# SHARE PROGRAM LIBRARY AGENCY


PROGRAM NUMBER

13401

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA
(305) - 284-6257

**CONTRIBUTED PROGRAM LIBRARY SUBMITTAL**
**(for IBM S/360, 1130 and 1800)**

SHARE Program Library Agency
Triangle Universities Computation Center
P. O. Box 12076
Research Triangle Park, N. C. 27709

This form should be completed and submitted with the program package to PID at the address shown above. Standards and instructions for submitting programs are in your *User Group Reference Manual* or the *Contributed Program Submittal Standards Manual* available from PID.

(1) **Program Order Number** *(to be filled in by PID)* . . . . . 360D-13.4.001

(2) **System Type** *(machine)* . . . . . 

(3) **Search Key** . . . . . COOLEY-TUKEY
FAST
FOURIER TRANSFORM
FOURT

(4) **Name of Author** *(if different than submitter's)* . . . . . 

(5) **Submitter's Name** *(direct technical inquiries to)* . . . . Norman Brenner

(6) **Submitter's Address** . . . . . 

Norman Brenner, Rm. 36-005
IBM Research
P.O. Box 218
Yorktown Heights, NY 10598

(7) **Title of Program** . . . . Cooley-Tukey Fast Fourier Transform--FOURT

(8) **Submitter's User Group Affiliation Code and Installation Code** . . . . . MIT

(9) **Submitter's Own Program Identification and Suffix** *(optional)* . . . . . FFTT

(10) **Primary Subject Code** . . . . . 13.4

(11) **Secondary Subject Codes** . . . . . 43.3 44.0

(12) **Operating or Monitor System Required** . . . . . 

(13) **New or Revision Code** *(if revision, show prior Program Order Number in item 1)* . . . . . R

(14) **Year Completed** . . . . . 68

(15) **Date of Submittal** . . . . . 092168

(16) **Documentation** *(number of original pages submitted)* . . . . . 7

(17) **Abstract** *(should contain sufficient information for a reader to determine the value of the program). Listed on the reverse side of this form are subjects which may serve as a guide for a descriptive abstract.*

**DISCLAIMER**

Triangle Universities Computation Center (TUCC) serves solely as the distribution agent for contributed programs and does not test or maintain them. They are distributed essentially in the original form submitted by the author. Neither TUCC nor SHARE, INC., make any warranty, expressed or implied, as to the documentation, function, or performance of the contributed programs.

## CONTRIBUTED PROGRAM LIBRARY SUBMITTAL FORM

**Subject Guide**

a. Purpose
b. Programming Language used
c. Version and modification level or release number of IBM Programming System used, or program order number for non-IBM authored program used
d. Field of application
e. Type of routine (main program, subroutine, etc.)
f. Specific description of machine requirements
g. Engineering Changes (EC) level of equipment (if pertinent)

**ABSTRACT**

SUBROUTINE FOURT(DATA,N,NDIM,ISIGN,IFORM,WORK,NWORK)

The Cooley-Tukey Fast Fourier Transform in USASI Basic Fortran.

For N data points , FOURT runs in time proportional to N log N, while pre-FFT methods run in N**2. For large N, this is seconds versus hours. The FFT is the fastest and most accurate way of computing periodograms of time series, convolutions and correlations, Fourier coefficients, digital filters, partial differential equations, etc. See IEEE Transactions on Audio and Electroacoustics (June 1967), a special issue on the FFT. DATA is a multi-dimensional array of floating-point data which is transformed and replaced with the transform values. It may be complex, real or conjugate-symmetric complex (IFORM=1,0 or -1). The latter cases run in half the time and half the storage of the first. N is an array of length NDIM giving the dimensions of DATA. NDIM is unrestricted; each N(I) may be any positive integer, even prime. ISIGN is +1 or -1, the transform direction. WORK is a complex array of length NWORK; it may be quite short, depending on the N(I).

*(Please attach additional pages if necessary)* . . . . . **Total pages attached** 0

**Permission to Publish**

*"I hereby give anyone permission to reprint, reproduce, and distribute this program to anyone else."*

(18) **Signature of Submitter and Date** [signature]

(19) **Signature of Installation Addressee** _____

T4SF

Table of Contents





## Deck Key Sheet

Nine Fortran decks are present in the FOURT package.

| Deck Name | ID in cc 73-75 | Last Seq. no. in cc 77-79 (first is 1) |
|---|---|---|
| (main program for testing) | TFF | 134 |
| FOURT | FFT | 146 |
| FACTR | FAC | 24 |
| SMFAC | SMF | 46 |
| SYMRV | SYM | 39 |
| ASMRV | ASM | 35 |
| COOL | COO | 128 |
| GOERT | GOE | 87 |
| FIXRL | FIX | 111 |

1. Title — Fast Fourier Transform--FOURT

2. Language — USASI Basic Fortran

3. Programmer — Norman Brenner

4. Date — 20 September 1968

5. Description

FOURT is a version of the Cooley-Tukey Fast Fourier Transform. It computes the discrete Fourier transform of complex or real floating-point arrays, multi-dimensional, whose size is arbitrary. For example, it can transform an array dimensioned 50x40.

The finite discrete Fourier transform of an array DATA is defined

$$TRAN(K1,K2,\ldots) = \sum_{J1=1}^{N1} \sum_{J2=1}^{N2} \ldots DATA(J1,J2,\ldots) \exp(ISIGN*2\pi i* [(J1-1)(K1-1)/N1 + (J2-1)(K2-1)/N2 + \ldots]), \quad (ISIGN = \pm 1)$$

for K1 from 1 to N1, K2 from 1 to N2, etc. It is related to the more usual integral definition over the same finite range by Filon's Integration Formula (Handbook of Mathematical Functions, National Bureau of Standards, Washington, 1965, pp. 890-1).

Let the total number of points be NTOT = N1*N2*...*N(NDIM), where both DATA and TRAN are NDIM-dimensional arrays. Pre-Fast Fourier Transform methods that directly implement the above summation run in time proportional to $NTOT^2$. The Fast Fourier Transform runs in time NTOT log(NTOT). For several thousand data points, this is the difference between hours and seconds. FFT is also more accurate.

FOURT accepts complex, real or conjugate symmetric data arrays. The latter two cases require approximately half as much storage and half as much running time as the correspondingly dimensioned complex array. The amount of working storage needed is usually quite small.

6. Entry/Calling Sequence

To transform in place a complex array dimensioned 50 by 40:

With COMPLEX arithmetic

```
      DIMENSION DATA(50,40),N(2),WORK(10)
      COMPLEX DATA,WORK
      DATA N/50,40/
      DO 10 J1=1,50
      DO 10 J2=1,40
10    DATA(J1,J2)=complex value
      CALL FOURT(DATA,N,2,-1,1,WORK,10)
```



Without COMPLEX arithmetic

```
      DIMENSION DATA(2,50,40),N(2),WORK(2,10)
      N(1)=50
      N(2)=40
      DO 10 J1=1,50
      DO 10 J2=1,40
      DATA(1,J1,J2)=real part
10    DATA(2,J1,J2)=imaginary part
      CALL FOURT(DATA,N,2,-1,+1,WORK,10)
```

In general, CALL FOURT(DATA,N,NDIM,ISIGN,IFORM,WORK,NWORK).

DATA is a floating-point array dimensioned N(1) by N(2) by ... by N(NDIM). It is complex or real; complex data must have adjacent real and imaginary parts. The transform values are returned to array DATA, replacing the input. They are complex or real, depending on DATA. See below.

N is an integer array of length NDIM. Each N(I) may be any positive integer, even a prime. For fastest running, however, each N(I) should be highly composite. NDIM, the number of dimensions, must be positive and may be as large as desired.

ISIGN = -1 or +1, the direction of the transform. If a set of data are -1 transformed, and the transform values +1 transformed (or vice versa), the original data reappear, multiplied by NTOT.

IFORM = +1, 0 or -1, as DATA is complex, real or conjugate-symmetric complex. The transform values are complex, conjugate-symmetric complex or real, respectively.

WORK is a complex array used for working storage. Its length, NWORK, is variable, but can generally be very small. It can be at worst as large as the largest N(I). FOURT computes the length of needed working storage as shown below. If NWORK is not at least as big, an error message is written and control returned to the calling program. For a one-dimensional array, factor N(1) (use N(1)/2 if IFORM = 0 or -1) into its prime factors. NWORK must be as big as the largest prime factor. Secondly, NWORK must be at least as big as the quotient of N(1) by the largest square dividing it (again, use N(1)/2 if IFORM = 0 or -1). For example, let IFORM = 1, NDIM = 1 and N(1) = 1960 = 2x2x2x5x7x7. It is clear that 196 is the largest square dividing N(1). Hence, NWORK must be at least max(7,1960/196) = 10. For multi-dimensional arrays, compute NWORK for each N(I) and take the largest such NWORK (always take N(2), N(3), for all IFORM).

If IFORM = 0 or -1 and N(1) is not even, an error message is written and control returned to the calling program.

If a set of real data are to be transformed, two courses are possible. One can disguise the real data as complex by appending zero imaginary parts. This is wasteful both of storage and of running time. Alternately, one can note that the transform of a real array, tho complex and ostensibly requiring twice as much storage, has conjugate symmetry. That is,

TRAN(K1,K2,...) = conjugate(TRAN(K1',K2',...))

where $K_i' = 1$ if $K_i = 1$, and $K_i' = N(i)+2-K_i$ if $K_i > 1$. (Note the 1-origin indexing). Since for $K_i = 1$ and $K_i = N(i)/2+1$, $K_i' = K_i$, those values are self-conjugate, i.e. real. For example, the transform of a real array of length ten has the following symmetry:

| K | TRAN(K) | |
|---|---|---|
| 1 | A | real |
| 2 | B | complex |
| 3 | C | " |
| 4 | D | " |
| 5 | E | " |
| 6 | F | real |
| 7 | E* | complex |
| 8 | D* | " |
| 9 | C* | " |
| 10 | B* | " |

Hence it would be sufficient to return only TRAN(K), K = 1,6 to the calling program since K = 7,10 are easily obtainable. For consistency, FOURT returns TRAN(1) and TRAN(6) as complex numbers. Note that six complex numbers, requiring twelve units of storage, have been returned for ten input real numbers.

For a multi-dimensional real array DATA sized N(1) by N(2) by ... by N(NDIM), FOURT returns a complex array sized [N(1)/2+1] by N(2) by ... by N(NDIM), which takes up slightly more storage.

For example, do our previous example for real input:

With COMPLEX arithmetic

```
      DIMENSION DATA(50,40),TRAN(26,40),N(2),WORK(10)
      COMPLEX TRAN,WORK
      EQUIVALENCE (DATA,TRAN)
      DATA N/50,40/
      DO 10 J1=1,50
      DO 10 J2=1,40
10    DATA(J1,J2)=real value
      CALL FOURT(DATA,N,2,-1,0,WORK,10)
```



Without COMPLEX arithmetic

```
      DIMENSION DATA(50,40),TRAN(2,26,40),N(2),WORK(2,10)
      EQUIVALENCE (DATA,TRAN)
      N(1)=50
      N(2)=40
      DO 10 J1=1,50
      DO 10 J2=1,40
10    DATA(J1,J2)=real value
      CALL FOURT(DATA,N,2,-1,0,WORK,10)
```

Note that only one large area has been allocated, a block of 2080 floating-point numbers, known under two names: DATA and TRAN. Initially, the program would compute or read in real values and store them in DATA. After the above transformation, the complex transform values would be referred to as TRAN(K1,K2) (or TRAN(1 or 2,K1,K2)). For K1 > 26, we have from the rule given above that

$$TRAN(K1,K2) = \begin{cases} \text{conjugate}(TRAN(52-K1,1)), & K2 = 1 \\ \text{conjugate}(TRAN(52-K1,42-K2)), & K2 > 1 \end{cases}$$

To inverse transform a -1 transform, one performs a +1 transform and divides by 2000 as before. Continuing from above:

```
      CALL FOURT(DATA,N,2,+1,-1,WORK,10)
      DO 20 J1=1,50
      DO 20 J2=1,40
20    DATA(J1,J2)=DATA(J1,J2)/2000.
```

Note that IFORM = -1. DATA now contains the original input plus some roundoff error.

Similarly, if one starts off with a conjugate-symmetric complex array, he would transform it into a real array with IFORM = -1 and back again with IFORM = 0.

7. Return

Normal

8. Storage/Common

The program is $5630_8$ ($2968_{10}$) on the CDC 3300. No COMMON storage. 650 Fortran statements.

9. Algorithm

See the IEEE Transactions on Audio and Electroacoustics (June 1967), a special issue on the FFT and its applications.

10. Special Cautions and Features

Real or complex data, as explained. Arbitrary number of data, as explained.

11. Timing information

Let $N_{total}$ = N(1)xN(2)x...xN(NDIM) (use N(1)/2 if IFORM = 0 or -1). Factor $N_{total}$ into its prime factors, $N_{total} = f_1 f_2 \cdots f_n$. Running time on the CDC 3300 (floating multiply of 16 microseconds) is:

$$T = N_{total} \left(154 + \sum_{i=1}^{n} COST(f_i)\right) \text{ microseconds } \pm 13 \text{ percent,}$$

where a factor of 2 costs 134 microseconds, a factor of 3 costs 275, and a prime factor $f \geq 5$ costs 486+72f. Factors of 2 and 3 are almost equally fast: N = 243 is calculated to run in .370 seconds and N = 256 in .314 (measured times are .354 and .290), a difference of only twenty percent.

12. Error messages

The following messages are written on unit 6:

ERROR IN FOURT. NWORK = nnnnn IS TOO SMALL FOR N(iii) = nnnnn, WHOSE CENTER = ccccc, AND WHOSE PRIME FACTORS ARE-- fffff fffff fffff fffff

(The "center" is the quotient of N(1) and the largest square dividing it.)

ERROR IN FOURT. IFORM = xx (REAL OR HALF-COMPLEX), BUT N(1) IS NOT EVEN. DIMENSIONS = nnnnn nnnnn nnnnn nnnnn

Control is then returned to the calling program.

13. Subroutines used

Internal subroutines used are FACTR, SMFAC, SYMRV, ASMRV, COOL, GOERT, and FIXHL. Sine and cosine values are computed by SIN.

FACTR may be of use to the user. It is called by CALL FACTR(N,IFACT,NFACT). N is input, along with an integer array IFACT. On return, IFACT contains the prime factors of N in increasing order, NFACT in number. That is, N = IFACT(1) * ... * IFACT(NFACT).

14. Accuracy

An upper bound for the root-mean-square of the relative errors of the transform values is

$$3 \cdot 2^{-b} \cdot \sum_{i=1}^{n} f_i^{3/2}$$

where b is the number of bits in the floating point fraction (36 in the CDC 3300, 23 in the IBM System/360).

Because fewer arithmetic operations are performed by the FFT than by conventional programs directly implementing the summation, the FFT builds up less roundoff error. See Gentleman and Sande, "FFT--for Fun and Profit", 1966 FJCC Proceedings, Spartan Books, 1966.



15. Comments on usage

i. If the input DATA(J) are considered to be located at times (J-1)*I, then the transform values TRAN(K) are located at frequencies (K-1)*F, F = 2π/(NT). By periodicity, all frequencies above the "foldover frequency" π/I may be identified with the corresponding frequency 2π/T lower. This is true for each dimension separately.

ii. The number of output transform values must be the same as the number of input data, due to the nature of the algorithm. If the DATA(J) are padded out with zeroes to a longer array, the transform values will be interpolated as given in i. with the new N--"spectrum broadening".

iii. This program was very carefully written to be compatible with as many Fortran compilers as possible. No DO-loop ever has a larger initial value than final, no multiple subscripts are used, and complex arithmetic is performed by operating on the real and imaginary parts separately. Even if complex arithmetic is available, rewriting is not recommended, so long as real and imaginary parts are adjacent in storage. Finally, the order of dimension sizes N(1), N(2), etc., is from that subscript varying fastest in storage order to that varying slowest. In most Fortrans, this is left-to-right order; in PL/I, it is right-to-left.

iv. Sine and cosine values are computed by Singleton's recursion (see the Audio Transactions cited). Tabling them would speed up the program less than ten percent, while calculating them as needed would slow the program by fifty percent, both at a small gain in accuracy.

v. In testing the program with a simple function such as a square wave, do not expect the transform to be that given by the infinite integral Fourier transform. It will be that given by the summation definition above.

A good test function is a ramp. Define DATA(J) = $\alpha + \beta*(J-1)$, for complex $\alpha$ and $\beta$, J from 1 to N. Then

$$TRAN(K) = \begin{cases} \alpha N + \beta\frac{N(N-1)}{2}, & K = 1, \\ -\beta\frac{N}{2}\left(1 + ISIGN*i*\cot\frac{\pi(K-1)}{N}\right), & K > 1. \end{cases}$$

Similarly, in two dimensions, let

$$DATA(J1,J2) = \alpha + \beta*(J1-1 + N1*(J2-1)), \quad 1 \leq J1 \leq N1, \; 1 \leq J2 \leq N2.$$

Define N = N1*N2; then

$$TRAN(K1,K2) = \begin{cases} \alpha N + \beta\frac{N(N-1)}{2}, & K1 = 1, \; K2 = 1, \\ -\beta\frac{N}{2}\left(1 + ISIGN*i*\cot\frac{\pi(K1-1)}{N1}\right), & K1 > 1, \; K2 = 1, \\ -\beta\frac{N*N1}{2}\left(1 + ISIGN*i*\cot\frac{\pi(K2-1)}{N2}\right), & K1 = 1, \; K2 > 1, \\ 0, & K1 > 1, \; K2 > 1. \end{cases}$$

vi. A double precision version of this program may be obtained by changing the name to DFOURT, declaring all floating-point variables double precision [360 Fortran: IMPLICIT REAL*8 (A-H,O-Z)] in all the subroutines, changing SIN to DSIN and setting TWOPI = 6.2831853071795865*FLOAT(ISIGN)

vii. If DATA is always a power of two size, use program FOUR2. If it is always one-dimensional complex, use program FOURG. If it is always one-dimensional complex, sized a power of two, use program FOUR1. All are shorter programs than FOURT. If DATA is so big that it must be kept on direct access storage, use program FOR2D.

Since arrays can be padded with zeroes to attain a length which is a power of two, there are only two reasons for using FOURT over FOUR2:

1. Convenience

2. Interpolation to specific frequency values, or by transforming, padding with zeroes, and retransforming, to specific time values.

viii. Calling FOURT with arbitrary N can cause running time to mushroom if N contains a large prime factor; also WORK may not be large enough. For example, consider the following running times for a one-dimensional complex array, computed from the timing formula:

| N | Factors | Running time (sec) | NWORK needed |
|---|---|---|---|
| 1020 | $2^2 \cdot 3 \cdot 5 \cdot 17$ | 3.32 | 255 |
| 1021 | 1021 | 75.71 | 1021 |
| 1022 | $2 \cdot 7 \cdot 73$ | 7.17 | 1022 |
| 1023 | $3 \cdot 11 \cdot 31$ | 4.53 | 1023 |
| 1024 | $2^{10}$ | 1.53 | 2 |
| 1025 | $5^2 \cdot 41$ | 5.42 | 41 |
| 1026 | $2 \cdot 3^3 \cdot 19$ | 3.04 | 114 |
| 1027 | $13 \cdot 79$ | 2.96 | 1027 |
| 1028 | $2^2 \cdot 257$ | 19.96 | 257 |
| 1029 | $3 \cdot 7^3$ | 3.50 | 21 |
| 1030 | $2 \cdot 5 \cdot 103$ | 9.31 | 1030 |

By comparison, the fastest non-FFT program known to the author (based on Goertzel's Algorithm) will transform one-dimensional complex arrays in time

T = N (540+76 N) microseconds

on the CDC 3300, which gives 79.62 to 31.18 seconds for the cases above.