U. S. DEPARTMENT OF THE INTERIOR

U.S. GEOLOGICAL SURVEY

The Inverse of Winnowing: a Fortran Subroutine and Discussion of Unwinnowing Discrete Data

by

Robert E. Bracken[1]

Open-File Report 03-229

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

Table of Contents

Abstract

This report describes an unwinnowing algorithm that utilizes a discrete Fourier transform, and a resulting Fortran subroutine that winnows or unwinnows a 1-dimensional stream of discrete data; the source code is included. The unwinnowing algorithm effectively increases (by integral factors) the number of available data points while maintaining the original frequency spectrum of a data stream. This has utility when an increased data density is required together with an availability of higher order derivatives that honor the original data.

Introduction

A process known as *winnowing* (also called decimation, subsampling, or resampling to a courser interval) is used with data streams either to reduce the size of the data set or to increase the sampling interval. Winnowing typically involves the removal of one or more data points adjacent to every data point retained in the data stream; the retained data points are selected at a constant sampling interval. An inverse process here called *unwinnowing* (known as reconstitution or resampling to a finer interval) involves the insertion of one or more data points adjacent to every data point extant in a data stream.

This report describes an unwinnowing algorithm that utilizes a discrete Fourier transform, and a resulting Fortran subroutine that winnows or unwinnows a 1-dimensional stream of discrete data; the source code is included. The unwinnowing algorithm effectively increases (by integral factors) the number of available data points while maintaining the original frequency spectrum of a data stream. This has utility when an increased data density is required together with the availability of higher order derivatives that honor the original data.

The unwinnowing capability has application to certain forms of data analysis and the merging of lower-frequency with higher-frequency sampled data streams. It is particularly helpful in working with data from multiple-instrument platforms, as may be found in airborne geophysical surveys or UXO detection and discrimination equipment. Typically, each instrument will have an inherent or favored sampling interval that differs from the other instruments on the platform. Yet, proper reduction often requires that the data streams be analyzed simultaneously – at identical sampling intervals. Winnowing the higher-frequency data stream (to match the lower) will remove potentially important information, while simple interpolation between the data points in the lower-frequency data stream can add noise, dishonor the data, or distort its derivatives. In these instances an unwinnowing algorithm, such as is presented here, that maintains the spectrum of the original data can be quite useful.

A Basic Winnowing Process

A clear understanding of the winnowing process forms a basis for the unwinnowing process. As an example of winnowing, assume that a winnow factor of 3 is applied to a time-domain profile of data (a data stream) that was originally sampled at an interval of 1 second for a period of 12 seconds (that is, 12 samples). The first point is kept, the second and third points are removed, the fourth point is kept, and so on. So, the original and winnowed data sets have points at times:

```
samp times
original   0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11 seconds
winnowed   0,       3,       6,       9,      seconds
```

Now, a phenomenon becomes apparent after examining the function values of the original and the winnowed data streams:

```
function values
original   0, 3,-3, 0,-3, 3, 0, 3,-3, 0,-3, 3
winnowed   0,       0,       0,       0,
```

The winnowed function is all zeros! So, why was the function completely destroyed by winnowing? The problem has to do with the fact that the frequency content of the original profile exceeds the Nyquist frequency of the winnowed profile. The Nyquist frequency is the highest frequency that can be represented at a given sample interval. There must be a minimum of 2 samples during each cycle of the signal; if not, the high-frequency signal will fold back and appear as a lower frequency, at or below the Nyquist frequency of the sample interval. The following illustrates the frequency content of the original signal by showing that its function is the sum of two sine-wave functions:

```
samp times 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11 seconds

func. A   -2, 2,-2, 2,-2, 2,-2, 2,-2, 2,-2, 2  1/2 Hz
func. B    2, 1,-1,-2,-1, 1, 2, 1,-1,-2,-1, 1  1/6 Hz
orig.= A+B 0, 3,-3, 0,-3, 3, 0, 3,-3, 0,-3, 3
```

Function A, with a Nyquist of 1/2 Hz, cannot be properly represented by the winnowed sample interval that has a Nyquist of only 1/6 Hz. So upon winnowing, the 1/2-Hz signal folds back to a 1/6-Hz aliased waveform. However, function B, already with a Nyquist of 1/6 Hz, can be represented (barely) by the winnowed sample interval; no information is lost from this signal upon winnowing. Because the aliased function A is out of phase with function B, the two functions sum to zero. It follows then, that any signals with frequencies above the winnowed Nyquist must be removed (attenuated to an acceptably small level) BEFORE winnowing takes place. This is done by applying a low-pass filter to the original data.

```
func. A    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  Removed
func. B    2, 1,-1,-2,-1, 1, 2, 1,-1,-2,-1, 1  1/6 Hz
filtered   2, 1,-1,-2,-1, 1, 2, 1,-1,-2,-1, 1
winnowed   2,      -2,       2,      -2,
```

The amplitude spectrum of the original signal is illustrated in the following frequency-domain chart. Each frequency, shown along the horizontal, is represented by a fraction in which the numerator (0, 1, … 6) is the wave number, and the denominator (/12) is the period of the time-domain function in seconds. (The period happens to equal the number of points in this case, because the sample interval is 1 second). Function A appears as a single spectral line (of amplitude 2) at 1/2 Hz (6/12), which is also the Nyquist frequency; and function B is (also of amplitude 2) at 1/6 Hz (2/12):

```
Amplitude              |                      |
                       |                      |
Frequency-> 0/12 1/12 2/12 3/12 4/12 5/12 6/12
```

After low-pass filtering, the 1/2-Hz signal has been removed (its amplitude has been reduced to insignificance):

```
Amplitude              |
                       |
Frequency-> 0/12 1/12 2/12 3/12 4/12 5/12 6/12
```

After winnowing, the spectrum has lost the higher frequency portion (3/12 – 6/12) and a new Nyquist frequency is established at 2/12. In the following spectrum, information about the sample interval is preserved implicitly. But, it is not obvious that Nyquist is at 2/12:

```
Amplitude              |
                       |
Frequency-> 0/12 1/12 2/12
```

To make the Nyquist frequency obvious at the standard "1/2", the sample interval can be normalized to 1 second, thus the specific sample interval is no longer available as part of the spectrum. The number of points in the winnowed function is now shown explicitly in the denominator (/4) of each frequency; the wave numbers are in the numerators (0, 1, 2):

```
Amplitude              |
                       |
Frequency-> 0/4  1/4  2/4
```

The Basic Unwinnowing Process

The unwinnowing process can be understood as the inverse of the winnowing process. However, there is a caveat. A true inverse will completely undo the forward process. For example, the time-domain and the frequency-domain are inverse transforms of one another. But, the typical winnowing process destroys the higher frequency information from the original data stream and therefore cannot have an inverse. It would be possible to keep a copy of the higher frequencies that were removed before winnowing, and then add them back in during "unwinnowing", but that would be a nearly trivial exercise. It therefore is appropriate to pursue the unwinnowing process with the realization that an original (pre-winnowed) data stream is fictitious, even though it is used here as a tool for deriving, understanding, and describing unwinnowing.

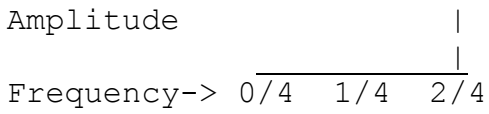
So, unwinnowing does not restore a previously winnowed data stream, but rather decreases the sampling interval (by inverse integral multiples) of an existing data stream in the smoothest possible way. Therefore, it is not necessary to have winnowed a data stream before unwinnowing is performed on it. One the other hand, if the data stream had been previously winnowed, it is important to realize that the unwinnowing process will not restore the higher frequency information. Rather, it will restore the original sampling interval but with the spectrum of the winnowed data stream.

The objective then of unwinnowing is to *restore the sampling interval* and the correct number of time-domain points from the original data stream *without altering the frequency spectrum* of the winnowed data stream and *without changing the function values* of the winnowed data points. This is done by replacing the intervening data points (points removed during winnowing) with interpolated values that have no power at frequencies above the winnowed Nyquist frequency. These points will not add any new spectral information to the data; they will maintain continuity in the greatest possible number of derivatives; and they will honor the information that remains from the winnowed data stream.

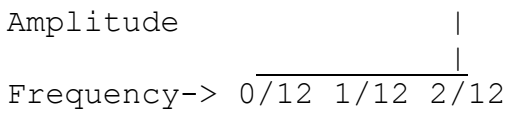
The basic unwinnowing process is fairly simple. It is nearly the reverse of the steps given by example for the winnowing process. The following unwinnowing example starts with the final winnowed profile from the winnowing example:

```
winnowed data stream
samp times 0,       3,       6,       9,      seconds
function   2,      -2,       2,      -2,
```

Using a Discrete Fourier Transform (DFT or FFT), change the data stream into a frequency spectrum. The spectrum has both real and complex components that can be divided into an amplitude and a phase spectrum. Only the amplitude spectrum is illustrated here:

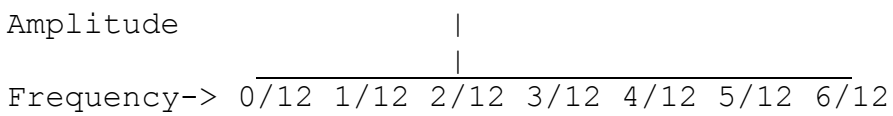


Determine the number of samples desired for the unwinnowed data stream. It should be an integral multiple of the number of samples in the winnowed data stream. In this case the winnow-factor is 3 and the number of samples is 4, so the number of points in the unwinnowed data stream will be 12. Convert the frequencies according to the new sample interval while keeping the wave numbers constant:



(Note: There is nothing about the DFT that would constrain the number of samples in the unwinnowed data stream to be an integral multiple of the number of samples in the winnowed data stream. This constraint is only given to meet the objective that *the function values of the winnowed data points not be changed*. It is apparent that, if this objective is ignored, modifications to the algorithm could be made that would allow any number of discrete data points to exist in the unwinnowed data stream.)

Fill out the new spectrum by appending additional zero-amplitude frequencies up to the new (unwinnowed) Nyquist:



Apply the inverse Discrete Fourier Transform (DFT; FFT only works with powers of 2 and 12 points is not a power of 2) to change the spectrum back into a data stream:

```
unwinnowed data stream
samp times 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11 seconds
function   2, 1,-1,-2,-1, 1, 2, 1,-1,-2,-1, 1
```

Here you can see that the original data stream is not quite restored because the higher frequencies (function A) had been irretrievably filtered out. However, the original filtered data stream is exactly restored. Also, notice that winnowed data points at times 0, 3, 6, and 9 seconds have remained intact; this will always be the case when the spectrum is appended with an integral multiple of the winnowed number of points. However, the intervening points are subject to additional constraints that will be discussed next.

Finally, it should now be apparent that this unwinnowing algorithm must be applied in post processing; it cannot be applied in real time. Also, the segment of the function sampled has a definite beginning and end that are not usually part of the random process that produced the function.

Data Stream Preparation

The Discrete Fourier Transform operates on the assumption that the time-domain function repeats endlessly from time equals minus infinity to plus infinity. This means that the transform will contain spectral information of an implied step from the function value at the last point in the time-domain data stream to the function value at the first point in the same data stream, in addition to all of the explicit variations within the data stream.

For this reason, before unwinnowing, the data stream must be prepared in such a way as to minimize the step, or to make a "step" that does not alter the basic spectrum of the random process that originally produced the data in the segment. In the example above, the signals, the sample intervals, and the number of sample points were chosen such that the implicit step exactly matched the function itself; and therefore, no particular data-stream preparations were necessary. However, a fortuitous convergence of this nature does not generally occur.

If data preparations are not performed, the spectrum that includes the implied step may differ from the spectrum of the random process that produced the segment of the data stream being unwinnowed. Additionally, because the unwinnowing process adds data points, the implied step will contain one or more of the new points; and these points will smoothly conform to the implied step. The absence of frequencies above the winnowed Nyquist forces the derivatives at the last and first points of the winnowed profile (unchanged points before and after the implied step) to be smooth. In turn, this leads to ringing that is greatest at the ends of the profile and damps toward the center. An interesting (yet unwanted) aspect of the ringing is that it occurs entirely among the new points added by the unwinnowing process. And, because the winnowed points remain unchanged, the ringing oscillates about them, as illustrated in figure 1.

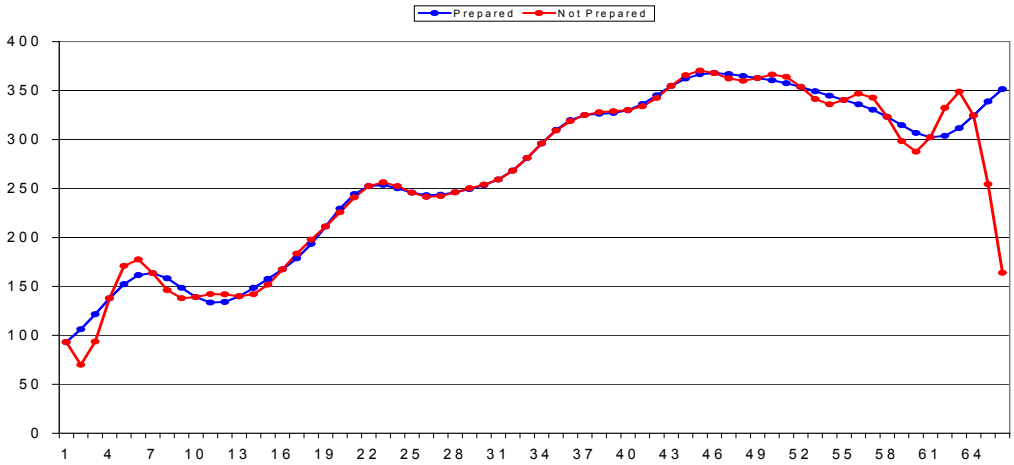


Figure 1. Ringing due to the implicit step in an unwinnowed profile that was not prepared before unwinnowing, compared to a profile that was properly prepared before unwinnowing. Notice that the winnowed points (1,4,7, …) are identical in both profiles and that the ringing affects only the additional points added by unwinnowing. Also, notice that additional points conform to the implicit step in the non-prepared profile; but in the prepared profile, they follow the apparent trend of the function.

The data preparations that this algorithm performs are: 1) removal of a least-squares-fit line, 2) Extension by a Burg Spectral Estimation algorithm (Claerbout, 1976), and 3) application of a Kaiser window (Harris, 1976).

Removal of the least-squares line is a well-known calculation and data-preparation step. If the line has a slope, the step will be reduced by the amount of difference between the two ends of the line segment. It also reduces the absolute values of the function values so that computational precision is increased.

Extension of the data stream (in both directions) is important because it moves the implicit step away from the data segment of interest. The ringing damps toward the center of the profile so that by extending an arbitrarily long distance, the ringing can be reduced to arbitrarily small levels.

The extension itself must not introduce any of its own ringing. This is accomplished through use of Burg Spectral Estimation, which produces an extension function based on the estimated spectrum of the random process that produced the winnowed profile. The data stream should represent a stationary random process (Bendat and Piersol, 1986) for the Burg Spectral Estimation to work properly. Non-stationary components of the data stream cannot be extended and will be ignored (filtered out) in the estimation process. Some "ringing" may be identifiable stemming from extension of non-stationary functions. (A real difficulty with non-stationary functions occurs if the original profile had one or more "steps" or "spikes". Quite unrelated to extension-caused ringing, these may lead to Gibb's phenomenon ringing in mid profile upon unwinnowing.)

In the unwinnowing algorithm, the necessary length of extension is calculated by an empirically derived ratio between the rms (root mean square) amplitude of the data stream and the rate that an rms-sized step (representing the power of the signal) will be damped. This calculation is implemented such that the ringing is damped to below –72 dB (decibels) within the end-points of the original data stream. The extension calculation includes reduction of the size of the rms step by the height of the windowing function at the extended end-points. (It is important to keep the extension length as short as possible commensurate with the specified ringing suppression because the extension algorithm is extremely time consuming.)

Finally, after the extension is performed, a windowing function is applied to minimize the actual size of the implied step. The Kaiser window with an alpha of 3.0 has been found empirically to perform best for most stationary data. Other windowing functions are acceptable but may require greater extension lengths to attain the same ringing suppression. The Kaiser is a full-length window that reduces the relative weighting of the waveforms away from the center of the profile. This may become a problem for long data segments with non-stationary components. The weighting issue can be circumvented by use of a *split* window such that the weighting is one (100%) throughout the length of the data profile with the windowing taper only on the extended portion. A split Riesz window (Harris, 1976) empirically produces 10 dB greater ringing on stationary functions. Consequently, the Kaiser window was "hard-wired" into the subroutines that implement this algorithm.

Description of the Unwinnowing Subroutine

The unwinnowing subroutine (UNWIN0.F) is written in standard Fortran 77 and automatically performs all necessary unwinnowing procedures, including the data preparations described above. It runs under an umbrella subroutine, called WINNOW.F, that performs either winnowing or unwinnowing according the setting of an input argument. The source code of the entire library of subroutines necessary for running the unwinnowing process is provided with the digital release of this report. However, the only entry-point is WINNOW.F; all the other subroutines are operationally transparent to the user. Complete instructions for use of WINNOW.F (as well as for all the others) 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.

Generalizing the Unwinnowing Algorithm

There is nothing about the DFT that would constrain the number of samples in the unwinnowed data stream to be an integral multiple of the number of samples in the winnowed data stream. This constraint is only given to meet the objective that *the function values of the winnowed data points not be changed*. It is apparent that, if this objective is ignored, modifications to the algorithm could be made that would allow any number of discrete data points to exist in the unwinnowed data stream. Implementation of this modification to the existing subroutine would be tedious, but possible. Information to help in this generalization may be found in the explanatory notes of subroutine DFTFTR.F.

References Cited

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

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

APPENDIX A

Brief Descriptions of Subroutines Needed for this Unwinnowing Algorithm

bold - Indicates subroutines that perform critical functions for this algorithm and which are included in subsequent appendixes.

unbold – Indicates subroutines that perform common or uncritical functions for this algorithm and which are included among the digital files of this report but not in the appendixes.

Entry Level Subroutine

winnow.f – Call this one for either winnowing or unwinnowing

2nd Level Subroutines

errfor.f – Produces a fatal error to stop processing for subroutine-determined non-system errors
unwin0.f – Handles the various major steps for unwinnowing
bwtdf.f – Butterworth time-domain filter with data preparations
factr.f – Factors an integer

3rd Level Subroutines

bwring.f – Estimates lengths needed to damp ringing of a butterworth filter from a given "step".
extend.f – Produces a extensions of various types, including Burg Spectral Estimation
varfqd.f – Finds the variance of a profile of data about a line
window.f – Applies or Removes windowing functions of various types including Kaiser and Riesz
dftftr.f – Produces a DFT (Discrete Fourier Transform) with the assumption of a real time domain (vanishing imaginary components) and complex frequency domain. Also, includes options for differing lengths of time and frequency domains.
bwlow.f – Basic Butterworth low-pass filter
trendr.f – Calculates, removes, or adds polynomial trends of various orders, including linear using lsqply.f

4th Level Subroutines

lsqply.f - Calculates, removes, or adds polynomial trends of various orders, including linear.
hann3.f – Applies or removes a Hanna window
kais3.f – Applies or removes a Kaiser window
fejer.f – Applies or removes a Triangular window
rclow.f – Basic RC low-pass filter
riesz.f – Applies or removes a Riesz window
burgdp.f – Finds the coefficients for Burg Spectral Estimation
hamm3.f – Applies or removes a Hamming window
tukey.f – Applies or removes a Cosine tapered window

5th Level Subroutine

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

APPENDIX B

Subroutine WINNOW.F

```
C
C_____________________________________________________________________
C
C     SUBROUTINE  W I N N O W
C_____________________________________________________________________
C
C SUBROUTINE WINNOW WINNOWS (DECIMATES) AND "UNWINNOWS" A PROFILE
C OF DATA.
C
C WINNOWING IS SIMPLY THE PROCESS OF REMOVING INTERVENING DATA
C POINTS AND COMPACTING THE DATA PROFILE.  AFTER WINNOWING, THE
C NEW PROFILE WILL CONTAIN ONLY EVERY NTH DATA POINT OF THE
C ORIGINAL.  THE EFFECT OF WINNOWING IS TO LOWER THE NYQUIST
C FREQUENCY OF THE DATA AND THEREFORE IMPOSE ALIASING ON THE NEW
C NYQUIST INTERVAL.  CONSEQUENTLY, BEFORE WINNOWING, A LOW-PASS
C FILTER IS APPLIED TO THE ORIGINAL DATA PROFILE TO REMOVE OR
C MINIMIZE ANY FREQUENCIES THAT ARE ABOVE THE NEW NYQUIST.
C
C TECHNICALLY, "UNWINNOWING" SHOULD PERFORM THE EXACT INVERSE OF
C WINNOWING, THAT IS, RESTORE THE ORIGINAL PROFILE.  HOWEVER, THE
C WINNOWING PROCESS ESSENTIALLY ERASES THE HIGHER FREQUENCIES
C MAKING THEM UNAVAILABLE FOR RESTORING.  THEREFORE, THE
C "UNWINNOWING" AS PERFORMED BY THIS SUBROUTINE RESTORES THE
C ORIGINAL SAMPLING INTERVAL AND NUMBER OF POINTS BUT WITH THE
C SPECTRUM OF ONLY THE WINNOWED PROFILE.
C
C
C THIS SUBROUTINE USES A LOW-PASS TIME-DOMAIN FILTER WITH A
C SQUARED BUTTERWORTH CHARACTERISTIC AND CUTOFF FREQUENCY 0.8053
C OF THE WINNOWED NYQUIST FREQUENCY (WC=0.8053*PI/KWIN).  THE
C CHARACTERISTICS OF THIS FILTER INCLUDE ZERO PHASE SHIFT, AND A
C TRANSIENT BAND WHOSE ROLLOFF CAN BE SPECIFIED WITH ARGUMENT
C NPOLE.  IF NPOLE=16, THE FOLLOWING ROLLOFF WILL OCCUR:
C
C   FREQUENCY      POWER(DB) POWER(RATIO)  COMMENTS
C 0.4686*PI/KWIN      0 DB = 1-(1/2**24)  REAL*4 PREC LIMIT
C 0.5000*PI/KWIN     -0 DB = 1-(1/2**21)  HALF NYQUIST
C 0.5110*PI/KWIN     -0 DB = 1-(1/2**20)
C 0.7834*PI/KWIN     -3 DB =    1/2       SQUARED HALF POWER
C 0.8053*PI/KWIN     -6 DB =    1/2**2    UNSQUARED CUTOFF FQ
C 1.0000*PI/KWIN    -60 DB =    1/2**20   NYQUIST
C 1.0444*PI/KWIN    -72 DB -    1/2**24   REAL*4 PREC LIMIT
C
C THE SQUARED BUTTERWORTH WITH NPOLE=16 (EFFECTIVE NUMBER OF
C POLES IS 32) HAS A ROLLOFF OF 12 DB PER POLE PER OCTAVE.
C HOWEVER, WITH THIS LARGE NUMBER OF POLES, RINGING CAN BECOME A
C SIGNIFICANT ISSUE.  SEE DISCUSSION UNDER ARGUMENT NPOLE.
C
C BEFORE FILTERING, THE ORIGINAL DATA PROFILE WILL BE EXTENDED
C WITH A BURG PREDICTION ALGORITHM.  THIS PROCESS DIMINISHES THE
C END EFFECTS (RINGING) LIKELY TO OCCUR WITH THE TIME-DOMAIN
C FILTER.  HOWEVER, IT SHOULD BE NOTED THAT IF STEPS OCCUR WITHIN
C THE PROFILE, RESULTANT RINGING MAY BE VISIBLE (AS RINGING)
C WITHIN THE WINNOWED PROFILE.
C
C
C INPUT ARGUMENTS:
```

```
C   KWIN   - INTEGER*4.  THE WINNOW FACTOR, AN INTEGER FACTOR BY
C            WHICH THE ORIGINAL SAMPLING INTERVAL IS TO BE
C            MULTIPLIED VIA REMOVAL OF INTERVENING DATA POINTS.
C            THE FIRST POINT OF THE ORIGINAL PROFILE WILL ALWAYS
C            BE KEPT, AS WELL AS EVERY KWINTH POINT AFTERWARDS.
C            FOR EXAMPLE, IF KWIN=3, POINTS 1,2,3, ... N OF THE
C            WINNOWED PROFILE WILL HAVE COME FROM POINTS 1,4,7,
C            ... 3N+1 OF THE ORIGINAL PROFILE.  THE TOTAL NUMBER
C            OF POINTS IN THE WINNOWED PROFILE WILL EQUAL
C            1+(NPTS-1)/KWIN (INTEGER DIVISION ASSUMED).
C
C            IF KWIN IS GREATER THAN 1, WINNOWING WILL BE DONE.
C            IF KWIN IS LESS THAN -1, UNWINNOWING WILL BE DONE.
C            IF KWIN IS -1, 0, OR 1, NOTHING WILL BE DONE.
C            (IF YOU SIMPLY NEED A PROFILE LOW-PASS FILTERED, SEE
C            SUBROUTINE BWTDF)
C
C            NOTE:  TO MINIMIZE WINNOWING COMPUTATION TIME FOR
C            LARGE PROFILES, KWIN SHOULD BE FACTORABLE WITH
C            SEVERAL SMALL FACTORS SUCH AS 2, 3, AND 5 AND LSTAGE
C            SHOULD BE SET TO 1.  CONVERSELY, IF KWIN IS ALLOWED
C            TO BE A LARGE PRIME NUMBER, THE TIME REQUIRED FOR
C            THE EXTENSION PART OF THE FILTERING PROCESS MAY
C            BECOME UNREASONABLY LARGE AS WILL THE NECESSARY
C            EXTENSION LENGTH.
C
C   LSTAGE - INTEGER*4.  DISABLES OR ENABLES STAGING DURING
C            WINNOWING.  IF LSTAGE IS 0, STAGING WILL NOT BE
C            USED; IF LSTAGE IS 1, STAGING WILL BE USED.  LSTAGE
C            IS NOT USED DURING UNWINNOWING.  WINNOWING IN STAGES
C            INVOLVES COMPOUNDED WINNOWINGS BY EACH OF THE
C            FACTORS OF KWIN.  BY WINNOWING IN STAGES, TIME IS
C            SAVED BUT ACCURACY IS COMPROMISED BECAUSE THE
C            REPEATED FILTERING CAUSED BY STAGING OVERPRINTS THE
C            UPPER HALF OF THE NYQUIST INTERVAL WITH VARIOUS
C            PARTS OF THE FILTER CURVE.  THE GREATEST ACCURACY IS
C            OBTAINED BY CHOOSING NO STAGING OR BY CHOOSING
C            WINNOW FACTORS THAT ARE PRIME NUMBERS.
C
C   NPOLE  - INTEGER*4.  THE NUMBER OF POLES DESIRED FOR THE
C            BUTTERWORTH FILTER DURING WINNOWING.  IF NPOLE=0,
C            THE FILTERING STEP WILL BE SKIPPED.  NPOLE IS NOT
C            USED DURING UNWINNOWING.  THE FILTER HAS BEEN
C            PRE-DEFINED AS A SQUARED TIME-DOMAIN BUTTERWORTH (NO
C            PHASE SHIFT) WITH A CUT-OFF FREQUENCY OF 0.8053 *
C            (THE NEW NYQUIST FREQ).  BECAUSE THE FILTER IS
C            SQUARED, THE EFFECTIVE NUMBER OF POLES IS TWICE
C            NPOLE.  THE FOLLOWING TABLE GIVES A -72 dB STOP
C            FREQUENCY IN OCTAVES RELATIVE TO THE CUTOFF
C            FREQ, DERIVED FROM NPOLE=LOG(64)/LOG(FSTOP/FCUT):
C
C                  NPOLE   STOP FQ    SUPPRESSION AT NYQUIST
C                         (OCTAVES)      (dB)   (AMPLITUDE)
C                     0       FILTERING IS SKIPPED FOR NPOLE=0
C                     1       6           -3.25    2/3
C                     2       3           -7.5     1/2
C                     4       1.5        -15       1/5
```

```
C                    8       0.75       -30       1/31
C                   16       0.375      -60       1/1000
C                   32       0.1875    -120       1/1000000
C
C            THE PRE-DEFINED ASPECTS OF THE FILTER WERE
C            ORIGINALLY DESIGNED TO PRODUCE A VERY SHARP CUTOFF
C            USING NPOLE=16.  HOWEVER, THIS PRODUCES SIGNIFICANT
C            RINGING IF THE FUNCTION CONTAINS ANY "SHARP
C            CORNERS".  BY REDUCING TO NPOLE=4 RINGING DECREASES
C            SUBSTANTIALLY, BUT THE SUPPRESSION OF FREQUENCIES
C            ABOVE NYQUIST IS COMPROMISED.  NEVERTHELESS, IN
C            THE NORMALLY-PINK POTENTIAL-FIELD FUNCTIONS, NPOLE=4
C            IS USUALLY SUFFICIENT.
C
C   NPTS   - INTEGER*4.  THE ORIGINAL, UNWINNOWED NUMBER OF
C            POINTS IN ARRAY YFNC.  NPTS SHOULD ALWAYS BE THE
C            UNWINNOWED NUMBER OF POINTS REGARDLESS OF WHETHER
C            THE PROFILE IS BEING WINNOWED (POSITIVE KWIN) OR
C            UNWINNOWED (NEGATIVE KWIN).  IF NPTS IS NEGATIVE
C            DURING WINNOWING, ONLY THE LOW-PASS FILTERING
C            STEP(S) OF THE WINNOWING PROCESS WILL BE PERFORMED;
C            THE ACTUAL WINNOWING WILL BE SKIPPED (IF UNDER THESE
C            CIRCUMSTANCES NDIM=0, THE NEW DIMENSION RETURNED
C            WILL BE GREATER THAN ITS VALUE HAD NPTS BEEN
C            POSITIVE).  SIMILARLY, IF NPTS IS NEGATIVE DURING
C            UNWINNOWING, THE UNWINNOWING WILL BE SKIPPED.
C            (HOWEVER, BECAUSE UNFILTERING IS NOT PERFORMED
C            DURING UNWINNOWING, NEGATIVE NPTS DURING UNWINNOWING
C            ACCOMPLISHES NOTHING; THE OPTION IS INCLUDED MERELY
C            FOR SYMMETRY).
C
C INPUT/OUTPUT ARGUMENTS:
C   YFNC   - REAL*8.  ARRAY OF DIMENSION NDIM CONTAINING NPTS OF
C            LEFT-JUSTIFIED DATA TO BE WINNOWED OR
C            1+(NPTS-1)/ABS(KWIN) POINTS TO BE UNWINNOWED.
C            NDIM MUST BE LARGER THAN NPTS TO ACCOMMODATE
C            SUBROUTINE COMPUTATIONS.  THE MINIMUM VALUE OF NDIM
C            MAY BE FOUND AS DESCRIBED IN NDIM.
C
C            NOTE:  THE NUMBER OF WINNOWED POINTS IS NEVER TAKEN
C            AS AN INPUT ARGUMENT.  SEE DESCRIPTION OF NPTW.
C
C   NDIM   -  INTEGER*4.  THE DIMENSION OF YFNC.  DURING WINNOWING
C            AND UNWINNOWING NDIM MUST BE GREATER THAN NPTS TO
C            ACCOMMODATE PROFILE EXTENSIONS NEEDED FOR FILTERING
C            AND TRANSFORMATIONS.  IF THE SPACE REQUIREMENTS IN
C            YFNC ARE GREATER THAN NDIM, AN ERROR WILL BE
C            GENERATED.  HOWEVER, IF NDIM IS GIVEN A VALUE OF
C            ZERO DURING THE CALL, THE MINIMUM NEEDED DIMENSION
C            OF YFNC WILL BE CALCULATED AND RETURNED IN NDIM
C            WITHOUT ALTERING YFNC.  THIS WILL BE DONE REGARDLESS
C            OF THE SIGN OF NPTS.  FOLLOWING IS AN EXAMPLE OF THE
C            USAGE OF NDIM:
C
C            integer*4 MXYFNC
C            parameter(MXYFNC = 200 000 )
C            real*8 yfnc(MXYFNC)
```

```
C            integer*4 ndim
C                .   .   .
C                .   .   .
C                .   .   .
C      C     FIND A MINIMUM VALUE FOR NDIM
C            ndim=0
C            call winnow(kwin,lstage,npole,npts, yfnc,ndim, nptw)
C            if(ndim.gt.MXYFNC) then
C              write(6,888) ndim
C        888   format('Size requirements of array yfnc are ',
C           &   ,'greater than its allocation.',/,'Please modify'
C           &   ,' parameter MXYFNC to be greater than ndim=',i7)
C              stop
C            endif
C      C
C      C     THE VALUE OF NDIM IS NOW ASSURED TO BE LARGE ENOUGH
C            call winnow(kwin,lstage,npole,npts, yfnc,ndim, nptw)
C
C OUTPUT ARGUMENT:
C   NPTW   - INTEGER*4.  THE NUMBER OF POINTS IN THE WINNOWED
C            PROFILE.  THE NUMBER OF WINNOWED POINTS IS
C            1+(ABS(NPTS)-1)/ABS(KWIN).  THIS VALUE CAN ALWAYS BE
C            CALCULATED EXACTLY FROM KWIN AND NPTS; BUT, THE
C            VALUE OF NPTS CANNOT BE CALCULATED EXACTLY FROM KWIN
C            AND THE WINNOWED NUMBER OF POINTS.  THEREFORE, THE
C            NUMBER OF WINNOWED POINTS IS NEVER TAKEN AS AN INPUT
C            ARGUMENT.  RATHER, IT IS PROVIDED HERE AS AN OUTPUT
C            ARGUMENT STRICTLY FOR INFORMATIONAL PURPOSES.  A
C            VALUE FOR NPTW IS PROVIDED WHENEVER A CALL IS MADE
C            TO THIS SUBROUTINE REGARDLESS OF THE VALUE OF NDIM
C            OR KWIN (ALTHOUGH THE VALUE OF KWIN INFLUENCES
C            NPTW).
C
C
C SUBROUTINE WINNOW WRITTEN BY ROB BRACKEN, USGS.
C FORTRAN 77, HP FORTRAN/9000, HP-UX RELEASE 11.0
C VERSION 1.0, 19970506, (INITIAL ROUTINE).
C VERSION 2.0, 20000201, (ADDED ARGUMENTS, LSTAGE AND NPOLE).
C
C
      subroutine winnow(kwin,lstage,npole,npts, yfnc,ndim, nptw)
C
C DECLARATIONS
C
C     INPUT ARGUMENTS
      integer*4 kwin,lstage,npole,npts
C
C     INPUT/OUTPUT ARGUMENTS
      real*8 yfnc(*)
      integer*4 ndim
C
C     OUTPUT ARGUMENT
      integer*4 nptw
C
C     WINNOWING STAGES
      integer*4 mxfac
      parameter(mxfac=60)
```

```
      real*8 winfac(mxfac)
      integer*4 nstage
C
C     BUTTERWORTH-FILTER PARAMETERS
C
C     KSQ=SQUARED FILTER (TO MAKE ZERO PHASE), 0=OFF, 1=ON
C     RINGDB=FILTER END-EFFECT RINGING IN DECIBELS
C     WCRNYQ=CUTOFF FREQUENCY RELATIVE TO NYQUIST
C     WSRWC=STOP BAND (-36 DB UNSQRD, -72 DB SQRD) FQ REL TO FCUT
C     (STOP BAND FREQ AT 1.29*FCUT PRODUCES A 16-POLE FILTER)
      integer*4 ksq
      real*8 ringdb,wcrnyq,wsrwc
      parameter(ringdb=-72.d0,ksq=1)
c      parameter(wcrnyq=0.8053d0,wsrwc=1.2968d0)
      parameter(wcrnyq=0.8053d0)
      real*8 fcut,fstop
C
C     UNWINNOWING
      real*8 fdavg,fdnyq
C
C
C FIND THE WINNOWED NUMBER OF POINTS, NPTW
C
      if(kwin.ne.0) then
        nptw=1+(jiabs(npts)-1)/jiabs(kwin)
      else
        nptw=jiabs(npts)
      endif
C
C DETERMINE WHETHER TO WINNOW OR UNWINNOW
C
      if(kwin.ge.2) goto 101
      if(kwin.le.-2) goto 102
      if(ndim.eq.0) ndim=jiabs(npts)
      goto 990
C
C WINNOW
C
C     DETERMINE THE FACTORS OF WINNOWING STAGES
  101 if(lstage.ge.1) then
C
C       STAGING IS DESIRED
        call factr(dflotj(kwin),winfac,nstage)
        if(nstage.gt.mxfac) call errfor(nstage,
     &   '(winnow)  nstage exceeds mxfac')
      else
C
C       STAGING IS NOT DESIRED
        nstage=3
        winfac(3)=dflotj(kwin)
      endif
C
C     DO WINNOWING IN STAGES FROM SMALLEST FACTORS TO LARGEST
      ndim0=ndim
      npts2=jiabs(npts)
      do j=3,nstage
C
```

```
C       GET THE NEXT LARGEST FACTOR IN KWIN
        kfac=jidnnt(winfac(j))
C
C       CHECK WHETHER TO FILTER
        if(npole.ge.1) then
C
C         FILTER THIS STAGE
          if(npts.gt.0.or.j.eq.3) then
            fcut=0.5d0*wcrnyq/dflotj(kfac)
          else
            fcut=fcut/dflotj(kfac)
          endif
          wsrwc=dexp(dlog(64.d0)/dble(npole))
          fstop=wsrwc*fcut
          ndim2=ndim0
          call bwtdf(fcut,fstop,ksq,ringdb,npts2, yfnc,ndim2)
        else
C
C         DO NOT FILTER THIS STAGE
          ndim2=ndim0
          if(ndim2.le.0) then ndim2=npts2
        endif
C
        if(ndim0.le.0) then
C
C         ADJUST THE DIMENSION
          if(ndim2.gt.ndim) ndim=ndim2
          if(npts.gt.0) npts2=1+(npts2-1)/kfac
        else if(npts.gt.0) then
C
C         WINNOW THE PROFILE
C
C         IF ONLY 2 POINTS LEFT IN PROFILE, TAKE THE AVERAGE
          if(npts2.eq.2.and.kfac.gt.1)
     &     yfnc(1)=(yfnc(1)+yfnc(2))/2.d0
C
          i=1
          do k=1+kfac,npts2,kfac
            i=i+1
            yfnc(i)=yfnc(k)
          enddo
          npts2=i
        endif
       enddo
C
C      CHECK THE NUMBER OF POINTS IN THE WINNOWED PROFILE
       if(ndim0.gt.0.and.npts.gt.0
     & .and.npts2.ne.1+(jiabs(npts)-1)/kwin)
     & call errfor(0,'(winnow)  Algorithm error')
       goto 990
C
C UNWINNOW
C
C      CHECK THE VALUE OF NDIM
  102 ndim2=0
      call unwin0(jiabs(kwin),jiabs(npts),yfnc(1),ndim2,
     & yfnc(2),fdavg,fdnyq)
```

```
      ndim0=jmax0(jiabs(npts),ndim2)
C     IF KRIESZ=1 IN UNWIN0, THIS STATEMENT MUST BE MODIFIED
      ndim2=((ndim2/jiabs(kwin)-1)/2)*2
      if(ndim.le.0) then
        ndim=ndim0+ndim2
        goto 990
      else
        if(ndim.lt.ndim0+ndim2) call errfor(ndim0+ndim2,
     &   '(winnow)  ndim is too small')
      endif

C     UNWINNOW THE PROFILE
      if(npts.gt.0)
     & call unwin0(jiabs(kwin),jiabs(npts),yfnc(1),ndim,
     & yfnc(ndim0+1),fdavg,fdnyq)
C
C EXIT PROCEDURE
C
  990 return
      end
```

APPENDIX C

Subroutine UNWIN0.F

```
C
C_____________________________________________________________________
C
C     SUBROUTINE  U N W I N 0
C_____________________________________________________________________
C
C SUBROUTINE UNWIN0 "UNWINNOWS" A PROFILE OF DATA.
C
C WINNOWING IS SIMPLY THE PROCESS OF REMOVING INTERVENING DATA
C POINTS AND COMPACTING THE DATA PROFILE.  SO TECHNICALLY,
C "UNWINNOWING" SHOULD PERFORM THE EXACT INVERSE OF WINNOWING,
C THAT IS, RESTORE THE ORIGINAL PROFILE.  HOWEVER, THE WINNOWING
C PROCESS ESSENTIALLY ERASES THE HIGHER FREQUENCIES MAKING THEM
C UNAVAILABLE FOR RESTORING.  THEREFORE, THE "UNWINNOWING" AS
C PERFORMED BY THIS SUBROUTINE RESTORES THE ORIGINAL SAMPLING
C INTERVAL AND NUMBER OF POINTS BUT WITH THE SPECTRUM OF ONLY THE
C WINNOWED PROFILE.
C
C IN THE UNWINNOWING PROCESS, ALL OF THE WINNOWED POINTS ARE
C RESTORED TO THE ORIGINAL PROFILE WITH THEIR EXACT WINNOWED
C VALUES.  THE FOURIER TRANSFORM METHOD (WHICH IS EMPLOYED BY
C THIS SUBROUTINE) INSURES THROUGH ORTHOGONALITY THAT THESE
C POINTS WILL IN FACT BE RESTORED.  HOWEVER, THE INTERVENING
C POINTS MUST BE INTERPOLATED AND THEIR VALUES DEPEND UPON
C SELECTION OF EXTENSION PROCESSES AND WINDOWING METHODS.  THESE
C HAVE BEEN OPTIMIZED FOR THIS SUBROUTINE BUT VERY SMALL ERRORS
C REMAIN.  TYPICALLY, MID-PROFILE ERRORS SHOULD BE WELL BELOW -72
C DB REFERENCED TO THE PROFILE VARIANCE.  END POINT ERRORS
C (WITHIN THE FIRST AND LAST 2*KWIN POINTS) HAVE BEEN FOUND AS
C LARGE AS -32 DB.
C
C THE FREQUENCY SPECTRUM OF THE UNWINNOWED PROFILE MATCHES VERY
C WELL TO THE ORIGINAL PROFILE IN THE LOWER HALF OF THE WINNOWED
C NYQUIST INTERVAL.  OF COURSE, THE UPPER HALF OF THE WINNOWED
C NYQUIST INTERVAL MAY DEPART SIGNIFICANTLY FROM THE ORIGINAL
C BECAUSE OF THE LOW-PASS FILTERING WHICH WOULD HAVE OCCURED
C BEFORE WINNOWING.
C
C
C INPUT ARGUMENTS:
C   KWIN   - INTEGER*4.  THE WINNOW FACTOR, AN INTEGER FACTOR BY
C            WHICH THE ORIGINAL SAMPLING INTERVAL WAS MULTIPLIED
C            VIA WINNOWING.  THE FIRST POINT OF THE ORIGINAL
C            PROFILE WAS KEPT, AS WELL AS EVERY KWINTH POINT
C            AFTERWARDS.  FOR EXAMPLE, IF KWIN=3, POINTS 1,2,3,
C            ... N OF THE WINNOWED PROFILE WILL HAVE COME FROM
C            POINTS 1,4,7, ... 3N+1 OF THE ORIGINAL PROFILE.  THE
C            TOTAL NUMBER OF POINTS IN THE WINNOWED PROFILE WILL
C            EQUAL 1+(NPTS-1)/KWIN (INTEGER DIVISION ASSUMED).
C            IF UNWINNOWING IS TO BE PERFORMED, ABS(KWIN) MUST BE
C            GREATER THAN 1.
C   NPTS   - INTEGER*4.  THE ORIGINAL, UNWINNOWED NUMBER OF
C            POINTS IN ARRAY YFNC.
C
C            NOTE:  THE NUMBER OF WINNOWED POINTS IS ALWAYS
C            1+(NPTS-1)/KWIN.  THIS VALUE IS NOT GIVEN AS AN
C            ARGUMENT BECAUSE ITS VALUE CAN ALWAYS BE CALCULATED
```

```
C            EXACTLY FROM KWIN AND NPTS; BUT, THE VALUE OF NPTS
C            CANNOT BE CALCULATED EXACTLY FROM KWIN AND THE
C            WINNOWED NUMBER OF POINTS.
C
C INPUT/OUTPUT ARGUMENTS:
C   YFNC   - REAL*8.  A SINGLE DIMENSIONED ARRAY CONTAINING
C            1+(NPTS-1)/KWIN DATA POINTS TO BE UNWINNOWED.  UPON
C            OUTPUT, YFNC WILL CONTAIN NPTS OF UNWINNOWED DATA.
C            YFNC DOUBLES AS A WORKSPACE DURING THE UNWINNOWING
C            PROCESS AND THE REQUIRED WORKSPACE (THE DIMENSION OF
C            YFNC) MAY EXCEED NPTS.  TO CALCULATE THE NECESSARY
C            DIMENSION, SEE NDIM.
C   NDIM  -  INTEGER*4.  THE DIMENSION OF YFNC.  NDIM CAN BE
C            GIVEN EITHER OF TWO VALUES, ZERO (0) OR A POSITIVE
C            VALUE THAT IS EQUAL TO OR GREATER THAN THE LARGEST
C            DIMENSION REQUIRED FOR YFNC.  IF THE POSITIVE VALUE
C            IS GIVEN, PROCESSING WILL OCCUR.  BUT IF THE VALUE
C            IS LESS THAN THE REQUIRE SPACE, AN ERROR WILL BE
C            GENERATED.  IF THE ZERO VALUE IS GIVEN, NO
C            PROCESSING WILL OCCUR DURING THAT CALL, BUT RATHER
C            THE MINIMUM REQUIRED DIMENSION OF YFNC WILL BE
C            CALCULATED AND RETURNED IN NDIM.
C
C OUTPUT ARGUMENTS:
C   FDH    - COMPLEX*16.  WORK ARRAY OF OF DIMENSION
C            (NDIM/KWIN-1)/2 WHEN NDIM HAS BEEN GIVEN A VALUE OF
C            0 AND UNWIN0 ALLOWED TO CALCULATE A LENGTH.  FDH IS
C            A HALF SPECTRUM (SEE SUBROUTINES DFTFTR.F AND
C            CNVSPC.F) OF THE WINNOWED, EXTENDED PROFILE.  THE
C            COMPLEX*16 HALF SPECTRUM IS DESIGNED TO REQUIRE NO
C            MORE REAL*8 ELEMENTS THAN ITS REAL*8 TIME-DOMAIN
C            COUNTERPART.  IF NECESSARY, FDH MAY BE COMBINED WITH
C            FDAVG AND FDNYQ TO RECONSTRUCT A FULL SPECTRUM OF
C            THE WINNOWED EXTENDED PROFILE.  HOWEVER, FOR THE C
C            PURPOSES OF UNWINNOWING, FDH IS SIMPLY A WORKSPACE.
C   FDAVG  - REAL*8.  WORK VARIABLE WHICH IS THE ZERO FREQUENCY
C            PART OF THE HALF SPECTRUM AS DESCRIBED ABOVE AND IN
C            SUBROUTINE DFTFTR.F.
C   FDNYQ  - REAL*8.  WORK VARIABLE WHICH IS THE NYQUIST
C            FREQUENCY PART OF THE HALF SPECTRUM AS DESCRIBED
C            ABOVE AND IN SUBROUTINE DFTFTR.F.
C
C
C SUBROUTINE UNWIN0 WRITTEN BY ROB BRACKEN, USGS, 19970506.
C HP-9000 SERIES 700/800 VERSION 1.0, 19970506
C
C
      subroutine unwin0(kwin,npts,yfnc,ndim,fdh,fdavg,fdnyq)
C
C DECLARATIONS
C
C     INPUT ARGUMENTS
      integer*4 kwin,npts
C
C     INPUT/OUTPUT ARGUMENTS
      real*8 yfnc(*)
      integer*4 ndim
```

```
C
C     OUTPUT ARGUMENTS
      complex*16 fdh(*)
      real*8 fdavg,fdnyq
C
C     UNWINNOWING AND DFT VARIABLES
      integer*4 nptw,kfac
      real*8 signi
C
C     LEAST SQUARES LINE REMOVAL
      integer*4 ktype,ncoef,nverse
      real*8 slope,rcept
      real*8 acoef(2)
C
C     FREQUENCY DEPENDENT VARIANCE
      real*8 varf
C
C     EXTENSION-AMOUNT CALCULATION
      real*8 rdb,profdb,thresh,step
      real*8 tenpct
      integer*4 jleft,kright,jmn,jlefw
C
C     EXTENSION PROCEDURES
      real*8 tparms(3)
      integer*4 just,npxt,npxw
C
C     WINDOWING FUNCTION
      real*8 alpha,wfrac
      integer*4 kdft
C
C     BUTTERWORTH-FILTER PARAMETERS
C
C     KSQ=SQUARED FILTER (TO MAKE ZERO PHASE), 0=OFF, 1=ON
C     RINGDB=FILTER END-EFFECT RINGING IN DECIBELS
C     WCRNYQ=CUTOFF FREQUENCY RELATIVE TO NYQUIST
C     WSRWC=STOP BAND (-36 DB UNSQRD, -72 DB SQRD) FQ REL TO FCUT
C     (STOP BAND FREQ AT 1.29*FCUT PRODUCES A 16-POLE FILTER)
      integer*4 ksq
      real*8 ringdb,wcrnyq,wsrwc
      parameter(ringdb=-72.d0,ksq=1)
      parameter(wcrnyq=0.8053d0,wsrwc=1.2968d0)
C
      real*8 fcut
      integer*4 npole
C
C     SELECT SPLIT RIESZ WINDOW (KRIESZ=1) OR KAISER (KRIESZ=0)
      integer*4 kriesz
      parameter(kriesz=0)
C
C DETERMINE WHETHER TO UNWINNOW
C
      if(jiabs(kwin).le.1) goto 990
C
C
C UNWINNOW
C
C     CONVERT THE WINNOW FACTOR
```

```
      kfac=jiabs(kwin)
C
C     FIND THE NUMBER OF POINTS IN THE WINNOWED PROFILE
      nptw=1+(npts-1)/kfac
C
C     FIND WINNOWED CUTOFF FREQ REFERENCED TO UNWINNOWED NYQUIST
      fcut=0.5d0*wcrnyq/dble(kfac)
      npole=16
C
C FIND COEFS OF A LEAST SQUARES LINE LATER TO BE REMOVED
C
      if(ndim.ge.1.or.ringdb.ge.0.d0) then
        ktype=1
        ncoef=2
        nverse=0
        call trendr(ktype,ncoef,acoef,nverse,-nptw,yfnc)
      endif
C
C FIND THE VARIANCE (POWER) OF THE WINNOWED PROFILE
C (SAME AS THE FREQ-DEPENDENT VARIANCE OF THE UNWINNOWED PROFILE)
C
      if(ringdb.lt.0.d0) then
        rdb=ringdb
C
C       RINGDB IS A RELATIVE THRESHOLD; EXTENSION LENGTHS WILL BE
C       CALCULATED BY BWRING SUCH THAT POWER IS RELATIVE TO THE
C       PROFILE VARIANCE.  ADJUSTMENT OF THE REQUESTED RINGING
C       POWER IS UNNECESSARY.
        profdb=0.d0
      else
        rdb=-ringdb
C
C       RINGDB IS AN ABSOLUTE THRESHOLD; BECAUSE EXTENSION
C       LENGTHS ARE CALCULATED BY BWRING RELATIVE TO PROFILE
C       VARIANCE, THE EFFECT OF VARIANCE MUST BE REMOVED FROM THE
C       REQUESTED RINGING POWER IN ORDER TO TIE IT TO THE
C       ABSOLUTE STANDARD (VARIANCE = 1/2).
C
C       (NOTE: INTERCEPT OCCURS AT I=0 NOT I=1)
        slope=acoef(2)
        rcept=acoef(1)
        varf=0.d0
        do i=1,nptw
          varf=varf+( yfnc(i)-(slope*dflotj(i)+rcept) )**2.d0
        enddo
        varf=varf/dflotj(nptw)
        profdb=10.d0*dlog10(2.d0*varf)
      endif
C
C FIND AMOUNT THE UNWINNOWED PROFILE SHOULD HAVE BEEN EXTENDED
C
C     FIND A THE RINGING THRESHOLD
      thresh=rdb-profdb
C
C     CONVERT THRESH TO AN AMPLITUDE FOR SUBR BWRING
      thresh=10.d0**(thresh/20.d0)
      step=1.d0
```

```
      call bwring(thresh,step, fcut,npole,ksq, jleft,kright)
C
C     ADJUST JLEFT TO EXTEND FAR ENOUGH THAT THE ENDS OF THE
C     PROFILE ARE HIGHER THAN 10% ON A KAISER WINDOW WITH ALPHA=3
      tenpct=0.1632d0
      jmn=dflotj(3*kfac+npts)*tenpct/(1.d0-2.d0*tenpct)+1.d0
      if(jleft.lt.jmn) jleft=jmn
C
C     ADJUST JLEFT TO THE WINNOWED & UNWINNOWED EXTENSION AMOUNTS
      jlefw=1+(jleft-1)/kfac
      jleft=jlefw*kfac
C
C     FIND THE WINNOWED AND UNWINNOWED NUMBER OF POINTS EXTENDED
      npxw=1+(jleft+npts+jleft-1)/kfac
      npxt=npxw*kfac
C
C     CHECK NDIM
      if(ndim.le.0) then
C
C       CALCULATE THE VALUE OF NDIM AND RETURN TO CALLING ROUTINE
        if(kriesz.eq.1) then
C
C         SPLIT RIESZ WINDOW INTERNALLY SELECTED
          ndim=jmax0(npts,npxw)
        else
C
C         KAISER WINDOW INTERNALLY SELECTED
          ndim=npxt
        endif
        goto 990
      endif
C
C REMOVE AN LSQ LINE FROM THE WINNOWED PROFILE USING COEFS
C CALCULATED EARLIER IN THIS SUBR
C
      ktype=1
      ncoef=2
      nverse=0
      call trendr(ktype,-ncoef,acoef,nverse,nptw,yfnc)
C
C EXTEND THE PROFILE WITH A BURG PREDICTION FILTER
C
C NOTE:  THE EXTENSION USED HAS EVERYTHING TO DO WITH HOW WELL
C THE RESTORED WINNOWED POINTS NEAR THE PROFILE ENDS WILL MATCH
C THE ORIGINAL.  HOWEVER, MATCHING THE ORIGINAL IS A BIT
C SUBJECTIVE BECAUSE FREQUENCIES ARE LOST GOING THROUGH THE
C WINNOWING STAGE.  GENERALLY, THE LONGER ROOT IN THE BURG
C EXTENSION IS BETTER.  BUT, THERE IS A TRADE-OFF WITH TIME ...
C AND THE BURG EXTENSION IS VERY SLOW.  BECAUSE OF THIS, THE
C COMPROMISE HAS BEEN TAKEN TO MAKE THE ROOT LENGTH EQUAL THE
C EXTENSION LENGTH.
C
C     BE SURE THE EXTENSIONS WILL FIT IN THE ALOTTED DIMENSION
      if(ndim.lt.npxw)
     & call errfor(npxw,
     & '(unwin0)  dimension ndim of array yfnc is too small')
C
```

```
C     SELECT BURG PREDICTION FILTER FOR EXTENSION
      ktype=7
C
C     DEPTH OF BURG FILTER COEFS = DEPTH OF EXTENSION
      tparms(1)=-1.d0
C     DEPTH OF BURG FILTER COEFS = LENGTH OF PROFILE
c      tparms(1)=0.d0
C     DEPTH OF BURG FILTER COEFS = VALUE OF TPARMS(2) AND (3)
c      tparms(1)=-3.d0
c      tparms(2)=2048.d0
c      tparms(3)=2048.d0
      just=jlefw+1
      call extend(ktype,tparms,just,nptw,npxw,yfnc)
C
C WINDOW THE PROFILE WITH EITHER A SPLIT RIESZ OR KAISER WINDOW
C
C NOTE:  WINDOWING WITH A SPLIT RIESZ PRODUCES ABOUT 10 DB MORE
C NOISE THAN THE KAISER THROUGHOUT THE PROFILE.  HOWEVER, THE
C SPLIT RIESZ WINDOW REMOVES THE DANGER OF DIFFERING WEIGHTS IN
C VARIOUS PARTS OF THE PROFILE.  SELECTION OF THE WINDOWING
C FUNCTION DOES NOT EFFECT THE TIME-DOMAIN ENDS.
C
      if(kriesz.eq.1) then
C
C       WINDOWING WITH A SPLIT RIESZ WINDOW FUNCTION (ALPHA=3.0)
        ktype=4
        alpha=2.5d0
        wfrac=dflotj(jleft)/dflotj(npxt)
      else
C
C       WINDOWING WITH A KAISER WINDOW FUNCTION (ALPHA=3.0)
        ktype=14
        alpha=3.0d0
        wfrac=1.d0
      endif
      kdft=1
      nverse=0
      call window(ktype,alpha,wfrac,kdft,nverse,npxw,yfnc)
C
C EXPAND THE WINNOWED EXTENDED PROFILE BACK TO THE UNWINNOWED
C EXTENDED PROFILE WHILE IMPLICITLY REMOVING EXTENSIONS
C
C NOTE:  DFTFDR IS THE SAME AS DFTFTR EXCEPT THAT IT CALCULATES
C THE WHOLE TIME-DOMAIN PROFILE INCLUDING THE PARTS THAT ARE TO
C BE REMOVED.  WHEREAS DFTFTR SAVES TIME, IMPLICITLY REMOVING
C THE EXTENSIONS BY NOT CALCULATING THEM.
C
C     BE SURE THE UNWINNOWED PROFILE WILL FIT IN THE ALOTTED DIM
      if(ndim.lt.npts)
     & call errfor(npts,
     & '(unwin0)  dimension ndim of array yfnc is too small')
C
      kscale=1
      signi=-1.d0
      call dftftr(kscale,
     & 1,npxw,npxw,npxw,signi,yfnc,fdh,fdavg,fdnyq)
c      call dftfdr(npxw,npxw,signi,yfnc,fdh)
```

```
      jb=jleft+1
      je=jleft+npts
      signi=+1.d0
      call dftftr(kscale,
     & jb,je,npxt,npxw,signi,yfnc,fdh,fdavg,fdnyq)
c      call dftfdr(npxt,npxw,signi,yfnc,fdh)
C
C REMOVE THE FULL-LENGTH KAISER WINDOW
C
      if(kriesz.ne.1) then
        if(ndim.lt.npxt) call errfor(ndim,
     &   '(unwin0)  ndim is less than npxt')
        ktype=0
        just=jleft+1
        call extend(ktype,tparms,just,npts,npxt,yfnc)
C
        ktype=14
        alpha=3.0d0
        wfrac=1.d0
        kdft=1
        nverse=1
        call window(ktype,alpha,wfrac,kdft,nverse,npxt,yfnc)
        ktype=0
        just=jleft+1
        call extend(ktype,tparms,just,npxt,npts,yfnc)
      endif
C
C UNWINNOW THE LSQ LINE AND REPLACE INTO UNWINNOWED PROFILE
C
C     IMPLICITLY UNWINNOW THE LSQ LINE BY CHANGING THE COEFS
      slope=acoef(2)
      rcept=acoef(1)
      acoef(2)=slope/dble(kfac)
      acoef(1)=rcept+slope-acoef(2)
C
C     REPLACE THE LSQ LINE
      ktype=1
      ncoef=2
      nverse=1
      call trendr(ktype,ncoef,acoef,nverse,npts,yfnc)
C
C
C EXIT PROCEDURE
C
  990 return
      end
```

APPENDIX D

Subroutine BWRING.F

```
C
C_____________________________________________________________________
C
C     SUBROUTINE  B W R I N G
C_____________________________________________________________________
C
C SUBROUTINE BWRING CALCULATES THE NUMBER OF SAMPLES IN A TIME
C DOMAIN PROFILE NEEDED TO DAMP RINGING FROM A BUTTERWORTH
C LOW-PASS FILTER TO A VALUE BELOW A GIVEN THRESHOLD.  NUMBERS
C MAY BE CALCULATED FOR EITHER UNSQUARED OR SQUARED TRANSFER
C FUNCTIONS.  (UNSQUARED ARE FOR A SINGLE PASS OF THE FILTER;
C SQUARED ARE FOR A FORWARD AND REVERSE PASS WHICH ELIMINATES
C PHASE SHIFTS).
C
C THE EQUATIONS USED FOR THIS CALCULATION HAVE BEEN DERIVED
C EMPIRICALLY FROM SAMPLE DATA USING A TIME-DOMAIN SQUARED
C BUTTERWORTH FILTER.  CALCULATIONS FOR A STANDARD BUTTERWORTH
C HAVE BEEN ESTIMATED FROM THE SQUARED DATA.
C
C INPUT ARGUMENTS:
C   THRESH - REAL*8.  THE THRESHOLD BELOW WHICH RINGING MAY BE
C            TOLERATED.  A TYPICAL VALUE OF THRESH MIGHT BE
C            1/4096 WHICH WOULD BE AN ABSOLUTE POWER OF -72 DB.
C            THRESH MUST NOT BE ZERO.
C   STEP   - REAL*8.  THE SIZE OF THE STEP CAUSING THE RINGING.
C            THIS VALUE IS THE EXPECTED PEAK VALUE OF THE WHOLE
C            PROFILE CALCULATED AS THE SQUARE ROOT OF TWO TIMES
C            THE PROFILE VARIANCE -> SQRT(2*VAR) = SQRT(2)*SIGMA.
C   FQC    - REAL*8.  THE CUTOFF FREQUENCY (-3 DB FOR UNSQUARED
C            AND -6 DB FOR SQUARED) WHERE 0.5 IS NYQUIST.  FQC
C            MUST NOT BE ZERO.
C   NPOLE  - INTEGER*4.  THE NUMBER OF POLES IN THE UNSQUARED
C            FILTER.  FOR AN UNSQUARED BUTTERWORTH, THE ROLL-OFF
C            IS ABOUT -6 DB PER OCTIVE PER POLE; FOR A SQUARED
C            BUTTERWORTH, ABOUT -12 DB.
C   KSQ    - INTEGER*4.  IF KSQ IS 0, THE NUMBERS FOR THE
C            UNSQUARED FILTER WILL BE CALCULATED.  IF KSQ IS 1,
C            THE SQUARED FILTER NUMBERS WILL BE FOUND.
C
C OUTPUT ARGUMENTS:
C   JLEFT  - INTEGER*4.  THE NUMBER OF DATA SAMPLES FROM THE LEFT
C            TO WHERE THE RINGING DAMPS BELOW THE SIZE OF THE
C            THRESHOLD.
C   KRIGHT - INTEGER*4.  THE NUMBER OF DATA SAMPLES FROM THE
C            RIGHT TO WHERE THE RINGING DAMPS BELOW THE SIZE OF
C            THE THRESHOLD.
C
C INPUT/OUTPUT ARGUMENTS:
C
C SUBROUTINE BWRING WRITTEN BY ROB BRACKEN, USGS.
C HP-9000 SERIES 700/800 UNIX FORTRAN 77.
C SUBROUTINE BWRING VERSION 1.0, 19970204.
C
C
      subroutine bwring(thresh,step, fqc,npole,ksq, jleft,kright)
C
C DECLARATIONS
```

```
C
C     INPUT ARGUMENTS
      real*8 thresh,step,fqc
      integer*4 npole,ksq
C
C     OUTPUT ARGUMENTS
      integer*4 jleft,kright
C
C     INTERNAL VARIABLES
      real*8 tos,fqc2,pole,ipl,ipr
C
C CHECK THRESHOLD AND STEP
C
      if(thresh.eq.0.d0)
     & call errfor(0,'(bwring)  thresh = 0')
      tos=1.d0
      if(step.ne.0.d0) then
        tos=dabs(thresh/step)
        if(tos.gt.1.d0) tos=1.d0
      endif
C
C CHECK LIMITS OF FREQUENCY
C
      if(fqc.eq.0.d0)
     & call errfor(0,'(bwring)  fqc = 0')
      fqc2=dabs(fqc)
      if(fqc2.gt.0.5d0) fqc2=0.5d0
C
C FIND WHETHER TO SQUARE THE XFER FUNCTION
C
      pole=dflotj(jiabs(npole))
      if(ksq.eq.0) pole=pole/2.d0
C
C FIND DAMPING VALUES
C
      jl=jidnnt((pole**1.4d0)/(10.58d0*fqc2))
      kright=jidnnt((-pole*dlog10(tos))/(4.23d0*fqc2))
      if(kright.lt.jl) kright=jl
      jleft=kright-jl
C
      if(tos.lt.1.d0) then
        ipl=jidnnt(1.d0/(2.d0*fqc2))
        ipr=jidnnt(pole/(9.52d0*fqc2))+ipl
        if(jleft.lt.ipl) jleft=ipl
        if(kright.lt.ipr) kright=ipr
      endif
C
C
C EXIT PROCEDURE
C
  990 return
      end
```

APPENDIX E

Subroutine DFTFTR.F

```
C
C_____________________________________________________________________
C
C     SUBROUTINE  D F T F T R
C_____________________________________________________________________
C
C SUBROUTINE DFTFTR 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.  DFTFTR IS SIMILAR TO
C DFTFDR BUT HAS AN ADDITIONAL FEATURE THAT ALLOWS THE TIME
C DOMAIN TO BE TRUNCATED IMPLICITLY AT THE ENDS THUS GIVING A
C FASTER TRANSFORM WHEN TRUNCATION IS CONTEMPLATED.
C
C DIFFERING LENGTH DOMAINS ARE INTERPRETED BY DFTFTR 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 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, DFTFTR.  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 DFTFTR 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, DFTFTR PRODUCES THE FREQUENCY TRUNCATION INTERNALLY.)
C
C
C
C
C DFTFTR 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 DFTFTR 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 DFTFTR 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 AN ADDITIONAL FUNCTION OF DFTFTR IS THAT IT ALLOWS THE TIME
C DOMAIN TO BE TRUNCATED IMPLICITLY USING THE TWO ARGUMENTS JB
C AND JE.  THIS TRUNCATION IS INTENDED PURELY AS A TIME SAVINGS
C WHICH IS USEFUL ONLY IF THE TIME DOMAIN IS TO BE TRUNCATED
C IMMEDIATELY AFTER INVERSE TRANSFORM OR EXTENDED WITH ZEROS
C IMMEDIATELY BEFORE FORWARD TRANSFORM.  IF THIS FEATURE WAS NOT
C AVAILABLE THE DFT WOULD HAVE TO SPEND TIME FINDING OR USING
C VALUES THAT DO NOT CONTRIBUTE TO THE RESULT.
C
C A SPECIAL FEATURE OF DFTFTR IS THAT THE FREQUENCY DOMAIN DOES
C NOT INCLUDE THE NEGATIVE FREQUENCIES AND ZERO & NYQUIST
C FREQUENCIES ARE PASSED THROUGH DEDICATED ARGUMENTS.  BECAUSE
C THE TIME DOMAIN IS REAL, THE NEGATIVE FREQUENCIES DO NOT ADD
C ANY NEW INFORMATION.  THEREFORE, DROPPING THE NEGATIVE
C FREQUENCIES DOES NOT ALTER THE FREQUENCY DOMAIN WHILE ALLOWING
C IT TO TAKE THE SAME ARRAY SPACE AS THE TIME DOMAIN (RATHER THAN
C TWICE).
C
C
C INPUT ARGUMENTS:
C   KSCALE - INTEGER*4.  THE TYPE OF SCALE FACTOR TO BE USED:
C               KSCALE     DESCRIPTION
C                  0       FFT SCALE FACTOR
C                  1       DFT SCALE FACTOR
C            THE FFT SCALE FACTOR TRIES TO MAKE THE TIME AND
C            FREQUENCY DOMAINS MATHEMATICALLY EQUIVALENT BY
C            SCALING THE SQUARE ROOT OF THE NUMBER OF POINTS ONCE
C            ON FORWARD TRANSFORM AND AGAIN ON INVERSE TRANSFORM.
C            THE DFT SCALE FACTOR PRODUCES A FREQUENCY DOMAIN
C            WITH TRUE AMPLITUDES ON FORWARD TRANSFORM AND
C            THEREFORE DOES NO SCALING ON INVERSE TRANSFORM.  THE
C            DFT SCALE FACTOR IS GENERALLY MORE PHYSICALLY
C            MEANINGFUL AND IS MORE LIKELY TO REMAIN MEANINGFUL
C            WHEN TRUNCATIONS AND EXTENSIONS ARE BEING USED.
C   JB     - INTEGER*4.  THE BEGINNING LOCATION IN THE TIME
C            DOMAIN OF THE INTERVAL (JB,JE) FROM WHICH
C            CALCULATIONS WILL BE MADE.  ANY VALUES OUTSIDE OF
C            THIS INTERVAL WILL HAVE AN EFFECTIVE VALUE OF ZERO.
C            THE VALUE OF JB MUST BE WITHIN THE RANGE 1 TO NRX
C            WHERE NRX IS THE NUMBER OF POINTS IN THE TIME
C            DOMAIN.  UPON INVERSE TRANSFORM ALL POINTS BEFORE JB
C            WILL BE IMPLICITLY TRUNCATED MAKING THE TRUNCATED
C            ARRAY LOCATION, RX(1) HAVE THE VALUE OF THE
C            UNTRUNCATION ARRAY LOCATION, RX(JB).
C   JE     - INTEGER*4.  THE ENDING LOCATION IN THE TIME DOMAIN
C            OF THE INTERVAL (JB,JE) FROM WHICH CALCULATIONS WILL
C            BE MADE.  ANY VALUES OUTSIDE OF THIS INTERVAL WILL
C            HAVE AN EFFECTIVE VALUE OF ZERO.  THE VALUE OF JE
C            MUST BE GREATER THAN OR EQUAL TO JB AND WITHIN THE
C            RANGE 1 TO NRX WHERE NRX IS THE NUMBER OF POINTS IN
```

```
C            THE TIME DOMAIN.  UPON INVERSE TRANSFORM ALL POINTS
C            AFTER JE WILL BE IMPLICITLY TRUNCATED.  THE
C            TRUNCATED ARRAY LOCATION, RX(JE-JB+1) HAVE THE VALUE
C            OF THE UNTRUNCATION ARRAY LOCATION, RX(JE).
C   NRX    - INTEGER*4.  THE NUMBER OF TIME-DOMAIN PTS.  NRX IS
C            ALWAYS USED AS THE EFFECTIVE LENGTH OF THE
C            TIME-DOMAIN REGARDLESS OF THE THE VALUES IN JB AND
C            JE.  BUT, UPON FORWARD TRANSFORM, NRX IS THE ACTUAL
C            NUMBER OF POINTS IN AND THE DIMENSION OF ARRAY RX;
C            UPON INVERSE TRANSFORM THE DIMENSION IS REDUCED TO
C            JE-JB+1 BECAUSE OF IMPLICIT TRUNCATION.
C   NCY    - INTEGER*4.  THE NUMBER OF FREQUENCY-DOMAIN PTS.  NCY
C            IS ALWAYS USED AS THE EFFECTIVE LENGTH OF THE
C            FREQUENCY-DOMAIN.  BUT, BECAUSE NEGATIVE FREQUENCIES
C            ARE NOT USED, NCY IS ROUGHLY TWICE THE ACTUAL NUMBER
C            OF POINTS IN ARRAY CYH.  THE ACTUAL DIMENSION OF CYH
C            IS (NCY-1)/2.
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            THE DIMENSION OF RX IS NRX DURING FORWARD TRANSFORM;
C            IT IS REDUCED TO JE-JB+1 DURING INVERSE.
C   CYH    - COMPLEX*16.  ARRAY OF DIMENSION (NCY-1)/2 CONTAINING
C            A HALF SPECTRUM OF THE COMPLEX FREQUENCY-DOMAIN
C            DATA.  IF SIGNI IS ZERO OR NEG (FORWARD TRANSFORM),
C            CYH WILL BE AN OUTPUT ARGUMENT.  IF SIGNI IS GREATER
C            THAN ZERO (INVERSE TRANSFORM), CYH WILL BE AN INPUT
C            ARGUMENT.  THE HALF SPECTRUM IS STRUCTURED AS
C            FOLLOWS:  CYH(1)=WAVE#1, CYH(2)=WAVE#2, ...
C            CYH((NCY-1)/2)=HIGHEST POSITIVE FREQUENCY THAT IS
C            NOT NYQUIST.  ZERO FREQUENCY (WAVE#0) AND NYQUIST
C            (WAVE#(NCY/2)) ARE BOTH REAL AND SEPARATED TO THE
C            ARGUMENTS RYA (ZERO) AND RYN (NYQUIST).  NOTE THAT
C            IF NCY IS ODD, NYQUIST DOES NOT EXIST.  (A FULL
C            SPECTRUM WOULD CONTAIN NCY EVENLY-SPACED FREQUENCIES
C            IN AN ASCENDING SEQUENCE FROM ZERO TO NYQUIST AND ON
C            UP TO 1 POINT BEFORE TWICE NYQUIST.)
C   RYA    - REAL*8.  THE ZERO FREQUENCY (WAVE#0 OR THE AVERAGE
C            VALUE).  THE VALUE IN RYA NORMALLY APPEARS AS A
C            COMPLEX NUMBER IN ARRAY LOCATION 1 OF A FULL
C            SPECTRUM.  HOWEVER, HERE IT IS A SEPARATE ARGUMENT
C            TO INSURE THAT THE HALF SPECTRUM HAVE A DIMENSION NO
C            LARGER THAN HALF OF THE FULL SPECTRUM.
C   RYN    - REAL*8.  THE NYQUIST FREQUENCY (WAVE#(NCY/2) OR THE
C            HIGHEST POSSIBLE FREQUENCY).  THE VALUE IN RYN
C            NORMALLY APPEARS AS A COMPLEX NUMBER IN ARRAY
C            LOCATION NCY/2+1 OF A FULL SPECTRUM.  HOWEVER, HERE
C            IT IS A SEPARATE ARGUMENT TO INSURE THAT THE HALF
C            SPECTRUM HAVE A DIMENSION NO LARGER THAN HALF OF THE
C            FULL SPECTRUM.  NOTE THAT NYQUIST ONLY EXISTS IF NCY
C            IS EVEN.  IF NCY IS ODD OR NCY IS GREATER THAN NRX,
```

```
C            THE VALUE IN RYN WILL BE ZERO.
C
C SUBROUTINE DFTFTR WRITTEN BY ROB BRACKEN, USGS, 19970613.
C HP-9000 SERIES 700/800 UNIX VERSION 1.0, 19970613.
C
C
      subroutine dftftr(kscale,
     &                  jb,je, nrx,ncy,signi,rx,cyh,rya,ryn)
C
C DECLARATIONS
C
C     INPUT ARGUMENTS
      integer*4 kscale
      integer*4 jb,je,nrx,ncy
      real*8 signi
C
C     INPUT/OUTPUT ARGUMENTS
      real*8 rx(*)
      complex*16 cyh(*)
      real*8 rya,ryn
C
C     INTERNAL VARIABLES AND CONSTANTS
      complex*16 ci,cw0,cwk,cwj,csum
      real*8 rsum
      real*8 pi,dscale
      real*8 powmin
      parameter(pi = 3.1415 92653 58979 32384 62643)
      integer*4 ksc2
C
C
C SET UP INTERNAL SCALE TYPE
C
C     FFT TYPE (KSC2=0) OR DFT TYPE (KSC2=1)
      ksc2=0
      if(kscale.ge.1) ksc2=1
C
C CALCULATE DISCRETE FOURIER TRANSFORM WITH REAL TIME DOMAIN,
C COMPLEX HALF SPECTRUM, HIGH FREQUENCY TRUNCATION/EXTENSION, AND
C IMPLICIT SPECIFIABLE TIME-DOMAIN TRUNCATION.
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     CALCULATE THE DISCRETE KERNEL OF INCREMENTAL NORMALIZED
C     ANGULAR FREQ, CW0 FOR FREQUENCY DOMAIN TRUNCATION/EXTENSION
      ci=dcmplx(0.d0,1.d0)
      cw0=cdexp(-ci*2.d0*pi/dflotj(nrx))
C
C     ADJ BEG & END TIM-DOM LOCS (JB, JE) INTO INTERVAL (1,NRX)
      jb2=jb
      je2=je
      if(jb2.lt.1) jb2=1
      if(je2.lt.1) je2=1
      if(jb2.gt.nrx) jb2=nrx
      if(je2.gt.nrx) je2=nrx
      if(je2.lt.jb2) then
```

```
        jhold=je2
        je2=jb2
        jb2=jhold
      endif
C
C     DETERMINE WHETHER TO PERFORM FORWARD OR INVERSE TRANSFORM
      if(signi.gt.0.d0) goto 101
C
C
C FORWARD TRANSFORM (TIME DOMAIN TO FREQUENCY DOMAIN)
C
C     INITIALIZE MINIMUM POWER CHECKER
c      powmin=1.1d+38
C
C     SET UP INTRINSIC DFT TYPE OF SCALE FACTOR
c      dscale=1.d0/dflotj(je2-jb2+1)
      dscale=1.d0/dflotj(nrx)
C
C     CHANGE SCALE FACTOR IF FFT TYPE IS DESIRED
      if(ksc2.eq.0) dscale=dsqrt(dscale)
C
C     FIND THE AVERAGE VALUE (ZERO FREQUENCY OR WAVENUMBER ZERO)
      rsum=0.d0
      do j=jb2,je2
        rsum=rsum+rx(j)
      enddo
      rya=rsum*dscale
C
C     INITIALIZE NYQUIST VARIABLE
      ryn=0.d0
C
C     FIND THE NUMBER OF FREQS TO CALCULATE (INTERVAL (ZERO,NYQ))
      nfq=jmin0(nrx,ncy)/2+1
C
C     FIND FREQUENCIES:  WAVENUMBER 1 UP THROUGH NYQUIST
      cwk=cw0
      do k=2,nfq
        csum=dcmplx(0.d0,0.d0)
        cwj=cwk**dflotj(jb2-1)
        do j=jb2,je2
          csum=csum+rx(j)*cwj
          cwj=cwj*cwk
        enddo
        cwk=cwk*cw0
c        if(cdabs(csum).lt.powmin) powmin=cdabs(csum)
C
C       SCALE AND PUT IN LEFT HALF (AVG IS NOT IN CYH(1))
        if(k.lt.nfq) cyh(k-1)=csum*dscale
      enddo
      k=k-1
C
C     MAKE ADJUSTMENTS AT AND ABOVE THE NYQUIST FREQUENCY
      nyqck=(nfq-1)*2
      if(ncy.gt.nrx) then
C
C       IF CSUM IS THE OLD NYQUIST, SPLIT IT
        if(nyqck.eq.nrx) csum=csum/2.d0
```

```
        cyh(k-1)=csum*dscale
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*dscale/4096.d0
c        powmin=powmin*dscale/2.d0
        do i=k,(ncy-1)/2
          cyh(i)=dcmplx(powmin,0.d0)
        enddo
      else
C
C       IF CSUM IS NEW NYQUIST, COMBINE WITH ITS REFLECTION
        if(nyqck.eq.ncy) then
          if(ncy.lt.nrx) csum=csum+dconjg(csum)
          ryn=dreal(csum)*dscale
        else
          cyh(k-1)=csum*dscale
        endif
      endif
C
      goto 990
C
C
C INVERSE TRANSFORM (FREQUENCY DOMAIN TO TIME DOMAIN)
C
C     INVERT THE KERNEL
  101 cw0=dconjg(cw0)
C
C     SET UP INTRINSIC DFT TYPE OF INVERSE SCALE FACTOR ( = 1 )
      dscale=1.d0
C
C     CHANGE INVERSE SCALE FACTOR IF FFT TYPE IS DESIRED
c      if(ksc2.eq.0) dscale=1.d0/dsqrt(dflotj(je2-jb2+1))
      if(ksc2.eq.0) dscale=1.d0/dsqrt(dflotj(nrx))
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=cw0**dflotj(jb2-1)
      do j=jb2,je2
        rsum=0.d0
C
C       ADD IN FREQS WITH WAVENUMBERS 1 THROUGH ALMOST NYQUIST
        cwk=cwj
        do k=1,kbn-1
          rsum=rsum+dreal(cyh(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((cyh(k)+dconjg(cyh(k)))*cwk)
          else
            rsum=rsum+dreal(ryn*cwk)
          endif
        endif
C
C       ADD IN AVERAGE
        rsum=rsum+rya
C
C       SCALE THE SUM
        rx(j-jb2+1)=rsum*dscale
      enddo
C
C EXIT PROCEDURE
C
  990 return
      end
```

APPENDIX F

Subroutine KAIS3.F

```
C
C_____________________________________________________________________
C
C     SUBROUTINE  K A I S 3
C_____________________________________________________________________
C
C SUBROUTINE KAIS3 REMOVES OR APPLIES A KAISER-BESSEL WINDOW IN
C AN EVENLY SPACED ARRAY OF REAL DATA.
C
C INPUT ARGUMENTS:
C   ALPHA  - REAL*8.  DETERMINES THE INTENSITY OF THE WINDOW.
C            ALPHA = 0.0 DOES NO WINDOWING; THE BEST VALUE IS
C            ALPHA = 3.0; VALUES OF ALPHA > 3.5 ARE NOT
C            RECOMMENDED.
C   NREM   - INTEGER*4.  DETERMINES WHETHER TO REMOVE (NREM = 1)
C            OR APPLY (NREM = 0) THE WINDOW.
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   NIO    - INTEGER*4.  THE NUMBER OF DATA POINTS IN ARRAY YIO.
C
C INPUT/OUTPUT ARGUMENT:
C   YIO    - REAL*8.  ARRAY CONTAINING THE DATA TO BE WINDOWED.
C
C
C SUBROUTINE KAISER WRITTEN BY MIKE WEBRING, USGS, DATE?
C MODIFIED TO KAIS2 BY ROB BRACKEN, USGS, 11OCT91.
C MODIFIED TO KAIS3 BY ROB BRACKEN, USGS, 20AUG96.
C HP-9000 SERIES 700/800 UNIX VERSION 2.1, 9AUG96.
C HP-9000 SERIES 700/800 UNIX VERSION 3.0,20AUG96.
C
C
      subroutine kais3(alpha,nrem,kdft,nio,yio)
C
C DECLARATIONS
C
C     INPUT ARGUMENTS
      real*8, alpha
      integer*4 nrem,kdft,nio
C
C     INPUT/OUTPUT ARGUMENT
      real*8 yio(*)
C
C     INTERNAL VARIABLES
      integer*4 nwp,jdmax
      real*8 half,pi,pial,pialx,xd,winval,zmbes5,zmpial
      parameter(pi = 3.1415 92653 58979 32384 62643)
C
C INITIALIZATIONS
C
C     EFFECTIVE NUMBER OF WINDOW POINTS
      nwp=nio
      if(kdft.ge.1) nwp=nwp+1
C     THE MAXIMUM DISTANCE FROM THE LEFT MOST PROFILE POINT
```

```
      jdmax=nwp/2-1
C     THE EXACT LENGTH OF HALF THE PROFILE
      half=dflotj(nwp-1)/2.d0
C     PI * ALPHA
      pial=pi*alpha
      zmpial=zmbes5(pial)
C
C WINDOW ONE POINT AND IT'S REFLECTION ON EACH LOOP
C
      do 701 jd=0,jdmax
C
C       THE DISTANCE TRANSLATED TO XD=(n/(N/2))
        xd=1.d0-dflotj(jd)/half
C
C       THE ARGUMENT OF THE BESSEL FUNCTION IN THE NUMERATOR
        pialx=pial*dsqrt(1.d0-xd*xd)
C
        if(nrem.le.0) then
C
C         THE KAISER-BESSEL WINDOW VALUE
          winval=zmbes5(pialx)/zmpial
        else
C
C         THE KAISER-BESSEL WINDOW INVERSE VALUE
          winval=zmpial/zmbes5(pialx)
        endif
C
C       WINDOW POINT NUMBERS JD+1 AND NWP-JD
        yio(jd+1)=yio(jd+1)*winval
        if(jd.gt.0.or.kdft.le.0) yio(nwp-jd)=yio(nwp-jd)*winval
  701 continue
C
C EXIT PROCEDURE
C
  990 return
      end
```

APPENDIX G

Subroutine BURGDP.F

```
C
C_____________________________________________________________________
C
C     SUBROUTINE  B U R G D P
C_____________________________________________________________________
C
C SUBROUTINE BURGDP FINDS THE DOUBLE-PRECISION COEFFICIENTS OF A
C BURG ERROR-PREDICTION FILTER (UNIT SPAN).  THE TIME SERIES MUST
C BE OF DOUBLE-PRECISION DATA TYPE.  IF PROCESSING COMPLEX DATA,
C USE SUBROUTINE BURGDC.
C
C THE BURG ERROR-PREDICTION FILTER IS A REVERSIBLE MINIMUM-PHASE
C TIME-DOMAIN FILTER WITH A ZERO OUTPUT.  IT'S PREDICTION
C CHARACTERISTIC RESULTS FROM HOLDING THE FIRST COEFFICIENT TO
C ONE DURING DERIVATION.  ALL OF THE OTHER COEFFICIENTS OPERATE
C ON THE DATA TO CANCEL THE FIRST POINT.
C
C THERE ARE THREE PRIMARY USES OF THE BURG FILTER.
C
C A FIRST USE IS TO EXTEND A SHORT DATA PROFILE WITH ARTIFICIAL
C DATA OF A SIMILAR FREQUENCY SPECTRUM.  TO DO THIS, EXTEND ONE
C POINT AT A TIME ACCORDING TO: X(N+1) = -(X(N)*A(2)+X(N-1)*A(3)+
C X(N-2)*A(4)+ ... ).  THEN SLIDE THE FILTER COEFFICIENTS ONE
C POINT TO X(N+1) AND ITERATE TO THE DESIRED LENGTH.  THIS
C OPERATION MAY BE DONE IN BOTH DIRECTIONS.
C
C NOTE THAT THE BURG ALGORITHM DISCOVERS THE TRUE FREQUENCY
C SPECTRUM SO ACCURATELY THAT IT CAN BE USED FOR EXTENDING
C PROFILES TO MANY TIMES THEIR ORIGINAL LENGTH BEFORE FOURIER
C TRANSFORMING.  IN FACT, IF USED IN CONJUNCTION WITH WINDOWING,
C EXTREMELY CLOSE SPECTRAL PEAKS CAN BE SEPARATED THAT OTHERWISE
C WOULD HAVE BEEN BLURRED BY THE WINDOW OR IT'S SIDE LOBES.
C
C A SECOND USE IS TO FIND THE SPECTRAL CONTENT OF A DATA PROFILE
C WITHOUT THE CLUTTERING END EFFECTS.  THE FILTER IS DERIVED SUCH
C THAT IT ASSUMES THE FUNCTION TO CONTINUE "IN A REASONABLE WAY"
C INSTEAD OF TRUNCATING WITH ZEROS OR REPEATING THE INTERVAL.
C THE SPECTRUM OF THE FUNCTION IS THE INVERSE OF THE FILTER
C SPECTRUM.  TO FIND THE FUNCTION SPECTRUM, EXTEND THE FILTER
C COEFFICIENTS WITH ZEROS TO A REASONABLE LENGTH; THEN FOURIER
C TRANSFORM THE EXTENDED FILTER COEFFICIENTS INTO THE FREQUENCY
C DOMAIN; FINALLY, INVERT EACH FREQUENCY.  TO PERFORM THE
C FREQUENCY INVERSION AND PRODUCE A NATURAL-LOG POWER SPECTRUM IN
C ONE EASY STEP, SIMPLY TAKE THE COMPLEX LOG AND MULTIPLY BY
C NEGATIVE 2.  (NOTE:  THIS USE IS NOT EASILY INTERPRETED BECAUSE
C THE SPECTRUM DOES NOT EVEN RESEMBLE A SPECTRUM FROM A FOURIER
C TRANSFORM.)
C
C A THIRD USE IS TO PREDICT FUTURE VALUES IN A DATA STREAM.  IF
C THE DATA STREAM IS STATIONARY, ANOMALOUS VALUES WILL NOT BE
C PREDICTED CORRECTLY AND CAN THEREFORE BE ISOLATED.  IT MAY BE
C REASONABLE TO AUGMENT THIS TECHNIQUE BY THE USE OF AN ADAPTIVE
C FILTER SUCH AS THE WIDROW ALGORITHM (SEE CLAERBOUT CHAPTER 7-3,
C PAGES 136-139).
C
C A COMPLETE DESCRIPTION OF THE BURG FILTER (INCLUDING A DETAILED
C DERIVATION) MAY BE FOUND IN NOTES ENTITLED BURG SPECTRAL
```

```
C ESTIMATION, 6NOV96.  THE FOUNDATIONAL MATERIAL WAS OBTAINED
C FROM "FUNDAMENTALS OF GEOPHYSICAL DATA PROCESSING" BY JON F.
C CLAERBOUT (ISBN 0-86542-305-9), CHAPTER 7-2, PAGES 133-137.
C
C NOTE:  TESTING HAS SHOWN THAT THE OPTIMUM FILTER LENGTH (NF) IS
C ABOUT .707*ND.  A SHORTER LENGTH CANNOT SUFFICIENTLY DETERMINE
C ALL OF THE FREQUENCIES AND THEIR AMPLITUDES (ALTHOUGH FAIRLY
C GOOD FREQUENCY DETERMINATION OCCURS DOWN TO ABOUT .250*ND; IF
C SHORTER THAN THAT, IT RAPIDLY DETERIORATES).  A GREATER LENGTH
C BEGINS INTRODUCING MINOR NOISE.
C
C UNDER CERTAIN CIRCUMSTANCES THE BASIC BURG PREDICTION FILTER
C CAN BECOME UNSTABLE.  THIS OCCURS WHEN A BAND OF FREQUENCIES
C (PARTICULARLY THE HIGH FREQUENCIES) IS MISSING FROM THE
C SPECTRUM OF THE DATA TO BE FILTERED.  TO AVERT THE POSSIBILITY
C OF THIS HAPPENING, THIS SUBROUTINE ARTIFICIALLY INJECTS A
C RANDOM (ALMOST WHITE) NOISE SIGNAL INTO THE COEFFICIENT
C GENERATING PROCESS WITH AN AMPLITUDE OF -235 DB BELOW THE
C VARIANCE OF THE PROFILE.  THIS SIGNAL LEVEL IS SO SMALL THAT IT
C WOULD PROBABLY NEVER BE SEEN IN SUBSEQUENT SIGNAL PROCESSING
C (EVEN IF ONE IS LOOKING SPECIFICALLY FOR IT!).
C
C THE PROCESS THAT GENERATES THE RANDOM NOISE CAN BE MADE TO
C GENERATE IDENTICAL NOISE PROFILES IN SUBSEQUENT SUBROUTINE
C CALLS OR TO GENERATE DIFFERING NOISE PROFILES.  TO CHANGE, THE
C TWO COMMENTED STATEMENTS IN THE CODE MUST BE SWAPPED WITH THEIR
C UNCOMMENTED COUNTERPARTS (IQX, ISEED AND RANDM4, RAN).  THE
C IDENTICAL NOISE PROFILES WILL RESULT IN IDENTICAL BURG
C EXTENSIONS AND COEFFICIENTS FROM CALL TO CALL, BUT IF STACKING,
C THEY COULD ADD CONSTRUCTIVELY.  THE DIFFERING NOISE WILL CAUSE
C SLIGHT VARIATIONS IN BURG EXTENSIONS IF THE ORIGINAL DATA
C PROFILE HAD AN INCOMPLETE SPECTRUM (WAS UNSTABLE).  ON BALANCE,
C THE INJECTED NOISE IS SO SMALL THAT IT PROBABLY WOULD NEVER
C STACK TO SIGNIFICANT LEVELS; THEREFORE, THE IDENTICAL NOISE
C PROFILES ARE LIKELY THE BEST SELECTION.
C
C
C INPUT ARGUMENTS:
C   ND     - INTEGER*4.  THE NUMBER OF POINTS IN XDAT.  ND MUST
C            NOT EXCEED MXARR GIVEN AS A PARAMETER IN THE
C            SUBROUTINE.  THE CALCULATION TIME WILL BE
C            PROPORTIONAL TO ND*NF.
C   XDAT   - REAL*8.  ARRAY OF DIMENSION ND CONTAINING THE DATA
C            FOR FILTER DERIVATION.  IDEALLY, XDAT SHOULD BE SOME
C            KIND OF ERGODIC STATIONARY DATA IF EXTENSION IS
C            CONTEMPLATED FOR PREDICTIVE PURPOSES.
C   NF     - INTEGER*4.  THE DESIRED NUMBER OF FILTER
C            COEFFICIENTS.  NF MUST NOT EXCEED ND.  FOR A
C            MEANINGFUL FILTER, NF SHOULD NOT BE SMALLER THAN 2.
C
C OUTPUT ARGUMENT:
C   APF    - REAL*8.  ARRAY OF DIMENSION NF CONTAINING THE
C            TIME-DOMAIN COEFFICIENTS OF THE BURG ERROR
C            PREDICTION FILTER DERIVED FROM THE DATA IN XDAT.
C            THE FIRST COEFFICIENT, APF(1), ALWAYS HAS A VALUE OF
C            ONE.
C
```

```
C SUBROUTINE BURGC GIVEN BY JON CLAERBOUT, STANFORD UNIVERSITY.
C SUBROUTINE BURGDP ADAPTED BY ROB BRACKEN, USGS, 11NOV96.
C HP-9000 SERIES 700/800 UNIX VERSION 1.0, 11NOV96.
C
C
      subroutine burgdp(nd,xdat,nf,apf)
C
C DECLARATIONS
C
C     INPUT ARGUMENTS
      integer*4 nd
      real*8 xdat(0:*)
      integer*4 nf
C
C     OUTPUT ARGUMENT
      real*8 apf(0:*)
C
C     NUMBERS OF ELEMENTS
      integer*4 nx,mf
C
C     INTERNAL ARRAYS
      integer*4 mxarr
      parameter(mxarr=200000)
      real*8 ep(0:mxarr),em(0:mxarr),bpf(0:mxarr)
      common /work1/ ep,em,bpf
C
C     MISC VARIABLES
      real*8 ck,top,bot,epj
C
C     NOISE INJECTION
      real*8 acoef(2),varf,varu,dbnoi,ampnoi
      parameter(dbnoi=-235.d0)
      integer*4 iseed
c      integer*4 iqx
c      real*4 randm4
C
C CHECK ARRAY SIZES
C
      if(nd.gt.mxarr+1) call errfor(mxarr,
     & '(burgdp)  Array xdat too large')
      if(nf.gt.nd) call errfor(nf,
     & '(burgdp)  Too many filter elements')
C
C INITIALIZE ARRAYS AND VARIABLES
C
C     ADJUST UPPER LIMITS TO A ZERO TO N-1 SYSTEM (FROM ONE TO N)
      nx=nd-1
      mf=nf-1
C
C     LOCATION OF CURRENT FILTER COEFS (ISW=0 APF, ISW=1 BPF)
      isw=0
C
C     ZEROTH FILTER COEFFICIENT
      apf(0)=1.d0
C
C     FIND AMPLITUDE OF STABILIZING RANDOM NOISE SIGNAL TO INJECT
      call trendr(1,2,acoef, 0,-nd,xdat(0))
```

```
      call varfqd(0.d0,0,nd,xdat(0),acoef(2),acoef(1), varf,varu)
      ampnoi=dsqrt(varu*10.d0**(dbnoi/10.d0))
      if(ampnoi.eq.0.d0) then
C
C       PROF IS ALL 0; FLTR WILL BE UNSTBLE; MAKE ARTIFICIAL FLTR
        do i=1,mf
          apf(i)=-1.d0/dflotj(mf)
        enddo
        goto 990
      endif
C
C     PUT DATA VALUES INTO THE EP AND EM ARRAYS AND INJECT NOISE
C     (NOTE:  USE OF FUNCTION RANDM4 AUTOMATICALLY SELECTS A
C     DIFFERENT SEED VALUE FOR EACH CALL TO ISEED().  THEREFORE,
C     IF SEVERAL PROFILES ARE EXTENDED USING THIS SUBROUTINE FOR
C     THE BURG FILTER COEFS AND THEN STACKED, THE NOISE WILL TEND
C     TO BE SUMMED OUT.)
c      iqx=iseed(0)
      iseed=15377317
      do i=0,nx
c        ep(i)=xdat(i)
c        ep(i)=xdat(i)+ampnoi*dble(randm4()-0.5)
        ep(i)=xdat(i)+ampnoi*dble(ran(iseed)-0.5)
        em(i)=ep(i)
      enddo
C
C ITERATE FILTER COEFS OVER LENGTHS K=1,MF
C
      do 701 k=1,mf
C
C       FIND THE REFLECTION COEFFICIENT, CK
        top=0.d0
        bot=0.d0
        do j=k,nx
          jm=j-k
          top=top+em(jm)*ep(j)
          bot=bot+em(jm)*em(jm)+ep(j)*ep(j)
        enddo
        ck=2.d0*top/bot
C
C       UPDATE EP AND EM
        do j=k,nx
          jm=j-k
          epj=ep(j)
          ep(j)=epj-ck*em(jm)
          em(jm)=em(jm)-ck*epj
        enddo
C
C       FIND THE NEXT FILTER COEFS USING CK & LEVINSON RECURSION
        if(isw.eq.0) then
C
C         CURRENT FILTER IN APF; PUT NEW IN BPF
          isw=1
          apf(k)=0.d0
          do i=0,k
            bpf(i)=apf(i)-ck*apf(k-i)
          enddo
```

```
        else
C
C         CURRENT FILTER IN BPF; PUT NEW IN APF
          isw=0
          bpf(k)=0.d0
          do i=0,k
            apf(i)=bpf(i)-ck*bpf(k-i)
          enddo
        endif
  701 continue
C
C     MOVE NEW FILTER BACK INTO APF
      if(isw.eq.0) goto 990
      do i=0,mf
        apf(i)=bpf(i)
      enddo
C
C EXIT PROCEDURE
C
  990 return
      end
```