General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.
- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.
- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.
- This document is paginated as submitted by the original source.
- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

FAR-FIELD RADIATION PATTERNS OF APERTURE ANTENNAS BY THE WINOGRAD FOURIER TRANSFORM ALGORITHM

Rodney Heisler

School of Engineering

Walla Walla College

(NASA-CR-159911) FAR-FIELD RADIATION PATTERNS OF APERTURE ANTENNAS BY THE WINOGRAD FOURIER TRANSFORM ALGORITHM (Walla Walla College, College Place) 46 p HC A03/MF A01 CSCL 20N G3/32

N79-16171

Unclas 43309

NAS 5-25063

October 1978

GODDARD SPACE FLIGHT CENTER

GREENBELT, MARYLAND 20771

FAR-FIELD RADIATION PATTERNS OF APERTURE ANTENNAS
BY THE WINOGRAD FOURIER TRANSFORM ALGORITHM

Rodney Heisler
School of Engineering
Walla Walla College

ABSTRACT

The far-field radiation pattern of an antenna may be determined as the fast Fourier transform (FFT) of the aperture distribution. When the antenna is electrically large and a detailed pattern in two dimensions is required, computer run-times exceeding an hour may result. A more time-efficient algorithm for computing the discrete Fourier transform (DFT) may result in a more effective analysis and design process. Significant savings in cpu time will improve the computer turn-around time and circumvent the need to resort to weekend runs.

A FORTRAN program to calculate the DFT using the Winograd Fourier transform algorithm was adapted to the IBM 360/91 computer and extended to handle complex input data vectors up to length N = 5040. Transforms may be computed for any data length given by the product of four mutually prime numbers selected from the integers 16, 9, 8, 7, 5, 4, 3, and 2 (e.g., $N = 9 \cdot 8 \cdot 7 \cdot 5 = 2520$). The WFT was used to compute antenna patterns for cophase and linear phase gradient apertures. The results were essentially identical to those previously computed with a conventional radix-2 FFT.

Significant time savings were realized with the WFT program. Run-time comparisons were made between WFT lengths of 1008, 2520 and 5040 and FFT lengths of 1024, 2048, and 4096, respectively. A minimum 4.6 to 1 speed advantage was demonstrated over this range. On the basis of the WFT timings it was estimated that two-dimensional transforms would require about one minute, ten minutes, and 40 minutes for the 1008, 2520, and 5040 point transforms, respectively. This

is sufficient to make two-dimensional transforms up to $N = 2520$ feasible within reasonable computer run-time limitations.

CONTENTS

ILLUSTRATIONS

TABLES

FAR-FIELD RADIATION PATTERNS OF APERTURE ANTENNAS
BY THE WINOGRAD FOURIER TRANSFORM ALGORITHM

INTRODUCTION

The computation of far-field radiation patterns of aperture antennas using the Fourier transform is well documented in the literature (Ref. 1 and 2) and in current use. The fast Fourier transform (FFT) algorithm is used in this application because of its computational speed advantage over the conventional discrete Fourier transform (DFT). Even so, computer run times exceeding one hour are expected as analysis techniques are extended to two-dimensional problems. A significant reduction in cpu run time would allow more effective analysis and design as computer runs may not be so time consuming as to be relegated to weekend runs only. Furthermore, the financial savings may be significant.

Recent studies (Ref. 3 and 4) have explored the feasibility of using the fast Walsh transform (FWT) for this application. While time savings on the order of ten to one were reported, the FWT gave only marginally satisfactory results for real data and failed to produce the normal squint associated with a linear phase gradient on the aperture distribution. Additionally, there were serious difficulties in calibrating the sequency axis.

Other investigators (Ref. 5, 6, 7, 8, 9, and 10) have reported on a new algorithm for computing the DFT. The Winograd Fourier transform algorithm (WFT) requires substantially fewer multiplications than the FFT while the number of additions, for some cases, remains near FFT levels. The WFT may be used effectively on short data sequences where the number of elements is prime. For long transform lengths, however, the number of additions becomes excessive and a direct application of the WFT becomes impractical. In this case it is useful to employ a combination of "small-N" WFT algorithms with a multidimensional expansion to extend the range of application to data lengths in excess of several thousand.

FROM DISCRETE FOURIER TRANSFORM TO DISCRETE CONVOLUTION

The DFT of a data vector $x(n) = \left\{x(0), x(1), \cdots\cdot x(N-1)\right\}$ is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\left(\frac{2\pi}{N}\right)nk} \quad k = 0, 1, 2, \cdots N-1 \tag{1}$$

$$= \sum_{n=0}^{N-1} x(n) w^{nk}$$

where

$$w = e^{-j\left(\frac{2\pi}{N}\right)}$$

The DFT is to discrete-time signals what the Fourier transform

$$X(jw) = \int_0^{\infty} x(t) e^{-j\left(\frac{2\pi}{T}\right)nt}\, dt \tag{2}$$

is to continuous-time signals. The extension of equation (2) to the DFT of equation (1) can be viewed intuitively.

Equation (1) describes the generation of N equations from which the elements in the DFT $X(0), X(1), \cdots X(N-1)$ may be computed:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \cdot \\ \cdot \\ \cdot \\ X(N-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w^1 & w^2 & \cdots & w^{N-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(N-1)} \\ \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & & \\ 1 & w^{N-1} & w^{2(N-1)} & \cdots & w^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \cdot \\ \cdot \\ \cdot \\ x(N-1) \end{bmatrix} \tag{3}$$

Using the FFT algorithm to evaluate this matrix product results in a very substantial computational reduction over a direct calculation. The complexity of computation arises from the $(N-1)$ by $(N-1)$ lower right-hand section of the transform matrix:

$$\overline{X}(k) = \sum_{n=1}^{N-1} x(n)w^{kn} \quad k = 1, 2, \cdots N - 1 \tag{4}$$

From $\overline{X}(k)$ we can retrieve $X(k)$ by

$$X(0) = \sum_{n=0}^{N-1} x(n)$$
$$X(k) = x(0) + \overline{X}(k) \quad k = 1, 2, \cdots N-1 \tag{5}$$

Consider the case where $N = 5$. Equation (4) then becomes:

$$\begin{bmatrix} \overline{X}(1) \\ \overline{X}(2) \\ \overline{X}(3) \\ \overline{X}(4) \end{bmatrix} = \begin{bmatrix} w^1 & w^2 & w^3 & w^4 \\ w^2 & w^4 & w^1 & w^3 \\ w^3 & w^1 & w^4 & w^2 \\ w^4 & w^3 & w^2 & w^1 \end{bmatrix} \begin{bmatrix} x(1) \\ x(2) \\ x(3) \\ x(4) \end{bmatrix} \tag{6}$$

where the exponents of w are written modulo 5, i.e., $w^8 = w^5 \cdot w^3 = w^3$, as $w^5 = 1$. A simple permutation of rows and columns in (6) will demonstrate how the DFT process may be converted to cyclic convolution. First interchange the last two columns and then the last two rows:

$$\begin{bmatrix} \overline{X}(1) \\ \overline{X}(2) \\ \overline{X}(4) \\ \overline{X}(3) \end{bmatrix} = \begin{bmatrix} w^1 & w^2 & w^4 & w^3 \\ w^2 & w^4 & w^3 & w^1 \\ w^4 & w^3 & w^1 & w^2 \\ w^3 & w^1 & w^2 & w^4 \end{bmatrix} \begin{bmatrix} x(1) \\ x(2) \\ x(4) \\ x(3) \end{bmatrix} \tag{7}$$

Next reverse the order of the input data $x(2)$, $x(3)$, $x(4)$:

$$\begin{bmatrix} \overline{X}(1) \\ \overline{X}(2) \\ \overline{X}(4) \\ \overline{X}(3) \end{bmatrix} = \begin{bmatrix} w^1 & w^3 & w^4 & w^2 \\ w^2 & w^1 & w^3 & w^4 \\ w^4 & w^2 & w^1 & w^3 \\ w^3 & w^4 & w^2 & w^1 \end{bmatrix} \begin{bmatrix} x(1) \\ x(3) \\ x(4) \\ x(2) \end{bmatrix} \tag{8}$$

Careful examination reveals that the above matrix product conforms to the definition of discrete circular convolution:

$$y(k) = x(n) * h(n) = \sum_{n=0}^{N-1} x(n)h(k-n) \tag{9}$$

$$k = 1, 2, \cdots N$$

Convolution is more easily understood from a graphical description as the mathematical process of equation (9) may not be clear. Figure 1 demonstrates graphically the convolution of x(n) and h(n). Both x(n) and h(n) are depicted as periodic discrete data series of length four. For the case of $k = 0, h(-n)$ is seen to be the mirror image of h(n). $h(1 - n)$ is simply $h(-n)$ shifted right one sampling interval. y(1) may then be computed as the sum of products of x and h as shown. The other y(k) terms may be similarly found as demonstrated in the figure.

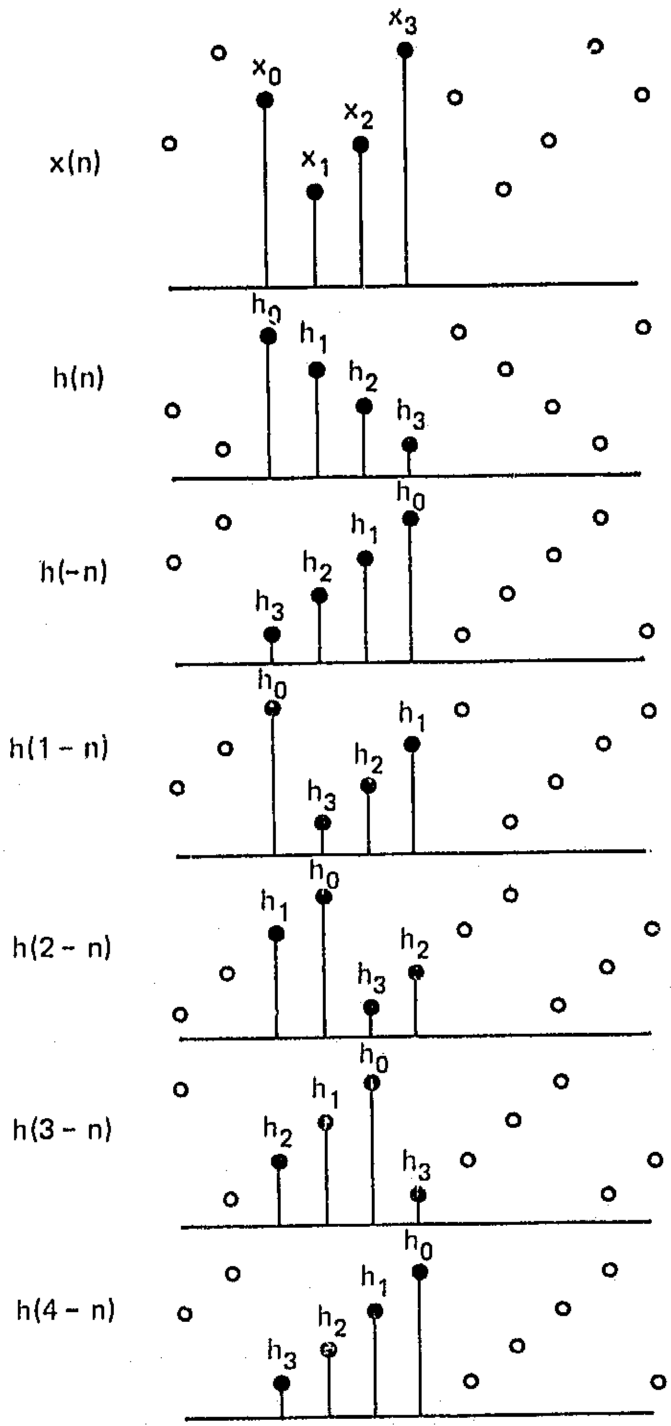
Equation (8) describes the convolution:

$$\left\{\overline{X}(1), \overline{X}(2), \overline{X}(4), \overline{X}(3)\right\} = \left\{x(1), x(3), x(4), x(2)\right\} * \left\{w^1, w^2, w^4, w^3\right\} \tag{10}$$

The graphical details of this convolution may be studied in Figure 2. Note that upon convolving the two data sets of equation (10), the elements $\overline{X}(1), \overline{X}(2), \overline{X}(4), \overline{X}(3)$ respectively are computed. Significantly, the elements $\overline{X}(k)$ of the DFT are now to be computed from a convolution operation. This transition was accomplished by a mapping of the matrix indices and is always possible for N equal to a prime or a prime power. For a mathematical description of the mapping process see Kolba and Parks (Ref. 9).

FAST DISCRETE CIRCULAR CONVOLUTION

Winograd (Ref. 6 and 7) has demonstrated an operational advantage to computing the DFT by changing to a discrete circular convolution operation. He presents an algorithm for performing short length cyclic convolution in a minimum number of multiplies. The concept employs polynomial multiplication modulo a third polynomial.



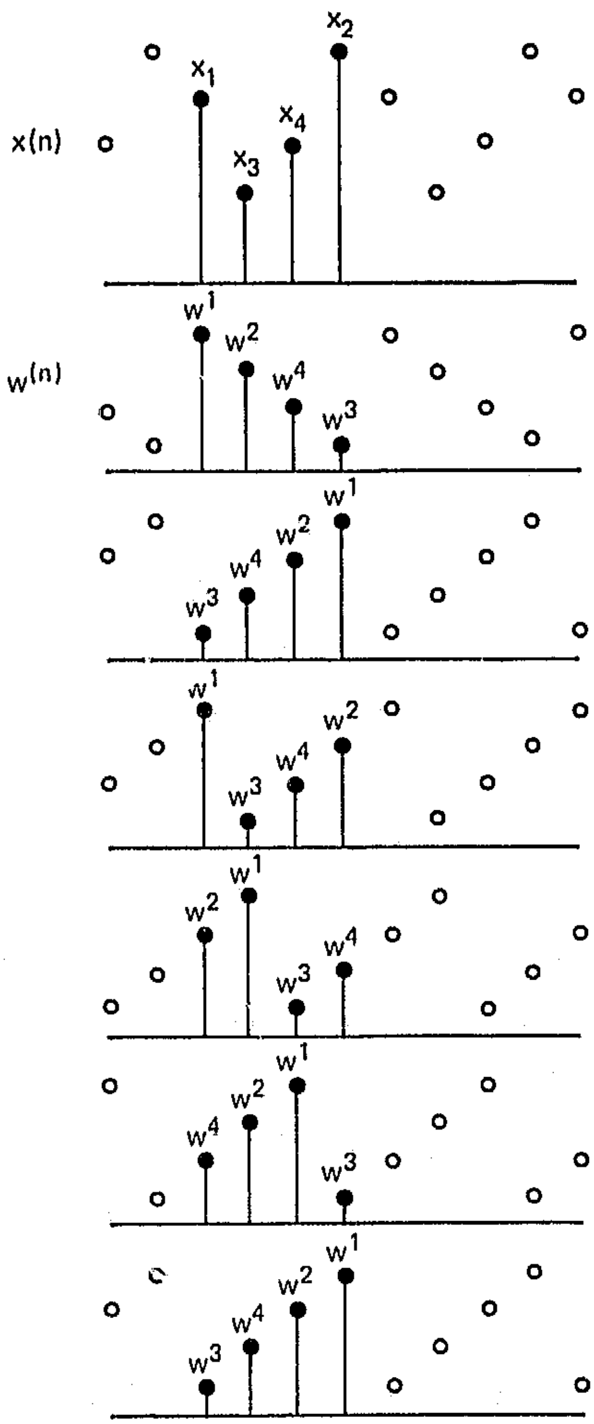
$$y(1) = h_0x_0 + h_3x_1 + h_2x_2 + h_1x_3$$

$$y(2) = h_1x_0 + h_0x_1 + h_3x_2 + h_2x_3$$

$$y(3) = h_2x_0 + h_1x_1 + h_0x_2 + h_3x_3$$

$$y(4) = h_3x_0 + h_2x_1 + h_1x_2 + h_0x_3$$

Figure 1. Graphical Convolution of the Periodic Data Sets x(n) and h(n)



$$\overline{X}(1) = w^1x(1) + w^3x(3) + w^4x(4) + w^2x(2)$$

$$\overline{X}(2) = w^2x(1) + w^1x(3) + w^3x(4) + w^4x(2)$$

$$\overline{X}(4) = w^4x(1) + w^2x(3) + w^1x(4) + w^3x(2)$$

$$\overline{X}(3) = w^3x(1) + w^4x(3) + w^2x(4) + w^1x(2)$$

Figure 2. Graphical Convolution of $\{\overline{X}(1), \overline{X}(2), \overline{X}(4), \overline{X}(3)\}$
$= \{x(1), x(3), x(4), x(2)\} * \{w^1, w^2, w^4, w^3\}$

The cyclic convolution $h(n) * x(n)$ can be evaluated from the N coefficients of:

$$Y(z) = H(z) \cdot X(z) \bmod (z^N - 1)$$

where

$$H(z) = \sum_{k=0}^{N-1} h_k z^k = h_0 + h_1 z^1 + h_2 z^2 + \ldots + h_{N-1} z^{N-1} \tag{11}$$

$$X(z) = \sum_{k=0}^{N-1} x_k z^k = x_0 + x_1 z^1 + x_2 z^2 + \ldots + x_{N-1} z^{N-1}$$

Winograd states that the minimum number of required multiplies is equal to $2N - K$ where K is the number of irreducible polynomials into which $z^N - 1$ may be factored, i.e.,

$$z^N - 1 = \prod_{i=1}^{K} Q_i(z) \tag{12}$$

To demonstrate this theorem consider the convolution $\{h_0, h_1, h_2, h_3\} * \{x_0, x_1, x_2, x_3\}$ as presented earlier in Figure 1.

$$\begin{aligned} y(1) &= x_0h_0 + x_1h_3 + x_2h_2 + x_3h_1 \\ y(2) &= x_0h_1 + x_1h_0 + x_2h_3 + x_3h_2 \\ y(3) &= x_0h_2 + x_1h_1 + x_2h_0 + x_3h_3 \\ y(4) &= x_0h_3 + x_1h_2 + x_2h_1 + x_3h_0 \end{aligned} \tag{13}$$

The y(k) may be evaluated from the polynomial multiplication:

$$\begin{aligned} Y_1(z) &= (x_0 + x_1 z + x_2 z^2 + x_3 z^3) \cdot (h_0 + h_1 z + h_2 z^2 + h_3 z^3) \\ &= (x_0h_0) + (x_0h_1 + x_1h_0)z + (x_0h_2 + x_1h_1 + x_2h_0)z^2 + (x_0h_3 + x_1h_2 + x_2h_1 + x_3h_0)z^3 \\ &\qquad + (x_1h_3 + x_2h_2 + x_3h_1)z^4 + (x_2h_3 + x_3h_2)z^5 + (x_3h_3)z^6 \\ &= y_0 + y_1 z + y_2 z^2 + y_3 z^3 + y_4 z^4 + y_5 z^5 + y_6 z^6 \end{aligned} \tag{14}$$

It is now required to find $Y(z) = Y_1(z) \bmod (z^4 - 1)$.

The notion of modulo arithmetic is from the concept of congruency. Two integers a and b are said to be congruent mod M if

$$a = b + kM \tag{15}$$

where k is an integer and M is the modulus. This may be written as:

$$a = b \bmod (M)$$

The integer b may be found as the remainder of the quotient a/M as may be easily demonstrated. From equation (15) we have

$$a - kM = b$$

and hence:

$$\begin{array}{r} k \\ M \;\overline{)\,a} \\ \underline{kM} \\ b \end{array}$$

gives b as a remainder.

Extending this concept to polynomials we compute $Y(z) = Y_1(z) \bmod (z^4 - 1)$ as the remainder in the synthetic division $Y_1(z)/(z^4 - 1)$. This results in

$$Y(z) = (y_0 + y_4) + (y_1 + y_5)z + (y_2 + y_6)z^2 + y_3 z^3 \tag{16}$$

Note that there are N = 4 coefficients resulting from

$$Y(z) = H(z) \cdot X(z) \bmod (z^4 - 1)$$

and that upon close observation they are found to be y(1), y(2), y(3), and y(4) of equation (13), i.e.,

$$y_0 + y_4 = x_0 h_0 + x_1 h_3 + x_2 h_2 + x_3 h_1 = y(1)$$
$$y_1 + y_5 = x_0 h_1 + x_1 h_0 + x_2 h_3 + x_3 h_2 = y(2)$$

$$y_2 + y_6 = x_0h_2 + x_1h_1 + x_2h_0 + x_3h_3 = y(3)$$
$$y_3 = x_0h_3 + x_1h_2 + x_2h_1 + x_3h_0 = y(4)$$

The above example has demonstrated the logistics of performing the multiplication of two polynomials modulo another polynomial and specifically how the convolution results are recovered. This process may be accomplished in a more computationally efficient manner using the polynomial version of the Chinese remainder theorem (Ref. 9). We may first evaluate H(z) and X(z) modulo the irreducible polynomial factors of $z^N - 1$, i.e., $Q_i(z)$ and then find Y(z) as

$$Y(z) = \left[\sum_{i=1}^{k} Y_i(z)S_i(z)\right] \mod (z^N - 1) \tag{17}$$

where

$$\begin{aligned} Y_i(z) &= H_i(z)X_i(z) && \mod (Q_i(z)) \\ H_i(z) &= H(z) && \mod (Q_i(z)) \\ X_i(z) &= X(z) && \mod (Q_i(z)) \\ S_i(z) &\triangleq 1 && \mod (Q_i(z)) \quad i = 1, 2, \cdots, K \end{aligned}$$

To demonstrate the operations described by equation (17) we may continue with the convolution example $\{h_0, h_1, h_2, h_3\} * \{x_0, x_1, x_2, x_3\}$ as presented earlier. For this case N = 4, hence

$$H(z) = h_0 + h_1z + h_2z^2 + h_3z^3$$
$$X(z) = x_0 + x_1z + x_2z^2 + x_3z^3$$

as before.

The irreducible real factors of $z^4 - 1$ are

$$(z^4 - 1) = (z + 1)(z - 1)(z^2 + 1)$$

Note that the number of irreducible factors K = 3, and hence the number of required multiplies to compute the convolution is 2N - K = 5.

The $X_i(z)$ and $H_i(z)$ polynomials are determined next. $X_1(z)$ is found by:

$$X_1(z) = x_0 + x_1 z + x_2 z^2 + x_3 z^3 \mod (z+1)$$

Computationally $X_1(z)$ is computed as the remainder in $X(z)/(z+1)$. Similarly $X_2(z)$ and $X_3(z)$ may be found and we have:

$$\begin{aligned} X_1(z) &= X(z) \mod (z+1) = x_0 - x_1 + x_2 - x_3 = x_0^1 \\ X_2(z) &= X(z) \mod (z-1) = x_0 + x_1 + x_2 + x_3 = x_0^2 \\ X_3(z) &= X(z) \mod (z^2+1) = (x_1 - x_3)z + (x_0 - x_2) = x_0^3 + x_1^3 z \end{aligned} \tag{18}$$

Since $H(z)$ has the same form as $X(z)$, the $H_i(z)$ polynomials will have the same form as the $X_i(z)$ polynomials:

$$\begin{aligned} H_1(z) &= H(z) \mod (z+1) = h_0 - h_1 + h_2 - h_3 = h_0^1 \\ H_2(z) &= H(z) \mod (z-1) = h_0 + h_1 + h_2 + h_3 = h_0^2 \\ H_3(z) &= H(z) \mod (z^2+1) = (h_1 - h_3)z + (h_0 - h_2) = h_0^3 + h_1^3 z \end{aligned} \tag{19}$$

We may now find the $Y_i(z)$ polynomials as

$$\begin{aligned} Y_1(z) &= H_1(z)X_1(z) \mod (z+1) = h_0^1 x_0^1 = y_0^1 \\ Y_2(z) &= H_2(z)X_2(z) \mod (z-1) = h_0^2 x_0^2 = y_0^2 \\ Y_3(z) &= H_3(z)X_3(z) \mod (z^2+1) \\ &= h_1^3 x_1^3 z^2 + (h_0^3 x_1^3 + h_1^3 x_0^3)z + h_0^3 x_0^3 \mod (z^2+1) \\ &= (h_0^3 x_1^3 + h_1^3 x_0^3)z + (h_0^3 x_0^3 - h_1^3 x_1^3) \\ &= y_0^3 + y_1^3 z \end{aligned} \tag{20}$$

The $Y_i(z)$ polynomials may be written as

$$
\begin{aligned}
Y_1(z) &= h_0^1 x_0^1 = y_0^1 \\
Y_2(z) &= h_0^2 x_0^2 = y_0^2 \\
Y_3(z) &= (h_0^3 x_0^3 - h_1^3 x_1^3) + [(h_0^3 - h_1^3)(x_1^3 - x_0^3) + h_0^3 x_0^3 + h_1^3 x_1^3]z \\
&= y_0^3 + y_1^3 z
\end{aligned} \tag{21}
$$

Next let

$$
\begin{aligned}
m_1 &= h_0^1 x_0^1 & m_4 &= h_0^3 x_0^3 \\
m_2 &= h_0^2 x_0^2 & m_5 &= h_1^3 x_1^3 \\
m_3 &= (h_0^3 - h_1^3)(x_1^3 - x_0^3)
\end{aligned} \tag{22}
$$

These are the five multiplies that will be required to complete the convolution in this example. The $Y_i(z)$ polynomials may be rewritten in the form

$$
\begin{aligned}
Y_1(z) &= m_1 \\
Y_2(z) &= m_2 \\
Y_3(z) &= (m_4 - m_5) + (m_3 + m_4 + m_5)z
\end{aligned} \tag{23}
$$

We need yet to express $Y(z)$ in terms of the $Y_i(z)$ factors as per equation (16):

$$
Y(z) = Y_1(z)S_1(z) + Y_2(z)S_2(z) + Y_3(z)S_3(z) \quad \mod (z^4 - 1)
$$

To do so, however, we must first determine the $S_i(z)$ polynomials. These are found in accordance with equation (16) to be:

$$
\begin{aligned}
S_1(z) &= -\tfrac{1}{4}(z^3 - z^2 + z - 1) \\
S_2(z) &= \tfrac{1}{4}(z^3 + z^2 + z + 1) \\
S_3(z) &= \tfrac{1}{4}(z^4 - 2z^2 + 1)
\end{aligned} \tag{24}
$$

These may be checked by verifying that $S_i(z) = 1 \mod (Q_i(z))$ as

$$-\tfrac{1}{4}(z^3 - z^2 + z - 1) = 1 \quad \mod (z + 1)$$

$$\tfrac{1}{4}(z^3 + z^2 + z + 1) = 1 \quad \mod (z - 1)$$

$$\tfrac{1}{4}(z^4 - 2z^2 + 1) = 1 \quad \mod (z^2 + 1)$$

Now $Y(z)$ is found to be:

$$\begin{aligned} Y(z) = {} & m_1[-\tfrac{1}{4}(z^3 - z^2 + z - 1)] \\ & + m_2[\tfrac{1}{4}(z^3 + z^2 + z + 1)] \\ & + [(m_4 - m_5) + (m_3 + m_4 + m_5)z][\tfrac{1}{4}(z^4 - 2z^2 + 1)] \quad \mod (z^4 - 1) \end{aligned} \tag{25}$$

$$\begin{aligned} 4Y(z) = {} & (m_1 + m_2 + m_4 - m_5) \\ & + (-m_1 + m_2 + m_3 + m_4 + m_5)z \\ & + (m_1 + m_2 - 2m_4 + 2m_5)z^2 \\ & + (-m_1 + m_2 - 2m_3 - 2m_4 - 2m_5)z^3 \\ & + (m_4 - m_5)z^4 \\ & + (m_3 + m_4 + m_5)z^5 \qquad \mod (z^4 - 1) \end{aligned} \tag{26}$$

$$\begin{aligned} 4Y(z) = {} & (m_1 + m_2 + 2m_4 - 2m_5) \\ & + (-m_1 + m_2 + 2m_3 + 2m_4 + 2m_5)z \\ & + (m_1 + m_2 - 2m_4 + 2m_5)z^2 \\ & + (-m_1 + m_2 - 2m_3 - 2m_4 - 2m_5)z^3 \end{aligned} \tag{27}$$

Recalling the method of determining one polynomial modulo another, equation (27) was found as the remainder in the synthetic division $4Y(z)/(z^4 - 1)$. As with equation (16), the coefficients of z are the elements of the convolution example, i.e.:

$$
\begin{aligned}
y(1) &= \tfrac{1}{4}(m_1 + m_2 + 2m_4 - 2m_5) &&= x_0h_0 + x_1h_3 + x_2h_2 + x_3h_1 \\
y(2) &= \tfrac{1}{4}(-m_1 + m_2 + 2m_3 + 2m_4 + 2m_5) &&= x_0h_1 + x_1h_0 + x_2h_3 + x_3h_2 \\
y(3) &= \tfrac{1}{4}(m_1 + m_2 - 2m_4 + 2m_5) &&= x_0h_2 + x_1h_1 + x_2h_0 + x_3h_3 \\
y(4) &= \tfrac{1}{4}(-m_1 + m_2 - 2m_3 - 2m_4 - 2m_5) &&= x_0h_3 + x_1h_2 + x_2h_1 + x_3h_0
\end{aligned}
\tag{28}
$$

This last result may be verified by performing the indicated multiplications and additions.

COMPUTING THE DISCRETE FOURIER TRANSFORM WITH FAST CONVOLUTION TECHNIQUES

The fast convolution algorithm may now be applied to the task of computing the discrete Fourier transform. This is done by performing the indicated convolution in equation (8) to find $\overline{X}(1)$, $\overline{X}(2)$, $\overline{X}(4)$, and $\overline{X}(3)$. In this application h_0, h_1, h_2, and h_3 are w^1, w^2, w^4, and w^3, respectively, as seen by comparing Figures 1 and 2. Similarly, x_0, x_1, x_2, and x_3 are $x(1)$, $x(3)$, $x(4)$, and $x(2)$, the input data vector.

The terms h_0^1, h_0^2, h_0^3, h_1^3, x_0^1, x_0^2, x_0^3 and x_1^3 of equations (18) and (19) may be found as:

$$
\begin{aligned}
h_0^1 &= w^1 - w^2 + w^4 - w^3 = 2.236 \\
h_0^2 &= w^1 + w^2 + w^4 + w^3 = 1.0 \\
h_0^3 &= w^1 - w^4 = -j1.902 \\
h_1^3 &= w^2 - w^3 = -j1.176 \\
x_0^1 &= x(1) - x(3) + x(4) - x(2) \\
x_0^2 &= x(1) + x(3) + x(4) + x(2) \\
x_0^3 &= x(1) - x(4) \\
x_1^3 &= x(3) - x(2)
\end{aligned}
$$

From these results the m_i terms in equation (22) may be computed and subsequently the convolution results of equation (28) found. This results in the intermediate transform values $\overline{X}(k)$. The DFT is then calculated from equation (5). The process is now complete and the results for an N = 5 computational algorithm are summarized in Table 1 [from Kolba and Parks (Ref. 9) and Winograd

Table 1

Computational Algorithm for an N = 5 DFT Using Fast Discrete Convolution

[from Kolba and Parks (Ref. 9) and Winograd (Ref. 7)]

$a_1 = x(1) + x(4)$	$a_5 = a_2 + a_4$
$a_2 = x(1) - x(4)$	$a_6 = a_1 - a_3$
$a_3 = x(2) + x(3)$	$a_7 = a_1 + a_3$
$a_4 = x(2) - x(3)$	$a_8 = x(0) + a_7$
$m_1 = 0.951\ a_5$	$c_1 = x(0) - m_5$
$m_2 = 1.539\ a_2$	$c_2 = c_1 + m_4$
$m_3 = 0.363\ a_4$	$c_3 = c_1 - m_4$
$m_4 = 0.559\ a_6$	$c_4 = m_1 - m_3$
$m_5 = \frac{1}{4} a_7$	$c_5 = m_2 - m_1$

$$X(0) = a_8$$
$$X(1) = c_2 - jc_4$$
$$X(2) = c_3 - jc_5$$
$$X(3) = c_3 + jc_5$$
$$X(4) = c_2 + jc_4$$

(Ref. 7)]. The number of calculations required is observed to be 17 additions and 5 multiplications (the multiplication by ¼ may be accomplished by two word shifts on some FORTRAN compilers). The x(n) input data vector may be complex in which case these are complex adds and multiplies.

Computational algorithms for other short length transforms may be found in Winograd (Ref. 7), Silverman (Ref. 8 and 10), and Kolba and Parks (Ref. 9). The number of computations required for these several short-length WFT algorithms is compared with radix-2 FFT requirements in Table 2. The computational advantage of the WFT is clearly seen.

Table 2

Number of Calculations for Short-Length WFT and FFT

N	WFT		FFT	
	Multiplies	Adds	Multiplies	Adds
2	0	2	1	2
3	2*	6		
4	0	8	4	8
5	5*	17		
7	8	36		
8	2	26	24	48
9	10*	49		
16	10	68	32	64

*The number of multiplies may be reduced by using word shifts.

For large N the WFT algorithm for fast convolution gets out of control. Furthermore, the number of additions becomes excessive and the WFT loses its computational advantage. Consequently it is necessary to use a combination of short-length transforms to realize the speed advantage of the WFT for transform lengths of practical importance.

LONG-LENGTH TRANSFORMS

Winograd Fourier transforms of practical lengths in the several thousands may be computed as a combination of short-length transforms. This is accomplished by converting an $N = M_1 M_2 \cdots M_\ell$ (where the M_i are mutually prime integers) length transform into ℓ shorter transforms of lengths M_i for $i = 1, 2, \cdots \ell$. This is equivalent to a mapping from one to ℓ dimensions.

The DFT of equation (1)

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

incorporates an index n which orders the input data vector and an index k ordering the transform output results. A mapping from one to two dimensions requires that each index map to two indices, i.e.,

$$n \rightarrow (n_1, n_2)$$
$$k \rightarrow (k_1, k_2)$$

With this mapping the DFT may be written as

$$X(k_1, k_2) = \sum_{n_1=0}^{M_1-1} \left(\sum_{n_2=0}^{M_2-1} x(n_1, n_2) W_{M_2}^{n_2 k_2} \right) W_{M_1}^{n_1 k_1} \qquad \begin{array}{l} k_1 = 0, 1, \cdots M_1 - 1 \\ k_2 = 0, 1, \cdots M_2 - 1 \end{array} \tag{29}$$

where

$$W_{M_1} = e^{-j(2\pi/M_1)}$$

$$W_{M_2} = e^{-j(2\pi/M_2)}$$

$$N = M_1 M_2$$

and M_1 and M_2 are relatively prime.

The two-dimensional DFT of equation (29) is found by first computing M_1 transforms of length M_2

$$y(n_1, k_2) = \sum_{n_2=0}^{M_2-1} x(n_1, n_2)\, W_{M_2}^{n_2 k_2}$$

and then M_2 transforms of length M_1

$$X(k_1, k_2) = \sum_{n_1=0}^{M_1-1} y(n_1, k_2)\, W_{M_1}^{n_1 k_1}$$

The above mapping follows the method of Kolba and Parks (Ref. 9) and is referred to as a prime factor FFT algorithm. Winograd (Ref. 6 and 7) presents another method of structuring short-length transforms to accomplish the same end. This technique is referred to as the nested algorithm.

From an implementation point of view the major consideration between the two algorithms is the amount of computational effort. The computational requirements of these algorithms for several values of N are compared in Table 3. In general, the nested algorithm requires fewer multiplies, while the prime factor algorithm requires substantially fewer adds. The total number of calculations is substantially fewer for the prime factored algorithm.

Table 3

Comparison of Prime Factor and Nested Algorithms

N	Factors	Prime Factor Algorithm			Nested Algorithm		
		Multiplies	Adds	Total	Multiplies	Adds	Total
252	9·7·4	1024	6344	7368	848	7128	7976
504	9·7·8	2300	13948	16248	1704	15516	17220
1260	9·5·7·4	7136	40288	47424	5168	50184	55352
2520	9·5·7·8	15532	86876	102408	10344	106667	117011

The particular computer available and the relative timings for floating-point multiplication and addition would dictate which algorithm would be more time efficient in any application. The computer used in this study was the IBM 360/91 at Goddard Space Flight Center. The 360/91 is a very fast system utilizing a high-speed "cashe-pipeline". Floating-point multiplication and addition require essentially the same execution time. Hence, the nested algorithm was judged faster and selected for application on this system.

APPLICATION TO APERTURE ANTENNAS

As discussed earlier, the far-field radiation pattern of an aperture antenna can be determined to a good approximation as the DFT of the aperture field distribution. In this application it appears desirable to replace the FFT, which is currently used, with the faster WFT algorithm. Perhaps the savings in cpu run time would permit extension of the analysis program to two-dimensional geometries with several thousand data elements along each axis. At this time a reasonable estimate of the maximum input data vector size is about five thousand. This would allow analysis of a one meter antenna at 180 GHz with half wavelength sampling.

A FORTRAN IV program to calculate the WFT was adapted to run on the IBM 360/91 computer and extended to handle data vectors up to 5040 points with double precision arithmetic (See Appendix). Transforms may be computed for any data length given by the product of four mutually prime numbers selected from the integers 2, 3, 4, 5, 7, 8, 9, and 16 (e.g., $N = 9 \times 8 \times 7 \times 5 = 2520$). As a basis for comparing the performance of this WFT program with the conventional FFT, a benchmark aperture distribution of special interest was selected. The distribution specified is sine on a pedestal with a 20 db edge taper and is representative of focal-point fed parabolic antennas exhibiting axial symmetry. This distribution is described by $0.0909 + 0.9091 \sin\left(\frac{m\pi}{N-1}\right)$, $m = 0, 1, 2, \cdots N-1$; where N is the number of samples in the aperture. The antenna diameter is specified as 20 feet and the wavelength as 0.44973 feet.

A 2520 and 5040 point WFT analysis were performed on this antenna and the results compared with a 4096 point FFT computation. Theoretically the WFT and FFT algorithms give exactly the same results so this comparison is intended as a verification of the correctness of the WFT program. Figure 3 presents the 5040 point WFT and 4096 point FFT results. The two patterns are essentially identical except for minor differences in the higher order lobes. This difference is on the order of half a db and is entirely the result of differences in precision in the two programs. The WFT was run in double precision (8 byte data length) and the FFT in quartic precision (16 byte data length).

A 2520 and 5040 point WFT are compared in Figure 4. The results are very nearly identical. The only significant difference is the amount of detail produced. Understandably, the 5040 point transform yields twice as much detail as the 2520 point transform.

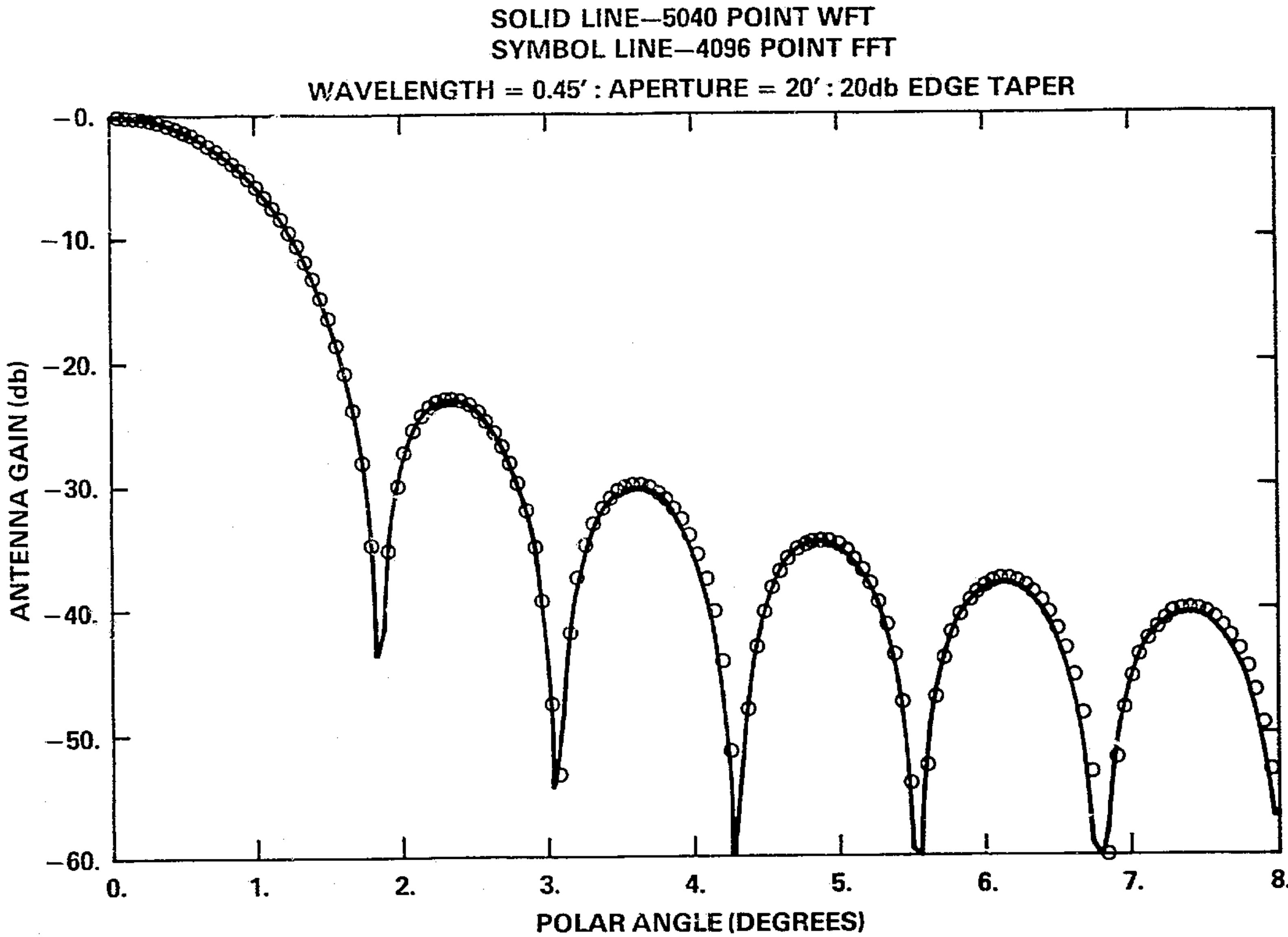


Figure 3. Far-field Radiation Pattern for Benchmark Aperture by 5040 Point WFT and 4096 Point FFT

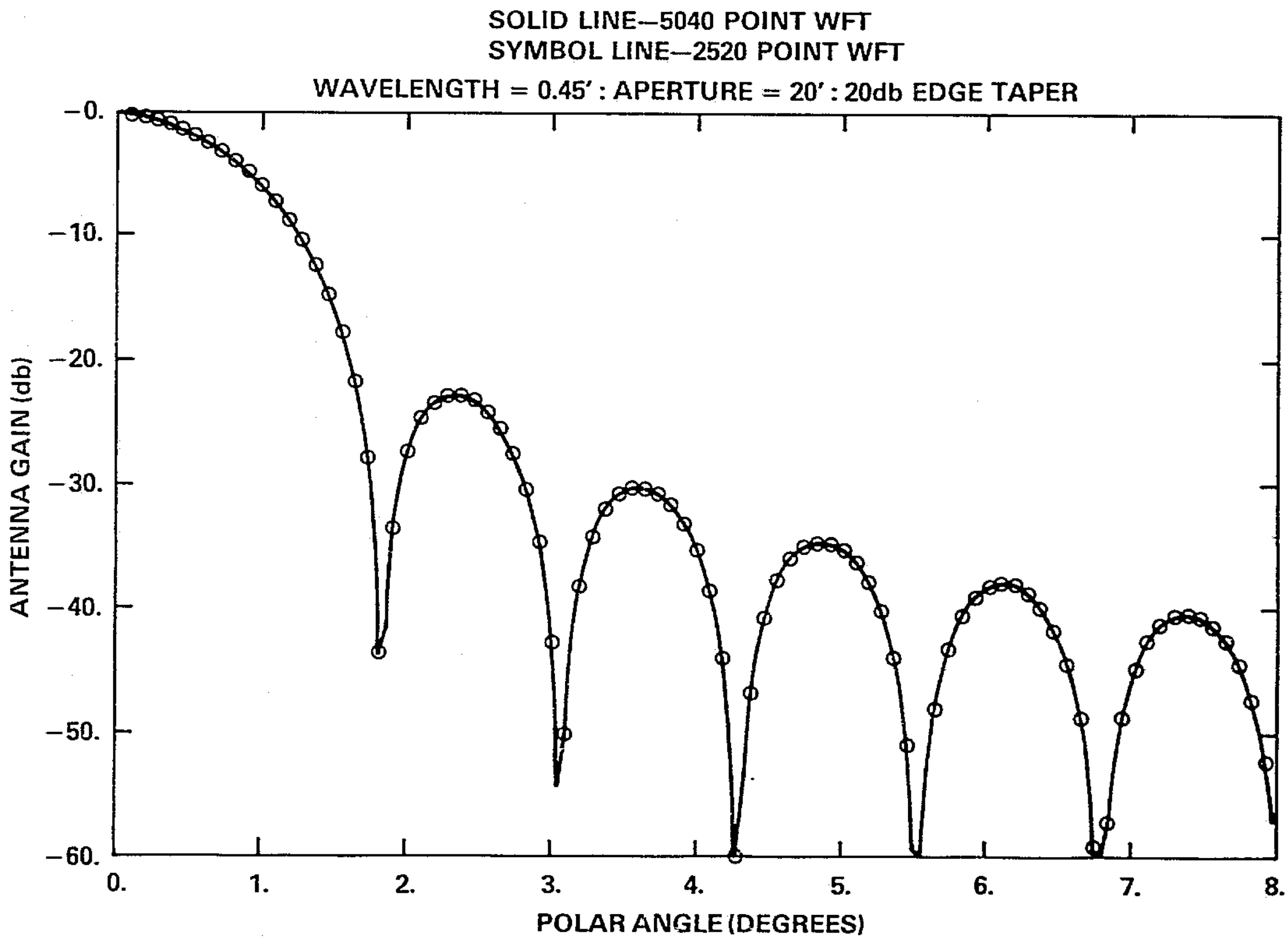


Figure 4. Far-field Radiation Pattern for Benchmark Aperture by 5040 Point and 2520 Point WFT

Figure 5 demonstrates the new algorithm's ability to handle complex input data. A 360° linear phase gradient was imposed on the benchmark aperture, resulting in the expected squint. The difference in results as computed by the FFT and WFT is again due to the greatly extended precision specified in the FFT program.

The interval timer available on the 360/91 system library at GSFC was used to time the WFT and FFT subroutines. For this comparison the FFT subroutine was rewritten in double precision math so that both transform algorithms would be judged on the same basis. The interval timer is represented as accurate to the nearest 0.01 second. The results of this test are presented in Table 4. WFT and FFT timings are compared for values of N that are as numerically close as possible given the different constraints on N for the two algorithms. The adjusted time ratio is determined by scaling the time ratio by the ratio of the number of data samples for the two algorithms.

Table 4

WFT and FFT Timing Results

Number of Data Samples		CPU Time (Sec)	Time Ratio	Adjusted Time Ratio
WFT	FFT			
1008		0.06	7.2	7.0
	1024	0.43		
	2048	0.90	3.8	4.6
2520		0.24		
	4096	1.90	4.0	4.9
5040		0.48		

Clearly the WFT shows a minimum 4.6 to 1 speed advantage over the FFT for the range of N considered. This is considered a significant improvement in speed. On the basis of these timings it is estimated that two-dimensional WFT computations would require about one minute, ten minutes, and 40 minutes for the 1008, 2520, and 5040 point transforms, respectively. Using the conventional

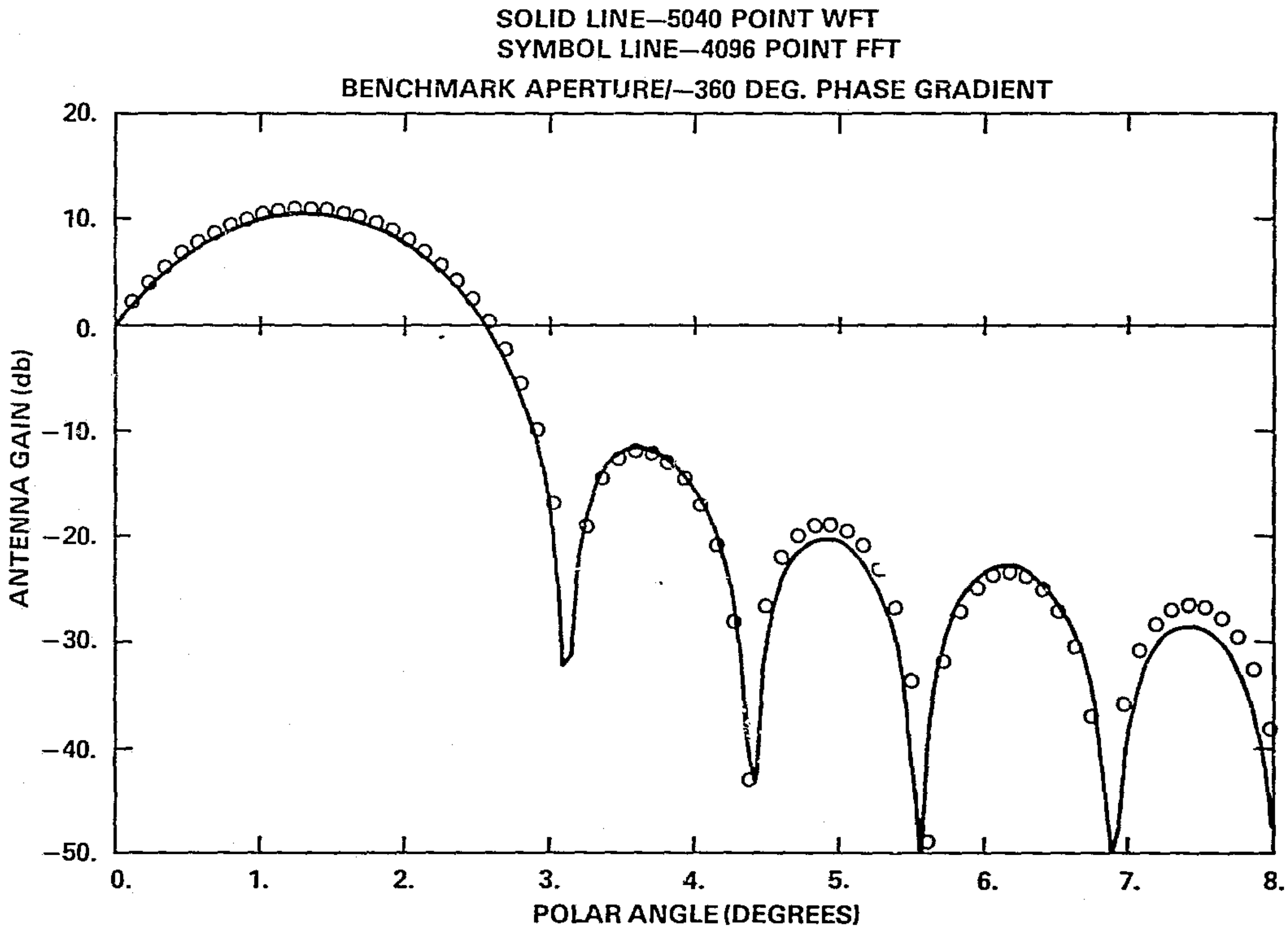


Figure 5. Far-field Radiation Pattern for Benchmark Aperture with Negative 360° Linear Phase Gradient by 5040 Point WFT and 4096 Point FFT

FFT program, approximately seven minutes, 31 minutes, and 130 minutes would be required for the 1024, 2048, and 4096 point transforms.

To achieve the large N capability of the WFT program it was necessary to nest the summations described by equation (29) three and four deep. The limits on these summations are the prime composites which make up N (e.g., $N = 2 \times 5 \times 7 \times 9 = 630$). The ordering of these factors greatly affects the run time for a given N. Minimum cpu time is realized when the composite transforms are ordered in some optimum way. Table 5 presents the optimum ordering for N = 504, 1008, 2520, and 5040. Any departure from this ordering will result in an increase in run time by as much as 100 percent.

Table 5

Optimum Ordering of Composite Transforms

for Large N WFT

N	Order
504	8•9•7•1
1008	7•16•9•1
2520	7•9•5•8
5040	16•5•7•9

OTHER ALGORITHMS

Other possibilities exist for fast DFT computations. Morris (Ref. 11) reports that a radix-4 FFT algorithm outperforms the WFT on some computer systems (DEC PDP-11/55 and the IBM-370/168). His work indicates that in this comparison the WFT execution times were about 20 to 40 percent longer.

Reed and Truong (Ref. 12) have suggested that replacing the convolution operation in equation (8) with a complex integer transform operation will result in fewer multiplications than either the FFT or the WFT. Several papers by these authors have demonstrated the feasibility of performing cyclic convolution by a combination of two N X N integer transforms, a multiplication of two

1 X N matrices, and an N X N inverse integer transform. The transforms and the inverse transform are performed with word shifts and integer additions. This results in a rather significant reduction in the total number of multiplications and converts them from floating-point to integer operations. However, the number of required additions is approximately tripled; consequently, the total number of mathematical operations is increased. On the basis of these observations it appears that the use of integer transforms offers no advantage over either the FFT or the WFT for systems like the IBM 360/91 where floating-point multiplication and addition require about the same time. A clear advantage can be demonstrated for micro-processor based systems for which the ratio of multiplication to addition times may be several orders of magnitude.

CONCLUSION

Use of the WFT algorithm in antenna analysis appears to be a very successful application. The radix-2 FFT as used in computing far-field radiation patterns of aperture antennas may be replaced by the WFT with no degradation in performance and with a considerable improvement in speed. Over the range of N from about 1000 to 5000 the WFT demonstrated a minimum 4.6 to 1 speed advantage over the presently used FFT. This is sufficient to make two-dimensional transforms up to N = 2520 feasible within reasonable computer run time limitations.

As new algorithms and transforms are introduced into the study of antennas, more powerful analysis and design techniques will become available to the design engineer.

ACKNOWLEDGMENTS

This work was supported in part by a NASA-ASEE Summer Faculty Fellowship and for this the author wishes to thank Professor Dan Fan of Howard University. Additional support came from the GSFC Contract No. NAS5-25063. The author also wishes to thank Mr. Richard Schmidt of GSFC for his support and encouragement.

REFERENCES

1. Bracewell, Ronald N.: The Fourier Transform and Its Applications, McGraw-Hill Book Co., 1978.
2. Collin, Robert E., and Zucker, Francis J.: Antenna Theory, Part One, McGraw-Hill Book Co., 1969.
3. Heisler, R.: Feasibility Study–Application of Walsh Transforms To Far-Field Antenna Pattern Computations, NASA Research Report X-811-77-215, GSFC, August 1977.
4. Ko, W.L., and Mittra, R.: On the Use of Walsh-Hadamard Transforms for the Computation of Radiation Patterns of Aperture Antennas, 1978 International IEEE/AP-S Symposium, USNC/URSI Spring Meeting, May 1978, University of Maryland.
5. Reed, I.S., Liu, K.Y., and Truong, T.K.: A New Fast Algorithm for Computing a Complex Number–Theoretic Transform, The Deep Space Progress Report 42-39, NASA-JPL, March and April 1977.
6. Winograd, Shmuel: On Computing the Discrete Fourier Transform, Proceedings National Academy of Science, USA, vol. 73, no. 4, pp. 1005-1006, April 1976.
7. Winograd, Shmuel: On Computing the Discrete Fourier Transform, IBM Research Report, RC-6291, November 1976.
8. Silverman, Harvey F.: An Introduction to Programming the Winograd Fourier Transform Algorithm (WFTA), IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-25, no. 2, pp. 152-165, April 1977.
9. Kolba, Dean P., and Parks, Thomas W.: A Prime Factor FFT Algorithm Using High-Speed Convolution, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-25, no. 4, pp. 281-294, August 1977.
10. Silverman, Harvey F.: Corrections and Addendum to "An Introduction to Programming the Winograd Fourier Transform Algorithm (WFTA), IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-26, no. 3, June 1978.

11. Morris, L. Robert: A Comparative Study of Time Efficient FFT and WFT Programs for General Purpose Computers, IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-26, no. 2, pp. 141-150, April 1978.

12. Reed, I.S., and Truong, T.K.: A Fast DFT Algorithm Using Complex Integer Transforms, Electronics Letters, vol. 14, no. 6, pp. 191-193, March 1978.

APPENDIX

FORTRAN PROGRAM FOR COMPUTING THE FAR-FIELD RADIATION PATTERN USING THE WFT ALGORITHM

This section presents two FORTRAN subroutines. The first (GOODFT) uses the WFT algorithm to compute the DFT for complex input data up to length N = 5040. The second (PATOUT) computes the far-field radiation pattern from the transform results. The WFT subroutine is an enhancement of a program contributed by Dean Kolba (Ref. 9) from Rice University. An N = 16 WFT algorithm was added to the original program structure to extend the maximum range from N = 2520 to N = 5040. In addition, the program was rewritten in double precision.

The length of the DFT, N, must be a product of no more than four mutually prime factors chosen from the integers 2, 3, 4, 5, 7, 8, 9, and 16. These factors are named M1, M2, M3, and M4. If not all four factors are used the unused factors are set equal to 1. The factors of one must be last in the sequence of M's in the program. The other I/O variables used in the subroutine are:

NFT = number of nonunity factors

KOUT = output indexing constant

= K1 + K2 + K3 + K4 (mod N)

where

$K1 = M2 \cdot M3 \cdot M4$ or $= 0$ for $M1 = 1$

$K2 = M1 \cdot M3 \cdot M4$ or $= 0$ for $M2 = 1$

$K3 = M1 \cdot M2 \cdot M4$ or $= 0$ for $M3 = 1$

$K4 = M1 \cdot M2 \cdot M3$ or $= 0$ for $M4 = 1$

XR(N) = real part of input data

XI(N) = imaginary part of input data

A(N) = real part of transform results

B(N) = imaginary part of transform results

To illustrate the above, consider the case $N = 5 \cdot 3 \cdot 2 \cdot 1 = 30$. The above input variables are:

N = 30	
NFT = 3	
M1 = 5	K1 = 6
M2 = 3	K2 = 10
M3 = 2	K3 = 15
M4 = 1	K4 = 0

KOUT = 31 (mod 30) = 1

The ordering of the M's is not significant with respect to the quality of the transform results, but does have a very important affect on the run-time for the subroutine. Table 5 presented the optimum ordering for the four cases N = 504, 1008, 2520, and 5040. Departure from these orders may increase the cpu timings by more than 100 percent.

The DFT is cyclic in nature and hence to compute the transform of a non-cyclic, finite data set $\{x(n)\}$ it is necessary to append a relatively large number of zeros to both ends of the aperture distribution. About 75 to 90 percent of the input data should be made up of these embedded zeros. This will establish an adequate ground plane about the aperture and reduce the effects of folding or aliasing.

The subroutine PATOUT requires as input the length of the DFT (N), the transform results from GOODFT (A&B), and the sampling interval (T) in wavelengths. To prevent aliasing $T \leqslant \lambda/2$. The output of the subroutine is the magnitude (in db) and the phase (in degrees) of the antenna gain as a function of polar angle starting at a line drawn broadside to the antenna and through its center. For this statement to be true it is necessary that the input aperture distribution to GOODFT be specified according to the same geometry. The aperture data must be input starting at the antenna midpoint and proceeding to the edge. The zeros are imbedded next, followed by the other half of the aperture data starting at the edge and ending at the center.

```
      SUBROUTINE GOODFT(XR,XI,N,M1,M2,M3,M4,NFT,KOUT,A,B)
C   THE SUBROUTINE GOODFT COMPUTES A LENGTH N DFT OF THE INPUT DATA WHICH IS IN
C   TWO VECTORS, XR THE REAL PART AND XI THE IMAGINAGY PART. BOTH XR AND XI ARE
C   LENGTH N VECTORS. THE LENGTH OF THE DFT, N, MUST BE A PRODUCT OF AT MOST
C   FOUR MUTUALLY PRIME FACTORS. THE POSSIBLE FACTORS ARE 2,3,4,5,7,8,9 AND 16.
C   THESE FACTORS ARE M1, M2, M3, AND M4. IF THE FOUR FACTORS ARE NOT ALL USED,
C   THE UNUSED FACTORS ARE SET EQUAL TO 1. FOR EXAMPLE WITH N=30, WE HAVE
C   M1=5, M2=3, M3=2, AND M4=1. THE FACTORS OF ONE MUST BE THE LAST OF THE M'S.
C   THE NUMBER OF NONUNITY FACTORS IS NFT. KOUT IS AN OUTPUT INDEXING CONSTANT
C   WHICH IS PRECOMPUTED. KOUT=(K1+K2+K3+K4)MOD N WHERE K1=M2*M3*M4,
C   K2=M1*M3*M4, K3=M1*M2*M4, K4=M1*M2*M3, AND K2=0 IF M2=1, K3=0 IF M3=1, AND
C   K4=0 IF M4=1. FOR EXAMPLE, N=30, K1=6, K2=10, K3=15, K4=0 AND KOUT=31MOD 30
C   =1. THE TRANSFORMED RESULTS ARE STORED IN TWO LENGTH N VECTORS, A AND B. A
C   CONTAINS THE REAL PART AND B CONTAINS THE IMAGINARY PART OF THE RESULTS.
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION XR(5040),XI(5040),UR(16),UI(16),I(16),A(5040),B(5040)
      REAL*8  MR1,MR2,MR3,MR4,MR5,MR6,MR7,MR8,MR9,MR10,MR11,MR12,MR13
      REAL*8  MR14,MR15,MR16,MR17,MR18,MR19,MR20,MR21,MR22,MR23,MR24
      REAL*8  MR25,MR26,MR27,MR28,MR29,MR30,MI1,MI2,MI3,MI4,MI5,MI6,MI7
      REAL*8  MI8,MI9,MI10,MI11,MI12,MI13,MI14,MI15,MI16,MI17,MI18,MI19
      REAL*8  MI20,MI21,MI22,MI23,MI24,MI25,MI26,MI27,MI28,MI29,MI30
      NF=NFT
C   ORDER FACTORS FOR TRANSFORMS OF LENGTH M1
      MM1=M1
      MM2=M2
      MM3=M3
      MM4=M4
      GO TO 20
   10 GO TO(12,13,14),NF
C   ORDER FACTORS FOR TRANSFORMS OF LENGTH M2
   12 MM1=M2
      MM2=M1
      MM3=M3
      MM4=M4
      GO TO 20
C   ORDER FACTORS FOR TRANSFORMS OF LENGTH M3
   13 MM1=M3
      MM2=M1
      MM3=M2
      MM4=M4
      GO TO 20
C   ORDER FACTORS FOR TRANSFORMS OF LENGTH M4
   14 MM1=M4
      MM2=M1
      MM3=M2
```

```
      MM4=M3
C   INDEXING INITIALIZATION FOR THE TRANSFORMS
    20 N2=0
      N3=0
      N4=0
      K1=MM2*MM3*MM4
      K2=MM1*MM3*MM4
      K3=MM1*MM2*MM4
      K4=MM1*MM2*MM3
      I(1)=0
C   INPUT INDEXING ALONG ONE DIMENSION
    21 DO 22 J=2,MM1
      I(J)=I(J-1)+K1
      IF(I(J).LT.N) GO TO 22
      I(J)=I(J)-N
    22 CONTINUE
C   TRANSFERRING DATA TO TEMPORARY VECTORS UR AND UI
    30 DO 31 J=1,MM1
      IJ=I(J)+1
      UR(J)=XR(IJ)
    31 UI(J)=XI(IJ)
C   TRANSFORM UR,UI
      GO TO(50,200,300,400,500,50,700,800,900,50,50,50,50,50,50,1600),MM
     21
C   PLACE RESULTS OF TRANSFORM BACK IN XR AND XI
    40 DO 41 J=1,MM1
      IJ=I(J)+1
      XR(IJ)=UR(J)
    41 XI(IJ)=UI(J)
C   TESTING FOR COMPLETION OF THIS FACTOR'S TRANSFORMS
      IF(N2.NE.MM2-1) GO TO 51
      N2=0
      IF(N3.NE.MM3-1) GO TO 52
      N3=0
      IF(N4.NE.MM4-1) GO TO 53
    50 NF=NF-1
      IF(NF.EQ.0) GO TO 1000
      GO TO 10
C   INPUT INDEXING ALONG OTHER DIMENSIONS
    51 N2=N2+1
      DO 54 J=1,MM1
      I(J)=I(J)+K2
      IF(I(J).LT.N) GO TO 54
      I(J)=I(J)-N
    54 CONTINUE
```

```
      GO TO 30
   52 N3=N3+1
      I(1)=K3*N3+K4*N4
      IF(I(1).LT.N) GO TO 21
      I(1)=I(1)-N
      GO TO 21
   53 N4=N4+1
      I(1)=K4*N4
      GO TO 21
C  UNSCRAMBLING TRANSFORM RESULTS
 1000 II=1
      J=1
      GO TO 1001
 1002 IF(J.GT.N) GO TO 1003
      II=II+KOUT
 1004 IF(II.LE.N) GO TO 1001
      II=II-N
      GO TO 1004
 1001 A(J)=XR(II)
      B(J)=-XI(II)
      J=J+1
      GO TO 1002
C   2 POINT TRANSFORM
  200 URX=UR(1)+UR(2)
      UIX=UI(1)+UI(2)
      UR(2)=UR(1)-UR(2)
      UI(2)=UI(1)-UI(2)
      UR(1)=URX
      UI(1)=UIX
      GO TO 40
C   3 POINT TRANSFORM
  300 AR=UR(2)+UR(3)
      AI=UI(2)+UI(3)
      MR1=-1.5D0*AR
      MI1=-1.5D0*AI
      MR2=0.8660254038D0*(UR(2)-UR(3))
      MI2=0.8660254038D0*(UI(2)-UI(3))
      UR(1)=AR+UR(1)
      UI(1)=AI+UI(1)
      MR1=UR(1)+MR1
      MI1=UI(1)+MI1
      UR(2)=MR1-MI2
      UI(2)=MI1+MR2
      UR(3)=MR1+MI2
      UI(3)=MI1-MR2
```

```
      GO TO 40
C   4 POINT TRANSFORM
  400 AR1=UR(1)+UR(3)
      AI1=UI(1)+UI(3)
      AR2=UR(1)-UR(3)
      AI2=UI(1)-UI(3)
      AR3=UR(2)+UR(4)
      AI3=UI(2)+UI(4)
      AR4=UR(2)-UR(4)
      AI4=UI(2)-UI(4)
      UR(1)=AR1+AR3
      UI(1)=AI1+AI3
      UR(2)=AR2-AI4
      UI(2)=AI2+AR4
      UR(3)=AR1-AR3
      UI(3)=AI1-AI3
      UR(4)=AR2+AI4
      UI(4)=AI2-AR4
      GO TO 40
C   5 POINT TRANSFORM
  500 AR1=UR(2)+UR(5)
      AI1=UI(2)+UI(5)
      AR2=UR(2)-UR(5)
      AI2=UI(2)-UI(5)
      AR3=UR(3)+UR(4)
      AI3=UI(3)+UI(4)
      AR4=UR(3)-UR(4)
      AI4=UI(3)-UI(4)
      AR5=AR1+AR3
      AI5=AI1+AI3
      MR1=0.9510565163D0*(AR2+AR4)
      MI1=0.9510565163D0*(AI2+AI4)
      MR2=1.538841769D0*AR2
      MI2=1.538841769D0*AI2
      MR3=0.3632712640D0*AR4
      MI3=0.3632712640D0*AI4
      MR4=0.559016994 4D0*(AR1-AR3)
      MI4=0.5590169944D0*(AI1-AI3)
      MR5=-1.25D0*AR5
      MI5=-1.25D0*AI5
      UR(1)=UR(1)+AR5
      UI(1)=UI(1)+AI5
      MR5=UR(1)+MR5
      MI5=UI(1)+MI5
      AR1=MR5+MR4
```

```
      AI1=MI5+MI4
      AR2=MR5-MR4
      AI2=MI5-MI4
      AR3=MR1-MR3
      AI3=MI1-MI3
      AR4=MR1-MR2
      AI4=MI1-MI2
      UR(2)=AR1-AI3
      UI(2)=AI1+AR3
      UR(3)=AR2+AI4
      UI(3)=AI2-AR4
      UR(4)=AR2-AI4
      UI(4)=AI2+AR4
      UR(5)=AR1+AI3
      UI(5)=AI1-AR3
      GO TO 40
C   7 POINT TRANSFORM
  700 AR1=UR(2)+UR(7)
      AI1=UI(2)+UI(7)
      AR2=UR(2)-UR(7)
      AI2=UI(2)-UI(7)
      AR3=UR(3)+UR(6)
      AI3=UI(3)+UI(6)
      AR4=UR(3)-UR(6)
      AI4=UI(3)-UI(6)
      AR5=UR(4)+UR(5)
      AI5=UI(4)+UI(5)
      AR6=UR(4)-UR(5)
      AI6=UI(4)-UI(5)
      AR7=AR1+AR3+AR5
      AI7=AI1+AI3+AI5
      MR1=-1.166666667D0*AR7
      MI1=-1.166666667D0*AI7
      MR2=0.790156468 8D0*(AR1-AR5)
      MI2=0.7901564688D0*(AI1-AI5)
      MR3=0.055854268D0*(AR5-AR3)
      MI3=0.055854268D0*(AI5-AI3)
      MR4=0.734302201D0*(AR3-AR1)
      MI4=0.734302201D0*(AI3-AI1)
      MR5=0.440958552D0*(AR2+AR4-AR6)
      MI5=0.440958552D0*(AI2+AI4-AI6)
      MR6=0.340872931D0*(AR2+AR6)
      MI6=0.340872931D0*(AI2+AI6)
      MR7=-0.533969361D0*(-AR6-AR4)
      MI7=-0.533969361D0*(-AI6-AI4)
```

```
      MR8=0.874842291D0*(AR4-AR2)
      MI8=0.874842291D0*(AI4-AI2)
      UR(1)=UR(1)+AR7
      UI(1)=UI(1)+AI7
      AR1=UR(1)+MR1
      AI1=UI(1)+MI1
      AR2=AR1+MR2+MR3
      AI2=AI1+MI2+MI3
      AR3=AR1-MR2-MR4
      AI3=AI1-MI2-MI4
      AR4=AR1-MR3+MR4
      AI4=AI1-MI3+MI4
      AR5=MR5+MR6+MR7
      AI5=MI5+MI6+MI7
      AR6=MR5-MR6-MR8
      AI6=MI5-MI6-MI8
      AR7=MR5-MR7+MR8
      AI7=MI5-MI7+MI8
      UR(2)=AR2-AI5
      UI(2)=AI2+AR5
      UR(3)=AR3-AI6
      UI(3)=AI3+AR6
      UR(4)=AR4+AI7
      UI(4)=AI4-AR7
      UR(5)=AR4-AI7
      UI(5)=AI4+AR7
      UR(6)=AR3+AI6
      UI(6)=AI3-AR6
      UR(7)=AR2+AI5
      UI(7)=AI2-AR5
      GO TO 40
C   8 POINT TRANSFORM
  800 AR1=UR(2)-UR(8)
      AI1=UI(2)-UI(8)
      AR2=UR(2)+UR(8)
      AI2=UI(2)+UI(8)
      AR3=UR(4)-UR(6)
      AI3=UI(4)-UI(6)
      AR4=UR(4)+UR(6)
      AI4=UI(4)+UI(6)
      AR5=UR(1)-UR(5)
      AI5=UI(1)-UI(5)
      AR6=UR(1)+UR(5)
      AI6=UI(1)+UI(5)
      AR7=UR(3)-UR(7)
```

```
      AI7=UI(3)-UI(7)
      AR8=UR(3)+UR(7)
      AI8=UI(3)+UI(7)
      MR1=0.7071067812D0*(AR1+AR3)
      MI1=0.7071067812D0*(AI1+AI3)
      MR2=0.7071067812D0*(AR2-AR4)
      MI2=0.7071067812D0*(AI2-AI4)
      MR3=AR2+AR4
      MI3=AI2+AI4
      MR4=AR6+AR8
      MI4=AI6+AI8
      MR5=AR6-AR8
      MI5=AI6-AI8
      MR6=AR1-AR3
      MI6=AI1-AI3
      MR7=AR5+MR2
      MI7=AI5+MI2
      MR8=AR5-MR2
      MI8=AI5-MI2
      MR9=AR7+MR1
      MI9=AI7+MI1
      MR10=AR7-MR1
      MI10=AI7-MI1
      UR(1)=MR4+MR3
      UI(1)=MI4+MI3
      UR(2)=MR7-MI9
      UI(2)=MI7+MR9
      UR(3)=MR5-MI6
      UI(3)=MI5+MR6
      UR(4)=MR8+MI10
      UI(4)=MI8-MR10
      UR(5)=MR4-MR3
      UI(5)=MI4-MI3
      UR(6)=MR8-MI10
      UI(6)=MI8+MR10
      UR(7)=MR5+MI6
      UI(7)=MI5-MR6
      UR(8)=MR7+MI9
      UI(8)=MI7-MR9
      GO TO 40
C   9 POINT TRANSFORM
  900 AR1=UR(2)+UR(9)
      AI1=UI(2)+UI(9)
      AR2=UR(2)-UR(9)
      AI2=UI(2)-UI(9)
```

```
AR3=UR(3)+UR(8)
AI3=UI(3)+UI(8)
AR4=UR(3)-UR(8)
AI4=UI(3)-UI(8)
AR5=UR(5)+UR(6)
AI5=UI(5)+UI(6)
AR6=UR(5)-UR(6)
AI6=UI(5)-UI(6)
AR7=UR(4)+UR(7)
AI7=UI(4)+UI(7)
AR8=UR(4)-UR(7)
AI8=UI(4)-UI(7)
AR=AR1+AR3+AR5
AI=AI1+AI3+AI5
MR1=-0.5D0*AR7
MI1=-0.5D0*AI7
MR2=0.8660254038D0*AR8
MI2=0.8660254038D0*AI8
MR3=0.19746542D0*(-AR1+AR5)
MI3=0.19746542D0*(-AI1+AI5)
MR4=0.56857902D0*(AR1-AR3)
MI4=0.56857902D0*(AI1-AI3)
MR5=0.3711136D0*(-AR3+AR5)
MI5=0.3711136D0*(-AI3+AI5)
MR6=0.54253179D0*(AR2-AR6)
MI6=0.54253179D0*(AI2-AI6)
MR7=0.10025579D0*(AR2+AR4)
MI7=0.10025579D0*(AI2+AI4)
MR8=0.44227597D0*(-AR4-AR6)
MI8=0.44227597D0*(-AI4-AI6)
MR9=-1.5D0*AR
MI9=-1.5D0*AI
MR10=0.8660254038D0*(AR2-AR4+AR6)
MI10=0.8660254038D0*(AI2-AI4+AI6)
AR1=UR(1)+MR1
AI1=UI(1)+MI1
UR(1)=AR+AR7+UR(1)
UI(1)=AI+AI7+UI(1)
AR=UR(1)+MR9
AI=UI(1)+MI9
AR2=MR4-MR5
AI2=MI4-MI5
AR3=MR3+MR4
AI3=MI3+MI4
AR4=MR7-MR8
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      AI4=MI7-MI8
      AR5=MR6-MR7
      AI5=MI6-MI7
      AR6=AR2-MR5-MR3+AR1
      AI6=AI2-MI5-MI3+AI1
      AR7=AR3+MR3+MR5+AR1
      AI7=AI3+MI3+MI5+AI1
      AR8=-AR3-AR2+AR1
      AI8=-AI3-AI2+AI1
      MR1=MR6-MR8
      MI1=MI6-MI8
      MR3=AR4+MR1+MR2
      MI3=AI4+MI1+MI2
      MR4=AR5+MR1-MR2
      MI4=AI5+MI1-MI2
      MR5=AR5-AR4+MR2
      MI5=AI5-AI4+MI2
      UR(2)=AR6-MI3
      UI(2)=AI6+MR3
      UR(3)=AR7-MI4
      UI(3)=AI7+MR4
      UR(4)=AR-MI10
      UI(4)=AI+MR10
      UR(5)=AR8-MI5
      UI(5)=AI8+MR5
      UR(6)=AR8+MI5
      UI(6)=AI8-MR5
      UR(7)=AR+MI10
      UI(7)=AI-MR10
      UR(8)=AR7+MI4
      UI(8)=AI7-MR4
      UR(9)=AR6+MI3
      UI(9)=AI6-MR3
      GO TO 40
C 16 POINT TRANSFORM
 1600 AR1=UR(1)+UR(9)
      AI1=UI(1)+UI(9)
      AR2=UR(5)+UR(13)
      AI2=UI(5)+UI(13)
      AR3=UR(3)+UR(11)
      AI3=UI(3)+UI(11)
      AR4=UR(3)-UR(11)
      AI4=UI(3)-UI(11)
      AR5=UR(7)+UR(15)
      AI5=UI(7)+UI(15)
```

```
AR6=UR(7)-UR(15)
AI6=UI(7)-UI(15)
AR7=UR(2)+UR(10)
AI7=UI(2)+UI(10)
AR8=UR(2)-UR(10)
AI8=UI(2)-UI(10)
AR9=UR(4)+UR(12)
AI9=UI(4)+UI(12)
AR10=UR(4)-UR(12)
AI10=UI(4)-UI(12)
AR11=UR(6)+UR(14)
AI11=UI(6)+UI(14)
AR12=UR(6)-UR(14)
AI12=UI(6)-UI(14)
AR13=UR(8)+UR(16)
AI13=UI(8)+UI(16)
AR14=UR(8)-UR(16)
AI14=UI(8)-UI(16)
AR15=AR1+AR2
AI15=AI1+AI2
AR16=AR3+AR5
AI16=AI3+AI5
AR17=AR15+AR16
AI17=AI15+AI16
AR18=AR7+AR11
AI18=AI7+AI11
AR19=AR7-AR11
AI19=AI7-AI11
AR20=AR9+AR13
AI20=AI9+AI13
AR21=AR9-AR13
AI21=AI9-AI13
AR22=AR18+AR20
AI22=AI18+AI20
AR23=AR8+AR14
AI23=AI8+AI14
AR24=AR8-AR14
AI24=AI8-AI14
AR25=AR10+AR12
AI25=AI10+AI12
AR26=AR12-AR10
AI26=AI12-AI10
AR31=UR(1)-UR(9)
AI31=UI(1)-UI(9)
UR(1)=AR17+AR22
```

```
UI(1)=AI17+AI22
UR(9)=AR17-AR22
UI(9)=AI17-AI22
AR29=AR15-AR16
AI29=AI15-AI16
AR30=AR1-AR2
AI30=AI1-AI2
MR1=0.7071067812D0*(AR19-AR21)
MI1=0.7071067812D0*(AI19-AI21)
MR2=0.7071067812D0*(AR4-AR6)
MI2=0.7071067812D0*(AI4-AI6)
MR3=0.3826834324D0*(AR24+AR26)
MI3=0.3826834324D0*(AI24+AI26)
MR4=1.306562965D0*AR24
MI4=1.306562965D0*AI24
MR5=-0.5411961001D0*AR26
MI5=-0.5411961001D0*AI26
AR32=AR18-AR20
AI32=-AI18+AI20
AR33=AR3-AR5
AI33=-AI3+AI5
AR34=UR(5)-UR(13)
AI34=-UI(5)+UI(13)
MR6=0.7071067812D0*(AR19+AR21)
MI6=-0.7071067812D0*(AI19+AI21)
MR7=0.7071067812D0*(AR4+AR6)
MI7=-0.7071067812D0*(AI4+AI6)
MR8=0.9238795325D0*(AR23+AR25)
MI8=-0.9238795325D0*(AI23+AI25)
MR9=-0.5411961001D0*AR23
MI9=0.5411961001D0*AI23
MR10=1.306562965D0*AR25
MI10=-1.306562965D0*AI25
MR11=AR30+MR1
MI11=AI30+MI1
MR12=AR30-MR1
MI12=AI30-MI1
MR13=AR33+MR6
MI13=AI33+MI6
MR14=MR6-AR33
MI14=MI6-AI33
MR15=AR31+MR2
MI15=AI31+MI2
MR16=AR31-MR2
MI16=AI31-MI2
```

```
MR17=MR4-MR3
MI17=MI4-MI3
MR18=MR5-MR3
MI18=MI5-MI3
MR19=MR15+MR17
MI19=MI15+MI17
MR20=MR15-MR17
MI20=MI15-MI17
MR21=MR16+MR18
MI21=MI16+MI18
MR22=MR16-MR18
MI22=MI16-MI18
MR23=AR34+MR7
MI23=AI34+MI7
MR24=AR34-MR7
MI24=AI34-MI7
MR25=MR8+MR9
MI25=MI8+MI9
MR26=MR8-MR10
MI26=MI8-MI10
MR27=MR23+MR25
MI27=MI23+MI25
MR28=MR23-MR25
MI28=MI23-MI25
MR29=MR24+MR26
MI29=MI24+MI26
MR30=MR24-MR26
MI30=MI24-MI26
UR(2)=MR19+MI27
UI(2)=MI19+MR27
UR(3)=MR11+MI13
UI(3)=MI11+MR13
UR(4)=MR22-MI30
UI(4)=MI22-MR30
UR(5)=AR29+AI32
UI(5)=AI29+AR32
UR(6)=MR21+MI29
UI(6)=MI21+MR29
UR(7)=MR12+MI14
UI(7)=MI12+MR14
UR(8)=MR20-MI28
UI(8)=MI20-MR28
UR(10)=MR20+MI28
UI(10)=MI20+MR28
UR(11)=MR12-MI14
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      UI(11)=MI12-MR14
      UR(12)=MR21-MI29
      UI(12)=MI21-MR29
      UR(13)=AR29-AI32
      UI(13)=AI29-AR32
      UR(14)=MR22+MI30
      UI(14)=MI22+MR30
      UR(15)=MR11-MI13
      UI(15)=MI11-MR13
      UR(16)=MR19-MI27
      UI(16)=MI19-MR27
      GO TO 40
 1003 RETURN
      END
      SUBROUTINE PATOUT(N,T)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/DFT/A(5040),B(5040)
      WRITE(6,45)
   45 FORMAT(1H1,43X,25HFAR FIELD ANTENNA PATTERN/7X,5HANGLE,4X,7HMAG(DB
     2),5X,5HPHASE)
      C12=A(1)**2+B(1)**2
      DEG=57.29577951310D0
      DO 50 I=1,N
      S=(I-1)/(N*T)
      IF(S.GT.1.0) GO TO 60
      C2=A(I)**2+B(I)**2
      CDB=10.0D0*DLOG10(C2/C12)
      PHA=DATAN(B(I)/A(I))*DEG
      IF(B(I).LT.0.0) PHA=PHA+180.0D0
      ANG=DARSIN(S)*DEG
      WRITE(6,70)ANG,CDB,PHA
   50 CONTINUE
   60 CONTINUE
   70 FORMAT(3(3X,F10.4,F10.4,F10.4))
      RETURN
      END
```