DEPARTMENT OF THE INTERIOR

U.S. GEOLOGICAL SURVEY

TERRACE: A terracing procedure for gridded data, with Fortran programs, and VAX Command Procedure, Unix C-Shell, and DOS batch file implementations

By

Jeffrey D. Phillips[1]

1992

Open File Report 92-5

92-5A Documentation (Paper Copy)
92-5B Source Code Diskette

[1]U.S. Geological Survey, Denver, Colorado

CONTENTS

Page

Introduction 1

The terracing operator 1

Algorithm 1

Installation 3

Acknowledgments 4

References 4

Appendix 1: The standard grid 5

Appendix 2: Program listings 6

Listing 1: TERRACE.SH C-Shell script 6

Listing 2: TERRACE.COM VAX/VMS Command Procedure . . . 7

Listing 3: TERRACE.BAT DOS Batch File 9

Listing 4: GRADIENT.F 10

Listing 5: STDBNDY.F 11

Listing 6: COMBGR.F 12

Listing 7: TERRACE4.F 13

Listing 8: TRANSPOS.F 15

Listing 9: FIXTER.F 17

Listing 10: MEDIPLUG.F 18

Listing 11: MEDIFILT.F 19

INTRODUCTION

Procedure TERRACE performs terracing of gridded data. A terraced data set consists of constant amplitude domains separated by abrupt domain boundaries. Terracing was initially developed by Cordell and McCafferty (1989) for application to gridded gravity anomaly or magnetic anomaly data (after a pseudo-gravity transformation). The terraced data provided an estimate of the density or magnetization distribution producing the anomalies. Initial terracing algorithms developed for this purpose proved to be unstable with regard to the locations of the domain boundaries. Procedure TERRACE fixes the locations of domain boundaries to correspond to the lines tracking the maxima of the horizontal gradient of the gridded data, thus solving the instability problem.

THE TERRACING OPERATOR

Terracing has proven to be an effective tool for interpretation of gravity and aeromagnetic data, because it converts the smoothly varying anomaly data into something resembling a source map, i.e. domains of constant physical properties separated by abrupt boundaries. The transformation is accomplished by using an operator that iteratively steepens the highest gradients of the starting grid and flattens the areas of lower gradient.

Cordell and McCafferty (1989) describe a terracing operator in which the value of each grid point is compared to the values of the four adjacent grid points to the east, west, north and south. If the center point is lower than the average of the four adjacent points, it is assigned to the minimum of the four adjacent points. If it is greater than the average, it is assigned to the maximum of the four adjacent points. If the center point is outside the range of the adjacent points, it is unchanged. By iteratively applying this operator, a terraced grid can be constructed.

Ideally, the boundaries of the domains in the terraced grid should correspond to the locations of the steepest horizontal gradients in the gravity or pseudo-gravity data. This is because these locations provide the best estimates of vertical edges of source bodies (Cordell and Grauch, 1985; Blakely and Simpson, 1986). In early applications of terracing (Cordell and McCafferty, 1989), the domain boundaries were allowed to move during the iteration. This resulted in displacement of the domain boundaries away from the most likely locations of geologic contacts. Furthermore the movement of the boundaries was erratic, so that the positions of the boundaries could not be predicted from one iteration to the next. Finally, the domains resulting from this procedure had a blocky appearance, with their edges tending to be aligned with the rows and columns of the grid. To correct these problems, a multi-step procedure has been developed to insure that the locations of the domain boundaries correspond to the steepest horizontal gradients of the input grid.

ALGORITHM

The terracing procedure consists of a series of Fortran programs that are accessed through a command language shell. VAX/VMS, Unix C-Shell, and PC-DOS versions of TERRACE are provided. TERRACE assumes that the input data are in standard grid format (Appendix 1). The input grid may contain flagged

areas of no data (DVALs, Appendix 1), but the user should be aware that some of these flagged values will be overwritten by the procedure. It is recommended that a regional surface be removed from the gravity or aeromagnetic grid before terracing in order to increase the dynamic range of the data. Gridded gravity data can be directly processed using TERRACE. Gridded magnetic data should first be transformed to pseudogravity data by applying a filter to the grid. Either of the computer programs FFTFIL (Hildenbrand, 1983; Godson and Mall, 1989; U.S. Geological Survey, 1989) or BOUNDARY (Blakely and Simpson, 1986; Godson and Mall, 1989; U.S. Geological Survey, 1989) can be used to perform the pseudogravity transformation. This transformation is necessary because the magnetic data represent the component of a vector field along a direction that is dependent upon the geomagnetic latitude at the site of the survey. Consequently the amplitudes of the anomalies and the positions of the gradients are latitude dependent. The physical properties that TERRACE is intended to estimate are not latitude dependent, therefore these effects must be removed prior to terracing. Furthermore, magnetic fields contain sidelobes due to the dipolar nature of magnetic sources. These sidelobes will produce fictitious domains in the terraced data. The pseudogravity transformation results in removal of both latitude effects and dipolar sidelobes.

The shell procedure TERRACE asks for the name of the input grid and the number of iterations to be used for terracing. Past experiance has established that 20 to 30 iterations will generally result in a terrace grid with greater than 85% flat slopes. The procedure calls eight different programs, and produces two different output grids. The names of the output grids consist of the same prefix as the input grid, but different suffixes. The secondary output grid has the suffix '.ter'. When viewed as an image, this grid will emphasize the domain boundaries as data gaps. The primary output grid has the suffix '.fil'. This grid has had the domain boundary data gaps filled in by median filtering.

The steps in terracing are as follows:

1. Compute the magnitude of the horizontal gradient of the input grid by using the program GRADIENT. The results are stored in a temporary gradient grid file.

2. Preserve the maxima (ridge crests) of the gradient grid, but set all other grid cells to a 'no data' value (called a DVAL) by using the program STDBNDY. A temporary output grid file is created.

3. Modify the original input grid by changing the grid cells corresponding to the ridge crests of the horizontal gradient (read from the output grid of step 2) to 'no data' (DVAL) values by using the program COMBGR. These lines of DVALs act as barriers to the terracing operator, effectively forcing the major domain boundaries to correspond to the horizontal gradient crests. A temporary output grid file is created.

4. Terrace the output of COMBGR by using the program TERRACE4. A temporary output grid file is created. TERRACE4 will report

the percentage of flat slopes at each iteration. The terracing algorithm makes no modification to DVALS and cells adjacent to DVALS. Therefore these cells need to be handled separately.

5. Transpose the rows and columns of the terraced grid by using the program TRANSPOS. A temporary output grid file is created.

6. Along rows of the transposed grid, replace grid cells adjacent to DVALS (maximum gradient lines) by medians of values two grid cells away from the DVALS by using the program FIXTER. This will insure that grid cells at the edge of a domain have the same value as the rest of the domain. A temporary output grid file is created.

7. Repeat steps 5 and 6 to complete the repair of grid cells adjacent to DVALS. The resulting grid file has the suffix '.ter', and is one of the two output grid files. Because of the two transpositions, the grid is returned to its original orientation. This grid, when viewed as an image, will emphasizes the domain boundaries as black or white data gaps.

8. Replace each DVAL with the median of adjacent non-DVALS by using the program MEDIPLUG. This fills in the gaps at the domain boundries, assigning the boundary cell the value of the dominant adjacent domain. A temporary output grid file is created.

9. Pass a 3x3 median filter over the output of MEDIPLUG by using the program MEDIFILT. This is done to remove single-cell highs and lows resulting from instabilities in the terracing algorithm of TERRACE4, particularly in areas where domain boundaries are not constrained by lines of maximum horizontal gradient. This is the primary output grid of the procedure and has the suffix '.fil'.

INSTALLATION

The Fortran programs on the distribution diskette need to be compiled and installed, along with the command language shell, in an appropriate directory on the Unix, DOS, or VAX computer. The VMS command procedure assumes that the compiled programs are installed in a directory with the system logical name 'geophys'. The Unix shell script and the DOS batch file assume only that the compiled programs are in the user's search path. The command language shells can be edited to reflect alternate directory structures.

ACKNOWLEDGMENTS

Development of this procedure has benefited from the programming skills of many U.S. Geological Survey personnel. The C-Shell script TERRACE.SH was modified form a script written by Joseph Duval, Reston, Virginia. The program GRADIENT was written by Robert Simpson, Menlo Park, California. The program TRANSPOS was written by Mike Webring, Denver, Colorado, with modifications to the DOS version by Dick Godson and Margaret Mall, Denver, Colorado. The program TERRACE4 was written by Lindrith Cordell and A.E. McCafferty, Denver, Colorado.

REFERENCES

Blakely, R.J., and Simpson, R.W., 1986, Locating edges of source bodies for magnetic or gravity anomalies: Geophysics, v. 51, no. 7, p. 1494-1498.

Cordell, Lindrith, and Grauch, V.J.S., 1985, Mapping basement magnetization zones from aeromagnetic data in the San Juan Basin, New Mexico, in Hinze, W.J., ed., The Utility of Regional Gravity and Magnetic Anomaly Maps: Society of Exploration Geophysicists, p. 181-197.

Cordell, Lindrith, and McCafferty, A.E., 1989, A terracing operator for physical property mapping with potential field data: Geophysics, v. 54, no. 5, p. 621-634.

Godson, Richard H., and Mall, Margaret R., 1989, Potential-field geophysical programs for IBM compatible microcomputers, version 1.0: U.S. Geological Survey Open-File Report 89-197, parts A through F, 25 p., 5 diskettes.

Hildenbrand, T.G., 1983, FFTFIL: A filtering program based on two-dimensional Fourier analysis of geophysical data: U.S. Geological Survey Open-file Report 83-237.

U.S. Geological Survey, 1989, Potential-field geophysical programs for VAX 7xx computers: U.S. Geological Survey Open-File Report 89-115, parts A through D, 23 p., 3 diskettes.

APPENDIX 1: THE STANDARD GRID

The binary standard grid file used by these programs consists of: (1) a header record and (2) a series of data records corresponding to rows of the grid. The origin of the grid is in the lower left corner. Rows are read from left to right.

A. Header record (23 4-byte words long)

id:	56 ASCII characters of identification (14 words)
pgm:	8 ASCII characters identifying the creation program (2 words)
ncol:	number of columns of data (integer, 1 word)
nrow:	number of rows of data (integer, 1 word)
nz:	number of words per data element (integer, 1 word)
xo:	x-position of first (leftmost) column of data (real, 1 word)
dx:	equal spacing of columns (real, 1 word)
yo:	y-position of first (bottommost) row of data (real, 1 word)
dy:	equal spacing of rows (real, 1 word)

B. Data record (ncol + 1 words long assuming nz = 1)

dlt:	a dummy value (real, 1 word)
x():	ncol data values corresponding to a row of data (real, ncol words)

Empty grid cells are flagged by a special 'no data' value called a DVAL. The value of DVAL is octal constant '37777677777'o for VAX/VMS systems and decimal constant 1.0e+38 for DOS and Unix systems.

APPENDIX 2: PROGRAM LISTINGS

Listing 1: TERRACE.SH C-Shell script

```
#       @(#)terrace.sh   1.3    command procedure written in Cshell script
#       J.Duval, USGS, Reston, Virginia, and J.Phillips, USGS, Denver,
#       Colorado
#
if ($#argv <= 0) then
        echo "************ @(#)terrace.sh 1.3 ********"
        echo " "
        echo "This procedure terraces an input grid file which is"
        echo "assumed to be in the old USGS standard grid format."
        echo " "
        echo "USAGE: terrace P1 P2"
        echo " "
        echo "P1 is the input grid file name."
        echo "P2 is the number of interations to be used."
        goto done
else
#
#       check on the data file
#
if ( -e terrace$$) then
        rm -f terrace$$
endif
set ofil1 = `expr $1".tmp"`
set ofil2 = `expr $1".ter"`
set ofil3 = `expr $1".fil"`
echo "1" > terrace$$
echo $1 >> terrace$$
echo $ofil1 >> terrace$$
if ( -e $ofil1 ) then
        rm -f $ofil1
endif
echo "Running the gradient program."
gradient < terrace$$ >> /dev/null
rm -f terrace$$
echo $ofil1 > terrace$$
echo $ofil2 >> terrace$$
echo "0" >> terrace$$
if ( -e $ofil2 ) then
        rm -f $ofil2
endif
echo "Running the stdbndy program."
stdbndy < terrace$$ >> /dev/null
rm -f terrace$$
echo $1 > terrace$$
echo $ofil2 >> terrace$$
echo $ofil1 >> terrace$$
if ( -e $ofil1 ) then
        rm -f $ofil1
endif
echo "Running the combgr program."
combgr < terrace$$ >> /dev/null
rm -f terrace$$
echo $ofil1 > terrace$$
echo $ofil2 >> terrace$$
echo $2 >> terrace$$
echo " " >> terrace$$
if ( -e $ofil2 ) then
        rm -f $ofil2
endif
if ( -e terrace.tm1 ) then
        rm -f terrace.tm1
endif
if ( -e terrace.tmp ) then
        rm -f terrace.tmp
endif
echo "Running the terrace4 program."
terrace4 < terrace$$
rm -f terrace$$
if ( -e terrace.tm1 ) then
        rm -f terrace.tm1
endif
```

```
if ( -e terrace.tmp ) then
        rm -f terrace.tmp
endif
echo $ofil2 > terrace$$
echo $ofil1 >> terrace$$
echo "1" >> terrace$$
echo " " >> terrace$$
if ( -e $ofil1 ) then
        rm -f $ofil1
endif
echo "Running the transpos program."
transpos < terrace$$ >> /dev/null
rm -f terrace$$
echo $ofil1 > terrace$$
echo $ofil2 >> terrace$$
if ( -e $ofil2 ) then
        rm -f $ofil2
endif
echo "Running the fixter program."
fixter < terrace$$ >> /dev/null
rm -f terrace$$
echo $ofil2 > terrace$$
echo $ofil1 >> terrace$$
echo "1" >> terrace$$
echo " " >> terrace$$
if ( -e $ofil1 ) then
        rm -f $ofil1
endif
echo "Running the transpos program."
transpos < terrace$$ >> /dev/null
rm -f terrace$$
echo $ofil1 > terrace$$
echo $ofil2 >> terrace$$
if ( -e $ofil2 ) then
        rm -f $ofil2
endif
echo "Running the fixter program."
echo "Creating" $ofil2
fixter < terrace$$ >> /dev/null
rm -f terrace$$
echo $ofil2 > terrace$$
echo $ofil1 >> terrace$$
if ( -e $ofil1 ) then
        rm -f $ofil1
endif
echo "Running the mediplug program."
mediplug < terrace$$ >> /dev/null
rm -f terrace$$
echo $ofil1 > terrace$$
echo $ofil3 >> terrace$$
if ( -e $ofil3 ) then
        rm -f $ofil3
endif
echo "1" >> terrace$$
echo "Running the medifilt program."
echo "Creating" $ofil3
medifilt < terrace$$ >> /dev/null
rm -f terrace$$
endif
rm -f $ofil1
done:
```

Listing 2: TERRACE.COM VAX/VMS Command Procedure

```
!$ terracing command procedure - vax/vms version
!$ J.Phillips, USGS, Denver, Colorado
!$
!$ assumes that compiled programs are in a directory
!$ with the system logical name 'geophys'
!$
$ set noverify
$ if p1.eqs."" then inquire p1 "input file"
$ if p1.eqs."" then exit
$ inquire p2 "iterations"
$ ifil=f$parse(p1)
$ o=f$parse(p1,,,"name")
```

```
!$ write sys$output o
$ ofil1=f$parse(o,".tmp")
$ ofil2=f$parse(o,".ter")
$ ofil3=f$parse(o,".plg")
!$ write sys$output ofil1
$ tmpfil= "temp.dat"
$ open/write f 'tmpfil'
$ tmpfil=f$search(tmpfil)
$ write f 1
$ write f ifil
$ write f ofil1
$ close f
$ assign/user 'tmpfil' sys$input
$ set verify
$ r geophys:gradient
$ set noverify
$ delete 'tmpfil'
$ open/write f 'tmpfil'
$ write f ofil1
$ write f ofil2
$ write f 0
$ close f
$ assign/user 'tmpfil' sys$input
$ set verify
$ r geophys:stdbndy
$ set noverify
$ delete 'tmpfil'
$ open/write f 'tmpfil'
$ write f ifil
$ write f ofil2
$ write f ofil1
$ close f
$ assign/user 'tmpfil' sys$input
$ set verify
$ r geophys:combgr
$ set noverify
$ delete 'tmpfil'
$ open/write f 'tmpfil'
$ write f ofil1
$ write f ofil2
$ write f p2
$ write f " "
$ close f
$ assign/user 'tmpfil' sys$input
$ set verify
$ r geophys:terrace4
$ set noverify
$ delete 'tmpfil'
$ open/write f 'tmpfil'
$ write f ofil2
$ write f ofil1
$ write f 1
$ write f " "
$ close f
$ assign/user 'tmpfil' sys$input
$ set verify
$ r geophys:transpos
$ set noverify
$ delete 'tmpfil'
$ open/write f 'tmpfil'
$ write f ofil1
$ write f ofil2
$ close f
$ assign/user 'tmpfil' sys$input
$ set verify
$ r geophys:fixter
$ set noverify
$ delete 'tmpfil'
$ open/write f 'tmpfil'
$ write f ofil2
$ write f ofil1
$ write f 1
$ write f " "
$ close f
$ assign/user 'tmpfil' sys$input
$ set verify
$ r geophys:transpos
```

```
$ set noverify
$ delete 'tmpfil'
$ open/write f 'tmpfil'
$ write f ofil1
$ write f ofil2
$ close f
$ assign/user 'tmpfil' sys$input
$ set verify
$ r geophys:fixter
$ set noverify
$ delete 'tmpfil'
$ open/write f 'tmpfil'
$ write f ofil2
$ write f ofil1
$ close f
$ assign/user 'tmpfil' sys$input
$ set verify
$ r geophys:mediplug
$ set noverify
$ delete 'tmpfil'
$ open/write f 'tmpfil'
$ write f ofil1
$ write f ofil3
$ write f 1
$ close f
$ assign/user 'tmpfil' sys$input
$ set verify
$ r geophys:medifilt
$ set noverify
$ delete 'tmpfil'
$ ofil1=f$fao("!2(AS)",o,".tmp;*")
$ delete 'ofil1'
$ ofil2=f$fao("!2(AS)",o,".ter")
$ purge 'ofil2'
```

Listing 3: TERRACE.BAT DOS Batch File

```
@echo off
echo .
if exist %1.%2 goto begin
echo This procedure terraces an input grid file that is
echo in USGS standard grid format.
echo .
echo USAGE: terrace P1 P2 P3
echo .
echo P1 is the input file name prefix
echo P2 is the input filename suffix
echo P3 is the number of iterations to be used
echo .
echo EXAMPLE:
echo To terrace the grid file MYDATA.GRD with 20 iterations,
echo type: terrace MYDATA GRD 20
echo .
goto done
:begin
echo . > temp
for %%f in (%1.ter %1.fil) do if exist %%f echo Warning: %%f will be overwritten.
for %%f in (%1.ter %1.fil) do if exist %%f if exist temp del temp
if not exist temp echo .
if not exist temp echo Hit Ctrl-Break to abort, otherwise . . .
if not exist temp pause
echo %1.%2 > temp
echo %1.tmp >> temp
echo gradient of %1.%2 >> temp
echo .
echo ** Running the gradient program **
gradient < temp >> nul
echo %1.tmp > temp
echo %1.ter >> temp
echo 0 >> temp
echo ** Running the stdbndy program **
stdbndy < temp >> nul
echo %1.%2 > temp
echo %1.ter >> temp
echo %1.tmp >> temp
echo ** Running the combgr program **
```

```
combgr < temp >> nul
echo %1.tmp > temp
echo %1.ter >> temp
echo %3 >> temp
echo terrace of %1.%2 with %3 iterations >> temp
echo ** Running the terrace4 program **
echo .
terrace4 < temp
echo %1.ter > temp
echo %1.tmp >> temp
echo 1 >> temp
echo transpose of %1.ter >> temp
echo ** Running the transpos program **
transpos < temp >> nul
echo %1.tmp > temp
echo %1.ter >> temp
echo ** Running the fixter program **
fixter < temp >> nul
echo %1.ter > temp
echo %1.tmp >> temp
echo 1 >> temp
echo transpose of %1.ter >> temp
echo ** Running the transpos program **
transpos < temp >> nul
echo %1.tmp > temp
echo %1.ter >> temp
echo ** Running the fixter program **
fixter < temp >> nul
echo %1.ter > temp
echo %1.tmp >> temp
echo ** Running the mediplug program **
mediplug < temp >> nul
echo %1.tmp > temp
echo %1.fil >> temp
echo 1 >> temp
echo ** Running the medifilt program **
medifilt < temp >> nul
echo ** Deleting temporary files
del %1.tmp
del temp
echo .
echo ** Done:  %1.ter, %1.fil created **
:done
```

Listing 4: GRADIENT.F

```
c     gradient.f - unix version
c     original vms version code indicated by cv
c
c     vms version by R.Simpson, USGS, Menlo Park, California
c
      character infil*50, outfil*50
      character id*56, pgm1*8, pgm2*8
      dimension a(1000), b(1000), c(1000), grad(1000)
cv    data dcval / '37777677777'o /
      data dcval / 1.0e+38 /
      data pgm2 / 'gradient' /
      rad = 3.14159 / 180.
c
      print *, ' *Mode: 1=magnitude of gradient'
      print *, '        2=direction of gradient (0 to 360)'
      print *, '        3=trend of gradient (0 to 180)'
      print *, ' *Give mode'
      read(5, *) mode
    5 format(a50)
c
      print *, ' *Give input grid:'
      read(5, 5) infil
      print *, ' *Give output grid:'
      read(5, 5) outfil
      open(10, file=infil, status='old', form='unformatted')
      open(11, file=outfil, status='new', form='unformatted')
c
      read(10) id, pgm1, ncol, nrow, nz, xo, dx, yo, dy
      print *, id
      print *, pgm1
```

```
      print *, ' ncol=', ncol, '     nrow=', nrow
      print *, ' xo=  ', xo, '     yo=  ', yo
      print *, ' dx=  ', dx, '     dy=  ', dy
      write(11) id, pgm2, ncol, nrow, nz, xo, dx, yo, dy
c
      read(10) dum, (b(i),i = 1, ncol)
      read(10) dum, (c(i),i = 1, ncol)
c
      do 10 i = 1, ncol
   10 grad(i) = dcval
      write(11) dum, (grad(i),i = 1, ncol)
c
      do 200 j = 2, nrow - 1
      do 120 i = 1, ncol
      a(i) = b(i)
  120 b(i) = c(i)
      read(10) dum, (c(i),i = 1, ncol)
      do 40 i = 1, ncol
   40 grad(i) = dcval
      do 100 i = 2, ncol - 1
      if ((b(i - 1) .eq. dcval) .or. (b(i + 1) .eq. dcval)) goto 100
      if ((a(i) .eq. dcval) .or. (c(i) .eq. dcval)) goto 100
      dzdx = (b(i + 1) - b(i - 1)) / (2 * dx)
      dzdy = (c(i) - a(i)) / (2 * dy)
      if (mode .eq. 1) grad(i) = sqrt((dzdx ** 2) + (dzdy ** 2))
      if (mode .eq. 2) grad(i) = atan2(dzdy,dzdx) / rad
      if (mode .eq. 3) grad(i) = amod((atan2(dzdy,dzdx) + 360.) / rad,
     &180.)
  100 continue
c
      write(11) dum, (grad(i),i = 1, ncol)
  200 continue
c
      do 510 i = 1, ncol
  510 grad(i) = dcval
      write(11) dum, (grad(i),i = 1, ncol)
c
      close(10)
      close(11)
      stop
      end
```

Listing 5: STDBNDY.F

```
c     stdbndy.f - unix version
c     J.Phillips, USGS, Denver, Colorado
c     original vms version code indicated by cv
c
      program stdbndy
      dimension xbuf(1000, 3), ybuf(1000), id(14), pgm(2)
      character ifile*50, ofile*50
      data ndim / 1000 /
cv    data dval / '3777767777'o /
      data dval / 1.0e+38 /
    1 write(*, 800)
  800 format(8h ifile: ,$)
      read(5, 801) ifile
  801 format(a50)
      open(10, file=ifile, status='old', form='unformatted', err=1)
cv    open(10,file=ifile,status='old',form='unformatted',iostat=ios)
cv    if(ios.ne.0) then
cv    open(10,file=ifile,status='old',form='unformatted',readonly)
cv    endif
      write(*, 802)
  802 format(8h ofile: ,$)
      read(5, 801) ofile
      open(11, file=ofile, status='unknown', form='unformatted')
      write(*, 804)
  804 format(39h background dvals (0) or negative (1): ,$)
      read(5, *) jcode
      val = -1.0
      if (jcode .eq. 0) val = dval
      read(10) id, pgm, nc, nr, nz, xo, dx, yo, dy
      write(11) id, pgm, nc, nr, nz, xo, dx, yo, dy
      do 42 j = 1, 2
   42 read(10) delta, (xbuf(i,j),i = 1, nc)
```

```
      call thinx(xbuf, ybuf, nc, val)
      write(11) delta, (ybuf(i),i = 1, nc)
      do 50 j = 3, nr
      read(10) delta, (xbuf(i,3),i = 1, nc)
      call thinxy(xbuf, ybuf, nc, ndim, val)
      write(11) delta, (ybuf(i),i = 1, nc)
      do 50 i = 1, nc
      xbuf(i,1) = xbuf(i,2)
   50 xbuf(i,2) = xbuf(i,3)
      call thinx(xbuf(1,2), ybuf, nc, val)
      write(11) delta, (ybuf(i),i = 1, nc)
      close(10)
      close(11)
      stop
      end
      subroutine thinx(xbuf, ybuf, n, val)
      dimension xbuf(n), ybuf(n)
      ist = 0
      xb1 = xbuf(1)
      ybuf(1) = val
      do 40 i = 2, n - 1
      xb2 = xbuf(i)
      ybuf(i) = val
      if (xb1 - xb2) 10, 40, 20
   10 ist = i
      goto 40
   20 if (ist .eq. 0) goto 40
      do 30 j = ist, i - 1
   30 ybuf(j) = xb1
      ist = 0
   40 xb1 = xb2
      ybuf(n) = val
      return
      end
      subroutine thinxy(xbuf, ybuf, n, ndim, val)
      dimension xbuf(ndim, 3), ybuf(ndim)
      ist = 0
      xb1 = xbuf(1,2)
      ybuf(1) = val
      if ((xb1 .ge. xbuf(1,1)) .and. (xb1 .ge. xbuf(1,3))) ybuf(1) = xb1
      do 40 i = 2, n - 1
      xb2 = xbuf(i,2)
      ybuf(i) = val
      if ((xb2 .ge. xbuf(i,1)) .and. (xb2 .ge. xbuf(i,3))) ybuf(i) = xb2
      if (xb1 - xb2) 10, 40, 20
   10 ist = i
      goto 40
   20 if (ist .eq. 0) goto 40
      do 30 j = ist, i - 1
   30 ybuf(j) = xb1
      ist = 0
   40 xb1 = xb2
      ybuf(n) = val
      xb1 = xbuf(n,2)
      if ((xb1 .ge. xbuf(n,1)) .and. (xb1 .ge. xbuf(n,3))) ybuf(n) = xb1
      return
      end
```

Listing 6: COMBGR.F

```
c     combgr.f - unix version
c     J.Phillips, USGS, Denver, Colorado
c     original vms version code indicated by cv
c
      dimension gr(1024), cr(1024), id(14), pgm(2)
      character ifile1*50, ifile2*50, ofile*50
cv    data dval / '37777677777'o /
      data dval / 1.0e+38 /
      data dlim / 1.0e+30 /
  799 format(a50)
    1 write(*, 800)
  800 format(15h primary file: ,$)
      read(5, 799) ifile1
cv    open(10,file=ifile1,status='old',form='unformatted',iostat=ios)
cv    if(ios.ne.0) then
cv    open(10,file=ifile1,status='old',form='unformatted',readonly)
```

```
cv    endif
      open(10, file=ifile1, status='old', form='unformatted', err=1)
    2 write(*, 801)
  801 format(13h crest file: ,$)
      read(5, 799) ifile2
cv    open(11,file=ifile2,status='old',form='unformatted',iostat=ios)
cv    if(ios.ne.0) then
cv    open(11,file=ifile2,status='old',form='unformatted',readonly)
cv    endif
      open(11, file=ifile2, status='old', form='unformatted', err=2)
      write(*, 802)
  802 format(14h output file: ,$)
      read(5, 799) ofile
cv    open(12,file=ofile,status='new',form='unformatted')
      open(12, file=ofile, status='unknown', form='unformatted')
      read(10) id, pgm, nc, nr, nz, xo, dx, yo, dy
      read(11) id
      write(12) id, pgm, nc, nr, nz, xo, dx, yo, dy
      do 80 k = 1, nr
      read(10) dlt, (gr(i),i = 1, nc)
      read(11) dlt, (cr(i),i = 1, nc)
      do 70 i = 1, nc
cv    if (cr(i) .eq. dval) goto 70
      if (cr(i) .ge. dlim) goto 70
      gr(i) = dval
   70 continue
      write(12) dlt, (gr(i),i = 1, nc)
   80 continue
      close(10)
      close(11)
      close(12)
      stop
      end
```

Listing 7: TERRACE4.F

```
c  terrace4.f - unix version
c  original vms version code indicated by cv
c  vms version by L.Cordell and A.McCafferty, USGS, Denver, Colorado
c
c   To steepen gradients on basis of local curvature.
c
      character file1*50, file2*50
      character pgm*8
      dimension a1(1000), a2(1000), a3(1000), id(14)
      real it, ib, il, ir, ic2, ic3
      data ddval / 1.e30 /
  111 write(6, 1)
    1 format(1x,22hEnter input file name:/1x1h*$)
      read(5, 101) file1
  101 format(a50)
cv    open(10,file=file1,form='unformatted',status='old',iostat=ios)
cv    if(ios.ne.0) then
cv    open(10,file=file1,form='unformatted',status='old',readonly)
cv    endif
      open(10, file=file1, form='unformatted', status='old', err=111)
      write(6, 2)
    2 format(1x,23hEnter output file name:/1x1h*$)
      read(5, 101) file2
cv    open (11,file='terrace.tmp',status='new',form='unformatted')
cv    open (12,file=file2,form='unformatted',status='new')
      open(11, file='terrace.tmp', status='unknown',
     &form='unformatted')
      open(12, file=file2, form='unformatted', status='unknown')
      write(6, 7)
    7 format(1x,27hEnter number of iterations;/1x1h*$)
      read(5, *) iterations
      write(6, 4)
    4 format(1x,21hEnter output-file id:/1x1h*$)
      read(5, 5) (id(i),i = 1, 14)
    5 format(14a4)
      read(10) id, pgm, ncol, nrow, nz, x0, dx, y0, dy
      pgm = 'terrace '
      if ((nz .ne. 1) .or. (ncol .gt. 1000)) goto 99
      write(12) id, pgm, ncol, nrow, nz, x0, dx, y0, dy
      do 30 iter = 1, iterations
```

```
      icount = 0
      call rowio(a1, y, nz, ncol, 10, 1)
      call rowio(a2, y, nz, ncol, 10, 1)
      call rowio(a1, y, nz, ncol, 11, 2)
      do 10 j = 3, nrow
      call rowio(a3, y, nz, ncol, 10, 1)
      do 21 i = 2, ncol - 1
      if (a2(i) .ge. ddval) goto 13
      ic2 = 0.
      nc2 = 0
      if (a1(i) .lt. ddval) then
        it = a1(i)
        ic2 = ic2 + it
        nc2 = nc2 + 1
      else
        it = a2(i)
      end if
      if (a3(i) .lt. ddval) then
        ib = a3(i)
        ic2 = ic2 + ib
        nc2 = nc2 + 1
      else
        ib = a2(i)
      end if
      if (a2(i - 1) .lt. ddval) then
        il = a2(i - 1)
        ic2 = ic2 + il
        nc2 = nc2 + 1
      else
        il = a2(i)
      end if
      if (a2(i + 1) .lt. ddval) then
        ir = a2(i + 1)
        ic2 = ic2 + ir
        nc2 = nc2 + 1
      else
        ir = a2(i)
      end if
      ic3 = ic2 - (nc2 * a2(i))
      if (ic3) 12, 13, 14
   12 a1(i) = max(it,ib,il,ir,a2(i))
      goto 20
   13 a1(i) = a2(i)
      goto 20
   14 a1(i) = min(it,ib,il,ir,a2(i))
   20 if (a1(i) .eq. a2(i)) icount = icount + 1
   21 continue
      a1(1) = a2(1)
      a1(ncol) = a2(ncol)
      call rowio(a1, y, nz, ncol, 11, 2)
      do 11 i = 1, ncol
      a1(i) = a2(i)
   11 a2(i) = a3(i)
   10 continue
      call rowio(a2, y, nz, ncol, 11, 2)
      count = (icount * 100.) / float(ncol * nrow)
      write(6, 31) iter, count
   31 format(1x,11hIteration =,i5,27h   Percent of flat slopes =,f9.5)
      close(10)
      close(11)
cv    open (10,file='terrace.tmp',status='old',disp='delete',
cv   1  form='unformatted')
      open(10, file='terrace.tmp', status='old', form='unformatted'
     &)
      if (iter .eq. iterations) goto 30
cv    open (11,file='terrace.tmp',status='new',form='unformatted')
      open(11, file='terrace.tmp', status='unknown',
     &form='unformatted')
   30 continue
   33 continue
      do 32 j = 1, nrow
      call rowio(a1, y, nz, ncol, 10, 1)
   32 call rowio(a1, y, nz, ncol, 12, 2)
      goto 999
   99 write(6, 98)
   98 format(1x,16hCan't handle it.)
  999 continue
```

```
      stop
      end
      subroutine rowio(a, y, nz, ncol, ld, key)
      dimension a(nz, ncol)
      goto (1, 2), key
    1 read(ld) y, a
      goto 90
    2 write(ld) y, a
   90 return
      end
```

Listing 8: TRANSPOS.F

```
c  transpos.f - unix version
c  original vms version code indicated by cv
c  vms version by M.Webring, USGS, Denver, Colorado
c
c  flips grid about the diagonal, rotates 90 deg., or reverses columns
c
      dimension buf(50000)
      character id*56, p*8, p2*8
      character if*72, jf*72, kf*72
      data p2 / 'trnsposd' /
      data nz / 1 /
      data kf / 'mtx.tmp' /
      data inp / 5 /
c
      write(*, 1)
    1 format(8h ifile: $)
      read(inp, 100) if
      write(*, 2)
  100 format(a72)
    2 format(8h ofile: $)
      read(inp, 100) jf
      write(*, 3)
    3 format(22h enter 1 for transpose,/,28h       2 for 90 deg rotation
     &,/,28h       3 for column reverse:$)
      irot = 0
      read(inp, *) irot
      if (irot .lt. 3) goto 6
      open(12, file=if, form='unformatted', status='old')
cv    open(10, file=jf, form='unformatted', status='new')
      open(10, file=jf, form='unformatted', status='unknown')
      goto 40
    6 open(10, file=if, form='unformatted', status='old')
cv    open(12,file=jf,form='unformatted',status='new')
      open(12, file=jf, form='unformatted', status='unknown')
      read(10) id, p, nc, nr, nz, xo, dx, yo, dy
      if (nc .gt. 2500) stop 'x too big'
      if (nr .gt. 2500) stop 'y too big'
      n = int((float(nr) / 20.) + .999)
      m = int((float(nc) / 20.) + .999)
      n10 = n * 20
      m10 = m * 20
      nm = n10 * m10
      open(11, file=kf, access='direct', status='scratch', recl=
     &1600)
      call rio(nc, buf, m, n, 1, 1, m10, -1)
      write(*, 4)
    4 format(7h title:)
      read(inp, 5) id
    5 format(14a4)
      write(12) id, p2, nr, nc, nz, yo, dy, xo, dx
      ic = 1
      do 10 ibuf = 1, m
      loc = ibuf
      call rio(nr, buf, n, 0, loc, m, n10, 1)
      ndx = 1
      do 20 i = 1, 20
      call rowio(nr, buf(ndx), 0, 12, 12, iend)
      if (ic .eq. nc) goto 99
      ic = ic + 1
   20 ndx = ndx + n10
   10 continue
   99 continue
      close(11)
```

```
      rewind(12)
      close(10)
c
   40 if (irot .le. 1) goto 90
cv    open(10, file=jf, form='unformatted', status='new')
      open(10, file=jf, form='unformatted', status='unknown')
      read(12) id, p, nc, nr, nz, xo, dx, yo, dy
      p = '90d rota'
      xo = ((nc - 1) * dx) + xo
      dx = - dx
      if (irot .lt. 3) goto 41
      p = 'rev col'
   41 write(10) id, p, nc, nr, nz, xo, dx, yo, dy
      do 45 j = 1, nr
      i2 = nc + nc
      call rowio(nc, buf(1), -1, 12, 12, ie)
      do 42 i = 1, nc
      buf(i2) = buf(i)
   42 i2 = i2 - 1
      call rowio(nc, buf(nc + 1), 0, 10, 10, ie)
   45 continue
      close(10)
   90 close(12)
      stop
      end
      subroutine rio(na, z, nblk, nbuf, loc2, idl, n10, iop)
      dimension w(400)
      dimension z(1), is(2)
cv    data dval/'37777677777'o/
      data dval / 1.0e+38 /
      loc = loc2
      assign 999 to ir
      if (iop) 1, 2, 3
    1 iflag = 1
      assign 99 to ir
      is(1) = na + 1
      is(2) = 1
      assign 10 to ig
      if (n10 .eq. na) goto 15
      assign 12 to ig
   15 do 100 ibuf = 1, nbuf
      ii = 1
      do 10 irow = 1, 20
      goto (11, 12), iflag
   11 call rowio(na, z(ii), -1, 10, 10, ie)
      if (ie .eq. 1) goto 14
      ii = ii + n10
      goto ig
   14 iflag = 2
   12 kk = (irow - 1) * n10
      do 13 k = kk + is(iflag), kk + n10
   13 z(k) = dval
   10 continue
      goto 2
   99 continue
  100 continue
      loc1 = loc - 1
c
      return
    2 do 30 iblk = 1, nblk
      lw = 1
      ls = ((iblk - 1) * 20) + 1
      do 21 j = 1, 20
      l2 = ls
      l = lw
      do 20 i = 1, 20
      w(l) = z(l2)
      l = l + 20
   20 l2 = l2 + 1
      lw = lw + 1
   21 ls = ls + n10
      write(11, rec=loc) w
   30 loc = loc + idl
c
      goto ir
    3 do 40 iblk = 1, nblk
      read(11, rec=loc) w
```

```
      loc = loc + idl
      l = 1
      ls = ((iblk - 1) * 20) + 1
      do 51 j = 1, 20
      l2 = ls
      do 50 i = 1, 20
      z(l2) = w(l)
      l = l + 1
   50 l2 = l2 + 1
   51 ls = ls + n10
   40 continue
  999 return
      end
      subroutine rowio(n, z, iop, idev, jdev, iend)
c  WHERE IOP<0 READ; IOP=0 WRITE; IOP>0 READ&WRITE
      dimension z(n)
      iend = 0
      if (iop) 1, 2, 1
    1 read(idev, end=10) xo, z
      if (iop) 9, 9, 2
    2 write(jdev) xo, z
    9 return
   10 iend = 1
      return
      end
```

Listing 9: FIXTER.F

```
c  fixter.f - unix version
c  J.Phillips, USGS, Denver, Colorado
c  original vms version code indicated by cv
c
c   To convert unchanged grid cells to dvals.
c
      character file1*50, file2*50
      character pgm*8
      dimension a1(2000), id(14)
      data ddval / 1.e30 /
    1 write(6, 2)
    2 format(1x,32hEnter terrace4 output file name:/1x1h*$)
      read(5, 101) file1
  101 format(a50)
cv    open(11,file=file1,form='unformatted',status='old',iostat=ios)
cv    if(ios.ne.0) then
cv    open(11,file=file1,form='unformatted',status='old',readonly)
cv    endif
      open(11, file=file1, form='unformatted', status='old', err=1)
      write(6, 3)
    3 format(1x,23hEnter output file name:/1x1h*$)
      read(5, 101) file2
cv    open (12,file=file2,form='unformatted',status='new')
      open(12, file=file2, form='unformatted', status='unknown')
      read(11) id, pgm, ncol, nrow, nz, x0, dx, y0, dy
      pgm = 'terrace '
      if ((nz .ne. 1) .or. (ncol .gt. 2000)) goto 99
      write(12) id, pgm, ncol, nrow, nz, x0, dx, y0, dy
      do 10 j = 1, nrow
      call rowio(a1, y, nz, ncol, 11, 1)
      do 21 i = 3, ncol - 2
      if (a1(i) .ge. ddval) then
      if ((a1(i + 1) .lt. ddval) .and. (a1(i + 2) .lt. ddval)) a1(i + 1)
     & = a1(i + 2)
      if ((a1(i - 1) .lt. ddval) .and. (a1(i - 2) .lt. ddval)) a1(i - 1)
     & = a1(i - 2)
      end if
   21 continue
      call rowio(a1, y, nz, ncol, 12, 2)
   10 continue
   99 continue
      stop
      end
      subroutine rowio(a, y, nz, ncol, ld, key)
      dimension a(nz, ncol)
      goto (1, 2), key
    1 read(ld) y, a
      goto 90
    2 write(ld) y, a
```

```
   90 return
      end
```

Listing 10: MEDIPLUG.F

```
c       mediplug.f - unix version
c      J.Phillips, USGS, Denver, Colorado
c       original vms version code indicated by cv
c
c       fills holes using the median of surrounding values
c
        dimension a(1000),b(1000,3),c(1000),id(14),pgm(2)
        character*50 ifile
        data ddval/1.0e+36/
        print 200
200     format(' enter ifile: ',$)
        read(5,100) ifile
100     format(a50)
        open(10,file=ifile,status='old',form='unformatted')
        print 201
201     format(' enter ofile: ',$)
        read(5,100) ifile
        open(11,file=ifile,status='new',form='unformatted')
        read(10) id,pgm,nc,nr,nz,xo,dx,yo,dy
        write(11) id,pgm,nc,nr,nz,xo,dx,yo,dy
        read(10) dlt,(b(i,3),i=1,nc)
        write(11) dlt,(b(i,3),i=1,nc)
        read(10) dlt,(b(i,2),i=1,nc)
        nd=0
        do ii=3,nr
                read(10) dlt,(b(i,1),i=1,nc)
                c(1)=b(1,2)
                c(nc)=b(nc,2)
                do i=2,nc-1
                        c(i)=b(i,2)
                        if(c(i).gt.ddval) then
                                j=0
                                do k=1,3
                                        do l=i-1,i+1
                                                if(b(l,k).lt.ddval) then
                                                        j=j+1
                                                        a(j)=b(l,k)
                                                endif
                                        enddo
                                enddo
                                if(j.gt.0) call median(j,a)
                                if(j.lt.1) then
                                        nd=nd+1
                                else
                                        c(i)=a(1)
                                endif
                        endif
                enddo
                write(11) dlt,(c(i),i=1,nc)
                do i=1,nc
                        b(i,3)=b(i,2)
                        b(i,2)=b(i,1)
                enddo
        enddo
        write(11) dlt,(b(i,1),i=1,nc)
        write(*,101) nd
101     format(i10,' dvals remain')
        stop
        end
c
        subroutine median(n,a)
        dimension a(n)
10      if(n.le.2) return
        ax=a(1)
        an=a(1)
        iax=1
        ian=1
        do i=2,n
                if(a(i).gt.ax) then
                        ax=a(i)
                        iax=i
```

```
                endif
                if(a(i).lt.an) then
                        an=a(i)
                        ian=i
                endif
        enddo
        if(an.eq.ax) then
                a(1)=an
                return
        endif
        j=0
        do 20 i=1,n
                temp=a(i)
                if(i.eq.iax.or.i.eq.ian) go to 20
                j=j+1
                a(j)=temp
20      continue
        n=j
        go to 10
        end
```

Listing 11: MEDIFILT.F

```
c     medifilt.f - unix version
c     J.Phillips, USGS, Denver, Colorado
c     original vms version code indicated by cv
c
c     median filtering using a 3x3, 5x5, 7x7, 9x9, or 11x11 window
c
      dimension a(1000), b(1000, 11), c(1000), id(14), pgm(2)
      character ifile*50
      data ddval / 1.0e+36 /
    1 write(*, 200)
  200 format(14h enter ifile: ,$)
      read(5, 100) ifile
  100 format(a50)
cv    open(10,file=ifile,status='old',form='unformatted')
      open(10, file=ifile, status='old', form='unformatted', err=1)
      write(*, 201)
  201 format(14h enter ofile: ,$)
      read(5, 100) ifile
cv    open(11,file=ifile,status='new',form='unformatted')
      open(11, file=ifile, status='unknown', form='unformatted')
      read(10) id, pgm, nc, nr, nz, xo, dx, yo, dy
      write(11) id, pgm, nc, nr, nz, xo, dx, yo, dy
  203 write(*, 202)
  202 format(43h enter window index (1=3x3, 2=5x5, 3=7x7): ,$)
      read(5, *) iw
      if ((iw .lt. 1) .or. (iw .gt. 5)) goto 203
      do j = (2 * iw) + 1, iw + 2, -1
        read(10) dlt, (b(i,j),i = 1, nc)
        write(11) dlt, (b(i,j),i = 1, nc)
      end do
      do j = iw + 1, 2, -1
        read(10) dlt, (b(i,j),i = 1, nc)
      end do
      nd = 0
      do ii = (2 * iw) + 1, nr
        read(10) dlt, (b(i,1),i = 1, nc)
        do i = 1, nc
          c(i) = b(i,iw + 1)
        end do
        do i = iw + 1, nc - iw
          j = 0
          do k = 1, (2 * iw) + 1
            do l = i - iw, i + iw
              if (b(l,k) .lt. ddval) then
                j = j + 1
                a(j) = b(l,k)
              end if
            end do
          end do
          if (j .gt. 0) call median(j, a)
          if (j .lt. 1) then
            nd = nd + 1
          else
```

```
          c(i) = a(1)
        end if
      end do
      write(11) dlt, (c(i),i = 1, nc)
      do i = 1, nc
        do j = (2 * iw) + 1, 2, -1
          b(i,j) = b(i,j - 1)
        end do
      end do
    end do
    do j = iw, 1, -1
      write(11) dlt, (b(i,j),i = 1, nc)
    end do
    write(*, 101) nd
101 format(i10,13h dvals remain)
    stop
    end

    subroutine median(n, a)
    dimension a(n)
 10 if (n .le. 2) return
    ax = a(1)
    an = a(1)
    iax = 1
    ian = 1
    do i = 2, n
      if (a(i) .gt. ax) then
        ax = a(i)
        iax = i
      end if
      if (a(i) .lt. an) then
        an = a(i)
        ian = i
      end if
    end do
    if (an .eq. ax) then
      a(1) = an
      return
    end if
    j = 0
    do 20 i = 1, n
      temp = a(i)
      if ((i .eq. iax) .or. (i .eq. ian)) goto 20
      j = j + 1
      a(j) = temp
 20 continue
    n = j
    goto 10
    end
```