COMPUTER SOFTWARE PACKAGE FOR 3-D PLOTTING

DEVELOPMENT OF COMPUTER SOFTWARE PACKAGE FOR 3-D PLOTTING AND SOME APPLICATIONS

by

PETER P. ZACHAR (Dipl. Eng.)

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Engineering

McMaster University

March, 1972

MASTER OF ENGINEERING (1971) McMASTER UNIVERSITY

(Design) Hamilton, Ontario

TITLE: Development of Computer Software Package for 3-D Plotting and Some Applications

AUTHOR: Peter P. Zachar (Dipl. Eng.)

SUPERVISOR: Professor M. C. de Malherbe

NUMBER OF PAGES: 227, xiv

SCOPE AND CONTENTS:

This thesis is a design of a three-dimensional plotting routine for the computer facilities at McMaster University with a subroutine to adopt the package for use on other computer systems and is divided into four sections.

Section A is a review of the necessity of computerized plotting in science and engineering design with particular emphasis on software sophistication, which is the subject of this work.

Section B describes the principles employed and the basic logic of the software package. The method by which a three-dimensional solid is mapped is also explained.

Section C is a complete user's guide to the package.

Section D deals with some applications for the package such as the solution of design optimization problems, visual analysis of data sets, an aid to teaching and combined interpolation and display of experimental results.

ACKNOWLEDGEMENTS

The author expresses his sincere gratitude to Dr. M. C. de Malherbe of McMaster University for invaluable help and guidance given during and before this project.

Dr. G. L. Keech, Director of McMaster Computing Centre is gratefully acknowledged for his help and understanding and his assistance in making available the facilities of his department.

Co-operation and help of Computer Centre's Academic Systems Group and computer operating personnel is also acknowledged and appreciated.

The author also wishes to thank Marg Hruboska for her dedicated and expert typing of the manuscript.

The author is particularly indebted to his parents whose guidance and encouragement is much appreciated.

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF TABLES

SECTION A

BACKGROUND

CHAPTER 1

INTRODUCTION

1.1 Abstract

A computer program and a user's guide is developed for representing three-dimensional surfaces on a two-dimensional plane.

The program is user-oriented and the way the input is handled by the program makes it very easy and convenient for the user to prepare.

Given a function $z = z(x,y)$ the program evaluates and plots, in orthogonal dimetric projection the values of z for any x and y variables while eliminating all hidden lines. The program also plots and scales on the same diagram the axes, marks the maximum and the minimum points if they are visible, writes on the plot the parameters controlling the plotting, and prints any heading specified.

An important feature of this package is that the function $z(x,y)$ does not have to be a continuous function.

The function can also be supplied as discreet points and if the function is known to be smooth and the array is not complete, the program can interpolate between known values using the method of "cubic spline fit" and plot the smooth 'complete' surface.

Various applications are given in the field of optimization techniques, function analysis, data evaluation, and teaching.

1.2 Fundamentals of the Problem

It is well known that, in general, pictures of surfaces can convey certain messages faster and easier than any other form of communication.

A designer's or scientist's world is full of formulas, equations, sets of data or experimental results and he or she is very often forced with the problem of presenting the results or analyzing the formulas or data sets obtained.

Figures, graphs and models are used to describe this abstract data in many cases.

For the relationship between two variables, one independent, the other dependent, it is easy to draw a plot in two dimensions. Such a plot may be drawn by hand, but for a fast display or a display of repeating data sets a computer may be utilized and its use in this field is growing rapidly. The problems arising from the use of computers in a graphic field are not of hardware availability, but of software sophistication.

The basic level software for digital plotters is well developed and adequate for any two-dimensional drawing (1,2). However, if a relationship between three variables is to be represented on the plotter, a plotting program has to be written to perform the task. To satisfy the need for higher level software there are basically two ways of approaching the problem of representing a function of two independent variables.

The surface representing the function may be represented by a set of curves, that join the points of equal values (2,3). These are called contours and are analogous to isotypes on topographic maps or isobars on the weather maps and are very convenient for quantitative evaluation of the functions. The flexibility of contour plotting programs usually allows the user to specify the values at which the contours are to be drawn and, therefore, relevant numerical information may be extracted from the plot.

The other approach to representation of relationships between two independent variables is a planar projection of the surface, which is formed by a dependent variable as a function of the two independent variables. This type of representation emphasizes the qualitative aspects. Properties, rather than actual numerical values of the function are of interest in this case.

There are several types of projections, that may be used as well as several ways of visualizing the function.

Among the methods of representing three-dimensional figures on a two-dimensional surface, the following methods are most common: (5,6)

Perspective Projection

Perspective projection is as close as possible the same as viewing the object itself in the space (7). This type of projection, however, is not directly measurable.

Orthographic Projection

Orthographic projection produces an acceptable pictorial view and is easier to measure than perspective projection, as all parallel lines are equally foreshortened.

Among several types of orthographic projections such as trimetric, dimetric, isometric or oblique, the dimetric projection seems to be especially suitable for computer generated drawings. Two principle axes are foreshortened in the same ratio, i.e., the line of sight is equally inclined to two of the principle planes if using dimetric projection.

Once the type of projection is determined, there are various methods and techniques for displaying the surface. The surface is most commonly represented by a set of curves that join the points at the surface. The set of points that lie on each of the curves may be generated using four basic techniques:

- for constant z find the x and y values and plot z
- for constant x vary y and plot z
- for constant y vary x and plot z
- vary x and y using given algorithm and plot z

The first method is similar to contour plotting and not used much in computer derived planar drawings.

The last three methods are very common and the method chosen for this thesis is a combination of the second and third methods which results in shading of the drawing if the density of the curves is sufficient.

As the sophistication of the display increases, the program becomes more complicated. This can be shown if the length of the plot program in reference (4) for contours is compared with the length of the three-dimensional program in this work.

Most of the work in three-dimensional computer generated displays is centered on functions such that $z=z(x,y)$ where z is continuous and smooth (8). The need for plots of discreet and piecewise continuous functions arises when the fields of topography (9) statistics, vibrations or acoustics are studied. Wherever random numerical data is presented, a 3-D plot helps, when available, to visualize the data (10,11).

When functions are discontinuous at one or several points or along certain curves, the plots of the functions become more valuable since the visualizing of these functions becomes more difficult.

The visualizing of the intersection of two surfaces also becomes simple with the aid of 3-D plots. This thesis also deals with the plotting of any surface $z=z(x,y)$ intersecting the surface $y_1=y_1(x)$.

The art of plotting of intersections and/or discontinuous surfaces offers great possibility for advances in the computer graphics field.

SECTION B

BASIC PRINCIPLES

CHAPTER 2

ASSUMPTIONS AND DEFINITIONS

The basic logic of the program requires a two dimensional rectangular or square array (matrix) of values, which represents a surface, which is a function of two independent variables, x1 and x2 as an input. Elements of the matrix represent a grid of points of a rectangular mesh dividing the area defined by the lower and upper limits of x1 and x2 into equal intervals for each variable.

The values stored in the matrix are represented by the function:

$$u = u(x1,x2) \qquad (2.1)$$

Elements of the matrix are represented by the function

$$u_{ij} = u(x1_{min} + (i-1)\Delta x1, x2_{min} + (j-1)\Delta x2) \qquad (2.2)$$

where

$x1_{min}$ and $x2_{min}$ are the lower limits of the variables x1 and x2 respectively

$$\Delta x1 = \frac{x1_{max} - x1_{min}}{n} \qquad (2.3)$$

$$\Delta x2 = \frac{x2_{max} - x2_{min}}{m} \qquad (2.4)$$

where: n is the number of intervals along the x1 axis

m is the number of intervals along the x2 axis

To store all the values x1 and x2, a matrix of dimension (n+1,m+1) is required.

If the surface is restricted in an area within the range of x1 and x2, all elements falling within this restricted area are flagged. These flagged elements are assumed to have zero elevation with respect to the base even though their values might be known.

An algorithm was written to draw the surface by scanning the matrix, first column wise, then row wise. The pen is directed to join the neighbouring values, subject to the condition that hidden lines are not drawn. It is assumed that the function representing the surface exists and is continuous between two neighbouring non-flagged values. Thus it is possible to join the values by a straight line. The surface is assumed not to exist between a flagged and non-flagged value, as well as between two flagged values, thus a grid pattern on the base is mapped between all values where the surface does not exist.

It is further assumed that if both points of a line segment joining two elements are visible/hidden, then all points on this segment are visible/hidden. This assumption is true in the majority of cases and should it cause a significant inaccuracy in a particular mapped plot, a change in the x1 and/or the x2 intervals or a finer grid will remedy the problem.

The surface is assumed not to exist at all between two flagged values. It is assumed to exist between two non-flagged values and to either not exist or partly exist between a flagged and a non-flagged value.

CHAPTER 3

GENERATING AND PREPARING THE ARRAY FOR PLOTTING

The array of values defining the surface undergoes several processes before the drawing routines are called to map the matrix on the paper (see Figure 3.1).

3.1 Generating the Array

The matrix is initially stored for plotting by a subroutine called GETMAT. All elements are flagged as non-existent by setting them to a dummy quantity before any operation on the matrix starts.

The values of the matrix are supplied in two basic ways:

a) Internally, i.e., using a function representing the surface,

b) Externally, i.e., the matrix is supplied from a source outside the plotting package and is merely transferred to the program.

In the latter case, if not all the elements are known, or defined, the matrix can be completed by an interpolating routine included in the package.

3.2 Application of Area Restrictions

Area restrictions which represent constraints on the data contained in the matrix are introduced by means of a subroutine called CONST. The matrix is searched to find and flag any element that lies in the restricted area. In addition, the matrix is further searched after elimination of elements in restricted areas for single elements surrounded by the flagged ones. Such elements, if found are also flagged. The second elimination procedure is included because of the necessity to map the surface clearly, i.e., to eliminate the drawing of single lines that represent isolated values within a restricted area.

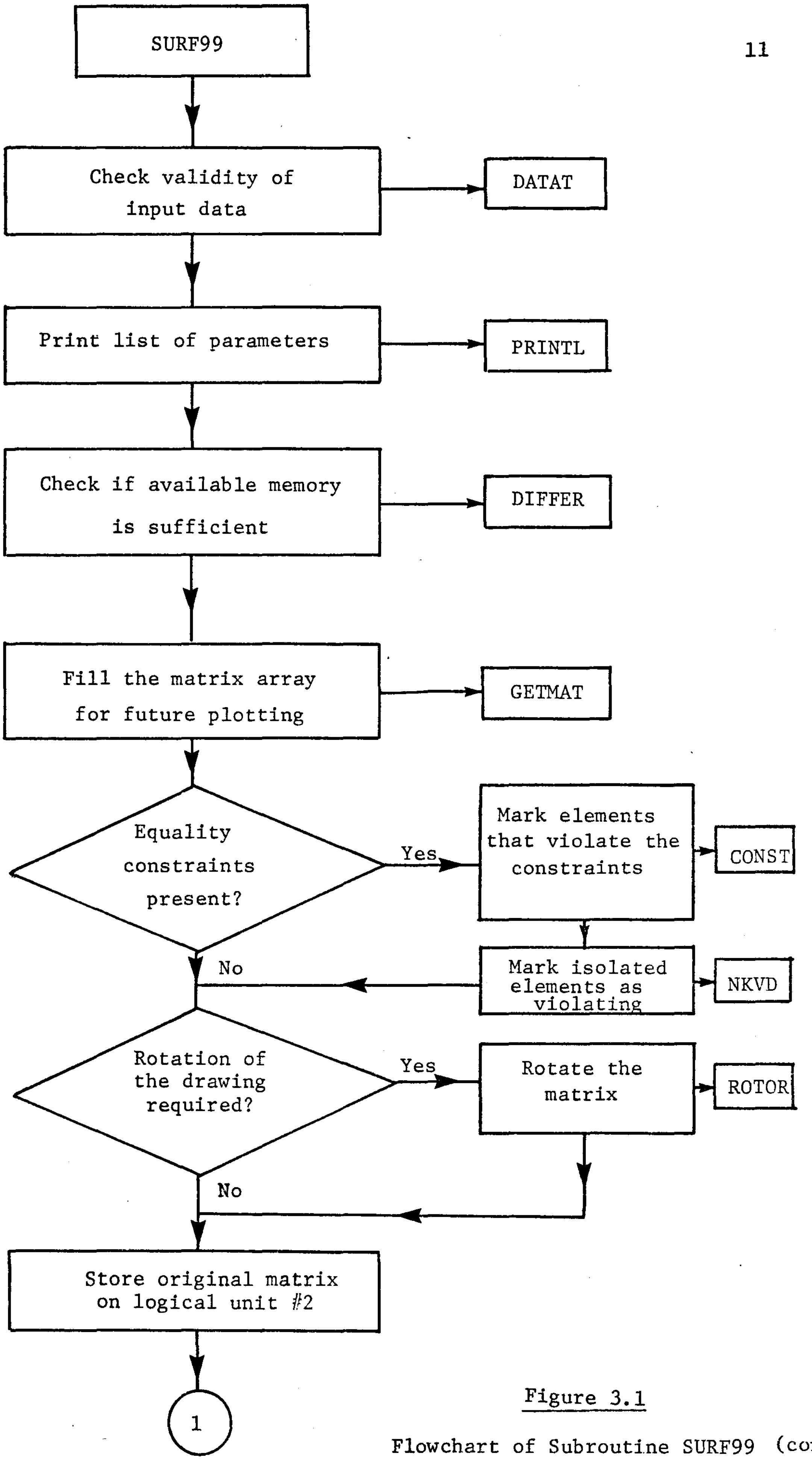

SURF99
Check validity of input data
DATAT
Print list of parameters
PRINTL
Check if available memory is sufficient
DIFFER
Fill the matrix array for future plotting
GETMAT
Equality constraints present?
Yes
Mark elements that violate the constraints
CONST
Mark isolated elements as violating
NKVD
No
Rotation of the drawing required?
Yes
Rotate the matrix
ROTOR
No
Store original matrix on logical unit #2
1

Figure 3.1

Flowchart of Subroutine SURF99 (cont'd)

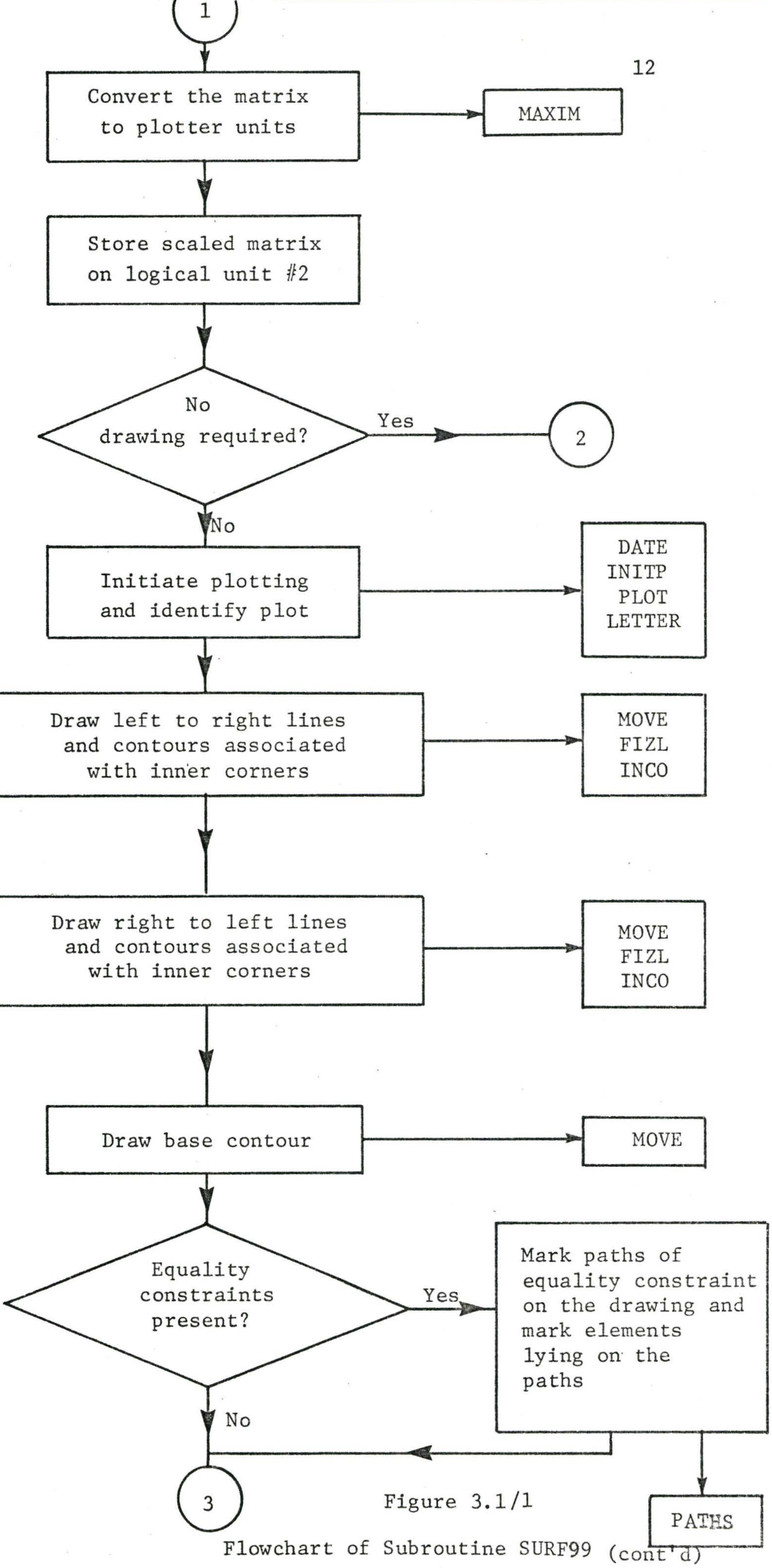

1
Convert the matrix to plotter units
MAXIM
Store scaled matrix on logical unit #2
No drawing required?
Yes
2
No
Initiate plotting and identify plot
DATE
INITP
PLOT
LETTER
Draw left to right lines and contours associated with inner corners
MOVE
FIZL
INCO
Draw right to left lines and contours associated with inner corners
MOVE
FIZL
INCO
Draw base contour
MOVE
Equality constraints present?
Yes
Mark paths of equality constraint on the drawing and mark elements lying on the paths
No
3
PATHS

Figure 3.1/1

Flowchart of Subroutine SURF99 (cont'd)

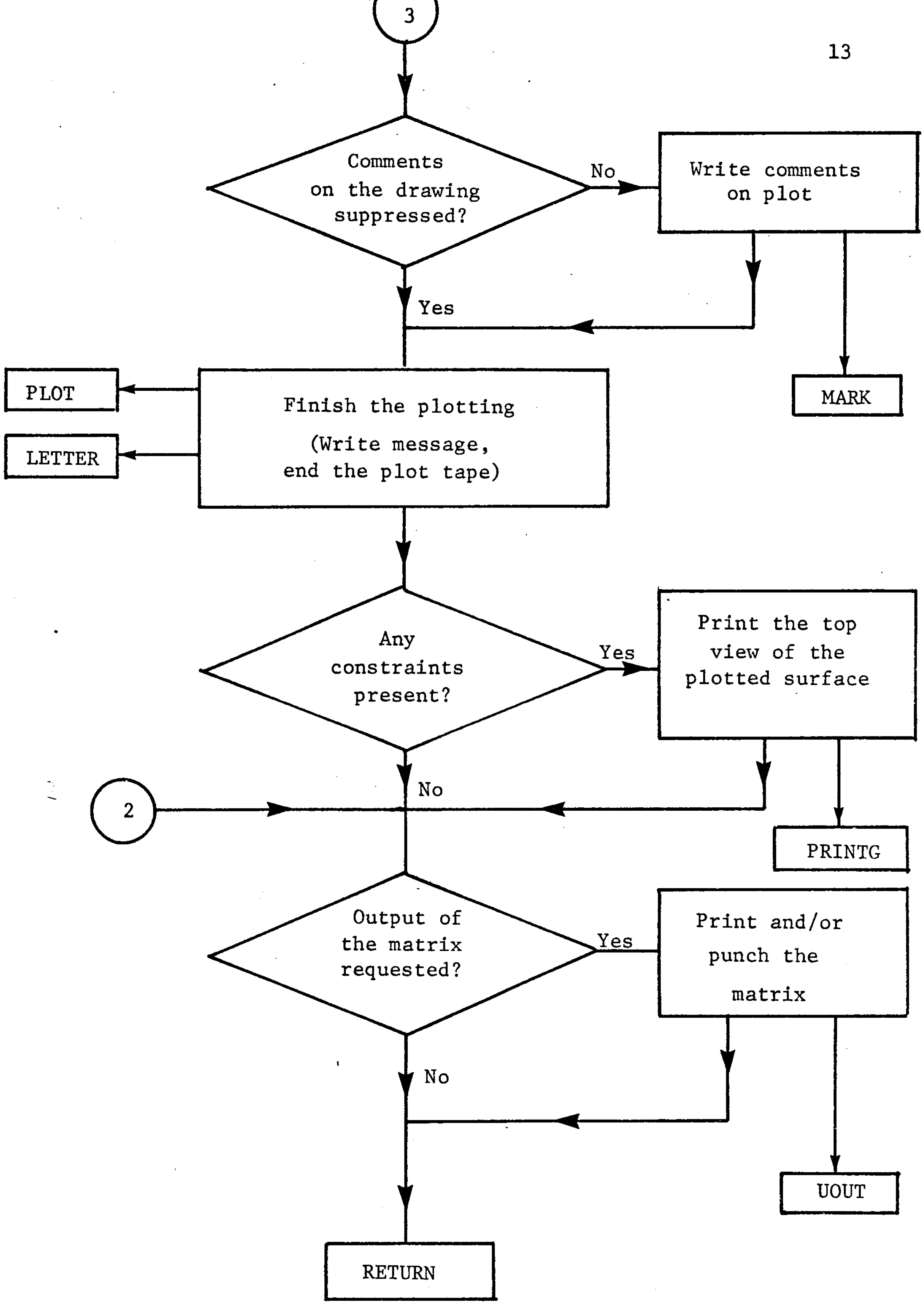

3
Comments on the drawing suppressed?
No
Write comments on plot
Yes
MARK
PLOT
LETTER
Finish the plotting (Write message, end the plot tape)
Any constraints present?
Yes
Print the top view of the plotted surface
No
2
PRINTG
Output of the matrix requested?
Yes
Print and/or punch the matrix
No
UOUT
RETURN

Figure 3.1/2

Fowchart of Subroutine SURF99

CHAPTER 4

OPERATIONS ON THE MATRIX

4.1 Rotation of the Matrix

The matrix can be rotated if a view of the surface is required from a point other than the lower left hand corner as shown in figure 4.1.

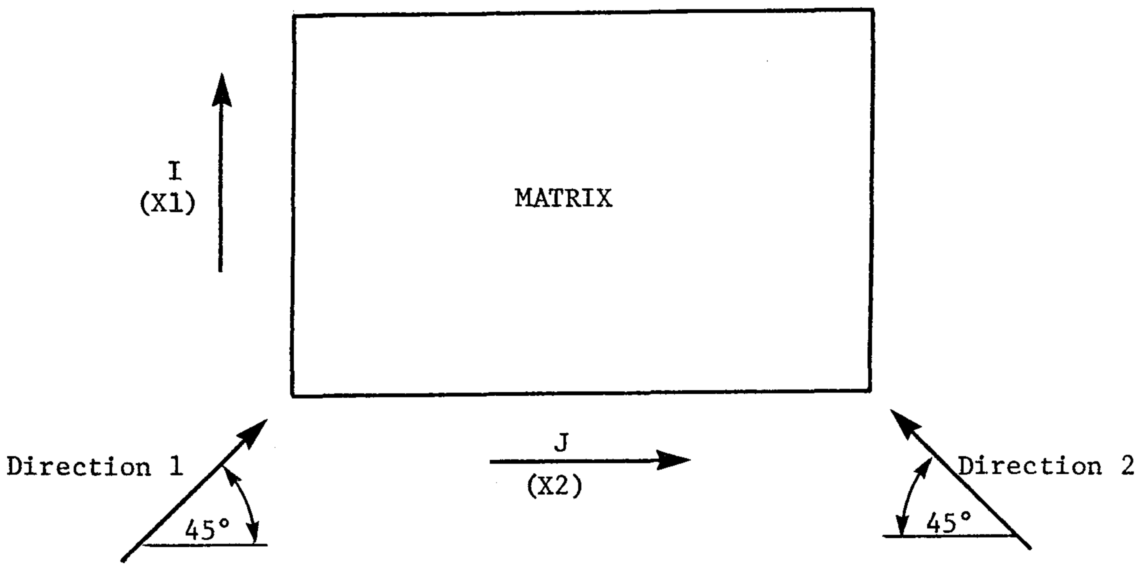


Figure 4.1

Directions of View

Rather than changing the algorithm for mapping the contour, which is set up for view from direction 1, the whole matrix is rotated within the core space allocated to it in the computer memory until the corner required can be seen from direction 1.

The rotating routines are called ROTOR and READU and they rotate the matrix clockwise so that the surface mapped is rotated anticlockwise. The matrix is written row-wise starting from the last row in the file, element by element, from which it is read back into the same array, column-wise. The necessity of doing this in two different routines arises from the need to rotate arrays that do not have equal sides in which case the dimensions of the matrix have to be mutually changed. Figures 4.2 through 4.5 show the rotation and its effect on the drawn surface.

4.2 Maximum and Minimum Values

The maximum and minimum values of the matrix have to be known primarily in order to scale the mapped plot so that it fits into the specified overall vertical size on the paper. As their locations are also recorded for the surface generated analytically, they are marked on the mapped plot if visible. In cases where there are multiple maxima and minima, the largest maximum and smallest minimum will be marked.

4.3 Horizontal Size and x and y Increments

The horizontal size of the mapped plot results from the specified length representing the larger of the two dimensions of the matrix and the angle of inclination of the side of the base as shown in Figure 4.6.

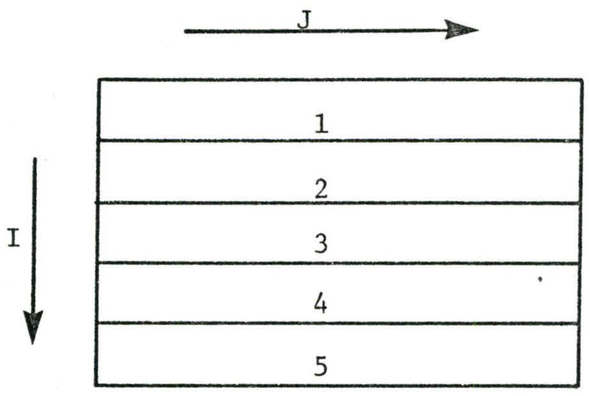


Figure 4.2

Original Matrix as Stored in the Memory

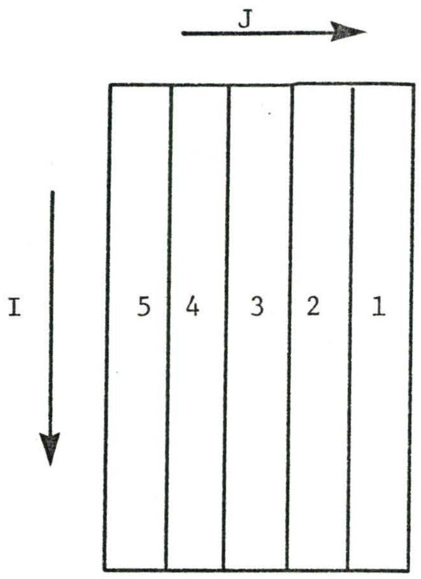


Figure 4.4

Rotated Matrix as Stored in the Memory

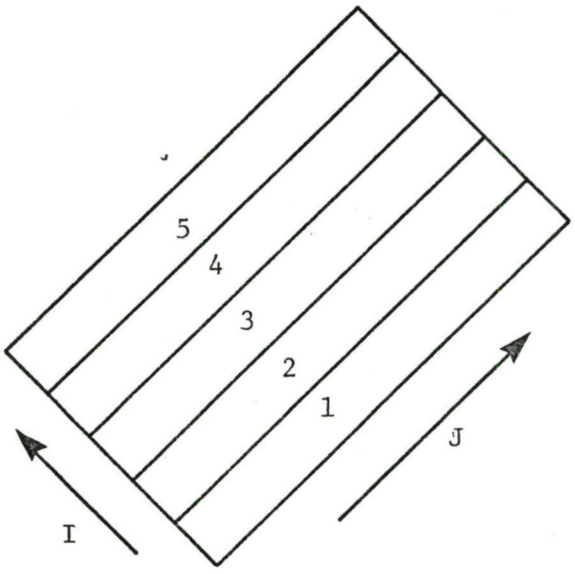


Figure 4.3

Original Matrix as Plotted

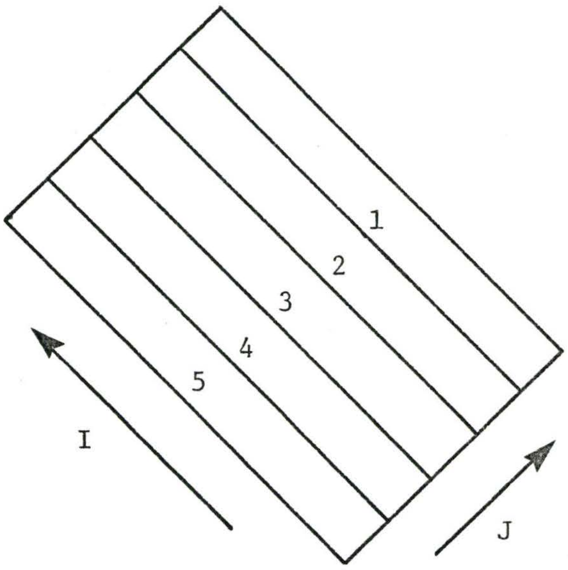


Figure 4.5

Rotated Matrix as Plotted

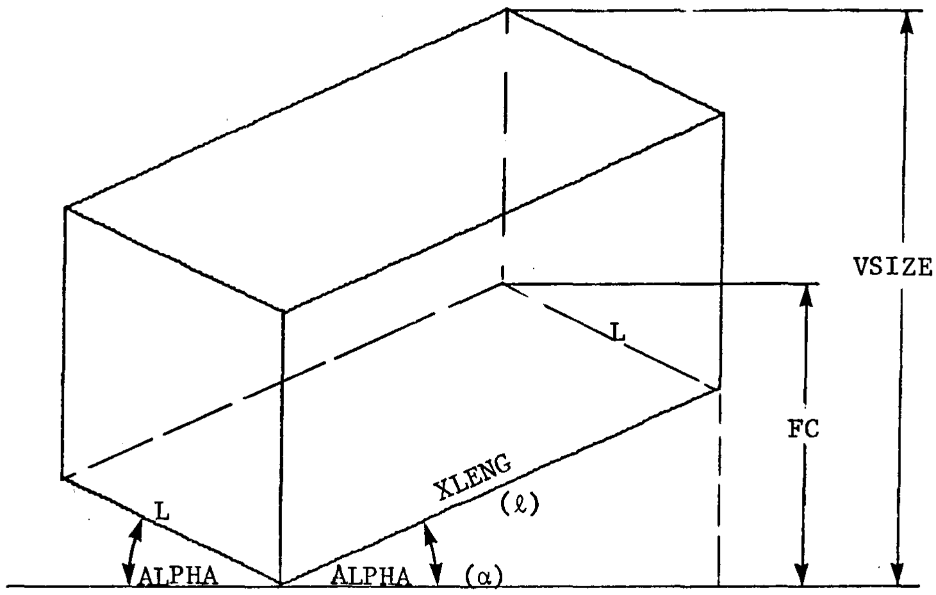


Figure 4.6

Principle Projection

Each element is located at the grid point whose location on the paper has to be known prior to the drawing of the surface. Depending on the position of the matrix which is discussed in Chapter 5, the location of the grid point of the element u_{ij} is calculated as follows:

$$x_{ij} = \Delta x(j-i) \qquad (4.1)$$

$$y_{ij} = \Delta y(j+i-2) \qquad (4.2)$$

where x_{ij} and y_{ij} are co-ordinates of the point with respect to the origin at u_{11}.

$\Delta x = \ell \cos\alpha/m$

$\Delta y = \ell \sin\alpha/n$

where ℓ is the length of the longer side and

α is the projected angle of the side

The overall horizontal size is given by the equation

$$h = \sum_{i=1}^{m} \sum_{i=1}^{n} \Delta x_{ij} \qquad (4.3)$$

4.4 Vertical Scale

The vertical scale is calculated with respect to the overall size of the mapped plot so that this overall size is equal to a pre-specified value. Two points have to be recognized before calculation of the scaling factor.

a) The mesh parallel to the x1-x2 plane is positioned at

$u = 0 \quad$ for $u_{min} \geqslant 0$

$u = u_{min} \quad$ for $u_{min} < 0$

This means that when the minimum value of the matrix is negative, the minimum value is subtracted from each value before the scaling factor is calculated.

b) The grid points of the mesh have their actual locations on the paper given by (4.1) and (4.2).

Thus the scaling factor is given as the minimum of all vs_{ij} as follows:

$$vs_{ij} = (v-y_{ij})/(u_{ij}-\Delta u) \quad (4.4)$$

where:	vs_{ij}	is the scaling factor for element u_{ij}
	v	is the overall vertical size
	y_{ij}	is the vertical co-ordinate of the grid point corresponding to the element u_{ij}
	u_{ij}	is any value of the matrix
	$\Delta u=0$	for $u_{min} \geq 0$
	$\Delta u=u_{min}$	for $u_{min} < 0$

4.5 Conversion of the Array of Values

Before mapping of the surface starts, the original matrix is recalculated so that each element which had a non-flagged value representing its elevation on the surface is scaled and elements that had their value flagged are set to -1.0. Because the origin for drawing of the mapped plot is located at the grid point u_{11}, all defined elements have positive values which distinguishes them from those which are not defined. Conversion of the matrix is then as follows:

$$u_{ij} = -1.0 \quad \text{for flagged elements}$$

$$u_{ij} = (u_{ij}-\Delta u) \cdot vs + y_{ij} \quad (4.5)$$

where u_{ij}, Δu, vs and y_{ij} have the same meaning as in equation (4.4).

CHAPTER 5
DRAWING OF THE SURFACE

5.1 Positioning of the Surface

The surface is mapped so that the observer's direction of view is 45° from a position in front of a corner of the mapped solid corresponding to the element u_{11} as shown in Figure 4.1 (i.e. a dimetric projection).

The variable subscripted by I is represented by the left hand horizontal axis. The J subscripted variable is represented by the right hand horizontal axis.

The surface is completed by the two front visible vertical planes to give an image of a solid body. The sides of the base are kept in the ratio of the dimensions of the sides of the matrix used for storage of the elevation of the points on the surface of the solid.

$$\frac{l_L}{l_R} = \frac{m'-1}{n'-1} \tag{5.1}$$

where	l_L	is the length of the left side of the base
	l_R	is the length of the right side of the base
	m'	is the first dimension of the matrix
	n'	is the second dimension of the matrix

By the above method, the grid points of the mapped plot are ordered in rows so that when projected they lie on vertical lines. This method has been adopted so that rapid search can be made by the algorithm that checks for hidden lines. Also, it limits rotation of the matrix to multiples of 90° which is not a limitation of the program since, besides rotation, the object may be tilted as well and so be seen from all sides. If the angle of rotation were allowed to take on any value, the time required to execute the program would be greatly increased with no benefit in terms of increased utility.

Figure 5.1 shows the projected base network of a 5x7 matrix.

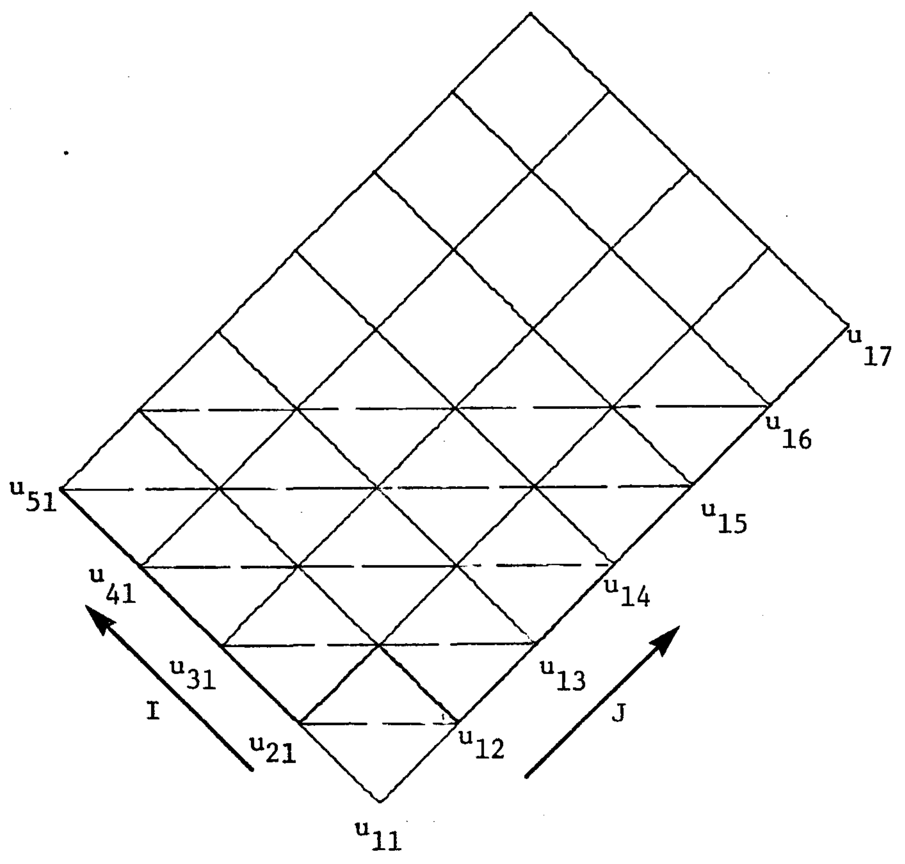


Figure 5.1

View of the Matrix as Plotted

5.2 Choice of a Path for the Pen

The surface is mapped by sending the pen from one element to the next according to an algorithm. There are three basic ways of drawing the lines.

a) The pen travels along the rows of the matrix, i.e. the I subscripted variable runs through a complete cycle before the J subscripted variable is changed (see Figure 5.1).

b) The pen travels along the columns of the matrix, i.e. the J subscripted variable runs through a complete cycle before the I subscript is changed.

c) The pen travels along diagonals, i.e. both subscripts vary simultaneously, the one increasing, the other decreasing as shown on the dashed line of Figure 5.1.

Each of the three algorithms may be used alone, or in combination with the others. Figures 5.2 through 5.4 show examples of some alternatives.

It is obvious that for a thin mesh, or large increments Δx and Δy the image of the mapped solid will not be satisfactory. Therefore, the algorithm built into the program draws the lines both row-wise and column-wise. This is especially necessary for cases where the program has to produce a clear plot of the edges of the solid and restricted areas as apparent from Figures 5.2, 5.3 and 5.4.

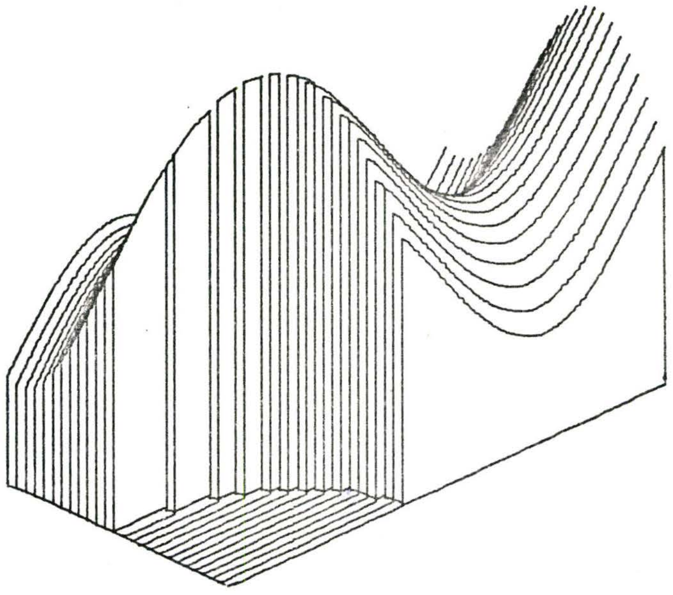


Figure 5.2

Left-to-right Pen Paths

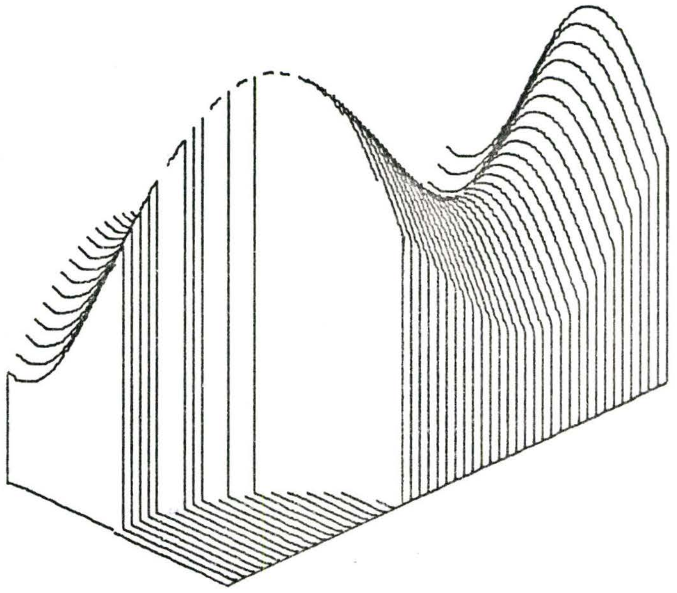


Figure 5.3

Right-to-left Pen Paths

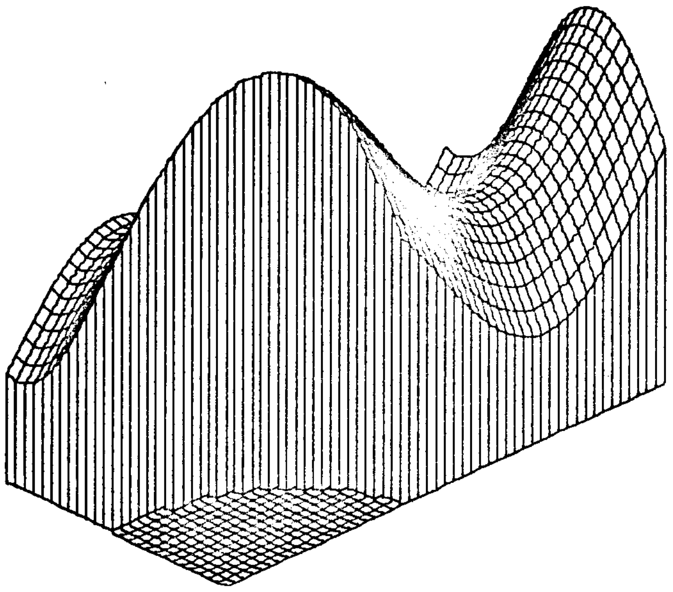


Figure 5.4

Combined Pen Paths

5.3 The Hidden Lines Algorithm

The fact that the hidden lines are not drawn is one of the major advantages of the package.

The conversion of the matrix as described in Chapter 4 and the choice of the position of the surface, in Chapter 5 enable the logic to perform a rapid check.

Before the pen is moved to its next position, the visibility of the point is checked against all points standing in front of the tested one. If the tested value is the largest value in the line of sight, the point is visible, otherwise the point could be hidden. Comparing the visibility of the next point with the point at which the pen has stopped , the line is either drawn fully or not drawn at all for the cases where the points are both visible or hidden respectively.

If only one of the points is visible, a point of intersection with the shading contour is found and the line is drawn up to this horizon (in the case where the next point is hidden) or starts at the horizon when the next point is visible.

The advantage of having the grid points ordered along the line of sight is apparent because they have to be checked for visibility. If the points did not run along the line of sight it would be necessary to interpolate to find a horizon. Such a procedure would involve added CPU time with no apparent advantage.

5.4 Discussion of Difficulties which may Occur

As has been mentioned, it is difficult to interpret a plot with a low density mesh, particularly so when the surface has restricted areas (constraints). This difficulty of interpretation can only be improved by increasing the density of the lines.

Since the surface is assumed not to exist between two flagged values, or between a flagged and a non-flagged value, lines joining such values represent the grid on the base only. This does not cause any difficulties within the restricted area; however, at the boundaries elements have two values, one on the base and the other on the surface. The pen has to be moved to the boundary between restricted and non-restricted areas and at this boundary it has to either be moved from the base to the surface or from the surface to the base depending on whether the boundary was approached from a restricted or a non-restricted area. These two types of moves have to be made in one step while maintaining the hidden line condition. Thus, the hidden line algorithm has to be modified to adopt straight vertical lines that jam the edge of the surface to the base at the intersection of restricted and non-restricted areas.

Figure 5.5 represents another major problem, namely the representation of inner corners. Comparing this figure with Figure 5.6, representing the top view of the surface, it can be seen that the contours of the corners resulting from the situation at the element U(6,6) (Figure 5.6) cannot be drawn by a simple algorithm that joins two consecutive points while handling the boundaries as described above.

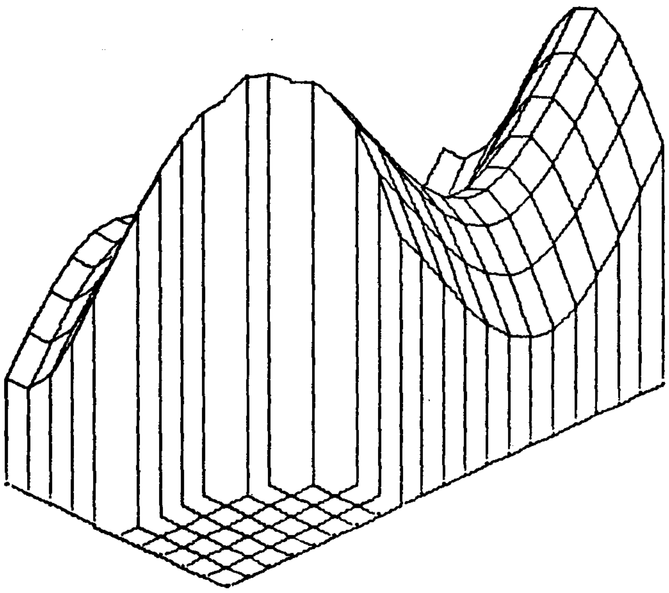


Figure 5.5

Plot Without Subroutine INCO

```
TOP VIEW OF THE PLOTTED SURFACE - MATRIX U(I,J)

        5    10   15   20
        I    I    I    I
11  +++++++++++++++++++++   11
10  +++++++++++++++++++++   10
 9  +++++++++++++++++++++    9
 8  +++++++++++++++++++++    8
 7  +++++++++++++++++++++    7
 6  -++++++++++++++++++++    6
 5  -----++++++++++++++++    5
 4  -------++++++++++++++    4
 3  --------+++++++++++++    3
 2  --------+++++++++++++    2
 1  --------+++++++++++++    1
        I    I    I    I
        5    10   15   20
```

Figure 5.6

Line Printer of Representation of Figure 5.5

In order to complete the plot, namely, to draw the vertical contour of the corner and the horizontal contour of the bottom (base) of the surface, a special routine has to be called before a move to the next point is made.

In contrast to the above, values at the corners that project out - element U(5,6) - are accessible from both the I and the J directions. Thus the vertical line would be drawn twice (Figures 5.5 and 5.6). To prevent this, the value that causes the problem is flagged once, the vertical line is drawn, and when found by drawing the lines in the other direction, mapping of the corner is bypassed.

CHAPTER 6

MAJOR SUBROUTINES EMPLOYED IN MAPPING

The main tasks during the generation of the mapped plot are accomplished by five subroutines, the overall logic of which are discussed in the following paragraphs.

6.1 Subroutine FIZL

This subroutine, once supplied with the identification of the matrix element to which the pen is supposed to move next, checks for various conditions before the move is made and calls for movement of the pen; see the flowchart represented on Figure 6.1.

The routine tests whether the boundary has been reached, and if so, supervises the repositioning of the pen from the surface to the base or vice versa, by calling subroutine ELEV. It also flags elements, when necessary,to prevent the drawing of a duplicate contour as discussed at the end of the previous chapter.

Before moving from one point to another, this subroutine checks visibility by calling a subroutine EYE and finally for the actual move a call to the subroutine MOVE is issued. When the move has been made, control is returned to the calling routine to obtain the location of the element through which the line has to be drawn next.

6.2 Subroutine ELEV

The purpose of this subroutine is to move the pen from the

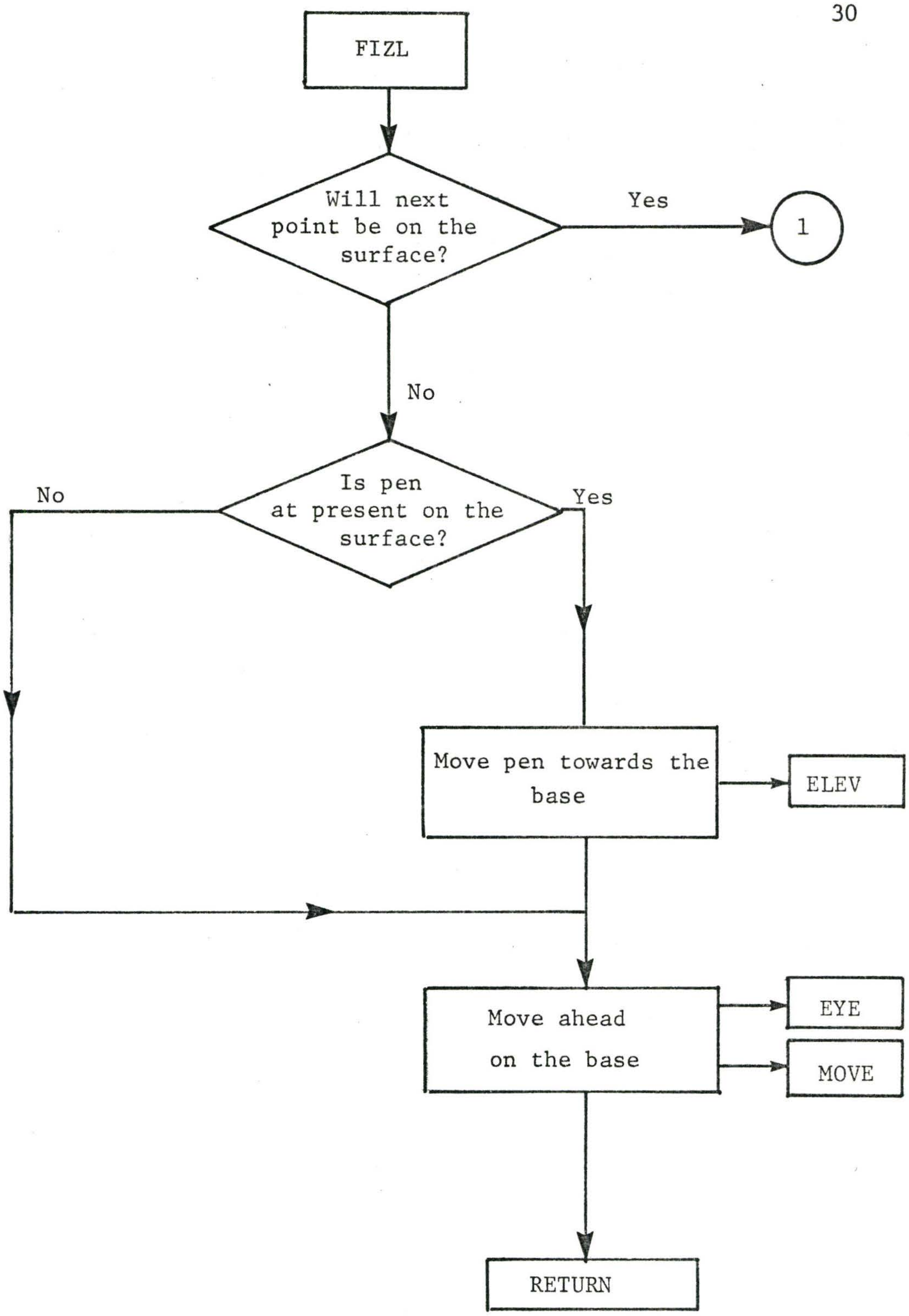

FIZL
Will next point be on the surface?
Yes
1
No
Is pen at present on the surface?
No
Yes
Move pen towards the base
ELEV
Move ahead on the base
EYE
MOVE
RETURN

Figure 6.1

Flowchart of Subroutine FIZL (cont'd)

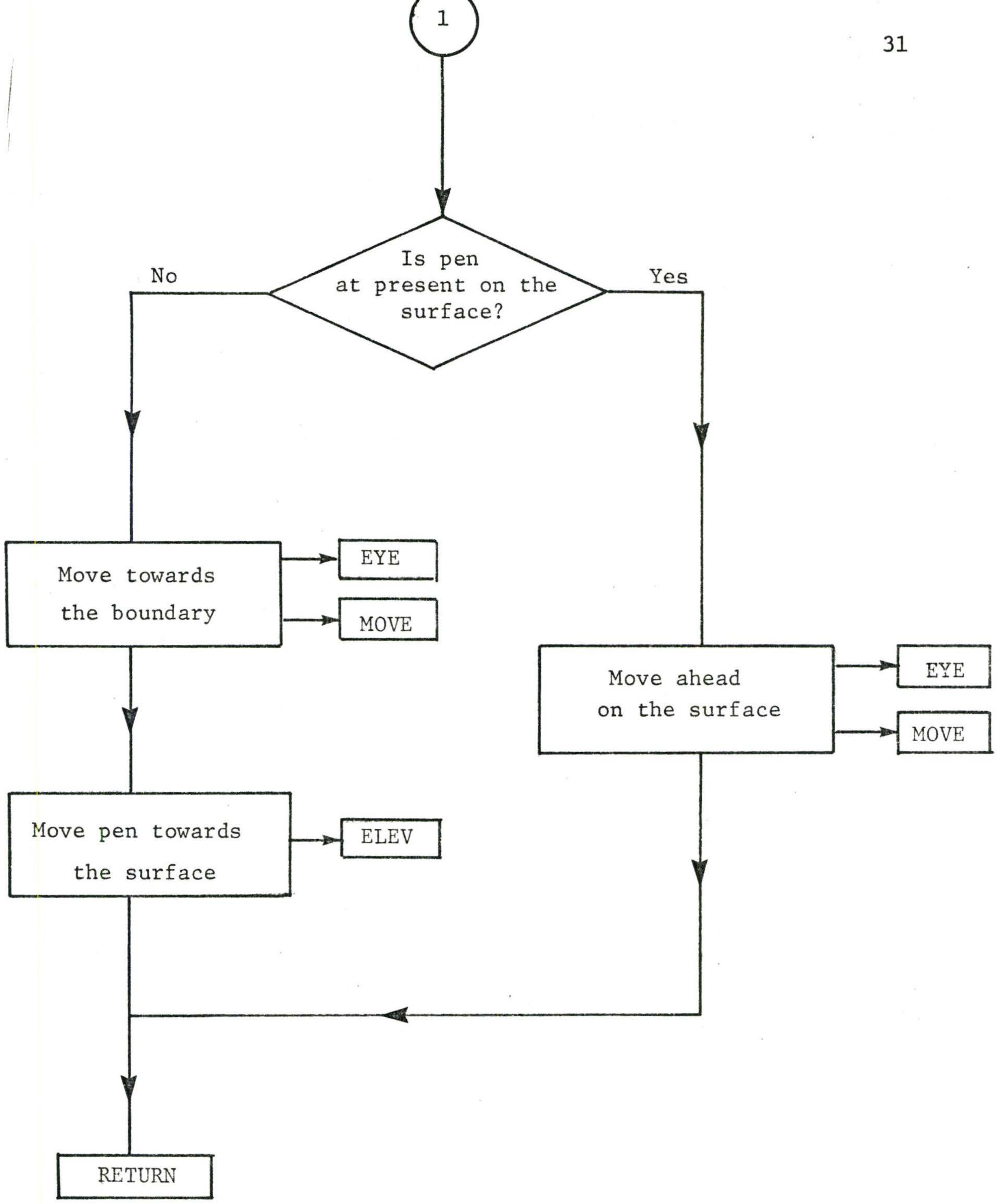

1
Is pen at present on the surface?
No
Yes
Move towards the boundary
EYE
MOVE
Move pen towards the surface
ELEV
Move ahead on the surface
EYE
MOVE
RETURN

Figure 6.1/2

Flowchart of Subroutine FIZL

surface to the base of the solid and vice versa while checking the visibility of the vertical lines. The visibility of the next point is checked by calling subroutine EYE and the move is made by subroutine MOVE. While the basic algorithm for hidden lines is still used, this subroutine has an extension to handle the conditions at the boundaries when the basic algorithm fails. It also takes care of the problem of not drawing double vertical lines in situations similar to the one shown in Figure 6.2.

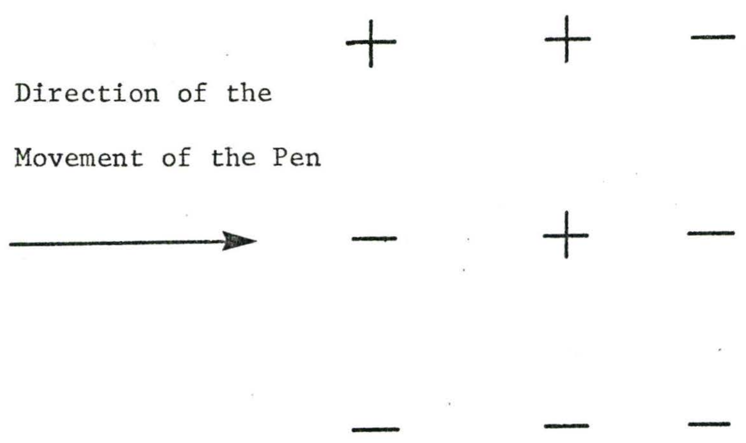


Figure 6.2

Special Configuration of Flagged and Nonflagged Values

6.3 Subroutine EYE

The hidden line algorithm is contained in this subroutine. The routine, once supplied with the location of the tested element and its value, returns the proper mode for the pen to make its next move. If the returned mode is opposite to the supplied one, it finds the point at which the mode changes and moves the pen up to this point; see the flowchart represented on Figure 6.3

The subroutine EYE uses several assisting subroutines, the most important of which is the subroutine CORNER, to determine whether the element causing the change in mode of the pen is an element at the boundary of the surface and a constrained area. When this check is positive, the mode of the pen has to be decided relevant to the particular circumstances and no point of intersection is looked for.

6.4 Subroutine INCO

Subroutine INCO (see the flowchart represented on Figure 6.4) is called immediately after a move of the pen has been made, i.e. after control has been returned from subroutine FIZL, in order to see if the element the pen was moved to is located at an inner corner as explained in the previous chapter. If the check is positive, the contour of the corner is drawn by subroutine ELEV and the base contours, Figure 5.5, are completed using subroutines EYE and MOVE.

This subroutine represents a break in the continuity of logic for drawing a line through a given path. For this reason, all indicators, pointers and the present position of the pen have to be saved and before leaving the routine the saved data has to be restored to its status when the routine was entered.

6.5 Subroutine MOVE

This subroutine controls the movements of the pen while generating the mapped contour in order to avoid inefficient travel of the pen on the paper. The routine issues calls to the library routine PLOT only

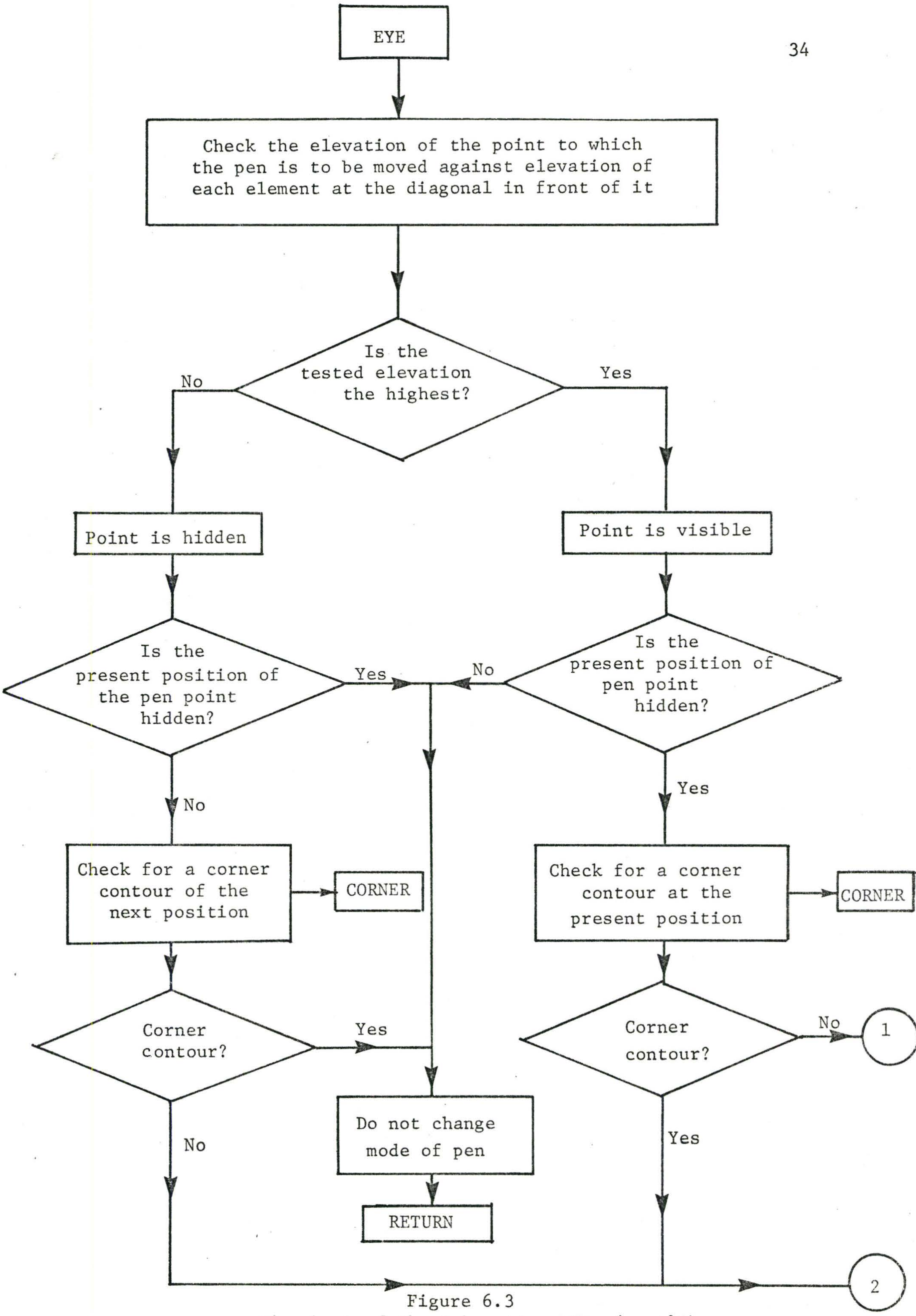

EYE
Check the elevation of the point to which the pen is to be moved against elevation of each element at the diagonal in front of it
Is the tested elevation the highest?
No
Yes
Point is hidden
Point is visible
Is the present position of the pen point hidden?
Yes
No
Is the present position of pen point hidden?
No
Yes
Check for a corner contour of the next position
CORNER
Check for a corner contour at the present position
CORNER
Corner contour?
Yes
No
Corner contour?
No
1
Yes
Do not change mode of pen
RETURN
2

Figure 6.3
Flowchart of the Subroutine EYE (cont'd)

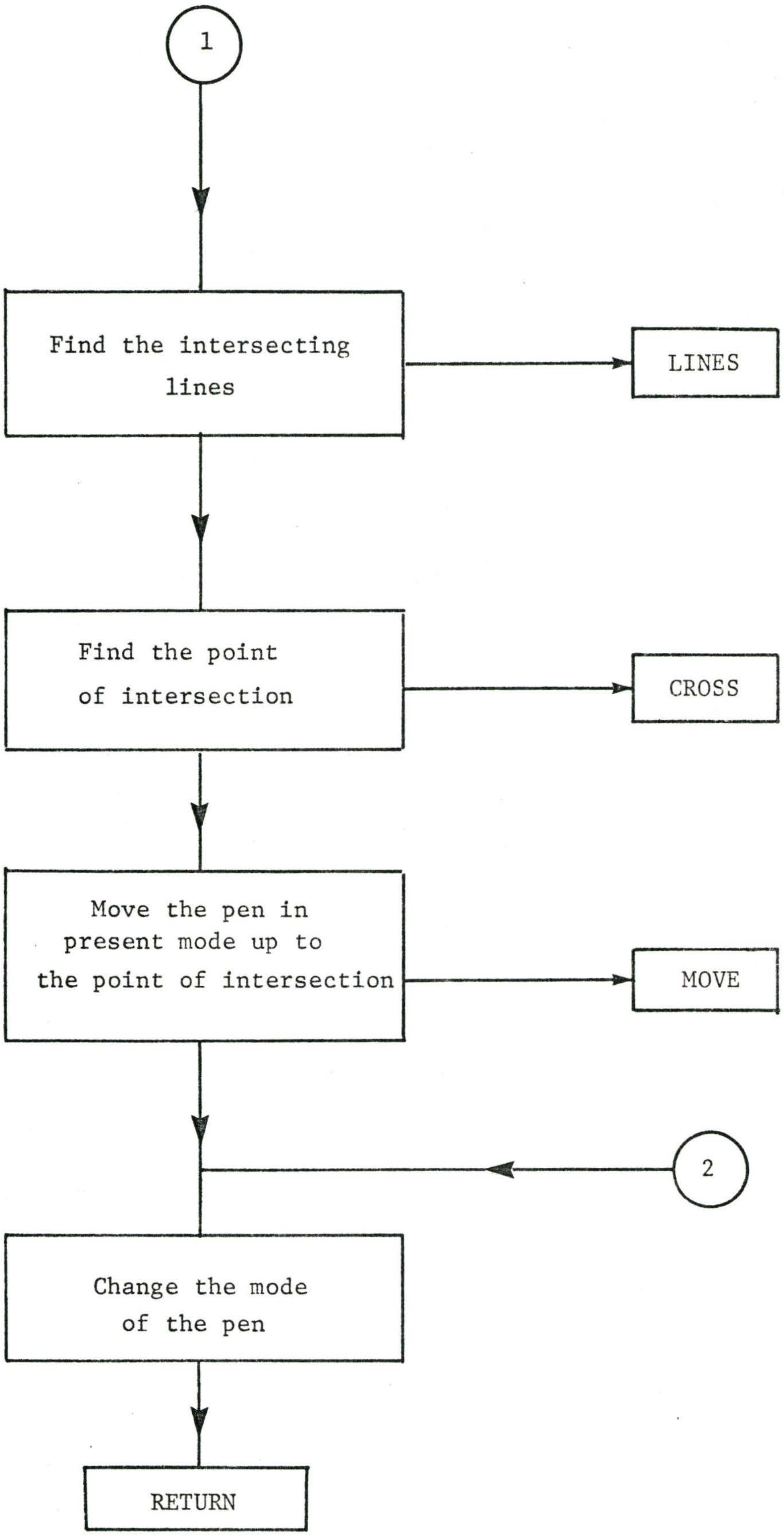

1
Find the intersecting lines
LINES
Find the point of intersection
CROSS
Move the pen in present mode up to the point of intersection
MOVE
2
Change the mode of the pen
RETURN

Figure 6.3/1
Flowchart of Subroutine EYE

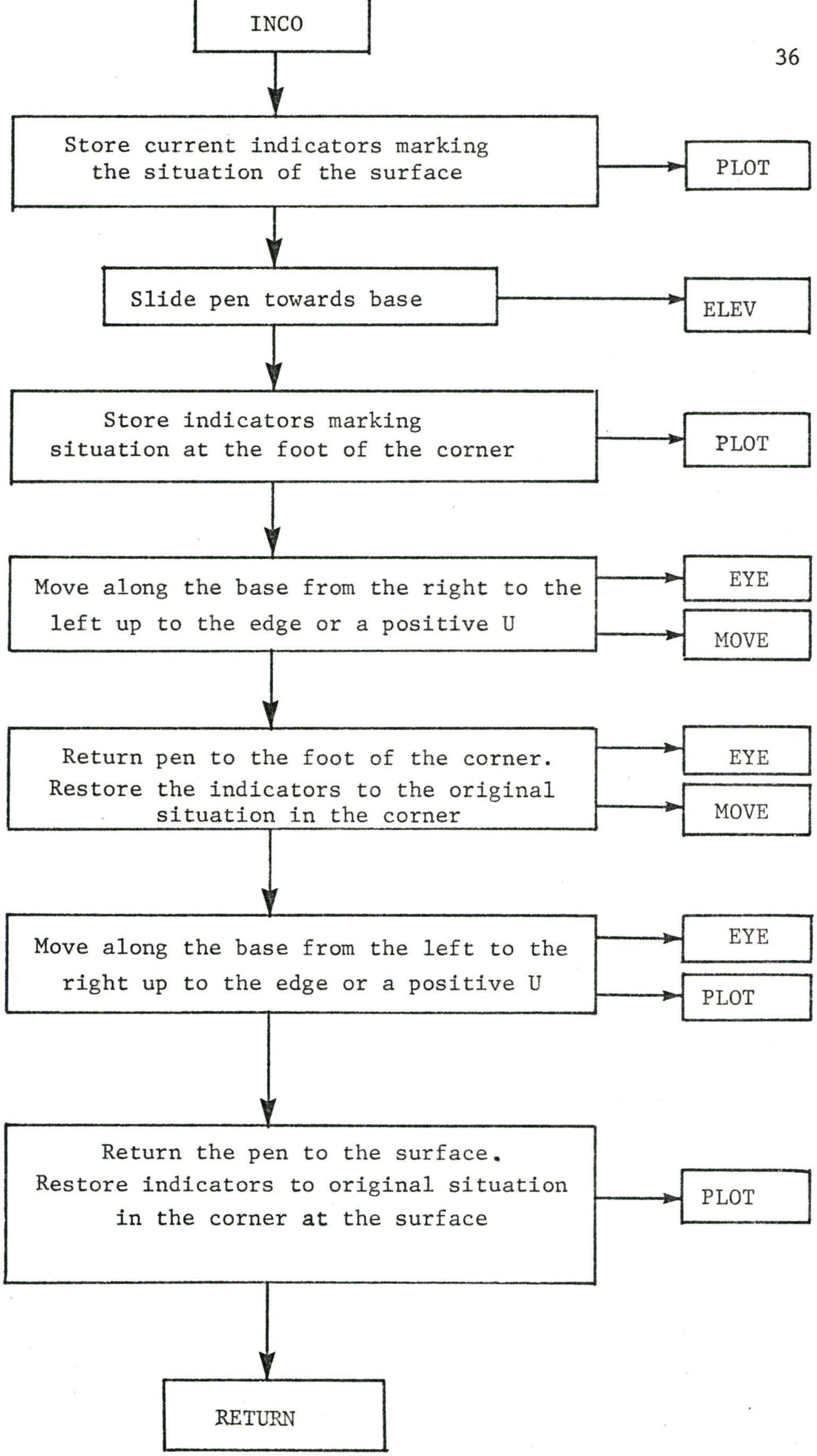

INCO
Store current indicators marking the situation of the surface
PLOT
Slide pen towards base
ELEV
Store indicators marking situation at the foot of the corner
PLOT
Move along the base from the right to the left up to the edge or a positive U
EYE
MOVE
Return pen to the foot of the corner. Restore the indicators to the original situation in the corner
EYE
MOVE
Move along the base from the left to the right up to the edge or a positive U
EYE
PLOT
Return the pen to the surface. Restore indicators to original situation in the corner at the surface
PLOT
RETURN

Figure 6.4

Flowchart of Subroutine INCO

if a line is actually drawn, rather than letting the pen follow the hidden lines as well. It has been found that the introduction of this subroutine reduced the time required for the actual drawing of the surface by 20 to 30% for the majority of surface shapes.

CHAPTER 7

TILTING OF THE MAPPED SOLID

7.1 General

The dimensions of the mapped plot result from three parameters supplied to the routine. They are, respectively:

v	the overall vertical size of the plot
1	the length of the longer side of the base
α	the angle that the sides of the base make with the horizontal, i.e. the angle of projection.

Using these physical dimensions of the plot and bearing in mind what was discussed in the previous two chapters, the angle of view in the vertical plane can be changed.

As the program is primarily supposed to analyze functions or sets of data, the plotted display can be purposely distorted and changed by varying the parameters listed above independently provided they do not interfere with one another or with other parameters or with the limits imposed on them. However, if a surface is to be studied in successive steps, looking at it from different angles of view, or some typical angle of view is required, any angle of view as measured from the horizontal plane may be specified. When an angle of view is specified, the dimensions of the mapped plot result from this particular angle and various assumptions built into the program.

7.2 The Cube Approximation

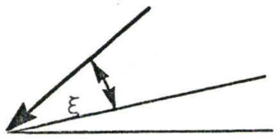
Because the values of the function are incompatible with the values of the independent variables as far as dimensions are concerned, an assumption has to be made regarding their mutual physical relationship.

The relationship between the x1 and x2 axes is given by their limits and the number of intervals through which each ranges. In order to generate plots of the same surface that are in proportion when viewed from different angles (compatible), the assumption has to be made that the maximum height of the surface is located at the corner opposite to the front corner of the base. This allows for the calculation of the scaling factor for conversion of the matrix so that the vertical dimension in the limiting case, corresponding to the limiting angle of view, is equal to the specified value. For all other angles of view, the vertical size is less.

Assuming that the maximum height of the surface when in the corner diagonally opposite the front corner of the base is the worst possible case ensures that the mapped plot does not exceed the maximum specified vertical dimension of the plot on the paper.

From the above it can be assumed that the object being viewed is a box and it simply has to be decided what the principle dimension, namely the height of the box has to be. Figures 7.1 and 7.2 illustrate the principle. It is assumed in the program that the box is a cube, therefore the ratio $\frac{a}{b}$ is simply

$$r = \frac{a}{b} = \frac{1}{\sqrt{2}} \tag{7.1}$$



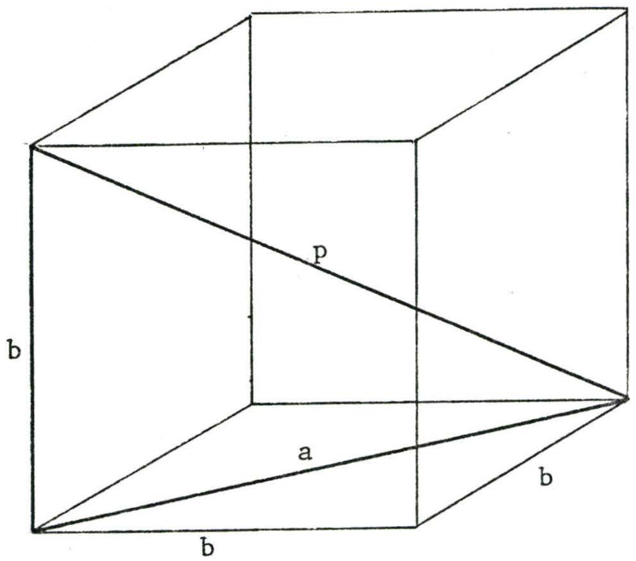


Figure 7.1

Cube Approximation

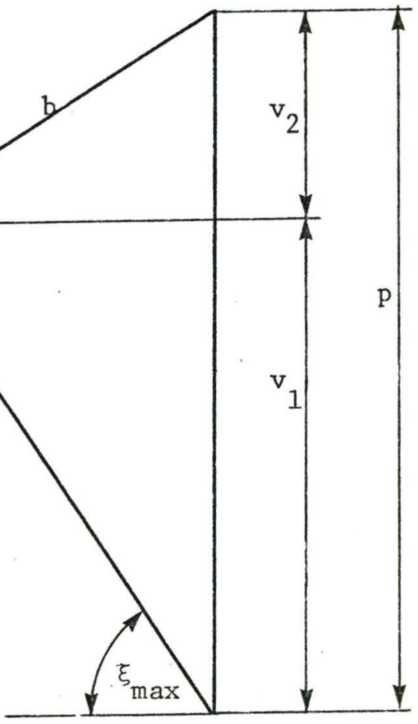


Figure 7.2

Maximum Angle of Tilt

This ratio can, however, be any other value without creating any difficulty in the program other than changing the cube approximation to a prism approximation.

7.3 <u>The Maximum Angle of Tilt</u>

The maximum angle is understood to be that angle of view at which the plot would reach its maximum size if in fact the maximum elevation of the surface was at the corner opposite the front corner of the base. From Figure 7.2 it is apparent that the maximum angle is

$$\xi_{max} = \arctan \left(\frac{a}{b}\right) \tag{7.2}$$

where ξ is the angle of view

7.4 <u>Calculation of the Dimensions</u>

7.4.1 <u>The Angle of the Side of the Base</u>

Figures 7.3 and 7.4 show the relationship between the vertical angle of view ξ and the angle of projection α.

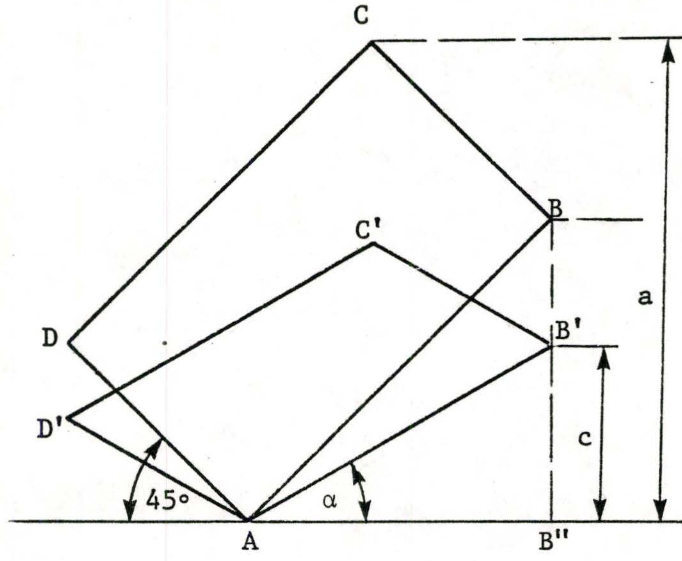


Figure 7.3

Relationship between Projection and Tilt Angles of Base

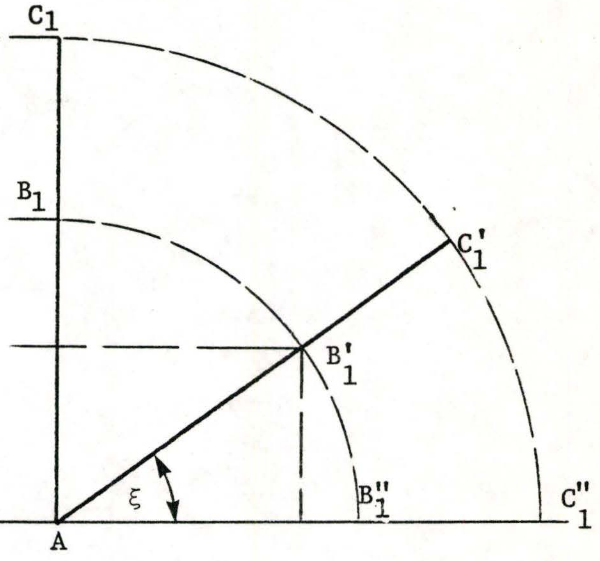


Figure 7.4

End Elevation of Figure 7.3

The angle ξ is simultaneously the angle between the horizontal plane and the angle of the plane of the base, since it may be assumed that rather than the observer change his angle of view, the object is titled towards the observer.

Figure 7.3 shows the projected bases corresponding to an angle of 90°, the rectangle A B C D and an angle ξ, the parallelogram A B'C'D'. Figure 7.4 shows an end elevation of the base. While the base is tilted, the distance A B'' does not change. From Figure 7.3

$$\tan \alpha = \frac{c}{A\,B''}$$

From Figure 7.4

$$\sin \xi = \frac{c}{A\,B'_1}$$

$$\text{Since } A\,B'' = B\,B'' = A\,B'_1$$

$$\tan \alpha = \sin \xi$$

Thus, the required angle α is:

$$\alpha = \arctan(\sin \xi) \qquad (7.3)$$

7.4.2 <u>The Length of the Side of the Base</u>

The length of the side of the base is the distance A B' in Figure 7.3. Since this distance is common to both triangles A B B'' and A B'B'', the following relations may be written for the distance A B''.

$$A\,B'' = A\,B \cos 45°$$

$$A\,B'' = A\,B' \cos \alpha$$

$$\text{thus } A\,B' = A\,B \frac{\cos 45°}{\cos \alpha} \qquad (7.4)$$

The angle α is known and the distance A B can be evaluated from Figure 7.3 and condition (5.1)

From (5.1);

$$A\ B = B\ C\ \frac{n'-1}{m'-1} \tag{7.5}$$

From Figure 7.3;

$$a = (A\ B + C\ B)\ \sin 45° \tag{7.6}$$

From (7.1);

$$a = b.r$$

From Figure 7.2;

$$a^2 + b^2 = p^2 \tag{7.7}$$

Solving equations (7.1) and (7.7) results in the relation

$$a = \left(\frac{p^2}{1+r^2}\right)^{\frac{1}{2}} . r \tag{7.7}$$

Equations (7.5) and (7.6) can now be solved for A B to give the relation;

$$A\ B = \frac{a\sqrt{2}}{\frac{m'-1}{n'-1} + 1} \tag{7.8}$$

The value for A B, substituted in equation (7.4) gives the length of A B'.

7.4.3 The Scaling Factors and the Vertical Size of the Mapped Plot

Assuming that the maximum elevation is as discussed above, the distance v2 (see Figure 7.2) is the projected maximum vertical size of the plot. This is the limiting case and, for a maximum elevation located at any other position, will cause the vertical size of the plot to be less than p. Thus the scaling factor is computed as

$$vs = \frac{v2}{u_{max} - \Delta u} \qquad (7.10)$$

where $v2 = b \cos \xi$

u_{max} is the largest value in the matrix

$\Delta u = 0$ for $u_{min} \geq 0$

$\Delta u = u_{min}$ for $u_{min} < 0$

The overall size of the mapped plot is then;

$$v = v1 + v2 \qquad (7.11)$$

where $v1 = a \sin \xi$

For a maximum value of ξ (discussed above) the overall size is:

$$v = p$$

SECTION C

GUIDE TO THE USE OF THE PACKAGE

CHAPTER 8

GENERAL DESCRIPTION

The program is written in FORTRAN IV and assumes use of a CDC 6400 computer with the RUN compiler and Benson-Lehner plotter and some of the associated plotting routines.

The entire program comprises a set of routines that may be divided into three sections.

a) Package subroutines,

b) Service subroutines,

c) Library subroutines.

Appendix B is a list of all package subroutines and functions with their brief diagnostic. A description of the most important routines is contained in Chapter 6. The package subroutines contain an independent subsection that has been added to the package to complete the incomplete sets of data. These interpolating subroutines are modified routines using the spline interpolation method with links designed to suit the concept of dynamic storage allocation.

Appendix C gives a full cross reference dictionary for the entire program. The service subroutines are described in the following section. Library subroutines are routines associated with the Benson-Lehner plotter; however, they may become service subroutines supplied by a user and calling corresponding routines from a set of routines for a different plotting system.

CHAPTER 9

SERVICE SUBROUTINES

9.1 General

Service subroutines are routines supplied by the user to generate the array for plotting and to apply any constraining conditions on the surface. There are four special purpose subroutines with reserved names and together with the input data they represent an interface with the user. However, these subroutines may call any other additional user's or library subroutines if it is necessary for them to fulfill their purpose.

9.2 The Calling Program

The calling program is the program containing the call to the package. It may be a subroutine but in the majority of cases it will be a small main program containing the call and perhaps, if the user specifies his own plot identification, the blank COMMON/A/ and statements redefining the array KODE(5).

An example of part of the calling routine is as follows:

```
      COMMON/A/ PHI(20),PSI(5),KODE(5)
C
      READ 100,KODE
100   FORMAT (5A4)
C
```

```
          CALL PLOT3D
C
          STOP
          END
```

9.3 Subroutine OBJECT

This subroutine contains the equation of the function that is to be displayed as a surface and is called to evaluate the function at a given point U(x1,x2). Calls are issued for each grid point so that the array to be plotted is systematically filled with values.

An example of the subroutine that has been used to generate Figures 11.1 through 11.20 is,

```
          SUBROUTINE OBJECT (X1,X2,U)
          U=SIN(X1)+SIN(X2)
          RETURN
          END
```

9.4 Subroutine UFILL

Subroutine UFILL serves the same purpose as subroutine OBJECT; i.e. it fills the plotting array with values. UFILL and OBJECT are mutually exclusive and unlike OBJECT, UFILL is called only once to supply the entire matrix. This subroutine enables the package to analyze a set of data that, in the general case, results from a previous calculation. However, these may be a set of values that do not lie in a smooth surface; for example, statistical data or experimental results. Figure 11.8 shows one such example. An example of the subroutine is,

```
SUBROUTINE UFILL(U,M,N)
DIMENSION U(M,N)
READ(1) U
RETURN
END
```

9.5 Subroutine CONST

The area restrictions that may be formulated analytically are passed to the package via this subroutine. While analogous to the inequality constraints in optimization problems and actual inequality constraints if the program analyzes an optimization surface, these constraints are defined as;

$$\phi_i = \phi_i(x_1,x_2,\dots .x_n)\geq 0 \text{ for } i=1,p$$

where n is the number of independent variables

p is the number of constraints

For the three dimensional display, the number of independent variables is limited to two. The total number of constraints is unlimited in the general case, but the program only allows up to 20 constraints in its present form. Thus, for three dimensional problems the constraints have the form;

$$\phi_i = \phi_i(x_1,x_2)\geq 0 \text{ for } i=1,20$$

By definition, the surface is defined in areas where all constraints are satisfied and not defined in areas where at least one constraint is violated.

An example of the subroutine that has been used to impose restrictions in Figure 11.4 is;

```
SUBROUTINE CONST(X1,X2)
COMMON/A/ PHI(20),PSI(5),KODE(5)
PHI(1)=X1**2+X2**2-2.5**2
RETURN
END
```

9.6 Subroutine EQUAL

This routine is used for marking of lines in intersection of the surface with a vertical plane in the general case. In the specific cases, where an optimization problem is being solved, these lines of intersection represent equality constraints defined as;

$$\psi_j = \psi_j(x_1,x_2,\ldots.x_n)=0 \text{ for } j=1,q$$

where n is the number of variables

q is the number of constraints

The program allows up to 5 equality constraints and for two independent variables, the equality constraints have the form;

$$\psi_j = \psi_j(x_1,x_2)=0 \text{ for } j=1,5$$

An example of the subroutine as used in generating Figure 11.5 is;

```
SUBROUTINE EQUAL (X1,X2)

COMMON / A / PHI(20),PSI(5),KODE(5)

PSI(1)=X1-0.25*X2-0.33

RETURN

END
```

CHAPTER 10

INPUT DATA

10.1 General

The necessary data required for execution of the program is supplied via logical unit no. 5 (card reader most commonly) and is read in two parts:

a) Parameters and switches,

b) Comment (or heading) card.

The parameters supply actual numerical values used during computation. Switches route the execution of the program depending on their state "on" or "off" (refer to chapters on input parameters and switches).

The comment card is a card whose punched characters represent any comment the user may wish to use. This card is simply copied on both the mapped plot and the printed report on plotting (see chapter on Output). This card may be omitted altogether or left blank if no comments are required.

10.2 Subroutine INPUT

The general form of the input data is as follows:

A parameter name followed by an equality sign, followed by a numerical value, followed by a comma, or a similar sequence with a switch name followed by a comma.

For example:

```
NCONS=1,X1MAX=3E0,PRINT,X2MAX=8.2E02,ENDDA
```

The data is decoded in the subroutine INPUT and the values are passed on to the other routines via labelled common. The data on the input cards is not order dependent and all blanks are ignored. The names on the cards are compared character by character with a table of specified 5 letter words and as soon as the first 5 letters are recognized, the rest of the name on the card is ignored up to the equality sign. Illegal names cause termination of the program with an error message.

Integer type parameters must be followed by values in I format such as 3,1,2. Real parameters must be followed by values in E format such as 3.E0, 2.E-1, -6.5E+01, etc.

Scanning of data cards is terminated by the key word ENDDA. There can be as many data cards as necessary provided that no parameter or switch is split between two cards. If a parameter is specified more than once, the last value read is recorded.

The following errors cause termination of the program:

a) Illegal parameter or switch names,

b) Illegal characters,

c) Integer type parameters not followed by an I format number,

d) Real type parameters not followed by an E format number,

e) Syntax errors in I or E format numbers,

f) A name split between two cards.

Any of the above errors are indicated by an error message together with a copy of the card with the error and a pointer showing the place where the error occurred. However, since the pointer is printed by a subroutine called DIFFER it could be missing if the package contains a modified version of this subroutine (refer to section on DIFFER).

An example of a typical error message is shown in Figure 10.1:

```
*****SYNTAX ERROR IN INPUT DATA IN OR AFTER COLUMN 22
  ****X2MAX=10E0,X1MAX=4E0,WSIZE=7.5E0,ENDDATA
                           ↑
```

Figure 10.1

Sample of Error Message

The manner in which input data is supplied to the package is very convenient because the user is not required to punch a given parameter in a given column or field and he does not have to punch them all. At the same time, the user has convenient visual control over the parameters being specified or changed. Punching data in a given column or field very often results in input errors. As most of the parameters and switches are present before execution of the program, the user does not have to specify all of them. This makes the input of data simple and safe.

CHAPTER 11

DESCRIPTION OF INPUT PARAMETERS

11.1 General

The input parameters are required to scale the results plotted by the package. The program uses up to 17 parameters for this control.

Some of the parameters are complementary and not all of them are used or applicable in a given problem. Furthermore, a value can be given to some of the parameters which causes other parameters to be ignored when the program is executed. For example, (refer to table TAB1) ALPHA, VSIZE and XLENG are ignored if an angle of observers view, ANGLE is specified. Values not given to some parameters are substituted for depending on the mode of the program. For example, limits XIMIN, XIMAX, X2MIN and X2MAX are substituted for when not specified and in addition KLASS 1 and no constraints are specified. In the first case, ignored parameters are computed by the program and, in the second case, the limits that are not specified are taken to be the dimensions of the matrix, that is X1MIN=X2MIN=1, X1MAX=ISIZE and X2MAX=JSIZE.

The parameters are divided into two groups:

a) Compulsory parameters,

b) Optional parameters.

	PARAMETERS	DEFAULT VALUES	EXAMPLE OF THE USE	
1	INTX1	50	INTX1	= 30
2	INTX2	50	INTX2	= 60
3	ISIZE	51	ISIZE	= 31
4	JSIZE	51	JSIZE	= 61
5	NCONS	0	NCONSTRAINTS	= 2
6	NEQUA	0	NEQUALITY	= 1
7	KLASS	1	KLASS	= 2
8	ALPHA	20.0	ALPHA	= 15E0
9	ANGLE	N/A	ANGLE	= 6.5E1
10	PAPER	10.0	PAPER	= 28E0
11	VSIZE	10.0	VSIZE	= 1.5E01
12	XLENG	7.5	XLENGTH	= 2E1
13	XYROT	0.0	XYROTATION	= 180.E0
14	X1MAX	0.0	X1MAX	= 0.1E0
15	X1MIN	0.0	X1MIN	= 0.01E0
16	X2MAX	0.0	X2MAX	= 2E-1
17	X2MIN	0.0	X2MIN	= 4E-02

TABLE 1

LIST OF PARAMETERS

The distribution of parameters belonging to each group depends on the method used for generating the matrix to be plotted which is governed by the value of the parameter KLASS.

The influence of the input parameters is illustrated in Figures 11.1 to 11.20. Each of these figures is the matrix plotted on Figure 11.1 with the appropriate parameter or group of parameters changed. This figure has been generated using subroutine OBJECT from paragraph 9.3 and the following parameters:

INTX1	=	10
INTX2	=	20
ISIZE	=	11
JSIZE	=	21
NCONS	=	0
NEQUA	=	0
KLASS	=	1
ALPHA	=	25.0
ANGLE	=	N/A
PAPER	=	10.0
VSIZE	=	3.0
XLENG	=	2.5
XYROT	=	0.0
X1MAX	=	5.0
X1MIN	=	0.0
X2MAX	=	6.5
X2MIN	=	0.0

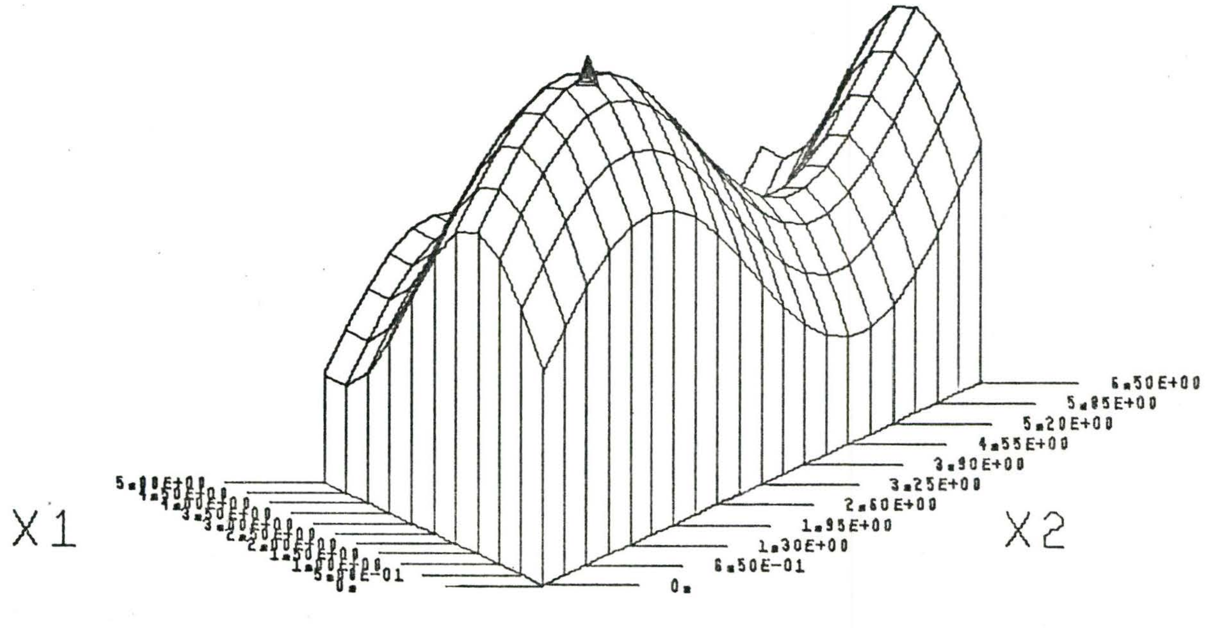


Figure 11.1

Basic Plot of 3-D Solid for Parameters listed on page 58

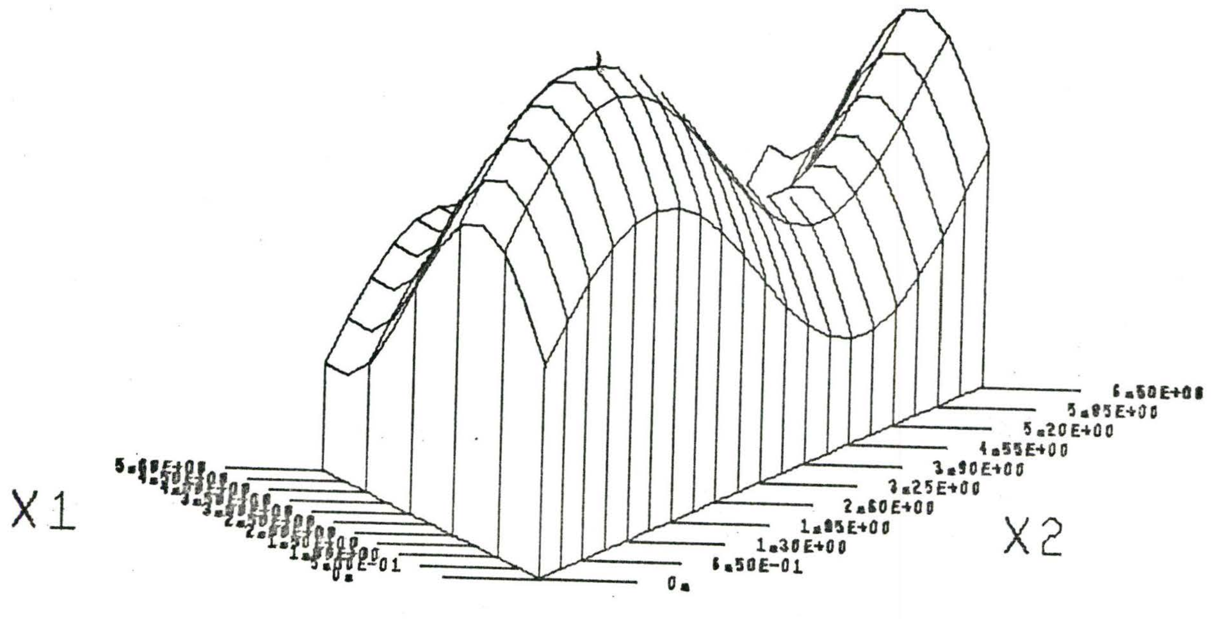


Figure 11.2

INTX1 = 5

11.2 INTX1,INTX2

Parameter INTX1 specifies the number of intervals to be plotted along the X1 axis. The number of lines drawn in the vertical plane is then INTX1+1 and the maximum value of INTX1 is ISIZE-1 where ISIZE is the appropriate side of the matrix to be plotted. INTX1 has a minimum value of 1. In order to draw the boundaries of the matrix, INTX1 has to be chosen so that it is a multiple of ISIZE-1. If both of the above conditions for INTX1 are not satisfied, the program is terminated with an appropriate message in the printed plotting report and dayfile.

The number of intervals to be plotted along the X2 axis is INTX2. The conditions applicable to INTX2 are the same as for INTX1 except for the size of the matrix which is JSIZE. Effect of a change in these parameters is shown Figures 11.2 and 11.3.

11.3 ISIZE, JSIZE

ISIZE and JSIZE are dimensions of the matrix to be plotted, i.e. U(ISIZE,JSIZE). The dimensions of the sides of the matrix can be any value greater than 1, provided that there is sufficient memory available to store the full matrix. The methods used to improve the use of core space are discussed in another section. If ISIZE or JSIZE are less than or equal to 1, or the matrix U does not fit into the core memory, the program is terminated with an appropriate message in the printed plotting report and dayfile.

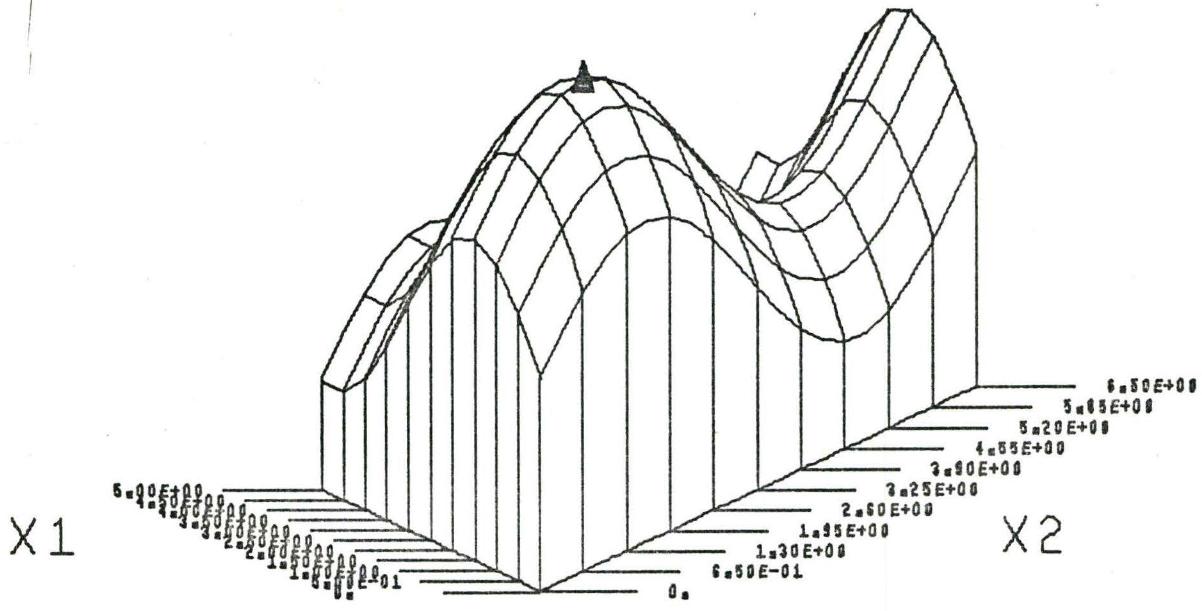


Figure 11.3

INTX2 = 10

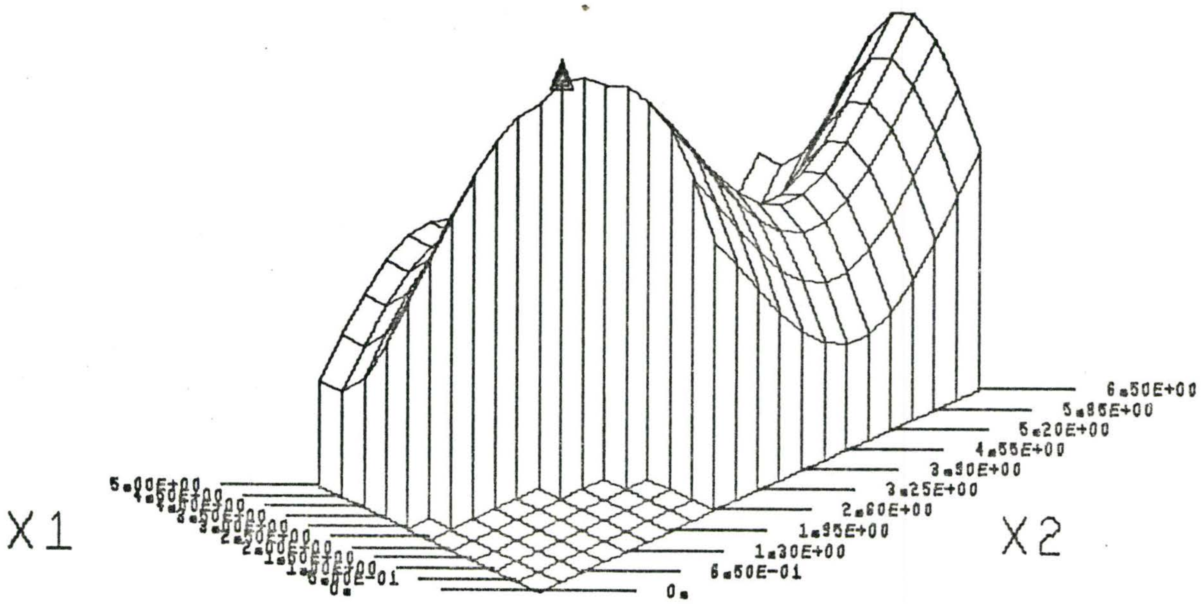


Figure 11.4

NCONS = 1, NEQUA = 0

11.4 NCONS

NCONS is the number of inequality constraints inserted when the package is used for the solution of optimization problems. Because an inequality constraint specifies an area in which the matrix is not defined, the use of this parameter is generally not restricted to the solution of optimization problems. In this case the inequality constraints are used merely to constrain an area in the matrix and not as in optimization procedure to determine maxima and minima. The allowable number of inequality constraints is 20 and if NCONS is out of limits, the program is terminated with appropriate messages.

If $0<\text{NCONS}\leq 20$, the program refers to a user's subroutine CONST (discussed in 9.5) and calls are issued to this subroutine whenever necessary. Figure 11.4 shows a plot with a constrained area in the matrix. In this case NCONS=1 and the user's subroutine CONST is that shown in paragraph 9.5.

11.5 NEQUA

NEQUA is the number of equality constraints and is a parameter primarily used in the solution of optimization problems. This parameter can be used in the same way as NCONS in general plotting routines, its function being to specify a number of paths to be marked on the surface of the solid mapped. The allowed number of equality constraints is 5 and the program terminates if NEQUA is specified outside this limit.

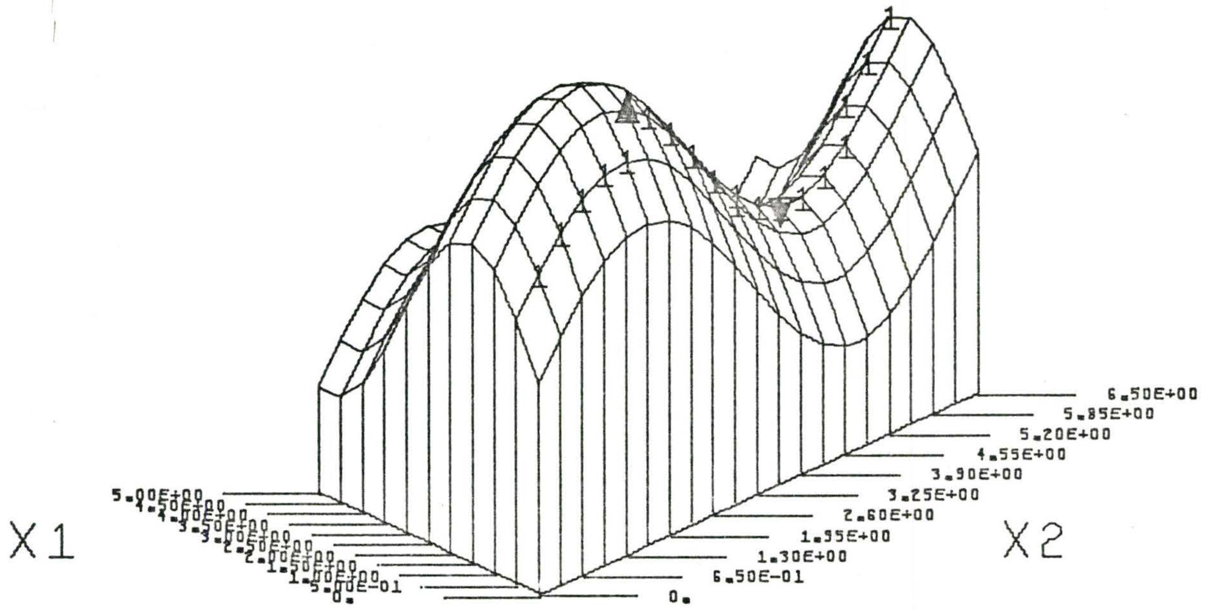


Figure 11.5

NCONS = 0, NEQUA = 1

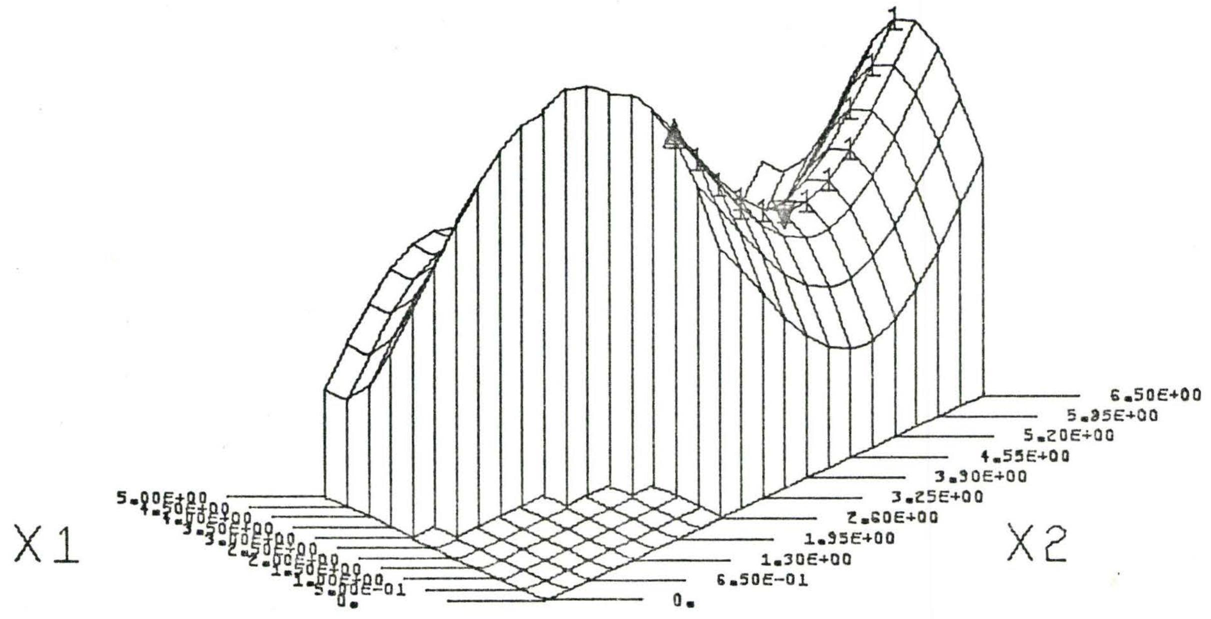


Figure 11.6

NCONS = 1, NEQUA = 1

If $0<\text{NEQUA}\leq 5$ the program refers to a user's subroutine EQUAL (discussed in paragraph 9.6) and calls are issued to this subroutine whenever necessary. Figure 11.5 shows the plot for NEQUA=1 and subroutine EQUAL from the paragraph 9.6. The combined effect of both subroutines is shown in Figure 11.6.

11.6 KLASS

KLASS is the parameter specifying two basic modes of generating the matrix to be plotted.

If KLASS=1 the matrix is generated by means of a user's subroutine called OBJECT (paragraph 9.3) which specifies the surface analytically within the interval $\langle$X1MIN,X1MAX$\rangle$ and $\langle$X2MIN,X2MAX$\rangle$. The subroutine OBJECT, when specified by the user is called periodically by the program to generate the value of the surface at each point U(I,J).

If KLASS=2 or KLASS=3, the matrix values are obtained externally by means of a user supplied subroutine called UFILL. This subroutine is called only once to supply either the entire matrix (for KLASS=2) or all the known values of the matrix (for KLASS=3).

As all the elements of the matrix have to be defined before plotting is possible, interpolation routines, internal to the program are called to complete the matrix when an incomplete matrix is specified, i.e. KLASS=3. If the number of elements specified for the matrix in the external routine are insufficient for successful

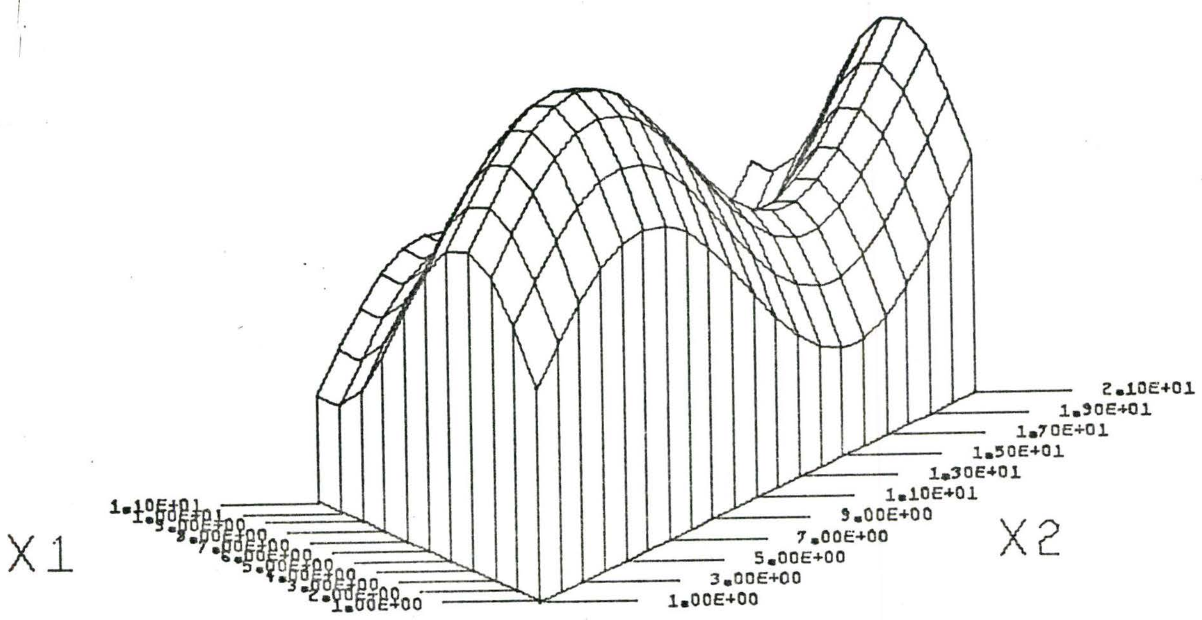


Figure 11.8

KLASS = 2

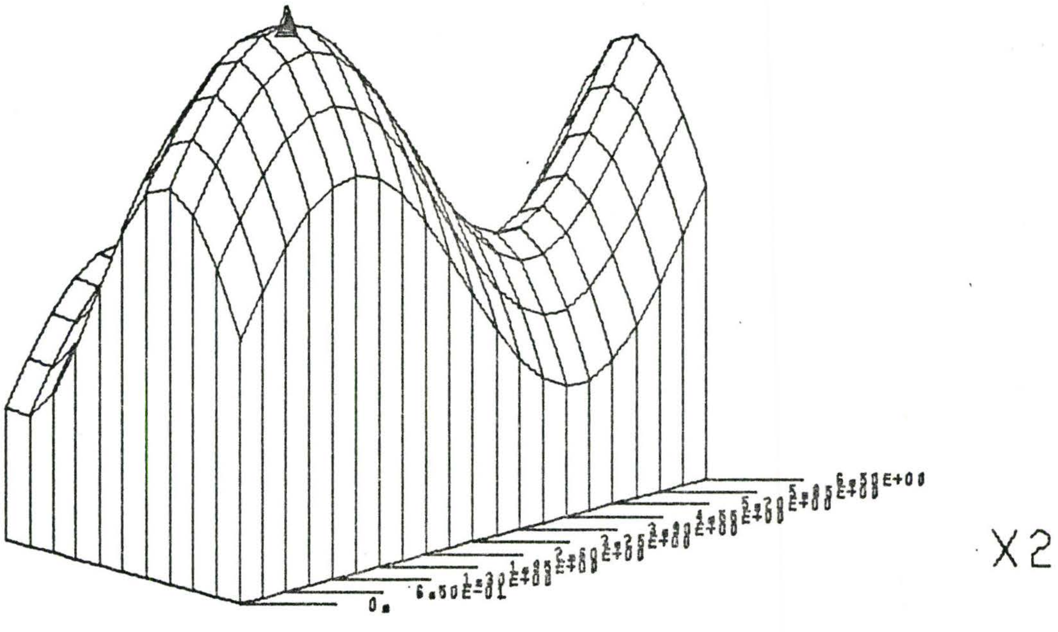


Figure 11.9

ALPHA = 15.0

completion of the data by interpolation, the program is terminated with error messages.

The most common way of obtaining the matrix for KLASS=2 and KLASS=3 is by reading it in. The user has the option of specifying a logical file unit on PROGRAM card which contain the data. The logical unit number has to be different from 2, 5, 6, 8 and 10 because these units are reserved for the program (discussed in Chapter 15).

If values of KLASS are used that differ from 1, 2 or 3 the program is terminated with error messages. Figure 11.8 shows the surface for the case when KLASS=2. As it can be seen, the limits, since not specified, have been substituted for by the dimensions of the matrix.

11.7 ALPHA

ALPHA is the angle that the base axes of the plot make with the horizontal when drawn by the plotter (see Figure 4.6). ALPHA can have any value between 0° and 90°. Although ALPHA is essential for the plotting routine, it is ignored as an input parameter if a value for the parameter ANGLE is specified at the same time, since ALPHA is calculated with respect to the parameter ANGLE. If the value specified for ALPHA is greater or equal to 90.0 degrees or less than 0.0 degrees, the program is terminated with error messages. Figures 11.9 and 11.10 show the effect of changes in angle ALPHA.

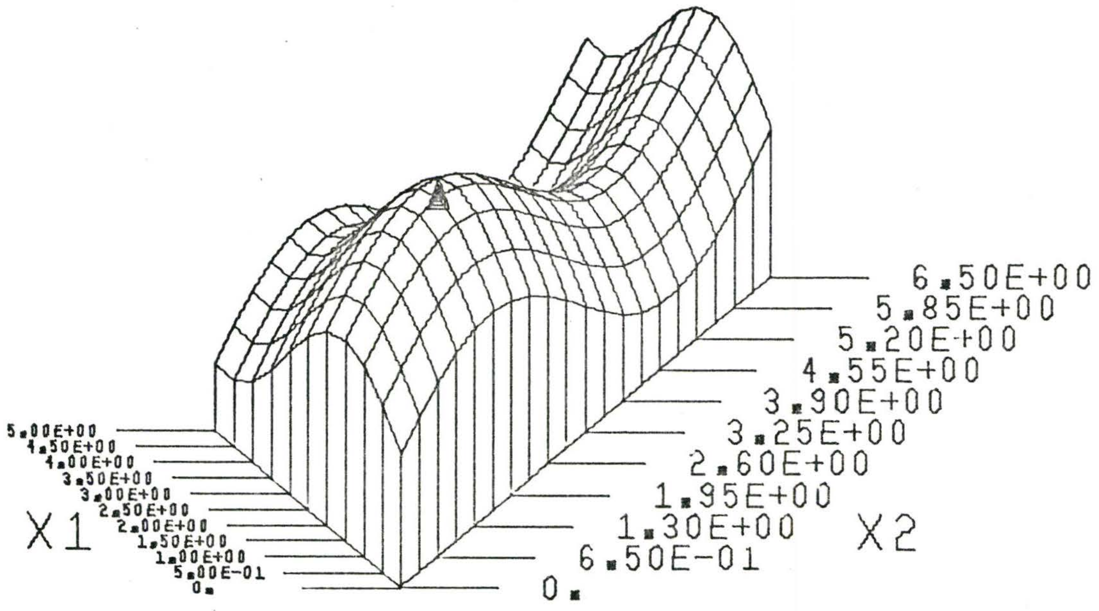


Figure 11.10

ALPHA = 40.0

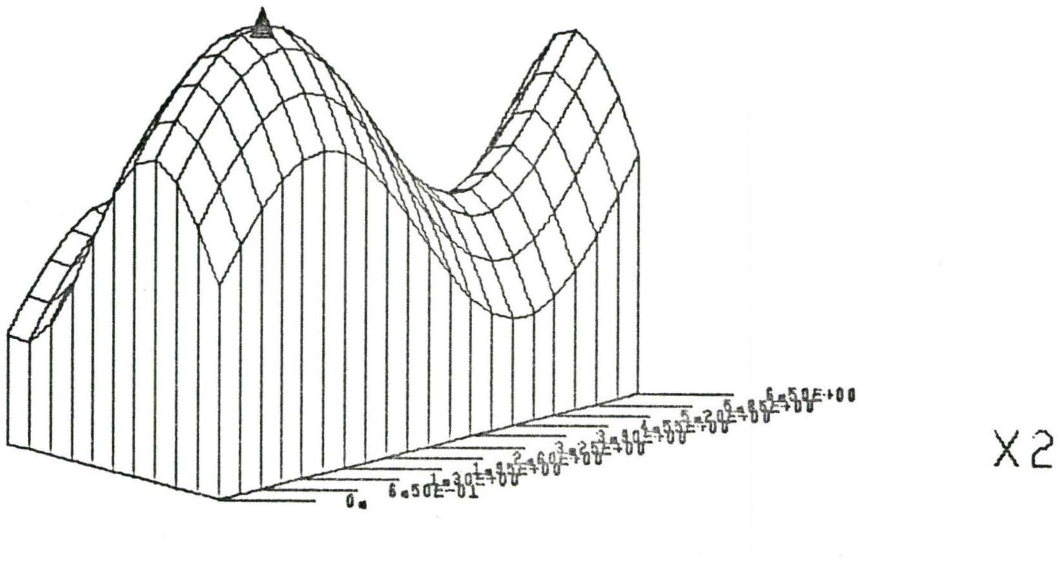


Figure 11.11

ANGLE = 15.0

11.8 ANGLE

This parameter represents the angle of view of the observer measured in vertical plane when looking at the solid plotted by the routine. The value of ANGLE is specified in degrees and may vary from 0 to 90 degrees including both limits, all the intermediate values being allowed. Variation of the parameter ANGLE results in tilting the solid plotted by the routine on a horizontal axis through the intersection of the X1 and X2 axes. When ANGLE has a specified value ALPHA, the parameter discussed above, XLENG, the length of the longer axis in the horizontal plane and, VSIZE, the overall height of the mappled plot, are calculated as explained in Chapter 7, and these parameters are ignored if specified together with ANGLE. If ANGLE is specified outside the allowed limits the program is terminated with error messages. Figures 11.11, 11.12 and 11.13 show views of the solid plotted as ANGLE is increased from 15° through 25° to 60°. When ANGLE is specified as 90° the mapped plot is a plan view of the matrix.

11.9 PAPER

Parameter PAPER specifies the assumed effective width of the paper in the plotter. The standard sizes of the plotting paper are 12 inches and 30 inches from edge to edge. The useful plotting width is one inch less than the nominal size and because it is difficult to use the whole plotting width of the paper (from a practical standpoint) a further inch should be subtracted from the useful width; this allows space for headings, etc. above the plot. From the above

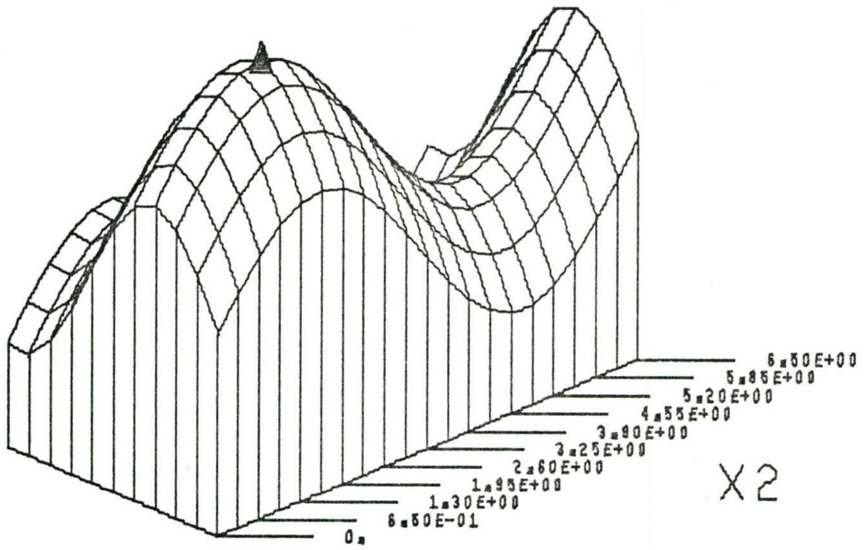


Figure 11.12

ANGLE = 25.0

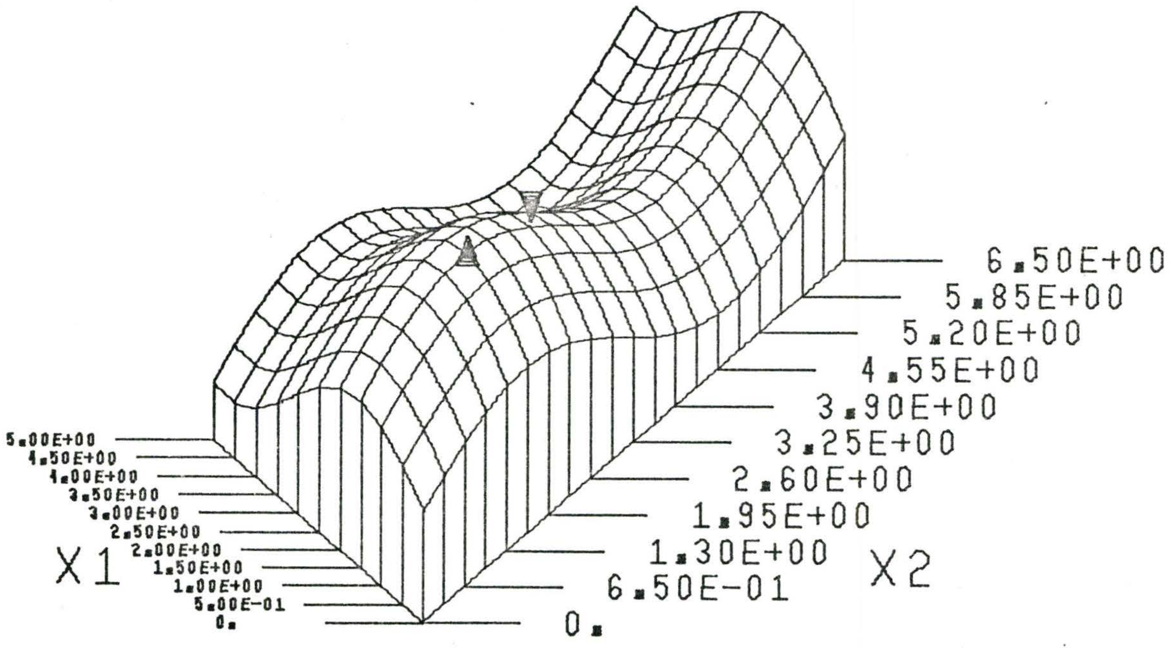


Figure 11.13

ANGLE = 60.0

considerations, the maximum allowed value for PAPER is 28 inches and if a value outside the limit is specified, the program terminates.

There are no means of checking the actual size of the plotting paper used and for this reason, the plotting may be unsuccessful if a matrix generated for 30 inch paper is plotted on 12 inch paper.

For the parameter ANGLE, PAPER has a special meaning as it represents the overall possible size of the mapped plot. If a maximum occurs at U(ISIZE,JSIZE) and the angle of view is $\xi_{max} = \arctan(a/b)$, see Chapter 7.

11.10 XLENG

XLENG is the length of the larger side of the matrix specified in inches as plotted by the program and forms the one dimension of the base sides. Both base dimensions correspond to the sides of the matrix and are in the mutual ratio given by (5.1). For the special case JSIZE=ISIZE, both dimensions of the base are equal, i.e.

$$L = \text{XLENG} \qquad (11.1)$$

where L is the length of the appropriate base line.
For JSIZE>ISIZE, the shorter base side is

$$L = \frac{\text{ISIZE}-1}{\text{JSIZE}-1}\,\text{XLENG} \qquad (11.2)$$

For JSIZE>ISIZE, the length of the shorter base side is

$$L = \frac{\text{JSIZE}-1}{\text{ISIZE}-1}\,\text{XLENG} \qquad (11.3)$$

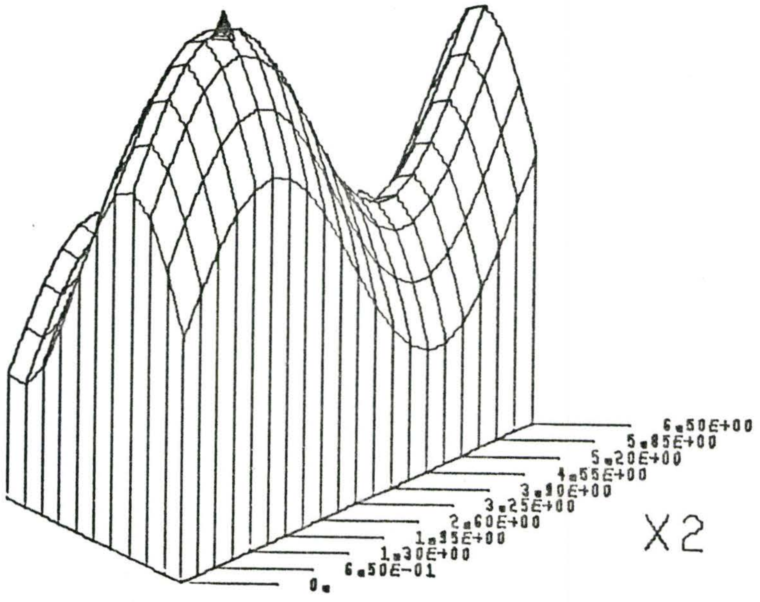


Figure 11.14

XLENG = 2.0

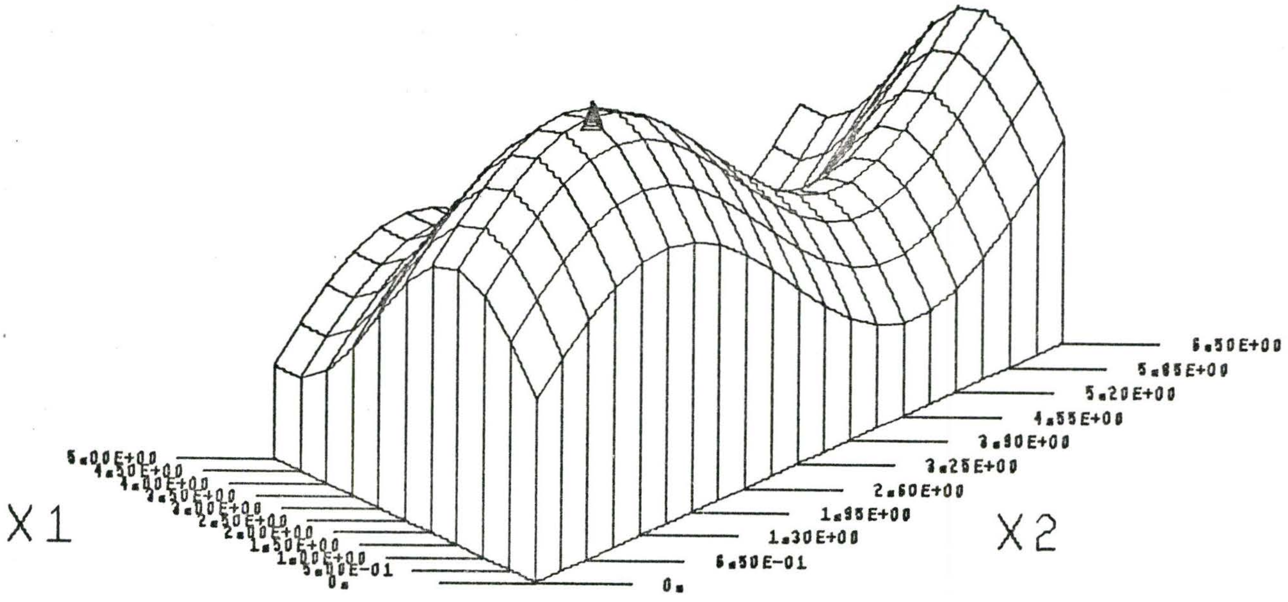


Figure 11.15

XLENG = 3.0

If XLENG is specified together with ANGLE, its value is ignored as a new angle is calculated that is compatible with the specified value of ANGLE. If XLENG is negative, the program terminates with an error message.

An upper limit for XLENG is not built into the program, but it is apparent from any plot that if ALPHA and XLENG (for a given size of matrix ISIZE and JSIZE) are not correctly chosen, the far corners of the base of the plot FC will exceed the overall size of the solid mapped (VSIZE). See Figure 4.6. Should this happen, the program will terminate with an error message. The effect of changes in XLENG is illustrated in Figures 11.14 and 11.15.

11.11 VSIZE

VSIZE is a parameter that specifies the overall size of the mapped plot in inches as shown on Figures 11.16 and 11.17 where VSIZE is specified as 2 and 4 inches respectively. VSIZE is tested for being less than or equal to the paper size and greater than zero. If the mapped plot will not fit onto the paper, the program terminates with a diagnostic message. This test is bypassed when the parameter ANGLE is specified,as VSIZE is then calculated with respect to parameters PAPER and ANGLE. If ANGLE is not specified, VSIZE, XLENG and ALPHA have to be matched in such a way that the projected distance of the far corner of the plotted matrix is less than the maximum specified height of the plot. This can be explained as follows:

Assume that Figure 4.6 shows the mapped solid in its original

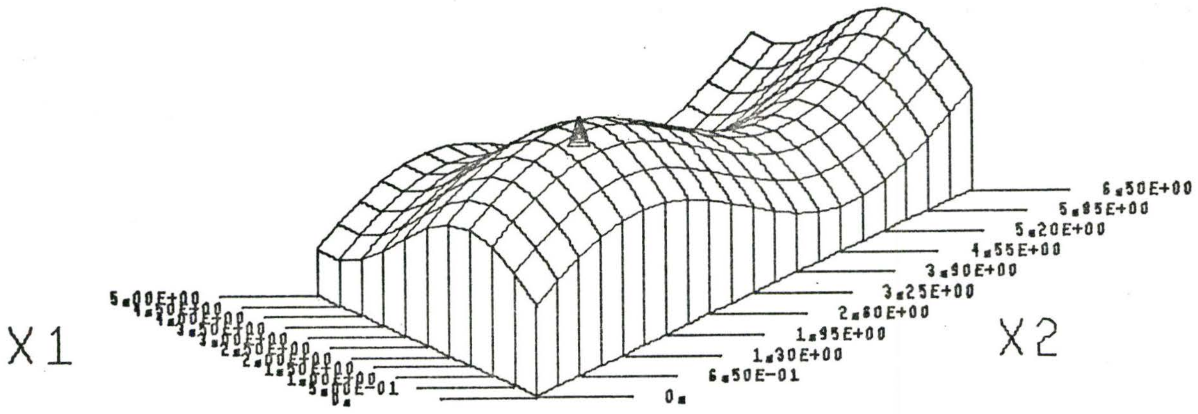


Figure 11.16

VSIZE = 2.0

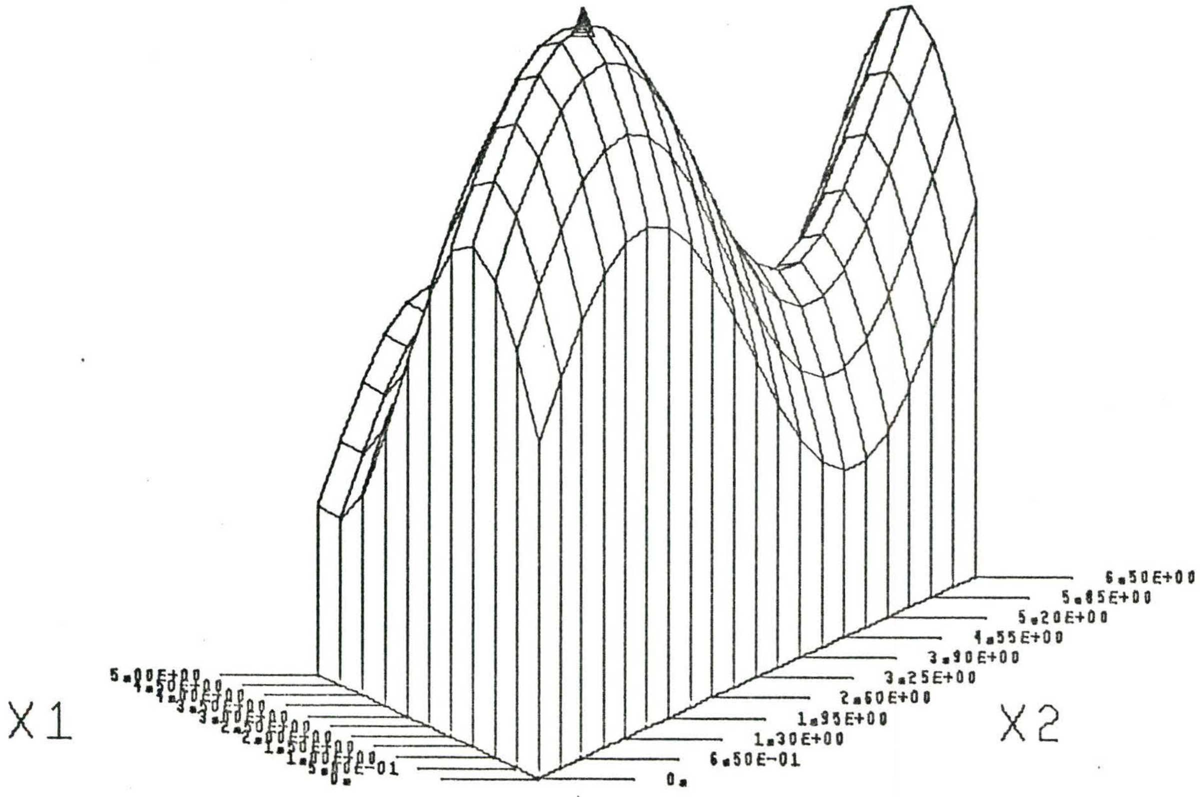


Figure 11.17

VSIZE = 4.0

position, i.e. for an angle of rotation XYROT = $k.2\pi$ where $k=0,1,2....,n$; then,

$$FC = XLENG.SIN(ALPHA)+L.SIN(ALPHA)$$

since JSIZE >ISIZE from (11.2)

$$L = \frac{ISIZE-1}{JSIZE-1}\ XLENG$$

therefore,

$$FC = XLENG.SIN(ALPHA)(1+\frac{ISIZE-1}{JSIZE-1}) \qquad (11.4)$$

It is obvious from (11.4) that for a square matrix where ISIZE=JSIZE

$$FC=2.XLENG.SIN(ALPHA)$$

VSIZE is tested against the computed FC and if less than or equal to FC the program is terminated with error messages.

11.12 XYROT

Parameter XYROT specifies the angle of rotation of the plot in the horizontal X-Y plane (or X1-X2) plane. The angle can be either positive or negative; however, it must be a multiple of 90°. Positive XYROT causes rotation in an anticlockwise direction and negative XYROT rotation in a clockwise direction. If XYROT is not a multiple of 90 degrees the program will terminate with an error message.

For rotation of the matrix, the program uses logical unit no. 2 which has to be reserved even if no rotation is going to be requested as this unit has another function as well, as will be discussed in another section. Figures 11.19 and 11.19 show the object rotated by +90 and -90 degrees respectively.

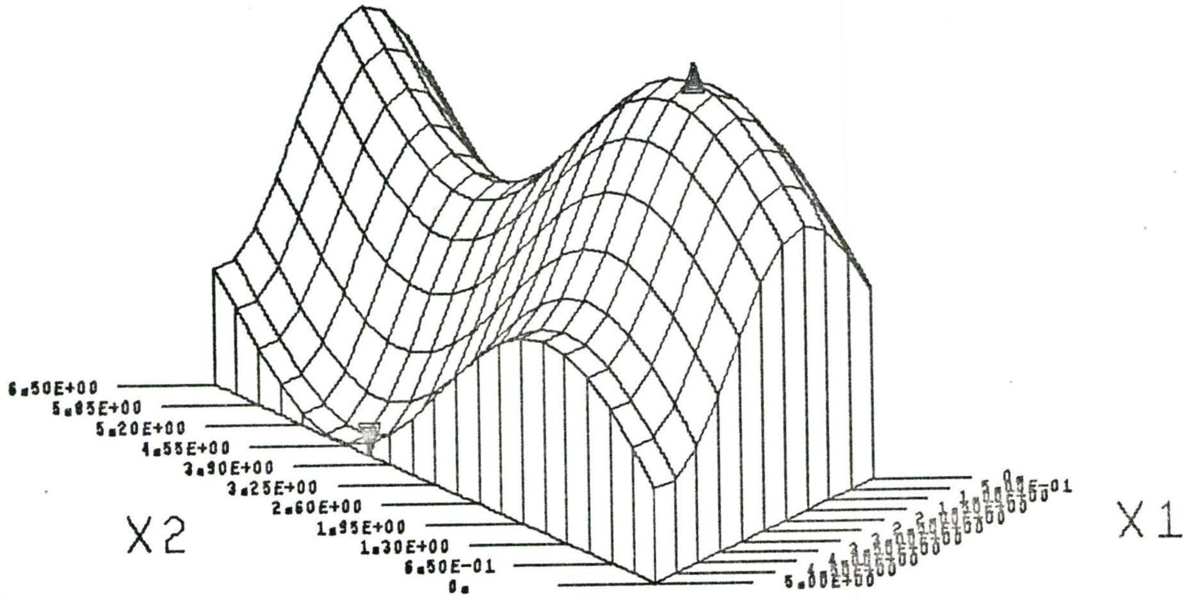


Figure 11.18

XYROT = 90.0

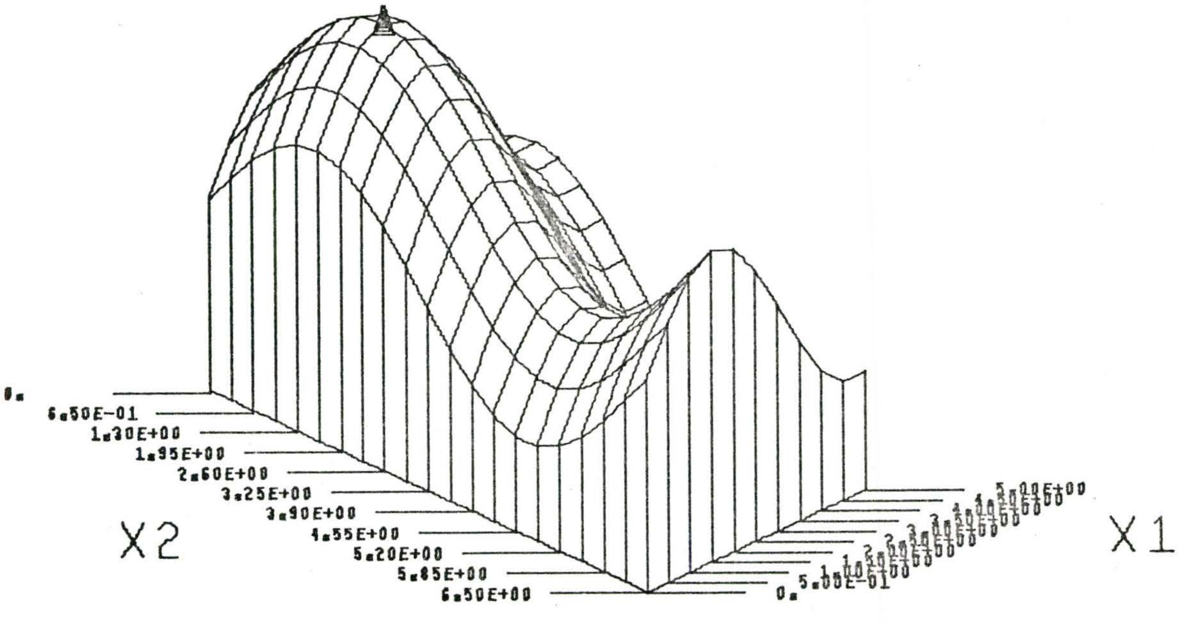


Figure 11.19

XYROT = -90.0

11.13 X1MIN, X1MAX and X2MIN, X2MAX

These parameters specify upper and lower limits for the independent variables X1 and X2 respectively. These limits are essential if the user subroutines OBJECT and/or CONST are inserted by the user because of the way in which the plotting routine calls these subroutines; i.e. in the case KLASS=1 and/or $0<\text{NCONS}\leq 20$ and/or $0<\text{NEQUA}\leq 5$.

The upper and lower limits of each variable are tested for mutual equality and, if this test is positive, the program is terminated with an error message.

For the case where the parameter KLASS=2 or KLASS=3 and NCONS=0, these parameters are ignored if specified as they are not needed for the plotting.

The change in limits is illustrated by Figure 11.20.

11.14 Defining of the Parameters

All the input parameters (except ANGLE) are preset to the default value shown in Table 1 before execution of the program starts. The parameter ANGLE is preset to the dummy quantity.

It will be apparent from the discussion of the input parameters that a simple form for the input data card deck would be the parameters X1MAX,X2MAX (assuming X1MIN and X2MIN=0) and the terminator ENDDA or KLASS=2 and the terminator ENDDATA. In this case the other parameters and switches would take their default values, KLASS would have a value of 1 and the user subroutine OBJECT would have to be supplied to generate the matrix in the first case. In the second case, the user subroutine UFILL together with a matrix 51x51 would have to be supplied.

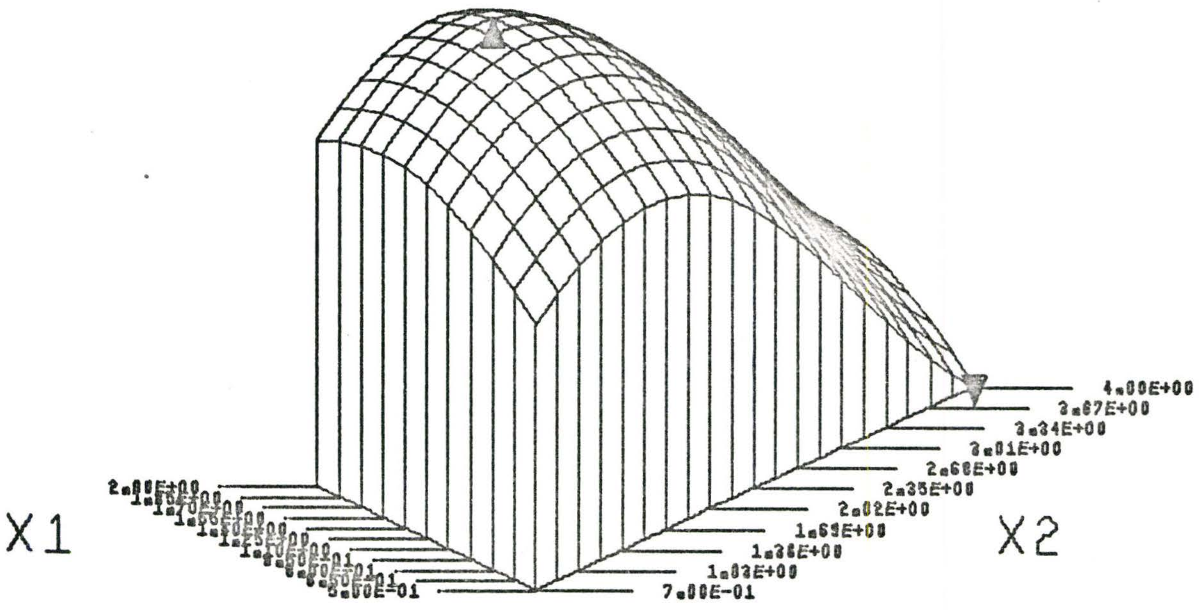


Figure 11.20

X1MAX=2.0,XMIN=0.5,X2MAX=4.0,X2MIN=0.7

CHAPTER 12

DESCRIPTION OF SWITCHES

12.1 General

The switches are required to route the program run and are turned off when equal to 0 and on when set to 1. When turned on, the action corresponding to a given switch will be initiated. There are five switches in the program, all of which have a default value of "off" (0) (refer to table TAB 2).

SWITCH		DEFAULT VALUE	EXAMPLE OF THE USE
1	NOLAB	0	NOLABEL
2	NOPLO	0	NOPLOT
3	PRINT	0	PRINT
4	PUNCH	0	PUNCH
5	ENDDA	N/A	ENDDATA

TABLE 2

LIST OF SWITCHES

12.2 NOLAB

This switch suppresses the printing of all comments and labels normally written on the mapped plot, except for the user's identification, checking marks and the plot terminating message.

Figure 12.1 shows an example of a plot with the labelling suppressed.

12.3 NOPLO

The switch NOPLO suppresses the entire plotting procedure and no plot file is written at all. This switch is useful when used with switches PRINT and/or PUNCH, for cases where only the interpolation routines included in the package are needed. In such cases, the user may enter an incomplete matrix which will be completed by interpolation with the plotter bypassed.

12.4 PRINT

The PRINT switch causes the matrix to be printed out in both the original, unscaled form as it enters the preparation procedure for plotting and in the scaled form as it would normally be plotted complete with marked elements that violate inequality constraints if any were specified.

12.5 PUNCH

When the switch PUNCH is turned on, the matrix is written in binary form in logical unit no. 8 as two files of a similar structure to those written when the switch PRINT is on.

The first file is the original unscaled matrix, the second file is the scaled matrix ready for plotting. Both matrices are written column-wise and are preceeded by two integers recording the dimensions of the matrix. The files are written in the same format as if the write statement for each was:

```
WRITE(8)ISIZE,JSIZE,((U(I,J),I=1,ISIZE),J=1,JSIZE)
```

This discussion is expanded in the section on "Input/Output Files".

12.6 ENDDA

ENDDA is the key word terminating the reading of input data (discussed further under "Input"). Since it has no "on" or "off" state, it is not a switch in the true sense; however, its function is that of a switch (i.e. decoded in the same way by the subroutine INPUT).

12.7 Setting the Switches

All the switches are automatically set to zero before the start of program execution. The switches remain off by default but can be turned on by naming and setting the appropriate switch to 1 on the input card(s). The routine decoding the input data then finds the key word and turns the switch on.

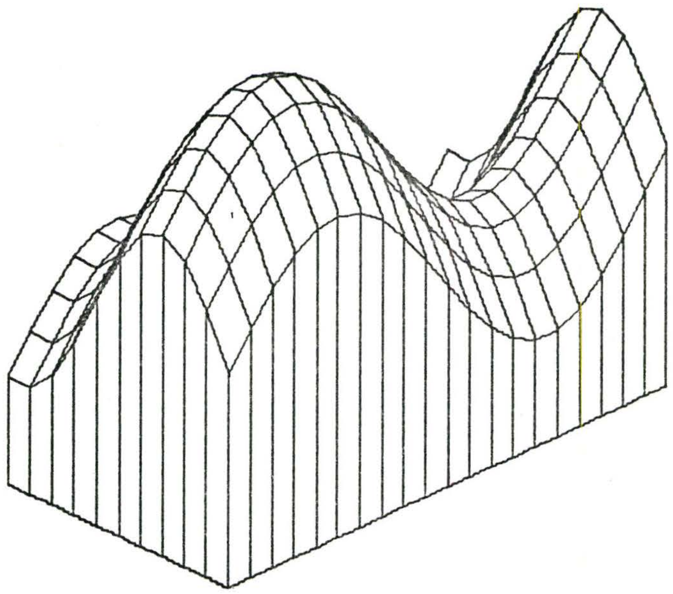


Figure 12.1

Switch NOLAB "On"

CHAPTER 13

OUTPUT

There are two independent outputs from the program.

a) The mapped plot (from a plotter),

b) Printed output with information pertaining to the plot (from a line printer).

13.1 The Plot

The plot is the primary output comprising a 3-D drawing and comments (or labels). The drawing represents a body with a square or rectangular base, positioned with one edge towards an observer so that the angle of projection ALPHA is equal on both sides as shown in Figure 4.6.

The reason for choosing equal angles of projection was discussed in 5.1.

The standard size paper used is 12", with an effective drawing width of 10 inches. If necessary, 30" paper may be used, with an effective drawing width of 28".

The vertical size of the drawing is given by VSIZE, a parameter that may vary from zero to the maximum effective width of the paper (discussed uner parameters). Unless the default value is changed, the vertical height of the plot will be 10 inches.

The horizontal size of the plot depends on the type of matrix and the value of the parameters XLENG and ALPHA. Because XLENG specifies the length of the longer side of the base and ALPHA the angle of projection (Figure 4.6), it is obvious that the horizontal size is a maximum when a square matrix is plotted and is equal to

$$2. \text{XLENG} \cdot \cos (\text{ALPHA}) \tag{13.1}$$

For a rectangular matrix the horizontal size will be;

$$\text{XLENG} . \cos (\text{ALPHA}) + k. \text{ XLENG} . \cos (\text{ALPHA}) \tag{13.2}$$

where

$$k = \frac{\text{i-dimension-1}}{\text{j-dimension-1}} \leq 1$$

The base of the mapped plot is scaled and marked within the limits X1MIN,X1MAX,X2MIN and X2MAX. The difference is divided into ten equal intervals and marks are drawn on the respective sides of the base together with a numerical value (for each mark).

The standard size of the numbers is 0.12"; however, if the numbers overlap, this size is reduced to 0.06", or the marking of numerals is suppressed entirely for that particular axis (compare Figure 11.8 and 11.9). The base axes are also marked to show which variable they represent, namely, X1 or X2. When the mapped plot is rotated, the scaling and symbols remain in their correct places (compare Figures 11.1, 11.17 and 11.18).

If the parameter KLASS is set equal to 1, the mapped plot is assumed to represent an optimization problem and additional symbols are drawn. Symbols Δ and ∇ mark the maximum and the minimum altitudes of the surface respectively; i.e. the maximum and minimum values of the function.

If equality constraints are specified, the paths of the constraints are marked with numerals 1 to 5. Each numeral represents an equality constraint as specified in the user subroutine EQUAL: for example, the numeral 1 could represent an equality constraint PSI(1) (see Chapter 9). When equality constraints are specified the symbols Δ and ∇ mark the maximum and the minimum elevations of the surface subject to the equality constraints.

The symbols representing maximum and minimum elevations of the surface do not appear on mapped plots where the matrix has been read into the program by means of a data file, i.e. for KLASS>1 because in such cases the package is intended for use in visualizing a set of given data (see Figure 11.8).

A heading is printed above the mapped plot containing up to 80 characters as read from a card (see Input Data, Chapter 10).

All parameters, together with their numerical value are listed on the left of the drawing to complete the information on the plot (Figure 13.1). If it is necessary to produce a plot with no labelling, the labelling can be suppressed by turning the switch NOLAB "on" (see NOLAB, paragraph 12.2). The list of activated switches is not printed on the plot, but can be found on the printed report from the line printer (Figure 13.2).

In addition to the comments pertaining to the problem being solved by the package, there is additional information printed on the plot dealing with the actual plotting procedure and bearing no relation to the problem being solved.

At the start of the plot there is a written identification of the plot and the date of the run. Identification is stored in the array KODE(5) which contains 20 characters and is preset to MCM/PPZ72-3D-PACKAGE before execution of the package starts. The identification characters can be redefined by the user in the calling program as shown in paragraph 9.2.

The data is supplied by the FORTRAN function DATE called by the package as CALL DATE (DD) (12) where DD is a ten character Hollerith constant containing the date in the form bmm/dd/yyb

ZACHAR P. P. 03/09/72

```
LIST OF PARAMETERS

INTX1  =    10
INTX2  =    20
ISIZE  =    11
JSIZE  =    21
NCONS  =     1
NEQUA  =     1
kLASS  =     1
ALPHA  =   2.50E+01
ANGLE  =     N/A
PAPER  =   1.00E+01
VSIZE  =   3.00E+00
XLENG  =   2.50E+00
XYROT  =   0.
X1MAX  =   5.00E+00
X1MIN  =   0.
X2MAX  =   6.50E+00
X2MIN  =   0.
```

Figure 13.1

Identification of Plot and List of Parameters where b is a blank

dd is a day of the month

mm is the month

yy is the year.

If necessary, the user may supply his own subroutine DATE and use it for generating a different hollerith constant to be used as additional identification or comment or date in a different form.

Following the identification, there are positioning marks drawn on the paper. These marks are 0.2 inches long and 10 inches apart and perform two functions as follows:

a) Since the lower one has the same Y-co-ordinate as the origin; i.e. the lower front corner of the mapped plot and the upper one is 10 inches above it, they mark the maximum size of the plot and therefore serve to locate the position of the paper in the plotter.

b) After the plotting process has been completed, the marks are extended by an additional 0.2 inches. If these extensions do not line up with the lines drawn at the start of the plotting process and form a continuous line 0.4 inches long, the plotter has malfunctioned and the plot must be discarded.

When a plot has been completed, the message END OF PLOT is written on the paper.

Printed Output

The printed output comes fromthe line printer and is called REPORT ON PLOTTING (Figure 13.2). This report is actually written on logical unit no. 6 and could be exited on a device other than a line printer.

The comment card read in as part of the input is printed under the label HEADING as a string of 80 characters between asterisks. A list of parameters follows the heading together with a numerical value and explanation for each parameter. If any switches have been activated they are listed after the list of parameters.

Before generating the matrix for plotting, the program tests for sufficient memory space and before the memory test is made the message "TEST FOR SUFFICIENT MEMORY SPACE" is printed. This message is followed by information on the space needed and the space available. If the memory is not large enough, the program terminates with an error message. However, since the information on the available space is calculated by subroutine DIFFER, it may not appear if the plotting package is assembled with a modified version of this subroutine (paragraph 16.2). If the memory is large enough, the message "TEST PASSED" is printed.

During the search for maximum and minimum elements, any number of "finds" will be recorded and when there is more than one of either or both, a message appears recording the values and frequency. When the extremes have been found, their values and location in the matrix are printed. For multiple maxima

```
 REPORT ON PLOTTING
==================

HEADING  ***  CHANGE OF PARAMETERS

LIST OF PARAMETERS
------------------

INTX1  =  10           NUMBER OF INTERVALS PLOTTED ALONG X1 AXIS
INTX2  =  20           NUMBER OF INTERVALS PLOTTED ALONG X2 AXIS
ISIZE  =  11           I-DIMENSION OF THE PLOTTED MATRIX U(I,J)
JSIZE  =  21           J-DIMENSION OF THE PLOTTED MATRIX U(I,J)
NCONS  =   1           NUMBER OF INEQUALITY CONSTRAINTS
NEQUA  =   0           NUMBER OF EQUALITY CONSTRAINTS
KLASS  =   1           MATRIX WAS GENERATED BY MEANS OF SUBROUTINE OBJECT
ALPHA  =  2.500E+01    ANGLE OF THE BASIS OF THE PICTURE (DEGREES)
ANGLE  =    N/A        ANGLE OF VIEW (DEGREES)
PAPER  =  1.000E+01    MINIMUM SIZE OF THE PLOT PAPER (INCHES)
VSIZE  =  2.500E+00    VERTICAL DIMENSION OF THE PICTURE (INCHES)
XLENG  =  2.100E+00    LENGTH OF THE LONGER SIDE OF THE BASIS (INCHES)
XYROT  =  0.           ANGLE OF ROTATION OF THE ORIGINAL MATRIX (DEGREES)
X1MAX  =  5.000E+00    UPPER BOUND OF THE VARIABLE X1
X1MIN  =  0.           LOWER BOUND OF THE VARIABLE X1
X2MAX  =  6.500E+00    UPPER BOUND OF THE VARIABLE X2
X2MIN  =  0.           LOWER BOUND OF THE VARIABLE X2


TEST FOR SUFFICIENT MEMORY SPACE
   SPACE NEEDED IS        231 LOCATIONS
   SPACE AVAILABLE IS    5498 LOCATIONS

TEST PASSED

MINIMUM VALUE  -1.96437E+00 FOUND AT I =  10, J =  15   FOR X1 =   4.500000E+00 AND X2 =

MAXIMUM VALUE   1.90783E+00 FOUND AT I =   5, J =   6   FOR X1 =   2.000000E+00 AND X2 =

*** PLOTTING COMPLETED ***
```

Figure 13.2
Report on Plotting

the element with the largest sum of element subscripts i+j is recorded. For multiple minima it is the element with the smallest sum of subscripts i and j. Additional information on extremes is printed when any equality constraints have been printed. This information comprises maxima and minima and their locations for each equality constraint listed in the same order as they have been specified in subroutine EQUAL.

At the end of the program run, the message ***PLOTTING COMPLETED*** is printed to confirm that the plot has been successfully generated.

If any constraints were specified, a symbolic printout of the plotted matrix called "TOP VIEW OF THE PLOTTED SURFACE" appears on the next page. Figure 13.3 shows a sample printout. Symbol + represents an element of the matrix that does not violate any inequality constraint. The symbol - represents the area in which at least one inequality constraint is violated. The numerals 1 to 5 represent paths of equality constraints (if any) in the same order in which they have been specified in subroutine EQUAL. The matrix is printed so that the I subscript varies along the vertical axis, starting with I=1 at the bottom of the printout. The J subscript varies along the horizontal axis from left to right. This orientation of the matrix was chosen because it corresponds visually to the drawing produced by the program, the lower left corner of the printed plot corresponding to the front vertical edge of the mapped plot.

```
TOP VIEW OF THE PLOTTED SURFACE - MATRIX U(I,J)

       5   10   15   20
       I    I    I    I
11 +++++++++++++++++++++   11
10 +++++++++++++++++++++   10
 9 +++++++++++++++++++++    9
 8 +++++++++++++++++++++    8
 7 +++++++++++++++++++++    7
 6 -++++++++++++++++++++    6
 5 -----+++++++++++++111    5
 4 -------+++++111111+++    4
 3 --------1111+++++++++    3
 2 --------+++++++++++++    2
 1 --------+++++++++++++    1
       I    I    I    I
       5   10   15   20
```

Figure 13.3

Line Printer Representation of the Figure 11.6

Both subscripts are printed around the plot. Because of the limited number of characters that can be printed on a single line, the J subscript is suppressed if it exceeds 123 and J values are truncated at this number. The I subscript is unconstrained as regards printing and the plot is printed with page skipping suppressed. If for some reason it is essential to print the full range of values for J, the matrix can be rotated and then the I subscript will be truncated if the value exceeds 123.

The "plotting completed" message may be followed by a listing of the matrix if switch PRINT is turned "ON" and/or by a message pertaining to punching of the matrix on cards if the switch PUNCH is turned "ON".

In all cases where the program was prematurely terminated by "fatal" errors, the diagnostic message appears as the last printed information, (see section on "diagnostic messages").

13.3 Diagnostic and Fatal Errors

All input data and partial results are checked for validity, limits and correct logical combination before the actual plotting is started. There are 18 possible fatal errors that may occur, that result from incorrect input data specification. These errors are detected during program execution and each of them causes premature termination of the program. Before the program terminates a diagnostic message is printed and the program stops with a corresponding STOP n in the dayfile where u is the octal constant from 1 to 22 referring to the particular error. A list of diagnostics is contained in APPENDIX B.

CHAPTER 14

COMPUTER MEMORY

14.1 General

The complete package with its 34 subroutines represents about 2,800 cards of FORTRAN coding and the memory requirement for storing the entire program with its arrays is 35K. This, however, does not mean that 35,000 memory locations are enough for execution of the program. There must be space for a loader to load the program into the memory which represents another 7,000 to 11,000 locations and in addition to all this, space is required for the user's subroutines. Although, in most cases, the coding of the user's subroutines does not occupy significant memory space, they have to be taken into account, particularly if they contain any arrays, because of the large space required for the I/O buffers. There are three ways to significantly reduce the memory required:

a) Dynamic storage allocation - this is built into the program,

b) Selection of a loader,

c) Specification of number of input/output files and the sizes of their buffers

14.2 Dynamic Storage Allocation

The 35K memory locations necessary for storage of the package coding does not include an array for the matrix to be plotted.

This matrix array is the largest array the program uses. Its size is dependent on the problem being solved and may cause the program to abort if it cannot be stored in the memory allocated to the program.

Because the loader that loads the program into the memory is only needed for this purpose has to be allocated a memory space, its space can be used as additional storage space at the time of execution. It can thus be used as additional storage after it has served its purpose if the arrays are assigned to blank COMMON. Blank COMMON begins immediately after the last routine loaded and may expand up to the last location of the memory allocated to the program, overlapping the loader and loader labels (13) (see Figure 14.1).

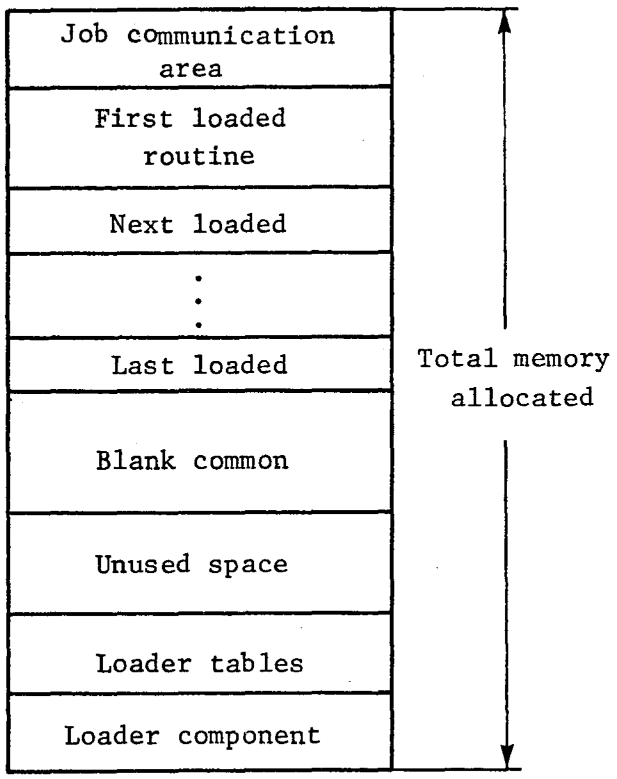


Figure 14.1

Computer Memory Structure

If the blank COMMON specified exceeds the memory allocated, it is truncated reserving all remaining core for storage purposes. The user is informed about this situation by the message "BLANK COMMON EXCEEDS AVAILABLE CORE, TRUNCATED" which is printed on the dayfile. This idea is employed in the program for storing the matrix to be plotted.

The routine PLOT3D called by the user specifies in the blank COMMON array WORK which has a size of 30K locations. It calls subroutine INPUT to obtain the size of the matrix. It then passes the address of the first word of the array WORK, together with the size of the matrix on to the plotting routine SURFGG, which handles the array WORK as a two-dimensional matrix u.

The calling sequence is (see flowchart 14.2):

```
      SUBROUTINE PLOT3D
C
      COMMON WORK(30000)
C
      CALL INPUT(ISIZE,JSIZE)
C
      CALL SURF99(WORK,ISIZE,JSIZE)
C
      RETURN
      END
      SUBROUTINE SURF99(U,NUI,NUJ)
      DIMENSION U(NUI,NUJ)
      .
      .
      .
```

$$\begin{tikzcd}
\boxed{\text{PLOT3D}} \arrow[d] & \\
\boxed{\text{Specify working array in blank COMMON}} \arrow[r] \arrow[d] & \boxed{\text{INPUT}} \\
\boxed{\text{Fetch input data}} \arrow[r] \arrow[d] & \boxed{\text{SURF99}} \\
\boxed{\text{Generate the drawing}} \arrow[d] & \\
\boxed{\text{RETURN}} &
\end{tikzcd}$$

Figure 14.2

Fowchart of Subroutine PLOT3D

The amount of core occupied by the loader is about 7 to 12K (i.e. 7,000 to 12,000 octal locations). The actual size varies depending on the user's subroutines but nevertheless, there are automatically a minimum of 4,500 decimal locations available as storage space for the matrix. The figures refer to the different loaders (see paragraph 14.3).

If the default storage is not sufficeint, it can be increased by requesting more core for the program and for a requested total storage of 124K the storage available varies from 23K to 26K decimal locations with respect to the number of input/output files and sizes of their buffers (see I/O buffers).

Table TAB3 shows the minimum memory requirements and guaranteed maximum storage space available for a number of different situations assuming reasonably small users subroutines.

A COMPASS memory searching routine called SPACE is supplied with the package. The routine using the job communication area and loader area obtains both the size of memory as requested by the user and the last word of the loaded program (reference 14).

This information is processed by subroutine DIFFER which, if the space available is not sufficient for storing the full matrix, terminates the program prematurely.

No. of Files	Size of I/O buffers (dec)	Minimum memory required (octal)		Maximum total storage for memory 124K (octal) (in decimal)
		PP Loader	CP Loader	
4	1024 (def)	54K	56K	25,000
5	1024 (def)	56K	60K	24,000
6	1024 (def)	60K	62K	23,000
4	512	50K	52K	26,000
5	512	51K	53K	25,500
6	512	52K	54K	25,000
Maximum default storage (in decimal)		3,500	4,500	

TABLE 3

MEMORY REQUIREMENTS

14.3 Selection of the Loader

For computers with two types of loaders, the user can cut down the memory requirement by selecting the PP loader, that is, if it is not selected automatically as the installation's default loader. This will decrease the memory required to load the program by approximately 2K when compared with the memory required if the CP loader is used (TAB 3). On the other hand, loading is faster when the program is loaded with the CP loader (reference 15).

14.4 Input/Output (I/O) Buffers

For execution, the program uses a minimum of four files (see I/O files) and the total number of files could be as many as six. Because each file has its own I/O buffer within the memory space assigned to the user's job, these buffers represent a significant portion of the core.

By default, the size of each buffer is specified to be 1024 decimal locations (i.e. 2K octal) by the FORTRAN compiler. For six files, this represents 14K (octal). It is possible to reduce the memory requirement by only assigning files the program is going to use and by the specification of shorter buffers for each file. The assignment of files is described in the section on "INPUT/OUTPUT files".

The size of the buffers can be decreased safely to 1011 octal locations (16), and this represents a saving of about 500 decimal locations per file. This shortening of the buffers causes a decrease in the efficiency of the I/O processes, however, it has been observed from test runs that this decrease in efficiency does not represent any significant decrease in the speed of the runs.

To shorten the memory using this feature, the main program has to be compiled with the buffer size specification. One way of doing this is to set the appropriate parameter on the compilation control card to the size of the buffer (this should not be less than 1011 octal locations).

Example: `RUN(S,,1011)`

The above example would cause the size of the buffers for all files to be equal (i.e. 1011 octal locations).

There is another way of specifying shorter buffers and that is by stating the size of each buffer individually on the PROGRAM card (16,17).

Example: PROGRAM MAIN(INPUT=1011,....)

This would cause the size of the buffer for file INPUT to be 1011 octal locations while the other files remain with buffers of the default size unless also specified.

A review of typical examples is shown in TAB 3. Generally, the minimum memory necessary for the loading and execution of the program is given by the formula:

$$CM = 35000 + \sum_{i=1}^{n} bi + \ell$$

where CM is the memory size in octal

3500 represents the size of the package and service subroutines (octal)

n	is the number of files
b_i	is the buffer size (octal)

$\ell = 7000$ (octal) for the PP loader

$\ell = 11000$ (octal) for the CP loader

CHAPTER 15

INPUT/OUTPUT FILES

15.1 General

During execution, the program uses up to six files, of which four are compulsory and two optional because they are job dependent.

The compulsory files are as follows:

a) TAPE2

b) INPUT,TAPE5=INPUT

c) OUTPUT,TAPE6=OUTPUT

d) TAPE10

Optional files are as follows:

e) PUNCHB,TAPE8=PUNCHB

f) TAPEn

15.2 TAPE2

The logical unit TAPE2 serves two purposes:

a) It is a working file used for rotation of the matrix if required.

b) During execution of the program, it is used for storing a plotted matrix. This is done primarily for printing or punching purposes if the corresponding switches have been activated.

After the program terminates successfully (after execution) TAPE2 is not destroyed (erased) and remains available to the user. The same applies for TAPE8 (paragraph 15.6).

TAPE2 contains two files, both of which are written in binary. The first file contains the plotted matrix in its unscaled form (i.e. as it was generated or read in). This matrix is already completed if KLASS=3 and interpolation was successful, rotated if rotation was requested and has its elements for which the inequality constraints have been violated set to the dummy quantity.

The second file contains the matrix in the form it was transposed to before plotting started, that is, scaled and adjusted for plotting. It contains values between zero and VSIZE that represent the actual elevation of the elements on the paper.

Both matrices are written column-wise as if the equivalent FORTRAN statement was

```
WRITE(2)((U(I,J),I=1, ISIZE),J=1,JSIZE)
```

where the original limits ISIZE and JSIZE have been mutually switched if the rotation angle was

$$w = \pm(90+k180) \qquad \text{for } k=0,1,2....n$$

15.3 INPUT,TAPE5

TAPE5 is the logical unit containing the input data. If TAPE5 is made equivalent to input as is usually done, the data is input from the INPUT file (i.e. card reader).

15.4 OUTPUT,TAPE6

TAPE6 is the logical unit containing the output generated by the program, other than the drawing and punched cards, making TAPE6 equivalent to OUTPUT directs the output to the line printer.

15.5 TAPE10

TAPE10 is the logical unit containing the plot file. It is the magnetic tape that is mounted on the plotting device after the program has successfully been executed. Although TAPE10 is a magnetic tape, it is not requested under normal circumstances by the user because the request is automatically generated by the SYSTEM on the first call of subroutine PLOT (18). This however, is a McMaster software modification and, therefore, the procedure to write the plot tape will be different for other computers. Regardless of the installation, the plot file is generated in all cases and has to have its logical unit specified in one way or another.

15.6 PUNCHB,TAPE8

TAPE8 is the logical unit onto which the matrix is written for further use outside the plotting program. The output written on TAPE8 consists of the two binary files representing the matrix in its unscaled and scaled form respectively as discussed in paragraph 15.2.

In contrast to TAPE2, each file on TAPE8 is preceded by two integers stating the size of the matrix. The write statement for each file can be assumed to be

```
WRITE(8) ISIZE,JSIZE,((U(I,J),I=1,ISIZE),J=1,JSIZE)
```

The file TAPE8 is primarily intended to serve the user in providing a source for the deck with the matrices. For this reason TAPE8 is commonly equated to PUNCHB, which directs that the output be punched in binary by the card punch.

The above feature is not necessary because it is possible to use the system copying utility (15) to control the output from TAPE8. In this case PUNCHB is not stated on the PROGRAM card and the user may select either of the files if it is not necessary to output both of them.

15.7 TAPEn

TAPEn is the logical unit containing the matrix if it is to be read in (i.e. for KLASS=2 or 3). If the matrix is punched in coded form it may be placed in the card reader and no special TAPEn has to be specified as in this case TAPE5 serves the purpose.

When the parameter KLASS is specified other than 1, a call is issued to the user's subroutine UFILL. UFILL reads the matrix using the logical unit number n, i.e.:

```
READ (n,format) list or READ(n) list
```

Since it is not safe to handle mixed mode files (reference (16), the matrix punched in binary may not be read into the program from the card reader; however, it still may be read by the reader and copied onto TAPEn prior to the program execution (see the example 5 in APPENDIX D).

CHAPTER 16

USE OF THE PACKAGE ON OTHER COMPUTERS

16.1 General

Although the program was set up for a CDC 6400 computer, the method and approach to the problem of 3-D plotting can be used on most systems with the necessary hardware and software. However, the program has to be modified for a particular computer system. Also, the differences lie mainly in the coding that solves assisting tasks, such as variable format for printing, writing numbers on the plot, etc.

16.2 Subroutine DIFFER

The parts of the coding that might be different are collected in the subroutine DIFFER where the purpose of a particular statement is explained by a comment preceding the particular statement. Therefore, to adopt the program for another computer it should be sufficient to replace the subroutine DIFFER with a new one with the appropriate modifications.

It might even be necessary to replace this subroutine if the program is assembled by the same computer using a different compiler. An example of this is RUN and FORTRAN Extended (FTN) compilers on CDC computers (reference 15,16).

Also, if FTN is to be used, a modified version of the COMPASS subroutine SPACE must be used because of different subroutine linkage (19).

16.3 Library Routines

If library plotting routines are not available in a particular installation under the names used in the package (see APPENDIX C), the user may supply his own routine in the set of service subroutines calling the correct available subroutine. For example, if the subroutine that moves the pen is called PLOTT instead of PLOT the subroutine supplied will be:

```
SUBROUTINE PLOT (X,Y,N)
CALL PLOTT (X,Y,N)
RETURN
END
```

Because the routines directly involved in generating the mapped plot do not use DIFFER, the conversion of the package for other computers can be done easily and without any chance of interfering with the main algorithm. This feature adds a significant degree of flexibility to the package.

CHAPTER 17

PERFORMANCE AND TIMING (SUMMARY)

During the test runs and a short period of actual use, the program performed very well, producing precise and accurate images of the surface.

It is difficult to give a guideline for estimating the total CPU time required to generate a given plot. This is because, although the time increases with the size of the matrix, it also depends significantly on the shape of the surface. The execution time required to generate figures 11.1 through 11.20 was of the order five to six seconds. The compilation time should also be taken into account, although it is insignificant for most usage of the package since the package will be used in object form. The entire package represents about 2,800 cards and the assembly time is about 20 seconds.

Another time that should be considered is the time the tape is engaged on the plotter while the drawing is being done. For the same drawing, the time is proportional to the size of the drawing, although the computer CPU time will remain the same.

When the subroutine MOVE, paragraph 6.5, was introduced, the plotting time became independent of the computing time and two drawings having close computing time may have vastly different plotting times. If, for instance, after the rotation the front part of the solid shaded the rest of it, the plotting time would be a minimum while the computing time remained without significant change.

As an illustration, the time required to map Figures 11.1 through 11.20 was about five minutes. For a smooth surface and larger matrices with about one third of the points hidden, it may be estimated that for each two seconds of generating time, the plot takes about one minute to be drawn.

SECTION D

SUMMARY - APPLICATIONS

CHAPTER 18

GENERAL

A three dimensional display finds its usage anywhere where the properties of a function of three variables are to be examined. This section deals with some of these application areas, some of which have been investigated in detail and some of which are little more than ideas at this time.

The usefulness of the package lies mainly in the fact that it enables the user to visualize a three dimensional problem, such as a complicated surface that is a function of three variables. Visualizing a function of two independent variables is particularly difficult when the function is not continuous. The introduction of inequality constraints has the effect of making the function discontinuous.

CHAPTER 19

OPTIMIZATION PROBLEMS

Optimization problems involve the optimization of some objective function

$$u = u(x_1, x_2, x_3 \ldots x_n)$$

where x_1, x_2, $x_3 \ldots x_n$ are independent variables.

If there are two independent variables, the objective function can be represented as a surface in three dimensional space. If there are more than two independent variables, the objective function represents a hyperplane and cannot be represented in three dimensional space. The plotting package is, therefore, limited to the solution of three dimensional problems.

Algorithms have been formulated to solve both linear and nonlinear optimization problems. The solution of such problems involves finding a maximum or a minimum value for the objective function, subject to equality or inequality constraints. When the constraints and the objective function are linear the problem is one of linear optimization and algorithms such as the "simplex method" (reference 20 and 2 1) can be used. Simple problems can be solved manually, but more complex problems require the use of a digital computer. When one or all of the constraints or the objective function are nonlinear, the problem of solving for an optimum value of the objective function is considerably more complicated and in some cases it becomes impossible to find a solution. The plotting package is equipped to find a numerical solution to a nonlinear optimization problem, but of considerable importance is the fact that the problem, with its solution is displayed visually.

The constraint functions may be inequality functions, such as

$$\phi_j = \phi_j(x_1,x_2,x_3\ldots x_n)\geq 0 \text{ for } j=1,m$$

or equality constraints such as;

$$\psi_k = \psi_k(x_1,x_2,x_3\ldots x_n) \text{ for } k=1,p$$

where x_i are independent variables

n is the number of independent variables

m is the number of inequality constraints

p is the number of equality constraints

The package has been designed with a view to the solution of optimization problems in design work and also to gain insight into the effect of changing variables in the objective function. Because the package can only handle problems with three variables, the number of independent design variables is limited to two giving the objective function the form

$$u = u(x_1,x_2)$$

Two independent variables could be regarded as a serious limitation of the objective function; however, a choice of the two variables that are allowed to vary independently can be made and in a manner that leaves the nature of the objective function unchanged. A single or a series of sensitivity tests may be performed on the objective function to make the choice of independent variables simpler. It is also often possible to reduce the number of independent variables and by so doing also reduce the number of constraints since constraints on independent variables that have been removed from the problem are redundant.

The program allows twenty inequality and five equality constraints because test runs of the program have shown that these are sufficient in almost all cases. Constrained regions in the mapped solid are defined by inequality constraints in such a way that only that part of the surface that represents feasible solutions is drawn. The grid pattern on the base of the solid is drawn in places where the solid has been "cut away" because of an inequality constraint. The boundaries are clearly represented by vertical lines and this ability to represent multivalue elements (sudden drops from the surface to the base) makes the package unique.

The equality constraints are mapped as paths consisting of numerals, each of which represents a particular constraint.

Both the maximum and the minimum values of the objective function, subject to the constraints are marked on the surface provided that the solid is oriented in such a way that these points are visible.

Figures 19.1, 19.2, 19.3 and 19.4 show views of a typical objective function and the change in the solutions when constraints on the objective function are changed and combined in different ways. By multiplying the original objective function by -1.0, the solution had its maximum and minimum reversed.

Use of the three dimensional plot for visualizing a designer's objective function makes it simple to recognize variables that have a marked effect on the solution and also how the variables affect the objective function. It is also possible to see whether the solution found represents a global or a local optimum.

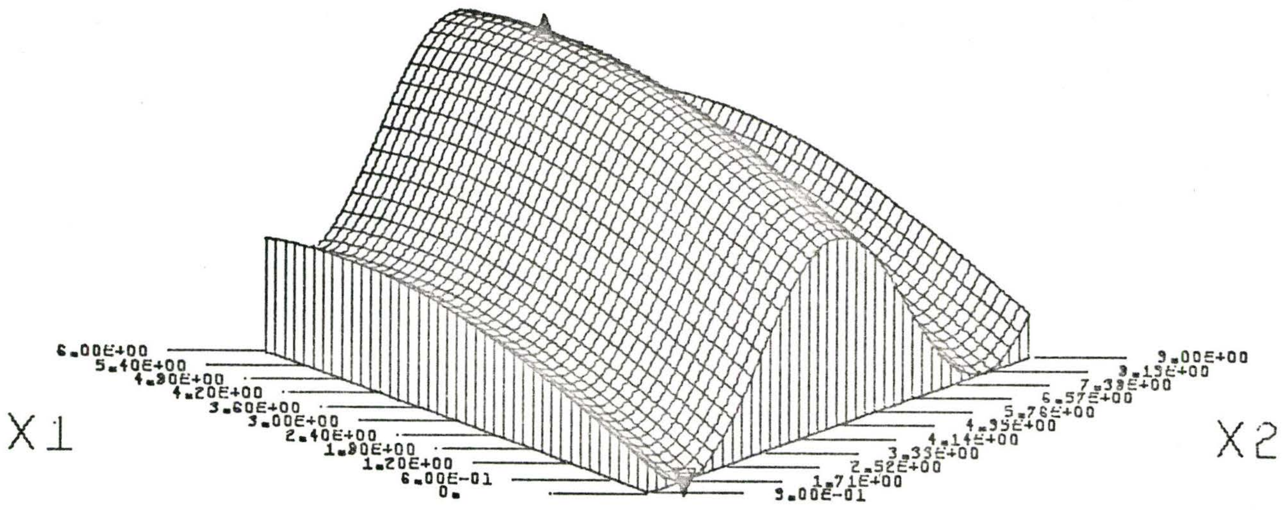


Figure 19.1

Solution of Optimization Problem with no Constraints

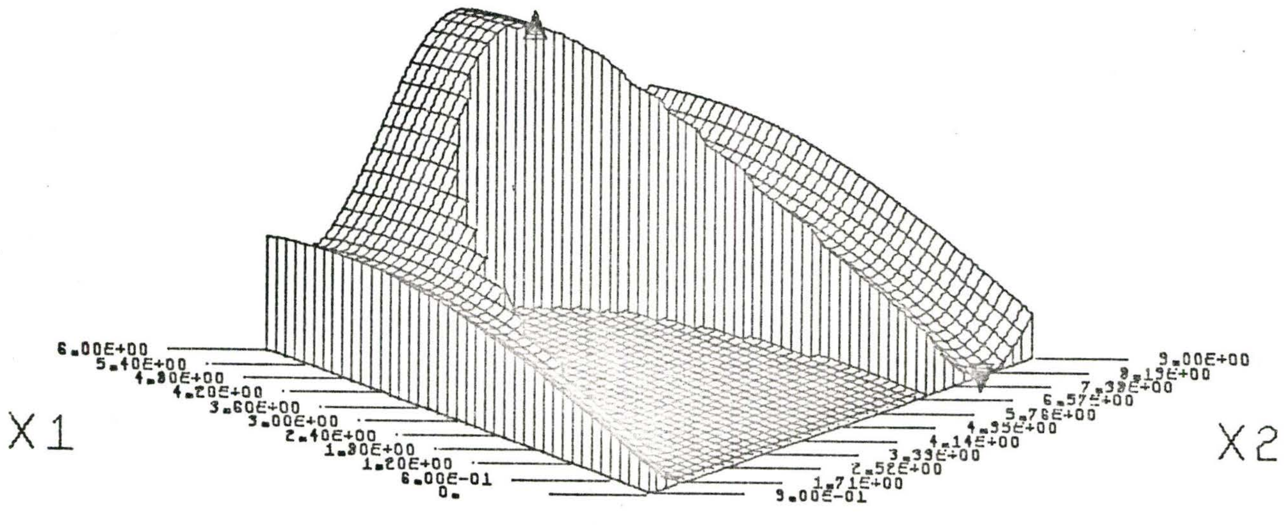


Figure 19.2

Solution of Optimization Problem with One Inequality Constraint

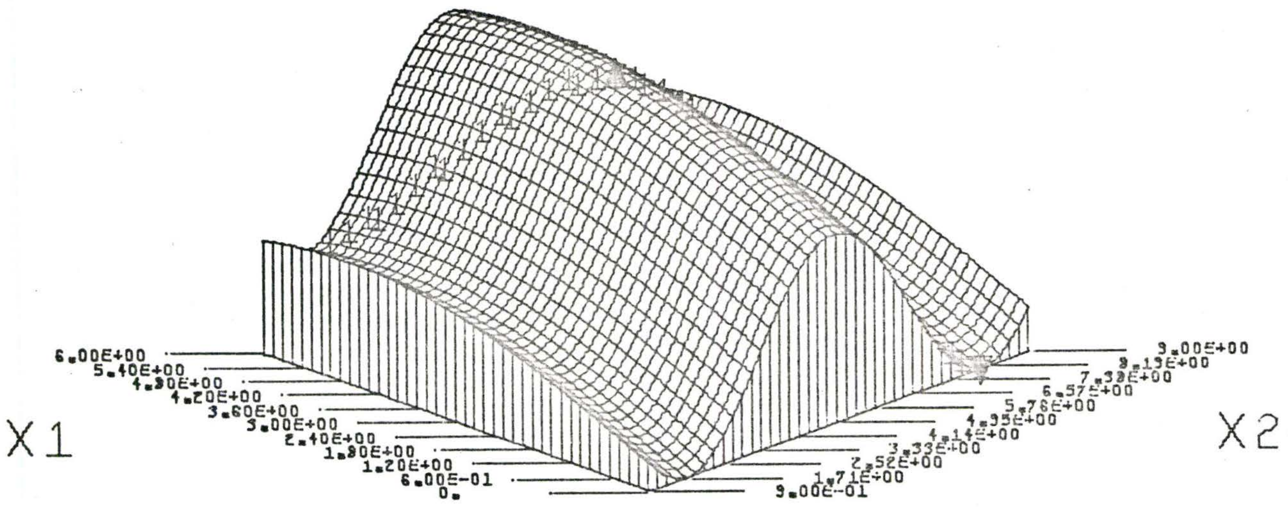


Figure 19.3

Solution of Optimization Problem with One Equality Constraint

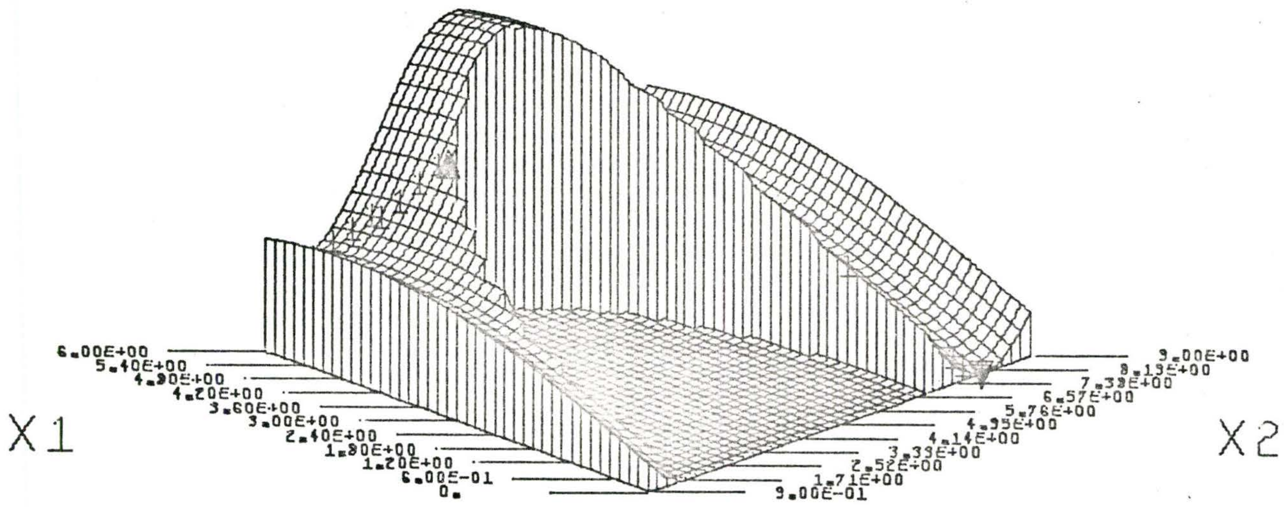


Figure 19.4

Solution of Optimization Problem with One Equality and One Inequality Constraint

CHAPTER 20

FUNCTIONAL ANALYSIS IN DESIGN

Figure 20.1 shows the application of the package to the analysis of the functional relationship between tool life, cutting speed, feed rate and depth of cut in a chip removable machining process.

An extended Taylor's basic equation for tool life (reference 22) defines tool life as:

$$T^n = \frac{k}{V.f^m d^p} \qquad (20.1)$$

where T is the tool life in minutes

V is the cutting speed in feet per minute

f is the feed rate in inches per minute

d is the depth of cut

k is a constant

m, n and p are experimentally determined constants

Combining the feed rate and depth of cut into a single variable representing the cross-sectional area of the cut, the tool life becomes a function of two independent variables.

$$T = T(V,M) \qquad (20.2)$$

where M is the volume of metal removed

Thus, the function is suitable for display by the three-dimensional plotting package. The surface resulting from the study of a model based on equation (20.1) is shown in Figure 20.1.

An analysis of the plot shows that the cutting speed has a bigger influence on tool life than the volume of metal removed. Using the scale along the vertical axis, readings of tool life may be taken from the surface for a given cutting speed and volume of metal (maximum and minimum values are obtained from the printout accompanying the plot). Plots similar to Figure 20.1 may be extremely useful in machine shops if supplied in a suitable form.

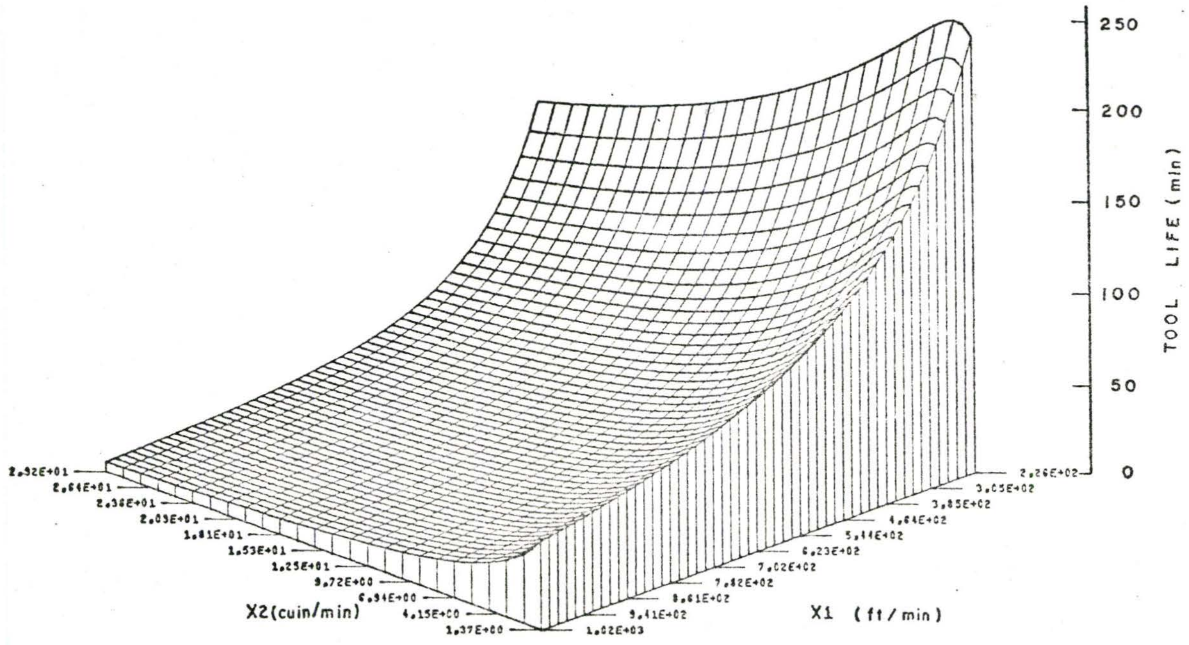


Figure 20.1

Response Surface for Tool Life

CHAPTER 21

EVALUATION OF DATA SETS

The three dimensional plotting package has application in the preparation of diagrams for visual aids. A fast and cheap line drawing of a family of curves, or a response surface can be drawn as viewed from each of four corners, and the plots quickly coloured (eg. using letra-tone). If all four views are photographed together in one illustration, readers are given maximum information in an easily assimilable form.

The 3-D package can also be used to plot a 2-D transient response (eg. longitudinal dispersion against time) or a 3-D response at various instances of time. Figures 21.1 and 21.2 show a flood wave propagation through a body of water as seen from opposite corners. Alternatively, in the latter case, a simple adaptation can be made by plotting a relavent parameter, eg. the instantaneous peak flood levels, against time.

In the case of such transients, the 3-D package is particularly suitable for illustrating the effect of several inputs distributed along the boundary of the configuration (eg. pollutant, flow, etc.), and provides a worthwhile condensation of otherwise unwieldy tables of numbers, inherent in the output of most models. Figures 21.3 and 21.4 show a graphical representation of the

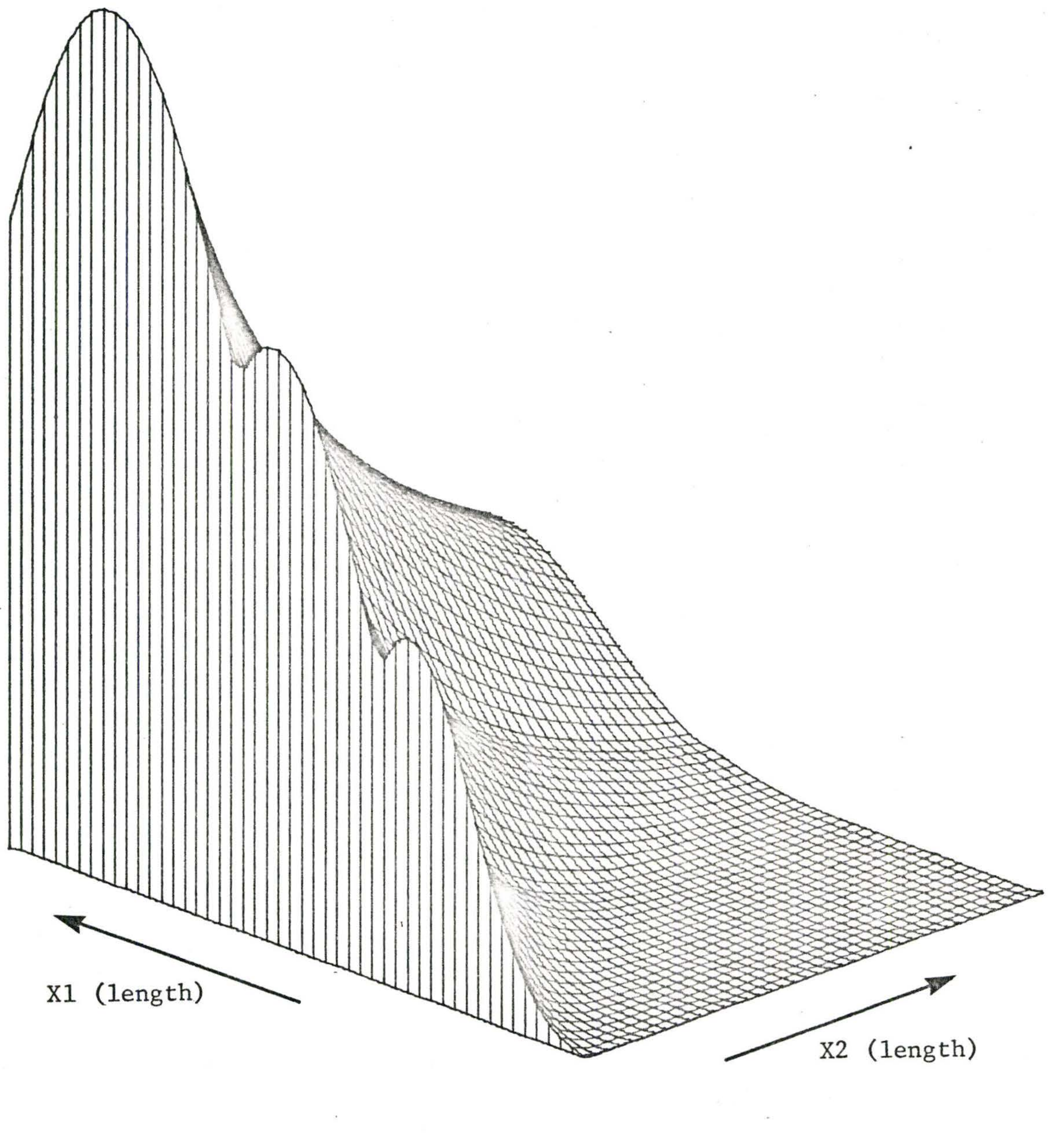


Figure 21.1

Flood Wave Propagation - Basic View

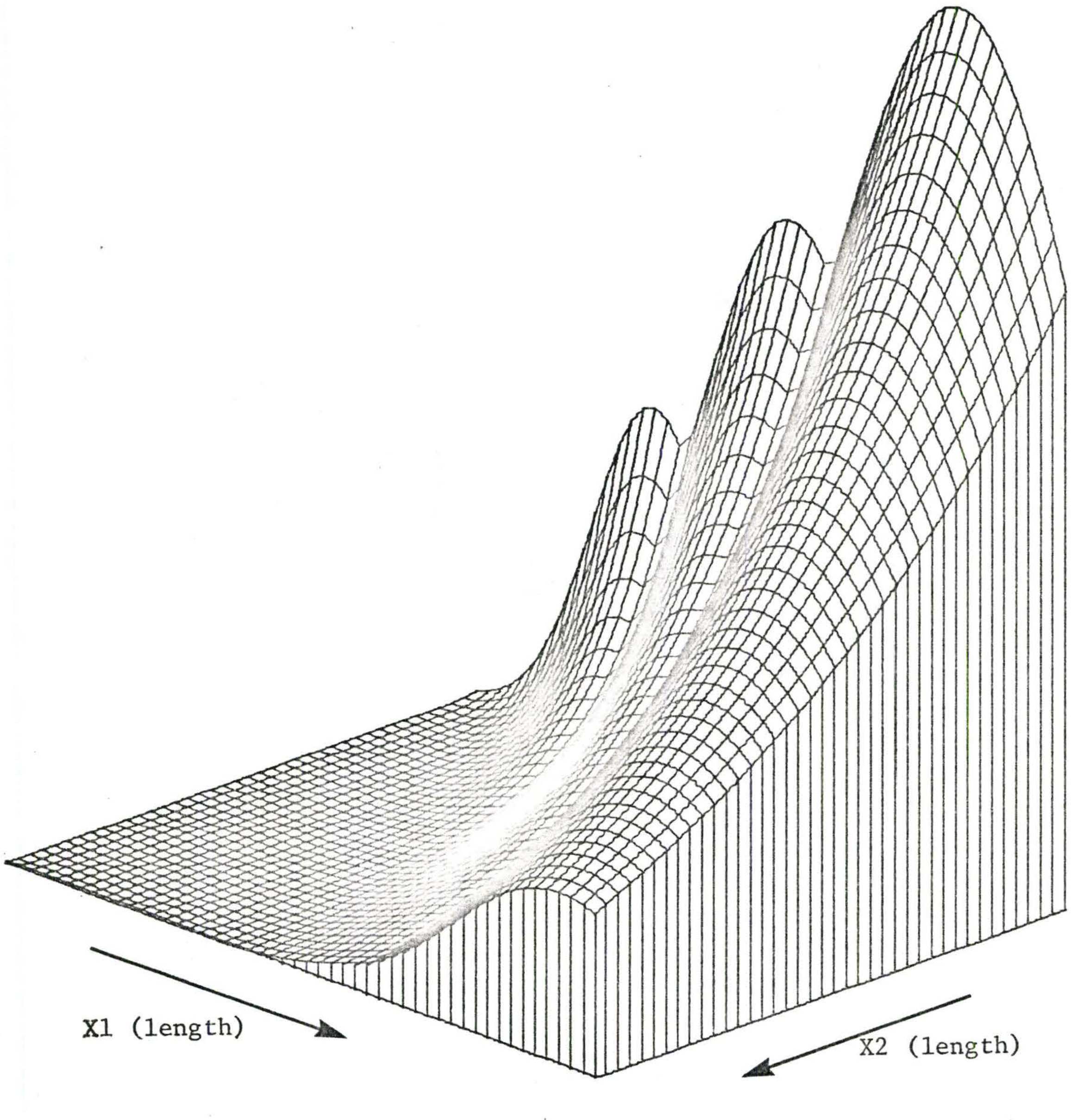


Figure 21.2

Flood Wave Propagation - Rotated View

dissipation of pollutants from two sources in a body of water. The X1 and X2 dimensions represent an area in the body of water and the altitude of the mapped solid represents the pollutant concentration. The manner in which they mix is of interest (reference 23).

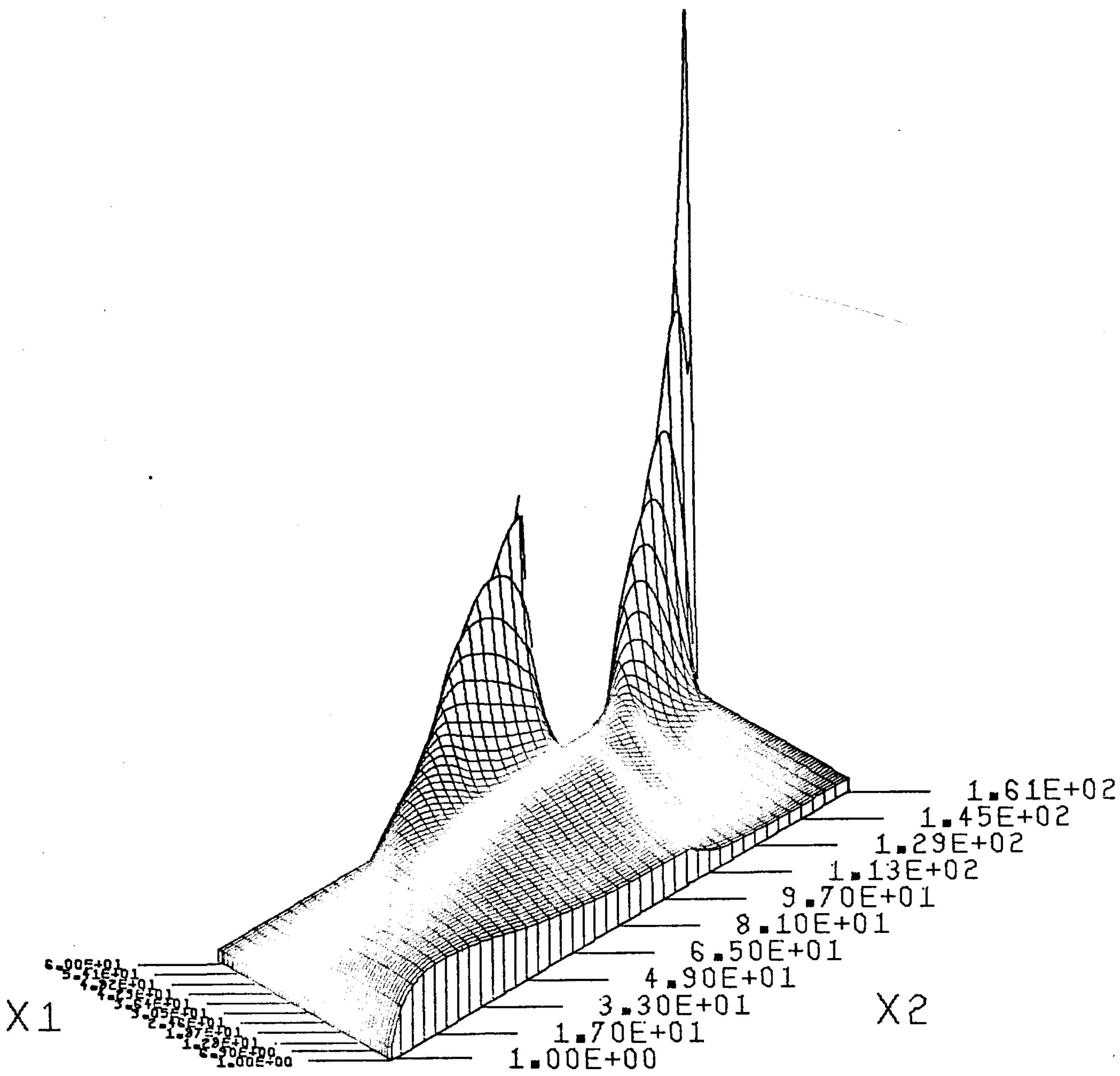


Figure 21.3

Concentration of Pollution from Two Sources - Basic View

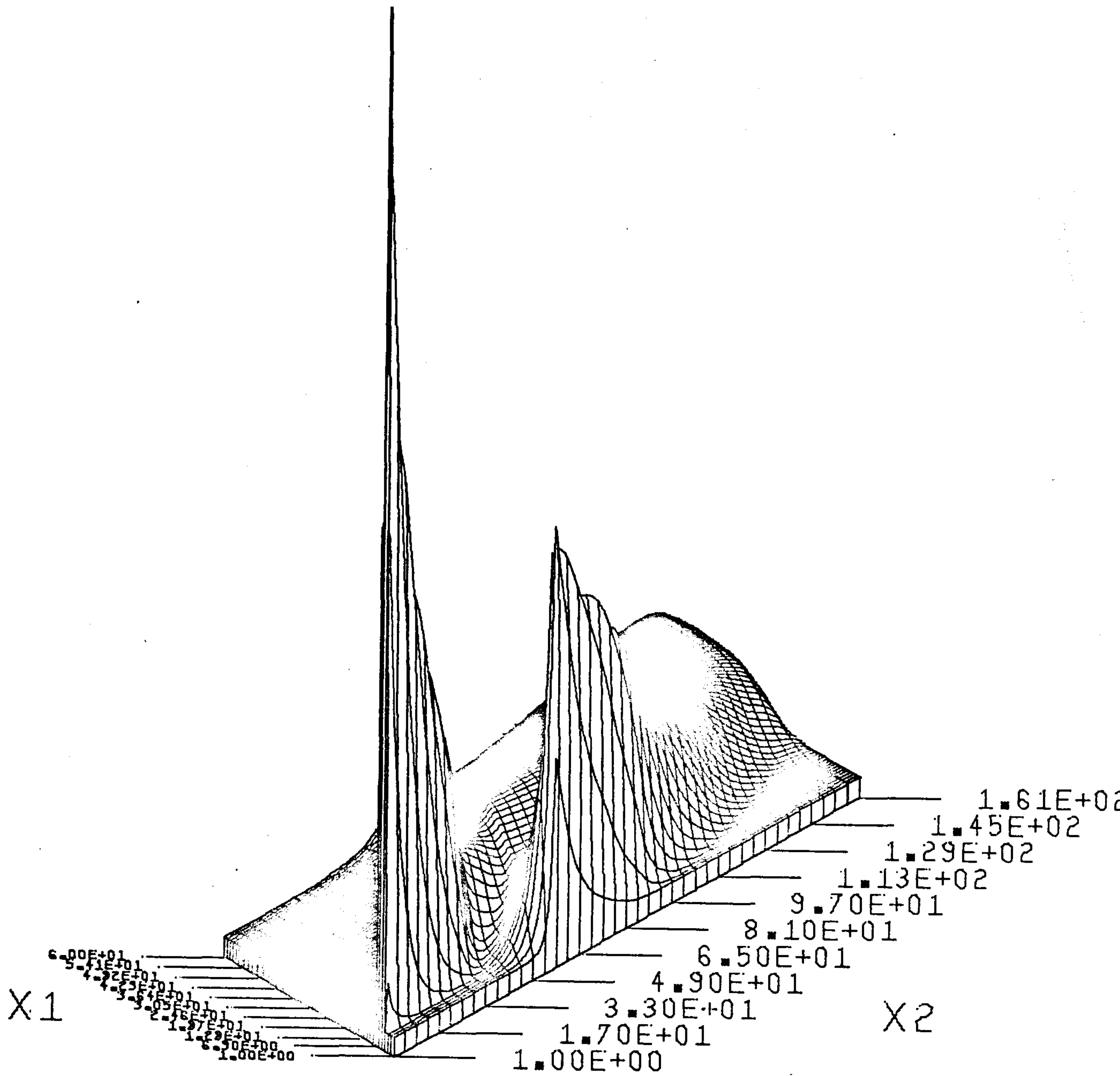


Figure 21.4

Concentration of Pollution from Two Sources - Rotated View

CHAPTER 22

TEACHING DESIGN OPTIMIZATION

The optimization feature of the package with its mapped display has interesting possibilities as a teaching tool.

The idea would be that students would try to search for maxima or minima in three dimensional engineering design optimization problems via computer experiments on a response surface fed into the computer by the instructor ahead of time. The student would be allowed several experiments and the results of each experiment should appear as printout of the numerical value of the objective function for each experiment. Furthermore, the experiment number could be stored at the appropriate location on the response surface for later display purposes.

When the students have computed their set of experiments, the surface, the optimum and the values of the objective functions corresponding to each experiment could be displayed (reference 24).

CHAPTER 23

IMPROVEMENT OF INCOMPLETE EXPERIMENTAL DATA

Often experiments are too costly or too time-consuming to determine sufficient data for a complete range of variables. Usually in such cases, only the minimum number of points are obtained experimentally and interpolation is necessary to complete a description of the phenomena.

Figure 23.1 shows a 3-D figure obtained on the plotter from an incomplete set of experimental data. In this case, the problem was to determine the shape and values of a surface describing the relationship between the nonstochiometry of cobaltwüstite (the vertical axis) with respect to the pressure (x2 axis) and weight percentage of ferrous oxide in the mixed oxide (x1 axis) at a given temperature (25).

Data was obtained for the nonstochiometry of cobaltwüstite at a temperature of 1000°C. Because of the high temperature, the experimental results were difficult to obtain. It was, therefore, decided to attempt to use the package to simultaneously double the number of results by using the spline interpolation routine built into the package and to map a three-dimensional display of the total data. Figure 23.1 shows a display of the three variables where every second value was obtained by interpolation (26).

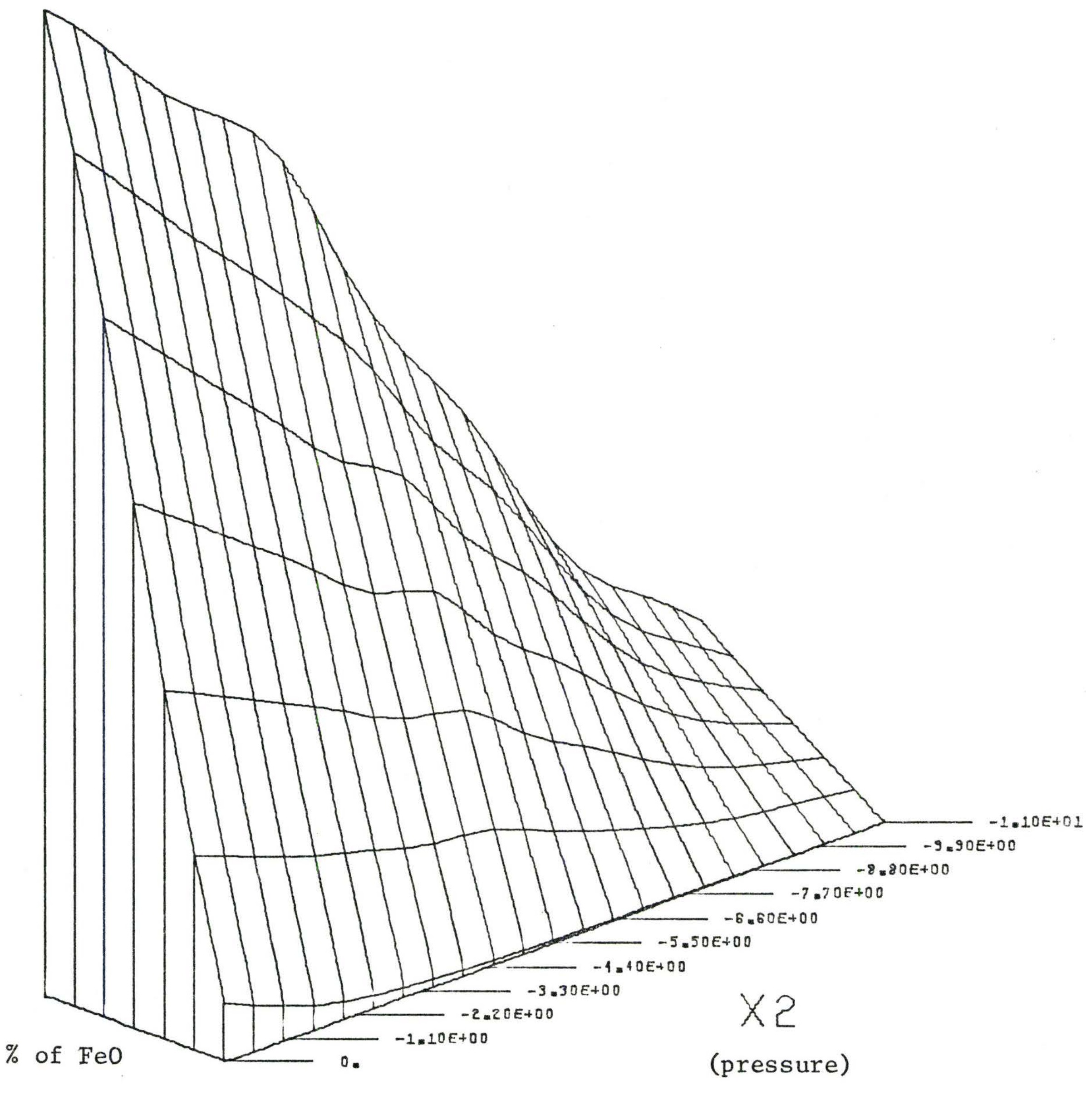


Figure 23.1

Interpolated Results from High Temperature Experiment

CHAPTER 24

VISUALIZATION OF DISCRETE EXPERIMENTAL DATA

A neutron beam was pointed at a detector and the flight time and energy of each neutron as it reflected off the detector and hit another detector was recorded (reference 27).

A plot was required to display the full spectrum of results. Therefore, the package described herein was used to plot the number of neutrons hitting the second detector (vertical axis), versus the pulse height representing the energy (x1 axis) versus the time of flight (x2 axis).

Figure 24.1 is a display of only a sample of the full range of the original data and the original plot. This figure demonstrates the performance of the package when applied to discrete "unsmoothed" data.

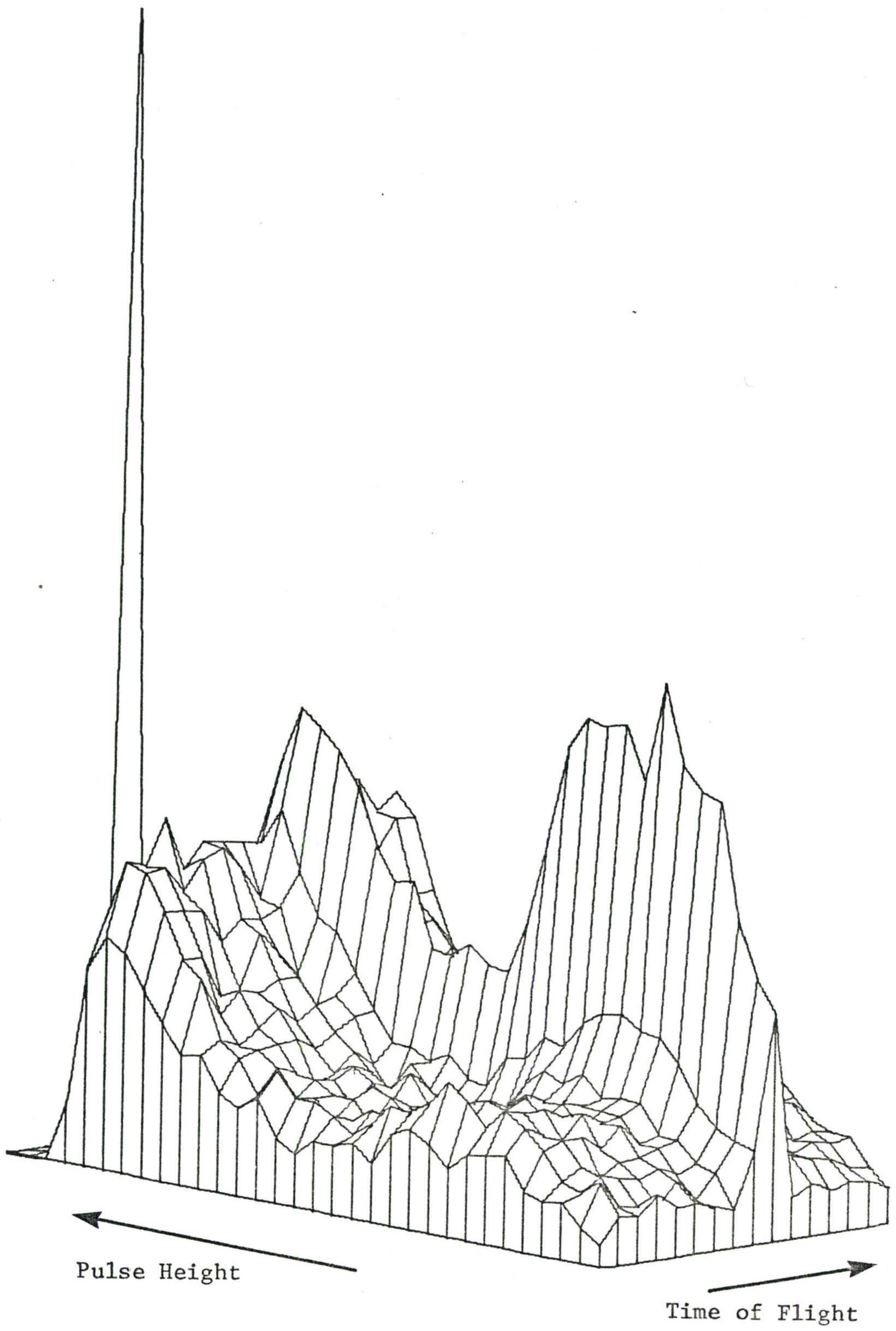


Figure 24.1

Display of Matrix With Eratic Data

REFERENCES

1. "Benson-Lehner Plotting System for CDC 6400", Data Processing & Computing Centre, Publication No. 5.2.1, November, 1971.

2. "Programming Calcomp Plotter", Calcomp manual, California Computer Products, Inc.

3. "Notes on Plotting Contours", Data Processing & Computing Centre, McMaster University, Hamilton, Publication 19.1B, June 12, 1969.

4. "Finite Element Method for the Stress Analysis of Saw Blades", M. Mahomed Master's Thesis, McMaster University, 1971.

5. "Graphics", J.T. Rule, S.A. Coons, McGraw-Hill Book Company, Inc., 1961.

6. "Graphic Science", T.E. French, Ch. J. Vierck, McGraw-Hill Book Company, Inc., 1963.

7. "Professional Perspective Drawing for Architects and Engineers", F.W. Capelle, McGraw-Hill Book Company, Inc., 1969.

8. "The Perspective Representation of Function of Two Variables", B. Kubert, J. Szabo, S. Giulieri, Journal of the Association for Computing Machinery, Vol. 15, No. 2, April 1968.

9. "The 'SYMVU' Program", User's Reference & Operator's Manuals, April 1971, F.J. Rens, Laboratory for Computer Graphics and Spatial Analysis, Harvard University, Cambridge, Version 1, 1970.

10. "Fundamentals of Acoustics", L.E. Kinsler, A.R. Frey, John Wiley & Sons, Inc., New York, 1967.

11. "Shock & Vibration Handbook", Vol. 2, C.M. Harris, C.E. Crede, McGraw-Hill Book Company, New York, 1961.

12. "Computing Centre Newsletter", Vol. 3, No. 8, Sept. 1971, McMaster University, Hamilton.

13. "Conserving Memory Space", Data Processing & Computing Centre, McMaster University, Hamilton, Publication 3.16, October, 1970.

14. "SCOPE 3.3 System Programmer's Reference Manual", Control Data 6000 Series Computer Systems, Control Data Corporation, Publication No. 60306800, 1971.

15. "SCOPE 3.3 Reference Manual", Control Data 6000 Series Computer Systems, Control Data Corp., Publication No. 60305200.

16. "FORTRAN Reference Manual", Control Data 6400/6500/6600 Computer Systems, Control Data Corporation, Publication No. 60174900, 1969.

17. "FORTRAN Extended Reference Manual", Control Data 6400/6500/6600 Computer Systems, Control Data Corporation, Publication No. 60176600, 1970.

18. "Use of Magnetic Tapes on the CDC 6400", Data Processing & Computing Centre, McMaster University, Hamilton, Publication No. 3.13.1, September, 1971.

19. "COMPASS Version 3 Reference Manual", Control Data CYBER 70 Series Computer Systems, 6000 Series Computer Systems, 7600 Computer System, Control Data Corporation, Publication No. 60360900, 1971.

20. "Analysis and Operations Management", E.H. Bowman, R.B. Fetter, Richard D. Irwin, Inc., 1967.

21. "Principles of Operations Research", H.M. Wagner, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1969.

22. "Tool Life Testing by Response Surface Methodology Coupled with a Random Strategy Approach", R. Vilencic, K. Ströbele, R. Venter, (To be published 1972).

23. Private contact with Dr. W. James, Civil Engineering, McMaster University, Hamilton.

24. Correspondence with Dr. D. R. Woods, Chemical Engineering, McMaster University, Hamilton.

25. "Thermodynamic Properties of Solid Solutions with Spinel-Type Structure: II. The System Co_3O_4-Fe_3O_4 at 1200°C", E. Aukrust, A. Muan, Transactions of the Metallurgical Society of Aime, Volume 230, October 1964.

26. "Oxidation of Cobalt-Iron Alloys in Various Oxygen Atmospheres at 1000°C", P. Mayer, (Ph.D. Thesis, to be submitted in 1972).

27. "Measurement of MeV-Range Total Cross-Sections using Reactor Neutrons", G.R.Norman, W.V. Prestwich, T.J. Kennett, Nuclear Inst. and Methods (in press 1972).

APPENDIX A

LIST OF STOPS, DIAGNOSTIC MESSAGES AND SUBROUTINES GENERATING THEM

If the program aborts, the following STOPn is printed in the dayfile and the associated message is printed on the printed output.

STOP 1 SUBROUTINE DATAT

"Number of intervals to be plotted along X1 or X2 axis is less than or equal to zero".

See Parameter INTX1.

STOP 2 SUBROUTINE DATAT

"Error in matrix dimensions. ISIZE or JSIZE are less than or equal to one".

See Parameter ISIZE.

STOP 3 SUBROUTINE DATAT

"Number of inequality constraints greater than twenty or less than zero".

See Parameter NCONS.

STOP 4 SUBROUTINE DATA

"Number of equality constraints greater than five or less than zero".

See Parameter NEQUA.

STOP 5 SUBROUTINE DATAT

"Klass is not equal 1, 2 or 3".

See Parameter KLASS.

STOP 6 SUBROUTINE DATA

"Angle of the matrix side is less than zero or greater than or equal to ninety".

See Parameter ALPHA.

STOP 7 SUBROUTINE DATA

"Size of the plot exceeds nnn inches or is less than or equal to zero".

See Parameter VSIZE.

STOP 10 SUBROUTINE DATA

"Length of the plot side is less than or equal to zero".

See Parameter XLENG.

<u>STOP 11</u> SUBROUTINE SURF99

"Total number of intervals (size of the matrix minus one is not an integer multiple of the number of the intervals to be plotted".

See Parameter INTX1.

<u>STOP 12</u> SUBROUTINE SURF99

"Upper limit of X1 or X2 is equal to the lower limit".

See Parameter X1MIN,X1MAX,X2MIN,X2MAX.

<u>STOP 13</u> SUBROUTINE SURF99

"Surface is not defined at specified interval. No feasible solution for optimization problem".

Although the surface exists at the interval specified by limits of X1 and X2, inequality constraints are violated at that particular interval. Therefore, the function is said to be "not defined" at the interval in question.

<u>STOP 14</u> SUBROUTINE SURF99

"Size of the plot is less than far corner of the basis".

See Parameter XLENG.

<u>STOP 15</u> SUBROUTINE DATAT

"Angle of view is less than zero or greater than 90 degrees".

See Parameter ANGLE.

STOP 16 SUBROUTINE DIFFER

"Memory allocated is not sufficient".

This message may not be issued for a modified subroutine DIFFER. If this subroutine does not supply the space available for the matrix, the program fails due to insufficient memory during the test done in Subroutine SURF99. (See Memory, page 96).

STOP 17 SUBROUTINE DATA

"Paper size is exceeding 28 inches or is less than zero".
See Parameter PAPER.

STOP 20 SUBROUTINE ROTOR

"Rotation angle is not an integer multiple of 90 degrees".
See Parameter XYROT

STOP 21 SUBROUTINE INPUT

"Syntax error in input data in or after column nnn".
See Input Data.

STOP 22 SUBROUTINE XINTER

"Ill conditions for interpolation".

This message is accompanied by an additional message specifying the cause of failure to interpolate and aprintout of the matrix at the instant of termination.

APPENDIX B

LIST OF THE PACKAGE SUBROUTINES AND FUNCTIONS

1. CHECK

Checks if criteria for interpolation to be possible are satisfied.

2. COORD

Calculates X and Y co-ordinates of a matrix element u_{ij}.

3. CORNER

Checks if the element which forces the pen to change its mode is an element standing on the edge of a constrained and non-constrained area.

4. CROSS

Finds X and Y co-ordinates of a point of intersection of two straight lines.

5. DATAT

Tests input parameters for specified limits.

6. DIAG3

Solves a tridiagonal system of equations.

7. DIFFER

Contains the instructions that may be different for different computers or different FORTRAN compilers.

8. ELEV

Lifts the pen from the base to the surface or slides it to the base from the surface if the edge of the surface due to the constrained area has been reached.

9. EYE

Finds the proper mode of the pen (2 or 3) for the next movement. It also moves the pen in its present mode up to the point of intersection if the pen changes its mode. (See Figure 6.3).

10. FIZL

FIZL checks the conditions for the next move of the pen, namely, whether the move is to be on the surface or on the base (for constrained areas), positions the pen for any surface - base interchanges, obtains the mode of the move and moves the pen to the next position (see Figure 6.1). It also marks the outer corner elements in order to prevent multiple drawing of their vertical contours.

11. GETMAT

Reads or generates the matrix for plotting, depending on the parameter KLASS.

12. IGET

Finds the position of the next nonblank character for its identification by subroutine INPUT.

13. <u>INCO</u>

Checks if element lies on the inner corner of the boundaries of constrained areas and if so, arranges for the plotting of both the vertical contour and the contours at the base.

14. <u>INITP</u>

Initiates the plot file by issuing the first call to subroutine PLOT.

15. <u>INPUT</u>

Reads the input data, decodes the key words and sets appropriate switches and/or parameters.

16. <u>INSERT</u>

Interpolates column-wise or row-wise.

17. <u>LINES</u>

Identifies the intersecting lines by two pairs of co-ordinates of the points lying on each one.

18. <u>MARK</u>

Controls the writing of all the text and symbols connected with the problem on the plot.

19. <u>MAXIM</u>

Locates the maxima and minima of the values in the generated matrix.

20. MOVE

Checks the mode of the pen and, while generating the drawing controls, movements of the pen so that the pen is moved only if it is supposed to draw a line.

21. NKVD

Searches the matrix to determine if any elements have been eliminated due to the constraining conditions and clears out elements that do not have any neighbouring ones in order to get a clear drawing.

22. PLOT3D

This is the driving routine for the plotting package. Specifies working space in blank corner, calls subroutine INPUT for input of parameters and passes control to the plotting routine (see Figure 14.2).

23. PATHS

Marks the paths of equality constraints on the drawing and marks the elements on the paths for further printer-plot.

24. PRINTG

Generates a printer-plot representing the top view of the plotted matrix.

25. PRINTL

Prints list of parameters, their values and meanings and list of activated switches.

26. READV

Reads the matrix column-wise from the logical unit #2 onto which it has been written by subroutine ROTOR, thus rotates the matrix clock-wise.

27. ROTOR

Checks validity of the angle of rotation, switches limits of variables and writes the matrix row-wise onto logical unit #2.

28. SPACE

This COMPASS subroutine searches the job communication area (Figure 14.1) for total memory allocated to the program and the first address of the blank COMMON. These values are analyzed by subroutine DIFFER to check if space in blank COMMON can accommodate the matrix with the working arrays, if an interpolation is necessary.

29. SPLINE

Executes cubic spline interpolation.

30. SPLIT

Calculates second derivatives for interpolation.

31. <u>SURF99</u>

This is the backbone of the program (see Figure 3.1). It controls and performs calculations and other tasks necessary to generate the matrix and prepare it for plotting, initiates and terminates the plotting, controls the generating of the drawing and servicing tasks such as printing of the report, outputting the matrix, etc.

32. <u>UOUT</u>

Outputs the matrix on the printer and/or binary punch with respect to the setting of appropriate switches.

33. <u>USO</u>

This function retrieves the actual value (altitude) to be used for plotting since some elements have been artificially modified-flagged by subroutine FIZL.

34. <u>XINTER</u>

Arranges for the completion of the matrix by the method of spline interpolation if the matrix has not been supplied fully.

LIST OF LIBRARY SUBROUTINES

1. DATE

FORTRAN function supplying the current date as a 10 character Hollerith constant.

2. GRAF

Draws symbols Δ and ∇ to mark maximum and minimum on the plot respectively.

3. LETTER

Draws string of characters used as comment on the plot.

4. PLOT

Moves the pen above or on the paper with respect to the mode specified.

APPENDIX C

CROSS REFERENCE DIRECTORY OF THE PACKAGE SUBROUTINES

NAME	CALLING	CALLED BY
1 CHECK		XINTER
2 COORD ENTRY XCO ENTRY YCO		ELEV, FIZL, INCO, LINES, PATHS, SURF99
3 CORNER	U50	EYE
4 CROSS		EYE
5 DATAT		SURF99
6 DIAG3		SPLIT
7 DIFFER	SPACE	INPUT, MARK, PRINTG, READU, ROTOR, UOUT, SURF99
8 ELEV	COORD, EYE, MOVE U50	FIZL, INCO
9 EYE	CORNER, CROSS, LINES, MOVE, U50	ELEV, FIZL, INCO

NAME	CALLING	CALLED BY
10 FIZL	COORD,ELEV,EYE, MOVE	SURF99
11 GETMAT	OBJECT,UFILL,XINTER	SURF99
12 IGET		INPUT
13 INCO	COORD,ELEV,EYE, MOVE,PLOT	SURF99
14 INITP	PLOT	SURF99
15 INPUT	DIFFER,IGET	PLOT3D
16 INSERT	SPLINE,SPLIT	XINTER
17 LINES	COORD,U50	EYE
18 MARK	DIFFER,LETTER, PATHS, PLOT	SURF99
19 MAXIM		SURF99
20 MOVE	PLOT	ELEV,EYE,FIZL,INCO,SURF99,
21 NKVD		SURF99
22 PATHS	COORD,EQUAL,GRAF, LETTER,OBJECT,U50	MARK,SURF99

NAME	CALLING	CALLED BY
23 PLOT3D	INPUT,SURF99	MAIN
24 PRINTG	DIFFER	SURF99
25 PRINTL		SURF99
26 READU	DIFFER	ROTOR
27 ROTOR	DIFFER,READU	SURF99
28 SPACE		DIFFER
29 SPLINE		INSERT
30 SPLIT	DIAG3	INSERT
31 SURF99	CONST,COORD,DATAT, DATE,DIFFER,FIZL, GETMAT,INCO,INITP, LETTER,MARK,MAXIM, MOVE,NKVD,PATHS, PLOT,PRINTG,PRINTL, ROTOR,UOUT	PLOT3D
32 UOUT	DIFFER	SURF99
33 U50		ELEV,EYE,LINES,PATHS
34 XINTER	CHECK,INSERT	GETMAT

REFERENCE DIRECTORY OF SERVICE SUBROUTINES

NAME	CALLING	CALLED BY
1 CONST		SURF99
2 EQUAL		PATHS
3 OBJECT		GETMAT,PATHS
4 UFILL		GETMAT

APPENDIX D

TYPICAL DECK STRUCTURES

1) Package in Source Form

```
abcd,MT1,T40,CM52000.                      name

LOADER(PPLOADR)

RUN(S,,1011)

SETINDF.

REDUCE.

LGO.

RETURN(TAPE10)

7/8/9       END OF RECORD

      PROGRAM MAIN(INPUT,OUTPUT,PUNCHB,TAPE1,

    * TAPE2,TAPE5=INPUT,TAPE6=OUTPUT,TAPE8=PUNCHB,

    * TAPE10)


               user's service subroutines


               package subroutines


7/8/9       END OF RECORD

               data

6/7/8/9     END OF FILE
```

2) <u>Package in Object Form as a Deck</u>

```
abcd,MT1,CM52000.                                   name

LOADER(PPLOADR)

COPYBF(INPUT,LGO)

RUN(S,,1011)

SETINDF.

REDUCE.

LGO.

RETURN(TAPE10)

7/8/9      END OF RECORD


                object deck with package subroutines


7/8/9      END OF RECORD

      PROGRAM MAIN(INPUT,................)


                user's service subroutines


7/8/9      END OF RECORD


                 data


6/7/8/9     END OF FILE
```

3) Package in Object Form on Magnetic Tape

```
abcd,MT1,CM52000.                                  name

LOADER(PPLOADR)

RUN(S,,1011)

REQUEST(TP)READ/tape identification

REWIND(TP)

COPYBF(TP,XX)

UNLOAD(TP)

LOAD(XX)

LGO.

7/8/9    END OF RECORD

     PROGRAM MAIN(INPUT,.........)


            user's service subroutines


7/8/9    END OF RECORD


            data


6/7/8/9    END OF FILE
```

4) Package in Object Form on Permanent File

```
abcd,MT1,CM52000.                                  name

LOADER(PPLOADR)

RUN(S,,1011)

ATTACH(XX,filename,ID=abcdname,MR=1)

LOAD(XX)

LGO.

7/8/9    END OF RECORD

       PROGRAM MAIN (INPUT,..........)


               user's service subroutines


7/8/9    END OF RECORD


               data


6/7/8/9    END OF FILE
```

5) Input Matrix as a Binary Deck

```
abcd,MT1,CM52000.                                 name

LOADER(PPLOADR)

RUN(S,,1011)

COPYBF(INPUT,TAPE1)

REWIND(TAPE1)

ATTACH(XX,..........)

LOAD(XX)

LGO.

7/8/9    END OF RECORD

       PROGRAM MAIN(TAPE1,INPUT,.....)


               user's service subroutines


7/8/9    END OF RECORD


               binary deck with the matrix


7/8/9    END OF RECORD


               data


6/7/8/9   END OF FILE
```

APPENDIX E

LISTING OF THE 3-D PACKAGE

```
      SUBROUTINE PLOT3D
C
C     SPECIFIES LARGE WORKING SPACE IN BLANK COMMON AND
C     CALLS FOR INPUT DATA AND PASSES CONTROL TO 3-D PLOTTING PACKAGE
C     *** USER IS BY NO MEANS ALLOWED TO PASS VARIABLES AMONG HIS
C     ROUTINES VIA BLANK COMMON ***
C
      COMMON WORK(30000)
C
      CALL INPUT (ISIZE,JSIZE)
C
      CALL SURF99 (WORK,ISIZE,JSIZE)
C
      RETURN
      END
      SUBROUTINE SURF99 (U,NUI,NUJ)
C
C     SUPERVISES BOTH THE PREPARATION OF THE MATRIX FOR THE
C     PLOTTING AND THE PLOTTING OF THE SURFACE
C
      DIMENSION U(NUI,N
C
      COMMON /DATA/ INTX1,INTX2,ISIZE,JSIZE,NCONS,IQUAL,KLASS,ALPHA,ANGL
     1E,PAPER,VSIZE,XLENG,XYROT,X1MAX,X1MIN,X2MAX,X2MIN
      COMMON /A/ PHI(20),FSI(5),KODE(5)
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,NRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
      COMMON /SWITCH/ KEYS(5,10),NKEYS,NSW(10)
C
      DATA XLETI/1HI/
C
C
C
C     TEST INPUT DATA
C
      CALL DATAT
C
C     CALL SUBROUTINE TO CALCULATE ALPHA, VSIZE AND XLENG FOR GIVEN
C     ANGLE OF VIEW AND TO PRINT A LIST OF PARAMETERS
C
      CALL PRINTL (V2)
C
C     TEST FOR AVAILABLE MEMORY SPACE
C     THIS TEST MAY DIFFER FOR DIFFERENT MACHINES. THE SIMPLEST WAY OF
C     TESTING IS USING THE VERY LAST LOCATION OF MATRIX U FOR STORING
C     OF A DUMMY QUANTITY. FOR KLASS=3 IT WILL BE THE LAST LOCATION OF
C     WORKING ARRAYS.
C     POSSIBLE FORM OF TEST IS   U(LAST)=0.0
C
      WRITE (6,61)
      IF (KLASS-3) 2,1,2
    1 K=NUI
      IF (NUJ.GT.NUI) K=NUJ
      K1=NUI*NUJ+7*K+1
      GO TO 3
    2 K1=NUI*NUJ
    3 WRITE (6,63) K1
C
      CALL DIFFER (6,K1,A,IC,NUI,U,NUJ)
C
      U(K1)=0.0
      WRITE (6,62)
```

```
C
C     PRESET U-MATRIX TO I
C
      DO 4  I=1,NUI
      DO 4  J=1,NUJ
      U(I,J)=XLETI
4     CONTINUE
C
      NI=NUI-1
      NJ=NUJ-1
      ISTEP=NI/INTX1
      JSTEP=NJ/INTX2
C
C     CHECK IF NUMBER OF INTERVALS TO BE PLOTTED IS AN INTEGER MULTIPLE
C     OF THE SIZE OF THE MATRIX
C
      IF ((ISTEP*INTX1).EQ.NI.AND.(JSTEP*INTX2).EQ.NJ) GO TO 5
      WRITE (6,65)
      STOP  11
5     IF (KLASS.NE.1.AND.NCONS.EQ.0.AND.IQUAL.EQ.0) GO TO 7
      DIF1=X1MAX-X1MIN
      DIF2=X2MAX-X2MIN
C
C     CHECK IF UPPER LIMIT OF X1 OR X2 IS DIFFERENT FROM THE LOWER
C     LIMIT
C
      IF (DIF1.NE.0.0.AND.DIF2.NE.0.0)  GO TO 6
      WRITE  (6,67)
      STOP  12
6     DEL1=DIF1/FLOAT(NI)
      DEL2=DIF2/FLOAT(NJ)
C
C     GET U-MATRIX FOR FUTURE PLOTTING
C
7     CALL  GETMAT (U,NUI,NUJ)
C
C     SET U-ELEMENTS TO NEGATIVE IF ANY INEQUALITY CONSTRAINT
C     AT  I-J  LOCATION IS VIOLATED
C
      IF (NCONS) 11,11,8
8     NVIOL=0
      DO 10 I=1,NUI
      XI=X1MIN+DEL1*FLOAT(I-1)
      DO 10 J=1,NUJ
      XJ=X2MIN+DEL2*FLOAT(J-1)
C
      CALL  CONST  (XI,XJ)
C
      DO 10 K=1,NCONS
      IF (PHI(K)) 9,9,10
```

```
9     IF (U(I,J).EQ.XLETI) GO TO 10
      U(I,J)=XLETI
      NVIOL=NVIOL+1
10    CONTINUE
C
C     CHECK IF THE SURFACE IS DEFINED AT THE GIVEN INTERVAL
C
      IF (NVIOL.LT.(NUI*NUJ)) GO TO 11
      WRITE (6,68)
      STOP 13
C
C     CLEAR THE MATRIX LIQUIDATING ANY SINGLE VALUE OF U
C
11    IF (NCONS.LE.0) GO TO 12
C
      CALL NKVD (U,NUI,NUJ)
C
C     ROTATE THE MATRIX IF NECESSARY
C
12    IF (XYROT) 13,14,13
C
C
C     STORE BOUNDS OF X1 AND X2
C
13    PHI(1)=X1MIN
      PHI(2)=X1MAX
      PHI(3)=X2MIN
      PHI(4)=X2MAX
C
      CALL ROTOR (U,NUI,NUJ)
14    CONTINUE
C
C     WRITE THE ORIGINAL MATRIX I.E. IN DATA FORM IN BINARY AS
C     THE FIRST FILE ON UNIT 2
C     THE MATRIX IS NOW ROTATED IF XYROT WAS DIFFERENT FROM ZERO
C
      REWIND 2
      WRITE (2) U
      END FILE 2
C
C     FIND LOCATIONS AND VALUES OF MINIMUM AND MAXIMUM U
C
      CALL MAXIM (U,NUI,NUJ)
C
      WRITE (6,58) UMIN,IMIN,JMIN
      IF (KLASS.NE.1.AND.NCONS.LE.0) GO TO 15
      XI=X1MIN+DEL1*FLOAT(IMIN-1)
      XJ=X2MIN+DEL2*FLOAT(JMIN-1)
      WRITE (6,59) XI,XJ
      GO TO 16
```

```
15      WRITE (6,60)
16      WRITE (6,58) UMAX,IMAX,JMAX
        IF (KLASS.NE.1.AND.NCONS.LE.0) GO TO 17
        XI=X1MIN+DEL1*FLOAT(IMAX-1)
        XJ=X2MIN+DEL2*FLOAT(JMAX-1)
        WRITE (6,59) XI,XJ
        GO TO 18
17      WRITE (6,60)
18      CONTINUE
C
C       CALCULATE DIMENSIONS AND INCREMENTS FOR PLOTTING
C
        DYL=XLENG*SIN(ALPHA*4.0*ATAN(1.0)/180.0)
        DXL=XLENG*COS(ALPHA*4.0*ATAN(1.0)/180.0)
        IF (NI.GT.NJ) GO TO 19
        DX=DXL/FLOAT(NJ)
        DY=DYL/FLOAT(NJ)
        GO TO 20
19      DX=DXL/FLOAT(NI)
        DY=DYL/FLOAT(NI)
C
C       CHECK IF THE SIZE OF THE PLOT IS GREATER THAN THE OPOSIT CORNER
C       OF THE BASIS
C
20      IF (YCO(NUI,NUJ).LE.VSIZE) GO TO 21
        WRITE (6,66)
        STOP 14
C
C       CALCULATE VERTICAL SCALE
C
21      IF (UMAX.EQ.UMIN) GO TO 22
        IF (UMIN) 23,22,22
22      UCOR=0
        GO TO 24
23      UCOR=UMIN
C
C        SET INITIAL VALUE OF VS
C
24      DO 25 I=1,NUI
        DO 25 J=1,NUJ
        IF (ABS(U(I,J)-UCOR).GT.1.0E-6) GO TO 26
25      CONTINUE
        WRITE (6,64)
        VS=0.0
        GO TO 30
C
C       CALCULATE VERTICAL SCALE FOR SPECIFIED ANGLE OF VIEW
C
26      IF (ANGLE.EQ.XLET1) GO TO 27
        VS=V2/(UMAX-UCOR)
```

```
      GO TO 30
C
27    VS=VSIZE/(U(I,J)-UCOR)
      DO 29 I=1,NUI
      DO 29 J=1,NUJ
      IF (U(I,J)-XLETI) 28,29,28
28    IF (ABS(U(I,J)-UCOR).LT.1.0E-6) GO TO 29
      VSS=(VSIZE-YCO(I,J))/(U(I,J)-UCOR)
      IF (VSS.LT.VS) VS=VSS
29    CONTINUE
C
C      CALCULATE U-NETWORK
C
30    CONTINUE
      DO 33 I=1,NUI
      DO 33 J=1,NUJ
      IF (U(I,J)-XLETI) 32,31,32
31    U(I,J)=-1.0
      GO TO 33
32    U(I,J)=(U(I,J)-UCOR)*VS+YCO(I,J)
33    CONTINUE
C
C     WRITE U-MATRIX SCALED FOR PLOTTING I.E. CONVERTED TO PLOTER
C     UNITS IN BINARY AS THE SECOND FILE ON UNIT 2
C
      WRITE (2) U
      END FILE 2
      REWIND 2
C
C     BYPASS PLOTTING IF NOPLOT SWITCH IS SET
C
      IF (NSW(3).NE.0) GO TO 55
C
C     PLOTTING STARTS HERE
C
C     WRITE CODE AND DATE
C
      CALL DATE (DD)
      CALL INITP (0.0)
      CALL PLOT (0.0,0.2,-3)
      XCH=1.5
      CALL PLOT (XCH,10.0,3)
      CALL PLOT (XCH+0.2,10.0,2)
      DO 34 I=1,5
      YKODE=10.0-FLOAT(I-1)*4.0*0.3
      CALL LETTER (4,0.3,270.0,1.0,YKODE,KODE(I))
34    CONTINUE
      CALL LETTER (10,0.3,270.0,1.0,4.0,DD)
      CALL PLOT (XCH+0.2,0.0,3)
      CALL PLOT (XCH,0.0,2)
      X1L=FLOAT(NUI-1)*DX
      XST=X1L+8.0
      CALL  PLCT  (XST,0.0,-3)
C
C     DRAW  LEFT-RIGHT  PATHS
C
      IK=0
      JK=1
      DO 37  I=1,NUI,ISTEP
      NUP=0
      IPEN=2
      CALL  MOVE  (XCO(I,1),YCO(I,1),3)
      DO 35  J=2,NUJ
      CALL  FIZL  (I,J,U,NUI,NUJ)
      NX1=1
      CALL  INCO  (I,J,U,NUI,NUJ)
35    CONTINUE
      IF  (I.EQ.1)  GO  TO  37
      IF  (U(I-1,NUJ))  36,37,37
36    NX1=2
      CALL  INCO  (I,NUJ+1,U,NUI,NUJ)
37    CONTINUE
      CALL  PLOT  (0.0,0.0,-3)
C
C     DRAW  RIGHT-LEFT  PATHS
C
      IK=1
      JK=0
      DO 40  J=1,NUJ,JSTEP
      NUP=0
      IPEN=2
      CALL  MOVE  (XCO(1,J),YCO(1,J),3)
      DO 38  I=2,NUI
      CALL  FIZL  (I,J,U,NUI,NUJ)
38    CONTINUE
      IF  (J.EQ.1)  GO  TO  40
      IF  (U(NUI,J-1))  39,40,40
39    NX1=3
      CALL  INCO  (NUI+1,J,U,NUI,NUJ)
40    CONTINUE
C
C     DRAW  BASE
C
C     RIGHT-LEFT
C
      IPEN=2
      CALL  PLOT  (0.0,0.0,-3)
      I=NUI
      J=1
      CALL  MOVE  (XCO(I,J),YCO(I,J),3)
```

```
41      I=I-1
        IF (U(I,J)) 42,43,43
42      IPEN=3
        GO TO 45
43      IF (U(I+1,J)) 42,44,44
44      IPEN=2
45      CALL MOVE (XCO(I,J),YCO(I,J),IPEN)
        IF (I-1) 46,46,41
C
C       LEFT-RIGHT
C
46      J=J+1
        IF (U(I,J)) 47,48,48
47      IPEN=3
        GO TO 50
48      IF (U(I,J-1)) 47,49,49
49      IPEN=2
50      CALL MOVE (XCO(I,J),YCO(I,J),IPEN)
        IF (J.LT.NUJ) GO TO 46
C
C       DRAW PATHS OF EQUALITY CONSTRAINTS
C
        IF (IQUAL) 52,52,51
C
51      CALL PATHS (U,NUI,NUJ)
C
C        MARK AXIS
C
52      IF (NSW(4)) 54,53,54
C
53      CALL MARK (U,NUI,NUJ)
C
C       FINISH UP PLOTTING
C
54      XCH=-X1L-6.3
        CALL PLOT (XCH,10.0,3)
        CALL PLOT (XCH+0.2,10.0,2)
        CALL PLOT (XCH+0.2,0.0,3)
        CALL PLOT (XCH,0.0,2)
        XEND=FLOAT(NUJ-1)*DX+4.0
        IF ((XEND+FLOAT(NUI-1)*DX).LT.20.0) XEND=20.0-FLOAT(NUI-1)*DX
        CALL LETTER (11,0.2,270.0,XEND,6.0,11HEND OF PLOT)
        CALL PLOT (XEND+17.0,0.0,-3)
        CALL PLOT (0.0,0.0,999)
        WRITE (6,57)
C
C       PRINT PLAIN VIEW OF THE SURFACE IF ANY CONSTRAINTS WERE PRESENT
C
        IF (NCONS.LE.0.AND.IQUAL.LE.0) GO TO 55
C
```

```
      CALL PRINTG (U,NU1,NUJ)
C
C     OUTPUT U-MATRIX IF THE CORRESPONDING SWITCHES ARE SET
C
55    IF (NSW(1).NE.0.OR.NSW(2).NE.0) CALL UOUT (U,NU1,NUJ)
C
      RETURN
C
56    FORMAT (//5X,14HMINIMUM VALUE ,E13.5,14H FOUND AT I = ,I3,6H, J =
     1,I3)
57    FORMAT (////5X,26H*** PLOTTING COMPLETED ***)
58    FORMAT (//5X,14HMAXIMUM VALUE ,E13.5,14H FOUND AT I = ,I3,6H, J =
     1,I3)
59    FORMAT (1H+,60X,8HFOR X1 =,E14.5,9H AND X2 =,E14.5)
60    FORMAT (1X)
61    FORMAT (//5X,32HTEST FOR SUFFICIENT MEMORY SPACE)
62    FORMAT (/5X,11HTEST PASSED)
63    FORMAT (8X,15HSPACE NEEDED IS,I10,10H LOCATIONS)
64    FORMAT (//5X,43HBOTH MINIMUM AND MAXIMUM ARE EQUAL TO ZERO /)
65    FORMAT (//3X,81H*****TOTAL NUMBER OF INTERVALS (SIZE OF THE MATRIX
     1 MINUS ONE) IS NOT AN INTEGER  /4X,51H MULTIPLE OF THE NUMBER OF I
     2NTERVALS TO BE PLOTTED /)
66    FORMAT (//3X,59H*****SIZE OF THE PLOT IS LESS THAN FAR CORNER OF T
     1HE BASIS /)
67    FORMAT (//3X,57H*****UPPER LIMIT OF X1 OR X2 IS EQUAL TO THE LOWER
     1 LIMIT /)
68    FORMAT (//3X,50H*****SURFACE IS NOT DEFINED AT SPECIFIED INTERVAL
     1/5X,46HNO FEASIBLE SOLUTION FOR OPTIMIZATION PROBLEM /)
      END
      FUNCTION COORD (I,J)
C
C     CALCULATES X AND Y COORDINATES OF THE MATRIX ELEMENT
C
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IARTS,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,J1,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,MRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
      ENTRY XCO
      COORD=DX*FLOAT(J-1)
      RETURN
C
      ENTRY YCO
      COORD=DY*FLOAT(J+I-2)
      RETURN
C
      END
      SUBROUTINE CORNER (U,NUI,NUJ)
C
C     CHECKS IF CONTOUR IS A CORNER LINE
C
      DIMENSION U(NUI,NUJ)
C
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,MRENCM,UP(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
      NCORN=0
      IF (NINCO) 4,1,4
1     IF (JK) 3,2,3
2     IF (I23) 5,9,5
3     IF (I23) 9,5,9
4     IF (INCOJK) 2,3,2
C
C     +-
C     --
C
5     IF (U(IT,JT+1)) 6,44,44
6     IF (IT-1) 7,43,7
C
7     IF (U(IT-1,JT)) 8,44,44
8     IF (U(IT-1,JT+1)) 13,44,44
C
C     --
C     -+
C
9     IF (U(IT+1,JT)) 10,44,44
10    IF (JT-1) 11,43,11
C
11    IF (U(IT,JT-1)) 12,44,44
12    IF (U(IT+1,JT-1)) 16,44,44
C
13    IF (I23) 43,14,43
14    ISW=1
      GO TO 18
C
15    IR=IT-1
      JR=JT
      ICK=1
      JCK=0
      ISW=1
      GO TO 32
C
16    IF (I23) 43,17,43
C
```

```
17    ISW=2
18    IR=IT
      JR=JT
19    IR=IR-1
      JR=JR-1
      IF (IR.LT.1.OR.JR.LT.1) GO TO 30
      IF (U(IR,JR)) 19,20,20
20    IF (U50(IR,JR,U,NUI,NUJ).GT.ULAST) GO TO 21
      GO TO 19
C
21    GO TO (22,26), ISW
C
22    IF (U(IR,JR+1)) 23,44,44
23    IF (IR-1) 19,19,24
24    IF (U(IR-1,JR)) 25,44,44
25    IF (U(IR-1,JR+1)) 19,44,44
C
26    IF (U(IR+1,JR)) 27,44,44
27    IF (JR-1) 19,19,28
28    IF (U(IR,JR-1)) 29,44,44
29    IF (U(IR+1,JR-1)) 19,44,44
C
30    GO TO (15,31), ISW
C
31    IR=IT
      JR=JT-1
      ICK=0
      JCK=1
      ISW=2
C
32    IF ((IR+ICK).EQ.1.OR.(JR+JCK).EQ.1) GO TO 43
      IR=IR-1
      JR=JR-1
      IF (U(IR,JR)) 32,33,33
C
33    II=IR+ICK
      JJ=JR+JCK
      GO TO (34,38), ISW
C
C     +-
C     --
C
34    IF (U(II,JJ+1)) 35,42,42
35    IF (II-1) 32,32,36
C
36    IF (U(II-1,JJ)) 37,42,42
37    IF (U(II-1,JJ+1)) 32,42,42
C
C     --
C     -+
C
38      IF (U(II+1,JJ)) 39,42,42
39      IF (JJ-1) 32,32,40
C
40      IF (U(II,JJ-1)) 41,42,42
41      IF (U(II+1,JJ-1)) 32,42,42
C
42      CONTINUE
        IF (U50(II,JJ,U,NUI,NUJ).LT.ULAST) GO TO 32
        RETURN
C
43      NCORN=1
44      RETURN
C
        END
```

```
      SUBROUTINE CROSS
C
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(30),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,NRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
C     SOLVES INTERSECTING POINT OF TWO SRAIGHT LINES
C      X1,Y1          POINT AT THE FIRST LINE
C      X2,Y2          POINT AT THE FIRST LINE
C      X1,YY1         POINT AT THE SECOND LINE
C      X2,YY2         POINT AT THE SECOND LINE
C      XN,YN          POINT OF INTERSECTION
C
      SIGN(X,X1,X2,Y,Y1,Y2)=(X2-X1)*(Y-Y1)-(Y2-Y1)*(X-X1)
C
C     CHECK IF LINES ARE INTERSECTING WITHIN THE INERVAL X1 - X2
C
      S1=SIGN(X1,X1,X2,YY1,Y1,Y2)
      ST=SIGN(X2,X1,X2,YY2,Y1,Y2)
      IF (S1*ST) 1,1,2
1     A=YY2-YY1
      B=Y2-Y1
      IF (A.NE.B) GO TO 3
2     INT=0
      RETURN
C
3     C=X2-X1
      XN=X1+(YY1-Y1)*C/(B-A)
      YN=Y1+B*(XN-X1)/C
      INT=1
C
      RETURN
      END
      SUBROUTINE DATAT
C
C     PERFORMES TEST ON INPUT DATA TO CHECK IF THEY ARE IN
C     ALLOWABLE LIMITS
C
      COMMON /DATA/ INTX1,INTX2,ISIZE,JSIZE,NCONS,IQUAL,KLASS,ALPHA,ANGL
     1E,PAPER,VSIZE,XLENG,XYROT,X1MAX,X1MIN,X2MAX,X2MIN
C
      DATA XLETI/1HI/
C
C
C     CHECK IF NUMBER OF SPECIFIED INTERVALS IS POSITIVE
C
      IF (INTX1.GT.0.AND.INTX2.GT.0) GO TO 1
      WRITE (6,12)
      STOP 1
C
C     TEST CORRECT SIZE OF THE MATRIX
C
1     IF (ISIZE.GT.1.AND.JSIZE.GT.1) GO TO 2
      WRITE (6,11)
      STOP 2
C
C     CHECK IF NUMBER OF INEQUALITY CONSTRAINTS DOES NOT EXCEEDE
C     LIMIT
C
2     IF (NCONS.LE.20.AND.NCONS.GE.0) GO TO 19
      WRITE (6,14)
      STOP 3
C
C     TEST IF NUMBER OF EQUALITY CONSTRAINTS DOES NOT EXCEEDE LIMIT
C
19    IF (IQUAL.LE.5.AND.IQUAL.GE.0) GO TO 3
      WRITE (6,15)
      STOP 4
C
C     CHECK IF KLASS IS EQUAL TO 1, 2 OR 3
C
3     IF (KLASS.GT.0.AND.KLASS.LT.4) GO TO 4
      WRITE (6,18)
      STOP 5
C
C     CHECK IF THE ANGLE OF THE SIDE OF THE MATRIX IS WITHIN LIMITS
C
4     IF (ALPHA.LT.90.0.AND.ALPHA.GE.0.0) GO TO 5
      WRITE (6,16)
      STOP 6
C
C     CHECK IF PAPER SIZE DOES NOT EXCEED 28 INCHES
C
```

```
5     IF (PAPER.GT.0.0.AND.PAPER.LE.28.0) GO TO 6
      WRITE (6,10)
      STOP 17
C
C     CHECK IF THE SIZE OF THE PLOT DOES NOT EXCEEDE THE SIZE OF
C     THE PAPER
C
6     IF (ANGLE.NE.XLETI) GO TO 7
      IF (VSIZE.GT.0.0.AND.VSIZE.LE.PAPER) GO TO 7
      WRITE (6,13) PAPER
      STOP 7
C
C     CHECK IF ANGLE OF THE VIEW (IF SPECIFIED) IS BETWEEN ZERO AND
C     90 DEGREES
C
7     IF (ANGLE.EQ.XLETI) GO TO 8
      IF (ANGLE.GE.0.0.AND.ANGLE.LE.90.0) RETURN
      WRITE (6,9)
      STOP 15
C
C     CHECK IF THE LENGTH OF THE LONGER SIDE OF THE PLOT IS POSITIVE
C
8     IF (XLENG.GT.0.0) RETURN
      WRITE (6,17)
      STOP 10
C
C
9     FORMAT (//3X,60HANGLE OF VIEW IS LESS THAN ZERO OR GREATER THAN 90
     1 DEGREES   )
10    FORMAT (//3X,60H*****PAPER SIZE IS EXCEEDING 28 INCHES OR IS LESS
     1THAN ZERO /)
11    FORMAT (//3X,78H*****ERROR IN MATRIX DIMENSIONS ISIZE OR JSIZE ARE
     1 LESS THAN OR EQUAL TO ONE  /)
12    FORMAT (//3X,83H*****NUMBER OF INTERVALS TO BE PLOTTED ALONG X1 OR
     1 X2 IS LESS THAN OR EQUAL TO ZERO/)
13    FORMAT (//3X,30H*****SIZE OF THE PLOT EXCEEDES,F5.0,41H INCHES OR
     1IS LESS THAN OR EQUAL TO ZERO /)
14    FORMAT (//3X,58H*****NUMBER OF INEQUALITY CONSTRAINTS GREATER THAN
     1 TWENTY ,18HOR LESS THAN ZERO /)
15    FORMAT (//3X,72H*****NUMBER OF EQUALITY CONSTRAINTS GREATER THAN F
     1 IVE OR LESS THAN ZERO/)
16    FORMAT (//3X,74H*****ANGLE OF THE MATRIX SIDE IS LESS THAN ZERO OR
     1 IS GREATER THAN OR EQUA,11HL TO NINETY/)
17    FORMAT (//3X, 9H*****LENGTH OF THE PLOT SIDE IS LESS THAN OR EQUAL
     1 TO ZERO /)
18    FORMAT (//3X,38H*****KLASS IS NOT EQUAL TO 1, 2 OR 3  /)
      END
      SUBROUTINE DIFFER (ISW,N,XX,IAR,N1,RAR,N2)
C
C     CONTAINS ALL INSTRUCTIONS THAT MAY DIFFER FROM ONE COMPUTER
C     TO ANOTHER
C
      DIMENSION IAR(N1), RAR(N2)
C
C
C     SUBROUTINE SETUP FOR CDC 6400
C
C     PRESET FORMAT OF ONE COMPUTER WORD
C
      DIMENSION WORD(2)
      DATA WORD/10H(020)      ,10H          /
C
      GO TO (1,2,3,4,5,6,7,8), ISW
C
C     COMPLETE VARIABLE FORMAT (5X,I3,1X,    A1,I4) IN ARRAY VAR
C     SUBSTITUTING FOR BLANKS INTEGER CONSTANT N IN FORMAT I4
C     VARIABLE FORMAT ARRAY DIMENSIONED 5 IS TRANSMITTED VIA ARRAY RAR
C
1     ENCODE (10,11,RAR(3) )RAR(3),N
      RETURN
C
C     COMPLETE VARIABLE FORMATS (9X,     (4X,1H1)) IN ARRAY VAR1 AND
C     (9X,     (3X,I2)) IN ARRAY VAR2 SUBSTITUTING FOR BLANKS INTEGER
C     VARIABLE FORMAT ARRAY DIMENSIONED 5 IS TRANSMITTED VIA ARRAY RAR
C     CONSTANT N IN FORMAT I4
C
2     ENCODE (4,12,RAR(2) )N
      RETURN
C
C     CONVERT INTEGER CONSTANT N TO BCD FORM IN FORMAT I4 AND
C     PLACE IT IN IAR
C
3     ENCODE (4,12,IAR(1) )N
      RETURN
C
C     CONVERT REAL CONSTANT XX TO BCD FORM IN FORMAT E9.2 AND
C     PLACE IT IN IAR
C
4     ENCODE (9,13,IAR(1) )XX
      RETURN
C
C     PUT THE VARIABLE FORMAT OF ONE COMPUTER WORD TO RAR
C
5     RAR(1)=WORD(1)
      RAR(2)=WORD(2)
      RETURN
C
```

```
C       TEST IF MEMORY SPACE ALLOCATED IS SUFFICIENT FOR DYNAMIC
C       ALLOCATION OF U-MATRIX AND/CR WORKING ARRAYS
C
6       CALL SPACE (LWA,MEMORY)
C
        MEMSP=MEMORY-LWA
        WRITE (6,14) MEMSP
        IF (N.LE.MEMSP) RETURN
        WRITE (6,15)
        STOP 16
C
C       INDICATE COLUMN WITH ERROR IN INPUT DATA
C       COMPLETE VARIABLE FORMAT RAR (    X,1H↑) WITH THREE DIGIT
C       INTEGER N AND PRINT IT OUT
C
7       ENCODE (4,16,RAR(1) )RAR(1),N
        WRITE (6,RAR)
        RETURN
C
C       TEST FOR END OF FILE ON UNIT NO. 2
C       SET N=1 IF EOF FOUND
C
8       IF (EOF,2) 9,10
9       N=1
10      RETURN
C
C
11      FORMAT (A2,I8)
12      FORMAT (I4)
13      FORMAT (E9.2)
14      FORMAT (8X,18HSPACE AVAILABLE IS,I7,10H LOCATIONS)
15      FORMAT (//5X,40H*****MEMORY ALLOCATED IS NOT SUFFICIENT )
16      FORMAT (A1,I3)
        END
```

```
      SUBROUTINE ELEV (IE,JE,U,NUI,NUJ)
C
C     MOVES PEN FROM THE BASE TO THE SURFACE AND VICE VERSA
C
      DIMENSION U(NUI,NUJ)
C
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,NRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
      NELEV=1
      STEP=0.05
      IF (NUP) 13,1,13
C
C     THE PEN IS AT PRESENT AT THE BASIS
C
1     IF (JE.EQ.1.OR.IE.EQ.1) GO TO 8
C
      MODE=0
      CALL  EYE (IE,JE,U50(IE,JE,U,NUI,NUJ),U,NUI,NUJ)
C
      IF (ISKIP) 7,2,7
2     IF (MODE-IPEN) 3,11,3
3     UTEST=YCO(IE,JE)+STEP
4     IF (UTEST.GE.U(IE,JE)) GO TO 7
C
      MODE=0
      CALL  EYE (IE,JE,UTEST,U,NUI,NUJ)
C
      IF (IPEN-MODE) 10,5,10
5     UTEST=UTEST+STEP
      GO TO 4
6     MODE=3
      GO TO 12
7     MODE=3
      GO TO 11
8     IF (ISKIP) 6,9,6
9     MODE=2
      GO TO 12
C
10    CALL  MOVE (XCO(IE,JE),U50(I3,J3,U,NUI,NUJ),IPEN)
      DUM=U50(I3,J3,U,NUI,NUJ)
11    CALL  EYE  (IE,JE,U50(IE,JE,U,NUI,NUJ),U,NUI,NUJ)
12    CALL  MOVE  (XCO(IE,JE),U50(IE,JE,U,NUI,NUJ),MODE)
      NUP=1
      GO TO 27
C
C     THE PEN IS AT PRESENT AT THE SURFACE
```

```
C
13      IF (IE.EQ.1.OR.JE.EQ.1) GO TO 22
C
C       SET MODE UNCONDITIONALY TO 3 IF THE VERTICAL LINE IS COVERED BY
C       ADJACENT POSITIVE U
C
        IF (U(IE-JK,JE-IK)) 16,16,14
14      IF (NINCO) 16,15,16
15      MODE=3
        I3=IE
        J3=JE
        IPEN=3
        GO TO 26
C
16      MODE=0
        CALL EYE (IE,JE,YCO(IE,JE),U,NUI,NUJ)
C
        IF (ISKIP) 14,17,14
17      IF (IPEN-MODE) 18,25,18
18      UTEST=U50(IE,JE,U,NUI,NUJ)-STEP
19      IF (UTEST.LE.YCO(IE,JE)) GO TO 22
C
        MODE=0
        CALL EYE (IE,JE,UTEST,U,NUI,NUJ)
C
        IF (IPEN-MODE) 23,20,23
20      UTEST=UTEST-STEP
        IFIRST=0
        GO TO 19
21      MODE=3
        GO TO 26
22      MODE=2
        GO TO 25
C
23      IF (IFIRST) 21,24,21
24      CALL MOVE (XCO(IE,JE),U50(I3,J3,U,NUI,NUJ),IPEN)
        DUM=U50(I3,J3,U,NUI,NUJ)
25      CALL EYE (IE,JE,YCO(IE,JE),U,NUI,NUJ)
26      CALL MOVE (XCO(IE,JE),YCO(IE,JE),MODE)
        NUP=0
        GO TO 36
C
C       CHECK IF ELEV IS GOING TO BE USED IN THE FOLLOWING MOVE IN
C       ORDER TO AVOID DOUBLE VERTICAL LINE
C
27      CONTINUE
        IF (NINCO) 36,28,36
28      IF (JK) 31,29,31
29      IF (IE.GE.NUI) GO TO 30
        IF (U(IE+1,JE)) 34,30,30
```

```
30    ISKIP=0
      GO TO 36
31    IF (JE.GE.NUJ) GO TO 30
      IF (U(IE,JE+1)) 32,30,30
32    IF (IE.LE.1) GO TO 35
      IF (U(IE-1,JE)) 35,33,33
33    IPEN=3
      I3=IE
      J3=JE
      GO TO 35
34    IF (JE.LE.1) GO TO 35
      IF (U(IE,JE-1)) 35,33,33
35    ISKIP=1
C
36    NELEV=0
      RETURN
      END
      SUBROUTINE EYE (I,J,UTEST,U,NUI,NUJ)
C
C     RETURNS MODE OF THE PEN FOR THE NEXT MOVEMENT
C
      DIMENSION U(NUI,NUJ)
C
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,MRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
      NCORN=0
      IT=I
      JT=J
C
      NEG=1
1     IF (IT.EQ.1.OR.JT.EQ.1) GO TO 7
      IT=IT-1
      JT=JT-1
      IF (U(IT,JT)) 1,2,2
2     NEG=0
      IF (UTEST.GE.U50(IT,JT,U,NUI,NUJ)) GO TO 1
      MEYE=3
      IF (NELEV) 4,3,4
3     IF (MEYE-IPEN) 21,4,21
4     I3=IT
      J3=JT
5     IF (MODE) 59,6,59
6     MODE=MEYE
      GO TO 60
C
7     IF (NELEV) 8,17,8
8     IF (NINCO) 17,9,17
9     IF (NEG) 10,17,10
10    IF (I-1) 17,17,11
11    IF (U(I-1,J)) 17,12,12
12    IF (J-1) 17,17,13
13    IF (U(I,J-1)) 17,14,14
14    ISKIP=1
      IF (MODE) 16,15,16
15    MODE=3
      GO TO 60
16    IPEN=3
      GO TO 60
C
17    MEYE=2
      IF (NELEV) 5,18,5
18    IF (MEYE-IPEN) 19,5,19
19    IF (IART3) 5,20,5
```

```
20    IT=I3
      JT=J3
      I23=0
C
C     IPEN WAS 3.  CHECK FOR CORNER CONTOUR
C
      CALL CORNER (U,NUI,NUJ)
C
      IF (NCORN) 5,26,5
C
C
C     IPEN WAS 2. CHECK FOR CORNER CONTOUR
C
21    I23=1
      ITS=IT
      JTS=JT
      IT3=IT
      JT3=JT
C
22    CALL CORNER (U,NUI,NUJ)
C
      IF (NCORN) 23,25,23
23    IF (IT3.EQ.1.OR.JT3.EQ.1) GO TO 60
      IT3=IT3-1
      JT3=JT3-1
      IF (U50(IT3,JT3,U,NUI,NUJ).GT.UTEST.AND.U50(IT3,JT3,U,NUI,NUJ).LT.
     1U50(IT,JT,U,NUI,NUJ)) GO TO 24
      GO TO 23
24    IT=IT3
      JT=JT3
      GO TO 22
C
C     NOT A CORNER, CONTOUR LINE
C     SET THE INDICATORS OF THE DIAGONALE ALONG WHICH POINTS OF
C     INTERSECTION ARE TO BE CHECKED
C
25    IMAT=ITS
      JMAT=JTS
      GO TO 27
26    IMAT=IT
      JMAT=JT
C
C     SET JUMP
C
27    JUMP=0
      IF (NINCO) 28,29,28
28    IF (INCOJK) 30,31,30
29    IF (JK) 31,30,31
30    JUMP=1
31    IF (I23) 35,33,35
```

```
32    JUMP=0
      GO TO 35
33    IF (JUMP) 32,34,32
34    JUMP=1
C
C     DEFINE COORDINATES OF THE PEN S LINE
C
35    CONTINUE
      CALL LINES (1,J,U,NUI,NUJ)
C
      NFAIL=1
C
C     SPECIFY YY1 AND YY2 AS COORDINATES OF INTERSECTING LINE
C
36    YY2=U50(IMAT,JMAT,U,NUI,NUJ)
      ISW=1
      IF (JUMP) 44,37,44
C
37    IF ((JMAT-1).LT.1) GO TO 42
      IF (U(IMAT,JMAT-1)) 42,38,38
38    YY1=U50(IMAT,JMAT-1,U,NUI,NUJ)
      GO TO 49
C
39    IF (JUMP) 41,40,41
40    IF (U(IMAT+1,JMAT)) 55,43,43
41    IF (U(IMAT,JMAT+1)) 55,47,47
C
42    IF (U(IMAT+1,JMAT)) 55,43,43
43    YY1=U50(IMAT+1,JMAT,U,NUI,NUJ)
      GO TO 48
C
C
44    IF ((IMAT-1).LT.1) GO TO 46
C
      IF (U(IMAT-1,JMAT)) 46,45,45
45    YY1=U50(IMAT-1,JMAT,U,NUI,NUJ)
      GO TO 49
C
46    IF (U(IMAT,JMAT+1)) 55,47,47
47    YY1=U50(IMAT,JMAT+1,U,NUI,NUJ)
C
48    ISW=2
C
C     FIND POINT OF INTERSECTION
C
49    CALL CROSS
C
C     RESET THE COORDINATES OF THE POINT OF INTERSECTION IF NEW
C     POINT HAS BEEN FOUND AND THE DISTANCE BETWEEN THE NEW
C     INTERSECTION AND THE POINT WHERE THE PEN IS OR WILL BE ON THE
```

```
C       PAPER IS LESS THAN THE PRESENT LEST ONE.
C       X1 AND Y1 ARE THE COORDINATES OF THE POINT WITH THE PEN ON
C       THE PAPER.
C
        IF (INT) 50,54,50
50      YY2=C50(IMAT,JMAT,U,NOI,NOJ)
        DISN=SQRT((X1-XN)**2+(Y1-YN)**2)
        IF (NFAIL) 51,52,51
51      NFAIL=0
        GO TO 53
52      IF (DISN.GE.DISCR) GO TO 54
53      XCROSS=XN
        YCROSS=YN
        DISCR=DISN
54      GO TO (59,55), ISW
C
C       CARRY THE PROCEDURE ALONG THE DIAGONALE UP TO THE EDGE OF
C       THE MATRIX
C
55      IMAT=IMAT-1
        JMAT=JMAT-1
        IF (IMAT.EQ.0.OR.JMAT.EQ.0) GO TO 56
        IF (U(IMAT,JMAT)) 55,36,36
56      IF (NFAIL) 57,58,57
57      IPEN=2
        GO TO 4
C
C       MOVE THE PEN IN THE PRESENT MODE UP TO THE POINT OF INTERSECTION
C
58      CALL MOVE (XCROSS,YCROSS,IPEN)
        IF (IPEN-2) 5,4,5
C
59      IPEN=MEYE
60      ULAST=UTEST
        RETURN
        END
```

```
      SUBROUTINE FIZL (I,J,U,NUI,NUJ)
C
C     SUBROUTINE FIZL IS FOLLOWING A GIVEN PATH, CHECKING U-FUNCTION
C     BEFORE ADVANCING THE PEN AND MOVING THE PEN TO THE NEXT POSITION
C
      DIMENSION U(NUI,NUJ)
C
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IEL,IC(50),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,NRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
C     I AND J ARE POINTING AT THE POSITION TO WHICH IS THE PEN TO BE
C     MOVED NEXT. SEE IF IT IS TO BE SURFACE OR BASIS
C
      IF (U(I,J)) 1,5,5
C
C     MOVE ON THE BASIS, CHECK PRESENT POSITION OF THE PEN
C
1     IF (NUF) 2,4,2
C
C     PEN IS AT THE SURFACE. SLIDE IT DOWN TO THE BASIS FIRST
C
2     CALL ELEV (I-IK,J-JK,U,NUI,NUJ)
      IF (JK) 3,4,3
3     IF (U(I-IK,J-JK).GE.50.0) GO TO 4
      U(I-IK,J-JK)=U(I-IK,J-JK)+50.0
4     ISKIP=0
C
C     THE PEN IS AT THE BASIS. LOOK FOR MODE OF THE MOVE AND MOVE.
C
      CALL EYE (I,J,YCO(I,J),U,NUI,NUJ)
      CALL MOVE (XCO(I,J),YCO(I,J),IPEN)
      GO TO 20
C
5     IF (U(I,J).LT.50.0) GO TO 6
      U(I,J)=U(I,J)-50.0
      ISKIP=1
6     IF (I.EQ.2.AND.IK.EQ.1) GO TO 7
      IF (J.EQ.2.AND.JK.EQ.1) GO TO 7
      GO TO 11
C
C     PATH STARTS HERE. THE PEN IS STIL AT THE BASIS SO IT MUST BE
C     LIFTED UP IF U AT THE STARTING POSITION WAS POSITIVE
C
7     IF (U(I-IK,J-JK)) 11,8,8
8     IF (U(I-IK,J-JK).LT.50.0) GO TO 9
      U(I-IK,J-JK)=U(I-IK,J-JK)-50.0
      ISKIP=1
      GO TO 10
9     ISKIP=0
10    CALL  ELEV (I-IK,J-JK,U,NUI,NUJ)
      IF (U(I-IK,J-JK).LT.50.0) U(I-IK,J-JK)=U(I-IK,J-JK)+50.0
C
C     MOVE  ON THE  SURFACE. CHECK PRESENT POSITION OF THE PEN
C
11    IF (NUP) 19,12,19
C
C     THE  PEN IS AT THE BASIS. MOVE IT PRIOR TO LIFTING UP TO THE
C     SURFACE
C
12    CALL  EYE (I,J,YCO(I,J),U,NUI,NUJ)
      CALL  MOVE (XCO(I,J),YCO(I,J),IPEN)
      ASSIGN 17 TO ISW
C
13    IF (NCORN-1) 16,14,16
14    IF (I23-1) 16,15,16
15    IPEN=3
      I3=IT
      J3=JT
16    NCORN=0
      GO TO ISW, (17,18)
C
17    CONTINUE
      CALL  ELEV (I,J,U,NUI,NUJ)
      IF (ISKIP.EQ.0) U(I,J)=U(I,J)+50.0
18    RETURN
C
C
C     THE  PEN IS ALREADY AT THE SURFACE. LOOK FOR MODE OF THE MOVEMENT
C     AND MOVE.
C
19    UIJ=U(I,J)
      CALL  EYE (I,J,UIJ,U,NUI,NUJ)
      CALL  MOVE (XCO(I,J),UIJ,IPEN)
20    ASSIGN 18 TO ISW
      GO TO 13
      END
      SUBROUTINE GETMAT (U,NUI,NUJ)
C
C     SUPPLIS THE PROGRAM WITH U-MATRIX IN A MANNER SPECIFIED BY KLASS
C     KLASS=1        MATRIX IS CALCULATED USING SUBROUTINE OBJECT
C     KLASS=2        MATRIX IS SUPPLIED BY USER↑S SUBROUTINE ↑UFILL↑
C     KLASS=3        MATRIX IS EXTERNALY SUPPLIED ONLY PARTIALY AND IS TO
C                    BE COMPLETED BY INTERPOLATION
C
      DIMENSION U(NUI,NUJ)
C
      COMMON /DATA/ INTX1,INTX2,ISIZE,JSIZE,NCONS,IQUAL,KLASS,ALPHA,ANGL
     1E,PAPER,VSIZE,XLENG,XYROT,X1MAX,X1MIN,X2MAX,X2MIN
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,NRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
1     IF (KLASS-1) 4,2,4
C
C     CALCULATE U-MATRIX
C
C     LEFT-RIGHT PATHS
C
2     DO 3 I=1,NUI
      XI=X1MIN+DEL1*FLOAT(I-1)
C
C     RIGHT-LEFT PATHS
C
      DO 3 J=1,NUJ
      XJ=X2MIN+DEL2*FLOAT(J-1)
C
      CALL OBJECT (XI,XJ,U(I,J))
C
3     CONTINUE
      RETURN
C
C     CALL SUBROUTINE TO FILL U-MATRIX
C
4     CALL UFILL (U,NUI,NUJ)
      IF (KLASS.EQ.2) RETURN
C
C     U-ARRAY IS FILLED PARTIALY, REMAINING POINTS HAVE TO BE GENERATED
C
C     CALCULATE VARIABLE DIMENSIONS OF WORKING ARRAYS
C
      K5=NUI*NUJ+1
      K=NUI
      IF (NUJ.GT.NUI) K=NUJ
      K1=K5+K+1
      K2=K1+K
      K3=K2+K
      K4=K3+3*K
C
C      CALL INTERPOLATION ROUTINES
C
      CALL XINTER (U,NUI,NCJ,U(K5,1),U(K1,1),U(K2,1),U(K3,1),U(K4,1))
C
      RETURN
C
      END
      SUBROUTINE IGET (I)
C
C     RETURNS POSITION OF THE NEXT NONBLANK CHARACTER TO BE ANALYSED
C     BY SUBROUTINE INPUT
C
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,MRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
C
1     I=I+1
      IF (I.LE.80) GO TO 2
      READ (5,3) IC
      I=1
2     IF (IC(I).EQ.IBL) GO TO 1
      RETURN
C
3     FORMAT (80A1)
      END
      SUBROUTINE INCO (I,J,U,NUI,NUJ)
C
C     CHECKS FOR INNER CORNER AND ARRANGES DRAWING OF ITS CONTOURS
C
      DIMENSION U(NUI,NUJ)
C
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(30),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KL,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,MRENUM,UP(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
      NINCO=1
      GO TO (1,7,28), NX1
1     CONTINUE
      IF (I-1) 45,45,2
2     IF (U(I,J)) 45,3,3
3     IF (U(I-1,J-1)) 4,45,45
4     IF (U(I-1,J)) 45,5,5
5     IF (U(I,J-1)) 45,6,6
C
C     STORE PRESENT INDICATORS
C
6     NUPS=NUP
      NCORNS=NCORN
      ULASTS=ULAST
      I3S=I3
      J3S=J3
      IPENS=IPEN
      CALL PLOT (XS,YS,0)
C
C     SLIDE THE PEN DOWN TO THE BASIS
C
      CALL ELEV (I,J,U,NUI,NUJ)
C
C     STORE INDICATORS
C
      ULASS=ULAST
      IPENSS=IPEN
      I3SS=I3
      J3SS=J3
      CALL PLOT (XSS,YSS,0)
C
C     DRAW LEFT-RIGHT CORNER CONTOUR AT THE BASIS
C
7     NUPOS=0
      NUNEG=0
      INCOIK=0
      INCOJK=1
      IART3=0
      JI=J
8     JI=JI-1
      CALL  EYE  (I,JI,YCO(I,JI),U,NUI,NUJ)
      IF  (NUNEG)  10,9,10
9     IF  (NUPOS)  10,11,10
10    IPS=IPEN
      IPEN=3
      IART3=1
      GO  TO  14
C
11    IF  (U(I-1,JI))  14,12,12
12    IF  (IPEN-2)  14,13,14
13    IPEN=3
      I3=I-1
      J3=JI-1
14    CALL  MOVE  (XCO(I,JI),YCO(I,JI),IPEN)
      IF  (NUPOS)  15,16,15
15    NUPOS=0
      IPEN=IPS
      IART3=0
16    IF  (NCORN)  17,19,17
17    IF  (I23)  18,19,18
18    IPEN=3
      I3=I
      J3=JI
19    IF  (JI-1)  24,24,20
20    IF  (U(I-1,JI-1).GE.0.0.AND.U(I,JI-1).GE.0.0)  GO  TO  24
      IF  (U(I,JI-1))  21,22,22
21    NUNEG=1
      GO  TO  8
22    IF  (NUNEG)  23,8,23
23    NUPOS=1
      NUNEG=0
      GO  TO  8
C
C     DRAW  RIGHT-LEFT  CORNER  CONTOUR  AT  THE  BASIS
C
24    IF  (NX1-1)  45,25,45
25    CALL  MOVE  (XSS,YSS,3)
      IPEN=IPENSS
      NCORN=NCORNS
      I3=I3SS
      J3=J3SS
      ULAST=ULASS
C
26    NUPOS=0
      NUNEG=0
      INCOIK=1
      INCOJK=0
      IART3=0
      II=I
27    II=II-1
      CALL EYE (II,J,YCO(II,J),U,NUI,NUJ)
      IF (NUNEG) 29,28,29
28    IF (NUPOS) 29,30,29
29    IPS=IPEN
      IPEN=3
      IART3=1
      GO TO 33
C
30    IF (U(II,J-1)) 33,31,31
31    IF (IPEN-2) 33,32,33
32    IPEN=3
      I3=II-1
      J3=J-1
33    CALL MOVE (XCO(II,J),YCO(II,J),IPEN)
      IF (NUPOS) 34,35,34
34    NUPOS=0
      IPEN=IPS
      IART3=0
35    IF (NCORN) 36,38,36
36    IF (I23) 37,38,37
37    IPEN=3
      I3=II
      J3=J
38    IF (II-1) 43,43,39
39    IF (U(II-1,J-1).GE.0.0.AND.U(II-1,J).GE.0.0) GO TO 43
      IF (U(II-1,J)) 40,41,41
40    NUNEG=1
      GO TO 27
41    IF (NUNEG) 42,27,42
42    NUPOS=1
      NUNEG=0
      GO TO 27
C
C     RESET INDICATORS
C
43    IF (NX1-1) 45,44,45
44    CALL MOVE (XS,YS,3)
      NUP=NUPS
      IPEN=IPENS
      NCORN=NCORNS
      ULAST=ULASTS
      I3=I3S
      J3=J3S
45    NINCC=0
      IART3=0
      RETURN
      END
      SUBROUTINE INITP (X)
C
C     INITIATES PLOTTING  ***MUST BE REMOVED IF RUN ON IBM 360
C
      CALL PLOT (0.0,0.0,3)
      RETURN
      END
      SUBROUTINE INPUT (NUI,NUJ)
C
C     READS INPUT DATA IN FORM   KEY WORD = INTEGER OR E-FORMAT NUMBER
C
      COMMON /DATA/ INTX1,INTX2,ISIZE,JSIZE,NCONS,IQUAL,KLASS,ALPHA,ANGL
     1E,PAPER,VSIZE,XLENG,XYROT,X1MAX,X1MIN,X2MAX,X2MIN
      COMMON /HEAD/ HEAD(20)
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,J1,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,NRENUM,OF(130),OLAST,OM
     4AX,OMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
      COMMON /SWITCH/ KEYS(5,10),NKEYS,NSW(10)
C
      DIMENSION ID(7), D(10), VAR(3)
C
      EQUIVALENCE (ID(1),INTX1), (D(1),ALPHA)
C
      DATA IS/80/,VAR/4H(    ,4HX,    ,4H1H↑)/
C
C
C
    1 CALL IGET (IS)
      I=IS
C
C     SET SWITCHES
C
      DO 6 J=1,NKEYS
      J1=0
    2 J1=J1+1
      IF (IC(I).NE.KEYS(J1,J)) GO TO 5
      IF (J1.GE.5) GO TO 3
      CALL IGET (I)
      GO TO 2
    3 IF (J.EQ.NKEYS) GO TO 40
C
C     SET APPROPRIATE SWITCH
C
      NSW(J)=1
C
C     SEARCH FOR COMMA
C
    4 CALL IGET (I)
      IF (IC(I).NE.ICOM) GO TO 4
      IS=I
      GO TO 1
    5 I=IS
    6 CONTINUE
C
C     GET NUMERICAL VALUES
```

```
C
      DO 38 J=1,NKEY
      J1=0
7     J1=J1+1
      IF (IC(I).NE.KEY(J1,J)) GO TO 37
      IF (5-J1) 9,9,8
8     CALL IGET (I)
      GO TO 7
C
C      SEARCH FOR = SIGN
C
9     CALL IGET (I)
      IF (IC(I).NE.IEQ) GO TO 9
      IF (J.GT.INTNUM) GO TO 20
C
C       GET INTEGER NUMBER
C
      IVAL=0
      ISIGN=1
      ISW=0
      CALL IGET (I)
      IF (IC(I).EQ.MINUS) GO TO 16
      IF (IC(I).EQ.IPLUS) GO TO 17
10    IF (ISW) 12,11,12
11    IF (IC(I).EQ.ICOM) GO TO 18
12    DO 13 J2=1,10
      IF (IC(I).EQ.IDIG(J2)) GO TO 14
13    CONTINUE
      GO TO 39
14    IVAL=IVAL*10+J2-1
      ISW=0
15    CALL IGET (I)
      GO TO 10
16    ISIGN=-1
17    ISW=1
      GO TO 15
18    ID(J)=IVAL*ISIGN
19    IS=I
      GO TO 1
C
C       GET E-FORMAT NUMBER
C
20    VAL=0.0
      NPER=0
      NSIGEX=0
      NE=0
      IEX=0
      DIV=1
      ISW=0
      CALL IGET (I)
      ISIGN=1
      IF (IC(I).EQ.IPLUS) GO TO 21
      IF (IC(I).NE.MINUS) GO TO 24
      ISIGN=-1
21    ISW=1
C
22    CALL IGET (I)
      IF (ISW) 24,23,24
23    IF (IC(I).EQ.ICOM) GO TO 36
24    DO 25 J2=1,10
      IF (IC(I).EQ.IDIG(J2)) GO TO 32
25    CONTINUE
      IF (NPER) 27,26,27
26    IF (IC(I).NE.IPER) GO TO 27
      NPER=1
      GO TO 22
C
27    IF (NE) 29,28,29
28    IF (IC(I).NE.KE) GO TO 39
      NE=1
      NPER=1
      GO TO 22
C
29    IF (NSIGEX) 39,30,39
30    IF (IC(I).EQ.IPLUS) GO TO 31
      IF (IC(I).NE.MINUS) GO TO 39
      NSIGEX=-1
      GO TO 21
31    NSIGEX=1
      GO TO 21
C
32    IF (NE.EQ.1.AND.NSIGEX.EQ.0) NSIGEX=1
      ISW=0
      IF (NSIGEX) 35,33,35
C
33    VAL=VAL*10.0+FLOAT(J2-1)
      IF (NPER) 34,22,34
C
34    DIV=DIV*10.0
      GO TO 22
C
35    IEX=IEX*10+J2-1
      GO TO 22
C
36    D(J-INTNUM)=FLOAT(ISIGN)*(VAL/DIV)*10.0**(IEX*NSIGEX)
      GO TO 19
37    I=IS
38    CONTINUE
C
39    WRITE (6,42) I,IC
```

```
      CALL DIFFER (7,I+7,A,IC,1,VAR,3)
      STOP 21
C
C     READ HEADING
C
40    READ (5,41) HEAD
C
      NUI=ISIZE
      NUJ=JSIZE
      RETURN
C
41    FORMAT (20A4)
42    FORMAT (//3X,50H*****SYNTAX ERROR IN INPUT DATA IN OR AFTER COLUMN
     1,I3//5X,3H***,80A1,3H***)
      END
      SUBROUTINE LINES (I,J,U,NUI,NUJ)
C
C      DEFINES INTERSECTING LINES
C
      DIMENSION U(NUI,NUJ)
C
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,MRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
      IF (NINCO) 4,1,4
C
C     PATH OF THE PEN
C
C     PEN IS AT I-IK,J-JK AND IS TO BE MOVED TO I,J
C
1     X1=XCO(I-IK,J-JK)
      X2=XCO(I,J)
      IF (NUP) 2,3,2
2     Y1=U50(I-IK,J-JK,U,NUI,NUJ)
      Y2=U50(I,J,U,NUI,NUJ)
      GO TO 5
3     Y1=YCO(I-IK,J-JK)
      Y2=YCO(I,J)
      GO TO 5
C
C     PEN IS AT I+INCOIK,J+INCOJK AND IS TO BE MOVED TO I,J
C
4     X2=XCO(I,J)
      X1=XCO(I+INCOIK,J+INCOJK)
      Y2=YCO(I,J)
      Y1=YCO(I+INCOIK,J+INCOJK)
C
C     CHECK FOR POSITION OF THE MAXIMUM
C
5     IF (I23) 7,6,7
C
C     MAXIMUM IS AT THE DIAGONALE OF THE PEN
C     SWITCH COORDINATES OF THE PEN↑S PATH
C
6     TEMP=X1
      X1=X2
      X2=TEMP
      TEMP=Y1
      Y1=Y2
      Y2=TEMP
7     RETURN
      END
      SUBROUTINE MARK (U,NUI,NUJ)
C
C     WRITES TEXT ON THE PLOT
C
      DIMENSION U(NUI,NUJ)
C
      COMMON /DATA/ INTX1,INTX2,ISIZE,JSIZE,NCONS,IQUAL,KLASS,ALPHA,ANGL
     1E,PAPER,VSIZE,XLENG,XYROT,X1MAX,X1MIN,X2MAX,X2MIN
      COMMON /A/ PHI(20),PSI(5),KODE(5)
      COMMON /HEAD/ HEAD(20)
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,PRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
      COMMON /SWITCH/ KEYS(5,10),NKEYS,NSW(10)
C
      DIMENSION ID(7), D(10)
C
      EQUIVALENCE (ID(1),INTX1), (D(1),ALPHA)
C
      DATA XLETI/1HI/
C
C     SET X1 AND X2 BOUNDS TO THE SIZE OF THE MATRIX IF BOUNDS WERE
C     NOT DEFINED
C
      IF (X1MIN) 5,1,5
1     IF (X1MAX) 5,2,5
2     IF (X2MIN) 5,3,5
3     IF (X2MAX) 5,4,5
4     X1M=1.0
      X2M=1.0
      X1ST=FLOAT(NUI-1)/10.0
      X2ST=FLOAT(NUJ-1)/10.0
      GO TO 6
C
C     SET VALUES FOR FUTURE MARKING OF THE AXIS
C
5     X1M=X1MIN
      X2M=X2MIN
      X1ST=(X1MAX-X1MIN)/10.0
      X2ST=(X2MAX-X2MIN)/10.0
C
C     RESET BOUNDS TO THEIR ORIGIN VALUES FOR CORRECT LISTING
C
6     IF (XYROT) 7,8,7
7     X1MIN=PHI(1)
      X1MAX=PHI(2)
      X2MIN=PHI(3)
      X2MAX=PHI(4)
```

```
C
C      WRITE INPUT DATA ON THE PLOT
C
8     SIZE=0.2
      XST=-FLOAT(NUI-1)*DX-5.5
      RADALP=ALPHA*4.0*ATAN(1.0)/180.0
      IF (RADALP.LT.0.01) RADALP=0.01
      YL=10.1
      DDY=1.5*SIZE
C
C     WRITE HEADING
C
      XH=-FLOAT(NUI-1)*DX
      DO 9 I=1,20
      CALL LETTER (4,SIZE,0.0,XH,YL,HEAD(I))
      XH=XH+4.0*SIZE-SIZE/4.0
9     CONTINUE
      YL=YL-0.4
      SIZE=0.12
C
      XL=XST
      CALL LETTER (18,SIZE,0.0,XL,YL,18HLIST OF PARAMETERS)
      YL=YL-0.1
      CALL PLOT (XL+18.*SIZE,YL,3)
      CALL PLOT (XL,YL,2)
      YL=YL-0.1
C
      NLOOP=NKEY-1
      DO 13 J=1,NLOOP
      YL=YL-DDY
      DO 10 I=1,5
      XL=XST+SIZE*FLOAT(I-1)
      CALL LETTER (1,SIZE,0.0,XL,YL,KEY(I,J))
10    CONTINUE
C
      XL=XL+2.0*SIZE
      CALL LETTER (1,SIZE,0.0,XL,YL,IEQ)
      XL=XL+3.0*SIZE
C
      IF (J.GT.INTNUM) GO TO 11
C
      CALL DIFFER (3,ID(J),0.0,IC,3,OF,3)
      CALL LETTER (4,SIZE,0.0,XL,YL,IC)
      GO TO 13
C
11    IF (J.EQ.(INTNUM+2).AND.ANGLE.EQ.XLET1) GO TO 12
      CALL DIFFER (4,NX,D(J-INTNUM),IC,3,OF,3)
      CALL LETTER (9,SIZE,0.0,XL,YL,IC)
      GO TO 13
12    CALL LETTER (6,SIZE,0.0,XL,YL,6H   N/A)
```

```
C
13    CONTINUE
C
C      MARK X1 AND X2 AXIS
C
C
      HOR=DX*FLOAT(NUJ-1)/10.0
      VERT=DY*FLOAT(NUJ-1)/10.0
      IF (VERT.LT.0.05) GO TO 15
      SIZE=0.12
      IF (VERT.LT.0.14) SIZE=0.06
      DV=SIZE/2.0
      DH=-0.6
      DO 14 I=1,11
      A=X2M+X2ST*FLOAT(I-1)
      V=VERT*FLOAT(I-1)
      H=HOR*FLOAT(I-1)
      CALL PLOT (H,V,3)
      CALL PLOT (H+0.5,V,2)
      CALL DIFFER (4,NX,A,IC,3,UF,3)
      CALL LETTER (9,SIZE,0.0,H-DH,V-DV,IC)
14    CONTINUE
15    CONTINUE
C
      DIS=1.5
      IF (SIZE.EQ.0.06) DIS=1.0
      XX2=DIS/SIN(RADALP)
      IF (XX2.GT.(FLOAT(NUJ-1)*DX+2.5)) XX2=FLOAT(NUJ-1)*DX+2.5
      CALL LETTER (1,0.2,0.0,XX2,0.2,1HX)
      LET=IDIG(3)
      IF (NROT.NE.2) LET=IDIG(2)
      CALL LETTER (1,0.2,0.0,XX2+0.2,0.2,LET)
C
C      MARK X1 AXIS
C
      HOR=DX*FLOAT(NUI-1)/10.0
      VERT=DY*FLOAT(NUI-1)/10.0
      IF (VERT.LT.0.05) GO TO 17
      SIZE=0.12
      IF (VERT.LT.0.14) SIZE=0.06
      DV=SIZE/2.0
      DH=SIZE*9.0+0.6
      DO 16 I=1,11
      A=X1M+X1ST*FLOAT(I-1)
      CALL DIFFER (4,NX,A,IC,3,UF,3)
      V=VERT*FLOAT(I-1)
      H=-HOR*FLOAT(I-1)
      CALL LETTER (9,SIZE,0.0,H-DH,V-DV,IC)
      CALL PLOT (H-0.5,V,3)
      CALL PLOT (H,V,2)
```

```
16    CONTINUE
17    CONTINUE
C
      DIS=1.5
      IF (SIZE.EQ.0.06) DIS=1.0
      XX1=2.0*0.2+DIS/SIN(RADALP)
      IF (XX1.GT.(FLOAT(NUI-1)*DX+2.5)) XX1=FLOAT(NUJ-1)*DX+2.5
      CALL LETTER (1,0.2,0.0,-XX1,0.2,1HX)
      LET=IDIG(2)
      IF (NRCT.NE.2) LET=IDIG(3)
      CALL LETTER (1,0.2,0.0,-XX1+0.2,0.2,LET)
      IF (IQUAL.GT.0) RETURN
      NX3=1
C
C     MARK MAXIMUM AND MINIMUM IF THE MATRIX HAS BEEN GENERATED
C
      IF (KLASS.EQ.1) CALL PATHS (0,NUI,NUJ)
C
      RETURN
      END
      SUBROUTINE MAXIM (U,NUI,NUJ)
C
C     LOCATES MAXIMUM AND MINIMUM VALUES IN U MATRIX
C
      DIMENSION U(NUI,NUJ)
C
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,NRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
      DATA XLETI/1HI/
C
C
      MULMIN=0
      MULMAX=0
C      FIND STARTING VALUES
C
      DO 1  I=1,NUI
      DO 1  J=1,NUJ
      IF  (U(I,J).NE.XLETI) GO TO 2
1     CONTINUE
C
C      SET STARTING VALUES
C
C
2     IMIN=I
      IMAX=I
      JMIN=J
      JMAX=J
      UMIN=U(I,J)
      UMAX=U(I,J)
C
C      FIND MINIMUM AND MAXIMUM
C
      DO 8  I=1,NUI
      DO 8  J=1,NUJ
      IF  (U(I,J).EQ.XLETI)  GO TO 8
      IF  (U(I,J)-UMIN)  4,3,5
3     MULMIN=MULMIN+1
      IF  ((I+J).GE.(IMAX+JMAX))  GO TO 5
4     UMIN=U(I,J)
      IMIN=I
      JMIN=J
      GO TO 8
5     IF  (U(I,J)-UMAX)  8,6,7
6     MULMAX=MULMAX+1
      IF  ((I+J).LE.(IMAX+JMAX))  GO TO 8
7     UMAX=U(I,J)
      IMAX=I
      JMAX=J
8     CONTINUE
C
      IF (MULMIN.GT.1) WRITE (6,9) UMIN,MULMIN
      IF (MULMAX.GT.1) WRITE (6,10) UMAX,MULMAX
C
      RETURN
C
9     FORMAT (//5X,5HVALUE,E13.5,7H OCCURS,I5,19H TIMES AS A MINIMUM)
10    FORMAT (//5X,5HVALUE,E13.5,7H OCCURS,I5,19H TIMES AS A MAXIMUM)
      END
      SUBROUTINE MOVE (XN,YN,N)
C
C     CHECKS THE MODE OF THE MOVEMENT OF THE PEN AND MOVES THE PEN
C     FOR N=2 ONLY PREVENTING THE PEN TRAVELLING INEFICIENTLY
C     ABOVE THE PAPER
C
      IF (N-3) 1,4,1
1     IF (NLAST-3) 3,2,3
C
2     CALL PLOT (XL,YL,3)
C
3     CALL PLOT (XN,YN,2)
C
4     NLAST=N
      XL=XN
      YL=YN
C
      RETURN
      END
      SUBROUTINE NKVD (U,NUI,NUJ)
C
C     LIQUDATES POSITIVE VALUES OF U-FUNCTION IF IT STANDS ALONE
C
      DIMENSION U(NUI,NUJ)
C
      DATA XLETI/1HI/
C
C
      DO 8 I=1,NUI
      DO 8 J=1,NUJ
      IF (U(I,J).EQ.XLETI) GO TO 8
      IF (I-1) 2,2,1
1     IF (U(I-1,J).NE.XLETI) GO TO 8
2     IF (I-NUI) 3,4,4
3     IF (U(I+1,J).NE.XLETI) GO TO 8
4     IF (J-1) 6,6,5
5     IF (U(I,J-1).NE.XLETI) GO TO 8
6     IF (J-NUJ) 7,8,8
7     IF (U(I,J+1).NE.XLETI) GO TO 8
C
      U(I,J)=XLETI
      WRITE (6,9) I,J
8     CONTINUE
C
      RETURN
C
9     FORMAT (5X,14HELEMENT AT I =,I4,6H,   J =,I4,10HLIQUIDATED)
      END
      SUBROUTINE PATHS (U,NUI,NUJ)
C
C     DRAWS PATHS OF EQUALITY CONSTRAINTS AND SETS CORRESPONDING
C     U-ELEMENTS TO -2.0 FOR FURTHER PRINTING PURPOSES
C
      DIMENSION U(NUI,NUJ)
      DIMENSION PSIL(5), PSM(10)
C
      COMMON /DATA/ INTX1,INTX2,ISIZE,JSIZE,NCONS,IQUAL,KLASS,ALPHA,ANGL
     1E,PAPER,VSIZE,XLENG,XYROT,X1MAX,X1MIN,X2MAX,X2MIN
      COMMON /A/ PH (20),PSI(5),KODE(5)
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,NRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
      COMMON /SWITCH/ KEYS(5,10),NKEYS,NSW(10)
C
      DATA UND/2HUN/
C
C
      IL=1
      JL=1
      ISW=0
      IF (NX3.NE.0) GO TO 22
      DEL1=(X1MAX-X1MIN)/FLOAT(NUI-1)
      DEL2=(X2MAX-X2MIN)/FLOAT(NUJ-1)
C
      DO 1 K=1,IQUAL
      UF(K)=UND
1     CONTINUE
      I=0
      J=1
2     I=I+1
      IF (I.LE.NUI) GO TO 4
      I=0
      J=J+1
      IF (J.LE.NUJ) GO TO 2
C
      ISW=1
      J=0
      I=1
3     J=J+1
      IF (J.LE.NUJ) GO TO 4
      J=1
      I=I+1
      IF (I.GT.NUI) GO TO 21
C
4     XI=X1MIN+DEL1*FLOAT(I-1)
      XJ=X2MIN+DEL2*FLOAT(J-1)
      IS=I
      JS=J
C
      CALL EQUAL (XI,XJ)
C
      DO 20 K=1,IQUAL
      IF (I.EQ.1.OR.J.EQ.1) GO TO 19
      IF (PSI(K)*PSIL(K)) 5,19,19
5     IF (ABS(PSI(K)).LT.ABS(PSIL(K))) GO TO 6
      IF (U(IL,JL)) 19,7,7
6     IF (U(I,J)) 19,8,8
C
7     XI=X1MIN+DEL1*FLOAT(IL-1)
      XJ=X2MIN+DEL2*FLOAT(JL-1)
      I=IL
      J=JL
C
8     CALL OBJECT (XI,XJ,UU)
C
C     SET INITIAL VALUES
C
      IF (UF(K)-UND) 10,9,10
9     UF(K)=UU
      IC(K)=I
      PSM(K)=U50(I,J,U,NUI,NUJ)
      PSM(K+IQUAL)=U50(I,J,U,NUI,NUJ)
      IC(K+2*IQUAL)=J
      UF(K+IQUAL)=UU
      IC(K+IQUAL)=I
      IC(K+3*IQUAL)=J
C
C     MAXIMUM
C
10    IF (UU-UF(K)) 12,12,11
11    UF(K)=UU
      IC(K)=I
      IC(K+2*IQUAL)=J
      PSM(K)=U50(I,J,U,NUI,NUJ)
      GO TO 14
C
C     MINIMUM
C
12    IF (UU-UF(K+IQUAL)) 13,14,14
C
13    UF(K+IQUAL)=UU
      PSM(K+IQUAL)=U50(I,J,U,NUI,NUJ)
      IC(K+IQUAL)=I
      IC(K+3*IQUAL)=J
C
14    UTEST=U50(I,J,U,NUI,NUJ)
```

```
      IT=I
      JT=J
15    IF (IT.EQ.1.OR.JT.EQ.1) GO TO 17
      IT=IT-1
      JT=JT-1
      IF (U(IT,JT)) 15,16,16
16    IF (UTEST-U50(IT,JT,U,NUI,NUJ)) 18,15,15
17    CALL LETTER (1,0.12,0.0,XCO(I,J)-0.06,U50(I,J,U,NUI,NUJ)-0.06,IDIG
     1(K+1))
18    U(I,J)=-K-1
19    PSIL(K)=PSI(K)
      I=IS
      J=JS
20    CONTINUE
      IL=I
      JL=J
      IF (ISW) 3,2,3
21    IF (NSW(4).NE.0) GO TO 37
C
C      MARK MAXIMUM AND MINIMUM
C
22    K=1
      IF (IQUAL) 23,23,24
23    PSM(K)=U50(IMAX,JMAX,U,NUI,NUJ)
      GO TO 25
24    IMAX=IC(K)
      JMAX=IC(K+2*IQUAL)
C
25    ISW=0
      I=IMAX
      J=JMAX
      UTEST=PSM(K)
C
26    IF (I.EQ.1.OR.J.EQ.1) GO TO 28
      I=I-1
      J=J-1
      IF (U(I,J)) 26,27,27
27    IF (UTEST-U50(I,J,U,NUI,NUJ)) 31,26,26
C
28    IF (ISW) 30,29,30
29    CALL GRAF (XCO(IMAX,JMAX),PSM(K),0.18,5)
      CALL GRAF (XCO(IMAX,JMAX),PSM(K),0.12,5)
      CALL GRAF (XCO(IMAX,JMAX),PSM(K),0.06,5)
      GO TO 31
30    CALL GRAF (XCO(IMIN,JMIN),PSM(K+IQUAL),0.18,2)
      CALL GRAF (XCO(IMIN,JMIN),PSM(K+IQUAL),0.12,2)
      CALL GRAF (XCO(IMIN,JMIN),PSM(K+IQUAL),0.06,2)
31    IF (ISW) 36,32,36
32    ISW=1
      IF (IQUAL) 33,33,34
```

```
33    PSM(K)=U50(IMIN,JMIN,0,NUI,NUJ)
      GO TO 35
34    IMIN=IC(K+IQUAL)
      JMIN=IC(K+3*IQUAL)
35    I=IMIN
      J=JMIN
      UTEST=PSM(K+IQUAL)
      GO TO 26
C
36    IF (IQUAL.LE.0) RETURN
      K=K+1
      IF (K-IQUAL) 24,24,37
C
C     WRITE MINIMUM AND MAXIMUM VALUES
C
37    WRITE (6,38)
      WRITE (6,39) (UF(K+IQUAL),IC(K+IQUAL),IC(K+3*IQUAL),UF(K),IC(K),IC
     1(K+2*IQUAL),K=1,IQUAL)
C
      RETURN
C
38    FORMAT (//5X,63HMINIMUM AND MAXIMUM VALUES WITH RESPECT TO EQUALIT
     1Y CONSTRAINTS)
39    FORMAT (/7X,7HMINIMUM,E13.5,7H AT I =,I4,6H,  J =,I4,5X,7HMAXIMUM,
     1E13.5,7H AT I =,I4,6H,  J =,I4)
      END
      SUBROUTINE PRINTG (U,NUI,NUJ)
C
C     PRINTS PLAIN VIEW OF THE SURFACE
C
      DIMENSION U(NUI,NUJ)
C
      COMMON /DATA/ INTX1,INTX2,ISIZE,JSIZE,NCONS,IQUAL,KLASS,ALPHA,ANGL
     1E,PAPER,VSIZE,XLENG,XYROT,X1MAX,X1MIN,X2MAX,X2MIN
      COMMON /A/ PHI(20),PSI(5),KODE(5)
      COMMON /D/ DIG(5)
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,NRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
      DIMENSION LINE(25)
      DIMENSION VAR(5), VAR1(5), VAR2(5)
C
      DATA PLUS,RMINUS/1H+,1H-/
      DATA VAR/4H(5X,,4HI3,1,4HX,   ,4H   A1,4H,I4)/
      DATA VAR1/4H(9X,,4H    ,4H(4X,,4H1H1),4H)    /
      DATA VAR2/4H(9X,,4H    ,4H(2X,,4HI3)),4H    /
C
C
      WRITE (6,8)
      WRITE (6,7)
      LINEJ=NUJ
      IF (LINEJ.LE.123) GO TO 1
      LINEJ=123
      WRITE (6,10) NUJ
1     CALL DIFFER (1,LINEJ,0.0,IC,1,VAR,5)
      LI=LINEJ/5
      CALL DIFFER (2,LI,0.0,IC,1,VAR1,5)
      CALL DIFFER (2,LI,0.0,IC,1,VAR2,5)
      DO 2 I=1,LI
      LINE(I)=5*I
2     CONTINUE
      WRITE (6,VAR2) (LINE(J),J=1,LI)
      WRITE (6,VAR1)
      DO 6 K=1,NUI
      I=NUI-K+1
      DO 5 J=1,NUJ
      UF(J)=PLUS
      IF (U(I,J)+1.0) 4,3,5
3     UF(J)=RMINUS
      GO TO 5
4     JN=-U(I,J)-1.0
      UF(J)=DIG(JN)
5     CONTINUE
```

```
      WRITE (6,VAR) I,(UF(J),J=1,LINEJ),I
6     CONTINUE
      WRITE (6,VAR1)
      WRITE (6,VAR2) (LINE(J),J=1,L1)
      WRITE (6,9)
      RETURN
C
C
7     FORMAT (5X,47HTOP VIEW OF THE PLOTTED SURFACE - MATRIX U(I,J)//)
8     FORMAT (1HQ)
9     FORMAT (1HR)
10    FORMAT (//5X,33HLENGTH OF THE LINE TRUNCATED FROM,I4,7H TO 123//)
      END
      SUBROUTINE PRINTL (V2)
C
C     CALCULATES ALPHA, VSIZE AND XLENG IF THE ANGLE OF THE VIEW IS
C     SPECIFIED AND PRINTS LIST OF PARAMETERS
C
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IARTS,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,NRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
      COMMON /DATA/ INTX1,INTX2,ISIZE,JSIZE,NCONS,IQUAL,KLASS,ALPHA,ANGL
     1E,PAPER,VSIZE,XLENG,XYROT,X1MAX,X1MIN,X2MAX,X2MIN
      COMMON /HEAD/ HEAD(20)
      COMMON /SWITCH/ KEYS(5,10),NKEYS,NSW(10)
C
      DIMENSION ID(7), D(10)
C
      EQUIVALENCE (ID(1),INTX1), (D(1),ALPHA)
C
      DATA XLETI/1HI/
C
C
C
C
C     CALCULATE ALPHA, XLENG AND VSIZE FOR SPECIFIED ANGLE OF VIEW
C
      IF (ANGLE.EQ.XLETI) GO TO 1
      RR=SQRT(2.0)
      PIM=4.0*ATAN(1.0)/180.0
      BB=SQRT(PAPER**2/(1.0+RR**2))
      AA=BB*RR
      UNIT=AA*SQRT(2.0)/FLOAT(ISIZE+JSIZE-2)
      RADAN=ANGLE*PIM
      ALPHA=ATAN(SIN(RADAN))
      NIJ=ISIZE-1
      IF (JSIZE.GT.ISIZE) NIJ=JSIZE-1
      XLENG=UNIT*FLOAT(NIJ)/(SQRT(2.0)*COS(ALPHA))
      V1=AA*SIN(RADAN)
      V2=BB*COS(RADAN)
      VSIZE=V1+V2
      ALPHA=ALPHA/PIM
C
1     WRITE (6,11) HEAD
      WRITE (6,12) (KEY(I,1),I=1,5),ID(1)
      WRITE (6,15) (KEY(I,2),I=1,5),ID(2)
      WRITE (6,16) (KEY(I,3),I=1,5),ID(3)
      WRITE (6,17) (KEY(I,4),I=1,5),ID(4)
      WRITE (6,18) (KEY(I,5),I=1,5),ID(5)
      WRITE (6,19) (KEY(I,6),I=1,5),ID(6)
      IF (KLASS-2) 2,3,4
2     WRITE (6,20) (KEY(I,7),I=1,5),ID(7)
```

```
      GO TO 5
3     WRITE (6,21) (KEY(I,7),I=1,5),ID(7)
      GO TO 5
4     WRITE (6,22) (KEY(I,7),I=1,5),ID(7)
5     WRITE (6,13) (KEY(I,8),I=1,5),D(1)
      IF (ANGLE.EQ.XLETI) GO TO 6
      WRITE (6,10) (KEY(I,9),I=1,5),D(2)
      GO TO 7
6     WRITE (6,9) (KEY(I,9),I=1,5)
7     WRITE (6,23) (KEY(I,10),I=1,5),D(3)
      WRITE (6,25) (KEY(I,11),I=1,5),D(4)
      WRITE (6,26) (KEY(I,12),I=1,5),D(5)
      WRITE (6,27) (KEY(I,13),I=1,5),D(6)
      WRITE (6,28) (KEY(I,14),I=1,5),D(7)
      WRITE (6,29) (KEY(I,15),I=1,5),D(8)
      WRITE (6,30) (KEY(I,16),I=1,5),D(9)
      WRITE (6,31) (KEY(I,17),I=1,5),D(10)
C
C     PRINT LIST OF SET SWITCHES
C
      WRITE (6,24)
      NL=NKEYS-1
      DO 8 I=1,NL
      IF (NSW(I).NE.0) WRITE (6,14) (KEYS(J,I),J=1,5)
8     CONTINUE
C
      RETURN
C
9     FORMAT (5X,5A1,9H =     N/A,7X,24HANGLE OF VIEW (DEGREES) )
10    FORMAT (5X,5A1,3H = ,E10.3,3X,24HANGLE OF VIEW (DEGREES) )
11    FORMAT (1H1,5X,20HREPORT ON PLOTTING  ,/5X,18(1H=)///5X,13HHEADING
     1  *** ,20A4,4H ***,///5X,18HLIST OF PARAMETERS/5X,18(1H-)//)
12    FORMAT (5X,5A1,3H = ,I3,10X,41HNUMBER OF INTERVALS PLOTTED ALONG X
     11 AXIS)
13    FORMAT (5X,5A1,3H = ,E10.3,3X,43HANGLE OF THE BASIS OF THE PICTURE
     1 (DEGREES))
14    FORMAT (5X,5A1)
15    FORMAT (5X,5A1,3H = ,I3,10X,41HNUMBER OF INTERVALS PLOTTED ALONG X
     12 AXIS)
16    FORMAT (5X,5A1,3H = ,I3,10X,40HI-DIMENSION OF THE PLOTTED MATRIX U
     1(I,J))
17    FORMAT (5X,5A1,3H = ,I3,10X,40HJ-DIMENSION OF THE PLOTTED MATRIX U
     1(I,J))
18    FORMAT (5X,5A1,3H = ,I3,10X,32HNUMBER OF INEQUALITY CONSTRAINTS)
19    FORMAT (5X,5A1,3H = ,I3,10X,30HNUMBER OF EQUALITY CONSTRAINTS)
20    FORMAT (5X,5A1,3H = ,I3,10X,50HMATRIX WAS GENERATED BY MEANS OF SU
     1BROUTINE OBJECT)
21    FORMAT (5X,5A1,3H = ,I3,10X,24HMATRIX WAS READ IN FULLY)
22    FORMAT (5X,5A1,3H = ,I3,10X,55HMATRIX WAS READ IN PARTIALY AND COM
     1PLETED BY INTERPOLATION)
```

```
23      FORMAT (5X,5A1,3H = ,E10.3,3X,39HMINIMUM SIZE OF THE PLOT PAPER (I
       1NCHES))
24      FORMAT (/)
25      FORMAT (5X,5A1,3H = ,E10.3,3X,42HVERTICAL DIMENSION OF THE PICTURE
       1 (INCHES))
26      FORMAT (5X,5A1,3H = ,E10.3,3X,47HLENGTH OF THE LONGER SIDE OF THE
       1BASIS (INCHES))
27      FORMAT (5X,5A1,3H = ,E10.3,3X,50HANGLE OF ROTATION OF THE ORIGINAL
       1 MATRIX (DEGREES))
28      FORMAT (5X,5A1,3H = ,E10.3,3X,30HUPPER BOUND OF THE VARIABLE X1)
29      FORMAT (5X,5A1,3H = ,E10.3,3X,30HLOWER BOUND OF THE VARIABLE X1)
30      FORMAT (5X,5A1,3H = ,E10.3,3X,30HUPPER BOUND OF THE VARIABLE X2)
31      FORMAT (5X,5A1,3H = ,E10.3,3X,30HLOWER BOUND OF THE VARIABLE X2)
        END
```

```
      SUBROUTINE READU (U,NUI,NUJ)
C
C
C     READS U-MATRIX COLUMN-WISE FROM THE UNIT 2
C     U-MATRIX WAS WRITTEN ON THE UNIT 2 BY SUBROUTINE ROTOR ROW-WISE
C     STARTING FROM THE LAST ROW, THUS SUROUTINE READU RETURNS
C     THE MATRIX ROTATED CLOCK-WISE BY 90 DEGREES
C
      DIMENSION U(NUJ,NUI)
      DIMENSION OCT(2)
C
      REWIND 2
      CALL DIFFER (5,0,0.0,IC,1,OCT,2)
C
      READ (2,OCT) ((U(I,J),I=1,NUJ),J=1,NUI)
C
      ITEMP=NUI
      NUI=NUJ
      NUJ=ITEMP
C
      RETURN
      END
      SUBROUTINE ROTOR (U,NUI,NUJ)
C
C     ROTATES UPPER AND LOWER LIMITS AND CALLS FOR ROTATION OF
C     WHOLE MATRIX
C
      DIMENSION OCT(2)
      DIMENSION U(NUI,NUJ)
C
      COMMON /DATA/ INTX1,INTX2,ISIZE,JSIZE,NCONS,IQUAL,KLASS,ALPHA,ANGL
     1E,PAPER,VSIZE,XLENG,XYROT,X1MAX,X1MIN,X2MAX,X2MIN
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,NRENCH,UF(150),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
C
C
C
      ROT=XYROT
      NROT=ROT/90.0
      IF (ABS(ROT-FLOAT(NROT*90)).LT.1.0) GO TO 1
      WRITE (6,9)
      STOP 20
C
1     IF (ABS(ROT).LT.360.0) GO TO 2
      ROT=ROT-SIGN(360.0,ROT)
      GO TO 1
C
2     IF (ROT) 3,8,4
C
3     IF (ROT.EQ.-90.0) ROT=270.0
      IF (ROT.EQ.-270.0) ROT=90.0
C
4     NROT=ABS(ROT)/90.0
      CALL DIFFER (5,0,0.0,IC,1,OCT,2)
      DO 7 II=1,NROT
      TEMP=X1MAX
      X1MAX=X2MAX
      X2MAX=X1MIN
      X1MIN=X2MIN
      X2MIN=TEMP
C
      REWIND 2
      I=NUI
5     WRITE (2,OCT) (U(I,J),J=1,NUJ)
      I=I-1
      IF (I) 6,6,5
C
6     CALL READU (U,NUI,NUJ)
C
```

```
C
7     CONTINUE
C
8     RETURN
C
9     FORMAT (//3X,61H*****ROTATION ANGLE IS NOT AN INTEGER MULTIPLE OF
     190 DEGREES /)
      END
```

```
         IDENT   SPACE
         ENTRY   SPACE
*
 NAME    VFD     42/5LSPACE,18/2
*
 SPACE   DATA    0
         MX0     18                      SET MASK IN X0 18 BITS LEFT JUSTIFIED
         LX0     18                      ROTATE THE MASK TO SET LOWEST 18 BITS
         SA1     67B                     FWA LOADER TO X1
         BX2     X0*X1                   EXTRACT FWA LOADER TO X2
         SA1     X2-2                    FWA OF BLANK COMMON TO X1
         BX6     X0*X1                   EXTRACT FWA BLANK COMMON TO X6
         SA6     B1                      PASS FWA BLANK COMMON TO CALLING PROG
*
         SA1     65B                     ADDRESS OF THE FIRST WORD AVAILABLE
*                                        FOR THE LOADER TO X1 (THIS IS THE
*                                        TOTAL MEMORY AVAILABLE SINCE BLANK
*                                        COMMON FILLS ENTIRE REMAINING SPACE)
         BX6     X1*X0                   TOTAL MEMORY TO X6
         SA6     B2                      PASS TOTAL MEMORY SPACE TO CALLING PR
         EQ      SPACE
*
      END
      SUBROUTINE UOUT (U,NUI,NUJ)
C
C     READS U-MATRIX IN BOTH DATA AND PLOTER UNITS FROM TAPE2 AND
C     UPON REQUEST PRINTS IT OUT AND/OR PUNCHES IT BINARY.
C
      COMMON /SWITCH/ KEYS(5,10),NKEYS,NSW(10)
C
      DIMENSION U(NUI,NUJ)
C
C
      DO 9 ISW=1,2
C
C     READ THE MATRIX
C
1     READ (2) U
C
C     SWALLOW END-OF-FILE MARK
C
      IEOF=0
      CALL DIFFER (8,IEOF,A,I,1,A,1)
      IF (IEOF) 1,2,1
2     IF (NSW(1)) 3,8,3
3     GO TO (4,5), ISW
4     WRITE (6,12)
      GO TO 6
5     WRITE (6,13)
C
6     DO 7 I=1,NUI
      WRITE (6,10) I,NUJ
      WRITE (6,11) (U(I,J),J=1,NUJ)
7     CONTINUE
      IF (NSW(2).EQ.0) GO TO 9
C
C     PUNCH BINARY U-MATRIX
C
8     WRITE (8) NUI,NUJ,U
      END FILE 8
9     CONTINUE
      REWIND 2
      IF (NSW(2).NE.0) WRITE (6,14)
C
      RETURN
C
10    FORMAT (/5X,3HI =,I4,11H,  J = 1 TO,I4/)
11    FORMAT (10E13.5)
12    FORMAT (1H1,5X,17HU-MATRIX UNSCALED/6X,17(1H=)/)
13    FORMAT (1H1,5X,28HU-MATRIX SCALED FOR PLOTTING/6X,28(1H=)/)
14    FORMAT (//5X,65HU-MATRIX PUNCHED IN UNSCALED AND SCALED FORM AS TW
     1O BINARY FILES.)
      END
      FUNCTION U50 (I,J,U,NUI,NUJ)
C
      DIMENSION U(NUI,NUJ)
C
      IF (U(I,J)-50.0) 2,1,1
1     U50=U(I,J)-50.0
      RETURN
C
2     U50=U(I,J)
      RETURN
      END
      SUBROUTINE XINTER (U,NI,NJ,XC,YC,D,A,UTEMP)
C
C     TO COMPLETE PARTIALLY FILLED SURFACE MATRIX U BY THE METHOD
C     OF SPLINE INTERPOLATION.
C     INPUT TO SUBROUTINE INTERP CONSISTS OF FOLLOWING-
C     1. MATRIX U(NI,NJ) WHOSE ELEMENTS ARE PRESET TO VALUE 1HI
C        EXCEPT FOR THE KNOWN ELEMENTS WHICH ARE DEFINED AT
C        GRID POINTS OF A RECTANGULAR MESH.
C     2. NI= NO. OF ROWS IN U, NJ=NO. OF COLUMNS IN U
C     3. XMIN = MINIMUM X1 VALUE
C     4. XMIN = MINIMUM X2 VALUE
C     5. DX   = INCREMENT IN X1 DIRECTION
C     6. DY   = INCREMENT IN X2 DIRECTION
C     THE FOLLOWING ARE WORKING ARRAYS TOGETHER WITH MAX. DIMENSIONS
C     XC(N),YC(N),D(N),A(3,N),UTEMP(N)
C     N= LARGER OF NI OR NJ
C
      DIMENSION U(NI,NJ), XC(1), YC(1), D(1), A(3,1), UTEMP(1)
C
      DATA ROW,COL/4H ROW,4H COL/
      DATA XLETI/1HI/
C
C     SET  LOWER LIMITS AND INCREMENTS
C
      XMIN=1.0
      YMIN=1.0
      DX=1.0
      DY=1.0
C
C     CHECK IF FOUR CORNER POINTS ARE DEFINED
C
      IF (U(1,1).EQ.XLETI) GO TO 22
      IF (U(1,NJ).EQ.XLETI) GO TO 22
      IF (U(NI,1).EQ.XLETI) GO TO 22
      IF (U(NI,NJ).EQ.XLETI) GO TO 22
C
      NI1=NI-1
      NJ1=NJ-1
      KR=NI
      KC=NJ
C
C     COMPLETE END COLUMNS
C
1     CONTINUE
      KROW=NI
      KCOL=NJ
      K3=0
      JJ=1
      DO 5 J=1,2
      DO 2 I=1,NI
      UTEMP(I)=U(I,JJ)
2     CONTINUE
C
      CALL  CHECK (UTEMP,NI,K1,K2,N)
C
      IF  (N.EQ.0)  GO  TO  4
      IF  (K1.GT.0)  GO  TO  10
      IF  (K2.GT.0)  GO  TO  10
C
      CALL  INSERT (XC,YC,D,A,UTEMP,NI,YMIN,DY,COL,JJ,KODE)
C
      IF  (KODE.GT.0)  GO  TO  25
      DO  3  I=1,NI
      U(I,JJ)=UTEMP(I)
3     CONTINUE
4     KCOL=KCOL-1
      JJ=NJ
5     CONTINUE
C
C     FILL  MATRIX  ROW-WISE
C
      DO  9  I=2,NI1
      DO  6  J=1,NJ
      UTEMP(J)=U(I,J)
6     CONTINUE
C
      CALL  CHECK (UTEMP,NJ,K1,K2,N)
C
      IF  (N.EQ.0)  GO  TO  8
      IF  (K1.GT.0)  GO  TO  9
      IF  (K2.GT.0)  GO  TO  9
C
      CALL  INSERT (XC,YC,D,A,UTEMP,NJ,XMIN,DX,ROW,I,KODE)
C
      IF  (KODE.GT.0)  GO  TO  25
      DO  7  J=1,NJ
      U(I,J)=UTEMP(J)
7     CONTINUE
8     KROW=KROW-1
9     CONTINUE
      GO  TO  11
C
C     COMPLETE  END  ROWS
C
10    K3=1
11    II=1
      DO  17  I=1,2
      DO  12  J=1,NJ
      UTEMP(J)=U(II,J)
12    CONTINUE
```

```
C
      CALL CHECK (UTEMP,NJ,K1,K2,N)
C
      IF (N.EQ.0) GO TO 16
      IF (K1.NE.0) GO TO 13
      IF (K2.EQ.0) GO TO 14
13    IF (K3.LE.0) GO TO 17
      GO TO 23
C
14    CALL INSERT (XC,YC,D,A,UTEMP,NJ,XMIN,DX,ROW,II,KODE)
C
      IF (KODE.GT.0) GO TO 25
      DO 15 J=1,NJ
      U(II,J)=UTEMP(J)
15    CONTINUE
16    KROW=KROW-1
      IF (KROW.LE.0) RETURN
17    II=NI
C
C     FILL MATRIX COLUMN-WISE
C
      DO 21 J=2,NJ1
      DO 18 I=1,NI
      UTEMP(I)=U(I,J)
18    CONTINUE
C
      CALL CHECK (UTEMP,NI,K1,K2,N)
C
      IF (N.EQ.0) GO TO 20
      IF (K1.GT.0) GO TO 21
      IF (K2.GT.0) GO TO 21
C
      CALL INSERT (XC,YC,D,A,UTEMP,NI,YMIN,DY,COL,J,KODE)
C
      IF (KODE.GT.0) GO TO 25
      DO 19 I=1,NI
      U(I,J)=UTEMP(I)
19    CONTINUE
20    KCOL=KCOL-1
      IF (KCOL.LE.0) RETURN
21    CONTINUE
      IF (KROW.GE.KC) GO TO 24
      IF (KCOL.GE.KR) GO TO 24
      KR=KROW
      KC=KCOL
      GO TO 1
C
22    WRITE (6,27)
      WRITE (6,28)
      GO TO 25
```

```
23    WRITE (6,29)                                                    AC 151
      WRITE (6,28)                                                    AC 152
      GO TO 25                                                        AC 153
24    WRITE (6,30)                                                    AC 154
      WRITE (6,28)                                                    AC 155
25    WRITE (6,32)                                                    AC 156
      WRITE (6,31)                                                    AC 157
      DO 26 I=1,NI                                                    AC 158
      WRITE (6,33) I                                                  AC 159
      WRITE (6,34)  (U(I,J),J=1,NJ)                                   AC 160
26    CONTINUE                                                        AC 161
      STOP 22                                                         AC 162
C                                                                     AC 163
C                                                                     AC 164
C                                                                     AC 165
27    FORMAT (7X,30HNOT ALL CORNER POINTS DEFINED )                   AC 166
28    FORMAT (7X,68HCOMPLETION OF SURFACE MATRIX ELEMENTS NOT POSSIBLE B AC 167
     1Y INTERPOLATION )                                               AC 168
29    FORMAT (7X,45HTOO MANY BOUNDARY POINTS CLUSTERED TOGETHER  )    AC 169
30    FORMAT (7X,41HTOO FEW POINTS DEFINED PER ROW OR COLUMN )        AC 170
31    FORMAT (/7X,30HSURFACE MATRIX AT TERMINATION /)                 AC 171
32    FORMAT (//3X,39H*****ILLCONDITIONS FOR INTERPOLATION    )       AC 172
33    FORMAT (/3X,5HROW =,I4/)                                        AC 173
34    FORMAT (10E13.5)                                                AC 174
      END                                                             AC 175-
      SUBROUTINE CHECK (UTEMP,NC,K1,K2,N)
C
C     CHECKS THE FOLLOWING CRITERIA TO BE SATISFIED FOR INTERPOLATION
C     TO BE POSSIBLE-
C     1. NOT LESS THAN PC1 PERCENT OF POINTS PER ROW OR COLUMN MUST
C        BE DEFINED
C     2. NOT MORE THAN PC2 PERCENT OF UNDEFINED POINTS MUST BE
C        CLUSTERED TOGETHER.
C
C     K1=0  CRITERION 1 SATISFIED , K1=1 CRITERION 1 VIOLATED
C     K2=0  CRITERION 2 SATISFIED , K2=1 CRITERION 2 VIOLATED
C     N= NUMBER OF UNDEFINED POINTS PER ROW (OR COLUMN)
C
      DIMENSION UTEMP(1)
C
      DATA XLETI/1HI/
C
      PC2=0.40
      PC1=0.50
C
      K1=0
      K2=0
      N=0
      JJ=0
      NN=0
      DO 1  J=1,NC
      IF  (UTEMP(J).NE.XLETI)  GO TO 1
      IF  (JJ.NE.(J-1))  NN=0
      JJ=J
      N=N+1
      NN=NN+1
      IF  (FLOAT(NN)/FLOAT(NC).GT.PC2)  GO TO 2
1     CONTINUE
      IF  (FLOAT(N)/FLOAT(NC).GT.PC1)  K1=1
      RETURN
2     K2=1
      RETURN
      END
      SUBROUTINE DIAG3 (A,X,N)
C
C      SOLUTION OF TRIDIAGONAL SYSTEM OF EQUATIONS
C
      DIMENSION A(3,1), XQ(1)
C
      DO 1  J=2,N
      Z=-A(1,J)/A(2,J-1)
      A(1,J)=0.
      IF  (Z.EQ.0.)  GO TO 1
      A(2,J)=A(2,J)+Z*A(3,J-1)
      XQ(J)=XQ(J)+Z*XQ(J-1)
1     CONTINUE
      JJ=N+1
      DO 2  J=1,N
      JJ=JJ-1
      IF  (JJ.EQ.N)  GO TO 2
      XQ(JJ)=XQ(JJ)-A(3,JJ)*XQ(JJ+1)
2     XQ(JJ)=XQ(JJ)/A(2,JJ)
      RETURN
      END
      SUBROUTINE INSERT (XC,YC,D,B,UTEMP,NC,AMIN,DA,ALPHA,JA,KODE)
C
C     COLUMN-WISE OR ROW-WISE INTERPOLATION
C     NST= FIRST DEFINED POINT RELATIVE TO POINTS IN ARRAY UTEMP
C     NEN=  LAST DEFINED POINT RELATIVE TO POINTS IN ARRAY UTEMP
C
      DIMENSION XC(1), YC(1), UTEMP(1), B(3,1), D(1)
C
      DATA XLETI/1HI/
C
      A=AMIN
      KODE=0
      NC2=NC-2
      N=0
      DO 1 J=1,NC
      A=A+DA
      IF (UTEMP(J).EQ.XLETI) GO TO 1
      JJ=J
      N=N+1
      IF (N.EQ.1) NST=JJ
      YC(N)=UTEMP(J)
      XC(N)=A
1     CONTINUE
      NEN=JJ
      IF (NST.EQ.1) GO TO 2
      IF (NST-3) 2,2,5
2     IF (NEN.EQ.NC) GO TO 3
      IF (NEN-NC2) 5,3,3
3     CALL SPLIT (N,XC,YC,D,B)
      A=AMIN
      DO 4 J=1,NC
      A=A+DA
      IF (UTEMP(J).NE.XLETI) GO TO 4
      UTEMP(J)=SPLINE(A,N,XC,YC,D)
4     CONTINUE
      RETURN
C
5     KODE=1
      RETURN
C
      END
      FUNCTION SPLINE (XP,N,X,Y,D)
C
C     CUBIC SPLINE INTERPOLATION
C
      DIMENSION XQ(1), YQ(1), D(1)
C
      N1=N-1
      DO 1 I=1,N1
      IF (XP.LT.XQ(I)) GO TO 1
      IF (XP.GT.XQ(I+1)) GO TO 1
      J=I
      GO TO 2
1     CONTINUE
2     CONTINUE
      IF (XP.LT.XQ(1)) J=1
      IF (J.GT.XQ(N)) J=N-1
C
      H=XQ(J+1)-XQ(J)
      CC=(YQ(J+1)-YQ(J))/H-H*(D(J+1)-D(J))/6.
      DD=YQ(J)-D(J)*H*H/6.
      A=(XQ(J+1)-XP)**3/6./H
      B=(XP-XQ(J))**3/6./H
      C=(XP-XQ(J))
      SPLINE=D(J)*A+D(J+1)*B+CC*C+DD
      RETURN
      END
      SUBROUTINE SPLIT (N,X,Y,D,A)
C
C     TO INITIALIZE. CALCULATES SECOND DERIVATIVES
C
      DIMENSION XQ(1), YQ(1), D(1), A(3,1)
C
      DEL=0.5
      A(1,1)=0.
      A(2,1)=2.0
      A(3,1)=-2.*DEL
      D(1)=0.
      NN=N-1
      DO 1  J=2,NN
      H1=XQ(J)-XQ(J-1)
      H2=XQ(J+1)-XQ(J)
      F2=(YQ(J+1)-YQ(J))/H2
      F1=(YQ(J)-YQ(J-1))/H1
      D(J)=6.*(F2-F1)/(H1+H2)
      AL=H2/(H1+H2)
      AU=1.-AL
      A(1,J)=AU
      A(2,J)=2.
      A(3,J)=AL
1     CONTINUE
      A(1,N)=-2.*DEL
      A(2,N)=2.0
      A(3,N)=0.
      D(N)=0.
      CALL DIAG3 (A,D,N)
      RETURN
      END
      BLOCK DATA
C
      COMMON /DATA/ INTX1,INTX2,ISIZE,JSIZE,NCONS,IQUAL,KLASS,ALPHA,ANGL
     1E,PAPER,VSIZE,XLENG,XYROT,X1MAX,X1MIN,X2MAX,X2MIN
      COMMON /A/ PHI(20),PSI(5),KODE(5)
      COMMON /B/ DIG(5)
      COMMON /BLOCK1/ DEL1,DEL2,DX,DY,D1,D2,IART3,IBL,IC(80),ICOM,IDIG(1
     10),IEQ,IK,IMAX,IMIN,INCOIK,INCOJK,INT,INTNUM,IPEN,IPER,IPLUS,ISKIP
     2,IT,I23,I3,JK,JMAX,JMIN,JT,JUMP,J3,KE,KEY(5,20),MINUS,MODE,NCOMP,N
     3CORN,NELEV,NINCO,NKEY,NROT,NUP,NX1,NX2,NX3,NRENUM,UF(130),ULAST,UM
     4AX,UMIN,XN,X1,X2,YN,YY1,YY2,Y1,Y2
      COMMON /SWITCH/ KEYS(5,10),NKEYS,NSW(10)
C
      DATA  KODE/4HMCM/,4HPP27,4H2-30,4H-PAC,4HKAGE/
      DATA  IART3,INCOJK,INTNUM,ISKIP,NINCO,NKEY,NRENUM/0,1,7,0,0,18,10/
      DATA  INTX1,INTX2,ISIZE,JSIZE,NCONS,IQUAL,KLASS/50,50,51,51,0,0,1/
      DATA  ALPHA,ANGLE,PAPER,VSIZE,XLENG,XYROT,X1MAX,X1MIN,X2MAX,X2MIN/2
     10.0,1HI,2*10.0,7.5,5*0.0/
      DATA  IBL,IEQ,IPLUS,MINUS,IPER,ICOM,KE/1H ,1H=,1H+,1H-,1H.,1H,,1HE/
      DATA  IDIG/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
      DATA  DIG/1H1,1H2,1H3,1H4,1H5/
      DATA  NX3/0/
      DATA  NROT,MODE/2,2/
      DATA  NKEYS,NSW/5,10*0/
      DATA  KEY/1HI,1HN,1HT,1HX,1H1,1HI,1HN,1HT,1HX,1H2,1HI,1HS,1HI,1HZ,1
     1HE,1HJ,1HS,1HI,1HZ,1HE,1HN,1HC,1HO,1HN,1HS,1HI,1HQ,1HU,1HA,1HK
     2,1HL,1HA,1HS,1HS,1HA,1HL,1HP,1HH,1HA,1HA,1HN,1HG,1HL,1HE,1HP,1HA,1
     3HP,1HE,1HR,1HV,1HS,1HI,1HZ,1HE,1HX,1HL,1HE,1HN,1HG,1HX,1HY,1HR,1HO
     4,1HT,1HX,1H1,1HM,1HA,1HX,1HX,1H1,1HM,1HI,1HN,1HX,1H2,1HM,1HA,1HX,1
     5HX,1H2,1HM,1HI,1HN,15*1H$/
      DATA  KEYS/1HP,1HR,1HI,1HN,1HT,1HP,1HU,1HN,1HC,1HH,1HN,1HO,1HP,1HL,
     11HO,1HN,1HO,1HL,1HA,1HB,1HE,1HN,1HD,1HD,1HA,25*1H$/
C
      END
```