</td>
<td>29.000</td>
<td>32.000</td>
<td>540</td>
<td>16(5)</td>
</tr>
<tr>
<td>15</td>
<td>15.000</td>
<td>14.000</td>
<td>448</td>
<td>19(1)</td>
<td>36.000</td>
<td>40.000</td>
<td>720</td>
<td>19(5)</td>
</tr>
<tr>
<td>16</td>
<td>18.000</td>
<td>17.000</td>
<td>560</td>
<td>22(1)</td>
<td>43.000</td>
<td>48.000</td>
<td>960</td>
<td>22(5)</td>
</tr>
</tbody>
</table>

\( R(k) \) = time for one real addition

\( N \) = number of data points

\( G(\mu) \) = speed-gain parameter

\( R \) = number of real multiplications

\( A \) = number of real additions

\( \mu \) = time for one real multiplication

\( \mu \) = time for one real addition

\( k \) = Tableau Realization Sequences
Let

\[ \mu = \frac{\text{time for one real multiplication}}{\text{time for one real addition}} \]

in the specific system considered. We define the gain \( G \), of the present algorithm over the (nominal) Cooley-Tukey algorithm by

\[ G = \frac{\mu \mathcal{M}_{CT} + \mathcal{A}_{CT}}{\mu \mathcal{M} + \mathcal{A}} \]  

(8.10)

where \( \mathcal{M}_{CT}, \mathcal{A}_{CT} \) are the Cooley-Tukey parameters introduced in (1.1). Obviously, \( G \) is the ratio of the time required by the Cooley-Tukey algorithm to the time required by Winograd's algorithm. We refer to it as the speed gain. It is a function of \( \mu \) and the four parameters appearing in (8.10). Eqn. (8.11) is a more convenient two-parameter formulation (see (1.1)).

\[ G(\mu) = G_\infty \frac{\mu + 1.5}{\mu + R} \]  

(8.11)

where

\[ G_\infty = \frac{\mathcal{M}_{CT}}{\mathcal{M}} \]  

(8.12)

\[ R = \frac{\mathcal{A}}{\mathcal{N}} \]  

(8.13)

\( G_\infty \) is the asymptotic speed gain that is approached with large \( \mu \). \( R \) prescribes a pole of the \( G(\mu) \) function (at \( \mu = -R \)) and thus determines the second asymptote. Adding to these two the zero of \( G(\mu) \) (at \( \mu = -1.5 \)) makes it very easy to sketch \( G(\mu) \) and get a sufficiently precise estimate of it.\(^\text{18}\)

We conclude with a few words regarding the real data case. The number of arithmetic operations here is not half the number in the corresponding complex data case. The main reason for that is that as the DFT of a real vector is, in general, complex, some of the intermediate entities will be complex too. For example, the 8-th order DFT tableau prescribes

\[ \beta_5 = e_2 + ie_5 \]  

(8.14)

and thus \( \beta_5 \) is complex even for real data. This means of course that the tableaus realizing \( \xi_5 = \Omega \beta_5 \) have complex data inputs.

Another factor to consider is the fact that the construction of a complex number, given its two components, does not involve any arithmetic addition in spite of the appearance of the plus symbol. In the example previously cited, both \( e_2 \) and \( e_5 \) are real when the \( f_i \)'s are real. Hence, the "addition" appearing in (8.14) is free.

\(^\text{18}\) In eqns. (8.10) - (8.13), replacing \( \mathcal{M} \) of (8.4) by \( \mathcal{N} \) of (8.6), yields the circumflexed entities \( \hat{G}, \hat{G}_\infty, \hat{R} \) appearing in Table 6.
IX. **Concluding Remarks**

We have tried to present here an orderly development of Winograd's DFT algorithm, starting with the general concept, continuing with the construction of the necessary building blocks and culminating in a detailed description of their incorporation into the overall algorithm.

In applying the algorithm, Table 6 (p. 84) is the starting point as it lists all the permissible N values with their associated performance parameters. Having chosen a particular N, the next step is to consult Table 5 (p. 80) in order to locate the specific tableaus called for in Table 6. In the actual implementation, one starts with the scrambling of the input vector and then applies phase 1 of the tableaus (in the order prescribed in Table 6) to smaller and smaller segments of the data vector in its various partially transformed states. This culminates in single component "segments" finally being multiplied by constants in phase 2. From here on, the process is reversed in the application of phase 3 of the tableaus: The scalars appearing at the output of phase 2 are combined into vectors of higher and higher dimensionality, finally culminating in an N-dimensional vector which is just a scrambled version of the transformed input vector.

The present paper contains sufficiently detailed information upon which one could base a direct, straightforward, implementation of the above process in either hardware or software. There are, however, some less obvious implementations which have various advantages. These are the subject of a forthcoming paper and will not be discussed here. Nevertheless, attention should be called to a certain feature of the tableaus specifically designed into them to facilitate the application of these special techniques. We are referring here to "in-place" transformation. Consider, for example, the 7-th order tableau (Fig. 8). Note that the input components \( f_2, f_5 \) are used to compute \( c_2, c_5 \) and nothing else. Hence, there is no need to assign additional storage for \( c_2, c_5 \). They may be stored back into the \( f \) array, overwriting \( f_2, f_5 \). The only requirement is for a temporary storage for, say, \( f_2 \) so that after we store \( c_2 \) we still have \( f_2 \) available for the computation of \( c_5 \). Note that even if \( f_2 \) represents a vector, we still need only a one-word temporary store since the computation is carried out one component at a time.

This property which we have illustrated here with the \( (f_2, f_5) \rightarrow (c_2, c_5) \) transformation is common to all variables in the DFT tableaus of orders 2, 3, 5, 7.
It is also valid for the remaining tableaus, if we regard \( \eta \) as the output vector. The only deviation from the above pattern is that, in some cases, groups of 3 components (rather than 2) have to be considered. In the above tableau, the computation of \( \beta_1, \beta_2, \beta_3 \) is such an example. Note however, that, even in this case, a one-word temporary store is sufficient.

It should be pointed out that in those applications in which this "in-place" feature is not utilized, the tableaus of orders 4, 9, 8, 16, may be simplified somewhat by permuting the \( F_4 \) rows/columns in the output square to yield an unscrambled output vector \( F \). In this case, of course, the vector \( \eta \) may be dispensed with.

We turn now to a brief consideration of the precision disadvantage mentioned in section I. We have already seen in Appendix A (last paragraph) that some of the manipulations generating the tableaus have a detrimental effect on precision. A similar situation afflicts the computation of \( \delta_1 \) in some of the tableaus. Examination of eqns. (4.11), (4.12) reveals that the adopted formulation (4.12) involves addition and subtraction of \( \triangle^1 \). Hence if \( |\triangle^1| >> |\delta_1| \) we are bound to have problems, namely, loss of significant bits in floating point arithmetic and tendency to overflow in fixed-point arithmetic. Similar addition-subtraction manipulations are dispersed in various disguises throughout the tableaus' derivations.

The effect of these peculiarities of the tableaus is that to guarantee a certain measure of precision in the transformation, we probably need more bits per word than in the Cooley-Tukey algorithm. We do not analyze this effect here but it should be pointed out that the structure of the algorithm as displayed in Fig. 21 makes such an analysis relatively simple.

Finally, we conclude with yet another important aspect of the algorithm brought forth in Fig. 21, namely, the suitability of its structure to the application of various schemes of parallel processing and pipelining. Indeed, there is fertile ground here for all sorts of ingenious designs and variations. As the algorithm becomes more widely known, more and more of these will undoubtedly materialize.

Acknowledgement

The author wishes to express his thanks to Dr. L. D. Baumert of JPL for his helpful comments regarding some number-theoretic aspects of this work.
Appendix: Polynomial Congruences

The derivations in section III require numerous evaluations of

\[ R(x) = S_n(x) \mod m(x) \]  \hspace{1cm} (A.1)

where

\[ S_n(x) = \sum_{k=0}^{n} s_k x^k \]  \hspace{1cm} (A.2)

and \( m(x) \) is a monic polynomial of degree 1 or 2, whose roots lie on the unit circle. We establish here all the needed results.

1. \( m(x) = x - x_0 \) \((x_0 = \pm 1)\)

\( R(x) \) must be of degree 0

\[ R(x) = r_0 \]  \hspace{1cm} (A.3)

and (A.1) is equivalent in this case to

\[ S_n(x) = (x - x_0) Q_{n-1}(x) + r_0 \]  \hspace{1cm} (A.4)

where \( Q_{n-1}(x) \) is a polynomial of degree \((n-1)\). Substituting \( x = x_0 \) in (A.4) we find

\[ R(x) = r_0 = S_n(x_0) \]  \hspace{1cm} (A.5)

Note that with \( x_0 = \pm 1 \), \( r_0 \) is a multiplication -free algebraic sum of the coefficients of \( S_n(x) \).

2. \( m(x) = x^2 - 2x \cos \theta + 1 = (x - x_0)(x - \overline{x_0}) \)

\( R(x) \) must be of degree 1

\[ R(x) = r_0 + r_1 x \]  \hspace{1cm} (A.6)

and (A.1) is equivalent to

\[ S_n(x) = (x - x_0)(x - \overline{x_0}) Q_{n-2}(x) + (r_0 + r_1 x) \]  \hspace{1cm} (A.7)

Hence

\[ S_n(x_0) = r_0 + r_1 x_0 \]  \hspace{1cm} (A.8)

\[ S_n(\overline{x_0}) = r_0 + r_1 \overline{x_0} \]  \hspace{1cm} (A.9)

Note that

\[ x_0 = \cos \theta + i \sin \theta = e^{i \theta} \]  \hspace{1cm} (A.10)
Hence, subtracting (A.9) from (A.8) yields

\[ 2i r_1 \sin \theta = S_n(x_0) - S_n(\overline{x}_0) = \sum_{k=0}^{n} s_k (e^{ik\theta} - e^{-ik\theta}) \]

\[ r_1 = \frac{1}{\sin \theta} \sum_{k=0}^{n} s_k \sin k\theta \]

\[ r_1 = \sum_{k=1}^{n} s_k U_{k-1}(\cos \theta) \quad (A.11) \]

where \( U_m(x) \) is the Chebyshev polynomial of the second kind.

To get \( r_0 \), we multiply (A.8) by \( \overline{x}_0 \) and (A.9) by \( x_0 \) and then subtract, getting

\[ 2i r_0 \sin \theta = x_0 S_n(\overline{x}_0) - \overline{x}_0 S_n(x_0) = \sum_{k=0}^{n} s_k (e^{-i(k-1)\theta} - e^{i(k-1)\theta}) \]

\[ = 2is_0 \sin \theta - 2i \sum_{k=2}^{n} s_k \sin(k-1)\theta \]

\[ \therefore r_0 = s_0 - \sum_{k=2}^{n} s_k U_{k-2}(\cos \theta) \quad (A.12) \]

The specific \( m(x) \) polynomials for which \( R(x) \) is required are listed in Table A1 with the corresponding \( \theta \) values. Note that for these values of \( \theta \), \( U_k(\cos \theta) \) takes only the values 0, ±1. Hence, both \( r_0 \) and \( r_1 \) are multiplication-free algebraic sums of the coefficients of \( S_n(x) \). The results for all required degrees of \( S_n(x) \) are shown in Table A1.

A specific application of Table A1 is the following: Given

\[ P(x) \mod m(x) = p_1 x + p_0 \quad (A.13) \]

\[ Q(x) \mod m(x) = q_1 x + q_0 \quad (A.14) \]

Find

\[ G(x) = g_1 x + g_0 = \{P(x)Q(x)\} \mod m(x) \quad (A.15) \]

\( G(x) \) can be expressed in terms of (A.13) (A.14) as follows:
Table A1: $S_n(x) \mod m(x)$

\[ S_n(x) = \sum_{k=0}^{n} a_k x^k \]

<table>
<thead>
<tr>
<th>$m(x)$</th>
<th>$\theta$</th>
<th>$S_2(x)$</th>
<th>$S_3(x)$</th>
<th>$S_5(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^2-x+1$</td>
<td>$60^\circ$</td>
<td>$(s_1+s_2)x + (s_0-s_2)$</td>
<td>$-----$</td>
<td>$(s_1+s_2-s_4-s_5)x + (s_0-s_2-s_3+s_5)$</td>
</tr>
<tr>
<td>$x^2+1$</td>
<td>$90^\circ$</td>
<td>$s_1 x + (s_0-s_2)$</td>
<td>$(s_1-s_3)x + (s_0-s_2)$</td>
<td>$-----$</td>
</tr>
<tr>
<td>$x^2+x+1$</td>
<td>$120^\circ$</td>
<td>$(s_1-s_2)x + (s_0-s_2)$</td>
<td>$(s_1-s_2)x + (s_0-s_2+s_3)$</td>
<td>$(s_1-s_2+s_4-s_5)x + (s_0-s_2+s_3+s_5)$</td>
</tr>
</tbody>
</table>
\[ G(x) = \left\{ \frac{P(x)}{Q(x)} \right\} \mod m(x) \]
\[ = \{(p_1 q_1) \cdot x^2 + (p_1 q_0 + p_0 q_1) \cdot x + p_0 q_0\} \mod m(x) \]  \hspace{1cm} (A.16)

Identifying the bracketed polynomial with \( S_2(x) \) in Table A1, we get the following results:

\[ G(x) = \left[ p_1 (q_1 + q_1) + p_0 q_1 \right] \cdot x + (p_0 q_0 - p_1 q_1) \] \hspace{1cm} (m(x) = x^2 - x + 1) \hspace{1cm} (A.17)

\[ G(x) = (p_1 q_0 + p_0 q_1) \cdot x + (p_0 q_0 - p_1 q_1) \] \hspace{1cm} (m(x) = x^2 + 1) \hspace{1cm} (A.18)

\[ G(x) = \left[ p_1 (q_0 - q_1) + p_0 q_1 \right] \cdot x + (p_0 q_0 - p_1 q_1) \] \hspace{1cm} (m(x) = x^2 + x + 1) \hspace{1cm} (A.19)

Starting with \( p_1, q_1 \), each of these formulae requires 4 multiplications. However, with the proper rearranging of terms, we can replace one of these multiplications with an extra addition, thus getting faster computation. This is accomplished by adding and subtracting \( p_0 q_0 \) from the \( g_1 \) term. This is sufficient for (A.17), (A.19). In (A.18) we also modify the \( g_0 \) term by adding and subtracting \( p_0 q_1 \).

The results are summarized in Table A2 and, as we see there, 3 multiplications are now sufficient. The \( x^2 + 1 \) case, however, seems to indicate that the price is higher than previously stated, namely, 3 extra additions. In general this is indeed true. However, we intend to apply this result to a situation where arithmetic operations involving \( p_0 \cdot p_1 \) only, do not count (precomputation). Under these conditions, the price is indeed 1 extra addition.

It should be noted that the higher speed realized by the formulae of Table A2 is accompanied by the disadvantage of requiring more bits per word. In fixed point arithmetic we will need more bits to prevent overflow of the intermediate results. In floating point arithmetic, we will need more bits to prevent loss of precision. Consider the extreme case in which \( p_0 q_0 \gg g_1 \). (A.17) would not be affected by that but, in floating point, Table A2 could yield a value for \( g_1 \) which would be pure noise.
Table A2: \( \{P(x)Q(x)\} \mod m(x) \)

\[
P(x) \mod m(x) = p_1 x + p_0
\]

\[
Q(x) \mod m(x) = q_1 x + q_0
\]

<table>
<thead>
<tr>
<th>( m(x) )</th>
<th>( {P(x)Q(x)} \mod m(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x^2-x+1 )</td>
<td>[(p_1+p_0)(q_1+q_0)-p_0q_0] (x) + ((p_0q_0-p_1q_1))</td>
</tr>
<tr>
<td>( x^2+1 )</td>
<td>[(p_1-p_0)q_0 + (q_1+q_0)p_0] (x) + ((q_1+q_0)p_0-(p_1+p_0)q_1)</td>
</tr>
<tr>
<td>( x^2+x+1 )</td>
<td>[(p_1-p_0)(q_0-q_1)+p_0q_0] (x) + ((p_0q_0-p_1q_1))</td>
</tr>
</tbody>
</table>
REFERENCES


End of Document