AR-001-683

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

ELECTRONICS RESEARCH LABORATORY

TECHNICAL REPORT
ERL-0085-TR

A PROCEDURE FOR PRODUCING A TERRAIN ELEVATION GRID FROM MAP CONTOUR DATA

M.J. Dowling

S U M M A R Y

A common method of representing the shape of the terrain, for use in various simulations or digital computer assisted studies, is to record the height of the terrain at regular intervals over a 2-dimensional grid and store this information in the computer. Manual interpolation of contour data from survey maps, to obtain the data for the grid, is a laborious and error-prone process.

This report describes an improved procedure for producing the terrain height grid. It involves a sonic digitizer to record a set of (X, Y) coordinates along the contour lines shown on the survey map. This information is stored in a disk file on a computer, and a two-stage cubic spline interpolation procedure is used to determine the terrain height at the intersection points of the nominated grid. The set of computer programs developed to implement the procedure with minimal storage requirements is described and listed.

POSTAL ADDRESS: Chief Superintendent, Electronics Research Laboratory, Box 2151, G.P.O., Adelaide, South Australia, 5001.

TABLE OF CONTENTS

LIST OF APPENDICES

LIST OF FIGURES

1. INTRODUCTION

There are numerous applications in which terrain must be modelled for use in a digital computer program. In order to simplify the programming, the terrain heights are usually specified at the intersections of a lattice of regularly spaced, rectangular grid lines. Reading the height at each grid point directly from a map is a slow, tedious, and error-prone process. The improved procedure outlined below uses a sonic digitizer to record (X, Y) coordinates at regular intervals as a pen is moved along a contour line. The points at which each contour crosses the grid lines are found by normal linear interpolation, and, from this data base, bicubic spline interpolation is used to obtain the height at each grid point.

Five steps are involved in the procedure, namely :

(1) Record a series of (X, Y) coordinates for each contour on paper tape and transfer to disk.

(2) Find the points at which the contour lines intersect the lattice of regularly spaced rectangular grid lines. These will be referred to as crossing points. For each intersection of a contour line with a horizontal grid line (row) store the X-value, the contour height, and the row number. For each intersection of a contour line with a vertical grid line (column) store the Y-value, the contour height, and the column number.

(3) Sort the two sets of crossing points into increasing row/column order, and increasing X/Y value within each row/column.

(4) For each grid point on the map boundary, read the terrain height, and the slope perpendicular to the map boundary. The slopes at the map corner points are not required.

(5) For each grid point within the map boundary, fit one cubic spline to the crossing points along the row (see figure 1), and another cubic spline to the crossing points along the column. A cubic spline is defined by a set of coefficients for a continuous cubic function within each interval between successive crossing points, such that the functions on either side of a crossing point each give the same height, and the same first and second derivatives, at that point. The two splines are used to calculate the slope at the closest crossing points on either side of the grid point, along both the row and the column. Thus the height and slope are known at four crossing points surrounding the grid point, and the bicubic spline interpolation method of ref. 1 can be used. It shows that there are four unique cubic functions which connect these crossing points with the grid point, and give :

   (a) the required heights and slopes at the crossing points,
   (b) the same height at the grid point, and
   (c) a continuous slope at the grid point along both the row and the column.

   The coefficients of these functions are calculated and these provide the information for the terrain height at the grid point.

In the following sections, the method will be described in detail, including a description of the computer programs. The major difficulty involved in programming the above procedure lies in constraining the storage requirements since for a large map the combined total of grid points and crossing points may exceed the capacity of all but the largest computers, and, as will be shown, not all of these points need to be stored at once. The programs have been designed to minimize storage requirements whilst maintaining efficiency.

2. THE MAP DIGITIZATION PROCESS

The initial source for the data is assumed to be a survey map of suitable scale which displays, as well as other information, terrain height contour lines within the area of interest. The first step involves digitization of the contour line data, and this is performed with the aid of a GRAF/PEN model GP-3/NT-301 sonic digitizer (Science Accessories Corporation) linked to a FACIT 4070 paper tape punch. The following procedure is involved.

Firstly, the survey map of the required area is attached to the tablet of the sonic digitizer, as shown in figure 2. When the stylus of the pen is depressed a hypersonic impulse is generated at the point of the stylus, and two linear sensor microphones measure the X and Y coordinates (X_D, Y_D),and record the values on paper tape. Initially, the four corners of the map are digitized, in the order bottom right, bottom left, top left, top right. The first program (Appendix I) uses this information to calculate firstly the scale of the map relative to the digitizer data, and secondly the angle of the map relative to the digitizer, because the map may not be placed exactly square on the tablet.

Along the bottom of the tablet a scale has been drawn approximately along the line $Y_D = 0$, and with the X_D value shown in millimetres. This line can be used to indicate the start of a new contour line, and the height of that contour. To do this, the map is attached to the tablet such that the bottom is at least 8 mm above the scale line. The digitizer records coordinates in tenths of a millimetre. If the new contour height is 350 m for example, then the pen is depressed at the 350 mm point on the scale line , and the digitizer records the coordinates (3500, 0), approximately. Thus a Y-coordinate of less than 80 flags the start of a new contour line, and the contour height is given (in Fortran 4 notation) by :

$$\text{Contour Height} = IC*IFIX\left(\frac{IX+5*IC}{10*IC}\right)$$

where IC is the contour interval in metres, and IX is the X-coordinate in tenths of a millimetre. The pen must be placed within IC/2 mm of the required X-value for the correct contour height to be obtained.

If at any time the operator realises that an incorrect point has been entered, the pen is moved to the top right hand corner of the tablet and depressed at that point. When the data have subsequently been loaded from the paper tape onto disk file, the data points are scanned, and when such a reading is encountered the preceding point and the reading itself are deleted.

To operate the digitizer, the operator moves the pen along the appropriate contour at roughly constant speed, and the digitizer records the (X, Y) coordinates (in tenths of a millimetre) on paper tape at regular time intervals. The time interval is set by the operator according to the space between the contours and/or the intricacy of the contour (see also Section 3). The process is repeated until all contour lines on the map have been traced. The data are then transferred from the paper tape to disk. Contours may be read from a number of detailed maps, and the data for each of these maps combined into one composite map, as shown in figure 3. In the figure, each square is an individual map, and the lines shown are the first and last gridlines (both northings and eastings) which are fully enclosed within that map.

The cubic spline process requires the height to be specified at all grid points on the boundary of the map. The slope perpendicular to the map boundary must also be specified at the same points, excluding the corner

points. The terrain height at each grid point is estimated directly from the map. The slope at the grid point is approximated from the terrain heights at the grid point and at a point located one half grid square inside the boundary.

3. INTERSECTION OF CONTOUR LINES AND GRID LINES

The digitizing process represents each contour line as a series of (X, Y) coordinates in tenths of a millimetre. The next step is to calculate the points at which the contour lines intersect the grid lines of a regularly spaced rectangular lattice. These points will be called crossing points. The (X, Y) coordinates along the contour must be sufficiently closely spaced such that, if successive points span a grid line, the crossing point can be obtained with sufficient accuracy by linear interpolation. The latter procedure is carried out in the first program (see Appendix I) in the manner described below.

Firstly, digitizer coordinates in tenths of a millimetre are translated into map coordinates in units appropriate to the given map. Assume, for example, that the latter are in km.
If we let

(X_{MBL}, Y_{MBL}) be the coords. of the bottom L.H. corner of the map (km),

(X_{DBL}, Y_{DBL}) be the coords. of the digitizer data (0.1 mm),

THETA the angle of the map relative to the digitizer,

SCALE the scale of the map relative to the digitizer data (km per 0.1 mm),

(X_D, Y_D) an arbitrary digitizer point (0.1 mm),

(X_M, Y_M) the corresponding map location (km),

then if

$$XX = (X_D - X_{DBL}) * SCALE, \quad YY = (Y_D - Y_{DBL}) * SCALE,$$

$$X_M = XX * COS(THETA) + YY * SIN(THETA) + X_{MBL} \qquad (1)$$

$$Y_M = -XX * SIN(THETA) + YY * COS(THETA) + Y_{MBL} \qquad (2)$$

Also the nearest column to the left of point (X_M, Y_M) is given by :

$$IC = \frac{X_M - X_C + DX}{DX}$$

where (X_C, Y_C) is the location of the bottom left hand corner of the composite map, and DX is the spacing of the vertical grid lines (km). Thus the column, $X_M = X_C$, is column number 1. Similarly, the nearest row below the point (X_M, Y_M) is given by :

$$IR = \frac{Y_M - Y_C + DY}{DY}$$

where DY is the spacing of the horizontal grid lines, and the row $Y_M = Y_C$, is row number 1.

Now, the points within each contour line are considered in turn. If the value of IC is different for successive contour points (X1, Y1) and (X2, Y2), then the points span a column. The crossing point (X, Y) is given by :

$$X = FLOAT(IC) * DX, \quad Y = Y1 + (X - X1) * \frac{Y2 - Y1}{X2 - X1}$$

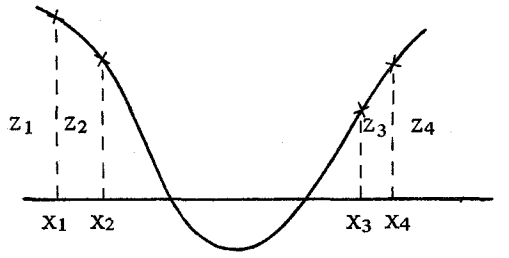
where IC is the larger of the two IC values. The Y-value, the contour height and the column number are written to a dataset containing column crossings. If the value of IR is different for these contour points, then the points span a row and the crossing point (X, Y) is given by :

$$Y = FLOAT(IR) * DY, \quad X = X1 + (Y - Y1) * \frac{X2 - X1}{Y2 - Y1}$$

where IR is the larger of the IR values. The X-value, the contour height, and the row number are written to another dataset containing row crossings. The process is repeated for all contours on the map, and for all maps within the composite map.

The crossing points in each dataset must now be sorted into order of increasing row/column number, and within each row/column into order of increasing X/Y respectively. Most computer installations have standard sort routines which will perform this type of task with maximum efficiency. The SORT/MERGE package supplied for the IBM 370/3033 sorted 5137 row crossings in 0.36 s of CPU time, and 5969 column crossings in 0.4 s of CPU time. The total number of row crossings, and the total number of column crossings were determined by a second program (Appendix II), which enabled the SORT to be performed with maximum efficiency.

If there is a large gap between consecutive crossing points along a row or column, the interpolating cubic function may take the shape shown.



The height calculated at a grid point in the middle of the gap will be incorrect. Physically, the large gap implies a gradual slope. Therefore, the program in Appendix III was written to detect consecutive crossing points of a row or column which are more than one grid spacing apart, and insert extra points at equal intervals of at most one grid spacing, with linearly interpolated height values.

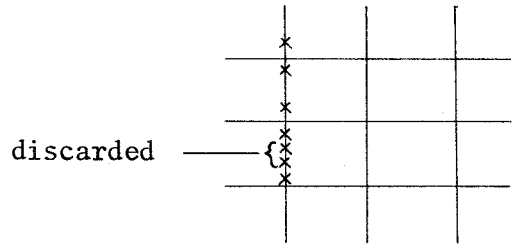
Similarly, if two crossing points are very close together, then the same situation can arise. In this case, the program detects consecutive crossing points of a row or column which are less than 0.5 mm apart (on the digitizer), and combines the two points into a single point with the average of the two heights at the average X or Y value.

The second program must now be run again with the new sets of row and column crossings, to determine the maximum number of crossings of any row and of any column, which will enable reduced array dimensions in the fourth and final program.

A block diagram of the programs and datasets involved in the whole procedure is shown in figure 4.

4. CUBIC SPLINE INTERPOLATION

Along each column there are now a number of crossing points, sorted into increasing Y-order. If more than two crossing points occur between consecutive rows, it is assumed that only those closest to the rows need be considered since the additional points would have little effect upon the interpolated height at the grid point. Hence the additional points are discarded (see figure below). This reduces the number of crossing points which need to be stored and fitted by the cubic spline.



The next program (see Appendix IV) reads all the column crossing points and stores the Y-value and the height in two two-dimensional arrays CCROSS and CHIGHT respectively : the first dimension is the number of the crossing points in that column and the second dimension is the column number. If there were two crossing points between every row in a particular column, then these arrays would have to be dimensioned (2*NROWS,NCOLS), where NROWS is the number of rows in the composite map and NCOLS is the number of columns in the composite map. However, normally the maximum number of crossing points of any column, MAX, (output from second program - see Section 3 or Appendix II) is much less than 2*NROWS, and the first dimension of the two arrays is reduced to MAX + 2. The number of crossing points along each column is stored in array NCC(NCOLS). The Y-value and height for the end points of each column are inserted into the arrays CCROSS and CHIGHT, and the slopes at the bottom and top are stored in arrays SLOPC1 (NCOLS) and SLOPCN(NCOLS) respectively. All the column information which is required for the cubic spline process has now been stored.

For each successive row, IR, the output array Z(NCOLS) which will contain terrain heights at the grid points along the row, is initialized to zero: grid points outside the composite map will retain this value. If this row is the first or the last row of the composite map, the array Z is set from the array CHIGHT and then written directly onto the output dataset; crossing point data for the first row are read but not used. For an intermediate row, the program reads the crossing points along the row into the arrays RCROSS and RHIGHT. If there are more than two crossing points between successive columns, only the crossing points nearest to each column are kept, and any in between are discarded. The variable NCROSS stores the number of crossing points along the row IR. The height and slope at the start and end of the row are read from file 4, the heights being inserted into the first and last elements of array RHIGHT, while the slopes are stored as SLOPR1 and SLOPRN respectively. RCROSS(1) is simply XA, the X-value at the start of this row,

and RCROSS(NCROSS) is XB, the X-value at the end of the row.

It should be noted that the arrays RCROSS, RHIGHT, and Z are all singly dimensioned, since it is possible to process one row at a time. The maximum dimension of arrays RCROSS and RHIGHT would be 2*NCOLS, but normally the dimension is MAX + 2, where in this case MAX is the maximum number of crossing points along any row (see Section 3).

There are an infinite number of cubic splines which interpolate the set of points {(RCROSS(i), RHIGHT(i)), i = 1, NCROSS }. However, by setting four boundary condition parameters (see ref. 2), a unique cubic spline is obtained which has the required slopes at the start and at the end of the row. The required parameters, BPAR, are :

$$BPAR(1) = 1.0,$$

$$BPAR(2) = \frac{6.}{RCROSS(2) - RCROSS(1)} \left[\frac{RHIGHT(2) - RHIGHT(1)}{RCROSS(2) - RCROSS(1)} - SLOPR1 \right],$$

$$BPAR(3) = 1.0,$$

$$BPAR(4) = \frac{6.}{RCROSS(NCROSS)-RCROSS(NCROSS-1)} \left[SLOPRN - \frac{RHIGHT(NCROSS)-RHIGHT(NCROSS-1)}{RCROSS(NCROSS)-RCROSS(NCROSS-1)} \right].$$

Subroutine ICSICU (Ref. 2) returns the cubic spline coefficients for the row IR in the NCROSS-1 by 3 array RCOEFF. The terrain height given by the spline approximation at a point T in the interval :

$$RCROSS(I) \leqq T < RCROSS(I+1)$$

is :

$$S(T) = RCOEFF(I, 3)*D^3 + RCOEFF(I, 2)*D^2 + RCOEFF(I, 1)*D + RHIGHT(I),$$

$$D = T - RCROSS(I).$$

The height and first derivative at T = RCROSS(I) are thus RHIGHT(I) and RCOEFF(I, 1) respectively. This information is that required by the bicubic spline interpolation process.

The first column in this row (IR) is L1, and the last column is L2: the terrain height at these points is RHIGHT(1) and RHIGHT(NCROSS) respectively. Also, for the intervening columns, if this row is the first or last of one of the maps the height is obtained from the array CHIGHT. Otherwise bicubic spline interpolation must be used to obtain the height at the row/column intersection point. Let the current column be IC, then the X-value of this column is FLOAT(IC-1) * DX, and the crossing point number LR (initially set to 2 for each row) is incremented until :

$$RCROSS\ (LR - 1) < X < RCROSS\ (LR).$$

The bicubic spline interpolation process (subroutine BICUBE) requires double

precision on the IBM 370/3033, so the following input variables must be double precision :

$$X1 = RCROSS\ (LR - 1)\ , \quad Z1 = RHIGHT\ (LR - 1),$$

$$X2 = RCROSS\ (LR)\ , \quad Z2 = RHIGHT\ (LR)\ ,$$

$$Z1D = RCOEFF\ (LR - 1,\ 1), \quad Z2D = RCOEFF\ (LR,\ 1),$$

$$XG = X.$$

The height and slope at the closest row crossing points on either side of the grid point are now available for the bicubic spline process.

The corresponding column information is obtained as follows. Subroutine ICSICU is called again to fit a cubic spline to the crossing points along the column IC, namely

$$\{(CCROSS\ (i,IC),\ CHIGHT\ (i,\ IC)),\ i = 1,\ NCC\ (IC)\}.$$

The spline has the slope SLOPC1(IC) at the bottom of the column, and SLOPCN(IC) at the top of the column, if the boundary condition parameters BPAR(2) and BPAR(4) are set as follows :

$$BPAR(2) = \frac{6.}{CCROSS(2,IC)-CCROSS(1,IC)}\left[\frac{CHIGHT(2,IC)-CHIGHT(1,IC)}{CCROSS(2,IC)-CCROSS(1,IC)}-SLOPC1(IC)\right],$$

$$BPAR(4) = \frac{6.}{CCROSS(N,IC)-CCROSS(N-1,IC)}\left[SLOPCN(IC)-\frac{CHIGHT(N,IC)-CHIGHT(N-1,IC)}{CCROSS(N,IC)-CCROSS(N-1,IC)}\right],$$

$$N = NCC(IC).$$

The spline coefficients are stored in array CCOEFF. The Y-value of row IR is FLOAT(IR-1) * DY; the crossing point number of the column, LC(IC) (initially set to 2 in a DATA statement), is incremented until :

$$CCROSS\ (LC(IC) - 1,\ IC)\ < Y < CCROSS\ (LC(IC),\ IC),$$

and then the following double precision variables can be fixed :

$$Y1 = CCROSS\ (LC(IC) - 1,\ IC), \quad Z3 = CHIGHT\ (LC(IC) - 1,\ IC),$$

$$Y2 = CCROSS\ (LC(IC),\ IC)\ , \quad Z4 = CHIGHT\ (LC(IC),\ IC),$$

$$Z3D = CCOEFF\ (LC(IC) - 1,\ 1), \quad Z4D = CCOEFF\ (LC(IC),\ 1),$$

$$YG = Y.$$

Thus the height and slope at the closest column crossing points on either side of the grid point are also available for the bicubic spline process.

5. BICUBIC SPLINE INTERPOLATION

The row data and the column data, shown in figure 5, are passed through labelled common to subroutine BICUBE (see Appendix V). The subroutine performs the bicubic spline interpolation and returns the value ZG, the height at the point (XG, YG) which is the intersection of row IR and column IC, and also the value ZGSD. The interpolation process calculates four values of height, which are averaged to give ZG, and ZGSD is the standard deviation of the four values. If the computations could be performed with perfect accuracy, then all four values of height would be identical; ZGSD reflects the inaccuracies of the floating point computations. The height ZG is then stored in array Z, and the above process is repeated for all the grid points in this row.

When the row has been completed, the array Z is written onto the output dataset, several variables are reset, and the next row is commenced. When all rows have been processed, the program terminates. For a sample map of a reasonably flat area, the program calculated the height of 17460 grid points in less than 17 s CPU time, on an IBM 370/3033 computer using the Fortran H (option 2) optimizing compiler. The average execution time per point was thus a little less than 0.001 s. The program execution time using the normal G compiler was almost 25 s.

A simple step-by-step procedure for transferring data from the digitizer onto disk, and for implementing the bicubic spline interpolation process, is included as Appendix VI. The command procedures and Job Control Language files referred to are not listed in this report, but are available upon request.

REFERENCES

No.	Author	Title
1	Fogg, D.A.B.	"Bicubic Spline Interpolation from Data in Contour Form" WRE Technical Report 1837(A)
2		International Mathematical and Statistical Libraries Inc.

APPENDIX I

PROGRAM TO CALCULATE GRIDLINE CROSSING POINTS FROM DIGITISED CONTOUR DATA

```
C*************************************************************************
C*                                                                       *
C*    AUTHOR: M.J.DOWLING                                                *
C*    DATE: 1/12/78                                                      *
C*    LATEST REVISION: 3/7/79                                            *
C*                                                                       *
C*    SUMMARY:                                                           *
C*      THE DATA ON FILE 5 ARE USED TO SET UP A RECTANGULAR GRID, AND    *
C*    THEN THE CORNERS ARE READ FROM THE CONTOUR DATA. THE SCALE OF      *
C*    THE MAP RELATIVE TO THE READER DATA IS CALCULATED, AND THE         *
C*    ROTATION OF THE MAP ON THE TABLET. THIS ENABLES READER COORDS      *
C*    TO BE TRANSLATED INTO MAP COORDINATES.                             *
C*      IF CONSECUTIVE POINTS IN A CONTOUR SPAN A ROW THEN THE X-        *
C*    VALUE OF THE CROSSING POINT (LINEAR INTERPOLATION), THE HEIGHT,    *
C*    AND THE ROW NUMBER (INTEGER*2) ARE STORED IN SPLINE.ROW.DATA.      *
C*    IF CONSECUTIVE POINTS SPAN A COLUMN THEN THE Y-VALUE, THE          *
C*    HEIGHT AND THE COLUMN NUMBER (INTEGER*2) ARE STORED IN SPLINE.     *
C*    COL.DATA.  DISPOSITION OF MOD FOR THE OUTPUT DATASETS ENABLES      *
C*    THIS STEP TO BE RUN SEVERAL TIMES WITH DIFFERENT MAPS; THE NEW     *
C*    CROSSING POINTS WILL BE ADDED TO THE EXISTING SETS.                *
C*                                                                       *
C*                                                                       *
C*    INPUT: FILE 2 - MJD.BODESERT.CONTOUR.DATA(MEMBERS)                 *
C*                    CONTOUR LINE DATA FROM THE XY READER (FORMAT       *
C*    XXXXYYYY). MUST START WITH FOUR CORNER POINTS AND THEREAFTER       *
C*    THE START OF A NEW CONTOUR LINE IS SIGNALLED BY A Y-VALUE OF       *
C*    LESS THAN 80 TENTHS OF A MM, AND THE CORRESPONDING X-VALUE         *
C*    THE CONTOUR HEIGHT. FORMAT(2I4).                                   *
C*                                                                       *
C*           FILE 5 - MJD.BODESERT.MAP.CORNERS.DATA(MEMBERS)             *
C*                    MAP COORDINATES OF THE FOUR CORNERS, THE GRID      *
C*    SPACING (KM), AND MAP COORDINATES OF THE BL CORNER OF THE          *
C*    COMPOSITE MAP. FORMAT(9F8.0).                                      *
C*                                                                       *
C*                                                                       *
C*    OUTPUT: FILE 3 - MJD.SPLINE.COL.DATA  COLUMN CROSSING POINTS       *
C*            FILE 4 - MJD.SPLINE.ROW.DATA  ROW CROSSING POINTS          *
C*                                                                       *
C*    EXECUTION: MJD.SPLINE.CNTL(GRIDXRUN)                               *
C*                                                                       *
C*                                                                       *
C*                                                                       *
C*************************************************************************
C
C
      DIMENSION XMAP(4),YMAP(4),XPPT(4),YPPT(4)
      DATA ISTOP/0/,NTOTR/0/,NTOTC/0/
      INTEGER*2 IROW,ICOL
C
C  READ CORNER POINTS FROM MAP( (X,Y) PAIRS IN THE ORDER BOTTOM RIGHT,
C    BOTTOM LEFT,TOP LEFT,TOP RIGHT) AND GRID SPACING(KILOMETRES)
C
```

```
      READ (5,10) (XMAP(I),YMAP(I),I=1,4),GRIDSP
   10 FORMAT(9F8.0)
C
C  READ BOTTOM LEFT HAND CORNER OF COMPOSITE MAP.  IF THERE IS ONLY ONE
C  MAP THEN RE-ENTER XMAP(2),YMAP(2)
C
      READ (5,10) XCOMP,YCOMP
      WRITE (6,20) XCOMP,YCOMP
   20 FORMAT('1BOTTOM LEFT HAND CORNER OF COMPOSITE MAP IS (',F8.2,1H,,
     * F8.2,1H) )
C  READ CORRESPONDING CORNER POINTS FROM READER DATA
          DO 30 I=1,4
   30     READ (2,40) XPPT(I),YPPT(I)
   40 FORMAT(2F4.0)
C  CALCULATE SCALE OF MAP RELATIVE TO READER DATA BY SUMMING THE
C  LENGTHS OF THE FOUR SIDES
      SUMMAP=SQRT( (XMAP(1)-XMAP(4))**2 + (YMAP(1)-YMAP(4))**2 )
      SUMPPT=SQRT( (XPPT(1)-XPPT(4))**2 + (YPPT(1)-YPPT(4))**2 )
          DO 50 I=1,3
          IP1=I+1
          SUMMAP=SUMMAP+SQRT((XMAP(IP1)-XMAP(I))**2+(YMAP(IP1)-YMAP(I))*
     *    *2)
   50     SUMPPT=SUMPPT+SQRT((XPPT(IP1)-XPPT(I))**2+(YPPT(IP1)-YPPT(I))*
     *    *2)
      SCALE=SUMMAP/SUMPPT
      WRITE (6,60) SCALE
   60 FORMAT(' READER DATA MUST BE SCALED BY THE FACTOR ',F9.4)
C  CALCULATE THETA, THE ROTATION OF THE MAP ON THE READER (POSITIVE
C  ANTICLOCKWISE)   BY COMPARING THE DIRECTIONS OF THE FOUR SIDES.
C  BEWARE THE SIDE FROM TOP LEFT TO TOP RIGHT (ANGLE APPROXIMATELY + OR
C    - PI)
      AP=ATAN2(YPPT(3)-YPPT(4),XPPT(3)-XPPT(4))
      IF (AP.LT.0.) AP=6.2832+AP
      AM=ATAN2(YMAP(3)-YMAP(4),XMAP(3)-XMAP(4))
      IF (AM.LT.0.) AM=6.2832+AM
      ANGERR=AP-AM
          DO 70 I=1,3
          IM1=I-1
          IF (I.EQ.1) IM1=4
          ANGERR=ANGERR+ATAN2(YPPT(IM1)-YPPT(I),XPPT(IM1)-XPPT(I))
     *                        -ATAN2(YMAP(IM1)-YMAP(I),XMAP(IM1)-XMAP(I))
   70     CONTINUE
      THETA=ANGERR/4.
      THEDEG=THETA*57.3
      WRITE (6,80) THEDEG
   80 FORMAT(' READER DATA MUST BE ROTATED THROUGH ',F8.4,' DEGREES')
      COSTH=COS(THETA)
      SINTH=SIN(THETA)
C  BOTTOM LEFT HAND CORNER OF MAP AND READER DATA
      XBLMAP=XMAP(2)
      YBLMAP=YMAP(2)
      XBLPPT=XPPT(2)
      YBLPPT=YPPT(2)
      WRITE (6,90) XBLMAP,YBLMAP,XBLPPT,YBLPPT
   90 FORMAT(' BOTTOM LEFT HAND CORNER OF MAP IS (',F8.2,1H,,F8.2,1H),/,
     * ' BOTTOM LEFT HAND CORNER OF READER DATA IS (',F8.2,1H,,F8.2,1H))
C  BOTTOM LEFT HAND CORNER OF THIS MAP RELATIVE TO BOTTOM LEFT HAND
```

```
C  CORNER OF COMPOSITE MAP
      XBLINC=XBLMAP-XCOMP
      YBLINC=YBLMAP-YCOMP
C  GRID LINE INCREMENTS(KM)
      DX=GRIDSP
      DY=DX
      WRITE (6,100) DX,DY
  100 FORMAT(' GRID LINE INCREMENTS; DX = ',F8.2,'   DY = ',F8.2)
C  READ CONTOUR HEIGHT FROM READER DATA
      READ (2,110) IX,IY
  110 FORMAT(2I4)
      IF (IY.LT.80) GO TO 130
      WRITE (6,120)
  120 FORMAT(' INITIAL CONTOUR HEIGHT IS MISSING FROM READER DATA')
      STOP
C
C  IX ON READER IS APPROXIMATELY 10 * CONTOUR HEIGHT
C
  130 ICONHT=((IX+50)/100)*10
      ZZ=ICONHT
C  ZERO NGR,THE NUMBER OF GRID LINE INTERSECTIONS OF THIS CONTOUR HEIGHT
      NGR=0
      NGC=0
C  READ FIRST CONTOUR POINT (X1,Y1)
      READ (2,40) XIN,YIN
      XX=(XIN-XBLPPT)*SCALE
      YY=(YIN-YBLPPT)*SCALE
      X1= XX*COSTH+YY*SINTH+XBLINC
      Y1=-XX*SINTH+YY*COSTH+YBLINC
      IX1=(X1+DX)/DX
      IY1=(Y1+DY)/DY
C  READ NEXT CONTOUR POINT (X2,Y2)
  140 READ (2,40,END=170) XIN,YIN
C  Y LESS THAN 80 IMPLIES START OF NEXT CONTOUR HEIGHT
      IF (YIN.LT.80.) GO TO 180
      XX=(XIN-XBLPPT)*SCALE
      YY=(YIN-YBLPPT)*SCALE
      X2= XX*COSTH+YY*SINTH+XBLINC
      Y2=-XX*SINTH+YY*COSTH+YBLINC
      IX2=(X2+DX)/DX
      IY2=(Y2+DY)/DY
C
      IF (IX2.EQ.IX1) GO TO 150
C
C  CROSS VERTICAL GRID LINE
C
C  SET X EQUAL TO GRID LINE VALUE
      IX=MIN0(IX1,IX2)
      XX=FLOAT(IX)*DX
C  LINEARLY INTERPOLATE TO GET Y-VALUE AT GRID LINE
      YY=Y1+(XX-X1)*(Y2-Y1)/(X2-X1)
      ICOL=IX
      WRITE (3) YY,ZZ,ICOL
      NGC=NGC+1
      NTOTC=NTOTC+1
C
  150 IF (IY2.EQ.IY1) GO TO 160
```

```
C
C  CROSS HORIZONTAL GRID LINE
C
C  SET Y EQUAL TO GRID LINE VALUE
      IY=MIN0(IY1,IY2)
      YY=FLOAT(IY)*DY
C  LINEARLY INTERPOLATE TO GET X-VALUE AT GRID LINE
      XX=X1+(YY-Y1)*(X2-X1)/(Y2-Y1)
      IROW=IY
      WRITE (4) XX,ZZ,IROW
      NGR=NGR+1
      NTOTR=NTOTR+1
C  NEXT CONTOUR POINT
  160 IX1=IX2
      IY1=IY2
      X1=X2
      Y1=Y2
      GO TO 140
C  END OF INPUT DATA - SET FLAG ISTOP
  170 ISTOP=1
C  END OF CONTOUR HEIGHT
  180 CONTINUE
      WRITE (6,190) ICONHT,NGR,NGC
  190 FORMAT(' CONTOUR HEIGHT = ',I4,' , NUMBER OF ROW CROSSINGS = ',I4,
     * ' , NUMBER OF COLUMN CROSSINGS = ',I4)
C  NEXT CONTOUR HEIGHT
      IX=XIN
      IF (ISTOP.EQ.0) GO TO 130
C  END OF INPUT DATA
      WRITE (6,200) NTOTR,NTOTC
  200 FORMAT(' TOTAL NUMBER OF ROW CROSSINGS =',I4,' , TOTAL NUMBER OF
     *COLUMN CROSSINGS =',I4)
      STOP
      END
```

APPENDIX II

PROGRAM TO LIST THE ROW AND COLUMN CROSSING POINTS

```
C*************************************************************************
C*                                                                       *
C*    AUTHOR: M.J.DOWLING                                                *
C*    DATE: 4/7/79                                                       *
C*    LATEST REVISION: 4/7/79                                            *
C*                                                                       *
C*    SUMMARY:                                                           *
C*       THE PROGRAM PROMPTS THE USER TO ENTER IPRINT.  IF THE REPLY     *
C*    IS ZERO, THE PROGRAM PRINTS THE TOTAL NUMBER OF ROW CROSSING       *
C*    POINTS AND OF COLUMN CROSSING POINTS, AND THE MAXIMUM NUMBER OF    *
C*    CROSSING POINTS ALONG ANY ROW AND ALONG ANY COLUMN.  IF THE        *
C*    REPLY IS ONE, THE PROGRAM WILL ALSO LIST ALL THE ROW AND           *
C*    COLUMN CROSSING POINTS.                                            *
C*                                                                       *
C*                                                                       *
C*    INPUT: FILE 2  MJD.SPLINE.COL.DATA                                 *
C*           FILE 3  MJD.SPLINE.ROW.DATA                                 *
C*           FILE 4  MJD.BODESERT.MAP.CORNERS.DATA                       *
C*                                                                       *
C*                                                                       *
C*    EXECUTION: MJD.SPLINE.CLIST(LISTTERM)  OUTPUT AT TERMINAL          *
C*               MJD.SPLINE.CLIST(LISTPRIN)  OUTPUT AT PRINTER           *
C*                                                                       *
C*                                                                       *
C*************************************************************************
C
C
      DIMENSION C(5),H(5),D(5)
      INTEGER*2 I
      WRITE (6,5)
    5 FORMAT(' ENTER IPRINT')
      READ (5,*) IPRINT
C
C  IPRINT  0: PRINT THE TOTAL NUMBER OF ROW OR COLUMN CROSSINGS, AND
C             THE MAXIMUM NUMBER OF CROSSINGS PER ROW OR COLUMN
C          1: PRINT ALSO A LIST OF THE ROW AND COLUMN CROSSINGS
C  INCX    ADDED TO THE ROW NUMBER TO GIVE THE NORTHING
C  INCY    ADDED TO THE COLUMN NUMBER TO GIVE THE EASTING
C  N       TOTAL NUMBER OF ROW OR COLUMN CROSSINGS
C  IROW    THE CURRENT ROW
C  NR      THE NUMBER OF CROSSINGS OF THE CURRENT ROW
C  MAX     THE MAXIMUM NUMBER OF CROSSINGS PER ROW OR COLUMN
C  ICOL    THE CURRENT COLUMN
C  NC      THE NUMBER OF CROSSINGS OF THE CURRENT COLUMN
C
      READ (4,10) GRIDSP,XCOMP,YCOMP
   10 FORMAT(64X,F8.0,/,2F8.0)
      IF (IPRINT.GT.0) WRITE (6,20) XCOMP,YCOMP
   20 FORMAT('1BOTTOM LEFT HAND CORNER OF COMPOSITE MAP IS (',F8.2,
     * 1H,,F8.2,1H) )
      INCX=IFIX(XCOMP)-1
      INCY=IFIX(YCOMP)-1
      DX=GRIDSP
```

```
      DY=GRIDSP
C
C  ROW CROSSINGS
C
      IF (IPRINT.GT.0) WRITE (6,30)
   30 FORMAT(//,8X,'ROW CROSSINGS',/,' EASTING HEIGHT NORTHING',
     * 4(3X,'EASTING HEIGHT NORTHING') )
      N=0
      IROW=1
      MAX=0
      NR=0
   40     DO 60 J=1,5
          READ (2,END=80) C(J),H(J),I
      II=I
          IF (II.EQ.IROW) GO TO 50
          IF (NR.GT.MAX) MAX=NR
          NR=0
          IROW=II
   50     NR=NR+1
          N=N+1
          C(J)=C(J)+XCOMP
      D(J)=FLOAT(II)*DY+YCOMP
   60     CONTINUE
      IF (IPRINT.GT.0) WRITE (6,70) (C(J),H(J),D(J),J=1,5)
   70 FORMAT(1X,F7.2,2X,F4.0,2X,F6.1,4(5X,F7.2,2X,F4.0,2X,F6.1) )
      GO TO 40
   80 JM1=J-1
      IF (JM1.GT.0 .AND. IPRINT.GT.0)WRITE(6,70)(C(J),H(J),D(J),J=1,JM1)
      WRITE (6,90) N
   90 FORMAT('0NUMBER OF ROW CROSSINGS =',I6)
      WRITE (6,95) MAX
   95 FORMAT(' MAXIMUM NUMBER OF CROSSINGS OF ANY ROW =',I5)
C
C  COLUMN CROSSINGS
C
      IF (IPRINT.GT.0) WRITE (6,100)
  100 FORMAT('1    COLUMN CROSSINGS',/,' NORTHING HEIGHT EASTING',
     * 4(3X,'NORTHING HEIGHT EASTING') )
      N=0
      ICOL=1
      MAX=0
      NC=0
  110     DO 130 J=1,5
          READ (3,END=140) C(J),H(J),I
      II=I
          IF (II.EQ.ICOL) GO TO 120
          IF (NC.GT.MAX) MAX=NC
          NC=0
          ICOL=II
  120     NC=NC+1
          N=N+1
          C(J)=C(J)+YCOMP
      D(J)=FLOAT(II)*DX+XCOMP
  130     CONTINUE
      IF (IPRINT.GT.0) WRITE (6,70) (C(J),H(J),D(J),J=1,5)
      GO TO 110
  140 JM1=J-1
```

```
      IF (JM1.GT.0 .AND. IPRINT.GT.0)WRITE(6,70)(C(J),H(J),D(J),J=1,JM1)
      WRITE (6,150) N
  150 FORMAT('ONUMBER OF COLUMN CROSSINGS =',I6)
      WRITE (6,160) MAX
  160 FORMAT(' MAXIMUM NUMBER OF CROSSINGS OF ANY COLUMN =',I5)
      STOP
      END
```

APPENDIX III

PROGRAM TO INSERT CROSSING POINTS OR COMBINE CROSSING POINTS WHENEVER NECESSARY

```
C*****************************************************************************
C*                                                                           *
C*    AUTHOR: M.DOWLING                                                      *
C*    DATE: 18/6/79                                                          *
C*    LATEST REVISION: 25/6/79 BY MJD                                        *
C*                                                                           *
C*    SUMMARY:  THE PROGRAM READS IN CROSSING POINTS OF ROWS/COLUMNS  *
C*    WHICH HAVE BEEN SORTED INTO INCREASING X/Y ORDER. IF SUCCESSIVE *
C*    POINTS IN THE SAME ROW/COLUMN ARE MORE THAN XSEP KM APART, THEN *
C*    EXTRA POINTS ARE INSERTED,AT EQUAL INTERVALS (OF APPROX XSEP KM)*
C*    WITH LINEARLY INTERPOLATED TERRAIN HEIGHTS.                     *
C*              IF SUCCESSIVE POINTS IN THE SAME ROW/COLUMN ARE WITHIN*
C*    25 METRES (0.5 MM ON XY READER) THEN THE TWO POINTS ARE COMBINED*
C*    SETTING THE AVERAGE OF THE TERRAIN HEIGHTS AT THE AVERAGE X/Y   *
C*    VALUE.                                                          *
C*                                                                    *
C*    INPUT:   FILE 2  MJD.SPLINE.ROW.DATA OR MJD SPLINE.COL.DATA     *
C*                     SORTED ROW OR COLUMN CROSSING POINTS           *
C*                                                                    *
C*    OUTPUT:  FILE 2  MJD.SPLINE.ROW.DATA OR MJD.SPLINE.COL.DATA     *
C*                                                                    *
C*                                                                    *
C*    EXECUTION:  MJD.SPLINE.CLIST(EXTRA)                             *
C*                                                                    *
C*****************************************************************************
C
      DIMENSION A(20000),B(20000)
      INTEGER*2 LINE,LINEP,IRC(20000)
      DATA XSEP/0.5/
C
C  READ ALL POINTS, THEN REWIND FILE
C
      N=0
   10 N=N+1
      READ (2,END=30) A(N),B(N),IRC(N)
      IF (N.LT.20000) GO TO 10
      WRITE (6,20)
   20 FORMAT(' MORE THAN 20000 CROSSING POINTS - INCREASE DIMENSIONS')
      STOP
   30 REWIND 2
      NPOINT=N-1
C
      CROSS=A(1)
      Z=B(1)
      LINE=IRC(1)
      N=0
          DO 80 L=2,NPOINT
          CROSSP=CROSS
          ZP=Z
          LINEP=LINE
          CROSS=A(L)
          Z=B(L)
```

```
          LINE=IRC(L)
          IF (LINE.EQ.LINEP) GO TO 40
          WRITE (2) CROSSP,ZP,LINEP
          N=N+1
          GO TO 80
C
C  SAME LINE. DOES DISTANCE BETWEEN CROSSING POINTS EXCEED XSEP?
C
   40     DIST=CROSS-CROSSP
          IDIST=DIST/XSEP
          IF (IDIST.EQ.0) GO TO 60
C
          DC=DIST/FLOAT(IDIST+1)
          DZ=(Z-ZP)/FLOAT(IDIST+1)
C
C  GENERATE IDIST DUMMY CROSSING POINTS
C
          WRITE (2) CROSSP,ZP,LINEP
          N=N+1
              DO 50 I=1,IDIST
              XCROSS=CROSSP+FLOAT(I)*DC
              XZ=ZP+FLOAT(I)*DZ
              WRITE (2) XCROSS,XZ,LINE
              N=N+1
   50         CONTINUE
          GO TO 80
C
C ARE CROSSING POINTS LESS THAN 25 METRES APART?
C
   60     IF (DIST.LE.0.025) GO TO 70
          WRITE (2) CROSSP,ZP,LINEP
          N=N+1
          GO TO 80
C
C  COMBINE THE TWO POINTS
C
   70     CROSS=(CROSSP+CROSS)/2.0
          Z=(ZP+Z)/2.0
   80     CONTINUE
C
C  END OF DATA
C
   90 WRITE (6,100) N
  100 FORMAT('0NUMBER OF CROSSING POINTS =',I6)
      STOP
      END
```

APPENDIX IV

PROGRAM TO CALCULATE THE TERRAIN HEIGHT AT THE RECTANGULAR GRID POINTS

```
C
C
C
C
C*************************************************************************
C*                                                                       *
C*    AUTHOR: M.J.DOWLING                                                *
C*    DATE: 11/12/78                                                     *
C*    LATEST REVISION: 4/7/79                                            *
C*                                                                       *
C*    SUMMARY:                                                           *
C*       THE PROGRAM READS IN AND STORES ALL THE COLUMN CROSSING         *
C*    POINTS, EXCEPT IF THERE ARE MORE THAN TWO IN A GRID SPACE THEN     *
C*    ONLY THE TWO CROSSING POINTS CLOSEST TO THE TWO GRID POINTS ARE    *
C*    STORED.  FOR EACH COLUMN, A CUBIC SPLINE IS FITTED TO THE          *
C*    CROSSING POINTS, WHICH  THEN SPECIFIES THE SLOPE AT EACH           *
C*    CROSSING POINT.  EACH ROW OF THE MAP IS THEN PROCESSED IN TURN.    *
C*    THE CROSSING POINTS ALONG THE ROW ARE READ, STORED, AND FITTED     *
C*    BY A CUBIC SPLINE.  FOR EACH GRID POINT ALONG THE ROW, THE         *
C*    HEIGHT AND SLOPE OF THE TWO NEAREST ROW CROSSING POINTS AND OF     *
C*    THE TWO NEAREST COLUMN CROSSING POINTS ARE PASSED TO A BICUBIC     *
C*    SPLINE INTERPOLATION SUBROUTINE, WHICH CALCULATES THE HEIGHT AT    *
C*    THE GRID POINT.  THE TERRAIN HEIGHT VALUES FOR THE ROW ARE         *
C*    STORED IN THE OUTPUT DATASET, AND THE NEXT ROW COMMENCED.          *
C*                                                                       *
C*                                                                       *
C*    INPUT: FILE 2  MJD.SPLINE.COL.DATA                                 *
C*                   SORTED COLUMN CROSSING POINTS, IN INCREASING        *
C*    COLUMN ORDER, AND INCREASING Y-VALUE WITHIN EACH COLUMN.           *
C*                                                                       *
C*           FILE 1  MJD.SPLINE.ROW.DATA                                 *
C*                   SORTED ROW CROSSING POINTS, IN INCREASING ROW       *
C*    ORDER, AND INCREASING X-VALUE WITHIN EACH ROW.                     *
C*
C*           FILE 4  HEIGHT AND SLOPE AT THE BOTTOM AND TOP OF EACH
C*                   COLUMN
C*                   HEIGHT AND SLOPE AT THE LEFT AND RIGHT OF EACH
C*                   ROW
C*                                                                       *
C*                                                                       *
C*    OUTPUT: FILE 3  MJD.SPLINE.Z.DATA                                  *
C*                    TERRAIN HEIGHT AT THE GRID POINTS, STORED          *
C*    ONE ROW AT A TIME.                                                 *
C*                                                                       *
C*                                                                       *
C*    EXECUTION: MJD.SPLINE.CNTL(Z)                                      *
C*                                                                       *
C*                                                                       *
C*************************************************************************
C
      DIMENSION CCROSS(266,97),CHIGHT(266,97),NCC(97),LC(97),
     * RCROSS(170),RHIGHT(170),CCOEFF(265,3),RCOEFF(169,3),BPAR(4),
```

```
     * Z(97),SLOPC1(97),SLOPCN(97)
       INTEGER*2 ICOL,ICOLP,IROW,IROWP
       INTEGER*2 NRSTRT(97),NRSTOP(97),NCSTRT(145),NCSTOP(145)
C
C  CCROSS  Y-VALUE AND HEIGHT FOR UP TO 2*NROWS CROSSINGS PER COLUMN,
C  CHIGHT    AND FOR NCOLS COLUMNS. MAXIMUM DIMENSIONS (2*NROWS,NCOLS),
C            OR THE FIRST DIMENSION MAY BE ACTUAL MAXIMUM NUMBER OF
C            CROSSINGS PER COLUMN (IF SMALLER)
C  NCC     NUMBER OF CONTOUR LINE CROSSINGS ALONG EACH COLUMN
C  LC      THE CURRENT CROSSING IN EACH COLUMN
C  RCROSS  X-VALUE AND HEIGHT FOR UP TO2*NCOLS CROSSINGS PER ROW.
C  RHIGHT    MAXIMUM DIMENSION (2*NCOLS) OR THE ACTUAL MAXIMUM NUMBER
C            OF CROSSINGS PER ROW (IF SMALLER)
C  CCOEFF  THE COEFFICIENTS OF THE CUBIC SPLINES FITTED TO THE COLUMNS
C  RCOEFF    AND ROWS RESPECTIVELY
C  BPAR    THE END CONDITIONS FOR THE SPLINE
C
C  Z (OUTPUT) HEIGHTS AT GRID LINE INTERSECTIONS: Z(I) IN ROW IR IS THE
C           HEIGHT AT THE POINT (FLOAT(I)*DX,FLOAT(IR)*DY)
C  SLOPC1  THE SLOPE AT THE BOTTOM OF EACH COLUMN
C  SLOPCN  THE SLOPE AT THE TOP OF EACH COLUMN
C
C                                                                  --
C  NRSTRT  START ROW FOR EACH COLUMN  CONTOURS MAY BE READ FROM |  |
C  NRSTOP   STOP ROW FOR EACH COLUMN  A NUMBER OF DETAILED MAPS |  |
C  NCSTRT  START COLUMN FOR EACH ROW  AND COMBINED INTO ONE COM  -----
C  NCSTOP   STOP COLUMN FOR EACH ROW  POSITE MAP WITH THIS SHAPE   |  |
C                                                                 |  |
C                                                                  --
C
      DATA NRSTRT/97*1/,NRSTOP/97*145/
      DATA NCSTRT/145*1/,NCSTOP/145*97/
      DATA DX,DY,ICDIM,IRDIM,NROWS,NCOLS /0.5,0.5,265,169,145,97/
      DATA LC/97*2/,ISTOP/0/,SLOPC1/97*0.0/,SLOPCN/97*0.0/
C
C  NROWS   NUMBER OF HORIZONTAL GRID LINES (ROWS) IN THE COMPOSITE MAP
C            DY LE Y LE FLOAT(NROWS)*DY
C  NCOLS   NUMBER OF VERTICAL GRID LINES (COLUMNS) IN THE COMPOSITE MAP
C            DX LE X LE FLOAT(NCOLS)*DX
C  DX      DISTANCE BETWEEN VERTICAL GRID LINES
C  DY      DISTANCE BETWEEN HORIZONTAL GRID LINES
C  ICDIM IS FIRST DIMENSION OF ARRAY CCOEFF (2*NROWS-1)
C  IRDIM  "   "       "      "   "   RCOEFF (2*NCOLS-1)
C
      REAL*8     X1,Y1,X2,Y2,XG,YG,Z1,Z1D,Z2,Z2D,Z3,Z3D,Z4,Z4D,ZG,ZGSD
      COMMON/BSP/X1,Y1,X2,Y2,XG,YG,Z1,Z1D,Z2,Z2D,Z3,Z3D,Z4,Z4D,ZG,ZGSD
C
C  READ ALL COLUMN CROSSINGS, ENSURING NO MORE THAN TWO BETWEEN EACH
C   ROW. START AT CROSSING POINT NUMBER TWO; THE FIRST IS THE BOUNDARY
C   VALUE OF HEIGHT WHICH WILL BE SUPPLIED SEPARATELY. SIMILARLY THE
C   LAST CROSSING POINT IS THE OTHER BOUNDARY VALUE, ALSO SUPPLIED
C   SEPARATELY.
C
      I=2
      READ (2) CCROSS(I,1),CHIGHT(I,1),ICOLP
C
          DO 90 IC=1,NCOLS
```

```
C  START AND STOP Y-VALUES FOR THIS COLUMN
          YA=NRSTRT(IC)*DY
          YB=NRSTOP(IC)*DY
   10     IF (CCROSS(I,IC).GE.YA) GO TO 20
C  FIRST CROSSING POINT IS OUTSIDE BOUNDARY
          READ (2) CCROSS(I,IC),CHIGHT(I,IC),ICOL
          GO TO 10
C  ICRP FIXES THE GRID SQUARE WHICH CONTAINS THIS CROSSING POINT
   20     ICRP=CCROSS(I,IC)/DY
C  READ NEXT CROSSING POINT
   30     I=I+1
          READ (2,END=80) CCROSS(I,IC),CHIGHT(I,IC),ICOL
          IF (ICOL.NE.ICOLP) GO TO 70
C  IN THE SAME COLUMN
          IF (CCROSS(I,IC).GT.YB) GO TO 50
C  AND WITHIN BOUNDARY
          ICR=CCROSS(I,IC)/DY
          IF (ICR.EQ.ICRP) GO TO 40
C  BUT DIFFERENT GRID SQUARE
          ICRP=ICR
          GO TO 30
C  SAME GRID SQUARE, ARE THERE ANY MORE?
   40     I=I+1
   50     READ (2,END=80) CCROSS(I,IC),CHIGHT(I,IC),ICOL
          IF (ICOL.NE.ICOLP) GO TO 70
C  IN THE SAME COLUMN
          IF (CCROSS(I,IC).GT.YB) GO TO 50
C  AND WITHIN BOUNDARY
          ICR=CCROSS(I,IC)/DY
          IF (ICR.EQ.ICRP) GO TO 60
C  BUT DIFFERENT GRID SQUARE
          ICRP=ICR
          GO TO 30
C  SAME GRID SQUARE AGAIN. KEEP THE FIRST AND LAST CROSSING POINTS, AND
C   IGNORE THE MIDDLE POINT.
   60     CCROSS(I-1,IC)=CCROSS(I,IC)
          CHIGHT(I-1,IC)=CHIGHT(I,IC)
          GO TO 50
C
C  END OF CURRENT COLUMN
C
   70     ICOLP=ICOL
          CCROSS(2,IC+1)=CCROSS(I,IC)
          CHIGHT(2,IC+1)=CHIGHT(I,IC)
C
C  END OF INPUT DATA
C
   80     NCC(IC)=I
C
C  READ HEIGHT AND SLOPE AT THE BOTTOM AND TOP OF THE COLUMN
C
          READ (4) ZBOT,ZDBOT,ZTOP,ZDTOP
          CCROSS(1,IC)=YA
          CHIGHT(1,IC)=ZBOT
          SLOPC1(IC)=ZDBOT
          CCROSS(I,IC)=YB
          CHIGHT(I,IC)=ZTOP
```

```
          SLOPCN(IC)=ZDTOP
          I=2
   90     CONTINUE
C
C  START LOOKING AT SUCCESSIVE ROWS
C
       I=2
       READ (1) RCROSS(I),RHIGHT(I),IROWP
C
          DO 370 IR=1,NROWS
C
C  ZERO THE HEIGHT ARRAY Z
C
              DO 100 IC=I,NCOLS
  100         Z(IC)=0.0
C
C  START AND STOP COLUMNS FOR ROW IR
C
          L1=NCSTRT(IR)
          L2=NCSTOP(IR)
C
          IF (IR.GT.1) GO TO 120
C  SET THE HEIGHT OF THE FIRST ROW OF THE COMPOSITE MAP
              DO 110 IC=L1,L2
  110         Z(IC)=CHIGHT(1,IC)
          GO TO 140
  120     IF (IR.LT.NROWS) GO TO 140
C  SET THE HEIGHT OF THE LAST ROW OF THE COMPOSITE MAP
              DO 130 IC=L1,L2
              NCCI=NCC(IC)
  130         Z(IC)=CHIGHT(NCCI,IC)
          ISTOP=1
          GO TO 360
  140     CONTINUE
C
C  READ CROSSINGS FOR ROW IR
C
C  START AND STOP X-VALUES FOR THIS ROW
          XA=FLOAT(L1)*DX
          XB=FLOAT(L2)*DX
  150     IF (RCROSS(I).GE.XA) GO TO 160
C  FIRST CROSSING POINT IS OUTSIDE MAP BOUNDARY
          READ (1) RCROSS(I),RHIGHT(I),IROW
          GO TO 150
C  ICRP FIXES THE GRID SQUARE WHICH CONTAINS THIS CROSSING POINT
  160     ICRP=RCROSS(I)/DX
C  READ THE NEXT CROSSING POINT
  170     I=I+1
          READ (1,END=210) RCROSS(I),RHIGHT(I),IROW
          IF (IROW.NE.IROWP) GO TO 220
C  IN THE SAME ROW
          IF (RCROSS(I).GT.XB) GO TO 190
C  AND WITHIN BOUNDARY
          ICR=RCROSS(I)/DX
          IF (ICR.EQ.ICRP) GO TO 180
C  BUT DIFFERENT GRID SQUARE
          ICRP=ICR
```

```
          GO TO 170
C  SAME GRID SQUARE, ARE THERE ANY MORE?
  180     I=I+1
  190     READ (1,END=210) RCROSS(I),RHIGHT(I),IROW
          IF (IROW.NE.IROWP) GO TO 220
C  IN THE SAME ROW
          IF (RCROSS(I).GT.XB) GO TO 190
C  AND WITHIN BOUNDARY
          ICR=RCROSS(I)/DX
          IF (ICR.EQ.ICRP) GO TO 200
C  BUT DIFFERENT GRID SQUARE
          ICRP=ICR
          GO TO 170
C  AND SAME GRID SQUARE AGAIN. KEEP THE FIRST AND LAST CROSSING POINTS,
C   AND IGNORE THE MIDDLE POINT.
  200     RCROSS(I-1)=RCROSS(I)
          RHIGHT(I-1)=RHIGHT(I)
          GO TO 190
C
C  END OF THIS ROW
C
  210     ISTOP=1
C  NCROSS  NUMBER OF CROSSING POINTS ALONG ROW IR
  220     NCROSS=I
          RCR=RCROSS(I)
          RCH=RHIGHT(I)
C
C  DISREGARD CROSSINGS OF FIRST ROW: HEIGHTS ALONG THE BOUNDARY ARE
C   KNOWN.
C
          IF (IR.EQ.1) GO TO 360
C
C  SET HEIGHT AT START AND END OF ROW
C
          READ (4) ZLEFT,ZDLEFT,ZRITE,ZDRITE
          RCROSS(1)=XA
          RHIGHT(1)=ZLEFT
          SLOPR1=ZDLEFT
          RCROSS(NCROSS)=XB
          RHIGHT(NCROSS)=ZRITE
          SLOPRN=ZDRITE
C
C  Y-VALUE OF THIS ROW
C
          Y=FLOAT(IR)*DY
C
C  VECTOR BPAR DEFINES END CONDITIONS FOR CUBIC SPLINE
C
          BPAR(1)=1.
          BPAR(2)=6.*((RHIGHT(2)-RHIGHT(1))/(RCROSS(2)-RCROSS(1))-SLOPR1
     *    )/     (RCROSS(2)-RCROSS(1))
          BPAR(3)=1.
          BPAR(4)=6.*(SLOPRN-(RHIGHT(NCROSS)-RHIGHT(NCROSS-1))/(RCROSS(N
     *    CROSS)-RCROSS(NCROSS-1)))/(RCROSS(NCROSS)-RCROSS(NCROSS-1))
C
C  OBTAIN CUBIC SPLINE COEFFICIENTS FOR ROW IR
C
```

```
          CALL ICSICU(RCROSS,RHIGHT,NCROSS,BPAR,RCOEFF,IRDIM,IER)
          IF (IER.GT.0) WRITE (6,230) (RCROSS(J),J=1,NCROSS)
  230 FORMAT(1X,16F8.2)
C
C  SET HEIGHT OF FIRST AND LAST COLUMNS
C
          Z(L1)=RHIGHT(1)
          Z(L2)=RHIGHT(NCROSS)
C
C  INNER COLUMNS
C
          LR=2
          L1P1=L1+1
          L2M1=L2-1
C
              DO 350 IC=L1P1,L2M1
C
              IF (NRSTRT(IC).NE.IR) GO TO 240
C  ROW IR IS THE FIRST ROW OF ONE OF THE MAPS
              Z(IC)=CHIGHT(1,IC)
              GO TO 350
C  NCCI    NUMBER OF CROSSING POINTS ALONG COLUMN IC
  240         NCCI=NCC(IC)
              IF (NRSTOP(IC).NE.IR) GO TO 250
C  ROW IR IS THE LAST ROW OF ONE OF THE MAPS
              Z(IC)=CHIGHT(NCCI,IC)
              GO TO 350
C
C  X-VALUE OF THIS COLUMN
C
  250         X=FLOAT(IC)*DX
C
C  FIND CLOSEST CROSSING POINTS ON EACH SIDE OF X
C
  260         IF (RCROSS(LR)-X) 270,280,290
  270         LR=LR+1
              GO TO 260
  280         Z(IC)=RHIGHT(LR)
              GO TO 350
C  RCROSS(LR-1) < X < RCROSS(LR)
  290         X1=RCROSS(LR-1)
              X2=RCROSS(LR)
              Z1=RHIGHT(LR-1)
              Z2=RHIGHT(LR)
              Z1D=RCOEFF(LR-1,1)
              Z2D=RCOEFF(LR,1)
              XG=X
C
C  OBTAIN CUBIC SPLINE COEFFICIENTS FOR COLUMN IC
C
              BPAR(2)=6.*((CHIGHT(2,IC)-CHIGHT(1,IC))/(CCROSS(2,IC)-CCRO
     *        SS(1,IC))-SLOPC1(IC))/(CCROSS(2,IC)-CCROSS(1,IC))
              BPAR(4)=6.*(SLOPCN(IC)-(CHIGHT(NCCI,IC)-CHIGHT(NCCI-1,IC))
     *        /(CCROSS(NCCI,IC)-CCROSS(NCCI-1,IC)))/(CCROSS(NCCI,IC)-CCR
     *        OSS(NCCI-1,IC))
              CALL ICSICU(CCROSS(1,IC),CHIGHT(1,IC),NCCI,BPAR,CCOEFF,ICD
     *        IM,IER)
```

```
              IF (IER.GT.0) WRITE (6,230) (CCROSS(J,IC),J=1,NCCI)
C
C  FIND CLOSEST CROSSING POINTS ON EACH SIDE OF Y
C
C  LCI      THE CURRENT CROSSING POINT NUMBER FOR COLUMN IC
              LCI=LC(IC)
  300         IF (CCROSS(LCI,IC)-Y) 310,320,330
  310         LC(IC)=LC(IC)+1
              LCI=LC(IC)
              GO TO 300
  320         Z(IC)=CHIGHT(LCI,IC)
              GO TO 350
C  CCROSS(LCI-1,IC) < X < CCROSS(LCI,IC)
  330         Y1=CCROSS(LCI-1,IC)
              Y2=CCROSS(LCI,IC)
              Z3=CHIGHT(LCI-1,IC)
              Z4=CHIGHT(LCI,IC)
              Z3D=CCOEFF(LCI-1,1)
              Z4D=CCOEFF(LCI,1)
              YG=Y
C
C  BICUBIC SPLINE INTERPOLATION
C
              CALL BICUBE
              IF (ZGSD.GT.100000.) WRITE (6,340) X1,Y1,X2,Y2,X,Y,Z1,Z1D,
     *        Z2,Z2D,  Z3,Z3D,Z4,Z4D,ZG,ZGSD
  340 FORMAT(1X,15F8.2,1PE12.4)
              Z(IC)=ZG
C
C  NEXT COLUMN
C
  350         CONTINUE
C
C  NEXT ROW
C
  360      IROWP=IROW
           RCROSS(2)=RCR
           RHIGHT(2)=RCH
           I=2
           WRITE (3) (Z(J),J=1,NCOLS)
           IF (ISTOP.EQ.1) GO TO 380
C
  370      CONTINUE
C
  380 STOP
      END
      SUBROUTINE BICUBE
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/BSP/X1,Y1,X2,Y2,XG,YG,Z1,Z1D,Z2,Z2D,Z3,Z3D,Z4,Z4D,ZG,ZGSD
```

```
C
C                                  |
C                                  X (XG,Y2,Z4,Z4D)
C                                  |
C                                  |
C            ----X-----------------*-----------------X-----------
C          (X1,YG,Z1,Z1D)        /|           (X2,YG,Z2,Z2D)
C                              /   |
C                          (XG,YG)|
C                                 |
C                                 X (XG,Y1,Z3,Z3D)
C                                 |
C
C  WITH THE INPUT SHOWN ABOVE, THE SUBROUTINE CALCULATES THE HEIGHT ZG
C    AT THE POINT (XG,YG) WITH STANDARD DEVIATION ZGSD
C
      DATA EPS/1.E-6/
C
C  IF ANY OF THE FOUR POINTS ARE WITHIN EPS OF (XG,YG), SET ZG EQUAL TO
C    THE Z-VALUE AT THE APPROPRIATE POINT AND RETURN
C
      ZGSD=0.
      IF (DABS(XG-X1).GT.EPS) GO TO 10
      ZG=Z1
      RETURN
   10 IF (DABS(XG-X2).GT.EPS) GO TO 20
      ZG=Z2
      RETURN
   20 IF (DABS(YG-Y1).GT.EPS) GO TO 30
      ZG=Z3
      RETURN
   30 IF (DABS(YG-Y2).GT.EPS) GO TO 40
      ZG=Z4
      RETURN
   40 CONTINUE
      NFLAG=0
C
C  COMPUTE POWERS REQUIRED LATER
C
      X1SQ=X1**2
      X2SQ=X2**2
      XGSQ=XG**2
      X1CU=X1*X1SQ
      X2CU=X2*X2SQ
      XGCU=XG*XGSQ
      Y1SQ=Y1**2
      Y2SQ=Y2**2
      YGSQ=YG**2
      Y1CU=Y1*Y1SQ
      Y2CU=Y2*Y2SQ
      YGCU=YG*YGSQ
C
      X1G=X1-XG
      X1GSQ=X1G**2
      X2G=X2-XG
      X2GSQ=X2G**2
C
```

```
      Y1G=Y1-YG
      Y1GSQ=Y1G**2
      Y1GCU=Y1G*Y1GSQ
      Y2G=Y2-YG
      Y2GSQ=Y2G**2
      X21=X2-X1
      Y21=Y2-Y1
C
C  CALCULATE TERMS FOR TRIANGULAR REDUCTION
C
      R01426=XG+2.*X2
      R01526=-Y2GSQ/X2GSQ
      R01626=R01526*(YG+2.*Y2)
      R0R26=(Z4-Z2+X2G*Z2D-Y2G*Z4D)/X2GSQ
C
      DENOM=-3.*X1GSQ
      R31434=(4.*X1G-X2G)*X2G/DENOM
      R31534=-2.*X21*R01526/DENOM
      R31634=-2.*X21*R01626/DENOM
      R3R34=(Z2D-Z1D-2.*X21*R0R26)/DENOM
C
      R51436=-2.*X1G*X2G*X21/3.
      R51536=R01526*X21*(3.*X2G-X1G)/3.
      R51636=R51536*(YG+2.*Y2)
      R5R36=Z4-Z1-Y2G*Z4D+X1G*(Z1D-X1G*(R0R26+2.*X1G*R3R34))
C
      R61437=2.*Y1GCU*(R31434+1.)
      R61537=-Y21*(Y1G+Y2G)+2.*Y1GCU*R31534
      R61637=(2.*Y1+YG)*Y1GSQ-(2.*Y2+YG)*Y2GSQ+2.*Y1GCU*R31634
      R6R37=Z4-Z3+Y1G*Z3D-Y2G*Z4D+2.*Y1GCU*R3R34
C
      IF (DABS(R31434+1.).LE.EPS) NFLAG=NFLAG+1
      DENOM=-3.*Y1GSQ*(1.+R31434)
      R41538=(2.*Y21-3.*Y1GSQ*R31534)/DENOM
      R41638=3.*(Y21*(Y2+Y1)-Y1GSQ*R31634)/DENOM
      R4R38=(Z4D-Z3D-3.*Y1GSQ*R3R34)/DENOM
C
      R51539=R51536-R51436*R41538
      R61540=R61537-R61437*R41538
C
      IF (DABS(R51539).LE.EPS) NFLAG=NFLAG+10
      R51641=(R51636-R51436*R41638)/R51539
      R5R41=(R5R36-R51436*R4R38)/R51539
C
      DENOM=R61637-R61437*R41638-R61540*R51641
      IF (DABS(DENOM).LE.EPS) NFLAG=NFLAG+100
C
C  COMPUTE COEFFICIENTS A(I,J)
C
      A43=(R6R37-R61437*R4R38-R61540*R5R41)/DENOM
      A42=R5R41-R51641*A43
      A23=R4R38-R41638*A43-R41538*A42
      A13=R3R34-R31634*A43-R31534*A42-R31434*A23
      A33=-A13+A23+A43
      A32=-3.*YG*A33+A42+3.*YG*A43
      A22=R0R26-R01626*A43-R01526*A42-R01426*A23
      A12=A22-3.*XG*A13+3.*XG*A23
```

```
      A41=Z4D-2.*Y2*A42-3.*Y2SQ*A43
      A31=Z3D-2.*Y1*A32-3.*Y1SQ*A33
      A21=Z2D-2.*X2*A22-3.*X2SQ*A23
      A11=Z1D-2.*X1*A12-3.*X1SQ*A13
      A40=Z4-Y2*Z4D+Y2SQ*A42+2.*Y2CU*A43
      A30=Z3-Y1*Z3D+Y1SQ*A32+2.*Y1CU*A33
      A20=Z2-X2*Z2D+X2SQ*A22+2.*X2CU*A23
      A10=Z1-X1*Z1D+X1SQ*A12+2.*X1CU*A13
C
C  COMPUTE HEIGHT VALUES AT (XG,YG)
C
      ZG1=A10+A11*XG+A12*XGSQ+A13*XGCU
      ZG2=A20+A21*XG+A22*XGSQ+A23*XGCU
      ZG3=A30+A31*YG+A32*YGSQ+A33*YGCU
      ZG4=A40+A41*YG+A42*YGSQ+A43*YGCU
C
      ZG=(ZG1+ZG2+ZG3+ZG4)/4.
      ZGSD=((ZG1-ZG)**2+(ZG2-ZG)**2+(ZG3-ZG)**2+(ZG4-ZG)**2)/3.
C
C  ERROR CODE: NFLAG=IJK
C       K=1    R31434+1.=0.
C       J=1    R51539=0.
C       I=1    R61637-R61437*R41638-R61540*R51641=0.
C
      IF (NFLAG.NE.0) WRITE (6,50) NFLAG
   50 FORMAT(' WARNING: ERROR IN BICUBE, NFLAG =',I3)
C
      RETURN
      END
C     SUBROUTINE ICSICU (X,Y,NX,BPAR,C,IC,IER)
C
C-ICSICU--------S/D-----LIBRARY 1--------------------------------------
C
C   FUNCTION            - INTERPOLATORY APPROXIMATION BY CUBIC SPLINES
C                         WITH ARBITRARY SECOND DERIVATIVE END
C                         CONDITIONS.
C   USAGE               - CALL ICSICU(X,Y,NX,BPAR,C,IC,IER)
C   PARAMETERS   X      - VECTOR OF LENGTH NX CONTAINING THE ABSCISSAE
C                         OF THE NX DATA POINTS (X(I),Y(I)) I=1,...,
C                         NX (INPUT). X MUST BE ORDERED SO THAT
C                         X(I) .LT. X(I+1).
C                Y      - VECTOR OF LENGTH NX CONTAINING THE ORDINATES
C                         (OR FUNCTION VALUES) OF THE NX DATA POINTS
C                         (INPUT).
C                NX     - NUMBER OF ELEMENTS IN X AND Y (INPUT). NX
C                         MUST BE .GE. 2.
C                BPAR   - VECTOR OF LENGTH 4 CONTAINING THE END
C                         CONDITION PARAMETERS (INPUT).
C                         2.0*SPP(1)+BPAR(1)*SPP(2) = BPAR(2),
C                         BPAR(3)*SPP(NX-1)+2.0*SPP(NX) = BPAR(4),
C                         WHERE SPP(I) = SECOND DERIVATIVE OF THE
C                         CUBIC SPLINE FUNCTION S EVALUATED AT X(I).
C                C      - SPLINE COEFFICIENTS (OUTPUT). C IS AN NX-1 BY
C                         3 MATRIX. THE VALUE OF THE SPLINE
C                         APPROXIMATION AT T IS
C                         S(T) = ((C(I,3)*D+C(I,2))*D+C(I,1))*D+Y(I)
C                         WHERE X(I) .LE. T .LT. X(I+1) AND
```

```
C                          D = T-X(I).
C               IC      - ROW DIMENSION OF MATRIX C IN THE CALLING
C                         PROGRAM (INPUT). IC MUST BE .GE. NX-1.
C               IER     - ERROR PARAMETER.
C                         TERMINAL ERROR
C                           IER = 129, IC IS LESS THAN NX-1.
C                           IER = 130, NX IS LESS THAN 2.
C                           IER = 131, INPUT ABSCISSA ARE NOT ORDERED
C                             SO THAT X(1) .LT. X(2) ... .LT. X(NX).
C   PRECISION           - SINGLE/DOUBLE
C   REQD. IMSL ROUTINES - UERTST
C   LANGUAGE            - FORTRAN
C-----------------------------------------------------------------------
C   LATEST REVISION     - JULY 29, 1974
C
      SUBROUTINE ICSICU  (X,Y,NX,BPAR,C,IC,IER)
C
      DIMENSION          X(NX),Y(NX),BPAR(4),C(IC,3)
C*    DOUBLE PRECISION   BPAR,C,DX,DXJ,DXJP1,DXP,DYJ,DYJP1,HALF,ONE,PJ,
CO   1                   SIX,SIXI,TWO,X,Y,YPPA,YPPB,ZERO
      EQUIVALENCE        (DXJ,YPPB),(PJ,SIXI),(DXJP1,YPPA)
C*    DATA               ZERO/0.0D0/,HALF/0.5D0/,ONE/1.0D0/,
C2   1                   TWO/2.0D0/,SIX/6.0D0/
      DATA               ZERO/0.0/,HALF/0.5/,ONE/1.0/,
     1                   TWO/2.0/,SIX/6.0/
C                                CHECK ERROR CONDITIONS
      IER = 0
      NXM1 = NX-1
      IF (IC .LT. NXM1) GO TO 60
      IF (NX .LT. 2) GO TO 70
      IF (NX .EQ. 2) GO TO 20
C                                COMPUTE COEFFICIENTS AND RIGHT
C                                HAND SIDE OF THE TRIDIAGONAL
C                                SYSTEM DEFINING THE SECOND
C                                DERIVATIVES OF THE SPLINE
C                                INTERPOLANT FOR (X,Y)
C                                C(J,1) = LAMBDA(J)
C                                C(J,2) = MU(J)
C                                C(J,3) = D(J)
      DXJ = X(2)-X(1)
      IF (DXJ .LE. ZERO) GO TO 80
      DYJ = Y(2)-Y(1)
          DO 10 J=2,NXM1
          DXJP1=X(J+1)-X(J)
          IF(DXJP1.LE.ZERO)GO TO 80
          DYJP1=Y(J+1)-Y(J)
          DXP=DXJ+DXJP1
          C(J,1)=DXJP1/DXP
          C(J,2)=ONE-C(J,1)
          C(J,3)=SIX*(DYJP1/DXJP1-DYJ/DXJ)/DXP
          DXJ=DXJP1
          DYJ=DYJP1
   10     CONTINUE
C                                FACTOR THE TRIDIAGONAL MATRIX
C                                AND SOLVE FOR U
C                                C(J,2) = U(J)
C                                C(J,1) = Q(J)
```

```
C                                     BPAR(1) = LAMBDA(1)
C                                     BPAR(2) = D(1)
C                                     BPAR(3) = MU(NX)
C                                     BPAR(4) = D(NX)
   20 C(1,1) = -BPAR(1)*HALF
      C(1,2) = BPAR(2)*HALF
      IF (NX .EQ. 2) GO TO 40
          DO 30 J=2,NXM1
          PJ=C(J,2)*C(J-1,1)+TWO
          C(J,1)=-C(J,1)/PJ
          C(J,2)=(C(J,3)-C(J,2)*C(J-1,2))/PJ
   30     CONTINUE
C                                     SOLVE FOR CUBIC COEFFICIENTS
C                                     OF SPLINE INTERPOLANT
C                                     C(J,1), C(J,2), AND C(J,3)
   40 YPPB = (BPAR(4)-BPAR(3)*C(NXM1,2))/(BPAR(3)*C(NXM1,1)+TWO)
      SIXI = ONE/SIX
          DO 50 I=1,NXM1
          J=NX-I
          YPPA=C(J,1)*YPPB+C(J,2)
          DX=X(J+1)-X(J)
          C(J,3)=SIXI*(YPPB-YPPA)/DX
          C(J,2)=HALF*YPPA
          C(J,1)=(Y(J+1)-Y(J))/DX-(C(J,2)+C(J,3)*DX)*DX
          YPPB=YPPA
   50     CONTINUE
      GO TO 100
   60 IER = 129
      GO TO 90
   70 IER = 130
      GO TO 90
   80 IER = 131
   90 CONTINUE
      CALL UERTST(IER,6HICSICU)
  100 RETURN
      END
C     SUBROUTINE UERTST (IER,NAME)
C
C-UERTST----------------LIBRARY 1-------------------------------------
C
C   FUNCTION            - ERROR MESSAGE GENERATION
C   USAGE               - CALL UERTST(IER,NAME)
C   PARAMETERS   IER    - ERROR PARAMETER. TYPE + N  WHERE
C                           TYPE= 128 IMPLIES TERMINAL ERROR
C                                  64 IMPLIES WARNING WITH FIX
C                                  32 IMPLIES WARNING
C                           N   = ERROR CODE RELEVANT TO CALLING ROUTINE
C                NAME   - INPUT VECTOR CONTAINING THE NAME OF THE
C                           CALLING ROUTINE AS A SIX CHARACTER LITERAL
C                           STRING.
C   LANGUAGE            - FORTRAN
C-----------------------------------------------------------------------
C   LATEST REVISION     - JANUARY 18, 1974
C
      SUBROUTINE UERTST(IER,NAME)
C
      DIMENSION          ITYP(5,4),IBIT(4)
```

```
      INTEGER*2          NAME(3)
      INTEGER            WARN,WARF,TERM,PRINTR
      EQUIVALENCE        (IBIT(1),WARN),(IBIT(2),WARF),(IBIT(3),TERM)
      DATA     ITYP      /'WARN','ING ','    ','    ','    ',
     *                    'WARN','ING(','WITH',' FIX',')   ',
     *                    'TERM','INAL','    ','    ','    ',
     *                    'NON-','DEFI','NED ','    ','    '/,
     *         IBIT      / 32,64,128,0/
      DATA     PRINTR    / 6/
      IER2=IER
      IF (IER2 .GE. WARN) GO TO 10
C                                     NON-DEFINED
      IER1=4
      GO TO 40
   10 IF (IER2 .LT. TERM) GO TO 20
C                                     TERMINAL
      IER1=3
      GO TO 40
   20 IF (IER2 .LT. WARF) GO TO 30
C                                     WARNING(WITH FIX)
      IER1=2
      GO TO 40
C                                     WARNING
   30 IER1=1
C                                     EXTRACT 'N'
   40 IER2=IER2-IBIT(IER1)
C                                     PRINT ERROR MESSAGE
      WRITE (PRINTR,50) (ITYP(I,IER1),I=1,5),NAME,IER2,IER
   50 FORMAT(' *** I M S L(UERTST) ***  ',5A4,4X,3A2,4X,I2,
     *   ' (IER = ',I3,')')
      RETURN
      END
```

APPENDIX V

THE BICUBIC SPLINE SUBROUTINE

The bicubic spline interpolation subroutine listed in Appendix III of reference 1 was modified as follows for the present application.

V.1 If any of the four crossing points $\{(X_i, Y_i), i=1,4\}$ are within a small increment (EPS) of the grid point (X_G, Y_G), then the height Z_G at the grid point is set to the height at the appropriate crossing point, ZGSD is set to zero, and control is returned to the main program.

V.2 The number of powers calculated initially has been increased.

V.3 There are now only three error conditions. In each case, a denominator is less than EPS; 1, 10 or 100 is added to the error flag, computation continues and the error flag is printed before control returns to the main program.

V.4 Many of the terms which are required for the triangular reduction have been algebraically simplified, which means that fewer operations need to be performed, and thus speed and accuracy are improved. The simplifications are detailed below.

$$(1)\ R01526 = \frac{(y_2^2-y_G^2)-(y_2-y_G)2y_2}{(x_G-x_2)^2} = -\frac{(y_G-y_2)^2}{(x_G-x_2)^2}$$

$$(2)\ R01626 = \frac{(y_2^3-y_G^3)-(y_2-y_G)3y_2^2}{(x_G-x_2)^2} = (y_G+2y_2)R01526$$

$$(3)\ R31434 = \frac{3x_2^2-3x_G^2+6(x_G-x_1)x_G-2(x_2-x_1)(x_G+2x_2)}{-3(x_1-x_G)^2}$$

$$= \frac{(4x_1-x_2-3x_G)(x_2-x_G)}{-3(x_1-x_G)^2}$$

$$(4)\ R51436 = (x_G-x_1)^2\ 3x_G-(x_G-x_1)^2\ R01426-2(x_1-x_G)^3\ R31434$$

$$= -\frac{2}{3}(x_1-x_G)(x_2-x_G)(x_1-x_2)$$

$$(5)\ R51536 = (y_2^2-y_G^2)-(y_2-y_G)2y_2-(x_G-x_1)^2\ R01526-2(x_1-x_G)^3\ R31534$$

$$= -\frac{1}{3}R01526(x_2-x_1)(3x_2-x_1-2x_G)$$

$$(6)\ R51636 = (y_2^3-y_G^3)-(y_2-y_G)3y_2^2-(x_G-x_1)^2\ R01626-2(x_1-x_G)^3\ R31634$$

$$= (y_G+2y_2)R51536$$

$$(7)\ R61537 = (y_G-y_1)^2-(y_G-y_2)^2+2(y_1-y_G)^3\ R31534$$

$$= (y_2-y_1)(2y_G-y_1-y_2)+2(y_1-y_G)^3\ R31534$$

$$(8)\ R61637 = 2y_G^3-2y_2^3+3y_2^2\ y_G+3y_1\ y_G\ (y_1-2y_G)+2(y_1-y_G)^3\ (1+R31634)$$

$$= (2y_1+y_G)(y_1-y_G)^2-(2y_2+y_G)(y_2-y_G)^2+2(y_1-y_G)^3\ R31634$$

$$(9)\ R41638 = 3y_2^2-3y_G^2+6(y_G-y_1)y_G-3(y_G-y_1)^2\ (1+R31634)$$

$$= 3(y_2-y_1)(y_2+y_1)-3(y_G-y_1)^2\ R31634$$

APPENDIX VI

A STEP-BY-STEP PROCEDURE FOR BICUBIC SPLINE INTERPOLATION OF DIGITISED CONTOUR DATA

The following is a step-by-step procedure for transferring paper tape contour data from the digitizer onto disk, and then implementing the bicubic spline interpolation procedure to obtain a rectangular grid of terrain heights. All dataset names are prefixed MJD.

(i) Create a partitioned dataset, which will contain the digitized contour data, one member for each map.

DCB = (RECFM=FB, LRECL=16, BLKSIZE=2560), and
SPACE = (TRK,(4,4,5)) should be sufficient.

(ii) In PAPATAPE.LOAD.CNTL change the job name to match the paper tape label, and specify the above partitioned dataset name and the member name (lines 170 and 210). The paper tape is read, the data are translated to EBCDIC, and stored in the specified member.

(iii) Edit the member, re-number, and save.

(iv) Execute PAPATAPE.CLIST(LIST) which runs PAPATAPE.FORT(LIST) and lists the digitizer data, ten points across each line, with an asterisk to indicate a contour height flag.

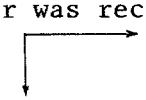
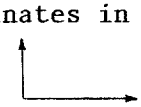
(v) Execute PAPATAPE.CLIST(DEBUG) which runs PAPATAPE.FORT(DEBUG) and detects and flags any of the following possible errors in the digitizer data, namely,

1. The distance between consecutive points exceeds 2 cm.
2. The Y-reading is less than 80 (contour height flag).
3. Both X and Y readings are less than 80 or greater than 7520 (error flag to indicate that previous point was wrong).

(vi) Edit the member and delete or correct the points - occasionally a digit is incorrect and plotting a few points by hand on graph paper will indicate what it should have been.

(vii) Execute PAPATAPE.CLIST(PLOTTEK) to run PAPATAPE.FORT(TEKTRONX) which plots the contour lines on the Tektronix screen. Check for any more odd points, and correct if necessary, OR execute PAPATAPE.CLIST(PLOTCAL) to run PAPATAPE.FORT(CALCOMP) which plots the contour lines on the Calcomp plotter.

(viii) If the digitizer was recording coordinates in reverse mode,

i.e.  instead of 

then execute PAPATAPE.CLIST(SWAPXY) to run PAPATAPE.FORT(SWAPXY) and translate the data back to normal coordinates.

(ix) Ensure the first four points in each member are corner points, and in the correct order, i.e. bottom right, bottom left, top

left, top right.

(x) Delete SPLINE.ROW.DATA and SPLINE.COL.DATA (if they exist). Submit SPLINE.CNTL(DATASETS) to re-create them - they will be used to store row and column crossing points.

(xi) Check if SPLINE.LOAD contains member GRIDCROS. If not, then submit SPLINE.CNTL(GRIDXCMP) to compile the gridline crossing points program and store the load module.

(xii) Set up a partitioned dataset (normal TSO format and size), with a member corresponding to each member of the contour data PDS, containing the map coordinates of the four corner points (again in the order bottom right, bottom left, top left, top right), the grid interval (Km), and the map coordinates of the bottom left corner of the composite map. If only one map is to be read, repeat the map coordinates of the bottom left corner. The format required is 9F8.0, hence there will be nine values in the first line and two in the second.

(xiii) Edit SPLINE.CNTL(GRIDXRUN), change the two PDS names(lines 70 and 80) and the member names (from line 140), and submit. The GRIDCROS load module calculates all the points at which a contour line crosses a row or column, and for each point stores the X or Y value, the height, and the row or column number.

(xiv) Execute SPLINE.CLIST(LISTTERM) to run SPLINE.FORT(LISTCROS). Set IPRINT = 0 when prompted - the program calculates the total numbers of row crossings and column crossings (required for the next step).

(xv) Edit SPLINE.CNTL(SORTX), set the number of row crossings (line 90) and the number of column crossings (line 170) and submit. The dataset SPLINE.ROW.DATA containing row crossing points will be sorted into increasing row number order, and increasing X-value within each row. Similarly SPLINE.COL.DATA containing column crossing points will be sorted into increasing column number order, and increasing Y-value within each column.

(xvi) Execute SPLINE.CLIST(EXTRA) to run SPLINE.FORT(EXTRAXPT). If successive crossing points in a row or column are more than 0.5 km apart, then the program inserts crossing points at equal intervals of up to 0.5 km, with linearly interpolated heights. If successive crossing points are too close together then they are combined into one point with the average of the two heights at the average X or Y value.
The choice of 0.5 Km as the maximum separation of crossing points coincided with the grid spacing, and may be changed if necessary (line 300 of the program above).

(xvii) Execute SPLINE.CLIST(LISTPRIN) to run SPLINE.FORT(LISTCROS). Set IPRINT =1 when prompted. The program will list the row and column crossings, and print the maximum number of crossing points along any row, and along any column (required for the next step).

(xviii) Let MR=maximum number of crossing points along any row+2,

MC=maximum number of crossing points along any column+2,
NROWS=number of rows,
NCOLS=number of columns.
Edit SPLINE.FORT(Z) and make the following changes:

1. Line 420 - set first dimension of CCROSS and CHIGHT = MC, set second dimension of CCROSS and CHIGHT = NCOLS, set dimension of NCC and LC = NCOLS.
2. Line 440 - set dimension of RCROSS and RHIGHT = MR, set first dimension of CCOEFF = MC-1, set first dimension of RCOEFF = MR-1.
3. Line 450 - set dimension of Z, SLOPC1 and SLOPCN = NCOLS.
4. Line 470 - set dimension of NRSTRT and NRSTOP = NCOLS, set dimension of NCSTRT and NCSTOP = NROWS.
5. Lines 750 to 760 - If the composite map is not square, then set start and stop rows for each column, and the start and stop columns for each row as indicated in the comments above. If the composite map is square, then the lines should be

```
DATA NRSTRT/NCOLS*1/,NRSTOP/NCOLS*NROWS/,
  *NCSTRT/*1/,NCSTOP/NROWS*NCOLS/
```

   with, of course, NCOLS and NROWS replaced by the appropriate numbers.
6. Line 770 - set DX and DY the X and Y grid increments, set ICDIM = MC-1, set IRDIM = MR-1, set NROWS and NCOLS.
7. Line 780 - set LC/NCOLS*2/, set SLOPC1/NCOLS*0.0/, set SLOPCN/NCOLS*0.0/.

(xix) Store in EDGES.DATA the height and slope at the bottom and top of each column of the map, and similarly the height and slope at the start and end of each row.

(xx) Delete SPLINE.Z.DATA if it exists. Edit SPLINE.CNTL(Z), set the logical record length in line 100 to 4+4*NCOLS, and the blocksize to the largest multiple of the record length which is less than 6233 bytes. Submit. The program SPLINE.FORT(Z) performs bicubic spline interpolation from the crossing point data, and stores the terrain height at rectangular grid points in SPLINE.Z.DATA, one row at a time.

(xxi) Edit SPLINE.FORT(READGRID) and set the dimensions of Z(NCOLS,NROWS) in line 390, and in line 430 set DX and DY, NCOLS and NROWS, and (XBL,YBL) the map coordinates of the bottom left corner of the composite map.

(xxii) To list up to 25 columns of the terrain grid at the terminal (format(F5.0)), execute SPLINE.CLIST(READGRID) and when prompted enter the first and last columns to be listed.
To list the whole terrain grid on the printer, edit SPLINE.CNTL(READGRID) and change lines 80 to 120 so that all columns are specified, then submit.

(xxiii) SPLINE.CLIST(PLOTTEK) will run SPLINE.FORT(CONTOUR) and plot contour lines on the Tektronix screen, or SPLINE.CLIST(PLOTCAL) will plot contour lines on the 30" Calcomp.
In SPLINE.FORT(CONTOUR) -

1. Line 370 - change the dimension of DZ to the number of contour heights and the dimension of ZNCOLS,NROWS), X(NCOLS) and Y(NROWS),
2. Lines 420 and 430 - set the number of contour heights, and the height of each contour.
3. Line 440 - set NCOLS and NROWS, (XBL,YBL), DX and DY as in step (xxi).

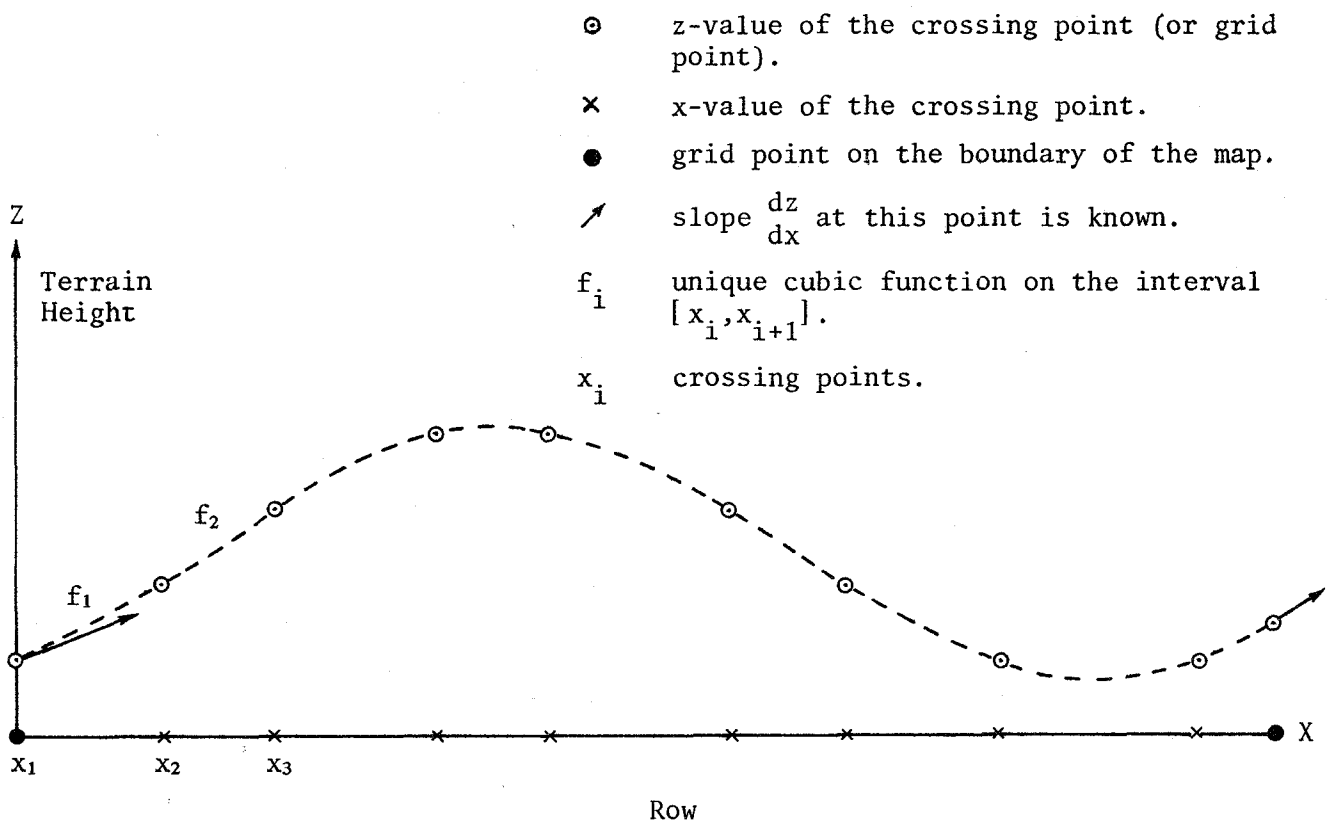


Figure 1. A cubic spline

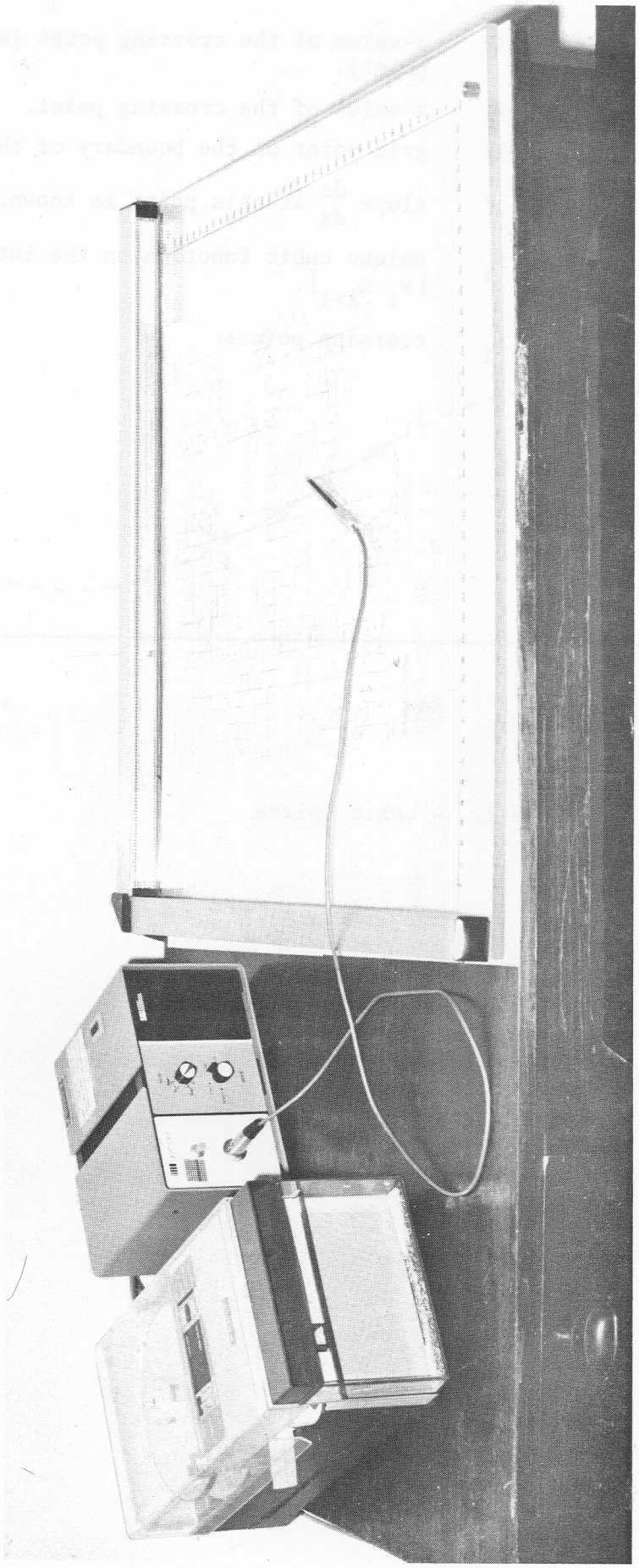


Figure 2. Survey map mounted on the Sonic Digitizer Tablet

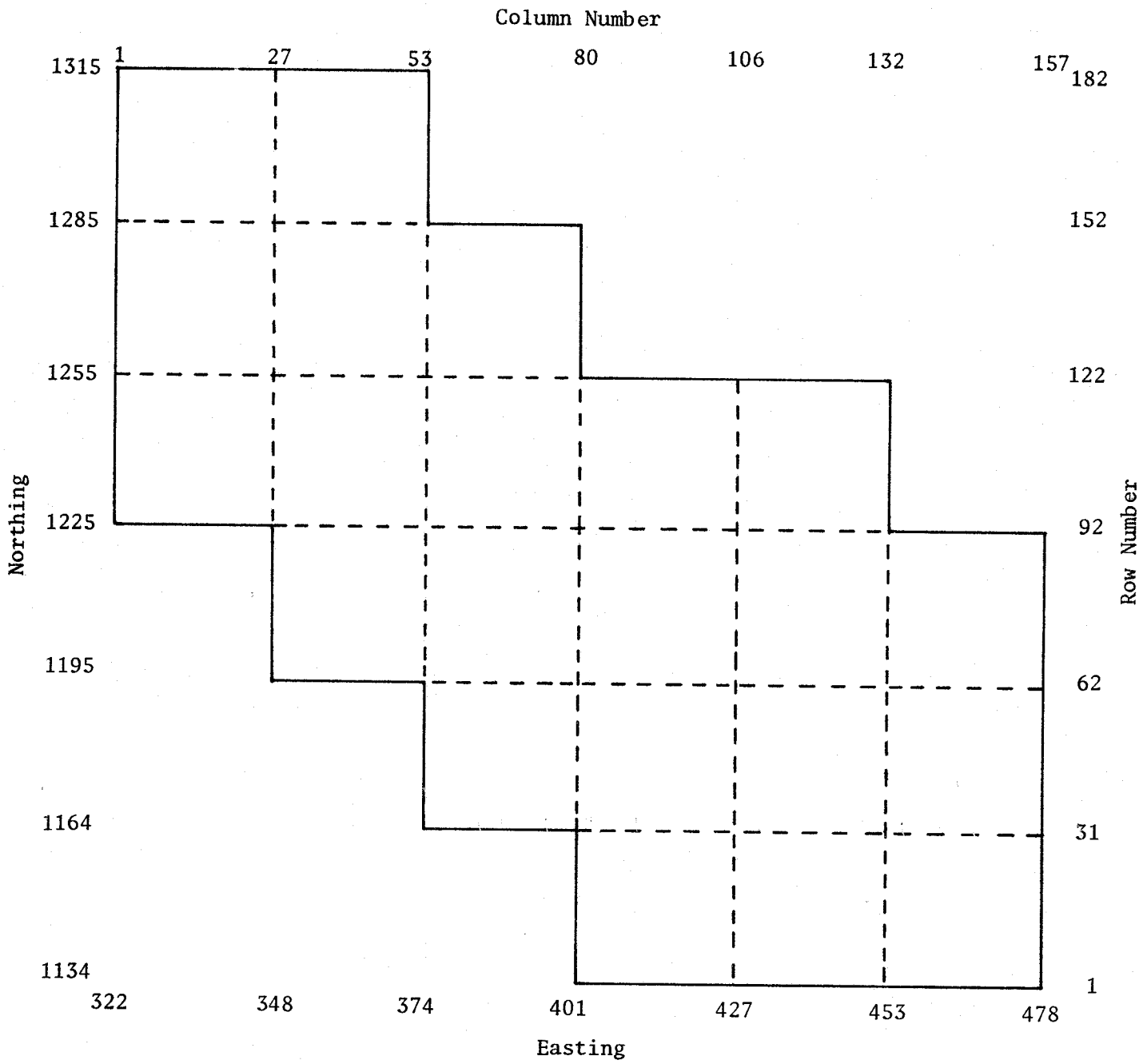


Figure 3. Composite map

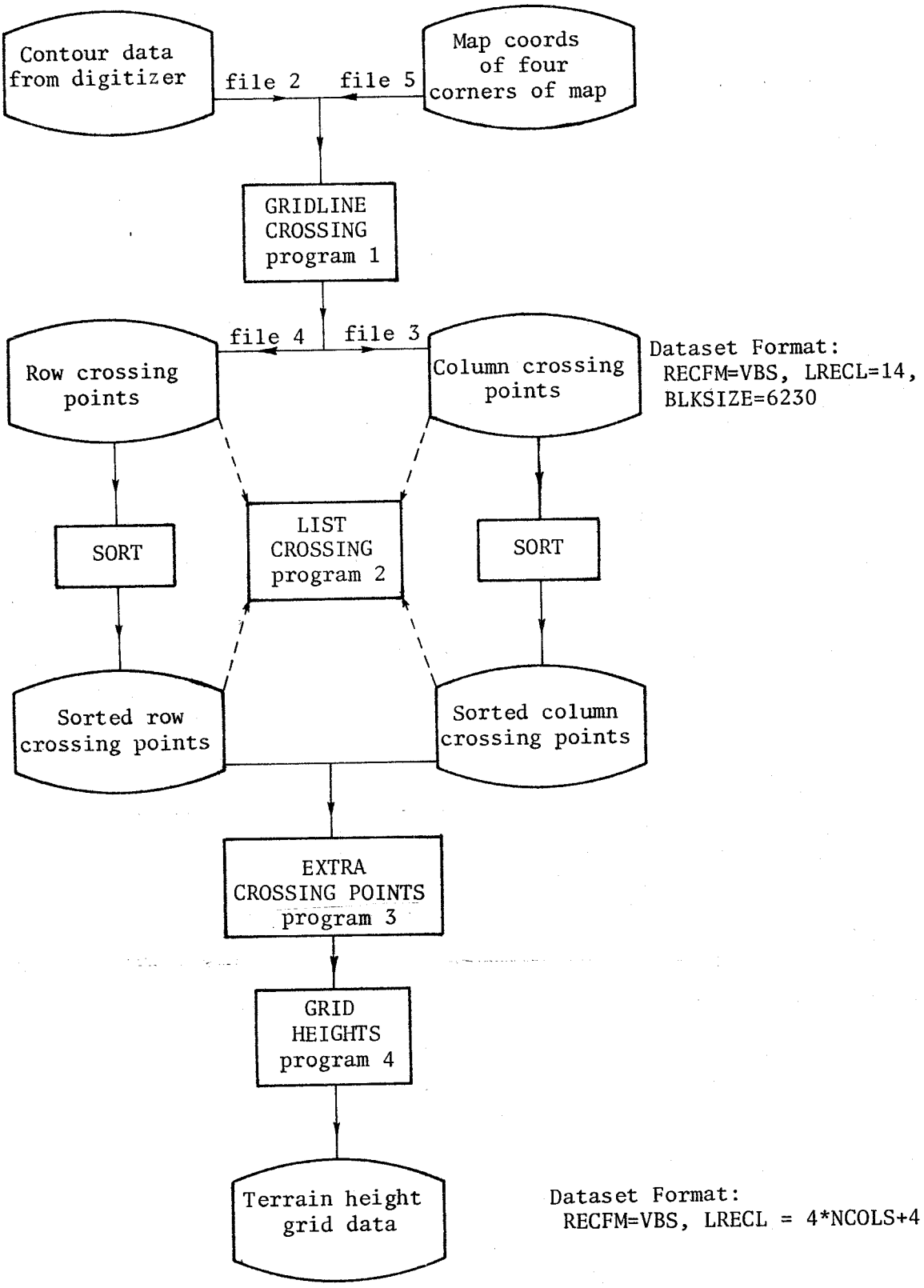


Figure 4. Block diagram of program structure

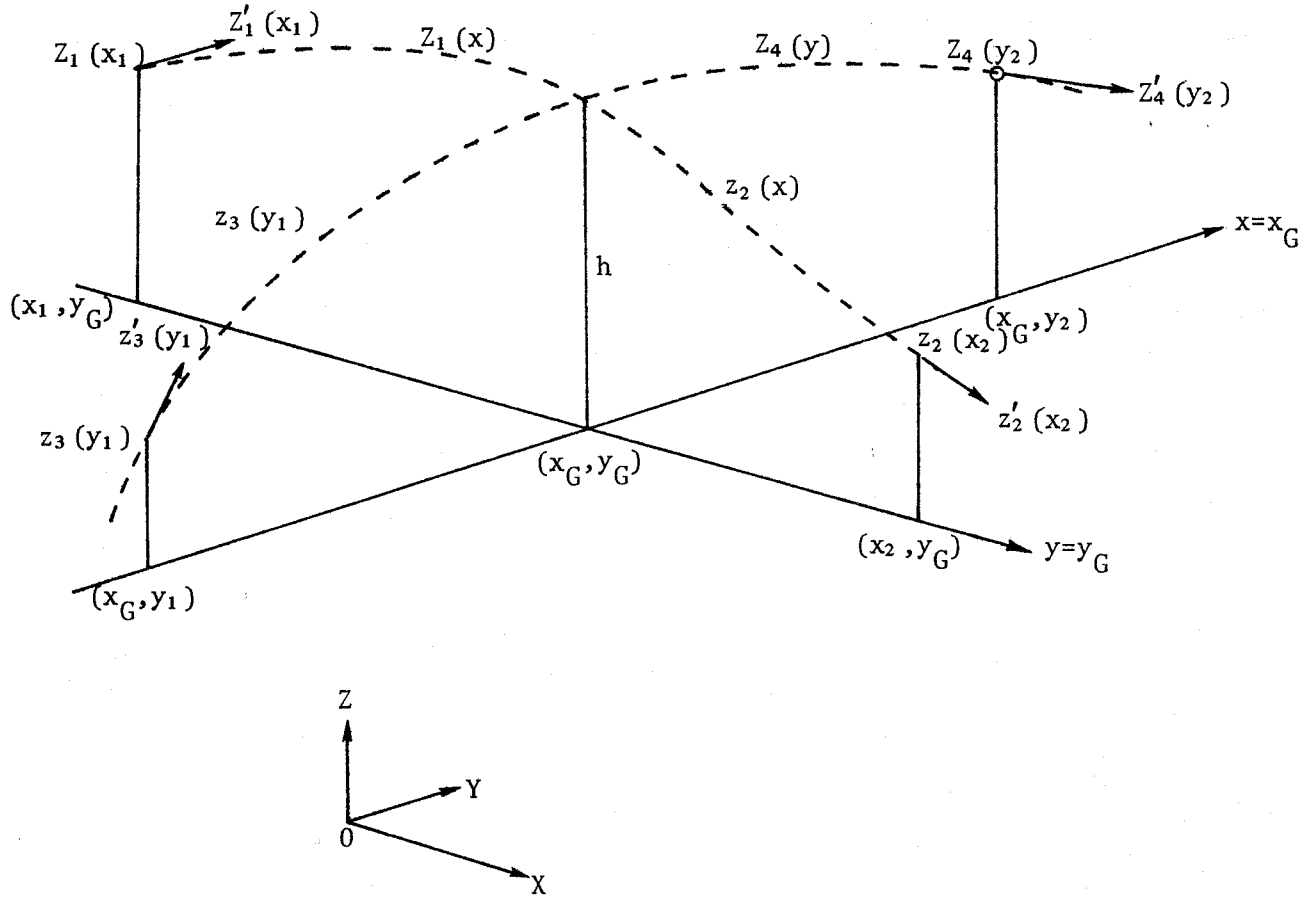


Figure 5. Input parameters for subroutine BICUBE

DOCUMENT CONTROL DATA SHEET

Security classification of this page: UNCLASSIFIED

1 DOCUMENT NUMBERS

AR Number: AR-001-683

Report Number: ERL-0085-TR

Other Numbers:

2 SECURITY CLASSIFICATION

a. Complete Document: Unclassified

b. Title in Isolation: Unclassified

c. Summary in Isolation: Unclassified

3 TITLE

A PROCEDURE FOR PRODUCING A TERRAIN ELEVATION GRID FROM MAP CONTOUR DATA

4 PERSONAL AUTHOR(S):

M.J. Dowling

5 DOCUMENT DATE:

June 1979

6 6.1 TOTAL NUMBER OF PAGES: 45

6.2 NUMBER OF REFERENCES: 2

7 7.1 CORPORATE AUTHOR(S):

Electronics Research Laboratory

7.2 DOCUMENT SERIES AND NUMBER

Electronics Research Laboratory 0085-TR

8 REFERENCE NUMBERS

a. Task: DST 76/240

b. Sponsoring Agency:

9 COST CODE:

347677/133

10 IMPRINT (Publishing organisation)

Defence Research Centre Salisbury

11 COMPUTER PROGRAM(S) (Title(s) and language(s))

12 RELEASE LIMITATIONS (of the document):

Approved for Public Release

12.0	OVERSEAS	NO		P.R.	1	A		B		C		D		E	

Security classification of this page: UNCLASSIFIED

Security classification of this page: UNCLASSIFIED

13 ANNOUNCEMENT LIMITATIONS (of the information on these pages):

No Limitation

14 DESCRIPTORS:

a. EJC Thesaurus Terms

Digital computers
Terrain
Terrain intelligence
Topography
Computer programs
Mapping
Data storage devices
Coordinates
Grid (coordinates)
Maps
Position (location)
Map projection

b. Non-Thesaurus Terms

Sonic digitizers

15 COSATI CODES:

0802

16 LIBRARY LOCATION CODES (for libraries listed in the distribution):

17 SUMMARY OR ABSTRACT:
(if this is security classified, the announcement of this report will be similarly classified)

A common method of representing the shape of the terrain, for use in various simulations or digital computer assisted studies, is to record the height of the terrain at regular intervals over a 2 dimensional grid and store this information in the computer. Manual interpolation of contour data from survey maps, to obtain the data for the grid, is a laborious and error-prone process.

This report describes an improved procedure for producing the terrain height grid. It involves a sonic digitizer to record a set of (X, Y) coordinates along the contour lines shown on the survey map. This information is stored in a disk file on a computer, and a two-stage cubic spline interpolation procedure is used to determine the terrain height at the intersection points of the nominated grid. The set of computer programs developed to implement the procedure with minimal storage requirements are described and listed.

Security classification of this page: UNCLASSIFIED