ESD-TR-67-462

Technical Note 1967-2

N. M. Brenner

Three Fortran Programs that Perform the Cooley-Tukey Fourier Transform

28 July 1967

Prepared under Electronic Systems Division Contract AF 19(628)-5167 by

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lexington, Massachusetts

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.

ESD-TR-67-462

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.

7 September 1967

Publications
M.I.T. Lincoln Laboratory
P. O. Box 73
Lexington, Massachusetts 02173

```
      THE FOLLOWING THREE PATTERNS OCCUR FREQUENTLY.
      BR=WR*AR-WI*AI
      BI=AI*WR+AR*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

      P. 15, L. 7
7     ISTEP=2*MMAX

      P. 21, L. 2 AND P. 17, L. 2
2     NTOT=NTOT*NN(IDIM)

      P. 22, L. 5-2 AND P. 17, L. 100-2
      NP2=NP1*N

      P. 22, L. 12 AND L. 51
12 OR 51   NTWO=NTWO+NTWO

      P. 22, L. 70+2
      I1RNG=NP1
      IF(IDIM-4)71,100,100

      P. 23, L. 72+1
      I1RNG=NP0*(1+NPREV/2)

      P. 23, L. 120 AND P. 17, L.110
110 OR 120   I1MAX=I2+NP1-2

      P. 23, L. 120+3 AND P. 17, L. 110+3
      J3=J+I3-I2

      P. 23, L. 200
200   NWORK=2*N

      P. 23, L.210-1
      IF(ICASE-3)210,220,210

      P. 23, L. 240+1
      J=J+IFP1
      IF(J-I3-IFP2)260,250,250

      P. 24, L. 420+1 AND P. 18, L. 420+1
      KMIN=IPAR*M+I1

      P. 24, L. 440 AND P. 18, L. 440
440   KDIF=IPAR*MMAX
450   KSTEP=4*KDIF

      P. 24, L. 520+1 AND P. 18, L. 520+1
      KMIN=4*(KMIN-I1)+I1
      KDIF=KSTEP
      IF(KDIF-NP2HF)450,450,530

      P. 25, L. 550+1 AND P. 19, L. 550+1
      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
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
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
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
```

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

THREE FORTRAN PROGRAMS THAT PERFORM THE COOLEY-TUKEY FOURIER TRANSFORM

N. M. BRENNER

Group 31

TECHNICAL NOTE 1967-2

28 JULY 1967

LEXINGTON MASSACHUSETTS

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 × 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 × 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 DATA(I1,I2,...),

$$\text{TRANSFORM}(J1,J2,\ldots) = \Sigma\,[\text{DATA}(I1,I2,\ldots)\ W1^{(I1-1)(J1-1)}\ W2^{(I2-1)(J2-1)}\ldots]\ ,$$

where $W1 = \exp(-2\pi i/N1)$, $W2 = \exp(-2\pi i/N2),\ldots$ and I1 and J1 run from 1 to N1, I2 and J2 run from 1 to N2, 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

$$F(y) = \int_{-\infty}^{\infty} f(x)\ e^{-2\pi ixy}\ dx\ .$$

By interpreting the subscripts modulo N1, N2, etc. 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 N1*N2*... .

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:

N	Factorization	Time for Complex Transform (sec)
4094	$2 \times 23 \times 89$	80
4095	$3^2 \times 5 \times 7 \times 13$	24
4096	2^{12}	6.2
4097	17×241	180
4098	$2 \times 3 \times 683$	480
4099	prime	2868
4100	$2^2 \times 5^2 \times 41$	39

Calling Sequence

The listings of three programs are given in the appendices. FOUR1 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. FOUR1 is much slower than FOURT or FOUR2, and performs only one-dimensional transforms on complex arrays whose lengths are powers of two. FOUR1 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 ($0.5^{\frac{1}{2}}$). 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

$$t_0 = d_0 + d_1 + d_2 + d_3 + d_4 + d_5$$
$$t_1 = d_0 + wd_1 + w^2d_2 + w^3d_3 + w^4d_4 + w^5d_5$$
$$t_2 = d_0 + w^2d_1 + w^4d_2 + w^6d_3 + w^8d_4 + w^{10}d_5$$

$$t_3 = d_0 + w^3d_1 + w^6d_2 + w^9d_3 + w^{12}d_4 + w^{15}d_5$$
$$t_4 = d_0 + w^4d_1 + w^8d_2 + w^{12}d_3 + w^{16}d_4 + w^{20}d_5$$
$$t_5 = d_0 + w^5d_1 + w^{10}d_2 + w^{15}d_3 + w^{20}d_4 + w^{25}d_5$$

Straightforward computation requires 25 complex multiplications and 30 complex additions. The fast Fourier transform computes as follows:

$$u_0 = d_0 + d_3$$
$$u_1 = d_0 + w^3d_3$$
$$u_2 = d_1 + d_4$$
$$u_3 = d_1 + w^3d_4$$
$$u_4 = d_2 + d_5$$
$$u_5 = d_2 + w^3d_5$$
$$t_0 = u_0 + u_2 + u_4$$
$$t_1 = u_1 + wu_3 + w^2u_5$$
$$t_2 = u_0 + w^2u_2 + w^4u_4$$
$$t_3 = u_1 + w^3u_3 + w^6u_5$$
$$t_4 = u_0 + w^4u_2 + w^8u_4$$
$$t_5 = u_1 + w^5u_3 + w^{10}u_5$$

which requires only 13 complex multiplications and 18 complex additions. Note that $w^3 = -1$ and $w^6 = 1$.

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 2π(N-1)/(Ndt) at interval 2π/(Ndt) as I runs from 1 to N. By periodicity, the upper limit is identified with -2π/(Ndt) and in fact all points above the "foldover frequency" π/(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_{total} be the total number of points in the data array. That is, N_{total} = N1*N2*.... Decompose N_{total} into its prime factors, such as $2^{K2}3^{K3}5^{K5}$ .... Let Σ_2 be the sum of all the factors of two in N_{total}, that is, $\Sigma_2 = 2*K2$. Let Σ_f be the sum of all the other factors, $\Sigma_f = 3*K3 + 5*K5 + \ldots$. The time taken for a multidimensional transform is

$$T = T_0 + N_{total}\,[T_1 + T_2\Sigma_2 + T_f\Sigma_f] \, .$$

For the CDC 3300,

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

The greater optimization apparent for factors of two is due to

1. The eight-fold symmetry of the trigonometric functions from 0 to 2π.
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 (IFORM = 0) can reduce running time by as much as forty percent. On the CDC 3300, a 64 × 64 complex array was transformed in

6.1 seconds; a 64×64 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

$$1.06\ N_{total}^{\frac{1}{2}}\ 2^{-b}\ \Sigma_j [2f_j]^{3/2}$$

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

Further error is introduced in this particular program by the use of recursive generation of sines and cosines for factors of N_{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

$$(f*g)_k = \Sigma_j\ f_j g_{k-j}\ ,$$

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 × 40 array in Fortran II

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

2. Same example as 1, but in Fortran IV

```
      DIMENSION DATA (50,40), WORK (100), NN (2)
      COMPLEX DATA
      DATA NN/50, 40/
      DO 1  I = 1, 50
      DO 1  J = 1, 40
   1  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:

```
      DOUBLE PRECISION DATA, WORK
```

   Change the call to:

```
      CALL DFOURT (DATA,NN,2,-1,1,WORK)
```

4. Inverse transform of real 64 × 32 array in Fortran IV

```
      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 × 32 array in Fortran IV

```
      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

```
      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.

BIBLIOGRAPHY

1. Cooley, J.W. and Tukey, J.W., "An Algorithm for the Machine Calculation of Complex Fourier Series," Mathematics of Computation, 19, 90 (April 1965), 297-301.

2. Danielson, G.C. and Lanczos, C., "Some Improvements in Practical Fourier Analysis and Their Application to X-ray Scattering from Liquids," Journal of the Franklin Institute, 233, April 1942, 365-380, and May 1942, 435-452.

3. Gentleman, W.M. and Sande, G., "Fast Fourier Transforms — for Fun and Profit," AFIPS, 29, 1966 Fall Joint Computer Conference, Spartan Books, Washington 1966.

4. Good, I.J., "The Interaction Algorithm and Practical Fourier Analysis," Journal of the Royal Statistical Society, B, 20 (1958), 361-373, 22 (1960), 372-375.

5. Stockham, T., "High-speed Convolution and Correlation," AFIPS, 28, 1966 Spring Joint Computer Conference, Spartan Books, Washington.

6. Runge, C., Zeitschrift für Math. und Physik., 48 (1903), 443, 52 (1905), 117.

7. Runge, C. and König, H., Vorlesungen über Numerisches Rechnen, 211-231, Springer, Berlin, 1924.

8. Wittaker, E.T. and Robinson, G., The Calculus of Observations, 260-284, Blackie and Son, London 1944.

9. Yates, F., The Design and Analysis of Factorial Experiments, 15, Commonwealth Agricultural Bureaux, Farnham Royal, 1937.

10. Box, G.E.P., Connor, L.R., Cousins, W.R., Davies, O.L. (Ed.), Himsworth, F.R., and Sillitto, G.P., The Design and Analysis of Industrial Experiments, 363-366, Oliver and Boyd, London, 1954.

11. Cooley, J.W., Miller, R.E., and Winograd, S., "Harmonic Analyzer," IBM Watson Research Center, Yorktown Heights, 1963.

12. Bergland, G.D. and Hale, H.W., "Digital Real-time Spectral Analysis," IEEE Trans. on Electronic Computers, EC-16 (April 1967), No. 2, 180-185.

13. Bowles, K.L., "Signal Processing — An Untapped Major Market for the Computer Industry," University of California, San Diego, 1967.

14. Rader, C., Lincoln Laboratory Memorandum, 1965.

15. Good, I.J., "Random Motion on a Finite Abelian Group," Proc. Camb. Phil. Soc., 47 (1951), 756-762.

16. Goertzel, G., "An Algorithm for the Evaluation of Finite Trigonometric Series," The American Mathematical Monthly, 65 (1958), 34-35.

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 regrouping 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

$$t_j = \Sigma\, d_i (-1)^{i_0 j_0 + i_1 j_1 + \cdots}$$

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 ([4], 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.

Appendix III

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 FOUR1(DATA,NN,ISIGN)
C     THE COOLEY-TUKEY FAST FOURIER TRANSFORM IN USASI BASIC FORTRAN
C     TRANSFORM(J) = SUM(DATA(I)*W**((I-1)*(J-1))), WHERE I AND J RUN
C     FROM 1 TO NN AND W = EXP(ISIGN*2*PI*SQRT(-1)/NN).  DATA IS A ONE-
C     DIMENSIONAL COMPLEX ARRAY (I.E., THE REAL AND IMAGINARY PARTS OF
C     THE DATA ARE LOCATED IMMEDIATELY ADJACENT IN STORAGE, SUCH AS
C     FORTRAN IV PLACES THEM) WHOSE LENGTH NN IS A POWER OF TWO.  ISIGN
C     IS +1 OR -1, GIVING THE SIGN OF THE TRANSFORM.  TRANSFORM VALUES
C     ARE RETURNED IN ARRAY DATA, REPLACING THE INPUT DATA.  THE TIME IS
C     PROPORTIONAL TO N*LOG2(N), RATHER THAN THE USUAL N**2.  WRITTEN BY
C     NORMAN BRENNER, JUNE 1967.  THIS IS THE SHORTEST VERSION
C     OF FFT KNOWN TO THE AUTHOR, AND IS INTENDED MAINLY FOR
C     DEMONSTRATION.  PROGRAMS FOUR2 AND FOURT ARE AVAILABLE THAT RUN
C     TWICE AS FAST AND OPERATE ON MULTIDIMENSIONAL ARRAYS WHOSE
C     DIMENSIONS ARE NOT RESTRICTED TO POWERS OF TWO.  (LOOKING UP SINES
C     AND COSINES IN A TABLE WILL CUT RUNNING TIME OF FOUR1 BY A THIRD.)
C     SEE-- IEEE AUDIO TRANSACTIONS (JUNE 1967), SPECIAL ISSUE ON FFT.
      DIMENSION DATA(1)
      N=2*NN
      J=1
      DO 5 I=1,N,2
      IF(I-J)1,2,2
1     TEMPR=DATA(J)
      TEMPI=DATA(J+1)
      DATA(J)=DATA(I)
      DATA(J+1)=DATA(I+1)
      DATA(I)=TEMPR
      DATA(I+1)=TEMPI
2     M=N/2
3     IF(J-M)5,5,4
4     J=J-M
      M=M/2
      IF(M-2)5,3,3
5     J=J+M
      MMAX=2
6     IF(MMAX-N)7,9,9
7     ISTEP=2*MMAX
      DO 8 M=1,MMAX,2
      THETA=3.1415926535*FLOAT(ISIGN*(M-1))/FLOAT(MMAX)
      WR=COS(THETA)
      WI=SIN(THETA)
      DO 8 I=M,N,ISTEP
      J=I+MMAX
      TEMPR=WR*DATA(J)-WI*DATA(J+1)
      TEMPI=WR*DATA(J+1)+WI*DATA(J)
      DATA(J)=DATA(I)-TEMPR
      DATA(J+1)=DATA(I+1)-TEMPI
      DATA(I)=DATA(I)+TEMPR
8     DATA(I+1)=DATA(I+1)+TEMPI
      MMAX=ISTEP
      GO TO 6
9     RETURN
      END
      SUBROUTINE FOUR2(DATA,NN,NDIM,ISIGN)
C
C     THE COOLEY-TUKEY FAST FOURIER TRANSFORM IN USASI BASIC FORTRAN
C
C     TRANSFORM(J1,J2,,,,) = SUM(DATA(I1,I2,,,,)*W1**((I1-1)*(J1-1))
C                                 *W2**((I2-1)*(J2-1))*,,,),
C     WHERE I1 AND J1 RUN FROM 1 TO NN(1) AND W1=EXP(ISIGN*2*PI*
C     SQRT(-1)/NN(1)), ETC,
C
C     DATA IS A MULTIDIMENSIONAL FLOATING POINT ARRAY ALL OF WHOSE
C     DIMENSIONS ARE POWERS OF TWO,  THE LENGTH OF EACH DIMENSION IS
C     STORED IN THE INTEGER ARRAY NN, OF LENGTH NDIM,  ISIGN IS
C     +1 OR -1, GIVING THE SIGN OF THE TRANSFORM,  THE REAL
C     AND IMAGINARY PARTS OF A DATUM ARE IMMEDIATELY ADJACENT IN STORAGE
C     (SUCH AS FORTRAN IV PLACES THEM),  TRANSFORM RESULTS ARE RETURNED
C     IN ARRAY DATA, REPLACING THE ORIGINAL DATA,  TIME IS PROPORTIONAL
C     TO N*LOG2(N), RATHER THAN THE USUAL N**2,  NOTE THAT IF A FORWARD
C     TRANSFORM IS FOLLOWED BY AN INVERSE TRANSFORM, THE ORIGINAL DATA
C     WILL REAPPEAR MULTIPLIED BY NN(1)*NN(2)*,,,,  EXAMPLE--
C     FORWARD FOURIER TRANSFORM OF A TWO-DIMENSIONAL ARRAY IN FORTRAN II
C     DIMENSION DATA(2,64,32),NN(2)
C     NN(1)=64
C     NN(2)=32
C     DO 1 I=1,64
C     DO 1 J=1,32
C     DATA(1,I,J)=REAL PART
C   1 DATA(2,I,J)=IMAGINARY PART
C     CALL FOUR2(DATA,NN,2,-1)
C
C     SAME EXAMPLE IN FORTRAN IV
C     DIMENSION DATA(64,32),NN(2)
C     COMPLEX DATA
C     DATA NN/64,32/
C     DO 1 I=1,64
C     DO 1 J=1,32
C   1 DATA(I,J)=COMPLEX VALUE
C     CALL FOUR2(DATA,NN,2,-1)
C
C     PROGRAM BY NORMAN BRENNER FROM THE BASIC PROGRAM BY CHARLES
C     RADER,  MAY 1967,  THE IDEA FOR THE DIGIT REVERSAL WAS SUGGESTED
C     BY RALPH ALTER,
C
C     THIS VERSION OF THE FAST FOURIER TRANSFORM IS THE FASTEST KNOWN
C     TO THE AUTHOR,  LOOKING UP SINES AND COSINES IN A TABLE INSTEAD OF
C     COMPUTING THEM WOULD DECREASE RUNNING TIME SEVEN PERCENT,
C     PROGRAMS FOURT AND FOUR1 ARE AVAILABLE FROM THE AUTHOR THAT ALSO
C     PERFORM THE FAST FOURIER TRANSFORM AND ARE WRITTEN IN USASI BASIC
C     FORTRAN,  FOURT IS THREE TIMES AS LONG, IS NOT RESTRICTED TO
C     POWERS OF TWO, AND RUNS UP TO FORTY PERCENT FASTER ON REAL DATA,
C     FOUR1 IS ONE FOURTH AS LONG, ONE HALF AS FAST, AND IS RESTRICTED
C     TO ONE DIMENSION AND POWERS OF TWO,
C
C     SEE-- IEEE AUDIO TRANSACTIONS (JUNE 1967), SPECIAL ISSUE ON FFT,
C
      DIMENSION DATA(1),NN(1)
      IF(NDIM-1)700,1,1
1     NTOT=2
      DO 2 IDIM=1,NDIM
      IF(NN(IDIM))700,700,2
```

```
2      NTOT=NTOT*NN(IDIM)
       RTHLF=.70710 67812
       TWOPI=6.28318 53070
C
C      MAIN LOOP FOR EACH DIMENSION
C
       NP1=2
       DO 600 IDIM=1,NDIM
       N=NN(IDIM)
       NP2=NP1*N
       IF(N-1)700,600,100
C
C      SHUFFLE DATA BY BIT REVERSAL, SINCE N=2**K.  AS THE SHUFFLING
C      CAN BE DONE BY SIMPLE INTERCHANGE, NO WORKING ARRAY IS NEEDED
C
100    NP2HF=NP2/2
       J=1
       DO 160 I2=1,NP2,NP1
       IF(J-I2)110,130,130
110    I1MAX=I2+NP1-2
       DO 120 I1=I2,I1MAX,2
       DO 120 I3=I1,NTOT,NP2
       J3=J+I3-I2
       TEMPR=DATA(I3)
       TEMPI=DATA(I3+1)
       DATA(I3)=DATA(J3)
       DATA(I3+1)=DATA(J3+1)
       DATA(J3)=TEMPR
120    DATA(J3+1)=TEMPI
130    M=NP2HF
140    IF(J-M)160,160,150
150    J=J-M
       M=M/2
       IF(M-NP1)160,140,140
160    J=J+M
C
C      MAIN LOOP.  PERFORM FOURIER TRANSFORMS OF LENGTH FOUR, WITH ONE OF
C      LENGTH TWO IF NEEDED.  THE TWIDDLE FACTOR W=EXP(ISIGN*2*PI*
C      SQRT(-1)*M/(4*MMAX)).  CHECK FOR THE SPECIAL CASE W=ISIGN*SQRT(-1)
C      AND REPEAT FOR W=W*(1+ISIGN*SQRT(-1))/SQRT(2).
C
       NP1TW=NP1+NP1
       IPAR=N
310    IF(IPAR-2)350,330,320
320    IPAR=IPAR/4
       GO TO 310
330    DO 340 I1=1,NP1,2
       DO 340 K1=I1,NTOT,NP1TW
       K2=K1+NP1
       TEMPR=DATA(K2)
       TEMPI=DATA(K2+1)
       DATA(K2)=DATA(K1)-TEMPR
       DATA(K2+1)=DATA(K1+1)-TEMPI
       DATA(K1)=DATA(K1)+TEMPR
340    DATA(K1+1)=DATA(K1+1)+TEMPI
350    MMAX=NP1
360    IF(MMAX-NP2HF)370,600,600
370    LMAX=MAX0(NP1TW,MMAX/2)
       DO 570 L=NP1,LMAX,NP1TW
       M=L
       IF(MMAX-NP1)420,420,380
380    THETA=-TWOPI*FLOAT(M)/FLOAT(4*MMAX)
```

```
       IF(ISIGN)400,390,390
390    THETA=-THETA
400    WR=COS(THETA)
       WI=SIN(THETA)
410    W2R=WR*WR-WI*WI
       W2I=2.*WR*WI
       W3R=W2R*WR-W2I*WI
       W3I=W2R*WI+W2I*WR
420    DO 530 I1=1,NP1,2
       KMIN=IPAR*M+I1
       IF(MMAX-NP1)430,430,440
430    KMIN=I1
440    KDIF=IPAR*MMAX
450    KSTEP=4*KDIF
       DO 520 K1=KMIN,NTOT,KSTEP
       K2=K1+KDIF
       K3=K2+KDIF
       K4=K3+KDIF
       IF(MMAX-NP1)460,460,480
460    U1R=DATA(K1)+DATA(K2)
       U1I=DATA(K1+1)+DATA(K2+1)
       U2R=DATA(K3)+DATA(K4)
       U2I=DATA(K3+1)+DATA(K4+1)
       U3R=DATA(K1)-DATA(K2)
       U3I=DATA(K1+1)-DATA(K2+1)
       IF(ISIGN)470,475,475
470    U4R=DATA(K3+1)-DATA(K4+1)
       U4I=DATA(K4)-DATA(K3)
       GO TO 510
475    U4R=DATA(K4+1)-DATA(K3+1)
       U4I=DATA(K3)-DATA(K4)
       GO TO 510
480    T2R=W2R*DATA(K2)-W2I*DATA(K2+1)
       T2I=W2R*DATA(K2+1)+W2I*DATA(K2)
       T3R=WR*DATA(K3)-WI*DATA(K3+1)
       T3I=WR*DATA(K3+1)+WI*DATA(K3)
       T4R=W3R*DATA(K4)-W3I*DATA(K4+1)
       T4I=W3R*DATA(K4+1)+W3I*DATA(K4)
       U1R=DATA(K1)+T2R
       U1I=DATA(K1+1)+T2I
       U2R=T3R+T4R
       U2I=T3I+T4I
       U3R=DATA(K1)-T2R
       U3I=DATA(K1+1)-T2I
       IF(ISIGN)490,500,500
490    U4R=T3I-T4I
       U4I=T4R-T3R
       GO TO 510
500    U4R=T4I-T3I
       U4I=T3R-T4R
510    DATA(K1)=U1R+U2R
       DATA(K1+1)=U1I+U2I
       DATA(K2)=U3R+U4R
       DATA(K2+1)=U3I+U4I
       DATA(K3)=U1R-U2R
       DATA(K3+1)=U1I-U2I
       DATA(K4)=U3R-U4R
520    DATA(K4+1)=U3I-U4I
       KMIN=4*(KMIN-I1)+I1
       KDIF=KSTEP
       IF(KDIF-NP2HF)450,450,530
```

```
530   CONTINUE
      M=M+LMAX
      IF(M-MMAX)540,540,570
540   IF(ISIGN)550,560,560
550   TEMPR=WR
      WR=(WR+WI)*RTHLF
      WI=(WI-TEMPR)*RTHLF
      GO TO 410
560   TEMPR=WR
      WR=(WR-WI)*RTHLF
      WI=(TEMPR+WI)*RTHLF
      GO TO 410
570   CONTINUE
      IPAR=3-IPAR
      MMAX=MMAX+MMAX
      GO TO 360
600   NP1=NP2
700   RETURN
      END
      SUBROUTINE FOURT(DATA,NN,NDIM,ISIGN,IFORM,WORK)
C
C     THE COOLEY-TUKEY FAST FOURIER TRANSFORM IN USASI BASIC FORTRAN
C
C     TRANSFORM(J1,J2,,,,) = SUM(DATA(I1,I2,,,,)*W1**((I1-1)*(J1-1))
C                                 *W2**((I2-1)*(J2-1))*,,,),
C     WHERE I1 AND J1 RUN FROM 1 TO NN(1) AND W1=EXP(ISIGN*2*PI*
C     SQRT(-1)/NN(1)), ETC.  THERE IS NO LIMIT ON THE DIMENSIONALITY
C     (NUMBER OF SUBSCRIPTS) OF THE DATA ARRAY.  IF AN INVERSE
C     TRANSFORM (ISIGN=+1) IS PERFORMED UPON AN ARRAY OF TRANSFORMED
C     (ISIGN=-1) DATA, THE ORIGINAL DATA WILL REAPPEAR,
C     MULTIPLIED BY NN(1)*NN(2)*,,,  THE ARRAY OF INPUT DATA MUST BE
C     IN COMPLEX FORMAT.  HOWEVER, IF ALL IMAGINARY PARTS ARE ZERO (I.E.
C     THE DATA ARE DISGUISED REAL) RUNNING TIME IS CUT UP TO FORTY PER-
C     CENT.  (FOR FASTEST TRANSFORM OF REAL DATA, NN(1) SHOULD BE EVEN.)
C     THE TRANSFORM VALUES ARE ALWAYS COMPLEX, AND ARE RETURNED IN THE
C     ORIGINAL ARRAY OF DATA, REPLACING THE INPUT DATA.  THE LENGTH
C     OF EACH DIMENSION OF THE DATA ARRAY MAY BE ANY INTEGER.  THE
C     PROGRAM RUNS FASTER ON COMPOSITE INTEGERS THAN ON PRIMES, AND IS
C     PARTICULARLY FAST ON NUMBERS RICH IN FACTORS OF TWO.
C
C     TIMING IS IN FACT GIVEN BY THE FOLLOWING FORMULA.  LET NTOT BE THE
C     TOTAL NUMBER OF POINTS (REAL OR COMPLEX) IN THE DATA ARRAY, THAT
C     IS, NTOT=NN(1)*NN(2)*,,,  DECOMPOSE NTOT INTO ITS PRIME FACTORS,
C     SUCH AS 2**K2 * 3**K3 * 5**K5 * ,,,  LET SUM2 BE THE SUM OF ALL
C     THE FACTORS OF TWO IN NTOT, THAT IS, SUM2 = 2*K2.  LET SUMF BE
C     THE SUM OF ALL OTHER FACTORS OF NTOT, THAT IS, SUMF = 3*K3+5*K5+,,
C     THE TIME TAKEN BY A MULTIDIMENSIONAL TRANSFORM ON THESE NTOT DATA
C     IS T = T0 + NTOT*(T1+T2*SUM2+T3*SUMF).  ON THE CDC 3300 (FLOATING
C     POINT ADD TIME = SIX MICROSECONDS), T = 3000 + NTOT*(600+40*SUM2+
C     175*SUMF) MICROSECONDS ON COMPLEX DATA.
C
C     IMPLEMENTATION OF THE DEFINITION BY SUMMATION WILL RUN IN A TIME
C     PROPORTIONAL TO NTOT*(NN(1)+NN(2)+,,,).  FOR HIGHLY COMPOSITE NTOT
C     THE SAVINGS OFFERED BY THIS PROGRAM CAN BE DRAMATIC.  A ONE-DIMEN-
C     SIONAL ARRAY 4000 IN LENGTH WILL BE TRANSFORMED IN 4000*(600+
C     40*(2+2+2+2+2)+175*(5+5+5)) = 14.5 SECONDS VERSUS ABOUT 4000*
C     4000*175 = 2800 SECONDS FOR THE STRAIGHTFORWARD TECHNIQUE.
C
C     THE FAST FOURIER TRANSFORM PLACES THREE RESTRICTIONS UPON THE
C     DATA.
C     1.  THE NUMBER OF INPUT DATA AND THE NUMBER OF TRANSFORM VALUES
C     MUST BE THE SAME.
C     2.  BOTH THE INPUT DATA AND THE TRANSFORM VALUES MUST REPRESENT
C     EQUISPACED POINTS IN THEIR RESPECTIVE DOMAINS OF TIME AND
C     FREQUENCY.  CALLING THESE SPACINGS DELTAT AND DELTAF, IT MUST BE
C     TRUE THAT DELTAF=2*PI/(NN(I)*DELTAT).  OF COURSE, DELTAT NEED NOT
C     BE THE SAME FOR EVERY DIMENSION.
C     3.  CONCEPTUALLY AT LEAST, THE INPUT DATA AND THE TRANSFORM OUTPUT
C     REPRESENT SINGLE CYCLES OF PERIODIC FUNCTIONS.
C
C     THE CALLING SEQUENCE IS--
C     CALL FOURT(DATA,NN,NDIM,ISIGN,IFORM,WORK)
C
C     DATA IS THE ARRAY USED TO HOLD THE REAL AND IMAGINARY PARTS
C     OF THE DATA ON INPUT AND THE TRANSFORM VALUES ON OUTPUT.  IT
C     IS A MULTIDIMENSIONAL FLOATING POINT ARRAY, WITH THE REAL AND
C     IMAGINARY PARTS OF A DATUM STORED IMMEDIATELY ADJACENT IN STORAGE
C     (SUCH AS FORTRAN IV PLACES THEM).  NORMAL FORTRAN ORDERING IS
```

```
C     EXPECTED. THE FIRST SUBSCRIPT CHANGING FASTEST.  THE DIMENSIONS
C     ARE GIVEN IN THE INTEGER ARRAY NN, OF LENGTH NDIM.  ISIGN IS -1
C     TO INDICATE A FORWARD TRANSFORM (EXPONENTIAL SIGN IS -) AND +1
C     FOR AN INVERSE TRANSFORM (SIGN IS +).  IFORM IS +1 IF THE DATA ARE
C     COMPLEX, 0 IF THE DATA ARE REAL.  IF IT IS 0, THE IMAGINARY
C     PARTS OF THE DATA MUST BE SET TO ZERO.  AS EXPLAINED ABOVE, THE
C     TRANSFORM VALUES ARE ALWAYS COMPLEX AND ARE STORED IN ARRAY DATA.
C     WORK IS AN ARRAY USED FOR WORKING STORAGE.  IT IS FLOATING POINT
C     REAL, ONE DIMENSIONAL OF LENGTH EQUAL TO TWICE THE LARGEST ARRAY
C     DIMENSION NN(I) THAT IS NOT A POWER OF TWO.  IF ALL NN(I) ARE
C     POWERS OF TWO, IT IS NOT NEEDED AND MAY BE REPLACED BY ZERO IN THE
C     CALLING SEQUENCE.  THUS, FOR A ONE-DIMENSIONAL ARRAY, NN(1) ODD,
C     WORK OCCUPIES AS MANY STORAGE LOCATIONS AS DATA.  IF SUPPLIED,
C     WORK MUST NOT BE THE SAME ARRAY AS DATA.  ALL SUBSCRIPTS OF ALL
C     ARRAYS BEGIN AT ONE.
C
C     EXAMPLE 1.  THREE-DIMENSIONAL FORWARD FOURIER TRANSFORM OF A
C     COMPLEX ARRAY DIMENSIONED 32 BY 25 BY 13 IN FORTRAN IV.
C     DIMENSION DATA(32,25,13),WORK(50),NN(3)
C     COMPLEX DATA
C     DATA NN/32,25,13/
C     DO 1 I=1,32
C     DO 1 J=1,25
C     DO 1 K=1,13
C   1 DATA(I,J,K)=COMPLEX VALUE
C     CALL FOURT(DATA,NN,3,-1,1,WORK)
C
C     EXAMPLE 2.  ONE-DIMENSIONAL FORWARD TRANSFORM OF A REAL ARRAY OF
C     LENGTH 64 IN FORTRAN II.
C     DIMENSION DATA(2,64)
C     DO 2 I=1,64
C     DATA(1,I)=REAL PART
C   2 DATA(2,I)=0.
C     CALL FOURT(DATA,64,1,-1,0,0)
C
C     THERE ARE NO ERROR MESSAGES OR ERROR HALTS IN THIS PROGRAM.  THE
C     PROGRAM RETURNS IMMEDIATELY IF NDIM OR ANY NN(I) IS LESS THAN ONE.
C
C     PROGRAM BY NORMAN BRENNER FROM THE BASIC PROGRAM BY CHARLES
C     RADER.  JUNE 1967.  THE IDEA FOR THE DIGIT REVERSAL WAS
C     SUGGESTED BY RALPH ALTER.
C
C     THIS IS THE FASTEST AND MOST VERSATILE VERSION OF THE FFT KNOWN
C     TO THE AUTHOR.  A PROGRAM CALLED FOUR2 IS AVAILABLE THAT ALSO
C     PERFORMS THE FAST FOURIER TRANSFORM AND IS WRITTEN IN USASI BASIC
C     FORTRAN.  IT IS ABOUT ONE THIRD AS LONG AND RESTRICTS THE
C     DIMENSIONS OF THE INPUT ARRAY (WHICH MUST BE COMPLEX) TO BE POWERS
C     OF TWO.  ANOTHER PROGRAM, CALLED FOUR1, IS ONE TENTH AS LONG AND
C     RUNS TWO THIRDS AS FAST ON A ONE-DIMENSIONAL COMPLEX ARRAY WHOSE
C     LENGTH IS A POWER OF TWO.
C
C     REFERENCE--
C     IEEE AUDIO TRANSACTIONS (JUNE 1967), SPECIAL ISSUE ON THE FFT.
C
      DIMENSION DATA(1),NN(1),IFACT(32),WORK(1)
      TWOPI=6.283185307
      RTHLF=.70710 67812
      IF(NDIM-1)920,1,1
1     NTOT=2
      DO 2 IDIM=1,NDIM
      IF(NN(IDIM))920,920,2
2     NTOT=NTOT*NN(IDIM)
```

```
C
C     MAIN LOOP FOR EACH DIMENSION
C
      NP1=2
      DO 910 IDIM=1,NDIM
      N=NN(IDIM)
      NP2=NP1*N
      IF(N-1)920,900,5
C
C     IS N A POWER OF TWO AND IF NOT, WHAT ARE ITS FACTORS
C
5     M=N
      NTWO=NP1
      IF=1
      IDIV=2
10    IQUOT=M/IDIV
      IREM=M-IDIV*IQUOT
      IF(IQUOT-IDIV)50,11,11
11    IF(IREM)20,12,20
12    NTWO=NTWO+NTWO
      IFACT(IF)=IDIV
      IF=IF+1
      M=IQUOT
      GO TO 10
20    IDIV=3
      INON2=IF
30    IQUOT=M/IDIV
      IREM=M-IDIV*IQUOT
      IF(IQUOT-IDIV)60,31,31
31    IF(IREM)40,32,40
32    IFACT(IF)=IDIV
      IF=IF+1
      M=IQUOT
      GO TO 30
40    IDIV=IDIV+2
      GO TO 30
50    INON2=IF
      IF(IREM)60,51,60
51    NTWO=NTWO+NTWO
      GO TO 70
60    IFACT(IF)=M
C
C     SEPARATE FOUR CASES--
C        1. COMPLEX TRANSFORM OR REAL TRANSFORM FOR THE 4TH, 5TH,ETC.
C           DIMENSIONS.
C        2. REAL TRANSFORM FOR THE 2ND OR 3RD DIMENSION.  METHOD--
C           TRANSFORM HALF THE DATA, SUPPLYING THE OTHER HALF BY CON-
C           JUGATE SYMMETRY.
C        3. REAL TRANSFORM FOR THE 1ST DIMENSION, N ODD.  METHOD--
C           SET THE IMAGINARY PARTS TO ZERO.
C        4. REAL TRANSFORM FOR THE 1ST DIMENSION, N EVEN.  METHOD--
C           TRANSFORM A COMPLEX ARRAY OF LENGTH N/2 WHOSE REAL PARTS
C           ARE THE EVEN NUMBERED REAL VALUES AND WHOSE IMAGINARY PARTS
C           ARE THE ODD NUMBERED REAL VALUES.  SEPARATE AND SUPPLY
C           THE SECOND HALF BY CONJUGATE SYMMETRY.
C
70    ICASE=1
      IFMIN=1
      I1RNG=NP1
      IF(IDIM-4)71,100,100
```

```
71      IF(IFORM)72,72,100
72      ICASE=2
        I1RNG=NP0*(1+NPREV/2)
        IF(IDIM-1)73,73,100
73      ICASE=3
        I1RNG=NP1
        IF(NTWO-NP1)100,100,74
74      ICASE=4
        IFMIN=2
        NTWO=NTWO/2
        N=N/2
        NP2=NP2/2
        NTOT=NTOT/2
        I=1
        DO 80 J=1,NTOT
        DATA(J)=DATA(I)
80      I=I+2
C
C       SHUFFLE DATA BY BIT REVERSAL, SINCE N=2**K,  AS THE SHUFFLING
C       CAN BE DONE BY SIMPLE INTERCHANGE, NO WORKING ARRAY IS NEEDED
C
100     IF(NTWO-NP2)200,110,110
110     NP2HF=NP2/2
        J=1
        DO 150 I2=1,NP2,NP1
        IF(J-I2)120,130,130
120     I1MAX=I2+NP1-2
        DO 125 I1=I2,I1MAX,2
        DO 125 I3=I1,NTOT,NP2
        J3=J+I3-I2
        TEMPR=DATA(I3)
        TEMPI=DATA(I3+1)
        DATA(I3)=DATA(J3)
        DATA(I3+1)=DATA(J3+1)
        DATA(J3)=TEMPR
125     DATA(J3+1)=TEMPI
130     M=NP2HF
140     IF(J-M)150,150,145
145     J=J-M
        M=M/2
        IF(M-NP1)150,140,140
150     J=J+M
        GO TO 300
C
C       SHUFFLE DATA BY DIGIT REVERSAL FOR GENERAL N
C
200     NWORK=2*N
        DO 270 I1=1,NP1,2
        DO 270 I3=I1,NTOT,NP2
        J=I3
        DO 260 I=1,NWORK,2
        IF(ICASE-3)210,220,210
210     WORK(I)=DATA(J)
        WORK(I+1)=DATA(J+1)
        GO TO 230
220     WORK(I)=DATA(J)
        WORK(I+1)=0.
230     IFP2=NP2
        IF=IFMIN
240     IFP1=IFP2/IFACT(IF)
        J=J+IFP1
        IF(J-I3-IFP2)260,250,250
```

```
250    J=J-IFP2
       IFP2=IFP1
       IF=IF+1
       IF(IFP2-NP1)260,260,240
260    CONTINUE
       I2MAX=I3+NP2-NP1
       I=1
       DO 270 I2=I3,I2MAX,NP1
       DATA(I2)=WORK(I)
       DATA(I2+1)=WORK(I+1)
270    I=I+2
C
C      MAIN LOOP FOR FACTORS OF TWO.  PERFORM FOURIER TRANSFORMS OF
C      LENGTH FOUR, WITH ONE OF LENGTH TWO IF NEEDED.  THE TWIDDLE FACTOR
C      W=EXP(ISIGN*2*PI*SQRT(-1)*M/(4*MMAX)).  CHECK FOR W=ISIGN*SQRT(-1)
C      AND REPEAT FOR W=W*(1+ISIGN*SQRT(-1))/SQRT(2).
C
300    IF(NTWO-NP1)600,600,305
305    NP1TW=NP1+NP1
       IPAR=NTWO/NP1
310    IF(IPAR-2)350,330,320
320    IPAR=IPAR/4
       GO TO 310
330    DO 340 I1=1,I1RNG,2
       DO 340 K1=I1,NTOT,NP1TW
       K2=K1+NP1
       TEMPR=DATA(K2)
       TEMPI=DATA(K2+1)
       DATA(K2)=DATA(K1)-TEMPR
       DATA(K2+1)=DATA(K1+1)-TEMPI
       DATA(K1)=DATA(K1)+TEMPR
340    DATA(K1+1)=DATA(K1+1)+TEMPI
350    MMAX=NP1
360    IF(MMAX-NTWO/2)370,600,600
370    LMAX=MAX0(NP1TW,MMAX/2)
       DO 570 L=NP1,LMAX,NP1TW
       M=L
       IF(MMAX-NP1)420,420,380
380    THETA=-TWOPI*FLOAT(L)/FLOAT(4*MMAX)
       IF(ISIGN)400,390,390
390    THETA=-THETA
400    WR=COS(THETA)
       WI=SIN(THETA)
410    W2R=WR*WR-WI*WI
       W2I=2.*WR*WI
       W3R=W2R*WR-W2I*WI
       W3I=W2R*WI+W2I*WR
420    DO 530 I1=1,I1RNG,2
       KMIN=I1+IPAR*M
       IF(MMAX-NP1)430,430,440
430    KMIN=I1
440    KDIF=IPAR*MMAX
450    KSTEP=4*KDIF
       IF(KSTEP-NTWO)460,460,530
460    DO 520 K1=KMIN,NTOT,KSTEP
       K2=K1+KDIF
       K3=K2+KDIF
       K4=K3+KDIF
       IF(MMAX-NP1)470,470,480
470    U1R=DATA(K1)+DATA(K2)
       U1I=DATA(K1+1)+DATA(K2+1)
       U2R=DATA(K3)+DATA(K4)
```

```
      U2I=DATA(K3+1)+DATA(K4+1)
      U3R=DATA(K1)-DATA(K2)
      U3I=DATA(K1+1)-DATA(K2+1)
      IF(ISIGN)471,472,472
471   U4R=DATA(K3+1)-DATA(K4+1)
      U4I=DATA(K4)-DATA(K3)
      GO TO 510
472   U4R=DATA(K4+1)-DATA(K3+1)
      U4I=DATA(K3)-DATA(K4)
      GO TO 510
480   T2R=W2R*DATA(K2)-W2I*DATA(K2+1)
      T2I=W2R*DATA(K2+1)+W2I*DATA(K2)
      T3R=WR*DATA(K3)-WI*DATA(K3+1)
      T3I=WR*DATA(K3+1)+WI*DATA(K3)
      T4R=W3R*DATA(K4)-W3I*DATA(K4+1)
      T4I=W3R*DATA(K4+1)+W3I*DATA(K4)
      U1R=DATA(K1)+T2R
      U1I=DATA(K1+1)+T2I
      U2R=T3R+T4R
      U2I=T3I+T4I
      U3R=DATA(K1)-T2R
      U3I=DATA(K1+1)-T2I
      IF(ISIGN)490,500,500
490   U4R=T3I-T4I
      U4I=T4R-T3R
      GO TO 510
500   U4R=T4I-T3I
      U4I=T3R-T4R
510   DATA(K1)=U1R+U2R
      DATA(K1+1)=U1I+U2I
      DATA(K2)=U3R+U4R
      DATA(K2+1)=U3I+U4I
      DATA(K3)=U1R-U2R
      DATA(K3+1)=U1I-U2I
      DATA(K4)=U3R-U4R
520   DATA(K4+1)=U3I-U4I
      KDIF=KSTEP
      KMIN=4*(KMIN-I1)+I1
      GO TO 450
530   CONTINUE
      M=M+LMAX
      IF(M-MMAX)540,540,570
540   IF(ISIGN)550,560,560
550   TEMPR=WR
      WR=(WR+WI)*RTHLF
      WI=(WI-TEMPR)*RTHLF
      GO TO 410
560   TEMPR=WR
      WR=(WR-WI)*RTHLF
      WI=(TEMPR+WI)*RTHLF
      GO TO 410
570   CONTINUE
      IPAR=3-IPAR
      MMAX=MMAX+MMAX
      GO TO 360
C
C     MAIN LOOP FOR FACTORS NOT EQUAL TO TWO,  APPLY THE TWIDDLE FACTOR
C     W=EXP(ISIGN*2*PI*SQRT(-1)*(J1-1)*(J2-J1)/(IFP1+IFP2)), THEN
C     PERFORM A FOURIER TRANSFORM OF LENGTH IFACT(IF), MAKING USE OF
C     CONJUGATE SYMMETRIES,
C
600   IF(NTWO-NP2)605,700,700
```

```
605   IFP1=NTWO


      IF=INON2
      NP1HF=NP1/2
610   IFP2=IFACT(IF)*IFP1
      J1MIN=NP1+1
      IF(J1MIN-IFP1)615,615,640
615   DO 635 J1=J1MIN,IFP1,NP1
      THETA=-TWOPI*FLOAT(J1-1)/FLOAT(IFP2)
      IF(ISIGN)625,620,620
620   THETA=-THETA
625   WSTPR=COS(THETA)
      WSTPI=SIN(THETA)
      WR=WSTPR
      WI=WSTPI
      J2MIN=J1+IFP1
      J2MAX=J1+IFP2-IFP1
      DO 635 J2=J2MIN,J2MAX,IFP1
      I1MAX=J2+I1RNG-2
      DO 630 I1=J2,I1MAX,2
      DO 630 J3=I1,NTOT,IFP2
      TEMPR=DATA(J3)
      DATA(J3)=DATA(J3)*WR-DATA(J3+1)*WI
630   DATA(J3+1)=TEMPR*WI+DATA(J3+1)*WR
      TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI
635   WI=TEMPR*WSTPI+WI*WSTPR
640   THETA=-TWOPI/FLOAT(IFACT(IF))
      IF(ISIGN)650,645,645
645   THETA=-THETA
650   WSTPR=COS(THETA)
      WSTPI=SIN(THETA)
      J2RNG=IFP1*(1+IFACT(IF)/2)
      DO 695 I1=1,I1RNG,2
      DO 695 I3=I1,NTOT,NP2
      J2MAX=I3+J2RNG-IFP1
      DO 690 J2=I3,J2MAX,IFP1
      J1MAX=J2+IFP1-NP1
      DO 680 J1=J2,J1MAX,NP1
      J3MAX=J1+NP2-IFP2
      DO 680 J3=J1,J3MAX,IFP2
      JMIN=J3-J2+I3
      JMAX=JMIN+IFP2-IFP1
      I=1+(J3-I3)/NP1HF
      IF(J2-I3)655,655,665
655   SUMR=0.
      SUMI=0.
      DO 660 J=JMIN,JMAX,IFP1
      SUMR=SUMR+DATA(J)
660   SUMI=SUMI+DATA(J+1)
      WORK(I)=SUMR
      WORK(I+1)=SUMI
      GO TO 680
665   ICONJ=1+(IFP2-2*J2+I3+J3)/NP1HF
      J=JMAX
      SUMR=DATA(J)
      SUMI=DATA(J+1)
      OLDSR=0.
      OLDSI=0.
      J=J-IFP1
670   TEMPR=SUMR
```

```
      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
680   CONTINUE
      IF(J2-I3)685,685,686
685   WR=WSTPR
      WI=WSTPI
      GO TO 690
686   TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI
      WI=TEMPR*WSTPI+WI*WSTPR
690   TWOWR=WR+WR
      I=1
      I2MAX=I3+NP2-NP1
      DO 695 I2=I3,I2MAX,NP1
      DATA(I2)=WORK(I)
      DATA(I2+1)=WORK(I+1)
695   I=I+2
      IF=IF+1
      IFP1=IFP2
      IF(IFP1-NP2)610,700,700
C
C     COMPLETE A REAL TRANSFORM IN THE 1ST DIMENSION, N EVEN, BY CON-
C     JUGATE SYMMETRIES.
C
700   GO TO (900,800,900,701),ICASE
701   NHALF=N
      N=N+N
      THETA=-TWOPI/FLOAT(N)
      IF(ISIGN)703,702,702
702   THETA=-THETA
703   WSTPR=COS(THETA)
      WSTPI=SIN(THETA)
      WR=WSTPR
      WI=WSTPI
      IMIN=3
      JMIN=2*NHALF-1
      GO TO 725
710   J=JMIN
      DO 720 I=IMIN,NTOT,NP2
      SUMR=(DATA(I)+DATA(J))/2.
      SUMI=(DATA(I+1)+DATA(J+1))/2.
      DIFR=(DATA(I)-DATA(J))/2.
      DIFI=(DATA(I+1)-DATA(J+1))/2.
      TEMPR=WR*SUMI+WI*DIFR
      TEMPI=WI*SUMI-WR*DIFR
      DATA(I)=SUMR+TEMPR
      DATA(I+1)=DIFI+TEMPI
      DATA(J)=SUMR-TEMPR
      DATA(J+1)=-DIFI+TEMPI
```

```
720     J=J+NP2
        IMIN=IMIN+2
        JMIN=JMIN-2
        TEMPR=WR
        WR=WR*WSTPR-WI*WSTPI
        WI=TEMPR*WSTPI+WI*WSTPR
725     IF(IMIN-JMIN)710,730,740
730     IF(ISIGN)731,740,740
731     DO 735 I=IMIN,NTOT,NP2
735     DATA(I+1)=-DATA(I+1)
740     NP2=NP2+NP2
        NTOT=NTOT+NTOT
        J=NTOT+1
        IMAX=NTOT/2+1
745     IMIN=IMAX-2*NHALF
        I=IMIN
        GO TO 755
750     DATA(J)=DATA(I)
        DATA(J+1)=-DATA(I+1)
755     I=I+2
        J=J-2
        IF(I-IMAX)750,760,760
760     DATA(J)=DATA(IMIN)-DATA(IMIN+1)
        DATA(J+1)=0.
        IF(I-J)770,780,780
765     DATA(J)=DATA(I)
        DATA(J+1)=DATA(I+1)
770     I=I-2
        J=J-2
        IF(I-IMIN)775,775,765
775     DATA(J)=DATA(IMIN)+DATA(IMIN+1)
        DATA(J+1)=0.
        IMAX=IMIN
        GO TO 745
780     DATA(1)=DATA(1)+DATA(2)
        DATA(2)=0.
        GO TO 900
C
C       COMPLETE A REAL TRANSFORM FOR THE 2ND OR 3RD DIMENSION BY
C       CONJUGATE SYMMETRIES.
C
800     IF(I1RNG-NP1)805,900,900
805     DO 860 I3=1,NTOT,NP2
        I2MAX=I3+NP2-NP1
        DO 860 I2=I3,I2MAX,NP1
        IMIN=I2+I1RNG
        IMAX=I2+NP1-2
        JMAX=2*I3+NP1-IMIN
        IF(I2-I3)820,820,810
810     JMAX=JMAX+NP2
820     IF(IDIM-2)850,850,830
830     J=JMAX+NP0
        DO 840 I=IMIN,IMAX,2
        DATA(I)=DATA(J)
        DATA(I+1)=-DATA(J+1)
840     J=J-2
850     J=JMAX
        DO 860 I=IMIN,IMAX,NP0
        DATA(I)=DATA(J)
        DATA(I+1)=-DATA(J+1)
860     J=J-NP0
```

```
C
C        END OF LOOP ON EACH DIMENSION
C
900      NP0=NP1
         NP1=NP2
910      NPREV=N
920      RETURN
         END
```

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA – R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Lincoln Laboratory, M.I.T.	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP None

3. REPORT TITLE

Three Fortran Programs that Perform the Cooley-Tukey Fourier Transform

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Technical Note

5. AUTHOR(S) *(Last name, first name, initial)*

Brenner, Norman M.

6. REPORT DATE 28 July 1967	7a. TOTAL NO. OF PAGES 34	7b. NO. OF REFS 16

8a. CONTRACT OR GRANT NO. AF 19(628)5167 b. PROJECT NO. c. 649L d.	9a. ORIGINATOR'S REPORT NUMBER(S) Technical Note 1967-2 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* ESD-TR-67-462

10. AVAILABILITY/LIMITATION NOTICES

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES None	12. SPONSORING MILITARY ACTIVITY Air Force Systems Command, USAF

13. 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 80×80 can be transformed in 19.2 seconds. Besides the main array, only a working storage array of size 160 need be supplied.

14. KEY WORDS

Fortran   Fourier transforms   computer programs

UNCLASSIFIED
Security Classification

Printed by
United States Air Force
L. G. Hanscom Field
Bedford, Massachusetts