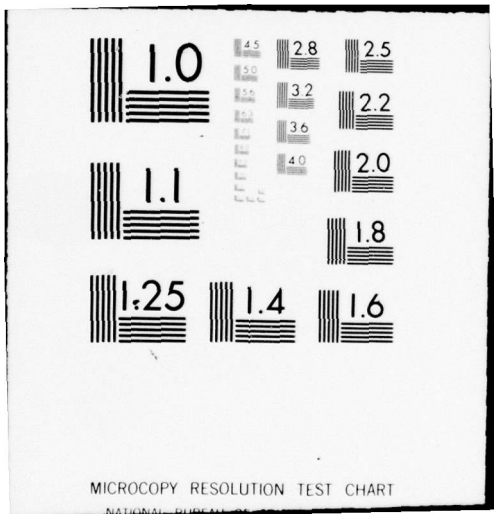
AD-A072 525 AIR FORCE AERO PROPULSION LAB WRIGHT-PATTERSON AFB OH F/G 9/2
A FORTRAN ALGORITHM FOR PLOTTING CONTOURS OVER A TRIANGULAR MES--ETC(U)
JUN 79 K D MACH

UNCLASSIFIED AFAPL-TR-79-2057 NL

| OF |
ADA
072525

END
DATE
FILMED
9-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART

AFAPL-TR-79-2057

2 B.S.

LEVEL II

A FORTRAN ALGORITHM FOR PLOTTING CONTOURS OVER A TRIANGULAR MESH

Turbine Engine Division
Components Branch

June 1979

TECHNICAL REPORT AFAPL-TR-79-2057

Final Report for the period July through December 1978

Approved for public release: distribution unlimited

AIR FORCE AERO-PROPULSION LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Kervyn D Mach

KERVYN D. MACH
Project Engineer

Joseph C Hurst

JOSEPH C. HURST, Maj, USAF
Chief, Components Branch
Turbine Engine Division

FOR THE COMMANDER

Ernest C. Simpson

ERNEST C. SIMPSON
Director
Turbine Engine Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFAPL/TBC ,*W-PAFB, OH 45433 to help us maintain a current mailing list".*

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFAPL-TR-79-2057	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): A FORTRAN ALGORITHM FOR PLOTTING CONTOURS OVER A TRIANGULAR MESH.	5. TYPE OF REPORT & PERIOD COVERED: INTERIM REPORT, JULY 1978 -- DECEMBER 1978
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Kervyn D. Mach	8. CONTRACT OR GRANT NUMBER(s): In-House
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Aero Propulsion Laboratory (AFAPL/TBC) Wright-Patterson Air Force Base, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: P. E. 62203F, Proj 3066, Task 06, W.U. 02 and 09
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Aero Propulsion Laboratory (AFAPL/TBC) Wright-Patterson Air Force Base, Ohio 45433	12. REPORT DATE: June 1979
	13. NUMBER OF PAGES: 27
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Final rept. Jul - Dec 78	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Contour Plotting
Computer Graphics

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes a contour plotting algorithm for data distributed on a triangular mesh such as might occur from numerical solutions of partial differential equations over an irregular domain. The method of application is described and illustrated with examples. The algorithm requires considerable memory but very little execution time. Suggestions for reducing the memory requirements and a complete program listing are included.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

011 570

FOREWORD

This report describes work conducted within the Air Force Aero Propulsion Laboratory, Turbine Engine Division, Components Branch (TBC), Wright-Patterson Air Force Base, Ohio. The work was accomplished under Project 3066, "Gas Turbine Technology," Task 06, "Turbine Technology," Work Units 02,"Turbine Aeromechanical Analysis," and 19, "Low Aspect Ratio Turbine Technology," between July 1978 and December 1978.

This report was submitted by the author in May 1979.

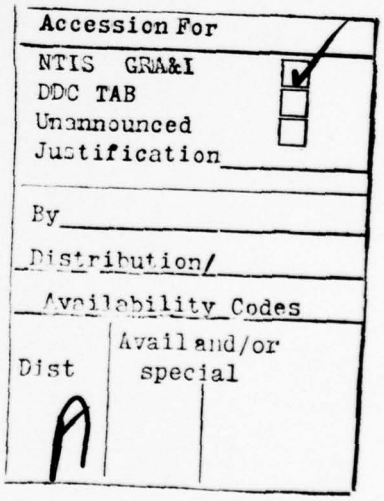


TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. INTRODUCTION

The use of digital computers to solve partial differential equations in two or more dimensions, e.g., in fluid dynamics or structural mechanics, typically involves the generation of numerical values for the dependent variable (or variables) at descrete points on a surface, which may or may not be a plane. If more than a few points are involved, graphical presentation of the results is virtually mandatory for human comprehension. Contour plots of the dependent variable convey information very effectively and hence are popular. Two algorithms which can be used when the descrete solution points are arranged on a rectangular grid are described in Reference 1.

However, not all problems are amenable to description on a rectangular grid. Finite element methods frequently use triangular elements and some solution domains simply are not rectangular. One also occasionally encounters data which is scattered irregularly over the surface. The algorithm described herein works very well in such environments. Though it requires a great deal more memory than LEVEL 1 or LEVEL 2 and requires more effort to set up, it is not difficult to use and produces high quality plots with little expenditure of computer time.

2. THE ALGORITHM

2.1 Overview

The algorithm presented here as ISOVAR2 and subsidiary routines ISOVAR3, FOLLOW, DRAWBND, DRAWL, and LABEL is a FORTRAN recoding of the Algol procedure tricont 2 described in Reference 2. It retains all the capabilities and limitations of the original. Because of the differences between FORTRAN and Algol, the method of invoking the algorithm is greatly different and is described in detail below and with some examples.

2.2 Operation

ISOVAR2 requires the user to supply several arrays and simple integers which describe the arrangement of the grid and the data on it. Some must be preset by the user while others may be treated as working storage. We shall describe them in order:

EL	Integer working array dimensioned (E, 3).
EP	Integer array dimensioned (E, 3) in which EP (R,S), (S = 1, 2, 3,) are the subscripts in PX and PT of the vertices of element R, proceeding clockwise around the element. See Figure 1.
E	The number of triangular elements in the grid. Integer.
PX, PY, PZ	Real arrays dimensioned (G) containing the data to be contoured, X, Y, and Z values respectively. They may be doubly dimensioned in the calling program, in which case G is the product of the two dimensions. Note: PX and PY must be scaled to represent actual inches of pen movement on the plot. See the examples.
G	The number of points in each of PX, PY, and PZ. Integer.
HTS	Real array containing the contour heights to be drawn.
K	The number of contours in HTS. Integer.
LL, LP, LR	Integer working arrays, dimensioned (N, 2).
UNUSED	Logical working array, dimensioned (N).
N	Maximum number of lines bounding the triangles. Lines common to two triangles are counted once. N is never more than 3*G, nor less than E + G - 1. See Figure 3.

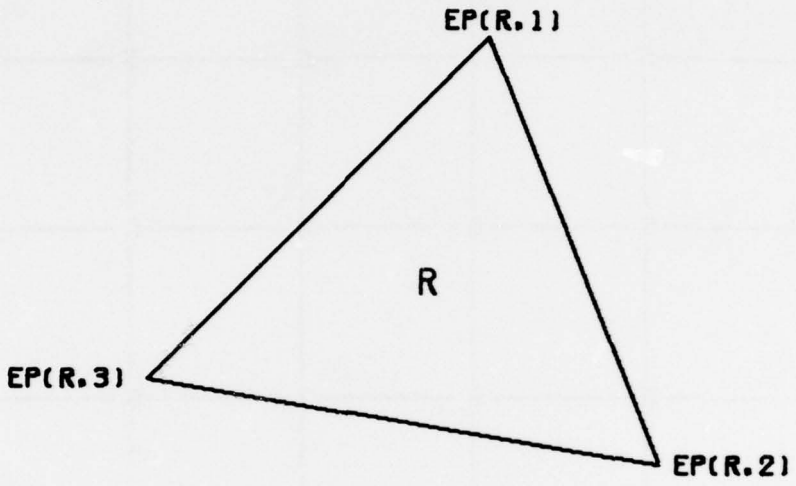


Figure 1
Definition of EP

ISOVAR2 will print the contents of the various arrays if the logical variable DBUG in labelled COMMON /BUGBUG/ is set .TRUE. This option should be used with discretion. Large meshes will yield very large quantities of printout.

2.3 A Simple Example

Consider an array of 30 Z values arranged on a 6 x 5 grid as shown in Figure 2. X values run from 0 to 5 and Y values from 0 to 4. Let us create a triangular mesh by running diagonals from upper left to lower right of each mesh rectangle as shown in Figure 3. Understand that this choice is completely arbitrary for this example and the other diagonal would have served just as well. In other applications, circumstances may favor a particular choice or combination of choices. In this example we could have taken some diagonals one way and some the other.

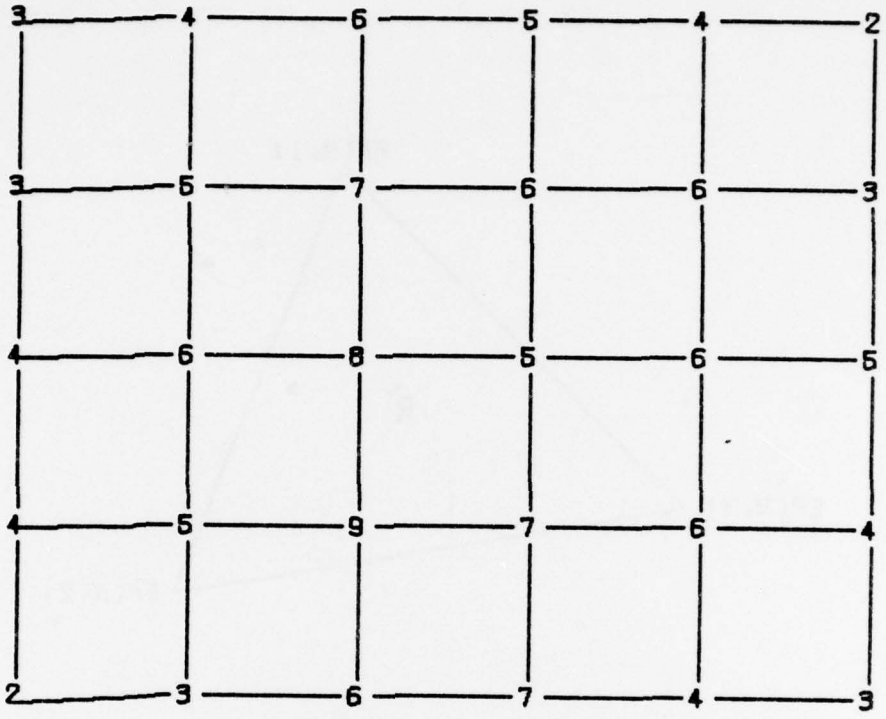


Figure 2
Sample Z Array

We can then create the PX, PY, and PZ arrays. From the description of the problem, G is 30. Thus:

```
DIMENSION PX(6, 5), PY(6, 5), PZ(6, 5)
DATA (PZ (I, 1), I = 1, 6) / 2.0, 3.0, 6.0, 7.0, 4.0, 3.0 /
. . .
DATA (PZ (I, 5), I = 1, 6) etc. DO 1 J = 1, 5
J1 - J - 1
DO 1 I = 1, 6
PX (I, J) = I - 1
PY (I, J) = J1
CONTINUE
```

Then the EP array is set up. From Figure 3, we know there are 40 triangles. Therefore E is 40. For our own reference (The information

is not passed explicitly to ISOVAR2) we need to adopt a numbering convention for the triangles and their vertices. Let us therefore start in the lower left corner of the mesh and proceed to the right, numbering the triangles in sequence, with odd numbers going to the lower triangles and even numbers to the uppers, as shown in Figure 3.

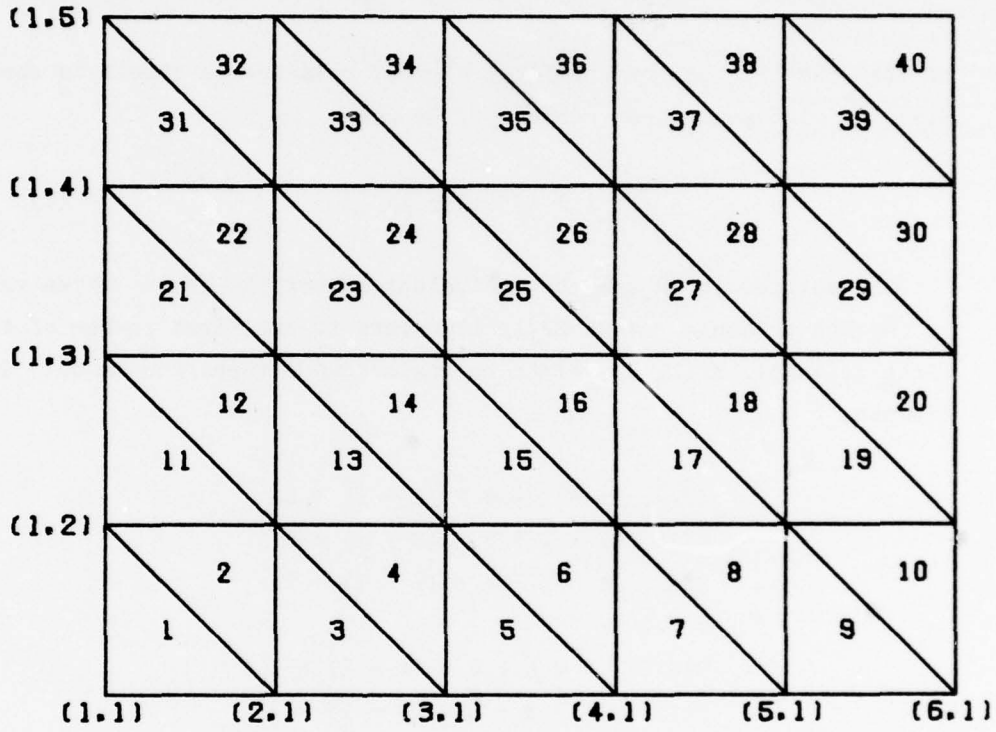


Figure 3
Triangular Mesh
Numbers in parentheses are FORTRAN subscripts for PX, PY, PZ
Numbers in triangles count the mesh elements and correspond to the first subscript R of EP (R, S).

We also need a system for identifying the vertices of the triangles. Therefore, let us start at the lower left vertex of each lower triangle and at the bottom vertex of each upper triangle and count clockwise around each.

All that remains is to convert the double FORTRAN subscripts to single integers for EP. For this we use the same method FORTRAN uses: The relative location of Azj in an array dimensioned A(I, J) is given by

$$L = i - 1 + I * (j - 1) \qquad (1)$$

so that the location of the first element relative to itself is zero. Since we want equivalent subscript, we use

$$L\,1 = i + I * (j - 1) \qquad (2)$$

The entries in EP are the equivalent subscripts of the three vertices of the Rth triangle. Thus EP(1, 1) points to the first corner of the first triangle, EP(1, 2) points to its second corner, and so on. In our example,

```
EP(1, 1) = 1 + 6 * (1 - 1) = 1
EP(1, 2) + 1 + 6 * (2 - 1) = 7
EP(1, 3) = 2 + 6 * (1 - 1) = 2
. . .
EP(40, 1) = 6 + 6 * (4 - 1) = 24
EP(40, 2) = 5 + 6 * (5 - 1) = 29
EP(40, 3) = 6 + 6 * (5 - 1) = 30
```

The choice of contours depends on the problem. We will use the sequence 2.5 to 8.5 in steps of 1.0, which makes K = 7.

Lastly, we must establish the size of the arrays LL, LP, LR, and UNUSED, i.e., the value of N. We know that N is not more than 90(= 3 * G) nor less than 69(=E + G - 1). If we count the lines in Figure 3, we will find 69. For simple geometries like this, the lower limit will always suffice.

With this, we are ready to plot. The complete driver program is listed with ISOVAR2 in the Appendix and the completed plot is shown in Figure 4. Note that ISOVAR2 draws the grid boundaries; no special provision for those is necessary.

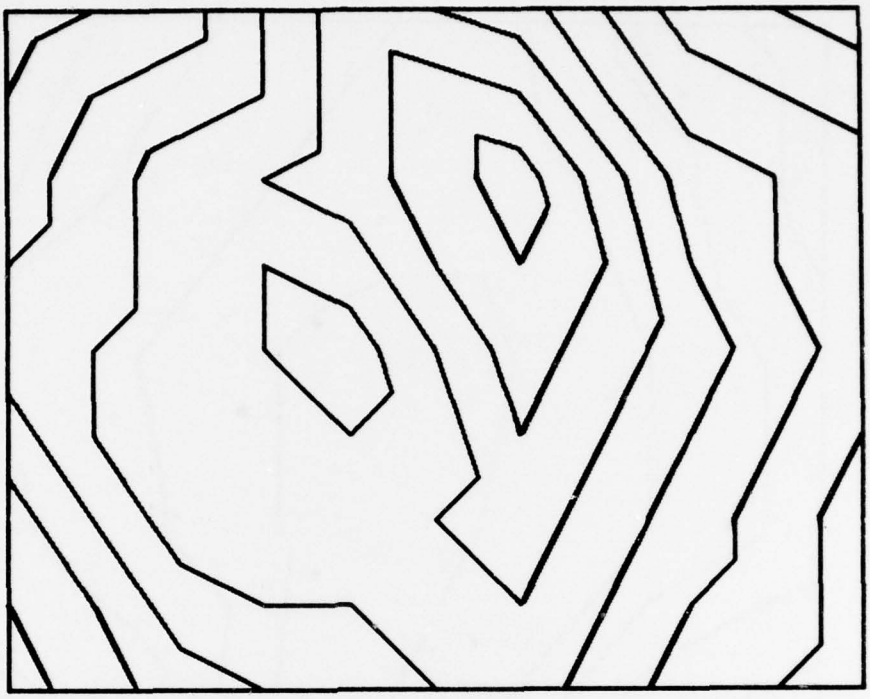


Figure 4
Sample Contour Plot

2.4 Multiple Data Sets on the Same Mesh

If one wishes to plot contours of more than one variable over the same mesh, he can bypass ISOVAR2 on the second and subsequent calls and enter the algorithm at ISOVAR3. The arguments are EL, EP, E, PX, PY, PZ, G, HTS, K, LL, LP, LR, UNUSED, and N. All have the same meanings as for ISOVAR2 except that PZ would be the alternate dependent variable set. The user must establish a new plot origin or the new plot will be superimposed on the old. Figure 5 shows a sample obtained by "relaxing" the first data set one cycle, i.e., for every point not on a boundary

ZP (I, J) = (PZ(I, J + 1) + PZ(I, J - 1) + PZ(I + 1, J) + PZ(I -1, J)) / 4.0

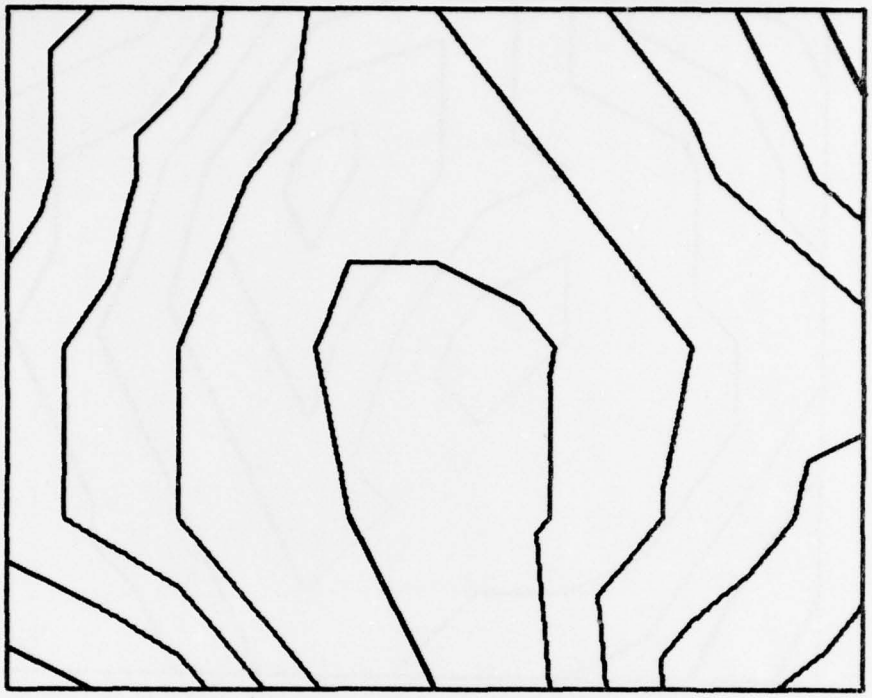


Figure 5
Contours of Second Data Set

2.5 An Example from Fluid Mechanics

The primary reason for developing ISOVAR2 from tricont 2 was to plot aerodynamic data at the exit of a turbine blade row. The general grid arrangement is similar to Figure 3, but with three differences. The upper and lower boundaries are circular arcs and the left and right boundaries are identical curves separated by the blade pitch angle. The grid is typically around 30 x 30 with density increasing toward each of the four walls. Figure 6 shows a typical plot of kinetic energy distribution, the highest levels being near the center.

Figure 6 required only five seconds of CDC 6600 central processor time to generate, but the total array storage (including PX, PY, and PZ) was more than 31000 (decimal) words.

2.6 Potential Improvements

2.6.1 Memory Requirements

As seen in the last example, one of the obstacles to the

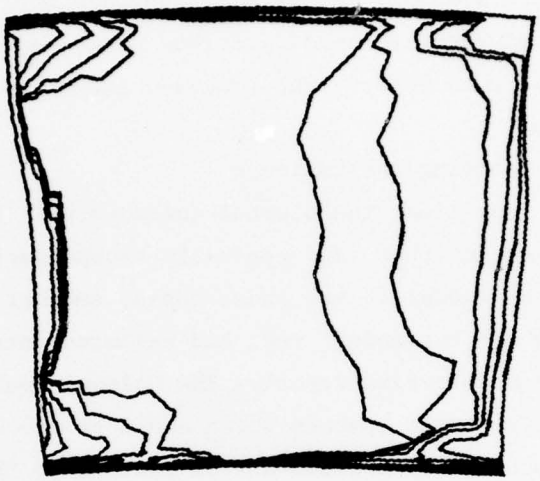


Figure 6
Kinetic Energy Contours at Stator Exit Plane

general application of ISOVAR2 is the large amount of memory it requires for the working arrays EL, EP, etc. Only the PX, PY, and PZ arrays are floating point and use an entire computer word for each element. The remaining arrays are integer or logical and therefore use only a fraction of a word per element. The EL and EP arrays for instance are dimensioned (E, 3). These could be packed by threes into a 60 bit CDC word. Allowing 18 bits each would permit a maximum value of 777777 octal or 262143 decimal, large enough for any application. The LL, LP, and LR arrays are dimensioned (N, 2). These could be packed two to a word or, by storing corresponding elements of LL, LP, and LR together, three to a word. Thus, half to two thirds of the integer array storage could be saved. The logical array UNUSED uses only one bit per word. These could be bunched 60 to a CDC word to save more than 98% of the required storage.

Packing and unpacking is a simple operation in CDC Fortran because it includes shift and mask instructions. IBM 360/370 Fortran allows a

length specification in type statements, e.g., INTEGER * 2 E, which specifies the number of bytes each variable is to occupy. Other computers may require assembly language subroutines for packing and unpacking. In any case, the modifications would require only a few hours' work, save much memory, and increase execution time by only a very small amount.

2.6.1 Labeling the Contours

Many times the plotted contours fall close together, double back, twist and turn, and generally become hard to distinguish. Figure 6 is a good example. For these cases, subroutine DRAWL can be modified to mark the beginning, end, and selected intermediate points of a contour with a distinctive symbol. The Calcomp routine SYMBOL has a repertoire of 15 suitable symbols which could be invoked by passing the value of the contour loop counter J from ISOVAR3 to DRAWL through a labelled COMMON block. Alternatively, one could cause DRAWL to print the X and Y coordinates at selected intervals on the contour. The latter method is used in the program which generated Figure 6. Annotating the contours directly is possible but would be difficult for the very reasons stated above.

3. CONCLUSIONS

ISOVAR2 has proven to be a useful and versatile algorithm which produces very satisfactory contour plots. It requires a lot of memory but very little execution time. The memory requirement can be reduced, as described in Section 2.6.1.

REFERENCES

1. J. S. Petty and K. D. Mach, Contouring and Hidden Line Algorithms for Vector Graphic Displays, AFAPL-TR-77-3, Air Force Aero Propulsion Laboratory, Wright-Patterson Air Force Base, Ohio, January 1977.

2. B. R. Heap, Algorithms for the Production of Contour Maps Over an Irregular Triangular Mesh, NPL-NAC-10, National Physical Laboratory, Teddington, England, February 1972.

3. FORTRAN Extended Version 4 Reference Manual, Publication No. 60497800, Control Data Corporation, Sunnyvale, California, November 1975.

4. IBM System/360 and System/370 FORTRAN IV Language, Eleventh Edition, Publication No. GC28-6516-10, IBM Corporation, Palo Alto, California, May 1974.

APPENDIX
PROGRAM LISTINGS

The complete FORTRAN code of ISOVAR2 and the driver program TWO which was used to generate Figures 4 and 5 are listed herein.

The listings contain occasional continuation lines marked with a dollar sign. These were created by the listing program to maintain the right-hand margin and do not appear in the actual code.

```
      PROGRAM TWO (PLOT, OUTPUT=4028, TAPE6=OUTPUT)
      DIMENSION H(7), LL(69,2), LF(69,2), LR(69,2), X(6,5),
     $ Y(6,5),
     1  Z(6,5), Z2(6,5)
      LOGICAL DBUG, L(68)
      INTEGER EP(40,3), EL(40,3), F, G, Q
      COMMON /BUGBUG/ DBUG
C
      DATA H / 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5 /
      DATA (Z(I,1),I=1,6) / 2.0, 3.0, 6.0, 7.0, 4.0, 3.0 /
      DATA (Z(I,2),I=1,6) / 4.0, 5.0, 9.0, 7.0, 6.0, 4.0 /
      DATA (Z(I,3),I=1,6) / 4.0, 6.0, 8.0, 5.0, 6.0, 5.0 /
      DATA (Z(I,4),I=1,6) / 3.0, 5.0, 7.0, 6.0, 6.0, 3.0 /
      DATA (Z(I,5),I=1,6) / 3.0, 4.0, 6.0, 5.0, 4.0, 2.0 /
C
      K(I,J)      = I + 6 * (J - 1)
C
        DO 1 I      = 1, 6
        FI          = I - 1
          DO 1 J      = 1, 5
          X(I,J)      = FI
          Y(I,J)      = J - 1
    1     CONTINUE
C
        DBUG        = .FALSE.
        F           = 40
        G           = 30
        N           = 69
        Q           = 7
C
C         LOAD EP WITH X, Y, AND Z SUBSCRIPTS
C
        DO 2 I      = 1, 5
          DO 2 J      = 1, 4
C
C         LOWER TRIANGLE HAS SUBSCRIPT M IN EP: UPPER HAS
C    *          SUBSCRIPT L.
C
            L           = 2 * (I + 5 * (J - 1))
            M           = L - 1
            EP(M,1)     = K(I,J)
            EP(M,2)     = K(I,J + 1)
            EP(M,3)     = K(I + 1,J)
            EP(L,1)     = K(I + 1,J)
            EP(L,2)     = K(I,J + 1)
            EP(L,3)     = K(I + 1,J + 1)
    2     CONTINUE
C
        CALL PLOT (2.0, 4.0, -3)
```

```
      CALL ISOVAR2 (EL, FP, F, X, Y, Z, G, H, Q, LL, LP,
     $ LR, U, N)
C
C          GENERATE SECOND DATA SET
C
          DO 3 I      = 1, 6
          Z2(I,1)     = Z(I,1)
          Z2(I,5)     = Z(I,5)
    3     CONTINUE
          DO 4 J      = 2, 4
          Z2(1,J)     = Z(1,J)
          Z2(6,J)     = Z(6,J)
    4     CONTINUE
          DO 5 I      = 2, 5
            DO 5 J    = 2, 4
            Z2(I,J)   = 0.25 * (Z(I,J + 1) + Z(I,J - 1) +
     $ Z(I + 1,J) +
     1      Z(I - 1,J))
    5     CONTINUE
C
      CALL PLOT (10.0, 0.0, -3)
      CALL ISOVAR3 (X, Y, Z2, G, H, Q, LL, LP, LR, U, N)
      CALL PLOT (8.5, -4.0, -3)
      CALL SYMBOL (0.0, 0.5, 0.105, "FINISHED", 90.0, 8)
      CALL PLOTE
      STOP
      END
```

```
      SUBROUTINE  ISOVAR2 (EL, EP, E, PX, PY, PZ, G, HTS,
     $ K, LL, LP,
     1   LR, UNUSED, N)
      COMMON  /BUGBUG/  DBUG
      COMMON  /ISOLIN/  L, LB
C
C        THIS ALGORITHM IS USED FOR CONTOURING OVER AN
C     $        IRREGULAR,
C        TRIANGULAR MESH.  THE PARAMETERS REQUIRED ARE:
C     E   - THE NUMBER OF TRIANGULAR ELEMENTS.  INTEGER.
C     EL  - INTEGER WORKING ARRAY, DIMENSIONED (E, 3).
C     EP  - INTEGER ARRAY DIMENSIONED (E, 3), IN WHICH
C     $        EP(R, S),
C              (S = 1, 2, 3), ARE THE SUBSCRIPTS IN PX AND
C     $        PY OF THE
C              CORNERS OF THE ELEMENT R, ARRANGED CLOCKWISE
C              AROUND THE ELEMENT.
C     PX  - REAL ARRAYS CONTAINING THE DATA TO BE
C     $        CONTOURED.
C     PY
C     PZ
C     G   - THE NUMBER OF POINTS IN EACH OF PX, PY, PZ.
C     $        INTEGER.
C     HTS - REAL ARRAY OF CONTOUR HEIGHTS TO BE DRAWN.
C     K   - THE NUMBER OF CONTOURS IN HTS.  INTEGER.
C     LL  - INTEGER WORKING ARRAYS, DIMENSIONED (N, 2).
C     LP
C     LR
C     UNUSED - LOGICAL WORKING ARRAY, DIMENSIONED (N).
C     N   - MAXIMUM NUMBER OF LINES BOUNDING THE
C     $        TRIANGLES.
C              NEVER MORE THAN 3 * G, NOR LESS THAN E + G -
C     $        1.
C
C        EXAMPLE
C
C        FOR A RECTANGULAR ARRAY CONTAINING M ELEMENTS IN
C     $        THE
C        X DIRECTION AND O ELEMENTS IN THE Y DIRECTION,
C        E = 2 * (M - 1) * (O - 1)
C        G = M * O
C        K = YOUR CHOICE
C        N = E + G
C
C        A DETAILED DESCRIPTION OF THE ALGORITHM IS GIVEN
C     $        IN
C        SECTION 3 OF THE ORIGINAL REPORT.
C
C        ADAPTED FROM THE ALGOL PROCEDURE TRICONT 2, FOUND
C     $        IN
```

```
C          (NPL-NAC-10) ALGORITHMS FOR THE PRODUCTION OF
C     $          CONTOUR MAPS,
C          B.R. HEAP (NATL. PHYS. LAB., TEDDINGTON, ENGLAND)
C          MARCH, 1972.  PAGE 21.
C
C          INITIALLY THE BOUNDARY LINE COUNT, LB, AND THE
C     $          ELEMENTS
C          OF THE ARRAY EL ARE SET TO ZERO AND THE TOTAL
C     $          LINE COUNT,
C          L, IS SET TO ITS MAXIMUM POSSIBLE VALUE.
C
        DIMENSION    HTS (K), LL (N,2), LP (N,2), LR (N,2),
     $ PX (G), PY
     1   (G), PZ (G)
        INTEGER      E, EL (E,3), EP (E,3), G, PA, PB, PC, R,
     $ S, SA, SB
        LOGICAL      DBUG, UNUSED (N)
        LE           = 3 * E
        L            = LE
        LB           = 0
          DO 1 R     = 1, E
            DO 1 S   = 1, 3
            EL(R, S) = 0
    1       CONTINUE
C
C          EACH ELEMENT OF THE MESH IS EXAMINED IN TURN, ITS
C     $          POINT
C          LABELS EXTRACTED, AND THE APPROPRIATE ENTRIES IN
C     $          THE
C          ARRAY EL MADE USING SUBROUTINE LABEL.
C
          DO 2 R     = 1, E
          PA         = EP(R, 1)
          PB         = EP(R, 2)
          PC         = EP(R, 3)
          IF (EL(R,1) .EQ. 0) CALL LABEL (R,1,PB,PC,E,EP,EL)
          IF (EL(R,2) .EQ. 0) CALL LABEL (R,2,PC,PA,E,EP,EL)
          IF (EL(R,3) .EQ. 0) CALL LABEL (R,3,PA,PB,E,EP,EL)
    2     CONTINUE
C
C          THE LINE COUNT, L, IS AMENDED AND THE LINE LABELS
C          ADJUSTED SO THAT THE BOUNDARY LINES HAVE LABELS
C     $          1,2,3,
C          ..., LB, AND THE INTERIOR LINES HAVE LABELS LE+1,
C     $          LB+2,
C          ..., L.
C
        IF (LB .EQ. LE) GO TO 4
        LC           = LE - LB
```

```
      L          = L - LC
        DO 3 R     = 1, E
          DO 3 S     = 1, 3
          IF  (EL(R,S) .GT. LB) EL (R,S) = EL (R,S) - LC
    3     CONTINUE
C
C         THE DIMENSIONS OF THE ARRAYS LR,LL,LP (L,2)
C     *         REQUIRED
C         FOR ISOVAR3 ARE NOW KNOWN.
C         INITIALLY ALL ENTRIES IN THE ARRAYS LR AND LL
C         ARE SET TO ZERO.
C
    4 CONTINUE
      IF (L .LE. N) GO TO 5
      PRINT *, "INCREASE N TO ", L
      STOP
    5 CONTINUE
        DO 6 R     = 1, L
          DO 6 S     = 1, 2
          LR(R, S)   = LL(R, S) = 0
    6     CONTINUE
C
C         THE ARRAYS LR, LP, AND LL ARE SET UP BY SCANNING
C     $         THROUGH
C       ALL ELEMENTS OF THE MESH AND EXTRACTING THE
C     $         APPROPRIATE
C         ENTRIES OF ARRAYS EP AND EL.
C
        DO 7 R     = 1, E
          DO 7 S     = 1, 3
          MA         = EL(R, S)
          SA         = S + 1
          IF  (S .EQ. 3) SA = 1
          SB         = S - 1
          IF  (S .EQ. 1) SB = 3
          NA         = 2
          IF  (LR(MA,1) .EQ. 0) NA = 1
          LR(MA, NA) = EL(R, SB)
          LL(MA, NA) = EL(R, SA).
          IF (NA .NE. 1) GO TO 7
          LP(MA, 1) = EP(R, SA)
          LP(MA, 2) = EP(R, SB)
    7     CONTINUE
      IF ( .NOT. DBUG) GO TO 12
      PRINT  100
        DO 8 I     = 1, E
        PRINT  101, I, (EP(I, J), J=1, 3)
    8   CONTINUE
      PRINT  102, L, LB
      PRINT  103
```

```
      DO 9 R    = 1, L
      PRINT  104, R, LP (R, 1), PX (LP (R, 1)), PY (LP
     $ (R, 1)), LP
     1    (R, 2), PX (LP(R, 2)), PY (LP (R, 2))
    9 CONTINUE
      PRINT  105
      DO 10 R   = 1, L
      PRINT  101, R, LL (R, 1), LL (R, 2), LR (R, 1), LR
     $ (R, 2)
   10 CONTINUE
      PRINT  106
      DO 11 R   = 1, F
      PRINT  101, R, (EL(R, S), S=1, 3)
   11 CONTINUE
   12 CONTINUE
C
C       CONTOURS ARE NOW DRAWN USING SUBROUTINE ISOVAR3.
C
      CALL ISOVAR3 (PX, PY, PZ, G, HTS, K, LL, LP, LR,
     $ UNUSED, N)
      RETURN
C
  100 FORMAT  ( 1H1, 9X, 1HI, 3X, 7HEP(I,1), 3X, 7HEP(I,2)
     $ , 3X,
     1   7HEP(I,3)  )
  101 FORMAT  ( 1H , 5I10 )
  102 FORMAT  (  *0  L =*, I3,  * LB =*, I3 )
  103 FORMAT  (  *1  LINE LIST* / 7X, 4HLINE, 6X, 4HEDGE,
     $ 9X, 1HX, 9X,
     1   1HY, 8X, 2HTO, 9X, 1HX, 9X, 1HY  )
  104 FORMAT  ( 1H , I10, 2(I10, 2F10.1) )
  105 FORMAT  (  *1  LINE LINKAGES* / 7X, 4HLINE, 3X,
     $ 7HLL(R,1), 3X,
     1   7HLL(R,2), 3X, 7HLR(R,1), 3X, 7HLR(R,2)  )
  106 FORMAT  (  *1  BOUNDARY LINES* / 3X, 8HTRIANGLE, 3X,
     $ 7HEL(R,1),
     1   3X, 7HEL(R,2), 3X, 7HEL(R,3)  )
C
      END
```

```
      SUBROUTINE  ISOVAR3 (PX, PY, PZ, G, HTS, K, LL, LP,
     $ LR, UNUSED,
     1   N)
      COMMON  /BUGBUG/  DBUG
      COMMON  /CLUES/  OPEN, FIRST, LAST, H
      COMMON  /ISOLIM/  L, LB
C
C         THIS PROCEDURE IS A SUBPROCEDURE OF ISOVAR2 BUT
C     $        CAN BE
C         USED SEPARATELY.  IT IS ALSO FOR CONTOURING OVER
C     $        AN
C         IRREGULAR TRIANGULAR MESH.  THE PARAMETERS
C     $        REQUIRED ARE
C         LB, THE NUMBER OF BOUNDARY LINES IN THE MESH,
C         L,  THE TOTAL NUMBER OF LINES IN THE MESH,
C     $        INTEGER ARRAYS
C         LP, LR, LL, WHERE LP(M,N), (N=1,2), ARE THE
C     $        LABELS OF THE
C         POINTS AT THE ENDS OF LINE M, AND THE ENTRIES IN
C     $        LR(M,N)
C         AND LL(M,N) SPECIFY THE LABELS OF THE LINES
C     $        JOINING AT
C         THESE POINTS.  ALSO NEEDED ARE THE ARRAYS PX, PY,
C     $        AND PZ
C         AND THE INTEGER K, WHICH ARE AS FOR ISOVAR2.
C     $        FULL DETAILS
C         ARE GIVEN IN SECTION 3 OF THE ORIGINAL REPORT.
C
C         THE BOUNDARY OF THE MESH IS NOW DRAWN.
C
      DIMENSION   HTS (K), LL (N,2), LP (N,2), LR (N,2),
     $ PX (G), PY
     1   (G), PZ (G)
      INTEGER     G, PA, PB, TEMP
      LOGICAL     DBUG, UNUSED (N)
      CALL DRAWBND (PX, PY, G, LP, N)
C
C         EACH LINE IS EXAMINED IN TURN, THE HEIGHTS OF ITS
C     $        END
C         POINTS EXTRACTED, AND THE ENTRIES IN THE ARRAYS
C     $        LP, LR,
C         AND LL INTERCHANGED IF NECESSARY SO THAT CONTOURS
C     $        ARE
C         TRACED WITH HIGH GROUND ON THE RIGHT.
C
      DO 1 M    = 1, L
      PA        = LP(M, 1)
      PB        = LP(M, 2)
      ZA        = PZ(PA)
```

```
      ZB          = PZ(PB)
      IF (ZA .GT. ZB) GO TO 1
      LP(M, 1)    = PB
      LP(M, 2)    = PA
      TEMP        = LP(M, 1)
      LP(M, 1)    = LP(M, 2)
      LP(M, 2)    = TEMP
      TEMP        = LL(M, 1)
      LL(M, 1)    = LL(M, 2)
      LL(M, 2)    = TEMP
    1 CONTINUE
      IF ( .NOT. DBUG) GO TO 3
      PRINT  100
      DO 2 M      = 1, L
      PRINT  101, M, LP (M, 1), LP (M, 2), LL (M, 1), LL
     $ (M, 2), LP
     1    (M, 1), LP (M, 2)
    2 CONTINUE
      PRINT  102
    3 CONTINUE
C
C       EACH CONTOUR HEIGHT IS DEALT WITH IN TURN.
C
      DO 6 J      = 1, K
      H           = HTS(J)
C
C       THE ARRAY UNUSED IS SET UP FOR THIS HEIGHT,
C    $       UNUSED(M)
C       BEING SET .TRUE. IF A CONTOUR LINE OF THIS HEIGHT
C       CROSSES LINE M.
C
        DO 4 M    = 1, L
        UNUSED(M) = PZ(LP(M, 1)).GE.H.AND.PZ(LP(M, 2))
     $ .LT.H
    4   CONTINUE
C
C       THE ARRAY UNUSED IS SCANNED FOR THE START OF A
C    *       CONTOUR
C       AND THE CONTOUR IS TRACED USING SUBROUTINE
C    *       FOLLOW.
C
        DO 5 M    = 1, L
C
C       WE USE I INSTEAD OF M HERE BECAUSE FOLLOW RESETS
C       THIS INTEGER AND WOULD MESS UP THE LOOP COUNT.
C
        I         = M
        IF (UNUSED(M) .AND. LP(M,1) .NE. 0) CALL FOLLOW
     $ (I,PX,PY,PZ,G
     1    ,LL,LP,LR,UNUSED,M)
```

```
    5       CONTINUE
    6     CONTINUE
        RETURN
C
  100   FORMAT  (  *1  REORDERED POINT AND LINE LISTS* /
     $ 10X, 1HM, 3X,
     1   7HLP(M,1), 3X, 7HLP(M,2), 3X, 7HLL(M,1), 3X,
     $ 7HLL(M,2), 3X,
     2   7HLR(M,1), 3X, 7HLR(M,2)  )
  101   FORMAT  ( 1H , 7I10 )
  102   FORMAT  ( 1H1  )
C
        END
```

```
      SUBROUTINE  FOLLOW (M, PX, PY, PZ, G, LL, LP, LR,
     $ UNUSED, N)
      COMMON  /BUGBUG/  DBUG
      COMMON  /ISOLIN/  L, LB
      COMMON  /CLUES/  OPEN, FIRST, LAST, H
C
C        THIS SUBROUTINE IS USED TO FOLLOW A PARTICULAR
C     $       CONTOUR
C        THROUGH THE MESH.  THE PARAMETER M IS THE LABEL
C     $       OF THE
C        LINE ON WHICH THE CONTOUR BEGINS.
C
      DIMENSION   LL (N,2), LP (N,2), LR (N,2), PX (G), PY
     $ (G), PZ (G)
      INTEGER     G, PA, PB, TEMP
      LOGICAL     DBUG, FIRST, LAST, OPEN, UNUSED (N)
      MSTART    = 0
C
C        THE PROCEDURE BEGINS BY SETTING THE INITIAL
C     $       VALUES OF
C        FIRST, LAST, AND OPEN.
C
      FIRST     = .TRUE.
      LAST      = .FALSE.
      OPEN      = M.LE.LB
      IF (DBUG) PRINT  100,H
C
C        NEXT POINT.
C        THE POINT WHERE THE LINE CROSSES THE LINE M IS
C        DETERMINED BY INVERSE LINEAR INTERPOLATION.
C
    1 PA        = LP(M, 1)
      PB        = LP(M, 2)
      ZA        = PZ(PA)
      ZB        = PZ(PB)
      TA        = 0.0
      DN        = ZA - ZB
      IF  (DN .NE. 0.0) TA =  (H - ZB) / DN
      TB        = 1.0 - TA
      XP        = TA * PX(PA) + TB * PX(PB)
      YP        = TA * PY(PA) + TB * PY(PB)
      IF (DBUG) PRINT  101,XP,YP
C
C        THE APPROPRIATE ENTRY IN THE ARRAY UNUSED IS SET
C     $       TO
C        .FALSE. UNLESS IT IS THE FIRST POINT OF A CLOSED
C        CONTOUR.  IF IT IS THE FIRST POINT OF A CONTOUR,
C     $       THE
C        LABEL OF THE INITIAL LINE IS STORED.  OTHERWISE,
C     $       TESTS
```

```
C          ARE CARRIED OUT TO SEE IF T IS THE LAST POINT ON
C     $          THE
C          CONTOUR.
C
           UNUSED(M) = FIRST.AND..NOT.OPEN
           IF (FIRST) GO TO 2
           LAST      = (OPEN.AND.M.LE.LB)
          $ .OR.(.NOT.OPEN.AND.M.EQ.MSTART)
           GO TO 3
         2 MSTART    = M
C
C          THE COORDINATES OF T ARE OUTPUT TO THE PROCEDURE
C     $          DRAWL.
C
         3 CONTINUE
           CALL DRAWL (XF, YF)
C
C          PROVIDED THAT IT IS NOT THE LAST POINT, THE LABEL
C     $          OF THE
C          NEXT LINE THAT THE CONTOUR CROSSES IS DETERMINED
C     $          AND THE
C          ROUTINE GOES BACK TO FIND A NEW POINT T.
C
           FIRST     = .FALSE.
           IF (LAST) RETURN
           TEMP      = LF(M, 1)
C
C          IN CASE WE END UP IN THE BOONIES
C
           IF (TEMP .EQ. 0) RETURN
           M         = LL(M, 1)
           IF  (UNUSED(TEMP)) M = TEMP
           GO TO 1
C
       100 FORMAT  (  *0  X AND Y FOR CONTOUR*, G15.7 )
       101 FORMAT  ( 5X, 2G15.7 )
C
           END
```

```
      SUBROUTINE  DRAWBND (PX, PY, G, LP, N)
      COMMON  /ISOLIM/  L, LB
C
C        THIS PROCEDURE IS USED TO DRAW THE BOUNDARY OF
C     *        THE MESH.
C        EACH BOUNDARY LINE IS SCANNED IN TURN, THE
C     *        COORDINATES OF
C        ITS END POINTS EXTRACTED, AND THE APPROPRIATE PEN
C        MOVEMENTS MADE.
C
      DIMENSION   LP (N,2), PX (G), PY (G)
      INTEGER     G, PA
        DO 1 I    = 1, LB
        PA        = LP(I, 1)
        CALL PLOT (PX(PA), PY(PA), 3)
        PA        = LP(I, 2)
        CALL PLOT (PX(PA), PY(PA), 2)
    1   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE  LABEL (R, S, PA, PB, E, EP, EL)
      COMMON  /ISOLIN/  L, LB
C
C         THIS PROCEDURE IS USED TO LABEL THE LINE JOINING
C         POINTS PA AND PB AND OPPOSITE TO THE POINT EP(R,
C     $         S)
C         IN ELEMENT R, AND TO PUT THE APPROPRIATE LABELS
C         IN THE ARRAY EL.
C
C         ALL ELEMENTS J (J .GT. R) ARE SEARCHED FOR A PAIR
C     $         OF
C         POINT LABELS IDENTICAL WITH PA AND PB.
C
      DIMENSION   EL (E,3), EP (E,3)
      INTEGER     E, EL, EP, PA, PB, QA, QB, QC, R, S, T
      JR          = R + 1
        DO 1 J    = JR, E
        QA        = EP(J, 1)
        QB        = EP(J, 2)
        QC        = EP(J, 3)
        T         = 3
        IF (PA .EQ. QB .AND. PB .EQ. QA) GO TO 2
        T         = 1
        IF (PA .EQ. QC .AND. PB .EQ. QB) GO TO 2
        T         = 2
        IF (PA .EQ. QA .AND. PB .EQ. QC) GO TO 2
    1   CONTINUE
C
C         NO PAIR OF LABELS HAS BEEN FOUND, SO THE LINE
C     $         MUST BE
C         A BOUNDARY LINE.  THE BOUNDARY LINE COUNT, LB, IS
C         UPDATED AND ASSIGNED TO EL(R,S).
C
      LB          = LB + 1
      EL(R, S)    = LB
      RETURN
C
C         INSIDE.
C
    2 CONTINUE
C
C         A PAIR OF LABELS HAS BEEN FOUND, SO THE LINE MUST
C     $         BE
C         AN INTERNAL LINE IN THE MESH.  THE LINE COUNT, L,
C     $         IS
C         UPDATED AND ASSIGNED TO THE APPROPRIATE TWO
C     $         ELEMENTS
C         IN THE ARRAY EL.
C
      L           = L + 1
      EL(R, S)    = EL(J, T) = L
      RETURN
      END
```

```
      SUBROUTINE  DRAWL (X, Y)
      COMMON  /CLUES/  OPEN, FIRST, LAST, H
      LOGICAL      FIRST, LAST, OPEN
      IF ( .NOT. FIRST) GO TO 1
      CALL PLOT (X, Y, 3)
      RETURN
    1 CALL PLOT (X, Y, 2)
      RETURN
      END
```