U. S. DEPARTMENT OF THE INTERIOR

U.S. GEOLOGICAL SURVEY

FFTDC2: A One-Dimensional Fourier Transform with Forward and Inverse Data Conditioning for Non-Complex Data

by

Robert E. Bracken[1]

Open-File Report 03-228

[1]USGS, MS 964, Federal Center, Denver, Colorado 80225

This report is preliminary and has not been reviewed for conformity with U.S. Geological Survey editorial standards. Although all software released in this report have been used by the USGS, no warranty, expressed or implied, is made by the USGS as to the functioning of the software. Any use of trade, product, or firm names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

Table of Contents

Abstract

A subroutine (FFTDC2) coded in Fortran 77 is described, which performs a Fast Fourier Transform or Discrete Fourier Transform together with necessary conditioning steps of trend removal, extension, and windowing. The source code for the entire library of required subroutines is provided with the digital release of this report. But, there is only one required entry point, the subroutine call to FFTDC2; all the other subroutines are operationally transparent to the user. Complete instructions for use of FFTDC2.F (as well as for all the other subroutines) and some practical theoretical discussions are included as comments at the beginning of the source code. This subroutine is intended to be an efficient tool for the programmer in a variety of production-level signal-processing applications.

Introduction

In signal processing applications, a digital version of the one-dimensional *Fourier transform* is commonly used to convert a finite stream of evenly spaced discrete samples from the time domain to the frequency domain and back again to the time domain, usually with an intervening frequency-domain operation. To obtain a useful spectral representation in the frequency domain, the data typically require a series of conditioning steps in the time domain prior to application of the transform. Upon inverse transformation, the conditioning steps must be reversed and inverted to obtain the original time-domain function modified according to the intended results of the operation.

This report describes and supplies source code of a subroutine that performs a one-dimensional FFT (Fast Fourier Transform) or DFT (Discrete Fourier Transform), as appropriate, and a reasonable series of conditioning steps on an arbitrarily long stream of digital real data. The subroutine, called fftdc2.f, is coded in Fortran 77 and performs all of its available functions from one call statement according to the values of arguments passed to the subroutine. This subroutine is intended to be an efficient tool for the programmer in a variety of production-level signal-processing applications. Hence, there is a selection of conditioning and transformation options intended to support various signal-processing requirements.

Because sampled data commonly have no imaginary component (e.g. potential-field data are usually real-only), the subroutine assumes that the time domain is real, not complex. Therefore in the frequency domain, the real part is always an even function and the imaginary part is always an odd function. The conditioning steps performed are trend removal, extension, and windowing. A variety of options is available for each conditioning step as well as for the transform itself.

Various discussions about the time domain, the frequency domain, and the Fourier transform may be found in the subroutine's source code (appendix B). A discussion about the FFT may be found in Claerbout (1976) pages 6-12. A discussion of the DFT may be found in Bloomfield (1976) pages 46-49. An excellent in-depth discussion of the Fourier transform may be found in Bracewell (1986).

Description of FFTDC2.F

This conditioning and transform subroutine (FFTDC2.F) is written in standard Fortran 77 and automatically performs all necessary conditioning procedures before transforming from time to frequency domain. The source code for the entire library of subroutines that are necessary for running the conditioning and transform process is provided with the digital release of this report. However, the only entry-point is FFTDC2.F; all the other subroutines are operationally transparent to the user. Complete instructions for use of FFTDC2.F (as well as for all the other subroutines) and some practical theoretical discussions are included as comments at the beginning of the source code. There are one or two arrays within the tree of subroutines whose dimensions are locked by parameter statements at several hundred thousand double precision elements. If these need to be changed, it must be done in the source code and then re-compiled.

Conventions and Constraints

This subroutine is intended to distill a number of tedious programming steps into a single call statement with functional control implemented via a few arguments. To do this, various conventions are followed, and certain constraints are implemented.

Arithmetic

All non-integer storage is done in double precision real or complex variables. This implies that, on most machines, the computational noise floor will be less than -275 dB referenced to the RMS power level of the data profile.

Definition of the Forward and Inverse Transform

The transform exponent, $-i\omega$, is defined implicitly as the *forward* transform; and, the forward transform is further defined as going from *time* to *frequency* domain. Therefore the transform exponent, $+i\omega$, is the implicit *inverse* transform; and it goes from *frequency* to *time* domain.

Scale Factor

In a procedure to make the time and frequency domains mathematically equivalent, Clearbout (1976) includes a "scale factor" in the FFT. The scale factor equals $n^{-1/2}$ for either forward or inverse transform. FFTDC2 may be called optionally, "with" or "without" a scale factor. If it is called *without* the scale factor, the amplitudes in the frequency domain will be true; and a non-symmetric scaling will be used, which is 1/n for the forward transform and 1 for the inverse.

Data Requirements

This subroutine has been developed around the assumption that the data stream is generally stationary and ergodic. While the conditioning processes and the Fourier transform will work with any data stream, the assumptions of the conditioning steps may not hold, and the spectra of non-stationary functions may not be useful. For a discussion of stationarity, see Bendat (1976).

Consequently, it is advised that spikes and certain other non-stationary phenomena be removed before calling this subroutine. Because of the variety and non-invertability of these removal processes, the subroutine does not include any of them.

Time-Domain Restricted to a Real Array

Although the Fourier transform requires complex domains in both time and frequency, the time domain in this subroutine is restricted to be real only, because sampled data rarely have an imaginary component. Therefore the time-domain array is declared real, making all imaginary components implicitly equal to zero. If it becomes necessary to process complex time-domain data, subroutine modifications would not be difficult.

Frequency Domain in a Complex Array

The frequency domain array is declared complex. As a consequence of the real-only time domain restriction, the frequency domain contains an *even* real function and an *odd* imaginary function. Therefore the negative abscissa of the frequency domain maps redundant values.

In order to avoid having negative indices in the complex frequency-domain array, periodicity is used to increment the negative portion 2π to the right. As a result, a function of length n (the length is the same in both time and frequency domains), will have the zero frequency at array index number 1, and the Nyquist frequency at array index n/2+1. The "negative" frequencies will occupy index locations n/2+2 to n. If n is odd, Nyquist will exist implicitly halfway between n/2+1/2 and n/2+3/2.

Array Usage during Forward and Inverse Transform

All conditioning steps are performed on the time-domain data *in the real time-domain array*, regardless whether forward or inverse transform has been selected.

During forward transform, the time-domain data are conditioned in the real array and copied into the real component of the complex array, the imaginary component of the complex array is set to zero, and then the contents of the complex array are transformed, within the array, to the frequency domain. So, upon return from the subroutine (forward) call, the real array contains the *conditioned* pre-transformed time-domain data and the complex array contains the conditioned transformed frequency-domain data.

During inverse transform, the frequency-domain data in the complex array are first transformed back to the time-domain and the real component copied into the real array, (the imaginary component should have vanished during transform), and then the conditioning is removed from the contents of the real array. So, upon return from the subroutine (inverse) call, the complex array contains the *inverse transformed* conditioned time-domain data and the real array contains the same time-domain data only with the conditioning removed.

Descriptions of Subroutine Operations

A description of each subroutine argument is given in the source code. General descriptions, comments, and suggestions are given here.

The Sequence of Steps for Conditioning and Transform

For the forward transform (time to frequency domain) the following conditioning steps are automatically performed on the time-domain data in the real array and in the following prescribed sequence:

1) Trend removal (real array),
2) Extension (real array),
3) Windowing (real array),
4) Copy from real to complex array,
5) Forward transformation from time to frequency domain (complex array).

If particular processing requires one or more of the conditioning or transform steps out of the prescribed sequence, repeated calls to FFTDC2 can be made with appropriate combinations of the steps allowed or disallowed. This pattern can be followed in either forward or inverse transform.

For the inverse transform, the reverse sequence is followed and each step is the inverse of its forward counterpart. The inverse transform and conditioning is intended to undo exactly all steps performed during the forward transform. This in turn, is useful if a function is to be modified in the frequency domain (e.g. a filter operation) and then transformed back to the time domain:

5) Inverse transformation from frequency to time domain (complex array),
4) Copy real component from complex to real array,
3) Inverse windowing (real array),
2) Truncation (real array),
1) Trend restoration (real array).

In the subroutine call, the arguments reflect the forward-transform processing sequence. (The argument names are given in parenthesis).

0) Selection of forward or inverse	(nverse),
1) Trend removal	(ktrend, nlsq, alsq),
2) Extension	(kexten, xparms, justx),
3) Windowing	(kwindo, alpha, wfrac, ywind),
5) Transform selection and arrays	(kdxfrm, ntmd, tmd, nfqd, fqd)

Trend Removal and Restoration (ktrend)

Least-squares polynomial trends (ktrend) up to third order (nlsq) may be removed. However, for best preservation of spectral content and for making a first-cut at minimizing end effects, a linear trend (first order) is recommended. Higher order polynomials will better reduce end effects but at the expense of proper low-frequency representation. For reducing end effects, a combination of extension and windowing should be done.

The coefficients of the selected trend will be generated and output in an argument array (alsq) during forward transform. During inverse transform, the coefficients will be read back in from the argument array.

Extension and Truncation (kexten)

A key simplification of this subroutine is in the method of determining the extension/truncation lengths. In effect, the lengths are given implicitly using the numbers of samples specified for each array, the real (tmd) and the complex (fqd). In other words, the explicit length of the time-domain function (ntmd), the explicit length of frequency-domain function (nfqd), and the direction of transform (nverse) implicitly determine whether to extend or truncate and by how much. An additional argument (justx) allows the user to specify the justification (left, right, center, or explicit amount) of the shorter domain within the longer.

Normally, a function would be extended before forward transform and the extensions then truncated after inverse transform. However, if the function is truncated before forward transform, it will then be extended back to its original length after inverse transform. This entire procedure is transparent to the user provided the extension-affecting arguments have not been altered between forward and inverse calls.

As with any conditioning step in this subroutine, extension and truncation occur entirely within the *time* domain. There is, however, one special situation that occurs if the user disallows extension/truncation (kexten = -1) *and* specifies a frequency domain of a different length than the time domain. In this case, the operation moves to the *frequency* domain and the subroutine makes an extension of zeros beyond the Nyquist frequency or truncates the high-frequency end of the spectrum.

Extending with zeros (kexten = 0) in the time domain does not alter the spectrum in the frequency domain. A zero-extension's primary effect is to increase the number of points between the zero and Nyquist frequencies. For example, if the time domain function was extended to triple its length with zeros (adding a series of zeros with length equaling the function at both the beginning and end), a particular peak in its frequency domain that had spanned only one frequency bin would now span three bins. Conversely, if a spectrum is extended beyond Nyquist with zeros, its corresponding time-domain function will have a smaller sampling interval. This type of operation is used in subroutine winnow.f (Bracken, 2003).

Extension can be used in conjunction with windowing to minimize profile end effects and sharpen spectral peaks … in effect, to approach the spectrum of the random process that produced the time-domain function. The end-effect that propels the need for extension and windowing is an implied discontinuity at the beginning or end of the time-domain function, which has a magnitude equal to the difference between the value of the first point and the last point. The discontinuity is an artifact of forcing the Fourier transform, which is supposed to operate on an infinitely long function, to operate on a function of finite length. The result is that the finite-length function is assumed to repeat, as if it was laid end-to-end from minus infinity to plus infinity.

Proper windowing can reduce the size of the implied step to an arbitrarily small value, but at the expense of significantly de-weighting function values that are away from the center of the profile. In some cases, de-weighting is not a problem because the function is sufficiently ergodic, but in other cases the spectrum may be altered. For these, an appropriate extension should precede windowing. The best extension not only preserves derivatives at the beginning and end transitions but also produces the extensions by the same random process that produced the function. An in-depth discussion of random processes may be found in Bendat (1976).

Burg prediction (kexten=7 or 8) will accomplish this. For an optimal narrowing of spectral peaks and minimal introduction of noise, the extended function would normally be 2 to 4 times the length of original function (nfqd=2*ntmd to 4*ntmd and nfqd = a power of 2 to get faster FFT processing). Empirical results indicate that a good "root" length should be $7/10^{th}$ the original function length (xparms(1)= ntmd/√2). The Burg prediction method is described by Claerbout (1976).

Windowing

A variety of windows (kwindo, alpha) is available and a description of each may be found in Harris (1976). Each windowing function may, optionally, be split (wfrac). A split window is simply the specified window applied only to the ends of the time-domain function. Splitting a window is most useful when de-weighting away from the center of the original function cannot be tolerated. However, splitting a window significantly (negatively) alters the window's characteristics.

All windows are applied to the entire, extended function with *DFT-even symmetry* (Harris, 1976) except when extension with zeros is specified. In that case, only the non-extended part of the function is windowed and a *symmetric* window is applied instead.

Of the installed windows, it appears that the Hamming (kwindo=3) and the Kaiser-Bessel (kwindo=14, alpha=3.5) have the least amount of side-lobing and signal loss. The commonly used Gaussian window (kwindo=12) has not been installed. But, unless the specific characteristic of a Gaussian window (minimum time-bandwidth product) is required, the Kaiser-Bessel is in many respects a better choice.

The generalized uncertainty principal guarantees a trade-off between the width of the transform-pulse main lobe and levels of the sidelobes. Most of the window functions are parameterized, using argument alpha to adjust this trade-off. Generally, a larger alpha means a wider transform-pulse main lobe and smaller sidelobe levels. Detailed discussions of these window characteristics can be found in Harris (1976).

The Transforms

This subroutine automatically selects from among three subroutines to obtain the optimal transform. If the FFT can be used, forkdc.f will be selected. If the DFT will be used, dfttdc.f will be selected. If the DFT is required to have zero-extension or truncation in the frequency domain, dftfdr.f will be selected.

If the frequency-domain length (nfqd) is a power of two (4, 8, 16, …), the FFT can be used, otherwise a DFT must be used. The DFT is faster than the FFT for profiles of less than 64 points. However, at 4096 points the DFT is *67 times slower* than the FFT. This is because the DFT is an n*n algorithm but the FFT is n*log(n).

Handling of Array Dimensions

Because there are several arrays in the argument list, array dimensions are not checked. The additional arguments required for checking array limits make the argument list cumbersome. Therefore, the calling routine must ensure that all arrays have been dimensioned large enough to handle their expected uses. Details of required dimensions are given in the comments provided at the beginning of FFTDC2.F.

An additional argument-simplifying step is to lock certain large arrays inside their subroutines. These arrays have been dimensioned to several hundred thousand elements. The two problems that can result are 1) exceeding the computer memory, or 2) array requirements exceeding built-in dimensions. The first is generally manifested by a system error during compile or first execution. The second is checked by the respective subroutines during execution. Execution will be halted if an array overrun becomes imminent.

Contemplated Modifications to this Subroutine

It is possible to conceive of a number of changes that could be made to this subroutine. First, various contemplated extension and windowing functions have not been installed. These could be easily coded and installed on an *as-need* basis. Additionally, as has been suggested earlier, the *real* time-domain array could be changed to *complex* in order to accommodate the possibility of a complex time-domain data stream.

Under rare circumstances, a problem can result from the few large arrays buried in the library that cannot be accessed by the calling routine; modifications could be advisable. The simplest solution is to put those arrays into the argument list of all calling subroutines. Although this can produce a cumbersome set of arguments, the author believes it to be programmatically more tractable than the use of common blocks.

One of the great difficulties of the DFT is that for long profiles, it is incredibly slow. The best solution is simply to extend the time-domain profile to the next power-of-two length and use the FFT. However, extension can, itself, be a slow process; and making an already long profile even longer may approach the absurd. Another, less desirable solution is to truncate the time-domain profile to a tractable length.

If truncation is not advisable because of a need to preserve low frequencies, a third solution may be to winnow the extremely long profile, thus truncating the high frequencies. Therefore, a winnowing conditioning step could be included. In order to maintain invertability, an "unwinnowing" procedure should also be included. A subroutine that does both of these is available (Bracken, 2003), and could be easily incorporated into FFTDC2 with only a few additional arguments in the call statement.

Another approach to the long-profile dilemma is to break it up into pieces and transform each piece individually. This is a step in the process of *stacking*. Therefore, it may be reasonable to consider incorporating a "stacking algorithm" in FFTDC2. However, so many stacking and/or splicing scenarios are possible that it may be better to do this process outside and use FFTDC2 as a tool.

References Cited

Bendat, Julius S., Piersol, Allan G., 1986, Random Data Analysis and Measurement Procedures, Second Edition (Revised and Expanded): John Wiley & Sons, New York.

Bloomfield, Peter, 1976, Fourier Analysis of Time Series: An Introduction: John Wiley & Sons. 258 p.

Bracewell, Ronald N., 1986, The Fourier Transform and its Applications: McGraw-Hill. 474 p.

Bracken, Robert E., 2003, The Inverse of Winnowing: a Fortran Subroutine and Discussion of Unwinnowing Discrete Data: U.S. Geological Survey open file report 03-229, Link to Digital File

Claerbout, Jon F., 1976, Fundamentals of Geophysical Data Processing: McGraw-Hill (out of print). Currently: http://sepwww.stanford.edu/sep/prof/

Harris, Fredric J., 1976, Windows, Harmonic Analysis, and the Discrete Fourier Transform: Naval Undersea Surveillance Department, San Diego, CA

APPENDIX A

Brief Descriptions of Required Subroutines

bold - Indicates required subroutines that are included in subsequent appendixes.

unbold – Indicates required subroutines that are included among the digital files of this report but not in the appendixes.

Entry Level Subroutine

fftdc2.f – This is the entry point subroutine; the only one that must be called. All others are transparent. Instructions for use, including argument descriptions, are given in the source code.

2nd Level Subroutines

dftfdr.f – Produces a DFT (Discrete Fourier Transform) with the assumption of a real time domain (vanishing imaginary components) and complex frequency domain. If domains differ in length, truncation or zero-extension is made in the *frequency* domain

dfttdc.f – Produces a DFT (Discrete Fourier Transform) with a shared complex array for both time and frequency domains. If domains differ in length, truncation or zero-extension is made in the *time* domain

extend.f – Produces extensions of various types, including Burg Spectral Estimation.

forkdc.f – Produces an FFT (Fast Fourier Transform) with a shared complex array for both time and frequency domains. Differing-length domains are not an option

trendr.f – Calculates, removes, or adds polynomial trends of various orders, including linear using lsqply.f

window.f – Applies or Removes windowing functions of various types including Kaiser and Riesz

3rd Level Subroutines

burgdp.f – Finds the coefficients for Burg Spectral Estimation

errfor.f – Produces a fatal error to stop processing for subroutine-determined non-system errors

lsqply.f - Calculates, removes, or adds polynomial trends of various orders, including linear.

fejer.f – Applies or removes a Triangular window

hamm3.f – Applies or removes a Hamming window

hann3.f – Applies or removes a Hanna window

kais3.f – Applies or removes a Kaiser window

riesz.f – Applies or removes a Riesz window

tukey.f – Applies or removes a Cosine tapered window

4th Level Subroutines

varfqd.f – Finds the variance of a profile of data about a line

zmbes5.f – Zero-order modified Bessel function of the first kind.

APPENDIX B

Subroutine FFTDC2.F

```
C
C_____________________________________________________________________
C
C     SUBROUTINE  F F T D C 2
C_____________________________________________________________________
C
C SUBROUTINE FFTDC2 PERFORMS FORWARD AND INVERSE FOURIER
C TRANSFORMS (FT), WINDOWS, EXTENDS, AND REMOVES TRENDS IN A
C DOUBLE PRECISION, 1-DIMENSIONAL ARRAY.
C
C                     RECOMMENDED PROCEDURE
C
C THE BEST SEPARATION OF SPECTRAL PEAKS, AND DISCERNMENT OF
C SPECTRAL CONTENT CAN BE OBTAINED BY THE FOLLOWING PROCEDURE.
C (OTHER COMBINATIONS OF TREND REMOVAL, EXTENSION, AND WINDOWING
C MAY OPTIMIZE OTHER USEFUL CHARACTERISTICS, DEPENDING ON THE
C APPLICATION):
C
C    1) REMOVE THE AVERAGE OR LEAST SQUARES LINE
C          KTREND=1, NLSQ=1 OR 2
C    2) EXTEND THE FUNCTION IN BOTH DIRECTIONS 2 TO 4 TIMES IT'S
C       ORIGINAL LENGTH USING BURG PREDICTION
C          KEXTEN=7, XPARMS=0, JUSTX=0
C          NFQD = 2*NTMD TO 4*NTMD (NFQD = 2 ** INTEGER)
C    3) WINDOW THE FUNCTION WITH THE KAISER-BESSEL WINDOW
C          KWINDO=14, ALPHA=3 OR 3.5
C    4) PERFORM AN FFT
C          KDXFRM=0
C
C
C                           DESCRIPTION
C
C THE PURPOSE OF THIS SUBROUTINE IS TO PERFORM A FOURIER
C TRANSFORM AND PROVIDE ALL REASONABLE DATA-PREPARATION STEPS IN
C A SINGLE PACKAGE.  ALTHOUGH THE FOURIER TRANSFORM IS A GENERAL
C ABSTRACT MATHEMATICAL OPERATION, THIS SUBROUTINE PLACES ON IT A
C FEW PRACTICAL CONSTRAINTS:  1) -IW IS DEFINED AS FORWARD
C TRANSFORM FROM THE TIME-DOMAIN TO THE FREQUENCY-DOMAIN (+IW IS
C INVERSE);  2) THE TIME DOMAIN IS CONSIDERED TO BE REAL AND THE
C FREQUENCY DOMAIN COMPLEX; 3) ALL DATA PREPARATION OCCURS IN THE
C TIME-DOMAIN BEFORE FORWARD TRANSFORM OR AFTER INVERSE
C TRANSFORM.
C
C IF FORWARD TRANSFORM IS SELECTED:  1) A LEAST SQUARES TREND
C WILL BE REMOVED; 2) THE PROFILE WILL THEN BE EXTENDED OR
C TRUNCATED; 3) IT WILL BE WINDOWED; AND FINALLY 4) IT WILL BE
C TRANSFORMED FROM THE TIME DOMAIN TO THE FREQUENCY DOMAIN VIA AN
C FFT OR DFT.  IF INVERSE IS SELECTED, THE INVERSE OF THE ABOVE
C STEPS WILL BE TAKEN IN THE REVERSE SEQUENCE.  THE ORDER OF
C ARGUMENTS ROUGHLY REFLECTS THE ORDER OF STEPS TAKEN UPON
C FORWARD TRANSFORM.
C
C IF THE FAST FOURIER TRANSFORM (FFT:  TIME REQUIRED IS
C PROPORTIONAL TO N*LOG(N)/LOG(2)) IS DESIRED, THE
C FREQUENCY-DOMAIN ARRAY IS RESTRICTED IN LENGTH TO POWERS OF
C TWO.  OTHERWISE, THE DISCRETE FOURIER TRANSFORM (DFT:  TIME
C REQUIRED IS PROPORTIONAL TO N*N) WILL BE USED.  SPIKES AND
```

```
C DVALS (NO-DATA VALUES) MUST BE REMOVED BEFORE CALLING FFTDC2.
C
C THE FOURIER TRANSFORM SUBROUTINES (FORKDC AND DFTTDC) CALLED BY
C THIS ROUTINE BOTH INCLUDE A SCALING FACTOR EQUAL TO 1/SQRT(N).
C (WHERE N IS THE TOTAL NUMBER OF POINTS IN THE FUNCTION).  THE
C FOURIER SUMS IN EITHER DIRECTION (-IW OR +IW) ARE MULTIPLIED BY
C THIS SCALING FACTOR.  THE PURPOSE IS TO MAKE THE FORWARD AND
C INVERSE TRANSFORMS MATHEMATICALLY INDISTIQUISHABLE (EXCEPT FOR
C PHASE) AS DESCRIBED IN CLAERBOUT.  THE FORWARD AND INVERSE
C FUNCTIONS ARE ILLUSTRATED AS FOLLOWS:
C
C    F.D. = (1/SQRT(N))*FSUM(-IW)  T.D. = (1/SQRT(N))*FSUM(+IW)
C
C IN CONTRAST, THE TRUE DISCRETE FOURIER TRANSFORM APPEARS AS:
C
C    F.D. =       (1/N)*FSUM(-IW)  T.D. =           1*FSUM(+IW)
C
C THEREFORE, IF ACTUAL FREQUENCY DOMAIN AMPLITUDES ARE REQUIRED
C AFTER TRANSFORMING WITH THIS ROUTINE, ALL VALUES IN FQD (REAL
C AND IMAGINARY) MUST BE MULTIPLIED BY 1/SQRT(NFQD).  IF HOWEVER
C A FORWARD TRANSFORM IS TO BE FOLLOWED BY AN INVERSE TRANSFORM,
C THE SCALING FACTORS WILL PROPERLY MULTIPLY TO 1/N AFTER BOTH
C TRANSFORMS, LEAVING THE AMPLITUDES IN THE FINAL TIME DOMAIN
C UNCHANGED.  (NOTE THAT IF THE DFT IS USED WITH DISPARATE
C NUMBERS OF INPUT AND OUTPUT POINTS, THE SCALING FACTORS WILL
C NOT BEHAVE IN A MEANINGFUL WAY).
C
C
C
C THE TIME DOMAIN
C
C IN THE CONTEXT OF A DISCRETE FOURIER TRANSFORM, THE TIME DOMAIN
C IS AN ARRAY OF EVENLY-SPACED REAL (IMPOSED BY THIS SUBROUTINE)
C VALUES COMPRISING A 1-DIMENSIONAL FUNCTION OF TIME.  IN
C FORTRAN, ARRAYS BEGIN WITH THE FIRST ELEMENT NUMBERED 1 (NOT
C 0).  THEREFORE, THE NUMBER OF POINTS IN THE ARRAY EQUALS THE
C DIMENSION OF THE ARRAY.  HOWEVER, IN THIS DESCRIPTION, IT WILL
C BE ASSUMED THAT ARRAYS OF N ELEMENTS BEGIN WITH ELEMENT NUMBER
C 0 (NOT 1) AND EXTEND TO ELEMENT NUMBER M = N-1.  THE TIMES OF
C THE POINTS ARE IMPLIED TO EXTEND FROM T0 TO TM:
C
C     ARRAY OF FUNCTION VALUES X(T) = X0, X1, X2, ..., XM
C              TIME              T  = T0, T1, T2, ..., TM
C
C BECAUSE THE FOURIER TRANSFORM IS BASED ON PERIODIC FUNCTIONS,
C IT ASSUMES THE INTERVAL OF DATA [X0...XM] TO REPEAT:
C
C  FUNC VALS ...   XM, X0, X1, X2, ..., XM,  X0,   X1,   X2,  ...
C    TIME    ..., T-1, T0, T1, T2, ..., TM, TM+1, TM+2, TM+3, ...
C
C THEREFORE, AS FAR AS THE FOURIER TRANSFORM IS CONCERNED, THE
C VALUE OF THE POINT IMMEDIATELY AFTER THE LAST POINT IS KNOWN
C IMPLICITLY TO EQUAL THE VALUE OF THE FIRST POINT IN THE TIME
C SERIES.  (THERE IS OFTEN A LARGE STEP BETWEEN THESE POINTS
C WHICH WOULD NOT REALLY EXIST IN THE TRUE EXTENDED FUNCTION.
C HENCE, VARIOUS DATA PREPARATIONS ARE NEEDED TO CALM DOWN THE
C IMPLIED DISCONTINUITY).
```

```
C
C THE IMPLIED POINT MUST BE INCLUDED WHENEVER SYMMETRIES ARE
C CONSIDERED.  FOR EXAMPLE, THE CENTER TIME IS NOT TM/2 BUT
C RATHER (TM+1)/2.  THIS LEADS TO THE CONCEPT OF DFT-EVEN
C SYMMETRY WHICH REQUIRES THE CENTER LOCATION IN A SERIES TO BE
C 1/2 OF AN INTERVAL TO THE RIGHT OF THE ACTUAL CENTER; IN OTHER
C WORDS, THE END-POINT OF THE SERIES IS NOT M BUT RATHER M+1.
C (IF THE THE ACTUAL DIMENSION IS N, THE IMPLIED DFT-EVEN
C DIMENSION IS N+1.)
C
C THE DFT-EVEN CONCEPT IS USED HERE FOR TIME-DOMAIN WINDOWING BUT
C IS ALSO HELPFUL IN UNDERSTANDING THE SYMMETRIES APPEARING IN
C THE FREQUENCY DOMAIN.  BECAUSE THE FOURIER TRANSFORM DOES NOT
C DISTINGUISH BETWEEN FREQUENCY AND TIME, THE FREQUENCY DOMAIN
C ALSO CONTAINS DFT-EVEN SYMMETRY.
C
C
C
C FREQUENCY DOMAIN
C
C JUST AS IN THE TIME DOMAIN, THE FREQUENCY DOMAIN IS PERIODIC.
C BECAUSE OF THE PERIODICITY, IT IS USEFUL TO THINK OF THE
C FREQUENCIES IN THE FREQUENCY DOMAIN AS ANGLES OR LOCATIONS ON A
C UNIT CIRCLE.  THE FULL RANGE OF TRANSFORMED VALUES WOULD THEN
C BE REPRESENTED AFTER ONE TURN AROUND THE CIRCLE.  IF THE ANGLE
C W REPRESENTS A GIVEN FREQUENCY, THEN W, W+2PI, W+4PI, ETC ALL
C REPRESENT THE SAME FREQUENCY BECAUSE THEY ALL ARE LOCATED AT
C THE SAME PLACE ON THE CIRCLE.
C
C TO SIMPLIFY MATTERS, THE DOMAIN CAN BE CONFINED TO ONE COMPLETE
C CIRCLE OR 2PI INTERVAL.  IT IS CONVENIENT TO PICK THE INTERVAL
C -PI < W <= +PI.  IN THIS INTERVAL THE ABSOLUTE VALUES OF
C FREQUENCIES ARE CONFINED TO 0 <= |W| <= +PI; THE MAXIMUM
C FREQUENCY IS +PI.  +PI IS ACTUALLY THE NYQUIST FREQUENCY; ANY
C FREQUENCY ABOVE NYQUIST FOLDS BACK TO BECOME A LOWER FREQUENCY.
C
C THE TRANSFORM OF A REAL (TIME-DOMAIN) FUNCTION WILL CONSIST OF
C A REAL EVEN FUNCTION AND AN IMAGINARY ODD FUNCTION; THE REAL
C EVEN FUNCTION IS MIRRORED ABOUT 0 AND THE IMAGINARY ODD
C FUNCTION IS MIRRORED AND INVERTED.  THIS IMPLIES THAT WHILE THE
C NEGATIVE FREQUENCIES (-PI TO 0) ARE IMPORTANT IN THE TRANSFORM,
C THEY DO NOT CONTAIN ANY INFORMATION THAT CANNOT BE OBTAINED
C FROM THE POSITIVE FREQUENCIES (0 TO PI).
C
C BECAUSE IT IS EASIER TO HANDLE ALL-POSITIVE ARRAY INDICIES, THE
C VALUES IN THE NEGATIVE FREQUENCIES CAN BE COPIED INTO THE
C POSITIVE INTERVAL, +PI < W < +2PI.  THIS WOULD BE THE SAME AS
C ROTATING THE INTERVAL A HALF CIRCLE (PI RADIANS) RE-DEFINING IT
C 0 <= W < 2PI.  IN THE NEW INTERVAL, NYQUIST FREQUENCY STILL
C OCCURS AT PI.  BUT BECAUSE OF FOLDING, THE EFFECTIVE FREQUENCY
C DECREASES TO ZERO IN BOTH DIRECTIONS FROM NYQUIST WITH ZERO
C FREQUENCY EFFECTIVELY OCCURING AT BOTH ZERO AND 2PI.
C
C THE DISCRETE FOURIER TRANSFORM (DFT) DOES NOT PRODUCE A
C CONTINUUM OVER THE DOMAIN; BUT RATHER, IT PRODUCES A SET OF
C VALUES AT DISCRETE FREQUENCIES EVENLY SPACED AROUND THE CIRCLE.
C FURTHERMORE, IF ALL OF THE INFORMATION IN THE TIME-DOMAIN IS TO
```

```
C BE TRANSFORMED TO THE FREQUENCY DOMAIN, IT IS NECESSARY AND
C SUFFICIENT THAT THE FREQUENCY DOMAIN HAVE EXACTLY THE SAME
C NUMBER OF POINTS AS THE TIME DOMAIN.  IN OTHER WORDS, IF THE
C TIME DOMAIN FUNCTION HAD N EVENLY-SPACED POINTS OVER THE TIME
C INTERVAL 0 <= T < TN, THE FREQUENCY DOMAIN SHOULD HAVE N
C POINTS EVENLY SPACED OVER THE FREQUENCY INTERVAL 0 <= W < 2PI.
C
C IT SHOULD BE NOTED THAT THE DFT-EVEN SYMMETRY EXISTS IN BOTH
C DOMAINS.  PARTICULARLY, THERE IS NO EXPLICIT POINT IN THE
C FREQUENCY DOMAIN AT 2PI; THE VALUE AT W=2PI IS IMPLIED EQUAL TO
C THE VALUE AT W=0.  THE LEFT-MOST POINT IS AT ZERO FREQUENCY;
C THE REST OF THE POINTS ARE EVENLY SPACED OVER THE INTERVAL 0 TO
C 2PI WITH THE POINT AT 2PI MERELY IMPLIED.
C
C THE ANGULAR FREQUENCY (NOT ACCOUNTING FOR FOLDING) OF THE KTH
C POINT IS:
C
C              WK = 2PI*K/N    K = 0, 1, 2, ..., N-1.
C
C THE TEMPORAL FREQUENCY (ACCOUNTING FOR FOLDING) IS:
C
C      FK =    WK   /(2PI*DT) =   K  /(N*DT) K=0,1,2,...N/2
C      FK = (2PI-WK)/(2PI*DT) = (N-K)/(N*DT) K=N/2+1,...N-1
C
C          WHERE DT = THE TEMPORAL SAMPLING INTERVAL
C
C IF THERE IS AN EVEN NUMBER OF POINTS IN THE FUNCTION, THE
C NYQUIST FREQUENCY WILL HAVE A CORROSPONDING POINT AT K=N/2+1.
C IF HOWEVER, THERE IS AN ODD NUMBER OF POINTS, THE NYQUIST
C FREQUENCY WILL FALL HALF WAY BETWEEN K=(N+1)/2 AND K=(N+1)/2+1
C
C
C
C                           REFERENCES
C
C A DISCUSSION OF THE FFT MAY BE FOUND IN "FUNDAMENTALS OF
C GEOPHYSICAL DATA PROCESSING" BY JON F. CLAERBOUT (ISBN
C 0-86542-305-9), CHAPTER 1, PAGES 6-12.  (THE OPERATION OF THE
C DFT MAY ALSO BE UNDERSTOOD FROM THIS DISCUSSION.)
C
C A DISCUSSION OF THE DFT MAY BE FOUND IN "FOURIER ANALYSIS OF
C TIME SERIES:  AN INTRODUCTION" BY PETER BLOOMFIELD, JOHN WILEY
C & SONS, PAGES 46-49.
C
C AN EXCELLENT IN-DEPTH DISCUSSION OF THE FOURIER TRANSFORM MAY
C BE FOUND IN "THE FOURIER TRANSFORM AND ITS APPLICATIONS", BY
C RONALD N. BRACEWELL, MCGRAW-HILL.
C
C A RATHER INTENSE DISCUSSION OF THE BURG ERROR-PREDICTION FILTER
C USED IN EXTENSION MAY BE FOUND IN CLAERBOUT CHAPTER 7-2, PAGES
C 133-137.
C
C AN IN-DEPTH DISCUSSION OF WINDOWING FUNCTIONS MAY BE FOUND IN
C "WINDOWS, HARMONIC ANALYSIS, AND THE DISCRETE FOURIER
C TRANSFORM" BY FREDRIC J. HARRIS, UNDERSEA SURVEILLANCE
C DEPARTMENT, SEPTEMBER 1976.
C
```

```
C
C
C                     DIRECTION OF TRANSFORM
C
C INPUT ARGUMENT:
C
C   NVERSE - INTEGER*4.  DIRECTION OF TRANSFORM.  NVERSE <= 0 IS
C            FORWARD TRANSFORM (DEFINED HERE AS TIME-DOMAIN TO
C            FREQUENCY DOMAIN).  NVERSE => 1 IS INVERSE TRANSFORM
C            (FREQUENCY-DOMAIN TO TIME-DOMAIN).
C
C            MOST ARGUMENTS TO THIS SUBROUTINE ARE AT ALL TIMES
C            INPUT ONLY.  ONE ARGUMENT (YWIND) IS AT ALL TIMES
C            OUTPUT ONLY.  THREE OF THE ARGUMENTS (ALSQ, TMD, AND
C            FQD) CHANGE AMONG INPUT ONLY, OUTPUT ONLY, OR
C            INPUT/OUTPUT DEPENDING UPON THE VALUE OF NVERSE.
C            THIS IS ILLUSTRATED IN THE FOLLOWING TABLE:
C
C               ARGUMENT        FORWARD        INVERSE
C                              (NVERSE<=0)    (NVERSE=>1)
C
C              (MOST ARGS)     INPUT ONLY     INPUT ONLY
C                 ALSQ         OUTPUT ONLY    INPUT ONLY
C                 YWIND        OUTPUT ONLY    OUTPUT ONLY
C                 TMD          INPUT/OUTPUT   OUTPUT ONLY
C                 FQD          OUTPUT ONLY    INPUT/OUTPUT
C
C
C
C                         TREND REMOVAL
C
C INPUT ARGUMENTS:
C
C   KTREND - INTEGER*4.  NUMBER INDICATING THE TYPE OF TREND TO
C            REMOVE.  A STAR (*) INDICATES THAT THE OPTION HAS
C            BEEN INSTALLED:
C
C               KTREND   TYPE OF TREND
C               *  0 = NO TREND REMOVAL
C               *  1 = POLYNOMIAL OF ORDER NLSQ-1
C                  2 = COSINE CURVE
C
C   NLSQ   - INTEGER*4.  NUMBER OF CONSTANTS ASSOCIATED WITH THE
C            DESIRED LEAST-SQUARES-TREND REMOVAL.  POLYNOMIAL
C            FUNCTIONS TO ORDER 9 ARE INSTALLED; HOWEVER GREATER
C            THAN ORDER 3 IS NOT RECOMMENDED.  TYPICAL VALUES ARE
C            AS FOLLOWS:
C
C                NLSQ    ORDER OF TREND
C                  0 = (NO TREND REMOVAL IF POLYNOMIAL)
C                  1 = ZEROTH ORDER (AVERAGE IF POLYNOMIAL)
C                  2 = FIRST ORDER OR (LINEAR IF POLYNOMIAL)
C                  3 = SECOND ORDER (PARABOLIC IF POLYNOMIAL)
C                  4 = THIRD ORDER
C
C OUTPUT OR INPUT ARGUMENT:
C
```

```
C   ALSQ   - REAL*8.  ARRAY OF DIMENSION NLSQ CONTAINING THE
C            COEFFICIENTS OF THE TREND.  IF FORWARD TRANSFORM,
C            ALSQ WILL BECOME AN OUTPUT ARGUMENT; IF INVERSE
C            TRANSFORM, ALSQ WILL BE AN INPUT ARGUMENT.
C
C
C
C                     EXTENSION AND TRUNCATION
C
C INPUT ARGUMENTS:
C
C   KEXTEN - INTEGER*4.  NUMBER INDICATING THE TYPE OF EXTENSION
C            THAT IS DESIRED.  IF EXTENSION WITH ZEROS IS
C            SELECTED, WINDOWING (THE DATA PREPARATION STEP
C            FOLLOWING EXTENSION) WILL BE PERFORMED ONLY OVER THE
C            ORIGINAL FUNCTION.  FOR ALL OTHER EXTENSION TYPES,
C            THE WINDOWING WILL BE PERFORMED OVER THE ENTIRE
C            EXTENDED LENGTH.  A STAR (*) INDICATES  THAT THE
C            OPTION HAS BEEN INSTALLED:
C
C             KEXTEN   TYPE OF EXTENSION
C              * -1 = EXTENSION/TRUNCATION DISALLOWED
C              *  0 = ZEROS
C              *  1 = AVERAGE VALUE
C              *  2 = FIRST/LAST VALUE
C              *  3 = REPEAT ORIGINAL FUNCTION
C              *  4 = REFLECT ORIGINAL FUNCTION
C                 5 = REFLECT AND INVERT (XPARMS NEEDED)
C                 6 = SPLINE TO ZERO (XPARMS NEEDED)
C              *  7 = BURG PREDICTION WITH A LINEAR FUNCTION
C                     REMOVED BEFORE PREDICTION AND ADDED BACK
C                     AFTER PREDICTION.  ONE OR MORE VALUES MUST
C                     BE SPECIFIED IN THE ARGUMENT, XPARMS().
C
C                     FOR KEXTEN=7, XPARMS(1) SPECIFIES THE ROOT
C                     LENGTH OF THE BURG FILTER.  THE ROOT LENGTH
C                     IS THE NUMBER OF DATA POINTS COUNTED FROM
C                     THE FIRST PREDICTED POINT BACKWARDS INTO
C                     THE EXISTING PROFILE.  THESE POINTS ARE
C                     USED TO PRODUCE THE PREDICTION-FILTER
C                     COEFFICIENTS.  FOR EXAMPLE, IF XPARMS(1)=3,
C                     AND THE TIME-DOMAIN PROFILE HAS NTMD=20
C                     POINTS, THEN POINTS 18, 19, AND 20 WILL BE
C                     USED TO PREDICT THE NEXT POINT, 21.  THEN,
C                     POINT 22 WILL BE PREDICTED FROM POINTS 19,
C                     20, AND 21; AND SO FORTH.  THIS PROCESS IS
C                     REFLECTED AT THE LEFT END OF THE PROFILE,
C                     IF EXTENSION IS OCCURRING THERE.
C
C                     BECAUSE THE ROOT LENGTH CANNOT BE LESS THAN
C                     2, VALUES OF XPARMS(1) THAT ARE LESS THAN 2
C                     ARE USED TO INDICATE SPECIFIC METHODS OF
C                     DERIVING THE ROOT LENGTH.  THESE METHODS
C                     ARE GIVEN IN THE FOLLOWING TABLE, AND SOME
C                     REQUIRE ADDITIONAL VALUES TO BE SPECIFIED
C                     IN XPARMS(2) AND XPARMS(3):
C
```

```
C                    XPARMS(1)   ROOT-LENGTH CALCULATION METHOD
C                       -3    LEFT ROOT = XPARMS(2)
C                             RIGHT ROOT = XPARMS(3)
C                       -2    LEFT ROOT =
C                                XPARMS(2)*(LEFT EXTENSION)
C                             RIGHT ROOT =
C                                XPARMS(2)*(RIGHT EXTENSION)
C                       -1    LEFT ROOT = LEFT EXTENSION
C                             RIGHT ROOT = RIGHT EXTENSION
C                        0    EACH ROOT = INPUT ARRAY LEN (NTMD)
C                        1    EACH ROOT =
C                                HALF THE TOTAL EXTENDED LENGTH
C                                WHICH IS (NFQD-NTMD)/2
C                       >1    EACH ROOT = XPARMS(1)
C
C             *  8 = BURG PREDICTION AS IN KEXTEN=7 ONLY WITHOUT
C                    THE LINEAR FUNCTION.
C
C           NOTE:  WHETHER THE PROFILE IS EXTENDED OR TRUNCATED
C           DEPENDS ON WHETHER NTMD IS LESS THAN OR GREATER
C           THAN NFQD;  THE AMOUNT OF EXTENSION IS EQUAL TO THE
C           DIFFERENCE BETWEEN NTMD AND NFQD; AND THE LOCATION
C           WITHIN THE EXTENDED PROFILE IS GOVERNED BY THE
C           JUSTIFICATION, JUSTX.
C
C           NOTE:  USE OF THE BURG PREDICTION IS AN EXCELLENT
C           WAY OF EXTENDING THE PROFILE TO ANY ARBITRARY LENGTH
C           (UP TO 4 TIMES THE ORIGINAL PROFILE) WHILE
C           PRESERVING EXACTLY THE FREQUENCY CONTENT.  UNLIKE
C           THE FOURIER TRANSFORM, BURG PREDICTION IS IMMUNE TO
C           THE REPETITION DISCONTINUITY ASSOCIATED WITH THE
C           BEGINNING AND END OF THE PROFILE.  THE RESULT IS TO
C           NARROW SPECTRAL PEAKS TO AN ARBITRARILY THIN LINE.
C           WHEN USED IN CONJUNCTION WITH THE KAISER-BESSEL
C           WINDOW, RESULTS CAN APPROACH AN IDEAL NARROWING OF
C           THE PEAK AND REMOVAL OF SIDE LOBES.
C
C           NOTE:  THE BURG PREDICTION IS AN N*N PROCEDURE AND
C           HENCE IS QUITE SLOW.  TO REDUCE THE TIME REQUIRED, A
C           SUITABLY SHORT "ROOT" LENGTH SHOULD BE CHOSEN USING
C           XPARMS.  KEEP IN MIND HOWEVER, THAT SHORTENING THE
C           "ROOT" WILL CAUSE POORER FREQUENCY REPRODUCTION IN
C           THE EXTENDED SECTION.  A RULE OF THUMB IS THAT THE
C           "ROOT" SHOULD NOT BE LESS THAN HALF OF THE AMOUNT TO
C           BE EXTENDED.
C
C   XPARMS - REAL*8.  ARRAY CONTAINING ANY PARAMETERS ASSOCIATED
C            WITH ONE OF THE KEXTEN NUMBERS.  THIS ARGUMENT WILL
C            NOT BE USED UNLESS DESCRIBED UNDER KEXTEN.
C
C   JUSTX  - INTEGER*4.  JUSTIFICATION OF THE ORIGINAL PROFILE
C            WITHIN THE EXTENDED PROFILE.  FOLLOWING ARE POSSIBLE
C            VALUES:
C
C                <-1 = EXACT LOCATION OF LAST POINT FROM RIGHT
C                 -1 = RIGHT JUSTIFY (LAST POINT AT END),
C                  0 = CENTER JUSTIFY,
```

```
C                  1 = LEFT JUSTIFY (FIRST POINT AT LOCATION 1)
C                 >1 = EXACT LOCATION OF FIRST POINT FROM LEFT
C
C
C
C                             WINDOWING
C
C INPUT ARGUMENTS:
C
C   KWINDO - INTEGER*4.  NUMBER INDICATING THE TYPE OF WINDOW.
C            EVERY WINDOW WILL BE APPLIED AS DFT-EVEN.  THAT IS,
C            THE END-POINT SYMMETRIC WITH THE BEGINNING POINT IS
C            ASSUMED TO EXIST IMPLICITLY AFTER THE LAST
C            TIME-DOMAIN ARRAY POINT.  AN EXCEPTION OCCURS WHEN
C            THE TIME-DOMAIN PROFILE HAS BEEN EXTENDED WITH
C            ZEROS; IN THIS CASE, A SYMMETRIC WINDOW IS APPLIED
C            ONLY TO THE NON-EXTENDED PART OF THE DATA.  A STAR
C            (*) INDICATES THAT THE OPTION HAS BEEN INSTALLED:
C
C               KWINDO   NAME (OTHER NAMES OR DESC)       ALPHA
C               *  0 = RECTANGLE (BOXCAR, NO WINDOWING)     -
C               *  1 = TRIANGLE  (FEJER OR BARTLET)         -
C               *  2 = COS**ALPHA (HANNING)             1.0 - 4.0
C               *  3 = HAMMING (MODIFIED HANNING)           -
C               *  4 = RIESZ (SIMPLEST CONT POLYNOM)       2.0
C                  5 = RIEMANN                              -
C                  6 = DE LA VALLE-POUSSIN (CUBIC)          -
C               *  7 = TUKEY (COSINE TAPERED)           0.0 - 1.0
C                  8 = BOHMAN                               -
C                  9 = POISSON (2-SIDED EXPONENTIAL)    2.0 - 4.0
C                 10 = HANNING-POISSON                  0.5 - 2.0
C                 11 = CAUCHY                           3.0 - 5.0
C                 12 = GAUSSIAN (MIN TIME/BANDWID PROD) 2.5 - 3.5
C                 13 = DOLPH-TCHEBYSHEV                 2.5 - 4.0
C               * 14 = KAISER-BESSEL (OVERALL BEST)     2.0 - 3.5
C                 15 = BARCILON-TEMES                   3.0 - 4.0
C
C   ALPHA  - REAL*8.  THE DEGREE OF WINDOWING FOR WINDOWS WHICH
C            ALLOW A SELECTION.  IF ALPHA=0.D0, THE BEST VALUE
C            FOR THAT WINDOW WILL BE SELECTED AUTOMATICALLY
C            (ALPHA WILL NOT BE CHANGED HOWEVER).
C   WFRAC  - REAL*8.  WINDOW FRACTION FOR SPLITTING WINDOWS.  IF
C            WFRAC IS GREATER THAN ZERO AND LESS THAN ONE, A
C            SPLIT WINDOW WILL BE INDICATED.  A SPLIT WINDOW IS
C            SIMPLY THE SPECIFIED WINDOW, KWINDO, APPLIED TO ONLY
C            THE ENDS OF THE PROFILE SUCH THAT THE PROFILE TAPERS
C            TO ZERO AT EACH END AND HAS ITS VALUES UNCHANGED IN
C            THE MIDDLE.  WFRAC SPECIFIES THE FRACTION OF THE
C            ORIGINAL PROFILE LENGTH TAKEN BY WINDOW TAPERING.
C            NOTE THAT A BOX CAR WINDOW (KWINDO=0) CANNOT BE
C            SHORTENED USING WFRAC; THAT PROCEDURE MUST BE DONE
C            WITH ZERO EXTENSION (KTYPE=0 IMPLIES ZERO EXTENSION,
C            OR USE KTYPE=4, RIESZ WINDOW, AND ALPHA NEARLY 0).
C
C OUTPUT ARGUMENT:
C
C   YWIND  - REAL*8.  ARRAY OF DIMENSION NTMD CONTAINING THE
```

```
C            WINDOWING VALUES NORMALIZED TO 1 AT THE MAXIMUM.
C            YWIND IS INTENDED TO SERVE AS A WEIGHTING FUNCTION
C            FOR RESTORING CONTIGUOUS DATA THAT WAS STACKED AND
C            FILTERED USING OVERLAPPING WINDOWING.  (IF EXTENSION
C            IS BEING PERFORMED AND NFQD IS GREATER THAN NTMD AND
C            A WINDOW IS BEING APPLIED OR UNAPPLIED, THEN
C            WORKSPACE MUST BE SUPPLIED IN YWIND FROM LOCATION
C            NTMD+1 THROUGH NFQD.  THEREFORE, ALTHOUGH THE USEFUL
C            INFORMATION IN YWIND IS ALWAYS IN ELEMENTS 1-NTMD,
C            YWIND SHOULD BE DIMENSIONED THE GREATER OF NTMD AND
C            NFQD.)
C
C
C
C                        TRANSFORM AND DATA
C
C INPUT ARGUMENTS:
C
C   KDXFRM - INTEGER*4.  THE TYPE OF TRANSFORM DESIRED.
C
C               KDXFRM    DESCRIPTION
C                 -1      FOURIER TRANSFORM NOT PERFORMED
C                  0      FFT WITH SCALE FACTOR (IF POSSIBLE)
C                  1      DFT WITH SCALE FACTOR
C                  2      FFT WITHOUT SCALE FACTOR (IF POSSIBLE)
C                  3      DFT WITHOUT SCALE FACTOR
C
C            IF KDXFRM IS NEGATIVE, THE TRANSFORM STEP WILL NOT
C            BE PERFORMED.  INSTEAD, THE TIME-DOMAIN DATA (AFTER
C            TREND REMOVAL, EXTENSION, AND WINDOWING) WILL BE
C            COPIED INTO THE REAL PART OF FQD.  THE IMAGINARY
C            PART WILL BE SET TO ZERO.  THIS IS EXACTLY AS IT
C            WOULD HAVE APPEARED AT THE INPUT TO A FOURIER
C            TRANSFORM ROUTINE.  IF NVERSE IS GREATER THAN ZERO,
C            THE REAL PART OF FQD WILL BE COPIED DIRECTLY INTO
C            TMD AND THE IMAGINARY PART DROPPED.  (THEN
C            UNWINDOWING, TRUNCATION, AND TREND ADD-BACK WILL BE
C            PERFORMED.)
C
C            IF KDXFRM IS EQUAL TO ZERO, THE FFT WILL BE USED
C            WHEN POSSIBLE (NFQD MUST BE A POWER OF 2); WHEN NOT
C            POSSIBLE, THE DFT WILL BE SUBSTITUTED AUTOMATICALLY.
C
C            IF KDXFRM IS GREATER THAN ZERO, THE DFT WILL BE
C            USED.  THIS DFT IS DESIGNED TO HAVE THE SAME SCALING
C            FACTOR AS FFT.  THEREFORE WHEN EITHER FFT OR DFT CAN
C            BE USED THEIR RESULTS ARE INDISTINGUISHABLE.  NOTE,
C            HOWEVER THAT THE DFT IS CONSIDERABLY SLOWER THAN THE
C            FFT TAKING SEVERAL SECONDS FOR PROFILES AROUND 1000
C            POINTS AND INCREASING BY THE SQUARE OF N.
C
C            IF THE DFT IS SELECTED, IT IS POSSIBLE TO HAVE
C            DIFFERENT LENGTH TIME AND FREQUENCY DOMAINS
C            (OCCURING WHEN EXTENSION/TRUNCATION HAS BEEN
C            DISALLOWED AND NTMD IS NOT EQUAL TO NFQD).  THIS
C            OPTION MAY HOWEVER BE OF LIMITED UTILITY BECAUSE THE
C            TRANSFORM IS RENDERED INCOMPLETE BY THE LENGTH
```

```
C            DISPARITY.
C
C   NTMD   - INTEGER*4.  THE NUMBER OF POINTS IN THE TIME-DOMAIN
C            ARRAY, TMD.  NTMD MAY BE ANY REASONABLE NUMBER AND
C            IS NOT RESTRICTED TO POWERS OF 2.  (IF THE FFT IS
C            DESIRED FOR EITHER FORWARD OR INVERSE, EXTENSION
C            MUST BE ENABLED AND NFQD MUST BE A POWER OF 2).
C
C            NOTE:  WHETHER THE PROFILE IS EXTENDED OR TRUNCATED
C            DEPENDS ON WHETHER NTMD IS LESS THAN OR GREATER
C            THAN NFQD;  THE AMOUNT OF EXTENSION IS EQUAL TO THE
C            DIFFERENCE BETWEEN NTMD AND NFQD; AND THE LOCATION
C            WITHIN THE EXTENDED PROFILE IS GOVERNED BY THE
C            JUSTIFICATION, JUSTX.
C
C INPUT/OUTPUT OR OUTPUT ARGUMENT:
C
C   TMD    - REAL*8.  SINGLE DIMENSION ARRAY CONTAINING THE
C            TIME-DOMAIN FUNCTION.  TMD MUST BE LARGE ENOUGH TO
C            ACCOMODATE THE GREATER OF NTMD AND NFQD.
C
C            IF FORWARD TRANSFORM, AT THE TIME OF CALLING TMD
C            MUST CONTAIN A TIME-DOMAIN FUNCTION OF LENGTH NTMD.
C            THE CONTENTS OF THE ARRAY WILL BE MODIFIED DURING
C            PROCESSING TO INCLUDE:  TREND REMOVAL, EXTENSION (OR
C            TRUNCATION), AND WINDOWING.  THEREFORE, TMD CAN BE
C            VIEWED AS AN INPUT/OUTPUT ARRAY.
C
C            IF INVERSE TRANSFORM, TMD WILL BE AN OUTPUT-ONLY
C            ARRAY CONTAINING THE FINAL TIME-DOMAIN FUNCTION OF
C            LENGTH NTMD.  HOWEVER, DURING PROCESSING IT WILL BE
C            USED AS A WORK ARRAY FOR UNWINDOWING, TRUNCATION (OR
C            EXTENSION), AND TREND ADD-BACK.
C
C
C INPUT ARGUMENT:
C
C   NFQD   - INTEGER*4.  THE NUMBER OF POINTS IN THE
C            FREQUENCY-DOMAIN ARRAY, FQD.  NFQD MAY BE ANY
C            REASONABLE NUMBER AND IS NOT RESTRICTED TO POWERS OF
C            2.  HOWEVER, NFQD MUST BE A POWER OF 2 IF THE FFT IS
C            DESIRED FOR EITHER FORWARD OR INVERSE TRANSFORM.  IF
C            EXTENSION IS DISABLED AND NFQD DOES NOT EQUAL NTMD,
C            THE DFT WILL BE USED REGARDLESS OF WHETHER NFQD IS A
C            POWER OF 2.
C
C            NORMALLY, THE FREQUENCY-DOMAIN PROFILE WOULD BE
C            EQUAL OR LONGER THAN THE TIME-DOMAIN.  HOWEVER,
C            CERTAIN OPERATIONS MAY REQUIRE IT TO BE SHORTER.
C            THIS CASE CAN BE ACCOMODATED SIMPLY BY MAKING NFQD
C            SMALLER THAN NTMD.  IF THIS OCCURS, TRUNCATION WILL
C            BE PERFORMED IN THE TIME-DOMAIN DURING FORWARD
C            TRANSFORM, AND EXTENSION DURING INVERSE TRANSFORM.
C
C            NOTE:  WHETHER THE PROFILE IS EXTENDED OR TRUNCATED
C            DEPENDS ON WHETHER NTMD IS LESS THAN OR GREATER
C            THAN NFQD;  THE AMOUNT OF EXTENSION IS EQUAL TO THE
```

```
C            DIFFERENCE BETWEEN NTMD AND NFQD; AND THE LOCATION
C            WITHIN THE EXTENDED PROFILE IS GOVERNED BY THE
C            JUSTIFICATION, JUSTX.
C
C OUTPUT OR INPUT/OUTPUT ARGUMENT:
C
C   FQD    - COMPLEX*16.  SINGLE DIMENSION ARRAY CONTAINING THE
C            FREQUENCY-DOMAIN FUNCTION.  FQD MUST BE LARGE ENOUGH
C            TO ACCOMODATE THE GREATER OF NTMD AND NFQD.
C
C            IF FORWARD TRANSFORM, FQD WILL BE AN OUTPUT-ONLY
C            ARRAY CONTAINING THE FINAL FREQUENCY-DOMAIN FUNCTION
C            OF LENGTH NFQD.
C
C            IF INVERSE TRANSFORM, AT THE TIME OF CALLING FQD
C            MUST CONTAIN A FREQUENCY-DOMAIN FUNCTION OF LENGTH
C            NFQD.  THE CONTENTS OF THE ARRAY WILL BE TRANSFORMED
C            BACK TO THE TIME DOMAIN DURING PROCESSING.
C            THEREFORE, FQD CAN BE VIEWED AS AN INPUT/OUTPUT
C            ARRAY.
C
C
C SUBROUTINE FFTDCX WRITTEN BY ROB BRACKEN, USGS, 23AUG96.
C HP-9000 SERIES 700/800 UNIX VERSION 1.0, 23AUG96.
C SUBROUTINE FFTDC2 WRITTEN BY ROB BRACKEN, USGS, 18OCT96.
C HP-9000 SERIES 700/800 UNIX VERSION 2.0, 18NOV96.
C VERSION 2.1, 20020824 ( ADDED DFTFDR CAPABILITY & ARG WFRAC ).
C
C
      subroutine fftdc2(nverse,
     &                  ktrend,nlsq,alsq,
     &                  kexten,xparms,justx,
     &                  kwindo,alpha,wfrac, ywind,
     &                  kdxfrm, ntmd,tmd, nfqd,fqd)
C
C DECLARATIONS
C
C     ARGUMENTS
C
C     FORWARD (-1)/INVERSE (+1)
      integer*4 nverse
C
C     TREND REMOVAL/ADDITION
      integer*4 ktrend,nlsq
      real*8 alsq(*)
C
C     FUNCTION EXTENSION/TRUNCATION
      integer*4 kexten
      real*8 xparms(*)
      integer*4 justx
C
C     WINDOW APPLICATION/REMOVAL
      integer*4 kwindo
      real*8 alpha,wfrac,ywind(*)
C
C     TRANSFORM-TYPE SELECTION (FFT OR DFT)
      integer*4 kdxfrm
```

```
C
C     TIME-DOMAIN FUNCTION ARRAY
      integer*4 ntmd
      real*8 tmd(*)
C
C     FREQUENCY-DOMAIN FUNCTION ARRAY
      integer*4 nfqd
      complex*16 fqd(*)
C
C     INTERNAL VARIABLES
C
      real*8 signi,scale
C
c      wfrac=1.d0
C
C DETERMINE WHETHER EXTENSION/TRUNCATION IS ALLOWED
C
C     DEFAULT IS EXTENSION/TUNCATION NOT ALLOWED
      jxt=0
C     ALLOWED IF EXTENSION IS NOT BARRED & NTMD DIFFERS FROM NFQD
      if(kexten.ge.0.and.ntmd.ne.nfqd) jxt=1
C
C DETERMINE IF SYMMETRIC WINDOWING IN MIDDLE OF PROFILE IS NEEDED
C
C     DEFAULT IS DFT-EVEN WINDOWING OVER ENTIRE LENGTH OF PROFILE
      jw0=0
C     SYMMETRIC WINDOWING IF EXTENSION WITH ZEROS AND NTMD < NFQD
      if(jxt.eq.1.and.kexten.eq.0.and.ntmd.lt.nfqd) jw0=1
C
C DETERMINE WHETHER TO USE FFT OR DFT
C
C     FFT IS DEFAULT
      jdx=0
C     DFT IS NEEDED IF EXT NOT ALLOWED & NTMD DIFFERS FROM NFQD
      if(jxt.eq.0.and.ntmd.ne.nfqd) jdx=2

C     DFT IS NEEDED IF NFQD IS NOT A POWER OF 2
      nchk=1
  205 nchk=nchk*2
      if(nchk.lt.nfqd) goto 205
      if(nchk.gt.nfqd.and.jdx.eq.0) jdx=1
C     DFT IF SPECIFICALLY REQUESTED
      if((kdxfrm.eq.1.or.kdxfrm.eq.3).and.jdx.eq.0) jdx=1
C     NO TRANSFORM IF SPECIFICALLY REQUESTED
      if(kdxfrm.le.-1) jdx=-1
C
C FIND BEG LOCATION OF INPUT PROFILE WITHIN EXTENDED PROFILE
C
      inloc=justx
      if(justx.lt.0) inloc=nfqd-ntmd+justx+2
      if(justx.eq.0) inloc=(nfqd-ntmd)/2+1
C
C INITIALIZE VALUES IN NORMALIZED WINDOW FUNCTION
C
      do 704 i=1,ntmd
        ywind(i)=1.d0
  704 continue
```

```
C
C CHECK DIRECTION OF TRANSFORM
C
      if(nverse.ge.1) goto 101
C
C FORWARD TRANSFORM
C
C     REMOVE TREND
      if(ktrend.ge.1)
     & call trendr(ktrend,nlsq,alsq, nverse,ntmd,tmd)
C
C     EXTEND
      if(jxt.eq.0) then
        npts=ntmd
      else
        call extend(kexten,xparms,justx,ntmd,nfqd,tmd)
        npts=nfqd
      endif
C
C     APPLY WINDOW
      if(kwindo.ge.1) then
        if(jw0.eq.0) then
C
C         WINDOW ENTIRE PROFILE
          kdftev=1
          call window(kwindo,alpha,wfrac,kdftev, nverse,npts,tmd)
C
C         SET UP THE NORMALIZED WINDOW FUNCTION
          do 706 i=ntmd+1,npts
            ywind(i)=1.d0
  706     continue
          call window(kwindo,alpha,wfrac,kdftev,
     &     nverse,npts,ywind)
          ixwind=0
          call extend(ixwind,xparms,justx,npts,ntmd,ywind)
        else
C
C         WINDOW NON-ZERO PORTION OF THE EXTENDED PROFILE
          kdftev=0
          call window(kwindo,alpha,wfrac,kdftev,
     &     nverse,ntmd,tmd(inloc))
C
C         SET UP THE NORMALIZED WINDOW FUNCTION
          call window(kwindo,alpha,wfrac,kdftev,
     &     nverse,ntmd,ywind)
        endif
      endif
C
C     FORWARD TRANSFORM
      signi=-1.d0
      do i=1,npts
        fqd(i)=dcmplx(tmd(i),0.d0)
      enddo
      if(jdx.eq.0) then
C
C       USE FFT (SCALE FACTOR INCLUDED IN FORKDC)
        call forkdc(npts,fqd,signi)
```

```
        if(kdxfrm.eq.2) then
C
C         REMOVE SCALE FACTOR
          scale=1.d0/dsqrt(dflotj(npts))
          do i=1,npts
            fqd(i)=fqd(i)*scale
          enddo
        endif
      else if(jdx.eq.1) then
C
C       USE DFT (SCALE FACTOR NOT INCLUDED DFTTDC)
        call dfttdc(npts,nfqd,signi,fqd)
        if(kdxfrm.ne.2.and.kdxfrm.ne.3) then
C
C         ATTACH SCALE FACTOR
          scale=dsqrt(dflotj(nfqd))
          do i=1,nfqd
            fqd(i)=fqd(i)*scale
          enddo
        endif
      else if(jdx.eq.2) then
C
C       USE DFT WITH FREQUENCY DOMAIN ZERO-EXTENSION/TRUNCATION
C       (SCALE FACTOR NOT INCLUDED DFTFDR)
        call dftfdr(npts,nfqd,signi,tmd,fqd)
        if(kdxfrm.ne.2.and.kdxfrm.ne.3) then
C
C         ATTACH SCALE FACTOR
          scale=dsqrt(dflotj(nfqd))
          do i=1,nfqd
            fqd(i)=fqd(i)*scale
          enddo
        endif
      else
C
C       USE NO TRANSFORM
        if(nfqd.gt.npts) then
          do i=npts+1,nfqd
            fqd(i)=dcmplx(0.d0,0.d0)
          enddo
        endif
      endif
      goto 990
C
C INVERSE TRANSFORM
C
C     INVERSE FFT
  101 signi=1.d0
      npts=nfqd
      if(jxt.eq.0) npts=ntmd
C
      if(jdx.eq.0) then
C
C       USE FFT (SCALE FACTOR INCLUDED IN FORKDC)
        call forkdc(npts,fqd,signi)
        if(kdxfrm.eq.2) then
C
```

```
C         REMOVE SCALE FACTOR
          scale=dsqrt(dflotj(npts))
          do i=1,npts
            fqd(i)=fqd(i)*scale
          enddo
        endif
      else if(jdx.eq.1) then
C
C       USE DFT (SCALE FACTOR NOT INCLUDED IN DFTTDC)
        call dfttdc(nfqd,npts,signi,fqd)
        if(kdxfrm.ne.2.and.kdxfrm.ne.3) then
C
C         ATTACH SCALE FACTOR
          scale=1.d0/dsqrt(dflotj(nfqd))
          do i=1,npts
            fqd(i)=fqd(i)*scale
          enddo
        endif
      else if(jdx.eq.2) then
C
C       USE DFT WITH FREQUENCY DOMAIN ZERO-EXTENSION/TRUNCATION
C       (SCALE FACTOR NOT INCLUDED DFTFDR)
        call dftfdr(nfqd,npts,signi,tmd,fqd)
        do 708 i=1,npts
          fqd(i)=dcmplx(tmd(i),0.d0)
  708   continue
C
        if(kdxfrm.ne.2.and.kdxfrm.ne.3) then
C
C         ATTACH SCALE FACTOR
          scale=1.d0/dsqrt(dflotj(nfqd))
          do i=1,npts
            fqd(i)=fqd(i)*scale
          enddo
        endif
      else
C
C       USE NO TRANSFORM
        if(nfqd.lt.npts) then
          do i=nfqd+1,npts
            fqd(i)=dcmplx(0.d0,0.d0)
          enddo
        endif
      endif
      do 703 i=1,npts
        tmd(i)=dreal(fqd(i))
  703 continue
C
C     REMOVE WINDOW
      if(kwindo.ge.1) then
        if(jw0.eq.0) then
C
C         UNWINDOW ENTIRE PROFILE
          kdftev=1
          call window(kwindo,alpha,wfrac,kdftev, nverse,npts,tmd)
C
C         SET UP THE NORMALIZED WINDOW FUNCTION
```

```
          do 707 i=ntmd+1,npts
            ywind(i)=1.d0
  707     continue
          call window(kwindo,alpha,wfrac,kdftev,
     &     -nverse,npts,ywind)
          ixwind=0
          call extend(ixwind,xparms,justx,npts,ntmd,ywind)
        else
C
C         UNWINDOW NON-ZERO PORTION OF THE EXTENDED PROFILE
          kdftev=0
          call window(kwindo,alpha,wfrac,kdftev,
     &     nverse,ntmd,tmd(inloc))
C
C         SET UP THE NORMALIZED WINDOW FUNCTION
          call window(kwindo,alpha,wfrac,kdftev,
     &     -nverse,ntmd,ywind)
        endif
      endif
C
C     TRUNCATE
      if(jxt.ne.0)
     &  call extend(kexten,xparms,justx,nfqd,ntmd,tmd)
C
C     ADD BACK TREND
      if(ktrend.ge.1)
     & call trendr(ktrend,nlsq,alsq, nverse,ntmd,tmd)
C
C EXIT PROCEDURE
C
  990 return
      end
```

APPENDIX C

Subroutine DFTFDR.F

```
C
C_____________________________________________________________________
C
C     SUBROUTINE  D F T F D R
C_____________________________________________________________________
C
C SUBROUTINE DFTFDR PERFORMS A DISCRETE FOURIER TRANSFORM ON ONE
C DIMENSIONAL DATA.  THE TIME DOMAIN IS DOUBLE-PRECISION REAL AND
C THE FREQUENCY DOMAIN IS DOUBLE-COMPLEX.  THE TIME AND FREQUENCY
C DOMAINS ARE ALLOWED TO DIFFER IN LENGTH.
C
C DIFFERING LENGTH DOMAINS ARE INTERPRETED BY DFTFDR AS REQUIRING
C A HIGH FREQUENCY TRUNCATION OR EXTENSION WITH ZEROS (SEE
C INTERPRETATION 2, BELOW).  UPON FORWARD AND INVERSE
C TRANSFORMING HOWEVER, RESTORATION OF THE ORIGINAL TIME-DOMAIN
C PROFILE CAN ONLY BE ASSURED IF THE FREQUENCY DOMAIN HAS THE
C SAME OR MORE POINTS THAN THE TIME DOMAIN.
C
C IF THE NUMBERS OF POINTS IN THE TIME AND FREQUENCIES DOMAINS
C ARE EQUAL, THIS SUBROUTINE PRODUCES A TRUE DFT (SIGNI=-1 FOR
C FORWARD AND SIGNI=+1 FOR INVERSE).  THE SCALE FACTORS ARE 1/NRX
C FOR FORWARD TRANSFORM AND 1 FOR INVERSE TRANSFORM.
C
C THE FOLLOWING TABLE SHOWS TIMES REQUIRED FOR VARIOUS ROUTINES
C TO PERFORM A FORWARD AND INVERSE TRANSFORM WHEN THE NUMBERS OF
C TIME AND FREQUENCY DOMAIN POINTS ARE EQUAL.  (TIMES TAKEN ON
C MUSETTE, HP-9000 SERIES 700/800 COMPUTER, ON 14MAY97.  THIS
C COMPUTER RUNS ROUGHLY 30 TIMES FASTER THAN A VAX 11/750.)
C
C    ROUTINE   TYPE      RULE      LENGTH    TIME (SEC)
C
C     FORKDC    FFT      N*LOGN     4096        0.80
C     DFTTDR    DFT      N*N        4096       31.69
C  -> DFTFDR    DFT      N*N        4096       31.69 <-
C     DFTTDC    DFT      N*N        4096       66.98
C
C    (THE DFT IS FASTER THAN THE FFT FOR N < 64)
C    (THE BREAK-EVEN POINT IS ACTUALLY N = 48 BUT THE FFT
C     CANNOT ACCEPT A 48 POINT PROFILE)
C
C
C
C
C A DFT CAN INTERPRET DIFFERING LENGTH DOMAINS IN TWO WAYS:  1)
C THE BEGINNING AND END OF THE TIME-DOMAIN ARE TO BE TRUNCATED OR
C EXTENDED WITH ZEROS, 2) THE HIGH FREQUENCIES IN THE FREQUENCY
C DOMAIN ARE TO BE TRUNCATED OR EXTENDED WITH ZEROS
C
C 1) TIME-DOMAIN TRUNCATION/EXTENSION IS THE SAME AS CHANGING THE
C DENSITY OF FREQUENCIES WITHIN THE FREQUENCY DOMAIN.  THE
C PROCEDURE IS DONE BY SUBROUTINE DFTTDR.  BY EXTENDING THE TIME
C DOMAIN WITH ZEROS, THE NUMBER OF POINTS IN THE FREQUENCY DOMAIN
C INCREASES BUT THE OVERALL SHAPE OF THE FREQUENCY-DOMAIN PROFILE
C REMAINS THE SAME.  SIMILARLY WITH TRUNCATION.  (NOTE THAT
C TRUNCATION OF THE TIME DOMAIN IS SO SIMPLE THAT IT SHOULD BE
C DONE IN THE SUBROUTINE CALL.  IF IN THE FORWARD CALL THE
C TIME-DOMAIN HAS MORE POINTS THAN THE FREQUENCY DOMAIN, DFTTDR
```

```
C WILL PRODUCE A DEGENERATE FREQUENCY DOMAIN RATHER THAN THE
C FREQUENCY DOMAIN OF A TRUNCATED TIME DOMAIN PROFILE.)
C
C 2) FREQUENCY-DOMAIN TRUNCATION/EXTENSION IS THE SAME AS
C CHANGING THE DENSITY OF SAMPLES WITHIN THE TIME DOMAIN.  THIS
C PROCEDURE IS DONE BY THIS SUBROUTINE, DFTFDR.  BY EXTENDING THE
C HIGH FREQUENCIES IN THE FREQUENCY DOMAIN WITH ZEROS, THE NUMBER
C OF POINTS IN THE TIME DOMAIN (AFTER INVERSE TRANSFORM)
C INCREASES BUT THE OVERALL SHAPE OF THE TIME-DOMAIN PROFILE
C REMAINS THE SAME.  SIMILARLY WITH TRUNCATION.  (TRUNCATION OF
C THE FREQUENCY DOMAIN IS NOT AS SIMPLE AS IN THE TIME DOMAIN;
C THEREFORE DFTFDR DOES NOT ALLOW THE POSSIBILITY OF A DEGENERATE
C TIME DOMAIN.  RATHER, IF ON INVERSE TRANSFORM THE NUMBER OF
C TIME-DOMAIN POINTS IS LESS THAN THE NUMBER OF FREQUENCY-DOMAIN
C POINTS, DFTFDR PRODUCES THE FREQUENCY TRUNCATION INTERNALLY.)
C
C
C
C
C DFTFDR PROVIDES THE OPTION OF PRODUCING A NEW TIME DOMAIN WHICH
C HAS BEEN ACCORDIONED-IN OR -OUT TO HAVE FEWER OR MORE POINTS
C THAN THE ORIGINAL TIME DOMAIN.  THIS IS DONE IMPLICITLY BY
C TRUNCATING OR EXTENDING THE HIGH FREQUENCIES IN THE FREQUENCY
C DOMAIN.  THE IMPLICIT TRUNCATION/EXTENSION IS FASTER THAN USING
C A STANDARD DFT TO TRANSFORM INTO THE FREQUENCY DOMAIN AND THEN
C MANUALLY TRUNCATE OR EXTEND THE FREQUENCIES.  TO ACCORDION A
C TIME DOMAIN, THE RECOMMENDED PROCEDURE IS TO FORWARD TRANSFORM
C THE ORIGINAL TIME DOMAIN INTO A FREQUENCY DOMAIN WITH EQUAL OR
C FEWER POINTS; THEN INVERSE TRANSFORM INTO THE NEW TIME DOMAIN
C WITH EQUAL OR MORE POINTS THAN THE FREQUENCY DOMAIN.  THE
C FREQUENCY DOMAIN PRODUCED BY DFTFDR IS NOT PARTICULARLY
C INTERESTING BECAUSE IT SIMPLY HAS THE HIGH FREQUENCIES
C TRUNCATED OR EXTENDED WITH ZEROS.
C
C THE FOLLOWING TABLE GIVES THE EFFECTS DFTFDR HAS ON VARIOUS
C EXAMPLE-LENGTH TIME AND FREQUENCY DOMAIN PROFILES:
C
C  TD1 -1 FD +1 TD2             DESCRIPTION
C  NRX    NCY   NRX2
C
C  100                 100 PT ORIGINAL TIME DOMAIN (TD1)
C
C  100    100          100 PT FREQ DOMAIN (FD) NORMAL DFT-STYLE
C  100    100   100    100 PT NEW TIME DOM (TD2) IDENTICAL TO TD1
C  100    100   200    TD2 = TD1 ACCORDIONED TO 200 PTS (FASTER)
C
C  100    200          100 PT FD WITH 100 ADDITIONAL ZERO HF PTS
C  100    200   100    TD2 = TD1
C  100    200   200    TD2 = TD1 ACCORDIONED TO 200 PTS (SLOWER)
C
C  200                 200 PT TD1
C
C  200    100          200 PT FD WITH 100 HF POINTS TRUNCATED
C  200    100   100    TD2 = TD1 ACCORDIONED TO 100 PTS (FASTER)
C  200    100   200    TD2 = TD1 WITH HIGH FREQUENCIES MISSING
C
C  200    200          200 PT FD NORMAL DFT-STYLE
```

```
C  200    200   100    TD2 = TD1 ACCORDIONED TO 100 PTS (SLOWER)
C  200    200   200    TD2 = TD1
C
C
C
C
C INPUT ARGUMENTS:
C   NRX    - INTEGER*4.  NUMBER OF TIME-DOMAIN PTS IN ARRAY RX.
C   NCY    - INTEGER*4.  NUMBER OF FREQUENCY-DOMAIN PTS IN CY.
C   SIGNI  - REAL*8.  DIRECTION OF TRANSFORM:
C               -1.D0 = FORWARD TRANSFORM (TIME TO FQ) (E^-IW).
C               +1.D0 = INVERSE TRANSFORM (FQ TO TIME) (E^+IW).
C
C INPUT/OUTPUT ARGUMENT:
C   RX     - REAL*8.  ARRAY CONTAINING REAL TIME-DOMAIN DATA.  IF
C            SIGNI IS ZERO OR NEG (FORWARD TRANSFORM), RX WILL BE
C            AN INPUT ARGUMENT.  IF SIGNI IS GREATER THAN ZERO
C            (INVERSE TRANSFORM), RX WILL BE AN OUTPUT ARGUMENT.
C   CY     - COMPLEX*16.  ARRAY CONTAINING COMPLEX FREQUENCY
C            DOMAIN DATA.  IF SIGNI IS ZERO OR NEG (FORWARD
C            TRANSFORM), CY WILL BE AN OUTPUT ARGUMENT.  IF SIGNI
C            IS GREATER THAN ZERO (INVERSE TRANSFORM), CY WILL BE
C            AN INPUT ARGUMENT.
C
C SUBROUTINE DFTFDR WRITTEN BY ROB BRACKEN, USGS, 13MAY97.
C HP-9000 SERIES 700/800 UNIX VERSION 1.0, 13MAY97.
C
C
      subroutine dftfdr(nrx,ncy,signi,rx,cy)
C
C DECLARATIONS
C
C     INPUT ARGUMENTS
      integer*4 nrx,ncy
      real*8 signi
C
C     INPUT/OUTPUT ARGUMENTS
      real*8 rx(*)
      complex*16 cy(*)
C
C     INTERNAL VARIABLES AND CONSTANTS
      complex*16 ci,cw0,cwk,cwj,csum
      real*8 rsum
      real*8 pi,scale
      real*8 powmin
      parameter(pi = 3.1415 92653 58979 32384 62643)
C
C
C CALCULATE DISCRETE FOURIER TRANSFORM
C
C NOTE:  IF NCY IS ODD, NYQUIST DOES NOT EXIST IF NCY IS EVEN,
C NYQUIST EXISTS, IS REAL, AND IS DOUBLE AMPLITUDE (AS IS THE
C ZERO FREQUENCY)
C
C     DISCRETE KERNEL OF INCREMENTAL NORMALIZED ANGULAR FREQ, CW0
C     FOR FREQUENCY DOMAIN TRUNCATION/EXTENSION
      ci=dcmplx(0.d0,1.d0)
```

```
      cw0=cdexp(-ci*2.d0*pi/dflotj(nrx))
C
C     DETERMINE WHETHER TO PERFORM FORWARD OR INVERSE TRANSFORM
      if(signi.le.0.d0) then
C
C FORWARD TRANSFORM (TIME DOMAIN TO FREQUENCY DOMAIN)
C
C       USE INTRINSIC DFT SCALING FACTOR
        scale=1.d0/dflotj(nrx)
C
C       FIND THE NUMBER OF FREQUENCIES TO CALCULATE
        nfq=jmin0(nrx,ncy)/2+1
C
C       FIND AVERAGE VALUE (ZERO FREQUENCY, WAVENUMBER = 0)
        rsum=0.d0
        do j=1,nrx
          rsum=rsum+rx(j)
        enddo
        cy(1)=dcmplx(rsum*scale,0.d0)
C
C       INITIALIZE MINIMUM POWER CHECKER
c        powmin=1.1d+38
C
C       FIND NON-ZERO FREQUENCIES INCLUDING NYQUIST
        cwk=cw0
        do k=2,nfq
          csum=dcmplx(0.d0,0.d0)
          cwj=dcmplx(1.d0,0.d0)
          do j=1,nrx
            csum=csum+rx(j)*cwj
            cwj=cwj*cwk
          enddo
          cwk=cwk*cw0
C
C         MAKE ADJUSTMENTS AT AND ABOVE THE NYQUIST FREQUENCY
c          if(cdabs(csum).lt.powmin) powmin=cdabs(csum)
          if(k.eq.nfq) then
            if(ncy.gt.nrx) then
C
C             IF CSUM IS NYQUIST, SPLIT IT
              if((k-1)*2.eq.nrx) csum=csum/2.d0
C
C             FILL EXTRA FREQUENCIES WITH ZEROS
              powmin=0.d0
C             (OTHER OPTIONS ARE PICK A POWER LEVEL BELOW POWMIN
C             OR CREATE WHITE NOISE AT SOME SPECIFIED POWER)
c              powmin=powmin*scale/4096.d0
c              powmin=powmin*scale/2.d0
              do i=k+1,ncy-k+1
                cy(i)=dcmplx(powmin,0.d0)
              enddo
            else if(ncy.lt.nrx) then
C
C             IF CSUM IS NEW NYQUIST, COMBINE WITH ITS REFLECTION
              if((k-1)*2.eq.ncy) csum=csum+dconjg(csum)
            endif
          endif
```

```
C
C         SCALE & REFLECT THE CONJG OF THE NEWLY CALCULATED FREQ
          cy(k)=csum*scale
          cy(ncy-k+2)=dconjg(cy(k))
        enddo
      else
C
C INVERSE TRANSFORM (FREQUENCY DOMAIN TO TIME DOMAIN)
C
C
C       INVERT THE KERNEL
        cw0=dconjg(cw0)
C
C       INVERSE SCALING FACTOR IS 1
        scale=1.d0
C
C       FIND LOC OF FREQ BEFORE NYQ & FIND WHETHER NYQUIST EXISTS
        kbn=(jmin0(nrx,ncy)-1)/2+1
        nyq=0
        if(kbn*2.eq.jmin0(nrx,ncy)) nyq=1
C
C       FIND TIME-DOM POINTS; UPPER HALF OF FREQS CAN BE IGNORED
        cwj=dcmplx(1.d0,0.d0)
        do j=1,nrx
          rsum=0.d0
C
C         ADD IN FREQS WITH WAVENUMBERS 1 THROUGH ALMOST NYQUIST
          cwk=cwj
          do k=2,kbn
            rsum=rsum+dreal(cy(k)*cwk)
            cwk=cwk*cwj
          enddo
          rsum=2.d0*rsum
          cwj=cwj*cw0
C
C         ADD IN NYQUIST IF IT EXISTS (MODIFY IT IF NECESSARY)
          if(nyq.eq.1) then
            if(ncy.gt.nrx) then
              rsum=rsum+dreal((cy(k)+dconjg(cy(k)))*cwk)
            else
              rsum=rsum+dreal(cy(k)*cwk)
            endif
          endif
C
C         ADD IN AVERAGE
          rsum=rsum+dreal(cy(1))
C
C         SCALE THE SUM
          rx(j)=rsum*scale
        enddo
      endif
C
C EXIT PROCEDURE
C
  990 return
      end
```

APPENDIX D

Subroutine EXTEND.F

```
C
C_____________________________________________________________________
C
C     SUBROUTINE  E X T E N D
C_____________________________________________________________________
C
C SUBROUTINE EXTEND EXTENDS A FUNCTION PROFILE IN ONE OR BOTH
C DIRECTIONS USING VARIOUS OPTIONS FOR TYPE, LENGTH, AND
C DIRECTION OF EXTENSION.  IT WILL ALSO PERFORM THE INVERSE BY
C TRUNCATION.
C
C INPUT ARGUMENTS:
C   KTYPE  - INTEGER*4.  THE TYPE OF EXTENSION TO PRODUCE.
C            FOLLOWING ARE POSSIBLE VALUES.  A STAR (*) INDICATES
C            THAT THE OPTION HAS BEEN INSTALLED:
C               * 0 = ZEROS
C               * 1 = AVERAGE VALUE
C               * 2 = FIRST/LAST VALUE
C               * 3 = REPEAT
C               * 4 = REFLECT
C               * 5 = REFLECT AND INVERT (TPARMS NEEDED)
C                     TPARMS(1) IS THE VALUE ABOUT WHICH TO
C                     INVERT THE LEFT EXTENSION.  TPARMS(2) IS
C                     THE VALUE ABOUT WHICH TO INVERT THE RIGHT
C                     EXTENSION.
C                 6 = SPLINE TO ZERO (TPARMS NEEDED)
C               *  7 = BURG PREDICTION WITH A LINEAR FUNCTION
C                     REMOVED BEFORE PREDICTION AND ADDED BACK
C                     AFTER PREDICTION.  ONE OR MORE VALUES MUST
C                     BE SPECIFIED IN THE ARGUMENT, TPARMS().
C
C                     FOR KTYPE=7, TPARMS(1) SPECIFIES THE ROOT
C                     LENGTH OF THE BURG FILTER.  THE ROOT LENGTH
C                     IS THE NUMBER OF DATA POINTS COUNTED FROM
C                     THE FIRST PREDICTED POINT BACKWARDS INTO
C                     THE EXISTING PROFILE.  THESE POINTS ARE
C                     USED TO PRODUCE THE PREDICTION-FILTER
C                     COEFFICIENTS.  FOR EXAMPLE, IF TPARMS(1)=3,
C                     AND THE PROFILE TO BE EXTENDED HAS NIN=20
C                     POINTS, THEN POINTS 18, 19, AND 20 WILL BE
C                     USED TO PREDICT THE NEXT POINT, 21.  THEN,
C                     POINT 22 WILL BE PREDICTED FROM POINTS 19,
C                     20, AND 21; AND SO FORTH.  THIS PROCESS IS
C                     REFLECTED AT THE LEFT END OF THE PROFILE,
C                     IF EXTENSION IS OCCURRING THERE.
C
C                     BECAUSE THE ROOT LENGTH CANNOT BE LESS THAN
C                     2, VALUES OF TPARMS(1) THAT ARE LESS THAN 2
C                     ARE USED TO INDICATE SPECIFIC METHODS OF
C                     DERIVING THE ROOT LENGTH.  THESE METHODS
C                     ARE GIVEN IN THE FOLLOWING TABLE, AND SOME
C                     REQUIRE ADDITIONAL VALUES TO BE SPECIFIED
C                     IN TPARMS(2) AND TPARMS(3):
C
C                     TPARMS(1)   ROOT-LENGTH CALCULATION METHOD
C                        -3    LEFT ROOT = TPARMS(2)
C                              RIGHT ROOT = TPARMS(3)
```

```
C                         -2    LEFT ROOT =
C                                  TPARMS(2)*(LEFT EXTENSION)
C                               RIGHT ROOT =
C                                  TPARMS(2)*(RIGHT EXTENSION)
C                         -1    LEFT ROOT = LEFT EXTENSION
C                               RIGHT ROOT = RIGHT EXTENSION
C                          0    EACH ROOT = INPUT ARRAY LEN (NIN)
C                          1    EACH ROOT =
C                                  HALF THE TOTAL EXTENDED LENGTH
C                                  WHICH IS (NOUT-NIN)/2
C                         >1    EACH ROOT = TPARMS(1)
C
C               * 8 = BURG PREDICTION AS IN KTYPE=7 ONLY WITHOUT
C                     THE LINEAR FUNCTION. (TPARMS SAME AS 7).
C
C   TPARMS - REAL*8.  ARRAY CONTAINING ANY PARAMETERS ASSOCIATED
C            WITH ONE OF THE KTYPE NUMBERS.  THIS ARGUMENT WILL
C            NOT BE USED UNLESS DESCRIBED UNDER KTYPE.
C   JUST   - INTEGER*4.  JUSTIFICATION OF THE ORIGINAL PROFILE
C            WITHIN THE EXTENDED PROFILE.  FOLLOWING ARE POSSIBLE
C            VALUES:
C               <-1 = EXACT LOCATION OF LAST POINT FROM RIGHT
C                -1 = RIGHT JUSTIFY (LAST POINT AT END),
C                 0 = CENTER JUSTIFY,
C                 1 = LEFT JUSTIFY (FIRST POINT AT LOCATION 1)
C                >1 = EXACT LOCATION OF FIRST POINT FROM LEFT
C   NIN    - INTEGER*4.  THE NUMBER OF POINTS INPUT.  IF NIN IS
C            LESS THAN NOUT, PROFILE WILL BE EXTENDED.
C   NOUT   - INTEGER*4.  THE NUMBER OF POINTS TO OUTPUT.  IF NOUT
C            IS LESS THAN NIN, PROFILE WILL BE TRUNCATED.
C
C INPUT/OUTPUT ARGUMENT:
C   YFNC   - REAL*8.  ARRAY OF DIMENSION NIN OR NOUT (WHICH EVER
C            IS GREATER) CONTAINING THE DATA TO BE EXTENDED (OR
C            TRUNCATED).
C
C SUBROUTINE EXTEND WRITTEN BY ROB BRACKEN, USGS, 12NOV96.
C HP-9000 SERIES 700/800 UNIX VERSION 1.0, 12NOV96.
C VERSION 1.1, 20010608.  (FIXED BUG IN KTYPE=4)
C
C
      subroutine extend(ktype,tparms,just,nin,nout,yfnc)
C
C DECLARATIONS
C
C     INPUT ARGUMENTS
      integer*4 ktype
      real*8 tparms(*)
      integer*4 just,nin,nout
C
C     INPUT/OUTPUT ARGUMENT
      real*8 yfnc(*)
C
C     INTERNAL VARIABLES
      integer*4 mxburg
      parameter(mxburg=200000)
      real*8 avg,apf(mxburg),acoef(2)
```

```
C
C DO NOTHING IF EQUAL NUMBERS OF INPUT AND OUTPUT POINTS
C
      if(nout.eq.nin) goto 990
C
C DETERMINE NUMBER OF EXTENDED ELEMENTS ON THE LEFT SIDE
C
C     FIND NPTS AND NPTSX
      npts=jmin0(nin,nout)
      nptsx=jmax0(nin,nout)
C
C     MAKE DETERMINATION
      if(just.ge.1) left=just-1
      if(just.eq.0) left=(nptsx-npts)/2
      if(just.le.-1) left=nptsx-npts+just+1
      if(left.lt.0.or.left.gt.nptsx-npts) call errfor(left,
     &  '(extend)  npts and nptsx inconsistant w/r just')
C
C PERFORM TRUNCATION IF FEWER OUTPUT THAN INPUT POINTS
C
      if(nout.lt.nin) then
C       PERFORM TRUNCATION
        if(left.gt.0) then
          do i=1,npts
            yfnc(i)=yfnc(i+left)
          enddo
        endif
        goto 990
      endif
C
C PERFORM EXTENSION
C
C MOVE PROFILE TO RIGHT TO MAKE ROOM FOR THE EXTENSION
C
      if(left.gt.0) then
        do i=npts,1,-1
          yfnc(left+i)=yfnc(i)
        enddo
      endif
C
C FIND LIMITS OF EXTENSION SECTIONS
C
C     LEFT SIDE
      il=1
      jl=left
C
C     RIGHT SIDE
      ir=left+npts+1
      jr=nptsx
C
C DETERMINE THE TYPE OF EXTENSION
C
      goto (100,101,102,103,104,105,106,107,108),ktype+1
      call errfor(ktype,
     & '(extend)  Non-existant extension-type number')
C
C ZERO EXTENSION
```

```
C
  100 do i=il,jl
        yfnc(i)=0.d0
      enddo
      do i=ir,jr
        yfnc(i)=0.d0
      enddo
      goto 990
C
C AVERAGE-VALUE EXTENSION
C
  101 avg=0.d0
      do i=jl+1,ir-1
        avg=avg+yfnc(i)
      enddo
      avg=avg/dflotj(npts)
      do i=il,jl
        yfnc(i)=avg
      enddo
      do i=ir,jr
        yfnc(i)=avg
      enddo
      goto 990
C
C FIRST AND LAST VALUE EXTENSION
C
  102 do i=il,jl
        yfnc(i)=yfnc(jl+1)
      enddo
      do i=ir,jr
        yfnc(i)=yfnc(ir-1)
      enddo
      goto 990
C
C REPEAT EXTENSION
C
  103 do i=jl,il,-1
        yfnc(i)=yfnc(i+npts)
      enddo
      do i=ir,jr
        yfnc(i)=yfnc(i-npts)
      enddo
      goto 990
C
C REFLECT EXTENSION
C
  104 j=jl+1
      kdir=-1
      do i=jl,il,-1
        if(j.ge.ir-1.or.j.le.jl+1) kdir=-kdir
        j=j+kdir
        yfnc(i)=yfnc(j)
      enddo
C
      j=ir-1
      kdir=1
      do i=ir,jr
```

```
        if(j.ge.ir-1.or.j.le.jl+1) kdir=-kdir
        j=j+kdir
        if(j.le.0) then
          yfnc(i)=yfnc(1)
        else
          yfnc(i)=yfnc(j)
        endif
      enddo
      goto 990
C
C REFLECT-AND-INVERT EXTENSION
C
  105 j=jl+1
      kdir=-1
      do i=jl,il,-1
        if(j.ge.ir-1.or.j.le.jl+1) kdir=-kdir
        j=j+kdir
        yfnc(i)=tparms(1)+(tparms(1)-yfnc(j))
      enddo
C
      j=ir-1
      kdir=1
      do i=ir,jr
        if(j.ge.ir-1.or.j.le.jl+1) kdir=-kdir
        j=j+kdir
        yfnc(i)=tparms(2)+(tparms(2)-yfnc(j))
      enddo
      goto 990
C
C SPLINE-TO-ZERO EXTENSION
C
  106 call errfor(ktype,
     & '(extend)  Spline-to-zero extension not installed')
C
C BURG-PREDICTION EXTENSION
C NOTE:  FILTER LENGTH IS NOW .707*(FILTER "ROOT" LENGTH)
C
C     FIND LEFT AND RIGHT FILTER "ROOT" LENGTHS
  108 continue
  107 jp1=jidnnt(tparms(1))
      if(jp1.lt.-3) jp1=0
C
      if(jp1.lt.0) then
        lenl=jl-il+1
        lenr=jr-ir+1
        if(jp1.eq.-2) then
          lenl=jidnnt(dflotj(lenl)*tparms(2))
          lenr=jidnnt(dflotj(lenr)*tparms(2))
        else if(jp1.eq.-3) then
          lenl=jidnnt(tparms(2))
          lenr=jidnnt(tparms(3))
        endif
      else
        lenl=npts
        if(jp1.eq.1) lenl=(nptsx-npts)/2
        if(jp1.gt.1) lenl=jp1
        lenr=lenl
```

```
      endif
C
      if(lenl.lt.3) lenl=3
      if(lenr.lt.3) lenr=3
      if(lenl.gt.npts) lenl=npts
      if(lenr.gt.npts) lenr=npts
c      lenlf=lenl
c      lenrf=lenr
      lenlf=jnint(floatj(lenl-2)/sqrt(2.))+2
      lenrf=jnint(floatj(lenr-2)/sqrt(2.))+2
      if(npts.le.1) goto 102
C
C     REMOVE LSQ LINE
      if(ktype.eq.7) call trendr(1,2,acoef,0,npts,yfnc(jl+1))
C
C     EXTEND TO THE LEFT
      if(lenlf.gt.mxburg) call errfor(lenlf,
     & '(extend)  lenlf exceeded upper dimension of array apf()')
      call burgdp(lenl,yfnc(jl+1),lenlf,apf)
      do j=jl,il,-1
c        yfnc(j)=0.d0
        jm1=j-1
        avg=0.d0
        do i=2,lenlf
c          yfnc(j)=yfnc(j)-yfnc(j+i-1)*apf(i)
          avg=avg-yfnc(jm1+i)*apf(i)
        enddo
        yfnc(j)=avg
      enddo
C
C     EXTEND TO THE RIGHT
      if(lenrf.gt.mxburg) call errfor(lenrf,
     & '(extend)  lenrf exceeded upper dimension of array apf()')
      if(lenr.ne.lenl.or.lenr.ne.npts)
     & call burgdp(lenr,yfnc(ir-lenr),lenrf,apf)
      do j=ir,jr
c        yfnc(j)=0.d0
        jp1=j+1
        avg=0.d0
        do i=2,lenrf
c          yfnc(j)=yfnc(j)-yfnc(j-i+1)*apf(i)
          avg=avg-yfnc(jp1-i)*apf(i)
        enddo
        yfnc(j)=avg
      enddo
C
C     ADD-BACK LSQ LINE
      if(ktype.eq.7) then
        acoef(1)=acoef(1)-acoef(2)*dflotj(left)
        call trendr(1,2,acoef,1,nptsx,yfnc)
      endif
      goto 990
C
C EXIT PROCEDURE
C
  990 return
      end
```

APPENDIX E

Subroutine FORKDC.F

```
C
C_______________________________________________________________________
C
C     SUBROUTINE  F O R K D C
C_______________________________________________________________________
C
C SUBROUTINE FORKDC PERFORMS A FAST FOURIER TRANSFORM (FFT) ON A
C SAMPLE FUNCTION (SEE NOTE BELOW ON TERMINOLOGY) IN A
C DOUBLE-PRECISION COMPLEX ARRAY.  THE TRANSFORM MAY BE PERFORMED
C EITHER FORWARD (TIME TO FREQUENCEY DOMAIN) OR INVERSE
C (FREQUENCY TO TIME DOMAIN).  THE INPUT/OUTPUT ARRAY LENGTH IS
C RESTRICTED TO POWERS OF 2 (SEE NOTE BELOW ON POWERS OF 2).
C
C THE FOLLOWING TABLE SHOWS TIMES REQUIRED FOR VARIOUS ROUTINES
C TO PERFORM A FORWARD AND INVERSE TRANSFORM.  (TIMES TAKEN ON
C MUSETTE, HP-9000 SERIES 700/800 COMPUTER, ON 14MAY97.  THIS
C COMPUTER RUNS ROUGHLY 30 TIMES FASTER THAN A VAX 11/750.)
C
C    ROUTINE   TYPE      RULE      LENGTH    TIME (SEC)
C
C  -> FORKDC    FFT      N*LOGN     4096        0.80 <-
C     DFTTDR    DFT      N*N        4096       31.69
C     DFTFDR    DFT      N*N        4096       31.69
C     DFTTDC    DFT      N*N        4096       66.98
C
C    (THE DFT IS FASTER THAN THE FFT FOR N < 64)
C
C
C THE FFT ALGORITHM WAS FIRST PUBLICIZED BY COOLEY AND TUKEY IN
C 1965.  THE FORTRAN CODING, SUBROUTINE FORK (2/15/69), WAS
C MODIFIED FROM BRENNER BY JON F. CLAERBOUT, DEPARTMENT OF
C GEOPHYSICS, STANFORD UNIVERSITY, AND PUBLISHED IN "FUNDAMENTALS
C OF GEOPHYSICAL DATA PROCESSING:  WITH APPLICATIONS TO PETROLEUM
C PROSPECTING", 1976 MCGRAW-HILL, PAGE 12.  IT WAS LATER ADAPTED
C TO A USGS COMPUTER BY MIKE WEBRING, THEN CONVERTED TO DOUBLE
C COMPLEX ARITHMETIC (FORKDC) AND NOTES SUPPLIED BY ROB BRACKEN
C IN 1996.  THE DFT PROCEDURE IS SUMMARIZED AS FOLLOWS (THE FFT
C IS A STREAMLINED VERSION WHICH TAKES ADVANTAGE OF SYMMETRIES IN
C A MATRIX VERSION OF THE FOLLOWING EQUATION):
C
C                    NX
C CX(K)=SQRT(1/NX)* SUM (CX(J)*EXP(2*PI*SIGNI*I*(J-1)*(K-1)/NX))
C                   J=1
C   WHERE   K   =  1,2, ... (NX=2**INTEGER)
C         SIGNI = -1 (FORWARD),
C               =  1 (INVERSE)
C
C NOTE ON TERMINOLOGY:  A SAMPLE FUNCTION (IN THIS CONTEXT) IS
C THE TIME-DOMAIN FUNCTION OF INTEREST WHICH IS BEING INPUT TO
C THE FFT ALGORITHM.  IT IS USUALLY A FINITE-LENGTH PIECE OF A
C LONGER FUNCTION AND IS, HOPEFULLY, REPRESENTATIVE OF THAT
C FUNCTION IN TERMS OF SPECTRAL CONTENT.  THE FOURIER TRANSFORM
C CONVERTS A TIME-DOMAIN FUNCTION INTO A FREQUENCY DOMAIN
C SPECTRUM AND VICE-VERSA.  THE ACTUAL FOURIER TRANSFORM HAS
C INFINITE LIMITS OF INTEGRATION; THAT IS, IT REQUIRES A SAMPLE
C FUNCTION OF INFINITE LENGTH.  A VERSION OF THE FOURIER
C TRANSFORM WHICH HAS FINITE LIMITS (AND THUS USES A PHYSICALLY
```

```
C REALIZABLE SAMPLE FUNCTION) IS CALLED THE FINITE FOURIER
C TRANSFORM.  THE FINITE FOURIER TRANSFORM REQUIRES A
C CONTINUOUSLY VARYING (OR ANALOG) SAMPLE FUNCTION WHICH CANNOT
C BE SUPPLIED IN A DIGITAL COMPUTER.  A DIGITAL VERSION IS THE
C DISCRETE FOURIER TRANSFORM (DFT) WHICH USES AN EVENLY SAMPLED
C (DISCRETE) SAMPLE FUNCTION AND PRODUCES SPECTRA WITH DISCRETE
C FREQUENCIES.  THE FAST FOURIER TRANSFORM IS A VERSION OF THE
C DFT WHICH REDUCES COMPUTATION TIME.
C
C NOTE ON DATA PREPARATION:  BEFORE TRANSFORMING A SAMPLE
C FUNCTION, THE DATA SHOULD BE PREPARED SO AS TO MINIMIZE NOISE
C AND FOCUS THE SPECTRUM APPROPRIATELY.  THE PREPARATION SHOULD
C INCLUDE THE FOLLOWING OPERATIONS:  1) REMOVE SPIKES, STEPS, AND
C OTHER EXTRANEOUS NOISES WHICH ARE NOT OF INTEREST;  2) REMOVE A
C TREND SUCH AS A LEAST-SQUARES-FIT LINE (KEEP THE COEFFICIENTS
C IF THE LINE IS TO BE ADDED BACK LATER); AND 3) SHAPE THE DATA
C WITH A WINDOWING FUNCTION.
C
C NOTE ON POWER OF 2 RESTRICTION:  IF THE AVAILABLE FUNCTION IS
C NOT AN INTEGER POWER OF 2 IN LENGTH AND IT CANNOT BE TRUNCATED,
C IT MAY BE PADDED WITH ZEROS TO THE NEXT POWER OF 2.  THIS
C PROCEDURE IS GENERALLY ACCEPTED AS THE BEST THAT CAN BE DONE
C WITH NON-POWER-OF-TWO DATA IN AN FFT.  HOWEVER, BE WARNED THAT
C IT MAY INCREASE BACKGROUND NOISE BY 10 dB OR MORE.  THE NOISE
C INCREASES IN PROPORTION TO THE PERCENTAGE OF PADDING.
C
C NOTE ON PERODIC EXTENSION:  THE FINITE FOURIER TRANSFORM (AND
C ALL SUBSETS) ASSUMES PERIODIC EXTENSION BEYOND IT'S LIMITS OF
C INTEGRATION.  IT EXTENDS THE SAMPLE RECORD IN BOTH DIRECTIONS
C (POSITIVE AND NEGATIVE) BY REPLICATION.  FOR EXAMPLE, AT THE
C END OF THE ACTUAL SAMPLE FUNCTION, AN IDENTICAL COPY OF THE
C ORIGINAL SAMPLE FUNCTION STARTS ALL OVER AGAIN.  THIS
C REPLICATION EXTENDS TO POSITIVE AND NEGATIVE INFINITY; AND
C DISCONTINUITIES BETWEEN REPLICATIONS USUALLY REQUIRE WINDOWING
C TO LESSEN THE EFFECTS.  IN THE CASE OF A DFT, THE REPLICATION
C BEGINS AT THE NEXT SAMPLE POINT AFTER THE END OF THE PREVIOUS
C SAMPLE FUNCTION.  THIS IS A SUBTLETY WHICH REQUIRES THE LAST
C POINT OF AN OTHERWISE SYMMETRIC WINDOWING FUNCTION TO BE CUT
C OFF.
C
C NOTE ON WINDOWING:  ALL TIME-DOMAIN SAMPLE FUNCTIONS SHOULD BE
C WINDOWED APPROPRIATELY BEFORE TRANSFORMING.  BECAUSE THE
C TRANSFORM ASSUMES PERIODIC EXTENSION OF THE FINITE SAMPLE
C FUNCTION, DISCONTINUITIES MAY BE IMPLIED AT THE BEGINNING AND
C END; THOSE DISCONTINUITIES ARE FOLDED BACK INTO THE FREQUENCY
C DOMAIN AND PRODUCE SERIOUS SIDE LOBING ADJACENT TO EACH
C DISCRETE FREQUENCY (BIN).  A GOOD WINDOWING FUNCTION SLOWLY
C DIMINISHES THE TIME-DOMAIN AMPLITUDE TOWARD THE END POINTS
C (MINIMIZING REPLICATED DISCONTINUITIES); AND, IN SO DOING, IT
C MINIMIZES SIDE LOBING AND FREQUENCY SMEARING WITHIN THE
C FREQUENCY DOMAIN.
C
C NOTE ON THE KAISER-BESSEL WINDOW:  ALTHOUGH THERE ARE MANY
C WINDOWING FUNCTIONS AVAILABLE, DEMONSTRABLY, ONE OF THE BEST IS
C THE KAISER-BESSEL WINDOW.  IF THE RECOMMENDED ALPHA=3.0 IS
C USED, IT HAS A MAXIMUM SIDE LOBE LEVEL OF -69 dB AND A 6-dB
C BANDWIDTH OF 2.39 BINS.  AN IN-DEPTH DISCUSSION OF THE
```

```
C KAISER-BESSEL WINDOW (AND OTHER WINDOWS) MAY BE FOUND IN
C "WINDOWS, HARMONIC ANALYSIS, AND THE DISCRETE FOURIER
C TRANSFORM" BY FREDRIC J.HARRIS, UNDERSEA SURVEILLANCE
C DEPARTMENT, SEPTEMBER 1976, NAVAL UNDERSEA CENTER, SAN DIEGO,
C CALIFORNIA.  THIS INFORMATION WAS SUPPLIED BY MIKE WEBRING,
C USGS.  THE KAISER-BESSEL-WINDOW EQUATION IS:
C
C   w(n) = Io[pi*alpha*sqrt( 1-(n/(N/2))**2 )] / Io[pi*alpha]
C
C   WHERE    0 <= |n| <= N/2     0 < alpha <= 3.5
C                    inf
C            Io[X] = sum[ ( ((X/2)**k)/k! )**2 ]
C                    k=0
C
C NOTE ON RESTORING WINDOWED DATA:  A FULL LENGTH WINDOW (SUCH AS
C THE KAISER-BESSEL) SIGNIFICANTLY REDUCES THE AMPLITUDE OF THE
C ORIGINAL TIME DOMAIN SAMPLE FUNCTION ESPECIALLY NEAR THE END
C POINTS.  IF A SAMPLE FUNCTION HAS BEEN WINDOWED, FORWARD
C TRANSFORMED, MODIFIED IN THE FREQUENCY DOMAIN, AND THEN
C TRANSFORMED BACK TO THE TIME DOMAIN, OBVIOUSLY THE INVERSE
C WINDOW MUST BE APPLIED TO RESTORE THE MODIFIED SAMPLE FUNCTION.
C THIS PROCEDURE WORKS WELL EXCEPT FOR TWO MINOR PITFALLS:
C 1) RESTORED FUNCTION VALUES NEAR THE ENDPOINTS MAY SUFFER
C MACHINE PRECISION ERRORS; AND 2) CONTRIBUTIONS FROM WAVEFORMS
C IN THE BEGINNING AND ENDING (31% FOR KAISER-BESSEL WITH
C ALPHA=3.0) OF THE TIME DOMAIN FUNCTION WILL BE ATTENUATED BY
C MORE THAN 6 dB.  THIS MEANS THAT THE SPECTRUM WILL BE BIASED
C TOWARD THE MIDDLE SECTION OF THE TIME DOMAIN FUNCTION AND THE
C VERY LOW FREQUENCIES MAY BE COMPROMISED.  HOWEVER, THESE
C PITFALLS ARE MUCH LESS SEVERE THAN THOSE OF NOT USING A GOOD
C WINDOW.
C
C NOTE ON OVERLAPPING WINDOWS:  AS HAS BEEN EXPLAINED BY MIKE
C WEBRING, THE CENTER BIAS PROBLEM MAY BE MINIMIZED BY WINDOWING
C OVERLAPPING SECTIONS OF DATA AND THEN STACKING IN THE FREQUENCY
C DOMAIN.  OVERLAPPING AND STACKING OPTIMIZES THE EXPECTED (MEAN)
C VALUE FOR THE SPECTRAL COMPONENTS.  THE SEPARATION BETWEEN
C CENTERS OF OVERLAPPING WINDOWS WILL VARY CONSIDERABLY DEPENDING
C ON THE TYPE AND SPECIFICATIONS OF THE WINDOW USED.  FOR
C EXAMPLE, THE KAISER-BESSEL WINDOW WITH ALPHA=3.0 SHOULD HAVE
C CENTER SEPARATIONS NO MORE THAN 38% OF EACH WINDOW LENGTH.
C AMPLITUDES MAY INCREASE IN OVERLAPPING PARTS OF THE SAMPLE
C FUNCTION.
C
C NOTE ON TESTS:  MANY OF THE NOTES ABOVE WERE DERIVED FROM TESTS
C DONE BY ROB BRACKEN, USGS.  THE RESULTS OF THESE TESTS ARE IN
C RING CORE NBF#6 9AUG96 - 16AUG96.
C
C
C
C INPUT ARGUMENTS:
C   NX     - INTEGER*4.  NUMBER OF POINTS IN ARRAY CX.  NX MUST
C            BE AN INTEGER POWER OF 2.
C   SIGNI  - REAL*8.  DIRECTION OF TRANSFORM:
C               -1.D0 = FORWARD TRANSFORM (TIME TO FQ).
C               +1.D0 = INVERSE TRANSFORM (FQ TO TIME).
C
```

```
C INPUT/OUTPUT ARGUMENT:
C   CX     - COMPLEX*16.  ARRAY CONTAINING THE FUNCTION.  CX MUST
C            HAVE DIMENSION NX (OR LARGER).  ONLY ELEMENTS 1
C            THROUGH NX WILL BE TRANSFORMED.
C
C SUBROUTINE FORK WRITTEN BY JON F. CLAERBOUT, STANFORD, 15FEB69.
C ADAPTED TO USGS COMPUTER BY MIKE WEBRING, USGS, (DATE?)
C CONVERSION TO DBLE CMPLX & NOTES BY ROB BRACKEN, USGS, 15AUG96.
C HP-9000 SERIES 700/800 UNIX VERSION 1.0, 15AUG96.
C
C
        subroutine forkdc( nx, cx, signi )
C
C DECLARATIONS
C
C     INPUT ARGUMENTS
      integer*4 nx
      real*8 signi
C
C     INPUT/OUTPUT ARGUMENT
      complex*16 cx(1)
C
C     INTERNAL VARIABLES
      complex*16 carg,cw,ctemp
      real*8 spi,pi,sc,arg
      parameter(pi=3.1415 92653 58979 32384 62643)
C
C INITIALIZATIONS
C
      spi=signi*pi
      sc=dsqrt( 1.d0/dflotj(nx) )
C
C COMPUTATIONS
C
      j=1
      do 701 i=1,nx
        if(i.le.j) then
          ctemp=cx(j)*sc
          cx(j)=cx(i)*sc
          cx(i)=ctemp
        endif
        m=nx/2
  202   if(j.gt.m) then
          j=j-m
          m=m/2
          if(m.ge.1) goto 202
        endif
        j=j+m
  701 continue
C
      ne=1
      dowhile(ne.lt.nx)
        istep=2*ne
        do 703 m=1,ne
          arg=(spi*dflotj(m-1))/dflotj(ne)
          carg=dcmplx(0.d0,arg)
          cw=cdexp(carg)
```

```
          do 704 i=m,nx,istep
            k=i+ne
            ctemp=cw*cx(k)
            cx(k)=cx(i)-ctemp
            cx(i)=cx(i)+ctemp
  704     continue
  703   continue
        ne=istep
      enddo
C
C EXIT PROCEDURE
C
  990 return
      end
```

APPENDIX F

Subroutine TRENDR.F

```
C
C_____________________________________________________________________
C
C     SUBROUTINE  T R E N D R
C_____________________________________________________________________
C
C SUBROUTINE TRENDR REMOVES OR ADDS BACK A TREND FROM AN
C EVENLY-SPACED DOUBLE-PRECISION FUNCTION ARRAY.  THE TYPE OF
C TREND MAY BE SELECTED USING KTYPE.
C
C INPUT ARGUMENTS:
C   KTYPE  - INTEGER*4.  THE TYPE OF TREND.  OPTIONS ARE:
C            * 0 = NO TREND
C            * 1 = POLYNOMIAL OF ORDER NCOEF-1
C              2 = COS FNC WITH AMPS, FQS, AND PHASES IN ACOEF
C   NCOEF  - INTEGER*4.  NUMBER OF COEFS IN ACOEF.  IF NCOEF IS
C            MADE NEGATIVE AND NVERSE <= 0, INSTEAD OF FINDING
C            NEW COEFS, THE COEFS GIVEN IN ACOEF WILL BE USED FOR
C            REMOVING THE TREND.  IF NCOEF IS NEGATIVE AND NVERSE
C            => 1, THERE WILL BE NO EFFECT DIFFERENT FROM THAT OF
C            POSITIVE NCOEF (I.E. THE COEFS IN ACOEF WILL BE
C            USED FOR ADDING BACK THE TREND).
C
C            NOTE:  CURRENTLY, NCOEF MAY NOT EXCEED 3 WHEN KTYPE
C            IS 1.
C
C   ACOEF  - REAL*8.  THE COEFS DESCRIBING THE TREND FUNCTION.
C            IF THE TREND IS BEING REMOVED, ACOEF IS AN
C            INPUT/OUTPUT ARGUMENT.  IF THE TREND IS BEING ADDED
C            BACK, ACOEF IS AN INPUT ARGUMENT.
C
C   NVERSE - INTEGER*4.  DETERMINES WHETHER TO REMOVE OR ADD BACK
C            THE TREND:
C               NVERSE <= 0 MEANS TO REMOVE THE TREND.
C               NVERSE => 1 MEANS TO ADD BACK THE TREND.
C   NPTS   - INTEGER*4.  THE NUMBER OF DATA POINTS IN ARRAY YFNC.
C            IF NPTS IS MADE NEGATIVE AND NVERSE <= 0, THE TREND
C            WILL NOT BE REMOVED (THE COEFS CAN THEN BE FOUND
C            WITHOUT CHANGING THE DATA ARRAY).  IF NPTS IS
C            NEGATIVE AND NVERSE => 1, THE TREND WILL NOT BE
C            ADDED BACK (THIS HAS NO PARTICULAR USE BUT IS
C            PROVIDED FOR FUNCTIONAL BALANCE).
C INPUT/OUTPUT ARGUMENT:
C   YFNC   - REAL*8.  ARRAY OF DIMENSION NPTS CONTAINING THE
C            FUNCTION TO HAVE THE TREND REMOVED OR ADDED BACK.
C
C
C SUBROUTINE TRENDR WRITTEN BY ROB BRACKEN, USGS, 26AUG96.
C HP-9000 SERIES 700/800 UNIX VERSION 1.0, 26AUG96.
C HP-9000 SERIES 700/800 UNIX VERSION 1.1, 19970611.
C   ADDITION OF OPTION TO FIND COEFS BUT NOT REMOVE THE TREND
C
C
      subroutine trendr(ktype,ncoef,acoef,
     &                  nverse,npts,yfnc)
C
C DECLARATIONS
```

```
C
C     INPUT ARGUMENTS
      integer*4 ktype,ncoef,nverse,npts
C
C     INPUT/OUTPUT ARGUMENTS
      real*8 acoef(*),yfnc(*)
C
C SELECT THE TYPE OF TREND
C
      goto (101,102),ktype
      goto 990
C
C TRENDS
C
C     POLYNOMIAL
  101 continue
      call lsqply(ncoef,acoef, nverse,npts,yfnc)
      goto 990
C
C     COSINE
  102 continue
      call errfor(ktype,'cosine trend not installed')
      goto 990
C
C EXIT PROCEDURE
C
  990 return
      end
```

APPENDIX G

Subroutine WINDOW.F

```
C
C_____________________________________________________________________
C
C     SUBROUTINE  W I N D O W
C_____________________________________________________________________
C
C SUBROUTINE WINDOW APPLIES OR REMOVES A SELECTED WINDOW FROM AN
C EVENLY-SPACED DOUBLE-PRECISION FUNCTION ARRAY.  WINDOWS MAY BE
C SPECIFIED AS DFT-EVEN OR SYMMETRIC.  ALSO, A NUMBER OF
C WINDOWING FUNCTIONS ARE AVAILABLE AND MAY BE SELECTED WITH
C KTYPE.
C
C   KTYPE  - INTEGER*4.  NUMBER INDICATING THE TYPE OF WINDOW.
C            FOLLOWING IS A LIST OF WINDOWS.  A STAR (*)
C            INDICATES THAT THE OPTION HAS BEEN INSTALLED:
C
C                KTYPE   NAME (OTHER NAMES OR DESC)       ALPHA
C               *  0 = RECTANGLE (BOXCAR, NO WINDOWING)     -
C               *  1 = TRIANGLE  (FEJER OR BARTLET)         -
C               *  2 = COS**ALPHA (HANNING)             1.0 - 4.0
C               *  3 = HAMMING (MODIFIED HANNING)           -
C               *  4 = RIESZ (SIMPLEST CONT POLYNOM)       2.0
C                  5 = RIEMANN                              -
C                  6 = DE LA VALLE-POUSSIN (CUBIC)          -
C               *  7 = TUKEY (COSINE TAPERED)           0.0 - 1.0
C                  8 = BOHMAN                               -
C                  9 = POISSON (2-SIDED EXPONENTIAL)    2.0 - 4.0
C                 10 = HANNING-POISSON                  0.5 - 2.0
C                 11 = CAUCHY                           3.0 - 5.0
C                 12 = GAUSSIAN (MIN TIME/BANDWID PROD) 2.5 - 3.5
C                 13 = DOLPH-TCHEBYSHEV                 2.5 - 4.0
C               * 14 = KAISER-BESSEL (OVERALL BEST)     2.0 - 3.5
C                 15 = BARCILON-TEMES                   3.0 - 4.0
C
C   ALPHA  - REAL*8.  THE DEGREE OF WINDOWING FOR WINDOWS WHICH
C            ALLOW A SELECTION.  IF ALPHA=0.D0, THE BEST VALUE
C            FOR THAT WINDOW WILL BE SELECTED AUTOMATICALLY
C            (ALPHA WILL NOT BE CHANGED HOWEVER).
C   WFRAC  - REAL*8.  WINDOW FRACTION FOR SPLITTING WINDOWS.  IF
C            WFRAC IS GREATER THAN ZERO AND LESS THAN ONE, A
C            SPLIT WINDOW WILL BE INDICATED.  A SPLIT WINDOW IS
C            SIMPLY THE SPECIFIED WINDOW, KTYPE, APPLIED TO ONLY
C            THE ENDS OF THE PROFILE SUCH THAT THE PROFILE TAPERS
C            TO ZERO AT EACH END AND HAS ITS VALUES UNCHANGED IN
C            THE MIDDLE.  WFRAC SPECIFIES THE FRACTION OF THE
C            ORIGINAL PROFILE LENGTH TAKEN BY WINDOW TAPERING.
C            NOTE THAT A BOX CAR WINDOW (KTYPE=0) CANNOT BE
C            SHORTENED USING WFRAC; THAT PROCEDURE MUST BE DONE
C            WITH ZERO EXTENSION (SEE SUBROUTINE EXTEND OR USE
C            KTYPE=4, RIESZ WINDOW AND ALPHA NEARLY 0).
C   KDFT   - INTEGER*4.  DETERMINES WHETHER THE WINDOW IS
C            DFT-EVEN (KDFT => 1) OR SYMMETRIC (KDFT <= 0).
C            DFT-EVEN WINDOWING SHOULD BE USED WITH DISCRETE
C            FOURIER TRANSFORMS.  HOWEVER, IF THE DATA ARE TO BE
C            PADDED WITH ZEROS AFTER WINDOWING, SYMMETRIC
C            WINDOWING IS MORE APPROPRIATE.
C   NVERSE - INTEGER*4.  DETERMINES WHETHER TO APPLY OR REMOVE
```

```
C            THE WINDOW:
C               NVERSE <= 0 MEANS TO APPLY THE WINDOW.
C               NVERSE => 1 MEANS TO REMOVE THE WINDOW.
C            NOTE: SOME WINDOWS GO TO ZERO AT THE END POINTS.  IN
C            THESE CASES, WHEN THE WINDOW IS REMOVED, EACH END
C            POINT OF THE FUNCTION WILL BE SET EQUAL TO THE
C            ADJACENT FUNCTION VALUE.
C   NPTS   - INTEGER*4.  THE NUMBER OF DATA POINTS IN ARRAY YFNC.
C
C INPUT/OUTPUT ARGUMENT:
C   YFNC   - REAL*8.  ARRAY CONTAINING THE DATA TO BE WINDOWED.
C
C
C SUBROUTINE WINDOW WRITTEN BY ROB BRACKEN, USGS, 26AUG96.
C HP-9000 SERIES 700/800 UNIX VERSION 1.0, 26AUG96.
C HP-9000 SERIES 700/800 UNIX VERSION 1.1, 19970325.
C HP-9000 SERIES 700/800 UNIX VERSION 1.2, 19970602.
C HP-9000 SERIES 700/800 UNIX VERSION 2.0, 19970602.
C HP-9000 SERIES 700/800 UNIX VERSION 2.1, 19970609.
C
C
      subroutine window(ktype,alpha,wfrac,kdft,nverse,npts,yfnc)
      SAVE
C
C DECLARATIONS
C
C     INPUT ARGUMENTS
      integer*4 ktype
      real*8 alpha,wfrac
      integer*4 kdft,nverse,npts
C
C     INPUT/OUTPUT ARGUMENT
      real*8 yfnc(*)
C
C     INTERNAL VARIABLES
      real*8 alpha2,alpha3,wfrac2
      integer*4 ktype2,kdft2,npts2
      data ktype2,alpha2,wfrac2,kdft2,npts2
     & / -999, 1.7d+37, 1.7d+37, -999, -999 /
C
C     INTERNAL WINDOW
      integer*4 mxw
      parameter(mxw= 800 000 )
      real*8 dwindo(mxw)
C
C
C CHECK WINDOW TYPE
C
      if(ktype.ge.1.and.ktype.le.15) goto 254
      if(ktype.eq.0) goto 990
      call errfor(ktype,'(WINDOW)  Non-existant window type')
C
C
C USE THE PREVIOUSLY CALCULATED WINDOW
C
  254 if(ktype.eq.ktype2.and.alpha.eq.alpha2.and.wfrac.eq.wfrac2
     & .and.kdft.eq.kdft2.and.npts.eq.npts2) then
```

```
C
C       DWINDO CONTAINS THE DESIRED WINDOW
        if(nverse.le.0) then
C
C         APPLY THE PREVIOUSLY CALCULATED WINDOW
          do 751 i=1,npts2
            yfnc(i)=yfnc(i)*dwindo(i)
  751     continue
        else
C
C         REMOVE THE PREVIOUSLY CALCULATED WINDOW
          do 752 i=2,npts2-1
            yfnc(i)=yfnc(i)/dwindo(i)
  752     continue
C
C         FIRST WINDOW POINT MAY BE ZERO
          if(dwindo(1).eq.0.d0) then
            yfnc(1)=yfnc(2)
          else
            yfnc(1)=yfnc(1)/dwindo(1)
          endif
C
C         LAST WINDOW POINT MAY BE ZERO
          if(dwindo(npts2).eq.0.d0) then
            yfnc(npts2)=yfnc(npts2-1)
          else
            yfnc(npts2)=yfnc(npts2)/dwindo(npts2)
          endif
        endif
        goto 990
      endif
C
C
C CALCULATE A NEW WINDOW
C
C     CHECK SIZE OF WINDOW AGAINST INTERNAL ARRAY SIZE
      if(npts.gt.mxw) call errfor(mxw,
     & '(WINDOW)  Internal array dwindo too small for data')
C
C     UPDATE INTERNAL WINDOW CHARACTERISTICS
      ktype2=ktype
      alpha2=alpha
      wfrac2=wfrac
      kdft2=kdft
      npts2=npts
C
C     CHECK VALUE OF WFRAC TO SET UP FOR POSSIBLE SPLIT WINDOW
      if(wfrac2.gt.0.d0.and.wfrac2.lt.1.d0) then
C
C       WINDOW TO BE SPLIT WITH ONES FILLING THE MIDDLE
        ksym=0
        if(kdft2.le.0) ksym=1
        npts3=jidnnt(wfrac2*dflotj(npts2-ksym))+ksym
        ncopy=(npts3-ksym)/2+ksym
        nfill=npts2-npts3
      else
C
```

```
C       NORMAL WINDOW
        npts3=npts2
        ncopy=0
        nfill=0
      endif
C
C     INITIALIZE INTERNAL ARRAY, DWINDO
      do 753 i=1,npts3
        dwindo(i)=1.d0
  753 continue
C
C     SET INTERNAL INVERSE TO APPLY THE WINDOW
      nversx=0
C
C     SELECT THE DESIRED WINDOW
      goto (101,102,103,104,105,106,107,108,109,
     &      110,111,112,113,114,115),ktype2
      goto 990
C
C
C ================ BEGIN WINDOWS ================
C
C     TRIANGLE  (FEJER OR BARTLET)     ALPHA =     -
  101 continue
      alpha3=alpha2
      if(alpha3.eq.0.d0) alpha3=1.d0
      call fejer(alpha3,nversx,kdft2,npts3,dwindo)
      goto 150
C
C     COS**ALPHA (HANNING)             ALPHA = 1.0 - 4.0
  102 continue
      alpha3=alpha2
      if(alpha3.eq.0.d0) alpha3=2.d0
      call hann3(alpha3,nversx,kdft2,npts3,dwindo)
      goto 150
C
C     HAMMING (MODIFIED HANNING)       ALPHA =     -
  103 continue
      alpha3=alpha2
      if(alpha3.eq.0.d0) alpha3=0.d0
      call hamm3(alpha3,nversx,kdft2,npts3,dwindo)
      goto 150
C
C     RIESZ (SIMPLEST CONT POLYNOM)    ALPHA =    2.0
  104 continue
      alpha3=alpha2
      if(alpha3.eq.0.d0) alpha3=2.d0
      call riesz(alpha3,nversx,kdft2,npts3,dwindo)
      goto 150
C
C     RIEMANN                          ALPHA =     -
  105 continue
      call errfor(ktype2,
     & '(WINDOW)  riemann window not installed')
      goto 150
C
C     DE LA VALLE-POUSSIN (CUBIC)      ALPHA =     -
```

```
  106 continue
      call errfor(ktype2,
     & '(WINDOW)  de la valle-poussin window not installed')
      goto 150
C
C     TUKEY (COSINE TAPERED)           ALPHA = 0.0 - 1.0
  107 continue
      alpha3=alpha2
      if(alpha3.eq.0.d0) alpha3=0.5d0
      call tukey(alpha3,nversx,kdft2,npts3,dwindo)
      goto 150
C
C     BOHMAN                           ALPHA =     -
  108 continue
      call errfor(ktype2,
     & '(WINDOW)  bohman window not installed')
      goto 150
C
C     POISSON (2-SIDED EXPONENTIAL)    ALPHA = 2.0 - 4.0
  109 continue
      call errfor(ktype2,
     & '(WINDOW)  poisson window not installed')
      goto 150
C
C     HANNING-POISSON                  ALPHA = 0.5 - 2.0
  110 continue
      call errfor(ktype2,
     & '(WINDOW)  hanning-poisson window not installed')
      goto 150
C
C     CAUCHY                           ALPHA = 3.0 - 5.0
  111 continue
      call errfor(ktype2,
     & '(WINDOW)  cauchy window not installed')
      goto 150
C
C     GAUSSIAN (MIN TIME/BANDWID PROD) ALPHA = 2.5 - 3.5
  112 continue
      call errfor(ktype2,
     & '(WINDOW)  gaussian window not installed')
      goto 150
C
C     DOLPH-TCHEBYSHEV                 ALPHA = 2.5 - 4.0
  113 continue
      call errfor(ktype2,
     & '(WINDOW)  dolph-tchebyshev window not installed')
      goto 150
C
C     KAISER-BESSEL (OVERALL BEST)     ALPHA = 2.0 - 3.5
  114 continue
      alpha3=alpha2
      if(alpha3.eq.0.d0) alpha3=3.d0
      call kais3(alpha3,nversx,kdft2,npts3,dwindo)
      goto 150
C
C     BARCILON-TEMES                   ALPHA = 3.0 - 4.0
  115 continue
```

```
      call errfor(ktype2,
     & '(WINDOW)  barcilon-temes window not installed')
      goto 150
C
  150 continue
C
C ================= END WINDOWS =================
C
C
C     IF SPLIT WINDOW, SLIDE RIGHT HALF RIGHT & FILL SPLIT W/ONES
      if(nfill.gt.0) then
        do i=npts2,npts2-ncopy+1,-1
          dwindo(i)=dwindo(i-nfill)
        enddo
        do j=i,i-nfill+1,-1
          dwindo(j)=1.d0
        enddo
      endif
C
C     RETURN TO APPLY OR REMOVE THE WINDOW FROM THE DATA
      goto 254
C
C EXIT PROCEDURE
C
  990 continue
      end
```