UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

MINC: A gridding program based on minimum curvature

by

Michael Webring

Open File Report 81-1224

1981

Table of Contents

Preface

The program in Appendix B is written mostly in ANSI standard FORTRAN, the exceptions are generally confined to the driver program. Character variables are used in conjunction with file attachments which follow the Honeywell/Multics conventions. The no-data flag is defined as the maximum floating point number, on the Multics octal 376777777777 or approximately 1.7e38. This number may be redefined wherever it appears. Some real number data transfers are accomplished using integer variables, therefore real and integer word lengths must be declared equal. The program declares 41K of main memory for arrays, an experienced programmer should be able to modify this to fit a specific machine.

Abstract

A FORTRAN program is presented which performs grid interpolation from randomly located data using the principle of minimum curvature. The smooth and continuous output grid is suitable for use with functions possessing similar qualities. Gravity and magnetic data are two examples. The program has the capability to handle unlimited input and produce up to 10^6 gridded values. A typical execution time is 5 ms per 20 iterations per grid point, on a machine which executes a floating-point multiply in 4.3 μs.

Introduction

This program uses the biharmonic difference equation to generate a smooth surface that has the property of minimum total curvature, where curvature refers to the estimate of the second horizontal derivatives in x and y at each grid location. The specific algorithm was developed by Briggs (1974), who presented a method for including randomly placed data values as boundary conditions for the difference equations. These conditions cause the gridded surface to converge to the data while maintaining the minimum curvature property in sparse data areas. The algorithm does not return a unique solution but it does produce a grid with a high degree of internal consistency.

Advantages to this algorithm include the ability to fit a surface to large amounts of data without potentially unstable matrix techniques, and the comparable computation times with other gridding techniques. The primary disadvantage stems from the iterative method of solution. This requires that the grid be initialized by an independent technique which should not introduce false trends and then the difference equations are applied using a finite number of iterations.

Theory

It can be shown that the second derivative for a parabola through three equally spaced points is

$$\frac{\partial^2 U_i}{\partial x^2} = U_{i-1} + U_{i+1} - 2U_i = C_i \tag{1}$$

where U_i are incremental samples of the dependent variable. A summation in x and y provides a simple approximation for curvature in two dimensions

$$C_{ij} = \frac{\partial^2 U_{ij}}{\partial x^2} + \frac{\partial^2 U_{ij}}{\partial y^2} = U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - 4U_{ij} \tag{2}$$

minimizing this function over the extent of the grid leads to

$$4C_{ij} = C_{i+1,j} + C_{i-1,j} + C_{i,j+1} + C_{i,j-1} \tag{3}$$

which can be solved for U_{ij}

$$\begin{aligned} - \frac{1}{20} [U_1 + U_5 + U_8 + U_{12} + 2\,(U_2 + U_4 + U_9 + U_{11}) \\ - 8\,(U_3 + U_6 + U_7 + U_{10})] = U_{ij} \end{aligned} \tag{4}$$

where the elements of the difference equation are grid locations

$$\begin{array}{ccccc} & & U_1 & & \\ & U_2 & U_3 & U_4 & \\ U_5 & U_6 & U_{ij} & U_7 & U_8 \\ & U_9 & U_{10} & U_{11} & \\ & & U_{12} & & \end{array}$$

Equation 4 is the biharmonic operator and is the general difference equation used in the program listed in Appendix B. Boundary conditions of (4) are zero curvature along grid edges and data located inside the grid. Briggs presents an expression for use when data is nearby as the curvature

$$C_{ij} = \sum_{k=1}^{4} b_k U_k - U_{ij} \sum_{k=1}^{5} b_k + b_5 W_n \qquad (5)$$

where W_n is the data value and b_k are weights applied to $\{U_k$: $U_{i+1,j-1}$, $U_{i,j-1}$, $U_{i-1,j}$, $U_{i-1,j+1}\}$. Using (5) in (3) yields an expression which is applied where data is available. There are two limitations on (5). The matrix yielding b_k becomes singular as the data value moves into coincidence with grid location U_{ij}, and the data must be close enough that the approximation leading to (5) is valid. In the program the grid location takes on the data value whenever the separation is less than .05 of the grid spacing and beyond .75 grid spacing data are ignored. These two distances are a matter of choice, .75 completely covers a grid cell with enough overlap that a centrally located data value will affect all four corners equally, and when less than .05 the data value weight is much greater than the other four.

A common practice for solving difference equations is to start with a coarse grid and divide the interval by two after completing the iterations until the final interval is reached. That idea is employed here with certain modifications. Equation (5) makes no provision for multiple data points within .75 grid units, therefore aliasing occurs whenever the gridding interval is larger than the wavelengths present in the data and false regional trends occur which cannot be removed by subsequent interval divisions. This program uses a regional grid interpolated at four times the interval of the final grid and where data is sparse these regional values are inserted into the final grid before initialization is complete. The effect is to coach the final grid in the direction of the regional grid. Aliasing in the regional grid is minimized by combining multiple data points with a distance weighting function to produce one psuedo-data value to be used in (5). The weighting operations can also be applied to the final grid.

Program Usage

Program input is a data file containing x,y,z coordinate sets. The number of input coordinates is limited only by the amount of disk storage available. There are three types of data records accepted which are discussed in the parameter list that follows. Input coordinates are assumed to be right cartesian, the data units in x,y,z are unspecified, however x and y are assumed to be the same.

Control parameters are contained in a separate disk file created by the user prior to program execution, so that normally the only user response is the name of the control file.

The output is a grid of equi-spaced interpolated values written to disk in binary form. The file consists of a header record containing size and location information followed by row records that start at the minimum y coordinate specified, details are in Appendix A.

The one critical parameter is gridding interval. A coarse interval results in aliasing and loss of information, too fine an interval results in isolated anomalies and an obscured regional picture. Convergence speed is also affected by relative distance between grid and data values. The optimum interpolation interval is generally 1/2 to 1/5 the data spacing, but since data density can vary greatly the final choice is one dictated by the features the user wants to emphasize. The worst case is aeromagnetic data when it contains short wavelengths along profiles but only longer ones in the cross profile direction. One method of handling this data is to filter the individual profiles to exclude wavelengths shorter than those represented by profile separation. This program cannot directly utilize 1-dimensional filters but has a data averaging option which can be used in cases where a coarse grid would alias high frequencies. The weighting option should not be used without critical evaluation of the resulting grid.

The program breaks the grid into blocks that are iterated in turn. The first program response tells how many tiers broken into sections are being used internally. When a block is completed, a message listing convergence information is printed.

The command file consists of a Fortran namelist, title 'parms', some of the parameters of which have default values and may be left out.

[grid parameters]

xo	x coordinate of the lower-left corner of the grid
yo	y coordinate of the lower-left corner of the grid
del	x and y gridding interval - must be positive
	xo, yo, and del are specified in x,y data units
nc	number of columns (samples in the x direction)
nr	number of rows (samples in the y direction)
idirx	(default = 0) when set ≠ 0 causes x = -x. This allows positive west longitude to be gridded without conversion to negative west longitude

[file control parameters]

ifile	name of the xyz input file, $\leq$ 50 characters long. (default is blank, and the program will prompt the user at run time for ifile, ofile, ifmt, ianom, and id)
ofile	name of the output grid file. $\leq$ 50 characters
ifmt	fortran format of the input file specifying floating-point data fields for x,y, and z. When blank (default), the input is assumed to be unformatted binary records three words long unless modified by parameter ianom. format example: ifmt = "(3F10.3)" or ifmt = "(3E15.4)"
ianom	anomaly selection from an unformatted record of the form: station-id (8 characters), x,y and up to 10 z values which can be gridded independently. Note that setting both ifmt and ianom is inconsistent. (default = 0, xyz records)
id	grid title consisting of a character string $\leq$ 56 in length (default is blank)

[optional control parameters]

radius	a distance, in x,y units which controls where no-data flags are inserted into the grid (default = 0, completely defined grid). Radius does not affect grid generation
npmin	number of data values within radius distance necessary for a grid location to remain unflagged (default = 1)
nim	maximum number of iterations per block (default = 20)
epsm	in z data units, stops iteration when maximum change per iteration drops below this value (default = 0, nim iterations per block)

lapovr — blocks overlap by this number of columns and rows to ensure continuity. This parameter should not be changed without study of the input data density. (default = 10)

slope — a distance weighting parameter that combines all data within .75 'del' to partially compensate for aliasing caused by a coarse gridding interval. The function is of the form

$$w = \frac{1}{r^2 + 1/slope}$$

where r is the distance from grid location to data location. A slope of 5 causes about a 5:1 weighting in favor of data near grid locations. (default = 0, only closest data point is used)

region — set $\neq 0$ to save the coarse grid named regional.tmp which is used internally to aid sparse data areas. (default = 0, automatic deletion)

whole — set $\neq 0$ to save the final grid under the name whole.tmp before the radius parameter is applied. This option is supplied for use with subsequent processing steps (filtering, continuation, etc) which require completely defined grids. When processing is complete a simple external program can mask the finished product with no-data areas of the radiused version. (default = 0)

EXAMPLES

In this sample run the data file consists of 819 records and the control file contains the namelist 'parms' which includes the &'s. The program is started on the USGS Honeywell/Multics by typing minc. The underlines indicate user input.

```
   data file  (grv.dat)
      x          y          z

    43.997     58.517   -212.949
    37.684     57.793   -232.034
    38.954     64.595   -230.364
                 .
                 .
                 .
```

```
   control file  (minc.cmd)

&parms
id=" bouguer gravity, 2.60 gm/cc"
ifile="grv.dat",ofile="grv.grd"
xo=-81.,yo=0.,del=1.,nc=162,nr=116
ifmt="(3f10.3)",radius=2.
&
```

```
   program execution

minc
 enter command filename :minc.cmd
nsec=  2,  ntier=  3,  block size: nc= 88,  nr= 47
     819 data points in area

  tier  1

initial error= 1.87E+00 end error= 6.04E-02 iterations= 20

initial error= 2.46E+00 end error= 1.03E-01 iterations= 20

  tier  2

initial error= 2.24E+00 end error= 6.17E-02 iterations= 20

initial error= 3.49E+00 end error= 1.28E-01 iterations= 20

  tier  3

initial error= 1.67E+00 end error= 5.82E-02 iterations= 20

initial error= 1.93E+00 end error= 1.10E-01 iterations= 20

STOP
cpu sec : 82.764
```

This example contains an interactive option that allows multiple grids to be generated without editing the command file. Whenever 'ifile' is blank in the command file, minc will prompt the user for the information shown below. The example command file contains the minimum number of specified parameters.

```
&parms
xo=-81,yo=0,del=1,nc=100,nr=50
&
```

```
minc
 enter command filename :minc.cmd
 enter ifile ofile:
tmp1.tc.bxyz grav.grd
 enter input format:
                   [blank format]
 z anomaly number :3
 enter title:
bouguer gravity, 2.60 gm/cc
 nsec=  2,  ntier=  1,  block size: nc= 57,   nr= 54
       219 data points in area

   tier  1

initial error= 1.42E+00 end error= 6.04E-02 iterations= 20

initial error= 2.12E+00 end error= 7.80E-02 iterations= 20

STOP
cpu sec : 25.106
```

note: The z anomaly number is nonzero indicating a multiple z data file.
Enter a zero for xyz data files.

Reference

Briggs, I. C., 1974, Machine contouring using minimum curvature: Geophysics, v. 39, no. 1, p. 39-48.

Appendix A

The output grid is written in unformatted binary records, and serves as a link between the programs used in the USGS geophysics branches. The following description is a subset of the more general specification.

header record

id	56 character title of the grid
pgm	8 character program identifier
nc	number of columns in a row record
nr	number of row records
nz	number of data values associated with each grid mesh location (always equals 1)
xo	x coordinate of lower left corner of grid
dx	x increment, always +del or -del
yo	y coordinate of lower left corner
dy	y increment, always equals +del

row records

There are 'nr' row records each of which is nc+1 words in length.

ycoord	y coordinate (1 word) of the row is always equal to zero. y position is defined from yo and dy
z	an array 'nc' words long which is one row of the output grid

```
c
c                                   Appendix B
c
c
c                    minimum curvature gridding routine
c*****************************************************************************
c  this program generates a 2-dimensional grid, equally
c  incremented in x and y, from randomly placed data points.
c  the algorithm (Briggs) produces a smooth grid by iteratively
c  solving a set of difference equations which minimize the total
c  2nd horizontal derivative and attempt to honor input data.
c  (ref: I.C. Briggs, 1974, Geophysics, v 39, no 1)
c
c  namelist parameters:
c  id      56 character title of output grid
c  ifile   input file containing xyz data records
c  ofile   output grid, consisting of a header record and row records
c  ifmt    input format: present if input is ascii
c  ianom   selection of z anomaly
c  xo      x coordinate of lower left corner of grid
c  yo      y coordinate of lower left corner
c  del     x and y increment (must be positive)
c  idirx   set to 1 when x coordinates decrease with increase in column,
c           postive west longitude for instance.
c  nc      number of columns
c  nr      number of rows (nc*nr < 1.3e6)
c radius   in horizontal data units, grid points with no data inside
c          this radius have a "no data" value inserted (dval).
c  npmin   number of data points within "radius" distance
c           before grid point considered valid
c  epsm    in z data units, iteration cutoff
c  nim     maximum iterations per block
c lapovr   number of rows overlapping next block
c  slope   a distance weighting ratio to decrease aliasing
c           by combining all data in a small area
c region   set .ne. 0. to save regional grid
c  whole   set .ne. 0. to save unradiused grid
c
c program breaks grid area into blocks containing no more than
c 5000 points.  for each block: a temporary binary file containing the
c input data is read, an initial grid is interpolated using
c one-dimensional interpolation to fill holes, data points are assigned
c to grid points, and iteration using minimum curvature difference
c equations attempts to honor the data points.
c continuity between blocks is provided by initializing the next block
c with whatever overlap is available, and inserting values from
c the regional grid where data are sparse.
c
c                         Mike Webring
c          U.S. Geological Survey, Regional Geophysics
c                    po box 25046, stop 964
c                    Denver Federal Center
c                    Denver, Colorado 80225
c
c                         Oct. 1978
c
c*****************************************************************************
          external dl(descriptors)
          dimension za(10),ida(14),p(2),idim(3),xloc(4)
          common /fmt/ifmta(14),ibr,idirx
```

```
            common /datai/ xyz(300)
            common /array/ wrk(40000)
            common /contn1/ zl(375),zb(375)
            common /gparm/mxc,mxr,nc,nr,nsec,ntier,lapovr,nim,epsm,
            & dv,slope,mdel
            common /qparm/ihfw,mxcq,mxrq,maxq,maxg
            common /assem/ ntot(2)
            character*56 id,ifmt,cf,ifile,ofile,blank
            character tmp*13,tmp2*13,mask*11
            equivalence (ida(1),id),(ifmta(1),ifmt)
            namelist /parms/id,nc,nr,xo,yo,del,idirx,nim,epsm,whole,
            & ifile,ofile,lapovr,ifmt,npmin,radius,ianom,region,slope
            data inp/5/,dval/o37677777777/,p/"min-","curv"/,nz/1/,blank/" "/
            data tmp/"minc_data.tmp"/,tmp2/"sub_grids.tmp"/,mask/"masking.tmp"/
c   dval (default grid value) is the no-data flag and may be any number.
c   the approx value is 1.7e38
c
            dv=dval
            nwrk=40000
            npidg=375
            npb=100
c   care must be taken when adapting these
c   array sizes to other machines
            wrk(nwrk)=0.
            zb(npidg)=0.
            npb3=npb*3
            xyz(npb3)=0.
            mdel=0
            maxg=0
            ntot(1)=0
            ntot(2)=0
            do 1 i=1,14
            ida(i)="    "
1           ifmta(i)="    "
            ifile=" "
            ofile=" "
            del=-1
            nc=0
            nr=0
            epsm=0.
            npmin=1
            lapovr=10
            nim=20
            idirx=0
            radius=0.
            ianom=0
            region=0.
            slope=0.
            whole=0.
c   clear previous file attachments
c   (site dependent)
            call minc_clr
            print 2
2           format(" enter command filename :"$)
            read(inp,37)cf
37          format(v)
            open(9,file=cf,mode="in",form="formatted")
            read(09,parms)
            close(9)
            if(del.le.0.) stop "negative or zero del"
```

```
      if(nc.le.4 .or. nr.le.4) stop " nc or nr < 5"
      if(ifile.ne.blank) go to 4
      print 5
5     format(" enter ifile ofile :")
      read(inp,37) ifile,ofile
      print 6
6     format(" enter input format :")
      read(inp,38) ifmt
38    format(a56)
      if(ifmta(1).ne."    ") go to 9
      print 10
10    format(" z anomaly number :"$)
      read(inp,37) ianom
9     print 7
7     format(" enter title :")
      read(inp,38) id
4     continue
      if(ifmt.eq." ") open(9,file=ifile,mode="in",form="unformatted")
      if(ifmt.ne." ") open(9,file=ifile,mode="in",form="formatted")
      open(12,file=ofile,mode="inout",form="unformatted")
      if(lapovr.lt.4) lapovr=4
      delx=del
      xout=xo
      if(idirx.eq.0) go to 8
      delx=-del
      idirx=-1
      xo=-xo
8     write(12)id,p,nc,nr,nz,xout,delx,yo,del
      nco=nc
      nro=nr
      mxc=nc+4
      mxr=nr+4
      if(slope.ne.0.0) slope=1./slope
      ihfw=int(abs(radius)/del+.5)
      if(ihfw.eq.0) go to 20
      open(14,file=mask,mode="inout",form="unformatted")
      iwind=2*ihfw
      mxcq=nco+iwind
      mxrq=nro+iwind
      iwind=iwind+1
      xo2=xo-ihfw*del
      yo2=yo-ihfw*del
      write(14) id,p,mxcq,mxrq,nz,xo2,del,yo2,del
20    xo=xo-2.*del
      yo=yo-2.*del
c
c  partition grid into blocks
      call prtish(nwrk,npidg,mxc,mxr,nc,nr,nsec,ntier,lapovr)
      print 30,nsec,ntier,nc,nr
30    format(/," nsec=",i3,",  ntier=",i3,",  block size: nc=",
      & i3,",  nr=",i3)
      if(nsec.eq.0) stop
      ng=nc*nr
      nblk=nsec*ntier
      if(nblk.gt.npidg) stop "argh...grid too large"
      n=int(float(nwrk)/float(nblk*3))
      if(n.gt.npb) n=npb
      if(nblk.eq.1) lapovr=1
      npb3=3*n
c
```

```
c   input data, prepare random access file
c   all direct access files are indexed with integer variables
c   record length in tmp is 2+npb3 words
c   number of records depends on input data file
          open(13,file=tmp,access="direct",form="unformatted")
          call randp(del,xo,yo,npb3,za,ianom)
          close(9)
c   record length in tmp2 is nc*nr words
c   maximum number of records is 2*nsec
          open(9,file=tmp2,access="direct",form="unformatted")
c
c   calculate coarse regional grid
          open(16,file="regional.tmp",mode="inout")
          open(17,file="regional.flag",mode="inout")
          call rejonl(xo,yo,del,xyz,npb3)
c
c   produce grid at specified interval.
          call pcontl(nco,nro,ng,wrk,npb3,xyz)
          close(9)
          close(13)
          close(16)
          close(17)
c
          if(ihfw.eq.0.) go to 999
          open(13,file="whole.tmp",mode="inout")
          rewind(12)
c   copy completely defined grid into file whole.tmp
          read(12) id,p,idim,xloc
          write(13) id,p,idim,xloc
          do 40 j=1,idim(2)
          call rowio(idim(1),wrk,1,12,13,ie)
40        continue
          rewind(12)
          rewind(13)
          rewind(14)
c   trim completed grid to data coverage
          n1=mxc+1
          n2=mxcq+n1
          if(mxcq*iwind-1+n2 .gt. nwrk) go to 888
          call fitr(wrk(n2),mxcq,iwind,wrk(n1),nco,nro,wrk(1),
          & 13,14,12,npmin)
888       continue
          close(12)
          close(13)
          close(14)
c   dl is used to delete tmp files
          call dl("masking.tmp")
          if(whole.eq.0.) call dl("whole.tmp")
999       continue
          if(region.eq.0.) call dl("regional.tmp")
          call dl("regional.flag")
          call dl("minc_data.tmp")
          call dl("sub_grids.tmp")
          stop
          end
```

```
      subroutine prtish(nwrk,nsave,mc,mr,nc,nr,nsec,ntier,lap)
c  optimize subdivision of the main grid.
c  maximize block size subject to the side ratio less than 3:1
      dimension nc1(3),nr1(3)
c
      nmax=nwrk/8
      is=0
      js=0
      width=aint(sqrt(float(nmax))-float(lap))
      nsec1=int(float(mc)/width+.5)-1
      ntier1=int(float(mr)/width+.5)-1
      if(nsec1.lt.1) nsec1=1
      if(ntier1.lt.1) ntier1=1
      n=nsec1
      do 1 i=1,3
      nc1(i)=int(float(mc-lap)/float(n)+.9999)+lap
1     n=n+1
      n=ntier1
      do 2 i=1,3
      nr1(i)=int(float(mr-lap)/float(n)+.9999)+lap
2     n=n+1
      nb=1000
      do 3 i=1,3
      do 3 j=1,3
      ratio=float(nc1(i))/float(nr1(j))
c  ratio limits match zl&zb array size
      if(ratio.gt.3.0 .or. ratio.lt..3333) go to 3
      if(nc1(i)*nr1(j).gt.nmax) go to 3
c  3 cols and rows are saved to provide continuity between blocks
      if(nc1(i)*3. .gt. nsave) go to 3
      if(nr1(j)*3. .gt. nsave) go to 3
      nblk=(nsec1+i-1)*(ntier1+j-1)
      if(nblk.ge.nb) go to 3
      nb=nblk
      js=j
      is=i
3     continue
      if(is.eq.0) go to 9
      nc=nc1(is)
      nr=nr1(js)
      nsec=nsec1+is-1
      ntier=ntier1+js-1
      return
9     print 8
8     format(" problem with partition")
      nsec=0
      return
      end
```

```
      subroutine randp(del,xo,yo,npb3,za,iz)
c  there are nsec*ntier pigeon holes, each of
c  which contains all data necessary for iterating
c  a subgrid.  because of overlap on left and
c  bottom sides one data point can appear in several holes.
      common /array/ wrk(40000)
      common /fmt/ ifmt(14),ibr,idirx
      common /gparm/ mxc,mxr,nc,nr,nsec,ntier,lap,nim,epsm,dv
c     temporary use of contin by counters
      common /contn1/ loc(375),ioff(375)
      dimension za(iz)
      character*8 id
      character*4 blank,test
      equivalence (ifmt(1),test)
      data ind/9/,iu/13/,blank/"    "/
c
      ibr=-1
      if(iz.gt.0) ibr=1
      if(test.ne.blank) ibr=0
      ic=0
      nblk=nsec*ntier
      nchk=npb3-3
      npb=npb3/3
      err=1.e-2
      fudg=del*err
      xmax=float(mxc-1)*del+xo-fudg
      ymax=float(mxr-1)*del+yo-fudg
      xmin=xo+fudg
      ymin=yo+fudg
c  dimension of pigeon hole minus overlap
      cl=float(nc-lap)
      rl=float(nr-lap)
      dux=1./cl
      duy=1./rl
      fx=(float(lap)-.1)*dux
      fy=(float(lap)-.1)*duy
c  ensures smallest index (igx,y) in 'bwts' is 1
      err2=err*.5+1.
      xlmt=1.-err2*dux
      ylmt=1.-err2*duy
      rdel=1./del
      x2=1.-xo*rdel
      y2=1.-yo*rdel
c   'loc' contains the address where a block will be written.
c   a linked list is formed by 'next'.
      do 11 i=1,nblk
      ioff(i)=1
11    loc(i)=i
      next=nblk+1
      do 12 i=1,npb3*nblk
12    wrk(i)=dv
c
c  read data, find pigeon hole
100   if( ibr ) 101,102,103
101   read(ind,end=50) x,y,z
      go to 104
102   read(ind,ifmt,end=50) x,y,z
      go to 104
103   read(ind,end=50) id,x,y,za
      z=za(iz)
```

```
104       if(idirx)105,106,106
105       x=-x
106       if(x.gt.xmax .or. x.lt.xmin) go to 100
          if(y.gt.ymax .or. y.lt.ymin) go to 100
c
          ib=0
          ic=ic+1
c     x&y converted to grid units
          x=x*rdel+x2
          y=y*rdel+y2
          bx=x*dux
          by=y*duy
          ibx=int(bx+xlmt)
          iby=int(by+ylmt)
          tstx=bx-float(ibx-1)
          tsty=by-float(iby-1)
          if(ibx.le.nsec) go to 17
          ibx=nsec
          tstx=1.
17        if(iby.le.ntier) go to 18
          iby=ntier
          tsty=1.
18        ibset=(iby-1)*nsec+ibx
          mblk=ibset
c
c  put data in pigeon hole, output when full
19        ixs=(mblk-1)*npb3
          ip=ixs+ioff(mblk)
          wrk(ip)=x
          wrk(ip+1)=y
          wrk(ip+2)=z
          ioff(mblk)=ioff(mblk)+3
          if(ioff(mblk).lt.npb3) go to 22
          ndp=(ioff(mblk)-1)/3
          call wrblk2(loc(mblk),next,ndp,wrk(ixs+1),npb3,iu)
          loc(mblk)=next
          next=next+1
          ioff(mblk)=1
c
c  is cata in overlap area ?
22        ib=ib+1
          go to (23,24,25,100)ib
23        if(tsty.gt.fy .or. iby.eq.1) go to 22
          mblk=ibset-nsec
          go to 19
24        if(ibx.eq.1 .or. iby.eq.1) go to 22
          if(tsty.gt.fy .or. tstx.gt.fx) go to 22
          mblk=ibset-nsec-1
          go to 19
25        if(tstx.gt.fx .or. ibx.eq.1) go to 100
          mblk=ibset-1
          go to 19
c
c   output unfilled pigeon holes
50        ip=1
          next=0
          do 51 i=1,nblk
          ndp=(ioff(i)-1)/3
          call wrblk2(loc(i),next,ndp,wrk(ip),npb3,iu)
51        ip=ip+npb3
```

```
c
      print 52,ic
52    format(i8," data points in area")
      if(ic.gt.0) return
      print 53
53    format(" coordinate mismatch or incorrect ianom parameter ?")
      stop
      end
```

```
      subroutine rejonl(xo,yo,del,xyz,npb3)
c   the regional grid is used in data sparse areas
c   to provide continuity and faster convergence.
      common /array/zg(5000),iod(5000),b(30000)
      common /gparm/ mxc,mxr,nc,nr,nsec,ntier,lap,nim,epsm,
      & dv,slope1,mdel
      common /contn1/ tmp(375),tmp2(375)
      dimension izg(5000),xyz(npb3)
      equivalence (izg,zg)
      character id*56,p*8
      logical lastt,lasts
      data iu/13/,id/"regional grid"/,p/"minc"/,nz/1/
c
      nimr=nim
      if(nimr.lt.20) nimr=20
      mdel=4
10    mc=(mxc-1)/mdel+6
      mr=(mxr-1)/mdel+6
      nn=mc*mr
      if(nn.le.5000) go to 11
      mdel=mdel+1
      go to 10
c
11    do 12 i=1,nn
12    zg(i)=dv
      do 13 i=1,nn
13    iqd(i)=0
      do 14 i=1,nn*6
14    b(i)=0.0
      slope=.2
      lastt=.false.
      dx=nc-lap
      dy=nr-lap
      endy=dy
      rdel=1./float(mdel)
c
      do 50 j=1,ntier
      endx=dx
      lasts=.false.
      if(j.eq.ntier) lastt=.true.
      do 40 i=1,nsec
      if(i.eq.nsec) lasts=.true.
      iadr=(j-1)*nsec+i
20    i2=1
      read(iu'iadr) next,np,xyz
      iadr=next
      if(np.eq.0) go to 40
      do 25 k=1,np*3,3
      k1=k+1
c   eliminate overlap in data
      if(lasts) go to 22
      if(xyz(k).gt.endx) go to 25
22    if(lastt) go to 23
      if(xyz(k1).gt.endy) go to 25
c   convert data to regional grid units
23    xyz(i2)=(xyz(k)-1.0)*rdel+3.0
      xyz(i2+1)=(xyz(k1)-1.0)*rdel+3.0
      xyz(i2+2)=xyz(k+2)
      i2=i2+3
25    continue
```

```
      np2=i2-1
      call bwts(0.,0.,mc,mr,zg,iqd,b,xyz,np2,slope,0)
      if(next.ne.0) go to 20
40    endx=endx+dx
50    endy=endy+dy
c
      np=0
      call bwts(0.,0.,mc,mr,zg,iqd,b,xyz,np,slope,1)
      call gridr(mc,mr,zg,tmp,ier)
      call curvmn(zg,iqd,b,mc,mr,epsm,nimr,st,end,ni)
c     print 60,st,end,ni
60    format(" start",1pe15.4," end",e15.4," iter.",i4)
c
c   remove border
      ib=mc*2+3
      ie=mc*3-2
      n=1
      do 100 j=3,mr-2
      do 101 i=ib,ie
      zg(n)=zg(i)
      iqd(n)=iqd(i)
101   n=n+1
      ib=ib+mc
100   ie=ie+mc
      mc=mc-4
      mr=mr-4
      dr=mdel*del
      write(16) id,p,mc,mr,nz,xo,dr,yo,dr
      write(17) id,p,mc,mr,nz,xo,dr,yo,dr
      ir=1
      do 200 j=1,mr
      call rowio(mc,izg(ir),0,16,16,ie)
      call rowio(mc,iqd(ir),0,17,17,ie)
200   ir=ir+mc
      return
      end
```

```
      subroutine pcontl(nco,nro,ng,zg,npb3,xyz)
      common /array/ w(40000)
      common /contn1/zl(375),zb(375)
      common /gparm/mxc,mxr,nc,nr,nsec,ntier,lap
      common /qparm/ihfw,mxcq,mxrq,maxq,maxg
      dimension iw(40000),zg(ng)
      equivalence (w(1),iw(1))
      data nsav/3/,nwrk/40000/
      data lout/6/,isub/9/,igrid/12/,iu/13/,msk/14/
c
      mswt=-1
      maxg=(nwrk-ng)/nco
      nn=nwrk/8
      n2=nn+1
      n3=nn+n2
c
      if(ihfw.eq.0) go to 3
c  write ihfw rows as border for masking grid
      maxq=(nwrk-ng)/mxcq
99    do 1 i=1,mxcq
1     iw(i)=0
      do 2 i=1,ihfw
2     call rowio(mxcq,iw,0,msk,msk,ie)
      if(mswt.eq.0) return
c
3     ir=0
      nbot=nc*nsav
      dx=float(nc-lap)
      dy=float(nr-lap)
      byo=0.
c  for each block, iterate a subgrid
      do 100 j=1,ntier
      bxo=0.
      write(lout,4) j
4     format(/,"   tier ",i2)
      do 50 i=1,nsec
      if(j.eq.1) go to 10
c  get lower boundary condition
c   caution zg(1) is w(1)
      read(isub'i) zg
      i2=(nr-lap)*nc+1
      do 5 ii=1,nbot
      zb(ii)=zg(i2)
5     i2=i2+1
10    mblk=(j-1)*nsec+i
      call icontl(i,j,ihfw,bxo,byo,xyz,npb3)
      if(i.eq.nsec) go to 50
c  save leftside boundary
      is=nc-lap+1
      i3=1
      do 14 jj=1,nr
      i2=is
      do 12 ii=1,nsav
      zl(i3)=w(i2)
      i3=i3+1
12    i2=i2+1
14    is=is+nc
50    bxo=bxo+dx
      call assemb(nc,nr,iw(1),nco,maxg,iw(ng+1),j,0,isub,igrid)
      if(ihfw.eq.0) go to 100
```

```
      call assemb(nc,nr,iw(1),mxcq,maxq,iw(ng+1),j,ihfw,isub,msk)
1C0   byo=byo+dy
      if(ihfw.eq.0) return
      mswt=0
      go to 99
      end
```

```
      subroutine fitr(m,mxcq,iwindw,n,mxc,mxr,r,ing,inq,jgrd,npmin)
c  blanks no-data areas by comparing the number of data points
c  in a square 2*radius on a side with the npmin parameter.
      dimension idim(3),xloc(4)
      dimension m(mxcq,iwindw),n(mxcq),r(mxc)
      character id*56,p*8
      data dv/o37677777777/
      read(ing) id,p,idim,xloc
      write(jgrd) id,p,idim,xloc
      read(inq)
      do 1 j=1,iwindw
1     call rowio(mxcq,m(1,j),-1,inq,inq,ie)
      do 2 i=1,mxcq
2     n(i)=0
      do 3 j=1,iwindw
      do 3 i=1,mxco
3     n(i)=n(i)+m(i,j)
      iptr=1
      do 8 jout=1,mxr
      read(ing) yo,r
      n2=0
      do 4 i=1,iwindw
4     n2=n2+n(i)
      if(n2.lt.npmin) r(1)=dv
      ndxl=1
      ndxr=iwindw+1
      do 5 i=2,mxc
      n2=n2-n(ndxl)+n(ndxr)
      if(n2.lt.npmin) r(i)=dv
      ndxl=ndxl+1
5     ndxr=ndxr+1
      write(jgrd) yo,r
      do 6 i=1,mxcq
6     n(i)=n(i)-m(i,iptr)
      call rowio(mxcq,m(1,iptr),-1,inq,inq,ie)
      do 7 i=1,mxcq
7     n(i)=n(i)+m(i,iptr)
      iptr=iptr+1
      if(iptr.gt.iwindw) iptr=1
8     continue
      return
      end
```

```
      subroutine icontl(ns,nt,ihfw,xo,yo,xyz,npb3)
c  iteration control for one block
c  input xyz data, output iterated block
      common /array/ zg(5000),iqd(5000),b(30000)
      common /gparm/mxc,mxr,nc,nr,nsec,ntier,lap,nim,epsm,dv,slope
      common /contn1/ zl(375)
      dimension izg(5000),xyz(npb3)
      equivalence (zg(1),izg(1))
      data lout/6/,isub/9/,iu/13/
c
      nn=nc*nr
      iadr=(nt-1)*nsec+ns
      binit=1.0
      if(slope.ne.0.0) binit=0.0
      do 5 i=1,nn
5     zg(i)=dv
      do 10 i=1,nn
10    iqd(i)=0
      do 15 i=1,nn*6
15    b(i)=binit
      idata=0
20    read(iu'iadr) next,np,xyz
      np3=np*3
123   format(3i6)
      if(np.eq.0) go to 30
      call bwts(xo,yo,nc,nr,zg,iqd,b,xyz,np3,slope,0)
      idata=1
      if(next.eq.0) go to 30
      iadr=next
      go to 20
30    if(idata.eq.0) go to 40
      call bwts(xo,yo,nc,nr,zg,iqd,b,xyz,np3,slope,1)
40    continue
c
      write(lout,220)
220   format(" ")
c  write out mask before boundary conditions set
      if(ihfw.gt.0) call wrblk(ns+nsec,iqd,nn,isub)
      call contin(ns,nt,xo,yo)
c     temporary use of zl as work array
      call gridr(nc,nr,zg,zl,ier)
      if(ier.ne.0) stop " gridr error"
c   begin iteration
      call curvmn(zg,iqd,b,nc,nr,epsm,nim,st,end,ni)
      write(lout,26) st,end,ni
26    format(" initial error=",1pe9.2," end error=",e9.2,
     & " iterations=",i3)
c  write iterated subgrid
      call wrblk(ns,izg,nn,isub)
      return
      end
```

```
      subroutine bwts(xo,yo,nc,nr,zg,iqd,b,xyz,npb3,slope,icalc)
c   associate data with grid locations and calculate weights
c   for minimum curvature equations.
c   'slope' > 0 indicates that all data values within .75 grid unit
c   radius will be combined by distance weighting to produce one
c   pseudo-data value to be used by the minimum curvature equations.
c   'slope' = 0 indicates that the closest data value is the only
c   one used.
      dimension dxs(4),dys(4),itabl(4)
      dimension xyz(npb3),zg(1),iqd(1),b(1)
      logical lslope
      data dv/o377777777777/,itabl/3,4,2,1/
c
      nn=nc*nr
      if(icalc.ne.0) go to 200
      if(npb3.le.0) return
      np=npb3/3
      nc1=nc-1
      nr1=nr-1
      lslope=.false.
      if(slope.gt.0.0) lslope=.true.
      n=1
c   cycle through data array
      do 100 i=1,np
      x=xyz(n)-xo
      igx=int(x)
      dx=x-float(igx)
      y=xyz(n+1)-yo
      igy=int(y)
c  debugging tests, is data inside block?
c     if(igx.gt.nc1 .or. igx.lt.1) go to 997
c     if(igy.gt.nr1 .or. igy.lt.1) go to 999
c     go to 9
c997  print 996,igx
c996  format(i3,"x"$)
c     go to 100
c999  print 998,igy
c998  format(i3,"y"$)
c     go to 100
9     dy=y-float(igy)
      ig=(igy-1)*nc+igx
      dx1=dx-1.
      dxs(1)=dx*dx
      dxs(4)=dxs(1)
      dxs(2)=dx1*dx1
      dxs(3)=dxs(2)
      dy1=dy-1.
      dys(1)=dy*dy
      dys(2)=dys(1)
      dys(3)=dy1*dy1
      dys(4)=dys(3)
      x2=dx
      y2=dy
c   test the four cell corners
      do 50 k=1,4
      go to (13,10,11,12)k
10    ig=ig+1
      x2=dx1
      go to 13
11    ig=ig+nc
```

```
        y2=dy1
        go to 13
12      ig=ig-1
        x2=dx
13      r2=dxs(k)+dys(k)
c   data must be within .75 grid units
        if(r2.gt..5625) go to 50
        ndx1=(ig-1)*6+1
        if(lslope) go to 30
        if(r2.ge.b(ndx1)) go to 50
        z=xyz(n+2)
        zg(ig)=z
        iqd(ig)=k
        if(r2.lt..0025) iqd(ig)=-1
c  array iqd contains quadrant number which the
c  datapoint is in, -1 locks zg value.
        b(ndx1)=r2
        b(ndx1+1)=x2
        b(ndx1+2)=y2
        b(ndx1+3)=dys(k)
        b(ndx1+5)=z
        go to 50
30      dwt=1./(r2+slope)
        n1=ndx1+1
        n2=ndx1+2
        n5=ndx1+5
        b(ndx1)=b(ndx1)+dwt
        b(n1)=b(n1)+x2*dwt
        b(n2)=b(n2)+y2*dwt
        b(n5)=b(n5)+xyz(n+2)*dwt
50      continue
100     n=n+3
        return
c
c   calculate pseudo data x,y,z,quadrant
200     if(slope.eq.0.0) go to 219
        ndx1=1
        do 210 ndx=1,nn
        zg(ndx)=dv
        iqd(ndx)=0
        if(b(ndx1).eq.0.0) go to 210
        rwt=1./(b(ndx1))
        n1=ndx1+1
        n2=ndx1+2
        n3=ndx1+3
        b(n1)=b(n1)*rwt
        b(n2)=b(n2)*rwt
        b(n3)=b(n2)*b(n2)
        n5=ndx1+5
        b(n5)=b(n5)*rwt
        zg(ndx)=b(n5)
        r2=b(n1)*b(n1)+b(n2)*b(n2)
        if(r2.lt..0025) go to 208
        ix=1
        if(b(n1).ge.0.0) ix=2
        iy=0
        if(b(n2).ge.0.0) iy=2
        iqd(ndx)=itabl(ix+iy)
        go to 210
208     iqd(ndx)=-1
```

```
210         ndx1=ndx1+6
c
c  solution of weighting matrix
219         ib=nc*2+3
            ie=nc*3-2
            do 220 j=3,nr-2
            ndx1=(ib-1)*6+1
            do 222 i=ib,ie
            if(iqd(i))222,222,221
221         dx=abs(b(ndx1+1))
            dy=abs(b(ndx1+2))
            dy2=b(ndx1+3)
            f1=dx*(dx+dy+dy+1.)
            b5=4./(f1+dy2+dy)
            b4=(b5*f1*.5)-1.
            b5dx=b5*dx
            b4b4=b4+b4
            b3=b5dx*(dy+1.)-b4b4
            b(ndx1+1)=2.+b5dx-(b5*dy2+b4b4+b3)
            b(ndx1)=b3+b4-b5dx
            b(ndx1+2)=b3
            b(ndx1+3)=b4
            b(ndx1+4)=b5*b(ndx1+5)
            b(ndx1+5)=1./(1.+b(ndx1)+b(ndx1+1)+b3+b4+b5)
222         ndx1=ndx1+6
            ib=ib+nc
220         ie=ie+nc
            return
            end
```

```
          subroutine contin(ns,nt,bxo,byo)
c   continuity is provided by locking previously iterated
c   grid values along left and bottom sides of the block
          common /array/ zg(5000),iqd(5000),b(30000)
          common /contn1/zl(375),zb(375)
          common /gparm/mxc,mxr,nc,nr,nsec,ntier,lap,nim,epsm,
          & dv,slope,mdel
          common /datai/ irow(150),iflag(150)
          dimension r(150),ins(35)
          equivalence (irow,r)
          character id*56,p*8
          data nsav/3/,lout/6/,ntmp/150/
c
          if(ns.eq.1) go to 20
c   insert leftside boundary condition
          j=1
          igs=1
          do 15 ii=1,nr
          ig=igs
          do 14 i=1,nsav
          zg(ig)=zl(j)
          if(i.ne.nsav) iqd(ig)=-1
          ig=ig+1
14        j=j+1
15        igs=igs+nc
c
20        if(nt.eq.1) go to 30
c   insert bottom boundary condition
          do 24 i=1,2*nc
          zg(i)=zb(i)
24        iqd(i)=-1
          do 25 i=2*nc+1,3*nc
25        zg(i)=zb(i)
c
c   insert control data from regional surface
c   into data sparse areas.
30        do 35 i=1,ntmp
35        r(i)=dv
          do 36 i=1,35
36        ins(i)=0
          fmdel=1./float(mdel)
          rewind(16)
          rewind(17)
          read(16) id,p,mc,mr,nz,xo,del,yo,dely
          read(17)
          if(mc.gt.ntmp) return
          call rowio(mc,irow,-1,16,16,ie)
          call rowio(mc,iflag,-1,17,17,ie)
c   iflag array contains quadrant info from the regional
c   surface, iflag(n) ne 0 indicates data nearby.
c
          iyo=int(byo+.0001)+1
          iyr=1
110       if(iyr.ge.iyo) go to 120
          call rowio(mc,irow,-1,16,16,ie)
          call rowio(mc,iflag,-1,17,17,ie)
          if(ie.ne.0) go to 999
          iyr=iyr+mdel
          go to 110
c
```

```
120       ixo=int(bxo+.0001)+1
          ixc=1
130       if(ixc.ge.ixo) go to 140
          ixc=ixc+mdel
          go to 130
c
140       mfc=ixc-int(bxo+.0001)
          mfr=iyr-int(byo+.0001)
          n=mfr
c
150       m=mfc
          m2=(ixc-1)/mdel + 1
          mx=2
c
160       if(m.gt.nc) go to 170
          mn=(n-1)*nc+m
          if(m2.gt.mc) stop "contin: indexing"
          itmp=0
          if(iflag(m2).ne.0) go to 161
c  insert control value
          if(zg(mn).ne.dv) go to 161
          zg(mn)=r(m2)
          itmp=1
c  interpolate between adjacent controls
          if(ins(mx-1).eq.0) go to 162
          dz=(zg(mn)-zg(mn-mdel))*fmdel
          do 163 ix=mn-mdel+1,mn-1
163       zg(ix)=zg(ix-1)+dz
162       if(ins(mx).eq.0) go to 161
          dz=(zg(mn)-zg(mn-mdel*nc))*fmdel
          do 164 iy=mn-(mdel-1)*nc,mn-nc,nc
164       zg(iy)=zg(iy-nc)+dz
161       ins(mx)=itmp
          mx=mx+1
          m=m+mdel
          m2=m2+1
          go to 160
170       n=n+mdel
          if(n.gt.nr) go to 999
          call rowio(mc,irow,-1,16,16,ie)
          call rowio(mc,iflag,-1,17,17,ie)
          if(ie.ne.0) go to 999
          go to 150
999       continue
          if(ie.ne.0) print 888
888       format(" eof regional grid")
          return
          end
```

```
          subroutine gridr(nc,nr,zg,wz,ier)
c   initialize grid with reasonable anomalies
c   wz at least max(nc,nr)
          dimension iw1(3),jset(3)
          dimension zg(1),wz(1)
          data dval/o37677777777/,iw1/3,7,9/
          ier=0
          nn=nc*nr
          nsep=(iw1(3)-1)/2
          ns1=nsep+1
c
c   insert control point net
c      distances for ring averages specified by iw1
          if(nc.lt.nsep+ns1 .or. nr.lt.nsep+ns1) go to 140
          do 135 i=1,3
          ihf=(iw1(i)-1)/2
135       jset(i)=ihf*nc+ihf
          ipass=1
          iflag=1
130       iw=iw1(iflag)
          iset=jset(iflag)
          ibnd=(iw-1)/2
131       nass=0
          do 110 jj=ns1,nr-ibnd,nsep
          ip=(jj-1)*nc+ns1
          do 100 ii=ns1,nc-ibnd,nsep
          if(zg(ip).ne.dval) go to 100
          ip2=ip-iset
          it=0
          t=0.0
          do 121 j=1,iw
          ips=ip2
          do 120 i=1,iw
          if(zg(ip2).eq.dval) go to 120
          t=zg(ip2)+t
          it=it+1
120       ip2=ip2+1
121       ip2=ips+nc
          if(it.eq.0) go to 100
          zg(ip)=t/float(it)
          nass=nass+1
100       ip=ip+nsep
110       continue
          if(nass.eq.0 .and. ipass.gt.2) go to 140
          if(ipass.gt.10) go to 140
          ipass=ipass+1
          if(iflag-2)133,132,131
133       iflag=2
          go to 130
132       iflag=3
          go to 130
140       continue
c
c   fill holes
          inc=nc*nsep
          j=inc+1
          if(nr.lt.ns1) go to 21
          do 20 irow=ns1,nr-nsep,nsep
          call plugm3(nc,zg(j),dval)
20        j=j+inc
```

```
21        do 28 icol=1,nc
          j=icol
          do 22 k=1,nr
          wz(k)=zg(j)
22        j=j+nc
          call plugm3(nr,wz,dval)
          j=icol
          do 24 k=1,nr
          zg(j)=wz(k)
24        j=j+nc
28        continue
c  final check
          do 40 i=1,nn
          if(zg(i).eq.dval) go to 41
40        continue
          return
41        t=0.0
          it=0
          do 42 i=1,nn
          if(zg(i).eq.dval) go to 42
          t=t+zg(i)
          it=it+1
42        continue
          if(it.eq.0) stop " cannot init grid"
          t=t/float(it)
          print 43,t
43        format(" gridr init with",1pe15.5)
          do 44 i=1,nn
          if(zg(i).eq.dval) zg(i)=t
44        continue
          return
          end
```

```
        subroutine curvmn(zg,iqd,b,nc,nr,epsmx,nim,eps1,dn1,ni)
c    applies minimum curvature equations to the first
c   nc*nr elements of array zg.
c    array iqd contains nc*nr elements which indicate
c   for each mesh location the quadrant where a data
c   value is located. an iqd value of zero indicates
c   no data and -1 locks the present mesh value.
c    array b should contain 6*nc*nr elements used for
c   weighting when iqd is 1 to 4, in the case where
c   iqd is only 0 or -1, b can be of length one.
c    the over-relaxation parameter w increases
c   as the system converges until 1.7 is reached.
        dimension zg(1),iqd(1),b(1)
        data nimn/5/,lmtc/1/
        if(nc.lt.5 .or. nr.lt.5) return
        ni=0
        dn=1.e20
        w=1.3
        eps=0.
        eps1=0.
        epsm=abs(epsmx)
111     continue
        if(ni.ge.nim) go to 72
        eps=0.
c first row
        if(iqd(1))2,1,1
1       zg(1)=(( (2.*(zg(2)+zg(nc+1))-zg(nc+nc+1)-zg(3))*.5 )-
        & zg(1))*w+zg(1)
2       j1=nc+2
        j2=j1+nc
        if(iqd(2))4,3,3
3       zg(2)=(( (4.*(zg(3)+zg(j1))+2.*zg(1)-zg(4)-zg(j1-1)-
        & zg(j1+1)-zg(j2))*.16666667 )-zg(2))*w+zg(2)
4       do 6 i=3,nc-2
        j1=i+nc
        j2=j1+nc
        if(iqd(i))6,5,5
5       zg(i)=(( (4.*(zg(i-1)+zg(j1)+zg(i+1))-zg(j1+1)-zg(j1-1)-
        & zg(j2)-zg(i+2)-zg(i-2))*.14285714 )-zg(i))*w+zg(i)
6       continue
        if(iqd(nc-1))8,7,7
7       i=nc-1
        j1=i+nc
        zg(i)=(( (4.*(zg(i-1)+zg(j1))+2.*zg(i+1)-zg(i-2)-
        & zg(j1+1)-zg(j1-1)-zg(j1+nc))*.16666667 )-zg(i))*w+zg(i)
8       if(iqd(nc))10,9,9
9       j1=nc+nc
        zg(nc)=(( (2.*(zg(j1)+zg(nc-1))-zg(nc-2)-zg(j1+nc))*.5 )-
        & zg(nc))*w+zg(nc)
c second row
10      if(iqd(nc+1))12,11,11
11      i=nc+1
        j1=i+nc
        zg(i)=(( (4.*(zg(j1)+zg(i+1))+2.*zg(1)-zg(2)-
        & zg(i+2)-zg(j1+1)-zg(j1+nc))*.16666667 )-zg(i))*w+zg(i)
12      if(iqd(nc+2))14,13,13
13      i=nc+2
        j1=i+nc
        jm=i-nc
        zg(i)=(( (8.*(zg(j1)+zg(i+1))+4.*(zg(jm)+zg(i-1))-
```

```
          & 2.*zg(j1+1)-zg(jm+1)-zg(j1-1)-zg(i+2)-zg(j1+nc))*
          & 5.5555556e-2 )-zg(i))*w+zg(i)
 14       do 16 i=nc+3,nc+nc-2
          j1=i+nc
          jm=i-nc
          if(iqd(i))16,15,15
 15       zg(i)=(( (8.*(zg(i-1)+zg(j1)+zg(i+1))+4.*(zg(jm))-
          & 2.*(zg(j1-1)+zg(j1+1))-zg(jm-1)-zg(jm+1)-
          & zg(j1+nc)-zg(i+2)-zg(i-2))*5.263158e-2 )-zg(i))*w+zg(i)
 16       continue
          i=nc+nc-1
          if(iqd(i))18,17,17
 17       j1=i+nc
          jm=i-nc
          zg(i)=(( (8.*(zg(j1)+zg(i-1))+4.*(zg(jm)+zg(i+1))-2.*zg(j1-1)-
          & zg(jm-1)-zg(j1+1)-zg(i-2)-zg(j1+nc))*5.5555556e-2 )-
          & zg(i))*w+zg(i)
 18       i=nc+nc
          if(iqd(i))20,19,19
 19       j1=i+nc
          jm=i-nc
          zg(i)=(( (4.*(zg(j1)+zg(i-1))+2.*zg(jm)-zg(jm-1)-
          & zg(i-2)-zg(j1-1)-zg(j1+nc))*.16666667 )-zg(i))*w+zg(i)
c rows 3 to nr-2
 20       do 39 j=3,nr-2
          i=(j-1)*nc+1
          if(iqd(i))22,21,21
 21       j1=i+nc
          jm=i-nc
          zg(i)=(( (4.*(zg(i+1)+zg(j1)+zg(jm))-zg(j1+nc)-zg(j1+1)-zg(i+2)-
          & zg(jm+1)-zg(jm-nc))*.14285714 )-zg(i))*w+zg(i)
 22       i=i+1
          if(iqd(i))24,23,23
 23       j1=i+nc
          jm=i-nc
          zg(i)=(( (8.*(zg(j1)+zg(i+1)+zg(jm))+4.*zg(i-1)
          & -2.*(zg(j1+1)+zg(jm+1))-zg(j1-1)-zg(j1+nc)-zg(i+2)-
          & zg(jm-nc)-zg(jm-1))*5.2631578e-2 )-zg(i))*w+zg(i)
 24       do 35 j2=3,nc-2
          i=i+1
          if(iqd(i))35,25,25
 25       j1=i+nc
          jm=i-nc
          d=zg(i)
          if(iqd(i))26,26,27
 26       d=(( (8.*(zg(i+1)+zg(i-1)+zg(jm)+zg(j1))-2.*(zg(j1+1)+zg(jm+1)+
          & zg(jm-1)+zg(j1-1))-zg(j1+nc)-zg(jm-nc)-zg(i-2)-zg(i+2))*
          & .05 )-d)*w+d
          go to 33
 27       ndx=(i-1)*6+1
          b1=b(ndx)
          b2=b(ndx+1)
          b3=b(ndx+2)
          b4=b(ndx+3)
          b5=b(ndx+4)
          b6=b(ndx+5)
          go to (28,29,30,31)iqd(i)
 28       bu=b1*zg(jm+1)+b2*zg(jm)+b3*zg(i-1)+b4*zg(j1-1)
          go to 32
 29       bu=b1*zg(j1+1)+b2*zg(i+1)+b3*zg(jm)+b4*zg(jm-1)
```

```
      go to 32
30    bu=b1*zg(j1-1)+b2*zg(j1)+b3*zg(i+1)+b4*zg(jm+1)
      go to 32
31    bu=b1*zg(jm-1)+b2*zg(i-1)+b3*zg(j1)+b4*zg(j1+1)
32    t=.25*(zg(j1+nc)+zg(i-2)+zg(jm-nc)+zg(i+2))
     & +.5*(zg(j1-1)+zg(jm-1)+zg(jm+1)+zg(j1+1))-
     & (zg(j1)+zg(i-1)+zg(jm)+zg(i+1))
      d=(( (bu+b5-t)*b6 )-d)*w+d
33    epsln=d-zg(i)
      if(abs(epsln).lt.abs(eps)) go to 34
      eps=epsln
      ieps=i
34    zg(i)=d
35    continue
      i=i+1
      if(iqd(i))37,36,36
36    j1=i+nc
      jm=i-nc
      zg(i)=(( (8.*(zg(j1)+zg(i-1)+zg(jm))+4.*zg(i+1)-2.*(zg(j1-1)+
     & zg(jm-1))-zg(jm+1)-zg(jm-nc)-zg(i-2)-
     & zg(j1+nc)-zg(j1+1))*5.2631578e-2 )-zg(i))*w+zg(i)
37    i=i+1
      if(iqd(i))39,38,38
38    j1=i+nc
      jm=i-nc
      zg(i)=(( (4.*(zg(j1)+zg(i-1)+zg(jm))-zg(jm-nc)-zg(jm-1)-zg(i-2)-
     & zg(j1-1)-zg(j1+nc))*.14285714 )-zg(i))*w+zg(i)
39    continue
c row nr-1
40    i=(nr-2)*nc+1
      if(iqd(i))42,41,41
41    j1=i+nc
      jm=i-nc
      zg(i)=(( (4.*(zg(jm)+zg(i+1))+2.*zg(j1)-zg(jm-nc)-zg(jm+1)-
     & zg(i+2)-zg(j1+1))*.16666667 )-zg(i))*w+zg(i)
42    i=i+1
      if(iqd(i))44,43,43
43    j1=i+nc
      jm=i-nc
      zg(i)=(( (8.*(zg(i+1)+zg(jm))+4.*(zg(i-1)+zg(j1))-
     & 2.*zg(jm+1)-zg(jm-1)-zg(jm-nc)-zg(i+2)-
     & zg(j1+1))*5.5555556e-2 )-zg(i))*w+zg(i)
44    do 46 j=3,nc-2
      i=i+1
      if(iqd(i))46,45,45
45    j1=i+nc
      jm=i-nc
      zg(i)=(( (8.*(zg(i-1)+zg(jm)+zg(i+1))+4.*zg(j1)-
     & 2.*(zg(jm-1)+zg(jm+1))-zg(j1-1)-zg(i-2)-
     & zg(jm-nc)-zg(i+2)-zg(j1+1))*5.2631578e-2 )-zg(i))*w+zg(i)
46    continue
      i=(nr-1)*nc-1
      if(iqd(i))48,47,47
47    j1=i+nc
      jm=i-nc
      zg(i)=(( (8.*(zg(i-1)+zg(jm))+4.*(zg(j1)+zg(i+1))-2.*zg(jm-1)-
     & zg(j1-1)-zg(i-2)-zg(jm-nc)-zg(jm+1))*5.5555556e-2 )-
     & zg(i))*w+zg(i)
48    i=i+1
      if(iqd(i))50,49,49
```

```
49          j1=i+nc
            jm=i-nc
            zg(i)=(( (4.*(zg(i-1)+zg(jm))+2.*zg(j1)-zg(jm-nc)-zg(jm-1)-
            & zg(i-2)-zg(j1-1))*.16666667 )-zg(i))*w+zg(i)
c last row
50          i=i+1
            if(iqd(i))52,51,51
51          jm=i-nc
            zg(i)=(( (2.*(zg(i+1)+zg(jm))-zg(i+1)-zg(jm-nc))*.5 )-
            & zg(i))*w+zg(i)
52          i=i+1
            if(iqd(i))54,53,53
53          jm=i-nc
            zg(i)=(( (4.*(zg(i+1)+zg(jm))+2.*zg(i-1)-zg(i+2)-zg(jm+1)-
            & zg(jm-nc)-zg(jm-1))*.16666667 )-zg(i))*w+zg(i)
54          do 56 j=3,nc-2
            i=i+1
            if(iqd(i))56,55,55
55          jm=i-nc
            zg(i)=(( (4.*(zg(i-1)+zg(i+1)+zg(jm))-zg(i-2)-zg(jm-1)-
            & zg(jm-nc)-zg(jm+1)-zg(i+2))*.14285714 )-zg(i))*w+zg(i)
56          continue
            i=i+1
            if(iqd(i))58,57,57
57          jm=i-nc
            zg(i)=(( (4.*(zg(i-1)+zg(jm))+2.*zg(i+1)-zg(i-2)-
            & zg(jm-1)-zg(jm-nc)-zg(jm+1))*.16666667 )-zg(i))*w+zg(i)
58          i=i+1
            if(iqd(i))60,59,59
59          jm=i-nc
            zg(i)=(( (2.*(zg(i-1)+zg(jm))-zg(i-2)-zg(jm-nc))*.5 )-
            & zg(i))*w+zg(i)
60          if(ni)70,70,71
70          eps1=abs(eps/w)
71          ni=ni+1
            if(eps.eq.0) go to 72
            dn1=abs(eps/w)
            if(dn1.le.epsm .and. ni.ge.nimn) go to 72
            dlam=dn1/dn
            dn=dn1
            if(dlam.gt.1.) go to 74
            if(dlam.lt..8) go to 75
            if(w.ge.1.6) go to 75
            w=w+.1
            go to 75
74          if(iconv.eq.lmtc) go to 76
            iconv=iconv+1
            go to 75
76          w=w-.1*aint(dlam*10.-9.11)
            iconv=0
            if(w.lt.1.)w=1.
75          continue
            go to 111
72          return
            end
```

```
      subroutine assemb(nc,nr,m,ncout,nmax,mw,nt,ihfw,inp,jput)
c  input a completed tier of blocks and assemble row records
c  for output as the finished grid.
      dimension m(nc,nr),mw(ncout,nmax)
      common /gparm/mxc,mxr,idum(2),nsec,ntier,lap
      common /assem/ ntot(2)
      nco=mxc-4
      nro=mxr-4
      nc1=nc-lap
      nr1=nr-lap
      it=1
      ioff=0
      if(ihfw.eq.0) go to 30
      ioff=nsec
      it=2
30    js=1
      nt1=nt
      nrout=nro
      if(nt.gt.1) go to 31
      js=3
      nt1=999
31    if(ntier.eq.1) go to 3
      nrout=nr1-2
      if(ntier-nt1) 3,2,1
1     nrout=nr1
      go to 3
2     nrout=nro-(ntier-1)*nr1+2
      if(nrout.gt.nr) stop 999
3     ntime=0
4     ntime=ntime+1
      if(nrout-ntime*nmax.gt.0) go to 4
      nrow=nrout
      if(nrout.gt.nmax) nrow=nrout/ntime+1
c
      do 20 itime=1,ntime
      istop=nco+2
      itot=0
      is=3
      ie=nc1
      iws=ihfw+1
      if(itime.eq.ntime) nrow=nrout-(itime-1)*nrow
      do 9 isec=1,nsec
      l=isec+ioff
      read(inp'l) m
      if(ihfw.eq.0) go to 6
c  set data flags in masking grid for
c  use by subroutine fitr
      do 5 j=1,nr
      do 5 i=1,nc
      if(m(i,j).ne.0) m(i,j)=1
5     continue
6     jin=js
      if(isec.eq.nsec) ie=istop
      do 8 j=1,nrow
      iwcol=iws
      do 7 i=is,ie
      mw(iwcol,j)=m(i,jin)
7     iwcol=iwcol+1
8     jin=jin+1
      itot=(ie-is+1)+itot
```

```
        istop=nco-itot
        iws=iwcol
9       is=1
        izr=ihfw+nco+1
        do 19 j=1,nrow
        if(ihfw.eq.0) go to 18
        do 16 i=1,ihfw
16      mw(i,j)=0
        do 17 i=izr,ncout
17      mw(i,j)=0
18      call rowio(ncout,mw(1,j),0,jput,jput,ie)
        ntot(it)=ntot(it)+1
19      if(ntot(it).eq.nro) return
20      js=js+nrow
        if(nt.eq.ntier .and. ntot(it).ne.nro) print 21,it,ntot(it),nro
21      format(" assembly error: grid",i2," output",i4," rows out of",i4)
        return
        end
```

```
      subroutine plugm3(n,z,dv)
c  plug holes using linear interpolation
      dimension z(n)
      do 1 is=1,n
      if(z(is) .ne. dv) go to 2
1     continue
      return
2     ix=is
3     ix=ix-1
      if(ix.lt.1) go to 4
      z(ix)=z(is)
      go to 3
4     do 5 idv=is,n
      if(z(idv) .eq. dv) go to 6
5     continue
      return
6     is=idv-1
      do 7 ie=idv,n
      if(z(ie) .ne. dv) go to 10
7     continue
      ix=is
9     ix=ix+1
      if(ix.gt.n) return
      z(ix)=z(is)
      go to 9
10    dz=(z(ie)-z(is))/float(ie-is)
      do 11 i=is+1,ie-1
11    z(i)=z(i-1)+dz
      is=ie
      go to 4
      end
      subroutine rowio(n,iz,iop,idev,jdev,iend)
c  where read iop<0; write iop=0; r&w iop>0
      dimension iz(n)
      y=0.
      iend=0
      if(iop)1,2,1
1     read(idev,end=10) y,iz
      if(iop)9,9,2
2     write(jdev) y,iz
9     return
10    iend=1
      return
      end
      subroutine wrblk(loc,m,n,iu)
      dimension m(n)
      write(iu'loc) m
      return
      end
      subroutine wrblk2(loc,m,n,xyz,nb,iu)
      dimension xyz(nb)
      write(iu'loc) m,n,xyz
      return
      end
```

```
      subroutine minc_clr
c  cleanup routine written for the honeywell/multics.
      external io(descriptors),close_file(descriptors),
     & dl(descriptors)

      call io("detach","error_output")
      call io("attach","error_output","dump_")
      call close_file("-all")
      call io("detach","file09")
      call io("detach","file12")
      call io("detach","file13")
      call io("detach","file14")
      call io("detach","file16")
      call io("detach","file17")
      call dl("masking.tmp")
      call dl("minc_data.tmp")
      call dl("sub_grids.tmp")
      call dl("regional.tmp")
      call dl("whole.tmp")
      call io("detach","error_output")
      call io("attach","error_output","syn_","user_i/o")
      return
      end
```