The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology, with the support of the U.S. Air Force under Contract AF 19(628)-5167.

This report may be reproduced to satisfy needs of U.S. Government agencies.

This document has been approved for public release and sale; its distribution is unlimited.
ERRATA SHEET
for Technical Note 1967-2

Because of unclear printing in Technical Note 1967-2 (N. M. Brenner, "Three Fortran Programs that Perform the Cooley-Tukey Fourier Transform," 28 July 1967), the distinction between + and * was often lost. A list of clarifications follows on the attached sheets.
THE FOLLOWING THREE PATTERNS OCCUR FREQUENTLY.

\[ BR = WR \times AR - WI \times AI \]

\[ BI = AI \times WR + AR \times WI \]

\[ DATA(J) = DATA(I) - TEMPR \]

\[ DATA(J+1) = DATA(I+1) - TEMPI \]

\[ DATA(I) = DATA(I) + TEMPR \]

\[ DATA(I+1) = DATA(I+1) - TEMPI \]

\[ INDEX2MAX = INDEX1 + N1 - N2 \]

ISTEP = 2 * MMAX

NTOT = NTOT * NN(IDIM)

NP2 = NP1 * N

NP0 = NP1 * (1 + NPREV/2)

UMAX = I2 + NP1 - 2

INDEX1 = INDEX1 + N1 - N2

NWORK = 2 * N

IF (ICASE = 3) 210, 220, 210

J = J + IFP1

IF (J + I3 - IFP2) 260, 250, 250

KMIN = 4 * (KMIN - I1) + U

KSTEP = 4 * KDIF

KMIN = 4 * (KMIN - I1) + I1

KDIF = KSTEP

IF (KDIF - NP2HF) 450, 450, 530

WR = (WR + WI) * RTHLF
P. 25, L. 560+2 AND P. 19, L. 560+2
WI=(TEMPR+WI)*RTHLF

P. 25, L. 570+2 AND P. 19, L. 570+2
MMAX=MMAX+MMAX

P. 26, L. 650+2
J2RNG=IFP1*(1+IFACT(IF)/2)

P. 26, L. 655+2
I=1+(J3-I3)/NP1HF

P. 26, L. 665
ICONJ=1+(IFP2-2*J2+I3+J3)/NP1HF

P. 27, L. 670+1
TEMPI=SUMI
SUMR=TWOWR*SUMR-OLDSR+DATA(J)
SUMI=TWOWR*SUMI-OLDSI+DATA(J+1)
OLDSR=TEMPR
OLDSI=TEMPI
J=J-IFP1
IF(J-JMIN)675,675,670

675 TEMPR=WR*SUMR-OLDSR+DATA(J)
TEMPI=WI*SUMI
WORK(I)=TEMPR-TEMPI
WORK(ICONJ)=TEMPR+TEMPI
TEMPR=WR*SUMI-OLDSI+DATA(J+1)
TEMPI=WI*SUMR
WORK(I+1)=TEMPR+TEMPI
WORK(ICONJ+1)=TEMPR-TEMPI

P. 27, L. 690+2
I2MAX=I3+NP2-NP1

P. 27, L. 710-2
JMIN=2*NHALF-1

P. 28, L. 740
NP2=NP2+NP2

P. 28, L. 745-1
IMAX=NTOT/2+1

745 IMIN=IMAX-2*NHALF

P. 28, L. 805+1
I2MAX=I3+NP2-NP1

P. 28, L. 805+3
IMIN=I2+I1RNG
IMAX=I2+NP1-2
JMAX=2*I3+NP1-IMIN

P. 28, L. 810
JMAX=JMAX+NP2
820 IF(IDIM-2)850,850,830
830 J=JMAX+NP0

P. 28, L. 840
840 J=J-2

P. 28, L. 860
860 J=J-NP0
THREE FORTRAN PROGRAMS THAT PERFORM
THE COOLEY-TUKEY FOURIER TRANSFORM

N. M. BRENNER

Group 31

TECHNICAL NOTE 1967-2

28 JULY 1967
ABSTRACT

This note describes and lists three programs, all written in USASI Basic Fortran, which perform the discrete Fourier transform upon a multi-dimensional array of floating point data. The data may be either real or complex, with a savings in running time for real over complex. The transform values are always complex and are returned in the array used to carry the original data. The running time is much shorter than that of any program performing a direct summation, even when sine and cosine values are precalculated and stored in a table. For example, on a CDC 3300 with floating point add time of six microseconds, a complex array of size 80 X 80 can be transformed in 19.2 seconds. Besides the main array, only a working storage array of size 160 need be supplied.

Accepted for the Air Force
Franklin C. Hudson
Chief, Lincoln Laboratory Office
This note describes and lists three programs, all written in USASI Basic Fortran, which perform the discrete Fourier transform upon a multi-dimensional array of floating point data. The data may be either real or complex, with a savings in running time for real over complex (see Timing). The transform values are always complex and are returned in the array used to carry the original data. The running time is much shorter than that of any program performing a direct summation, even when sine and cosine values are precalculated and stored in a table. For example, on a CDC 3300 with floating point add time of six microseconds, a complex array of size 80 x 80 can be transformed in 19.2 seconds. Besides the main array, only a working storage array of size 160 need be supplied.

The exact operation performed is called finite discrete Fourier transformation, also known as harmonic analysis or trigonometric interpolation. Given an array of data \( \text{DATA(I1,I2,...)}, \)

\[
\text{TRANSFORM(J1,J2,...)} = \sum \left[\text{DATA(I1,I2,...)} \cdot W_1^{(I1-1)(J1-1)} \cdot W_2^{(I2-1)(J2-1)} \ldots \right],
\]

where \( W_1 = \exp(-2\pi i/N_1), \) \( W_2 = \exp(-2\pi i/N_2), \ldots \) and \( I_1 \) and \( J_1 \) run from 1 to \( N_1 \), \( I_2 \) and \( J_2 \) run from 1 to \( N_2 \), etc. The Fortran convention of subscripts beginning at one is adhered to. This summation possesses many of the properties of the more usual infinite integral

\[
F(y) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i xy} \, dx.
\]

By interpreting the subscripts modulo \( N_1, N_2, \ldots \) and requiring the data to represent equispaced points, we can easily prove the usual properties about linearity, orthogonality, inverse transform and relationship to convolution. See Gentleman and Sande ([3], 1966).
There is no limit on the dimensionality (number of subscripts) of the data array. A three-dimensional transform can be performed as easily as a one-dimensional transform, though in a proportionately greater time. An inverse transform can be performed, in which the sign in the exponentials is +, instead of -. If an inverse transform is performed upon an array of transformed data, the original data will reappear multiplied by \( N_1 \cdot N_2 \cdot \ldots \).

The length of each dimension may be any integer, and as large as storage will permit. However, the program runs faster on composite integers than on primes, and is particularly fast on numbers rich in factors of two. For example, on the CDC 3300, the following timings for a one-dimensional transform have been calculated from the timing formula:

<table>
<thead>
<tr>
<th>( N )</th>
<th>Factorization</th>
<th>Time for Complex Transform (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1094</td>
<td>( 2 \times 23 \times 89 )</td>
<td>80</td>
</tr>
<tr>
<td>1095</td>
<td>( 3^2 \times 5 \times 7 \times 13 )</td>
<td>24</td>
</tr>
<tr>
<td>1096</td>
<td>( 2^{12} )</td>
<td>6.2</td>
</tr>
<tr>
<td>1097</td>
<td>( 17 \times 241 )</td>
<td>180</td>
</tr>
<tr>
<td>1098</td>
<td>( 2 \times 3 \times 683 )</td>
<td>480</td>
</tr>
<tr>
<td>1099</td>
<td>prime</td>
<td>2868</td>
</tr>
<tr>
<td>1100</td>
<td>( 2^2 \times 5^2 \times 41 )</td>
<td>39</td>
</tr>
</tbody>
</table>

**Calling Sequence**

The listings of three programs are given in the appendices. FOURL is a subset of FOUR2, which in turn is a subset of FOURT. FOURT is the most general, accepting multidimensional arrays of any size. FOUR2 is the same speed as FOURT but accepts only complex multidimensional arrays whose dimensions are powers of two. FOURL is much slower than FOURT or FOUR2, and performs only one-dimensional transforms on complex arrays whose lengths are powers of two. FOURL is intended mainly for pedagogical purposes; it is half a page of Fortran, the others being much longer.
The calling sequences are:

CALL FOURT (DATA,NN,NDIM,ISIGN,IFORM,WORK)
CALL FOUR2 (DATA,NN,NDIM,ISIGN)
CALL FOUR1 (DATA,NN,ISIGN)

In all cases, DATA is the array used to hold the real and imaginary parts of the input data and the transform values on output. The real and imaginary parts of a datum must be placed into immediately adjacent locations in storage. This is the form of storage used by Fortran IV, and may be accomplished in Fortran II by making the first dimension of DATA of length two, referring to the real and imaginary parts. If the data placed in DATA on input are real, they must have imaginary parts of zero appended. The transform values are always complex and replace the input data. Hence, the array DATA must always be of complex format.

For FOUR1, array DATA must be one-dimensional, of length NN. For FOUR2 and FOURT, it may be multidimensional. The extent of each dimension (except for the possible first dimension referring to the real and imaginary parts) is given in the integer array NN, which is of length NDIM, the number of dimensions. That is, NN(1) = N1, NN(2) = N2, etc.*

ISIGN is an integer used to indicate the direction of the transform. It is minus one to indicate a forward transform (exponential sign is -) and plus one to indicate an inverse transform (sign is +). The scale factor \( 1/(N1*N2*...) \) frequently seen in definitions of the Fourier transform must be applied by the user.

If the data being passed to FOURT are real (i.e., have zero imaginary parts), the integer IFORM should be set to zero. This will speed execution (see Timing). For complex data, IFORM must be plus one.

WORK is an array used by FOURT when any of the dimensions of DATA is not a power of two. Since FOUR2 and FOUR1 are restricted to powers of two, WORK is not needed. If the dimensions of DATA are all powers of two in FOURT, WORK may be replaced by a zero in the calling sequence. Otherwise, it must be

* As usual, the first subscript varies the fastest in storage order.
supplied, a real floating point array of length twice the longest dimension of DATA which is not a power of two. In one dimension, for the length not a power of two, WORK occupies as many storage locations as DATA. If given, it may not be the same array as DATA.

Double precision versions of these programs may be obtained by changing the names to DFOURT, DFOUR2, and DFOUR1, declaring double precision all variables not beginning with the letters I, J, K, L, M or N, changing the references to COS and SIN to DCOS and DSIN and assigning the correct precision constants to TWOPI (2π) and RTHLF (0.5^½). DATA and WORK must then be double precision arrays.

Storage and Common

No common of any kind is used. An integer array of length thirty-two is used by FOURT. FOURT is about four hundred Fortran statements long, FOUR2 about one hundred and twenty and FOUR1 thirty-seven.

Return and Error Messages

There are no error messages, error halts or error returns in this program. If NDIM or any NN(I) is less than one, the program returns immediately.

Algorithm

A heavily modified version of the algorithm discovered independently by Danielson and Lanczos ([2], 1942), Good ([4], 1958), and Cooley and Tukey ([1], 1965) is used. The following example is an application to a one-dimensional transform of length six.

Let \( w = e^{-2\pi i/6} \). The transformation is written

\[
\begin{align*}
t_0 &= d_0 + d_1 + d_2 + d_3 + d_4 + d_5 \\
t_1 &= d_0 + w d_1 + w^2 d_2 + w^3 d_3 + w^4 d_4 + w^5 d_5 \\
t_2 &= d_0 + w^2 d_1 + w^4 d_2 + w^6 d_3 + w^8 d_4 + w^{10} d_5
\end{align*}
\]
Straightforward computation requires 25 complex multiplications and 30 complex additions. The fast Fourier transform computes as follows:

\[
\begin{align*}
t_3 &= d_0 + \omega^2d_1 + \omega^6d_2 + \omega^9d_3 + \omega^{12}d + \omega^{15}d_5 \\
t_4 &= d_0 + \omega^4d_1 + \omega^8d_2 + \omega^{12}d_3 + \omega^{16}d_4 + \omega^{20}d_5 \\
t_5 &= d_0 + \omega^5d_1 + \omega^{10}d_2 + \omega^{15}d_3 + \omega^{20}d_4 + \omega^{25}d_5
\end{align*}
\]

which requires only 13 complex multiplications and 18 complex additions. Note that \(\omega^3 = -1\) and \(\omega^6 = 1\).

Such a reduction in computation can be found for any length which is a composite integer. The algebraic proof may be found in the appendix. Also, the various techniques for performing multidimensional transforms, real transforms, etc., are discussed there.

Special Cautions and Features

The finite discrete Fourier transform places three restrictions upon the data:

1. The data must form one cycle of a periodic function. Alternately stated, the subscripts are interpreted modulo \(N\).
2. The number of input data and the number of transform values must be the same.
3. The data must be equispaced in each dimension (though, of course, the interval need not be the same for each dimension). Further, if in any dimension the input data are spaced at interval dt, the resulting transform values will be spaced from 0 to 2\( \pi (N-1)/(Ndt) \) at interval \( 2\pi/(Ndt) \) as \( I \) runs from 1 to \( N \). By periodicity, the upper limit is identified with \(-2\pi/(Ndt)\) and in fact all points above the "foldover frequency" \( \pi/(Ndt) \) are to be identified with the corresponding negative frequency.

Those familiar with other implementations of the fast Fourier transform may be aware that the order of the data is scrambled in the course of execution. Unscrambling is performed automatically, however, and both the input and output values are placed in ordinary sequential arrangement.

**Timing**

Let \( N_{\text{total}} \) be the total number of points in the data array. That is, \( N_{\text{total}} = N_1*N_2*... \). Decompose \( N_{\text{total}} \) into its prime factors, such as \( 2^2*3*5*... \). Let \( \Sigma_2 \) be the sum of all the factors of two in \( N_{\text{total}} \), that is, \( \Sigma_2 = 2*K_2 \). Let \( \Sigma_f \) be the sum of all the other factors, \( \Sigma_f = 3*K_3 + 5*K_5 + ... \).

The time taken for a multidimensional transform is

\[
T = T_0 + N_{\text{total}} \left[ T_1 + T_2 \Sigma_2 + T_f \Sigma_f \right].
\]

For the CDC 3300,

\[
T = 3000 + N_{\text{total}} \left[ 600 + 40 \Sigma_2 + 175 \Sigma_f \right] \text{ microseconds}.
\]

The greater optimization apparent for factors of two is due to

1. The eight-fold symmetry of the trigonometric functions from 0 to \( 2\pi \).
2. The fact that Fourier transforms of length two and four require fewer complex multiplies than transforms of other lengths.

The above timing formula is accurate for complex data.

The use of real data (\( \text{IFORM} = 0 \)) can reduce running time by as much as forty percent. On the CDC 3300, a 64 \times 64 complex array was transformed in
6.1 seconds; a 64 x 64 real array took 4.2 seconds. A complex array 1500 long took 6.1 seconds, while a real 1500 array ran only 3.4 seconds.

**Accuracy**

The simplistic idea about accuracy is apparently correct: because the fast Fourier transform takes fewer steps in execution, less error creeps in. Gentleman and Sande ([3], 1966) show theoretically that the root-mean-square relative error is bounded by

\[
1.06 \frac{N_{\text{total}}}{3} 2^{-b} \sum_{j} [2f_j]^{3/2}
\]

where \( b \) is the number of bits in the floating-point fraction and \( f_j \) are the factors of \( N_{\text{total}} \).

Further error is introduced in this particular program by the use of recursive generation of sines and cosines for factors of \( N_{\text{total}} \) other than two. Sines and cosines needed for factors of two are computed precisely. In actual practice, out of eleven and a half digits representable on the CDC 3300, about four were lost on long one-dimensional sequences like 1500 and 4096.

**Applications**

Besides all the direct uses of discrete Fourier transforms in signal processing, lens design, crystallography, seismic studies, etc., Fourier transforms find application in techniques of correlation and convolution. The principal tool here is the convolution theorem. Denoting the convolution of two discrete functions \( f \) and \( g \) by \( f*g \)

\[
(f*g)_k = \sum_j f_j g_{k-j}
\]

where both \( j \) and \( k \) run from 1 to \( N \) and subscripts are interpreted modulo \( N \), and denoting the discrete Fourier transform of \( f \) by \( F(f) \), the convolution theorem states

\[
F(f*g) = F(f) F(g).
\]
The difficulties here are that cyclical interpretation of subscripts may not be desirable and that $N$ may not be convenient for fastest processing via the fast Fourier transform. Appendage of zeroes to the ends of the sequences solves both problems. See Stockham ([5], 1966) and Gentleman and Sande ([3], 1966).

**Examples of Use**

A. **FOURT**

1. **Forward transform of complex $50 \times 40$ array in Fortran II**

   ```fortran
   DIMENSION DATA (2,50,40), WORK (100), NN (2)
   NN (1) = 50
   NN (2) = 40
   DO 1 I = 1, 50
   DO 1 J = 1, 100
   DATA (I,J) = real part
   CALL FOURT (DATA,NN,2,-1,1*WORK)
   ```

2. **Same example as 1, but in Fortran IV**

   ```fortran
   DIMENSION DATA (50,40), WORK (100), NN (2)
   COMPLEX DATA
   DATA NN/50, 40/
   DO 1 I = 1, 50
   DO 1 J = 1, 40
   DATA (I,J) = complex value
   CALL FOURT (DATA,NN,2,-1,1,WORK)
   ```

3. **Same example as 2, but in double precision**

   Add the following statement:
   ```fortran
   DOUBLE PRECISION DATA, WORK
   ```

   Change the call to:
   ```fortran
   CALL DFOURT (DATA,NN,2,-1,1,WORK)
   ```
4. Inverse transform of real $64 \times 32$ array in Fortran IV

```fortran
DIMENSION DATA (64,32), NN(2)
COMPLEX DATA
DATA NN/64,32/
DO 1 I = 1, 64
   DO 1 J = 1, 32
      1 DATA(I,J) = real value
CALL FOURT (DATA,NN,2,+1,0,0)
```

B. FOUR2

Inverse transform of real $64 \times 32$ array in Fortran IV

```fortran
DIMENSION DATA (64,32), NN(2)
COMPLEX DATA
DATA NN/64,32/
DO 1 I = 1, 64
   DO 1 J = 1, 32
      1 DATA(I,J) = real value
CALL FOUR2 (DATA,NN,2,+1)
```

C. FOUR1

Forward transform of real array of length 2048 in Fortran II

```fortran
DIMENSION DATA (2,2048)
DO 1 I = 1, 2048
   DATA(1,I) = real part
   1 DATA(2,I) = 0
CALL FOUR1 (DATA,2048,-1)
```

Acknowledgments

The author's interest in the fast Fourier transform was sparked by Thomas Stockham. The original program was written by Charles Rader, and the idea for digit reversal was contributed by Ralph Alter. Additional ideas were gleaned from papers by Langdon and Sande, and Bingham.


Appendix I

Historical Sketch

In 1903 Runge published schemes for the optimal computation of twelve and twenty-four point Fourier transforms ([6]). They involved grouping and re-grouping of values in a manner similar to the modern FFT. Runge's schemes are well known and appear in many works on numerical analysis, including Runge and König ([7], 1924) and Whittaker and Robinson ([8], 1944). Nevertheless, no one thought of generalizing Runge's ideas until 1942 when Danielson and Lanczos ([2]) published an optimal algorithm for \( N \cdot 2^k \) point transforms. Their paper passed unnoticed.

Meanwhile, in 1937 Yates ([9]) had devised an algorithm for the efficient computation of the interactions of \( 2^N \) factorial experiments. This involves sums of the form

\[ t_j = \sum d_i (-1)^{i_0 j_0 + i_1 j_1 + \ldots} \]

where \( i_0 i_1 \ldots \) and \( j_0 j_1 \ldots \) are the binary representations of \( i \) and \( j \). Davies et al extended the method to \( 3^N \) experiments ([10], 1954); three years later, Good, in an abstruse paper, extended it to general factorial experiments ([14], 1958). In the same paper, Good devised analogous algorithms for \( N \) point Fourier transforms, where \( N \) is decomposable into mutually prime factors. Cooley and Tukey removed this restriction and clarified Good's argument ([1], 1965). They also wrote what was probably the first computer program to perform FFT.

Cooley and Tukey's paper sparked a resurgence of interest in the Fourier transform. Despite its indispensability in many areas of signal processing, the Fourier transform had long been avoided for reasons of long computation time. The FFT revived interest to such an extent that the IEEE Audio Transactions has devoted an entire issue to it (June 1967) and three groups have proposed implementing it in hardware ([11], 1963; [12], 1967; [13], 1967).
Appendix II

The Mathematics of the Fast Fourier Transform

Mathematical descriptions of the algorithms used in the Fast Fourier Transform subroutines will be published in the near future.

Punched decks for these three subroutines are available from J. J. Fitzgerald, J-105, or from SHARE.
Listing of the Fortran Subroutines

The listings of the three subroutines FOUR1, FOUR2, and FOURT are given on the following pages. All three are written in USASI Basic Fortran, and, as such are compatible with the great majority of Fortran compilers.
SUBROUTINE FOURK DATA (NN, ISIGN)

THE COOLEY-TUKEY FAST FOURIER TRANSFORM IN USASI BASIC FORTRAN

TRANSFORM(J) = SUM (DATA(I) * W**(I*J/M)) WHERE I AND J RUN
FROM 1 TO NN AND W = EXP(ISIGN*2*PI/SORT(NN)), DATA IS A ONE-

DIMENSIONAL COMPLEX ARRAY (I.E., THE REAL AND IMAGINARY PARTS OF
THE DATA ARE LOCATED IMMEDIATELY ADJACENT IN STORAGE, SUCH AS
FORTH IV PLACES THEM WHOSE LENGTH NN IS A POWER OF TWO, ISIGN
IS 1 OR -1, GIVING THE SIGN OF THE TRANSFORM; TRANSFORM VALUES
ARE RETURNED IN ARRAY DATA, REPLACING THE INPUT DATA, THE TIME IS
PROPORTIONAL TO NLOG2(N), RATHER THAN THE USUAL N*2, WRITTEN BY
NORMAN BRENNER; JUNE 1987; THIS IS THE SHORTEST VERSION
OF FFT KNOWN TO THE AUTHOR, AND IS INTENDED MAINLY FOR
DEMONSTRATION, PROGRAMS FOUR2 AND FOURT ARE AVAILABLE THAT RUN
THICE AS FAST AND OPERATE ON MULTIDIMENSIONAL ARRAYS WHOSE
DIMENSIONS ARE NOT RESTRICTED TO POWERS OF TWO, (LOOKING UP SINES
AND COSINES IN A TABLE WILL CUT RUNNING TIME OF FOUR1 BY A THIRD.)

SEE IEEE AUDIO TRANSACTIONS (JUNE 1987), SPECIAL ISSUE ON FFT,
DIMENSION DATA(I)

N = 2*NN

J = 1
DO 5 I = 1, NN
IF (I/J) = 2, 2, 2
TEMPR = DATA(J)
TEMPI = DATA(J/I)
DATA(J) = DATA(I) + DATA(I+J)
DATA(I) = TEMPR
DATA(I+J) = TEMP1
5 CONTINUE

M = NN/2

IF (J = M, 3, 4)
M = M/2

IF (M = 2, 3, 3)
M = M/2

IF (M = 2, 3, 3)
M = M/2

M = 2
THETA = 3.1415926535 * FLOAT(ISIGN) * FLOAT(M)/FLOAT(MMAX)
W = COS(THETA)
WI = SIN(THETA)
DO 8 M = M, MMAX
J = M
TEMP = DATA(J) * W * DATA(J+I)
TEMPR = DATA(J+I) * W * DATA(J)
DATA(J) = DATA(J) + DATA(J+I)
DATA(J+I) = TEMP
DATA(I) = DATA(I) + DATA(I+J)
DATA(I+J) = TEMP
8 CONTINUE

RETURN
END
The Cooley-Tukey Fast Fourier Transform in BASIC FORTRAN

SUBROUTINE FOUR2(DATA, NN, NDIM, SIGN)

THE COOLEY-TUKEY FAST FOURIER TRANSFORM IN USASI BASIC FORTRAN

TRANSFORM(J1,J2,...) = SUM(DATA(I1,J2,...)) * W1 * W2 * ... * WNDIM

WHERE I1 AND J1 RUN FROM 1 TO NN(I1) AND W1 = EXP(2*PI*SIGN/NN(I1))

DATA IS A MULTIDIMENSIONAL FLOATING-POINT ARRAY ALL OF WHOSE
DIMENSIONS ARE POWERS OF TWO; THE LENGTH OF EACH DIMENSION IS
STORED IN THE INTEGER ARRAY NN, OF LENGTH NDIM. SIGN IS
-1 OR +1, GIVING THE SIGN OF THE TRANSFORM, THE REAL
AND IMAGINARY PARTS OF A DATUM ARE IMMEDIATELY ADJACENT IN STORAGE
(SUCH AS FORTRAN IV PLACES THEM), TRANSFORM RESULTS ARE RETURNED
IN ARRAY DATA, REPLACING THE ORIGINAL DATA. TIME IS PROPORTIONAL
TO N*LOG2(N), RATHER THAN THE USUAL N*N. NOTE THAT IF A FORWARD
TRANSFORM IS FOLLOWED BY AN INVERSE TRANSFORM, THE ORIGINAL DATA
WILL REAPPEAR MULTIPLIED BY NN(I1)*NN(I2)*... EXAMPLES
FORWARD FOURIER TRANSFORM OF A TWO-DIMENSIONAL ARRAY IN FORTRAN II
DIMENSION DATA(2,64,32), NN(2)

NN(1) = 64
NN(2) = 32
DO 1 J = 1, 64
DATA(1,J) = REAL PART
1 DATA(2,J) = IMAGINARY PART
CALL FOUR2(DATA, NN, 2, -1)

SAME EXAMPLE IN FORTRAN IV
DIMENSION DATA(64,32), NN(2)
COMPLEX DATA
DATA NN/64,32/
DO 1 J = 1, 64
DATA(1,J) = COMPLEX VALUE
1 DATA(2,J) = COMPLEX VALUE
CALL FOUR2(DATA, NN, 2, -1)

PROGRAM BY NORMAN BRENNER FROM THE BASIC PROGRAM BY CHARLES
RADEAR, MAY 1967; THE IDEA FOR THE DIGIT REVERSAL WAS SUGGESTED
BY RALPH ALLEN;

THIS VERSION OF THE FAST FOURIER TRANSFORM IS THE FASTEST KNOWN
TO THE AUTHOR, LOOKING UP SINES AND COSINES IN A TABLE INSTEAD OF
COMPUTING THEM WOULD DECREASE RUNNING TIME SEVEN PERCENT.
PROGRAMS FOURT AND FOUR1 ARE AVAILABLE FROM THE AUTHOR THAT ALSO
PERFORM THE FAST FOURIER TRANSFORM AND ARE WRITTEN IN USASI BASIC
FORTRAN, FOURT IS THREE TIMES AS LONG, IS NOT RESTRICTED TO
POWERS OF TWO, AND RUNS UP TO FORTY PERCENT FASTER ON REAL DATA;
FOUR1 IS ONE FOURTH AS LONG, ONE HALF AS FAST, AND IS RESTRICTED
TO ONE DIMENSION AND POWERS OF TWO.

SEE-- IEEE AUDIO TRANSACTIONS (JUNE 1967), SPECIAL ISSUE ON FFT.
DIMENSION DATA(I1), NN(I1)
IF(NDIM<1700) STOP 1
1 NTOT = 2
DO 2 IDIM = 1, NDIM
IF(NN(IDIM)<1700) STOP 2
2
NTOT=NTOT*NN(IDIM)
RTMFL=73710.67812
TWOPi=6.28318.53070

MAIN LOOP FOR EACH DIMENSION

NP1=2
DO 600 IDIM=1,NDIM
N=NN(IDIM)
NP2=NP1*N
IF(N=1700,600,100)

SHUFFLE DATA BY BIT REVERSAL; SINCE N=2**K, AS THE SHUFFLING
CAN BE DONE BY SIMPLE INTERCHANGE, NO WORKING ARRAY IS NEEDED

100 NP2HF=NP2/2
J=1
DO 160 12*1,NP2,NP1
IF(J=12)110,130,130
110 I1MAX=I12=NP1-2
DO 120 11=I12,I1MAX,2
DO 120 13=I11,NTOT,NP2
J3=J+I3-12
TEMPR=DATA(I3)
TEMPJ=DATA(I3+1)
DATA(I3+1)=DATA(I3)
DATA(I3)=TEMPJ
120 DATA(J3+1)=TEMPR
130 M=NP2HF
140 IF(J=M)160,160,190
150 J=J+M
IF(M=NP1)160,140,140
160 J=J+M

MAIN LOOP, PERFORM FOURIER TRANSFORMS OF LENGTH FOUR; WITH ONE OF
LENGTH TWO IF NEEDED; THE TWIDDLE FACTOR W=EXP(I*SIGN*2*M/)
SORT(I)*M/(4*MMAX)), CHECK FOR THE SPECIAL CASE W=SIGN*SORT(I)
AND REPEAT FOR W=W*W*W*SIGN/3/SORT(2)

NP1TW=NP1*NP1
IPAR=IPAR/N
310 IF(IPAR=3)330,330,330
320 IPAR=IPAR/4
GO TO 310
360 l=IPAR*350,330,320
DO 340 11=1,NP1,2
DO 340 K1=1,NTOT,NP1TW
K2=K1*NPI
TEMPR=DATA(K2)
TEMPJ=DATA(K2+1)
DATA(K2)=DATA(K1+1)
DATA(K2+1)=DATA(K1)+TEMPJ
DATA(K1)=DATA(K2)+TEMPR
340 DATA(K1+1)=DATA(K1+1)+TEMPR
350 MMAX=NP1
360 IF(MMAX=NP2HF)370,600,600
370 LMAX=MAMX(NP1TW,MMAX/2)
DO 570 1=NP1,LMAX,NP1TW
M=I
IF(M=MAMX)520,420,380
380 THTA=THM1*FLOAT(M)/FLOAT(4*MMAX)
IF(ISIGN)400 TO 390
390 THETA = THETA
400 WR = COS(THETA)
410 W1 = SIN(THETA)
420 DO 330 I = 1, NP1, 2
430 KMIN = IPAR + 11
440 IF(MAX = NP1) 430, 430, 440
450 KSTEP = 4 * KDIFF
460 DO 520 K = KMIN, NTOT, KSTEP
470 IF(NMAX = NP1) 460, 460, 480
480 UIR = DATA(K1) + DATA(K2)
490 GO TO 510
500 U4 = DATA(K4) + DATA(K3)
510 GO TO 510
520 T2R = WR + DATA(K2) + W2 + DATA(K2)
530 T3R = WR + DATA(K3) + W1 + DATA(K2)
540 T4R = W1 + DATA(K4) + W3 + DATA(K2)
550 UIR = DATA(K1) + T2R
560 U2 = DATA(K2) + T2
570 U3 = DATA(K3) + T3
580 UI2 = DATA(K2) + T1
590 U4 = DATA(K4) + T3
600 GO TO 510
610 DATA(K1) = UIR + UIR
DATA(K2) = UIR + UIR
DATA(K3) = UIR + UIR
DATA(K4) = UIR + UIR
DATA(K5) = UIR + UIR
DATA(K6) = UIR + UIR
DATA(K7) = UIR + UIR
DATA(K8) = UIR + UIR
DATA(K9) = UIR + UIR
MM = NMIN + 11
CONTINUE:
MMAX=LHAX
IF(M=MMAX) 540, 540, 570
540 IF(IPTION) 550, 560, 560
550 TEMP=WR
WR=WR+1=RTHLF
W=WN=TEMP=RTMLF
GO TO 410
560 TEMP=WR
WR=WR+1=RTHLF
W=WN=TEMP=RTMLF
GO TO 410
570 CONTINUE:
IF=IF=3=IPAR
MMAX=MMAX=MMAX
GO TO 360
600 NPL=NP2
700 RETURN
SUBROUTINE FOURT(DATA,NN,NDIM,ISIGN,IFORM,WORK)

THE COOLEY-TUKEY FAST FOURIER TRANSFORM IN BASIC FORTRAN

TRANSFORM(J1,J2,...) = SUM(DATA(I1,I2,...)*W1*W2*...*WJ1*WJ2*...)

WHERE I1 AND J1 RUN FROM 1 TO NN(1) AND W1=EXP(ISIGN*PI*SORT(I1)/NN(1)); ETC. THERE IS NO LIMIT ON THE DIMENSIONALITY (NUMBER OF SUBSCRIPTS) OF THE DATA ARRAY, IF AN INVERSE TRANSFORM (ISIGN=-1) IS PERFORMED UPON AN ARRAY OF TRANSFORMED DATA, THE ORIGINAL DATA WILL REAPPEAR. MULTIPLIED BY NN(1)*NN(2)*... THE ARRAY OF INPUT DATA MUST BE IN COMPLEX FORMAT, HOWEVER; IF ALL IMAGINARY PARTS ARE ZERO (I.E., THE DATA ARE DISGUISED REAL) RUNNING TIME IS CUT UP TO FORTY PERCENT. (FOR FASTEST TRANSFORM OF REAL DATA, NN(1) SHOULD BE EVEN.) THE TRANSFORM VALUES ARE ALWAYS COMPLEX, AND ARE RETURNED IN THE ORIGINAL ARRAY OF DATA, REPLACING THE INPUT DATA. THE LENGTH OF EACH DIMENSION OF THE DATA ARRAY MAY BE ANY INTEGER. THE PROGRAM RUNS FASTER ON COMPOSITE INTEGERS THAN ON PRIMES AND IS PARTICULARLY FAST ON NUMBERS RICH IN FACTORS OF TWO. TIMING IS IN FACT GIVEN BY THE FOLLOWING FORMULA. LET NTOT BE THE TOTAL NUMBER OF POINTS (REAL OR COMPLEX) IN THE DATA ARRAY, THAT IS, NTOT=NN(1)*NN(2)*... DECOMPOSE NTOT INTO ITS PRIME FACTORS, SUCH AS 2**K2 * 3**K3 * 5**K5 * ... LET SUM2 BE THE SUM OF ALL THE FACTORS OF TWO IN NTOT, THAT IS, SUM2 = 2**K2, LET SUMF BE THE SUM OF ALL OTHER FACTORS OF NTOT, THAT IS, SUMF = 3**K3*5**K5*... THE TIME TAKEN BY A MULTIDIMENSIONAL TRANSFORM ON THESE NTOT DATA IS T = T0 + NTOT*(T1*T2*SUM2*T3*SUMF), ON THE CDC 3300 (FLOATING POINT ADD TIME = SIX MICROSECONDS), T = 3000 + NTOT*(400*400*SUM2 = 175*SUMF) MICROSECONDS ON COMPLEX DATA.

IMPLEMENTATION OF THE DEFINITION BY SUMMATION WILL RUN IN A TIME PROPORTIONAL TO NTOT*(NN(1)*NN(2)*...), FOR HIGHLY-COMPOSITE NTOT, THE SAVINGS OFFERED BY THIS PROGRAM CAN BE DRAMATIC; A ONE-DIMENSIONAL ARRAY 400 IN LENGTH WILL BE TRANSFORMED IN 4000*400 = 14,400 SECONDS USING THE STRAIGHTFORWARD TECHNIQUE.

THE FAST-FOURIER TRANSFORM PLACES THREE RESTRICTIONS UPON THE DATA:
1. THE NUMBER OF INPUT DATA AND THE NUMBER OF TRANSFORM VALUES MUST BE THE SAME;
2. BOTH THE INPUT DATA AND THE TRANSFORM VALUES MUST REPRESENT EQUISPACED POINTS IN THEIR RESPECTIVE DOMAINS OF TIME AND FREQUENCY, CALLING THESE SPACEINGS DELTAT AND DELTAF, IT MUST BE TRUE THAT DELTAF=2*PI/NN(1)*DELTAQ, OF COURSE, DELTAT NEED NOT BE THE SAME FOR EVERY DIMENSION;
3. CONCEPTUALLY AT LEAST, THE INPUT DATA AND THE TRANSFORM OUTPUT REPRESENT SINGLE CYCLES OF PERIODIC FUNCTIONS;

THE CALLING SEQUENCE IS:
CALL FOURT(DATA,NN,NDIM,ISIGN,IFORM,WORK)

DATA IS THE ARRAY USED TO HOLD THE REAL AND IMAGINARY PARTS OF THE DATA ON INPUT AND THE TRANSFORM VALUES ON OUTPUT; IT IS A MULTIDIMENSIONAL FLOATING POINT ARRAY, WITH THE REAL AND IMAGINARY PARTS OF A DATUM STORED IMMEDIATELY ADJACENT IN STORAGE (SUCH AS FORTRAN IV PLACES THEM), NORMAL FORTRAN ORDERING IS...
EXPECTED: THE FIRST SUBSCRIPT CHANGING FASTEST, THE DIMENSIONS ARE GIVEN IN THE INTEGER ARRAY NN(i) OF LENGTH NDIM. A SIGN IS +1 TO INDICATE A FORWARD TRANSFORM (EXPONENTIAL SIGN IS +1) AND -1 FOR AN INVERSE TRANSFORM (SIGN IS -1). IF IFORM IS -1, IF THE DATA ARE COMPLEX, 0 IF THE DATA ARE REAL, IF IT IS 0, THE IMAGINARY PARTS OF THE DATA MUST BE SET TO ZERO; AS EXPLAINED ABOVE, THE TRANSFORM VALUES ARE ALWAYS COMPLEX AND ARE STORED IN ARRAY DATA. WORK IS AN ARRAY USED FOR WORKING STORAGE, IT IS FLOATING POINT REAL; ONE DIMENSIONAL OF LENGTH EQUAL TO TWICE THE LARGEST ARRAY DIMENSION NN(1) THAT IS NOT A POWER OF TWO. IF ALL NN(i) ARE POWERS OF TWO, IT IS NOT NEEDED AND MAY BE REPLACED BY ZERO IN THE CALLING SEQUENCE. THIS, FOR A ONE-DIMENSIONAL ARRAY, NN(1) ODD, WORK OCCUPIES AS MANY STORAGE LOCATIONS AS DATA; IF SUPPLIED, WORK MUST NOT BE THE SAME ARRAY AS DATA; ALL SUBSCRIPTS OF ALL ARRAYS BEGIN AT ONE.

EXAMPLE 1. THREE-DIMENSIONAL FORWARD FOURIER TRANSFORM OF A COMPLEX ARRAY DIMENSIONED 32 BY 25 BY 13 IN FORTRAN IV.

DIMENSION DATA(32,25,13), WORK(50) : NN(3)

COMPLEX DATA
DATA NN/32,25,13/
DO 1 I=1,32
DO 1 J=1,25
DO 1 K=1,13
1 DATA(I,J,K):COMPLEX VALUE
CALL FOUR ? DATA, NN(3), IFORM, WORK

EXAMPLE 2. ONE-DIMENSIONAL FORWARD TRANSFORM OF A REAL ARRAY OF LENGTH 64 IN FORTRAN II.

DIMENSION DATA(2,64)
DO 2 I=1,64
2 DATA(I):REAL PART
CALL FOUR ? DATA, 64, IFORM, WORK

THERE ARE NO ERROR MESSAGES OR ERROR HALTS IN THIS PROGRAM. THE PROGRAM RETURNS IMMEDIATELY IF NDIM OR ANY NN(i) IS LESS THAN ONE.

PROGRAM BY NORMAN BRENNER FROM THE BASIC PROGRAM BY CHARLES RADER, JUNE 1967. THE IDEA FOR THE DIGIT REVERSAL WAS SUGGESTED BY RALPH ALTER.

THIS IS THE FASTEST AND MOST VERSATILE VERSION OF THE FFT KNOWN TO THE AUTHOR. A PROGRAM CALLED FOUR2 IS AVAILABLE THAT ALSO PERFORMS THE FAST FOURIER TRANSFORM AND IS WRITTEN IN USASI BASIC FORTRAN. IT IS ABOUT ONE THIRD AS LONG AND REQUIRES THE DIMENSIONS OF THE INPUT ARRAY WHICH MUST BE COMPLEX TO BE POWERS OF TWO; ANOTHER PROGRAM, CALLED FOUR3, IS ONE TENTH AS LONG AND RUNS TWO THIRDS AS FAST ON A ONE-DIMENSIONAL COMPLEX ARRAY WHOSE LENGTH IS A POWER OF TWO.

REFERENCE--

IEEE AUDIO TRANSACTIONS (JUNE 1967), SPECIAL ISSUE ON THE FFT,

DIMENSION DATA(1), NN(1), IFACT(32), WORK(1)

N=6, 2, 3, 3, 1, 9, 5, 3, 7
W=7, 6, 8, 1, 0, 6, 7, 8, 1

1 NDIM=2
DO 2 I=1, NDIM
2 NDIM=NDIM/2
MAIN LOOP FOR EACH DIMENSION

DO 910 IDIM=1,NDIM
  N=NN(IDIM)
  IF(N=1)GO TO 900
910 CONTINUE

IS N A POWER OF TWO AND IF NOT, WHAT ARE ITS FACTORS

MAN
  NTWO=NP1
  IF=1
  IDIV=2
  IQUOT=M/IDIV
  RHM=M-IDIV*IQUOT
  IF(RHM<>0)G0 TO 11
  BALL
  IF=IF*1
  IF=IF*1
  IDIV=3
  NONZ=IF
  IQUOT=M/IDIV
  RHM=M-IDIV*IQUOT
  IF(RHM<0)G0 TO 33
  IQUOT=M/IDIV
  IDIV=4
  NONZ=IF
  IQUOT=M/IDIV
  IF(RHM<0)G0 TO 33
  IQUOT=M/IDIV
  IDIV=5
  NONZ=IF
  IQUOT=M/IDIV
  IF(RHM<0)G0 TO 33
  GO TO 30
30 CONTINUE

SEPARATE FOUR CASES:

1. COMPLEX TRANSFORM OR REAL TRANSFORM FOR THE 4TH, 9TH, ETC.
   DIMENSIONS;
2. REAL TRANSFORM FOR THE 2ND OR 3RD DIMENSION, METHOD=
   TRANSFORM HALF THE DATA, SUPPLYING THE OTHER HALF BY CONJUGATE SYMMETRY;
3. REAL TRANSFORM FOR THE 1ST DIMENSION, N ODD, METHOD=
   SET THE IMAGINARY PARTS TO ZERO;
4. REAL TRANSFORM FOR THE 1ST DIMENSION, N EVEN, METHOD=
   TRANSFORM A COMPLEX ARRAY OF LENGTH N/2 WHOSE REAL PARTS ARE THE EVEN NUMERATED REAL VALUES;
   AND WHOSE IMAGINARY PARTS ARE THE ODD NUMERATED REAL VALUES; SEPARATE AND SUPPLY THE SECOND HALF BY CONJUGATE SYMMETRY;

70 M=1
   IF(N=1)
   IQUOT=M/IDIV
   IF(RHM<0)G0 TO 33
   IQUOT=M/IDIV
   IDIV=4
   IQUOT=M/IDIV
   IF(RHM<0)G0 TO 33
   IQUOT=M/IDIV
   GO TO 70
SHUFFLE DATA BY BIT REVERSAL SING N=2^M, AS THE SHUFFLING CAN BE DONE BY SIMPLE INTERCHANGE, NO WORKING ARRAY IS NEEDED

C

100 IF(NTH=NPR)200 J=1,J10
110 N=2HF=NPR/2
120 DO 130 J=1,NPR,NPR+1
130 IF(J=1)GOTO 120
140 IF(NPR=NPR+1)GOTO 120
150 IF(N=J)GOTO 120
160 IF(NPR=NPR+1)GOTO 120
200 IF(N=J)GOTO 120

SHUFFLE DATA BY DIGIT REVERSAL FOR GENERAL N

C

200 N=WORK=N
270 DO 280 J=2,J,2
280 IF(J%)GOTO 270
210 IF(J=1)GOTO 270
220 WORK(J)=DATA(J)
230 IF(J%)=20
240 IF(J=1)GOTO 230
250 IF(J%)=20
260 IF(J=1)GOTO 230
270 IF(J%)=20
280 IF(J=1)GOTO 230
290 IF(J%)=20
300 IF(J=1)GOTO 230
310 IF(J%)=20
320 IF(J=1)GOTO 230
330 IF(J%)=20
340 IF(J=1)GOTO 230
MAIN LOOP FOR FACTORS OF TWO: PERFORM FOURIER TRANSFORMS OF LENGTH FOUR, WITH ONE OF LENGTH TWO IF NEEDED. THE TWIDDLE FACTOR \( w_{k} = \exp(-2\pi i / \text{length}) \). CHECK FOR \( i \text{SIGN} = \text{SORT}(i) \) AND REPEAT FOR \( w_{k} = \exp(i \text{SIGN} \text{SORT}(i) / \text{SORT}(2) \).
U21 = DATA(K3) = DATA(K4)
U3R = DATA(K1) = DATA(K2)
U3I = DATA(K1) = DATA(K2)
IF(ISIGN) = 471, 472, 472
471
U4R = DATA(K3) = DATA(K4)
U4I = DATA(K4) = DATA(K3)
GO TO 510
472
U4R = DATA(K4) = DATA(K3)
U4I = DATA(K3) = DATA(K4)
GO TO 510
480
T2R = W2R = DATA(K2) = W2I = DATA(K2)
T2I = W2R = DATA(K2) = W2I = DATA(K2)
T3R = W3R = DATA(K3) = W3I = DATA(K3)
T3I = W3R = DATA(K3) = W3I = DATA(K3)
T4R = W4R = DATA(K4) = W4I = DATA(K4)
T4I = W4R = DATA(K4) = W4I = DATA(K4)

U1R = DATA(K1) = T2R
U1I = DATA(K1) = T2I
U2R = T2R = T4R
U2I = T2I = T4I
U3R = DATA(K1) = T2R
U3I = DATA(K1) = T2I
IF(ISIGN) = 490, 500, 500
490
U4R = T3I = T4I
U4I = T4I = T3I
GO TO 510
500
U4R = T3I = T4I
U4I = T3I = T4I
510
DATA(K1) = U1R = U2R
DATA(K1) = U1I = U2I
DATA(K2) = U3R = U4R
DATA(K2) = U3I = U4I
DATA(K3) = U1R = U2R
DATA(K3) = U1I = U2I
DATA(K4) = U3R = U4R
DATA(K4) = U3I = U4I
KDF = KSTEP
KMIN = 4*(KMIN = 1) + 1
GO TO 520
520
DATA(K1) = U1R = U2R
DATA(K1) = U1I = U2I
DATA(K2) = U3R = U4R
DATA(K2) = U3I = U4I
DATA(K3) = U1R = U2R
DATA(K3) = U1I = U2I
DATA(K4) = U3R = U4R
DATA(K4) = U3I = U4I
530 CONTINUE
MNUM = LMAX
IF(M = MMAX) = 540, 540, 570
540 IF(ISIGN) = 550, 560, 560
550 TEMPREW
WR = (WR + 1) = RTHLF
WT = (WT + TEMPRE) = RTHLF
GO TO 410
560 TEMPREW
WR = (WR + 1) = RTHLF
WT = (TEMPRE + 1) = RTHLF
GO TO 410
570 CONTINUE
IPAR = 3 = IPAR
MAX = MMAX = MMAX
GO TO 360
C MAIN LOOP FOR FACTORS NOT EQUAL TO TWO, APPLY THE TWIDDLE FACTOR
WHEN THE SIGN OF F = 1, 0, -1, THEN
PERFORM A FOURIER TRANSFORM OF LENGTH IFACT(IF), MAKING USE OF
CONJUGATE SYMMETRIES.
600 IF(NTO = NPAR) = 605, 700, 700
IF=INON2
NPH=NPH1/2
610 IFP2=IFACT(IF)*IFP1
615 J1MIN=NPH1
620 IF(J1MIN=IFP1)630,640,640
625 DO 630 J1=J1MIN,NPH1,NPH2
630 THEETA=TWOP/FLOAT(J1-1)/FLOAT(IFP2)
635 IF(ISIGN)625,620,680
640 THE eta=THETA
645 WSTPR=COS(THETA)
650 WSTP1=SIN(THETA)
655 WSTP=WSTP1
660 J2MIN=J1=IFP1
665 J2MAX=J1+1=IFP2
670 DO 630 J2=J2MIN,J2MAX,IFP1
675 J1MAX=J2+1=J2 RN0=E
680 DO 630 J3=J2+1=J3MAX,E
685 J3MAX=J2+1=J3 RN0=2
690 DATA(J3)=DATA(J3)+WR=DATA(J3)+1=W
695 TEM PR=DATA(J3)
700 WSTPR=WR
705 WSTP=WR
710 J1INT0T=J1,IFP1
715 DO 695 J2=J1INT0T,IFP1
720 DATA(J2)=DATA(J2)+WR=DATA(J2)*WR
725 WSTP=WSTP+WR=WR
730 WSTPR=WSTPR+WR=WR
735 THE eta=TWOP/FLOAT(IFP2)
740 IF(ISIGN)635,630,680
745 THE eta=THETA
750 WSTPR=COS(THETA)
755 WSTP=SIN(THETA)
760 J2RN0=IFP1+(1=FACT(IF)/2)
765 DO 695 J2=J2RN0,E
770 J2MAX=J2+1=J2 RN0,2
775 DO 630 J3=J2+1=J3MAX,IFP2
780 J3MAX=J2+1=J3 RN0=1
785 DATA(J3)=DATA(J3)+WR=DATA(J3)+1=W
790 TEM PR=DATA(J3)
795 WSTPR=WR
800 WSTP=WR
805 J3INT0T=J3,IFP2
810 DO 695 J2=J3INT0T,IFP2
815 DATA(J2)=DATA(J2)+WR=DATA(J2)*WR
820 WSTP=WSTP+WR=WR
825 WSTPR=WSTPR+WR=WR
830 THE eta=TWOP/FLOAT(IFP2)
835 IF(ISIGN)635,630,680
840 THE eta=THETA
845 SUMR=0
850 SUH=0
855 DO 660 J1=J1MIN,J1MAX,IFP1
860 SUH=SUH+DATA(J)
865 SUM=SUH+DATA(J+1)
870 WORK(J)=SUMR
875 WORK(J+1)=SUM
880 DO TO 680
885 JONUM1=1=IFP2=2=J2=13=J31=NP5HF
890 J2MAX
895 SUH=DATA(J)
900 SUM=DATA(J+1)
905 OLDSR=0
910 OLDS=0
915 J2=IFP2
920 TEM P=SUMR
COMPLETE A REAL TRANSFORM IN THE $1ST$ DIMENSION, $N$ EVEN, BY CON-
JUGATE SYMMETRIES,

GO TO 700

CASE

GO TO 700, 800, 900, 701, 702, 703, 704

DO 720 JMIN, JMAX, NHP2

SUMR = DATA(I) + DATA(J) / 2,
SUMI = DATA(I+1) + DATA(J+1) / 2,
DIFR = DATA(I+1) - DATA(J+1) / 2,

TEMPR = SUMR * DirR
TEMP = SUMI * DirI

DATA(I) = SUMR * TEMPR
DATA(I+1) = DIFR * TEMPE
DATA(J) = SUMR * TEMPI
DATA(J+1) = DIFR * TEMPI

720 JMIN = JMAX / 2

STOP
IF(1=IMAX)780,760,760
DATA(J)=DATA(IMIN)=DATA(IMAX+1)
DATA(J)=DATA(J)+0;
IF(J=1)775,775,770
DATA(J)=DATA(J)
DATA(J)=DATA(J)+0;
IMAX=IMIN
GO TO 745
DATA(J)=DATA(J)+DATA(2)
GO TO 740
C
COMPLETE A REAL TRANSFORM FOR THE 2ND OR 3RD DIMENSION BY
C
CONJUGATE SYMMETRIZING,
C
DO 880 13=1,NTOT,NP2
3MAX=13*NP2
DO 880 13=1,3*NP2
IMIN=13*NR1
IMAX=13*NP1
I3MAX=213*NP3=1
IF(I=I3MAX)830,830,830
I3MAX=I3MAX+NP2
820 IF(I3MAX)850,850,850
830 IF(I3MAX)890,890,890
DO 840 14=1,MIN,M2
DATA(J)=DATA(J)
DATA(J)=DATA(J)+1
840 IMAX=M3
850 IF(I=IMAX)890,890,890
DATA(J)=DATA(J)
DATA(J)=DATA(J)+1
880 IMAX=M3
C       END OF LOOP ON EACH DIMENSION
C
900    NR0=NP1
       NP1=NP2
910    NPREV=N
920    RETURN
END
### Abstract

Three programs are described and listed, all written in USASI Basic Fortran, which perform the discrete Fourier transform upon a multidimensional array of floating point data. The data may be either real or complex, with a savings in running time for real over complex. The transform values are always complex and are returned in the array used to carry the original data. The running time is much shorter than that of any program performing a direct summation, even when sine and cosine values are precalculated and stored in a table. For example, on a CDC 3300 with floating point add time of six microseconds, a complex array of size 80 x 80 can be transformed in 19.2 seconds. Besides the main array, only a working storage array of size 160 need be supplied.