MATHEMATICS OF COMPUTATION, VOLUME 32, NUMBER 141
JANUARY 1978, PAGES 175–199

On Computing the Discrete Fourier Transform

By S. Winograd

Abstract. A new algorithm for computing the Discrete Fourier Transform is described. The algorithm is based on a recent result in complexity theory which enables us to derive efficient algorithms for convolution. These algorithms are then used to obtain the new Discrete Fourier Transform algorithm.

I. Introduction. A previous paper [1] investigated the minimum number of multiplications needed to obtain the coefficients of the product of two $(n-1)$st degree polynomials modulo an nth degree polynomial. In this paper we will use the results of [1] to obtain new algorithms for computing the Discrete Fourier Transform (DFT). These new algorithms use about the same number of additions as the algorithms proposed by Cooley and Tukey [2], but only about 20% of the number of multiplications which their algorithm requires.

In the second section we will summarize the results needed for the construction of the algorithms. The third section will describe the derivation of the algorithms for cyclic convolutions, the fourth section will use these algorithms to derive the algorithm for DFT's of a few tens to a few thousands of points. In the last section we will discuss algorithms for multidimensional DFT's as well as algorithms for computing the DFT of very large numbers.

II. Theoretical Background. Let

$$R_l(u) = \sum_{i=0}^{l} x_i u^i, \qquad S_m(u) = \sum_{i=0}^{m} y_i u^i$$

be two polynomials with indeterminate coefficients, and let $P(u) = u^n + \Sigma_{i=0}^{n-1} a_i u^i$ be a monic polynomial of degree n with coefficients in a field G. (In the applications we will use G as the field Q of the rationals, only in the last section we will use other fields.) Assume $P(u) = P_1(u) \cdot P_2(u)$ such that $P_1(u)$ and $P_2(u)$ are relatively prime, and let $n_1 = \deg(P_1)$ and $n_2 = \deg(P_2)$.

Using the Chinese Remainder Theorem, we obtain:

$$\begin{aligned} &R_l \cdot S_m \bmod P \\ &\quad = (Q_2 \cdot P_2(R_l \cdot S_m \bmod P_1) + Q_1 \cdot P_1 \cdot (R_l \cdot S_m \bmod P_2)) \bmod P, \end{aligned} \tag{1}$$

where Q_1 and Q_2 are polynomials such that

$$Q_1 P_1 + Q_2 \cdot P_2 = 1 \bmod P. \tag{2}$$

Received November 30, 1976; revised June 9, 1977.
AMS (MOS) subject classifications (1970). Primary 68A20.

Copyright © 1978, American Mathematical Society

Let $\tilde{T}$ be the set of coefficients of $R_l \cdot S_m$ and let $\tilde{T}_P$ be the set of coefficients of $R_l \cdot S_m \bmod P$. It was shown in [3] that at least $l + m + 1$ multiplications are needed to compute $\tilde{T}$ (multiplication by a fixed element $g \in G$ is not counted), and using the algorithm of [4] one can actually obtain an algorithm for computing $\tilde{T}$ using $l + m + 1$ multiplications. Clearly, we can obtain $\tilde{T}_P$ from $\tilde{T}$ using only additions and multiplications by elements $g \in G$; thus, the number of multiplications, which are counted, needed to compute $\tilde{T}_P$ is at most $l + m + 1$.

Another way of computing $\tilde{T}_P$ when P has more than one irreducible factor is by the use of the identity (1), i.e., if $P = P_1 \cdot P_2$ such that P_1 and P_2 are relatively prime then we can use the algorithm for $\tilde{T}_{P_1}$ to multiply $(R \bmod P_1) \cdot (S \bmod P_1) \bmod P_1$, and the algorithm for $\tilde{T}_{P_2}$ to multiply $(R \bmod P_2) \cdot (S \bmod P_2) \bmod P_2$, and then obtain the algorithm for $\tilde{T}_P$ using only additional additions and multiplications by elements of G. It was shown in [1] that for the case $l = m = n - 1$ the number of multiplications needed to compute $\tilde{T}_P$ is $2n - k$, where k is the number of distinct irreducible factors of P. Moreover, every algorithm which computes $\tilde{T}_P$ in $2n - k$ multiplications uses (1).

When P has only one irreducible factor, i.e., when P is a power of irreducible polynomial, we cannot use (1); but then we compute $\tilde{T}_p$ by computing $\tilde{T}$ first and then reducing modulo P.

There are two ways for computing $\tilde{T}$ using only $l + m + 1$ multiplications. The first one uses the identity

$$R_l(u) \cdot S_m(u) = R_l(u) \cdot S_m(u) \bmod \prod_{i=0}^{m+l} (u - \alpha_i), \tag{3}$$

where the α_i's are distinct elements of G. (We assume that G is large enough. Actually, we will use in this paper only G of characteristic 0.) The right-hand side of (3) can be computed using (1) in $m + l + 1$ multiplications. This algorithm is the same as the one described in [4].

A second algorithm uses the identity

$$R_l(u) \cdot S_m(u) = R_l(u) \cdot S_m(u) \bmod \prod_{i=1}^{m+l} (u - \beta_i) + x_l y_m \prod_{i=1}^{m+l} (u - \beta_i), \tag{4}$$

where the β_i's are distinct elements of G. It was shown in [1] that every algorithm for computing $\tilde{T}$ in $m + l + 1$ multiplication uses either (3) or (4).

At times it is desirable to avoid the constants which the algorithms of (3) or (4) necessitate, even at the expense of additional multiplication. One way of accomplishing this is to choose a polynomial $P(u)$ of degree $l + m + 1$ with many distinct irreducible factors, but not necessarily only linear factors. Identity (3) is then modified to:

$$R_l(u) \cdot S_m(u) = R_l(u) \cdot S_m(u) \bmod P. \tag{5}$$

Similarly, we can modify identity (4).

Another theoretical development which will be needed in this paper is that of the dual or transpose of system. Let

$$\sum_{j=1}^{s}\sum_{i=1}^{r} a_{ijk}x_i y_j, \qquad k = 1, 2, \ldots, t, \tag{6}$$

be a system of bilinear forms, and assume that we have found an algorithm for computing this system using n multiplications without using the commutative law. That is,

$$\sum_{j=1}^{s}\sum_{i=1}^{r} a_{ijk}x_i y_j = \sum_{l=1}^{n} \gamma_{k,l}\left(\sum_{i=1}^{r} \alpha_{i,l}x_i\right)\left(\sum_{j=1}^{s} \beta_{j,l}y_j\right), \qquad k = 1, 2, \ldots, t. \tag{7}$$

Multiplying both sides of (7) by z_k and summing over k, we obtain

$$\sum_{k=1}^{t}\sum_{j=1}^{s}\sum_{i=1}^{r} a_{ijk}z_k x_i y_j = \sum_{l=1}^{n} \left(\sum_{k=1}^{t} \gamma_{k,l}z_k\right)\left(\sum_{i=1}^{r} \alpha_{i,l}x_i\right)\left(\sum_{j=1}^{s} \beta_{j,l}y_j\right). \tag{8}$$

Equating the coefficients of x_i, we obtain

$$\sum_{k=1}^{t}\sum_{j=1}^{r} a_{ijk}z_k y_j = \sum_{l=1}^{n} \alpha_{i,l}\left(\sum_{k=1}^{t} \gamma_{k,l}z_k\right)\left(\sum_{j=1}^{s} \beta_{j,l}y_j\right), \qquad i = 1, 2, \ldots, r. \tag{9}$$

The left-hand side of (9) is called a dual (or transpose) system, and the right-hand side of (9) provides an algorithm for computing it using n multiplications.

III. Cyclic Convolution. Consider the problem of computing the cyclic convolution of two sets of n points $(x_0, x_1, \ldots, x_{n-1})$ and $(y_0, y_1, \ldots, y_{n-1})$. This can be written as

$$\begin{pmatrix} x_0 & x_1 & \cdots & x_{n-2} & x_{n-1} \\ x_1 & x_2 & \cdots & x_{n-1} & x_0 \\ x_3 & x_4 & \cdots & x_0 & x_1 \\ \cdot & & & & \\ x_{n-1} & x_0 & \cdots & x_{n-3} & x_{n-2} \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \cdot \\ \cdot \\ y_{n-1} \end{pmatrix}. \tag{10}$$

It is readily verified that (10) is the system of coefficients of the polynomial

$$\begin{aligned}(x_0 + x_1u + x_2u^2 + \cdots + x_{n-1}u^{n-1})& \\ (y_0 + y_{n-1}u + y_{n-2}u^2 + \cdots + y_1u^{n-1}) \bmod u^n - 1,&\end{aligned} \tag{11}$$

and we can use the results described in the previous section to compute this system.

As an example we will take $n = 3$, that is, we consider the system

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} x_0 & x_1 & x_2 \\ x_1 & x_2 & x_0 \\ x_2 & x_0 & x_1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix}, \tag{12}$$

which is the system of coefficients of

$$(x_0 + x_1u + x_2u^2)(y_0 + y_2u + y_1u^2) \bmod (u^3 - 1). \tag{13}$$

Since $u^3 - 1 = (u - 1)(u^2 + u + 1)$, we have to compute

$$\begin{aligned}(x_0 + x_1u + x_2u^2)(y_0 + y_2u + y_1u^2) &\bmod (u - 1)\\ &= (x_0 + x_1 + x_2)(y_0 + y_1 + y_2)\end{aligned} \tag{14}$$

and

$$\begin{aligned}(x_0 + x_1u + x_2u^2)(y_0 + y_2u + y_1u^2) &\bmod (u^2 + u + 1)\\ &= ((x_0 - x_2) + (x_1 - x_2)u) \cdot ((y_0 - y_1) + (y_2 - y_1)u) \bmod (u^2 + u + 1).\end{aligned} \tag{15}$$

The part of the computation (14) can be done in one multiplication. To compute (15) we first compute $\tilde{T}$, that is, the coefficients of $((x_0 - x_2) + (x_1 - x_2)u) \cdot ((y_0 - y_1) + (y_2 - y_1)u)$. This is done using the identity:

$$\begin{aligned}&((x_0 - x_2) + (x_1 - x_2)u)((y_0 - y_1) + (y_2 - y_1)u)\\ &\quad = ((x_0 - x_2) + (x_1 - x_2)u)((y_0 - y_1) + (y_2 - y_1)u) \bmod u(u + 1)\\ &\qquad + (x_1 - x_2)(y_2 - y_1)u(u + 1),\end{aligned} \tag{16}$$

which leads to the algorithm:

$$\begin{aligned}&m_1 = (x_0 - x_2)(y_0 - y_1), \qquad m_2 = (x_1 - x_2)(y_2 - y_1),\\ &m_3 = ((x_0 - x_2) - (x_1 - x_2))((y_0 - y_1) - (y_2 - y_1)) = (x_0 - x_1)(y_0 - y_2).\end{aligned}$$

$$\begin{aligned}&(x_0 - x_2)(y_0 - y_1) = m_1,\\ &(x_0 - x_2)(y_2 - y_1) + (x_1 - x_2)(y_0 - y_1) = m_1 + m_2 - m_3,\\ &(x_1 - x_2)(y_2 - y_1) = m_2.\end{aligned} \tag{17}$$

And consequently, the coefficients of (15) are

$$m_1 - m_2 \quad \text{and} \quad m_1 + m_2 - m_3 - m_2 = m_1 - m_3.$$

Defining $m_0 = (x_0 + x_1 + x_2)(y_0 + y_1 + y_2)$, we obtain

$$\begin{aligned}&(x_0 + x_1u + x_2u^2)(y_0 + y_2u + y_1u^2) \bmod (u - 1) = m_0,\\ &(x_0 + x_1u + x_2u^2)(y_0 + y_2u + y_1u^2) \bmod (u^2 + u + 1)\\ &\qquad = (m_1 - m_2) + (m_1 - m_3)u;\end{aligned} \tag{18}$$

and using (1), we obtain

$$
\begin{aligned}
&(x_0 + x_1u + x_2u^2)(y_0 + y_2u + y_1u^2) \bmod (u^3 - 1)\\
&\quad = \frac{u^2 + u + 1}{3} m_0 + \left(\frac{-(u-1)(u+2)}{3}\right)((m_1 - m_2) + (m_1 - m_3)u)\\
&\qquad\qquad \bmod (u^3 - 1)\\
&\quad = \left(\frac{m_0}{3} + \frac{m_1}{3} - \frac{2m_2}{3} + \frac{m_3}{3}\right) + \left(\frac{m_0}{3} + \frac{m_1}{3} + \frac{m_2}{3} - \frac{2m_3}{3}\right)u\\
&\qquad + \left(\frac{m_0}{3} - \frac{2m_1}{3} + \frac{m_2}{3} + \frac{m_3}{3}\right)u^2.
\end{aligned} \tag{19}
$$

In many applications either the x_i's or the y_j's are known a priori, for example, they are the tap values; and therefore, computations involving only these variables can be done beforehand, and thus should not be counted. Assuming that the operations on the x_i's are not counted, we define

$$
m'_0 = \frac{x_0 + x_1 + x_2}{3}(y_0 + y_1 + y_2), \qquad m'_1 = \frac{x_0 - x_2}{3}(y_0 - y_1), \tag{20}
$$

$$
m'_2 = \frac{x_1 - x_2}{3}(y_2 - y_1), \qquad m'_3 = \frac{x_0 - x_1}{3}(y_0 - y_2); \tag{20a}
$$

and the three desired quantities are:

$$
m'_0 + m'_1 - 2m'_2 + m'_3, \quad m'_0 + m'_1 + m'_2 - 2m'_3, \quad m'_0 - 2m'_1 + m'_2 + m'_3. \tag{20b}
$$

Another algorithm is obtained by noticing that the transpose of (12) is

$$
\begin{pmatrix} z_0 & z_2 & z_1 \\ z_1 & z_0 & z_2 \\ z_2 & z_1 & z_0 \end{pmatrix}
\begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix}
=
\begin{pmatrix} z_0 & z_1 & z_2 \\ z_1 & z_2 & z_0 \\ z_2 & z_0 & z_1 \end{pmatrix}
\begin{pmatrix} y_0 \\ y_2 \\ y_1 \end{pmatrix}. \tag{21}
$$

Transposing algorithm (20), we obtain

$$
m_0 = \frac{z_0 + z_1 + z_2}{3}(y_0 + y_1 + y_2), \qquad m_1 = \frac{z_0 + z_1 - 2z_2}{3}(y_0 - y_1), \tag{22a}
$$

$$
m_2 = \frac{-2z_0 + z_1 + z_2}{3}(y_2 - y_1), \qquad m_3 = \frac{z_0 - 2z_1 + z_2}{3}(y_0 - y_2);
$$

and the three quantities to be computed are:

$$
m_0 + m_1 + m_3, \quad m_0 + m_2 - m_3, \quad m_0 - m_1 - m_2. \tag{22b}
$$

This method of obtaining simpler algorithms by transposing the system of bilinear forms is useful for other cyclic convolutions as well. Using the Chinese Remainder Theorem usually results in $Q_1 \cdot P_1$ and $Q_2 \cdot P_2$ coefficients other than 0, 1, -1, and transposing the algorithm results in moving these coefficients to what part which can be precomputed.

The matrix in (10) can be viewed as the "multiplication table" for the group z_n of addition modulo n. In case $n = n_1 \cdot n_2$ where n_1 and n_2 are relatively prime, then z_n is isomorphic to $z_{n_1} \times z_{n_2}$. Therefore, there exists a permutation of the rows and columns of the matrix of (10) such that the resulting matrix can be partitioned into blocks of $n_2 \times n_2$ cyclic matrices, and such that the blocks form an $n_1 \times n_1$ cyclic matrix.

For example, since $6 = 2 \times 3$ we have the isomorphism

$$\begin{aligned} &0 \longrightarrow (0, 0), \quad 1 \longrightarrow (1, 1), \quad 2 \longrightarrow (0, 2), \\ &3 \longrightarrow (1, 0), \quad 4 \longrightarrow (0, 1), \quad 5 \longrightarrow (1, 2); \end{aligned} \tag{23}$$

and therefore, if we have the cyclic convolution

$$\begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \end{pmatrix} = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_0 \\ x_2 & x_3 & x_4 & x_5 & x_0 & x_1 \\ x_3 & x_4 & x_5 & x_0 & x_1 & x_2 \\ x_4 & x_5 & x_0 & x_1 & x_2 & x_3 \\ x_5 & x_0 & x_1 & x_2 & x_3 & x_4 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} \tag{24}$$

and we arrange it in the order 0, 4, 2, 3, 1, 5 (that is, first those indices whose first coordinate is 0 and the second coordinate in ascending order, and then those indices whose first coordinate is 1 and the second coordinate in ascending order), we obtain

$$\left(\begin{array}{c} \psi_0 \\ \psi_4 \\ \psi_2 \\ \hline \psi_3 \\ \psi_1 \\ \psi_5 \end{array}\right) = \left(\begin{array}{ccc:ccc} x_0 & x_4 & x_2 & x_3 & x_1 & x_5 \\ x_4 & x_2 & x_0 & x_1 & x_5 & x_3 \\ x_2 & x_0 & x_4 & x_5 & x_3 & x_1 \\ \hdashline x_3 & x_1 & x_5 & x_0 & x_4 & x_2 \\ x_1 & x_5 & x_3 & x_4 & x_2 & x_0 \\ x_5 & x_3 & x_1 & x_2 & x_0 & x_4 \end{array}\right) \left(\begin{array}{c} y_0 \\ y_4 \\ y_2 \\ \hline y_3 \\ y_1 \\ y_5 \end{array}\right), \tag{25}$$

which is the same as (24), yet exhibits the block structure.

This block structure can be used to derive an algorithm by composing two different algorithms. Using $u^2 - 1 = (u + 1)(u - 1)$, we immediately obtain:

$$\begin{pmatrix} x_0 & x_1 \\ x_1 & x_0 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} \frac{x_0 + x_1}{2}(y_0 + y_1) + \frac{x_0 - x_1}{2}(y_0 - y_1) \\ \frac{x_0 + x_1}{2}(y_0 + y_1) - \frac{x_0 - x_1}{2}(y_0 - y_1) \end{pmatrix} \tag{26}$$

for a cyclic convolution of two elements. If we define:

$$\Phi_0 = \begin{pmatrix} \psi_0 \\ \psi_4 \\ \psi_2 \end{pmatrix}, \quad \Phi_1 = \begin{pmatrix} \psi_3 \\ \psi_1 \\ \psi_5 \end{pmatrix}, \quad Y_0 = \begin{pmatrix} y_0 \\ y_4 \\ y_2 \end{pmatrix}, \quad Y_1 = \begin{pmatrix} y_3 \\ y_1 \\ y_5 \end{pmatrix},$$

(27)

$$X_0 = \begin{pmatrix} x_0 & x_4 & x_2 \\ x_4 & x_2 & x_0 \\ x_2 & x_0 & x_4 \end{pmatrix}, \quad X_1 = \begin{pmatrix} x_3 & x_1 & x_5 \\ x_1 & x_5 & x_3 \\ x_5 & x_3 & x_1 \end{pmatrix},$$

then we can write (25) as

$$\begin{pmatrix} \Phi_0 \\ \Phi_1 \end{pmatrix} = \begin{pmatrix} X_0 & X_1 \\ X_1 & X_0 \end{pmatrix} \begin{pmatrix} Y_0 \\ Y_1 \end{pmatrix}; \tag{28}$$

and using algorithm (26), we get

$$M_1 = \frac{X_0 + X_1}{2}(Y_0 + Y_1), \qquad M_2 = \frac{X_0 - X_1}{2}(Y_0 - Y_1), \tag{29}$$

$$\Phi_0 = M_1 + M_2, \qquad \Phi_1 = M_1 - M_2.$$

Computing M_1 and M_2, we use algorithm (22). Thus we obtained an algorithm for (24) which uses eight multiplications and 34 additions. It should be noted that we could have factored 6 as 3×2 and obtained a different block structure, namely that of three point convolution of 2×2 blocks. In this case we would have obtained another algorithm for (24) using eight multiplications and 38 additions.

In Appendix A we give the algorithms derived for cyclic convolution of 2, 3, 4, 5 and 6 points. These algorithms are summarized in Table 1. The algorithm given for five point cyclic convolution does not use the minimum number of multiplications. Another algorithm could have been derived using only eight multiplications, but then the number of additions would have been much larger, and the constant coefficients would not have been $0, \pm 1$.

TABLE I

n	# mult.	# add.
2	2	4
3	4	11
4	5	15
5	10	31
6	8	34

IV. One Dimensional Fourier Transform. The Discrete Fourier Transform of n points

$$A_k = \sum_{j=0}^{n-1} w^{kj} a_j, \qquad i = 0, 1, \ldots, n-1, \quad w = e^{2\pi i/n}, \tag{30}$$

can be written as $\mathbf{A} = W\mathbf{a}$ where $W_{i,j} = w^{ij}$. We will consider first the case that n is a prime. In this case the matrix $W|_{i,j\neq 0}$ can be viewed as the "multiplication table" for the group M_n of nonzero integers relatively prime to n with group operation of multiplication modulo n. As is well known, $M_{p^r} \cong z_{(p-1)p^{r-1}}$ for $p \neq 2$ a prime and $M_{2^r} \cong z_2 \times z_{2^{r-2}}$. That means that if n is a prime, we can rearrange the rows and columns of $W|_{i,j\neq 0}$ so the resulting matrix is cyclic. (The idea of rearranging the indices of the Discrete Fourier Transform of a prime number of points so as to obtain cyclic convolution was first suggested by C. M. Rader [5].) This is illustrated in (31) for the case $n = 7$, i.e., (31) is another way of writing the Discrete Fourier Transform for seven points.

$$\begin{pmatrix} A_0 \\ A_1 \\ A_3 \\ A_2 \\ A_6 \\ A_4 \\ A_5 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w^1 & w^3 & w^2 & w^6 & w^4 & w^5 \\ 1 & w^3 & w^2 & w^6 & w^4 & w^5 & w^1 \\ 1 & w^2 & w^6 & w^4 & w^5 & w^1 & w^3 \\ 1 & w^6 & w^4 & w^5 & w^1 & w^3 & w^2 \\ 1 & w^4 & w^5 & w^1 & w^3 & w^2 & w^6 \\ 1 & w^5 & w^1 & w^3 & w^2 & w^6 & w^4 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_3 \\ a_2 \\ a_6 \\ a_4 \\ a_5 \end{pmatrix}, \quad w = e^{2\pi i/7}. \tag{31}$$

We can now use the algorithm for six point cyclic convolution developed in the previous section to compute the seven point Fourier Transform. Actually, for later use, it is better to compute first $A_i - A_0$, $i = 1, 2, \ldots, 6$. (Note that we have not disturbed the symmetries and, therefore, can still use the algorithms developed in the previous section.) The resulting algorithm appears in Appendix B.

In case $n = p^r$ is a power of a prime number, the situation is very similar. We can permute the rows and columns of W so as to have copies of $M_{p^r}, M_{p^{r-1}}, \ldots, M_{p^0}$. This permutation is best explained by means of an example. In (32) we illustrate the permutation of W for a nine point Fourier Transform. The algorithm for the nine points Fourier Transform is in Appendix B.

$$\left(\begin{array}{c} A_0 \\ A_3 \\ A_6 \\ A_1 \\ A_2 \\ A_4 \\ A_8 \\ A_7 \\ A_5 \end{array}\right) = \left(\begin{array}{ccc:ccc:ccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & w^3 & w^6 & w^3 & w^6 & w^3 & w^6 \\ 1 & 1 & 1 & w^6 & w^3 & w^6 & w^3 & w^6 & w^3 \\ \hdashline 1 & w^3 & w^6 & w^1 & w^2 & w^4 & w^8 & w^7 & w^5 \\ 1 & w^6 & w^3 & w^2 & w^4 & w^8 & w^7 & w^5 & w^1 \\ 1 & w^3 & w^6 & w^4 & w^8 & w^7 & w^5 & w^1 & w^2 \\ \hdashline 1 & w^6 & w^3 & w^8 & w^7 & w^5 & w^1 & w^2 & w^4 \\ 1 & w^3 & w^6 & w^7 & w^5 & w^1 & w^2 & w^4 & w^8 \\ 1 & w^6 & w^3 & w^5 & w^1 & w^2 & w^4 & w^8 & w^7 \end{array}\right) \left(\begin{array}{c} a_0 \\ a_3 \\ a_6 \\ a_1 \\ a_2 \\ a_4 \\ a_8 \\ a_7 \\ a_5 \end{array}\right). \tag{32}$$

An examination of the algorithms for the seven and nine point Fourier Transform reveals that the multiplicand which depends on the powers of w is either a real number or an imaginary number—never a general complex number. This is not a peculiarity of these two numbers, but a general property of these algorithms. In the case that $n = p^r$, $p \neq 2$, the group M_{p^r} is isomorphic to $Z_{(p-1)p^{r-1}}$, and the element -1 of M_{p^r} is mapped into $\frac{1}{2}(p-1)p^{r-1}$ under the isomorphism. But since $u^{(p-1)p^{r-1}} - 1 = (u^{\frac{1}{2}(p-1)p^{r-1}} - 1)(u^{\frac{1}{2}(p-1)p^{r-1}} + 1)$, the part of the algorithm for cyclic convolution which is based on computing modulo $u^{\frac{1}{2}(p-1)p^{r-1}} - 1$ depends on $w^j + w^{-j}$ which are real numbers; and the part of the algorithm which is based on $u^{\frac{1}{2}(p-1)p^{r-1}} + 1$ depends on $w^j - w^{-j}$ which are imaginary numbers. A similar argument establishes this fact for $n = 2^r$. Table 2 summarizes the algorithms for computing the Discrete Fourier Transform of 2, 3, 4, 5, 7, 8, 9, and 16 points. The actual algorithms are given in Appendix B. Since we will later have to consider multiplication by $w^0 = 1$ as a multiplication, this is also summarized in Table 2.

TABLE 2

n	# Mult.	# Mult. by w^0	# Add.
2	0	2	2
3	2	1	6
4	0	4	8
5	5	1	17
7	8	1	36
8	2	6	26
9	10	1	45
16	10	8	74

We now turn our attention to performing the Discrete Fourier Transform of n points where n is not a power of a prime. The idea of using the Chinese Remainder Theorem for "building up" an algorithm for computing the Discrete Fourier Transform of composite numbers originated with I. J. Good [6]. Since the way we "build up" the algorithm is somewhat different from Good's method, we will describe the whole process in detail.

Assume $n = n_1 \cdot n_2$ where n_1 and n_2 are relatively prime. By the Chinese Remainder Theorem we can represent every integer $0 \leqslant i \leqslant n$ by the pair (i_1, i_2) such that if i is represented by (i_1, i_2) and j by (j_1, j_2), then $i + j \bmod n$ is represented by $(i_1 + j_1 \bmod n, i_2 + j_2 \bmod n_2)$ and $i \cdot j \bmod n$ is represented by $(i_1 \cdot j_1 \bmod n, i_2 \cdot j_2 \bmod n_2)$.

Therefore, if we let w be the nth root of unity then:

$$(33)\quad w^{k\cdot j} = w^{k\cdot j \bmod n} = w^{(k_1,k_2)(j_1,j_2)} = w^{(k_1\cdot j_1, k_2\cdot j_2)}$$
$$= w^{(k_1 j_1,0)+(0,k_2 j_2)} = w^{(k_1 j_1,0)} w^{(0,k_2 j_2)} = (w^{(1,0)})^{k_1 j_1} \cdot (w^{(0,1)})^{k_2 j_2}.$$

That means that if we permute the rows and columns of the Discrete Fourier Transform matrix so as to arrange the indices to be in the lexicographical order of their representation, it can be partitioned in $n_1 \times n_1$ blocks each of dimensions $n_2 \times n_2$. The block in position r, s will be $(w^{(1,0)})^{r \cdot s} W_2$ when the u, v entry of W_2 is $(w^{(0,1)})^{u \cdot v}$. But $w^{(1,0)}$ is w_1^a where w_1 is the n_1th root of unity (note that the number represented by $(1, 0)$ is divisible by n_2), and $w^{(0,1)}$ is w_2^b where w_2 is the n_2th root of unity. Consequently, w_2 is the same as the Discrete Fourier Transform matrix for n_2 points except that w_2 is replaced by w_2^b. If we denote by W_1 the matrix of the Discrete Fourier Transform of n_1 points where w_1 is replaced by w_1^a, then the matrix of Discrete Fourier Transform of $n_1 \cdot n_2$ points has been transformed to the direct product of W_1 and W_2.

For example, take $n = 12 = 3 \cdot 4$. The correspondence according to the Chinese Remainder Theorem is

$$\begin{array}{llll} 0-(0,0) & 1-(1,1) & 2-(2,2) & 3-(0,3) \\ 4-(1,0) & 5-(2,1) & 6-(0,2) & 7-(1,3) \\ 8-(2,0) & 9-(0,1) & 10-(1,2) & 11-(2,3) \end{array}$$

and put in lexicographical order we get the rearrangement: 0, 9, 6, 3, 4, 1, 10, 7, 8, 5, 2, 11. Thus the Discrete Fourier Transform for 12 points can be written as:

(34)

$$\begin{pmatrix} A_0 \\ A_9 \\ A_6 \\ A_3 \\ A_4 \\ A_1 \\ A_{10} \\ A_7 \\ A_8 \\ A_5 \\ A_2 \\ A_{11} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1-i & 1-1 & 1\cdot i & 1 & 1\cdot -i & 1-1 & 1\cdot i & 1 & 1\cdot -i & 1-1 & 1\cdot i \\ 1 & 1-1 & 1 & 1-1 & 1 & 1-1 & 1 & 1-1 & 1 & 1-1 & 1 & 1-1 \\ 1 & 1\cdot i & 1-1 & 1\cdot -i & 1 & 1\cdot i & 1-1 & 1\cdot -i & 1 & 1\cdot i & 1-1 & 1\cdot -i \\ 1 & 1 & 1 & 1 & w & w & w & w & w^2 & w^2 & w^2 & w^2 \\ 1 & 1-i & 1-1 & 1\cdot i & w & w-i & w-1 & w\cdot i & w^2 & w^2-i & w^2-1 & w^2\cdot i \\ 1 & 1-1 & 1 & 1-1 & w & w-1 & w & w-1 & w^2 & w^2-1 & w^2 & w^2-1 \\ 1 & 1\cdot i & 1-1 & 1\cdot -i & w & w\cdot i & w-1 & w-i & w^2 & w^2\cdot i & w^2-1 & w^2\cdot -i \\ 1 & 1 & 1 & 1 & w^2 & w^2 & w^2 & w^2 & w & w & w & w \\ 1 & 1-i & 1-1 & 1\cdot i & w^2 & w^2-i & w^2-1 & w^2\cdot i & w & w-i & w-1 & w\cdot i \\ 1 & 1-1 & 1 & 1-1 & w^2 & w^2-1 & w^2 & w^2-1 & w & w-1 & w & w-1 \\ 1 & 1\cdot i & 1-1 & 1\cdot -i & w^2 & w^2\cdot i & w^2-1 & w^2\cdot -i & w & w\cdot i & w-1 & w\cdot -i \end{pmatrix} \begin{pmatrix} a_0 \\ a_9 \\ a_6 \\ a_3 \\ a_4 \\ a_1 \\ a_{10} \\ a_7 \\ a_8 \\ a_5 \\ a_2 \\ a_{11} \end{pmatrix},$$

where w is the cubic root of unity. Since $(1, 0)$ corresponds to four, we have $a = 1$, and since $(0, 1)$ corresponds to nine we have $b = 3$.

The decomposition of the 12 point Discrete Fourier Transform leads to an algorithm for its computation. If we define

(35)

$$\mathbf{a}_0 = \begin{pmatrix} a_0 \\ a_9 \\ a_6 \\ a_3 \end{pmatrix}, \quad \mathbf{a}_1 = \begin{pmatrix} a_4 \\ a_1 \\ a_{10} \\ a_7 \end{pmatrix}, \quad \mathbf{a}_2 = \begin{pmatrix} a_8 \\ a_5 \\ a_2 \\ a_{11} \end{pmatrix},$$

$$\mathbf{A}_0 = \begin{pmatrix} A_0 \\ A_9 \\ A_6 \\ A_3 \end{pmatrix}, \quad \mathbf{A}_1 = \begin{pmatrix} A_4 \\ A_1 \\ A_{10} \\ A_7 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} A_8 \\ A_5 \\ A_2 \\ A_{11} \end{pmatrix},$$

then using Algorithm B2 we obtain:

$$M_0 = W_2 \cdot (a_0 + a_1 + a_2), \quad M_1 = \left(\cos\frac{2\pi}{3} - 1\right) W_2 \cdot (a_1 + a_2),$$

$$M_2 = i \sin\frac{2\pi}{3} W_2 \cdot (a_1 - a_2), \tag{36}$$

$$A_0 = M_0, \quad A_1 = M_0 + M_1 + M_2, \quad A_2 = M_0 + M_1 - M_2,$$

where W_2 is the four point discrete Fourier Transform with i replaced by $-i$ (since $b = 3$). Therefore, we can use the Algorithm B3 to compute M_0, M_1, and M_2. In computing M_1, for example, we have to modify Algorithm B3 by first replacing i by $-i$ and second multiplying the constants in the multiplication steps by $(\cos 2\pi/3 - 1)$. These modifications are done initially when we derive the algorithm and, therefore, are not counted in analyzing the computation complexity of the algorithm. In the end of this section we will show how one can avoid the first modification.

It should be clear that the way we derived the algorithm for computing the 12 points Discrete Fourier Transform is quite general. If $n = n_1 \cdot n_2$ (where n_1, n_2 are relatively prime) and we have algorithms for computing the Discrete Fourier Transform of n_1 points using a_1 additions and m_1 multiplications, (including multiplication by 1) and of n_2 points using a_2 additions and m_2 multiplications, we can combine them to obtain an algorithm for computing the n points Discrete Fourier Transform using $m_1 \cdot m_2$ multiplications and $n_2 \cdot a_1 + m_1 \cdot a_2$ additions. Since we would have decomposed the n points discrete Fourier Transform using $n = n_2 \cdot n_1$ as well, we could have derived an algorithm using $m_1 \cdot m_2$ multiplications and $n_1 \cdot a_2 + m_2 \cdot a_1$ additions. In general, these two algorithms will differ in their number of additions. In Table 3 we summarize the number of multiplications and additions used in algorithms derived this way for various values of n. For the sake of comparison with FFT we also tabulate $2n \log_2 n$ and $3n \log_2 n$ (the formulas for the number of real multiplications and real additions, respectively, in FFT).

n	# Mult. Complex Data	# Add. Complex Data	$2n \log_2 n$	$3n \log_2 n$
30	72	384	295	442
48	108	636	537	805
60	144	888	709	1064
120	288	2076	1658	2487
168	432	3492	2484	3726
240	648	5016	3796	5693
420	1296	11352	7320	10980
504	1584	14642	9050	13574
840	2592	24804	16320	24480
1008	3564	34920	20115	30172
2520	9504	100188	56949	85423

TABLE 3

As we saw before, one of the modifications needed to compute the Discrete Fourier Transform of $n_1 \cdot n_2$ points is to replace w_1 by w_1^a and w_2 by w_2^b. One way of avoiding this modification is to use different permutations on the rows and columns of the matrix, that is, different permutations of the input and output data.

Let $b_0, b_1, \ldots, b_{n-1}$ be the reordering of the input data $a_0, a_1, \ldots, a_{n-1}$; and let $B_0, B_1, \ldots, B_{n-1}$ be the reordering of the output data $A_0, A_1, \ldots, A_{n-1}$. Choose r_1, r_2, s_1, s_2 such that

$$r_1 \cdot s_1 \cdot n_1 = 1 \bmod n_2, \qquad r_2 \cdot s_2 \cdot n_2 = 1 \bmod n_1. \tag{37}$$

If we choose $b_{j_1 n_2 + j_2} = a_{r_1 j_1 n_1 + r_2 j_2 n_2 \bmod n}$ and $B_{k_1 n_2 + k_2} = A_{s_1 k_1 n_1 + s_2 k_2 n_2 \bmod n}$, then the resulting entry of $(k_1 n_2 + k_2, j_1 n_2 + j_2)$ of the Discrete Fourier Transform matrix is (remember that $w^n = 1$):

$$\begin{aligned} W_{(k_1,k_2)(j_1,j_2)} &= w^{(s_1 k_1 n_1 + s_2 k_2 n_2)(r_1 j_1 n_1 + r_2 j_2 n_2)} \\ &= w^{(s_1 r_1 n_1) k_1 j_1 n_1 + (s_2 r_2 n_2) k_2 j_2 n_2} \\ &= (w^{n_2})^{(s_2 r_2 n_2) k_2 j_2} \cdot (w^{n_1})^{(r_1 s_1 n_1) k_1 j_1} \\ &= w_1^{(s_2 r_2 n_2) k_2 j_2} \cdot w_2^{(r_1 s_1 n_1) k_1 j_1} \\ &= w_1^{k_2 j_2} \cdot w_2^{k_1 j_1} \qquad (w_1^{n_1} = 1, w_2^{n_2} = 1). \end{aligned} \tag{38}$$

Therefore, this matrix is the direct product of the matrix for n_1 point Discrete Fourier Transform and n_2 point Discrete Fourier Transform.

One easy way of getting r_1, r_2, s_1 and s_2 is to choose $r_1 = r_2 = 1$ and s_1, s_2 according to the Chinese Remainder Theorem.

We will end this section with the remark that the algorithms developed here can be used in conjunction with FFT. The identity behind FFT states that computing the Discrete Fourier Transform of $n_1 \cdot n_2$ points (n_1, n_2 are not necessarily relatively prime) can be done by first performing the Discrete Fourier Transform of n_1 points n_2 times, then one performs $(n_1 - 1)n_2$ complex multiplications, and then one performs n_1 times the Discrete Fourier Transform of n_2 points. It is, of course, possible to use the algorithms developed here in the first and third stages of the FFT identity.

V. Multidimensional Fourier Transform. For the sake of concreteness we will consider only two dimensional Fourier Transform, even though it should be clear that the results apply to all dimensions. The $n_1 \times n_2$ points Discrete Fourier Transform is

$$A_{k,k'} = \sum_{j=0}^{n_1-1} \sum_{j'=0}^{n_2-1} w_1^{kj} w_2^{k'j'} a_{j,j'}, \qquad 0 \leqslant k \leqslant n_1 - 1, 0 \leqslant k' \leqslant n_2 - 1. \tag{39}$$

It is apparent from (39) that the matrix for the Discrete Fourier Transform of $n_1 \times n_2$ points is the direct product of the matrix of n_1 points by the matrix for n_2 points, and consequently the methods of the end of the last section are immediately

applicable to multidimensional Discrete Fourier Transform. This can be stated even more strongly by noting that underlying the method for one dimensional Discrete Fourier Transform is the transformation to multidimensional transform.

The main results in this section consist of illustrating how the full strength of the investigation of a product of polynomials modulo a polynomial, *and their dependence on the field of scalars,* can be utilized. The discussion in the preceding paragraph indicates that the techniques to be described are applicable in the one dimensional case as well, but their exposition is simpler in the multidimensional case.

As was mentioned in Section II, the minimum number of multiplications needed to compute $\widetilde{T}_p$ is $2n - k$ where n is the degree of P and k is the number of distinct irreducible factors of P. By choosing a larger field of constants we can increase k and thus decrease the number of multiplications. For example, $\widetilde{T}_{u^4-1}$ requires five multiplications over the field Q of rationals, but only four over the field $Q(i)$. Similarly, $\widetilde{T}_{u^6-1}$ requires eight multiplications over Q, but only six over $Q(e^{2\pi i/3})$. Recalling that $\widetilde{T}_{u^4-1}$ is the cyclic convolution of four points and $\widetilde{T}_{u^6-1}$ is the cyclic convolution of six points, we see that if somehow we could take advantage of the larger fields we could reduce the complexity of cyclic convolutions and, therefore, of the Discrete Fourier Transform.

One way of utilizing algorithms over fields which are algebraic extensions of the rationals is to use them in the situation that the Discrete Fourier Transform is to be performed on more than one set of data. For the sake of concreteness assume that we have two independent sets of data: $\{a_0^{(1)}, a_1^{(2)}, \ldots, a_{N-1}^{(2)}\}$ and $\{a_0^{(2)}, a_1^{(2)}, \ldots, a_{N-1}^{(2)}\}$. We can "group" them together as $\{a_0, a_1, \ldots, a_{n-1}\}$ where $a_j = (a_j^{(1)}, a_j^{(2)})$. Assume that the field of constants we want to use is $Q(I)$, where $I^2 = -1$, i.e., the field of Gaussian rationals.

We can transform the vectors a_j into an algebra over $Q(I)$ by defining:

$$
\begin{aligned}
&1.\quad (a_j^{(1)}, a_j^{(2)}) + (a_k^{(1)}, a_k^{(2)}) = (a_j^{(1)} + a_k^{(1)}, a_j^{(2)} + a_k^{(2)}),\\
&\qquad I \cdot (a_j^{(1)}, a_j^{(2)}) = (-a_j^{(2)}, a_j^{(1)}),\\
&(40)\qquad (a_j^{(1)}, a_j^{(2)})(a_k^{(1)}, a_k^{(2)}) = (b^{(1)}, b^{(2)}),\\
&\qquad b^{(1)} = a_j^{(1)}a_k^{(1)} - a_j^{(2)}a_k^{(2)} = a_j^{(1)}(a_k^{(1)} + a_k^{(2)}) - (a_j^{(1)} + a_j^{(2)})a_k^{(2)},\\
&\qquad b^{(2)} = a_j^{(1)}a_k^{(2)} + a_j^{(2)}a_k^{(1)} = a_j^{(1)}(a_k^{(1)} + a_k^{(2)}) - (a_j^{(1)} - a_j^{(2)})a_k^{(1)}.
\end{aligned}
$$

That is, we view the vector a_j as standing for $a_j^{(1)} + I \cdot a_j^{(2)}$ where $I^2 = -1$. This is of course possible whenever the number of independent sets of data is the same as the dimension of the extension field. We see that in this setting, multiplication by I is not counted (it amounts to interchanging the components of the vector and changing one of the signs), while multiplication of two elements of the algebra amounts to three multiplications of the components plus a certain number of additions.

The pairs of data could have been transformed into an algebra of $Q(\phi)$ where $\phi^2 + \phi + 1 = 0$, if instead of (40) we would have defined:

$$
\begin{aligned}
&(a_j^{(1)}, a_j^{(2)}) + (a_k^{(1)}, a_k^{(2)}) = (a_j^{(1)} + a_k^{(1)}, a_j^{(2)} + a_k^{(2)}),\\
&\phi \cdot (a_j^{(1)}, a_j^{(2)}) = (-a_j^{(2)}, a_j^{(1)} - a_j^{(2)}),\\
&(a_j^{(1)}, a_j^{(2)}) \cdot (a_k^{(1)}, a_k^{(2)}) = (b^{(1)}, b^{(2)}),\\
&b^{(1)} = (a_j^{(1)} + a_j^{(2)})a_k^{(1)} - a_j^{(2)}(a_k^{(1)} - a_k^{(2)}),\\
&b^{(2)} = (a_j^{(1)}a_k^{(2)} + a_j^{(2)}(a_k^{(1)} - a_k^{(2)}).
\end{aligned}
\tag{41}
$$

That is if we view the vector a_j as standing for $a_j^{(1)} + \phi a_j^{(2)}$, where $\phi^2 = -\phi - 1$.

Having computed the Discrete Fourier Transform of the pair gives us the two desired Discrete Fourier Transforms.

As an example, consider performing the three dimensional Discrete Fourier Transform on $120 \times 120 \times 120$ data points, which are assumed to be complex. Using the method described in the beginning of this section would require 5,971,968 real multiplications and 97,203,456 real additions for each set of data. (For the sake of comparison with FFT, note that $2 \times 120^3 \times \log_2 120^3 = 71{,}610{,}641$ and $3 \times 120^3 \times \log_2 120^3 = 107{,}415{,}962$.) Since $120 = 8 \times 3 \times 5$, we could have reduced the number of multiplications if we could have done the four point cyclic convolution (which appears in the five point Fourier Transform) in four multiplications instead of five. Since $Q(I)$ splits $u^4 - 1$, we choose to view the pair of input data as an algebra over $Q(I)$. In Appendix C we give an algorithm for the five point Fourier Transform over $Q(I)$. Using this algorithm, we can perform the $120 \times 120 \times 120$ Fourier Transform in $2 \times 5{,}184{,}000$ real multiplications and $2 \times 96{,}840{,}800$ real additions. But since this yields the results of performing the Fourier Transform on two sets of data, we obtained 13% savings of the number of multiplications (and a slight reduction of the number of additions).

It should be clear that this construction is general. For example, computing the Discrete Fourier Transform of 252×252 points may be advantageously done over $Q(\phi)$; and the Fourier Transform of $140 \times 140 \times 140$ may be sped up by doing it over $Q(I, \phi)$.

Another, more subtle, way of utilizing the fact that $\widetilde{T}_p$ may require fewer multiplications when the field of constants is enlarged, is based on the construction in the beginning of [1].

The Chinese Remainder Theorem states that when P_1 and P_2 are relatively prime, the system $\widetilde{T}_{P_1 \cdot P_2}$ can be transformed, by appropriate change of variables, to the direct sum of $\widetilde{T}_{P_1}$ and $\widetilde{T}_{P_2}$. We will illustrate this by considering the four point cyclic convolution, i.e., $\widetilde{T}_{u^4-1}$. Since $u^4 - 1 = (u^2 - 1)(u^2 + 1)$, we obtain:

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_1 \\ x_3 & x_4 & x_1 & x_2 \\ x_4 & x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix}$$

(42)

$$\cdot \begin{pmatrix} \frac{x_1 + x_3}{2} & \frac{x_2 + x_4}{2} & 0 & 0 \\ \frac{x_2 + x_4}{2} & \frac{x_1 + x_3}{2} & 0 & 0 \\ 0 & 0 & \frac{x_2 - x_4}{2} & -\frac{x_1 - x_3}{2} \\ 0 & 0 & \frac{x_1 - x_3}{2} & \frac{x_2 - x_4}{2} \end{pmatrix} \begin{pmatrix} y_1 + y_3 \\ y_2 + y_4 \\ y_1 - y_3 \\ y_2 - y_4 \end{pmatrix}.$$

This transformation can be carried directly into the appropriate Discrete Fourier Transform. Thus, the decomposition (42) translates into the following decomposition of the five point Discrete Fourier Transform

$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & -1 \\ 1 & 0 & 1 & -1 & 0 \end{pmatrix}$$

$$\cdot \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos u - 1 & \cos 2u - 1 & 0 & 0 \\ 0 & \cos 2u - 1 & \cos u - 1 & 0 & 0 \\ 0 & 0 & 0 & i \sin 2u & -i \sin u \\ 0 & 0 & 0 & i \sin u & i \sin 2u \end{pmatrix} \begin{pmatrix} a_0 + a_1 + a_2 + a_3 + a_4 \\ a_1 + a_4 \\ a_2 + a_3 \\ a_1 - a_4 \\ a_2 - a_3 \end{pmatrix}.$$

If we consider the two dimensional Fourier Transform of 5×5 points it can be decomposed as

$$(\tilde{T}_u \oplus \tilde{T}_{u^2-1} \oplus \tilde{T}_{u^2+1}) \otimes (\tilde{T}_u \oplus \tilde{T}_{u^2-1} \oplus \tilde{T}_{u^2+1})$$

$$= \tilde{T}_u \oplus 2 \cdot \tilde{T}_{u^2-1} \oplus 2 \cdot \tilde{T}_{u^2+1} \oplus (\tilde{T}_{u^2-1} \otimes \tilde{T}_{u^2+1})$$

$$\oplus 2 \cdot (\tilde{T}_{u^2-1} \otimes \tilde{T}_{u^2+1}) \oplus (\tilde{T}_{u^2+1} \otimes \tilde{T}_{u^2+1}).$$

In [1] we showed how to compute $\tilde{T}_{u^2+1} \otimes \tilde{T}_{u^2+1}$ using six multiplications, and therefore the total number of multiplications needed to perform the 5×5 two dimensional Fourier Transform is $1 + 4 + 6 + 4 + 12 + 6 = 33$ (instead of 36). Using this construction to obtain an algorithm for the $5 \times 5 \times 5$ Fourier Transform, and

then incorporating it in computing the 120 × 120 × 120 points Fourier Transform we obtain an algorithm which uses 90,706,176 real additions and 4,810,752 real multiplications. (That is, 6.7% of the number of multiplications and 84% of the number of additions of FFT.)

It should be emphasized that the savings of the last construction occur at the expense of the length of the program. This construction calls for writing an algorithm to compute the 5 × 5 × 5 Discrete Fourier Transform.

Acknowledgment. The author wishes to thank Dr. Ramesh C. Agarwal of IBM Research for helping in simplifying the algorithm for DFT of nine points.

Appendix A

A1.
$$\begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} = \begin{pmatrix}x_1 & x_2\\ x_2 & x_1\end{pmatrix}\begin{pmatrix}y_1\\ y_2\end{pmatrix}$$

Algorithm:

$$s_1 = y_1+y_2 \qquad s_2 = y_1 - y_2$$
$$m_1 = \frac{x_1+x_2}{2}\cdot s_1 \qquad m_2 = \frac{x_1- x_2}{2}\cdot s_2$$
$$s_3 = m_1+ m_2 \qquad s_4 = m_1- m_2$$
$$\psi_1 = s_3 \qquad \psi_4 = s_4$$

A2.
$$\begin{pmatrix}\psi_1\\ \psi_2\\ \psi_3\end{pmatrix} = \begin{pmatrix}x_1 & x_2 & x_3\\ x_2 & x_3 & x_1\\ x_3 & x_1 & x_2\end{pmatrix}\begin{pmatrix}y_1\\ y_2\\ y_3\end{pmatrix}$$

Algorithm:

$$s_1 = y_1+ y_2 \qquad s_2 = y_1-y_2 \qquad s_3 = y_2- y_3$$
$$s_4 = y_3- y_1 \qquad s_5 = y_3+ s_1$$
$$m_1 = \frac{x_1+x_2+x_3}{3}\cdot s_5 \qquad m_2 = \frac{2x_1-x_2-x_3}{3}\cdot s_2$$
$$m_3 = \frac{x_1+x_2-2x_3}{3}\cdot s_3 \qquad m_4 = \frac{x_1-2x_2+x_3}{3}\cdot s_4$$
$$s_6 = m_2+ m_3 \qquad s_7 = m_4- m_3 \qquad s_8 = m_2+ m_4$$
$$s_9 = m_1+ s_6 \qquad s_{10} = m_1+ s_7 \qquad s_{11} = m_1- s_8$$
$$\psi_1 = s_9 \qquad \psi_2 = s_{10} \qquad \psi_3 = s_{11}$$

A3.
$$\begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_1 \\ x_3 & x_4 & x_1 & x_2 \\ x_4 & x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

Algorithm:

$s_1 = y_1 + y_3 \qquad s_2 = y_1 - y_3 \qquad s_3 = y_4 + y_2 \qquad s_4 = y_4 - y_2$

$s_5 = s_1 + s_3 \qquad s_6 = s_1 - s_3 \qquad s_7 = s_2 + s_4$

$m_1 = \frac{x_1+x_2+x_3+x_4}{4} \cdot s_5 \qquad m_2 = \frac{x_1-x_2+x_3-x_4}{4} \cdot s_6$

$m_3 = \frac{x_1+x_2-x_3-x_4}{2} \cdot s_2 \qquad m_4 = \frac{x_2-x_4}{2} \cdot s_7 \qquad m_5 = \frac{x_1-x_2-x_3+x_4}{2} \cdot s_4$

$s_8 = m_1 + m_2 \qquad s_9 = m_1 - m_2 \qquad s_{10} = m_3 - m_4 \qquad s_{11} = m_4 + m_5$

$s_{12} = s_8 + s_{10} \qquad s_{13} = s_8 - s_{10} \qquad s_{14} = s_9 + s_{11} \qquad s_{15} = s_9 - s_{11}$

$\psi_1 = s_{12} \qquad \psi_2 = s_{14} \qquad \psi_3 = s_{13} \qquad \psi_4 = s_{15}$

A4.
$$\begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ x_2 & x_3 & x_4 & x_5 & x_1 \\ x_3 & x_4 & x_5 & x_1 & x_2 \\ x_4 & x_5 & x_1 & x_2 & x_3 \\ x_5 & x_1 & x_2 & x_3 & x_4 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}$$

$s_1 = y_1 + y_4 \qquad s_2 = y_2 + y_3 \qquad s_3 = s_1 + s_2 \qquad s_4 = s_1 - s_2 \qquad s_5 = s_3 + y_5$

$s_6 = y_1 - y_5 \qquad s_7 = y_1 - y_2 \qquad s_8 = y_2 - y_5 \qquad s_9 = y_1 - y_3 \qquad s_{10} = y_2 - y_4$

$s_{11} = y_3 - y_5 \qquad s_{12} = y_3 - y_4 \qquad s_{13} = y_4 - y_5$

$m_0 = \frac{x_1+x_2+x_3+x_4+x_5}{5} \cdot s_5 \qquad m_1 = \frac{x_1+x_2+x_3+x_4-4x_5}{5} \cdot s_6$

$m_2 = \frac{2x_1-3x_2+2x_3-3x_4+2x_5}{5} \cdot s_7 \qquad m_3 = \frac{-4x_1+x_2+x_3+x_4+x_5}{5} \cdot s_8$

$m_4 = \frac{2x_1+2x_2-3x_3-3x_4+2x_5}{5} \cdot s_9 \qquad m_5 = \frac{-x_1-x_2-x_3+2x_4-x_5}{5} \cdot s_4$

$m_6 = \frac{2x_1+2x_2+2x_3-3x_4-3x_5}{5} \cdot s_{10} \qquad m_7 = \frac{x_1-4x_2+x_3+x_4+x_5}{5} \cdot s_{11}$

$m_8 = \frac{-3x_1+2x_2+2x_3-3x_4+2x_5}{5} \cdot s_{12} \qquad m_9 = \frac{x_1+x_2-4x_3+x_4+x_5}{5} \cdot s_{13}$

$$s_{14} = m_4+m_5 \qquad s_{15} = m_5-m_6 \qquad s_{16} = m_0+m_1 \qquad s_{17} = s_{16}+m_2$$

$$s_{18} = s_{17}+s_{14} \qquad s_{19} = m_0-m_2 \qquad s_{20} = s_{19}+m_3 \qquad s_{21} = s_{20}-s_{15}$$

$$s_{22} = m_0-s_{14} \qquad s_{23} = s_{22}+m_7 \qquad s_{24} = s_{23}+m_8 \qquad s_{25} = m_0+s_{15}$$

$$s_{26} = s_{25}-m_8 \qquad s_{27} = s_{26}+m_9 \qquad s_{28} = m_0-m_1 \qquad s_{29} = s_{28}-m_3$$

$$s_{30} = s_{23}-m_7 \qquad s_{31} = s_{30}-m_9$$

$$\psi_1 = s_{18} \qquad \psi_2 = s_{21} \qquad \psi_3 = s_{24} \qquad \psi_4 = s_{27} \qquad \psi_5 = s_{31}$$

A5.
$$\begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \\ \psi_6 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ x_2 & x_3 & x_4 & x_5 & x_6 & x_1 \\ x_3 & x_4 & x_5 & x_6 & x_1 & x_2 \\ x_4 & x_5 & x_6 & x_1 & x_2 & x_3 \\ x_5 & x_6 & x_1 & x_2 & x_3 & x_4 \\ x_6 & x_1 & x_2 & x_3 & x_4 & x_5 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}$$

$$s_1 = y_1+y_4 \qquad s_2 = y_1-y_4 \qquad s_3 = y_2+y_5 \qquad s_4 = y_5-y_2$$

$$s_5 = y_3+y_6 \qquad s_6 = y_3-y_6 \qquad s_7 = s_1+s_3 \qquad s_8 = s_7+s_5$$

$$s_9 = s_2+s_4 \qquad s_{10} = s_9+s_6 \qquad s_{11} = s_1-s_3 \qquad s_{12} = s_3-s_5$$

$$s_{13} = s_5-s_1 \qquad s_{14} = s_2-s_4 \qquad s_{15} = s_4-s \qquad s_{16} = s_6-s_2$$

$$m_1 = \frac{x_1+x_2+x_3+x_4+x_5+x_6}{6} \cdot s_8 \qquad m_2 = \frac{2x_1-x_2-x_3+2x_4-x_5-x_6}{6} \cdot s_{11}$$

$$m_3 = \frac{x_1\ x_2-2x_3+x_4+x_5-2x_6}{6} \cdot s_{12} \qquad m_4 = \frac{x_1-2x_2+x_3+x_4-2x_5+x_6}{6} \cdot s_{13}$$

$$m_5 = \frac{x_1-x_2+x_3-x_4+x_5-x_6}{6} \cdot s_{10} \qquad m_6 = \frac{2x_1+x_2-x_3-2x_4-x_5+x_6}{6} \cdot s_{14}$$

$$m_7 = \frac{x_1-x_2-2x_3-x_4+x_5+2x_6}{6} \cdot s_{15} \qquad m_8 = \frac{x_1+2x_2+x_3-x_4-2x_5-x_6}{6} \cdot s_{16}$$

$$s_{17} = m_1+m_2 \qquad s_{18} = s_{17}+m_3 \qquad s_{19} = m_1-m_3 \qquad s_{20} = s_{19}+m_4$$

$$s_{21} = m_1-m_2 \qquad s_{22} = s_{21}-m_4 \qquad s_{23} = m_5+m_6 \qquad s_{24} = s_{23}+m_7$$

$$s_{25} = m_5-m_7 \qquad s_{26} = s_{25}+m_8 \qquad s_{27} = m_5-m_6 \qquad s_{28} = s_{27}-m_8$$

$$s_{29} = s_{18}+s_{24} \qquad s_{30} = s_{18}-s_{24} \qquad s_{31} = s_{20}-s_{26} \qquad s_{32} = s_{20}+s_{26}$$

$$s_{33} = s_{22}+s_{28} \qquad s_{34} = s_{22}-s_{28}$$

$$\psi_1 = s_{29} \qquad \psi_2 = s_{31} \qquad \psi_3 = s_{33} \qquad \psi_4 = s_{30} \qquad \psi_5 = s_{32} \qquad \psi_6 = s_{34}$$

Appendix B

B1.
$$\begin{pmatrix} A_0 \\ A_1 \end{pmatrix} = \begin{pmatrix} w^0 & w^0 \\ w^0 & -w^0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \qquad w = e^{\frac{2\pi i}{2}} = -1$$

Algorithm:

$s_1 = a_0+a_1$ $\qquad$ $s_2 = a_0-a_1$

$m_0 = 1 \cdot s_1$ $\qquad$ $m_1 = 1 \cdot s_2$

$A_0 = m_0$ $\qquad$ $A_1 = m_1$

B2.
$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \end{pmatrix} = \begin{pmatrix} w^0 & w^0 & w^0 \\ w^0 & w^1 & w^2 \\ w^0 & w^2 & w^1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \qquad w = e^{\frac{2\pi i}{3}}$$

Algorithm:

$s_1 = a_1+a_2$ $\qquad$ $s_2 = a_1-a_2$ $\qquad$ $s_3 = s_1+a_0$

$m_0 = 1 \cdot s_3$ $\qquad$ $m_1 = (\cos u - 1) \cdot s_1$ $\qquad$ $m_2 = i \sin u \cdot s_2$ $\qquad$ $u = \frac{2\pi}{3}$

$s_4 = m_0+m_1$ $\qquad$ $s_5 = s_4+m_2$ $\qquad$ $s_6 = s_4-m_2$

$A_0 = m_0$ $\qquad$ $A_1 = s_5$ $\qquad$ $A_2 = s_6$

B3.
$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{pmatrix} = \begin{pmatrix} w^0 & w^0 & w^0 & w^0 \\ w^0 & w^1 & -w^0 & -w^1 \\ w^0 & -w^0 & w^0 & -w^0 \\ w^0 & -w^1 & -w^0 & w^1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} \qquad w = e^{\frac{2\pi i}{4}}$$

$s_1 = a_0+a_2$ $\qquad$ $s_2 = a_0-a_2$ $\qquad$ $s_3 = a_1+a_3$ $\qquad$ $s_4 = a_1-a_3$

$s_5 = s_1+s_3$ $\qquad$ $s_6 = s_1-s_3$

$m_1 = 1 \cdot s_5$ $\qquad$ $m_2 = 1 \cdot s_6$ $\qquad$ $m_3 = 1 \cdot s_2$ $\qquad$ $m_4 = i \sin u \cdot s_4$ $\qquad$ $u = \frac{2\pi}{4}$

$s_7 = m_3+m_4$ $\qquad$ $s_8 = m_3-m_4$

$A_0 = m_1$ $\qquad$ $A_1 = s_7$ $\qquad$ $A_2 = m_2$ $\qquad$ $A_3 = s_8$

B4.
$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{pmatrix} = \begin{pmatrix} w^0 & w^0 & w^0 & w^0 & w^0 \\ w^0 & w^1 & w^2 & w^3 & w^4 \\ w^0 & w^2 & w^4 & w^1 & w^3 \\ w^0 & w^3 & w^1 & w^4 & w^2 \\ w^0 & w^4 & w^3 & w^2 & w^1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} \qquad w = e^{\frac{2\pi i}{5}}$$

Algorithm:

$s_1 = a_1+a_4 \qquad s_2 = a_1-a_4 \qquad s_3 = a_3+a_2 \qquad s_4 = a_3-a_2$

$s_5 = s_1+s_3 \qquad s_6 = s_1-s_3 \qquad s_7 = s_2+s_4 \qquad s_8 = s_5+a_0$

$m_0 = 1\cdot s_8 \qquad m_1 = \left(\frac{\cos u+\cos 2u}{2} - 1\right)\cdot s_5 \qquad m_2 = \left(\frac{\cos u-\cos 2u}{2}\right)\cdot s_6 \qquad u = \frac{2\pi}{5}$

$m_3 = i(\sin u+\sin 2u)\cdot s_2 \qquad m_4 = i \sin 2u\cdot s_7 \qquad m_5 = i(\sin u-\sin 2u)\cdot s_4$

$s_9 = m_0+m_1 \qquad s_{10} = s_9+m_2 \qquad s_{11} = s_9-m_2 \qquad s_{12} = m_3-m_4$

$s_{13} = m_4+m_5 \qquad s_{14} = s_{10}+s_{12} \qquad s_{15} = s_{10}-s_{12} \qquad s_{16} = s_{11}+s_{13}$

$s_{17} = s_{11}-s_{13}$

$A_0 = m_0 \qquad A_1 = s_{14} \qquad A_2 = s_{16} \qquad A_3 = s_{17} \qquad A_4 = s_{15}$

B5.
$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \end{pmatrix} = \begin{pmatrix} w^0 & w^0 & w^0 & w^0 & w^0 & w^0 & w^0 \\ w^0 & w^1 & w^2 & w^3 & w^4 & w^5 & w^6 \\ w^0 & w^2 & w^4 & w^6 & w^1 & w^3 & w^5 \\ w^0 & w^3 & w^6 & w^2 & w^5 & w^1 & w^4 \\ w^0 & w^4 & w^1 & w^5 & w^2 & w^6 & w^3 \\ w^0 & w^5 & w^3 & w^1 & w^6 & w^4 & w^2 \\ w^0 & w^6 & w^5 & w^4 & w^3 & w^2 & w^1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} \qquad w = e^{\frac{2\pi i}{7}}$$

Algorithm:

$s_1 = a_1+a_6 \qquad s_2 = a_1-a_6 \qquad s_3 = a_4+a_3 \qquad s_4 = a_4-a_3$

$s_5 = a_2-a_5 \qquad s_6 = a_2-a_5 \qquad s_7 = s_1+s_3 \qquad s_8 = s_7+s_5$

$s_9 = s_8+a_0 \qquad s_{10} = s_1-s_3 \qquad s_{11} = s_3-s_5 \qquad s_{12} = s_5-s_1$

$s_{13} = s_2+s_4 \qquad s_{14} = s_{13}+s_6 \qquad s_{15} = s_2-s_4 \qquad s_{16} = s_4-s_6$

$s_{17} = s_6-s_2$

$m_0 = 1\cdot s_9 \qquad m_1 = \left(\frac{\cos u+\cos 2u+\cos 3u}{3} - 1\right)\cdot s_8 \qquad u = \frac{2\pi i}{7}$

$m_2 = \left(\frac{2\cos u-\cos 2u-\cos 3u}{3}\right)\cdot s_{10} \qquad m_3 = \left(\frac{\cos u-2\cos 2u+\cos 3u}{3}\right)\cdot s_{11}$

$m_4 = \left(\frac{\cos u+\cos 2u-2\cos 3u}{3}\right)\cdot s_{12} \qquad m_5 = i\left(\frac{\sin u+\sin 2u-\sin 3u}{3}\right)\cdot s_{14}$

$$m_6 = i\left(\frac{2\sin u-\sin 2u+\sin 3u}{3}\right)\cdot s_{15} \qquad m_7 = i\left(\frac{\sin u-2\sin 2u-\sin 3u}{3}\right)\cdot s_{16}$$

$$m_8 = i\left(\frac{\sin u+\sin 2u+2\sin 3u}{3}\right)\cdot s_{17}$$

$$s_{18} = m_0+m_1 \qquad s_{19} = s_{18}+m_2 \qquad s_{20} = s_{19}+m_3 \qquad s_{21} = s_{18}-m_2$$

$$s_{22} = s_{21}-m_4 \qquad s_{23} = s_{18}-m_3 \qquad s_{24} = s_{23}+m_4 \qquad s_{25} = m_5+m_6$$

$$s_{26} = s_{25}+m_7 \qquad s_{27} = m_5-m_6 \qquad s_{28} = s_{27}-m_8 \qquad s_{29} = m_5-m_7$$

$$s_{30} = s_{29}+m_8 \qquad s_{31} = s_{20}+s_{26} \qquad s_{32} = s_{20}-s_{26} \qquad s_{33} = s_{22}+s_{28}$$

$$s_{34} = s_{22}-s_{28} \qquad s_{35} = s_{24}+s_{30} \qquad s_{36} = s_{24}-s_{30}$$

$$A_0 = m_0 \qquad A_1 = s_{31} \qquad A_2 = s_{33} \qquad A_3 = s_{36}$$

$$A_4 = s_{35} \qquad A_5 = s_{34} \qquad A_6 = s_{32}$$

B6.

$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \end{pmatrix} = \begin{pmatrix} w^0 & w^0 & w^0 & w^0 & w^0 & w^0 & w^0 & w^0 \\ w^0 & w^1 & w^2 & w^3 & w^4 & w^5 & w^6 & w^7 \\ w^0 & w^2 & w^4 & w^6 & w^0 & w^2 & w^4 & w^6 \\ w^0 & w^3 & w^6 & w^1 & w^4 & w^7 & w^2 & w^5 \\ w^0 & w^4 & w^0 & w^4 & w^0 & w^4 & w^0 & w^4 \\ w^0 & w^5 & w^2 & w^7 & w^4 & w^1 & w^6 & w^3 \\ w^0 & w^6 & w^4 & w^2 & w^0 & w^6 & w^4 & w^2 \\ w^0 & w^7 & w^6 & w^5 & w^4 & w^3 & w^2 & w^1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} \qquad w = e^{\frac{2\pi i}{8}}$$

Algorithm.

$$s_1 = a_0+a_4 \qquad s_2 = a_0-a_4 \qquad s_3 = a_2+a_6 \qquad s_4 = a_2-a_6$$

$$s_5 = a_1+a_5 \qquad s_6 = a_1-a_5 \qquad s_7 = a_3+a_7 \qquad s_8 = a_3-a_7$$

$$s_9 = s_1+s_3 \qquad s_{10} = s_1-s_3 \qquad s_{11} = s_5+s_7 \qquad s_{12} = s_5-s_7$$

$$s_{13} = s_9+s_{11} \qquad s_{14} = s_9-s_{11} \qquad s_{15} = s_6+s_8 \qquad s_{16} = s_6-s_8$$

$$m_1 = 1\cdot s_{13} \qquad m_2 = 1\cdot s_{14} \qquad m_3 = 1\cdot s_{10} \qquad m_4 = i \sin 2u\cdot s_{12} \qquad u = \frac{2\pi}{8}$$

$$m_5 = 1\cdot s_2 \qquad m_6 = i\cdot\sin 2u\cdot s_4 \qquad m_7 = i \sin u\cdot s_{15} \qquad m_8 = \cos u\cdot s_{16}$$

$$s_{17} = m_3+m_4 \qquad s_{18} = m_3-m_4 \qquad s_{19} = m_5+m_8 \qquad s_{20} = m_5-m_8$$

$$s_{21} = m_6+m_7 \qquad s_{22} = m_6-m_7 \qquad s_{23} = s_{19}+s_{21} \qquad s_{24} = s_{19}-s_{21}$$

$$s_{25} = s_{20}+s_{22} \qquad s_{26} = s_{20}-s_{22}$$

$A_0 = m_1$ $A_1 = s_{23}$ $A_2 = s_{17}$ $A_3 = s_{26}$

$A_4 = m_2$ $A_5 = s_{25}$ $A_6 = s_{18}$ $A_7 = s_{24}$

B7.

$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \\ A_5 \\ A_6 \\ A_7 \\ A_8 \end{pmatrix} = \begin{pmatrix} w^0 & w^0 & w^0 & w^0 & w^0 & w^0 & w^0 & w^0 & w^0 \\ w^0 & w^1 & w^2 & w^3 & w^4 & w^5 & w^6 & w^7 & w^8 \\ w^0 & w^2 & w^4 & w^6 & w^8 & w^1 & w^3 & w^5 & w^7 \\ w^0 & w^3 & w^6 & w^0 & w^3 & w^6 & w^0 & w^3 & w^6 \\ w^0 & w^4 & w^8 & w^3 & w^7 & w^2 & w^6 & w^1 & w^5 \\ w^0 & w^5 & w^1 & w^6 & w^2 & w^7 & w^3 & w^8 & w^4 \\ w^0 & w^6 & w^3 & w^0 & w^6 & w^3 & w^0 & w^6 & w^3 \\ w^0 & w^7 & w^5 & w^3 & w^1 & w^8 & w^6 & w^4 & w^2 \\ w^0 & w^8 & w^7 & w^6 & w^5 & w^4 & w^3 & w^2 & w^1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix}$$

Algorithm:

$s_1 = a_1+a_8$ $s_2 = a_1-a_8$ $s_3 = a_7+a_2$ $s_4 = a_7-a_2$

$s_5 = a_3+a_6$ $s_6 = a_3-a_6$ $s_7 = a_4+a_5$ $s_8 = a_4-a_5$

$s_9 = s_1+s_3$ $s_{10} = s_9+s_7$ $s_{11} = s_{10}+s_5$ $s_{12} = s_{11}+a_0$

$s_{13} = s_2+s_4$ $s_{14} = s_{13}+s_8$ $s_{15} = s_1-s_3$ $s_{16} = s_3-s_7$

$s_{17} = s_7-s_1$ $s_{18} = s_2-s_4$ $s_{19} = s_4-s_8$ $s_{20} = s_8-s_2$

$m_0 = 1 \cdot s_{12}$ $m_1 = (-\frac{1}{2}) \cdot s_{10}$ $m_2 = i \sin 3u \cdot s_{14}$ $u = \frac{2\pi}{9}$

$m_3 = (\cos 3u - 1) \cdot s_5$ $m_4 = i \sin 3u \cdot s_6$ $m_5 = (\frac{2\cos u - \cos 2u - \cos 4u}{3}) \cdot s_{15}$

$m_6 = (\frac{\cos u + \cos 2u - 2\cos 4u}{3}) \cdot s_{16}$ $m_7 = (\frac{\cos u - 2\cos 2u + \cos 4u}{3}) \cdot s_{17}$

$m_8 = i(\frac{2\sin u + \sin 2u - \sin 4u}{3}) \cdot s_{18}$ $m_9 = i(\frac{\sin u - \sin 2u - 2\sin 4u}{3}) \cdot s_{19}$

$m_{10} = i(\frac{\sin u + 2\sin 2u + \sin 4u}{3}) \cdot s_{20}$

$s_{21} = m_1+m_1$ $s_{22} = s_{20}+m_1$ $s_{23} = m_0+s_{22}$ $s_{24} = s_{23}\ m_2$

$s_{25} = s_{23}-m_2$ $s_{26} = m_0+m_3$ $s_{27} = s_{26}+s_{21}$ $s_{28} = s_{27}+m_5$

$s_{29} = s_{28}+m_6$ $s_{30} = s_{27}-m_6$ $s_{31} = s_{30}+m_7$ $s_{32} = s_{27}-m_5$

$s_{33} = s_{32}-m_7$ $s_{34} = m_4+m_8$ $s_{35} = s_{34}+m_9$ $s_{36} = m_4-m_9$

$s_{37} = s_{36}+m_{10}$ $s_{38} = m_4-m_8$ $s_{39} = s_{38}-m_{10}$ $s_{40} = s_{29}+s_{35}$

$s_{41} = s_{29}-s_{35}$ $s_{42} = s_{31}+s_{37}$ $s_{43} = s_{31}-s_{37}$ $s_{44} = s_{33}+s_{39}$

$s_{45} = s_{33}-s_{39}$

$A_0 = m_0$ $A_1 = s_{40}$ $A_2 = s_{43}$ $A_3 = s_{24}$ $A_4 = s_{43}$

$A_5 = s_{45}$ $A_6 = s_{23}$ $A_7 = s_{42}$ $A_8 = s_{41}$

B8. $A_k = \sum_{j=0}^{15} w^{kj} a_j$ $k=0,1,\ldots,15$ $w = e^{\frac{2\pi i}{16}}$.

Algorithm.

$s_1 = a_0+a_8$ $s_2 = a_0-a_8$ $s_3 = a_4+a_{12}$ $s_4 = a_4-a_{12}$

$s_5 = a_2+a_{10}$ $s_6 = a_2-a_{10}$ $s_7 = a_6+a_{14}$ $s_8 = a_6-a_{14}$

$s_9 = a_1+a_9$ $s_{10} = a_1-a_9$ $s_{11} = a_5+a_{13}$ $s_{12} = a_5-a_{13}$

$s_{13} = a_3+a_{11}$ $s_{14} = a_3-a_{11}$ $s_{15} = a_7+a_{15}$ $s_{16} = a_7-a_{15}$

$s_{17} = s_1+s_3$ $s_{18} = s_1-s_3$ $s_{19} = s_5+s_7$ $s_{20} = s_5-s_7$

$s_{21} = s_9+s_{11}$ $s_{22} = s_9-s_{11}$ $s_{23} = s_{13}+s_{15}$ $s_{24} = s_{13}-s_{15}$

$s_{25} = s_{17}+s_{19}$ $s_{26} = s_{17}-s_{19}$ $s_{27} = s_{21}+s_{23}$ $s_{28} = s_{21}-s_{23}$

$s_{29} = s_{25}+s_{27}$ $s_{30} = s_{25}-s_{27}$ $s_{31} = s_{22}+s_{24}$ $s_{32} = s_{22}-s_{24}$

$s_{33} = s_6+s_8$ $s_{34} = s_6-s_8$ $s_{35} = s_{10}+s_{16}$ $s_{36} = s_{10}-s_{16}$

$s_{37} = s_{12}+s_{14}$ $s_{38} = s_{12}-s_{14}$ $s_{39} = s_{35}+s_{37}$ $s_{40} = s_{36}+s_{38}$

$m_1 = 1\cdot s_{29}$ $m_2 = 1\cdot s_{30}$ $m_3 = 1\cdot s_{26}$ $m_4 = i \sin 4u\cdot s_{28}$ $u = \frac{2\pi}{16}$

$m_5 = 1\cdot s_{18}$ $m_6 = i \sin 4u\cdot s_{20}$ $m_7 = i \sin 2u\cdot s_{31}$ $m_8 = \cos 2u\cdot s_{32}$

$m_9 = 1\cdot$ 2 $m_{10} = i \sin 4u\cdot$ 4 $m_{11} = i \sin 2u\cdot$ 33 $m_{12} = \cos 2u\cdot s_{34}$

$m_{13} = i \sin 3u\cdot s_{39}$ $m_{14} = i(\sin u - \sin 3u)\cdot s_{35}$ $m_{15} = i(\sin u + \sin 3u)\cdot s_{37}$

$m_{16} = \cos 3u\cdot s_{40}$ $m_{17} = (\cos u + \cos 3u)\cdot s_{36}$ $m_{18} = (\cos 3u - \cos u)\cdot s_{38}$

$s_{41} = m_3 + m_4$ $s_{42} = m_3 - m_4$ $s_{43} = m_5 + m_7$ $s_{44} = m_5 - m_7$

$s_{45} = m_6 + m_8$ $s_{46} = m_6 - m_8$ $s_{47} = s_{43} + s_{45}$ $s_{48} = s_{43} - s_{45}$

$s_{49} = s_{44} + s_{46}$ $s_{50} = s_{44} - s_{46}$ $s_{51} = m_9 + m_{12}$ $s_{52} = m_9 - m_{12}$

$s_{53} = m_{10} + m_{11}$ $s_{54} = m_{10} - m_{11}$ $s_{55} = m_{13} + m_{14}$ $s_{56} = m_{13} - m_{15}$

$s_{57} = m_{17} - m_{16}$ $s_{58} = m_{18} - m_{16}$ $s_{59} = s_{51} + s_{55}$ $s_{60} = s_{51} - s_{55}$

$s_{61} = s_{52} + s_{56}$ $s_{62} = s_{52} - s_{56}$ $s_{63} = s_{53} + s_{57}$ $s_{64} = s_{53} - s_{57}$

$s_{65} = s_{54} + s_{58}$ $s_{66} = s_{54} - s_{58}$ $s_{67} = s_{59} + s_{63}$ $s_{68} = s_{59} - s_{63}$

$$s_{69} = s_{60} + s_{64} \qquad s_{70} = s_{60} - s_{64} \qquad s_{71} = s_{61} + s_{65} \qquad s_{72} = s_{61} - s_{65}$$

$$s_{73} = s_{62} + s_{66} \qquad s_{74} = s_{62} - s_{66}$$

$$A_0 = m_1 \qquad A_1 = s_{67} \qquad A_2 = s_{47} \qquad A_3 = s_{72} \qquad A_4 = s_{41} \qquad A_5 = s_{71}$$

$$A_6 = s_{48} \qquad A_7 = s_{68} \qquad A_8 = m_2 \qquad A_9 = s_{69} \qquad A_{10} = s_{49} \qquad A_{11} = s_{74}$$

$$A_{12} = s_{42} \qquad A_{13} = s_{73} \qquad A_{14} = s_{50} \qquad A_{15} = s_{70}$$

Appendix C

C1.

$$\begin{pmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_2 & x_3 & x_4 & x_1 \\ x_3 & x_4 & x_1 & x_2 \\ x_4 & x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} .$$

Algorithm:

$$s_1 = y_1 + y_3 \qquad s_2 = y_1 - y_3 \qquad s_3 = y_4 + y_2 \qquad s_4 = y_4 - y_2$$

$$s_5 = s_1 + s_3 \qquad s_6 = s_1 - s_3 \qquad s_7 = s_2 + I \cdot s_4 \qquad s_8 = s_2 - I \cdot s_4$$

$$m_1 = \frac{x_1+x_2+x_3+x_4}{4} \cdot s_5 \qquad m_2 = \frac{x_1-x_2+x_3-x_4}{4} \cdot s_6$$

$$m_3 = \frac{(x_1-x_3)+ I(x_2-x_4)}{4} \cdot s_7 \qquad m_4 = \frac{(x_1-x_3) - I(x_2-x_4)}{4} \cdot s_8$$

$$s_9 = m_1 + m_2 \qquad s_{10} = m_1 - m_2 \qquad s_{11} = m_4 + m_3 \qquad s_{12} = m_4 - m_3$$

$$s_{13} = s_9 + s_{11} \qquad s_{14} = s_9 - s_{11} \qquad s_{15} = s_{10} + I \cdot s_{12} \qquad s_{16} = s_{10} - I \cdot s_{12}$$

$$\psi_1 = s_{13} \qquad \psi_2 = s_{15} \qquad \psi_3 = s_{14} \qquad \psi_4 = s_{16}$$

C2.

$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{pmatrix} = \begin{pmatrix} w^0 & w^0 & w^0 & w^0 & w^0 \\ w^0 & w^1 & w^2 & w^3 & w^4 \\ w^0 & w^2 & w^4 & w^1 & w^3 \\ w^0 & w^3 & w^1 & w^4 & w^2 \\ w^0 & w^4 & w^3 & w^2 & w^1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} .$$

Algorithm:

$$s_1 = a_1 + a_4 \qquad s_2 = a_1 - a_4 \qquad s_3 = a_3 + a_2 \qquad s_4 = a_3 - a_2$$

$$s_5 = s_1 + s_3 \qquad s_6 = s_1 - s_3 \qquad s_7 = s_2 + I \cdot s_4 \qquad s_8 = s_2 - I \cdot s_4$$

$$s_9 = a_0 + s_5$$

$$m_0 = 1 \cdot s_9 \qquad m_1 = \left(\frac{\cos u + \cos 2u}{2} - 1\right) \cdot s_5 \qquad u = \frac{2\pi}{5}$$

$$m_2 = \frac{\cos u - \cos 2u}{2} \cdot s_6 \qquad m_3 = \frac{i(\sin u + I \sin 2u)}{2} \cdot s_7$$

$$m_5 = \frac{i(\sin u - I \sin 2u)}{2} \cdot s_8$$

$$s_{10} = m_0 + m_1 \qquad s_{11} = s_{10} + m_2 \qquad s_{12} = s_{10} - m_2 \qquad s_{13} = m_4 + m_3$$

$$s_{14} = m_4 - m_3 \qquad s_{15} = s_{11} + s_{13} \qquad s_{16} = s_{11} - s_{13} \qquad s_{17} = s_{12} + I \cdot s_{14}$$

$$s_{18} = s_{12} - I \cdot s_{14}$$

$$A_0 = m_0 \qquad A_1 = s_{15} \qquad A_2 = s_{17} \qquad A_3 = s_{18} \qquad A_4 = s_{16}$$

Mathematical Science Department
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

1. S. WINOGRAD, "Some bilinear forms whose multiplicative complexity depends on the field of constants," to be published in *Mathematical Systems Theory*, Vol. 10.

2. J. W. COOLEY & J. W. TUKEY, "An algorithm for the machine calculation of complex Fourier series," *Math. Comp.*, v. 19, 1965, pp. 297–301.

3. C. M. FIDUCCIA & Y. ZALCSTEIN, *Algebras Having Linear Multiplicative Complexities*, Technical Report 46, Dept. of Computer Science, State University of New York, Stony Brook, August 1975.

4. A. L. TOOM, "The complexity of a scheme of functional elements simulating the multiplication of integers," *Dokl. Akad. Nauk SSSR*, v. 150, 1963, pp. 496–498 = *Soviet Math. Dokl.*, v. 4, 1963, pp. 714–716.

5. C. M. RADER, "Discrete Fourier transforms when the number of data samples is prime," *Proc. IEEE*, v. 5, no. 6, June 1968, pp. 1107–1108.

6. I. J. GOOD, "The interaction of algorithm and practical Fourier series," *J. Roy. Statist. Soc. Ser. B*, v. 20, 1958, pp. 361–372; Addendum, v. 22, 1960, pp. 372–375.