A SIMPLE CONTOURING PROGRAM FOR GRIDDED DATA

By Arlen W. Harbaugh

U.S. GEOLOGICAL SURVEY

Open-File Report 90-144

Reston, Virginia

1990

DEPARTMENT OF THE INTERIOR
MANUEL LUJAN, JR., Secretary

U.S. GEOLOGICAL SURVEY
Dallas L. Peck, Director

For additional information write to:

Chief,
Branch of Computer Technology
U.S. Geological Survey, MS 440
12201 Sunrise Valley Drive
Reston, VA 22092

Copies of this report can be purchased from:

U.S. Geological Survey
Books and Open-File Reports Section
Federal Center, Bldg. 810
Box 25425
Denver, Colorado 80225

CONTENTS

ILLUSTRATIONS

A SIMPLE CONTOURING PROGRAM FOR GRIDDED DATA

by Arlen W. Harbaugh

ABSTRACT

A computer program that contours a two-dimensional array of data has been developed. The purpose of this report is to document the contouring method and program design and to provide user instructions. The two-dimensional array contains Z values for each node in a grid, and the grid dimensions define the (X,Y) coordinates for each node. Branches connect adjacent nodes vertically and horizontally. A contour line is formed by connecting successive contour points with straight lines. Contour points occur only on branches. If the value being contoured is between the Z values at two adjacent nodes, the connecting branch contains a contour point. Linear interpolation is used to find the location of a contour point on a branch. The program is written in the FORTRAN programming language, and it uses Graphics Kernel System (GKS) subroutine calls.

INTRODUCTION

Many computer applications use large two-dimensional data arrays as input or produce them as ouput. It is often helpful to be able to graphically look at this data in the form of contour plots. A simple contouring program, named CONTOUR, was developed to contour such arrays. The program is written in the FORTRAN 77 language (ANSI, 1978), and Graphics Kernel System (GKS) (ANSI, 1985) subroutine calls are used to draw the contours. By today's standards, CONTOUR requires only limited computational resources; it can be run on small personal computers. The purpose of this report is to describe the contouring method and program design, and to provide user instructions.

CONTOUR is designed as a working tool where quick viewing is the primary need. Unlike more sophisticated contouring software, CONTOUR does not produce smoothed contours, fancy annotation, or publication quality; however, CONTOUR can efficiently and conveniently draw simple contours. Input can come from almost any fixed format file or from the unformatted files produced by the U.S. Geological Survey Modular Ground-Water Flow Model (McDonald and Harbaugh, 1988). CONTOUR can also draw the grid, blank out areas through which contours should not be drawn, and plot the data values at each grid point.

A major part of most commercial software is the gridding component, which creates gridded data from non-gridded data. CONTOUR works only with data that are already gridded, such as data produced and used by finite-difference simulation models. An advantage of CONTOUR, however, is that it directly contours data in unequally spaced grids without the need to convert the data to an equally-spaced grid, as commonly required by other contouring software.

FUNCTIONAL DESIGN OF CONTOUR

A contour plot is a way to display a three-dimensional surface using only two dimensions. On a plan view of a two-dimensional area, lines are drawn for which the surface has an equal value for the third dimension. The most common application is for elevation maps, where lines of equal elevation are shown. In CONTOUR, the two-dimensional area shown in plan view is defined by a grid as might be used in a finite-difference model (McDonald and Harbaugh). The grid consists of a rectangular area made of rows and columns of rectangular blocks called cells. Each cell is located by a row and column. Row one, column one is located in the upper, left corner. User specified distances between cell walls in the X direction, one-dimensional array DELX, and distances between cell walls in the Y direction, one-dimensional array DELY, define the locations of the cell walls. The center point of each cell is called a node, and values in the third dimension, Z, are defined at each node (figure 1). Nodes are designated by their column and row grid location. The location of each node in a two-dimensional coordinate system can be calculated using the known dimensions of grid cells and the assumption that nodes are in the center of cells. The Z values are known only at the nodes, but a continuous surface is assumed to pass through all the Z values. The Z values are stored in a two-dimensional array that is indexed by column and row to match the grid index. The Z values are read one row at a time starting with the first row.

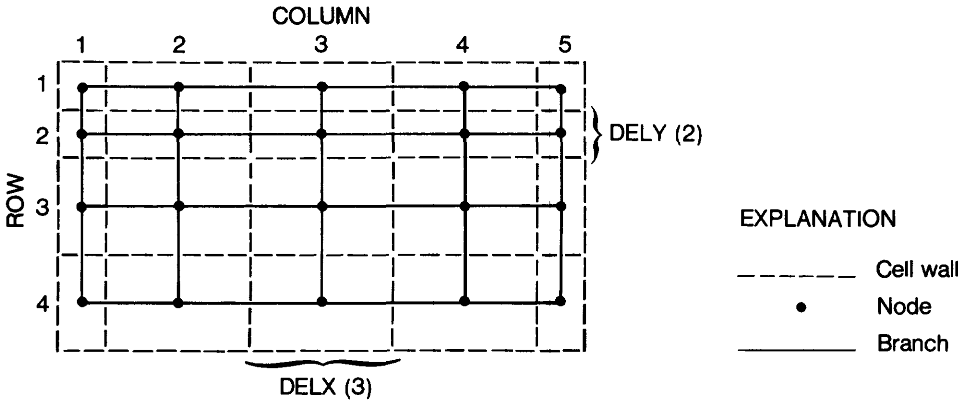

Figure 1. -- The grid that defines the area in which contours will be drawn.

Although the user defines the grid and associated Z values using standard matrix column and row notation, contouring is done in a Cartesian coordinate system where X and Y are the coordinates in the grid plane, and coordinate (0,0) is the lower, left corner. All user input dealing with rows and columns assumes that column 1, row 1 is at the upper left corner, but the data are then transformed into Cartesian coordinates immediately upon input. For example, DELY for the first row becomes DELY for the last row, and Z values for the first row become Z values for the last row. Other grid orientations could be handled by using a different transformation of the input data.

An additional grid component called a branch (figure 1) is an important part of the contouring method used by CONTOUR. Branches connect each pair of adjacent nodes in the horizontal and vertical directions. It is assumed that Z values vary linearly along branches. Thus the Z value at any point along a branch can be easily calculated. Although cell dimensions are used to define node locations, cells and cell walls are not directly involved in the contouring method.

Contours are constructed much like one would do by hand. For a given contour value, the branches are scanned to find one that contains the contour value. By linear interpolation, the (X,Y) coordinates of the point on the branch where Z is equal to the contour value are calculated. Once a starting point is found, the closest adjacent branches are scanned to see which way to continue the contour. There are always six adjacent branches to which a contour could connect (figure 2). If the current branch is a horizontal branch, there are three branches above (one to each side and one directly across) and three branches below. If the current branch is a vertical branch, there are three branches to the left and three branches to the right. After the second contour point has been found, the choices for the next point are reduced using knowledge of the contour direction. That is, it is assumed that a contour cannot cross the same rectangle of four connected branches twice in succession. For example, if the contour line has reached a horizontal branch, the contour direction will be either up (that is, coming from below the branch) or down. If the direction is up, there are only three possible branches to connect to (figure 3). Going to any of the three downward branches would cause a rectangle of four connected branches to be crossed twice in succession.

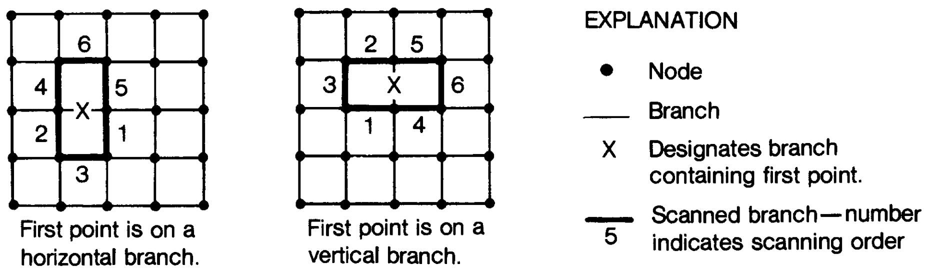
6
4
X
5
2
1
3
First point is on a horizontal branch.
2
5
3
X
6
1
4
First point is on a vertical branch.
EXPLANATION
● Node
Branch
X Designates branch containing first point.
5 Scanned branch—number indicates scanning order

Figure 2. -- The six branches that are checked to find the second contour point.

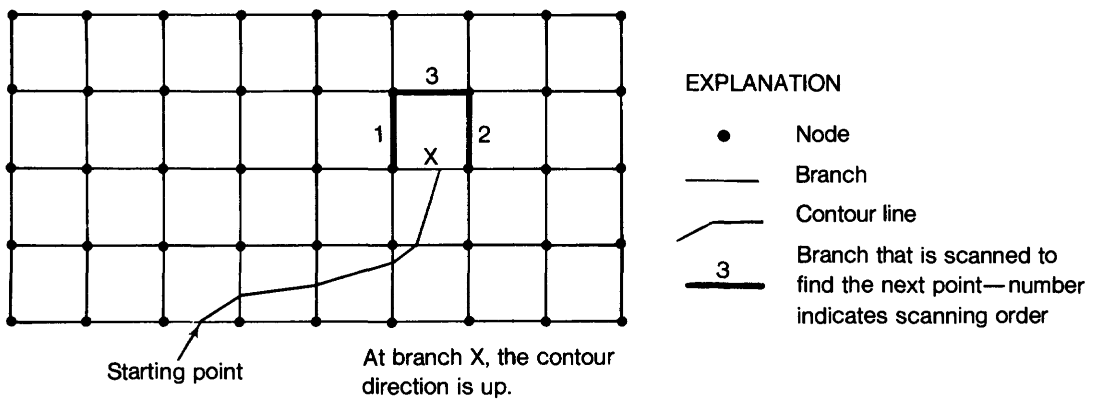
3
1
2
X
Starting point
At branch X, the contour direction is up.
EXPLANATION
Node
Branch
Contour line
Branch that is scanned to find the next point—number indicates scanning order

Figure 3. -- The three branches that are scanned to find successive points after contour direction is established.

A contour line is drawn by successively connecting points on branches with straight line segments. Thus, if the grid is coarse, the contours will not be smooth. This is actually considered an asset in the situation where the user is developing a model, which was the original application for CONTOUR. Rough-looking contours are an indication that the grid spacing is too coarse to represent the system being modeled.

In cases where the four connected branches that form a rectangle all contain the contour value, it is not apparent how to construct the contour. That is, there are multiple ways that a contour could go. For example, if the four branches labeled X, 1, 2, and 3 in figure 3 all contain the contour value, then the contour could conceivably go from branch X to branches 1, 2, and 3. This is where the surface modeling capabilities of a more sophisticated program would provide considerable improvement. Rather than perform complex calculations to make optimum contour decisions, very simple choices are made. Of the three possible branches, the two side branches (branches 1 and 2 in figure 3) are scanned first so that the contour line segment currently being drawn will not cross another contour segment that would eventually be drawn connecting the other two branches. Having contours cross would be an invalid physical interpretation in this situation. Instead, the contour line is drawn diagonally across such areas making the physical interpretation more realistic. No further effort is made to choose the best diagonal direction. That is, such a contour line could be drawn in either of two diagonal directions, but this choice is made arbitrarily. When data from a finite-difference model are being contoured, the unusual contours that sometimes result when there are multiple choices for the contour path are an indication that the grid is too coarse.

When a contour line ends, either by intersecting the edge of the grid or coming to an area where the user has specified that contours should not pass, the scanning of the branches continues in order to find starting points for other contour lines having the same contour value. The contouring process is completed for a contour value only when all branches have been scanned for the value and contour lines have been drawn to all branches that contain the value. Array IBRMAP is used to keep track of the branches through which a line has passed for a given contour value. Before starting a contour line on a branch or continuing a started contour line to a branch, IBRMAP is checked to prevent a contour line for a given value from passing through a branch more than once. As a contour line is extended to a branch, the IBRMAP value for that branch is set.

There is one element in IBRMAP for each node, and each element in IBRMAP contains information for two branches. IBRMAP(I,J), where I and J are the column and row respectively for a node in the grid, contains information for the branch to the right and the branch above node (I,J). If the branch to the right has been traversed by a contour, the first bit of IBRMAP is set. If the branch above has been traversed, the second bit is set. If only bit 1 is set, IBRMAP has a value of 1; if only bit 2 is set, IBRMAP has a value of 2; and if both bits are set, the value of IBRMAP is 3. IBRMAP also provides the mechanism that allows a user to specify areas through which contours should not pass. When this option is used, elements in IBRMAP corresponding to the branches that are to be avoided are set to appropriate values prior to drawing the lines for each contour value.

This contouring method works well as long as the points on a contour do not fall exactly on a node. If that happened, the method for scanning branches would have to be modified. Rather than adding additional scanning logic, CONTOUR prevents contours from falling exactly on a node. Before drawing the contours for a particular contour value, the Z values at all nodes are scanned. At each node where the Z value is equal to the contour value, the Z value is changed by adding a small amount to its value. The change does not significantly alter the location of the contour except that in some cases, it may cause a contour to completely disappear. For example, consider a grid where the outer rows and columns of nodes all have a Z value that is exactly 0, and all the interior nodes have positive values. If a 0 contour is requested, CONTOUR will change all the 0 values to a very small, but positive, number. Thus, there will be no branches on which a value of 0 will be interpolated, and no contour will be drawn. In this example, the user could specify a contour value that is slightly greater than 0 in order to cause a contour line to be drawn.

The program assumes that DELX and DELY values are specified in feet and converts these to real world inches; therefore, the plot generally requires scaling so that it will fit within the page size of the graphics device. (Other DELX and DELY units can be used by modifying the program variable GRDUNT.) The user can choose to use manual or automatic scaling. When using manual scaling, the user is prompted to enter a scaling factor to convert from real world inches to plotter inches. If, after manual scale selection, the plot is still too big for the display area of the graphics device, the plot will be automatically reduced (by Subroutine PLTBEG). If automatic scaling is selected, the program chooses a scale that will cause the plot to fill as much of the page of the graphics device as possible. The program does this scaling in a two step process. First, a scale is selected that will allow the plot to fit on an 8 by 8 inch page. The grid can occupy an 8 by 7 inch area, and an 8 by 1 inch area at the top of the plot is used for annotation. Subroutine PLTBEG then reduces or enlarges the 8 by 8 inch area to fill as much of the actual page size as possible. The prescaling to the 8 by 8 page size is done to avoid having to specify annotation text size and coordinate locations using real world coordinates. Because of the prescaling, plot annotation can be specified in scaled inches for both automatic and manual scaling.

PROGRAM DESIGN

To enhance portability and ease of modification, CONTOUR has a modular structure. There is a main program that only sets up array storage and calls the user-interaction routine, CON. Except in rare situations, the only array dimensions that a user might need to change are in the MAIN program. Subroutine CON prompts for options and controls the flow of the program. Many aspects of the user interface can be changed by modifying CON. Although CON controls the flow of the program, most of the work is done in small subroutines, each with a limited, independent purpose. This makes it easy to modify a program function by providing a replacement subroutine. The subroutines can be classified by three groups -- graphic primitives, contouring, and support.

The graphics primitive subroutines perform the basic graphics output. These routines start plotting (PLTBEG), end plotting (PLTEND), change color (PLTCOL), plot a string of characters (PLTSTR), and plot a line consisting of multiple connected points (PLTLIN). These are the only routines that directly call GKS subroutines; thus, to use non-GKS subroutines or to modify how GKS is used, these are the only subroutines that must be modified.

The support subroutines are supplemental to the actual process of contouring. Included are routines to prompt for a "yes" or "no" answer, input an array of data, draw the grid, plot data values at each cell, set up contour blanking, and label the plot. It is expected that many users might want to modify or replace the support subroutines to fit their needs. For example, the input routine might be changed to read data a different way.

The contouring subroutines are highly interdependent, and their only function is to draw the contours, including contour labels. These routines are the most complex and the least likely to require change; however, the contouring code was divided into fairly small subroutines that are easy to comprehend should change be necessary. The contouring process is initiated by calling a single subroutine, DRWCON. The contouring subroutines could be removed from CONTOUR and inserted into other programs. Having one integrated application that does contouring in combination with some other function, such as array generation, is likely to be more convenient to use than having a separate program for each function.

The program code contains many comments to aid users who want to understand the details of how the program works.

USER INSTRUCTIONS

The following is an interactive session using CONTOUR combined with explanations of each prompt. Figure 4 contains the DELX, DELY, and Z arrays that are used, and figure 5 shows the output from this interactive session.

File DELX.DATA:

```
 1500. 1000.  750.  500.  500.  500.  500.  500.  500.  500.
  500.  500.  500.  500.  500.  750. 1000. 1500.
```

File DELY.DATA:

```
 1000.  750.  500.  500.  500.  500.  500.  500.  500.  500.
  500.  500.  500.  500.  500.  500.  750. 1000.
```

File Z.DATA:

```
   0.000    0.655    1.205    1.674    2.083    2.444    2.766    3.057    3.321    3.561
   3.778    3.969    4.133    4.268    4.373    4.449    4.498    4.521
   0.000    0.665    1.220    1.694    2.105    2.467    2.789    3.081    3.346    3.589
   3.808    4.004    4.171    4.308    4.413    4.487    4.533    4.555
   0.000    0.683    1.252    1.734    2.149    2.512    2.834    3.126    3.394    3.642
   3.870    4.075    4.251    4.391    4.494    4.563    4.604    4.622
   0.000    0.712    1.301    1.795    2.214    2.577    2.898    3.191    3.464    3.721
   3.963    4.186    4.378    4.527    4.625    4.682    4.710    4.722
   0.000    0.751    1.366    1.876    2.300    2.660    2.978    3.270    3.549    3.821
   4.088    4.342    4.566    4.732    4.818    4.848    4.852    4.849
   0.000    0.798    1.448    1.977    2.402    2.755    3.065    3.355    3.641    3.934
   4.239    4.550    4.841    5.049    5.099    5.068    5.023    4.995
   0.000    0.851    1.541    2.094    2.514    2.851    3.147    3.432    3.727    4.046
   4.406    4.813    5.245    5.579    5.508    5.342    5.210    5.142
   0.000    0.898    1.631    2.216    2.617    2.928    3.207    3.486    3.788    4.134
   4.557    5.103    5.824    6.615    6.092    5.640    5.375    5.257
   0.000    0.920    1.681    2.306    2.670    2.958    3.225    3.500    3.804    4.164
   4.625    5.294    6.470    9.202    6.739    5.834    5.448    5.297
   0.000    0.894    1.623    2.204    2.603    2.912    3.189    3.466    3.766    4.112
   4.533    5.078    5.799    6.588    6.065    5.613    5.348    5.230
   0.000    0.842    1.524    2.071    2.486    2.818    3.110    3.392    3.684    4.001
   4.359    4.764    5.194    5.527    5.455    5.288    5.156    5.088
   0.000    0.784    1.423    1.942    2.360    2.706    3.010    3.295    3.577    3.866
   4.168    4.476    4.765    4.971    5.019    4.988    4.942    4.914
   0.000    0.732    1.332    1.829    2.242    2.594    2.904    3.189    3.463    3.730
   3.992    4.243    4.464    4.627    4.712    4.740    4.743    4.740
   0.000    0.688    1.257    1.735    2.141    2.493    2.805    3.089    3.355    3.606
   3.843    4.061    4.249    4.395    4.491    4.546    4.574    4.584
   0.000    0.654    1.199    1.661    2.060    2.409    2.720    3.003    3.263    3.503
   3.725    3.923    4.094    4.231    4.332    4.399    4.438    4.456
   0.000    0.629    1.155    1.606    1.998    2.344    2.654    2.934    3.190    3.424
   3.636    3.825    3.986    4.119    4.220    4.292    4.337    4.359
   0.000    0.612    1.127    1.570    1.957    2.301    2.609    2.888    3.141    3.371
   3.578    3.760    3.917    4.047    4.149    4.223    4.271    4.294
   0.000    0.604    1.113    1.552    1.937    2.279    2.586    2.864    3.116    3.344
   3.549    3.729    3.884    4.012    4.114    4.189    4.238    4.262
```

Figure 4. -- Input data used in the example interactive session.

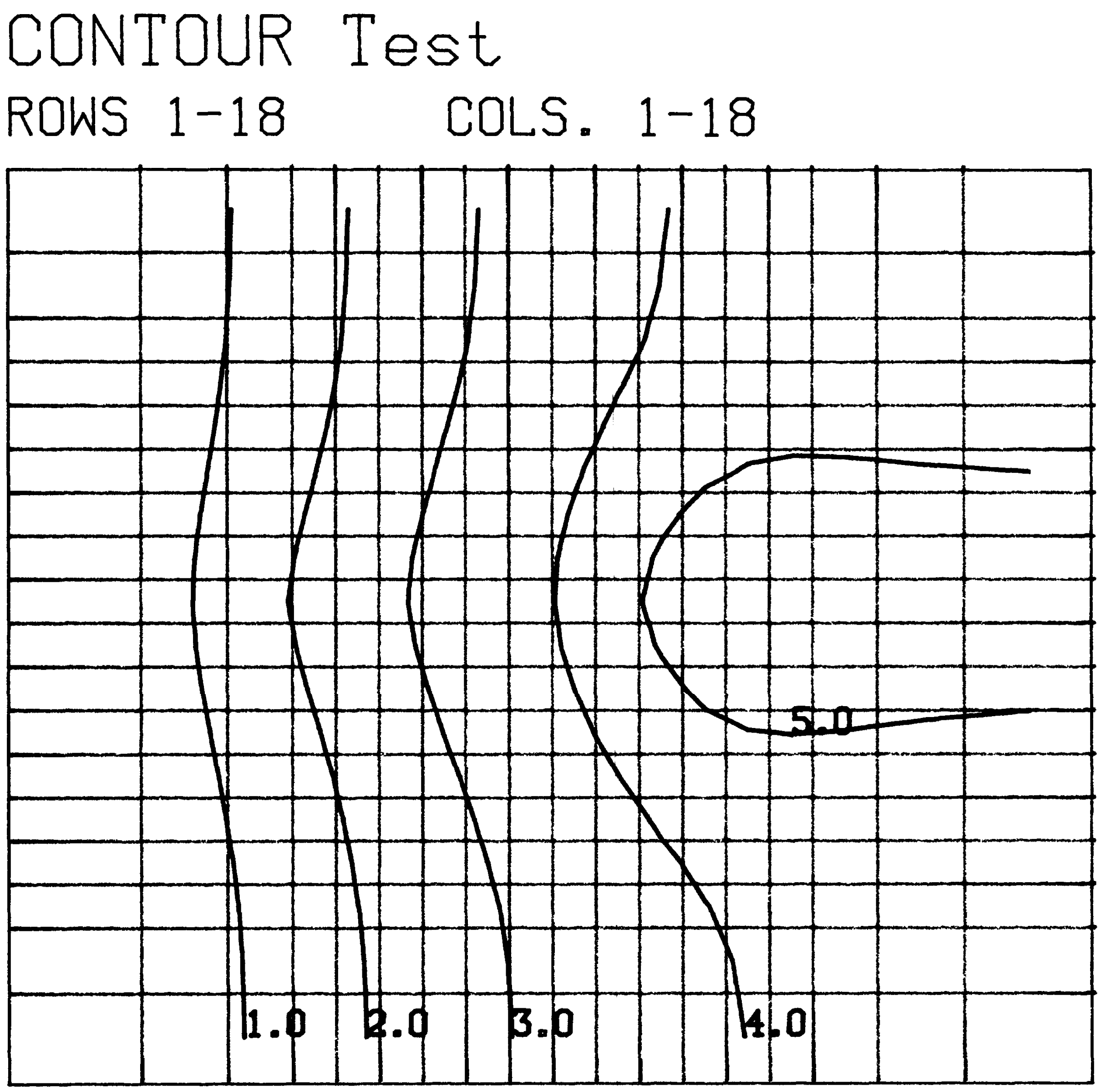

Figure 5. -- Example output from CONTOUR program.

```
ENTER THE NUMBER OF ROWS, NUMBER OF COLUMNS:
18,18
```

Grid dimensions must be entered here. The two values can be separated by either a blank or a comma.

```
ENTER NAME OF FILE CONTAINING DELX VALUES (CR DEFAULTS TO 1 INCH SPACING):
DELX.DATA
```

Distance between cell walls in the X direction (DELX) can be read from a file, or the spacing can default to one inch. The default is useful when the grid has equal spacing and when there is no need to have the contours drawn to a specific map scale. DELX values must be input from a file if grid spacing is unequal or if the plot will be overlaid on a scaled map. If DELX is defaulted, then DELY is also defaulted.

```
 ENTER FORMAT OF DATA  -- FOR EXAMPLE: (15F8.0)
  OR ENTER CR FOR UNFORMATTED MODULAR MODEL DATA:
(10F6.0)
```

To avoid the need to change the program when different data formats are needed, the user is prompted to enter the data format. The format is specified using standard FORTRAN rules. The unformatted option is not likely to be useful when reading grid spacing; this option was incorporated into the input routine for use when reading Z values.

```
  ENTER LINE NUMBER OF START OF DATA, MULTIPLIER:
1,1
```

The starting line number allows data to be read from the middle of a file. This makes it possible to read an output file generated by another program, which might contain some data preceding the data required by CONTOUR. The multiplier scales the data that are input. Use a multiplier of 1.0 to avoid changing the data.

```
 ENTER NAME OF FILE CONTAINING DELY VALUES:
DELY.DATA


  ENTER FORMAT OF DATA  -- FOR EXAMPLE: (15F8.0)
  OR ENTER CR FOR UNFORMATTED MODULAR MODEL DATA:
(10F6.0)
  ENTER LINE NUMBER OF START OF DATA, MULTIPLIER:
1,1

 DO YOU WANT TO CONSIDER THE ENTIRE GRID(Y/N)?
Y
```

The user can contour a subarea of the grid if desired. If the prompt is answered with "Y", the user is prompted to enter row and column ranges.

```
 DO YOU WANT THE PLOT AUTOMATICALLY SCALED(Y/N)?
N
```

If a "Y" response is given, the plot will be automatically scaled to fill as much of the page of the graphics device as possible. Automatic scaling is more convenient than manual scaling and is generally the option to use with video display devices. Manual scaling allows the user to produce a plot to a specified scale. This makes it possible to produce a contour plot that can be overlaid on a scaled map. Manual scaling can also be used to maintain a constant scale when contour plots are being produced on many different graphics devices that have different page sizes. If automatic scaling is used, the following prompts for scaling information will not be issued.

```
  THE GRID LENGTH (X) AND WIDTH (Y) ARE (INCHES):    150000.
     126000.
  ENTER MAP SCALE DIVISOR (E.G. IF SCALE IS 1:24000, ENTER 24000):
24000
```

Grid spacing is divided by the scale divisor, which provides a mechanism for the user to reduce the plot to a specific map scale. If the plot size is left larger than will fit within the size of a page on the graphics device being used; the plot will be automatically reduced to fit, and a message will be written into the GKS.ERR file. Note also that 1.0 inch is required at the top of the plot for labeling; the grid cannot extend into this area.

```
  THE SCALED GRID LENGTH (X) AND WIDTH (Y) ARE (INCHES):    6.25000
     5.25000

 IS THIS SIZE ACCEPTABLE(Y/N)?
Y
```

This prompt gives the user a chance to respecify the scale if the plot is bigger than the page size of the graphics device.

```
  ENTER DATA FILE NAME:
Z.DATA
```

This data file contains the two-dimensional array of Z data to be contoured.

```
  ENTER FORMAT OF DATA  -- FOR EXAMPLE: (15F8.0)
  OR ENTER CR FOR UNFORMATTED MODULAR MODEL DATA:
(10F8.0)
```

As with the input of DELX and DELY, the format of the data is user specified. Unformatted head or drawdown data from the U.S. Geological Survey Modular Ground-Water Flow Model (McDonald and Harbaugh) is assumed if the format is left blank. These unformatted files can contain multiple two-dimensional arrays (data sets), each containing head or drawdown for one model layer at a specific simulation time. If the format is left blank, the user is prompted for the number of the desired data set.

```
  ENTER LINE NUMBER OF START OF DATA, MULTIPLIER:
1,1

 DO YOU WANT TO USE A BOUNDARY MASK(Y/N)?
N
```

This option allows the user to specify areas of the grid through which contours will not pass -- that is, a blanking option. This is useful, for example, for no-flow zones in a ground-water flow model. If a "Y" answer is given, the user is prompted for a Z data value that indicates that a node is to be blanked out. No contour lines will be drawn to any branch that connects to a node having the specified value.

```
  YOUR MINIMUM AND MAXIMUM DATA VALUES ARE:   0.000000
     9.20200
  ENTER THE NUMBER OF VALUES TO CONTOUR :
5
```

Up to 25 values can be contoured. Zero values can also be contoured, which is not an absurd possibility. A user might want to draw only the grid without any contours. The range of data values is displayed as an aid to choosing how many contours to request.

```
  ENTER THE  5 CONTOUR VALUES
1,2,3,4,5
```

These are the values to be contoured. Although in this example the interval between contours is constant, any contour values can be specified.

```
 DO YOU WANT THE GRID DRAWN(Y/N)?
Y
```

This option causes the grid to be drawn. A block-centered grid is assumed.

```
 DO YOU WANT EACH DATA VALUE PLOTTED(Y/N)?
N
```

This option causes the data value at each node to be plotted. If a "Y" answer is given, the user is prompted for the number of digits to include after the decimal point. If "-1" is entered for the number of digits, there will be no decimal point. Plotting the data values can take a long time on a slow pen plotter. The plotted numbers will overlap each other if nodes are close together; the program makes no attempt to check if there is adequate room.

```
  ENTER PLOT TITLE:
CONTOUR Test
```

Any title up to 80 characters may be entered, but only those characters that fit on the plot page will be plotted. In most situations, at least 30 characters can be plotted.

```
 ENTER WORKSTATION TYPE:
1
```

When using GKS, the graphics device (workstation type) is specified by a number. Each number is linked to a specific graphics device by the GKS software. The possible device numbers and what devices they represent depend on the computer system and GKS software being used.

After the workstation type is entered, the plot is generated.

```
  PLOTTING HAS ENDED
**** STOP
```

When the plot is done, the program generally pauses until a carriage return is entered before continuing to completion. This depends on the graphics device. The program will stop after the carriage return is entered.

REFERENCES CITED

American National Standards Institute, 1978, Programming language FORTRAN: American National Standards Institute, X3.9-1978, 18 ch.

American National Standards Institute, 1985, Computer Graphics - Graphics Kernel System (GKS) functional description, American National Standards Institute, X3.124-1985, 268 p.

McDonald, M.G. and Harbaugh, A.W., 1988, A modular three-dimensional finite-difference ground-water flow model: U.S. Geological Survey Techniques of Water Resources Investigations, Book 6, Chapter A1, 14 ch.

APPENDIX

PROGRAM LISTING

Main program and Subroutine CON:

```
C  CONTOUR --  A Simple Contouring Program for Gridded Data
C    *****************************************************************
C          Documented in U.S. Geological Survey Open-File Report 90-144,
C             titled "A Simple Contouring Program for Gridded Data"
C                                             by Arlen W. Harbaugh
C    *****************************************************************
C
C    THIS MAIN PROGRAM EXISTS FOR THE PURPOSE OF ALLOWING RUN TIME ALLOCATION
C    OF MEMORY.  RATHER THAN HAVING TO DIMENSION A NUMBER OF ARRAYS TO BE THE
C    CORRECT SIZE IN 2 AND 1 DIMENSIONS, A USER NEED DIMENSION ONLY TWO
C    ARRAYS, NAMED X AND IX, IN 1 DIMENSION.  SPACE WITHIN X AND IX IS
C    AUTOMATICALLY DIVIDED INTO THE REQUIRED ARRAYS.  OF COURSE X AND IX MUST
C    BE DIMENSIONED LARGE ENOUGH TO HOLD THE DATA.  IF NOT, AN ERROR MESSAGE
C    IS PRINTED, TELLING THE USER WHAT DIMENSION SIZE TO USE.  THIS METHOD
C    MINIMIZES THE NEED FOR REDIMENSIONING.
C    X MUST CONTAIN NX*NY + 2*NX + 2*NY ELEMENTS
C    IX MUST CONTAIN 2*NX*NY ELEMENTS
C
C    IX IS DECLARED INTEGER*2, WHICH CUTS THE MEMORY REQUIRED FOR IX IN HALF
C    COMPARED TO THE DEFAULT OF INTEGER*4 ON MOST SYSTEMS.  IF YOUR
C    SYSTEM DOES NOT SUPPORT INTEGER*2, DELETE THE INTEGER*2 LINES IN
C    ALL SUBROUTINES.
*****************************************************************************
      INTEGER*2 IX
      DIMENSION X(10000), IX(18000)
C-----LENX MUST BE SET TO THE DIMENSIONED SIZE OF X
C-----LENIX MUST BE SET TO THE DIMENSIONED SIZE OF IX
      LENX=10000
      LENIX=18000
C
10    WRITE(*,*) ' ENTER THE NUMBER OF ROWS, NUMBER OF COLUMNS:'
      READ(*,*) NY,NX
      IF(NX.LT.1 .OR. NY.LT.1) GO TO 10
C
      LCZ=1
      LCDELX=LCZ+NX*NY
      LCDELY=LCDELX+NX
      LCXLOC=LCDELY+NY
      LCYLOC=LCXLOC+NX
      ISUMX=LCYLOC+NY-1
C
C-----INTEGER ARRAYS
      LCIBRM=1
      LCIOUT=LCIBRM+NX*NY
      ISUMIX=LCIOUT+NX*NY-1
```

```
C
C-----CHECK IF X IS DIMENSIONED BIG ENOUGH
      IF(ISUMX.LE.LENX) GO TO 50
      WRITE(*,11) ISUMX,LENX
11    FORMAT(1X,I7,' ELEMENTS IN THE X ARRAY HAVE BEEN USED OUT OF',I7)
      WRITE(*,*) '   ***X ARRAY MUST BE DIMENSIONED LARGER***'
C
C-----CHECK IF IX IS DIMENSIONED BIG ENOUGH
50    IF(ISUMIX.LE.LENIX) GO TO 90
      WRITE(*,12) ISUMIX,LENIX
12    FORMAT(1X,I7,' ELEMENTS IN THE IX ARRAY HAVE BEEN USED OUT OF',I7)
      WRITE(*,*) '   ***IX ARRAY MUST BE DIMENSIONED LARGER***'
      STOP
C
90    IF(ISUMX.GT.LENX) STOP
C
C-----PASS ARRAYS TO THE CONTOURING ROUTINES
100   CALL CON(X(LCZ),X(LCDELX),X(LCDELY),X(LCXLOC),X(LCYLOC),
     1         IX(LCIBRM),IX(LCIOUT),NX,NY)
      STOP
      END
      SUBROUTINE CON(Z,DELX,DELY,XLOC,YLOC,IBRMAP,IOUT,NX,NY)
C     **************************************************************************
C     INTERACTIVE PART OF CONTOUR PROGRAM -- READ DATA, ASK FOR
C     OPTIONS, ETC.  THEN CALL THE ROUTINES THAT DRAW THE CONTOURS
C     AND DO THE OPTIONS.  THIS ROUTINE CHANGES FROM A COORDINATE SYSTEM
C     IN WHICH ROW 1, COLUMN 1 IS IN THE UPPER LEFT HAND CORNER TO A CARTESIAN
C     COORDINATE SYSTEM.
C     **************************************************************************
      INTEGER*2 IBRMAP,IOUT
      DIMENSION Z(NX,NY), DELX(NX), DELY(NY), XLOC(NX), YLOC(NY)
      DIMENSION IBRMAP(NX,NY), CLEV(25), IOUT(NX,NY)
      CHARACTER*80 TITLE
      CHARACTER*132 NAMIN
C
C     DATA DEFINITION AND REQUIRED DIMENSIONS OF ARRAYS
C     *************************************************************************
C     NX IS # OF NODES IN X DIRECTION
C     NY IS # OF NODES IN Y DIRECTION
C     DELX AND DELY ARE BLOCK SIZES -- ONE VALUE IN DELY FOR EACH GRID ROW AND
C          ONE VALUE IN DELX FOR EACH GRID COLUMN -- ROW ONE IS THE TOP ROW,
C          COLUMN 1 IS THE LEFTMOST COLUMN -- DIMENSION DELX(NX) , DELY(NY)
C     XLOC AND YLOC ARE NODE LOCATIONS IN CARTESIAN COORDINATES ( NODES ARE
C          CENTERED IN BLOCKS ) -- DIMENSION XLOC(NX), DELY(NY)
C     Z MATRIX CONTAINS Z VALUES FOR EACH NODE -- DIMENSION Z(NX,NY)
C     IBRMAP IS  A WORK AREA USED TO STORE GRID BRANCHES THAT HAVE A CONTOUR
C          DRAWN THROUGH THEM - INTEGER*2 IBRMAP(NX,NY)
C     IOUT IS USED TO PREVENT CONTOURS FROM GOING INTO SPECIFIED AREAS --
C          INTEGER*2 IOUT(NX,NY)
C     CLEV STORES THE CONTOUR VALUES - DIMENSION CLEV(25)
C     NLEVS IS THE NUMBER OF CONTOUR VALUES BEING USED
```

```
C     TITLE CONTAINS UP TO 80 CHARACTERS FOR THE PLOT THE TITLE
C     IXF,IXL,IYF,IYL MAY BE SET TO ANY RECTANGULAR SUBAREA
C           WITHIN THE BOUND OF NX AND NY IF YOU DON'T WANT TO
C           USE THE ENTIRE GRID
C     SCALE IS MAP SCALE (IE INCHES OF REAL WORLD PER 1 INCH OF MAP)
C     GRDUNT IS THE NUMBER OF INCHES IN THE UNIT THAT IS USED TO
C           SPECIFY DELX AND DELY -- E.G. 12 IF THE UNIT USED IS FEET
C     UPI IS THE NUMBER OF UNITS USED FOR SPECIFYING DELX AND DELY
C           PER PLOTTER INCH -- I.E. DIVIDE A REAL WORLD COORDINATE BY
C           UPI TP GET A COORDINATE IN PLOTTER INCHES
C     IFILL IS A FILL FLAG TO DETERMINE HOW THE PLOT IS ADJUSTED TO FIT
C           ON THE PLOTTING DEVICE.  IF THE FLAG IS NOT 0, THE PLOT IS
C           EXPANDED OR REDUCED TO FILL THE PAGE.  IF THE FLAG IS 0, THE
C           PLOT SIZE IS NOT ADJUSTED PROVIDED THAT IT WILL FIT ON THE
C           PAGE.
C     ILABEL IS A FLAG TO DETERMINE IF CONTOUR LABELS ARE PLOTTED.  IF
C            0, LABELS ARE NOT PLOTTED
C   ***************************************************************
C
C-----DELX AND DELY HAVE UNITS OF FEET
      GRDUNT=12.
C
C-----READ IN DELX AND DELY (DELR AND DELC IN USGS MODULAR GROUND-WATER
C-----FLOW MODEL)
C-----PROMPT FOR A FILE NAME
      WRITE(*,*)
50    WRITE(*,51)
51    FORMAT(' ENTER NAME OF FILE CONTAINING DELX VALUES',
     1                 ' (CR DEFAULTS TO 1 INCH SPACING):')
      READ(*,53) NAMIN
53    FORMAT(A)
C-----IF NO FILE NAME IS GIVEN FOR DELX, THEN
C-----USE DEFAULT SPACING OF 1 REAL INCH PER GRID CELL, WHICH MEANS THAT
C-----A SCALE OF 1 WILL PRODUCE A PLOT NX BY NY INCHES
      IF(NAMIN.EQ.' ') THEN
         WRITE(*,*) ' USING DEFAULT OF 1 INCH FOR DELX AND DELY'
         DO 100 I=1,NX
         DELX(I)=1./GRDUNT
100      CONTINUE
         DO 110 I=1,NY
         DELY(I)=1./GRDUNT
110      CONTINUE
      ELSE
C-----IF FILE NAME WAS ENTERED, READ THE DATA
         CALL DREAD(7,NAMIN,DELX,NX,1,IERR)
         IF(IERR.NE.0) GO TO 50
120      WRITE(*,*) ' ENTER NAME OF FILE CONTAINING DELY VALUES:'
         READ(*,53) NAMIN
         CALL DREAD(7,NAMIN,DELY,NY,1,IERR)
         IF(IERR.NE.0) GO TO 120
      END IF
C
C-----REVERSE DELY SO THAT WE'RE USING A CARTESIAN SYSTEM
      DO 150 J=1,NY/2
      I=NY-J+1
      TEMP=DELY(J)
      DELY(J)=DELY(I)
      DELY(I)=TEMP
150   CONTINUE
C
C
C-----RESTRICT THE GRID IF DESIRED
      IXF=1
      IXL=NX
      IYF=1
      IYL=NY
      IYGRDF=1
      IYGRDL=NY
      CALL YESNO('DO YOU WANT TO CONSIDER THE ENTIRE GRID?',I)
      IF(I.EQ.1) GO TO 300
220   WRITE(*,*) ' ENTER 1ST ROW, LAST ROW :'
      READ(*,*) IYGRDF,IYGRDL
      IF((IYGRDL.GT.NY).OR.(IYGRDF.LT.1).OR.(IYGRDF.GT.IYGRDL))GO TO 220
240   WRITE(*,*) ' ENTER 1ST COL., LAST COL. :'
      READ(*,*) IXF,IXL
      IF((IXL.GT.NX).OR.(IXF.LT.1).OR.(IXF.GT.IXL)) GO TO 240
C-----REVERSE Y GRID VALUES SO THAT ROW 1 IS BOTTOM RATHER THAN THE TOP
      IYL=NY-IYGRDF+1
      IYF=NY-IYGRDL+1
C
C-----DEFINE A SCALE FOR CONVERTING TO PLOTTER INCHES
C-----FIND GRID SIZE IN REAL WORLD COORDINATES
300   UPI=1./GRDUNT
      CALL NODLOC(DELX,DELY,XLOC,YLOC,UPI,XTOT,YTOT,NX,NY,IXF,IXL,IYF,
     1            IYL)
C
C-----ASK IF AUTOMATIC SCALING IS DESIRED
      CALL YESNO('DO YOU WANT THE PLOT AUTOMATICALLY SCALED?',IFILL)
      IF(IFILL.NE.0) THEN
C
C-----FOR AUTOMATIC SCALING, SET SCALE SO THAT THE PLOT (INCLUDING 1
C-----INCH AT THE TOP FOR ANNOTATION) FITS ON AN 8 X 8 INCH PAGE.
C-----PLTBEG WILL FURTHER FIT THE PLOT TO THE ACTUAL PLOTTER PAGE SIZE.
         UPI=XTOT/8./GRDUNT
         IF(XTOT/YTOT .LT. 1.143) UPI=YTOT/7./GRDUNT
      ELSE
C
C-----FOR MANUAL SCALING, PROMPT FOR A MAP SCALE
         WRITE(*,*)
         WRITE(*,*) ' THE GRID LENGTH (X) AND WIDTH (Y) ARE (INCHES):',
     1               XTOT,YTOT
```

```
320        WRITE(*,*) ' ENTER MAP SCALE DIVISOR (E.G. IF SCALE IS',
     1                ' 1:24000, ENTER 24000): '
           READ(*,*) SCALE
           WRITE(*,*)
           WRITE(*,*) ' THE SCALED GRID LENGTH (X) AND WIDTH (Y) ARE',
     1                ' (INCHES):',XTOT/SCALE,YTOT/SCALE
           CALL YESNO('IS THIS SIZE ACCEPTABLE?',I)
           IF(I.EQ.0) GO TO 320
           UPI=SCALE/GRDUNT
      END IF
C
C-----CALCULATE THE LOCATION OF NODES IN PLOT INCHES
      CALL NODLOC(DELX,DELY,XLOC,YLOC,UPI,XMAX,YMAX,NX,NY,IXF,IXL,IYF,
     1            IYL)
C
C-----READ Z VALUES
C-----MODIFY THIS SECTION TO MATCH THE FORMAT OF YOUR PARTICULAR DATA.
C-----THE DATA SHOULD BE ARRANGED IN Z SO THAT THE LOWER LEFT HAND CORNER
C-----IS THE ORIGIN.
      WRITE(*,*)
500   WRITE(*,*) ' ENTER DATA FILE NAME:'
      READ(*,501) NAMIN
501   FORMAT(A)
      CALL DREAD(7,NAMIN,Z,NX,NY,IERR)
      IF(IERR.NE.0) GO TO 500
C-----EXCHANGE ROWS TO GET A CARTESIAN SYSTEM
      DO 550 J=1,NY/2
      K=NY-J+1
      DO 550 I=1,NX
      TEMP=Z(I,J)
      Z(I,J)=Z(I,K)
      Z(I,K)=TEMP
550   CONTINUE
C
C-----SPECIFY AREA TO IGNORE -- I.E. THE AREA TO BE BLANKED
      CALL YESNO('DO YOU WANT TO USE A BOUNDARY MASK?',IMASK)
      IF(IMASK.EQ.0) GO TO 555
      WRITE(*,*) ' ENTER DATA VALUE THAT INDICATES OUTSIDE OF BOUNDARY:'
      READ(*,*) BOGUS
C
C-----READ IN CONTOUR LEVELS
C-----BUT BEFORE DOING THIS, TELL THE USER WHAT THE RANGE OF THE DATA
C-----IS AS AN AID TO CHOOSING THE CONTOUR LEVELS
555   CALL ZRANGE(Z,NX,NY,IXF,IXL,IYF,IYL,IMASK,BOGUS,ZMIN,ZMAX)
      WRITE(*,*)
      WRITE(*,*) ' YOUR MINIMUM AND MAXIMUM DATA VALUES ARE:',
     1                  ZMIN,ZMAX
560   WRITE(*,*) ' ENTER THE NUMBER OF VALUES TO CONTOUR:'
      READ(*,*) NLEVS
      IF(NLEVS.GT.25) GO TO 560
      IF(NLEVS.LE.0) GO TO 600
      WRITE(*,570) NLEVS
```

```
570   FORMAT(1X,' ENTER THE',I3,' CONTOUR VALUES')
      READ(*,*) (CLEV(I),I=1,NLEVS)
C
C-----ASK ABOUT ALL OPTIONS BEFORE PLOTTING STARTS
600   CALL YESNO('DO YOU WANT THE GRID DRAWN?',IGRID)
C
680   CALL YESNO('DO YOU WANT EACH DATA VALUE PLOTTED?',IDPLT)
      IF(IDPLT.EQ.0) GO TO 700
      WRITE(*,*) ' ENTER NUMBER OF DIGITS TO PLOT AFTER DECIMAL POINT:'
      READ(*,*) NDEC
C
C-----READ IN A TITLE
700   WRITE(*,*)
      WRITE(*,701)
701   FORMAT(1X,' ENTER PLOT TITLE:')
      READ(*,702) TITLE
702   FORMAT(A)
C
C
C-----INITIALIZE PLOTTING
      CALL PLTBEG(XMAX,YMAX+1.0,IFILL)
C
C-----DRAW THE GRID IF THAT OPTION WAS SELECTED
      CALL PLTCOL(4)
      IF(IGRID.EQ.1) CALL GRID(DELX,DELY,NX,NY,UPI,XMAX,YMAX,IXF,IXL,
     1                         IYF,IYL)
C
C  DRAW SURROUNDING BOX AND LABEL PLOT
      CALL PLTCOL(1)
      CALL ANOTE(XMAX,YMAX,TITLE,IXF,IXL,IYGRDF,IYGRDL)
C
C-----INITIALIZE BLANKING ARRAY SO THAT THE ENTIRE AREA MAY HAVE
C-----CONTOURS CROSSING
      DO 820 J=1,NY
      DO 820 I=1,NX
820   IOUT(I,J)=0
C
C-----NOW SET BOUNDARY ARRAY IF THAT OPTION WAS SELECTED
      IF(IMASK.EQ.1 .AND. NLEVS.GT.0) CALL BOUND(Z,IOUT,NX,NY,BOGUS)
C
C-----DRAW CONTOURS UNLESS THERE WERE NO CONTOUR LEVELS SPECIFIED
      CALL PLTCOL(2)
      ILABEL=1
      IF(NLEVS.GT.0) CALL DRWCON(NX,NY,Z,XLOC,YLOC,CLEV,NLEVS,
     1       IBRMAP,IOUT,IXF,IXL,IYF,IYL,ILABEL)
C
C-----PLOT THE VALUE AT EACH NODE IF THAT OPTION WAS SELECTED.
C-----     NDEC IS NUMBER OF DECIMAL PLACES FOLLOWING DECIMAL POINT
C-----     HEIGHT IS THE HEIGHT OF THE SYMBOLS
C-----     IMASK VALUE OF 1 INDICATES THAT VALUES IN BLANKED OUT AREAS
C-----            SHOULD BE LEFT BLANK
C-----     BOGUS IS DATA VALUE INDICATING A BOUNDARY AREA
```

```
900   CALL PLTCOL(3)
      IF(IDPLT.EQ.1) CALL DATAPL(NX,NY,Z,XLOC,YLOC,NDEC,.10,IMASK,
     1           BOGUS,IXF,IXL,IYF,IYL)
C
C-----END OF PLOT
1000  CALL PLTEND
      WRITE(*,*)
      WRITE(*,*) ' PLOTTING HAS ENDED'
      RETURN
      END
```

Support subroutines:

```
      SUBROUTINE YESNO(MES,IANS)
C    *******************************************************************
C    ROUTINE TO GET YES OR NO RESPONSE TO A QUESTION
C    *******************************************************************
      CHARACTER*80 MES
      CHARACTER*1 IA
      DO 10 I=1,80
      IF(MES(I:I).EQ.'?') GO TO 20
10    CONTINUE
20    I=I-1
30    WRITE(*,2)
2     FORMAT(1X)
      WRITE(*,4) MES(1:I),'(Y/N)? '
4     FORMAT(1X,A,A)
      READ(*,3) IA
3     FORMAT(A)
      IANS=-1
      IF(IA.EQ.'Y' .OR. IA.EQ.'y') IANS=1
      IF(IA.EQ.'N' .OR. IA.EQ.'n') IANS=0
      IF(IANS.EQ.-1) GO TO 30
      RETURN
      END
      SUBROUTINE DREAD(IUNIT,NAMIN,D,NX,NY,IERR)
C    *******************************************************************
C    READ A 2-DIMENSIONAL ARRAY
C    *******************************************************************
      DIMENSION D(NX,NY)
      CHARACTER*16 TEXT
      CHARACTER*132 FMT,NAMIN
      INTEGER*4 KSTP,KPER,NCOL,NROW,ILAY
C
      IERR=0
C
C-----INPUT THE FORMAT
      WRITE(*,*) ' ENTER FORMAT OF DATA  -- FOR EXAMPLE: (15F8.0)'
      WRITE(*,*) ' OR ENTER CR FOR UNFORMATTED MODULAR MODEL DATA:'
      READ(*,51) FMT
51    FORMAT(A)
C
C-----IF FORMAT IS BLANK, USE AN UNFORMATTED READ
      IF(FMT.EQ.' ') THEN
         OPEN(IUNIT,FILE=NAMIN,STATUS='OLD',FORM='UNFORMATTED',
     1         ACCESS='SEQUENTIAL',ERR=110)
C
C-----READ THE UNFORMATTED FILE
C-----THERE MAY BE MORE THAN 1 DATA SET IN THE SAME FILE -- FIND OUT
C-----WHICH ONE IS DESIRED.  ALSO GET A MULTIPLIER THAT CAN BE USED
```

```
C-----TO SCALE THE DATA
         WRITE(*,*) ' ENTER THE DATA SET NUMBER, MULTIPLIER:'
         READ(*,*) NREAD,AMULT
         IF(NREAD.LT.1) NREAD=1
         DO 5 I=1,NREAD
         READ(IUNIT,ERR=100) KSTP,KPER,PERTIM,TOTIM,TEXT,NCOL,NROW,ILAY
         IF(I.EQ.NREAD) THEN
            WRITE(*,15) TEXT,ILAY,KSTP,KPER
15          FORMAT(1X,A,' IN LAYER',I3,' AT END OF TIME STEP',I3,
     1                                 ' IN STRESS PERIOD',I3)
         END IF
C-----MAKE SURE THAT NROW AND NCOL MATCH NX AND NY
         IF(NX.NE.NCOL .OR. NY.NE.NROW) THEN
            WRITE(*,*) ' GRID SIZE CONFLICT:'
            WRITE(*,17) NX,NY,NCOL,NROW
17          FORMAT(' THE GRID SIZE YOU SPECIFIED AT THE START IS',2I5/
     1             ' THE GRID SIZE SPECIFIED IN THE DATA FILE IS',2I5)
            GO TO 100
         END IF
C-----NOW READ THE ENTIRE ARRAY WITH 1 STATEMENT
         READ(IUNIT,ERR=100) ((D(K,L),K=1,NX),L=1,NY)
5        CONTINUE
C
      ELSE
C
C-----OPEN A FORMATTED FILE
         OPEN(IUNIT,FILE=NAMIN,STATUS='OLD',FORM='FORMATTED',
     1        ACCESS='SEQUENTIAL',ERR=110)
C
C-----READ THE FORMATTED FILE
C-----BY SPACIFYING THE START LINE AND FORMAT (FORMAT HAS ALREADY BEEN
C-----READ), DATA CAN BE EXTRACTED FROM MOST ANY LISTING FILE.  ALSO, A
C-----MULTIPLIER CAN BE USED TO SCALE THE DATA.
         WRITE(*,*) ' ENTER LINE NUMBER OF START OF DATA, MULTIPLIER:'
         READ(*,*) NSKIP,AMULT
         NSKIP=NSKIP-1
         IF(NSKIP.LE.0) GO TO 31
         DO 33 I=1,NSKIP
         READ(IUNIT,29,ERR=100)
29       FORMAT(1X)
33       CONTINUE
31       DO 30 M=1,NY
         READ(IUNIT,FMT,ERR=100) (D(L,M),L=1,NX)
30       CONTINUE
      END IF
C
C-----SCALE THE DATA
      DO 32 I=1,NY
      DO 32 J=1,NX
      D(J,I)=D(J,I)*AMULT
32    CONTINUE
C
```

```
      CLOSE(IUNIT)
      RETURN
C
C-----ERROR WHILE READING
100   CLOSE(IUNIT)
      WRITE(*,*) ' ERROR WILE READING THE FILE'
      IERR=1
      RETURN
C-----ERROR WHILE OPENING
110   WRITE(*,*) ' ERROR WHEN TRYING TO OPEN THE FILE'
      IERR=1
      RETURN
      END
      SUBROUTINE ANOTE(XMAX,YMAX,TITLE,IXF,IXL,IYGRDF,IYGRDL)
C    ********************************************************************
C    THIS SUBROUTINE DRAWS THE GRID BORDER AND PLOT ANNOTATION
C    ********************************************************************
      CHARACTER*80 TITLE
      CHARACTER*40 LABEL
      CHARACTER*20 CROW1,CROW2,CCOL1,CCOL2
      DIMENSION XB(5),YB(5)
C
C-----DRAW GRID BORDER
      XB(1)=0.
      YB(1)=0.
      XB(2)=XMAX
      YB(2)=0.
      XB(3)=XMAX
      YB(3)=YMAX
      XB(4)=0.
      YB(4)=YMAX
      XB(5)=0.
      YB(5)=0.
      CALL PLTLIN(XB,YB,5)
C
C-----GENERATE # OF ROWS AND COLS. LABEL
      TMP=IYGRDF
      CALL NUMCNV(TMP,-1,CROW1,NCR1)
      TMP=IYGRDL
      CALL NUMCNV(TMP,-1,CROW2,NCR2)
      TMP=IXF
      CALL NUMCNV(TMP,-1,CCOL1,NCC1)
      TMP=IXL
      CALL NUMCNV(TMP,-1,CCOL2,NCC2)
      LABEL='ROWS '//CROW1(1:NCR1)//'-'//CROW2(1:NCR2)//'     COLS. '
     1             //CCOL1(1:NCC1)//'-'//CCOL2(1:NCC2)
C
C----- WRITE ROWS AND COLS. LABEL AND TITLE
      CALL PLTSTR(0.,YMAX+.2,.21,LABEL,40)
      CALL PLTSTR(0.,YMAX+.6,.28,TITLE,80)
C
      RETURN
      END
      SUBROUTINE BOUND(Z,IOUT,NX,NY,BOGUS)
```

```
C     *******************************************************************
C     ROUTINE TO SET UP IOUT ARRAY SO THAT SPECIFIED AREAS DO NOT HAVE
C          CONTOURS PASS THROUGH
C     SKIP AREAS WHERE Z IS EQUAL BOGUS, BY SETTING THE BRANCH OCCUPIED
C     FLAGS IN ARRAY IBOUND.  THE OCCUPIED FLAGS ARE IDENTICAL TO THOSE
C     USED DURING CONTOURING IN ARRAY IBRMAP.  ARRAY IBRMAP IS INITIALIZED
C     WITH THE VALUES FROM IOUT, THUS SAVING THE ADDITION OF EXTRA
C     CODE WITHIN THE ACTUAL CONTOURING ROUTINES TO DO CHECKING FOR THE
C     BOUNDARY.
C     IF AT NODE (I,J), THE RIGHT BRANCH IS IOUT(I,J), FIRST BIT
C                       THE UPPER BRANCH IS IOUT(I,J), 2ND BIT
C                       THE LEFT BRANCH IS IOUT(I-1,J), FIRST BIT
C                       THE LOWER BRANCH IS IOUT(I,J-1), 2ND BIT
C     *******************************************************************
      INTEGER*2 IOUT
      DIMENSION Z(NX,NY), IOUT(NX,NY)
C
      DO 20 J=1,NY
      DO 10 I=1,NX
      IF(Z(I,J).NE.BOGUS) GO TO 10
C-----SET THE 4 BRANCHES WHICH SURROUND NODE AS OCCUPIED.
      IUP=IOUT(I,J)/2
      IRIGHT=IOUT(I,J) - IUP*2
      IF(IUP.EQ.0) IOUT(I,J)=IOUT(I,J) + 2
      IF(IRIGHT.EQ.0) IOUT(I,J)=IOUT(I,J) + 1
C
      IF(J.EQ.1) GO TO 8
      IDOWN=IOUT(I,J-1)/2
      IF(IDOWN.EQ.0) IOUT(I,J-1)=IOUT(I,J-1) + 2
8     IF(I.EQ.1) GO TO 10
      ILEFT=IOUT(I-1,J) - IOUT(I-1,J)/2*2
      IF(ILEFT.EQ.0) IOUT(I-1,J)=IOUT(I-1,J) + 1
10    CONTINUE
20    CONTINUE
      RETURN
      END
      SUBROUTINE GRID(DELX,DELY,NX,NY,UPI,XMAX,YMAX,IXF,IXL,IYF,IYL)
C     *******************************************************************
C     THIS SUBROUTINE DRAWS GRID LINES
C     *******************************************************************
      DIMENSION DELX(NX), DELY(NY), XPLT(2), YPLT(2)
C
C-----DRAW LINES EFFICIENTLY ON A PEN PLOTTER
C-----DRAW VERTICAL LINES
      Y=0.
      XP=0.
      N=IXL-1
C
      DO 10 I=IXF,N
      XP=XP+DELX(I)/UPI
      XPLT(1)=XP
      YPLT(1)=Y
      Y=YMAX-Y
      XPLT(2)=XP
      YPLT(2)=Y
      CALL PLTLIN(XPLT,YPLT,2)
   10 CONTINUE
C
C-----DRAW HORIZONTAL LINES
      X=XMAX
      YP=YMAX
      N=IYF+1
C
      DO 20 I=N,IYL
      J=IYL+N-I
      YP=YP-DELY(J)/UPI
      XPLT(1)=X
      YPLT(1)=YP
      X=XMAX-X
      XPLT(2)=X
      YPLT(2)=YP
      CALL PLTLIN(XPLT,YPLT,2)
   20 CONTINUE
C
      RETURN
      END
      SUBROUTINE DATAPL (NX,NY,Z,XC,YC,NDEC,HEIGHT,IMASK,BOGUS,
     1                   IXF,IXL,IYF,IYL)
C     ******************************************************************
C     PLOT DATA VALUES AT EACH NODE
C     ******************************************************************
      DIMENSION Z(NX,NY) , XC(NX) , YC(NY)
      CHARACTER*40 LABEL
C
C-----PLOT EFFICIENTLY ON A PEN PLOTTER
      JCK=1
      DO 50 J=IYF,IYL
      JCK=-JCK
      DO 50 K=IXF,IXL
      I=K
      IF(JCK.GT.0) I=IXL-K+IXF
      IF(IMASK.EQ.0) GO TO 40
      IF(Z(I,J).EQ.BOGUS) GO TO 50
 40   CALL NUMCNV(Z(I,J),NDEC,LABEL,NC)
      CALL PLTSTR(XC(I),YC(J),HEIGHT,LABEL,NC)
 50   CONTINUE
      RETURN
      END
      SUBROUTINE ZRANGE(Z,NX,NY,IXF,IXL,IYF,IYL,IMASK,BOGUS,ZMIN,ZMAX)
C     ******************************************************************
C     FIND THE MINIMUM AND MAXIMUM VALUES IN THE Z ARRAY WITHIN THE
C     ACTIVE AREA
```

```
C     ****************************************************************
      DIMENSION Z(NX,NY)
C
      IF(IMASK.EQ.0) GO TO 6
      DO 5 J=IYF,IYL
      DO 5 I=IXF,IXL
      IF(Z(I,J).NE.BOGUS) GO TO 7
5     CONTINUE
6     I=IXF
      J=IYF
7     ZMAX=Z(I,J)
      ZMIN=ZMAX
      DO 10 J=IYF,IYL
      DO 10 I=IXF,IXL
      IF(IMASK.EQ.1 .AND. Z(I,J).EQ.BOGUS) GO TO 10
      IF(Z(I,J).GT.ZMAX) ZMAX=Z(I,J)
      IF(Z(I,J).LT.ZMIN) ZMIN=Z(I,J)
10    CONTINUE
      RETURN
      END
      SUBROUTINE NUMCNV(Z,NDEC,STRING,NCHAR)
C *********************************************************************
C  CONVERT A REAL NUMBER TO A CHARACTER STRING
C  IF NDEC IS NEGATIVE, THERE WILL BE NO DECIMAL DIGITS AND NO DECIMAL
C     POINT IN THE RETURNED STRING
C *********************************************************************
      CHARACTER*20 STRING
      CHARACTER*20 BUF
      CHARACTER*7 FMT
      CHARACTER*10 DEC
      DATA FMT/'(F20.X)'/
      DATA DEC/'0123456789'/
C
C-----USE INTERNAL WRITE TO CONVERT THE REAL NUMBER
      ND=NDEC+1
      IF(ND.GT.10) ND=10
      IF(ND.LT.1) ND=1
      FMT(6:6)=DEC(ND:ND)
      WRITE(BUF,FMT) Z
C
C-----FIND HOW MANY CHARACTERS ARE IN THE STRING
      DO 10 ISTART=1,20
      IF(BUF(ISTART:ISTART).NE.' ') GO TO 20
10    CONTINUE
C
C-----MOVE THE STRING TO THE TARGET STRING VARIABLE
20    ISTOP=20
      IF(NDEC.LT.0) ISTOP=19
      NCHAR=ISTOP-ISTART+1
      STRING(1:NCHAR)=BUF(ISTART:ISTOP)
      RETURN
      END
```

Contouring subroutines:

```
      SUBROUTINE DRWCON(NX,NY,Z,XLOC,YLOC,CLEV,NLEVS,IBRMAP,
     1                  IOUT,IXF,IXL,IYF,IYL,ILABEL)
C     ******************************************************************
C     ENTRY ROUTINE FOR DRAWING CONTOURS
C     ******************************************************************
      INTEGER*2 IBRMAP, IOUT
      DIMENSION Z(NX,NY), IBRMAP(NX,NY), IOUT(NX,NY)
      DIMENSION XLOC(NX),YLOC(NY),CLEV(NLEVS)
C
C-----SEND ONE CONTOUR LEVEL AT A TIME THROUGH SUBROUTINES
      IF(NLEVS.LE.0) RETURN
      DO 55 I=1,NLEVS
C
C-----IF A Z VALUE IS EXACTLY EQUAL TO A CONTOUR VALUE - CHANGE IT SLIGHTLY
C-----BECAUSE THE PROGRAM DOES NOT TEST FOR CURVES THROUGH A NODE
      CHNGR=CLEV(I)+CLEV(I)*10.E-5
      IF ( CLEV(I).EQ.0. ) CHNGR=1.E-20
C
      C=CLEV(I)
      DO 50 J=IXF,IXL
      DO 50 K=IYF,IYL
      IF(Z(J,K).EQ.C) Z(J,K)=CHNGR
   50 CONTINUE
C
      CALL POINTS (NX,NY,Z,CLEV(I),IBRMAP,XLOC,YLOC,IOUT,IXF,IXL,
     1             IYF,IYL,ILABEL)
C
   55 CONTINUE
C
      RETURN
      END
      SUBROUTINE POINTS (NX,NY,Z,CL,IBRMAP,XC,YC,IOUT,IXF,IXL,IYF,IYL,
     1                   ILABEL)
C     ******************************************************************
C      THIS SUBROUTINE WILL CAUSE ONE CONTOUR LEVEL TO BE PLOTTED.
C      IT WILL GENERATE AS MANY CURVES AS REQUIRED FOR THAT LEVEL.
C      IT SCANS ALL NODES LOOKING FOR PLACES TO START A CURVE.  WHEN
C      A START IS FOUND, "FOLLOW" IS CALLED TO DRAW THE CURVE.  AFTER
C      "FOLLOW" COMPLETES, SCANNING CONTINUES -- ALL CPOSSIBLE CURVES
C      FOR THIS CONTOUR LEVEL WILL HAVE BEEN FOUND BY THE TIME ALL
C      NODES ARE SCANNED.
C     ******************************************************************
      INTEGER*2 IBRMAP, IOUT
      DIMENSION Z(NX,NY)
      DIMENSION IBRMAP(NX,NY),XC(NX),YC(NY),IOUT(NX,NY)
C
C-----INITIALIZE IBRMAP FOR EACH LEVEL-- THIS INDICATES WHICH BRANCHES
C-----SHOULDN'T BE DRAWN TO -- EITHER BECAUSE THE USER WANTS THEM BLANKED
C-----OUT OR BECAUSE A CONTOUR HAS ALREADY GONE THROUGH.
      DO 5 I=IXF,IXL
      DO 5 J=IYF,IYL
      IBRMAP(I,J) = IOUT(I,J)
      IBRMAP(I,J) = IOUT(I,J)
5     CONTINUE
      NC=0
C
C-----THIS LOOP FINDS STARTING POINTS
      DO 20 J=IYF,IYL-1
      DO 20 I=IXF,IXL-1
C
C-----CHECK IF A POINT IS ON THE HORIZONTAL BRANCH BETWEEN NODE
C-----I,J AND NODE I+1,J.
      IF (   CL  .LT.   Z(I,J)     .AND.  CL   .LT.   Z(I+1,J)  .OR.
     +       CL  .GT.   Z(I,J)     .AND.  CL   .GT.   Z(I+1,J)) GO TO 10
C
C-----POINT IS ON HORIZONTAL BRANCH
      IBRN=1
C
C-----IF POINT HAS BEEN PLOTTED - SKIP IT
      IRIGHT=IBRMAP(I,J)-IBRMAP(I,J)/2*2
      IF(IRIGHT.NE.0) GO TO 10
C
C-----DRAW THE CURVE
      CALL FOLLOW(NX,NY,Z,IBRMAP,XC,YC,I,J,IBRN,CL,IXF,IXL,IYF,IYL,
     1            ILABEL)
      NC=NC+1
C
C-----CURVE FOR THIS POINT COMPLETED - CONTINUE SCANNING
C
C-----CHECK IF A POINT IS ON THE VERTICAL BRANCH BETWEEN NODE
C-----I,J AND NODE I,J+1.
   10 IF (  CL   .LT.   Z(I,J)   .AND.    CL   .LT.   Z(I,J+1)  .OR.
     +      CL   .GT.   Z(I,J)   .AND.    CL   .GT.   Z(I,J+1)) GO TO 20
C
C-----POINT IS ON VERTICAL BRANCH
      IBRN=2
      IUP=IBRMAP(I,J)/2
      IF(IUP.NE.0) GO TO 20
      CALL FOLLOW(NX,NY,Z,IBRMAP,XC,YC,I,J,IBRN,CL,IXF,IXL,IYF,IYL,
     1            ILABEL)
      NC=NC+1
C
   20 CONTINUE
      RETURN
      END
      SUBROUTINE FOLLOW(NX,NY,Z,IBRMAP,XC,YC,L,M,IBRN,CL,IXF,IXL,
     1                  IYF,IYL,ILABEL)
C     ******************************************************************
C     This subroutine draws curves by connecting points on branches.  The
C     curve is constructed much as one might draw contours by hand.  To
C     continue the contour from a current branch, the surrounding
C     branches in the same general direction are scanned for the same
C     contour value.  If found, the contour is drawn to the new
C     branch.  Contours are not allowed to double back within the same
C     block of 4 branches because this is physically impossible and
```

```
C    would result in discontinuities in the contours.  Also, to avoid
C    the potential for 2 lines representing the same contour value
C    from crossing if it were to happen that all 4 branches of a
C    rectangle include the contour value, the 2 side branches in the
C    direction of the contour path are given drawing precedence over
C    the opposite branch.
C
C    Once a contour has been drawn to a given branch, there are
C    only 3 surrounding branches that the contour might continue to.
C    These 3 branches depend on the current direction of the contour
C    and the orientation of the current branch -- i.e either
C    horizonatl or vertical.  Based on which branch the curve has
C    come to and the direction that the curve is going, the order in
C    which surrounding branches are scanned to try to continue the
C    contour is shown below.  Note that when moving from the first
C    branch of a contour, it is necessary to scan all 6 directions
C    for the next branch because the direction has not been
C    established.
C
C                      CURRENT BRANCH IS HORIZONTAL (IBRN=1)
C    CONTOUR DIRECTION IS UP (IDIR=1)    CONTOUR DIRECTION IS DOWN (IDIR=-1)
C
C            ---3---
C           !       !
C           1       2
C           !       !
C           *---X---                                       *---X---
C                                                          !       !
C                                                          2       1
C                                                          !       !
C                                                           ---3---
C
C
C                               CURRENT BRANCH IS VERTICAL (IBRN=2)
C  CONTOUR DIRECTION IS LEFT (IDIR=-1)    CONTOUR DIRECTION IS RIGHT (IDIR=1)
C
C          ---2---                                          ---2---
C         !       !                                        !       !
C         3       X                                        X       3
C         !       !                                        !       !
C          ---1---*                                        *---1---
C
C     ******************************************************************
       INTEGER*2 IBRMAP
       DIMENSION XC(NX),YC(NY),IBRMAP(NX,NY),Z(NX,NY)
       DIMENSION XPLT(0:50), YPLT(0:50)
       CHARACTER*20 LABEL
C
C-----LABEL CONTOURS EVERY MAXSEG POINTS.  XPLT AND YPLT MUST BE DIMENSIONED
C-----FROM 0 TO MAXSEG.  THE ZERO ELEMENT HOLDS THE START POINT.
       MAXSEG=50
       IBRNBG=IBRN
       IDIR=0
```

```
C
C-----FIND COORDINATES OF STARTING POINT AND MARK BRANCH AS OCCUPIED
      CALL COORDS (NX,NY,Z,XC,YC,L,M,IBRN,CL,XBG,YBG)
      IBRMAP(L,M)=IBRMAP(L,M)+IBRN
C
C-----LABEL THE START OF THE CONTOUR
      IF(ILABEL.NE.0) THEN
      CALL NUMCNV(CL,1,LABEL,NC)
      CALL PLTSTR(XBG+.01,YBG+.01,.14,LABEL,NC)
      END IF
C
C-----ENTER HERE TO GO BACK TO THE START OF THE CURVE AND TRY CONTINUING
C-----THE CURVE IN THE OTHER DIRECTION
1     I=L
      J=M
      XPLT(0)=XBG
      YPLT(0)=YBG
      NUMSEG=0
C
C-----ENTER HERE TO CONTINUE DRAWING AFTER THE START POINT AND AS LONG
C-----AS MORE POINTS CAN BE FOUND.
5     IF(IBRN.EQ.2) GO TO 50
C
C     CONTINUE FOR A POINT ON A HORIZONTAL BRANCH
C     GO TO 50 FOR A POINT ON A VERTICAL BRANCH
C
C
      IF(IDIR.GT.0) GO TO 25
      IF(J.EQ.IYF) GO TO 24
C
C-----SEARCH FOR A POINT ON THE 1ST OF 3 POSSIBLE BRANCHES GOING DOWN
      IF (  CL  .LT.   Z(I+1,J)  .AND.  CL  .LT.   Z(I+1,J-1)   .OR.
     +      CL  .GT.   Z(I+1,J)  .AND.  CL  .GT.   Z(I+1,J-1) ) GO TO 15
C-----THERE IS A POINT, SO SET PARAMETERS FOR THAT BRANCH
      IBRNNX=2
      INX=I+1
      JNX=J-1
      IDIRNX=1
      IDIRRV=1
C-----TEST TO SEE IF THE BRANCH ALREADY HAS A CONTOUR GOING THROUGH IT
C-----IF NOT, GO AND DRAW THE BRANCH TO THE POINT
      IF(IBRMAP(INX,JNX)/2 .EQ.0) GO TO 100
C-----IF THE BRANCH IS OCCUPIED ALREADY, THAT MIGHT MEAN THAT YOU'VE
C-----RETURNED NEAR TO THE START POINT AND NEED TO CLOSE THE CURVE.
C-----IF NOT THE 1ST BRANCH AND YOU'RE NEXT TO THE START POINT(L,M), GO
C-----AND CLOSE.  OTHERWISE CONTINUE LOOKING FOR A VALID BRANCH
C-----NEARBY.
      IF(INX.EQ.L.AND.JNX.EQ.M .AND. IBRNNX.EQ.IBRNBG) GO TO 90
C
   15 IF (  CL  .LT.   Z(I,J-1)  .AND.  CL  .LT.   Z(I,J)       .OR.
     +      CL  .GT.   Z(I,J-1)  .AND.  CL  .GT.   Z(I,J)  ) GO TO 20
      IBRNNX=2
      INX=I
      JNX=J-1
      IDIRNX=-1
      IDIRRV=1
      IF(IBRMAP(INX,JNX)/2 .EQ.0) GO TO 100
      IF(INX.EQ.L.AND.JNX.EQ.M .AND. IBRNNX.EQ.IBRNBG) GO TO 90
C
   20 IF (  CL  .LT.   Z(I,J-1)  .AND.  CL  .LT.   Z(I+1,J-1)   .OR.
     +      CL  .GT.   Z(I,J-1)  .AND.  CL  .GT.   Z(I+1,J-1))  GO TO 24
      IBRNNX=1
      INX=I
      JNX=J-1
      IDIRNX=-1
      IDIRRV=1
      IF((IBRMAP(INX,JNX)-IBRMAP(INX,JNX)/2*2) .EQ. 0 ) GO TO 100
      IF(INX.EQ.L.AND.JNX.EQ.M .AND. IBRNNX.EQ.IBRNBG) GO TO 90
C
24    IF(IDIR.NE.0) GO TO 80
25    IF(J.EQ.IYL) GO TO 80
C
C-----SEARCH UPWARD FROM HORIZONTAL BRANCH
      IF (  CL  .LT.   Z(I,J)    .AND.  CL  .LT.   Z(I,J+1)     .OR.
     +      CL  .GT.   Z(I,J)    .AND.  CL  .GT.   Z(I,J+1) ) GO TO 30
      IBRNNX=2
      INX=I
      JNX=J
      IDIRNX=-1
      IDIRRV=-1
      IF(IBRMAP(INX,JNX)/2 .EQ.0) GO TO 100
      IF(INX.EQ.L.AND.JNX.EQ.M .AND. IBRNNX.EQ.IBRNBG) GO TO 90
C
   30 IF (  CL  .LT.   Z(I+1,J)  .AND.  CL  .LT.   Z(I+1,J+1)   .OR.
     +      CL  .GT.   Z(I+1,J)  .AND.  CL  .GT.   Z(I+1,J+1) ) GO TO 35
      IBRNNX=2
      INX=I+1
      JNX=J
      IDIRNX=1
      IDIRRV=-1
      IF(IBRMAP(I+1,J)/2 .EQ.0) GO TO 100
      IF(INX.EQ.L.AND.JNX.EQ.M .AND. IBRNNX.EQ.IBRNBG) GO TO 90
C
   35 IF (  CL  .LT.   Z(I,J+1)  .AND.  CL  .LT.   Z(I+1,J+1)   .OR.
     +      CL  .GT.   Z(I,J+1)  .AND.  CL  .GT.   Z(I+1,J+1)) GO TO 80
      IBRNNX=1
      INX=I
      JNX=J+1
      IDIRNX=1
      IDIRRV=-1
      IF((IBRMAP(INX,JNX)-IBRMAP(INX,JNX)/2*2) .EQ. 0 ) GO TO 100
      IF(INX.EQ.L.AND.JNX.EQ.M .AND. IBRNNX.EQ.IBRNBG) GO TO 90
C
C-----ALL POSSIBLE BRANCHES HAVE BEEN CHECKED WITHOUT FINDING ANY PATH.
C-----SO THE CURVE HAS DEAD-ENDED.
```

```
C-----GO BACK TO START OF THE CURVE AND TRY TO CONTINUE FROM THERE
      GO TO 80
C
C-----VERTICAL BRANCH
   50 IF(IDIR.GT.0) GO TO 65
      IF (I.EQ.IXF) GO TO 64
C
C-----SEARCH LEFT FROM A VERTICAL BRANCH
      IF (  CL  .LT.   Z(I-1,J)  .AND.  CL  .LT.   Z(I,J)       .OR.
     +      CL  .GT.   Z(I-1,J)  .AND.  CL  .GT.   Z(I,J)    ) GO TO 55
      IBRNNX=1
      INX=I-1
      JNX=J
      IDIRNX=-1
      IDIRRV=1
      IF((IBRMAP(INX,JNX)-IBRMAP(INX,JNX)/2*2) .EQ. 0 ) GO TO 100
      IF(INX.EQ.L.AND.JNX.EQ.M .AND. IBRNNX.EQ.IBRNBG) GO TO 90
C
   55 IF (  CL  .LT.   Z(I-1,J+1).AND.  CL  .LT.   Z(I,J+1)     .OR.
     +      CL  .GT.   Z(I-1,J+1).AND.  CL  .GT.   Z(I,J+1) ) GO TO 60
      IBRNNX=1
      INX=I-1
      JNX=J+1
      IDIRNX=1
      IDIRRV=1
      IF((IBRMAP(INX,JNX)-IBRMAP(INX,JNX)/2*2) .EQ. 0 ) GO TO 100
      IF(INX.EQ.L.AND.JNX.EQ.M .AND. IBRNNX.EQ.IBRNBG) GO TO 90
C
   60 IF (  CL  .LT.   Z(I-1,J)  .AND.  CL  .LT.   Z(I-1,J+1)   .OR.
     +      CL  .GT.   Z(I-1,J)  .AND.  CL  .GT.   Z(I-1,J+1) ) GO TO 64
      IBRNNX=2
      INX=I-1
      JNX=J
      IDIRNX=-1
      IDIRRV=1
      IF(IBRMAP(INX,JNX)/2*2 .EQ. 0 ) GO TO 100
      IF(INX.EQ.L.AND.JNX.EQ.M .AND. IBRNNX.EQ.IBRNBG) GO TO 90
C
   64 IF(IDIR.NE.0) GO TO 80
   65 IF(I.EQ.IXL) GO TO 80
C
C-----SEARCH RIGHT FROM A VERTICAL BRANCH
      IF (  CL  .LT.   Z(I,J)    .AND.  CL  .LT.   Z(I+1,J)     .OR.
     +      CL  .GT.   Z(I,J)    .AND.  CL  .GT.   Z(I+1,J)   ) GO TO 70
      IBRNNX=1
      INX=I
      JNX=J
      IDIRNX=-1
      IDIRRV=-1
      IF((IBRMAP(INX,JNX)-IBRMAP(INX,JNX)/2*2) .EQ. 0 ) GO TO 100
      IF(INX.EQ.L.AND.JNX.EQ.M .AND. IBRNNX.EQ.IBRNBG) GO TO 90
C
```

```
   70 IF (  CL  .LT.   Z(I,J+1)  .AND.  CL  .LT.   Z(I+1,J+1)   .OR.
     +      CL  .GT.   Z(I,J+1)  .AND.  CL  .GT.   Z(I+1,J+1) ) GO TO 75
      IBRNNX=1
      INX=I
      JNX=J+1
      IDIRNX=1
      IDIRRV=-1
      IF((IBRMAP(INX,JNX)-IBRMAP(INX,JNX)/2*2) .EQ. 0 ) GO TO 100
      IF(INX.EQ.L.AND.JNX.EQ.M .AND. IBRNNX.EQ.IBRNBG) GO TO 90
C
   75 IF (  CL  .LT.   Z(I+1,J)  .AND.  CL  .LT.   Z(I+1,J+1)   .OR.
     +      CL  .GT.   Z(I+1,J)  .AND.  CL  .GT.   Z(I+1,J+1)) GO TO 80
      IBRNNX=2
      INX=I+1
      JNX=J
      IDIRNX=1
      IDIRRV=-1
      IF(IBRMAP(INX,JNX)/2*2 .EQ. 0 ) GO TO 100
      IF(INX.EQ.L.AND.JNX.EQ.M .AND. IBRNNX.EQ.IBRNBG) GO TO 90
C
C-----HAVE NOT BEEN ABLE TO FIND ANY PLACE TO GO FROM HERE
C-----CURVE HAS DEAD-ENDED -- GO BACK TO THE START OF IT AND TRY TO
C-----CONTINUE FROM THERE IN A NEW DIRECTION UNLESS THIS HAS ALREADY BEEN
C-----TRIED BEFORE
80    N=MOD(NUMSEG,MAXSEG)
      IF(N.GT.0) CALL PLTLIN(XPLT,YPLT,N+1)
      IF(NUMSEG.EQ.0) RETURN
      IBRN=IBRNBG
      IDIR=IDIRBG
      GO TO 1
C
C-----CLOSE THE CURVE
90    N=MOD(NUMSEG,MAXSEG)+1
      NUMSEG=NUMSEG+1
      XPLT(N)=XBG
      YPLT(N)=YBG
      CALL PLTLIN(XPLT,YPLT,N+1)
      RETURN
C
C-----PLOT A FOUND BRANCH
  100 CALL COORDS (NX,NY,Z,XC,YC,INX,JNX,IBRNNX,CL,XNX,YNX )
      N=MOD(NUMSEG,MAXSEG)+1
      NUMSEG=NUMSEG+1
      XPLT(N)=XNX
      YPLT(N)=YNX
      IF(N.NE.MAXSEG) GO TO 110
      CALL PLTLIN(XPLT,YPLT,MAXSEG+1)
      IF(ILABEL.NE.0) THEN
      CALL NUMCNV(CL,1,LABEL,NC)
      CALL PLTSTR(XNX+.01,YNX+.01,.14,LABEL,NC)
      END IF
      XPLT(0)=XNX
      YPLT(0)=YNX
```

```
C-----NOW SET THE BRANCH-OCCUPIED FLAG SO THAT A 2ND CONTOUR CAN'T
C-----GO TO THIS POINT
 110  IBRMAP(INX,JNX)=IBRMAP(INX,JNX)+IBRNNX
C-----ADVANCE POINTERS TO THE NEXT POINT
      I=INX
      J=JNX
      IF(NUMSEG.EQ.1) IDIRBG=IDIRRV
      IDIR=IDIRNX
      IBRN=IBRNNX
C
C-----GET NEXT POINT ON CURVE
      GO TO 5
      END
      SUBROUTINE COORDS (NX,NY,Z,XC,YC,I,J,IBRN,CL,XP,YP )
C     *****************************************************************
C     THIS SUBROUTINE FINDS THE COORDINATES OF A CONTOUR POINT ON A BRANCH.
C     LINEAR INTERPOLATION BETWEEN THE 2 NODES IS USED.
C     *****************************************************************
      DIMENSION Z(NX,NY)
      DIMENSION  XC(NX) , YC(NY)
C
      IF ( IBRN.EQ.2 ) GO TO 20
C
C-----HORIZONTAL BRANCH
   10 I2=I+1
      J2=J
      B=ABS(XC(I2)-XC(I))
      GO TO 30
C
C-----VERTICAL BRANCH
   20 I2=I
      J2=J+1
      B=ABS(YC(J2)-YC(J))
C
   30 A=ABS(CL-Z(I,J))
      C=ABS (Z(I2,J2)-Z(I,J))
      X=A*B/C
C
C
      IF ( IBRN.EQ.2 ) GO TO 50
C
C-----HORIZONTAL BRANCH
   40 XP=(XC(I)+X)
      YP=YC(J)
      RETURN
C
C-----VERTICAL BRANCH
   50 YP=(YC(J)+X)
      XP=XC(I)
      RETURN
      END
      SUBROUTINE NODLOC(DELX,DELY,XLOC,YLOC,UPI,XMAX,YMAX,NX,NY,
     1                  IXF,IXL,IYF,IYL)
C    *****************************************************************
C    CALCULATE NODE LOCATIONS IN PLOTTER COORDINATES
C    *****************************************************************
      DIMENSION DELX(NX),DELY(NY),XLOC(NX),YLOC(NY)
C
C-----CREATE X COORDINATES
      SP=0.
      DO 10 I=IXF,IXL
      TEMP=DELX(I)/UPI
      XLOC(I)=SP+TEMP/2.
   10 SP=SP+TEMP
      XMAX=SP
C
C-----CREATE Y COORDINATES
      SP=0.
      DO 20 I=IYF,IYL
      TEMP=DELY(I)/UPI
      YLOC(I)=SP+TEMP/2.
   20 SP=SP+TEMP
      YMAX=SP
C
      RETURN
      END
```

Graphics primitive subroutines:

```
      SUBROUTINE PLTBEG(XPAGE,YPAGE,IFILL)
C *********************************************************************
C  INITIALIZE PLOTTING
C *********************************************************************
      COMMON /GKSCOM/NCOLDF
      DIMENSION LASF(13)
      DATA LASF/13*1/
C
C-----PROMPT FOR WORKSTATION TYPE
      WRITE(*,*)
      WRITE(*,*) ' ENTER WORKSTATION TYPE:'
      READ(*,*) IWSTYP
C
C-----OPEN GKS
      OPEN(UNIT=10,FILE='GKS.ERR', STATUS='UNKNOWN')
      WRITE(10,*) 'GKS ERROR FILE:'
      CALL GOPKS(10,1000)
C
C-----OPEN THE WORKSTATION AND FIND HOW BIG ITS PAGE IS
      CALL GOPWK(1,0,IWSTYP)
      CALL GACWK(1)
      CALL GQDSP(IWSTYP,IERR,IUNIT,RX,RY,LX,LY)
      PICASP=XPAGE/YPAGE
      WSASP=RX/RY
      RMTOIN=39.3701
      RXM=RX*RMTOIN
      RYM=RY*RMTOIN
C
C-----DEFINE THE WORLD COORDINATE WINDOW
C-----EITHER FILL THE DISPLAY COMPLETELY OR SCALE ABSOLUTELY
      IF(IFILL.NE.0) GO TO 50
C
C-----USE ABSOLUTE SCALING IF THE PLOT WILL FIT ON THE PAGE
      IF(IUNIT.NE.0) THEN
         WRITE(10,*) 'GRAPHICS DEVICE COORDINATE UNITS ARE UNKNOWN'
         WRITE(10,*) 'REQUEST TO USE ABSOLUTE SCALING HAS BEEN ABORTED'
         GO TO 50
      END IF
      IF(XPAGE.GT.RXM .OR. YPAGE.GT.RYM) THEN
         WRITE(10,*) 'PLOT IS TOO BIG TO FIT ON THE PLOTTER PAGE'
         WRITE(10,*) 'REQUEST TO USE ABSOLUTE SCALING HAS BEEN ABORTED'
         GO TO 50
      END IF
      XWC=RXM
      YWC=RYM
      GO TO 60
C
C-----AUTOMATIC SCALING -- FILL THE DISPLAY COMPLETELY
50    IF(WSASP.GT.PICASP) THEN
         YWC=YPAGE
         XWC=YPAGE*WSASP
      ELSE
         XWC=XPAGE
         YWC=XPAGE/WSASP
      ENDIF
C
C-----SET THE WORLD COORDINATE WINDOW AND THE CORRESPONDING VIEWPORT
60    CALL GSWN(1,0.,XWC,0.,YWC)
      SIZE=XWC
      IF(YWC.GT.XWC) SIZE=YWC
      XNDC=XWC/SIZE
      YNDC=YWC/SIZE
      CALL GSVP(1,0.,XNDC,0.,YNDC)
      CALL GSELNT(1)
C
C-----SET WORKSTATION TRANSFORMATION
      CALL GSWKWN(1,0.,XNDC,0.,YNDC)
      CALL GSWKVP(1,0.,RX,0.,RY)
C
C-----SETUP CHARACTER FONT 1, WHICH SHOULD EXIST IN ALL GKS VERSIONS
      CALL GSASF(LASF)
      CALL GSTXFP(1,1)
      CALL GSCHXP(1.)
      CALL GSCHSP(.15)
C
C-----SETUP COLORS IF GRAPHICS DEVICE SUPPORTS COLOR
      CALL GQCF(IWSTYP,IERR,NCOL,ICLCAP,NDEF)
      NCOLDF=2
      IF(ICLCAP.EQ.0) RETURN
      NCOLDF=NCOL
      IF(NCOLDF.EQ.0 .OR. NCOLDF.GT.5) NCOLDF=5
      IF(NCOLDF.GE.3) CALL GSCR(1,2,1.,0.,0.)
      IF(NCOLDF.GE.4) CALL GSCR(1,3,0.,1.,0.)
      IF(NCOLDF.GE.5) CALL GSCR(1,4,0.,0.,1.)
C
      RETURN
      END
      SUBROUTINE PLTEND
C **********************************************************************
C  TERMINATE PLOT
C **********************************************************************
      CHARACTER*1 ANS
C
C-----SOME GKS SYSTEMS REQUIRE A PAUSE FOR THE USER TO VIEW THE PLOT
C-----BEFORE GKS IS CLOSED -- PRESS RETURN KEY TO CONTINUE.
C-----SOME GKS SYSTEMS HAVE A BUILT-IN PAUSE AND DO NOT REQUIRE THIS
C-----ADDITIONAL PAUSE -- TO DELETE THE PAUSE, PUT "C" IN COLUMN 1 OF
C-----THE FOLLOWING 2 LINES:
      READ(*,1) ANS
1     FORMAT(A)
      CALL GDAWK(1)
      CALL GCLWK(1)
      CALL GCLKS
      RETURN
      END
      SUBROUTINE PLTCOL(ICOLOR)
C ***********************************************************************
C  CHANGE PLOT COLOR
C ***********************************************************************
      COMMON /GKSCOM/NCOLDF
C
C-----IF COLOR INDEX IS UNDEFINED, DON'T CHANGE THE COLOR (COLOR INDEX
C-----CAN RANGE FROM 0 THROUGH NCOLDF-1)
      IF(ICOLOR.GE.NCOLDF .OR. ICOLOR.LT.0) RETURN
      CALL GSTXCI(ICOLOR)
      CALL GSPLCI(ICOLOR)
      RETURN
      END
      SUBROUTINE PLTLIN(X,Y,N)
C ***********************************************************************
C  PLOT A POLYLINE
C ***********************************************************************
      DIMENSION X(N),Y(N)
C
      CALL GPL(N,X,Y)
      RETURN
      END
      SUBROUTINE PLTSTR(X,Y,H,STRING,NC)
C ***********************************************************************
C  PLOT A STRING OF CHARACTERS
C ***********************************************************************
      CHARACTER*(*) STRING
C
      CALL GSCHH(H)
      CALL GTX(X,Y,STRING(1:NC))
      RETURN
      END
```