OAK RIDGE NATIONAL LABORATORY
operated by
UNION CARBIDE CORPORATION
for the
U.S. ATOMIC ENERGY COMMISSION

ORNL-TM-1778

COPY NO. - 51

DATE - July 12, 1967

A DIGITAL FILTERING TECHNIQUE FOR EFFICIENT FOURIER TRANSFORM CALCULATIONS

S. J. Ball

ABSTRACT

A method for calculating Fourier transforms is presented which capitalizes on the similarity between the Fourier transform integral and the convolution integral. The method is implemented by an efficient digital computer code, which uses a digital simulation of a sine-cosine filter. Although this method is mathematically equivalent to a direct numerical calculation of the Fourier transform, it takes about one-third the time required for the direct integration. The code has been used extensively for calculating frequency response functions from pulse response test data and from cross-and auto-correlation functions.

NOTICE This document contains information of a preliminary nature and was prepared primarily for internal use at the Oak Ridge National Laboratory. It is subject to revision or correction and therefore does not represent a final report.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

PAGES 1 to 2

WERE INTENTIONALLY

LEFT BLANK

CONTENTS

1. INTRODUCTION

The similarity between the convolution integral[1] and the Fourier transform integral has probably been recognized by mathematicians for many years; however, an application of this similarity to the task of calculating Fourier transforms numerically has apparently not been widely appreciated. The basic method described in this paper was reported by Broome and Cooper[2] who implemented it using an analog computer. In many applications, however, it is more convenient to work with digitized data and a digital computer analysis code even though sampling problems might be introduced.

This method of digital filtering has several advantages over the conventional, direct integration method:

1. Computing time for the filter method is roughly one-third that of the direct method, which may be significant either if large volumes of data are to be processed or if the computation is to be done "on-line."

2. Linear filtering theory has been developed extensively, while the problems of numerical integration of products of sampled data functions are apparently not well understood.[3] It appears that extensions of this method could easily lead to a better understanding of the errors in numerical Fourier transform calculations due to noise in the function being transformed, and hence to better ways of estimating these errors.

Subsequent to the development (but not the reporting) of this technique, a method devised by Cooley and Tukey has been reported to be

[1] Convolution is alternatively referred to as the superposition theorem, Green's theorem, and Duhamel's theorem.

[2] P. W. Broome and G. C. Cooper, "Fourier Spectrum Analysis by Analog Methods," Instr. Control Sys. **35**(5), 155-60 (May 1962).

[3] S. Lees and R. C. Dougherty, Refinement of the Pulse Testing Procedure-Computer Limitations, Dartmouth College Research Report (Oct.'64)

orders of magnitude faster than direct integration.[4] Consequently, for applications requiring significant reductions in computing time, the Cooley-Tukey algorithm would be the likely choice.

2. DESCRIPTION OF THE METHOD

The Fourier transform of a time-varying function f(t) is a frequency-domain function $F(\omega)$ given by

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\, \epsilon^{-j\omega t}\, dt, \qquad (1)$$

where

ω = radian frequency,
$j = \sqrt{-1}$.

Equation (1) can be divided into its real and imaginary parts:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \cos \omega t \, dt - j \int_{-\infty}^{\infty} f(t) \sin \omega t \, dt. \qquad (2)$$

Thus to calculate the Fourier transform directly, the products of two time-varying functions must be integrated.

The convolution integral gives the output of a linear system, or filter, at any time t as a function of an input applied at previous times, $f(\tau)$:

$$g(t) = \int_{-\infty}^{t} f(\tau) \quad h(t-\tau)\, d\tau \qquad (3)$$

output of filter at time t	input to filter at time τ	response of filter at time t to an impulse applied at time τ

where the function h(t), the response of a linear filter to an impulse applied at $t = 0$, is known as the impulse response or weighting function of the filter. Thus the convolution integral is also an integration of the product of two time-varying functions similar to each term of Eq. (2). Furthermore, a few manipulations of Eqs. (2) and (3) will make them equivalent. Hence, to apply convolution

[4] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Fourier Series," J. Math. Comp., 19, 297,(April 1965)

to the problem of calculating Fourier transforms, the output time responses of filters with the appropriate weighting functions (i.e. cos ωt and sin ωt) are calculated for an input perturbation f. Then the outputs of the filters at a particular time will correspond to the real and imaginary parts of the Fourier transform of f(t).

To equate the convolution integral to the Fourier transform integral, we first make the upper limit of integration in Eq. (3) zero, which can be done if f(τ) can be made zero for all positive time:

$$g(0) = \int_{-\infty}^{0} f(\tau)\ h(-\tau)\ d\tau \qquad (4)$$

Equation (4) shows that the filter output at time t = 0 would be equivalent to the real (or imaginary) part of the Fourier transform of f(τ), Eq. (3), if h(-τ) were equal to cos ωt (or -sin ωt).

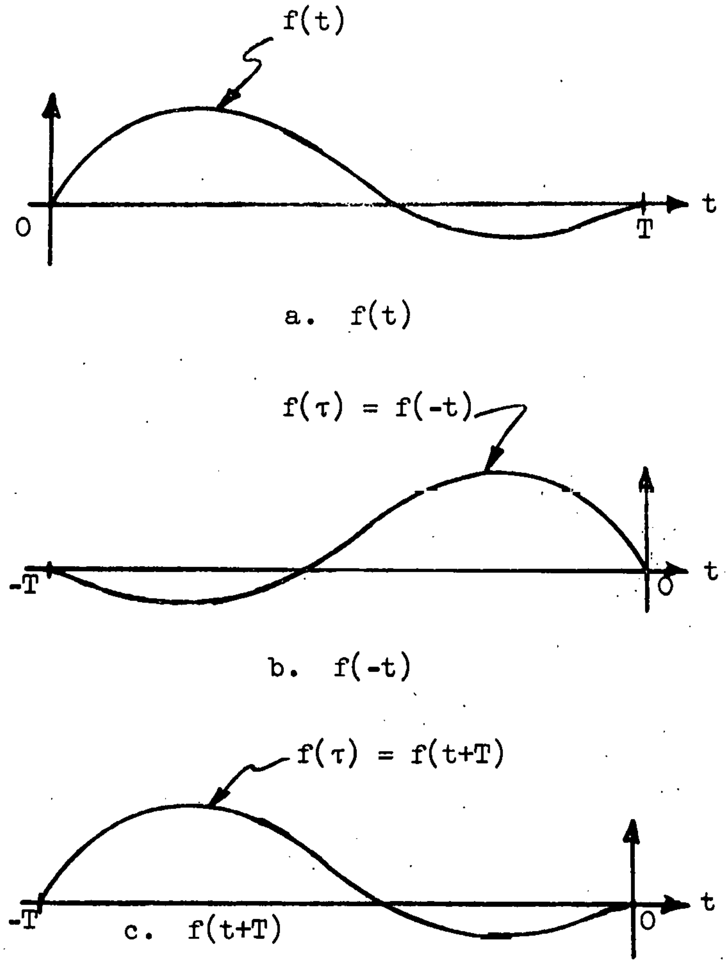
For the real part, since cos ωt is an even function, then cos ωt = cos(-ωt) and h(-τ) = h(τ); so Eq. (4) becomes

$$\mathrm{Re}\left\{F(\omega)\right\} = g_R(0) = \int_{-\infty}^{0} f(\tau)\ h(\tau)\ d\tau = \int_{-\infty}^{0} f(\tau)\ \cos(\tau)\ dt \ . \qquad (5)$$

On the other hand, since sin ωt is an odd function, then -sin ωt = sin(-ωt) and h(-τ) = -h(τ); so the solution for the imaginary part is

$$\mathrm{Im}\left\{F(\omega)\right\} = g_I(0) = -\int_{-\infty}^{0} f(\tau)\ h(\tau)\ d\tau = -\int_{-\infty}^{0} f(\tau)\ \sin(\tau)\ d\tau \ . \qquad (6)$$

Usually the response of the system to be analyzed, f(t), will be a function which has nonzero values for $0 \leq t \leq T$, and zero values for negative t and for times greater than T shown in Fig. 1a. There are two methods that we can use to make the function that we analyze zero for all positive time: we can either reverse the direction of f(t), i.e. f(-t) shown in Fig. 1b, or shift it by a time T, i.e. f(t+T) shown in Fig. 1c.



a. f(t)

b. f(-t)

c. f(t+T)

Fig. 1. Original Time Function with Time Reversal and Shift.

2.1 Method I, Reversing f(t) : f(-t)

If the actual function transformed is f(-t), rather than f(t), the relationship between this result and the desired transform of f(t) must be determined:

$$\mathcal{F}\left[f(-t)\right] = \int_{-\infty}^{\infty} f(-t)\epsilon^{-j\omega t}\,dt = \int_{-\infty}^{\infty} f(t)\epsilon^{+j\omega t}\,dt\ , \qquad (7)$$

which is just the complex conjugate of $F(\omega)$.

2.2 Method II, Shifting f(t) : f(t+T)

The equation is

$$\mathcal{F}\left[f(t+T)\right] = \int_{-\infty}^{\infty} f(t)\epsilon^{j\omega T}\,\epsilon^{-j\omega t}\,dt = \epsilon^{j\omega T}\int_{-\infty}^{\infty} f(t)\epsilon^{-j\omega t}\,dt \ . \qquad (8)$$

In this case, the desired result is obtained by correcting for a phase shift of ωT radians.

3. DIGITAL FILTER SIMULATION

A key factor in the method is the accurate digital simulation of the response of a cosine — sine filter (e.g. an undamped spring-mass system) to an arbitrary forcing function. This is accomplished by means of the matrix exponential technique.[5,6]

The differential equation for a system which oscillates with a frequency ω is

$$\frac{d^2x}{dt^2} + \omega^2 x = 0 \ . \qquad (9)$$

Alternatively, Eq. (9) can be put in the form of two first-order equations and made to include a forcing function $z_1(t)$:

$$\begin{aligned} \frac{dx_1}{dt} &= \omega x_2 + z_1(t), \\ \frac{dx_2}{dt} &= -\omega x_1 \ . \end{aligned} \qquad (10)$$

The solution of Eq. (10) when $x_1(0) = x_2(0) = 0$ and $z(t)$ is a unit impulse function applied at $t = 0^+$ is:

[5] H. M. Paynter and J. Suez, "Automatic Digital Setup and Scaling of Analog Computers," *Trans. Instr. Soc. Am.* **3**, 55-64 (Jan. 1964).

[6] S. J. Ball and R. K. Adams, "MATEXP" - A General Purpose Digital Computer Program for Solving Ordinary Differential Equations by the Matrix Exponential Method, ORNL-TM- (in preparation)

$$x_1(t) = \cos \omega t, \qquad (11)$$
$$x_2(t) = -\sin \omega t \ .$$

This corresponds precisely to the impulse responses of the sine and cosine filters which were required to satisfy Eq. (4).

To obtain the general solution to Eq. (10), it is convenient to convert Eq. (10) to matrix form:

$$\frac{dX}{dt} = AX + Z \ , \qquad (12)$$

where

$$X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

$$A = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix},$$

$$Z = \begin{pmatrix} z_1(t) \\ 0 \end{pmatrix}.$$

The exact "incremental" solution of Eq. (12), which updates X by a time τ, is

$$X(t + \tau) = \epsilon^{A\tau} X(t) + (\epsilon^{A\tau} - I)A^{-1} Z(t) \ , \qquad (13)$$

if we assume that Z is constant between t and $(t + \tau)$. Hence, by evaluating the $\epsilon^{A\tau}$ and $(\epsilon^{A\tau} - I)A^{-1}$ matrices once (for each ω), X can be successively updated by two matrix multiplications. The series form of $\epsilon^{A\tau}$ is

$$\epsilon^{A\tau} = I + \frac{A\tau}{1!} + \frac{(A\tau)^2}{2!} + \frac{(A\tau)^3}{3!} + \ldots\ldots \qquad (14)$$

So

$$\epsilon^{A\tau} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & \omega\tau \\ -\omega\tau & 0 \end{bmatrix} + \begin{bmatrix} \frac{-(\omega\tau)^2}{2!} & 0 \\ 0 & \frac{-(\omega\tau)^2}{2!} \end{bmatrix} + \begin{bmatrix} 0 & \frac{-(\omega\tau)^3}{3!} \\ \frac{(\omega\tau)^3}{3!} & 0 \end{bmatrix} \ldots\ldots$$

$$= \begin{bmatrix} 1 - \frac{(\omega\tau)^2}{2!} + \frac{(\omega\tau)^4}{4!} - \cdots\cdots & \omega\tau - \frac{(\omega\tau)^3}{3!} + \frac{(\omega\tau)^5}{5!} - \cdots\cdot \\ -\omega\tau + \frac{(\omega\tau)^3}{3!} - \frac{(\omega\tau)^5}{5!} + \cdots\cdots & 1 - \frac{(\omega\tau)^2}{2!} + \frac{(\omega\tau)^4}{4!} - \cdots\cdot \end{bmatrix}$$

Thus

$$\epsilon^{A\tau} = \begin{bmatrix} \cos \omega t & \sin \omega t \\ -\sin \omega t & \cos \omega t \end{bmatrix} \qquad (15)$$

Expanding $(\epsilon^{A\tau} - I)A^{-1}$ in the same manner, we eventually get

$$(\epsilon^{A\tau} - I)A^{-1} = \begin{bmatrix} \frac{\sin \omega\tau}{\omega} & \frac{1 - \cos \omega\tau}{\omega} \\ \frac{\cos(\omega\tau) - 1}{\omega} & \frac{\sin \omega\tau}{\omega} \end{bmatrix} \qquad (16)$$

Substituting Eqs. (15) and (16) into (13) and expanding, we get

$$x_1(t + \tau) = \cos \omega\tau \; x_1(t) + \sin \omega\tau \; x_2(t) + \frac{\sin \omega\tau}{\omega} \; z_1(t) \; ,$$

$$(17)$$

$$x_2(t + \tau) = -\sin \omega\tau \; x_1(t) + \cos \omega\tau \; x_2(t) + \frac{\cos \omega\tau - 1}{\omega} \; z_1(t) \; .$$

Since this solution is "exact" only when z_1 is a stair-step function, it might appear that greater accuracy could be achieved by using a trapezoidal (or higher order) approximation to $z_1(t)$. However, as Lees[3] has pointed out, this results in a correction term for the Fourier transform calculation of the form

$$F_{\text{true}} (\omega) \approx \left(F_{\text{stairstep}} (\omega)\right) \left(\text{correction term } (\omega)\right) \qquad (18)$$

However, the Fourier transforms are usually used to calculate the transfer function $G(\omega)$ of a system from the ratio of output-to-input Fourier transforms:

$$G(\omega) = \frac{F_{\text{output}}(\omega)}{F_{\text{input}}(\omega)} \; . \qquad (19)$$

With the assumption that the same sampling interval τ is used for both input and output, the correction terms for both the input and output transforms will be the same and will cancel out. Hence nothing is gained by using higher-order approximations to the forcing functions.

4. ESTIMATION OF ERRORS IN FOURIER TRANSFORM CALCULATIONS DUE TO NOISE

An elegant method for determining the mean-square response of linear, constant-coefficient filters to random noise inputs has been a part of the analog computer literature for many years.[7,8] This method (illustrated in Fig. 2) is based on the equivalence of the correlation function of white noise and a unit impulse function δ(t).

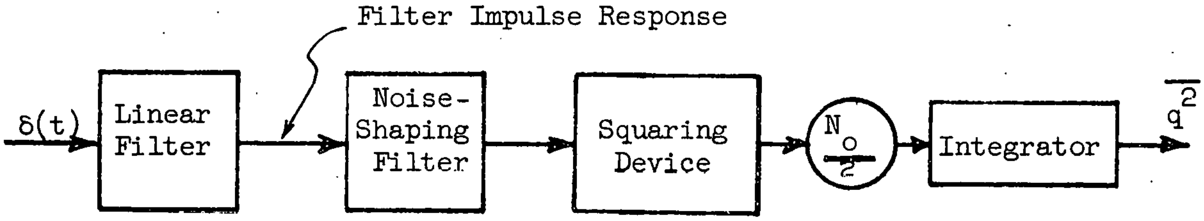


Fig. 2 Impulse-Response Method for Mean-Square Output Response.

A noise-shaping filter (Fig. 2) is used to account for differences between pure white noise and the actual input noise seen by the linear filter, which in this case would be noise in the signal to be Fourier transformed. The gain factor N_o is the power-spectral density (PSD) of the white noise input (before being shaped). Analysis shows that the steady-state output of the integrator equals the mean-square output of the filter due to the Gaussian noise input. The problem with applying this method to the sine or cosine filter, however, is that the integrator output would never reach a steady state, but would continue to increase with time, indicating that the error components of the real and imaginary part estimations would increase with

[7] A. E. Rogers and T. W. Connolly, Analog Computation in Engineering Design, McGraw-Hill, New York, 1960, Chapter 7.

[8] J. H. Laning and R. H. Battin, Random Processes in Automatic Control, McGraw-Hill, New York, 1956, pp 90-144.

integrating time.

A detailed analysis of the effects of noise is beyond the intent of this report, but it is hoped that these observations might serve as a starting point for further investigations.

5. DESCRIPTION OF THE FOURIER TRANSFORM CODE FOURCO

FOURCO has been set up as a general purpose Fortran IV code for calculating the Fourier transform of sampled input data (FI) and output data (FO), and for printing frequency response functions for up to 100 selected radian frequencies (W). The data is read in by the subroutine DATAIN, which can easily be altered to suit the format of the particular data. There is no nominal limit on the number of data points that can be processed, because the code will read in successive batches of 10,000 time samples as required, process each one as it goes until it reaches the end of a run, and then will compute the frequency response. Because of the batch data option, the second method of shifting f(t) is used in preference to the reversed input f(-t). The shifting method is also better suited to on-line calculations.

A variety of options is provided by FOURCO; the choice of which option to use sometime depends on the type of signal being analyzed (i.e. periodic or aperiodic) or on personal preferences. Specification of the choices by the user is made by setting the option flags M(1) through M(5), as follows:

1. Data Read-in and Processing Option, M(1)

M(1) > 0: Read in (and process) only M(1) input function data points (FI). Read in FO until a blank card is encountered.

M(1) = 0: Read in both FI and FO until a blank card is encountered.

M(1) < 0: Read in (and process) a single input function (FO) only.

2. Frequency Option, M(2)

   $M(2) > 0$: Read in all M(2) frequencies, W(1) up to W(100).

   $M(2) = 0$: Frequencies to be harmonics of the fundamental (WO), up to a limit WFIN, with a minimum spacing of DW; i.e., if $W(I+1)/W(I) < DW$, the next harmonic frequency would be substituted for W(I+1), etc., until $\frac{W(I+1)}{W(I)} \geq DW$.

   $M(2) < 0$: Frequencies to be equally spaced on a logarithmic plot between WO and WFIN, where $W(I) = DW * W(I-1)$; $DW > 1.0$.

3. Input Data Printout Option M(3)

   $M(3) > 0$: Print FI and FO data as read in.

   $M(3) = 0$: Print FI and FO data after subtracting bias.

   $M(3) < 0$: No printout.

4. Bias Selection Option, M(4)

   $M(4) > 0$: Subtract average values of FI and FO before processing.

   $M(4) = 0$: Subtract first-point values of FI and FO before processing.

5. Ensemble Average Option, M(5)

   $M(5) > 0$: Average and print M(5) data set transforms.

   $M(5) \leq 0$: Omit option.

A magnitude ratio correction factor COR is also read in so that different scale factors for FI and FO can be accounted for. The magnitude ratio printed is the ratio of output-to-input from raw data calculations multiplied by COR.

The computation times for FOURCO are roughly proportional to the number of input data points multiplied by the number of frequencies calculated. The basic (and most time consuming) calculation is updating the filter outputs; this requires six floating multiplies and four floating adds per step. Approximate running times on the IBM-7090 are given by

$$T \approx K * NT * NW,$$

where

T = running time, min,

K = 5.0×10^{-6},

NT = total number of data points and analyzed,

NW = number of frequencies.

K is somewhat larger if the input data printout option is selected. A typical analysis of 12,500 input and output data points at 47 frequencies took 4 min with input data printout.

6. APPENDIX

6.1 FOURCO Input Instructions

The input data arrangement for a typical case is shown in Fig. 3.

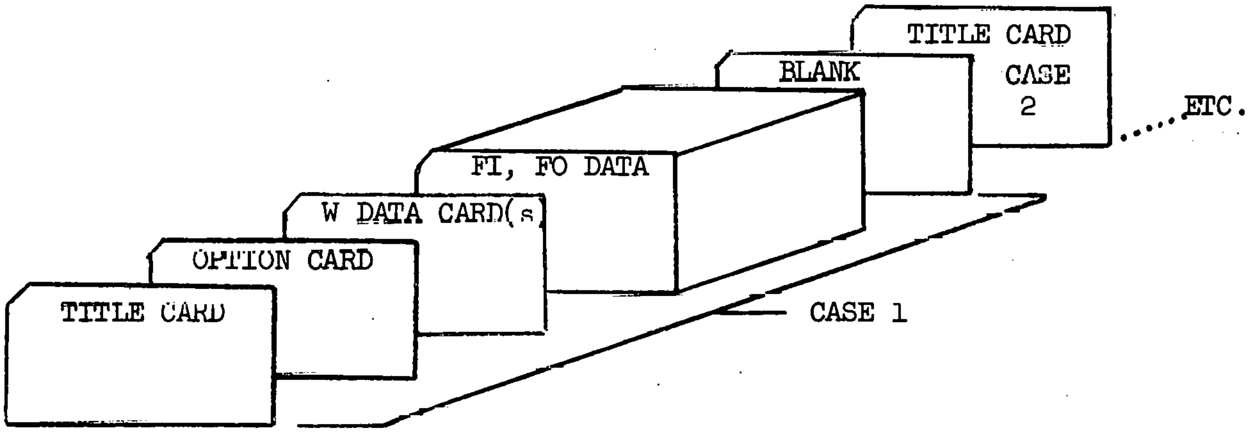


Fig. 3. FOURCO Input Data Layout

The title, option, and frequency cards are set up with the following format:

1. Title Card (12A6)

   Any 72 alphanumeric characters

2. Option Card (6I5, 2F10.0)

M(1)	M(2)	M(3)	M(4)	M(5)	(SPARE)	DT	COR
I5	I5	I5	I5	I5	I5	F10.0	F10.0

M(1) to M(5) = Option flags (see Sect. 5).

DT = Sampling interval

COR = Magnitude ratio correction factor

3. Frequency Card(s)

a. If M(2) $\leq$ 0: (3F10.0)

WO	DW	WFIN
F10.0	F10.0	F10.0

WO = minimum radian frequency

DW = frequency spacing factor (see Sect. 5)

WFIN = high frequency limit

b. If M(2) > 0: (8F10.0)

W(1)	W(2)	.... up to 8 per card
F10.0	F10.0	

Total number of W's read in = M(2)

W(I) = radian frequency.

6.2 FOURCO Fortran Listing

FOURCO is written in Fortran IV for the IBM-7090 computers at the Oak Ridge Computing Technology Center. Copies of the source or binary decks are available from the author on request.

```
$IBFTC MAIN    DECK
C     PROGRAM FOURCO
C     DIGITAL FILTER FOURIER TRANSFORM CODE
C
C     OPTION CODE
C     M(1)
C            POS # NO. DATA PTS. FOR INPUT FI  (PULSE)
C            0 # NO. DATA PTS FOR FI AND FO EQUAL
C            NEG # SINGLE INPUT (FO) ONLY
C     M(2)
C            POS # READ IN ALL M(2) W-S
C            0 # W0 + HARMONICS, DW#MIN. SPACING
C            NEG # W(I+1)#W(I)*DW
C     M(3)
C            POS # PRINT FI, FO DATA AS READ
C            0 # PRINT AFTER SUBTRACTING BIAS
C            NEG # NO PRINTOUT
C     M(4)
C            POS # SUBTRACT AVERAGES TO PROCESS FI, FO
C            0 # SUBTRACT FIRST POINTS
C     M(5)  #  NO. OF DATA SETS IN ENSEMBLE TO BE AVERAGED
C
C            ** PROGRAM SETS M(6) # -1 IF MORE THAN 10000 DATA PTS
C
      DIMENSION FI(10000),FO(10000),M(6),W(100)
      DIMENSION TITLE(12),CIA(100),COA(100)
      DIMENSION X1SI(100),X1SO(100),X2SI(100),X2SO(100)
      COMPLEX CI,CO,CIA,COA
      COMMON FI,FO,M,W,X1SI,X1SO,X2SI,X2SO
    1 READ(5,100)TITLE,M,DT,COR
  100 FORMAT(12A6/6I5,2F10.0)
      WRITE(6,101)TITLE,M,DT,COR
  1010FORMAT(1H1,12A6/6H0M 1-6,6I6,6H  DT #,F10.4,
     113X,14HCORR. FACTOR #,E20.8)
      ISET#0
      IF(M(2))15,15,16
   15 READ(5,103)W0,DW,WFIN
  103 FORMAT(8F10.0)
      W(1)#W0
      NW#1
      IF(M(2))17,18,16
   17 DO 19 I#2,100
      W(I)#W(I-1)*DW
      IF(WFIN-W(I))20,19,19
   19 NW#NW+1
      GO TO 20
   18 DWS#2.0
      DO 21 I#2,100
   22 W(I)#W(1)*DWS
      DWS#DWS+1.0
      WIM#W(I)/W(I-1)
      IF(WIM-DW)22,23,23
   23 IF(WFIN-W(I))20,21,21
   21 NW#NW+1
      GO TO 20
  16  NW#M(2)
      READ(5,103)(W(I),I#1,NW)
  20  DO 51 J#1,NW
      CIA(J)#CMPLX(0.0,0.0)
  51  COA(J)#CMPLX(0.0,0.0)
```

```
      TTOA#0.0
   52 NTO#0
      NGROUP#1
      DO 40 J#1,NW
      X1SI(J)#0.0
      X1SO(J)#0.0
      X2SI(J)#0.0
   40 X2SO(J)#0.0
    8 CALL DATAIN(NT)
      NTOP#NTO
      NTO#NTO+NT
      NTT#NGROUP*10000
      IF(NTO-NTT)7,6,6
    7 IRET#0
      GO TO 24
    6 NGROUP#NGROUP+1
      M(6)#-1
      IRET#1
   24 NTI#0
      IF(M(1).EQ.0)NTI#NT
      IF(M(1).GT.0.AND.NGROUP.EQ.1)NTI#M(1)
      CALL FOURT(NTI,NT,DT,NW)
      IF(IRET)2,2,8
    2 WRITE(6,102)
  102 FORMAT(1H0,16X,5HFREQ.,13X,15HMAGNITUDE RATIO,
     117X,12HPHASE (DEG.))
      TTO#FLOAT(NTO)*DT
      PI#0.0
      TTI#TTO
      IF(M(1).GT.0) TTI#FLOAT(M(1))*DT
      DO 41 J#1,NW
      IF(M(1).LT.0) GO TO 42
      SJWTI# TTI*W(J)
      CI#CMPLX(X1SI(J),X2SI(J))*CEXP(CMPLX(0.0,SJWTI))
      CALL AMPH(CI,AI,PI)
   42 SJWTO# TTO*W(J)
      CO#CMPLX(X1SO(J),X2SO(J))*CEXP(CMPLX(0.0,SJWTO))*COR
      CALL AMPH(CO,AO,PO)
      IF(M(1).LT.0) AI#TTO
      AO#AO/AI
      PO#PI-PO
      IF(M(5).LE.1) GO TO 41
      CIA(J)#CIA(J)+CI*TTO
      COA(J)#COA(J)+CO*TTO
      TTOA#TTOA+TTO
   41 WRITE(6,104)J,W(J),AO,PO
  104 FORMAT(1H ,I3,6X,3E20.8)
      IF(M(5).LE.1) GO TO 1
      ISET#ISET+1
      IF(M(5).GT.ISET) GO TO 52
      WRITE(6,105) M(5)
 105  FORMAT(22H1ENSEMBLE AVERAGES FOR,I4,14H SETS OF DATA.)
      WRITE(6,102)
      DO 50 J#1,NW
      IF(M(1).LT.0) CIA(J)#CMPLX(TTOA,0.0)
      CO#CONJG(COA(J)/CIA(J))
      CALL AMPH(CO,AO,PO)
   50 WRITF(6,104)J,W(J),AO,PO
      GO TO 1
      END
```

```
$IBFTC DIN     DECK
      SUBROUTINE DATAIN(NT)
C     DATA INPUT FOR FOURCO
      DIMENSION FI(10000),FO(10000),M(6),W(100),D(9)
      DIMENSION X1SI(100),X1SO(100),X2SI(100),X2SO(100)
      COMMON FI,FO,M,W,X1SI,X1SO,X2SI,X2SO
      NT#0
      FIA#0.0
      FOA#0.0
      IF(M(1).LE.0.OR.M(6).LT.0) GO TO 50
      NTI#M(1)
      READ(5,105)(FO(I),FI(I),I#1,NTI)
C     FORMAT FOR MSRE PULSE RESPONSE 105,6
 105  FORMAT(26X,2F3.0)
      IDB#NTI+1
      DO 33 I#IDB,10000
      READ(5,106)TIME,FO(I)
 106  FORMAT(F4.0,22X,F3.0)
      IF(TIME)34,34,33
   33 NT#NT+1
   34 NT#NT+NTI
      GO TO 35
  50  IF(M(1).EQ.0) GO TO 31
C     FORMAT FOR ORR NOISE - 109
      NS#1
      NF#20
      DO 52 I#1,500
      READ(5,109) TIME,(FO(J),J#NS,NF)
 109  FORMAT(15X,F5.0,20F3.0)
      IF(TIME)51,53,51
  51  NS#NS+20
      NF#NF+20
  52  NT#NT+20
  53  NTI#0
      GO TO 42
   31 DO 36 I#1,10000,4
C     FORMAT FOR MSRE PRBI TESTS  -  107
      READ(5,107) D
 107  FORMAT(F8.2,4(F7.3,E11.4))
      IF(D(1).LE.0.0) GO TO 37
      DO 5 J#1,4
      K#I+J-1
      KP#2*J
      FI(K)#D(KP)
  5   FO(K)#D(KP+1)
  36  NT#NT+4
  37  NTI#NT
  35  DO 40 I#1,NTI
   40 FIA#FIA+FI(I)
      FIA#FIA/(FLOAT(NTI))
   42 DO 43 I#1,NT
   43 FOA#FOA+FO(I)
      FOA#FOA/FLOAT(NT)
      IF(M(3))44,44,45
  45  IF(NTI.GT.0) WRITE(6,100)(FI(I),I#1,NTI)
  100 FORMAT(3H0FI/(1H ,1P10E11.3))
      WRITE(6,108)(FO(I),I#1,NT)
  108 FORMAT(3H0FO/(1H ,1P10E11.3))
  44  IF(M(6).LT.0) GO TO 6
      IF(M(4))7,7,8
```

```
C     SUBTRACT AVERAGE
  8   FIS#FIA
      FOS#FOA
      GO TO 6
C     SUBTRACT 1-ST POINT
  7   FIS#FI(1)
      FOS#FO(1)
C     SUBTRACT SAME AS IN DATA GROUP 1
  6   IF(NTI.EQ.0) GO TO 13
      DO 9 I#1,NTI
C     NEG FI FOR MSRE PRBS
  9   FI(I)#FIS-FI(I)
  13  DO 12 I#1,NT
  12  FO(I)#FO(I)-FOS
      IF(NTI.GT.0) WRITE(6,101)FIA,NTI
  101 FORMAT(13H0AVERAGE FI #,E20.8,
     14H FOR,I6,5H PTS.)
      WRITE(6,102)FOA,NT
  102 FORMAT(13H0AVERAGE FO #,E20.8,4H FOR,I6,5H PTS.)
      IF(M(3))46,47,46
   47 WRITE(6,107)(FI(I),I#1,MPRINT)
      WRITE(6,108)(FO(I),I#1,NT)
   46 RETURN
      END
```

```
$IBFTC FOUR1   DECK
      SUBROUTINE FOURT(NTI,NT,DT,NW)
      DIMENSION FI(10000),FO(10000),M(6),W(100)
      DIMENSION X1SI(100),X1SO(100),X2SI(100),X2SO(100)
      COMMON FI,FO,M,W,X1SI,X1SO,X2SI,X2SO
      DO 8 J#1,NW
      WT#W(J)*DT
      CWT#COS(WT)
      SWT#SIN(WT)
      SWTW#SWT/W(J)
      CWTMW#(CWT-1.0)/W(J)
      IF(NTI.EQ.0)GO TO 10
C     INPUT F(T)  INTO SINE FILTER
      X1T#X1SI(J)
      X2T#X2SI(J)
      DO 9 I#1,NTI
      X1TT#CWT*X1T+SWT*X2T+SWTW*FI(I)
      X2T#CWT*X2T-SWT*X1T+CWTMW*FI(I)
    9 X1T#X1TT
      X1SI(J)#X1T
      X2SI(J)#X2T
   10 X1T#X1SO(J)
      X2T#X2SO(J)
      DO 11 I#1,NT
      X1TT#CWT*X1T+SWT*X2T+SWTW*FO(I)
      X2T#CWT*X2T-SWT*X1T+CWTMW*FO(I)
   11 X1T#X1TT
      X1SO(J)#X1T
    8 X2SO(J)#X2T
      RETURN
      END
```

```
$IBFTC AMPH1   DECK
      SUBROUTINE AMPH(C,A,P)
C     CONVERTS COMPLEX NUMBER C TO MR AND PHASE (DEG)
      COMPLEX C
      A#CABS(C)
      IF(A)1,2,1
  2   P#0.0
      RETURN
    1 P#57.296*ATAN2(AIMAG(C),REAL(C))
      RETURN
      END
```

INTERNAL DISTRIBUTION

1. R. K. Adams
2-11. S. J. Ball
12. C. J. Borkowski
13. J. B. Bullock
14. O. W. Burke
15. F. H. Clark
16. R. A. Dandl
17. H. P. Danforth
18. S. J. Ditto
19. E. P. Epler
20. D. N. Fry
21. E. W. Hagen
22. C. S. Harrill
23. W. H. Jordan
24. R. C. Kryter
25. C. G. Lawson
26. J. B. Mankin, Jr. (K-25)
27. C. D. Martin
28. R. V. Miskell (Y-12)
29. C. H. Nowlin
30. H. G. O'Brien
31. C. L. Partain
32. R. W. Peelle
33. B. E. Prince
34. D. P. Roux
35. G. S. Sadowski
36. B. Squires
37. R. S. Stone
38. J. R. Trinko
39. R. E. Whitt
40. J. V. Wilson
41-42. Central Research Library
43. Document Reference Section
44-48. Laboratory Records Dept.
49. Laboratory Records, ORNL R.C.
50. ORNL Patent Office
51-65. Division of Technical Information Extension
66. Research and Development Division, ORO

EXTERNAL DISTRIBUTION

67. S. J. Gage, Univ. of Texas
68. R. P. Gardner, N. C. State Univ.
69. T. W. Kerlin, Univ. of Tenn.
70. J. W. Prados, Univ. of Tenn.
71. J. C. Robinson, Univ. of Tenn.
72. R. F. Saxe, N. C. State Univ.
73. S. E. Stephenson, Univ. of Ark.
74. Otis Updike, Univ. of Va.
75. W. C. Wright, Univ. of Tenn.