SANDIA REPORT SAND83—0011 • Unlimited Release • UC—32
Printed July 1983

SAND--83-0011
DE83 017422

A Portable FORTRAN Contour-Plotting Subprogram

Karen H. Haskell

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

DISCLAIMER

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

PAGES 1 to 2

WERE INTENTIONALLY

LEFT BLANK

SAND83-0011
Unlimited Release
Printed July, 1983

A PORTABLE FORTRAN CONTOUR-PLOTTING SUBPROGRAM

K. H. Haskell
Sandia National Laboratories
Albuquerque, NM 87185

ABSTRACT

A subprogram SCONTR( ) has been written that draws a contour plot of a function of two variables. The values of the function are required at a rectangular grid, but contour lines may be excluded on subregions of the rectangle. The subprogram allows the user to obtain a single plot with a minimum of difficulty. Other advanced features are available as options. The underlying graphics package is a small compact system developed at Sandia National Labs.

This page intentionally left blank

CONTENTS

This page intentionally left blank

I. Introduction

In this report we discuss a contour plotting Fortran subprogram. While contour plotting subroutines are available in many commercial plotting packages [5], this routine has the following advantages:

1. Since it uses the Weasel and VDI plot routines developed at Sandia [1,2], it occupies little storage and can be used on most of the Sandia time-sharing systems as part of a larger program. In the past, the size of plotting packages often forced a user to perform plotting operations in a completely separate program.

2. The contour computation algorithm (discussed in Section 2) is efficient and robust, and computes accurate contours for sets of data with low resolution.

3. The subprogram is easy to use. A simple contour plot can be produced with a minimum of information provided by a user in one Fortran subroutine call. Through the use of a wide variety of subroutine options, many additional features can be used. These include such items as plot titles, grid lines, placement of text on the page, etc.

The subroutine is written in portable Fortran 77, and is designed to run on any system which supports the Weasel and VDI plot packages. It also uses routines from the SLATEC mathematical subroutine library [4].

II. Computation of Contours

The contour values are computed using subprogram GCONTR( ), written by W. V. Snyder [3]. Much of the discussion in this section is from that paper.

A contour plotting algorithm can be constructed using one of two methods: 1) by following contours from some starting point until they either close or intersect a boundary; 2) or by examining each cell of the grid and drawing all contours found inside the cell. Advantages of contour following are that contour labeling is relatively easy and that the plotter pen makes fewer moves. Advantages of methods which draw all contours inside a cell are that less auxiliary storage is required, and that each cell can be processed before going on to the next. This would allow generation of contours over a much larger array than can be accommodated in main memory at one time. The subprogram GCONTR( ) uses method 1), following contours.

The data are presented at discrete points of a rectangular grid. The contour values are not necessarily equal to the dependent variable or function values at the nodes of the mesh. Thus a method must be used to estimate the point at which the contour intersects the edge of a cell. If nonlinear interpolation were used, there is the possibility of multiple intersections of a contour with an edge, or closed contours which intersect no edge. Faithfully following the interpolated surface would require finding the zeroes of a nonlinear bivariate function.

The subprogram GCONTR( ) uses linear interpolation. This interpolation method has the drawback that extrema can only occur at nodal points. If this does not produce sufficiently smooth or accurate contours, more function values could be computed, or the mesh could be refined. If the required storage is larger than desired, the independent variable plane could be divided into separately processed segments. Since different methods of interpolatory subtabulation work better for different problems, and GCONTR( ) uses a contour following method, it is more reasonable for the user to implement such a refinement.

Once the points of intersection of a contour with edges have been determined, they are joined by straight lines. When a contour intersects all four edges of a cell, a decision must be made as to how to connect the points. Many algorithms force the contours to cross inside the cell. The subprogram GCONTR( ) examines the linear interpolates along the "top" and "bottom" of the cell. If the top interpolate is less than the bottom interpolate, the contour line is drawn top-to-left, bottom-to-right. Otherwise, the contour line is drawn top-to-right, bottom-to-left. This method selects the same contours if the axes are exchanged.

The subprogram GCONTR( ) provides a means whereby designated elements of the array may be excluded. The edges connecting such elements to the other elements not excluded are not examined by the program. If data are not presented on a complete rectangle, extraneous and meaningless contours are avoided in the extension of the user's data domain.

The subprogram GCONTR( ) does not require the grid lines to be either straight or uniformly spaced. The output from GCONTR( ) is an ordered set of points which define the contour lines to be drawn. These points are connected using straight line segments by our subprogram SCONTR( ). Users with curved grid lines may wish to use GCONTR( ) directly and provide their own coordinate transformations to the sets of points before they are plotted.

III. Default Usage of the Subprogram

While SCONTR( ) permits a variety of options for the user, the subprogram is able to produce a reasonable plot using default parameters and a minimum amount of information from the user. (The options are discussed fully in the Section IV.)

The default usage will produce a contour plot with the following characteristics:

-- The lower left corner of the plot frame will occur at the lower left corner of the output device.

-- The plot will occupy a square area as large as the output device will permit.

-- The contour plot will be enclosed by a (square) frame.

-- Ten equally spaced contour values will be computed, labeled and plotted.

-- Contour lines will be computed for the entire rectangular input area.

The user must call SCONTR( ) with the following arguments:

```
CALL SCONTR (Z,LDZ,NX,NY,IOPT,COPT,ROPT,WORK,IWORK)
```

The subroutine arguments are defined as follows:

Input

Z(LDZ,*) A two-dimensional array of real values for which contours are to be drawn. The elements of Z(*,*) are assumed to lie on the nodes of a rectangular coordinate system.

LDZ The number of rows for the array Z(*,*), dimensioned in the calling program.

NX,NY The number of grid lines in the x and y coordinates, respectively

IOPT(*) An integer array for optional user input. For default usage it need be dimensioned at least 1, and IOPT(1)=99 assigned.

COPT A type CHARACTER string for optional user input. For default usage it need be only of length 1, and will not be referenced by the subroutine.

ROPT(*) A type REAL array for optional user input. For default usage it need be dimensioned only 1, and will not be referenced by the subroutine.

Work Arrays

WORK(*) A type REAL array of length at least max(NX,NY) used for internal working storage.

IWORK(*) A type INTEGER working array used for internal working storage. It must be declared of length at least

greatest integer part of [(2*NX*NY*NCV+ibit-1)/ibit]

where NCV = the number of contour values to be drawn (in the default case, NCV=10), and ibit = the number of useful bits in an integer word on the machine on which the subroutine is being called. (This value can usually be obtained as

ibit=I1MACH(8)

using the integer function subprogram I1MACH( ) from the SLATEC Library [4].) For example, on the CDC 6600, ibit=48; on the CRAY, ibit=63; on the VAX, ibit=31.

IV. Description and Usage of Subprogram Options

Through the use of optional input parameters in IOPT(*), COPT and ROPT(*), the user can request a variety of alternate output, such as labels and grid lines, or alter the position of the plot on the output device. A brief list of the options is given below. They are discussed in detail later in this section. It should be noted that the use of some options may cause the overall size of the contour plot itself to be reduced.

Option Key	Purpose
1	Print a title on the top of the page.
2	Label the horizontal axis.
Option Key	Purpose
3	Label the vertical axis.
4	Draw grid lines or tick marks.
5	Numerically label the axes.
6	Omit a portion of the Z(*,*) array from the plot.
7	Specify the number of contour levels to be drawn.
8	Provide the contour levels to be drawn.
9	Provide contour label symbols.
10	Do not list the contour levels drawn.
11	Omit the plot frame.
12	Omit the page frame.
13	Position the plot on the page.
14	Alter the overall size of the plot.
15	Give user control of plot initialization and termination.
16	Verify the lengths of the user-provided work arrays.
17	Return the contour values used for the plot.
99	No more options.

The integer array IOPT(*) is used to specify the desired user options. When one of the option keys above is used in IOPT(*), together with possible input in COPT or ROPT(*), optional subprogram action is requested. The usual format for the option array is:

```
IOPT(1)    = key1
IOPT(2)    = data for option specified by key1, or a pointer into
             COPT or ROPT(*) to obtain that data.
     .
     .
     .
IOPT(...) = key2
     .
     .
     .
IOPT(...) = 99, indicating no more options.
```

If the key value for a particular option in the array IOPT(*) has a negative value, that option will be ignored. This feature is useful for turning off an option without altering the entire option array's contents in the calling program.

The options can appear in IOPT(*) in any order. Most options require, in addition to the key value, additional locations in IOPT(*), COPT or ROPT(*) to transmit input values. Following the detailed discussion of the options and their required input (below), Table 1 summarizes the number of additional storage locations required in IOPT(*), COPT, and ROPT(*) for each option. The minimum storage required for any option is a single location in IOPT(*).

Key Value	Description of Option and Other Input
1	Print a plot title along the top of the plot. The data values for this option are: 1. The location in the COPT character string where the title begins. 2. The number of characters in the title.
2	Label the horizontal axis with a specified title. The data values for this option are: 1. The location in the COPT string where the title begins. 2. The number of characters in the title.
3	Label the vertical axis with a specified title. The data values for this option are: 1. The location in the COPT string where the title begins. 2. The number of characters in the title.
4	Draw grid lines on the plot area. The data values for this option are: 1. The type of grid lines to draw. Possible choices are: =1, draw plot frame only (this is default); =2, draw tick marks along the axes; =3, draw grid lines 2. The number of horizontal grid lines (equally spaced). 3. The number of vertical grid lines (equally spaced).

5 Numerically label the horizontal and vertical axes at grid lines. (Option #4 must also be specified.) The data values for this option are:

1. A pointer into the ROPT(*) array to the minimum horizontal axis value.
2. A pointer into the ROPT(*) array to the maximum horizontal axis value.
3. A pointer into the ROPT(*) array to the minimum vertical axis value.
4. A pointer into the ROPT(*) array to the maximum vertical axis value.

6 This option specifies that any point in the Z(*,*) array which is equal to a specified "special" value will not have any contours drawn through it. (This option is useful for excluding a region of the Z(*,*) array from the plot that does not correspond to the problem domain.) The data value for this option is:

1. A pointer into the ROPT(*) array to the special value.

7 This option changes NCV, the number of contour levels that are computed and drawn by the subroutine. The data value for this option is:

1. The number of contour levels to be drawn.

8 User-provided contour levels are to be drawn. If option #7 is not also specified, there must be 10 values provided. If option #7 is specified, the number of values specified there must be provided. The data value for this option is:

1. A pointer into the ROPT(*) array to the first contour value specified.

9 Label each contour value with a single character. The data value for this option is:

1. A pointer into COPT to the first of NCV (single) characters to be used to label the contours. (A blank is a valid character).

10 Do not list the contour values along the right of the plot. There are no additional data values for this option.

11 Do not draw a frame around the plot. (A frame will be drawn by default.) There are no additional data values for this option.

12 Do not draw a frame around the page. (A frame will be drawn by default.) There are no additional values for this option.

13 Draw the plot relative to a user-specified plot frame coordinate. (By default, the lower left corner of the plot frame is in the lower left corner of the output device.) The data values for this option are:

1. A pointer into the ROPT(*) array to the horizontal VDI coordinate for the lower left corner of the plot frame.
2. A pointer into the ROPT(*) array to the vertical VDI coordinate for the lower left corner of the plot frame.

14 Size the plot according to a user-specified plot frame coordinate. By default, the plot is drawn to use the maximum amount of the output device display area. This option allows the upper right corner of the plot frame to be specified by the user. The data values for this option are:

1. A pointer into the ROPT(*) array to the horizontal VDI coordinate for the upper right corner of the plot frame.
2. A pointer into the ROPT(*) array to the vertical VDI coordinate for the upper right corner of the plot frame.

15 Do not initialize or terminate the plot package within the subroutine. (This option, along with options #13 and #14, is useful for drawing several plots on the same page.) There are no additional data values for this option.
NOTE 1: If this option is specified, the user is responsible for calling the Weasel initialization and termination routines WSTART( ) and WEND( ) in the calling program for proper program execution.
NOTE 2: If several contour plots are drawn on the same page using this option, the subroutine will ring the terminal bell and pause after each plot. Entering a carriage return will cause it to continue.

16 Verify that the declared lengths of the work arrays WORK(*) and IWORK(*) are long enough. The data values for this option are:

1. The declared length of WORK(*) in the calling program.

2. The declared length of IWORK(*) in the calling program. It is an error if these input values do not satisfy the subprogram requirements.

17 Return the contour values used in the subprogram (whether computed by SCONTR( ) or provided by the user) to the user in the ROPT(*) array. This option is useful if the region to be plotted is subdivided, but it is desired to draw the same contour lines in each subregion. The data value for this option is:

1. A pointer into the ROPT(*) array where the NCV contour values will be returned.

99 No more options.

TABLE 1.

Number of locations required in option arrays

Option	IOPT(*)	COPT	ROPT(*)
1	3	No. of chars.	0
2	3	No. of chars.	0
3	3	No. of chars.	0
4	4	0	0
5	5	0	4
6	2	0	1
7	2	0	0
8	2	0	NCV
9	2	NCV	0
10	1	0	0
11	1	0	0
12	1	0	0
13	3	0	2
14	3	0	2
15	1	0	0
16	3	0	0
17	2	0	NCV
99	1	0	0

V. Demonstration Program and Examples

To demonstrate the use of SCONTR( ) and show examples of some of its options, a program has been written. For any of 8 example problems, the program will produce contour plots using a variety of the options discussed above. The examples available are:

1. Default usage

2. Plot title and grid lines

3. Axes labeling, tick marks, and axes numbering

4. User-specified contour values, labelled and listed (Page and plot frames are omitted)

5. Mask off a portion of the Z(*,*) array

6. Move plot to upper right corner of page

7. Draw two plots on the same page

8. Partition the region to be plotted; plot with consecutive calls to the subroutine. Refine contours in lower left corner of the frame.

The microfiche accompanying this report contains a complete listing of the demonstration program. In particular, the Fortran code required to produce each plot is listed.

The Appendix contains the eight sample plots produced by the demonstration program. Example 1 illustrates the default usage of the subroutine. In example 4, the use of several options is shown.

MRS:sdc:7205A:06/03/83X

VI. References

[1] WEASEL Reference Manual, September, 1982, Internal Sandia document, Div. 2644

[2] Karen M. Erickson and Randall W. Simons, "Functional Specification of the Sandia Virtual Device Interface (SVDI)", SAND81-1900, February, 1982.

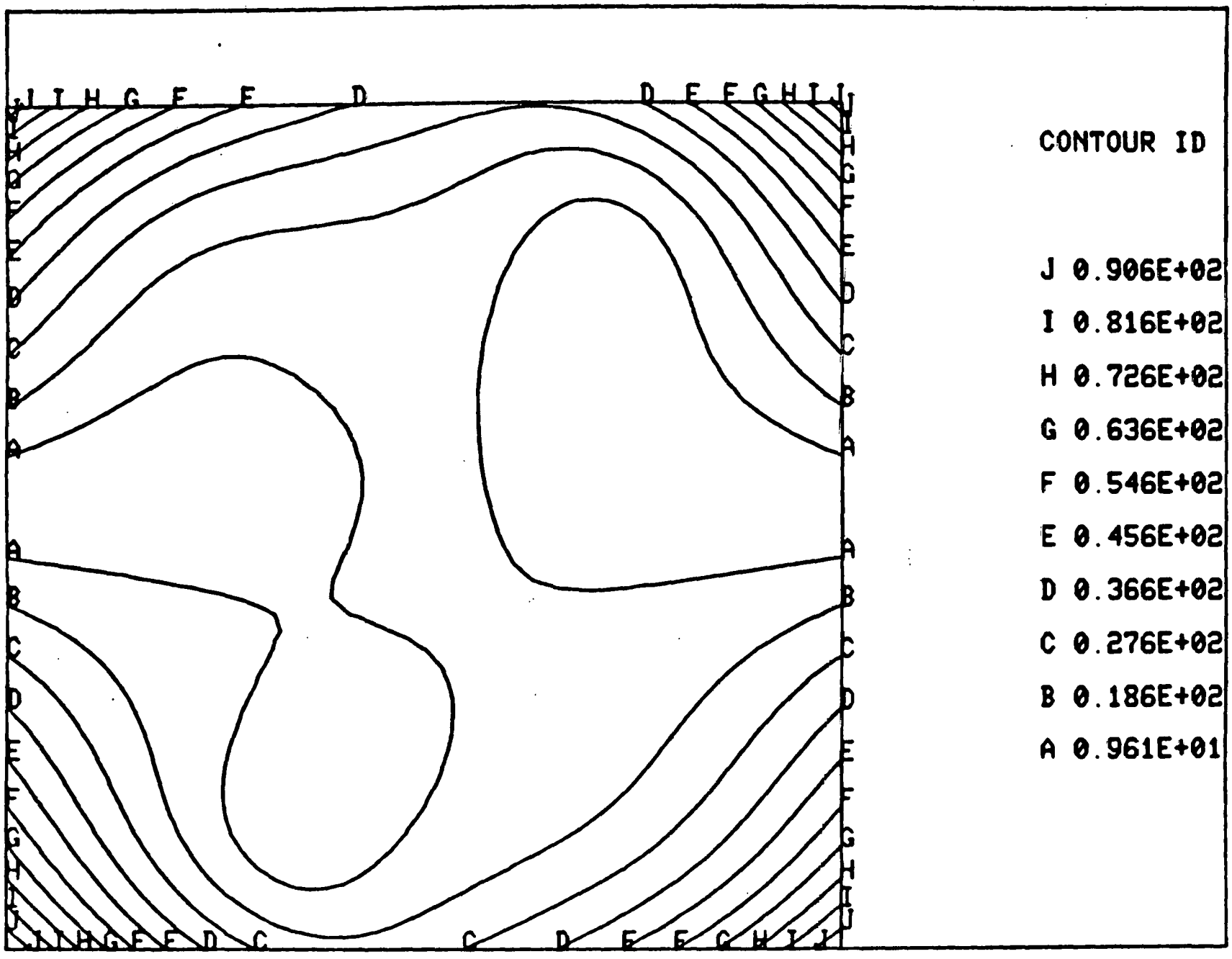
[3] W. V. Snyder, "Algorithm 531 Contour Plotting", ACM Trans. on Math. Software 4, No. 3, Sept., 1978, pp. 290-294.

[4] K. H. Haskell, W. H. Vandevender, E. L. Walton, "The SLATEC Common Mathematical Subprogram Library: SNLA Implementation," SAND80-2792, Dec., 1980.

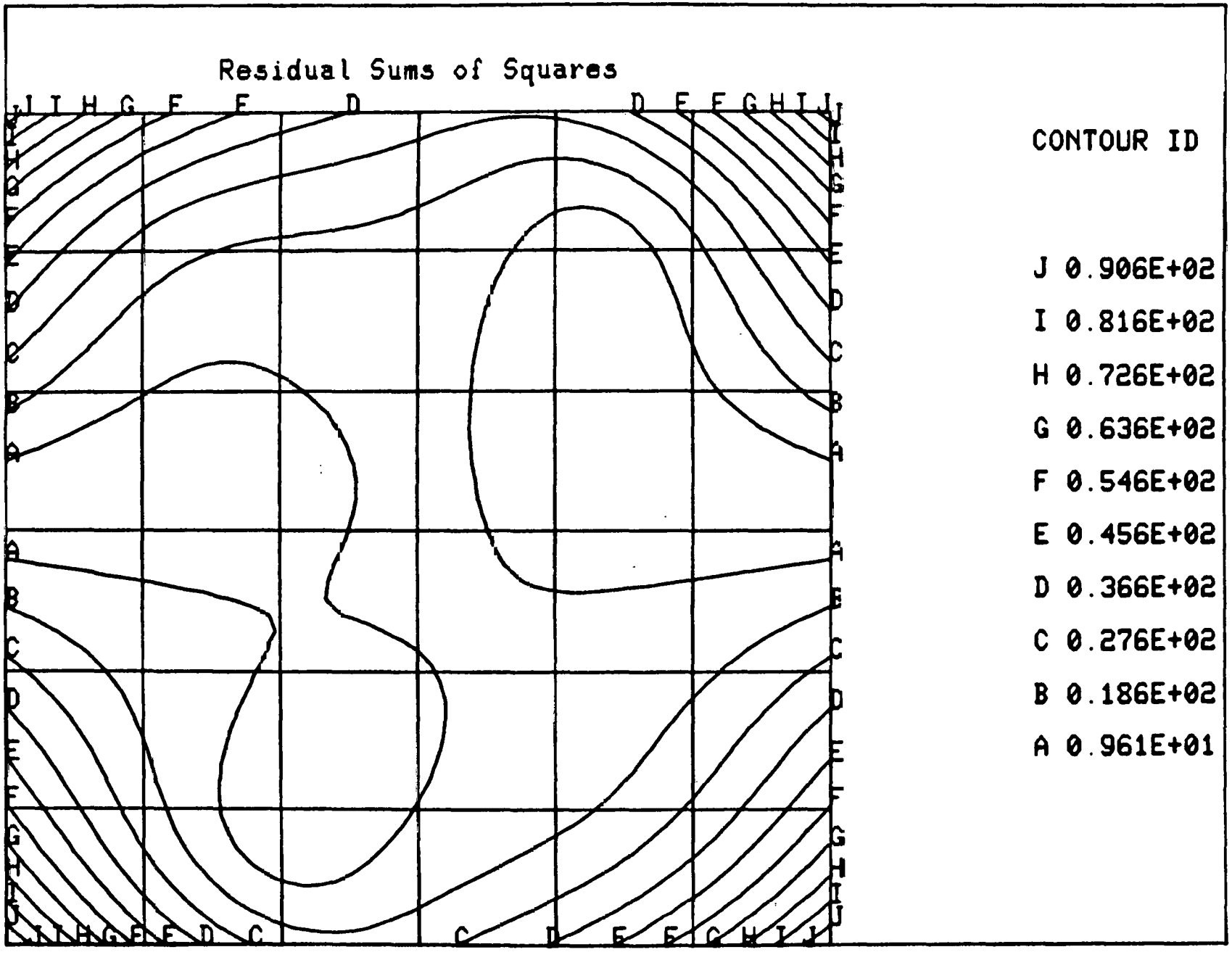
[5] DISSPLA User's Manual. Integrated Software Systems Corporation. Various dates, 1970-1978, San Diego, CA 92121

Remark on the Report

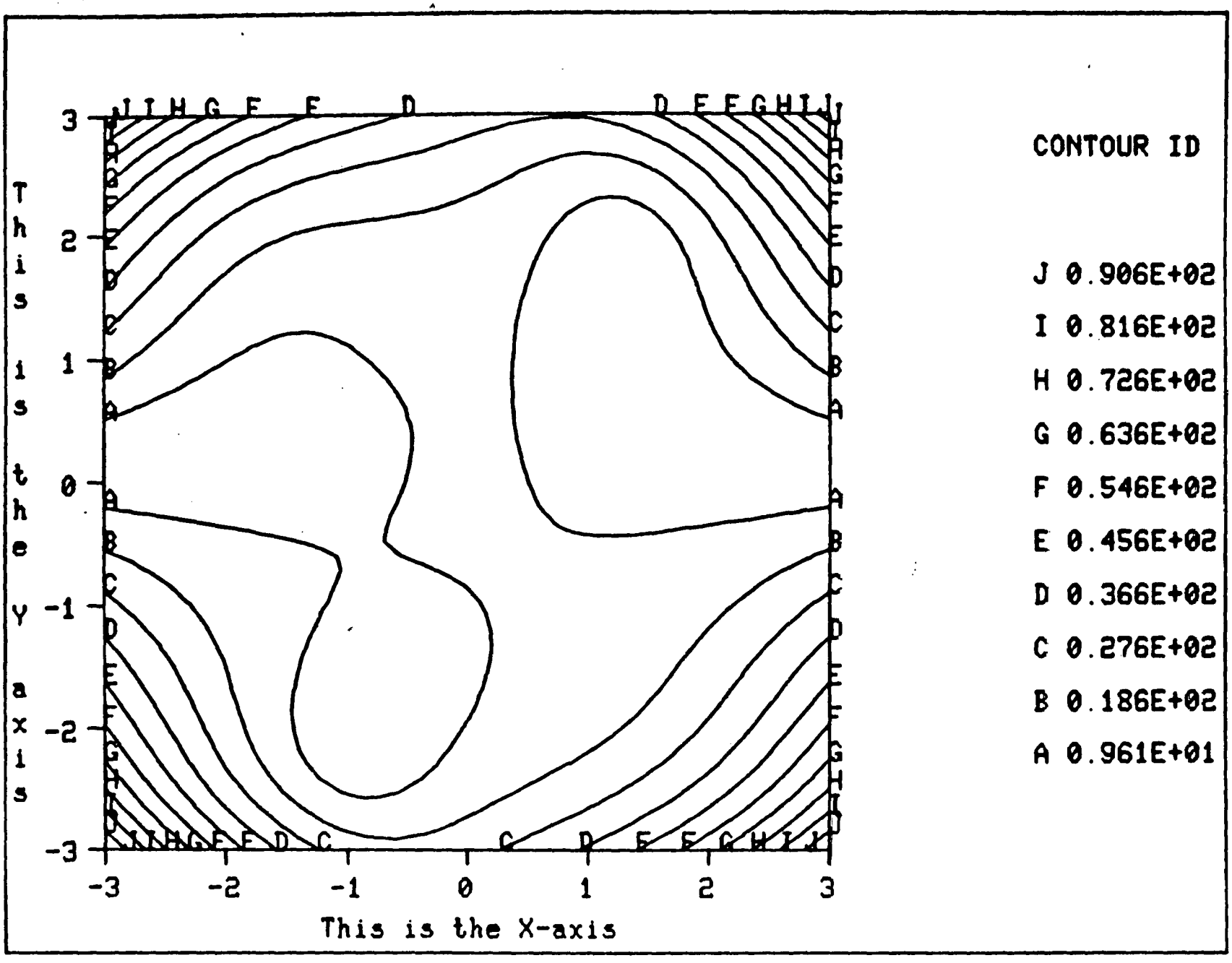
This report was partially prepared by R. J. Hanson, 2646. Some modification of the code package was also done by Hanson. (The author of the report is no longer employed at Sandia National Labs.)



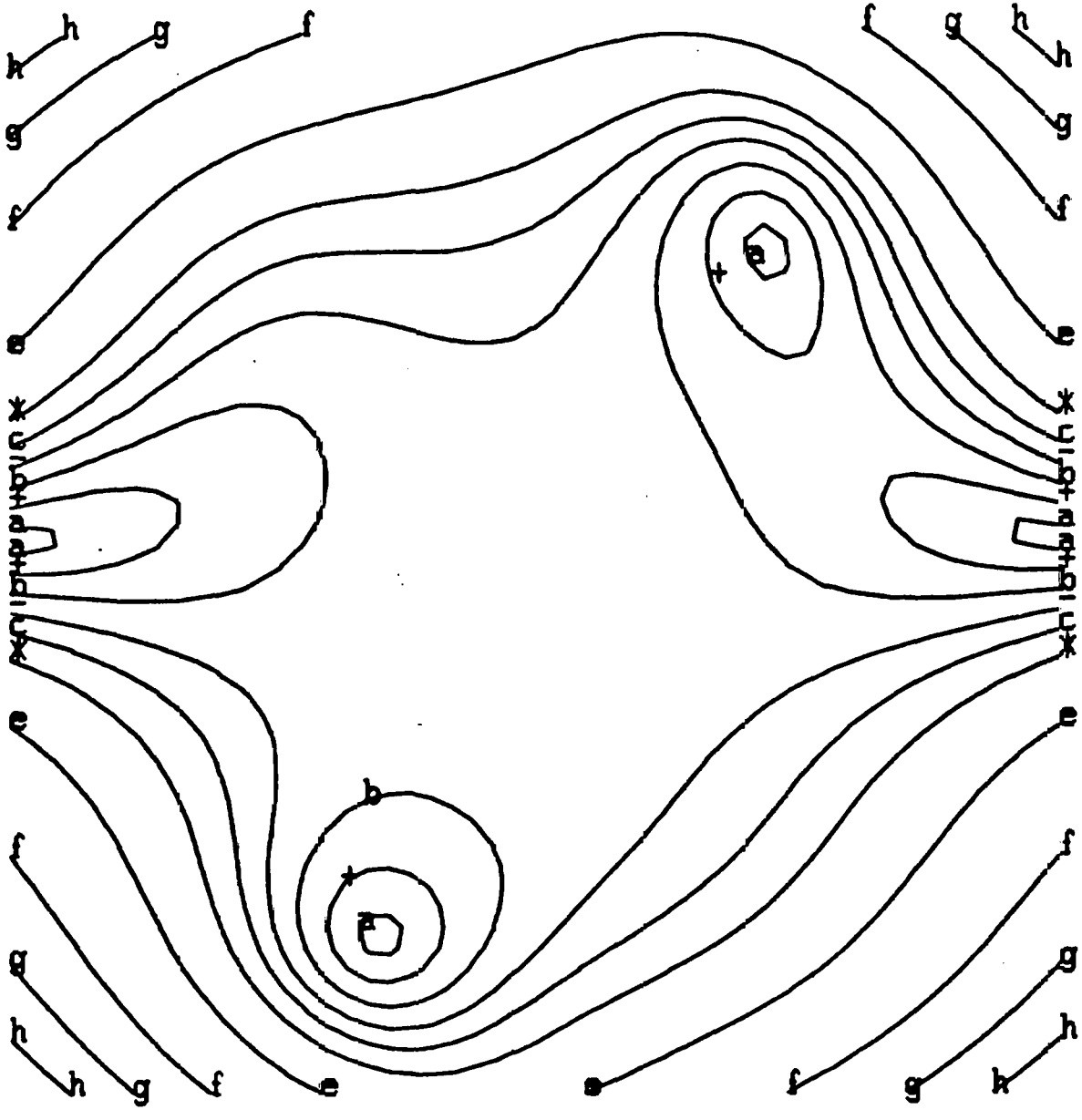
Example 1. Default Usage



Example 2. Usage with Plot Title and Grid Lines



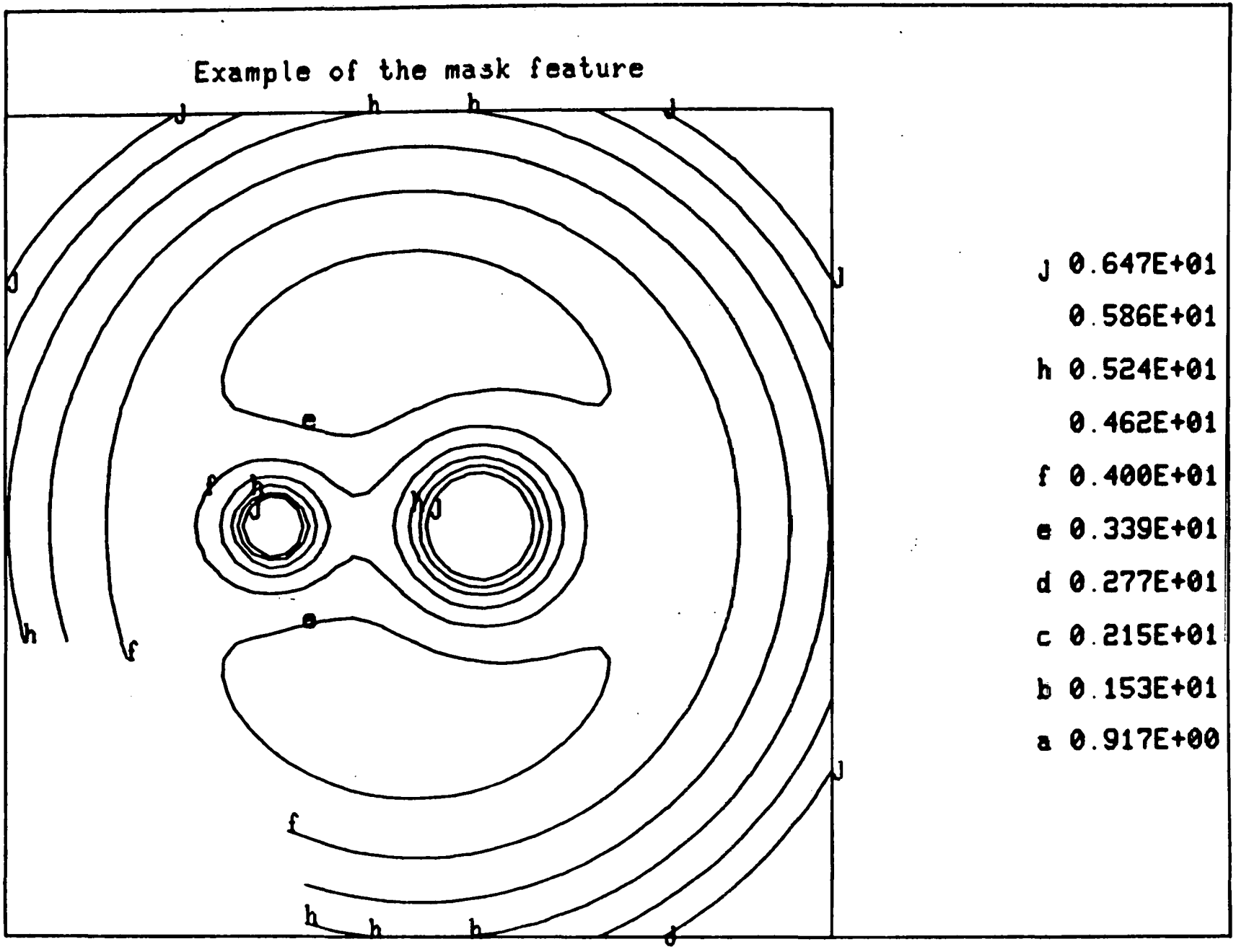
Example 3. Usage with Axis Labeling, Scale Tick Marks, and Axis Scaling



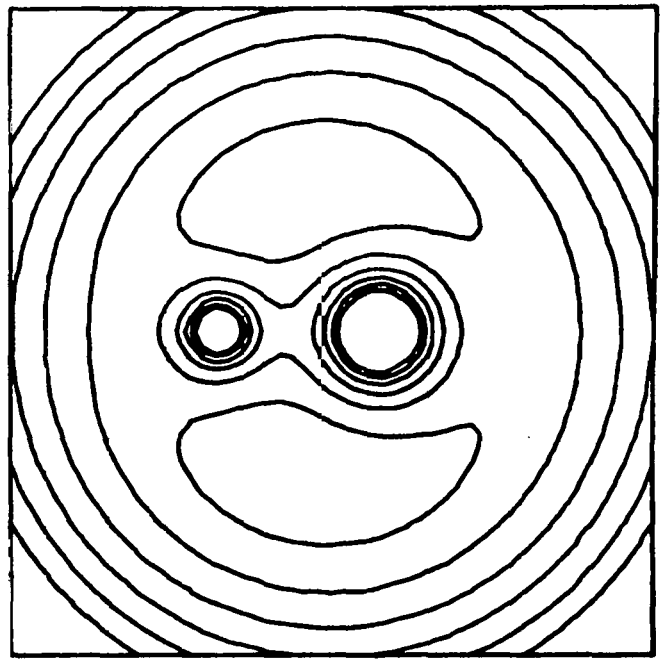
CONTOUR ID

h 0.816E+02
g 0.636E+02
f 0.456E+02
e 0.276E+02
* 0.180E+02
c 0.140E+02
- 0.110E+02
b 0.800E+01
+ 0.500E+01
a 0.200E+01

Example 4. User-Specified Contour Values and Labels. Page and Plot Frames are omitted.

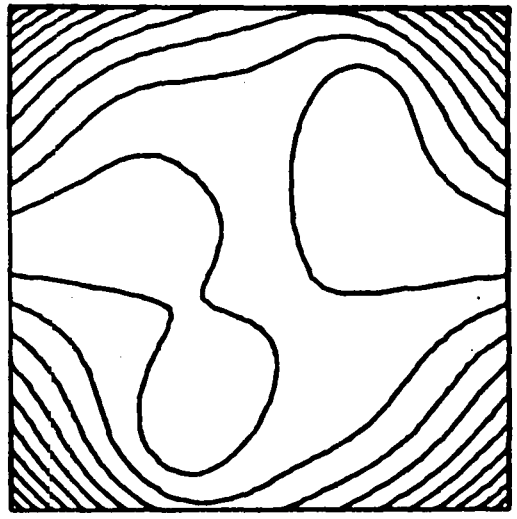


Example 5. Mask off a Near-Circular Segment of the Data Array

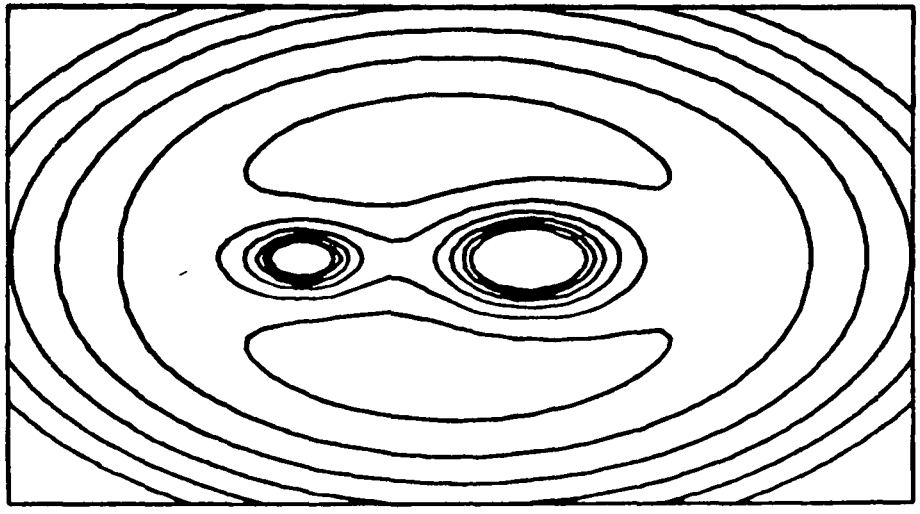


Example 6. Move the Plot to Upper Right Corner of the Page

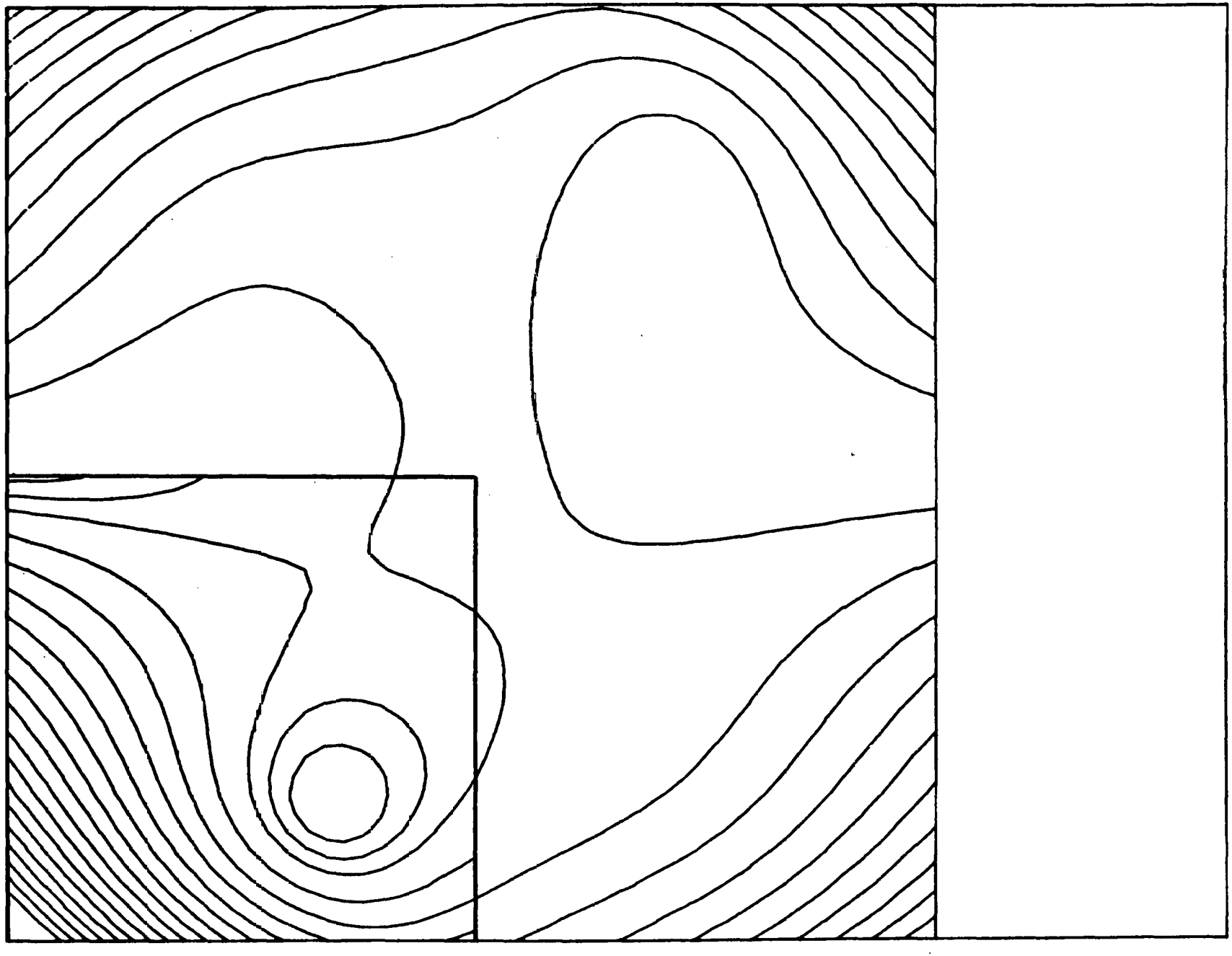
First Plot



Second Plot

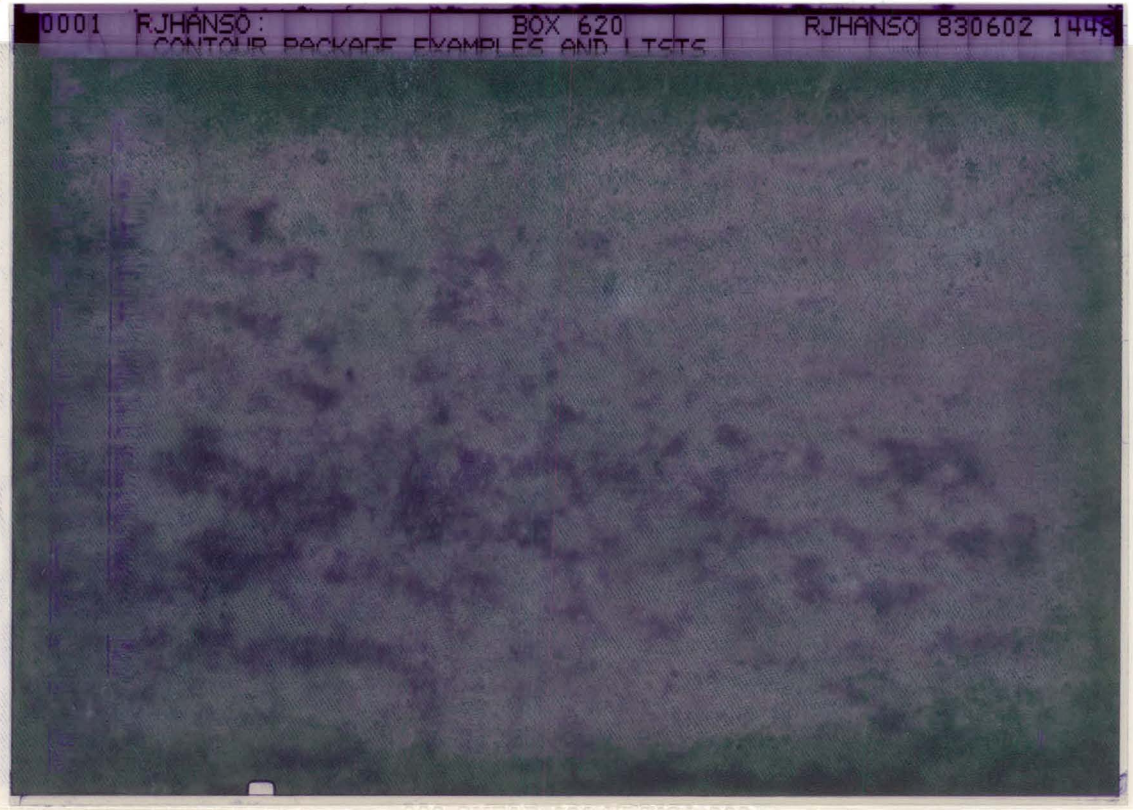


Example 7. Draw Two Plots on the Same Page



Example 8. Partition the Region to be Plotted. Refine Contours in Lower Left Corner of the Page.

Distribution:
1231 Lynn Kissel
1642 Don Amos
2640 Jack Tischhauser
2644 All Personnel
2646 All Personnel
2646 R. Hanson (5)
7112 Frank Biggs
8328 James Lathrop
9444 Sam Thompson
8214 M. A. Pound (1)
3141 L. J. Erickson (5)
3151 W. L. Garner (3)
For DOE/TIC
DOE/TIC (25)
(C. Dalin, 3154-3)


0001 RJHANSO:
BOX 620
RJHANSO 830602 1448

DO NOT MICROFILM
COVER

Org.	Bldg.	Name	Rec'd by	Org.	Bldg.	Name	Rec'd by

Sandia National Laboratories