UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

TAYLOR: a FORTRAN program using Taylor series expansion for level-surface or surface-level continuation of potential-field data

by

V.J.S. Grauch

Open-File Report 84-501

TABLE OF CONTENTS

PREFACE

All routines were written in VAX/VMS FORTRAN 77. Conversion to other computers would require a moderate amount of time for an experienced programmer. The largest floating point number on the VAX, hex ffff7fff (approximately 1.7 X 10^{38}), is used to flag areas in grids that have no data.

INTRODUCTION

This program uses a Taylor's series expansion to approximate the continuation of potential-field data observed on a level surface onto an irregular surface (level-to-drape continuation) or vice versa (drape-to-level continuation). The method is presented and discussed in detail by Cordell and Grauch (1984).

The vertical differentiation required by the Taylor series expansion and any level-to-level continuation or filtering that is required are accomplished by fast Fourier transform (FFT) methods coded by Hildenbrand (1983). Input data should be in USGS standard grid format (see Appendix A). Grids which are incompletely filled with data are allowed but the incomplete areas will be filled using the extrapolation technique described in Hildenbrand (1983) before FFT transformation begins.

THEORY

The Taylor series expansion method follows that of Cordell and Grauch (1984) and is only briefly described here.

Level-to-drape continuation.--Let f(z) be potential field data observed on an irregular surface which is described as a distance z (z positive down) from a certain reference level (z = 0). Then f(z) can be expanded as a Taylor's series about the point z = 0 (MacLaurin series) as

$$f(z) = f(0) + z \frac{\partial}{\partial z} f(0) + \frac{z^2}{2} \frac{\partial^2}{\partial z^2} f(0) + \cdots .$$

Data observed on the reference level then can be used to approximate data on the surface z, providing z is small enough so that the series converges. Assuring that z is small enough requires the proper choice of the reference level and a small relief of the surface compared to the grid interval (discussed below). If data were observed on a level that cannot be conveniently assigned as the reference level (fig. 1) then level-to-level continuation can be performed first before the Taylor's series expansion is applied.

Drape-to-level continuation.--If we know f(z) but do not know f(0) (fig. 2) we can rearrange the Taylor's series expansion of f(z) to become

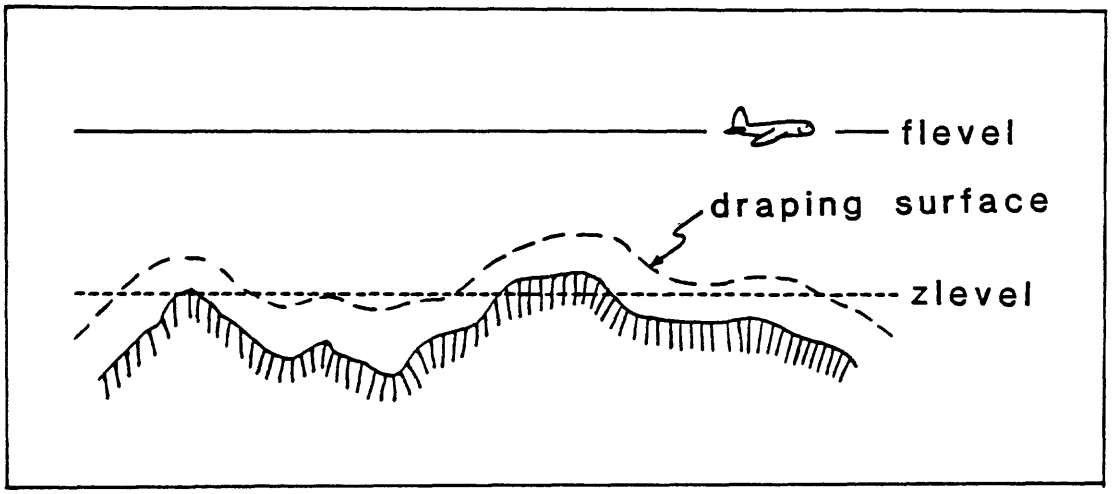


Figure 1. A common level-to-drape situation. Aeromagnetic data observed on flevel will be continued onto the draping surface. zlevel is the z reference level needed for the Taylor's series expansion.

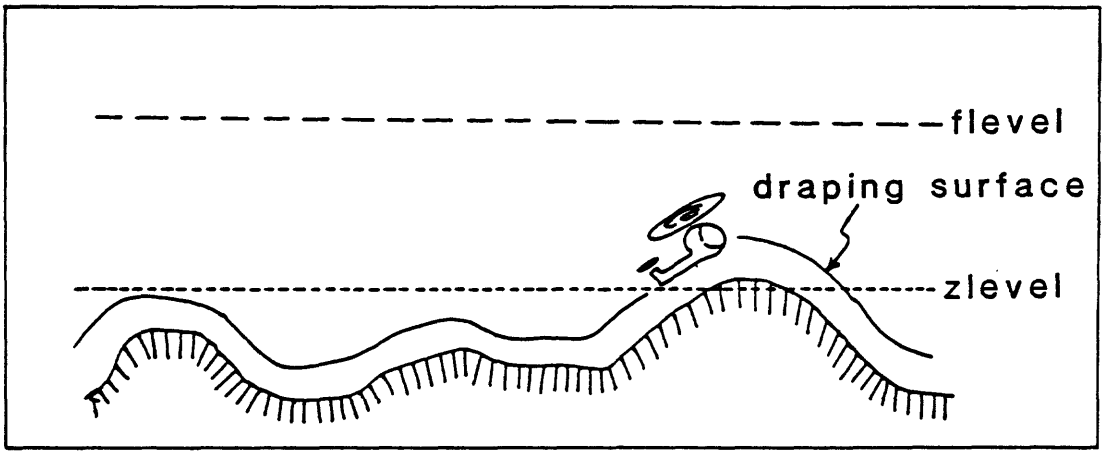


Figure 2. A common drape-to-level situation. A draped aeromagnetic survey will be incorporated in a more regional data set at the level flevel. The Taylor's series expands about zlevel.

$$f(0) = f(z) - z \frac{\partial}{\partial z} f(0) - \frac{z^2}{2} \frac{\partial^2}{\partial z^2} f(0) - \cdots .$$

This equation can act as a recursion formula to successively approximate f(0) thusly:

$$f_{n+1}(0) = f(z) - z \frac{\partial}{\partial z} f_n(0) - \frac{z^2}{2} \frac{\partial^2}{\partial z^2} f_n(0) - \cdots .$$

To see how good is the nth approximation, we can re-drape the level-data approximation using the level-to-drape operation and compare the results with the original, known, draped data. Thus the recovery error for the nth approximation is defined as

$$\varepsilon_n = f(z) - \left[f_n(0) + z \frac{\partial}{\partial z} f_n(0) + \frac{z^2}{2} \frac{\partial^2}{\partial z} f_n(0) + \cdots \right] .$$

and so

$$f_{n+1}(0) = f_n(0) + \varepsilon_n .$$

Thus the recovery error can provide some sort of estimate of error at successive approximations but does not constitute the error of the continuation process itself.

We now need only a beginning approximation to get started. Cordell and Grauch (1984) found that the best first approximation was to treat f(z) as though it were on a level and project it onto -z using the level-to-drape equation:

$$f_1(0) \equiv f(z) - z \frac{\partial}{\partial z} f(z) + \frac{z^2}{2} \frac{\partial^2}{\partial z^2} - \cdots .$$

Drape-to-drape continuation.-- Successive drape-to-level then level-to-drape operations constitute drape-to-drape continuation.

PRACTICAL CONSIDERATIONS

FFT problems.--Inherent in the FFT methods that are required for this continuation method are high-frequency artifacts of the discrete FFT (Cordell and Grauch, 1982), the problems of amplifying high frequency noise (due to various causes), and edge effects (discussed in Hildenbrand, 1983). Therefore it is usually desirable to lowpass filter the data while the derivative and downard-continuation operations are being performed. Hildenbrand (1983) describes the operation of lowpass filtering in detail. Briefly, two wavelengths describe the filter, w1 and w2, where w2 $\geq$ w1 (fig. 3). All wavelengths in the data that are less than w1 will be removed from the data

and all data that have wavelengths greater than w2 will remain. Data with wavelengths in between w1 and w2 will be partially removed.

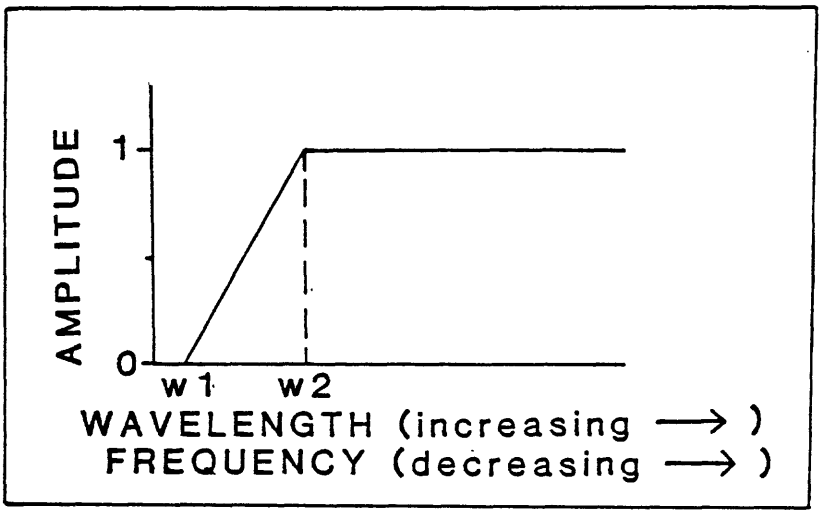


Figure 3. Picture of the lowpass filter used in TAYLOR in the frequency domain. Wavelengths greater than w2 (low frequencies) are preserved (passed), while wavelengths less than w1 (high frequencies) are squelched. Wavelengths between w1 and w2 are partially removed.

When to use the Taylor's series method.--Deciding when the Taylor's series expansion method is appropriate for a certain situation, how many terms of the series to use and/or how many iterations to make is very empirical. The main factors to consider, however, are relief of the irregular surface in terms of grid interval, and the frequency content of the observation surface and of the potential field data to be continued. The following is a list of empirical observations made from tests on theoretical data; they do not constitute any more than broad guidelines to follow. It should be emphasized that a lack of familiarity with the problems inherent in the use of FFT methods will severely handicap the usefulness of this continuation method.

Level to drape.

1. If the anomalies of the original level data are very smooth and broad, the Taylor series is quite stable up to a maximum magnitude of z greater than 4*dx, where dx is the raow and column spacing of the input grid. For "typical" aeromagnetic data, however, the series begins to diverge somewhere between $dx<|z|<2*dx$. The divergence can not always be discerned from the "smoothness" of the output contours!
2. Using 3 terms of the Taylor series is always recommended over only using 2 terms for level-to-drape.

Drape-to-level

1. The same goes as for #1 above except that the drape-to-level operation is more sensitive to high-frequency input.
2. Using 2 terms of the Taylor series works better than 3 terms most of the time for drape-to-level.
3. I recommend using 2 iterations when running the first time. Watch the recovery error (especially the standard deviation) to see if it is getting smaller with the second iteration. If so, the output can be used. If not, check the relief of your surface (you may have gotten a warning message). If the total surface relief is low enough, try a more severe lowpass filter (w1 and w2 larger). If the surface relef is too high, try a coarser grid interval.

EXAMPLE

The reasoning that determines the parameters to use in TAYLOR will be demonstrated through use of a practical example. Figure 4a shows data from an aeromagnetic survey that was flown on a tilted plane (figure 4b). We wish to drape the data 304.8 m (1000 feet) above ground, or onto the surface shown in figure 4c. The operation involves two steps: drape-to-level then level-to-drape. The following summarizes the information needed for the first step.

Minimum flight elevation:	304.8 m
Maximum flight elevation:	609.6 m
Mean of flight surface:	458 m
Grid interval:	.2 km

A reference level of 500 m is close to the mean and gives a maximum of 195.2 m to continue up to the 500 m level and a maximum of 109.6 m to continue down to the level, both maximums less than the grid interval. It is desirable to have the reference level a little higher than the mean of the surface for drape-to-level operations so that more data on the surface is continued up to the level rather than down to it.

Because so much data are missing in the original grid, the data were extrapolated to cover the entire grid using the same gridding routine used to grid the data in the first place. The plugging routine used in FFTFIL (Hildenbrand, 1983) and thus TAYLOR is not reliable in such extreme cases of missing data.

The first run of TAYLOR using two terms only and default filtering parameters w1 and w2 (see figure 3) of 0.4 and 0.8 km gave the following results:

	RECOVERY ERROR MAGNITUDE		
Iteration	Max	Mean	Std. Dev.
1	128.742	1.21609	5.50400
2	123.792	0.93172	5.01649

The method seemed to be converging but the maximum error and standard deviation especially are large compared to the desired contour interval of 10

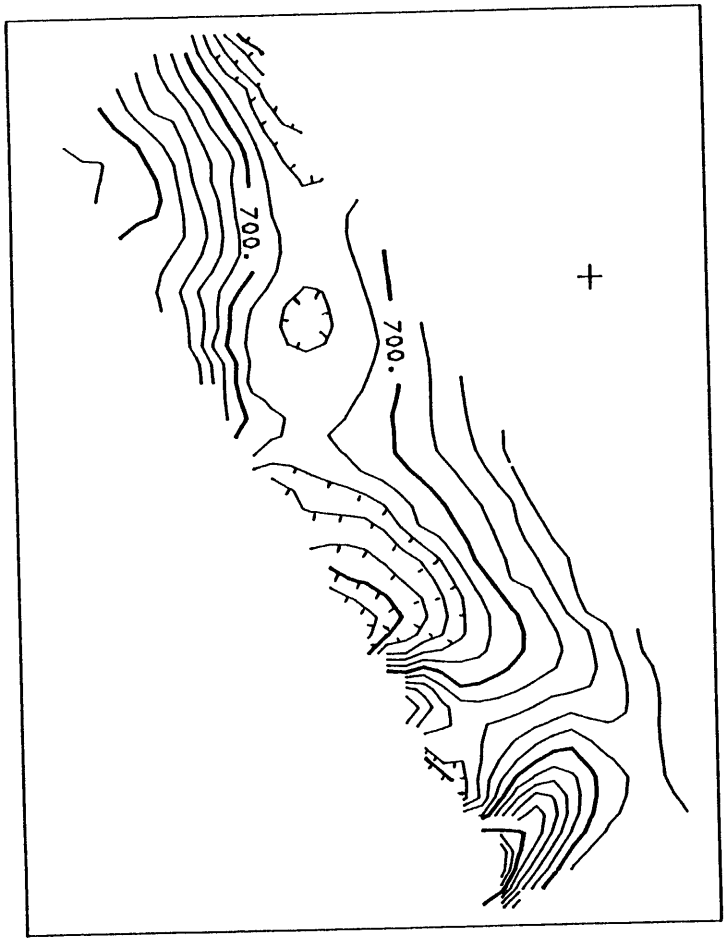


Figure 4a. Original aeromagnetic data used for example. Contour interval=10 gammas.

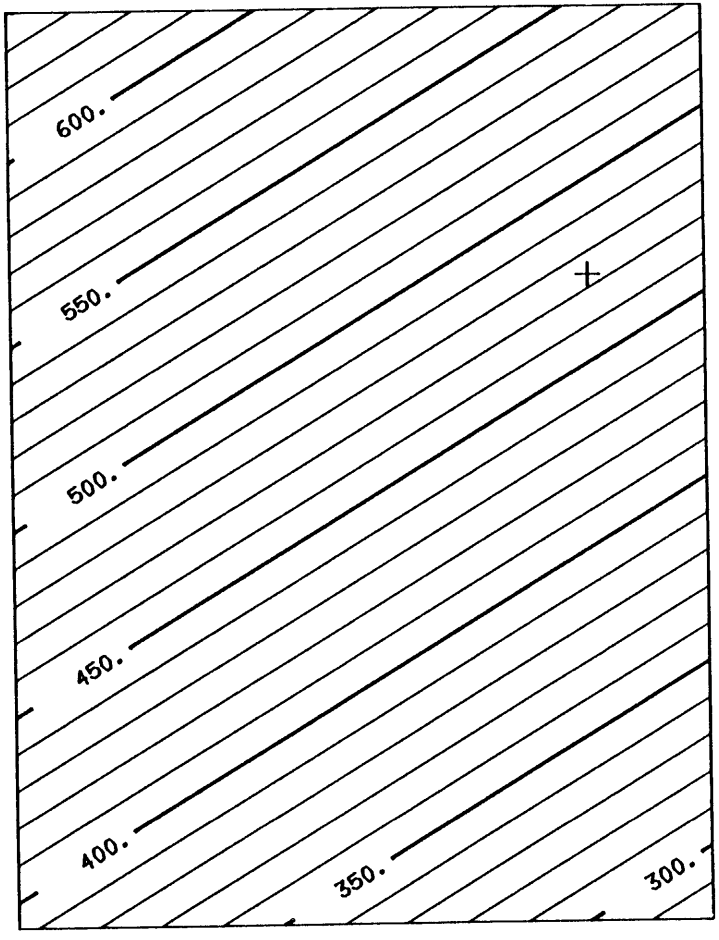


Figure 4b. Flight surface upon which the original aeromagnetic data were observed. Contour interval=10 meters. Flight lines were approximately perpendicular to contour lines.

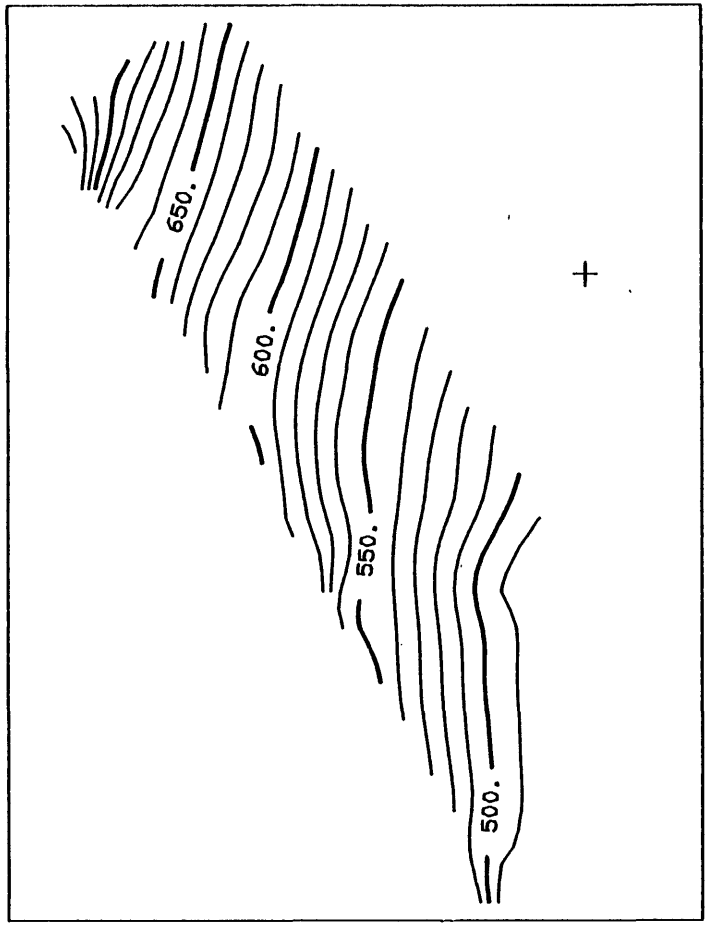


Figure 4c. Draping surface upon which the data are desired to be observed, 304.8 m (1000 feet) above ground. Contour interval=10 meters.

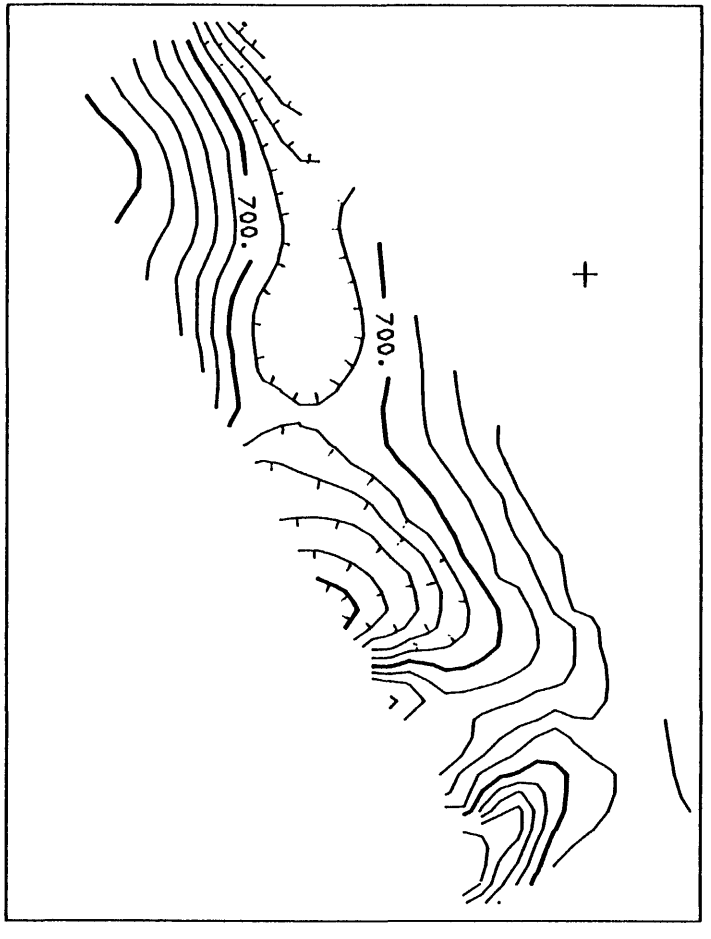


Figure 4d. Results of draping the original aeromagnetic data on the draping surface of figure 4c. Contour interval=10 gammas.

gammas. Using w1=0.8 and w2=1.2 gave

	RECOVERY ERROR MAGNITUDE		
Iteration	Max	Mean	Std. Dev.
1	46.1606	1.06558	3.64650
2	28.2923	0.58826	2.16909

and is considered acceptable.

To demonstrate the desirability of the use of only two terms of the expansion over three terms, the following results were obtained for three terms using w1=0.8 and w2=1.2.

	RECOVERY ERROR MAGNITUDE		
Iteration	Max	Mean	Std. Dev.
1	87.9274	1.72113	6.12554
2	95.3636	1.70490	6.69927

The method is not converging.

The information needed for the second step, the level-to-drape operation, is shown below.

Minimum of draping surface:	486 m
Maximum of draping surface:	742 m
Total relief:	256 m
Mean of surface:	561 m
Grid interval	0.2 km

A reference level of 600 m gives 114 m maximum to continue down to the surface, 142 m to continue up. Analogous to the drape-to-level situation, the maximum amount to continue downward should be less than the amount to continue upward, and the reference level should be near the mean value of the draping surface.

Note that from the drape-to-level operation we ended with the data on the 500 m level. A reference level of 600 m here requires upward continuation of 100 m before the Taylor's series is applied. This operation constitutes a lowpass filter on the data, so no filtering of the data during the level-to-drape operation is necessary.

Figure 4d shows the final results. Because the relief of both the flight surface and draping surface was so low compared to the grid interval, there is not a large difference between the original and the final result.

PROGRAM EXECUTION

TAYLOR may be run interactively or with a FORTRAN namelist command file. If run interactively one has the choice of being asked very descriptive questions or very brief questions. After answering the interactive questions a command file can be created by the program for later use by the user. NOTE: The command file that is created by the program does not write the following parameters into the file: iter, title, or any file names. These

parameters can be added with an editor or if left blank, they will be asked for during program run.

Begin program run. The first question asked is

```
1 - level to drape operation
2 - drape to level operation
            Enter option:
```

Enter number indicating the operation desired. The second question is

```
Enter command or command file name:
```

Commands are help, exit and term. term will start the interactive mode. help, exit are self-explanatory. Any other character string entered will be taken as a command file name.

Parameters

idval — Switch to tell whether input grids have incomplete data or not. For level-to-drape option, idval = 0 means input grid is complete, idval = 1 means it is incomplete, while the observation surface grid may have incomplete data or not but does not affect the idval parameter. For the drape-to-level operation, idval=0 means both the data grid and surface grid have complete data and idval=1 means that either the data grid or the surface grid have incomplete data. Default is 0.

flevel — level-to-drape: level at which input data are observed. drape-to-level: level to which data are desired to be continued. Units of flevel must be the same as the units of the irregular (draping) surface.

convf — Conversion factor to convert data of draping surface to those of the grid, negative if z is positive down, positive if z is positive up (e.g., elevation above sea level). For example, if the irregular surface is elevation in meters and the grid interval is kilometers, convf = .001. Default is 1.

zlevel — Reference level for the Taylor's series expansion. This level should be chosen in order to minimize the distance between the level and irregular surface. Default is that program chooses zlevel. Units the same as those of the irregular surface.

nterms — Number of terms to use in the Taylor's series expansion. 2 terms truncates after the 1st derivative term, 3 terms after the 2nd derivative term. nterms = 3 recommended for level-to-drape, nterms = 2 recommended for drape-to-level. Default is 3.

iter — drape-to-level only. Number of iterations to make in the recursion formula. Choice of 1 or 2 iterations, giving the 2nd or 3rd approximation, respectively. If iter = -1, 2 iterations will be made while saving the 2nd approx. (made before final continuation to flevel) in file fl.tmp. Default = 2.

ifilt — 0 - no filtering
1 - filter derivatives of Taylor series
2 - filter derivatives and filter during continuation operation (usually only used for downward continuation).
3 - filter during continuation only. Default = 0.

wl w2 — Important only if ifilt ≠ 0, w2>wl. wl and w2 describe the lowpass filter to be used (fig. 3). Units are the same as grid units. Defaults are wl=2*dx, w2=4*dx.

nadd — Number of rows and columns to add in order to extend the grid and thereby reduce edge effects. Same as in program FFTFIL (Hildenbrand, 1983). Default = 0.

ifile — Input data file in standard USGS grid format (see Appendix A), enclosed in single quotes. The first row of this file must not be entirely flagged data (incomplete). The grid interval in the x- and y-directions must be equal (dx=dy).

sfile — Input USGS standard grid of the irregular (draping) surface, enclosed in single quotes. The first row of this file must not be entirely flagged data (incomplete). Grid specifications must exactly match those of ifile.

ofile — Output grid filename, enclosed in single quotes.

title — Up to 56 characters of title for output file, enclosed in single quotes.

Example Run #1

```
1 - LEVEL TO DRAPE CONTINUATION
2 - DRAPE TO LEVEL CONTINUATION
       ENTER OPTION: 1
COMMAND OR COMMAND FILE NAME: TERM

BEGIN ENTERING PARMS INTERACTIVELY.
WANT ME TO ASK VERBOSE (AS OPPOSED TO BRIEF) QUESTIONS? Y

WILL INPUT DATA FILE HAVE DVALS? N

 WHAT IS THE FACTOR TO CONVERT THE UNITS OF MEASURE OF THE SURFACE INTO THE
 UNITS OF MEASURE OF THE GRID INTERVAL (E.G. .001 IF SURFACE IS IN M AND
 GRID UNITS IN KM).  ENTER THE NEGATIVE OF THIS NUMBER IF THE SENSE OF THE
 SURFACE FILE IS Z POSITIVE DOWN.
CONVERSION FACTOR = .001

 GIVE THE LEVEL AT WHICH THE INPUT DATA WERE OBSERVED IN THE SAME UNITS AS THOSE
 OF THE DRAPING SURFACE.  (REMEMBER TO BE CONSISTENT WITH WHETHER UP OR DOWN
 IS THE POSITIVE DIRECTION.)
FLEVEL= 500

 THE TAYLOR SERIES EXPANDS ABOUT A Z REFERENCE LEVEL THAT SHOULD BEST BE NEAR
 THE MEAN OF THE DRAPING SURFACE SO THAT THE DEVIATION OF THE SURFACE
 ABOVE AND BELOW THAT LEVEL IS THE LEAST AMOUNT.  ENTER THE LEVEL IN THE
 SAME UNITS USED FOR THE DRAPING SURFACE.
ENTER ZLEVEL (1.E+38 TO USE DEFAULT):  1.E+38

 BANDPASS (LOWPASS) FILTERING IS OFTEN DESIRABLE DURING THE 1ST & 2ND VERT.
 DERIVATIVE-TAKING OF THE TAYLOR SERIES EXPANSION OR IN THE CONTINUATION
 OF FLEVEL TO ZLEVEL OR VICE VERSA.  OF COURSE IF ZLEVEL=FLEVEL, FILTERING
 THE CONTINUATIO IS NOT DESIRABLE.
ENTER FILTERING OPTION (-1 FOR HELP): -1
  0 - NO FILTERING
  1 - FILTER DERIVS ONLY
  2 - FILTER DERIVS AND CONTINUATION
  3 - FILTER CONTINUATION ONLY
ENTER FILTERING OPTION (-1 FOR HELP): 1

 TWO WAVELENGTHS W1<=W2 DESCRIBE A (TAPERED) LOWPASS FILTER.  REMEMBER
 WAVELENGTH=2 * ANOMALY WIDTH.  DEFAULTS:  W1=2*GRID INTERVAL, W2=W1+2*GRID
 INTERVAL.  W1 & W2 ARE IN SAME UNITS AS GRID INTERVAL.
ENTER W1 AND W2 (TYPE 0 0 TO USE DEFAULT) 0 0

 IT OFTEN HELPS THE FFT OPERATION TO EXTEND THE NUMBER OF ROWS AND
 COLUMNS BY A SMALL PROPORTION OF THE TOTAL GRID  ENTER THE NUMBER TO
 ADD TO THE COLS AND ROWS:
NADD = 4

NOW I CAN CREATE A COMMAND FILE WITH ALL THE PARAMETERS YOU PICKED
SO THAT YOU CAN EDIT AND/OR USE IT FOR A LATER PROGRAM RUN.

WANT TO SAVE THIS INFO IN COMMAND FILE? Y

**COMMAND FILE TAYLOR.CMD CREATED**

INPUT FILE (LEVEL SURVEY): EAG3.TMP

ENTER DRAPING OR IRREGULAR SURFACE GRID: EAGD3.JIG

OUTPUT FILE: EAG3A.DRP
ENTER TITLE
LEVEL TO DRAPE

USING Z REF. LEVEL OF     670.000
USING W1,W2=    0.400000        0.800000

COMPUTING...

FORTRAN STOP
```

Command file created by previous example

```
 &PARMS FLEVEL=   500.000     ,CONVF=  0.100000E-02,NTERMS=3
 IFILT=1,NADD= 4,IDVAL=0 &
$
```

Example Run #2

```
1 - LEVEL TO DRAPE CONTINUATION
2 - DRAPE TO LEVEL CONTINUATION
        ENTER OPTION: 2
COMMAND OR COMMAND FILE NAME: TERM

BEGIN ENTERING PARMS INTERACTIVELY.
WANT ME TO ASK VERBOSE (AS OPPOSED TO BRIEF) QUESTIONS? N

WILL EITHER INPUT FILE HAVE DVALS? N
CONVERSION FACTOR = .001
ENTER ZLEVEL (1.E+38 TO USE DEFAULT):  500
FLEVEL = 500
WANT 2 OR 3 TERMS IN TAYLOR EXPANSION? 2
ENTER FILTERING OPTION (-1 FOR HELP): -1
  0 - NO FILTERING
  1 - FILTER DERIVS ONLY
  2 - FILTER DERIVS AND CONTINUATION
  3 - FILTER CONTINUATION ONLY
ENTER FILTERING OPTION (-1 FOR HELP): 1
ENTER W1 AND W2 (TYPE 0 0 TO USE DEFAULT) .8 1.2
NADD = 4
ENTER ITER: -1

WANT TO SAVE THIS INFO IN COMMAND FILE? N

INPUT FILE (DRAPED SURVEY): EAG3.GRD

ENTER DRAPING OR IRREGULAR SURFACE GRID: EAG3.PLN

OUTPUT FILE: EAG3.TMP
ENTER TITLE
DRAPE TO LEVEL OPERATION, TO LEVEL 500

                          RECOVERY ERROR MAGNITUDE
ITERATION          MAX                  MEAN              STD DEV.

    1            46.1606              1.06558            3.64650
    2            28.2923             0.588259            2.16909

1ST-ITER. APPROX. (ON Z REF. LEVEL) IS IN FILE F1.TMP
FORTRAN STOP
```

Command file used in the following example

```
$PARMS
FLEVEL=500,CONVF=.001,ZLEVEL=500,NTERMS=2,ITER=2,IFILT=1,NADD=4,
IFILE='EAG3.GRD',SFILE='EAG3.PLN',OFILE='EAG3.TMP',
TITLE='TEST OF EAGLE 3'
$
```

Example Run #3

```
1 - LEVEL TO DRAPE CONTINUATION
2 - DRAPE TO LEVEL CONTINUATION
       ENTER OPTION: 2
COMMAND OR COMMAND FILE NAME: MYTAYLOR.CMD
USING W1,W2=    0.400000        0.800000

                                RECOVERY ERROR MAGNITUDE
ITERATION          MAX                MEAN              STD DEV.

    1            128.742            1.21609             5.50400
    2            123.792           0.931721             5.01649
FORTRAN STOP
```

REFERENCES

Cordell, Lindrith and V.J.S. Grauch, 1982, Reconciliation of the discrete and integral Fourier transforms: Geophysics, V. 47, no. 2, p. 237-243.

Cordell, Lindrith and V.J.S. Grauch, 1984, Mapping basement magnetization zones from aeromagnetic data in the San Juan Basin, New Mexico: (in press for special Geophysics issue).

Hildenbrand, Thomas G., 1983, FFTFIL: A filtering program based on two-dimensional Fourier analysis: U.S. Geological Survey Open-File Report 83-237.

APPENDIX A - USGS Standard Grid File

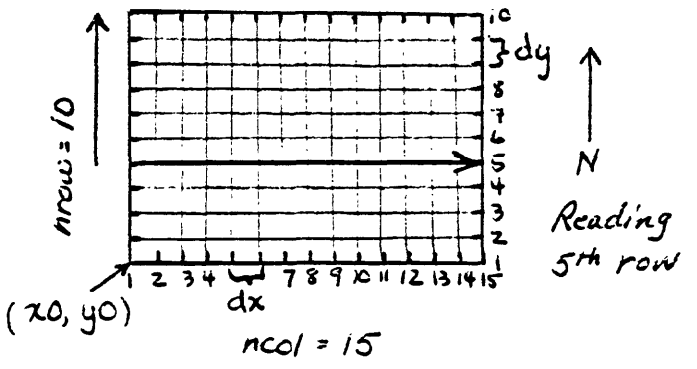
The USGS standard grid file accomodates a variety of ways to specify a binary grid of data, but is standardly used within the Geophysics Branch in the following less general form.

Header Record

id	56 ASCII characters of identification (14 words).
pgm	8 ASCII characters describing creation program (2 words).
ncol	number of columns of data (integer, one word).
nrow	number of rows of data (integer, one word).
nz	not used, must be 1.
x0	position of first (leftmost) column of data (real, one word).
dx	the interval of equal spacing between columns (real, one word).
y0	position of first (lowermost row of data (real, one word).
dy	the interval of equal spacing between rows (real, one word).

Data Records

Each unformatted data record contains one row of the grid, moving from left to right along the row, and moving from bottom row to top row as illustrated below.



```
c ****************************************************************
c
c  taylor.for
c
c  Driver for level-to-drape and drape-to-level continuation using
c  Taylor series expansion.
c
c  V.J.S. Grauch, USGS, March 1984.
c
c     f     =     input data
c     g     =     1st vertical derivative of f
c     h     =     2nd vertical derivative of f
c     z     =     surface data
c     iopt  =     1 if level-to-drape chosen, 2 if drape-to-level
c
c ****************************************************************
      character*50 ifile,ofile,outfile,ifilea,ifileb,sfile
      character*56 title
      dimension work(2048),f(2,1024,16),g(2,1024,16),h(2,1024,16),
     1 z(1024),id(14),pgm(2)
      common/12dparms/flevel,convf,zlevel,ifilt,w1,w2,idval,nadd,nterms,
     1ifile,ofile,sfile,title,iter
      common/12dinfo/p1,p2,nz,yo,xo,dx,dy,ncol,nrow
      common/units/iw,ir,in,iout,icmd,is,iunita,iunitb,iunitc,if,iunitg,
     1iunith
      data iw/6/,ir/5/,in/10/,iout/12/,icmd/9/,is/11/,iunita/20/,
     1iunitb/21/,iunitc/22/,if/14/,iunitg/23/,iunith/24/,lnri/16/
1     write(iw,800)
800   format(1x,'1 - level to drape continuation'/' 2 - drape to level
     1 continuation'/8x,'Enter option: ',$)
      read(ir,*) iopt
      if(iopt.lt.1.or.iopt.gt.2) go to 1
c
c  set up input parameters
c
      call setup(iopt)
c  find no. of rows for fft and extend cols by nadd
      call extend(nadd,n1,n2,id2,lnri,nri,nxa)
      if(iopt.eq.2.or.iopt.eq.4) go to 20
c
c   LEVEL TO DRAPE OPERATION
c
      p1='*1 t'
      p2='o d*'
      open(iout,file=ofile,form='unformatted',status='new')
      write(iout) title,p1,p2,ncol,nrow,nz,xo,dx,yo,dy
      write(iw,804)
804   format(/' Computing...'/)
      call 12d(f,g,h,work,id2,nri,n1,n2,z,nxa)
      close(in)
      close(is)
      close(iout)
      go to 300
c
c  DRAPE TO LEVEL OPERATION
c
20    finlevel=flevel
      flevel=zlevel
      isave=0
c get 1st approx by using level-to-drape with neg. z
      convf=-convf
      if(iter.eq.-1) then
         isave=1
         iter=2
      endif
      open(iout,file='f1.tmp',status='new',form='unformatted')
      write(iout) id,pgm,ncol,nrow,nz,xo,dx,yo,dy
      write(iw,805)
805   format(/28x,'RECOVERY ERROR MAGNITUDE'/' Iteration',10x,'Max',
     1 15x,'Mean',12x,'Std dev.'/)
      call 12d(f,g,h,work,id2,nri,n1,n2,z,nxa)
      close(in)
      rewind is
      close(iout)
c
c  Get next approximations
c
      convf=-convf
      outfile='f1d.tmp'
      ifilea='f1.tmp'
      k=0
25    k=k+1
      open(in,file=ifilea,status='old',form='unformatted',readonly)
      read(in) id,pgm,ncol,nrow,nz,xo,dx,yo,dy
      read(is) id,pgm,ncol,nrow,nz,xo,dx,yo,dy
      open(iout,file=outfile,status='new',form='unformatted')
      write(iout) id,pgm,ncol,nrow,nz,xo,dx,yo,dy
      call 12d(f,g,h,work,id2,nri,n1,n2,z,nxa)
      close(in)
      close(iout)
c  find error and add error to one approx to get next approx
      if(k.eq.2) go to 32
      outfile='f2.tmp'
      ifilea='f1d.tmp'
      ifileb='f1.tmp'
      go to 33
32    ifilea='f2d.tmp'
      ifileb='f2.tmp'
      outfile='f3.tmp'
33    if(k.ne.iter) go to 35
      close(is)
      if(abs(finlevel-zlevel).gt.0.0001) go to 36
      open(iout,file=ofile,form='unformatted',status='new')
      p1='*d t'
      p2='o 1*'
      write(iout) title,p1,p2,ncol,nrow,nz,xo,dx,yo,dy
      go to 40
35    rewind(is)
```

```
      open(iout,file=outfile,form='unformatted',status='new')
      write(iout) id,pgm,ncol,nrow,nz,xo,dx,yo,dy
      open(in,file=ifile,status='old',form='unformatted',readonly)
      open(iunita,file=ifilea,status='old',form='unformatted')
      open(iunitb,file=ifileb,status='old',form='unformatted')
      read(in) id,pgm,ncol,nrow,nz,xo,dx,yo,dy
      read(iunita) id,pgm,ncol,nrow,nz,xo,dx,yo,dy
      read(iunitb) id,pgm,ncol,nrow,nz,xo,dx,yo,dy
      errmax=-1.e+38
      sumerr=0.0
      sumerr2=0.0
      en=0.0
      do 50 j=1,nrow
      read(in) dum,(f(1,i,1),i=1,ncol)
      read(iunita) dum,(g(1,i,1),i=1,ncol)
      read(iunitb) dum,(h(1,i,1),i=1,ncol)
      do 45 i=1,ncol
      if(f(1,i,1).gt.1.e+38) go to 45
      err=f(1,i,1)-g(1,i,1)
      if(abs(err).gt.errmax) errmax=abs(err)
      sumerr=sumerr+abs(err)
      if(sumerr.ge.1.e+20) go to 299
      sumerr2=sumerr2+err*err
      en=en+1.0
      f(1,i,1)=err+h(1,i,1)
45    continue
      write(iout) dum,(f(1,i,1),i=1,ncol)
50    continue
      sumerr=sumerr/en
      stddev=(sumerr2-(sumerr*sumerr)/en)/(en-1.0)
      stddev=sqrt(stddev)
      write(iw,802) k,errmax,sumerr,stddev
802   format(5x,i1,9x,g14.6,2(5x,g14.6))
      close(in)
      close(iunita,status='delete')
      if(isave.ne.0) close(iunitb)
      if(isave.eq.0) close(iunitb,status='delete')
      close(iout)
      if(k.eq.iter) go to 100
      ifilea=outfile
      outfile='f2d.tmp'
      go to 25
100   if(isave.ne.0) write(iw,803)
803   format(/' last-iter. approx. (on z ref. level) is in file f1.tmp')
      if(abs(finlevel-zlevel).le.0.0001) go to 300
c  level-to-level continuation if needed for final level file
      flevel=zlevel
      zlevel=finlevel
      nterms=1
      open(in,file=outfile,status='old',form='unformatted')
      read(in) id,pgm,ncol,nrow,nz,xo,dx,yo,dy
      open(iout,file=ofile,status='new',form='unformatted')
      p1='*d t'
      p2='o 1*'
      write(iout) title,p1,p2,ncol,nrow,nz,xo,dx,yo,dy
      call l2d(f,g,h,work,id2,nr1,n1,n2,z,nxa)
      close(in,status='delete')
      close(iout)
      go to 300
299   write(iw,806)
806   format(' Method is rapidly diverging.  Abort.')
      close(iunita,status='delete')
      close(iunitb,status='delete')
      close(iout,status='delete')
300   stop
      end
c *************************************************************************
c
      subroutine setup(iopt)
c sets up the input parameters for the level to drape or
c drape to level operations.  Used with taylor.for
c
c *************************************************************************
      character answer*1,title*56
      character*50 ifile,ofile,sfile,command
      common/l2dparms/flevel,convf,zlevel,ifilt,w1,w2,
     1idval,nadd,nterms,ifile,ofile,sfile,title,iter
      common/l2dinfo/p1,p2,nz,yo,xo,dx,dy,ncol,nrow
      common/units/iw,ir,in,iout,icmd,is,iunita,iunitb,iunitc,if,iunitg,
     1iunith
      dimension id(14),pgm(2),z(1024)
c
c
      ifile=' '
      ofile=' '
      sfile=' '
      title='this is the default title'
1     write(iw,800)
800   format(' Command or command file name: ',$)
      read(ir,801) command
801   format(a50)
      if(command.eq.'exit'.or.command.eq.'EXIT') go to 300
      if(command.eq.'help'.or.command.eq.'HELP'.or.command.eq.' ') then
        write(iw,888)
        go to 1
 888  format(' Commands are'/,5x,'term - enter the parameters desired
     1interactively and create .cmd file at'/,10x,' the same time.',/,
     2 5x,'help'/ 5x,'exit'/' Anything else entered will be taken to
     3 be a command file name.')
      endif
      if(command.ne.'term'.and.command.ne.'TERM') go to 50
c
c  interactively set up parameters and command file.  command file will
c  be named taylor.cmd
      command='taylor.cmd'
      write(iw,802)
802   format(/,' Begin entering parms interactively.',/,' Want me to
     1 ask verbose (as opposed to brief) questions? '$)
```

```
      read(ir,803) answer
803   format(a1)
      if(answer.eq.'y'.or.answer.eq.'Y') iopt=iopt+2
c ask for parms
      if(iopt.eq.2.or.iopt.eq.4) write(iw,855)
855   format(/' Will either input file have dvals? '$)
      if(iopt.eq.1.or.iopt.eq.3) write(iw,856)
856   format(/' Will input data file have dvals? ',$)
      read(ir,803) answer
      idval=0
      if(answer.eq.'y'.or.answer.eq.'Y') idval=1
      if(iopt.gt.2) write(iw,709)
709   format(/'  What is the factor to convert the units of measure of
     1 the surface into the',/,' units of measure of the grid interval
     2 (e.g. .001 if surface is in m and'/'  grid units in km).  Enter
     3 the negative of this number if the sense of the'/'  surface file
     4 is z positive down.')
      write(iw,809)
809   format(' conversion factor = ',$)
      read(ir,*) convf
      if(iopt.eq.2.or.iopt.eq.4) go to 10
      if(iopt.eq.3) write(iw,706)
706   format(/'  Give the level at which the input data were observed in
     1 the SAME units as those'/'  of the draping surface.  (Remember to
     2 be consistent with whether up or down'/'  is the positive
     3 direction.)')
      write(iw,806)
806   format(' flevel= ',$)
      read(ir,*) flevel
10    if(iopt.gt.2) write(iw,711)
711   format(/'  The Taylor series expands about a z reference level
     1 that should best be near'/'  the mean of the draping surface ao
     2 that the deviation of the surface'/'  above and below that level
     3 is the least amount.  Enter the level in the'/'  SAME units used
     4 for the draping surface.')
      write(iw,811)
811   format(' Enter zlevel (1.e+38 to use default):  ',$)
      read(ir,*) zlevel
      kflag=0
      if(zlevel.ge.1.e+38) kflag=1
15      if(iopt.eq.1.or.iopt.eq.3) go to 17
      if(iopt.gt.2) write(iw,712)
712   format(/'  Enter the level to which to continue the data in the
     1 SAME units'/'  as the draping surface units.')
      write(iw,812)
812   format(' flevel = ',$)
      read(ir,*) flevel
17      nterms=3
      if(iopt.eq.1.or.iopt.eq.3) go to 18
      if(iopt.gt.2) write(iw,730)
730   format(/'  Read documentation for examples on when to use
     1 2 or 3 terms of the Taylor'/'  series expansion.')
      write(iw,830)
830   format(' Want 2 or 3 terms in Taylor expansion? ',$)
      read(ir,*) nterms
18    if(iopt.gt.2) write(iw,713)
713   format(/'  Bandpass (lowpass) filtering is often desirable during
     1 the 1st & 2nd vert.'/'  derivative-taking of the Taylor series
     2 expansion or in the continuation'/'  of flevel to zlevel or vice
     3 versa.  Of course if zlevel=flevel, filtering'/'  the continuation
     4 is not desirable.')
20    write(iw,813)
813   format(' Enter filtering option (-1 for help): ',$)
      read(ir,*) ifilt
      if(ifilt.eq.0) go to 30
      if(ifilt.gt.0.and.ifilt.le.3) go to 25
      write(iw,833)
833   format('   0 - no filtering'/'   1 - filter derivs only'/'   2 -
     1 filter derivs and continuation'/'   3 - filter continuation
     2 only')
      go to 20
25    if(iopt.gt.2) write(iw,714)
714   format(/'  Two wavelengths w1<=w2 describe a (tapered) lowpass
     1 filter.  Remember',/,' wavelength=2 * anomaly width.  Defaults:
     2  w1=2*grid interval, w2=w1+2*grid'/'  interval.  w1 &
     3 w2 are in SAME units as grid interval.')
      write(iw,814)
814   format(' Enter w1 and w2 (type 0 0 to use default) ',$)
      read(ir,*) w1,w2
      if(w1.gt.w2) go to 25
      if(w2.ne.0.) go to 30
      w1=1.e+38
      w2=1.e+38
      kflag=kflag+2
30    if(iopt.gt.2) write(iw,724)
724   format(/'  It often helps the FFT operation to extend the number
     2 of rows and',/,' columns by a small proportion of the total grid.
     3  Enter the number to'/'  add to the cols and rows:')
      write(iw,824)
824   format(' nadd = ',$)
      read(ir,*) nadd
      if(iopt.eq.1.or.iopt.eq.3) go to 32
31    if(iopt.gt.2) write(iw,725)
725   format(/'  The approximation of data on the level is found through
     1 successive'/'  iterations (1 or 2).  More iterations doesn''t
     2 neccessarily mean a better'/'  approximation though (see docu
     3mentation).  Enter iter=-1 if'/'  you want 2 iterations but also
     4 wish to save the grid of the 1st iteration.')
      write(iw,825)
825   format(' Enter iter: ',$)
      read(ir,*) iter
      if(iter.eq.-1.or.iter.le.2.and.iter.ge.1) go to 32
      write(iw,725)
      go to 31
c
c  create command file if requested
c
32    if(iopt.gt.2) write(iw,715)
```

```
715   format(/' Now I can create a command file with all the
     1 parameters you picked',/,' so that you can edit and/or use it
     2 for a later program run.')
      write(iw,860)
860   format(/' Want to save this info in command file? ',$)
      read(ir,803) answer
      if(answer.ne.'y'.and.answer.ne.'Y') go to 51
      open(icmd,file=command,status='unknown',form='formatted',
     1carriagecontrol='list')
      if(kflag.gt.0) go to 35
      write(icmd,818) flevel,convf,nterms,zlevel,ifilt,wl,w2,nadd,idval
818   format(' &parms  flevel=',gl4.6,',convf=',gl4.6,',nterms=',i1,
     2',zlevel=',gl4.6,',',/,' ifilt=',i1,',wl=',gl4.6,',w2=',gl4.6,/,
     3' nadd=',i2,',idval=',i1,' &')
      go to 40
35    if(kflag.gt.1) go to 37
      write(icmd,819) flevel,convf,nterms,ifilt,wl,w2,nadd,idval
819   format(' &parms flevel=',gl4.6,',convf=',gl4.6,',nterms=',i1,
     1/,' ifilt=',i1,',wl=',gl4.6,',w2=',gl4.6,/,' nadd=',i2,',idval=',
     2 i1,' &')
      go to 40
37    if(kflag.eq.2) go to 38
      write(icmd,719) flevel,convf,nterms,ifilt,nadd,idval
719   format(' &parms flevel=',gl4.6,',convf=',gl4.6,',nterms=',i1,
     1/,' ifilt=',i1,',nadd=',i2,',idval=',i1,' &')
      go to 40
38    write(icmd,720) flevel,convf,nterms,zlevel,ifilt,nadd,idval
720   format(' &parms flevel=',gl4.6,',convf=',gl4.6,',nterms=',i1,
     1/' zlevel=',gl4.6,',ifilt=',i1,',nadd=',i2,',idval=',i1,' &')
40    close(icmd)
      write(iw,820)
820   format(/' **command file taylor.cmd created**')
      go to 51
c
c  get parms from existing command file
c
c  set default values of parms
50    nadd=0
      convf=1.
      ifilt=0
      flevel=0.
      zlevel=1.e+38
      wl=1.e+38
      w2=1.e+38
      nterms=3
      iter=-3
      open(icmd,file=command,form='formatted',status='old',
     1readonly,carriagecontrol='list',err=299)
      call getparms(icmd,iw)
c check for parameter errors
      if(ifilt.lt.0.or.ifilt.gt.3) go to 290
      if(ifilt.ne.0.and.wl.gt.w2) go to 290
      if(idval.ne.0.and.idval.ne.1) go to 290
      if(nadd.lt.0) go to 290
      if(nterms.lt.1.or.nterms.gt.3) go to 290
      if(iopt.eq.1.or.iopt.eq.3) go to 51
      if(iter.eq.-1.or.iter.eq.2.or.iter.eq.1) go to 51
      write(iw,825)
      read(ir,*) iter
49    if(iter.eq.-1.or.iter.eq.2.or.iter.eq.1) go to 51
      write(iw,725)
      write(iw,825)
      read(ir,*) iter
      go to 49
c prompt for input files
51    if(ifile.ne.' ') go to 55
52    if(iopt.eq.1.or.iopt.eq.3) write(iw,704)
704   format(/' Input file (level survey): ',$)
      if(iopt.eq.2.or.iopt.eq.4) write(iw,804)
804   format(/' Input file (draped survey): ',$)
      read(ir,801) ifile
55    open(in,file=ifile,status='old',form='unformatted',readonly,
     1err=297)
      read(in)id,pgm,ncol,nrow,nz,xo,dx,yo,dy
      if(dx.ne.dy) go to 295
      if(sfile.ne.' ') go to 57
56    write(iw,807)
807   format(/' Enter draping or irregular surface grid: ',$)
      read(ir,801) sfile
57    open(is,file=sfile,form='unformatted',readonly,status='old',
     1err=298)
      read(is) id,pgm,ncol2,nrow2,nz,xo2,dx2,yo2,dy2
      if(abs(xo-xo2).gt.0.001.or.abs(yo-yo2).gt.0.001) go to 293
      if(abs(dx-dx2).gt.0.001*abs(dx).or.dx2.ne.dy2) go to 293
      if(nrow.ne.nrow2.or.ncol.ne.ncol2) go to 293
      if(ofile.ne.' ') go to 59
      write(iw,815)
815   format(/' Output file: ',$)
      read(ir,801)ofile
59    if(title.ne.'this is the default title') go to 100
      write(iw,816)
816   format(' Enter title')
      read(ir,817) title
817   format(a56)
c check zero level or set it if default used
100   call reflvl(is,reclevel,nrow,ncol,dx,iw,convf,z)
      if(zlevel.eq.1.e+38) go to 105
      if(abs((zlevel-reclevel)*convf).gt.abs(1.5*dx)) write(iw,821)
     1 zlevel,reclevel
821   format(/' **WARNING: zero level of ',gl4.6,' is more than 1.5*dx
     1 away from',/,'    recommended level of ',gl4.6)
      go to 110
105   zlevel=reclevel
      write(iw,822) zlevel
822   format(/' Using z ref. level of ',gl4.6)
c  set wl and w2 if default values picked
110   if(wl.lt.1.e+38.and.w2.lt.1.e+38.or.ifilt.eq.0) go to 300
      wl=2*dx
```

```
      w2=4*dx
      write(iw,823) w1,w2
823   format(' Using w1,w2= ',g14.6,1x,g14.6)
      go to 300
c
c ERROR  MESSAGES
c
290   write(iw,292)
292   format(' Command file has parameter error.  Abort.')
      stop
293   write(iw,294)
294   format(' Grids don''t match')
      stop
295   write(iw,296)
296   format(' dx must = dy')
      stop
297   write(iw,291) ifile
291   format(3x,a12,' not found or not binary.  Try again. ')
      go to 52
298   write(iw,291) sfile
      go to 56
299   write(iw,391) command
391   format(' File not found or not ascii.  Try again.')
      go to 1
300   return
      end
c
c this subroutine reads parameters from the command file in namelist
c format
c
      subroutine getparms(icmd,iw)
      character*56 title
      character*50 ifile,ofile,sfile
      common/12dparms/flevel,convf,zlevel,ifilt,w1,w2,
     1idval,nadd,nterms,ifile,ofile,sfile,title,iter
      namelist/parms/flevel,convf,zlevel,nterms,
     1ifilt,w1,w2,idval,nadd,ifile,ofile,sfile,title,iter
      read(icmd,parms,end=10)
      return
10    write(iw,800)
800   format(' odd EOF detected in cmd file.')
      stop
      end
c ********************************************************************
c      This subroutine recommends a reference level for the
c       level-to-drape operation.
c ********************************************************************
      subroutine reflvl(is,rlevel,nrow,ncol,dx,iw,convf,z)
      dimension z(ncol),id(14),pgm(2)
      data zmax/-1.e38/,zmin/1.e38/
c
c  calculate min, max and mean of the draping surface
c
      sumz=0.0
      en=0.0
      do 20 j=1,nrow
      read(11,end=80,err=70) dum,z
      do 10 i=1,ncol
c convert surf. file to z positive down and into units of dx
c calculate mean and std dev
      if(z(i).ge.1.e+38) go to 10
      z(i)=z(i)*convf
      sumz=sumz+z(i)
      if(z(i).gt.zmax) zmax=z(i)
      if(z(i).lt.zmin) zmin=z(i)
      en=en+1.0
10    continue
      if(j.eq.1.and.en.eq.0.0) go to 60
20    continue
      rewind is
      read(is) id,pgm,ncol,nrow,nz,xo,dx,yo,dy
      zmean=sumz/en
c
c  Ideally zmax-rlevel should be <= .9*dx (distance method will need to
c  approximate downward) and rlevel-zmin <= 1.5*dx (distance upward).
c  Try to get close to this rlevel.  Use the mean if conditions are
c  satisfied with rlevel=zmean.  Round off rlevel.
c
      rlevel=zmean
      if(zmax-rlevel.le.0.9*dx.and.rlevel-zmin.le.1.5*dx) go to 50
      rtest1=zmax-.9*dx
      rtest2=zmin+1.5*dx
      rlevel=(rtest1+rtest2)*.5
      if(zmax-rlevel.le.1.2*dx.and.rlevel-zmin.le.2.0*dx) go to 50
      write(iw,800)
800   format(/' **WARNING: Relief of draping surface is greater than
     1 dx.',/,' This could cause the method to diverge.  Try
     2 a coarser dx?')
c  round off rlevel
50    if(rlevel.eq.0.0) go to 55
      ntest=2-ifix(alog10(rlevel))
      test=10.e+0**float(ntest)
      rtest=float(nint(rlevel*test))
      rlevel=rtest/test
c change units of rlevel back and return
55    rlevel=rlevel/convf
      return
60    write(iw,801)
801   format(' No data in entire first row of surface file.  Can''t
     1 handle it.')
      stop
70    write(iw,802)
802   format(' Error in read of surface file.')
      stop
80    write(is,803)
803   format(' unexpected eof in surface file.')
      stop
      end
```

```
C*********************************************************************
C  subroutine '12d' removes flagged values from grid, transforms
C  data, calls routines to apply filter,  performs level-to-drape
C  Taylor's series expansion, and writes resulting grid.
C  Note that both input data files and output file must be open and
C  have had headers read before calling this subroutine.  They will
C  remain open upon return.
C  Guts of this subroutine lifted from program FFTFIL by Thomas G.
C  Hildenbrand, 1983, USGS Open-File Report 83-237.
C     f      =      input data
C     g      =      storage for 1st vertical derivative of f
C     h      =      storage for 2nd vertical derivative of f
C     work   =      work array for subroutine sfftmg
C     n1     =      number of rows
C     n2     =      number of cols
C     id2    =      2 * max(n1,n2)
C     nri    =
C     z      =      surface data
C     nxa    =
C
C*********************************************************************
      Subroutine 12d(f,g,h,work,id2,nri,n1,n2,z,nxa)
      Dimension f(2,n2,nri),g(2,n2,nri),h(2,n2,nri),work(id2),
     1jflag(2,20),z(n2)
      Character*50 ifile,ofile,sfile,command
      Character*56 title
      Common /12dparms/flevel,convf,rlevel,ifilt,w1,w2,idval,nadd,
     1nterms,ifile,ofile,sfile,title,iter
      Common /12dinfo/pi,p2,nz,yo,xo,dx,dy,ny,nx
      Common /units/iw,ir,in,iout,icmd,is,iunita,iunitb,iunitc,if,
     1iunitg,iunith
      Data pi2/6.28318531/,dval/'ffff7fff'x/
      n22=2*n2
      If (ifilt.EQ.0)Go To 20
      If (w1.NE.0.0)Go To 5
      f4=1.0e+30
      Go To 7
    5 f4=1./w1
    7 If (w2.NE.0.0)Go To 8
      f3=1.0e+30
      Go To 10
    8 f3=1./w2
   10 If (f3.NE.f4)denom=1./(f4-f3)
C  compute beginning and end column & row in original grid.
   20 Do 30 i=1,2
      Do 30 j=1,n2
   30 f(2,j,i)=0.0
      napl=nadd+1
      n2a=n2-nadd
      nxapl=nxa*0.5+1
      fxapl=float(nxa)/2.+1.
      If (abs(fxapl-float(nxapl)).GT.0.0001)nxapl=nxapl+1
      nla=nxapl-1+nx
C  remove flagged values and add '2*nadd' rows & columns to
C  reduce the effect of gibbs phenomena.
      Open (iunita,access='direct',status='new',form='unformatted',
     1file='slavel.tmp',recl=n22,maxrec=n1,
     1organization='relative',recordtype='fixed')
      If (idval.NE.0)Open (if,status='new',form='unformatted',
     1file='flag.loc')
      irr=1
      iwr=0
      jr=0
   40 jr=jr+1
      If (idval.EQ.0)Go To 310
C   idval=1: removal of flagged values, create file flag.loc
      If (jr.GT.1)Go To 140
C   read 1st row & remove flagged values.
      Read (in)dum,(f(1,i,1),i=napl,n2a)
      Read (in)dum,(f(1,i,2),i=napl,n2a)
      nflag=1
      i=napl
   50 jflag(1,1)=napl
      If (f(1,napl,1).LE.1.0e38)Go To 90
      Do 60 i=napl,n2a
   60 If (f(1,i,1).LE.1.0e38)Go To 70
      Go To 1270
   70 jflag(1,1)=i
      aval=f(1,i,1)
      Do 80 ii=napl,i-1
   80 f(1,ii,1)=aval
   90 If (i.GE.n2a)Go To 100
      Do 100 ii=i+1,n2a
      If (f(1,ii,1).LE.1.0e38)Go To 100
      jflag(2,nflag)=ii-1
      Go To 110
  100 Continue
      jflag(2,nflag)=n2a
      Write (if)nflag,(jflag(1,j),jflag(2,j),j=1,nflag)
      Go To 350
  110 aval=f(1,ii-1,1)
      Do 120 i=ii,n2a
      If (f(1,i,1).LE.1.0e38)Go To 130
  120 f(1,i,1)=aval
      Write (if)nflag,(jflag(1,j),jflag(2,j),j=1,nflag)
      Go To 350
  130 f(1,i-1,1)=(f(1,i-1,1)+f(1,i,1))*0.5
      nflag=nflag+1
      jflag(1,nflag)=i
C   1st row completed: now extend grid & write to disk.
      Go To 90
C   remove flagged values from remaining rows.
```

```
140 If (jr.GE.nx)Go To 150
    Read (in)dum,(f(1,i,3),i=napl,n2a)
    Go To 170
150 Do 160 i=napl,n2a
160 f(1,i,3)=dval
170 nflag=1
    i=napl
    jflag(1,1)=napl
    If (f(1,napl,2).LE.1.0e38)Go To 220
    Do 180 i=napl,n2a
180 If (f(1,i,2).LE.1.0e38)Go To 190
    i=napl
    jflag(1,1)=n2a+1
    If (f(1,napl,3).GE.1.0e38)f(1,napl,2)=f(1,napl,1)
    If (f(1,napl,3).LE.1.0e38)f(1,napl,2)=(f(1,napl,1)+
   1f(1,napl,3))*0.5
    Go To 220
190 jflag(1,1)=i
    ii=i
    Do 210 irev=napl,i-1
    ii=ii-1
    If (f(1,ii,3).GE.1.0e38)Go To 200
    f(1,ii,2)=(f(1,ii,1)+f(1,ii+1,2)+f(1,ii,3))*0.3333333
    Go To 210
200 f(1,ii,2)=(f(1,ii,1)+f(1,ii+1,2))*0.5
210 Continue
220 If (i.GE.n2a)Go To 230
    Do 230 ii=i+1,n2a
    If (f(1,ii,2).LE.1.0e38)Go To 230
    jflag(2,nflag)=ii-1
    Go To 240
230 Continue
    jflag(2,nflag)=n2a
    Write (if)nflag,(jflag(1,j),jflag(2,j),j=1,nflag)
    Go To 390
240 Do 290 i=ii,n2a
    If (i.EQ.n2a)Go To 250
    If (f(1,i+1,2).LE.1.0e38)Go To 270
250 If (f(1,i,3).GE.1.0e38)Go To 260
    f(1,i,2)=(f(1,i,1)+f(1,i-1,2)+f(1,i,3))*0.3333333
    Go To 290
260 f(1,i,2)=(f(1,i,1)+f(1,i-1,2))*0.5
    Go To 290
270 If (f(1,i,3).GE.1.0e38)Go To 280
    f(1,i,2)=(f(1,i,1)+f(1,i-1,2)+f(1,i+1,2)+f(1,i,3))*0.25
    Go To 300
280 f(1,i,2)=(f(1,i,1)+f(1,i-1,2)+f(1,i+1,2))*0.3333333
    Go To 300
290 Continue
    Write (if)nflag,(jflag(1,j),jflag(2,j),j=1,nflag)
    Go To 390
300 nflag=nflag+1
    i=i+1
    jflag(1,nflag)=i
    Go To 220
C   idval.eq.0:  check input data.
 310 If (jr.NE.1)irr=2
    Read (in)dum,(f(1,i,irr),i=napl,n2a)
    Do 320 ii=napl,n2a
 320 If (f(1,ii,irr).GE.1.0e38)Go To 330
    If (jr.NE.1)Go To 390
    Go To 350
 330 Write (iw,340)jr,ii
 340 Format (' #flagged value row=',i4,3x,'col=',i4,' may be more?')
    Close (in)
    Close (iunita)
    Return
C   extend row & write to disk.
 350 Do 360 i=1,napl
 360 f(1,i,1)=f(1,napl,1)
    Do 370 i=n2a,n2
 370 f(1,i,1)=f(1,n2a,1)
 380 iwr=iwr+1
    Call swrda(iunita,iwr,f,n22)
    If (iwr.EQ.nxapl)Go To 40
    Go To 380
 390 Do 400 i=1,napl
 400 f(1,i,2)=f(1,napl,2)
    Do 410 i=n2a,n2
 410 f(1,i,2)=f(1,n2a,2)
    iwr=iwr+1
    Call swrda(iunita,iwr,f(1,1,2),n22)
    If (iwr.GE.nla)Go To 430
    If (idval.LT.1)Go To 40
    Do 420 i=napl,n2a
    f(1,i,1)=f(1,i,2)
 420 f(1,i,2)=f(1,i,3)
    Go To 40
 430 If (iwr.GE.nl)Go To 450
 440 iwr=iwr+1
    Call swrda(iunita,iwr,f(1,1,2),n22)
    If (iwr.LT.nl)Go To 440
 450 If (idval.NE.0)Close (if)
 460 Format (i5)
C  obtain complex fourier coefficients (f.f.t. f).
 590 Open (iunitb,access='direct',status='new',form='unformatted',
   1file='slave2.tmp',recl=n22,maxrec=nl
   2,organization='relative',recordtype='fixed')
C
C    Transform data
C
 980 isign=-1
    Call sfftmg(iunita,iunitb,n2,nl,f,nri,isign,work)
C  f array now holds complex fourier coefficients.
C  start filtering
1000 Go To (1005,1003,1001),nterms
1001 Open (iunith,access='direct',status='new',form='unformatted',
    lfile='slave4.tmp',recl=n22,maxrec=nl,organization='relative',
```

```
     2recordtype='fixed')
 1003 Open (iunitg,access='direct',status='new',form='unformatted',
     1file='slave3.tmp',recl=n22,maxrec=nl,organization='relative',
     2recordtype='fixed')
 1005 d=(flevel-rlevel)*convf
      cd=pi2*d
      c1=pi2
      n1p1=n1+1
      n2p1=n2+1
      nnh1=(n1/2)+1
      nnh2=(n2/2)+1
C  operating on f fourier coefficients.
      rn1=1./(float(n1)*dx)
      rn2=1./(float(n2)*dy)
      jr=0
      jmr=n1p1
      igo=1
      If (ifilt.EQ.0)Go To 1010
      igo=2
      If (nterms.EQ.1.AND.ifilt.LE.1)igo=3
      If (nterms.EQ.1.AND.ifilt.GE.2)igo=4
      dumc=1.0
      dumd=1.0
      If (ifilt.LE.2)dumd=0.0
      If (ifilt.GE.2)dumc=0.0
 1010 jr=jr+1
      If (jr.GT.nnh1)Go To 1110
      jj=jr-1
      x=float(jj)*rn1
      xsq=x*x
      Call srdda(iunita,jr,f,n22)
      Do 1090 i=1,n2
      ii=i-1
      If (i.GT.nnh2)ii=-(n2p1-i)
      y=float(ii)*rn2
      xysq=xsq+y*y
      w=sqrt(xysq)
      fr=f(1,i,1)
      fi=f(2,i,1)
      Go To (1020,1030,1060,1070),igo
C modify coefs, no bandpass filtering
 1020 e=exp(cd*w)
      e1=e*c1*w
      e2=e1*c1*w
      f(1,i,1)=fr*e
      f(2,i,1)=fi*e
      g(1,i,1)=fr*e1
      g(2,i,1)=fi*e1
      h(1,i,1)=fr*e2
      h(2,i,1)=fi*e2
      Go To 1090
C  modify coefs while bandpass filtering, dumd and dumc are switches to
C  filter or not to filter depending on ifilt
C      dumd=1.0, no filter of derivs; dumc=1.0, no filter of continuation
 1030 If (w.LE.f4)Go To 1050
      e=exp(cd*w)
      f(1,i,1)=fr*e*dumc
      f(2,i,1)=fi*e*dumc
      e1=e*c1*w*dumd
      e2=e1*c1*w
      g(1,i,1)=fr*e1
      g(2,i,1)=fi*e1
      h(1,i,1)=fr*e2
      h(2,i,1)=fi*e2
      Go To 1090
 1050 factor=1.0
      If (w.GT.f3)factor=(f4-w)*denom
      e=exp(cd*w)
      factord=(1.0-dumd)*factor+dumd
      factorc=(1.-dumc)*factor+dumc
      e1=e*c1*w*factord
      e2=e1*c1*w
      f(1,i,1)=fr*e*factorc
      f(2,i,1)=fi*e*factorc
      g(1,i,1)=fr*e1
      g(2,i,1)=fi*e1
      h(1,i,1)=fr*e2
      h(2,i,1)=fi*e2
      Go To 1090
C continuation only (drape-to-level), no bandpass filtering
 1060 e=exp(cd*w)
      f(1,i,1)=fr*e
      f(2,i,1)=fi*e
      Go To 1090
C  continuation only with bandpass filtering
 1070 If (w.LE.f4)Go To 1075
      e=exp(cd*w)
      f(1,i,1)=0.0
      f(2,i,1)=0.0
      Go To 1090
 1075 factor=1.0
      If (w.GT.f3)factor=(f4-w)*denom
      e=exp(cd*w)*factor
      f(1,i,1)=fr*e
      f(2,i,1)=fi*e
 1090 Continue
      Call swrda(iunita,jr,f,n22)
      Go To (1095,1093,1092),nterms
 1092 Call swrda(iunith,jr,h,n22)
 1093 Call swrda(iunitg,jr,g,n22)
 1095 If (jr.EQ.1)Go To 1010
      jmr=jmr-1
      If (jr.EQ.jmr)Go To 1110
      f(2,1,1)=-f(2,1,1)
      g(2,1,1)=-g(2,1,1)
      h(2,1,1)=-h(2,1,1)
      iwr=n2p1
      Do 1100 i=2,nnh2
```

```
      iwr=iwr-1
      x=f(2,i,1)
      y=f(1,i,1)
      f(2,i,1)=-f(2,iwr,1)
      f(1,i,1)=f(1,iwr,1)
      f(1,iwr,1)=y
      f(2,iwr,1)=-x
      x=g(2,i,1)
      y=g(1,i,1)
      g(2,i,1)=-g(2,iwr,1)
      g(1,i,1)=g(1,iwr,1)
      g(1,iwr,1)=y
      g(2,iwr,1)=-x
      x=h(2,i,1)
      y=h(1,i,1)
      h(2,i,1)=-h(2,iwr,1)
      h(1,i,1)=h(1,iwr,1)
      h(1,iwr,1)=y
      h(2,iwr,1)=-x
 1100 Continue
      Call swrda(iunita,jmr,f,n22)
      Go To (1010,1105,1102),nterms
 1102 Call swrda(iunith,jmr,h,n22)
 1105 Call swrda(iunitg,jmr,g,n22)
      Go To 1010
C  f array now holds complex coefficients operated on by filter
C  compute desired anomaly = inv. f.f.t. of f
 1110 isign=1
      Call sfftmg(iunita,iunitb,n2,nl,f,nri,isign,work)
      Go To (1120,1115,1113),nterms
 1113 Call sfftmg(iunith,iunitb,n2,nl,h,nri,isign,work)
 1115 Call sfftmg(iunitg,iunitb,n2,nl,g,nri,isign,work)
C  f array now holds computed anomaly on datum = z.
C  do level to drape operation and write to output grid
 1120 If (idval.NE.0)Open (if,status='old',form=
     1'unformatted',file='flag.loc')
      area=1./float(nl*n2)
      dum=0.0
      iwr=0
      dumh=1.0
      If (nterms.EQ.2)dumh=0.0
      If (nterms.EQ.1)Go To 1215
      Do 1210 jr=nxapl,nla
      iwr=iwr+1
      Call srdda(iunita,jr,f,n22)
      Go To (300,1124,1123),nterms
 1123 Call srdda(iunith,jr,h,n22)
 1124 Call srdda(iunitg,jr,g,n22)
      Read (is)dum,(z(k),k=1,ny)
      If (idval.EQ.0)Go To 1170
      Read (if)nflag,(jflag(1,j),jflag(2,j),j=1,nflag)
      j=1
      ik=0
C   restore shape of original grid (i.e. insert flagged values).
      Do 1160 i=napl,n2a
      ik=ik+1
      If (i.LT.jflag(1,j))Go To 1130
      If (i.EQ.jflag(2,j))Go To 1140
      Go To 1150
 1130 f(1,i,1)=dval
      Go To 1160
 1140 j=j+1
      If (j.LE.nflag)Go To 1150
      j=j-1
      jflag(1,j)=1024
C  convert z(ik) to units of grid interval, positive down, and
C  into units of distance from reference level
 1150 If (z(ik).GE.1.e+38)Go To 1155
      z(ik)=(rlevel-z(ik))*convf
C  Taylor's series expansion.  dumh dummies out the 2nd deriv term
C    depending on nterms
      f(1,i,1)=f(1,i,1)+z(ik)*g(1,i,1)+z(ik)*z(ik)*h(1,i,1)*dumh
      f(1,i,1)=f(1,i,1)*area
      Go To 1160
 1155 f(1,i,1)=dval
 1160 Continue
      Go To 1190
 1170 ik=0
      Do 1180 i=napl,n2a
      ik=ik+1
      If (z(ik).GE.1.e+38)Go To 1175
      z(ik)=(rlevel-z(ik))*convf
      f(1,i,1)=f(1,i,1)+z(ik)*g(1,i,1)+z(ik)*z(ik)*h(1,i,1)*dumh
      f(1,i,1)=f(1,i,1)*area
      Go To 1180
 1175 f(1,i,1)=dval
 1180 Continue
 1190 Write (iout)dum,(f(1,i,1),i=napl,n2a)
 1210 Continue
      Go To 1250
C restore shape of grid- continuation only operation
 1215 Do 1247 jr=nxapl,nla
      iwr=iwr+1
      Call srdda(iunita,jr,f,n22)
      If (idval.EQ.0)Go To 1235
      Read (if)nflag,(jflag(1,j),jflag(2,j),j=1,nflag)
      j=1
      Do 1230 i=napl,n2a
      If (i.LT.jflag(1,j))Go To 1220
      If (i.EQ.jflag(2,j))Go To 1225
      Go To 1227
 1220 f(1,i,1)=dval
      Go To 1230
 1225 j=j+1
      If (j.LE.nflag)Go To 1227
      j=j-1
      jflag(1,j)=1024
 1227 f(1,i,1)=f(1,i,1)*area
```

```
 1230 Continue
      Go To 1245
 1235 Do 1240 i=napl,n2a
      f(1,i,l)=f(1,i,l)*area
 1240 Continue
 1245 Write (iout)dum,(f(1,i,l),i=napl,n2a)
 1247 Continue
 1250 If (idval.EQ.1)Close (if,status='delete')
      Close (iunitc,status='delete')
      Close (iunita,status='delete')
      Close (iunitb,status='delete')
      Go To (1290,1265,1263),nterms
 1263 Close (iunith,status='delete')
 1265 Close (iunitg,status='delete')
      Go To 1290
 1270 Write (iw,1280)
 1280 Format (' #entire first row has no data--program aborted')
      Close (in)
      Close (iunita,status='delete')
      If (idval.GT.0)Close (if,status='delete')
 1290 Return
      End
C *********************************************************************
      Subroutine extend(nadd,n1,n2,id2,lnri,nri,nxa)
C  this subroutine extends the cols of grid if nadd not 0 and finds the
C  best no. of rows to use in the fft routines.  Lifted from program
C  FFTFIL by Thomas G. Hildenbrand.
C *********************************************************************
      Common /12dinfo/p1,p2,nz,yo,xo,dx,dy,ncol,nrow
      n1=nrow
      n2=ncol
      l=lnri+1
      lnri2l=lnri/2+1
C  set no. of rows for fft: need m=1*2**k, m.gt. or .eq. nx+2*nadd.
C  m=no. of rows, l=no. from 9-16, k=interger.
      n1=n1+2*nadd
  190 l=l-1
      If (l.LT.lnri2l)Go To 260
      mr=n1/l+0.0000001
      k=1
      idiv=2
  200 iquot=mr/idiv+0.0000001
      If (iquot.LT.idiv)Go To 210
      k=k+1
      mr=iquot
      Go To 200
  210 k=k+1
      m=1*2**k
  220 mtest=1*2**(k-1)
      If (mtest.LT.n1)Go To 230
      k=k-1
      m=mtest
      If (k.EQ.0)Go To 230
      Go To 220
  230 lnxa=m-n1
      If (l.NE.lnri)Go To 250
      nxa16=lnxa
  240 nri=l
      nxa=lnxa
      Go To 190
  250 If (lnxa.GE.nxa)Go To 190
      Go To 240
  260 n1=n1+nxa
C  check to see if row block size of 16 will be more efficient
      n116=n1-nxa+nxa16
      ntest=0.9*n116
      If (ntest.GT.n1.OR.n116.GT.1024)Go To 270
      n1=n1-nxa+nxa16
      nxa=nxa16
      nri=16
  270 n2=n2+2*nadd
      If (n1.GT.1024.OR.n2.GT.1024)Go To 320
      nxa=nxa+2*nadd
      id2=n1
      If (n2.GT.n1)id2=n2
      id2=2*id2
      Go To 300
  320 Write (iw,330)nrow,ncol,n1,n2
  330 Format (' no. of extended rows or cols exceeds 1024:',/,' input
     1no. of rows and cols=',2i4,/,'no. of rows and cols required for
     2 filtering=',2i4,/)
  340 Stop
  300 Return
      End
c*********************************************************************
      subroutine sfftmg(unita,unitb,n,m,f,l,isign,work)
c     performs 2-dimensional fast fourier transform using disk
c     arguments -
c       unita - unit number of open random access disk file containing
c               one complex data row per record.
c               this file is used for input to fftmg and output
c               of the transform from fftmg.
c       unitb - unit number of open random access work file
c               the same as the file of unita.
c       n     - number of y columns = length of x rows.
c       m     - number of x rows = length of y columns,
c               must be of the form 1*2**k (k integer).
c       f     - complex work array for core manipulations.
c       l     - number of x rows which can be held in f.
c       isign - +1 or -1 for forward or reverse transform.
c       work  - work array for in-core transform routine sfourt.
c               work must be complex, of length max(n,l).
c   developed and coded by ray watts--usgs,denver
c   note: randin,randou,inset, and outset are entry points
c   in subroutine sdummy
c*********************************************************************
      integer unita,unitb,output,ndim(2)
      complex f(n,l),twidle,twidlf,w,work(l)
```

```
      nfourt=2*n*l
      nwork=2*n
      if(l.gt.n)nwork=2*l
c     ************************************************************
c     *    the number of disk accesses (reads+writes) used by    *
c     *    this program is m*(4*log2(m/l)+3).                    *
c     ************************************************************
c     ------------------------------------------------------------
c     check arguments
      if(l*(m/l).ne.m)stop 'l not a factor of m in fftmg'
      ml=m/l
      mt=l
   10 if(mt-ml)20,40,30
   20 mt=2*mt
      go to 10
   30 stop 'm/l not a power of 2 in fftmg'
   40 continue
c     ------------------------------------------------------------
c     ------------------------------------------------------------
c     start processing
c     skip decimation if l .eq. m
      if(l.eq.m)go to 80
c     set starting input and output units
      input=unita
      output=unitb
c     set starting blocksize for decimation in input domain
      iblock=m
c     ------------------------------------------------------------
c     decimate this blocksize
c     set up sub-block size
   50 isub=iblock/2
c     set up for random i/0
      call inset(input)
      call outset(output)
c     for each block of this size
      do 60 iblst=l,m,iblock
c     copy even members into first sub-block, odd members into second
      do 60 i=l,isub
      call randin(n,f,2*(i-l)+iblst)
      call randou(n,f,i-l+iblst)
      call randin(n,f,2*i-l+iblst)
   60 call randou(n,f,i-l+iblst+isub)
c     swap input/output functions
      inputt=input
      input=output
      output=inputt
c     next smaller blocksize
      iblock=isub
c     loop if not done
      if(iblock.gt.l)go to 50
c     decimation complete
c     ------------------------------------------------------------
c     ------------------------------------------------------------
c     ensure that data set is back on unita
      if(input.eq.unita)go to 80
      call inset(input)
      call outset(output)
      do 70 i=l,m
      call randin(n,f,i)
   70 call randou(n,f,i)
   80 continue
c     ------------------------------------------------------------
c     start transform.
c     set block size
      iblock=l
c     set up for random i/o
      call inset(unita)
      call outset(unita)
c     run through the blocks
      do 100 istart=l,m,iblock
      into=l
c     run through the records in a block
      do 90 irecrd=istart,istart+iblock-l
c     read the record
      call randin(n,f(l,into),irecrd)
c     proceed with next record in block
   90 into=into+l
c     fourier transform in both directions
      ndim(l)=n
      ndim(2)=l
      call sfourt(f,ndim,2,isign,l,work,nfourt,nwork)
c     ************************************************
c     write the block back onto disk
      ifrom=l
      do 100 irecrd=istart,istart+iblock-l
      call randou(n,f(l,ifrom),irecrd)
  100 ifrom=ifrom+l
c     if l.eq.m, the transform is now complete.
      if(l.eq.m)return
c     done with first-step processing
c     ------------------------------------------------------------
c     finish the processing by pairs of rows, applying twiddle factors
c     set the twiddle generator
      twidlf=cexp(cmplx(0.,3.141593*isign/(iblock)))
c     run through the blocks by pairs
  110 do 140 istart=l,m,iblock*2
c     start the twiddle factor
      twidle=(l.,0.)
c     scan the block pair
      do 130 irecrd=istart,istart+iblock-l
      jrecrd=irecrd+iblock
      call randin(n,f,irecrd)
      call randin(n,f(l,2),jrecrd)
c     combine, using twiddle factor
      do 120 i=l,n
      w=twidle*f(i,2)
      f(i,2)=f(i,1)-w
  120 f(i,1)=f(i,1)+w
```

```
c     write back onto disk
      call randou(n,f,irecrd)
      call randou(n,f(1,2),jrecrd)
c     increment the twiddle factor
  130 twidle=twidle*twidlf
c     go on to next block pair
  140 continue
c     increase the block size, change twiddle increment
      iblock=2*iblock
      twidlf=csqrt(twidlf)
      if(isign*aimag(twidlf).lt.0)twidlf=-twidlf
c     check if done
      if(iblock.lt.m)go to 110
c     ------------------------------------------------------------
c     done processing
      end
c ***********************************************************************
      subroutine srdda(no,ipos,dat,n)
c   this subroutine reads keyed sequential files.
c ***********************************************************************
      dimension dat(n)
      read(no'ipos)dat
      return
      end
c ***********************************************************************
      subroutine swrda(no,ipos,dat,n)
c   this subroutine writes keyed sequential files
c ***********************************************************************
      dimension dat(n)
      write(no'ipos)dat
      return
      end
c***********************************************************************
      subroutine sfourt(data,nn,ndim,isign,iform,work,nfourt,nwork1)
c  note that 'nfourt' and 'nwork1' in subroutine call statement are
c  revised features of 'sfourt'. they are used as dynamic dimensioning
c  for the 'data' and 'work' arrays
c     the cooley-tukey fast fourier transform in usasi basic fortran
c     transform(j1,j2,) = sum(data(i1,i2,)*w1**((i1-1)*(j1-1))
c                              *w2**((i2-1)*(j2-1))*,),
c     where i1 and j1 run from 1 to nn(1) and w1=exp(isign*2*pi*
c     sqrt(-1)/nn(1)), etc.  there is no limit on the dimensionality
c     (number of subscripts) of the data array.  if an inverse
c     transform (isign=+1) is performed upon an array of transformed
c     (isign=-1) data, the original data will reappear.
c     multiplied by nn(1)*nn(2)*,  the array of input data must be
c     in complex format.  however, if all imaginary parts are zero (i.e.
c     the data are disguised real) running time is cut up to forty per-
c     cent.  (for fastest transform of real data, nn(1) should be even.)
c     the transform values are always complex and are returned in the
c     original array of data, replacing the input data.  the length
c     of each dimension of the data array may be any integer.  the
c     program runs faster on composite integers than on primes, and is
c     particularly fast on numbers rich in factors of two.
c     timing is in fact given by the following formula.  let ntot be the
c     total number of points (real or complex) in the data array, that
c     is, ntot=nn(1)*nn(2)*...  decompose ntot into its prime factors,
c     such as 2**k2 * 3**k3 * 5**k5 * ...  let sum2 be the sum of all
c     the factors of two in ntot, that is, sum2 = 2*k2.  let sumf be
c     the sum of all other factors of ntot, that is, sumf = 3*k3*5*k5*..
c     the time taken by a multidimensional transform on these ntot data
c     is t = t0 + ntot*(t1+t2*sum2+t3*sumf).  on the cdc 3300 (floating
c     point add time = six microseconds), t = 3000 + ntot*(600+40*sum2+
c     175*sumf) microseconds on complex data.
c     implementation of the definition by summation will run in a time
c     proportional to ntot*(nn(1)+nn(2)+...).  for highly composite ntot
c     the savings offered by this program can be dramatic.  a one-dimen-
c     sional array 4000 in length will be transformed in 4000*(600+
c     40*(2+2+2+2+2)+175*(5+5+5)) = 14.5 seconds versus about 4000*
c     4000*175 = 2800 seconds for the straightforward technique.
c     the fast fourier transform places three restrictions upon the
c     data.
c     1.  the number of input data and the number of transform values
c     must be the same.
c     2.  both the input data and the transform values must represent
c     equispaced points in their respective domains of time and
c     frequency.  calling these spacings deltat and deltaf, it must be
c     true that deltaf=2*pi/(nn(i)*deltat).  of course, deltat need not
c     be the same for every dimension.
c     3.  conceptually at least, the input data and the transform output
c     represent single cycles of periodic functions.
c     the calling sequence is--
c     call fourt(data,nn,ndim,isign,iform,work)
c     data is the array used to hold the real and imaginary parts
c     of the data on input and the transform values on output.  it
c     is a multidimensional floating point array, with the real and
c     imaginary parts of a datum stored immediately adjacent in storage
c     (such as fortran iv places them).  normal fortran ordering is
c     expected, the first subscript changing fastest.  the dimensions
c     are given in the integer array nn, of length ndim.  isign is -1
c     to indicate a forward transform (exponential sign is -) and +1
c     for an inverse transform (sign is +).  iform is +1 if the data are
c     complex, 0 if the data are real.  if it is 0, the imaginary
c     parts of the data must be set to zero.  as explained above, the
c     transform values are always complex and are stored in array data.
c     work is an array used for working storage.  it is floating point
c     real, one dimensional of length equal to twice the largest array
c     dimension nn(i) that is not a power of two.  if all nn(i) are
c     powers of two, it is not needed and may be replaced by zero in the
c     calling sequence.  thus, for a one-dimensional array, nn(1) odd,
c     work occupies as many storage locations as data.  if supplied,
c     work must not be the same array as data.  all subscripts of all
c     arrays begin at one.
c     example 1.  three-dimensional forward fourier transform of a
c     complex array dimensioned 32 by 25 by 13 in fortran iv.
c     dimension data(32,25,13),work(50),nn(3)
c     complex data
c     data nn/32,25,13/
```

```
c     do 1 i=1,32
c     do 1 j=1,25
c     do 1 k=1,13
c  1  data(i,j,k)=complex value
c     call fourt(data,nn,3,-1,1,work)
c     example 2.  one-dimensional forward transform of a real array of
c     length 64 in fortran ii,
c     dimension data(2,64)
c     do 2 i=1,64
c     data(1,i)=real part
c  2  data(2,i)=0.
c     call fourt(data,64,1,-1,0,0)
c     there are no error messages or error halts in this program.  the
c     program returns immediately if ndim or any nn(i) is less than one.
c     program by norman brenner from the basic program by charles
c     rader,  june 1967.  the idea for the digit reversal was
c     suggested by ralph alter.
c     this is the fastest and most versatile version of the fft known
c     to the author.  a program called four2 is available that also
c     performs the fast fourier transform and is written in usasi basic
c     fortran.  it is about one third as long and restricts the
c     dimensions of the input array (which must be complex) to be powers
c     of two.  another program, called four1, is one tenth as long and
c     runs two thirds as fast on a one-dimensional complex array whose
c     length is a power of two.
c     reference--
c     ieee audio transactions (june 1967), special issue on the fft.
c*********************************************************************
      dimension data(nfourt),nn(2),ifact(32),work(nwork1)
      data twopi/6.2831853071796/,rthlf/0.70710678118655/
      if(ndim-1)1190,10,10
   10 ntot=2
      do 20 idim=1,ndim
      if(nn(idim))1190,1190,20
   20 ntot=ntot*nn(idim)
c     main loop for each dimension
      np1=2
      do 1180 idim=1,ndim
      n=nn(idim)
      np2=np1*n
      if(n-1)1190,1170,30
c     is n a power of two and if not, what are its factors
   30 m=n
      ntwo=np1
      if=1
      idiv=2
   40 iquot=m/idiv
      irem=m-idiv*iquot
      if(iquot-idiv)120,50,50
   50 if(irem)70,60,70
   60 ntwo=ntwo+ntwo
      ifact(if)=idiv
      if=if+1
      m=iquot
      go to 40
   70 idiv=3
      inon2=if
   80 iquot=m/idiv
      irem=m-idiv*iquot
      if(iquot-idiv)140,90,90
   90 if(irem)110,100,110
  100 ifact(if)=idiv
      if=if+1
      m=iquot
      go to 80
  110 idiv=idiv+2
      go to 80
  120 inon2=if
      if(irem)140,130,140
  130 ntwo=ntwo+ntwo
      go to 150
  140 ifact(if)=m
c     separate four cases--
c        1. complex transform or real transform for the 4th, 9th,etc.
c           dimensions.
c        2. real transform for the 2nd or 3rd dimension.  method--
c           transform half the data, supplying the other half by con-
c           jugate symmetry.
c        3. real transform for the 1st dimension, n odd.  method--
c           set the imaginary parts to zero.
c        4. real transform for the 1st dimension, n even.  method--
c           transform a complex array of length n/2 whose real parts
c           are the even numbered real values and whose imaginary parts
c           are the odd numbered real values.  separate and supply
c           the second half by conjugate symmetry.
  150 icase=1
      ifmin=1
      i1rng=np1
      if(idim-4)160,210,210
  160 if(iform)170,170,210
  170 icase=2
      i1rng=np0*(1+nprev/2)
      if(idim-1)180,180,210
  180 icase=3
      i1rng=np1
      if(ntwo-np1)210,210,190
  190 icase=4
      ifmin=2
      ntwo=ntwo/2
      n=n/2
      np2=np2/2
      ntot=ntot/2
      i=1
      do 200 j=1,ntot
      data(j)=data(i)
  200 i=i+2
c     shuffle data by bit reversal, since n=2**k.  as the shuffling
c     can be done by simple interchange, no working array is needed
```

```
  210 if(ntwo-np2)290,220,220
  220 np2hf=np2/2
      j=1
      do 280 i2=1,np2,npl
      if(j-i2)230,250,250
  230 i1max=i2+npl-2
      do 240 i1=i2,i1max,2
      do 240 i3=i1,ntot,np2
      j3=j+i3-i2
      tempr=data(i3)
      tempi=data(i3+1)
      data(i3)=data(j3)
      data(i3+1)=data(j3+1)
      data(j3)=tempr
  240 data(j3+1)=tempi
  250 m=np2hf
  260 if(j-m)280,280,270
  270 j=j-m
      m=m/2
      if(m-npl)280,260,260
  280 j=j+m
      go to 370
c     shuffle data by digit reversal for general n
  290 nwork=2*n
      do 360 i1=1,npl,2
      do 360 i3=i1,ntot,np2
      j=i3
      do 350 i=1,nwork,2
      if(icase-3)300,310,300
  300 work(i)=data(j)
      work(i+1)=data(j+1)
      go to 320
  310 work(i)=dats(j)
      work(i+1)=0.
  320 ifp2=np2
      if=ifmin
  330 ifpl=ifp2/ifact(if)
      j=j+ifpl
      if(j-i3-ifp2)350,340,340
  340 j=j-ifp2
      ifp2=ifpl
      if=if+1
      if(ifp2-npl)350,350,330
  350 continue
      i2max=i3+np2-npl
      i=1
      do 360 i2=i3,i2max,npl
      data(i2)=work(i)
      data(i2+1)=work(i+1)
  360 i=i+2
c     main loop for factors of two.  perform fourier transforms of
c     length four, with one of length two if needed.  the twiddle factor
c     w=exp(isign*2*pi*sqrt(-1)*m/(4*mmax)).  check for w=isign*sqrt(-1)
c     and repeat for w=w*(1+isign*sqrt(-1))/sqrt(2).
  370 if(ntwo-npl)680,680,380
  380 npltw=npl+npl
      ipar=ntwo/npl
  390 if(ipar-2)430,410,400
  400 ipar=ipar/4
      go to 390
  410 do 420 i1=1,i1rng,2
      do 420 k1=i1,ntot,npltw
      k2=k1+npl
      tempr=data(k2)
      tempi=data(k2+1)
      data(k2)=data(k1)-tempr
      data(k2+1)=data(k1+1)-tempi
      data(k1)=data(k1)+tempr
  420 data(k1+1)=data(k1+1)+tempi
  430 mmax=npl
  440 if(mmax-ntwo/2)450,680,680
  450 lmax=max0(npltw,mmax/2)
      do 670 l=npl,lmax,npltw
      m=l
      if(mmax-npl)500,500,460
  460 theta=-twopi*float(l)/float(4*mmax)
      if(isign)480,470,470
  470 theta=-theta
  480 wr=cos(theta)
      wi=sin(theta)
  490 w2r=wr*wr-wi*wi
      w2i=2.*wr*wi
      w3r=w2r*wr-w2i*wi
      w3i=w2r*wi+w2i*wr
  500 do 630 i1=1,i1rng,2
      kmin=i1+ipar*m
      if(mmax-npl)510,510,520
  510 kmin=i1
  520 kdif=ipar*mmax
  530 kstep=4*kdif
      if(kstep-ntwo)540,540,630
  540 do 620 k1=kmin,ntot,kstep
      k2=k1+kdif
      k3=k2+kdif
      k4=k3+kdif
      if(mmax-npl)550,550,580
  550 u1r=data(k1)+data(k2)
      u1i=data(k1+1)+data(k2+1)
      u2r=data(k3)+data(k4)
      u2i=data(k3+1)+data(k4+1)
      u3r=data(k1)-data(k2)
      u3i=data(k1+1)-data(k2+1)
      if(isign)560,570,570
  560 u4r=data(k3+1)-data(k4+1)
      u4i=data(k4)-data(k3)
      go to 610
  570 u4r=data(k4+1)-data(k3+1)
      u4i=data(k3)-data(k4)
```

```
      go to 610
  580 t2r=w2r*data(k2)-w2i*data(k2+1)
      t2i=w2r*data(k2+1)+w2i*data(k2)
      t3r=wr*data(k3)-wi*data(k3+1)
      t3i=wr*data(k3+1)+wi*data(k3)
      t4r=w3r*data(k4)-w3i*data(k4+1)
      t4i=w3r*data(k4+1)+w3i*data(k4)
      u1r=data(k1)+t2r
      u1i=data(k1+1)+t2i
      u2r=t3r+t4r
      u2i=t3i+t4i
      u3r=data(k1)-t2r
      u3i=data(k1+1)-t2i
      if(isign)590,600,600
  590 u4r=t3i-t4i
      u4i=t4r-t3r
      go to 610
  600 u4r=t4i-t3i
      u4i=t3r-t4r
  610 data(k1)=u1r+u2r
      data(k1+1)=u1i+u2i
      data(k2)=u3r+u4r
      data(k2+1)=u3i+u4i
      data(k3)=u1r-u2r
      data(k3+1)=u1i-u2i
      data(k4)=u3r-u4r
  620 data(k4+1)=u3i-u4i
      kdif=kstep
      kmin=4*(kmin-i1)+i1
      go to 530
  630 continue
      m=m+lmax
      if(m-mmax)640,640,670
  640 if(isign)650,660,660
  650 tempr=wr
      wr=(wr+wi)*rthlf
      wi=(wi-tempr)*rthlf
      go to 490
  660 tempr=wr
      wr=(wr-wi)*rthlf
      wi=(tempr+wi)*rthlf
      go to 490
  670 continue
      ipar=3-ipar
      mmax=mmax+mmax
      go to 440
c     main loop for factors not equal to two.  apply the twiddle factor
c     w=exp(isign*2*pi*sqrt(-1)*(j1-1)*(j2-j1)/(ifp1+ifp2)), then
c     perform a fourier transform of length ifact(if), making use of
c     conjugate symmetries.
  680 if(ntwo-np2)690,900,900
  690 ifp1=ntwo
      if=inon2
      nplhf=npl/2
  700 ifp2=ifact(if)*ifp1
      j1min=np1+1
      if(j1min-ifp1)710,710,760
  710 do 750 j1=j1min,ifp1,np1
      theta=-twopi*float(j1-1)/float(ifp2)
      if(isign)730,720,720
  720 theta=-theta
  730 wstpr=cos(theta)
      wstpi=sin(theta)
      wr=wstpr
      wi=wstpi
      j2min=j1+ifp1
      j2max=j1+ifp2-ifp1
      do 750 j2=j2min,j2max,ifp1
      i1max=j2+i1rng-2
      do 740 i1=j2,i1max,2
      do 740 j3=i1,ntot,ifp2
      tempr=data(j3)
      data(j3)=data(j3)*wr-data(j3+1)*wi
  740 data(j3+1)=tempr*wi+data(j3+1)*wr
      tempr=wr
      wr=wr*wstpr-wi*wstpi
  750 wi=tempr*wstpi+wi*wstpr
  760 theta=-twopi/float(ifact(if))
      if(isign)780,770,770
  770 theta=-theta
  780 wstpr=cos(theta)
      wstpi=sin(theta)
      j2rng=ifp1*(1+ifsct(if)/2)
      do 890 i1=1,i1rng,2
      do 890 i3=i1,ntot,np2
      j2max=i3+j2rng-ifp1
      do 880 j2=i3,j2max,ifp1
      j1max=j2+ifp1-np1
      do 850 j1=j2,j1max,np1
      j3max=j1+np2-ifp2
      do 850 j3=j1,j3max,ifp2
      jmin=j3-j2+i3
      jmax=jmin+ifp2-ifp1
      i=1+(j3-i3)/nplhf
      if(j2-i3)790,790,820
  790 sumr=0.
      sumi=0.
      do 810 j=jmin,jmax,ifp1
  800 sumr=sumr+data(j)
  810 sumi=sumi+data(j+1)
      work(i)=sumr
      work(i+1)=sumi
      go to 850
  820 iconj=1+(ifp2-2*j2+i3+j3)/nplhf
      j=jmax
      sumr=data(j)
      sumi=data(j+1)
      oldsr=0.
```

```
      oldsi=0.
      j=j-ifpl
  830 tempr=sumr
      tempi=sumi
      sumr=twowr*sumr-oldsr+data(j)
      sumi=twowr*sumi-oldsi+data(j+1)
      oldsr=tempr
      oldsi=tempi
      j=j-ifpl
      if(j-jmin)840,840,830
  840 tempr=wr*sumr-oldsr+data(j)
      tempi=wi*sumi
      work(i)=tempr-tempi
      work(iconj)=tempr+tempi
      tempr=wr*sumi-oldsi+data(j+1)
      tempi=wi*sumr
      work(i+1)=tempr+tempi
      work(iconj+1)=tempr-tempi
  850 continue
      if(j2-i3)860,860,870
  860 wr=wstpr
      wi=wstpi
      go to 880
  870 tempr=wr
      wr=wr*wstpr-wi*wstpi
      wi=tempr*wstpi+wi*wstpr
  880 twowr=wr+wr
      i=1
      i2max=i3+np2-np1
      do 890 i2=i3,i2max,np1
      data(i2)=work(i)
      data(i2+1)=work(i+1)
  890 i=i+2
      if=if+1
      ifpl=ifp2
      if(ifpl-np2)700,900,900
c     complete a real transform in the 1st dimension, n even, by con-
c     jugate symmetries.
  900 go to (1170,1090,1170,910),icase
  910 nhalf=n
      n=n+n
      theta=-twopi/float(n)
      if(isign)930,920,920
  920 theta=-theta
  930 wstpr=cos(theta)
      wstpi=sin(theta)
      wr=wstpr
      wi=wstpi
      imin=3
      jmin=2*nhalf-1
      go to 960
  940 j=jmin
      do 950 i=imin,ntot,np2
      sumr=(data(i)+data(j))/2.
      sumi=(data(i+1)+data(j+1))/2.
      difr=(data(i)-data(j))/2.
      difi=(data(i+1)-data(j+1))/2.
      tempr=wr*sumi+wi*difr
      tempi=wi*sumi-wr*difr
      data(i)=sumr+tempr
      data(i+1)=difi+tempi
      data(j)=sumr-tempr
      data(j+1)=-difi+tempi
  950 j=j+np2
      imin=imin+2
      jmin=jmin-2
      tempr=wr
      wr=wr*wstpr-wi*wstpi
      wi=tempr*wstpi+wi*wstpr
  960 if(imin-jmin)940,970,1000
  970 if(isign)980,1000,1000
  980 do 990 i=imin,ntot,np2
  990 data(i+1)=-data(i+1)
 1000 np2=np2+np2
      ntot=ntot+ntot
      j=ntot+1
      imax=ntot/2+1
 1010 imin=imax-2*nhalf
      i=imin
      go to 1030
 1020 data(j)=data(i)
      data(j+1)=-data(i+1)
 1030 i=i+2
      j=j-2
      if(i-imax)1020,1040,1040
 1040 data(j)=data(imin)-data(imin+1)
      data(j+1)=0.
      if(i-j)1060,1080,1080
 1050 data(j)=data(i)
      data(j+1)=data(i+1)
 1060 i=i-2
      j=j-2
      if(i-imin)1070,1070,1050
 1070 data(j)=data(imin)+data(imin+1)
      data(j+1)=0.
      imax=imin
      go to 1010
 1080 data(1)=data(1)+data(2)
      data(2)=0.
      go to 1170
c     complete a real transform for the 2nd or 3rd dimension by
c     conjugate symmetries.
 1090 if(i1rng-np1)1100,1170,1170
 1100 do 1160 i3=1,ntot,np2
      i2max=i3+np2-np1
      do 1160 i2=i3,i2max,np1
      imin=i2+i1rng
      imax=i2+np1-2
```

```
      jmax=2*i3+np1-imin
      if(i2-i3)1120,1120,1110
 1110 jmax=jmax+np2
 1120 if(idim-2)1150,1150,1130
 1130 j=jmax+np0
      do 1140 i=imin,imax,2
      data(i)=data(j)
      data(i+1)=-data(j+1)
 1140 j=j-2
 1150 j=jmax
      do 1160 i=imin,imax,np0
      data(i)=data(j)
      data(i+1)=-data(j+1)
 1160 j=j-np0
c     end of loop on each dimension
 1170 np0=np1
      np1=np2
 1180 nprev=n
 1190 return
      end
c ***************************************************************************
      subroutine sdummy
c     random access i/o routine to be used with sfftmg subroutine
c***************************************************************************
      integer out
      complex f(n),g(n)
c     entry to receive input setup
      entry inset(iin)
      in=iin
      return
c     entry to receive output setup
      entry outset(iout)
      out=iout
      return
c     entry to do input
      entry randin(n,f,nrec)
      read(in'nrec)f
      return
c     entry to do output
      entry randou(n,g,nrec)
      write(out'nrec)g
      return
      end
```