X-622-66-316

NASA TM X-55613

THE FAST FOURIER TRANSFORM IN FOURIER SPECTROSCOPY

BY

THOMAS E. MICHELS

JULY 1966

GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

X-622-66-316

THE FAST FOURIER TRANSFORM IN FOURIER SPECTROSCOPY

by

Thomas E. Michels

July 1966

Goddard Space Flight Center
Greenbelt, Maryland

CONTENTS

THE FAST FOURIER TRANSFORM IN FOURIER SPECTROSCOPY

by

Thomas E. Michels

N67-10863

ABSTRACT

The "Fast Fourier Transform" (suggested by Drs. J. W. Cooley and J. W. Tukey) is presented with special application to solving the interferogram integral obtained in Fourier Spectroscopy. Computational timing is tabulated and an explanation of a computer routine using this method to a binary base is presented.

Author

THE FAST FOURIER TRANSFORM IN FOURIER SPECTROSCOPY

INTRODUCTION

The problem of digitally reducing interferogram data obtained from an interferometer spectrometer is two fold; (1) numerical quadrature of the interferogram integral to obtain the spectral magnitudes, and (2) computation of the phase of the spectral magnitudes to determine the direction of radiation flow.

A third requirement can be placed on the computation if a great number of spectrums are to be analysed; that is fast calculation on a high speed digital computer. This is the problem with which this paper is primarily concerned.

Many methods of fast calculations of the interferogram integral have been suggested; however, the "Fast Fourier Transform" (Cooley, J. W. and Tukey, J. W., 1965) method offers many advantages in speed and accuracy which others do not have.

The application of this method to discrete interferogram data is discussed and the use of an existing computer program written by Dr. J. W. Cooley in FORTRAN IV using the "Fast Fourier Transform" to compute a Fourier transform or series is explained in Appendicies I and II. Only a small amount of interferometer theory is discussed so as to keep the general theme of the report.

It should be mentioned that the Fast Fourier Transform has been implemented with much success into the Infrared Interferometer Spectrometer (IRIS) Experiment Data Reduction Program as well as the theoretical simulating the IRIS experiment. Computational timings have been tabulated using these programs and are given in a following section. The reader should also be aware that now with the computational speeds obtained using the "Fast Fourier Transform," problems in Fourier Spectroscopy which previously were overly time consuming on the computer are now realistic for computer solution. These include taking convolutions for truncated Fourier Integrals, shown by Dr. J. W. Cooley, co-author of the Fast Fourier Transform (in an unpublished report) and correction of asymmetric interferograms (Forman, M. L., 1965).

The author is greatly indebted to Drs. B. Conrath and R. R. Hanel for many hours of helpful discussions on interferometer theory and Fourier analysis, and to Dr. J. W. Cooley for the use of his computer program.

DESCRIPTION OF THE PROBLEM

The instrument used to obtain interferogram data in the IRIS experiment is essentially a two beam Michelson interferometer, sketched in Figure 1.

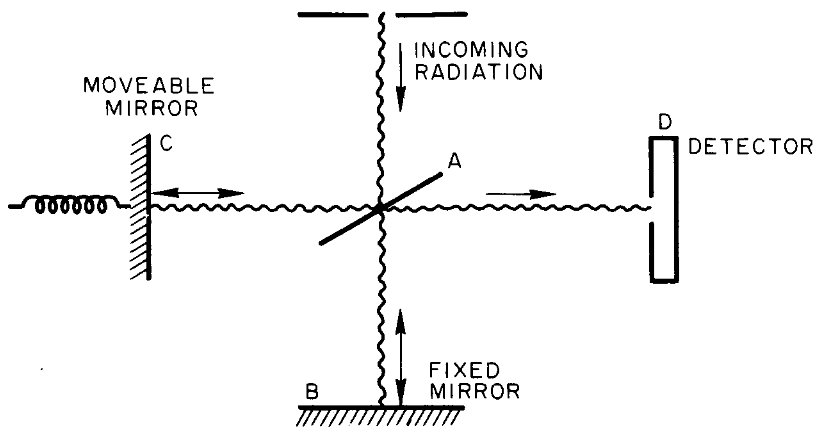

Figure 1–Sketch of Michelson Interferometer

The incoming heterochromatic radiation is split at the beam splitter, A, into two waves of equal amplitude; one directed toward a moveable mirror at C, and the other toward a fixed mirror at B. They are reflected back to the beam splitter, recombined, and directed to the detector at D. The recombined signal received at the detector is the interference pattern of the two beams called the interferogram and, ideally, can be defined as a function of path difference traveled by the two beams by

$$I(\delta) = \int_{-\infty}^{\infty} B_\nu \, e^{i 2\pi\nu\delta} \, d\nu \qquad (1)$$

The path difference, δ, is defined with the aid of Figure 1 by

$$\delta = 2\,(CA - BA)$$

The multiplying factor of two times the distance (CA-BA) arises from the fact that the radiation travels to the mirror and back again to transverse the added distance (δ can be given as a function of time and mirror velocity; however, we will use the path difference relationship here).

B_ν is the spectral distribution of the incoming radiation and is written here in terms of wave number, ν. It has been extended to include negative wave numbers by the definition

$$B_{-\nu} = B_\nu$$

Clearly, B_ν is real, and B_ν and $I(\delta)$ are transform pairs; thus one is able to obtain the spectrum B_ν by taking the Fourier transform of equation (1) which is

$$B_\nu = \int_{-\infty}^{\infty} I(\delta)\, e^{-i2\pi\nu\delta}\, d\delta. \tag{2}$$

If $I(\delta)$ were symmetric, which in the theoretical case is true, equation (2) reduces to

$$B_\nu = 2\int_0^{\infty} I(\delta) \cos(2\pi\nu\delta)\, d\delta$$

However, in actual interferometer use a phase shift, φ_ν, is introduced by the instrument components to the incoming radiation and equation (1) should be re-written as

$$I(\delta) = \int_{-\infty}^{\infty} B_\nu\, e^{i\varphi_\nu}\, e^{i2\pi\nu\delta}\, d\nu \tag{3}$$

$I(\delta)$ in this case will be an asymmetric function about zero path difference, δ_0. Upon taking the Fourier transform of equation (3), one obtains

$$B_\nu\, e^{i\varphi_\nu} = \int_{-\infty}^{\infty} I(\delta)\, e^{-i2\pi\nu\delta}\, d\delta. \tag{4}$$

Equation (4) is of the form

$$B_\nu e^{i\varphi_\nu} = C\nu + i\ S\nu$$

and one merely takes the absolute value of the right hand side to obtain the spectral magnitudes.

The phase shift φ_ν can, of course, be computed from

$$\varphi_\nu = \tan^{-1}\left(\frac{S_\nu}{C_\nu}\right) \tag{5}$$

The phase spectrum can be a useful tool in interpreting the amplitude spectrum. The direction of the net energy transfer between the detector and the target (determining whether the detector is warmer or colder than the target) changes direction from one spectral region to another and is indicated by an abrupt phase shift of 180°.

Therefore, the problem of digitally reducing interferogram data requires solution of the interferogram integral in equation (4) and computation of the spectral amplitude phases by equation (5).

NUMERICAL QUADRATURE OF THE INTERFEROGRAM INTEGRAL

The numerical calculation of the integral in equation (4) can be done by various quadrature methods. The Gaussian quadrature is perhaps the most accurate, but this method requires unequal spacing in the sampled data.

It is suggested that in practical use, when the truncated range $(-\delta_1, \delta_2)$ of the integral (see equation (6)) is large enough, many advantages are offered by use of the trapezoidal rule.

One advantage in accuracy is easily seen by looking at Eulers summation formula: (Scarborough, 1955)

$$\int_a^b f(x)\,dx = h\left[\frac{f(x_0)}{2} + f(x_1) + \ldots + f(x_{n-1}) + \frac{f(x_n)}{2}\right] - \frac{h}{12}\left[f'(b) - f'(a)\right]$$
$$+ \frac{h^3}{720}\left[f'''(b) - f'''(a)\right] - \frac{h^5}{30240}\left[f^{v}(b) - f^{v}(a)\right] + \cdots$$

This states that if the odd derivatives at the endpoints of an integration are small or equal, the trapezoidal rule is an excellent approximation to the integral.

In many cases, the derivatives at the endpoints of the interferogram are equal, and in the worst cases observed the derivatives were always of the same order. In these cases, the order of the error would be h, the separation distance ($\Delta\delta$ in the interferogram integral), and for the interferogram integral, this is small.

Therefore, the trapezoidal rule appears to be a satisfactory quadrature method and, as one will see, is a suitable form for the "Fast Fourier Transform" to obtain speed of calculation.

Rewriting equation (4) as a truncated inverse over the interval $(-\delta_1, \delta_2)$, and, for convenience, setting $A_\nu = B_\nu e^{i\varphi_\nu}$, we have

$$A_\nu = \int_{-\delta_1}^{\delta_2} I(\delta)\, e^{-i2\pi\nu\delta}\, d\nu \qquad (6)$$

Since the interferogram values sampled by the instrument are an average over an interval, the trapezoidal rule takes the form

$$A_\nu = B_\nu\, e^{i\varphi_\nu} = \Delta\delta \sum_{j=-m}^{n} I(\delta_j)\, e^{-i2\pi\nu\delta j} \qquad (7)$$

where the subscript, j, ranges over N sampled intervals with N defined as

$$N = m + n + 1$$

The point δ_0 is of course the sampled interval which normally will not correspond exactly to the point of zero path difference. Correction methods have been studied for correction of asymmetric interferograms of this type (Forman, M. L.); however, this paper will not concern itself with these methods.

Solution of equation (7) to obtain the spectral magnitudes and phase angles is easily accomplished using the "Fast Fourier Transform," and, as one will see, yields a very high degree of accuracy for computer solution.

THE "FAST FOURIER TRANSFORM"

1. Description of the Method

The form of the Fourier series equation required for the "Fast Fourier Transform" is

$$I(j) = \sum_{k=0}^{N-1} B(k) e^{i\frac{2\pi jk}{N}} \tag{8}$$

$$j = 0,\ 1,\ 2.\ .\ .\ N-1$$

and its transform is

$$B(k) = \frac{1}{N} \sum_{j=0}^{N-1} I(\delta_j)\, e^{-i\frac{2\pi jk}{N}} \tag{9}$$

Normal solution of equation (8) using a decimal based summation requires an order of N^2 operations where an operation is defined as one multiplication and addition. However, the number of operations using a different base, say r, is of the order $8N \log_r N$. A simple example will illustrate the saving.

If N is composite such that $N = r_1 \cdot r_2$, and j and k are written

$$j = r_1 j_1 + j_0,\ j_0 = 0, 1, 2, \ldots r_1 - 1;\ j_1 = 0, 1, \ldots r_2 - 1$$

and

$$k = r_2 k_1 + k_0,\ k_0 = 0,\ 1,\ \ldots\ r_2 - 1;\ k_1 = 0,\ 1,\ \ldots\ r_1 - 1$$

then equation (8) can be written

$$I(j) = \sum_{k_0} \sum_{k_1} B(k)\, e^{i \frac{2\pi}{N} j (r_2 k_1 + k_0)}$$

$$= \sum_{k_0} \sum_{k_1} B(k)\, e^{i \frac{2\pi}{N} j r_2 k_1}\, e^{i \frac{2\pi}{N} j k_0}$$

But

$$e^{i \frac{2\pi}{N} j r_2 k_1} = e^{i \frac{2\pi}{N} k_1 r_2 (r_1 j_1 + j_0)} = e^{i \frac{2\pi}{N} k_1 j_0 r_2}$$

Therefore, the inner sum over k_1, depends only on j_0 and k_0, which can be written as

$$B_1(j_0, k_0) = \sum_{k_1} B(r_2 k_1 + k_0)\, e^{i \frac{2\pi}{N} k_1 j_0 r_2}$$

and equation (8) now becomes

$$I(r_1 j_1 + j_0) = \sum_{k_0} B_1 (j_0, k_0)\, e^{i \frac{2\pi}{N} j k_0}$$

The total number of operations for this case is now $N(r_1 + r_2)$ as opposed to N^2 before.

It has been shown (Colley, J. W., Tukey, J. W., 1965) that the most saving is obtained when N is written as some number raised to an integer power, or

$$N = r^m$$

and, further, if $r = e$, one uses the least operations in solving equation (8). However, if $r = 2$, the saving is approximately the same, and from a digital computer standpoint, certain advantages in programming are obtained.

With this in mind, if one writes

$$N = 2^m$$

and

$$j = 2^{m-1} j_{m-1} + 2^{m-2} j_{m-2} + \dots + j_0$$

$$k = 2^{m-1} k_{m-1} + 2_{m-1} k_{m-2} + \dots + k_0$$

where each j_ℓ and k_ℓ, $\ell = 0, \dots m-1$ take on the values 0, 1, then equation (8) takes the form

$$I(2^{m-1} j_{m-1} + 2^{m-2} j_{m-2} + \dots + j_0) = B_m (j_0 + 2 j_1 + \dots + 2^{m-1} j_{m-1})$$

where

$$B_1(j_0, 2^{m-2} k_{m-2} + \dots + k_0) = \sum_{k_{m-1}} B(2^{m-1} k_{m-1} + \dots + k_0)\, e^{i \frac{2\pi}{N} j_0 k_{m-1} 2^{m-1}}$$

and

$$B_{\ell}(j_0 + \dots + 2^{\ell-1} j_{\ell-1}, k_0 + \dots + 2^{m-\ell-1} k_{m-\ell-1})$$

$$= \sum_{k_{m-\ell}} A_{\ell-1}(j_0 + \dots + 2^{\ell-2} j_{\ell-2}, k_0 + \dots + 2^{m-\ell} k_{m-\ell})$$

$$e^{i\frac{2\pi}{N}(j_{\ell-1} 2^{\ell-1} + \dots + j_0) \cdot k_{m-\ell} 2^{m-\ell}}$$

$$\ell = 1, 2, \dots, m$$

The routine listed in Appendix I uses the base two and is written such that either the Fourier series coefficients or their transform can be computed. That is, one is able to compute either equation (8) or equation (9).

2. Application to the Interferogram Integral.

We saw that the spectral magnitudes and phase angles can be obtained from equation (7),

$$A_\nu = B_\nu e^{i\varphi_\nu} = \Delta\delta \sum_{j=-m}^{n} I(\delta_j) e^{-i2\pi\nu\delta_j}.$$

If one sets,

$$\nu_k = k\Delta\nu, \qquad k = 0, 1, \dots N-1$$

and

$$\delta_j = j\,\Delta\delta, \qquad j = 0, 1, \dots N-1$$

where $\nu_0 = \delta_0 = 0$, we have

$$A\nu_k = \Delta\delta \sum_{j=-n}^{m} I(\delta_j)\, e^{-i2\pi jk\Delta\nu\Delta\delta}. \tag{10}$$

Further, if one makes the substitutions

$$j = j - n$$

$$\Delta\delta\Delta\nu = N^{-1} = (\text{Number of sampled intervals})^{-1},$$

one can write the summation in the form suitable for the "Fast Fourier Transform." That is,

$$A\nu_k = \Delta\delta\, e^{i\frac{2\pi kn}{N}} \sum_{j=0}^{N-1} I(\delta_j)\, e^{-i\frac{2\pi}{N}jk} \tag{11}$$

where

$$N = m + n + 1.$$

If the number of sampled intervals, N , does not equal a power of two and the interferogram has zero mean (no DC component), one is able to add values equal to zero to either or both ends of the interferogram without loss of accuracy and compute spectral magnitudes corresponding to $\nu = k\Delta\nu$. Here,

$$\Delta\nu = \frac{1}{\Delta\delta \cdot N'}$$

where

$$N' = N + \ell$$

such that N' is equal to two raised to some integral power.

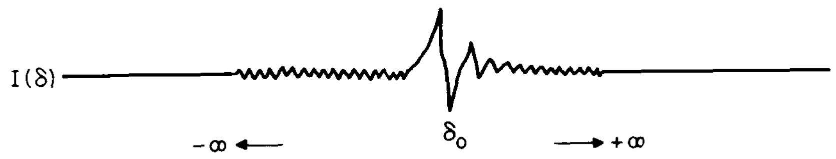

Figure 2–Interferometer extended with zeros on each end with δ_0 centered in area of N′ intervals.

Further, if the interferogram is positioned in the area of N′ intervals as shown in Figure 2, such that δ_0 is a distance N′/2 from the beginning, a particularly simple case arises. Keeping in mind that $A_\nu = B\nu\, e^{i\varphi\nu}$, equation (11) takes the following form for this case,

$$A\nu_k = \Delta\delta\, e^{i\pi k} \sum_{j=0}^{N-1} I(\delta_j)\, e^{-i\frac{2\pi}{N} jk} \tag{12}$$

and the spectral magnitudes are computed as before from

$$B\nu_k = |A\nu_k| \tag{13}$$

Re-writing equation (12) as

$$B\nu_k\, e^{i\varphi\nu_k} = e^{ik\pi}(C\nu_k - i\, S_{\nu k})$$

or

$$B\nu_k\, e^{i(\varphi\nu_k - k\pi)} = C\nu_k - i\, S_{\nu k}$$

the phase angle can then be computed from

$$\varphi\nu_k = \tan^{-1}\left(\frac{S\nu_k}{C\nu_k}\right) + k\,\pi \tag{14}$$

or

$$\varphi\nu_k = \tan^{-1}\left[\frac{(-1)^k\, S\nu_k}{(-1)^k\, C\nu_k}\right]$$

For the case when the interval δ_0 is not positioned in the center of the array, one still can use equation (12) to compute the spectral magnitudes, however, one must use an equation similar to equation (11) to compute the phase angles. That is,

$$\varphi\nu_k = \tan^{-1}\left(\frac{S\nu_k}{C\nu_k}\right) + \frac{2\pi kn}{N} \tag{15}$$

where, n , is the number of intervals from the beginning of the array to δ_0.

In using the routine in Appendix I, one must be careful applying it to compute the Interferogram integral. From experience it proved easier and faster to compute the Fourier series coefficients rather than the Fourier transform. This means that for the real input data, one obtains the complex conjugate of the vector $B\nu_k\, e^{i\varphi\nu_k}$ as output from the "Fast Fourier Transform."

Clearly, this makes no difference in computing the spectral magnitudes other than a factor of two, however, the phase angles must now be computed from

$$\varphi\nu_k = k\pi - \tan^{-1}\left(\frac{S\nu_k}{C\nu_k}\right) \tag{16}$$

or

$$\varphi\nu_k = \tan^{-1}\left[\frac{(-1)^{k+1}\, S\nu_k}{(-1)^{k+1}\, C\nu_k}\right] \tag{17}$$

or, in the case where δ_0 is not positioned in the center of the interferogram,

$$\varphi\nu_k = \frac{2\pi kn}{N} - \tan^{-1}\left(\frac{S\nu_k}{C\nu_k}\right) \tag{18}$$

3. Summary of the Application.

The method of computing the spectral magnitudes and phase angles can be summarized as follows:

1. Read the interferogram data into the "Fast Fourier Transform" (FFT) and compute a Fourier series yielding $C\nu_k$ and $S\nu_k$ as output.

$$\mathrm{FFT\,(series)} = C\nu_k + i\, S\nu_k$$

2. Compute the spectral magnitudes from

$$B\nu_k = 2 \cdot \Delta\delta \cdot |C\nu_k + i\, S\nu_k|$$

where

$$\nu_k = k\,\Delta\nu,\ k = 0,\ 1,\ \ldots\ N-1$$

and

$$\Delta\nu = \frac{1}{\Delta\delta \cdot N'}$$

3. Compute the phase angles from either equation (17) or (18) depending upon the position of the interferogram data in the array of size $N' = 2^m$.

In using the routine in Appendix I, one automatically obtains the same number of points of output as he has read in as input.

4. Computational Timing

Figure 3 shows the computational times obtained using the FORTRAN IV version of the "Fast Fourier Transform" routine (HARM) listed in Appendix I. The line indicating the times obtained on the IBM 360/65 system were obtained with the HARM subroutine, compiled with the optimization option. These times were obtained by Mr. Guy Marcot, Laboratory for Space Sciences, GSFC. A comparison with conventional times obtained with the algorithm by Goertzel indicates the saving, especially with a large number of data points.

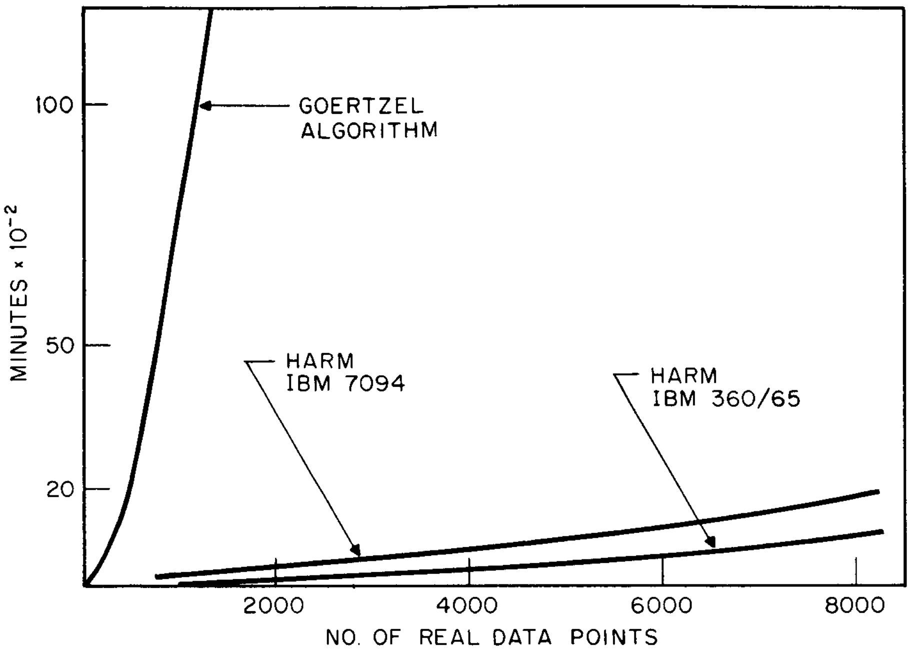

Figure 3–Computation Times using the FORTRAN IV version of HARM on the IBM 7094 and 360/65 Computers vs. the Goertzel algorithm.

Another version of the HARM subroutine, written in FORTRAN II and FAP, which is used in the theoretical simulation of the IRIS experiment was timed for 2^{11} and 2^{12} points on the IBM 7094 computer. These results are tabulated below.

No. of Points	Time for Fourier series calculation
2^{11}	1.3 sec
2^{12}	2.6 sec

The times tabulated above include the time obtained to compute the sine and cosine transform and do not include taking the absolute value or obtain the phase angle. This has been timed using the theoretical IRIS simulation. To compute 722 spectral magnitudes and phase angles out of the 2^{11} or 2^{12} available took approximately 3/10 seconds.

REFERENCES

1. Blackman, R. B. and Tukey, J. W., "The Measurement of Power Spectra," Dover Publications, Inc., New York, 1958.

2. Byrne, L. H., "Digital Computer Simulation of the Infrared Interferometer Spectrometer (IRIS) and Interferogram Analysis," Goddard Space Flight Center Report X-650-65-24, 1965.

3. Connes, J., "Recherches sur la Spectroscopie par Transformation de Fourier," Rev. d'Opt, Vol. 40, 1961.

4. Cooley, J. W. and Tukey, J. W., "An Algorithm for the Machine Calculation of Complex Fourier Series," Mathematics of Computation, Vol. 19, p. 297, 1965.

5. Forman, M. L., et al, "Nonlinear Phase Correction of Interferograms Obtained in Fourier Spectroscopy," Air Force Contract AF 19(628)-251, Scientific Report No. 2, AD 624-086, Clearinghouse for Federal Scientific and Technical Information, Washington, D. C., July 1965.

6. Forman, M. L., et al, "Correction of Asymmetric Interferograms Obtained in Fourier Spectroscopy," Journal of the Optical Society of America, Vol. 56, No. 1, Feb. 1966.

7. Hanel, R. A. and Chaney, L., "The Infrared Interferometer Spectrometer Experiment (IRIS)." Vol. I, Martian Fly-by Mission, Goddard Space Flight Center, Report X-650-64-204, 1964.

8. Hanel R. A. and Chaney, L., "The Infrared Interferometer Spectrometer Experiment (IRIS)." Vol. II, Meteorological Mission, Goddard Space Flight Center, Report X-650-65-75, 1965.

9. Mertz, L., "Astronomical Infrared Spectrometer," The Astronomical Journal, Vol. 70, No. 8, p. 548, October 1965.

10. Mertz, L., "Transformations In Optics," John Wiley and Sons, Inc., New York, 1965.

11. Strong, J. and Vanasse, G. A., "Interferometric Spectroscopy in the Far Infrared," Jour. of the Opt. Soc. of Am. Vol. 49, No. 9, September 1959.

12. Scarborough, J. B., "Numerical Mathematical Analysis," The Johns Hopkins Press, Baltimore, 1955.

13. Williams, R. A. and Chang, W. S. C., "Resolution and Noise in Fourier-Transform Spectroscopy," Journal of the Optical Society of America, Vol. 56, No. 2, Feb. 1966.

APPENDIX I

Sample Program Using the "Fast Fourier Transform" Routine "HARM"

The following program has been written exemplifying the use of the "HARM" subroutine to compute the transform of a set of real data and take the inverse of the transform to compare with the original data.

"HARM" is written in FORTRAN IV and is set up to accept complex input as well as to perform a three-dimensional sum in directions M(1), M(2), M(3). The return from "HARM" has the complex vector computed and stored in the array A, where

$$A(1) + i\ A(2) = C\nu_0 + i\ S\nu_0$$

$$\begin{array}{ccc} A(3) + i\ A(4) = & C\nu_1 + & i\ S\nu_1 \\ \vdots & \vdots & \vdots \end{array}$$

and so on. The array M and option IFS must be set prior to entry, and if one wishes to use the routine in one-dimension, as in the sample program, one sets

$$M(1) = \log_2 N$$

$$M(2) = 0$$

$$M(3) = 0$$

The call to the "HARM" subroutine is

CALL HARM (A, M, INV, S, IFS, IFERR)

A = Array of complex input, where the real and imaginary parts are in consecutive storage locations. A must be dimensioned 2N.

M = Array containing the logrithm to the base 2 of the length of the summations in directions M(1), M(2), M(3). M is dimensioned 3.

INV = Array computed in "HARM" for bit inverting. INV is dimensioned N/8.

S = Array computed in "HARM" containing the array of sines. S is dimensioned N/8.

IFS =	0 compute INV and S tables
	+1 compute INV and S tables and do Fourier series
	-1 compute INV and S tables and do Fourier transform
	+2 do Fourier series only
	-2 do Fourier transform only

IFERR = Error option. (See Appendix II)

Further amplification of the use of "HARM" can be obtained from the program write-up on file in the Laboratory for Atmospheric and Biological Sciences.

```
C.      TEST OF FAST FOURIER TRANSFORM ROUTINE(HARM).

C       THIS PROGRAM PERFORMS A TRANSFORM ON A SET OF DISCRETE REAL DATA,
C       THEN TAKES THE INVERSE OF THE TRANSFORM TO RECOVER THE ORIGINAL
C       DATA.  THE INPUT DATA REPRESENTS A BLACK BODY TEMPERATURE SPECTRU
C       COMPUTED IN THE 'IRIS' EXPERIMENT.

        DIMENSION A(16384),M(3),INV(2048),S( 2048)
C       SET UP DIMENSIONS
        M(1)=6
        M(2)=0
        M(3)=0
        N1 = 2**M(1)
        N2=2**M(2)
        N3=2**M(3)
        NTOT=N1*N2*N3
        DO 10   I=1,NTOT
     10 A(I)=0.
        READ(2,15) (A(2*I-1), I=1,NTOT)
     15 FORMAT(6E12.5)
     16 WRITE (3,17) (A(2*J-1),J=1,NTOT)
     17 FORMAT(26H1ORIGINAL DATA- REAL PART    //(5F13.5))
        WRITE (3,18) (A(2*J), J=1,NTOT)
     18 FORMAT(26H0ORIGINAL DATA- IMAG PART    //(5F13.5))

C       EVALUATE FOURIER TRANSFORM
C       IF IFS = -1, SETUP INV AND S TABLES AND AND COMPUTE FOURIER SERIES
        IFS = -1
        CALL HARM(A,M,INV,S,IFS,IFERR)
     25 WRITE (3,42) (A(2*J-1),J=1,NTOT)
     42 FORMAT(23H1REAL PART OF TRANSFORM//(5F13.5))
        WRITE (3,44) (A(2*J),J=1,NTOT)
     44 FORMAT(15H0IMAGINARY PART//(5F13.5))

C       TAKE INVERSE
C       IF IFS=+2, COMPUTE FOURIER SERIES.
        IFS = +2
        CALL HARM(A,M,INV,S,IFS,IFERR)
        WRITE (3,70) (A(2*J-1),J=1,NTOT)
     70 FORMAT(60H1REAL PART OF ORIGINAL INPUT OBTAINED BY INVERTING TRANS
       1FORM//(5F13.5))
        WRITE (3,71) (A(2*J),J=1,NTOT)
     71 FORMAT(15H0IMAGINARY PART//(5F13.5))
        PAUSE 77777
        END
```

```
ORIGINAL DATA- REAL PART

     230.09000     229.99000     229.99000     230.01000     230.09000
     230.03000     230.08000     230.08000     230.02000     229.90000
     230.03000     230.10000     229.96000     229.95000     229.99000
     230.10000     229.94000     229.98000     230.03000     230.08000
     230.01000     229.84000     229.99000     230.07000     229.98000
     229.87000     229.85000     230.00000     229.90000     229.74000
     230.05000     229.88000     229.84000     229.47000     229.73000
     229.92000     229.58000     229.89000     230.04000     230.06000
     229.85000     229.74000     229.61000     229.21000     229.80000
     229.52000     229.27000     229.63000     229.69000     229.09000
     229.10000     229.71000     229.63000     229.24000     229.05000
     229.42000     229.57000     229.59000     229.95000     229.88000
     230.00000     229.87000     229.67000     229.64000

ORIGINAL DATA- IMAG PART

        0.            0.            0.            0.            0.
        0.            0.            0.            0.            0.
        0.            0.            0.            0.            0.
        0.            0.            0.            0.            0.
        0.            0.            0.            0.            0.
        0.            0.            0.            0.            0.
        0.            0.         -0.00000      -0.00000      -0.00000
     -0.00000      -0.00000      -0.00000      -0.00000      -0.00000
     -0.00000      -0.00000      -0.00000      -0.00000      -0.00000
     -0.00000      -0.00000      -0.00000      -0.00000      -0.00000
     -0.00000      -0.00000      -0.00000      -0.00000      -0.00000
     -0.00000      -0.00000      -0.00000      -0.00000      -0.00000
     -0.00000      -0.00000      -0.00000      -0.00000
```

```
REAL PART OF TRANSFORM

   229.81062       0.00526      0.04921      0.03292     -0.01639
     0.00301      -0.03302     -0.01326     -0.00515     -0.01372
     0.02211       0.02160     -0.00379     -0.00196     -0.00677
     0.01359       0.02375      0.01425     -0.02524      0.01311
     0.02518      -0.00713      0.00676      0.01149      0.00546
    -0.00945       0.01539     -0.01854      0.00375      0.00945
     0.00907       0.00188      0.01375      0.00188      0.00907
     0.00945       0.00375     -0.01854      0.01539     -0.00945
     0.00546       0.01149      0.00676     -0.00713      0.02518
     0.01311      -0.02524      0.01425      0.02375      0.01359
    -0.00677      -0.00196     -0.00379      0.02160      0.02211
    -0.01372      -0.00515     -0.01326     -0.03302      0.00301
    -0.01639       0.03292      0.04921      0.00526

IMAGINARY PART

     0.           -0.13424     -0.03590      0.02090      0.00464
     0.00215       0.01459     -0.02894      0.00230     -0.03026
    -0.01345       0.00272      0.00449     -0.00504     -0.01928
    -0.03480       0.03250      0.01285      0.00126     -0.04073
     0.00637       0.00557     -0.00827      0.00460      0.00698
     0.00565      -0.00509      0.00575     -0.01536      0.00598
    -0.00021      -0.00435      0.           0.00435      0.00021
    -0.00598       0.01536     -0.00575      0.00509     -0.00565
    -0.00698      -0.00460      0.00827     -0.00557     -0.00637
     0.04073      -0.00126     -0.01285     -0.03250      0.03480
     0.01928       0.00504     -0.00449     -0.00272      0.01345
     0.03026      -0.00230      0.02894     -0.01459     -0.00215
    -0.00464      -0.02090      0.03590      0.13424
```

```
REAL PART OF ORIGINAL INPUT OBTAINED BY INVERTING TRANSFORM

     230.08999     229.98999     229.98999     230.00999     230.08999
     230.02999     230.07999     230.07999     230.01999     229.89999
     230.02999     230.09999     229.95999     229.94999     229.98999
     230.09999     229.93999     229.97999     230.02999     230.07999
     230.00999     229.83999     229.98999     230.06999     229.97999
     229.86999     229.84999     229.99999     229.89999     229.73999
     230.04999     229.88000     229.83999     229.46999     229.72999
     229.91999     229.57999     229.88999     230.03999     230.05999
     229.84999     229.73999     229.60999     229.21000     229.79999
     229.51999     229.26999     229.62999     229.68999     229.08999
     229.09999     229.70999     229.62999     229.23999     229.04999
     229.41999     229.56999     229.58999     229.94999     229.87999
     229.99999     229.86999     229.66999     229.64000

IMAGINARY PART

       0.           -0.00000      -0.00000      -0.00000       0.
       0.00000       0.00000      -0.00000       0.            0.00000
       0.00000      -0.00000       0.           -0.00000      -0.00000
      -0.00000       0.            0.00000       0.00000       0.00000
       0.            0.00000       0.            0.00000       0.
       0.00000      -0.00000       0.            0.            0.00000
       0.00000       0.00000       0.            0.00000       0.00000
       0.00000       0.           -0.00000      -0.00000       0.00000
       0.            0.00000      -0.00000       0.00000       0.
       0.00000       0.00000      -0.00000       0.           -0.00000
      -0.00000      -0.00000       0.           -0.00000       0.00000
      -0.00000       0.           -0.00000       0.00000       0.
       0.           -0.00000      -0.00000      -0.00000
```

APPENDIX II
"FAST FOURIER TRANSFORM" - "HARM"

```
C        HARM  DISCRETE FOURIER TRANSFORM. BASIC FORTRAN IV, IBSYS.
      SUBROUTINE HARM(A,M,INV,S,IFS,    IFERR)
      DIMENSION A(1),INV(1),S(1),N(3),M(3),NP(3),W(2),W2(2),W3(2)
      EQUIVALENCE (N1,N(1)),(N2,N(2)),(N3,N(3))

C     INPUT PARAMETERS TO BE SET BY USER BEFORE ENTERING HARM-

C     A IS A 3-DIMENSIONAL ARRAY OF COMPLEX COEFFICIENTS,
C     OF DIMENSION(N(1),N(2),N(3)).
C     THE A'S ARE STORED WITH REAL PART OF A(I1,I2,I3) IN THE LOCATION
C     WITH INDEX  2*(I3*N(1)*N(2)+I2*N(1)+I1)+1  AND THE IMAGINARY PART
C     IN THE LOCATION IMMEDIATELY FOLLOWING
C     IF THE FOURIER  SERIES   IS REQUESTED,  ARRAY  A  IS REPLACED BY
C     X(J1,J2,J3)=SUM A(K1,K2,K3)*W1**(K1*J1)*W2**(K2*J2)*W3**(K3*J3)
C     SUMMED OVER K1=0,N(1)-1,   K2=0,N(2)-1,   K3=0,N(3)-1
C     WHERE WI=N(I)-TH ROOT OF UNITY.

C     M(I),I=1,2,3,WHERE N(I)=2**M(I) IS THE NO.OF PTS.IN THE I-TH.DIM.
C     THE DIMENSION OF  A  IN THE CALLING PROGRAM SHOULD BE TWICE THE
C     NUMBER OF COMPLEX ELEMENTS OF THE LARGEST  A  ARRAY TO BE PROCESS-
C     ED
C     THE COMPLEX X'S ARE STORED IN THE SAME MANNER AS A.
C
C     IF THE FOURIER TRANSFORM IS REQUESTED, THE ARGUMENT  A  IS TAKEN
C     TO BE  X  AND IS REPLACED BY THE ARRAY  A  SATISFYING THE FOURIER
C     SERIES.

C     LET MT=MAX(M(1),M(2),M(3))-2, NT=2**MT, WITH  M  BEING THE  M
C     GIVEN WHEN THE TABLES ARE SET.

C     S(J)=SIN(J*PI/(2*NT ) ), J = 1,2,3,...,NT-1.

C     INV(J+1)=WORD CONTAINING BITS OF J IN INVERTED ORDER IN ITS
C     RIGHTMOST MT BIT POSITIONS, FOR J = 0,1,2,...,NT-1.
C
C     LET  IFS=0 TO SET UP SIN AND INV TABLES.
C          IFS=+1 TO SET UP SIN AND INV TABLES AND DO FOURIER SERIES.
C          IFS=-1 TO SET UP SIN AND INV TABLES AND DO FOURIER TRANSFORM.
C          IFS=+2  TO DO FOURIER SERIES ONLY.
C          IFS=-2 TO DO FOURIER TRANSFORM ONLY.

C     ONE DOES NOT HAVE TO REPEAT THE CALL TO 'HARM' WITH IFS=0,+1,-1
C     IF ONE DOES NOT CHANGE THE MAXIMUM  M.
C
C     IFERR=0 IF THE ARGUMENTS M ARE O.K.
C
C     IFERR=1 IF THERE IS AN ERROR IN CALLING 'HARM'
C     IF IFS=0,+1,-1,  IT MEANS THAT THE MAXIMUM M IS GREATER THAN 20
C     OR LESS THAN 3
C     IF IFS=+-2, IT MEANS THAT A SUFFICIENTLY LARGE SIN AND INV TABLE
C     HAS NOT BEEN COMPUTED.  ONE MUST CALL  'HARM' WITH  IFS=0,+-1 AND
C     WITH A MAX M(I) GREATER THAN OR EQUAL TO THE MAX M(I) FOR WHICH A
C     FOURIER TRANSFORM IS TO BE COMPUTED.
C
C     IFERR=-1 IF ONE IS CALLING ON 'HARM' WITH IFS=0,+-1  TO COMPUTE
```

```
C       SIN, INV TABLES WHICH IT ALREADY HAS COMPUTED ON A PREVIOUS
C       CALL TO HARM WITH THE SAME MAXIMUM M
C
   10 IF(IABS(IFS)-1) 900,900,12
   12 MTT=MAXO(M(1),M(2),M(3)) -2
      ROOT2 = SQRT(2.)
      IF (MTT-MT ) 14,14,13
   13 IFERR=1
    1 RETURN
   14 IFERR=0
      M1=M(1)
      M2=M(2)
      M3=M(3)
      N1=2**M1
      N2=2**M2
      N3=2**M3
      IF (IFS) 16,1,20
C       TO CALCULATE TRANSFORM REPLACE A BY CONJG(A)/N
   16 NTOT = N1*N2*N3
      FN = NTOT
      DO 18 I=1,NTOT
      A(2*I-1) = A(2*I-1)/FN
   18 A(2*I) = -A(2*I)/FN
   20 NP(1)=N1*2
      NP(2)= NP(1)*N2
      NP(3)=NP(2)*N3
      DO 250 ID=1,3
      IL = NP(3)-NP(ID)
      IL1 = IL+1
      MI = M(ID)
      IF (MI)250,250,30
   30 IDIF=NP(ID)
      KBIT=NP(ID)
      MEV = 2*(MI/2)
      IF (MI - MEV )60,60,40
C       M IS ODD. DO L=1 CASE
   40 KBIT=KBIT/2
      KL=KBIT-2
      DO 50 I=1,IL1,IDIF
      KLAST=KL+I
      DO 50 K=I,KLAST,2
      KD=K+KBIT
C       DO ONE STEP WITH L=1,J=0
C       A(K)=A(K)+A(KD)
C       A(KD)=A(K)-A(KD)
C
      T=A(KD)
      A(KD)=A(K)-T
      A(K)=A(K)+T
      T=A(KD+1)
      A(KD+1)=A(K+1)-T
   50 A(K+1)=A(K+1)+T
      IF (MI - 1)250,250,52
   52 LFIRST =3
C       DEF - JLAST = 2**(L-2) -1
      JLAST=1
      GO TO 70
C     M IS EVEN
   60 LFIRST = 2
      JLAST=0
   70 DO 240 L=LFIRST,MI,2
      JJDIF=KBIT
      KBIT=KBIT/4
      KL=KBIT-2
C     DO FOR J=0
      DO 80 I=1,IL1,IDIF
      KLAST=I+KL
      DO 80 K=I,KLAST,2
      K1=K+KBIT
      K2=K1+KBIT
      K3=K2+KBIT
C
C     DO TWO STEPS WITH J=0
C     A(K)=A(K)+A(K2)
C     A(K2)=A(K)-A(K2)
C     A(K1)=A(K1)+A(K3)
C     A(K3)=A(K1)-A(K3)
C
C     A(K)=A(K)+A(K1)
C     A(K1)=A(K)-A(K1)
C     A(K2)=A(K2)+A(K3)*I
C     A(K3)=A(K2)-A(K3)*I
C
      T=A(K2)
      A(K2)=A(K)-T
      A(K)=A(K)+T
      T=A(K2+1)
      A(K2+1)=A(K+1)-T
      A(K+1)=A(K+1)+T
C
      T=A(K3)
      A(K3)=A(K1)-T
      A(K1)=A(K1)+T
      T=A(K3+1)
      A(K3+1)=A(K1+1)-T
      A(K1+1)=A(K1+1)+T
C
      T=A(K1)
      A(K1)=A(K)-T
      A(K)=A(K)+T
      T=A(K1+1)
      A(K1+1)=A(K+1)-T
      A(K+1)=A(K+1)+T
C
      R=-A(K3+1)
      T = A(K3)
      A(K3)=A(K2)-R
      A(K2)=A(K2)+R
      A(K3+1)=A(K2+1)-T
   80 A(K2+1)=A(K2+1)+T
      IF (JLAST) 235,235,82
   82 JJ=JJDIF    +1
```

```
C
C       DO FOR J=1
        ILAST= IL +JJ
        DO 85 I = JJ,ILAST,IDIF
        KLAST = KL+I
        DO 85 K=I,KLAST,2
        K1 = K+KBIT
        K2 = K1+KBIT
        K3 = K2+KBIT
C       LETTING W=(1+I)/ROOT2,W3=(-1+I)/ROOT2,W2=I,
C       A(K)=A(K)+A(K2)*I
C       A(K2)=A(K)-A(K2)*I
C       A(K1)=A(K1)*W+A(K3)*W3
C       A(K3)=A(K1)*W-A(K3)*W3
C
C       A(K)=A(K)+A(K1)
C       A(K1)=A(K)-A(K1)
C       A(K2)=A(K2)+A(K3)*I
C       A(K3)=A(K2)-A(K3)*I
C
        R =-A(K2+1)
        T = A(K2)
        A(K2) = A(K)-R
        A(K)  = A(K)+R
        A(K2+1)=A(K+1)-T
        A(K+1)=A(K+1)+T
C
        AWR=A(K1)-A(K1+1)
        AWI = A(K1+1)+A(K1)
        R=-A(K3)-A(K3+1)
        T=A(K3)-A(K3+1)
        A(K3)=(AWR-R)/ROOT2
        A(K3+1)=(AWI-T)/ROOT2
        A(K1)=(AWR+R)/ROOT2
        A(K1+1)=(AWI+T)/ROOT2
        T= A(K1)
        A(K1)=A(K)-T
        A(K)=A(K)+T
        T=A(K1+1)
        A(K1+1)=A(K+1)-T
        A(K+1)=A(K+1)+T
        R=-A(K3+1)
        T=A(K3)
        A(K3)=A(K2)-R
        A(K2)=A(K2)+R
        A(K3+1)=A(K2+1)-T
   85   A(K2+1)=A(K2+1)+T
        IF(JLAST-1) 235,235,90
   90   JJ= JJ + JJDIF

C       NOW DO THE REMAINING J'S
        DO 230 J=2,JLAST
C
C       FETCH W'S
C       DEF- W=W**INV(J), W2=W**2, W3=W**3
   96   I=INV(J+1)
   98 IC=NT-I
      W(1)=S(IC)
      W(2)=S(I)
      I2=2*I
      I2C=NT-I2
      IF(I2C)120,110,100
C
C     2*I IS IN FIRST QUADRANT
  100 W2(1)=S(I2C)
      W2(2)=S(I2)
      GO TO 130
  110 W2(1)=0.
      W2(2)=1.
      GO TO 130
C
C     2*I IS IN SECOND QUADRANT
  120 I2CC = I2C+NT
      I2C=-I2C
      W2(1)=-S(I2C)
      W2(2)=S(I2CC)
  130 I3=I+I2
      I3C=NT-I3
      IF(I3C)160,150,140
C
C     I3 IN FIRST QUADRANT
  140 W3(1)=S(I3C)
      W3(2)=S(I3)
      GO TO 200
  150 W3(1)=0.
      W3(2)=1.
      GO TO 200
C
  160 I3CC=I3C+NT
      IF(I3CC)190,180,170
C
C     I3 IN SECOND QUADRANT
  170 I3C=-I3C
      W3(1)=-S(I3C)
      W3(2)=S(I3CC)
      GO TO 200
  180 W3(1)=-1.
      W3(2)=0.
      GO TO 200
C
C     3*I IN THIRD QUADRANT
  190 I3CCC=NT+I3CC
      I3CC = -I3CC
      W3(1)=-S(I3CCC)
      W3(2)=-S(I3CC)
  200 ILAST=IL+JJ
      DO 220 I=JJ,ILAST,IDIF
      KLAST=KL+I
      DO 220 K=I,KLAST,2
      K1=K+KBIT
      K2=K1+KBIT
      K3=K2+KBIT
```

```
C
C      DO TWO STEPS WITH J NOT 0
C      A(K)=A(K)+A(K2)*W2
C      A(K2)=A(K)-A(K2)*W2
C      A(K1)=A(K1)*W+A(K3)*W3
C      A(K3)=A(K1)*W-A(K3)*W3
C
C      A(K)=A(K)+A(K1)
C      A(K1)=A(K)-A(K1)
C      A(K2)=A(K2)+A(K3)*I
C      A(K3)=A(K2)-A(K3)*I
C
       R=A(K2)*W2(1)-A(K2+1)*W2(2)
       T=A(K2)*W2(2)+A(K2+1)*W2(1)
       A(K2)=A(K)-R
       A(K)=A(K)+R
       A(K2+1)=A(K+1)-T
       A(K+1)=A(K+1)+T
C
       R=A(K3)*W3(1)-A(K3+1)*W3(2)
       T=A(K3)*W3(2)+A(K3+1)*W3(1)
       AWR=A(K1)*W(1)-A(K1+1)*W(2)
       AWI=A(K1)*W(2)+A(K1+1)*W(1)
       A(K3)=AWR-R
       A(K3+1)=AWI-T
       A(K1)=AWR+R
       A(K1+1)=AWI+T
       T=A(K1)
       A(K1)=A(K)-T
       A(K)=A(K)+T
       T=A(K1+1)
       A(K1+1)=A(K+1)-T
       A(K+1)=A(K+1)+T
       R=-A(K3+1)
       T=A(K3)
       A(K3)=A(K2)-R
       A(K2)=A(K2)+R
       A(K3+1)=A(K2+1)-T
   220 A(K2+1)=A(K2+1)+T
C      END OF I AND K LOOPS
   230 JJ=JJDIF+JJ
C      END OF J-LOOP
   235 JLAST=4*JLAST+3
   240 CONTINUE
C      END OF  L  LOOP
   250 CONTINUE
C      END OF  ID  LOOP
C
C      WE NOW HAVE THE COMPLEX FOURIER SUMS BUT THEIR ADDRESSES ARE
C      BIT-REVERSED.  THE FOLLOWING ROUTINE PUTS THEM IN ORDER
       NTSQ=NT*NT
       M3MT=M3-MT
   350 IF(M3MT) 370,360,360
C      M3 GR. OR EQ. MT
   360 IGO3=1
       N3VNT=N3/NT
```

```
      MINN3=NT
      GO TO 380
C     M3 LESS THAN MT
  370 IGO3=2
      N3VNT=1
      NTVN3=NT/N3
      MINN3=N3
  380 JJD3 = NTSQ/N3
      M2MT=M2-MT
  450 IF (M2MT)470,460,460
C     M2 GR. OR EQ. MT
  460 IGO2=1
      N2VNT=N2/NT
      MINN2=NT
      GO TO 480
C     M2 LESS THAN MT
  470 IGO2 = 2
      N2VNT=1
      NTVN2=NT/N2
      MINN2=N2
  480 JJD2=NTSQ/N2
      M1MT=M1-MT
  550 IF(M1MT)570,560,560
C     M1 GR. OR EQ. MT
  560 IGO1=1
      N1VNT=N1/NT
      MINN1=NT
      GO TO 580
C     M1 LESS THAN MT
  570 IGO1=2
      N1VNT=1
      NTVN1=NT/N1
      MINN1=N1
  580 JJD1=NTSQ/N1
  600 JJ3=1
      J=1
      DO 880 JPP3=1,N3VNT
      IPP3=INV(JJ3)
      DO 870 JP3=1,MINN3
      GO TO (610,620),IGO3
  610 IP3=INV(JP3)*N3VNT
      GO TO 630
  620 IP3=INV(JP3)/NTVN3
  630 I3=(IPP3+IP3)*N2
  700 JJ2=1
      DO 870 JPP2=1,N2VNT
      IPP2=INV(JJ2)+I3
      DO 860 JP2=1,MINN2
      GO TO (710,720),IGO2
  710 IP2=INV(JP2)*N2VNT
      GO TO 730
  720 IP2=INV(JP2)/NTVN2
  730 I2=(IPP2+IP2)*N1
  800 JJ1=1
      DO 860 JPP1=1,N1VNT
      IPP1=INV(JJ1)+I2
      DO 850 JP1=1,MINN1
      GO TO (810,820),IGO1
  810 IP1=INV(JP1)*N1VNT
      GO TO 830
  820 IP1=INV(JP1)/NTVN1
  830 I=2*(IPP1+IP1)+1
      IF (J-I) 840,845,845
  840 T=A(I)
      A(I)=A(J)
      A(J)=T
      T=A(I+1)
      A(I+1)=A(J+1)
      A(J+1)=T
  845 CONTINUE
  850 J=J+2
  860 JJ1=JJ1+JJD1
C     END OF JPP1 AND JP2
  870 JJ2=JJ2+JJD2
C     END OF JPP2 AND JP3 LOOPS
  880 JJ3 = JJ3+JJD3
C     END OF JPP3 LOOP
      IF(IFS) 882,1,1

C     DOING TRANSFORM. REPLACE A BY CONJG(A).
  882 DO 884 I=1,NTOT
  884 A(2*I) = -A(2*I)
      GO TO 1
C     RETURN
C
C     THE FOLLOWING PROGRAM COMPUTES THE SIN AND INV TABLES.
C
  900 MT=MAXO(M(1),M(2),M(3)) -2
      MT = MAXO(2,MT)
  904 IF (MT-20)906,906,905
  905 IFERR = 1
      GO TO 1
C     RETURN
  906 IFERR=0
      NT=2**MT
      NTV2=NT/2
C     SET UP SIN TABLE
C     THETA=PIE/2**(L+1) FOR L=1
  910 THETA=.7853981634
C     JSTEP=2**(MT-L+1) FOR L=1
      JSTEP=NT
C     JDIF=2**(MT-L) FOR L=1
      JDIF=NTV2
      S(JDIF)=SIN(THETA)
      DO 950 L=2,MT
      THETA=THETA/2.
      JSTEP2=JSTEP
      JSTEP=JDIF
      JDIF=JSTEP/2
      S(JDIF)=SIN(THETA)
      JC1=NT-JDIF
      S(JC1)=COS(THETA)
      JLAST=NT-JSTEP2
      IF(JLAST - JSTEP) 950,920,920
  920 DO 940 J=JSTEP,JLAST,JSTEP
      JC=NT-J
      JD=J+JDIF
  940 S(JD)=S(J)*S(JC1)+S(JDIF)*S(JC)
  950 CONTINUE
C
C     SET UP INV(J) TABLE
C
  960 MTLEXP=NTV2
C     MTLEXP=2**(MT-L). FOR L=1
      LM1EXP=1
C     LM1EXP=2**(L-1). FOR L=1
      INV(1)=0
      DO 980 L=1,MT
      INV(LM1EXP+1) = MTLEXP
      DO 970 J=2,LM1EXP
      JJ=J+LM1EXP
  970 INV(JJ)=INV(J)+MTLEXP
      MTLEXP=MTLEXP/2
  980 LM1EXP=LM1EXP*2
      IF(IFS) 12,1,12
C     RETURN
      END
```