Engineering Sciences

MAR 25 '76

Branch Library

COMPUTATION OF POWER SPECTRAL DENSITIES AND CORRELATIONS USING DIGITAL FFT TECHNIQUES

By

R. E. Akins*

J. A. Peterka**

Sponsored by National Science Foundation
Grant ENG72-04261-A01

Fluid Mechanics and Wind Engineering Program
Department of Civil Engineering
Colorado State University
Fort Collins, Colorado

December 1975

CER75-76REA-JAP13

*Research Assistant
**Assistant Professor

U18401 0074302

ACKNOWLEDGMENTS

This work was supported by NSF Grant ENG72-04261-A01. Dr. A. C. Hansen helped with some of the programming details and was also involved in many discussions concerning problems and applications.

TABLE OF CONTENTS

LIST OF TABLES

1. INTRODUCTION

Spectral measurements are frequently required in fluid mechanics applications. Traditionally they have been made using analog techniques. With the development of the Fast Fourier Transform algorithms in the mid 1960's, digital techniques have evolved which enable power spectral densities and correlation functions to be calculated with costs much less than were previously possible. This report is intended to describe the Fast Fourier Transform algorithms available at Colorado State University, outline some of the difficulties encountered in using these algorithms, and provide a brief description of actual computer programs being used for spectral analysis on the CDC 6400 computer.

2. EXISTING FFT ROUTINES AT CSU

There are presently a number of computer programs used at CSU which use available FFT routines. Two FFT programs are being used extensively by the Fluid Mechanics and Wind Engineering group. These are FOR2D and FOURT, both a part of the IBM Contributed Program Library. FOURT (IBM Contributed Program No. 360D-13.4001) is presently on the system Fortran library (FTNLIB). FOR2D (IBM Contributed Program No. 360D-13.4006) is usually stored on a permanent file. For CSU users, a deck or access to this permanent file may be obtained by contacting Robert Akins or Dr. J. Peterka. The major difference between these two programs is that FOURT is written to use data located in the core of the computer and FOR2D is written to use data located on an external storage device. More detailed comments on these two specific subroutines appear in later sections.

3. SINGLE CHANNEL FORWARD/INVERSE TRANSFORM

Two separate uses of the FFT will be described; (1) calculation of a power spectral density from a time series and (2) transformation of a power spectral density to obtain an autocorrelation function. An explanation of the details of these types of calculations can be found in Bendat and Piersol (1). It will be assumed in the following discussions that the reader is familiar with this reference or an equivalent text.

The single-channel forward/inverse transform is perhaps the most straightforward application of the FFT, and is a good starting point for someone beginning to work with the FFT. A useful exercise is to select a known fourier transform pair and to perform the same transform using the FFT. An example utilizing this type approach will be discussed in order to illustrate usage of subroutine FOURT. Appendix A1 contains a program listing of subroutine FOURT. A short section of comments appears at the beginning of the listing and explains the calling parameters and some basic aspects of usage. Use of the program can be understood without a detailed understanding of the details of the program itself.

The example transform pair to be used consists of $R(t) = e^{-t}$, $t \geq 0$ and its inverse fourier transform $G(\omega) = 4/1+\omega^2$, $\omega \geq 0$. Such an R function is often used to represent the autocorrelation function of a fluctuating velocity signal and is not only an easy function to deal with, but also is of some physical significance. A sample program (Program CHECK) which was written to take a forward and inverse fourier transform is listed in Appendix B1. The following discussion will be based upon output from that program. References to the program will be by line number of the listing in the appendix.

The program was written to calculate a selected number of values of the function $R(t)$ at a time step specified by an input parameter. This array of values of $R(t)$, called D in the program, is reflected prior to performing the forward transform. This reflection is an important operation which is not adequately discussed in most texts. It is needed to satisfy continuous, even-function characteristics of the transform. In using a digital transform technique, one assumes that the data record is of infinite length. In order to create a record which resembles an infinite record, the function to be transformed is reflected about its endpoint, creating a symmetric, even, continuous function. A schematic of this reflection and the resulting periodic function is shown in Figure 1. Note that the function is one ΔT increment short of returning to the zero-time value of one at the time position $2N\ \Delta T$. This occurs because the reflection point is really at the $N\Delta T+1$ point rather than precisely on the $N\Delta T$ point as might be expected. The effect of not reflecting R prior to the transform is shown in Figure 2. In other words, the reflection should be done about a zero lag time, such that $R(\tau) = R(-\tau)$ so that the reflected correlation function is even. If there are $2N$ total points, N prior to reflection, then $R(N+I) = R(N+2-I)$ for $2 \leq I \leq N$. This scheme of reflection will result in the point $R(N+1)$ not being defined. Since a correlation is normally small at the maximum lag time, it is easiest to let $R(N+1)$ equal $R(N)$. The reflection in program check is performed in lines 35-40.

LIST OF FIGURES

After the data has been reflected and the D(2,I)'s set to zero, subroutine FOURT is called in line 43. It is necessary to create D as a two-dimensional array because the input to FOURT is complex. In many applications of these techniques only real signals are dealt with and the complex arithmetic capabilities of the program are not used. In these cases the fifth calling parameter of FOURT is set equal to zero indicating a real input only. It is important to understand the difference between NUMBER and NUMBE2. NUMBER as used in the program is the number of points in the unreflected R function. NUMBE2 is twice NUMBER or the length of the reflected R function.

After FOURT has been called, the output must be multiplied by a scaling factor in order to obtain the correct G. This factor for FOURT is 2 * DELTAT where DELTAT is the timestep of R. This multiplication is carried out in line 52 of Program Check. The G function returned from FOURT has data uniformly spaced in frequency with the data points at f = n/(NUMBER * DELTAT), n = 0, NUMBER-1.

Prior to performing the inverse transform, G is also reflected. In addition to reflecting G, the imaginary part of the D array is set to zero. Normally small values will appear in this position of the array during a transform and the inverse transform will be more accurate if the imaginary (D(2,I)) part of the D array is forced to zero. These operations are carried out in lines 57 to 65.

An inverse transform is performed to obtain the original R(t). Again the output of FOURT must be multiplied by a constant. For the inverse transform, this factor is 1/(4*NUMBER*DELTAT). The product of the factors for the forward and inverse transforms is 1/(2*NUMBER) or 1/NUMBER2. This agrees with the factor given in the write-up of FOURT in Appendix A1.

Several examples using the test functions R and G will now be discussed. A number of different experiments were conducted to determine the effect of the total time and the time increment on the accuracy of the results. In the tables and plots that follow, R* will denote the recovered R function after a forward and an inverse transform. Figure 3 is a comparison of G for various values of NUMBER, the total number of points, and DELTAT, the time step between points. These variables were selected to result in all of the R's being defined for the same total time. This plot is only for the higher frequencies; below $\omega = 3$, all of the cases tested agree. The N's on the plot are for the unreflected data. It can be seen that as NUMBER increases and DELTAT decreases the region over which the transform is accurate increases. For NUMBER equal to 2048, the results are very close to the actual function G for the entire range plotted. At higher frequencies, even the case for NUMBER equal to 2048 will deviate from the actual values.

A comparison of R* for the same cases is shown in Figure 4. R* is the initial function R after having been subjected to both a forward and inverse transform. In this case virtually all values of NUMBER yield an acceptable value for R*.

The central processor (Cp) times required for the various values of NUMBER2 are shown in Figure 5. It can be seen that there is an almost linear increase in Cp time with increasing NUMBER2. These times are for the actual transform only; any multiplication or other manipulations with the data would increase them. These test cases were run under the Scope 3.3.14 on the CSU CDC 6400 computer.

In summary, the FFT can be used rapidly and economically to perform a digital fourier transform of known data. Care must be taken to insure the data are reflected properly prior to the transform and that the appropriate factors are used after the transform. A user with no experience with FFT is strongly urged to experiment with this type of application prior to attempting to obtain a spectrum directly from digital data.

4. CALCULATION OF A POWER SPECTRAL DENSITY FROM A TIME SERIES

Another valuable application of the FFT is the calculation of a power spectral density function from a time series. A detailed explanation of this process is given in Chapter 9 of Bendat and Piersol (1) or in Chapter 6 of Enochson and Ontes (2). The basic equations used are straightforward and apply to any FFT routine. There is one significant difference between the procedures outlined in these references and the procedure recommended in this report.

This difference has to do with the addition of zeros to the initial time series to avoid having a distorted autocorrelation function as discussed on pages 312-314 of Bendat and Piersol (1). If one uses the reflection techniques described in Section 3 in obtaining an autocorrelation function from a power spectral density, the addition of zeros to the initial time series is unnecessary. This results in a significant advantage in that the same time series can be placed in a data array half the size required if the technique described in Bendat and Piersol (1) is used. In other words if a time series of data consisting of 2000 points was to be examined using standard procedures, an array of length 4000 would be required. If the reflection technique was used, the required array length would only be 2000 and there would be a savings in core of 50 percent. In most cases the size of the data array is limited by the available core of the computer, and therefore the ability to use a smaller array can be a significant advantage.

At this point nomenclature comparable to that in Bendat and Piersol will be introduced to make the following discussions easier to follow. Denote the time series by $x_n(t)$, $n = 1, N$, the fourier transform of this time series by $X(f_n, N)$ $n = 1, N$, and the spectral density function of the time series x_n by $\tilde{G}_x(f_n)$, $n = 1, N$. $f_n = (n-1)/T$ where $T = N\,\Delta t$. Δt is the time increment of the initial time series.

In terms of these variables, a technique for computation of power spectral densities is:

1. Truncate the data sequence or add zeros such that N is a power of 2. In most cases, the data should be taken to provide N data values without adding any zeros.

2. Taper this sequence using a cosine taper window. This process is discussed in Bendat and Piersol (1), pp. 322-324.

3. Compute $X(f_n, N)$ using a FFT routine.

4. Compute $\tilde{G}_x(f_n)$ using the equation $\tilde{G}_x(f_n) = \frac{2 \cdot \Delta t}{.875 \cdot N}|X_k|^2$

5. Smooth $\tilde{G}_x(f_n)$ using either frequency or segment averaging.

Frequency averaging averages together several values of $\tilde{G}_x$ from one transform about some value f_n and replaces all values averaged with one average value. Segment averaging is an ensemble average at each value of f_n of a number of separate transforms.

These steps are the basis for two programs which will be used as examples. It should be noted that a real data sequence x_n will have a complex fourier transform $X(f_n, N)$. In a sense the real part is the coefficient of the cosine term and the imaginary part is the coefficient of the sine term. Therefore in step (4) when the power spectral density estimate is computed, the sum of the square of these two terms is used. In all examples and figures, the power spectral density has been normalized with the variance of the time series. This normalized power spectral density will be called $F(f_n)$ or $F(n)$.

The smoothing in step (5) is one of the more subjective aspects of the procedure and the technique used will depend upon the type of signal being analyzed, the amount of computer time available, and the final use of the power spectral density. The smoothing and the choice of N and ΔT will determine the frequency range of the smoothed power spectral density. There is some choice available in the determination of these parameters, and this choice should be made prior to taking the data.

The largest value of N which can be used in core with the CDC 6400 is 8192 (2^{13}). This is the largest power of two which can be used for a data array and not exceed the available core. Frequencies will then run from 0 to $(\frac{N}{2} - 1)*\frac{1}{T}$. But $T = N\Delta T$, and therefore the frequencies will run from 0 to $(\frac{N}{2} - 1)*\frac{1}{N\Delta T}$. For large N this is approximately $1/2\Delta T$, the Nyquist frequency. The zero frequency value is generally not reliable because the record lengths are of a finite length. If the value were to be nearly exact, the total time of the input data record, T, should approach infinity. The increment between points is equal to $\frac{1}{T}$ where T is the length in time of the input data record. Recall T is equal to $N\Delta T$. Therefore the high frequency end of the power spectral density is determined by ΔT, the time interval of the data record, and the low frequency end is determined by the length of the data record.

Normally the type signal to be examined will dictate the sample rate, $1/\Delta T$. Once this is determined, and if the maximum range possible is desired, N is 8192, the low frequency end of the power spectral density is also set. The following table gives these limits for sample rates available on the Systems-Development A-D system currently in use.

TABLE 1 - LIMITS FOR POWER SPECTRAL DENSITY COMPUTATION - SYSTEMS - DEVELOPMENT A-D SYSTEM. RECORD LENGTH = 8192.

ΔT(SEC)	SAMPLE RATE (1/SEC)	LOWER LIMIT (HZ)	UPPER LIMIT (HZ)
.004	250	.031	125.0
.002	500	.061	250.0
.001	1000	.122	500.0
.0005	2000	.244	1000.0
.00025	4000	.488	2000.0

If a smaller range is desired, N may be reduced and there will be a savings in computer costs. If a larger range is desired, a program is available which allows larger N's to be used by employing an external storage device such as a disc. This program will be discussed later in this section.

Smoothing of the power spectral density is required. Two techniques are available: segment averaging and frequency averaging. These may be used independently or in a combined manner. In segment averaging, a number of power spectral densities are computed from separate records from the same signal. These estimates of the power spectral densities are treated as an ensemble, and an ensemble average computed. The number of segments used is determined by the quality of the smoothed power spectral density desired and the amount of computer time to be expended. Segment averaging will not alter the frequency range of the power spectral density, the upper limit will be $1/2\Delta T$ and the lower limit $1/T$.

Frequency averaging involves averaging adjacent points of the power spectral density estimate from one data record. For example every m points could be averaged and replaced by one point at the midpoint of the frequency range of the original m points. This type of averaging will have a negligible effect on the high frequency limit of the power spectral density, but will normally raise the lower limit substantially, depending, of course, on the choice of m and the original ΔT.

Factors which enter into the choice of frequency smoothing techniques are determined by the ultimate use of the power spectral density. If a well-smoothed plot is the desired output, a combination of frequency and segment averaging may be employed. If a correlation function is to be computed from the power spectral density function, then equal frequency spacing must be preserved. Also, the time spacing of the correlation obtained is determined by the frequency interval of the power spectral density and this relationship should be considered in any frequency smoothing.

4.1 Calculation of Power Spectral Densities Using Segment Averaging Techniques

In order to provide some examples of the use of both the FFT and the averaging techniques, output from a specific program will be presented. This program, SEGEMNT, is listed in Appendix B2, and references will again be made to line numbers in the program.

This program follows the suggested routine for computation of a power spectral density. Lines 107-133 read one block of data 8192 elements long off of the data tape, tape 1, and compute the mean and the rms of that data record. Lines 138-151 remove the mean from the data and divide by the rms to obtain a rms of 1.0. This section of the program also tapers the data. Lines 156-167 perform a forward fourier transform of the array D, and segment average into array SEGMEN. Lines 171-194 reflect the segment averaged spectra and perform an inverse transform to obtain a correlation function. The remainder of the program is concerned with output and plots of both the correlation and the power spectral density. Frequency averaging is performed in lines 246-260.

Some sample results from this program will now be used to illustrate the effect of segment and frequency averaging. Segment averaging can be evaluated using both qualitative and quantitative methods. The appearance of both the smoothed spectra and the autocorrelation can be compared for different numbers of segments. Figure 6 shows four different segment averaged spectra computed from the same data record. All four of these spectra were also smoothed using frequency averaging over the high frequency portion. The portion of the title which is of the form xx-8192 indicates how many segments of length 8192 were used in the calculation of the spectra. It can be easily seen that as the total number of records increases, the spectra become smoother. If the spectra are compared by laying one on another, there is no change in the best line that could be drawn through the data. In other words, if the 64-8192 case is compared with the 4-8192 case, the mean curves are identical. Additional qualitative comparisons can be made using a number of different criteria. The effective bandwidth, number of degrees of freedom and normalized standard error for the different cases can be computed using equations (9.140) to (9.149) of Bendat and Piersol (1). These values for the cases plotted in Figure 6 are shown in Table 2. As the number of segments averaged increases, the normalized standard error decreases. The effect of frequency averaging in reducing the normalized standard error can also be seen. Another means of comparison is available in terms of more physically relevant parameters. The area under the spectrum is compared in Table 3 for the four cases shown in Figure 6. There is very little difference in these integrated quantities as the number of segments increases. These values were all computed for the segment averaged spectra before frequency averaging. Close attention should be paid to the integral of F(n). A value which is not very close to 1.00 is an indication that, for some reason, an incorrect spectrum has been obtained.

Figure 7 shows the qualitative effect of frequency averaging. All three cases were averaged over the same number of segments, and the differences are a result of frequency averaging alone. The last line of the titles indicate the type of frequency averaging used. The different averaging schemes are: (1) no frequency averaging (2) HF AVG 10 - no frequency averaging from 0-5.98 HZ, 10 points averaged

TABLE 2 - EFFECT OF SEGMENT AVERAGING ON POWER SPECTRAL DENSITY ESTIMATES

CASE	NUMBER OF POINTS FREQUENCY AVERAGED	RANGE OF SMOOTHING (HZ)	EFFECTIVE BANDWIDTH (HZ)	NUMBER OF DEGREES OF FREEDOM	NORMALIZED STANDARD ERROR
4-8192	1	0-5.98	.122	8	.500
	10	5.93-500.0	1.220	80	.158
8-8192	1	0-5.98	.122	16	.353
	10	5.98-500.0	1.220	160	.112
16-8192	1	0-5.98	.122	32	.250
	10	5.98-500.0	1.220	320	.079
64-8192	1	0-1.09	.122	128	.125
	3	1.09-5.98	.366	384	.072
	10	5.98-500.0	1.220	1280	.039

TABLE 3 - COMPARISON OF INTEGRATED PROPERTIES OF POWER SPECTRAL DENSITY ESTIMATES

CASE	$\int_0^{500} F(n)dn$
4-8192	1.014
8-8192	1.004
16-8192	1.002
64-8192	1.007

from 5.98-500 HZ (3) HF AVG 10 LF AVG 3 - no frequency averaging from 0-1.098 HZ, 3 points averaged from 1.098-5.98 HZ and 10 points averaged from 5.98-500 HZ. Table 4 is comparable to Table 2 and shows the effective bandwidths, number of degrees of freedom and normalized standard error for the cases shown in Figure 7. These criteria are the only ways to evaluate frequency averaging. In most cases frequency averaging will be used to provide a smooth plot of the spectra, and the means of frequency smoothing selected will be dependent upon the type of data being considered, the frequency range of interest, and the ultimate use of the plot.

TABLE 4 - EFFECT OF FREQUENCY AVERAGING ON POWER SPECTRAL DENSITY ESTIMATES

CASE	NUMBER OF POINTS FREQUENCY AVERAGED	RANGE OF SMOOTHING (HZ)	EFFECTIVE BANDWIDTH (HZ)	NUMBER OF DEGREES OF FREEDOM	NORMALIZED STANDARD ERROR
4-8192	1	0-500.0	.122	8	.500
4-8192	1	0-5.98	.122	8	.500
	10	5.98-500.0	1.220	80	.158
4-8192	1	0-1.09	.122	8	.500
	3	1.09-5.98	.366	24	.289
	10	5.98-500.0	1.220	80	.158

Some additional guidelines which may be used in the selection of how many segments to average may be obtained from considerations of the autocorrelation function obtained from the segment averaged spectra. In order to compute an inverse fourier transform, the smoothed spectra must consist of equally spaced frequency increments. Generally the spectra to be used will only be segment averaged and not frequency averaged in order to preserve equal frequency spacing. In all of the cases which will be discussed, the segment averaged spectra was transformed using the techniques outlined in section 3. Figure 8 is a plot of the autocorrelation functions obtained from the spectra shown in Figure 6. In all cases the plots are quite similar up to a lag time of .2 seconds. For longer lag times there is more difference evident. As the number of segments used in the frequency averaging increases, the value of the autocorrelation stays closer to zero for lag times from .2 to 1.0 seconds. Table 5 shows the areas of the autocorrelation function up to the first zero crossing and also from 0 to 4.096 seconds. There is up to a 25 percent difference in the area to the first zero crossing between the different cases although the spectra of Figure 6 appear to be virtually identical. The areas computed over the full range of the autocorrelation are at least one order of magnitude less than the areas to the first zero crossing. A more detailed discussion of the reason for the difference in the areas is presented in the following paragraphs. These two problems represent a significant difficulty if one is interested in computing an integral scale.

TABLE 5 - EFFECT OF SEGMENT AVERAGING ON THE AREA UNDER THE AUTOCORRELATION FUNCTION

CASE	AREA TO FIRST ZERO CROSSING	AREA 0-4.096 SECONDS
4-8192	.0405	.00142
8-8192	.0363	.00157
16-8192	.0336	.00195
64-8192	.0280	.00187

In order to try to get a more accurate calculation of the autocorrelation function, a program was written to calculate the autocorrelation directly from the data record. This is a much more expensive method than the FFT technique and not as many cases were run. A comparison of the autocorrelations obtained using a direct calculation and using an inverse fourier transform of a spectra is shown in Figure 9. The plots with the title PROGRAM ACR were computed directly using a data record of the indicated length in 8 second segments. For a 32 second record, 4 separate autocorrelations were computed and averaged in a manner analogous to segment averaging of the spectra. The cost of calculation was such that in the direct case, the computation was only carried out to a lag time of .9 seconds. Therefore, the only direct comparison which can be made between the plots is the area up to the first zero crossing. For the 32 second record, the area to the first zero crossing is .0333 for the direct calculation and .0405 for the FFT calculation. For the 64 second record, the area is .0362 for the direct calculation and .0363 for the FFT calculation. It is interesting to note the comparison in cost to obtain an autocorrelation via the direct method with that for the FFT technique. For the lower two plots of Figure 10, both of which represent a data record of approximately 64 seconds of real time, the direct calculation for 100 values of time lag cost $28.00 while the FFT calculation costs $5.40 for 4096 values of time lag. This is a factor of 5 differences in cost for 40 times fewer correlation points. The FFT technique also provides a spectrum for the cost indicated.

In the course of the direct calculation of the autocorrelation function, an interesting effect of the length of the record used in the calculation was observed. The direct calculation was initially carried out using a record length of 2000 (2 seconds) and the maximum lag computed corresponded to 1000 data values (1 second). Two examples of this calculation are shown in Figure 10. In both cases where records of 2 seconds each were used, the autocorrelation is negative from a time lag of 0.3 seconds to a time lag of 1.0 seconds. This was not the case in any of the computations which used the FFT. In order to see what effect record length had on this negative region, the direct calculation program was modified to use a record length of 8000 (8 seconds). The results of these computations for the same total length of data are also shown in Figure 10. The negative region from .3 to 1.0 is no longer predominant, and these results agree well with the autocorrelations obtained from the FFT routines as shown in Figure 9.

An explanation for this difference can be made based on physical arguments. A time lag of .5 seconds corresponds to a frequency of 2HZ. In a 2 second record there would only be 4 cycles at this frequency and fewer cycles at any lower frequency (longer time lags). It seems that 4 cycles are not enough to adequately average in the calculation of an autocorrelation. By using a record length of 8 seconds, there will be 16 cycles of a 2HZ signal in one record, and the resolution at lag times of .5 seconds will be better. Based on a limited amount of experience with this particular record, it is felt that at least 8 cycles of a particular frequency should be present to obtain adequate resolution in an autocorrelation function at a lag time corresponding to the reciprocal of the frequency.

An additional effect of interest also arose in one case. A digital data tape was used which had more than one channel of data.

For a small portion of one record, the channels were reversed and the effect on the power spectral density is shown in Figure 11. The noise in the high frequency portion of the spectra is due to the channel switch. The second plot is of the same data but avoiding the record with the channel switch.

The cost of the various cases run with program SEGEMNT are listed in Table 6. These include the computation of a power spectral density, an autocorrelation and plots of both using the U200 plotting routines available at the Engineering Research Center, Colorado State University.

TABLE 6 - COST FOR TEST CASES - PROGRAM SEGEMNT
CENTRAL PROCESSOR COST = $290/hr

NUMBER OF SEGMENTS OF LENGTH 8192	TIME OF TOTAL AMOUNT OF DATA (SECONDS)	COST $
4	32.77	4.00
8	65.54	5.40
16	131.07	8.92
64	524.29	27.82

It is important to bear in mind that all of the examples in this section have been calculated using a record of pressure data obtained using a linear transducer. Non-linear transducers or signals of a different type which require different frequency range or which were taken at a different sample rate would alter the cost figures. As such, these examples should only be considered as guidelines in selecting a scheme for digital analysis.

4.2 Calculation of Power Spectral Densities Using an External Core FFT Algorithm

In some applications, it is desirable to have a greater frequency range of the power spectral density, or resolution of the autocorrelation at relatively large lag times. In order to obtain either of these results, a long record of data must be used for each segment. In order to stay within the present available core of the CDC 6400, (140000_8), the longest data record which is a power of two which may be used is 8192. A technique is available which allows longer data records to be considered by making use of disc storage and performing the FFT in pieces. The details of the algorithm are described by Brenner (3). A program titled FOR2D is available from the IBM Contributed Program Library (#360D-13.4006). This program was written by Norman Brenner and uses the algorithm of reference 3. The program allows record lengths limited only by the disc storage available on the computer system in use (presently between 2,000,000 and 3,000,000 for the CSU CDC 6400 system). This capability allows very long record lengths to be used if necessary. The cost of the calculations becomes large as longer records are used and in many cases becomes a limiting factor. A comparison of external core techniques and segment averaging techniques is discussed in section 4.3.

In order to use an external core type of program, the input data record is broken into a series of equal length records. It is necessary to be able to store 3 of these records in the core of the computer at any given time. This requirement will set the length of this array. The input data record is then stored on the disc and the FFT routine only calls a portion of the record at a time. It is important to understand that this is not a segment averaged technique, but that the resulting sequence of points is the same that would be obtained if the entire data record were transformed using a computer with a very large core.

A listing of a program written to utilize the external core technique, EXTCORE is in Appendix B3. A listing of subroutine FOR2D is in Appendix A2.

The steps necessary to calculate a power spectral density are basically the same as were listed in section 4.1. The only differences between program EXTCORE and SEGEMNT are in the input and averaging. These differences will be pointed out with reference to line numbers in Appendix B3.

In lines 145-170, the data is read from the data tape (tape 1) in units compatible with the length of the records to be stored in mass storage. These records are available to the program by calling subroutine DREAD. Lines 187-205 remove the mean from the data and taper the data. FOR2D is called in line 209. The remainder of the program involves frequency averaging, output, and plotting.

The frequency averaging is similar to that described in the previous section except that even the low frequency portion of the spectrum is frequency averaged. Since only one segment is run, some frequency averaging is necessary even in the low frequency portions in order to obtain acceptable levels of statistical reliability.

The power spectral densities obtained from four different cases using program EXTCORE are shown in Figure 12. The notation in the figures indicates how many portions were used to make up the entire record. The figure in the bottom right utilized a record made up of

512 parts, each consisting of 1024 data elements. The averaging in the three shorter cases was such that the bandwidths for all three were the same. The fourth case (512-1024) used a different scheme of frequency averaging. The details of the frequency averaging along with the number of degrees of freedom, and the normalized standard error are shown in Table 7. This table can be compared with Tables 2 and 4 of section 4.1. It can be easily seen that as the normalized standard error decreases, the power spectral density function becomes smoother.

TABLE 7 - EFFECT OF RECORD LENGTH ON POWER SPECTRAL DENSITY ESTIMATES, EXTERNAL CORE FFT

CASE	NUMBER OF POINTS FREQUENCY AVERAGED	RANGE OF SMOOTHING (HZ)	EFFECTIVE BANDWIDTH (HZ)	NUMBER OF DEGREES OF FREEDOM	NORMALIZED STANDARD ERROR
32-1024	8	0-31.25	.244	16	.353
(32.77 SEC)	16	31.25-62.50	.488	32	.250
	128	62.50-500.0	3.906	256	.088
64-1024	16	0-15.63	.244	32	.250
(65.54 SEC)	32	15.63-31.25	.488	64	.177
	256	31.25-500.0	3.906	512	.063
128-1024	32	0-7.81	.244	64	.177
(131.07 SEC)	64	7.81-15.63	.488	128	.125
	512	15.63-500.0	3.906	1024	.044
512-1024	16	0-1.95	.030	32	.250
(524.29 SEC)	64	1.95-21.48	.122	128	.125
	256	21.48-41.02	.488	512	.063
	512	41.02-500.0	.977	1024	.044

Table 8 shows the values of the areas under the spectra of Figure 12. This table is comparable to Table 3 of section 4.1. The first case (32-1024) shows more variation than any of the other cases in Table 3 or Table 8, but for many applications this error would be acceptable.

TABLE 8 - COMPARISON OF INTEGRATED PROPERTIES OF POWER SPECTRAL DENSITY ESTIMATES, EXTERNAL CORE FFT

CASE	$\int_0^\infty F(n)dn$
32-1024	.978
64-1024	1.016
128-124	1.009
512-1024	.997

Correlation functions were computed for three of the example cases and are shown in Figure 13 along with one case calculated with program SEGEMNT. A trend can be seen in these figures which is similar to that of Figure 8. As the length of record increases, the correlation at larger lag times is more nearly zero. The correlations computed using program EXTCORE all have very much larger record lengths than those computed using program SEGEMNT. It would be expected that the EXTCORE correlations would be valid for longer lag times. A listing of the areas for the three correlations computed is shown in Table 9. In all of the cases, the area to the first zero crossing is comparable to that for the entire autocorrelation (0-4.096 sec). This agreement is in contrast with the cases shown in Table 5 for the shorter records of program SEGEMNT. This is another example of the effect of record length on the calculation of autocorrelation functions. There is fair agreement between the areas out to the first zero crossing in both cases, and this may be an appropriate choice of area when only a limited record length is available. Care must be used in using the area to the first zero crossing, since not all correlations remain as close to zero as this demonstration case for regions beyond the first zero crossing.

TABLE 9 - EFFECT OF RECORD LENGTH ON THE AREA UNDER THE AUTOCORRELATION FUNCTION, EXTERNAL CORE FFT

CASE	RECORD LENGTH SECONDS	AREA TO FIRST ZERO CROSSING	AREA 0-4.096 SEC
32-1024	32.77	.0362	.0339
64-1024	65.54	.0378	.0376
128-1024	131.07	.0333	.0350

The costs for the EXTCORE examples are listed in Table 10. These include the cost of all calculations and plotting.

TABLE 10 - COST FOR TEST CASES PROGRAM EXTCORE. CENTRAL PROCESSOR COST = $290/hr

CASE	DATA LENGTH SECONDS	COST $
32-1024	32.77	12.10
64-1024	65.54	22.59
128-1024	131.07	46.37
512-1024	524.29	210.30

4.3 Comparison of Program SEGEMNT and Program EXTCORE

Many of the differences between and advantages of segment averaging and external core approaches are apparent after reading the previous section. These differences and advantages will be briefly summarized in order to point out the most significant.

The major advantages of the external core technique are that it allows a greater frequency range in the spectrum and provides an autocorrelation function which is valid at relatively long lag times. The advantage of being able to obtain more points at low frequency in the spectrum is offset somewhat by the need to perform some type of smoothing in order to obtain a statistically reliable value. For the external core case, the smoothing will be accomplished using frequency averaging which will reduce the number of data points available at the low frequency end of the spectrum.

The autocorrelation which may be obtained using the external core technique is of higher quality at higher lag times than the autocorrelation which may be obtained using segment averaging. This increase in quality is obtained at a corresponding increase in cost of computer time. This extra cost may be necessary if an accurate measure of integral scale is desired. The integral time scale and the low frequency end of the power spectral density are directly related, ($F(0) = 4 \int_0^\infty R(t)dt$), and if the low frequency end of the spectra has a standard error of .5, there can be up to 50 percent error in the integral scale.

The major advantage of segment averaging is cost. In all cases, comparable quality power spectral densities can be obtained (based on normalized standard error) for from 1/3 to 1/8 the cost using segment averaging instead of external core techniques.

The core requirements for each case are comparable based on the array sizes used in the example programs. Changing array sizes in either of the programs would have an effect on the core required, but a comparable change would have to be made to both programs and the core requirements would still be comparable.

A general guideline in selecting a technique would be to use segment averaging unless a special requirement exists which requires the external core technique.

5. TWO CHANNEL CALCULATIONS--CROSS-SPECTRAL DENSITIES AND CROSS-CORRELATIONS

Some applications require information concerning the relationship between two time series in either the frequency or time domains. Once the techniques described in the previous sections are understood, the computation of functions describing these relationships can be readily accomplished. Most computations of multichannel functions begin with a cross-spectral density function, a complex quantity. Once the cross-spectral density function is obtained, a number of additional quantities can be computed. A brief discussion of some of these functions can be found in Bendat and Piersol (1), pp. 25-34.

The equation for the cross-spectral density of two time series $x(t)$ and $y(t)$ is given by the equation $G_{xy}(f_n) = \frac{2}{T} X^*(f_n)\, Y\,(f_n)$. ($X^*$ is the complex conjugate of the transform of the $x(t)$ time series.) Thus, once the transforms of two simultaneous time series are available, the cross spectral density, and any other related quantities may be computed. As brief examples of both computation and averaging techniques, programs which compute a coherence function and a cross-correlation coefficient will be discussed.

The two most important aspects of these programs are the techniques of data storage and averaging. The data storage is common to both programs and will be explained with reference to PROGRAM CSPECT2 (Appendix B4). The single channel transforms have been computed and are stored on a master data tape (tape 2) as separate files, with each segment a separate record (logical record) of the file. The input portion of the program (lines 90-105) reads each file from tape 2 and stores them on tape 3 and tape 4 for $X(f_n)$ and $Y(f_n)$ respectively. All reads and writes are done using unformatted binary reads and writes. The use of this type statement instead of a formatted read or write results in savings of from 50 percent to 90 percent in the required computer central processor costs.

As shown in previous sections, some method of averaging will be required to obtain statistically reliable estimates. In the examples segment averaging is used as the primary method. It is necessary to segment average the cross-spectral density function and <u>not</u> the single channel transforms. Therefore, in lines 110 and 111 the single channel transform for each segment is read and the segment averaged cross-spectral density function is computed. In the calculation of the coherence function (lines 113-130) the cross-spectral density is also frequency averaged prior to the final calculation of the coherence function (lines 138-156).

A second example program which calculates a cross-correlation function (PROGRAM CSPECT3) is listed in Appendix B5. The input and smoothing sections of this program are the same as those in CSPECT2.

The cross-correlation is obtained from an inverse fourier transform of the cross-spectral density function. A segment averaged cross-spectral density is computed in lines 106-113 and reflected in lines 117-122. The cross-spectral density is reflected such that the real part is an even function and the imaginary part an odd function. The reflected cross-spectral density is transformed in line 126 to obtain a cross-correlation coefficient. The cross-correlation coefficient can be calculated directly from the time series, and a comparison of a direct computation and a FFT computation is shown in Figure 14. The two results are virtually identical.

This brief section shows just two of the many cross-channel computations possible. Costs of the different calculations will vary with the application and no definite guidelines can be stated. The two most important aspects of cross-channel calculations are (1) use of binary write and read statements (2) averaging the cross-spectrum and <u>not</u> the single channel transforms.

REFERENCES

1. Bendat, J. S., and Piersol, A. G., Random Data: Analysis and Measurement Procedures, Wyley-Interscience, New York, 1971.

2. Enochson, Soren D., and Ontes, Robert K., Programming and Analysis for Digital Time Series Data, The Shock and Vibration Information Center, United States Department of Defense, 1968.

3. Brenner, Norman M., "Fast Fourier Transform of Externally Stored Data," IEEE Transactions on Audio and Electroacoustics, Vol. AU17-No. 2, June 1969.

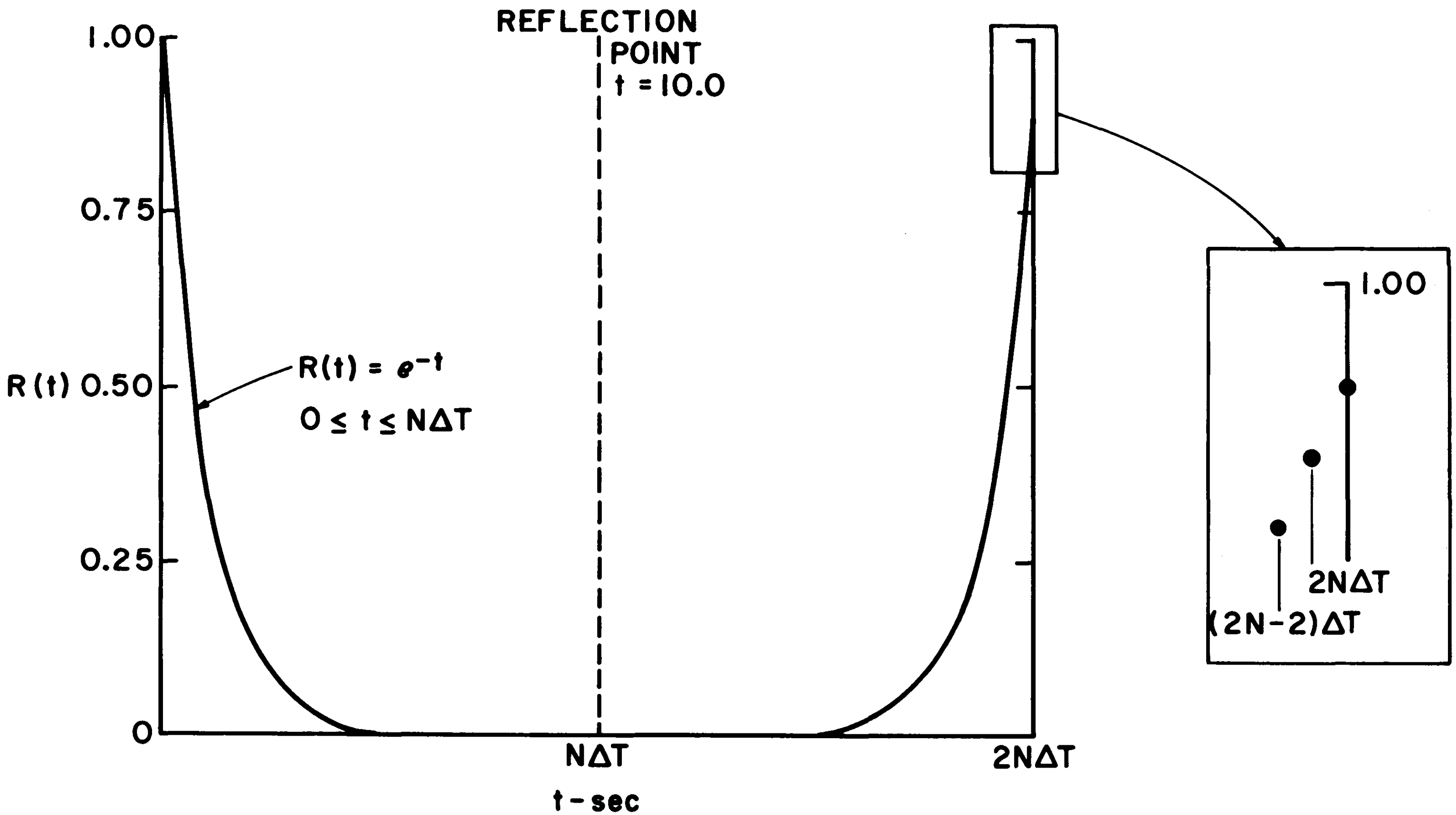


FIGURE 1. REFLECTION OF AUTOCORRELATION.

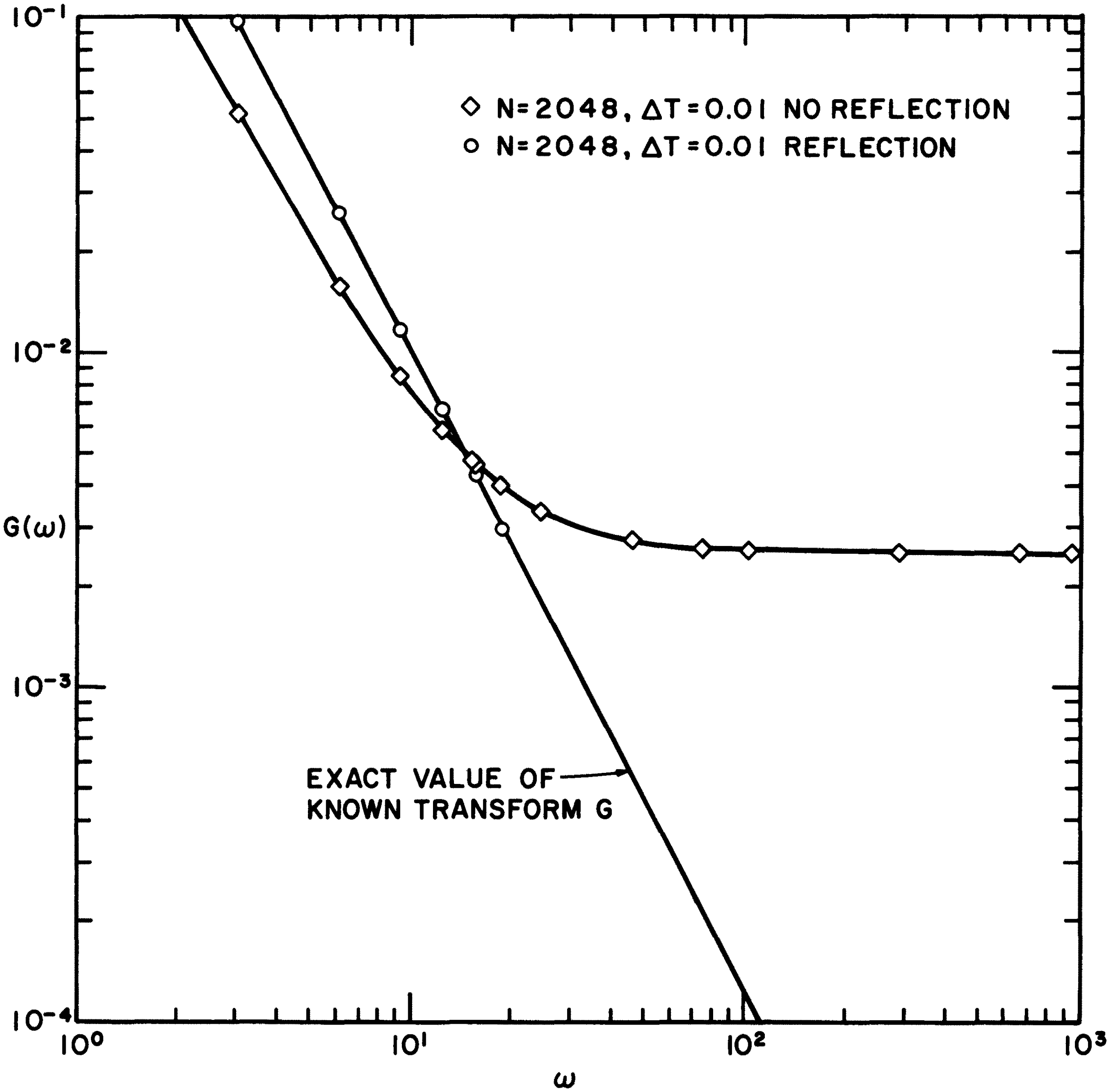


FIGURE 2. EFFECT OF TRANSFORMING R(t) PRIOR TO TRANSFORMING.

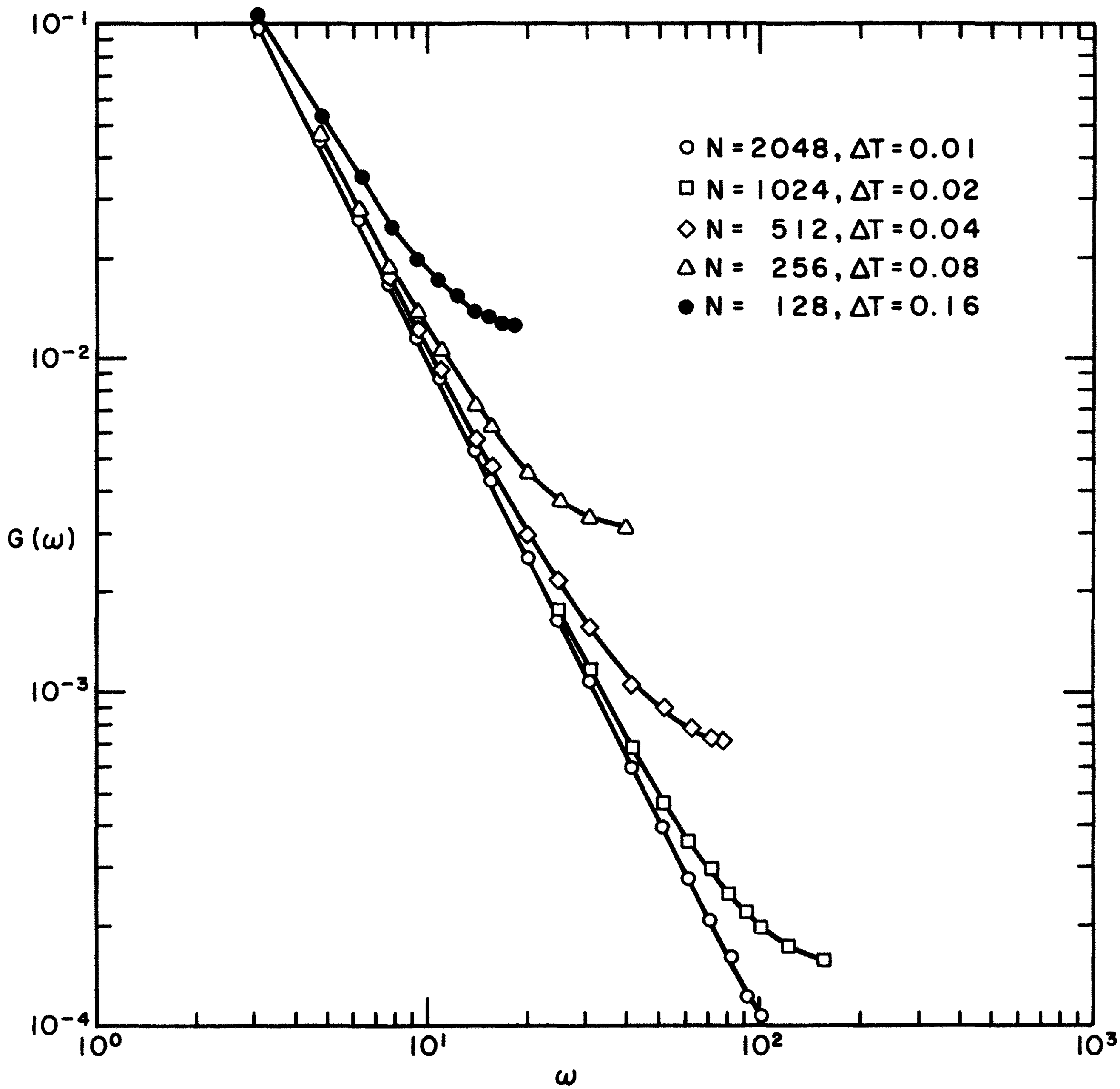


FIGURE 3. COMPARISON OF $G(\omega)$.

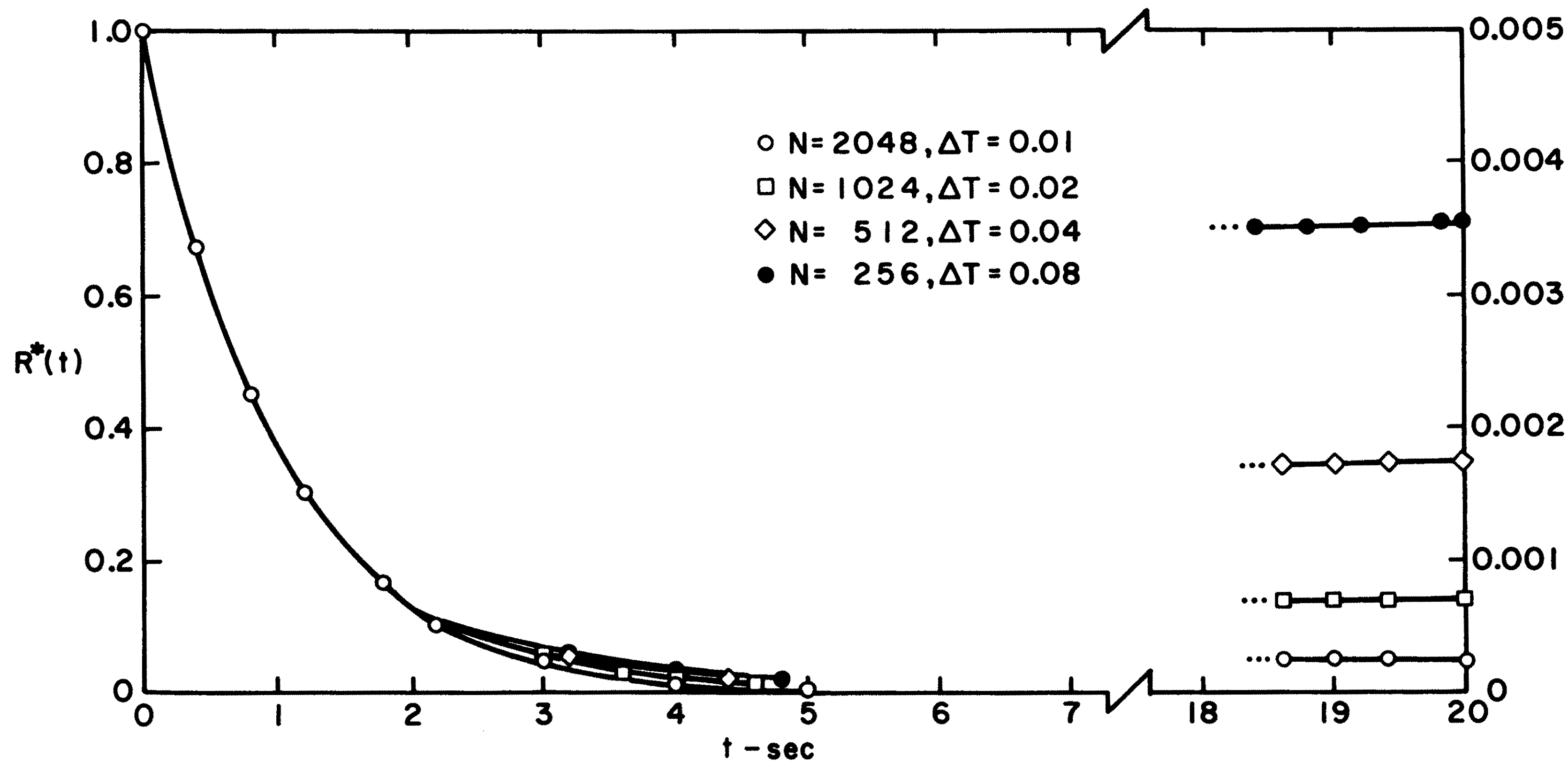


FIGURE 4. COMPARISON OF R*(t).

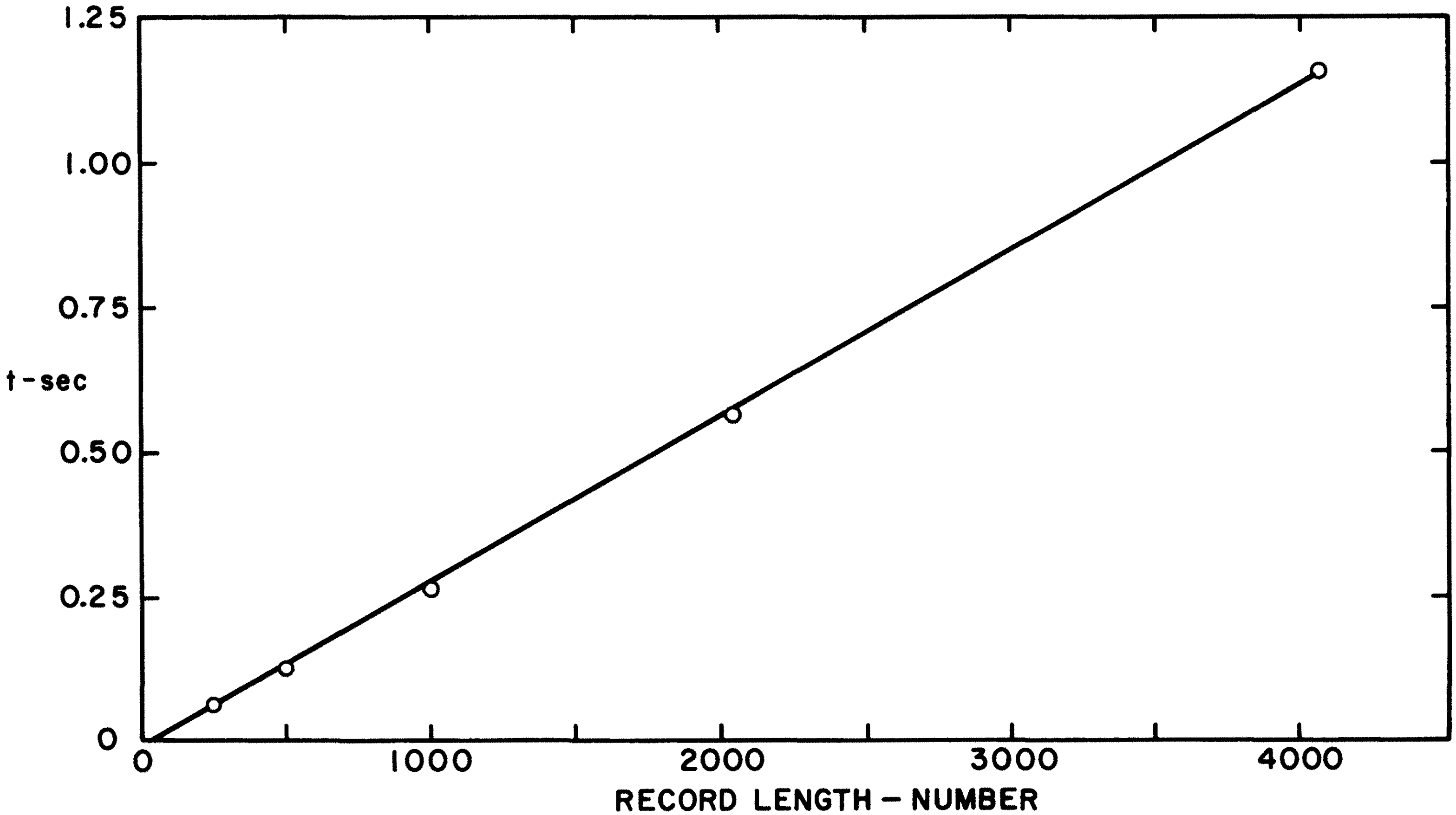


FIGURE 5. EXECUTION TIME--PROGRAM FOURT.

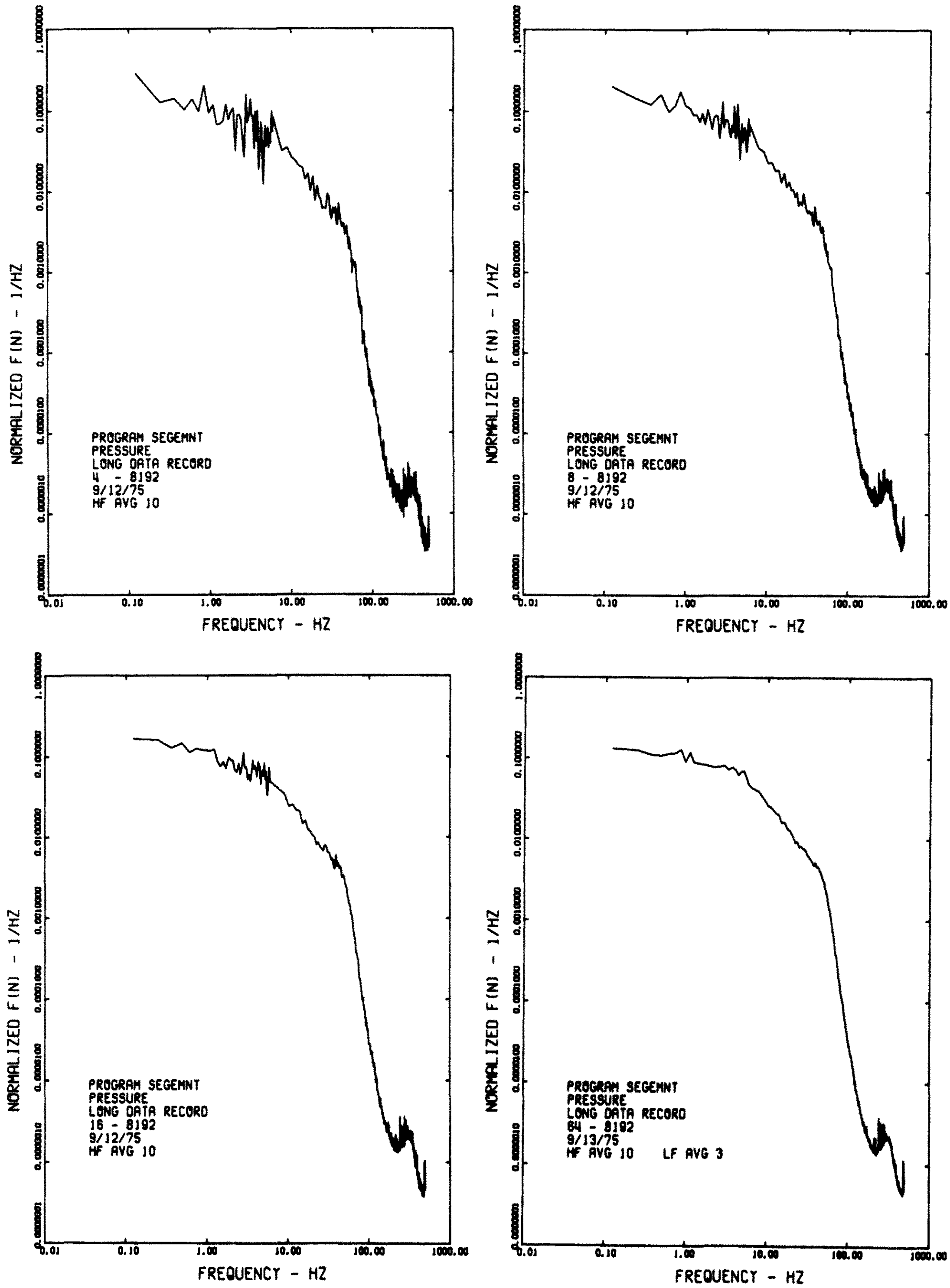


FIGURE 6. SEGMENT AVERAGED POWER SPECTRAL DENSITIES--
PROGRAM SEGEMNT.

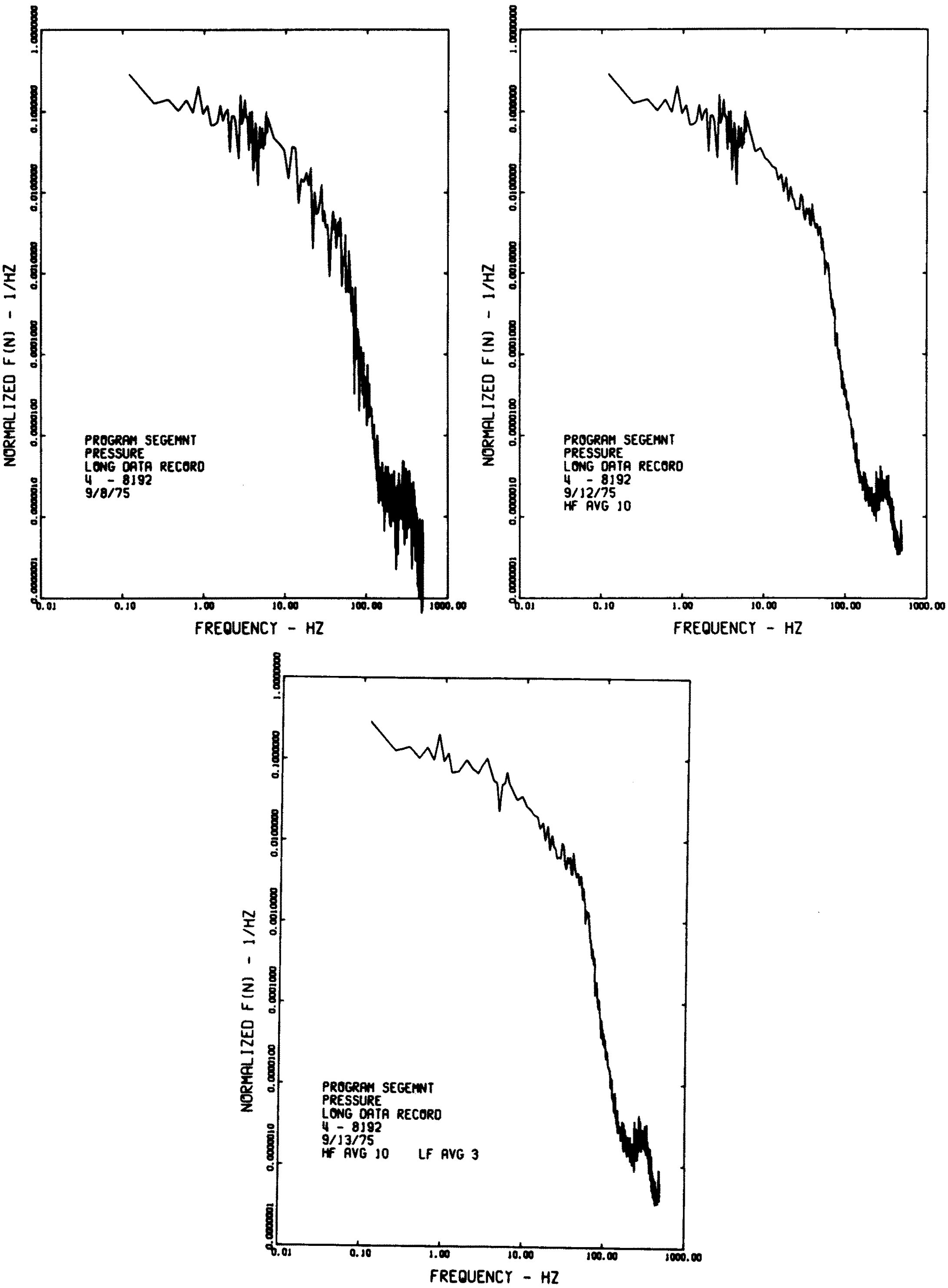


FIGURE 7. FREQUENCY AVERAGED POWER SPECTRAL DENSITIES--PROGRAM SEGEMNT.

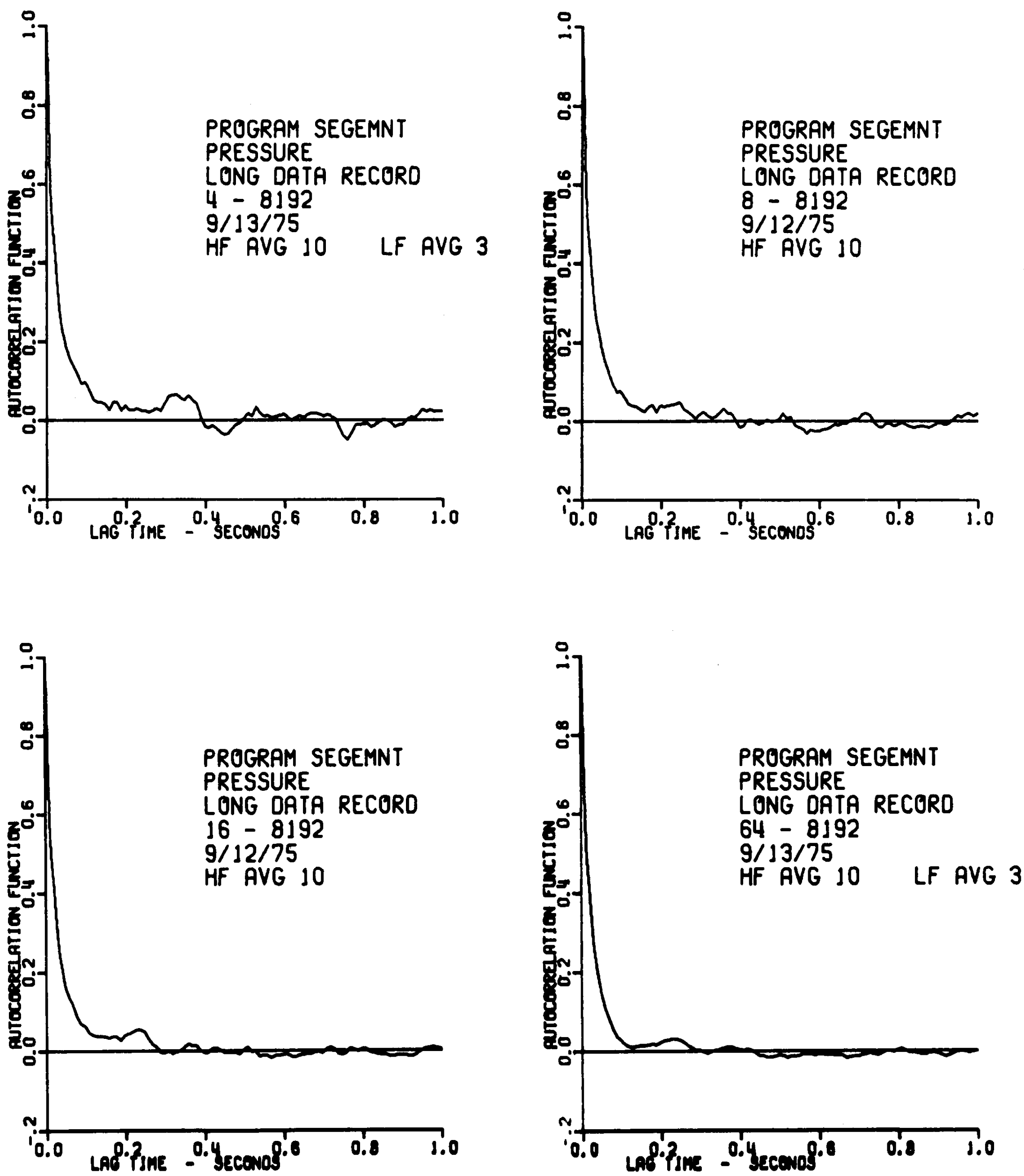


FIGURE 8. AUTOCORRELATIONS--PROGRAM SEGEMNT.

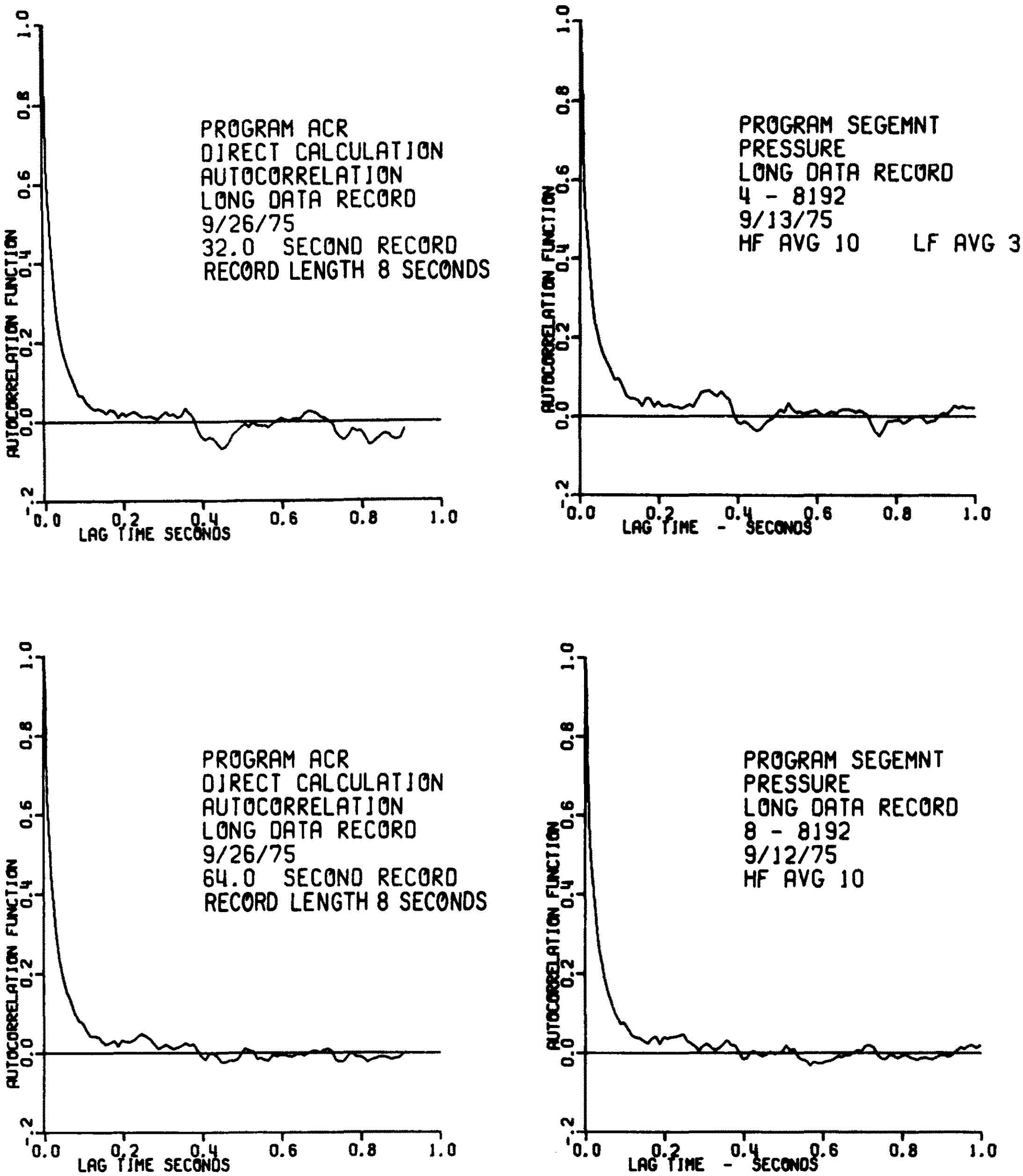


FIGURE 9. COMPARISON OF AUTOCORRELATIONS--PROGRAM SEGEMNT AND DIRECT CALCULATION.

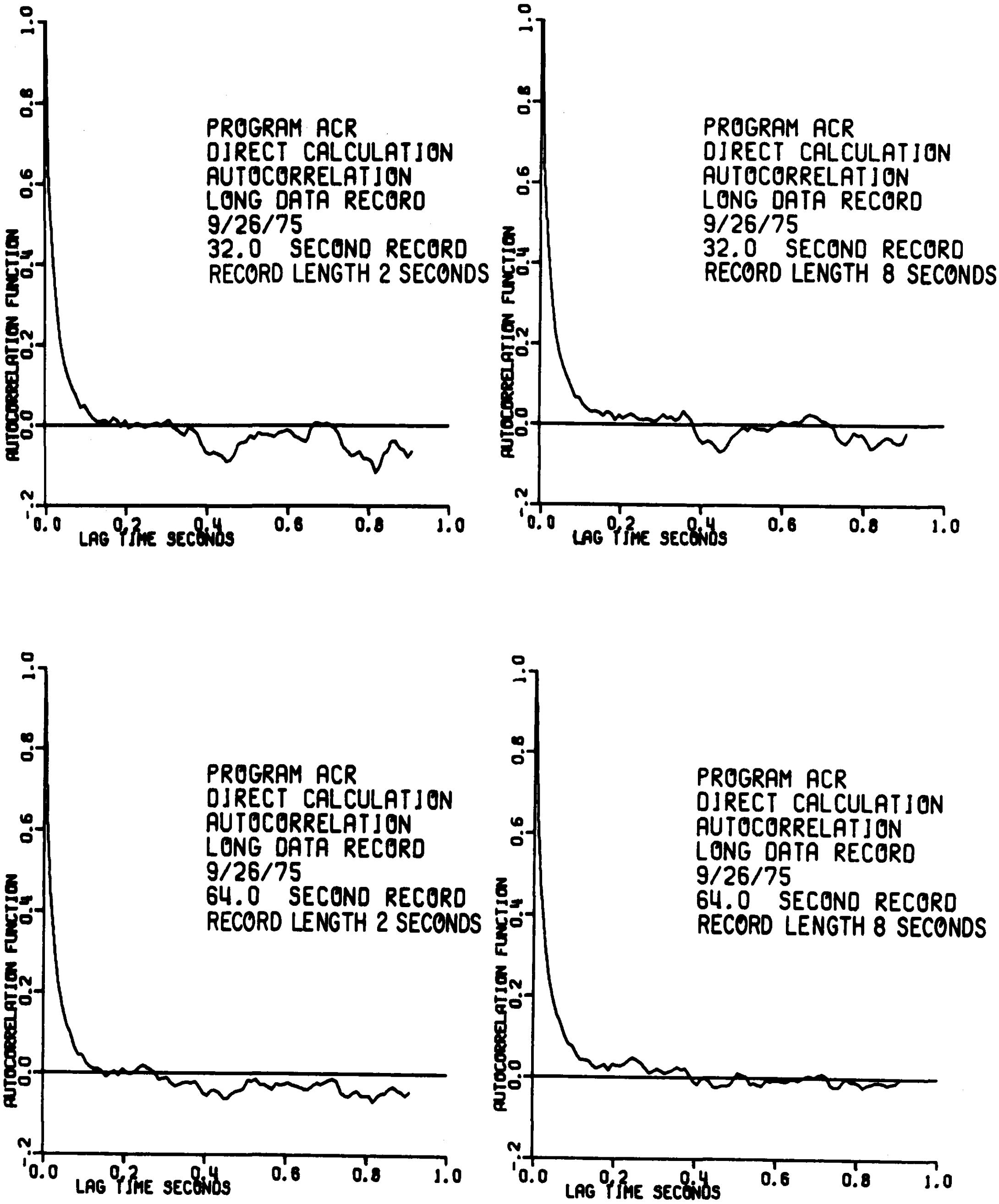
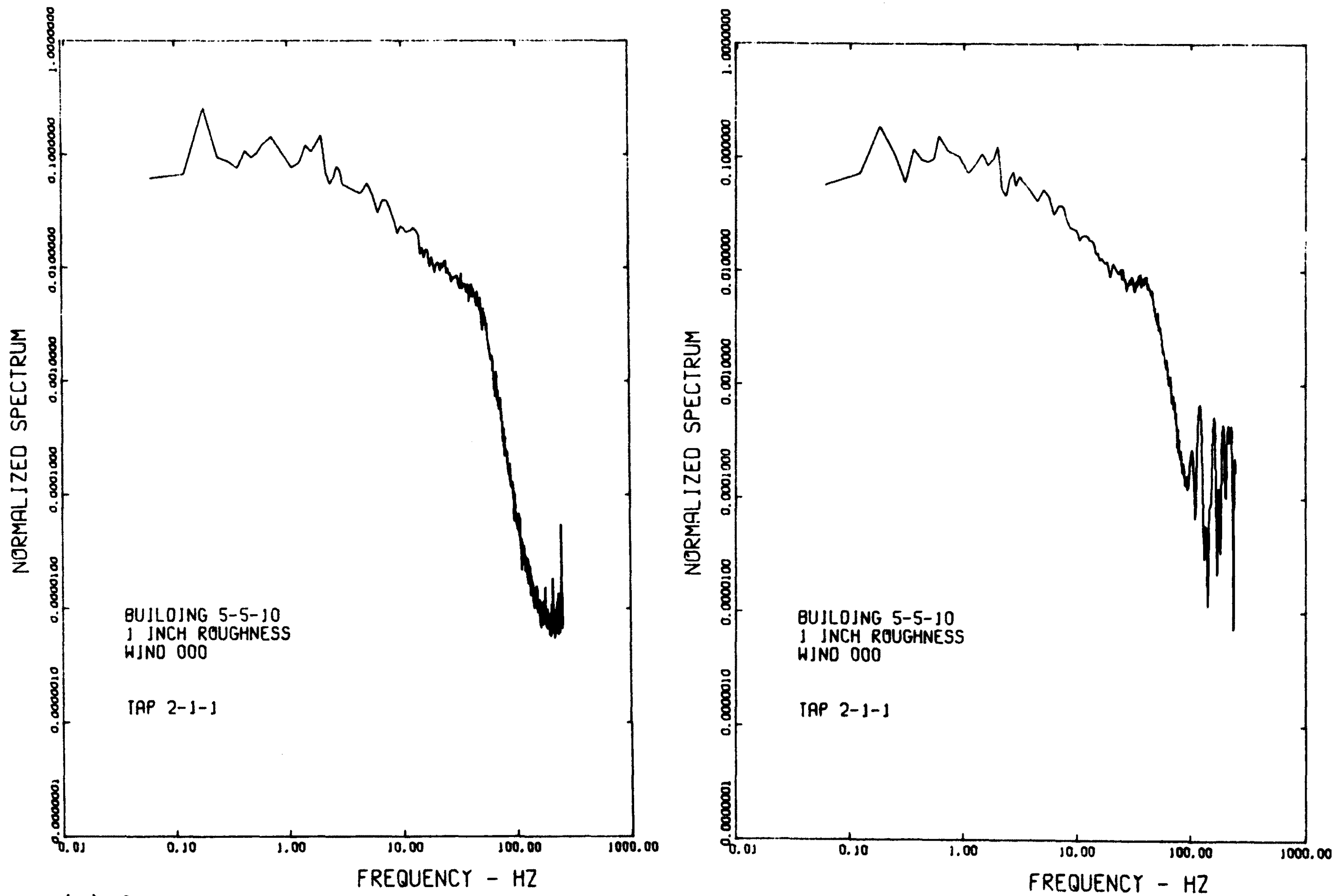


FIGURE 10. EFFECT OF RECORD LENGTH ON DIRECT CALCULATION OF AUTOCORRELATION.



(a) CHANNELS CORRECT

(b) CHANNELS REVERSED

FIGURE 11. EFFECT OF REVERSED CHANNELS.

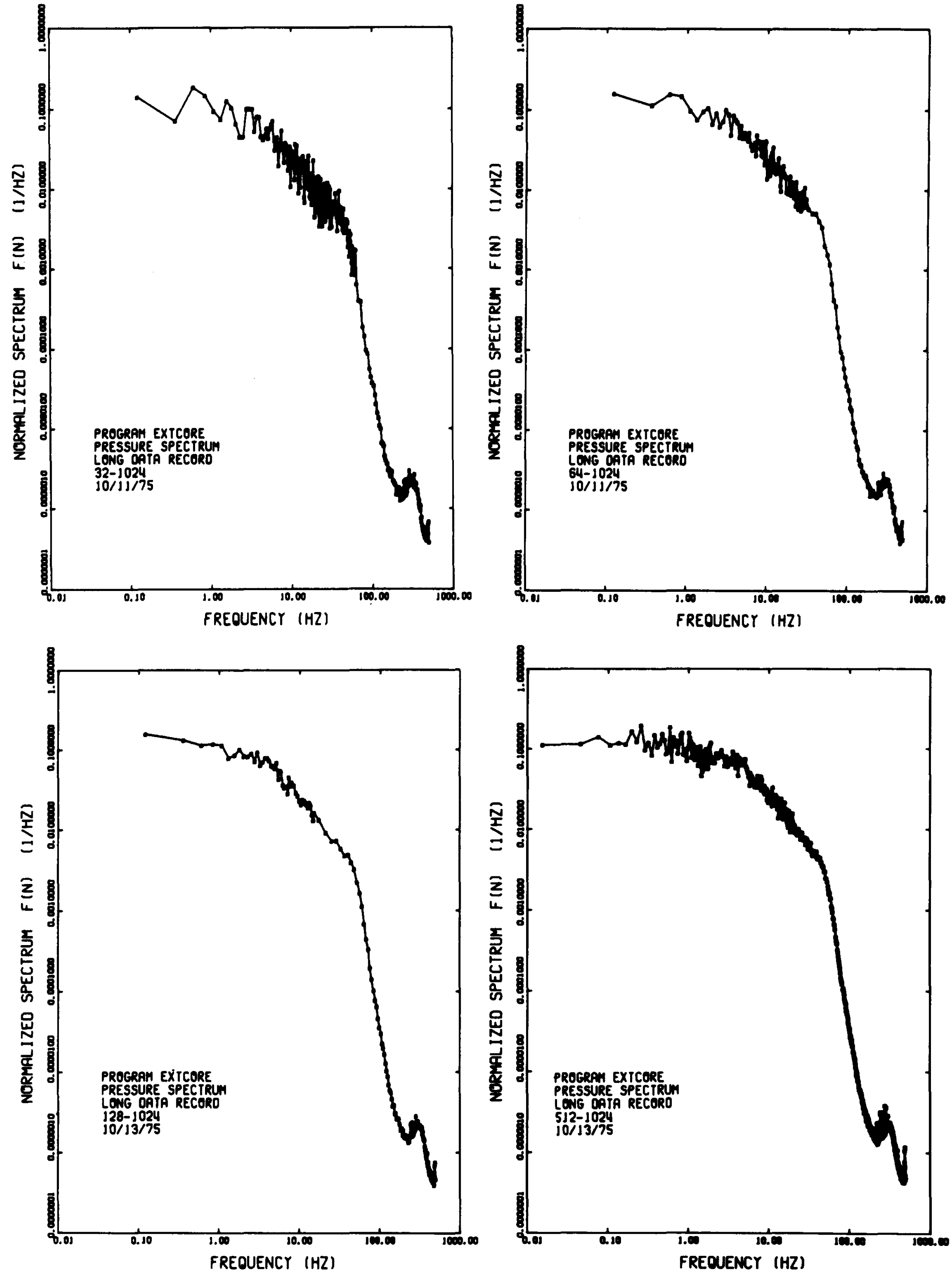


FIGURE 12. POWER SPECTRAL DENSITIES--PROGRAM EXTCORE.

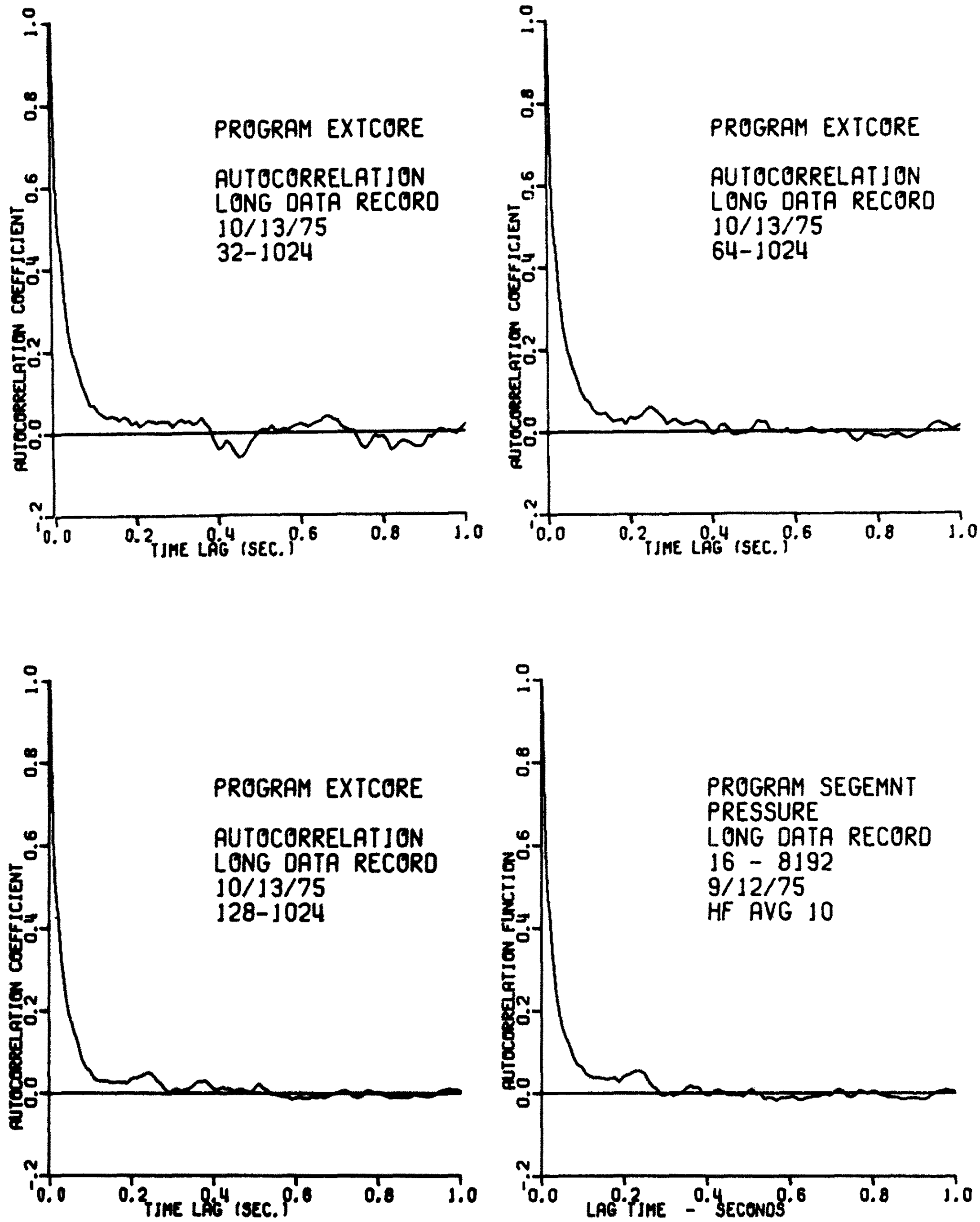
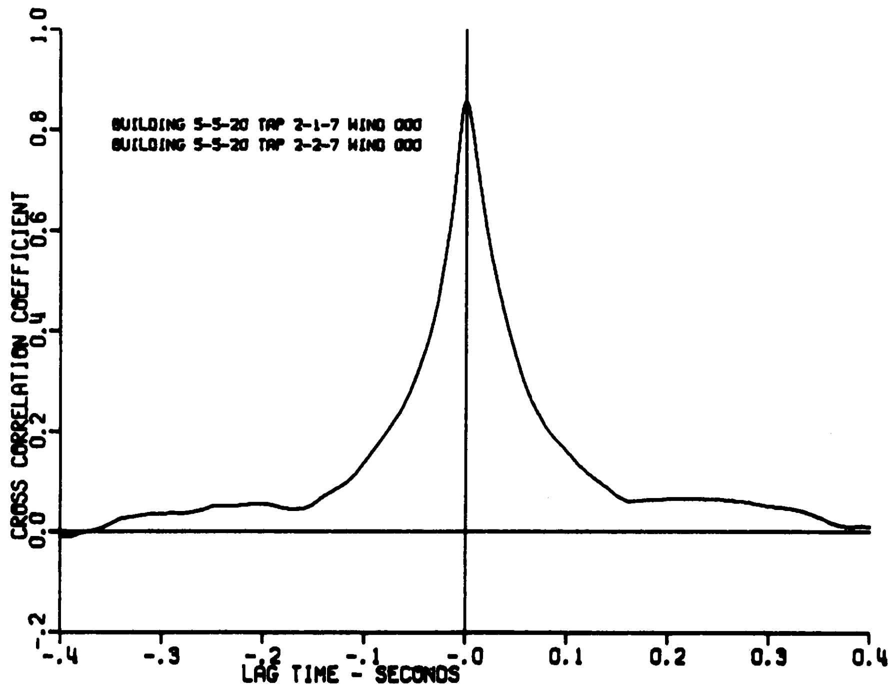
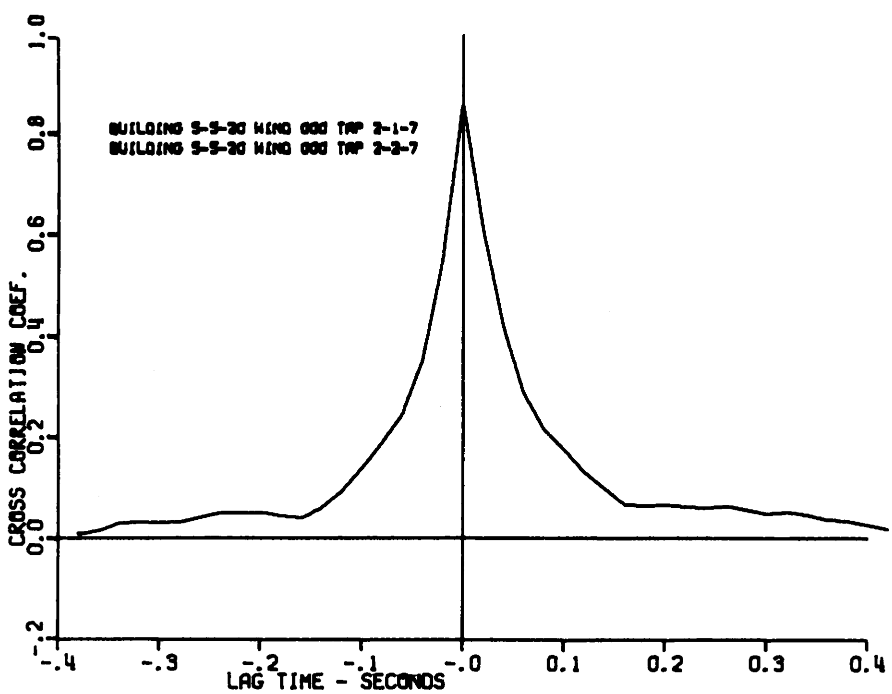


FIGURE 13. AUTOCORRELATIONS--PROGRAM EXTCORE.



(a) FFT COMPUTATION



(b) DIRECT COMPUTATION

FIGURE 14. COMPARISON OF CROSS-CORRELATION FUNCTIONS--DIRECT AND FFT COMPUTATION.

APPENDICES

A1 SUBROUTINE FOURT
IBM contributed Program No. 3600-13.4001

A2 SUBROUTINE FOR2D
IBM contributed Program No. 3600-13.4006

B1 PROGRAM CHECK

B2 PROGRAM SEGEMNT

B3 PROGRAM EXTCORE

B4 PROGRAM CSPECT2

B5 PROGRAM CSPECT3

```
      SUBROUTINE FOURT(DATA,NN,NDIM,ISIGN,IFORM,WORK)                   FFTT0000
C                                                                       FFTT0010
C     THE COOLEY-TUKEY FAST FOURIER TRANSFORM IN USASI BASIC FORTRAN    FFTT0020
C                                                                       FFTT0030
C     TRANSFORM(K1,K2,...) = SUM(DATA(J1,J2,...)*EXP(ISIGN*2*PI*SQRT(-1)FFTT0040
C     *((J1-1)*(K1-1)/NN(1)+(J2-1)*(K2-1)/NN(2)+...))), SUMMED FOR ALL  FFTT0050
C     J1, K1 BETWEEN 1 AND NN(1), J2, K2 BETWEEN 1 AND NN(2), ETC.      FFTT0060
C     THERE IS NO LIMIT TO THE NUMBER OF SUBSCRIPTS.  DATA IS A         FFTT0070
C     MULTIDIMENSIONAL COMPLEX ARRAY WHOSE REAL AND IMAGINARY           FFTT0080
C     PARTS ARE ADJACENT IN STORAGE, SUCH AS FORTRAN IV PLACES THEM.    FFTT0090
C     IF ALL IMAGINARY PARTS ARE ZERO (DATA ARE DISGUISED REAL), SET    FFTT0100
C     IFORM TO ZERO TO CUT THE RUNNING TIME BY UP TO FORTY PERCENT.     FFTT0110
C     OTHERWISE, IFORM = +1.  THE LENGTHS OF ALL DIMENSIONS ARE         FFTT0120
C     STORED IN ARRAY NN, OF LENGTH NDIM.  THEY MAY BE ANY POSITIVE     FFTT0130
C     INTEGERS, THO THE PROGRAM RUNS FASTER ON COMPOSITE INTEGERS, AND  FFTT0140
C     ESPECIALLY FAST ON NUMBERS RICH IN FACTORS OF TWO.  ISIGN IS +1   FFTT0150
C     OR -1.  IF A -1 TRANSFORM IS FOLLOWED BY A +1 ONE (OR A +1        FFTT0160
C     BY A -1) THE ORIGINAL DATA REAPPEAR, MULTIPLIED BY NTOT (=NN(1)*  FFTT0170
C     NN(2)*...).  TRANSFORM VALUES ARE ALWAYS COMPLEX, AND ARE RETURNEDFFTT0180
C     IN ARRAY DATA, REPLACING THE INPUT.  IN ADDITION, IF ALL          FFTT0190
C     DIMENSIONS ARE NOT POWERS OF TWO, ARRAY WORK MUST BE SUPPLIED,    FFTT0200
C     COMPLEX OF LENGTH EQUAL TO THE LARGEST NON 2**K DIMENSION.        FFTT0210
C     OTHERWISE, REPLACE WORK BY ZERO IN THE CALLING SEQUENCE.          FFTT0220
C     NORMAL FORTRAN DATA ORDERING IS EXPECTED, FIRST SUBSCRIPT VARYING FFTT0230
C     FASTEST.  ALL SUBSCRIPTS BEGIN AT ONE.                            FFTT0240
C                                                                       FFTT0250
C     RUNNING TIME IS MUCH SHORTER THAN THE NAIVE NTOT**2, BEING        FFTT0260
C     GIVEN BY THE FOLLOWING FORMULA.  DECOMPOSE NTOT INTO              FFTT0270
C     2**K2 * 3**K3 * 5**K5 * ....  LET SUM2 = 2*K2, SUMF = 3*K3 + 5*K5 FFTT0280
C     + ... AND NF = K3 + K5 + ....  THE TIME TAKEN BY A MULTI-         FFTT0290
C     DIMENSIONAL TRANSFORM ON THESE NTOT DATA IS T = T0 + NTOT*(T1+    FFTT0300
C     T2*SUM2+T3*SUMF+T4*NF).  ON THE CDC 3300 (FLOATING POINT ADD TIME FFTT0310
C     OF SIX MICROSECONDS), T = 3000 + NTOT*(500+43*SUM2+68*SUMF+       FFTT0320
C     320*NF) MICROSECONDS ON COMPLEX DATA.  IN ADDITION, THE           FFTT0330
C     ACCURACY IS GREATLY IMPROVED, AS THE RMS RELATIVE ERROR IS        FFTT0340
C     BOUNDED BY 3*2**(-B)*SUM(FACTOR(J)**1.5), WHERE B IS THE NUMBER   FFTT0350
C     OF BITS IN THE FLOATING POINT FRACTION AND FACTOR(J) ARE THE      FFTT0360
C     PRIME FACTORS OF NTOT.                                            FFTT0370
C                                                                       FFTT0380
C     PROGRAM BY NORMAN BRENNER FROM THE BASIC PROGRAM BY CHARLES       FFTT0390
C     RADER.  RALPH ALTER SUGGESTED THE IDEA FOR THE DIGIT REVERSAL.    FFTT0400
C     MIT LINCOLN LABORATORY, AUGUST 1967.  THIS IS THE FASTEST AND MOSTFFTT0410
C     VERSATILE VERSION OF THE FFT KNOWN TO THE AUTHOR.  SHORTER PRO-   FFTT0420
C     GRAMS FOUR1 AND FOUR2 RESTRICT DIMENSION LENGTHS TO POWERS OF TWO.FFTT0430
C     SEE-- IEEE AUDIO TRANSACTIONS (JUNE 1967), SPECIAL ISSUE ON FFT.  FFTT0440
C                                                                       FFTT0450
C     THE DISCRETE FOURIER TRANSFORM PLACES THREE RESTRICTIONS UPON THE FFTT0460
C     DATA.                                                             FFTT0470
C     1.  THE NUMBER OF INPUT DATA AND THE NUMBER OF TRANSFORM VALUES   FFTT0480
```

```
C      MUST BE THE SAME.                                                 FFTT0490
C      2.  BOTH THE INPUT DATA AND THE TRANSFORM VALUES MUST REPRESENT   FFTT0500
C      EQUISPACED POINTS IN THEIR RESPECTIVE DOMAINS OF TIME AND         FFTT0510
C      FREQUENCY.  CALLING THESE SPACINGS DELTAT AND DELTAF, IT MUST BE  FFTT0520
C      TRUE THAT DELTAF=2*PI/(NN(I)*DELTAT).  OF COURSE, DELTAT NEED NOT FFTT0530
C      BE THE SAME FOR EVERY DIMENSION.                                  FFTT0540
C      3.  CONCEPTUALLY AT LEAST, THE INPUT DATA AND THE TRANSFORM OUTPUTFFTT0550
C      REPRESENT SINGLE CYCLES OF PERIODIC FUNCTIONS.                    FFTT0560
C                                                                        FFTT0570
C      EXAMPLE 1.  THREE-DIMENSIONAL FORWARD FOURIER TRANSFORM OF A      FFTT0580
C      COMPLEX ARRAY DIMENSIONED 32 BY 25 BY 13 IN FORTRAN IV.           FFTT0590
C      DIMENSION DATA(32,25,13),WORK(50),NN(3)                           FFTT0600
C      COMPLEX DATA                                                      FFTT0610
C      DATA NN/32,25,13/                                                 FFTT0620
C      DO 1 I=1,32                                                       FFTT0630
C      DO 1 J=1,25                                                       FFTT0640
C      DO 1 K=1,13                                                       FFTT0650
C   1  DATA(I,J,K)=COMPLEX VALUE                                         FFTT0660
C      CALL FOURT(DATA,NN,3,-1,1,WORK)                                   FFTT0670
C                                                                        FFTT0680
C      EXAMPLE 2.  ONE-DIMENSIONAL FORWARD TRANSFORM OF A REAL ARRAY OF  FFTT0690
C      LENGTH 64 IN FORTRAN II.                                          FFTT0700
C      DIMENSION DATA(2,64)                                              FFTT0710
C      DO 2 I=1,64                                                       FFTT0720
C      DATA(1,I)=REAL PART                                               FFTT0730
C   2  DATA(2,I)=0.                                                      FFTT0740
C      CALL FOURT(DATA,64,1,-1,0,0)                                      FFTT0750
C                                                                        FFTT0760
       DIMENSION DATA(1),NN(1),IFACT(32),WORK(1)                         FFTT0770
       WR = 0.0
       WI = 0.0
       WSTPR = 0.0
       WSTPI = 0.0
       TWOPI=6.283185307                                                 FFTT0780
       IF(NDIM-1)920,1,1                                                 FFTT0790
1      NTOT=2                                                            FFTT0800
       DO 2 IDIM=1,NDIM                                                  FFTT0810
       IF(NN(IDIM))920,920,2                                             FFTT0820
2      NTOT=NTOT*NN(IDIM)                                                FFTT0830
C                                                                        FFTT0840
C      MAIN LOOP FOR EACH DIMENSION                                      FFTT0850
C                                                                        FFTT0860
       NP1=2                                                             FFTT0870
       DO 910 IDIM=1,NDIM                                                FFTT0880
       N=NN(IDIM)                                                        FFTT0890
       NP2=NP1*N                                                         FFTT0900
       IF(N-1)920,900,5                                                  FFTT0910
C                                                                        FFTT0920
C      FACTOR N                                                          FFTT0930
C                                                                        FFTT0940
5      M=N                                                               FFTT0950
```

```
      NTWO=NP1                                                          FFTT0960
      IF=1                                                              FFTT0970
      IDIV=2                                                            FFTT0980
10    IQUOT=M/IDIV                                                      FFTT0990
      IREM=M-IDIV*IQUOT                                                 FFTT1000
      IF(IQUOT-IDIV)50,11,11                                            FFTT1010
11    IF(IREM)20,12,20                                                  FFTT1020
12    NTWO=NTWO+NTWO                                                    FFTT1030
      M=IQUOT                                                           FFTT1040
      GO TO 10                                                          FFTT1050
20    IDIV=3                                                            FFTT1060
30    IQUOT=M/IDIV                                                      FFTT1070
      IREM=M-IDIV*IQUOT                                                 FFTT1080
      IF(IQUOT-IDIV)60,31,31                                            FFTT1090
31    IF(IREM)40,32,40                                                  FFTT1100
32    IFACT(IF)=IDIV                                                    FFTT1110
      IF=IF+1                                                           FFTT1120
      M=IQUOT                                                           FFTT1130
      GO TO 30                                                          FFTT1140
40    IDIV=IDIV+2                                                       FFTT1150
      GO TO 30                                                          FFTT1160
50    IF(IREM)60,51,60                                                  FFTT1170
51    NTWO=NTWO+NTWO                                                    FFTT1180
      GO TO 70                                                          FFTT1190
60    IFACT(IF)=M                                                       FFTT1200
C                                                                       FFTT1210
C     SEPARATE FOUR CASES--                                             FFTT1220
C        1. COMPLEX TRANSFORM OR REAL TRANSFORM FOR THE 4TH, 5TH,ETC.   FFTT1230
C           DIMENSIONS.                                                 FFTT1240
C        2. REAL TRANSFORM FOR THE 2ND OR 3RD DIMENSION.  METHOD--      FFTT1250
C           TRANSFORM HALF THE DATA, SUPPLYING THE OTHER HALF BY CON-   FFTT1260
C           JUGATE SYMMETRY.
C           TRANSFORM HALF THE DATA AT EACH STAGE, SUPPLYING THE OTHER  FFTT1290
C        3. REAL TRANSFORM FOR THE 1ST DIMENSION, N ODD.  METHOD--      FFTT1280
C           HALF BY CONJUGATE SYMMETRY.                                 FFTT1300
C        4. REAL TRANSFORM FOR THE 1ST DIMENSION, N EVEN.  METHOD--     FFTT1310
C           TRANSFORM A COMPLEX ARRAY OF LENGTH N/2 WHOSE REAL PARTS    FFTT1320
C           ARE THE EVEN NUMBERED REAL VALUES AND WHOSE IMAGINARY PARTS FFTT1330
C           ARE THE ODD NUMBERED REAL VALUES.  SEPARATE AND SUPPLY      FFTT1340
C           THE SECOND HALF BY CONJUGATE SYMMETRY.                      FFTT1350
C                                                                       FFTT1360
70    NON2=NP1*(NP2/NTWO)                                               FFTT1370
      ICASE=1                                                           FFTT1380
      IF(IDIM-4)71,90,90                                                FFTT1390
71    IF(IFORM)72,72,90                                                 FFTT1400
72    ICASE=2                                                           FFTT1410
      IF(IDIM-1)73,73,90                                                FFTT1420
73    ICASE=3                                                           FFTT1430
      IF(NTWO-NP1)90,90,74                                              FFTT1440
74    ICASE=4                                                           FFTT1450
      NTWO=NTWO/2                                                       FFTT1460
```

```
      N=N/2                                                             FFTT1470
      NP2=NP2/2                                                         FFTT1480
      NTOT=NTOT/2                                                       FFTT1490
      I=3                                                               FFTT1500
      DO 80 J=2,NTOT                                                    FFTT1510
      DATA(J)=DATA(I)                                                   FFTT1520
80    I=I+2                                                             FFTT1530
90    I1RNG=NP1                                                         FFTT1540
      IF(ICASE-2)100,95,100                                             FFTT1550
95    I1RNG=NP0*(1+NPREV/2)                                             FFTT1560
C                                                                       FFTT1570
C     SHUFFLE ON THE FACTORS OF TWO IN N.  AS THE SHUFFLING             FFTT1580
C     CAN BE DONE BY SIMPLE INTERCHANGE, NO WORKING ARRAY IS NEEDED     FFTT1590
C                                                                       FFTT1600
100   IF(NTWO-NP1)600,600,110                                           FFTT1610
110   NP2HF=NP2/2                                                       FFTT1620
      J=1                                                               FFTT1630
      DO 150 I2=1,NP2,NON2                                              FFTT1640
      IF(J-I2)120,130,130                                               FFTT1650
120   I1MAX=I2+NON2-2                                                   FFTT1660
      DO 125 I1=I2,I1MAX,2                                              FFTT1670
      DO 125 I3=I1,NTOT,NP2                                             FFTT1680
      J3=J+I3-I2                                                        FFTT1690
      TEMPR=DATA(I3)                                                    FFTT1700
      TEMPI=DATA(I3+1)                                                  FFTT1710
      DATA(I3)=DATA(J3)                                                 FFTT1720
      DATA(I3+1)=DATA(J3+1)                                             FFTT1730
      DATA(J3)=TEMPR                                                    FFTT1740
125   DATA(J3+1)=TEMPI                                                  FFTT1750
130   M=NP2HF                                                           FFTT1760
140   IF(J-M)150,150,145                                                FFTT1770
145   J=J-M                                                             FFTT1780
      M=M/2                                                             FFTT1790
      IF(M-NON2)150,140,140                                             FFTT1800
150   J=J+M                                                             FFTT1810
C                                                                       FFTT1820
C     MAIN LOOP FOR FACTORS OF TWO.  PERFORM FOURIER TRANSFORMS OF      FFTT1830
C     LENGTH FOUR, WITH ONE OF LENGTH TWO IF NEEDED.  THE TWIDDLE FACTORFFTT1840
C     W=EXP(ISIGN*2*PI*SQRT(-1)*M/(4*MMAX)).  CHECK FOR W=ISIGN*SQRT(-1)FFTT1850
C     AND REPEAT FOR W=ISIGN*SQRT(-1)*CONJUGATE(W).                     FFTT1860
C                                                                       FFTT1870
      NON2T=NON2+NON2                                                   FFTT1880
      IPAR=NTWO/NP1                                                     FFTT1890
310   IF(IPAR-2)350,330,320                                             FFTT1900
320   IPAR=IPAR/4                                                       FFTT1910
      GO TO 310                                                         FFTT1920
330   DO 340 I1=1,I1RNG,2                                               FFTT1930
      DO 340 J3=I1,NON2,NP1                                             FFTT1940
      DO 340 K1=J3,NTOT,NON2T                                           FFTT1950
      K2=K1+NON2                                                        FFTT1960
      TEMPR=DATA(K2)                                                    FFTT1970
```

```
      TEMPI=DATA(K2+1)                                                  FFTT1980
      DATA(K2)=DATA(K1)-TEMPR                                           FFTT1990
      DATA(K2+1)=DATA(K1+1)-TEMPI                                       FFTT2000
      DATA(K1)=DATA(K1)+TEMPR                                           FFTT2010
  340 DATA(K1+1)=DATA(K1+1)+TEMPI                                       FFTT2020
  350 MMAX=NON2                                                         FFTT2030
  360 IF(MMAX-NP2HF)370,600,600                                         FFTT2040
  370 LMAX=MAX0(NON2T,MMAX/2)                                           FFTT2050
      IF(MMAX-NON2)405,405,380                                          FFTT2060
  380 THETA=-TWOPI*FLOAT(NON2)/FLOAT(4*MMAX)                            FFTT2070
      IF(ISIGN)400,390,390                                              FFTT2080
  390 THETA=-THETA                                                      FFTT2090
  400 WR=COS(THETA)                                                     FFTT2100
      WI=SIN(THETA)                                                     FFTT2110
      WSTPR=-2.*WI*WI                                                   FFTT2120
      WSTPI=2.*WR*WI                                                    FFTT2130
  405 DO 570 L=NON2,LMAX,NON2T                                          FFTT2140
      M=L                                                               FFTT2150
      IF(MMAX-NON2)420,420,410                                          FFTT2160
  410 W2R=WR*WR-WI*WI                                                   FFTT2170
      W2I=2.*WR*WI                                                      FFTT2180
      W3I=W2R*WI+W2I*WR                                                 FFTT2200
      W3R=W2R*WR-W2I*WI                                                 FFTT2190
  420 DO 530 I1=1,I1RNG,2                                               FFTT2210
      DO 530 J3=I1,NON2,NP1                                             FFTT2220
      KMIN=J3+IPAR*M                                                    FFTT2230
      IF(MMAX-NON2)430,430,440                                          FFTT2240
  430 KMIN=J3                                                           FFTT2250
  440 KDIF=IPAR*MMAX                                                    FFTT2260
  450 KSTEP=4*KDIF                                                      FFTT2270
      DO 520 K1=KMIN,NTOT,KSTEP                                         FFTT2280
      K2=K1+KDIF                                                        FFTT2290
      K3=K2+KDIF                                                        FFTT2300
      K4=K3+KDIF                                                        FFTT2310
      IF(MMAX-NON2)460,460,480                                          FFTT2320
  460 U1R=DATA(K1)+DATA(K2)                                             FFTT2330
      U1I=DATA(K1+1)+DATA(K2+1)                                         FFTT2340
      U2R=DATA(K3)+DATA(K4)                                             FFTT2350
      U2I=DATA(K3+1)+DATA(K4+1)                                         FFTT2360
      U3R=DATA(K1)-DATA(K2)                                             FFTT2370
      U3I=DATA(K1+1)-DATA(K2+1)                                         FFTT2380
      IF(ISIGN)470,475,475                                              FFTT2390
  470 U4R=DATA(K3+1)-DATA(K4+1)                                         FFTT2400
      U4I=DATA(K4)-DATA(K3)                                             FFTT2410
      GO TO 510                                                         FFTT2420
  475 U4R=DATA(K4+1)-DATA(K3+1)                                         FFTT2430
      U4I=DATA(K3)-DATA(K4)                                             FFTT2440
      GO TO 510                                                         FFTT2450
  480 T2R=W2R*DATA(K2)-W2I*DATA(K2+1)                                   FFTT2460
      T2I=W2R*DATA(K2+1)+W2I*DATA(K2)                                   FFTT2470
      T3R=WR*DATA(K3)-WI*DATA(K3+1)                                     FFTT2480
```

```
      T3I=WR*DATA(K3+1)+WI*DATA(K3)                                     FFTT2490
      T4R=W3R*DATA(K4)-W3I*DATA(K4+1)                                   FFTT2500
      T4I=W3R*DATA(K4+1)+W3I*DATA(K4)                                   FFTT2510
      U1R=DATA(K1)+T2R                                                  FFTT2520
      U1I=DATA(K1+1)+T2I                                                FFTT2530
      U2R=T3R+T4R                                                       FFTT2540
      U2I=T3I+T4I                                                       FFTT2550
      U3R=DATA(K1)-T2R                                                  FFTT2560
      U3I=DATA(K1+1)-T2I                                                FFTT2570
      IF(ISIGN)490,500,500                                              FFTT2580

490   U4R=T3I-T4I                                                       FFTT2590
      U4I=T4R-T3R                                                       FFTT2600
      GO TO 510                                                         FFTT2610
500   U4R=T4I-T3I                                                       FFTT2620
      U4I=T3R-T4R                                                       FFTT2630
510   DATA(K1)=U1R+U2R                                                  FFTT2640
      DATA(K1+1)=U1I+U2I                                                FFTT2650
      DATA(K2)=U3R+U4R                                                  FFTT2660
      DATA(K2+1)=U3I+U4I                                                FFTT2670
      DATA(K3)=U1R-U2R                                                  FFTT2680
      DATA(K3+1)=U1I-U2I                                                FFTT2690
      DATA(K4)=U3R-U4R                                                  FFTT2700
520   DATA(K4+1)=U3I-U4I                                                FFTT2710
      KMIN=4*(KMIN-J3)+J3                                               FFTT2720
      KDIF=KSTEP                                                        FFTT2730
      IF(KDIF-NP2)450,530,530                                           FFTT2740
530   CONTINUE                                                          FFTT2750
      M=MMAX-M                                                          FFTT2760
      IF(ISIGN)540,550,550                                              FFTT2770
540   TEMPR=WR                                                          FFTT2780
      WR=-WI                                                            FFTT2790
      WI=-TEMPR                                                         FFTT2800
      GO TO 560                                                         FFTT2810
550   TEMPR=WR                                                          FFTT2820
      WR=WI                                                             FFTT2830
      WI=TEMPR                                                          FFTT2840
560   IF(M-LMAX)565,565,410                                             FFTT2850
565   TEMPR=WR                                                          FFTT2860
      WR=WR*WSTPR-WI*WSTPI+WR                                           FFTT2870
570   WI=WI*WSTPR+TEMPR*WSTPI+WI                                        FFTT2880
      IPAR=3-IPAR                                                       FFTT2890
      MMAX=MMAX+MMAX                                                    FFTT2900
      GO TO 360                                                         FFTT2910
C                                                                       FFTT2920
C     MAIN LOOP FOR FACTORS NOT EQUAL TO TWO.  APPLY THE TWIDDLE FACTOR FFTT2930
C     W=EXP(ISIGN*2*PI*SQRT(-1)*(J2-1)*(J1-J2)/(NP2*IFP1)), THEN        FFTT2940
C     PERFORM A FOURIER TRANSFORM OF LENGTH IFACT(IF), MAKING USE OF    FFTT2950
C     CONJUGATE SYMMETRIES.                                             FFTT2960
C                                                                       FFTT2970
600   IF(NTWO-NP2)605,700,700                                           FFTT2980
```

```
605   IFP1=NON2
      IF=1
      NP1HF=NP1/2
610   IFP2=IFP1/IFACT(IF)
      J1RNG=NP2
      IF(ICASE-3)612,611,612
611   J1RNG=(NP2+IFP1)/2
      J2STP=NP2/IFACT(IF)
      J1RG2=(J2STP+IFP2)/2
612   J2MIN=1+IFP2
      IF(IFP1-NP2)615,640,640
615   DO 635 J2=J2MIN,IFP1,IFP2
      THETA=-TWOPI*FLOAT(J2-1)/FLOAT(NP2)
      IF(ISIGN)625,620,620
620   THETA=-THETA
625   SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      WR=WSTPR+1.
      WI=WSTPI
      J1MIN=J2+IFP1
      DO 635 J1=J1MIN,J1RNG,IFP1
      I1MAX=J1+I1RNG-2
      DO 630 I1=J1,I1MAX,2
      DO 630 I3=I1,NTOT,NP2
      J3MAX=I3+IFP2-NP1
      DO 630 J3=I3,J3MAX,NP1
      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+WR
635   WI=TEMPR*WSTPI+WI*WSTPR+WI
640   THETA=-TWOPI/FLOAT(IFACT(IF))
      IF(ISIGN)650,645,645
645   THETA=-THETA
650   SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      KSTEP=2*N/IFACT(IF)
      KRANG=KSTEP*(IFACT(IF)/2)+1
      DO 698 I1=1,I1RNG,2
      DO 698 I3=I1,NTOT,NP2
      DO 690 KMIN=1,KRANG,KSTEP
      J1MAX=I3+J1RNG-IFP1
      DO 680 J1=I3,J1MAX,IFP1
      J3MAX=J1+IFP2-NP1
      DO 680 J3=J1,J3MAX,NP1
      J2MAX=J3+IFP1-IFP2
      K=KMIN+(J3-J1+(J1-I3)/IFACT(IF))/NP1HF
      IF(KMIN-1)655,655,665
```

```
655   SUMR=0.                                                      FFTT3500
      SUMI=0.                                                      FFTT3510
      DO 660 J2=J3,J2MAX,IFP2                                      FFTT3520
      SUMR=SUMR+DATA(J2)                                           FFTT3530
660   SUMI=SUMI+DATA(J2+1)                                         FFTT3540
      WORK(K)=SUMR                                                 FFTT3550
      WORK(K+1)=SUMI                                               FFTT3560
      GO TO 680                                                    FFTT3570
665   KCONJ=K+2*(N-KMIN+1)                                         FFTT3580
      J2=J2MAX                                                     FFTT3590
      SUMR=DATA(J2)                                                FFTT3600
      SUMI=DATA(J2+1)                                              FFTT3610
      OLDSR=0.                                                     FFTT3620
      OLDSI=0.                                                     FFTT3630
      J2=J2-IFP2                                                   FFTT3640
670   TEMPR=SUMR                                                   FFTT3650
      TEMPI=SUMI                                                   FFTT3660
      SUMR=TWOWR*SUMR-OLDSR+DATA(J2)                               FFTT3670
      SUMI=TWOWR*SUMI-OLDSI+DATA(J2+1)                             FFTT3680
      OLDSR=TEMPR                                                  FFTT3690
      OLDSI=TEMPI                                                  FFTT3700
      J2=J2-IFP2                                                   FFTT3710
      IF(J2-J3)675,675,670                                         FFTT3720
675   TEMPR=WR*SUMR-OLDSR+DATA(J2)                                 FFTT3730
      TEMPI=WI*SUMI                                                FFTT3740
      WORK(K)=TEMPR-TEMPI                                          FFTT3750
      WORK(KCONJ)=TEMPR+TEMPI                                      FFTT3760
      TEMPR=WR*SUMI-OLDSI+DATA(J2+1)                               FFTT3770
      TEMPI=WI*SUMR                                                FFTT3780
      WORK(K+1)=TEMPR+TEMPI                                        FFTT3790
      WORK(KCONJ+1)=TEMPR-TEMPI                                    FFTT3800
680   CONTINUE                                                     FFTT3810
      IF(KMIN-1)685,685,686                                        FFTT3820
685   WR=WSTPR+1.                                                  FFTT3830
      WI=WSTPI                                                     FFTT3840
      GO TO 690                                                    FFTT3850
686   TEMPR=WR                                                     FFTT3860
      WR=WR*WSTPR-WI*WSTPI+WR                                      FFTT3870
      WI=TEMPR*WSTPI+WI*WSTPR+WI                                   FFTT3880
690   TWOWR=WR+WR                                                  FFTT3890
      IF(ICASE-3)692,691,692                                       FFTT3900
691   IF(IFP1-NP2)695,692,692                                      FFTT3910
692   K=1                                                          FFTT3920
      I2MAX=I3+NP2-NP1                                             FFTT3930
      DO 693 I2=I3,I2MAX,NP1                                       FFTT3940
      DATA(I2)=WORK(K)                                             FFTT3950
      DATA(I2+1)=WORK(K+1)                                         FFTT3960
693   K=K+2                                                        FFTT3970
      GO TO 698                                                    FFTT3980
C                                                                  FFTT3990
C     COMPLETE A REAL TRANSFORM IN THE 1ST DIMENSION, N ODD, BY CON-   FFTT4000
```

```
C      JUGATE SYMMETRIES AT EACH STAGE.
C
695    J3MAX=I3+IFP2-NP1
       DO 697 J3=I3,J3MAX,NP1
       J2MAX=J3+NP2-J2STP
       DO 697 J2=J3,J2MAX,J2STP
       J1MAX=J2+J1RG2-IFP2
       J1CNJ=J3+J2MAX+J2STP-J2
       DO 697 J1=J2,J1MAX,IFP2
       K=1+J1-I3
       DATA(J1)=WORK(K)
       DATA(J1+1)=WORK(K+1)
       IF(J1-J2)697,697,696
696    DATA(J1CNJ)=WORK(K)
       DATA(J1CNJ+1)=-WORK(K+1)
697    J1CNJ=J1CNJ-IFP2
698    CONTINUE
       IF=IF+1
       IFP1=IFP2
       IF(IFP1-NP1)700,700,610
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    SINTH=SIN(THETA/2.)
       WSTPR=-2.*SINTH*SINTH
       WSTPI=SIN(THETA)
       WR=WSTPR+1.
       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                                                       FFTT4520
      JMIN=JMIN-2                                                       FFTT4530
      TEMPR=WR                                                          FFTT4540
      WR=WR*WSTPR-WI*WSTPI+WR                                           FFTT4550
      WI=TEMPR*WSTPI+WI*WSTPR+WI                                        FFTT4560
725   IF(IMIN-JMIN)710,730,740                                          FFTT4570
730   IF(ISIGN)731,740,740                                              FFTT4580
731   DO 735 I=IMIN,NTOT,NP2                                            FFTT4590
735   DATA(I+1)=-DATA(I+1)                                              FFTT4600
740   NP2=NP2+NP2                                                       FFTT4610
      NTOT=NTOT+NTOT                                                    FFTT4620
      J=NTOT+1                                                          FFTT4630
      IMAX=NTOT/2+1                                                     FFTT4640
745   IMIN=IMAX-2*NHALF                                                 FFTT4650
      I=IMIN                                                            FFTT4660
      GO TO 755                                                         FFTT4670
750   DATA(J)=DATA(I)                                                   FFTT4680
      DATA(J+1)=-DATA(I+1)                                              FFTT4690
755   I=I+2                                                             FFTT4700
      J=J-2                                                             FFTT4710
      IF(I-IMAX)750,760,760                                             FFTT4720
760   DATA(J)=DATA(IMIN)-DATA(IMIN+1)                                   FFTT4730
      DATA(J+1)=0.                                                      FFTT4740
      IF(I-J)770,780,780                                                FFTT4750
765   DATA(J)=DATA(I)                                                   FFTT4760
      DATA(J+1)=DATA(I+1)                                               FFTT4770
770   I=I-2                                                             FFTT4780
      J=J-2                                                             FFTT4790
      IF(I-IMIN)775,775,765                                             FFTT4800
775   DATA(J)=DATA(IMIN)+DATA(IMIN+1)                                   FFTT4810
      DATA(J+1)=0.                                                      FFTT4820
      IMAX=IMIN                                                         FFTT4830
      GO TO 745                                                         FFTT4840
780   DATA(1)=DATA(1)+DATA(2)                                           FFTT4850
      DATA(2)=0.                                                        FFTT4860
      GO TO 900                                                         FFTT4870
C                                                                       FFTT4880
C     COMPLETE A REAL TRANSFORM FOR THE 2ND OR 3RD DIMENSION BY         FFTT4890
C     CONJUGATE SYMMETRIES.                                             FFTT4900
C                                                                       FFTT4910
800   IF(I1RNG-NP1)805,900,900                                          FFTT4920
805   DO 860 I3=1,NTOT,NP2                                              FFTT4930
      I2MAX=I3+NP2-NP1                                                  FFTT4940
      DO 860 I2=I3,I2MAX,NP1                                            FFTT4950
      IMIN=I2+I1RNG                                                     FFTT4960
      IMAX=I2+NP1-2                                                     FFTT4970
      JMAX=2*I3+NP1-IMIN                                                FFTT4980
      IF(I2-I3)820,820,810                                              FFTT4990
810   JMAX=JMAX+NP2                                                     FFTT5000
820   IF(IDIM-2)850,850,830                                             FFTT5010
830   J=JMAX+NP0                                                        FFTT5020
```

```
      DO 840 I=IMIN,IMAX,2                                              FFTT5030
      DATA(I)=DATA(J)                                                   FFTT5040
      DATA(I+1)=-DATA(J+1)                                              FFTT5050
840   J=J-2                                                             FFTT5060
850   J=JMAX                                                            FFTT5070
      DO 860 I=IMIN,IMAX,NP0                                            FFTT5080
      DATA(I)=DATA(J)                                                   FFTT5090
      DATA(I+1)=-DATA(J+1)                                              FFTT5100
860   J=J-NP0                                                           FFTT5110
C                                                                       FFTT5120
C     END OF LOOP ON EACH DIMENSION                                     FFTT5130
C                                                                       FFTT5140
900   NP0=NP1                                                           FFTT5150
      NP1=NP2                                                           FFTT5160
910   NPREV=N                                                           FFTT5170
920   RETURN                                                            FFTT5180
      END
#
```

```
      SUBROUTINE FOR2D (IDATA,N,NDIM,ISIGN,IFORM,WORK,NELEM)
C     FOR2D COMPUTES A DISCRETE FOURIER TRANSFORM BY THE COOLEY-TUKEY
C     ALGORITHM.  THE ARRAY IS COMPLEX, MULTI-DIMENSIONAL AND KEPT ON
C     DIRECT ACCESS STORAGE.  THE NUMBER OF DATA IN EACH DIMENSION MUST
C     BE A POWER OF TWO.  RUNNING TIME IS PROPORTIONAL TO NTOT*
C     LOG2(NTOT), WHERE NTOT IS THE TOTAL NUMBER OF DATA.  ORDINARY
C     FOURIER TRANSFORM PROGRAMS RUN IN TIME NTOT**2.  THE TRANSFORM
C     IS DONE IN-PLACE ON THE DIRECT ACCESS STORAGE, AND AS MUCH OF THE
C     TRANSFORM AS POSSIBLE IS DONE IN CORE.  ENTIRELY IN-CORE
C     PROGRAMS ARE ALSO AVAILABLE (FOUR1, FOURG, FOUR2 AND FOURT).
C     WRITTEN BY NORMAN BRENNER, MIT LINCOLN LABORATORY, SEPTEMBER 1968.
C     SEE---IEEE AUDIO TRANSACTIONS (JUNE 1967), A SPECIAL ISSUE ON THE
C     FAST FOURIER TRANSFORM.
C
C     DIMENSION DATA(N(1),N(2),...,N(NDIM)),TRANSFORM(N(1),...,N(NDIM))
C     COMPLEX DATA,TRANSFORM
C     DIMENSION N(NDIM)
C     TRANSFORM(K1,K2,...) = SUM(DATA(J1,J2,...)*EXP(ISIGN*2*PI*I*
C     ((J1-1)*(K1-1)/N(1)+(J2-1)*(K2-1)/N(2)+...))), SUMMED FOR ALL
C     J1 FROM 1 TO N(1), J2 FROM 1 TO N(2), ETC., FOR ALL K1 FROM 1
C     TO N(1), K2 FROM 1 TO N(2), ETC., UP TO N(NDIM).  NDIM IS
C     UNLIMITED.  IF A SET OF DATA ARE ISIGN = -1 TRANSFORMED AND THEN
C     THE TRANSFORM VALUES +1 TRANSFORMED (OR VICE VERSA) THE RESULTS
C     WILL BE THE ORIGINAL DATA, MULTIPLIED BY NTOT = N(1)*...*N(NDIM).
C     IFORM MUST EQUAL 1.  FUTURE VERSIONS OF FOR2D WILL MAKE USE OF IT.
C     DATA ARE STORED ON DIRECT ACCESS STORAGE IN FILE NUMBER IDATA,
C     BROKEN INTO RECORDS OF LENGTH NELEM COMPLEX ELEMENTS (NELEM MUST
C     BE A POWER OF TWO).  TRANSFORM VALUES ARE RETURNED TO FILE IDATA,
C     REPLACING THE INPUT.
C
C     THE USER MUST SUPPLY TWO SUBROUTINES FOR I/O TO THE DIRECT
C     ACCESS STORAGE, DREAD AND DWRIT.  THE CALLING SEQUENCE IS CALL
C     DXXXX(IDATA,IREC,BUFFR,NREC,NELEM), MEANING NREC RECORDS (EACH
C     NELEM COMPLEX ELEMENTS LONG) ARE TO BE TRANSMITTED BETWEEN STORAGE
C     BUFFER BUFFR AND FILE NUMBER IDATA, RECORD NUMBER IREC (FROM 1
C     TO NTOT/NELRC).  THE BUFFER SUPPLIED WILL BE PART OF ARRAY WORK,
C     WHICH MUST BE SUPPLIED BY THE USER.  IT IS THREE RECORDS LONG.
C     FOR FASTEST RUNNING TIME, MAKE NELEM AS LARGE AS POSSIBLE.
      DIMENSION N(1), WORK(1)
      NTOT=1
      DO 10 IDIM=1,NDIM
   10 NTOT=NTOT*N(IDIM)
      NPREV=1
      DO 20 IDIM=1,NDIM
      NREM=NTOT/(N(IDIM)*NPREV)
      CALL BTRVD (IDATA,NPREV,N(IDIM),NREM,WORK,NELEM)
      CALL COL2D (IDATA,NPREV,N(IDIM),NREM,ISIGN,WORK,NELEM)
   20 NPREV=N(IDIM)*NPREV
      RETURN
```

```
      END

      SUBROUTINE BTRVD (IDATA,NPREV,N,NREM,BUFFR,NELEM)
C     SHUFFLE THE DATA BY BIT REVERSAL.
C     DIMENSION DATA(NPREV,N,NREM)
C     COMPLEX DATA
C     EXCHANGE DATA(J1,J2REV,J3) WITH DATA(J1,J2,J3), WHERE J2REV-1
C     IS THE BIT REVERSAL OF J2-1.  FOR EXAMPLE, LET N = 32.  THEN FOR
C     J2-1 = 10011, J2REV-1 = 11001, ETC.  DATA ARE COMPLEX AND STORED
C     ON DIRECT ACCESS STORAGE.  BUFFR IS A COMPLEX BUFFER THREE RECORDS
C     LONG, EACH RECORD OF LENGTH NELEM COMPLEX ELEMENTS.  NELEM MUST
C     BE LESS THAN HALF OF NPREV*N*NREM, THE TOTAL NUMBER OF ELEMENTS,
C     ELSE THE WHOLE TRANSFORM COULD BE DONE IN CORE.  NPREV, N, NREM
C     AND NELEM MUST BE POWERS OF TWO.
C     INTEGER INDICES MAY BECOME AS LARGE AS NPREV*N*NREM*2.
      DIMENSION BUFFR(1)
      IF (NELEM-NPREV) 10,10,20
C     DIMENSION DATA(NELEM,NPREV/NELEM,N,NREM)
 10   CALL SHUFD (IDATA,NELEM,NPREV/NELEM,N,NREM,BUFFR)
      RETURN
 20   IF (2*NELEM-N*NPREV) 50,30,30
C     DIMENSION DATA(2*NELEM,(NPREV*N*NREM)/(2*NELEM))
 30   IP0=2
      IP1=IP0*(2*NELEM)
      IP2=IP0*(NPREV*N*NREM)
      DO 40 I2=1,IP2,IP1
      IREC=1+(2*(I2-1))/IP1
      CALL DREAD (IDATA,IREC,BUFFR,2,NELEM)
      CALL BITRV (BUFFR,NPREV,N,(2*NELEM)/(NPREV*N))
 40   CALL DWRIT (IDATA,IREC,BUFFR,2,NELEM)
      RETURN
 50   NELRC=NELEM/NPREV
      NREC=N/NELRC
C     DIMENSION DATA(NPREV,NELRC,IREM,2,IPROD,NREM)
C     DEFINE R = LOG2(NREC) AND E = LOG2(NELRC).  THEN THE ENTIRE BIT
C     REVERSAL TAKES E STAGES, OF WHICH NO MORE THAN R+1 CAN TAKE FULL
C     PASSES THRU THE DATA.
      IP0=2
      IP1=IP0*NPREV
      IP2=IP1*NELRC
      IP5=IP2*NREC
      IP6=IP5*NREM
      IP4=IP5
 60   IF (IP4-IP1*MAX0(NELRC,NREC)) 170,170,70
C     IP4=IP5/2**(ISTAG-1)
C     IF ISTAG .GT. MIN(R,E) GO TO LAST TEST
 70   IP3=IP4/2
C     MERGE RECORDS DATA(I1,I2,I3,1,I5,I6) AND DATA(I1,I2,I3,2,I5,I6)
      DO 160 I6=1,IP6,IP4
      I3MAX=I6+IP3-IP2
      DO 160 I3=I6,I3MAX,IP2
```

```
      IREC0=1+(I3-1)/IP2
      IREC1=IREC0+IP3/IP2
      IF (IREC1-IREC0-1) 80,80,90
C     SAVE SOME ACCESS TIME IF THE RECORDS ARE ADJACENT
 80   CALL DREAD (IDATA,IREC0,BUFFR(IP2+1),2,NELEM)
      GO TO 100
 90   CALL DREAD (IDATA,IREC0,BUFFR(IP2+1),1,NELEM)
      CALL DREAD (IDATA,IREC1,BUFFR(2*IP2+1),1,NELEM)
 100  CALL MERGE (BUFFR(IP2+1),BUFFR(1),NPREV,NELRC)
C     MERGE THE EVEN NUMBERED ELEMENTS
      IBUFF=IP2+IP1+1
      CALL MERGE (BUFFR(IBUFF),BUFFR(IP2+1),NPREV,NELRC)
C     MERGE THE ODD-NUMBERED ELEMENTS
      IBUFF=1
C     THE RECORDS ARE NOW IN  BUFFERS 0 AND 1
      IF (IP5-NREC*IP3) 130,110,110
C     IF ISTAG .LT. R, GOTO WRITE
 110  IF (NREC-NELRC) 120,130,130
C     IF R .LT. E THEN DO SHUFC, ELSE WRITE OUT.
C     SUBROUTINES SHUFC AND SHUFD ARE MUTUALLY EXCLUSIVE--THE FIRST
C     REQUIRES THAT NELRC BE GREATER THAN NREC, WHILE THE LATTER
C     REQUIRES THE REVERSE.
 120  CALL SHUFC (BUFFR(IP2+1),BUFFR(2*IP2+1),NPREV,NELRC,NREC)
C     SHUFFLE BUFFER 1 AND PLACE INTO BUFFER 2
      CALL SHUFC (BUFFR(1),BUFFR(IP2+1),NPREV,NELRC,NREC)
C     SHUFFLE BUFFER 0 AND PLACE INTO BUFFER 1
      IBUFF=IP2+1
C     DATA ARE NOW IN BUFFERS 1 AND 2
 130  IF (IREC1-IREC0-1) 140,140,150
 140  CALL DWRIT (IDATA,IREC0,BUFFR(IBUFF),2,NELEM)
      GO TO 160
 150  CALL DWRIT (IDATA,IREC0,BUFFR(IBUFF),1,NELEM)
      IBUFF=IBUFF+IP2
      CALL DWRIT (IDATA,IREC1,BUFFR(IBUFF),1,NELEM)
 160  CONTINUE
      IP4=IP3
      GO TO 60
 170  IF (NREC-2*NELRC) 190,190,180
C     IF R .LE. E+1 RETURN
 180  CALL SHUFD (IDATA,NELEM,1,NREC/NELRC,NELRC*NREM,BUFFR)
C     BIT REVERSE THE RECORDS ON DISK.
 190  RETURN
      END

      SUBROUTINE BITRV (DATA,NPREV,N,NREM)
C     SHUFFLE THE DATA BY BIT REVERSAL.
C     DIMENSION DATA(NPREV,N,NREM)
C     COMPLEX DATA
C     EXCHANGE DATA(J1,J4REV,J5) WITH DATA(J1,J4,J5) FOR ALL J1 FROM 1
C     TO NPREV, ALL J4 FROM 1 TO N (WHICH MUST BE A POWER OF TWO), AND
C     ALL J5 FROM 1 TO NREM.  J4REV-1 IS THE BIT REVERSAL OF J4-1.  E.G.
```

```
C     SUPPOSE N = 32.  THEN FOR J4-1 = 10011, J4REV-1 = 11001, ETC.
      DIMENSION DATA(1)
      IP0=2
      IP1=IP0*NPREV
      IP4=IP1*N
      IP5=IP4*NREM
      I4REV=1
C     I4REV = 1+(J4REV-1)*IP1
      DO 60 I4=1,IP4,IP1
C     I4 = 1+(J4-1)*IP1
      IF (I4-I4REV) 10,30,30
 10   I1MAX=I4+IP1-IP0
      DO 20 I1=I4,I1MAX,IP0
C     I1 = 1+(J1-1)*IP0+(J4-1)*IP1
      DO 20 I5=I1,IP5,IP4
C     I5 = 1+(J1-1)*IP0+(J4-1)*IP1+(J5-1)*IP4
      I5REV=I4REV+I5-I4
C     I5REV = 1+(J1-1)*IP0+(J4REV-1)*IP1+(J5-1)*IP4
      TEMPR=DATA(I5)
      TEMPI=DATA(I5+1)
      DATA(I5)=DATA(I5REV)
      DATA(I5+1)=DATA(I5REV+1)
      DATA(I5REV)=TEMPR
 20   DATA(I5REV+1)=TEMPI
C     ADD ONE WITH DOWNWARD CARRY TO THE HIGH ORDER BIT OF J4REV-1.
 30   IP2=IP4/2
 40   IF (I4REV-IP2) 60,60,50
 50   I4REV=I4REV-IP2
      IP2=IP2/2
      IF (IP2-IP1) 60,40,40
 60   I4REV=I4REV+IP2
      RETURN
      END

      SUBROUTINE SHUFD (IDATA,NELEM,NPREV,N,NREM,BUFFR)
C     SHUFFLE THE RECORDS ON DIRECT ACCESS STORAGE BY BIT REVERSAL.
C     DIMENSION DATA(NELEM,NPREV,N,NREM)
C     COMPLEX DATA
C     EXCHANGE DATA(J1,J2,J4REV,J5) WITH DATA(J1,J2,J4,J5), WHERE
C     J4REV-1 IS THE BIT REVERSAL OF J4-1.  THIS CAN BE DONE BY AN
C     EXCHANGE OF RECORDS.
      DIMENSION BUFFR(1)
      IP0=2
      IP1=IP0*NELEM
      IP2=IP1*NPREV
      IP4=IP2*N
      IP5=IP4*NREM
      I4REV=1
      DO 60 I4=1,IP4,IP2
      IF (I4-I4REV) 10,30,30
 10   DO 20 I5=I4,IP5,IP4
```

```
      I2MAX=I5+IP2-IP1
      DO 20 I2=I5,I2MAX,IP1
      I2REV=I4REV+I2-I4
      IREC0=1+(I2-1)/IP1
      IREC1=1+(I2REV-1)/IP1
      CALL DREAD (IDATA,IREC0,BUFFR(1),1,NELEM)
      CALL DREAD (IDATA,IREC1,BUFFR(IP1+1),1,NELEM)
      CALL DWRIT (IDATA,IREC1,BUFFR(1),1,NELEM)
   20 CALL DWRIT (IDATA,IREC0,BUFFR(IP1+1),1,NELEM)
C     ADD ONE WITH DOWNWARD CARRY TO THE HIGH ORDER BIT OF J4REV-1.
   30 IP3=IP4/2
   40 IF (I4REV-IP3) 60,60,50
   50 I4REV=I4REV-IP3
      IP3=IP3/2
      IF (IP3-IP2) 60,40,40
   60 I4REV=I4REV+IP3
      RETURN
      END

      SUBROUTINE MERGE (FROM,TO,NPREV,NELRC)
C     MERGE TWO RECORDS INTO ONE.
C     DIMENSION FROM(NPREV,2,NELRC),TO(NPREV,NELRC)
C     COMPLEX FROM,TO
C     TO(J1,J3)=FROM(J1,1,J3)
      DIMENSION FROM(1), TO(1)
      IP0=2
      IP1=IP0*NPREV
      IP2=IP1*2
      IP3=IP2*NELRC
      ITO=1
      DO 10 I3=1,IP3,IP2
      I1MAX=I3+IP1-IP0
      DO 10 I1=I3,I1MAX,IP0
      TO(ITO)=FROM(I1)
      TO(ITO+1)=FROM(I1+1)
   10 ITO=ITO+IP0
      RETURN
      END

      SUBROUTINE SHUFC (FROM,TO,NPREV,NELRC,NREC)
C     SHUFFLE THE DATA IN CORE BY BIT REVERSAL.
C     DIMENSION FROM(NPREV,NELRC/NREC,NREC),TO(NPREV,NREC,NELRC/NREC)
C     COMPLEX FROM,TO
C     TO(J1,J4,J3REV)=FROM(J1,J3,J4) WHERE J3REV-1 IS THE BIT REVERSAL
C     OF J3-1.
      DIMENSION FROM(1), TO(1)
      IP0=2
      IP1=IP0*NPREV
      IP3=IP1*(NELRC/NREC)
      IP4=IP3*NREC
      I3REV=1
```

```
      DO 40 I3=1,IP3,IP1                                                SHC  13
      ITO=1+NREC*(I3REV-1)                                              SHC  14
      DO 10 I4=I3,IP4,IP3                                               SHC  15
      I1MAX=I4+IP1-IP0                                                  SHC  16
      DO 10 I1=I4,I1MAX,IP0                                             SHC  17
      TO(ITO)=FROM(I1)                                                  SHC  18
      TO(ITO+1)=FROM(I1+1)                                              SHC  19
 10   ITO=ITO+IP0                                                       SHC  20
C     ADD ONE WITH DOWNWARD CARRY TO THE HIGH ORDER BIT OF J3REV-1.     SHC  21
      IP2=IP3/2                                                         SHC  22
 20   IF (I3REV-IP2) 40,40,30                                           SHC  23
 30   I3REV=I3REV-IP2                                                   SHC  24
      IP2=IP2/2                                                         SHC  25
      IF (IP2-IP1) 40,20,20                                             SHC  26
 40   I3REV=I3REV+IP2                                                   SHC  27
      RETURN                                                            SHC  28
      END                                                               SHC  29-

      SUBROUTINE COL2D (IDATA,NPREV,N,NREM,ISIGN,BUFFR,NELEM)           C2D   1
C     DISCRETE FOURIER TRANSFORM OF LENGTH N.  IN-PLACE COOLEY-TUKEY    C2D   2
C     ALGORITHM, BIT-REVERSED TO NORMAL ORDER, SANDE-TUKEY PHASE SHIFTS.C2D   3
C     DIMENSION DATA(NPREV,N,NREM)                                      C2D   4
C     COMPLEX DATA                                                      C2D   5
C     DATA(J1,K4,J5) = SUM(DATA(J1,J4,J5)*EXP(ISIGN*2*PI*I*(J4-1)*      C2D   6
C     (K4-1)/N)), SUMMED OVER J4 = 1 TO N FOR ALL J1 FROM 1 TO NPREV,   C2D   7
C     K4 FROM 1 TO N AND J5 FROM 1 TO NREM.  N MUST BE A POWER OF TWO.  C2D   8
C     METHOD--LET IPREV TAKE THE VALUES 1, 2 OR 4, 4 OR 8, ..., N/16,   C2D   9
C     N/4, N.  THE CHOICE BETWEEN 2 OR 4, ETC., DEPENDS ON WHETHER N IS C2D  10
C     A POWER OF FOUR.  DEFINE IFACT = 2 OR 4, THE NEXT FACTOR THAT     C2D  11
C     IPREV MUST TAKE, AND IREM = N/(IFACT*IPREV).  THEN--              C2D  12
C     DIMENSION DATA(NPREV,IPREV,IFACT,IREM,NREM)                       C2D  13
C     COMPLEX DATA                                                      C2D  14
C     DATA(J1,J2,K3,J4,J5) = SUM(DATA(J1,J2,J3,J4,J5)*EXP(ISIGN*2*PI*I* C2D  15
C     (K3-1)*((J3-1)/IFACT+(J2-1)/(IFACT*IPREV)))), SUMMED OVER J3 = 1  C2D  16
C     TO IFACT FOR ALL J1 FROM 1 TO NPREV, J2 FROM 1 TO IPREV, K3 FROM  C2D  17
C     1 TO IFACT, J4 FROM 1 TO IREM AND J5 FROM 1 TO NREM.  THIS IS     C2D  18
C     A PHASE-SHIFTED DISCRETE FOURIER TRANSFORM OF LENGTH IFACT.       C2D  19
C     FACTORING N BY FOURS SAVES ABOUT TWENTY FIVE PERCENT OVER FACTOR- C2D  20
C     ING BY TWOS.  DATA MUST BE BIT-REVERSED INITIALLY.                C2D  21
C     IT IS NOT NECESSARY TO REWRITE THIS SUBROUTINE INTO COMPLEX       C2D  22
C     NOTATION SO LONG AS THE FORTRAN COMPILER USED STORES REAL AND     C2D  23
C     IMAGINARY PARTS IN ADJACENT STORAGE LOCATIONS.  IT MUST ALSO      C2D  24
C     STORE ARRAYS WITH THE FIRST SUBSCRIPT INCREASING FASTEST.         C2D  25
      DIMENSION BUFFR(1)                                                C2D  26
      TWOPI=6.2831853072*FLOAT(ISIGN)                                   C2D  27
      IF (2*NELEM-NPREV) 30,30,10                                       C2D  28
C     DIMENSION DATA(2*NELEM,(NPREV*N*NREM)/(2*NELEM))                  C2D  29
 10   IP0=2                                                             C2D  30
      IP1=IP0*(2*NELEM)                                                 C2D  31
      IP2=IP0*(NPREV*N*NREM)                                            C2D  32
      NMID=MIN0(N,(2*NELEM)/NPREV)                                      C2D  33
```

```
      NFIN=MAX0(1,(2*NELEM)/(NPREV*N))
      DO 20 I2=1,IP2,IP1
      IREC=1+(2*(I2-1))/IP1
      CALL DREAD (IDATA,IREC,BUFFR,2,NELEM)
      CALL COOL2 (BUFFR,NPREV,NMID,NFIN,ISIGN)
   20 CALL DWRIT (IDATA,IREC,BUFFR,2,NELEM)
C     DIMENSION DATA(NPREV,IPROD,2,IREM,NREM)
   30 IP0=2
      IP1=IP0*NPREV
      IP4=IP1*N
      IP5=IP4*NREM
      NWORD=IP0*NELEM
      IP2=IP0*MAX0(2*NELEM,NPREV)
   40 IF (IP2-IP4) 50,100,100
   50 IP3=IP2*2
      THETA=TWOPI/FLOAT(IP3/IP1)
      SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      IREC0=1
      IREC1=IREC0+IP2/NWORD
C     IREC0 AND IREC1 ARE NEVER ADJACENT RECORDS, SO MUST BE READ AND
C     WRITTEN SEPARATELY.
      CALL DREAD (IDATA,IREC0,BUFFR(1),1,NELEM)
      CALL DREAD (IDATA,IREC1,BUFFR(NWORD+1),1,NELEM)
      IELEM=1
      I3MIN=1
      DO 90 I5=1,IP5,IP3
      WR=1.
      WI=0.
      I2MAX=I5+IP2-IP1
      DO 90 I2=I5,I2MAX,IP1
      I1MAX=I2+IP1-IP0
      DO 80 I1=I2,I1MAX,IP0
      IF (IELEM-NELEM) 70,70,60
   60 CALL DWRIT (IDATA,IREC0,BUFFR(1),1,NELEM)
      CALL DWRIT (IDATA,IREC1,BUFFR(NWORD+1),1,NELEM)
      IREC0=1+(I1-1)/NWORD
      IREC1=IREC0+IP2/NWORD
      CALL DREAD (IDATA,IREC0,BUFFR(1),1,NELEM)
      CALL DREAD (IDATA,IREC1,BUFFR(NWORD+1),1,NELEM)
      IELEM=1
      I3MIN=I1
   70 I3A=I1-I3MIN+1
      I3B=I3A+NWORD
      TEMPR=WR*BUFFR(I3B)-WI*BUFFR(I3B+1)
      TEMPI=WR*BUFFR(I3B+1)+WI*BUFFR(I3B)
      BUFFR(I3B)=BUFFR(I3A)-TEMPR
      BUFFR(I3B+1)=BUFFR(I3A+1)-TEMPI
      BUFFR(I3A)=BUFFR(I3A)+TEMPR
      BUFFR(I3A+1)=BUFFR(I3A+1)+TEMPI
```

```
 80   IELEM=IELEM+1
      TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
 90   WI=TEMPR*WSTPI+WI*WSTPR+WI
      CALL DWRIT (IDATA,IREC0,BUFFR(1),1,NELEM)
      CALL DWRIT (IDATA,IREC1,BUFFR(NWORD+1),1,NELEM)
      IP2=IP3
      GO TO 40
 100  RETURN
      END

      SUBROUTINE COOL2 (DATA,NPREV,N,NREM,ISIGN)
C     DISCRETE FOURIER TRANSFORM OF LENGTH N.  IN-PLACE COOLEY-TUKEY
C     ALGORITHM, BIT-REVERSED TO NORMAL ORDER, SANDE-TUKEY PHASE SHIFTS.
C     DIMENSION DATA(NPREV,N,NREM)
C     COMPLEX DATA
C     DATA(J1,K4,J5) = SUM(DATA(J1,J4,J5)*EXP(ISIGN*2*PI*I*(J4-1)*
C     (K4-1)/N)), SUMMED OVER J4 = 1 TO N FOR ALL J1 FROM 1 TO NPREV,
C     K4 FROM 1 TO N AND J5 FROM 1 TO NREM.  N MUST BE A POWER OF TWO.
C     METHOD--LET IPREV TAKE THE VALUES 1, 2 OR 4, 4 OR 8, ..., N/16,
C     N/4, N.  THE CHOICE BETWEEN 2 OR 4, ETC., DEPENDS ON WHETHER N IS
C     A POWER OF FOUR.  DEFINE IFACT = 2 OR 4, THE NEXT FACTOR THAT
C     IPREV MUST TAKE, AND IREM = N/(IFACT*IPREV).  THEN--
C     DIMENSION DATA(NPREV,IPREV,IFACT,IREM,NREM)
C     COMPLEX DATA
C     DATA(J1,J2,K3,J4,J5) = SUM(DATA(J1,J2,J3,J4,J5)*EXP(ISIGN*2*PI*I*
C     (K3-1)*((J3-1)/IFACT+(J2-1)/(IFACT*IPREV)))), SUMMED OVER J3 = 1
C     TO IFACT FOR ALL J1 FROM 1 TO NPREV, J2 FROM 1 TO IPREV, K3 FROM
C     1 TO IFACT, J4 FROM 1 TO IREM AND J5 FROM 1 TO NREM.  THIS IS
C     A PHASE-SHIFTED DISCRETE FOURIER TRANSFORM OF LENGTH IFACT.
C     FACTORING N BY FOURS SAVES ABOUT TWENTY FIVE PERCENT OVER FACTOR-
C     ING BY TWOS.  DATA MUST BE BIT-REVERSED INITIALLY.
C     IT IS NOT NECESSARY TO REWRITE THIS SUBROUTINE INTO COMPLEX
C     NOTATION SO LONG AS THE FORTRAN COMPILER USED STORES REAL AND
C     IMAGINARY PARTS IN ADJACENT STORAGE LOCATIONS.  IT MUST ALSO
C     STORE ARRAYS WITH THE FIRST SUBSCRIPT INCREASING FASTEST.
      DIMENSION DATA(1)
      TWOPI=6.2831853072*FLOAT(ISIGN)
      IP0=2
      IP1=IP0*NPREV
      IP4=IP1*N
      IP5=IP4*NREM
      IP2=IP1
C     IP2=IP1*IPROD
      NPART=N
 10   IF (NPART-2) 60,30,20
 20   NPART=NPART/4
      GO TO 10
C     DO A FOURIER TRANSFORM OF LENGTH TWO
 30   IF (IP2-IP4) 40,160,160
 40   IP3=IP2*2
```

```
C       IP3=IP2*IFACT
        DO 50 I1=1,IP1,IP0
C       I1 = 1+(J1-1)*IP0
        DO 50 I5=I1,IP5,IP3
C       I5 = 1+(J1-1)*IP0+(J4-1)*IP3+(J5-1)*IP4
        I3A=I5
        I3B=I3A+IP2
C       I3 = 1+(J1-1)*IP0+(J2-1)*IP1+(J3-1)*IP2+(J4-1)*IP3+(J5-1)*IP4
        TEMPR=DATA(I3B)
        TEMPI=DATA(I3B+1)
        DATA(I3B)=DATA(I3A)-TEMPR
        DATA(I3B+1)=DATA(I3A+1)-TEMPI
        DATA(I3A)=DATA(I3A)+TEMPR
 50     DATA(I3A+1)=DATA(I3A+1)+TEMPI
        IP2=IP3
C       DO A FOURIER TRANSFORM OF LENGTH FOUR (FROM BIT REVERSED ORDER)
 60     IF (IP2-IP4) 70,160,160
 70     IP3=IP2*4
C       IP3=IP2*IFACT
        THETA=TWOPI/FLOAT(IP3/IP1)
        SINTH=SIN(THETA/2.)
        WSTPR=-2.*SINTH*SINTH
C       COS(THETA)-1, FOR ACCURACY.
        WSTPI=SIN(THETA)
        WR=1.
        WI=0.
        DO 150 I2=1,IP2,IP1
C       I2 = 1+(J2-1)*IP1
        IF (I2-1) 90,90,80
 80     W2R=WR*WR-WI*WI
        W2I=2.*WR*WI
        W3R=W2R*WR-W2I*WI
        W3I=W2R*WI+W2I*WR
 90     I1MAX=I2+IP1-IP0
        DO 140 I1=I2,I1MAX,IP0
C       I1 = 1+(J1-1)*IP0+(J2-1)*IP1
        DO 140 I5=I1,IP5,IP3
C       I5 = 1+(J1-1)*IP0+(J2-1)*IP1+(J4-1)*IP3+(J5-1)*IP4
        I3A=I5
        I3B=I3A+IP2
        I3C=I3B+IP2
        I3D=I3C+IP2
C       I3 = 1+(J1-1)*IP0+(J2-1)*IP1+(J3-1)*IP2+(J4-1)*IP3+(J5-1)*IP4
        IF (I2-1) 110,110,100
C       APPLY THE PHASE SHIFT FACTORS
 100    TEMPR=DATA(I3B)
        DATA(I3B)=W2R*DATA(I3B)-W2I*DATA(I3B+1)
        DATA(I3B+1)=W2R*DATA(I3B+1)+W2I*TEMPR
        TEMPR=DATA(I3C)
        DATA(I3C)=WR*DATA(I3C)-WI*DATA(I3C+1)
        DATA(I3C+1)=WR*DATA(I3C+1)+WI*TEMPR
```

```
      TEMPR=DATA(I3D)
      DATA(I3D)=W3R*DATA(I3D)-W3I*DATA(I3D+1)
      DATA(I3D+1)=W3R*DATA(I3D+1)+W3I*TEMPR
110   T0R=DATA(I3A)+DATA(I3B)
      T0I=DATA(I3A+1)+DATA(I3B+1)
      T1R=DATA(I3A)-DATA(I3B)
      T1I=DATA(I3A+1)-DATA(I3B+1)
      T2R=DATA(I3C)+DATA(I3D)
      T2I=DATA(I3C+1)+DATA(I3D+1)
      T3R=DATA(I3C)-DATA(I3D)
      T3I=DATA(I3C+1)-DATA(I3D+1)
      DATA(I3A)=T0R+T2R
      DATA(I3A+1)=T0I+T2I
      DATA(I3C)=T0R-T2R
      DATA(I3C+1)=T0I-T2I
      IF (ISIGN) 120,120,130
120   T3R=-T3R
      T3I=-T3I
130   DATA(I3B)=T1R-T3I
      DATA(I3B+1)=T1I+T3R
      DATA(I3D)=T1R+T3I
140   DATA(I3D+1)=T1I-T3R
      TEMPR=WR
      WR=WSTPR*TEMPR-WSTPI*WI+TEMPR
150   WI=WSTPR*WI+WSTPI*TEMPR+WI
      IP2=IP3
      GO TO 60
160   RETURN
      END
```

```
      PROGRAM CHECK(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
C        THIS PROGRAM WAS WRITTEN BY R. AKINS COLORADO STATE UNIVERSITY TO
C        ILLUSTRATE THE USE OF SUBROUTINE FOURT. A FORWARD AND INVERSE
C        TRANSFORM OF A KNOWN FUNCTION ARE PERFORMED AND THE RESULTS ARE
C        COMPARED WITH THE EXACT VALUES.
C
C        PROGRAM VARIABLES IN ALPHABETICAL ORDER ARE--
C
C            D - ARRAY USED AS INPUT AND OUTPUT FROM SUBROUTINE FOURT
C            DELTAT - TIME STEP OF INPUT FUNCTION
C            DELTAW - FREQUENCY STEP CORRESPONDING TO DELTAT
C            FREQ - ACTUAL FREQUENCY AT A GIVEN ELEMENT OF D
C            NUMBER - NUMBER OF DATA POINTS USED IN TRANSFORMS
C            NUMBE2 - NUMBER OF DATA POINTS AFTER REFLECTION USED IN TRANSFORMS
C            TIME - ACTUAL TIME AT A GIVEN ELEMTNT OF D
      DIMENSION D(2,4096)
C
C        READ INPUT VARIABLES
C
    3 READ(5,111)NUMBER,DELTAT
      IF(EOF(5))300,5
    5 NUMBE2=NUMBER*2
      DELTAW=6.2832/(DELTAT*FLOAT(NUMBE2))
C        COMPUTE INPUT EXPONENTIAL FUNCTION - STORE IT IT D(1,I) CORRESPONDING
C        TO THE REAL PART OF THE FOURT INPUT, PLACE A ZERO IN D(2,I)
C        CORRESPONDING TO THE IMAGINARY PART OF FOURT INPUT.
C
      DO 10 I=1,NUMBER
      D(1,I)=EXP(-FLOAT(I-1)*DELTAT)
   10 D(2,I)=0.0
C
C        REFLECT THE INPUT FUNCTION
C
      D(1,NUMBER+1)=D(1,NUMBER)
      DO 20 I=2,NUMBER
      K=NUMBER-I+2
      L=NUMBER+I
      D(2,L)=0.0
   20 D(1,L)=D(1,K)
C
C        PERFORM A FORWARD(-1) TRANSFORM ON THE DATA
C
      CALL FOURT(D,NUMBE2,1,-1,0,0)
C
C        COMPUTE ACTUAL TRANSFORM AND PRINT OUT A COMPARISON WITH THEOUTPUT
C        OF SUBROUTINE FOURT
C
      CPTIME=SECOND(A)
      WRITE(6,201)NUMBER,DELTAT,CPTIME
      WRITE(6,115)
      DO 30 I=1,NUMBER
   30 D(1,I)=D(1,I)*DELTAT*2.0
      DO 35 I=1,NUMBER,10
      ACTUAL=4.0/(1.0+(FLOAT(I-1)*DELTAW)**2)
      FREQ=FLOAT(I-1)*DELTAW
   35 WRITE(6,120)FREQ,D(1,I),ACTUAL
      D(2,1)=0
      D(2,NUMBER+1)=0
```

```
      D(1,NUMBER+1)=D(1,NUMBER)
      DO 40 I=2,NUMBER
      K=NUMBER-I+2
      L=NUMBER+I
      D(2,L)=0.0
      D(2,I)=0.0
   40 D(1,L)=D(1,K)
C
C        PERFORM AN INVERSE (+1) TRANSFORM OF THE DATA
C
      CALL FOURT(D,NUMBE2,1,1,0,0)
C
C        COMPARE THE RESULTS OF A FORWARD AND INVERSE TRANSFORM WITH
C        THE ORIGINAL DATA
C
      CPTIME=SECOND(A)
      WRITE(6,201)NUMBER,DELTAT,CPTIME
      WRITE(6,110)
      VALUE=D(1,1)
      DO 60 I=1,NUMBER,10
      TIME=FLOAT(I-1)*DELTAT
      D(2,I)=EXP(-TIME)
      D(1,I)=D(1,I)/(FLOAT(NUMBER)*DELTAT*4)
   60 WRITE(6,120)TIME,D(1,I),D(2,I)
  110 FORMAT(11X,*T(SEC)      COMPUTED   R(T)    ACTUAL R(T)*)
  111 FORMAT(I10,F10.3)
  115 FORMAT(11X,*W(RPS)      COMPUTED F(W)     ACTUAL F(W)*)
  120 FORMAT(10X,F7.3,5X,2E14.5)
  201 FORMAT(10X,*N = *,I4,5X,*DELTAT = *,F6.3,5X,*CPTIME = *,F8.5)
      GO TO 3
  300 CONTINUE
      END
```

```
      PROGRAM SEGEMNT(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE1)
C
C        THIS PROGRAM WAS WRITTEN 8/75 BY R. AKINS CSU TO COMPUTE POWER
C        SPECTRAL DENSITIES (PSD) FROM A TIME SERIES USING SUBROUTINE FOURT.
C        AND SEGMENT AVERAGING.  AN OPTION IS TO PERFORM AN INVERSE TRANSFORM OF
C        OF THE PSD AND OBTAIN AN AUTOCORRELATION (ACR) FUNCTION.  PLOTS OF BOTH
C        THE PSD AND THE ACR WILL BE MADE USING THE U200 HARD COPY PLOTTER
C
C
C
C         SUBROUTINES CALLED ARE
C         ALL PLOT SUBROUTINES ARE DESCRIBED IN THE CSU USERS MANUAL 1975 EDITIO
C
C         AXIS - PLOT ROUTINE
C         CURVE - PLOT ROUTINE
C         FRAME - PLOT ROUTINE
C         FIRSTPT - PLOT ROUTINE
C         FOURT - FFT SUBROUTINE CALLED FROM FTNLIB
C         IPS - SUBROUTINE TO INTEGRATE THE SPECTRA
C         LOCAT - PLOT ROUTINE
C         MACROT - CALCULATES INTEGRAL TIME SCALES FOR THE ACR
C         MICRO1 - CALCULATES MICROSCALE FROM (N**2)*F(N)
C         MICRO2 - CALCULATES MICROSCALE FROM ACR
C         PENZ - PLOT ROUTINE
C         READATA - READS DATA RECORD FORM TAPE1 (12 BIT WORDS)
C         SET - PLOT ROUTINE
C         SYMBOL - PLOT ROUTINE
C         UNPAK2 - CONVERTS DATA RECORD FROM 12 TO 60 BIT WORDS
C         VECCTR - PLOT ROUTINE
C
C         INPUT VARIABLES IN ALPHABETICAL ORDER ARE
C
C         GAIN - GAIN OF LINEAR TRANSDUCER
C         ICOR - CODE FOR CORRELATION CALCULATION
C         IRATE - SAMPLE RATE OF DATA
C         IREC - RECORD LENGTH OF TAPE1 (DATA TAPE)
C         KEY1-5 - PLOT LABELS FOR BOTH PSD AND ACR PLOT
C         LABX - X AXIS LABEL FOR PSD
C         LABY - Y AXIS LABEL FOR PSD
C         NSEGM - NUMBER OF SEGMENTS TO AVERAGE
C         TITLE - ALPHANUMERIC ARRAY USED TO LABEL PRINTED OUTPUT
C         XTIT - X AXIS LABEL FOR CORRELATION PLOT
C         YTIT - Y AXIS LABEL FOR CORRELATION PLOT
C
C         PROGRAM VARIABLES
C
C         A - ARRAY OF 12 BIT WORDS READ FROM TAPE INPUT TO UNPACK
C         B - ARRAY OF 60 BIT WORDS OUTPUT FROM UNPACK
C         CONST - NORMALIZING FACTOR FOR ACR
C         D - 2 DIMENSIONAL ARRAY USED TO SIMULATE COMPLEX NUMBERS
C         DELTAN - FREQUENCY INTERVAL OF SPECTRA
C         DELTAT - TIME STEP OF INPUT DATA
C         FACTOR - CONSTANT USED IN SPECTRA CALCULATIONS
C         IND - INDEX USED IN SETTING UP PLOT ARRAYS
C         KTAPER - UPPER LIMIT TAPER START
C         LTAPER - LOWER LIMIT TAPER CUTOFF
C         N - NUMBER/2
C         NEW - NUMBER/2
C         NPLOT - PLOT PARAMATER
C         NREC - NUMBER OF RECORDS TO BE READ FROM THE TAPE PER SEGMENT
```

```
C          NUMBER - LENGTH OF D ARRAY
C          RMS - COMPUTED VALUE OF RMS
C          SEGMEN - ARRAY USED TO STORE THE SEGMENT AVERAGED SPECTRA
C          TOTAL - FLOATING POINT VERSION OF NUMBER
C          UTAPER - TAPER FACTOR
C          X - INPUT ARRAY FOR PLOTS
C          XMEAN - RUNNING TOTAL USED IN MEAN CALCULATIONS
C          X2 - RUNNING TOTAL USED IN RMS  CALCULATIONS
C          Y - INPUT ARRAY FOR PLOTS
C
      COMMON D(2,8192),N,SEGMEN(4096)
      DIMENSION X(500),Y(500),TITLE(8)
      DIMENSION XTIT(4),YTIT(4),LABX(4),LABY(4)
      DIMENSION KEY1(3),KEY2(3),KEY3(3),KEY4(3),KEY5(3),KEY6(3)
      COMMON/UNPK/A(204),B(1020)
      COMMON/1/IREC,IRATE,K1
      DATA Y/500*0.0/
      DO 1 I=1,4096
    1 SEGMEN(I)=0.0
      NUMBER=8192
C
C         IN ORDER TO CHANGE THE SIZE OF ARRAY D, TWO CARDS NEED TO BE
C         CHANGED, THE DIMENSION CARD AND THE VALUE OF NUMBER
C
C         READ THE INPUT VARIABLES
C
       READ(5,501)TITLE
  501 FORMAT(8A10)
      READ(5,500)NSEGM,IREC,IRATE,ICOR,GAIN
  500 FORMAT(4I10,F10.3)
      READ(5,510)XTIT
      READ(5,510)YTIT
      READ(5,510)LABX
      READ(5,510)LABY
      READ(5,511)KEY1
      READ(5,511)KEY2
      READ(5,511)KEY3
      READ(5,511)KEY4
      READ(5,511)KEY5
      READ(5,511)KEY6
  510 FORMAT(4A10)
  511 FORMAT(3A10)
      CALL LOCAT(2RAT)
      CALL PENZ(5HBLACK,4HFELT)
C
C         READ INPUT DATA OFF OF TAPE1, COMPUTE THE MEAN AND THE RMS
C
C
      WRITE(6,600)TITLE,NSEGM,NUMBER,IREC,IRATE
      NREC=NUMBER/IREC+1
      DO 100 K=1,NSEGM
      ICOUNT=1
      XMEAN=0.0
      X2=0.0
  600 FORMAT(1H1,8A10,//,5X,*A SEGMENT AVERAGED SPECTRA WILL BE COMPUTED
     1 USING  *,I4,*  SEGMENTS*,/,5X,*OF LENGTH  *,I6,*. THE DATA IS IN
     2RECORDS *,I5,* VALUES LONG AT A SAMPLE*,/,5X,*RATE OF  *,I6,* SPS.
     3*,///,11X,*SEGMENT*,17X,*XMEAN*,16X,*RMS*)
       DO 10 K1=1,NREC
```

```
      CALL READATA
      CALL UNPAK2(A,B,IREC)
      DO 8 J=1,IREC
      B(J)=B(J)*GAIN
      D(1,ICOUNT)=B(J)
      D(2,ICOUNT)=0.0
      XMEAN=XMEAN+B(J)
      X2=X2+B(J)**2
      IF(ICOUNT.EQ.NUMBER)GO TO 12
    8 ICOUNT=ICOUNT+1
   10 CONTINUE
   12 TOTAL=FLOAT(NUMBER)
      XMEAN=XMEAN/TOTAL
      RMS=SQRT(ABS((X2-XMEAN*XMEAN*TOTAL)/TOTAL))
      WRITE(6,601)K,XMEAN,RMS
  601 FORMAT(13X,I4,15X,F10.7,10X,F10.7)
C
C          TAPER AND NORMALIZE THE DATA ( REMOVE MEAN, DIVIDE BY RMS)
C
      LTAPER=NUMBER/10
      KTAPER=NUMBER-LTAPER
      DO 15 I=1,NUMBER
      D(1,I)=(D(1,I)-XMEAN)/RMS
      IF(I.GT.LTAPER)GO TO 13
      FRAC=FLOAT(I-1)/FLOAT(LTAPER-1)
      UTAPER=COS(1.570796*(1-FRAC))**2
      D(1,I)=D(1,I)*UTAPER
      GO TO 15
   13 IF(I.LT.KTAPER)GO TO 15
      KK=(LTAPER-1)-(I-KTAPER)
      FRAC=FLOAT(KK)/FLOAT(LTAPER-1)
      UTAPER=COS(1.570796*(1.0-FRAC))**2
      D(1,I)=D(1,I)*UTAPER
   15 CONTINUE
C
C          PERFORM A FORWARD TRANSFORM OF THE DATA ARRAY D
C
      CALL FOURT(D,NUMBER,1,-1,0,0)
      NEW=NUMBER/2
      DELTAT=1.0/FLOAT(IRATE)
      FACTOR=2.0*1.143*DELTAT/TOTAL
      DELTAN=1./(TOTAL*DELTAT)
C
C          ADD THE INCREMENT INTO ARRAY SEGMEN, THE SEGMENT AVERAGED SPECTRA
C
      DO 20 I=1,NEW
   20 SEGMEN(I)=(FLOAT(K-1)*SEGMEN(I)+FACTOR*(D(1,I)**2+D(2,I)**2))/FLOA
     1T(K)
  100 CONTINUE
C
C         COMPUTE THE CORRELATION FUNCTION FROM THE N SEGM AVERAGED SPECTRA
C
      N=NUMBER/2
C
C          REFLECT THE SPECTRA INTO THE D ARRAY
C
      DO 110 I=1,N
      D(1,I)=SEGMEN(I)
  110 D(2,I)=0.0
```

```
      D(1,N+1)=D(1,N)
      D(2,N+1)=0.0
      DO 115 I=2,N
      K=N-I+2
      L=N+I
      D(2,L)=0.0
  115 D(1,L)=D(1,K)
C
C          PERFORM AN INVERSE TRANSFORM TO OBTAIN A CORRELATION FUNCTION
C
      CALL FOURT(D,NUMBER,1,1,0,0)
C
C          NORMALIZE THE CORRELATION FUNCTION
C
      CONST=D(1,1)
      DO 120 I=1,N
  120 D(1,I)=D(1,I)/CONST
C
C          PLACE THE NORMALIZED CORRELATION FUNCTION INTO ARRAY Y AND
C          GENERATE ARRAY X - TIME STEPS
C
      DO 130 I=1,50
      X(I)=FLOAT(I-1)*DELTAT
  130 Y(I)=D(1,I)
      IND=51
      DO 135 I=60,N,10
      Y(IND)=D(1,I)
      X(IND)=FLOAT(I-1)*DELTAT
      IF(X(IND).GT.1.0)GO TO 137
  135 IND=IND+1
  137 NPLOT=IND
C
C          OUTPUT AND PLOT THE CORRELATION FUNCTION
C
      WRITE(6,602)TITLE
      DO 140 I=1,NPLOT,5
  140 WRITE(6,603)X(I),Y(I),X(I+1),Y(I+1),X(I+2),Y(I+2),X(I+3),Y(I+3),X(
     1I+4),Y(I+4)
  602 FORMAT(1H1,8A10,/,10X,*AUTOCORRELATION FUNCTION*,//,5(*    TIME
     1    R(T)    *),//)
  603 FORMAT(10F10.6)
      CALL  SET(1.0,5.0,1.0,6.0,0.0,1.0,-.2,1.0,1,1,1)
      CALL  AXIS(0.0,0.0,YTIT,40,6.0,90.0,-.2,.2,1)
      CALL  AXIS(0.0,0.0,XTIT,-40,5.0,0.0,0.0,0.2,1)
      CALL  SYMBOL(2.0,4.6,.2,KEY1,0.0,30)
      CALL  SYMBOL(2.0,4.3,.2,KEY2,0.0,30)
      CALL  SYMBOL(2.0,4.0,.2,KEY3,0.0,30)
      CALL  SYMBOL(2.0,3.7,.2,KEY4,0.0,30)
      CALL  SYMBOL(2.0,3.4,.2,KEY5,0.0,30)
      CALL  SYMBOL(2.0,3.1,.2,KEY6,0.0,30)
      CALL  FRSTPT(0.0,0.0)
      CALL  VECTOR(1.0,0.0)
      NPLOT=NPLOT-1
      CALL  CURVE(X,Y,NPLOT,0,2)
      CALL  FRAME
      DO 153 I=1,500
  153 Y(I)=0.0
C
C          PLACE THE FIRST 10 POINTS OF THE SPECTRA INTO ARRAY Y AND ASSOCIATED
```

```
C          FREQUENCY INTO X
C
      DO 150 I=1,10
      X(I)=FLOAT(I)*DELTAN
  150 Y(I)=SEGMEN(I+1)
      IND=11
C
C          FREQUENCY AVERAGE 3 POINTS
C
      DO 154 I=11,50,3
      DO 157 J=1,3
  157 Y(IND)=Y(IND)+SEGMEN(I+J-1)
      Y(IND)=Y(IND)/3.0
      X(IND)=FLOAT(I+1)*DELTAN
  154 IND=IND+1
      N2=N-10
C
C          FREQUENCY AVERAGE 10 POINTS
C
      DO 155 I=60,N2,10
      DO 156 II=1,10
  156 Y(IND)=Y(IND)+SEGMEN(I+II-1)
      X(IND)=FLOAT(I+5-1)*DELTAN
      Y(IND)=Y(IND)/10.0
  155 IND=IND+1
      NPLOT=IND-1
      WRITE(6,607)SEGMEN(1)
  607 FORMAT(1H0,10X,*THE FIRST ELEMENT OF SEGMEN IS  *,E15.4)
      WRITE(6,604)TITLE
      DO 160 I=1,NPLOT,4
  160 WRITE(6,605)X(I),Y(I),X(I+1),Y(I+1),X(I+2),Y(I+2),X(I+3),Y(I+3)
  604 FORMAT(1H1,8A10,/,10X,*NORMALIZED POWER SPECTRAL DENSITY FUNCTION*
     1,//,4(*      FREQ-CPS         F(N)*),//)
  605 FORMAT(8E15.7)
      CALL IPS(SEGMEN,DELTAN,SUM,N,1)
      WRITE(6,608)SUM
  608 FORMAT(1H0,10X,*AREA OF SEGMEN = *,F10.5)
      CALL SET(1.50,9.25,1.75,12.95,0.01,1000.0,.0000001,1.0,2,7,4)
      CALL PERIM(5,0,7,0)
      CALL SYMBOL(3.5,-.8,.25,LABX,0.0,40)
      CALL SYMBOL(-.6,3.0,.25,LABY,90.0,40)
      CALL SYMBOL(1.0,3.5,.2,KEY1,0.0,30)
      CALL SYMBOL(1.0,3.2,.2,KEY2,0.0,30)
      CALL SYMBOL(1.0,2.9,.2,KEY3,0.0,30)
      CALL SYMBOL(1.0,2.6,.2,KEY4,0.0,30)
      CALL SYMBOL(1.0,2.3,.2,KEY5,0.0,30)
      CALL SYMBOL(1.0,2.0,.2,KEY6,0.0,30)
      CALL CURVE(X,Y,NPLOT,0,2)
      CALL FRAME
      CALL MACROT(DELTAT)
      CALL MICRO2(DELTAT)
      CALL MICRO1(DELTAN,RMICRO1)
      WRITE(6,606)RMICRO1
  606 FORMAT(1H0,10X,*MICROSCALE COMPUTED BY INTEGRATING N2F(N)*,F10.6)
      END
```

```
      PROGRAM EXTCORE(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE1,TAPE2,
     1TAPE3)
C
C          THIS PROGRAM WAS WRITTEN BY R. AKINS TO COMPUTE A POWER SPECTRAL
C          USING SUBROUTINE FOR2D, AN EXTERNAL CORE FFT ROUTINE
C
C          SUBROUTINES CALLED IN ALPHABETICAL ORDER ARE -
C          ALL PLOT ROUTINES ARE DISCUSSED IN THE CSU USERS MANUAL
C
C          CURVE - PLOT ROUTINE
C          DREAD - READS RECORDS FROM MASS STORAGE
C          DWRIT - WRITES ON MASS STORAGE, REWRITING OVER OLD DATA USED AFTER THE
C          DATA HAS BEEN WRITTEN ONCE
C          DWRIT1 - WRITES ON MASS STORAGE - FIRST TIME
C          FOR2D - EXTERNAL CORE FFT
C          FRAME - PLOT ROUTINE
C          IPS - INTEGRATION SUBROUTINE
C          LOCAT - PLOT ROUTINE
C          PENZ - PLOT ROUTINE
C          PERIM - PLOT ROUTINE
C          OPENMS - SETS UP MASS STORAGE - SYSTEM SUBROUTINE
C          READATA - READS 1 DATA RECORD IREC VALUES LONG FROM TAPE1 USING ARRAY
C          SYMBOL -PLOT ROUTINE
C          UNPAK2 - CAHNGES FROM 12 BIT TO 60 BIT WORDS
C
C          TAPE UNITS USED -
C
C          TAPE 1 - DATA TAPE
C          TAPE 2 - MASS STORAGE
C          TAPE 3 - OUTPUT FOR EQUALLY AVERAGED SPECGRA
C          TAPE 5 - CARD INPUT
C          TAPE 6 - PRINTED OUTPUT
C          A - ARRAY OF 12 BIT WORDS INPUT TO UNPACK, READ FROM TAPE1
C          B - ARRAY OF 60 BIT WORDS OUT PUT FROM UNPACK
C          DELATN - FREQUENCY STEP FOR GIVEN AVERAGING INTERVAL
C          DELTAT - TIME STEP OF DATA, 1/IRATE
C          FACTOR - FACTOR TO MULTIPLY OUTPUT OF FOR2D
C          FREQ - REAL ARRAY USED TO STORE THE FREQUENCY VALUES FOR SPECT
C          GAIN - CALIBRATION FACTOR
C          ICHAN - CHANNEL TO BE USED
C          ICOUNT - COUNTER USED IN TAPERING, AND IN INITIALLY PLACING THE DATA
C          INTO MASS STORAGE
C          INDEX - INTEGER ARRAY USED IN MASS STORAGE CONTROL
C          INDEX1 - COUNTER USED TO KEEP TRACK OF MASS STORAGE LOCATIONS ON INPUT
C          IRATE - SAMPLE RATE PER CHANNEL TAPE1
C          IREC - NUMBER OF DATA VALUES PER DATA RECORD, TAPE1
C          ISP - COUNTER USED IN FREQUENCY SMOOTHING
C          KEY 1 - TITLE CARD FOR PLOT OF SPECTRUM
C          KEY2 -  TITLE CARD FOR PLOT OF SPECTRUM
C          KEY3 - TITLE CARD FOR PLOT OF SPECTRUM
C          KEY4 - TITLE CARD FOR PLOT OF SPECTRUM
C          KEY5 - TITLE CARD FOR PLOT OF SPECTRUM
C          KEY6 - TITLE CARD FOR PLOT OF SPECTRUM
C          KTAPER - USED IN TAPERING THE DATA
C          LABX - PLOT AXIS LABEL
C          LABY - PLOT AXIS LABEL
C          LIMIT(1,I) - NUMBER OF POINTS TO AVERAGE ITH INTERVAL
C          LIMIT(2,I) - NUMBER OF RAW POINTS I-TH INTERVAL
C          LIMIT(3,I) - NUMBER OF AVERAGED POINTS I-TH INTERVAL
```

```
C          LTAPER - USED IN TAPERING THE DATA
C          N - ARRAY GIVING DIMENSION OF ENTIRE DATA ARRAY INPUT TO FOR2D
C          NAVG - NUMBER OF AVERAGING INTERVALS
C          NAVG1 - UNIFORM AVERAGING TO BE USED IN OUTPUT TO TAPE3
C          NCHAN - NUMBER OF CHANNELS OF DATA ON TAPE 1
C          NREC - COUNTER USED IN INTEGRATION
C          NRECORD - TOTAL NUMBER OF DATA RECORDS ON TAPE
C          NTREC - NUMBER OF RECORDS NEEDED TO READ N(1) VALUES FROM TAPE 1
C          NUMBER - LENGTH OF DATA RECORDS IN MASS STORAGE
C          N1 - NUMBER OF RECORDS TO BE USED IN MASS STORAGE +1
C          PFUNC(X) - GAIN*X - CALIBRATION FOR A LINEAR TRANSDUCER
C          PKHI - HIGHEST VALUE OF RECORD
C          PKLO - SMALLEST VALUE OF RECORD
C          RMS - ROOT-MEAN-SQUARE OF THE INPUT DATA
C          RMS2 - RMS**2
C          SPECT - REAL ARRAY USED TO STORE THE SMOOTHED SPECTRUM
C          SQ - SUM OF SQUARES OF DATA VALUES
C          STORE - REAL ARRAY USED IN UNIFORM SMOOTHING OF THE SPECTRUM
C          TEMP - COMPLEX ARRAY NUMBER ELEMENTS LONG, USED WITH FOR2D
C          TOTAL - TOTAL NUMBER OF POINTS USED IN THE FFT
C          UTAPER - USED IN TAPERING THE DATA
C          WORK - COMPLEX ARRAY 3*NUMBER ELEMENTS LONG USED WITH FOR2D AND MASS S
C          STORAGE READ AND WRITE ROUTINES
C          XINC - ARRAY USED IN INTEGRATION TO STORE FREQUENCY INCREMENTS
C          XINT - RUNNING VALUE OF INTEGRAL OS SPECTRA
C          XLIMIT(1,I) - BANDWIDTH FOR THE I-TH AVERAGING INTERVAL
C          XLIMIT(2,I) - UPPER LIMIT FOR THE I-TH AVERAGING INTERVAL
C          XMEAN - RUNNING MEAN
C
      COMPLEX TEMP
      COMPLEX WORK
      COMMON TEMP(1024)
      COMMON/UNPK/A(204),B(1020)
      COMMON/1/IREC,NCHAN,IRATE
      COMMON/2/IRUN,DIAM,LENGT,IWIND,J2
      DIMENSION WORK(3072),N(3),INDEX(513),SPECT(1200),FREQ(2000)
      DIMENSION STORE(8)
      DIMENSION LABX(4),LABY(4),KEY1(3),KEY2(3),KEY3(3),KEY4(3),KEY5(3),
     1KEY6(3)
      DIMENSION XLIMIT(2,6),LIMIT(3,6)
      DIMENSION XINC(6)
      DATA SPECT/1200*0.0/
      DATA STORE/8*0.0/
      PFUNC(X)=GAIN*X
C
C          READ IN PARAMETERS FOR PROGRAM EXECUTION
C
      XINT=0.0
      CALL PENZ(5HBLACK,4HFELT)
      GAIN = .04114
      READ(5,1000)IREC,NCHAN,IRATE,N(1),NAVG,NRECORD,NUMBER,N1
 1000 FORMAT(8I10)
      READ(5,1000)ICHAN,NAVG1
      READ(5,1001)LABX
      READ(5,1001)LABY
      READ(5,1002)KEY1
      READ(5,1002)KEY2
      READ(5,1002)KEY3
      READ(5,1002)KEY4
```

```
      READ(5,1002)KEY5
      READ(5,1002)KEY6
 1001 FORMAT(4A10)
 1002 FORMAT(3A10)
      READ(5,1003)(LIMIT(1,J),J=1,NAVG)
      READ(5,1003)(LIMIT(2,J),J=1,NAVG)
 1003 FORMAT(6I10)
C
C          OPEN MASS STORAGE
C
      CALL OPENMS(2,INDEX,N1,0)
      N1=N1-1
      NTREC=N(1)*NCHAN/IREC+1
      IF(NTREC.GT.NRECORD)STOP11
C
C          INITIALIZE PROGRAM PARAMETERS
C
      XMEAN=0.0
      PKHI=-100.0
      PKLO=100.0
      SQ=0.0
      ICOUNT=0
      INDEX1=1
      ICHAN=ICHAN-1
      DO 10 I=1,NTREC
C
C          READ THE DATA OFF OF TAPE1, UNPACK IT FORM 12 TO 60 BIT WORDS
C
      CALL READATA
      CALL UNPAK2(A,B,IREC)
      DO 5 JJ=1,IREC,NCHAN
      J=JJ+ICHAN
      B(J)=PFUNC(B(J))
      XMEAN=XMEAN+B(J)
      SQ=SQ+B(J)*B(J)
      IF(B(J).LT.PKHI)GO TO 3
      PKHI=B(J)
    3 IF(B(J).GT.PKLO)GO TO 5
      PKLO=B(J)
    5 CONTINUE
C
C          PLACE THE DATA INTO MASS STORAGE 1 RECORD NUMBER DATA VALUES LONG
C          AT A TIME
C
      DO 10 JJ=1,IREC,NCHAN
      J=JJ+ICHAN
      ICOUNT=ICOUNT+1
      WORK(ICOUNT)=B(J)
      IF(ICOUNT.NE.NUMBER) GO TO 10
      ICOUNT=0
      CALL DWRIT1(2,INDEX1,WORK,1,NUMBER)
      INDEX1=INDEX1+1
   10 CONTINUE
C
C          COMPUTE THE MEAN AND THE RMS
C
      TOTAL=FLOAT(NTREC)*FLOAT(IREC)/FLOAT(NCHAN)
      XMEAN=XMEAN/TOTAL
      RMS=SQRT(ABS((SQ-XMEAN*XMEAN*TOTAL)/TOTAL))
```

```
      WRITE(6,2000)IREC,IRATE,N1,NUMBER,XMEAN,RMS,PKHI,PKLO
 2000 FORMAT(1H1,10X,*TRIAL RUN OF FOR2D FOR PRESSURE SPECTRA*,//,10X,*R
     1ECORD LENGTH = *,I7,* SAMPLE RATE = *,I7,*SAMPLES/SECOND*,//,10X,*
     2FOR2D WAS CALLED USING*,I5,* RECORDS OF LENGTH*,I7,//,10X,*
     4MEAN = *,F10.6,20X,*(ALL UNITS PSI)*,/,10X,*RMS = *,F10.6,//,10X,*
     5PEAK HIGH = *,F10.6//,10X,*PEAK LOW = *,F10.6)
C
C         RECALL THE DATA,REMOVE THE MEAN,TAPER IF APPROPIATE ,RETURN TO STORAGE
C
      LTAPER=N(1)/10
      KTAPER=N(1)-LTAPER
      ICOUNT=1
      DO 20 J=1,N1
      CALL DREAD(2,J,WORK,1,NUMBER)
      DO 15 K=1,NUMBER
      WORK(K)=WORK(K)-XMEAN
      IF(ICOUNT.GT.LTAPER)GO TO 12
      FRAC=FLOAT(ICOUNT-1)/FLOAT(LTAPER-1)
      UTAPER=COS(1.570796*(1.0-FRAC))**2
      WORK(K)=WORK(K)*UTAPER
   12 IF(ICOUNT.LT.KTAPER)GO TO 14
      KK=(LTAPER-1)-(ICOUNT-KTAPER)
      FRAC=FLOAT(KK)/FLOAT(LTAPER-1)
      UTAPER=COS(1.570796*(1.0-FRAC))**2
      WORK(K)=WORK(K)*UTAPER
   14 ICOUNT=ICOUNT+1
   15 CONTINUE
   20 CALL DWRIT(2,J,WORK,1,NUMBER)
C
C         PERFORM A FORWARD TRANSFORM ON THE DATA
C
      CALL FOR2D(2,N,1,-1,1,WORK,NUMBER)
C
C         READ OUT THE TRANSFORMED VALUES, CONVERT TO A POWER SPECTRAL
C         DENSITY, FREQUENCY AVERAGE
C
      ISP=1
      TOTAL=FLOAT(N(1))
      DELTAT=1.0/FLOAT(IRATE)
      FACTOR=2.0*1.143*DELTAT/TOTAL
      RMS2=RMS**2
      DO 35 J=1,NAVG
      DELTAN=FLOAT(LIMIT(1,J))/(TOTAL*DELTAT)
      XLIMIT(1,J)=DELTAN
      LIMIT(3,J)=LIMIT(2,J)/LIMIT(1,J)
   35 XLIMIT(2,J)=LIMIT(3,J)*DELTAN
      DO 36 J=2,NAVG
   36 XLIMIT(2,J)=XLIMIT(2,J)+XLIMIT(2,J-1)
      WRITE(6,2004)(XLIMIT(2,J),LIMIT(1,J),XLIMIT(1,J),J=1,NAVG)
 2003 FORMAT(////,10X,*SCHEME OF VARIABLE BANDWIDTH SPECTRUM SMOOTHING*)
 2004 FORMAT(10X,*UPPER LIMIT CPS *,F10.2,5X,*NUMBER OF POINTS AVERAGED/
     1OUTPUT POINT*,I9,5X,*BANDWIDTH *,F10.4)
      NREC=1
      K1=0
      DO 40 J=1,NAVG
C
C        COMPUTE THE NUMBER OF RECORDS NECESSARY TO COMPUTE THIS PORTION
C         OF THE SPECTRUM
C
```

```
      J3=NUMBER/LIMIT(1,J)
      J2=LIMIT(1,J)
      J1=LIMIT(2,J)/NUMBER
      DELTN =FLOAT(J2)/(TOTAL*DELTAT)
      J1=NREC+J1-1
      DO 45 I=NREC,J1
      CALL DREAD(2,I,WORK,1,NUMBER)
      DO 37 K=1,NUMBER
      WRITE(3,300)WORK(K)
  300 FORMAT(2E12.4)
   37 WORK(K)=FACTOR*(REAL(WORK(K))**2+AIMAG(WORK(K))**2)
      IADD=0
C
C          SMOOTH THE SPECTRUM USING VARIABLE BANDWIDTH TECHNIQUES
C
      DO 39 KK=1,J3
      DO 38 L=1,J2
   38 SPECT(ISP)=SPECT(ISP)+REAL(WORK(IADD+L))
      SPECT(ISP)=SPECT(ISP)/(FLOAT(J2)*RMS2)
      IF(ISP.EQ.1)FREQ(ISP)=+DELTN/2.0
      IF(ISP.EQ.1)GO TO 30
      IF(I.EQ.NREC.AND.KK.EQ.1)FREQ(ISP)=FREQ(ISP-1)+DELTN/2.0+ODELTN/2.
     10
      IF(KK.EQ.1.AND.I.EQ.NREC)           GO TO 30
      FREQ(ISP)=FREQ(ISP-1)+DELTN
   30 IADD=IADD+J2
   39 ISP=ISP+1
   45 CONTINUE
C
C          INTEGRATE THE SPECTRUM LEAVING OUT THE END PORTIONS
C
      XINC(J)=DELTN
   60 K1=K1+LIMIT(3,J)
   63 CALL IPS(SPECT,XINC(J),SUM,K1,LIMIT(3,J))
      XINT=XINT+SUM
      ODELTN=DELTN
   40 NREC=J1+1
C
C         ADD ENDPOINTS AND OTHER ODD REGIONS
C
      IND=1
      KL=NAVG+1
      DO 80 I4=1,KL
      IF(I4.NE.1)GO TO 71
      XINT=XINT+XINC(I4)*SPECT(IND)/2.0
      GO TO 78
   71 IF(I4.NE.KL)GO TO 72
      XINT=XINT+XINC(I4-1)*SPECT(IND-1)/2.0
      GO TO 80
   72 XINT=XINT+SPECT(IND)*(XINC(I4)+XINC(I4-1))/2.0
   78 IND=IND+LIMIT(3,I4)
   80 CONTINUE
      WRITE(6,2008)XINT
 2008 FORMAT(1H0,10X,*THE AREA UNDER THE SPECTRUM IS *,F8.4)
C
C          OUTPUT THE SMOOTHED SPECTRUM
C
      WRITE(6,2001)
      M=ISP-1
```

```
      DO 50 J=1,M,4
   50 WRITE(6,2002)FREQ(J),SPECT(J),FREQ(J+1),SPECT(J+1),FREQ(J+2),SPECT
     1(J+2),FREQ(J+3),SPECT(J+3)
 2001 FORMAT(1H1,10X,*SMOOTHED SPECTRUM*,/,10X,*FREQ CPS*,10X,*G(N)*)
 2002 FORMAT(8E15.4)
C
C         PLOT THE SMOOTHED SPECTRUM
C
      CALL LOCAT(2RAT)
      CALL SET(1.50,9.25,1.75,12.95,0.01,1000.0,.0000001,1.0,2,7,4)
      CALL PERIM(5,0,7,0)
      CALL SYMBOL(3.5,-.8,.25,LABX,0.0,40)
      CALL SYMBOL(-.6,3.0,.25,LABY,90.0,40)
      CALL SYMBOL(1.0,3.5,.2,KEY1,0.0,30)
      CALL SYMBOL(1.0,3.2,.2,KEY2,0.0,30)
      CALL SYMBOL(1.0,2.9,.2,KEY3,0.0,30)
      CALL SYMBOL(1.0,2.6,.2,KEY4,0.0,30)
      CALL SYMBOL(1.0,2.3,.2,KEY5,0.0,30)
      CALL SYMBOL(1.0,2.0,.2,KEY6,0.0,30)
      CALL CURVE(FREQ,SPECT,M,4,2)
      CALL FRAME
      END
```

```
      PROGRAM CSPECT2(INPUT=101B,OUTPUT=202B,TAPE5=INPUT,TAPE6=OUTPUT,
     1TAPE2=513,TAPE3=513B,TAPE4=513B)
C
C          THIS PROGRAM WAS WRITTEN 11/75 BY R. AKINS TO COMPUTE AND PLOT
C          A COHERENCE FUNCTION USING SEGMENT AVERAGING AND READING THE
C          SINGLE CHANNEL TRANSFORMS FROM A DISC DEVICE, TAPE2 AND TAPE3.
C
C          SUBROUTINES CALLED (ALL PLOT SUBROUTINES ARE DESCRIBED IN THE
C          CSU USERS MANUAL, 1975 EDITION)
C
C          AXIS - PLOT ROUTINE
C          CURVE - PLOT ROUTINE
C          LOCAT - PLOT ROUTINE
C          PENZ - PLOT ROUTINE
C          RSTR - PLOT ROUTINE
C          SET - PLOT ROUTINE
C          SKIPF - TAPE CONTROL
C          SYMBOL - PLOT ROUTINE
C
C          INPUT VARIABLES ARE
C
C          IRATE - SAMPLE RATE OF ORIGINAL TIME SERIES
C          NRUN - NUMBER OF RUNS
C          NSEG - NUMBER OF SEGMENTS
C          NSKIP1 - TAPE CONTROL PARAMETER
C          NSKIP2 - TAPE CONTROL PARAMETER
C          NUMBER - LENGTH OF EACH SEGMENT
C          TITLE1 - LABEL FOR CHANNEL 1
C          TITLE2 LABEL FOR CHANNEL 2
C          X - COMPLEX ARRAY STORING TRANSFORM OF CHANNEL 1
C          XTIT - PLOT AXIS LABEL
C          Y - COMPLEX ARRAY STORING TRANSFORM OF CHANNEL 2
C          YTIT - PLOT AXIS LABEL
C
C          PROGRAM VARIABLES
C
C          A - FACTOR USED IN FREQUENCY AVERAGING
C          COH - ARRAY STORING FREQUENCY AVERAGED COHERENCE
C          DELTAN - FREQUENCY INCREMENT OF SPECTRA
C          FREQ - ARRAY STORING FREQUENCY STEPS FOR COHERENCE
C          IND - INDEX USED IN FREQUENCY AVERAGING
C          NPLOT - TOTAL NUMBER OF POINTS TO PLOT
C          N2 - NUMBER/2
C          GXY - SEGMENT AVERAGED CROSS SPECTRAL DENSITY
C          SPECT1 - SINGLE CHANNEL SEGMENT AVERAGED SPECTRA CHANNEL 1
C          SPECT2 - SINGLE CHANNEL SEGMENT AVERAGED SPECTRA CHANNEL 2
C
C          TAPE UNITS USED
C
C          TAPE2 - MASTER INPUT TAPE
C          TAPE3 - DISC USED AS INPUT FOR CHANNEL 1
C          TAPE4 - DISC USED AS INPUT FOR CHANNEL 2
C          TAPE5 - INPUT FILE
C          TAPE6 - OUTPUT FILE
C
      COMMON FREQ(500),TITLE1(8),TITLE2(8),COH(500),XTIT(4),YTIT(4),
     1SPECT1(500),SPECT2(500),GXY(500)
      COMMON X(4096),Y(4096)
      COMPLEX GXY
```

```
      COMPLEX X,Y
C
C         READ INPUT VARIABLES FOR ALL RUNS
C
      READ(5,500)NRUN
      READ(5,500)IRATE,NUMBER,NSEG
      READ(5,502)XTIT,YTIT
      CALL PENZ(5HBLACK,4HFELT)
      CALL LOCAT(2RAT)
      ICODE=1
      DO 100 KTOT=1,NRUN
C
C         ZERO NECESSARY ARRAYS
C
      DO 1 I=1,500
      SPECT2(I)=0.0
      SPECT1(I)=0.0
    2 COH(I)=0.0
    1 GXY(I)=(0.0,0.0)
C
C         READ INPUT VARIABLES FOR EACH RUN
C
      READ(5,501)TITLE1,TITLE2
      READ(5,500)NSKIP1,NSKIP2
  500 FORMAT(3I10)
  501 FORMAT(8A10)
  502 FORMAT(4A10)
      DELTAN=FLOAT(IRATE)/FLOAT(NUMBER)
C
C         COPY INPUT ARRAYS FROM TAPE TO DISCS
C
      REWIND 3
      REWIND 4
      DO 3 I=1,NSEG
      READ(2)X
    3 WRITE(3)X
      BACKSPACE2
      CALL SKIPF(2,NSKIP1,178,1)
      DO 4 I=1,NSEG
      READ(2)Y
    4 WRITE(4)Y
      BACKSPACE2
      CALL SKIPF(2,NSKIP2,178,1)
      REWIND 3
      REWIND 4
C
C         COMPUTE AND SEGMENT AVERAGE SINGLE CHANNEL SPECTRA AND
C         CROSS SPECTRAL DENSITY
C
      DO 30 JT=1,NSEG
      READ(3)X
      READ(4)Y
      DO 20 I=1,10
      SPECT1(I)=SPECT1(I)+CABS(X(I+1))**2
      SPECT2(I)=SPECT2(I)+CABS(Y(I+1))**2
   20 GXY(I)=GXY(I)+CONJG(X(I+1))*Y(I+1)
      IND=11
      DO 22 K=11,49,3
      DO 21 J=1,3
```

```
      SPECT1(IND)=SPECT1(IND)+CABS(X(K+J))**2
      SPECT2(IND)=SPECT2(IND)+CABS(Y(K+J))**2
   21 GXY(IND)=GXY(IND)+CONJG(X(K+J))*Y(K+J)
   22 IND=IND+1
      N2=NUMBER/2-25
      DO 25 I=50,N2,20
      DO 24 J=1,20
      SPECT1(IND)=SPECT1(IND)+CABS(X(I+J))**2
      SPECT2(IND)=SPECT2(IND)+CABS(Y(I+J))**2
   24 GXY(IND)=GXY(IND)+CONJG(X(I+J))*Y(I+J)
   25 IND=IND+1
   30 CONTINUE
C
C          SET UP THE FREQUENCY ARRAY
C
      A=FLOAT(NSEG)
C
C         COMPUTE AND FREQUENCY AVERAGE THE COHERENCE FUNCTION
C
      DO 35 I=1,10
      GXY(I)=GXY(I)/A
      COH(I)=(CABS(GXY(I))**2)*(A**2)/(SPECT1(I)*SPECT2(I))
   35 FREQ(I)=FLOAT(I)*DELTAN
      IND=11
      A=3.0*A
      DO 36 I=11,49,3
      GXY(IND)=GXY(IND)/A
      COH(IND)=(CABS(GXY(IND))**2)*(A**2)/(SPECT1(IND)*SPECT2(IND))
      FREQ(IND)=FLOAT(I+2)*DELTAN
   36 IND=IND+1
      A=20.0*A/3.0
      DO 37 I=50,N2,20
      GXY(IND)=GXY(IND)/A
      COH(IND)=(CABS(GXY(IND))**2)*(A**2)/(SPECT1(IND)*SPECT2(IND))
      FREQ(IND)=(FLOAT(I)+ 10.5 )DELTAN
      IF(FREQ(IND).GT.250.0)GO TO 60
   37 IND=IND+1
   60 NPLOT=IND-1
C
C         OUTPUT COHERENCE
C
      WRITE(6,610)TITLE1,TITLE2
      DO 50 I=1,NPLOT,3
   50 WRITE(6,611)FREQ(I),COH(I),FREQ(I+1),COH(I+1),FREQ(I+2),COH(I+2)
  611 FORMAT(6X,3(F9.2,8X,F7.4,7X))
  610 FORMAT(1H1,10X,*COHERENCE*,/,10X,*CHANNEL 1 *,8A10,/,10X,*CHANNEL
     12 *,8A10,//,6X,3(*FREQUENCY (HZ)    COHERENCE      *))
C         PLOT COHERENCE
C
      CALL SET(1.0,6.0,1.0,6.0,0.0,300.0,-.2,1.0,-1,1,1)
      CALL AXIS(0.0,0.0,XTIT,-40,6.0,0.0,0.0,50.0,-1)
      CALL AXIS(0.0,0.0,YTIT,40,6.0,90.0,-.2,.2,1)
      CALL SYMBOL(3.0,6.0,.1,TITLE1,0.0,80)
      CALL SYMBOL(3.0,5.8,.1,TITLE2,0.0,80)
      CALL CURVE(FREQ,COH,NPLOT,0,0)
      CALL RSTR(ICODE)
      ICODE=ICODE+1
      GO TO(100,90)ICODE
   90 ICODE=0
  100 CONTINUE
      END
```

```
      PROGRAM CSPECT3(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE2,TAPE3,
     1TAPE4)
C
C     THIS PROGRAM WAS WRITTEN 11/75 BY R. AKINS TO COMPUTE AND PLOT
C     CROSS-CORRELATION FUNCTIONS USING SEGMENT AVERAGING AND READING
C     THE SINGLE CHANNEL TRANSFORMS FROM A DISC DEVICE, TAPE 2 AND TAPE 3.
C
C     SUBROUTINES CALLED (ALL PLOT SUBROUTINES ARE DESCRIBED IN THE
C     CSU USERS MANUAL, 1975 EDITION)
C
C     AXIS - PLOT ROUTINE
C     CURVE - PLOT ROUTINE
C     FOURT - FFT ROUTINE
C     FRSTPT - PLOT ROUTINE
C     LOCAT - PLOT ROUTINE
C     PENZ - PLOT ROUTINE
C     ROTATE - PLOT ROUTINE
C     RSTR - PLOT ROUTINE
C     SET - PLOT ROUTINE
C     SKIPF - TAPE CONTROL SUBROUTINE - CSU USERS MANUAL
C     SYMBOL - PLOT ROUTINE
C     VECOTR - PLOT ROUTINE
C
C     INPUT VARIABLES ARE
C
C     IRATE - SAMPLE RATE OF INITIAL TIME SERIES
C     NRUN - NUMBER OF RUNS
C     NSEG - NUMBER OF SEGMENTS
C     NSKIP1 - TAPE CONTROL PARAMETER
C     NSKIP2 - TAPE CONTROL PARAMETER
C     NUMBER - LENGTH OF SINGLE CHANNEL TRANSFORMS
C     TITLE1 - CHANNEL 1 TITLE
C     TITLE2 - CHANNEL 2 TITLE
C     XTIT - PLOT TITLE X-AXIS
C     YTIT - PLOT TITLE Y-AXIS
C
C     PROGRAM VARIABLES
C
C     DELTAN - FREQUENCY STEP OF X,Y,GXY
C     DELTAT - TIME STEP OF CROSS-CORRELATION
C     FACTOR - USED TWICE - FACTOR IN CROSS-SPECTRUM CALCULATION AND LATER
C     CROSS CORRELATION CALCULATIONS
C     GXY - COMPLEX ARRAY WITH SEGMENT AVERAGED CROSS SPECTRUM
C     INDEX - USED IN OUTPUT AND PLOTTING
C     N2 - NUMBER/2
C     R12 - REAL ARRAY STORING CROSS-CORRELATION FUNCTION
C     TIME - REAL ARRAY WITH TIEM LAGS USED IN OUTPUT
C     X - COMPLEX ARRAY STORING TRANSFORM OF CHANNEL 1 TIME SERIES
C     Y - COMPLEX ARRAY STORING TRANSFORM OF CHANNEL 2 TIME SERIES
C
C
C     TAPE UNITS USED
C
C     TAPE2 - MASTER INPUT TAPE
C     TAPE3 - DISC USED AS INPUT FOR CHANNEL 1
C     TAPE4 - DISC USED AS INPUT FOR CHANNEL 2
C     TAPE5 - INPUT FILE
C     TAPE6 - OUTPUT FILE
C
```

```
      DIMENSION TIME(137),R12(137),TITLE1(4),TITLE2(4),XTIT(4),YTIT(4)
      COMMON GXY(8192),X(4096),Y(4096)
      COMPLEX GXY
      COMPLEX X,Y
C
C         READ INPUT VARIABLES FOR ALL RUNS
C
      READ(5,500)NRUN
      READ(5,500)IRATE,NUMBER,NSEG
      READ(5,502)XTIT,YTIT
      CALL LOCAT(2RAT)
      CALL PENZ(5HBLACK,4HFELT)
      ICODE=1
      DO 100 KLIM=1,NRUN
      DO 1 I=1,4096
    1 GXY(I)=(0.0,0.0)
C
C         READ INPUT VARIABLES WHICH CHANGE EACH RUN
C
      READ(5,501)TITLE1,TITLE2
      READ(5,500)NSKIP1,NSKIP2
  500 FORMAT(3I10)
  501 FORMAT(4A10)
  502 FORMAT(4A10)
      DELTAN=FLOAT(IRATE)/FLOAT(NUMBER)
      DELTAT=1.0/FLOAT(IRATE)
C
C         COPY INPUT ARRAYS X AND Y FROM DATA TAPE TO SEPARATE DISC FILES
C
      REWIND3
      REWIND4
      DO 3 I=1,NSEG
      READ(2)X
    3 WRITE(3)X
      BACKSPACE2
      CALL SKIPF(2,NSKIP1,178,1)
      DO 4 I=1,NSEG
      READ(2)Y
    4 WRITE(4)Y
      BACKSPACE2
      CALL SKIPF(2,NSKIP2,178,1)
      REWIND3
      REWIND4
C
C         CALCULATE SEGEMNT AVERAGED CROSS SPECTRAL DENSITY FUNCTION
C
      DO 30 JT=1,NSEG
      READ(3)X
      READ(4)Y
      DO 30 J=1,4096
   30 GXY(J)=GXY(J)+CONJG(X(J))*Y(J)
      FACTOR=2.0*1.143/FLOAT(IRATE)/FLOAT(NUMBER)/FLOAT(NSEG)
      DO 32 I=1,4096
   32 GXY(I)=FACTOR*GXY(I)
C
C         REFLECT THE CROSS SPECTRAL DENSITY FUNCTION
C
      NDOUB=NUMBER
      N2=NUMBER/2
```

```
      GXY(N2+1)=CONJG(GXY(N2))
      DO 33 I=1,4095
      K=NUMBER-I+1
   33 GXY(K)=CONJG(GXY(I+1))
C
C          PERFORM AN INVERSE  TRANSFORM TO OBTAIN THE CROSS-CORRELATION FUNCTION
C
      CALL FOURT(GXY,NDOUB,1,1,1,0)
      FACTOR=FLOAT(IRATE)/2.0/FLOAT(NUMBER)
C
C          PLACE SELECTED VALUES OF THE CROSS-CORRELATION FUNCTION INTO ARRAY
C          R12 AND ASSOCIATED TIME LAGS INTO ARRAY TIME FOR OUTPUT AND
C          PLOTTING
C
      R12(69)=GXY(1)*FACTOR
      TIME(69)=0.0
      INDEX=1
      DO 34 I=1,20
      K=69+INDEX
      L=69-INDEX
      R12(K)=GXY(I+1)*FACTOR
      R12(L)=GXY(NUMBER-I+1)*FACTOR
      TIME(K)=DELTAT*FLOAT(I)
      TIME(L)=-TIME(K)
   34 INDEX=INDEX+1
      DO 35 I=22,60,2
      K=69+INDEX
      L=69-INDEX
      R12(K)=GXY(I+1)*FACTOR
      R12(L)=GXY(NUMBER-I+1)*FACTOR
      TIME(K)=DELTAT*FLOAT(I)
      TIME(L)=-TIME(K)
   35 INDEX=INDEX+1
      DO 37 I=65,200,5
      K=69+INDEX
      L=69-INDEX
      R12(K)=GXY(I+1)*FACTOR
      R12(L)=GXY(NUMBER-I+1)*FACTOR
      TIME(K)=DELTAT*FLOAT(I)
      TIME(L)=-TIME(K)
   37 INDEX=INDEX+1
      WRITE(6,610)TITLE1,TITLE2
C
C          PRINT CROSS CORRELATION FUNCTION
C
      DO 39 I=1,137,2
   39 WRITE(6,611)TIME(I),R12(I),TIME(I+1),R12(I+1)
  610 FORMAT(1H1,9X,*CROSS CORRELATION COEFFICIENT*,/,10X,*CHANNEL 1 -
     1*,4A10,/,10X,*CHANNEL 2 - *,4A10)
  611 FORMAT(11X,F6.3,5X,F7.4,12X,F6.3,5X,F7.4)
C
C          PLOT CROSS CORRELATION FUNCTION
C
      CALL ROTATE(90.0)
      CALL SET(1.,8.,-7.,6.,-.4,.4,-.2,1.,1,1,0)
      CALL AXIS(0.,0.,XTIT,-40,8.0,0.0,-.4,.1,1)
      CALL AXIS(0.,0.,YTIT,40,6.,90.,-.2,.2,1)
      CALL FRSTPT(0.,-.2)
      CALL VECTOR(0.,1.)
```

```
      CALL FRSTPT(-.4,0.)
      CALL VECTOR(.4,0.)
      CALL CURVE(TIME,R12,137,0,0)
      CALL SYMBOL(0.5,5.0,.1,TITLE1,0.,40)
      CALL SYMBOL(0.5,4.8,.1,TITLE2,0.0,40)
      CALL RSTR(ICODE)
      ICODE=ICODE+1
      GO TO(100,90)ICODE
   90 ICODE=0
  100 CONTINUE
      END
```