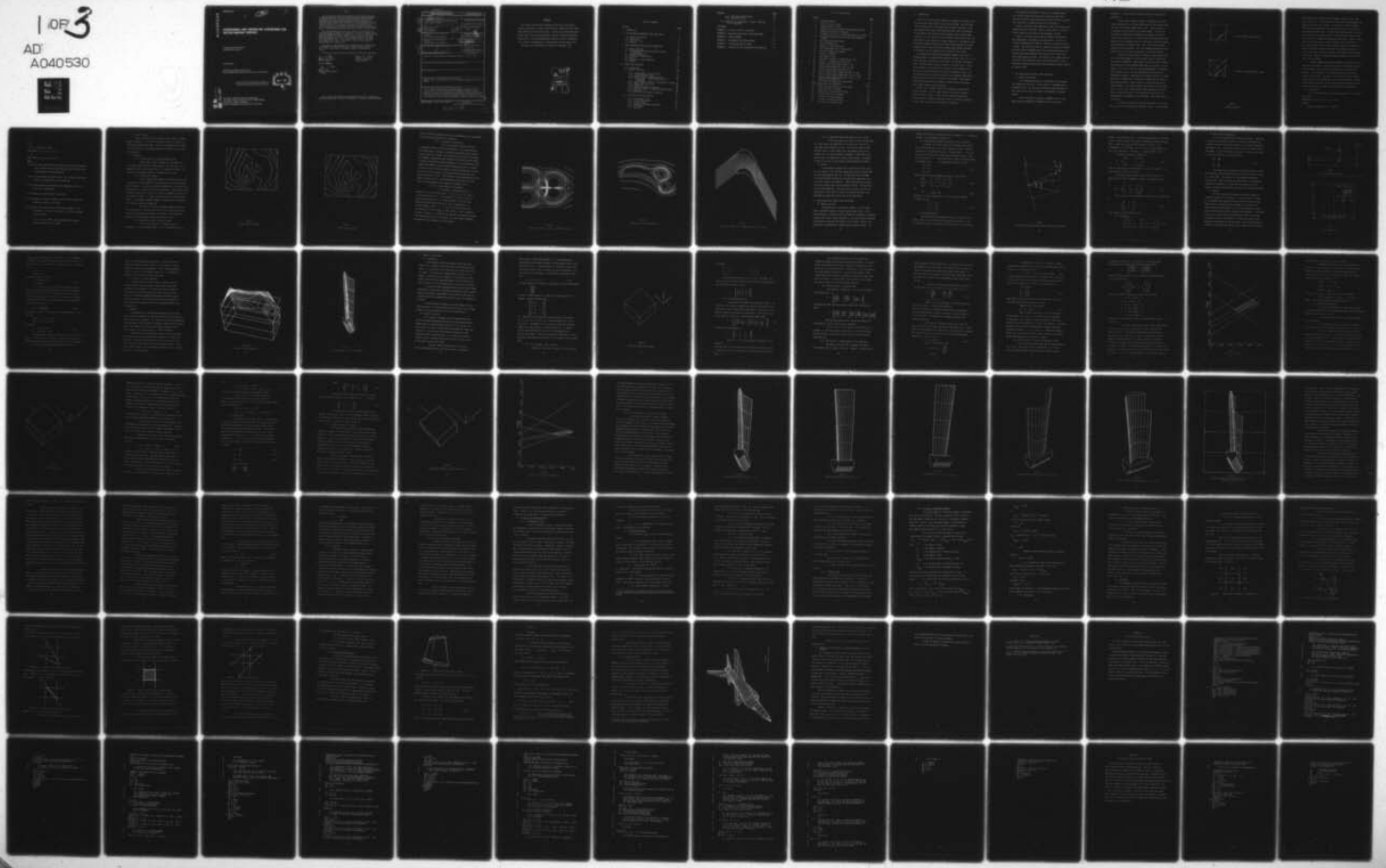
AD-A040 530 AIR FORCE AERO PROPULSION LAB WRIGHT-PATTERSON AFB OHIO F/G 12/1
CONTOURING AND HIDDEN-LINE ALGORITHMS FOR VECTOR GRAPHIC DISPLA--ETC(U)
JAN 77 J S PETTY, K D MACH

UNCLASSIFIED AFAPL-TR-77-3 NL

1 OF 3
AD A040530



AFAPL-TR-77-3

(12)

CONTOURING AND HIDDEN-LINE ALGORITHMS FOR VECTOR GRAPHIC DISPLAYS

TURBINE ENGINE DIVISION
COMPONENTS BRANCH

JANUARY 1977

TECHNICAL REPORT AFAPL-TR-77-3
FINAL REPORT FOR PERIOD AUGUST 1975 to AUGUST 1976

Approved for public release; distribution unlimited

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

NOTICE

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

This report contains the results of an effort to develop simple and effective methods for graphically presenting the results of turbine engine design and performance analysis computations. The work was performed in the Components Branch of the Turbine Engine Division of the Air Force Aero-Propulsion Laboratory, Air Force Systems Command, Wright-Patterson AFB, Ohio, under Project 3066, Task 06 and Work Units 02 and 27. The effort was conducted by Dr. James S. Petty and Dr. Kervyn D. Mach during the period August 1975 to August 1976.

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

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

James S Petty
JAMES S. PETTY, GS-13
Project Engineer

Kervyn D Mach
KERVYN D. MACH, GS-13
Project Engineer

FOR THE COMMANDER

Wayne A Tall
WAYNE A. TALL
Acting Tech Area Manager,
Turbines

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

AIR FORCE - 18 APRIL 77 - 100

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: (14) AFAPL-TR-77-3
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

4. TITLE *(and Subtitle)*: (6) Contouring and Hidden-Line Algorithms for Vector Graphic Displays.
5. TYPE OF REPORT & PERIOD COVERED: Interim, Aug 75 to Aug 76
6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s): (10) James S. Petty, Kervyn D. Mach (17) 06
8. CONTRACT OR GRANT NUMBER(s): In-House (16)

9. PERFORMING ORGANIZATION NAME AND ADDRESS: AFAPL/TBC, Wright-Patterson AFB OH 45433
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: P.E. 62203F, Proj 3066, Task 06, Work Units 02 and 27

11. CONTROLLING OFFICE NAME AND ADDRESS: (11) AFAPL/TBC, Wright-Patterson AFB OH 45433
12. REPORT DATE: January 1977
13. NUMBER OF PAGES: 189

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: (9) Final rept. Aug 75 - Aug 76,
15. SECURITY CLASS. *(of this report)*: Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT *(of this Report)*: (12) 195p.

Approved for Public Release; Distribution Unlimited.

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

18. SUPPLEMENTARY NOTES

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

Computer Graphics, Contour Plotting, Three-Dimensional Graphics, Hidden-Line Removal

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

This report describes several computer graphics techniques developed by the authors to display the results of turbomachinery design and performance calculations. It begins with basic contour plotting, then shows the procedures for drawing three-dimensional figures with or without perspective and concludes with descriptions of two methods of removing hidden lines from three-dimensional scenes. Numerous examples and program listings are included.

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

011570 4B

FOREWORD

This report describes work conducted within the Air Force Aero-Propulsion Laboratory, Turbine Engine Division, Components Branch (TBC), Wright-Patterson Air Force Base, Ohio. The work was accomplished under Project 3066, "Gas Turbine Technology," Task 06, "Turbine Technology," Work Units 02, "Turbine Aeromechanical Analysis," and 27, "Computation of 3-D Flows in Turbomachinery," between August 1975 and August 1976.

This report was submitted by the authors in September 1976.

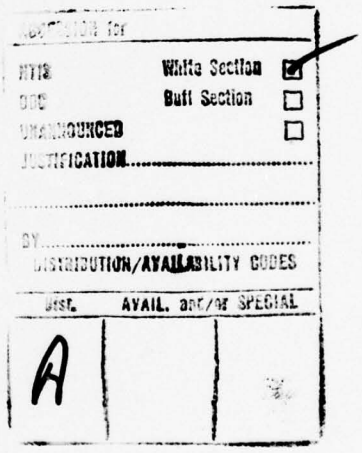


TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. INTRODUCTION

The utility of the digital computer for scientific calculation and the human labor saved through its use are well known. Not long after digital computers came into popular use for number crunching, it became apparent that a great deal more human labor could be eliminated if the computer could generate graphic output as well as the normal printout; it is now unusual to find a computer installation of any size which does not have some sort of graphic capability. The advent of the interactive graphics terminal which allows rapid generation of graphs, pictures, and in some cases moving pictures added another increment of flexibility and has indeed become a technical field itself.

The software provided with most computer graphics facilities is designed primarily to generate the familiar cartesian graphics of Y vs X, since most data is amenable to this format. Not all data, of course, fits. One often wants to draw a picture of a surface, say $Z = f(x,y)$, or he may wish to draw contours of constant Z on the surface or he may wish to view the surface from different viewpoints or both. One also wishes at times to draw a picture of a solid object either with or without the details of the side away from the viewer. Most graphics installations have available software to perform some of the above tasks though probably very few have all.

We offer in this report a selection of graphics software which we have developed. In most cases, program listings are included, along with sample results. Except for the contouring algorithms, we have included a technical discussion which is sufficiently detailed for the reader to write his own software if necessary.

The contouring algorithms in Section 2 are complete except for a driver program and have proven so versatile that there has been no need to change them since they were originally written. Their operation is completely described in Reference 1.

Section 3 addresses three-dimensional graphics with perspective, emphasizing basic principles. Only a simple program is listed in order to show one application of the procedures involved.

Finally, in Section 4 we present two hidden line algorithms and compare their performance on the same object. The first is a direct implementation of Roberts' algorithm; a program listing is included. The second can trace its ancestry to Warnock's algorithm. We found a direct implementation to be unsatisfactory because the original algorithm subdivides the picture until it finds a sub-scene simple enough to handle. The resulting line lengths were all too often at the resolution limit of the plotting device, which therefore increased the execution time tremendously besides yielding an untidy picture.

2. The Contouring Subroutines LEVEL1 and LEVEL2

2.1 General Description

It is sometimes necessary or convenient to plot contours of a function of the form $Z = f(x,y)$ where Z is tabulated over a rectangular field. The two sets of subroutines described here are fast, easy to use, and do not require large amounts of computer storage.

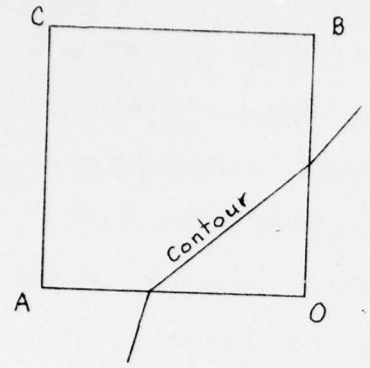
LEVEL1 and LEVEL2 are basically FORTRAN versions of the ALGOL procedures CONTOUR1 and CONTOUR2 which are described in

Reference 1. Certain features have been added to increase their generality.

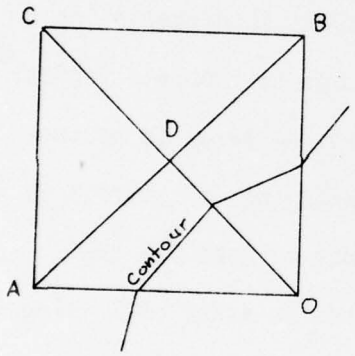
LEVEL1 traces contours through a rectangular mesh, using linear interpolation to find the intersections of the contours with the sides of each mesh element (See Fig 1a). LEVEL2 constructs a fifth point at the center of each mesh element. The height of this point is defined as the average of the heights of the four corner points (See Fig 1b). It then finds the intersections of the contour with the diagonals as well as the sides of the mesh element and thus produces a smoother curve. This is the only difference between the two subroutines. For a more thorough discussion of the internal logic, the reader is referred to the original report.

Usage of the two routines is identical, requiring only one call with eight arguments to obtain contour plots in either polar or cartesian coordinates. Both require the user to supply a logical array which has the same FORTRAN dimensions as the array to be contoured. This array is used for internal bookkeeping; if storage is short, another array in the calling program may be used via the EQUIVALENCE statement. Additionally, one needs an array of X values which must be monotonically increasing or decreasing, but need not be uniformly spaced; a similar array of Y values; and an array of contour heights. The contour heights need not be in any particular order. There is also no need to worry about requesting a contour above or below the Z array; if the subroutines do not find any intersections corresponding to a given contour, they simply move on to the next.

The general operation of the two subroutines is as follows: The array of contour heights is processed in order. For each height,



a. Basic rectangle used by LEVEL1



b. Augmented rectangle used by LEVEL2

Figure 1

Contouring Meshes

the subroutine scans along the mesh boundary, starting at the lower left corner, until it finds an intersection with the high ground on the right as one faces into the mesh. It follows this contour through the mesh, always keeping the high ground on the right, until the contour emerges from the mesh. It then resumes scanning where it left off and continues in this manner until all contours of the current height which intersect the edges of the mesh have been found. Because it must have the high ground on the right, it cannot "refind" contours it has already traced. When the open contours (those which intersect the edges of the mesh) have been processed, the subroutines search the interior for closed contours or islands, again keeping the high ground on the right. When all contours have been processed, control returns to the calling program.

Neither subroutine draws the boundaries of the mesh, nor do they label the contours. The user may not want the boundaries drawn or the boundaries may not describe a rectangle, as in some of the examples. If the boundary is a rectangle, one can easily draw it in the calling program. Similarly, it is difficult to place numeric contour labels on a plot where they will not be overwritten. The user usually knows enough about the data being contoured to mark the first and last points of each contour, as shown in one of the examples.

2.2 Application

The calling program must provide the arrays X, Y, Z, U, and H. For example,

```
DIMENSION X(i), Y(j), Z(i,j), U(i,j), H(k)
LOGICAL U
. . . (Statements defining X, Y, Z, and H)
```

I = i

J = j

K = k

U(1,1) = .FALSE. (or .TRUE.)

CALL LEVEL1 (X, Y, Z, H, U, I, J, K)

or

CALL LEVEL2 (X, Y, Z, H, U, I, J, K)

Where

X and Y are arrays describing the coordinate grid over the rectangular mesh. Uniform spacing is not necessary, but the arrays must be in ascending or descending order.

i and j are the number of elements in the X and Y arrays, respectively. There is no restriction on their magnitude.

Z is an array containing the heights of the dependent variable at each (X,Y) intersection.

H is the array of contour heights to be plotted.

K is the number of contours. Neither K nor the order in which the heights are given is restricted.

U is a logical array which, except for the first element, is set internally. If U(1,1) is set .FALSE., X and Y are assumed to be in inches.

If U(1,1) is set .TRUE., polar coordinates are assumed, with X in inches and Y in radians.

2.3 Source Listings

Source listings of LEVEL1, FOLLOW, LEVEL2, PURSUE, and DRAW appear in Appendix 4. LEVEL1 calls FOLLOW to trace the contours and FOLLOW calls DRAW to draw them. LEVEL2 calls PURSUE which calls DRAW. Most modifications of the package take place in DRAW, as will be seen in the examples.

2.4 Examples

2.4.1 Contouring Over a Simple Rectangular Array

Program ONE, listed in Appendix 4, reproduces the two example figures from the original report. The first (Fig. 2) is drawn with LEVEL1 and the second (Fig. 3) is drawn with LEVEL2. The effects of the different methods of interpolation are clear.

2.4.2 Labeling the Contours

One can identify the contours by printing out the X and Y arguments from DRAW; however, for large meshes this can produce many pages of output. A usually satisfactory alternative is to mark the first and last point of each contour with a unique symbol. This is easily done by adding a common block to LEVEL1 (or LEVEL2) and DRAW to pass the contour counter to DRAW. The counter can then be fed to a symbol drawing routine. The CALCOMP subroutine SYMBOL, for example, has a repertoire of 15 symbols, numbered 0 through 14.

The listings of LEVEL1 and DRAW in Appendix 4 show how this might be done. One might wish to make it an option and pass the controlling information through the array UNUSED, since none of the points on the mesh boundary (except the first) are used.

2.4.3 Contouring in Other Coordinate Systems

The only requirement here is that the mesh be rectangular in some coordinate system. Then a transformation can be

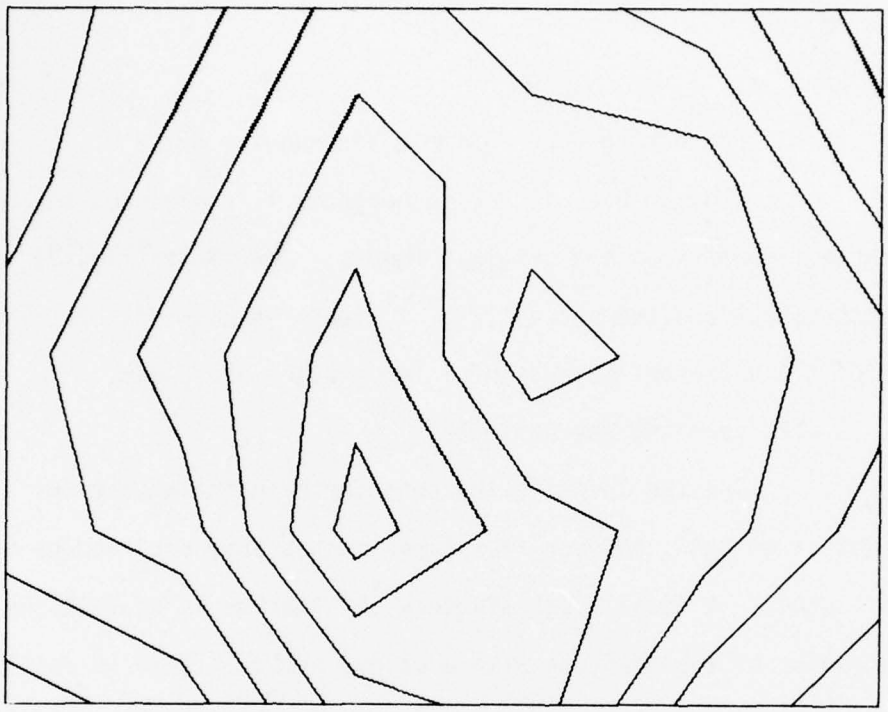


Figure 2

Contours Drawn by LEVEL1

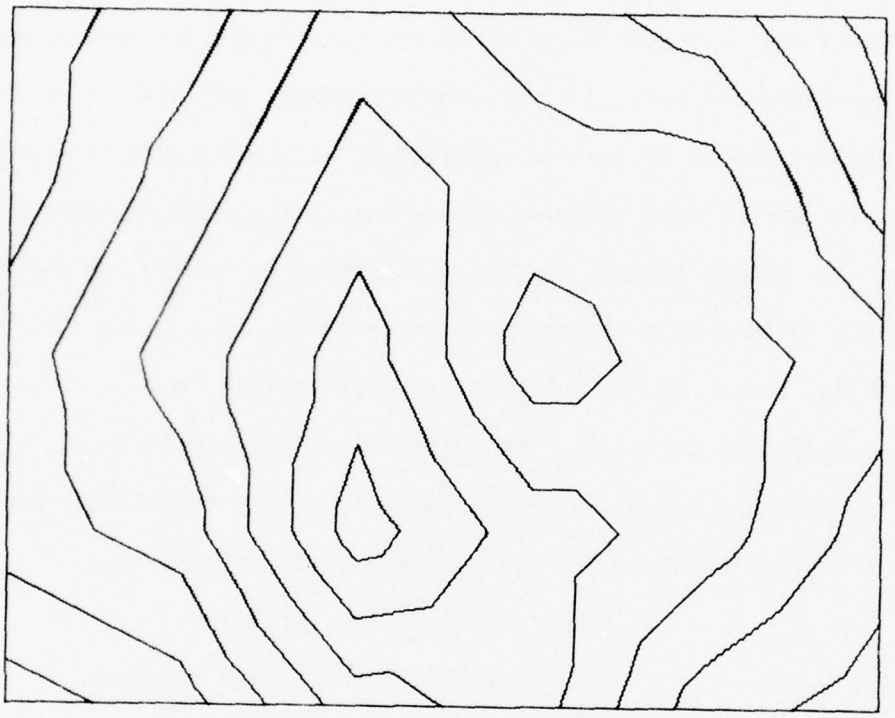


Figure 3

Contours Drawn by LEVEL2

built into DRAW to compute and plot the corresponding (X,Y) coordinates. The following two examples will illustrate.

2.4.3.1 A Joukowski Transformation

This is the familiar cylinder-to-airfoil transformation from classical fluid mechanics. The stream function was defined over a rectangle in cartesian coordinates and the streamlines (contours of constant stream function) were traced in this space. The Joukowski transformation was applied to each point passed to DRAW to draw the airfoil and selected streamlines. The variable RAM in the common block TRFORM is used to suppress drawing any part of a streamline which falls inside the cylinder in cartesian coordinates (If these are left in, there results a very messy plot because they map into points outside the airfoil). The cylinder and corresponding airfoil are shown in Figures 4 and 5. The protuberances at the two-o'clock and four-o'clock positions on the cylinder are standing vortices.

2.4.3.2 A User-Defined Transformation

In this example, the stream function is defined in (X,η) coordinates, where $\eta = (Y-Y_L)/(Y_U-Y_L)$ and Y_U and Y_L are tabular functions of X. The streamlines are traced in the X,η plane and when a point (X,η) is passed to DRAW, it uses the interpolating function ATKN to find the Y_L and Y_U corresponding to X. It then computes Y from $Y = Y_L + \eta\,(Y_U-Y_L)$ and displays the point. The variable SY in the listing is a scale factor. The finished plot is shown in Figure 6. The airfoils and upstream and downstream straight lines were drawn by the calling program. The polar option in DRAW was not needed for this problem, so it was omitted.

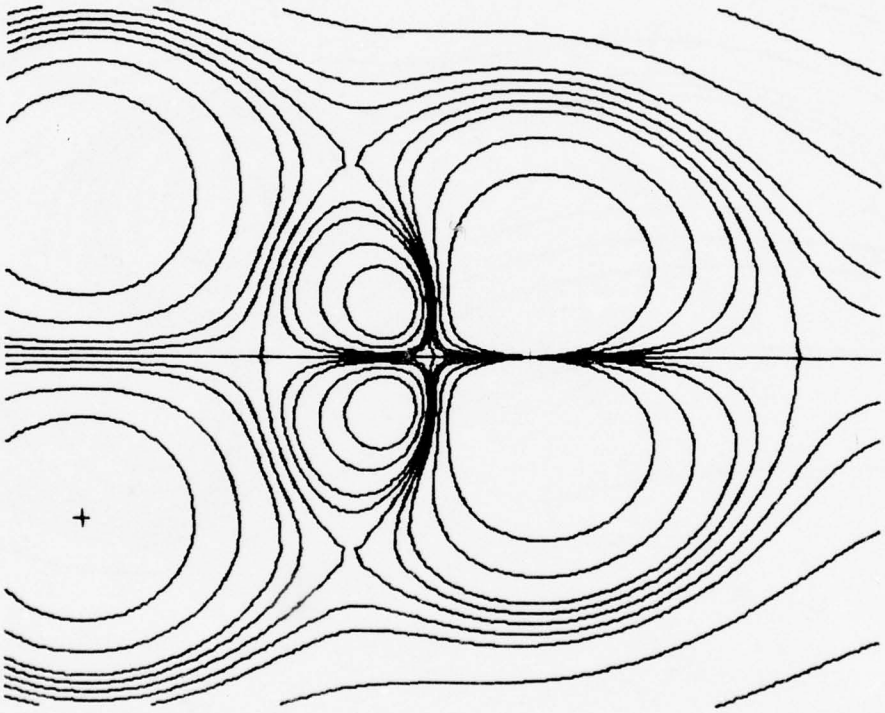


Figure 4

Streamlines Around a Cylinder with Standing Vortices

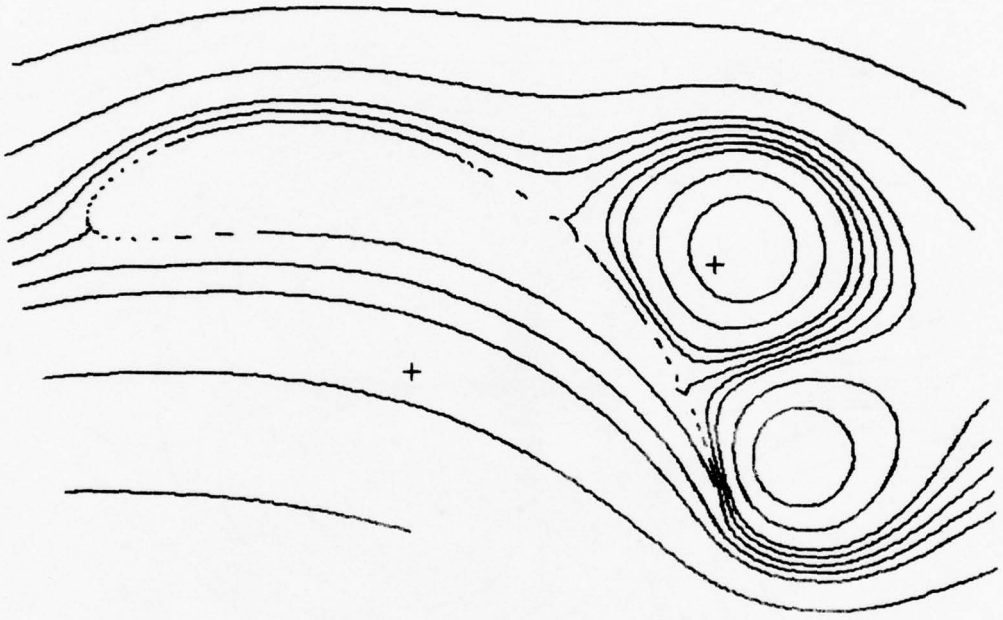


Figure 5

Joukowski Airfoil with Separation

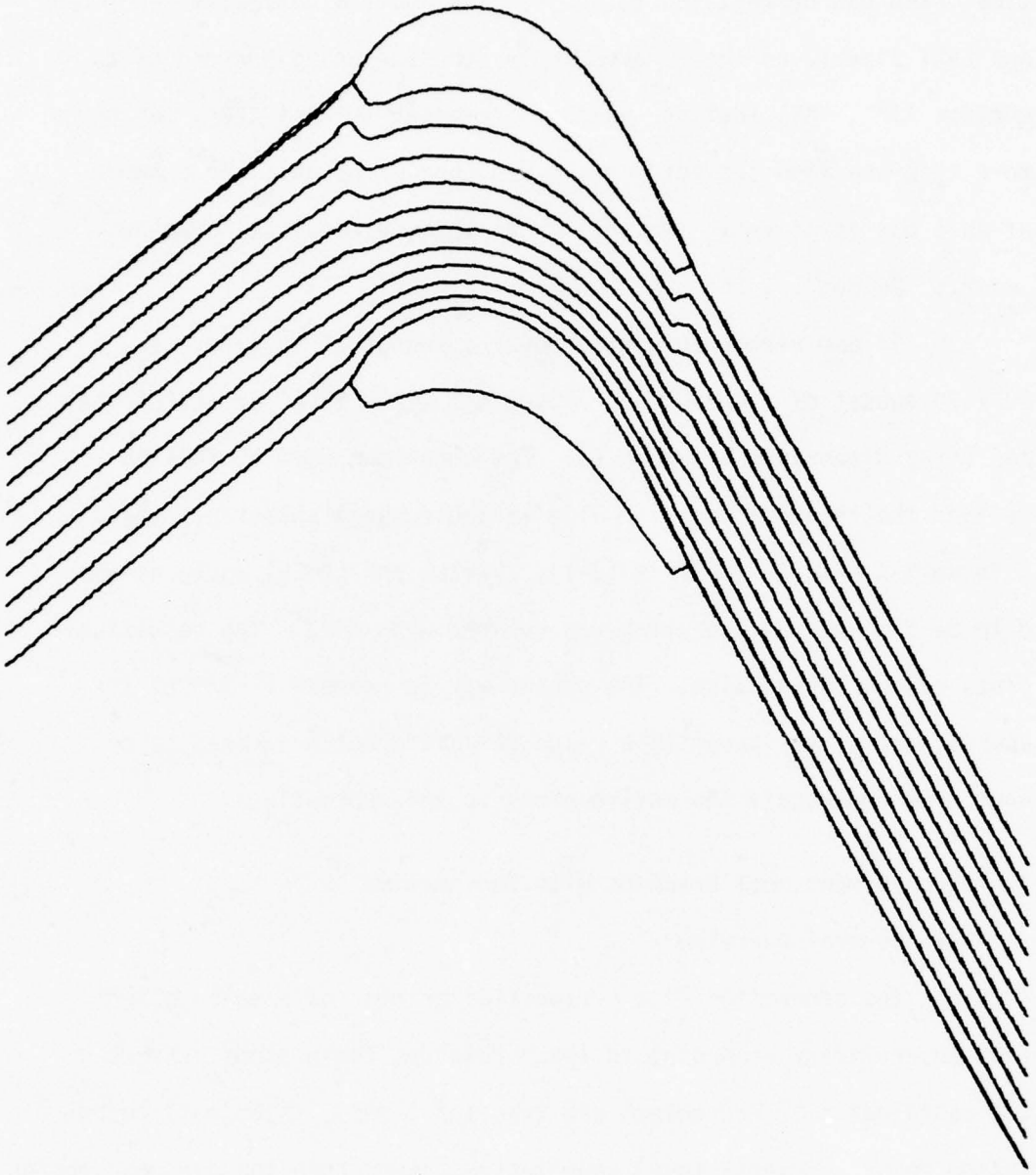


Figure 6

Streamlines Traced in X,η Plane and Drawn in X,Y Plane

2.4.4 Accumulating the Contour Points as an X-Y Array.

If for some reason, one wishes to store all the points in a given contour and draw them all at once rather than one at a time, DRAW can be modified to do so. The logical variables FIRST and LAST signal, as their names imply, the beginning and end of a contour line. All that one needs to remember is that there may be more than one line segment for a given contour height. An example of this use is given in the section on three-dimensional graphics.

2.5 Do Not . . .

If one wishes to contour over a subset of an array, say a 20 x 20 subset of a 40 x 20 array, do not tell LEVEL1 or LEVEL2 that the array dimensions are 20 x 20. The algorithm used by FORTRAN to find the relative location of a variable whose subscripts are I, J in an M x N array is I-1+Mx(J-1). LEVEL1 and LEVEL2 would assume M to be 20, whereas the array was defined with M=40. The resulting plots can be interesting. The proper way to proceed is to set the unused part of the array to a value far different from that to be contoured and supply the entire array to the subroutines.

3. Three-Dimensional Graphics with Perspective

3.1 General Overview

The generation of a perspective drawing of a solid object from an arbitrary viewpoint in space requires three steps. First, the coordinates of the object are translated from its(object) coordinate system to the viewer's (eye) coordinate system, then the eye coordinates are rotated to align with the viewer's line of sight. Finally, the perspective transformation is applied and the object is drawn. The

hidden line problem is not addressed here, although this is a necessary prelude. It is presented in Section 4.

3.2 Translation from Object Coordinates to Eye Coordinates

We assume, for reasons which will be apparent, the object to be located near the origin of its coordinate system. Then the translation is only mildly intricate. Consider Figure 7. The viewpoint (Xv,Yv,Zv) specified in the object coordinate system is taken as the origin of the eye coordinate system, in which coordinate points are denoted as (Xe,Ye,Ze). By inspection of Figure 7, we see

$$\begin{aligned} Xe &= Xv-X \\ Ye &= Z-Zv \\ Ze &= Yv-Y \end{aligned} \tag{3-1}$$

These relations can be expressed more neatly in matrix form:

$$\begin{pmatrix} Xe \\ Ye \\ Ze \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} X - Xv \\ Y - Yv \\ Z - Zv \end{pmatrix} \tag{3-2}$$

Or

$$\vec{Xe} = [F](\vec{X} - \vec{Xv}) \tag{3-2a}$$

Equation (2) holds for positive Yv. If Yv is zero or negative, [F] takes the form

$$F = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{3-3}$$

3.3 Coordinate Rotation

Now we rotate the eye coordinate system so that the Ze axis points at the origin of the object coordinate system (which is why we assumed the object to be located near the origin of its coordinate

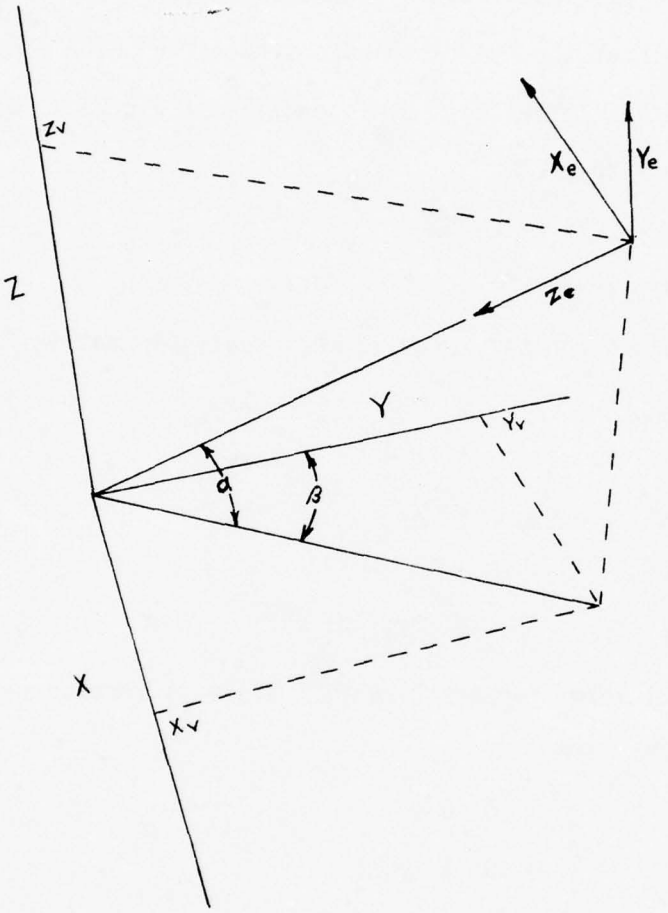


Figure 7

The Object Coordinate System and The Eye Coordinate System

system). This takes two steps -- a rotation through angle β about the Ye axis so that Ze points at (0,0,Zv), followed by a rotation through angle α about the Xe axis so that Ze points at (0,0,0). From trigonometry, $\beta = -\tan^{-1}(\frac{Xv}{Yv})$. If Yv is zero, $\beta = \frac{\pi}{2}$ for positive Xv and $-\frac{\pi}{2}$ for negative Xv. Similarly, $\alpha = -\tan^{-1}[Zv/\sqrt{(Xv^2 + Yv^2)}]$

The matrix for the first rotation is

$$R_1 = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \qquad (3\text{-}4)$$

After this rotation, the Xe component of (0,0,Zv) will be null. In eye coordinates (0,0,Zv) is (Xv,0,Yv).

The second rotation operates on the result of the first rotation to null the Ye component of $[R_1]$Xe. The rotation matrix is

$$R_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \qquad (3\text{-}5)$$

If we denote the translation matrix (3-2) and (3-3) symbolically as

$$F = \begin{pmatrix} f & 0 & 0 \\ 0 & 0 & 1 \\ 0 & f & 0 \end{pmatrix} \qquad (3\text{-}6)$$

we can combine all three matrices, thus:

$$[T] = [R_2][R_1][F]$$

$$= \begin{pmatrix} f\cos\beta & f\sin\beta & 0 \\ f\sin\alpha\sin & -f\sin\alpha\cos\beta & \cos\alpha \\ -f\sin\beta\cos\alpha & f\cos\alpha\cos\beta & \sin\alpha \end{pmatrix} \qquad (3\text{-}7)$$

3.4 Perspective Transformation

This is the simplest part of the entire process. Referring to Figure 8, we consider the point P (Xe,Ye,Ze) on the object and treat the screen (or the plotting surface) as a window through which the object is seen. Assume the viewer's eye to be a distance b from the window. By similar triangles, the point P'(Xs,Ys) on the screen is related to the point P on the object via

$$\frac{Xs}{b} = \frac{Xe}{Ze}$$

$$\frac{Ys}{b} = \frac{Ye}{Ze} \qquad (3\text{-}8)$$

The essence then of generating a perspective image is simply to divide the Xe and Ye coordinates by their corresponding Ze values; i.e., their depth. In practice, this can generate unacceptably small pictures. More satisfactory pictures result if we assume the distance b in equations (8) to be unity and scale the Xs and Ys values to give the largest possible picture within a specified frame.

3.5 Scaling

The final step before displaying the picture, whether on a terminal screen or an X-Y plotter, is to scale the Xs, Ys values so as to produce the largest picture which will fit inside a given frame and also to have the picture centered in the frame. This requires that the entire object (all its xyz coordinates) be translated, rotated, and converted to perspective coordinates. (If multiple views are to be drawn, the new coordinates will have to be stored in separate arrays or the originals will have to be kept on a mass storage file and reread for each new view). The new coordinates are scanned for the maximum and minimum X and Y values, which are used to

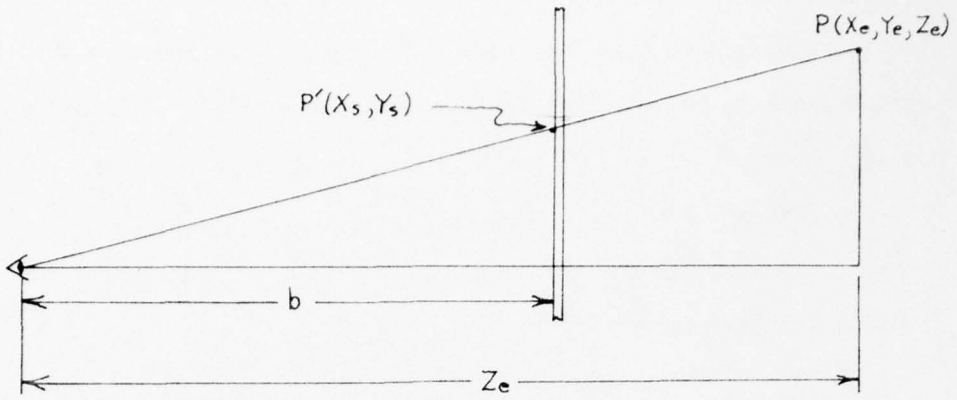


Figure 8

The Screen Coordinate System

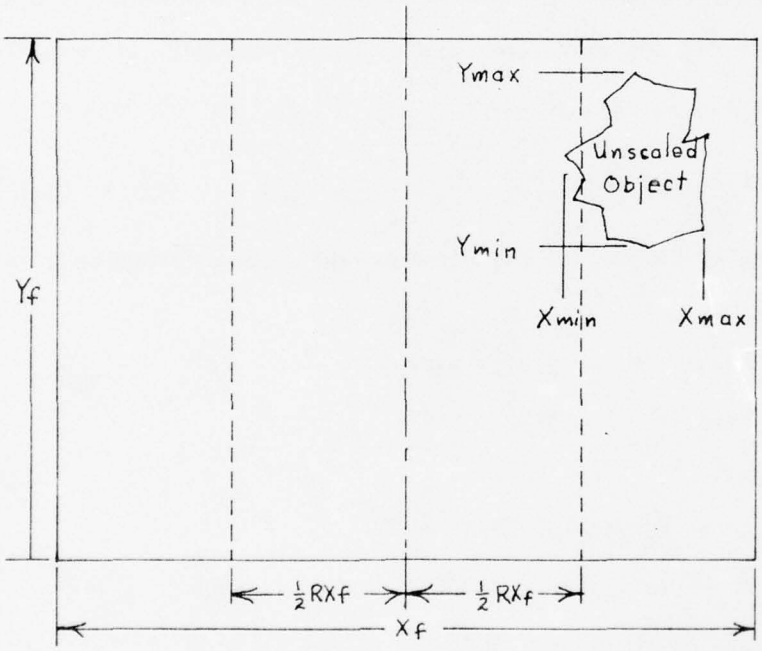


Figure 9

Scaling Nomenclature

compute linear transformations of the form Y' = Ay + B. Referring to Figure 9, we see that the frame runs from 0 to XF and 0 to YF. We want A and B to be such that Y'min is 0 and Y'max is YF. Accordingly, we set

$$AYmin + B = 0$$

$$AYmax + B = YF$$

Solving simultaneously yields

$$A = YF/(Ymax-Ymin)$$
$$B = -A*Ymin \qquad (3\text{-}9)$$

The X direction must be scaled by the same amount, but we can't use the same transformation because the resulting picture may be shifted too far left or right. Therefore, we define a subarea of the frame whose width is RXF (See Figure 9) and which is centered so that it extends RXF/2 left and right of the vertical centerline. The width RXF is defined as

$$RXF = XF \frac{Ymax-Ymin}{Xmax-Xmin} \qquad (3\text{-}10)$$

Proceeding as before, we compute the linear transformation X' = CX + D such that

$$CXmin + D = XF/2 - RXF/2$$

$$CXmax + D = XF/2 + RXF/2$$

From which

$$X = RXF/(Xmax-Xmin)$$
$$D = (XF-RXF)/2 - CXmin \qquad (3\text{-}11)$$

The above assumes that the projected view is higher than it is wide; i.e., that Ymax-Ymin > Xmax-Xmin. If the opposite is true, we follow the same procedure, but with the roles of X and Y interchanged.

Note that the scaling procedure described here is not

restricted to three-dimensional graphics. It will work for two-dimensional drawings as well; after all, the drawing of a three-dimensional object is two-dimensional. Also, a drawing generated with this system will not require clipping; i.e., the deletion of portions of lines which fall outside the frame.

3.6 Contouring in Three Dimensions

Drawing contours in three dimensions differs from the two-dimensional case only slightly. Rather than drawing each increment of the contour as it is found, the contour coordinates are stored until the entire contour is found, then the contour points are translated, rotated, converted to perspective, scaled, and drawn en masse. The hill picture in Figure 10 was drawn in this way using the accumulating version of subroutine DRAW listed in Appendix B.

3.7 Summary

The generation of a perspective drawing of a solid object from an arbitrary viewpoint in space has been shown to be a straightforward step-by-step procedure. The individual steps can be remembered from the mnemonic acronym TRAPS, for Translate, Rotate, Add Perspective, Scale. The drawing of a turbine blade shown in Figure 11 shows a typical result. The drawing program is listed in Appendix C.

Please note that the procedures described in this section do not delete hidden lines although they are integral to a hidden line algorithm. The pictures obtained here are the so-called wire frame variety and are entirely satisfactory for many applications. Hidden line elimination begins where the foregoing leaves off, and is treated in detail in the next sections.

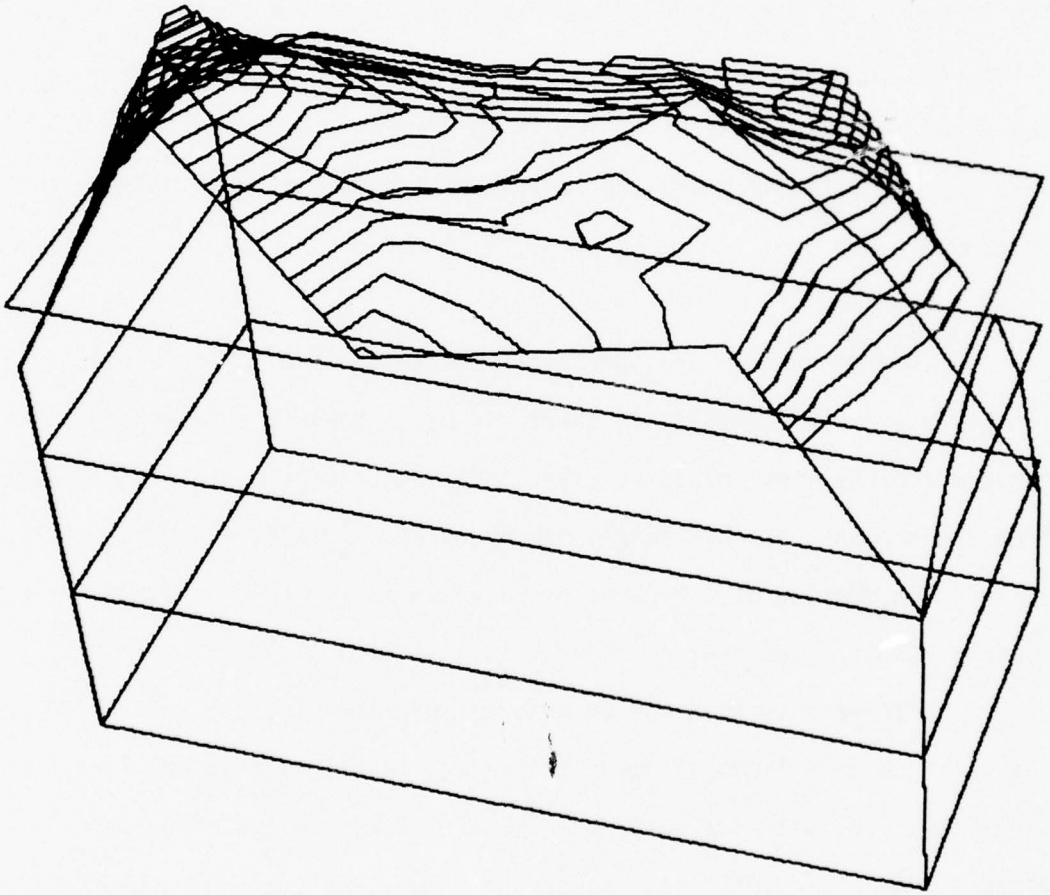


Figure 10

Contours in Three Dimensions

Figure 11

Wire Frame Drawing of a Turbine Blade

4. Hidden Line Algorithms

4.1 Introduction

One frequently wishes when drawing three dimensional objects with a computer that he had a way of eliminating the back lines; i.e., those which are on the other side of the object from the viewer. The drawing of a turbine blade in Figure 11 is an example. The back lines are so dense as to obscure the picture. Ideally, the line-hiding algorithm should not be too difficult to apply nor take a lot of storage nor take a lot of time to execute.

The two algorithms presented herein partially fulfill these requirements. Both require some data preparation but memory requirements are determined by the number of data points on the object to be drawn rather than by program size and run times are acceptable if not brief.

The descriptions to follow will include program listings and discussions sufficiently detailed that the reader can use either method on his own problems with a minimum of modification.

4.2 Roberts' Algorithm

4.2.1 Roberts' algorithm takes the approach that each line in the three-dimensional scene should be tested against every opaque surface to determine which portions, if any, of the line are visible. As such it is particularly convenient for scenes such as the hill in the previous section where one may wish to add selected contours to the basic hill, or for the turbine blade in Figures 11 and 17 through 22 where one may wish to leave out part of the object and show the interior.

The basic concepts involved are quite simple, but the implementation gets to be very intricate. Everything to

follow builds on these three concepts: (1) Solid objects are represented as plane convex polyhedra, convex because a solid is not able to hide itself. Concave solids; e.g., horseshoe, can be built of adjoining convex solids, say three (or more) straight bars. (2) Each plane of each polyhedron is represented by an equation of the form

$$ax + by + cz + d = 0 \qquad (4\text{-}1)$$

The plane equations are stored as column vectors of their coefficients

$$Po = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

The set of column sectors for a body can be considered as a 4 by n Matrix, called the Volume Matrix:

$$V = \begin{bmatrix} a_1, a_2 \ldots a_n \\ b_1, b_2 \ldots b_n \\ c_1, c_2 \ldots 1\ c_n \\ d_1, d_2 \ldots d_n \end{bmatrix}$$

(3) A general point in space can be represented by a row vector, So = (x, y, z, 1). If So is on Po, the dot product Po·So is zero by eq (4-1). Furthermore, if So is not on Po, the sign of the dot product tells which side of the plane the point is on. Roberts uses the convention that points inside the polyhedron yield positive dot products. Then points between the viewer and a visible plane will yield negative dot products. This fact will be very useful later.

4.2.2 First Example: Basic Concepts.

Consider the 2x2x2 cube in Figure 12. Its sides are

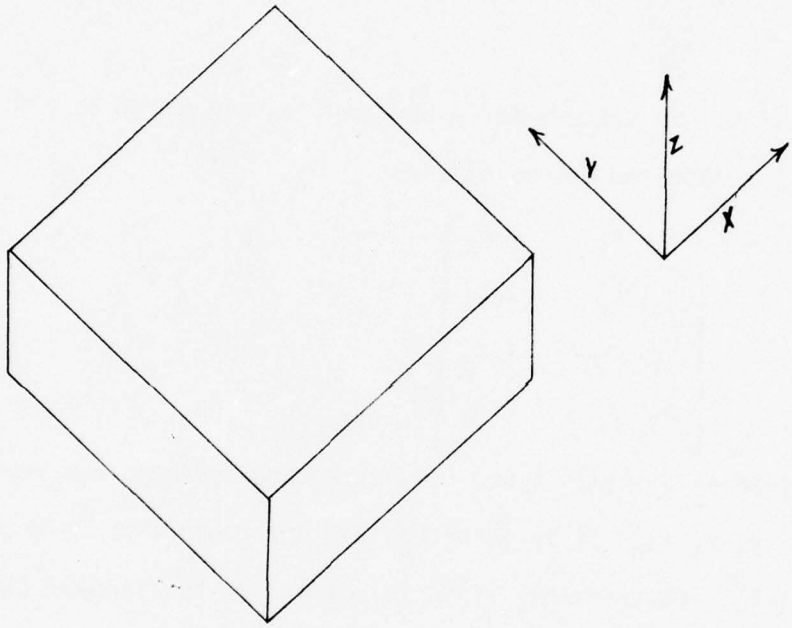


Figure 12

Cube with Hidden Lines Deleted

the planes:

1. $x = -1$
2. $x = -3$
3. $y = 0$
4. $y = 2$
5. $z = 0$
6. $z = 2$

A point inside the cube is (-2, 1, 1, 1). The reader can easily verify that the dot product of this point with every column in the volume matrix below is positive.

$$V = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ -1 & 3 & 0 & 2 & 2 \end{bmatrix}$$

4.2.3 Eliminating Back Lines

The easy part of hidden line elimination follows. The first step is to choose a viewpoint and translate and rotate the xyz coordinates of the vertices of the solid just as in Section 2.3 plus which we translate and rotate the volume matrix. The rotation matrix is the same as eq (3-7) with a fourth row and column added:

$$R = \begin{bmatrix} f\cos\beta & f\sin\beta & 0 & 0 \\ f\sin\alpha\sin\beta & -f\sin\alpha\cos\beta & \cos\alpha & 0 \\ -f\sin\beta\cos\alpha & f\cos\alpha\cos\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4\text{-}2)$$

The plane translation matrix is:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Xv & Yv & Zv & 1 \end{bmatrix}$$

Where (Xv, Yv, Zv) are the object coordinates of the viewpoint.

The translation matrix premultiplies the volume matrix and the result is premultiplied by the rotation matrix to yield the rotated volume matrix, B.

The rotated volume matrix B = RTV contains the information needed to determine which planes are hidden. As a result of the sign convention for planes, the signs of the entries of the third row (the one containing the z coefficients, C_j) will be negative for hidden planes and positive or zero for visible planes. Thus, if both planes whose intersection defines a line are invisible, that line is invisible. If either plane is visible the line is visible. Our cube in Figure 12 was drawn from these criteria:

4.2.4 Second Example. Drawing the Cube

The viewpoint in Figure 12 is (-4, -4, 16) for which the rotation matrix is:

$$R = \begin{bmatrix} 0.7071 & -0.7071 & 0 & 0 \\ 0.6667 & 0.6667 & 0.3333 & 0 \\ 0.2357 & 0.2357 & -0.9428 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Performing the matrix multiplication, we obtain the rotated volume matrix.

$$B = \begin{bmatrix} -0.7071 & 0.7071 & -0.7071 & 0.7071 & 0 & 0 \\ -0.6667 & 0.6667 & 0.6667 & -0.6667 & 0.3333 & -0.3333 \\ -0.2357 & 0.2357 & 0.2357 & -0.2357 & -0.9428 & 0.9428 \\ 3 & -1 & -4 & 6 & 16 & -14 \end{bmatrix}$$

Note that the above does not include the perspective transformation. We will deal with that presently.

The third row of B above contains negative entries in columns 1, 4, and 5 which means that planes 1, 4, and 5 of the cube are not visible. As we look at Figure 12, we can see that this is indeed the case.

4.2.5 The Hard Part: Adding Another Line to the Scene

The process just described is adequate for drawing simple objects such as cubes, prisms, etc. However, it does not take

long to exhaust all the possibilities. If we want to draw something more complex, such as two cubes, or even add the axes in Figure 12, we have to test each line against each volume in the scene. If there are two or more volumes, each line of each volume which passes the back line test must then be tested against each volume except its own.

Each line is represented parametrically in terms of its end points; thus a line from (X1, Y1, Z1) to (X2, Y2, Z2) is:

$$\bar{v} = \begin{pmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{pmatrix} + t \begin{pmatrix} X1 - X2 \\ Y1 - Y2 \\ Z1 - Z2 \\ 1 - 1 \end{pmatrix} \qquad (4\text{-}4)$$

or $\bar{v} = \bar{s} + t\,\bar{d}$

Where $0 \leq t \leq 1$ so that when $t = 0$ we have the end of the line and when $t = 1$ we have the beginning.

The 1 at the bottom of the column vectors permits the homogenous 4-space representation of points in 3-space. It may be thought of as a scale factor. See Appendix II of Newman and Sproull for a complete discussion.

Each value of t defines a point on the line (4-4). We can form another line from that point to the viewer's eye. (Recall that in eye coordinates, the viewer's eye is at the origin, z increases away from the viewer and x and y increase to the right and upward respectively.) Such a line is:

$$\bar{u} = \bar{s} + t\bar{d} + \alpha\bar{g} \qquad (4\text{-}5)$$

Where s, t and d are as above

$\bar{g}$ is the vector $\begin{pmatrix} 0 \\ 0 \\ -1 \\ 0 \end{pmatrix}$

$\alpha \geq 0$, scalar

To determine if line (4-4) is hidden by a volume, we post-multiply equation (4-5) by the rotated volume matrix of that volume; i.e., we form the product:

$$\vec{h} = \vec{u}\,[B] = \vec{s}\,[B] + t\vec{d}\,[B] + \alpha\vec{g}\,[B] \qquad (4\text{-}6)$$

If each component of the vector $\vec{h}$ is non-negative for some values of t, the line (4-5) passes through the volume, meaning that the line (4-4) is hidden by the volume for those values of t.

Let us define

$$\begin{aligned} \vec{p} &= \vec{s}\,[B] \\ \vec{q} &= \vec{d}\,[B] \\ \vec{w} &= \vec{g}\,[B] \end{aligned} \qquad (4\text{-}7)$$

Note that $\vec{w}$ is (from the definition of $\vec{g}$) identically the third row of [B], the row we checked for back lines.

Equation (4-6) then becomes

$$p_j + tq_j - xw_j \geq 0 \qquad (4\text{-}8)$$

Where j counts the columns in [B] (or the planes of the volume); i.e., for our cube we would have six equations. We then look for the largest and smallest values of t which will satisfy the set of inequalities (4-8) for α not negative. In essence, we have a linear programming problem with n equations and three constraints; namely, $\alpha \geq 0$, $t \geq 0$, and $t \leq 1$. How do we go about solving it? Let us look at another example, then we will consider some of the intricacies of the process in detail.

4.2.6 Third Example: Testing a Line Against a Volume

Let us look at our cube again from the same viewpoint; (-4, -4, 16). Say we want to add a line from (0, 1, -5) to (-3, 4, 5) in object coordinates, deleting the part behind the cube.

The rotated volume matrix is the same as in the second example and the line (4-4) become after translation and rotation:

$$\vec{v} = \begin{pmatrix} 4.9497 \\ 2.3333 \\ 12.4922 \\ 1 \end{pmatrix} + t \begin{pmatrix} 4.2426 \\ -3.3333 \\ 9.4281 \\ 0 \end{pmatrix}$$

Since we already know what $\vec{w}$ is, we need only to multiply $\vec{s}$ and $\vec{d}$ by R to obtain $\vec{p}$ and $\vec{q}$:

$$\vec{p} = \begin{pmatrix} 2 \\ 0 \\ 4 \\ -2 \\ 5 \\ -3 \end{pmatrix} \text{ and } \quad \vec{q} = \begin{pmatrix} -3 \\ 3 \\ -3 \\ 3 \\ -10 \\ 10 \end{pmatrix}$$

We can there form six equations in t and α using eq (4-8).

1. $2 - 3t + 0.2357\ \alpha \geq 0$
2. $3t - 0.2357\ \alpha \geq 0$
3. $4 - 3t - 0.2357\ \alpha \geq 0$
4. $-2 + 3t + 0.2357\ \alpha \geq 0$
5. $5 - 10t + 0.9428\ \alpha \geq 0$
6. $-3 + 10t - 0.9428\ \alpha \geq 0$

The graphs of these equations (taken as equalities) are plotted in Figure 13.

To find the largest and smallest t which satisfy these equations, we solve them simultaneously in pairs, then test the resulting t and α in the remaining inequalities, discarding any t and α which do not satisfy all of them. In other words, we are finding the intersections of the graphed equalities in Figure 13. In addition, we compute the intersections of each equality with the lines $t = 0$ and $t = 1$, discarding any for which $\alpha < 0$ and testing the rest in the inequalities as before. Also, we find the intersections for which $t > 0$ or $t < 1$ because solutions in these ranges indicate

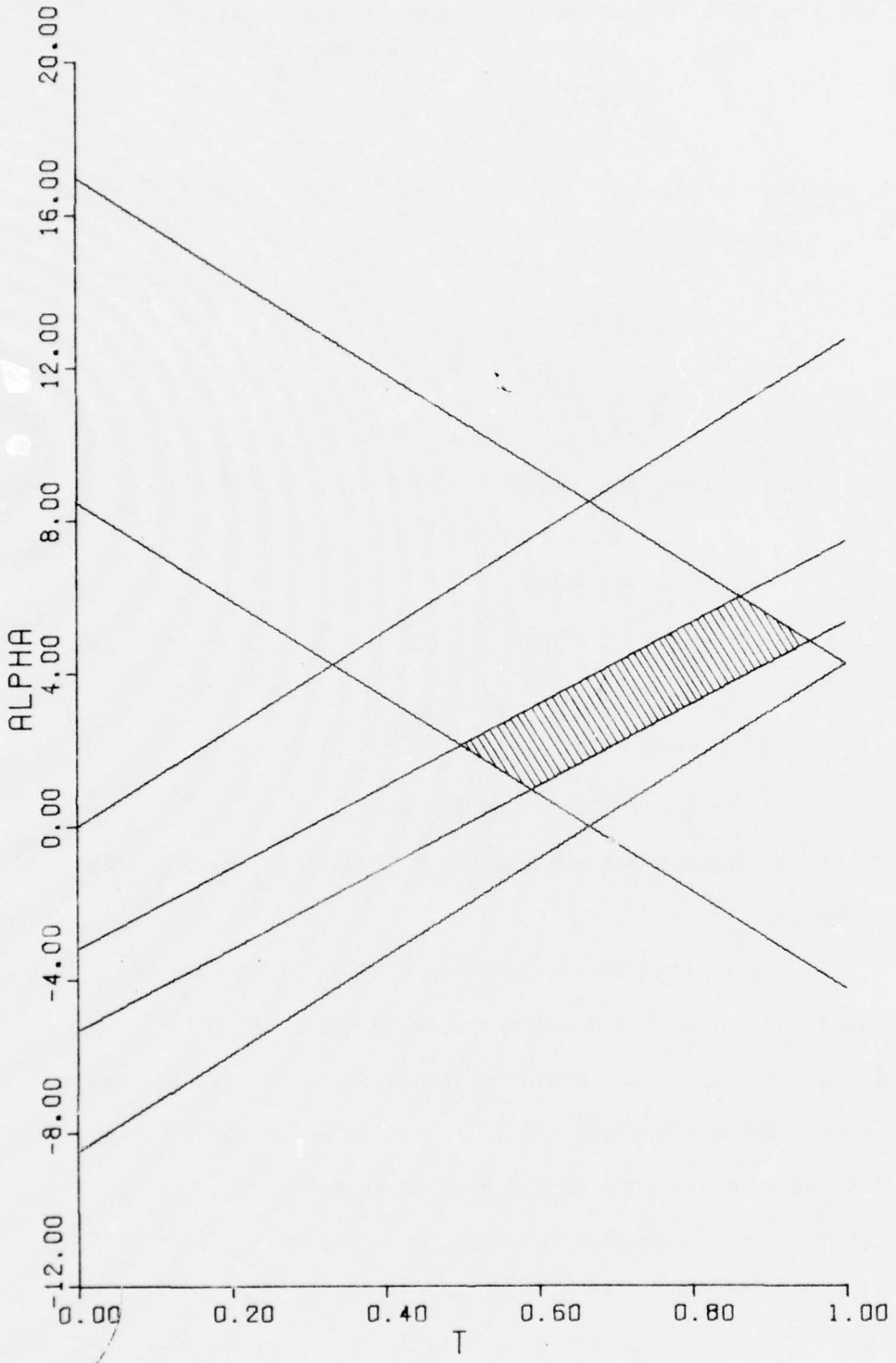


Figure 13

Typical t-α Diagram

that one end of the before given line is hidden.

Throughout this process, we keep the largest and smallest t's which survive all of the tests. The final pair are the left and right extremities of the shaded quadrilateral in Figure 13. The given line is hidden for all values of t between these two and visible otherwise, i.e.:

visible: $0 \leq t \leq t_{min}$

hidden: $t_{min} < t < t_{max}$

visible: $t_{max} \leq t \leq 1$

If $t_{min} < 0$ or $t_{max} > 1$, that end of the line is hidden.

In this example, we find $t_{min} = 0.5$ and $t_{max} = 0.9545$. The resulting segments are drawn in Figure 14.

4.2.7 Some of the Intricacies

It is possible to speed up the search for t_{min} and t_{max} by eliminating some of the graphed equalities from consideration. There are two criteria we can use:

1. The given line is entirely in front of the body and is not hidden.

2. The combination of p_j, q_j, and w_j is such that the inequality (4-8) will always be satisfied for $0 \leq t \leq 1$ and $\alpha > 0$.

To test for complete visibility, we go back to the definitions of $\vec{p}$ and $\vec{q}$, eq (4-7). By the rules of matrix multiplication, the jth component of $\vec{p}$ is the dot product of the end point of the given line with the jth plane of the solid. Likewise, the jth component of the sum $\vec{p} + \vec{q}$ (i.e., t = 1) is the dot product of the beginning of the given line with the jth plane of the solid. We

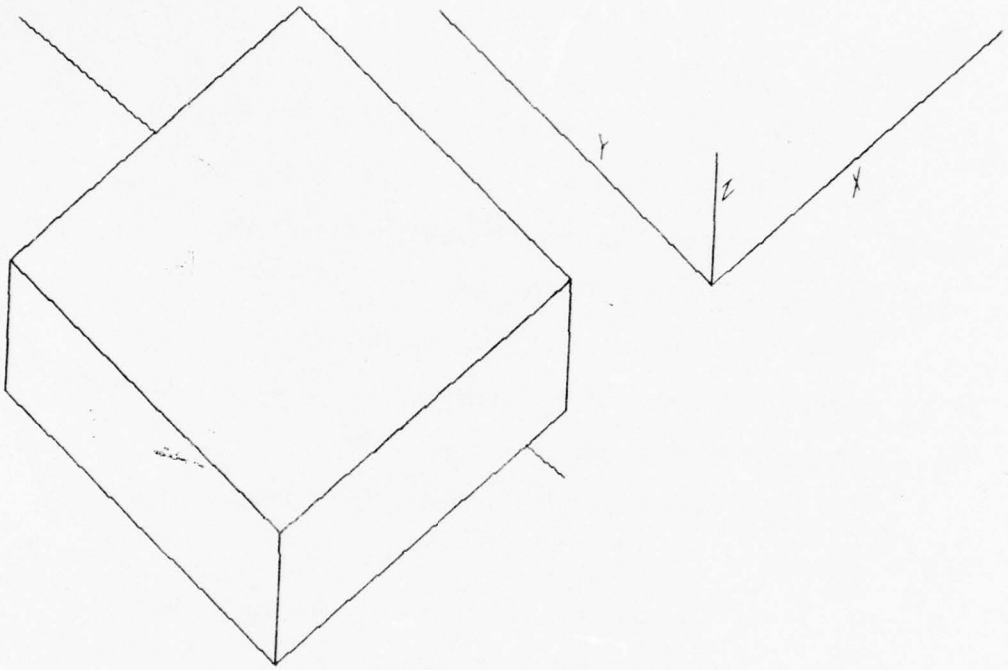


Figure 14

Cube Hiding a Line

the scale factors described in Section 2.5 are computed. We are ready

showed in Sections 4.2.3 and 4.2.4 that the jth plane is visible if $w_j \geq 0$ and in 4.2.1 that points between the viewer and a visible plane will yield negative dot products. Therefore, if $w_j \geq 0$, the plane is visible; if $p_j < 0$, the end of the line is between the viewer and the plane; and if $p_j + q_j < 0$, the beginning of the line is between the viewer and the plane. If both ends are between the viewer and the plane, all other points on the line must also be and the line is completely visible. Thus, our criterion is:

$$w_j \geq 0 \text{ and } p_j \leq 0 \text{ and } p_j + q_j \leq 0 \qquad (4\text{-}9)$$

The equalities of p_j and $p_j + q_j$ allow for the case where the line is on the plane (dot product identically zero from eq 4-1).

If the criterion (4-9) is met, the remainder of the procedure can be skipped and the line drawn immediately. Obviously, this can save considerable time in a complex figure where a given line is as likely to be in front of a volume as behind.

Roberts gives two criteria for eliminating a graphed equality on the grounds that (4-8) will always be satisfied. They are:

1. $w_j \leq 0$ and $p_j \geq 0$ and $q_j > 0$ (4-10)
2. $w_j \leq 0$ and $p_j > 0$ and $q_j < 0$ and $p_j + q_j \geq 0$

In terms of the dot products, these mean that the given line is on or behind an invisible plane. We cannot however assume the line to be completely hidden because although the plane is infinite in extent, that portion of it which corresponds to a side of the volume is finite and the ends of the line may stick out past it.

Roberts also gives two criteria for rejecting graphed equalities on the grounds that (4-8) will never be satisfied. They

are:

1. $w_j \geq 0$ and $p_j \leq 0$ and $q_j < 0$
2. $w_j \geq 0$ and $p_j < 0$ and $q_j > 0$ and $p_j + q_j \leq 0$

We can see by inspection that these are contained in (4-9).

In practice (i.e., in a computer program), it is necessary to assign tolerances to the tests in (4-8) and (4-9) to allow for round-off error. By trial and error, the following were found to work well:

(4-8): $p_j + tq_j - \alpha w_j \geq -10^{-6}$

(4-9): $p_j < 10^{-4}$ and $p_j + q_j < 10^{-4}$ and $w_j \geq 0$

4.2.8 Perspective

The process here is almost identical to that described in Section 3.4 except that we do not ignore the z values and we must transform the volume matrix to the new coordinate system, called the screen coordinate system by Newman and Sproull. The difference between screen coordinates and eye coordinates is that in screen coordinates the viewpoint is at $z_s = -\infty$, while in eye coordinates, the viewpoint is at the origin. The screen in screen coordinates is not at any particular z_s location. The coordinate transformations are simply:

$$Xs = \frac{Xe}{Ze} \tag{4-11}$$

$$Ys = \frac{Ye}{Ze} \tag{4-12}$$

$$Zs = -\frac{1}{Ze} \tag{4-13}$$

Or in matrix form,

$$\begin{pmatrix} Xs \\ Ys \\ Zs \end{pmatrix} = S \begin{pmatrix} Xe \\ Ye \\ Ze \end{pmatrix}$$

where

$$S = \begin{pmatrix} 1/ze & 0 & 0 & 0 \\ 0 & 1/ze & 0 & 0 \\ 0 & 0 & 0 & -1/ze \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (4\text{-}14)$$

Transforming the rotated volume matrix from eye to screen coordinates turns out to be very simple: $B' = ZB$, where

$$Z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad (4\text{-}15)$$

The line hiding procedure described above needs no changes; it works equally well in either coordinate system. Figure 15 is a perspective drawing of the same cube and the same line as the third example. The $t - \alpha$ diagram is shown in Figure 16.

4.2.9 Summary of Roberts' Algorithm

The foregoing has described a method of generating hidden line drawings of simple objects. We have seen how the volume matrix for a solid is generated and how the back lines of a solid are eliminated. Sections 4.2.4 through 4.2.6 dealt with the difficult problem of adding another line to the scene via the parametric representation of that line in space, the generation of the t-α diagram and the search for t_{min} and t_{max}. Finally, we added the the option of drawing perspective views.

Roberts' algorithm as presented here is capable of handling rather complicated objects if they are built up of simpler forms as the next section will show. It will fail however for two cases. First, if the given line pierces a body, the algorithm will find a t at one of the edges of the body rather than at the intersection of the line and the plane. Perhaps this could be worked out; it has

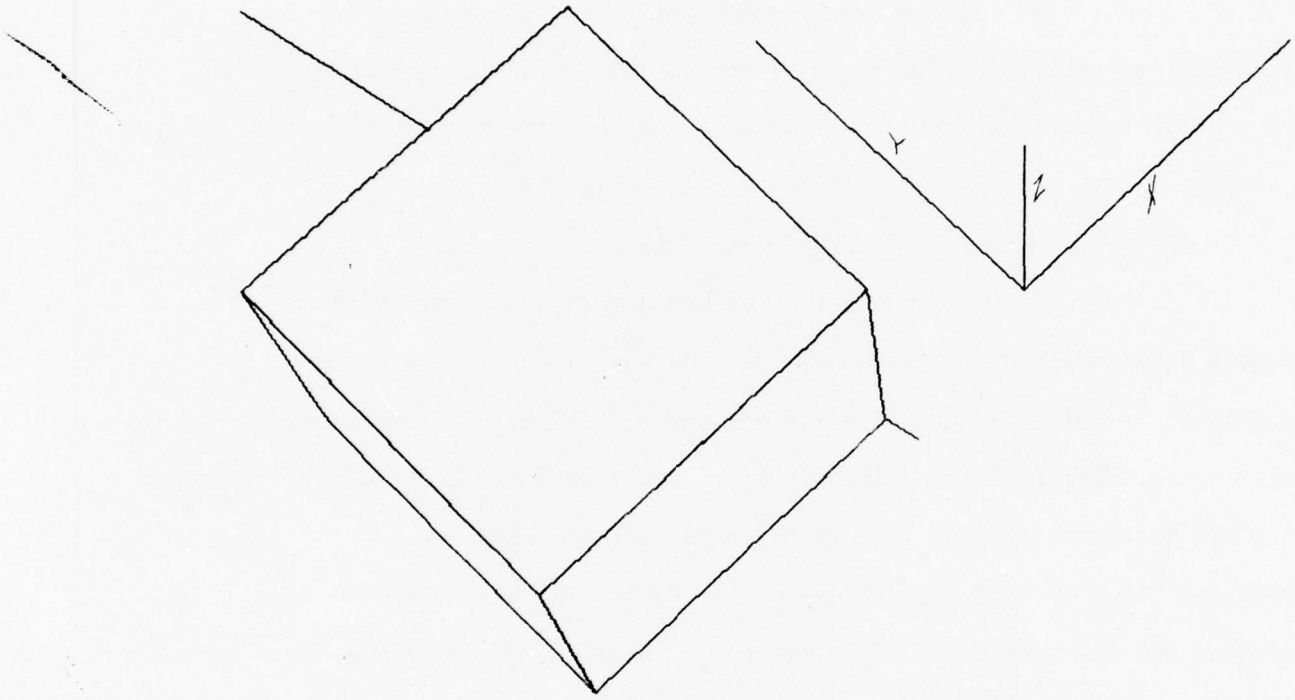


Figure 15

Perspective Drawing of a Cube Hiding a Line

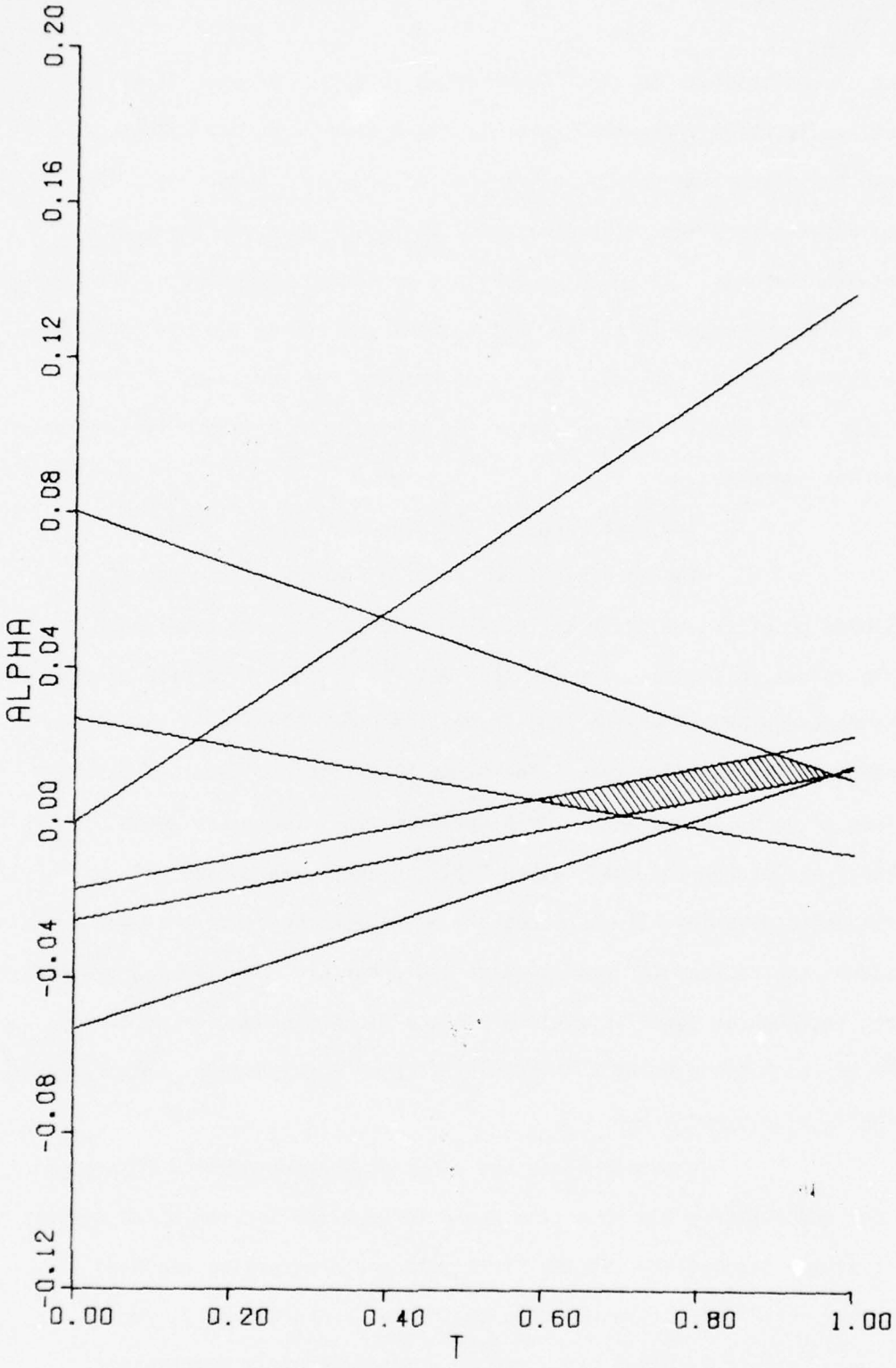


Figure 16

Typical t-α Diagram with Perspective

not caused problems in the figures drawn to date. Second, it will not handle cases where two segments of the given line are hidden, as when the given line passes behind the two arms of a horseshoe. The implementation given in the next section would leave out the section between the arms. It would be possible to handle this case by treating the surviving segments of the test against one arm as two new lines to be tested against the other arm, then drawing the survivors of those tests. This problem did not occur for the objects drawn so far and so was not addressed.

4.2.10 The Last Example: Drawing the Turbine Blade

The turbine blade shown in Figures 17 through 22 is made up of two major units, each of which is in turn subdivided into convex polyhedra. The fir tree base is divided into five polyhedra: the rectangular bar on top, the trapezoidal slab beneath it, and three hexagonal prisms below that. The blade proper is divided into four tiers of 26 polyhedra each, for a total of 104. These are wedge shaped except for the leading and trailing edges, which are made up of triangular pyramids. It was necessary to divide the blade this way because the rectangular panels which are drawn are not planar. Instead, each rectangular panel is divided along a diagonal and treated as two planes, each plane being a triangular face of a polyhedron. The diagonal line is simply not drawn.

Each polyhedron has a number, beginning with 1 through 5 for the fir tree top to bottom and 6 through 109 for the blade starting at the leading edge of the first tier and progressing rearward through each tier before going on to the next. Furthermore, each plane of each polyhedron has a number which ultimately corresponds to a column in the volume matrix. The actual plane numbering convention

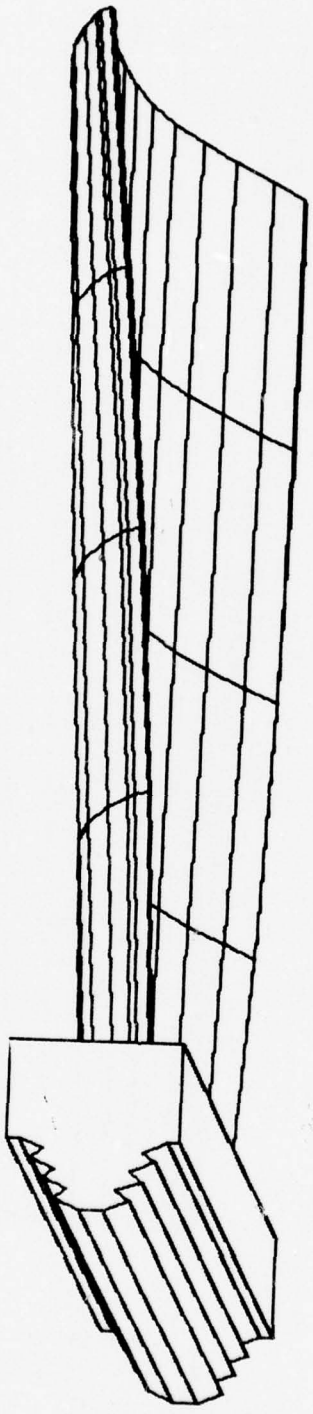


Figure 17
Turbine Blade Viewed from (-100, 1, -75)

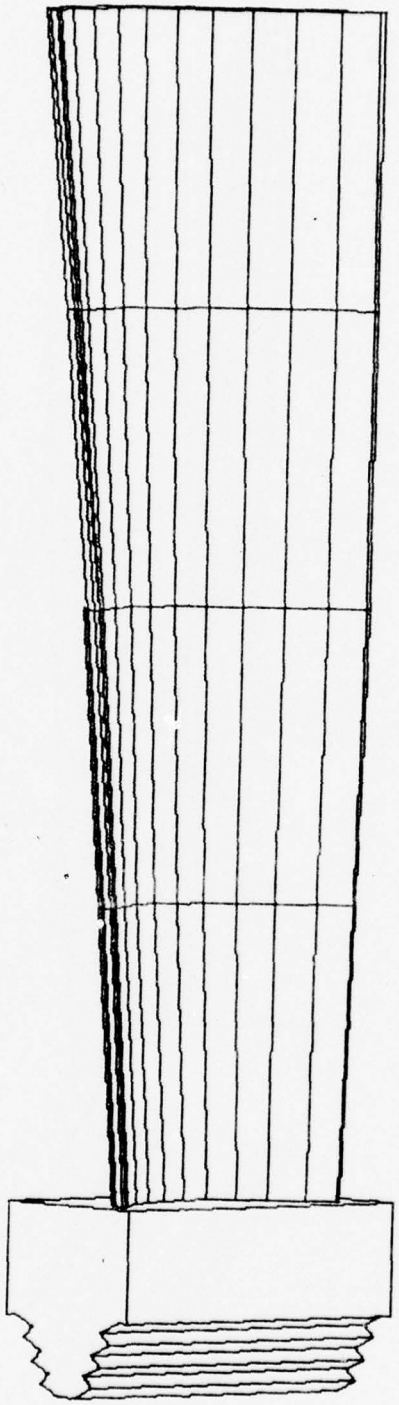


Figure 18
Turbine Blade Viewed from (-95, -81, 20)

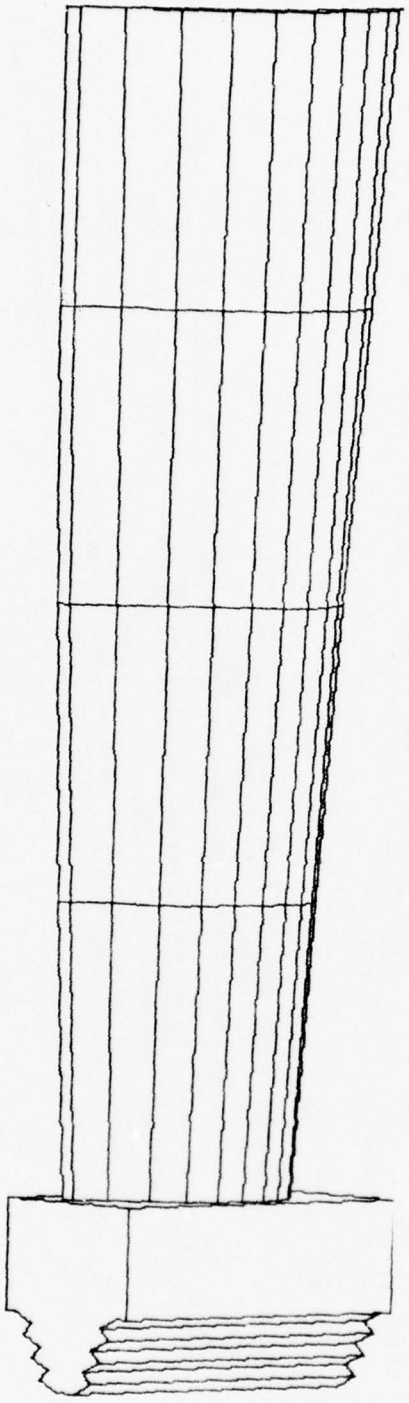


Figure 19

Turbine Blade Viewed from (95, 81, 20)

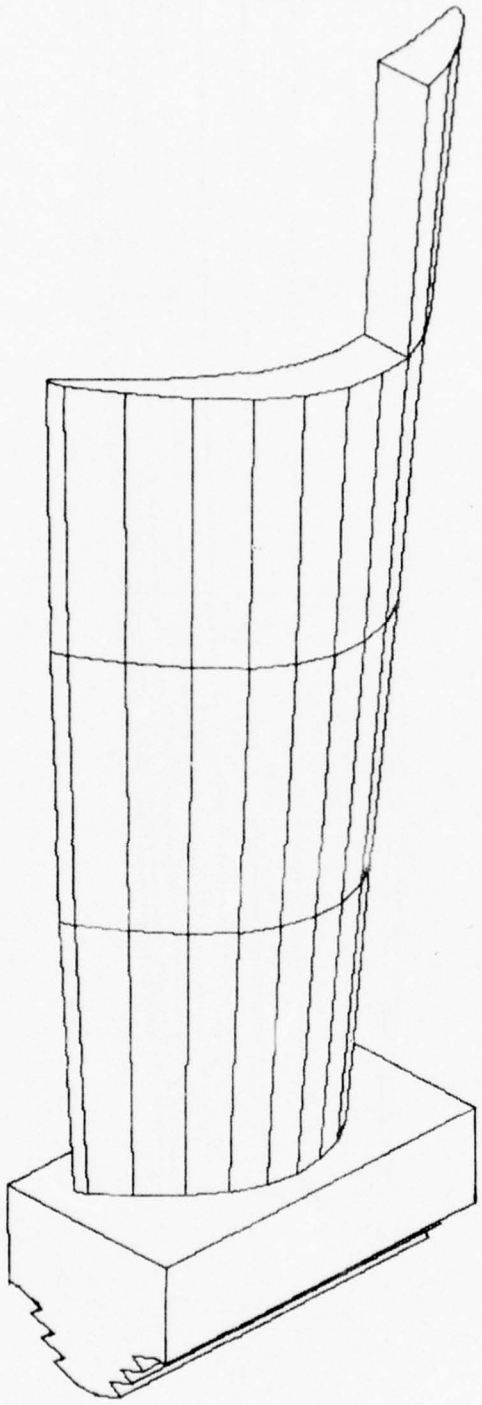


Figure 20
Partial Turbine Blade Viewed from (70, 51, 100)

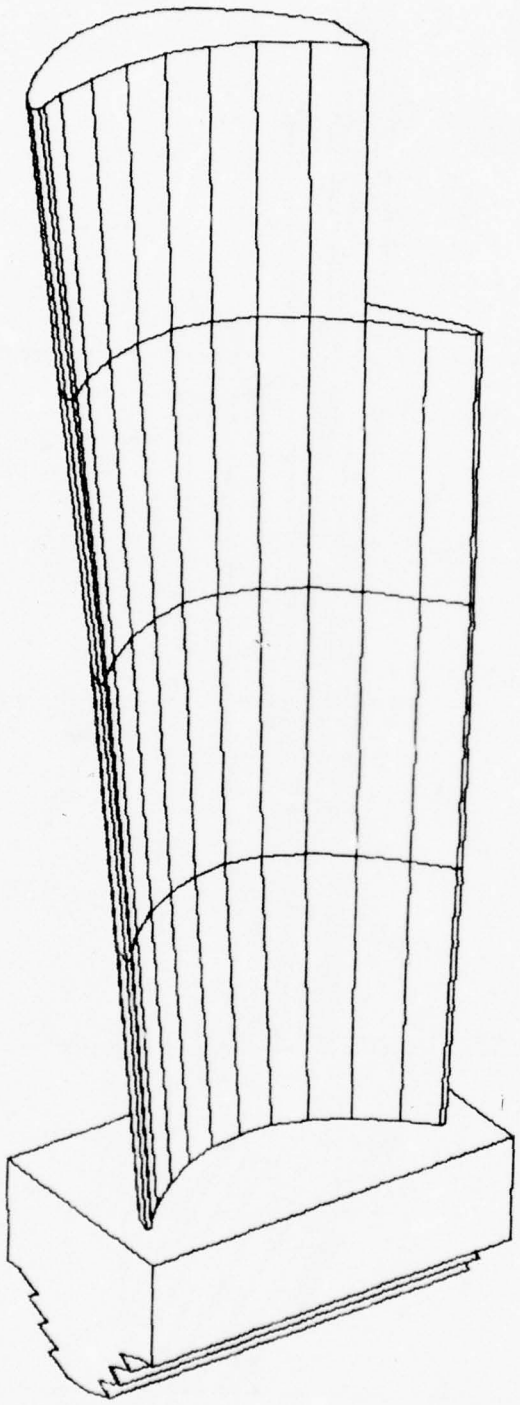


Figure 21
Partial Turbine Blade Viewed From (-61, -61, 100)

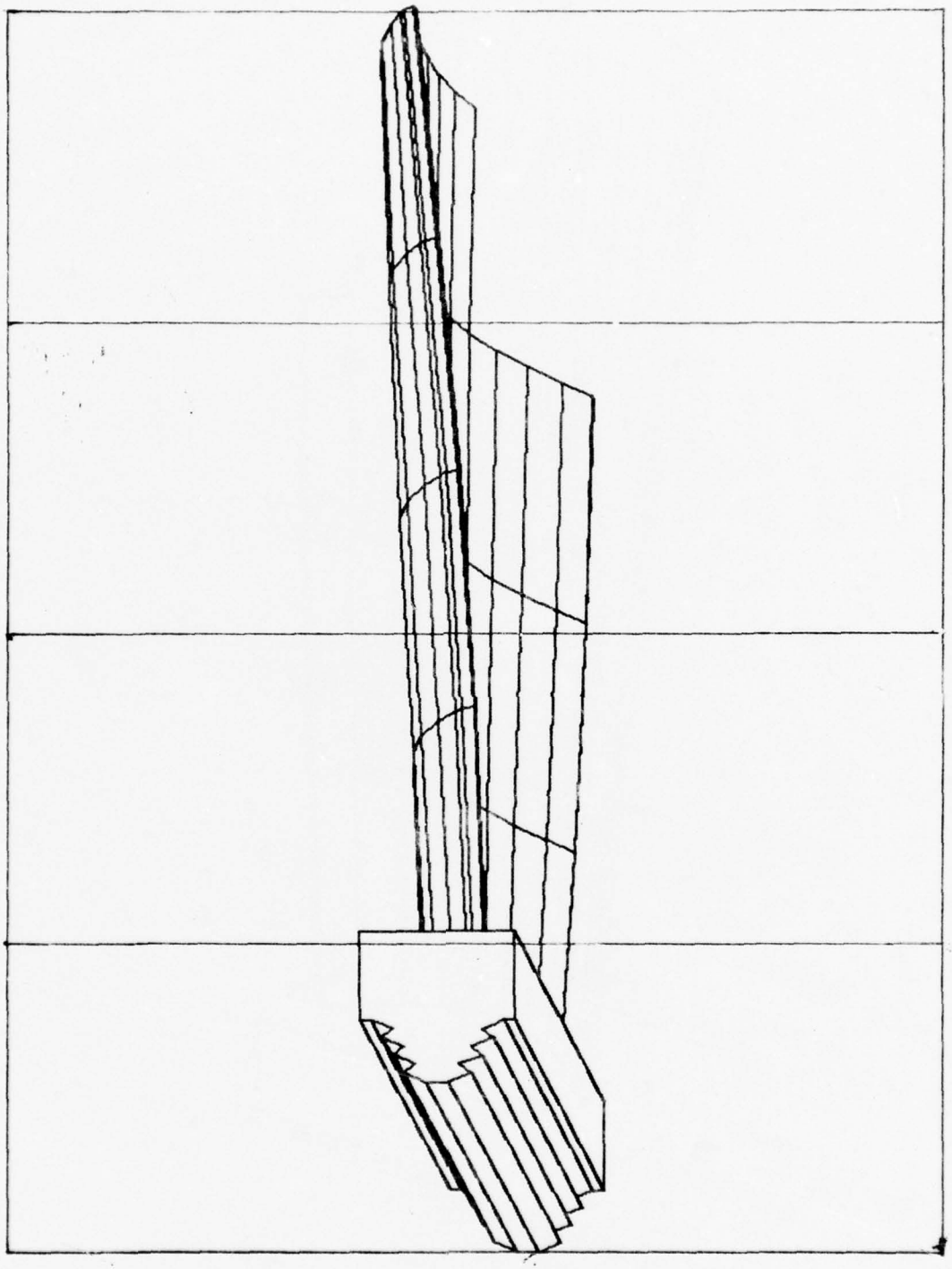


Figure 22

Partial Turbine Blade Viewed from (-100, 1, -75)
Showing Window Boundaries

does not matter. In this case, the bottom planes of all segments of the fir tree are assigned number 1 and numbering proceeds counter-clockwise as seen from the positive x axis (object coordinates). The near end is next to last and the far end last. The blade uses a different convention because of its different geometry. Each axial increment is made up of lower and upper prisms, the lower being forward. The bottom plane of the lower prism is number 1, the pressure (concave) surface is number 2, the suction (convex) surface is number 3, the diagonal plane is number 4 and the forward plane is number 5. The diagonal plane of the upper prism is number 1, the pressure and suction surfaces are 2 and 3, the top plane is number 4, and the rear plane is number 5. Leading and trailing edge segments are special cases having only four planes.

Each line which appears in the drawings represents the intersection of two planes of a polyhedron. Accordingly, each line is assigned a pair of numbers denoting which planes of which polyhedron, in the form: plane number + 10 times polyhedron number; e.g. 15 would denote plane 5 of segment 1; and 13, 15 would denote the intersection of planes 3 and 5 of segment 1. These numbers are stored as LLIST for the fir tree and LLISTB for the blade.

Finally, the XYZ coordinates of the end points of each line are stored and the plane equations for all the segments are stored in a single 4 x 548 volume matrix according to the numbering convention. A list called MLIST marks the beginning and length of the part of the volume matrix corresponding to a given segment. When a viewpoint is selected, the coordinates are translated, rotated and converted to perspective as described above. The transformed space coordinates are scanned for maximum and minimum x and y values and

the scale factors described in Section 2.5 are computed. We are ready to draw.

Drawing begins with the upper right corner of the front base of fir tree and proceeds clockwise around the 20 corners of that face. Each corner has three lines associated with it and each line has a pair of entries in LLIST. The first line goes from the current corner to the next corner on the same face, the second goes lengthwise from the current corner to the corresponding corner on the rear face, and the third goes from that corner to the next corner on the rear face. As each line is selected, the main program passes the appropriate pair of entries from LLIST to subroutine FINDM. FINDM decodes the LLIST numbers and returns the segment number in the variable NM and the volume matrix column subscripts for the two planes whose intersection is the line in the variables MV and NV. The main program then checks the signs of the Z coefficients in these columns of the rotated volume matrix. If both are negative, the line is a back line and is entirely hidden. If one or both are not, the main program forms two vectors consisting of the x, y and z components of the beginning and end points of the line and passes these to subroutine HIDE.

Subroutine HIDE tests the given line against several segments, excluding the line's own segment which is denoted by NB. The segments tested against are those which appear in the same region or "window" of the picture as the given line. Windowing is discussed in greater detail below. As HIDE tests the given line against each segment, it keeps the smallest t_{min} (called T1) and largest t_{max} (called T2) from all the tests and finally draws the line based on

these cumulative values. HIDE calls subroutine SEEK to perform the actual testing, giving it the s and d vectors for the line and the beginning column and number of columns in the volume matrix which correspond to that segment, via the variables J1 and J2.

Subroutine SEEK performs the functions described in Sections 4.2.5 and 4.2.6. It computes the p and q vectors and attempts to eliminate some of the inequalities via the tests (4-9) and (4-10). Subscripts of survivors are kept in the array M. If less than two survive, the given line is treated as visible. Otherwise, intersections with the lines $t = 0$, $t = 1$ and $\alpha = 0$ are computed and passed to subroutine VALID, which tests to see that 4-8 is satisfied for all subscripts in M. If it is and the t is smaller than the current t_{min} or larger than the current t_{max}, VALID updates the appropriate one. Otherwise, VALID simply returns. The intersections with each other are handled identically.

Drawing of the blade proceeds similarly. The lines to be drawn are associated with the lower of the two segments associated with each axial increment. The first goes rear-ward along the bottom of the pressure side, the second goes rear-ward along the suction side, the third goes upward on the pressure side and the last goes upward on the suction side. Thus, the program proceeds rearward along a tier drawing a pair of L's at each axial increment. The trailing edge is closed with a line from the trailing edge list, LTE, and the top of the final tier is drawn from the top list, LTOP.

In order to evaluate the time Roberts' algorithm would require to draw varying numbers of segments, an option to draw only part of the blade was included in the program. The user may specify 1 to 4 tiers and 2 - 26 segments in the uppermost tier drawn and thereby

obtain run times for 5 to 109 segments. The partial blade is furnished off by capping the partial tier with the bottom lines of the next tier above or with the top list as appropriate. The uncovered part of the next lower tier is capped with the remaining bottom lines from the unfurnished tier. The back of the last segment in the unfurnished tier is closed with the two vertical lines from the next segment and with two horizontal lines from the pressure side to the suction side, one at the bottom and one at the top. These last two do not appear in any line list, but they are from the same body as the two vertical lines; therefore NB is the same as is the windowing data.

A series of cases were run from the viewpoint (-100, 1, -65) because it yielded the longest observed run times and thus gave worst case values. The resulting values were fit via least squares to a curve of the form.

$$t = a\, n^{p} \qquad (4\text{-}16)$$

Where t is the cental processor time in seconds and n is the number of segments drawn. The curve fit yielded

$$a = 0.0162 \text{ and}$$

$$p = 1.491$$

The important value is the exponent p. Newman and Sproull give an exponent of 2 and conclude that the algorithm is extremely slow for complicated scenes. Indeed, this implementation without windowing is not very fast; it required 17.8 seconds of CDD 6600 time to draw the entire blade.

Then windowing was added as described below and the process repeated. As might be expected, the program ran slightly slower for a small number of segments (break-even is between 5 and 10)

but showed a marked improvement at higher numbers; it took only 8.84 seconds to draw the entire blade. The resulting coefficients from the curve fit were

$$a = 0.0529 \text{ and}$$

$$p = 1.089$$

Again, the important parameter is the exponent. Merely by dividing the scene into fourths and testing a line only against these segments which occupy the same regions and thereby have a chance of affecting it reduced the exponents from nearly 3/2 to nearly linear which then makes the algorithm viable for complicated figures. No attempt was made to optimize the number of windows although an optimum surely exists; rather the intent here was to demonstrate the concept.

4.2.11 Windowing

The turbine blade is essentially a vertical object, therefore the viewing area is divided into four horizontal windows stacked one on another as in Figure 22. After all plane and coordinate rotations are done, the main program calls subroutine WINDOW; WINDOW computes the height of the window segments as fractions of the frame height; i.e., they are in page (or screen) inches and stores them in the array YW. It then scans the Y coordinates of the ends of each segment, passing to function IN THERE the scaled (ready to draw) Y value. IN THERE checks the Y value against the window boundaries and if YW(I) < Y < YW (I+1), it returns the value I, where I is 1, 2, 3, or 4. WINDOW then stores the segment number L in location (I,L) of the array IN which is dimensioned (4,109). For example, if segment 43 appears in window 3 (numbered from the bottom up), IN (3,43) = 43. After assigning each end of each segment to a window, WINDOW checks to see if any segment is in three or four

windows, that is if, for example, one end is in window 1 and the other end is in window 3 or 4, the segment must also be in the intervening window. WINDOW adds these entries to the IN array as appropriate and returns.

Where the time to draw the a line arrives, it is a simple matter to prepare a list of segments in the same windows as the line. One takes the segment number for the line, finds which windows that segment is in, and prepares a list of all segments in those windows. Subroutine DOLIST prepares this list, which is called JLIST, for use by subroutine SEEK.

4.2.12 Conclusion for Roberts' Algorithm

Roberts' algorithm has been shown here to be a useful method of generating hidden-line drawings of complex objects, although there is considerable labor involved in setting up the necessary data for a given object; e.g., fitting equations to all the planes. The method used to fit planes to the turbine blade is described in Appendix E. Because of the necessary preliminaries, it is not likely that one could write a general program which would accept coordinate data for a given object and generate a hidden line drawing directly. The first problem one would encounter is how to divide the object into planar segments which are all convex. Nonetheless, one could easily adapt most of the subroutines used to draw the blade and write a new main program. The windowing scheme could also be adapted to the object. Note that there is no requirement that all windows be the same size.

Roberts' algorithm has the added advantage that it is no trick at all to generate cutaway drawings as shown by the partial blades drawn in Figures 20 through 22. A particularly appealing

application would be to show the internal geometry of a cooled turbine blade. Similarly, one could draw such machines as aircraft or automobiles and remove panels to show the internal machinery.

4.3 A Second Line-Hiding Algorithm

4.3.1 Introduction to LXL

A second, conceptually simple, line-hiding algorithm was developed as a parallel effort to the coding of Roberts' algorithm. This was done mainly because of the pessimistic reviews of the latter in the literature.

The present algorithm (which we will refer to as "LXL" for "line against line") is based to some extent on Warnock's algorithm and borrows intact some of Warnock's subprograms. Warnock's algorithm was tried and found to be unsatisfactory for our purposes, not only because computing times were too long, but also because it tended to draw lines in short segments (sometimes dot-by-dot). This wasted plotter time and gave poor looking lines due to plotter digital-to-analog converter inaccuracy.

LXL was developed to correct this latter problem by analyzing each line in turn. For a line "I" other lines are tested to see if they lie between the viewer and line I, and if their images on the screen plane intersect the image of line I. An ordered list of intersections is assembled for the line I and each line segment between intersections is tested to see if it is visible. The visible segments are drawn and the program moves on to the next line. Contiguous visible segments are drawn with a single pen movement.

This basic procedure was improved by dividing the scene into smaller parts with "intermediate windows." This helps increase speed since we only need to test lines and polygons which are

in the same intermediate window as the subject line.

There are few restrictions on the kinds of figures which can be drawn, aside from memory and time limitations:

a) The figure must lie wholely in front of the viewpoint.

b) Except for quadrilaterals, all polygons must be planar. Quadrilaterals may be nonplanar.

4.3.2 A Little More Detail

In a little more detail, the LXL algorithm works as follows:

1.a) A "large window rectangle" (LWR) is defined which just encloses the projections of all the data points on the screen plane. This is done so that the picture can be sized to fit the screen and also for efficient processing.

1.b) This LWR is subdivided into M x M equal "intermediate window rectangles" (IWR), where M depends on the number of lines, N_L, in the figure. A simple optimization gave

$$M = \text{integer part } (0.5 + NL^{1/4})$$

as a good choice. The analysis of the optimum IWR size is given in Section 4.3.4. For each IWR:

2.a) A list of unprocessed lines, whose projections intersect* the IWR, is compiled. We designate these lines as "I" lines -- they are the lines which will be processed in this IWR.

2.b) A new rectangular window, JWR, is found

---

* We will use the word "intersects" to mean that the projection of a line or polygon on the screen plane lies at least partly inside the window.

which just encloses all the "I" lines. All lines and polygons which affect the visibility of the "I" lines intersect JWR.

2.c) A list of all lines which intersect the JWR is compiled. These lines are designated "J" lines. This list includes all lines which intersect the "I" lines.

2.d) A list of all polygons which intersect the JWR is compiled. The polygons are designated "J" polygons. This list includes all polygons which cover any parts of the "I" lines.

For each "I" line in the IWR:

3.a) A small window rectangle (SWR) is constructed which encloses the"I" line, except for its end points. The line's end points are excluded from the SWR to avoid unnecessary consideration of lines which are connected to the "I" line.

3.b) From the "J" polygon list, a list is compiled of polygons which intersect the SWR and lie in front of the "I" line. These are designated "S" polygons. Polygons which have the "I" line as an edge are excluded from this list, because we assume that a polygon cannot hide any of its own edges.*

3.c) If the SWR is completely covered by an "S" polygon, then the "I" line is completely hidden--we just mark it "processed" and move on to the next "I" line. Otherwise--

3.d) If no "S" polygons are found, then the line is completely visible--we draw it, mark it "processed" and move on to the next "I" line. Otherwise--

3.e) We check each "J" line against the "I" line,

---

* This is not necessarily true if the polygon is non-planar.

to see if their projections on the screen plane intersect. If so, and if the "J" line lies in front of the "I" line, we store the intersection point.

3.f) If no intersections are found, we draw the line, mark it processed, and move on to the next line. Otherwise--

3.g) The intersections are arranged in an ordered list from one end of the "I" line to the other. We consider the "I" line to be cut into segments by the intersections.

3.h) If the mid-point of a segment is visible (not covered by an "S" polygon), we note it.

3.i) After all segments have been checked, we draw the visible portions of the line. Contiguous visible segments are drawn with single pen movements.

3.j) We mark the "I" line "processed" and move on to the next line.

2.e) After all "I" lines in the current IWR have been processed, we move on to the next IWR.

1.c) After all IWR's have been processed, the figure is complete.

4.3.3 Things to Come

In the following sections we will discuss some of the features of the LXL algorithm and its subprograms. First, the optimization of the IWR size is presented. This is followed by a brief paragraph on the input data format. Then we cover in some detail the "windowing" technique used throughout the algorithm and the way in which we define the surfaces of twisted (non-planar) quadrilaterals. Finally, some results, conclusions, and recommendations are presented.

4.3.4 Sizing the Intermediate Windows

The optimum number of intermediate windows is determined by a rather simple analysis. The basic assumption of this analysis is that the computer spends most of its time in a "windowing" subprogram named CLIP. (CLIP is used to determine whether a line intersects a window.) Numerical experiments show that this assumption is good--the computer spends about half of its time in CLIP.

We now simply minimize the number of calls to CLIP. A rough count of the number of calls is obtained from the code:

$$N_{CLIP} = M^2 \cdot [\bar{N}_E \cdot N_P + 2 \cdot N_L] + N_L \cdot \bar{N}_E \cdot \bar{N}_{P_{JWR}} + (\bar{N}_I + 1) \cdot N_L \cdot \bar{N}_{P_{SWR}} \quad (4\text{-}17)$$

Where M^2 is the number of IWR's

N_P is the number of polygons

$\bar{N}_E$ is the average number of edges per polygon

N_L is the number of lines

$\bar{N}_{P_{JWR}}$ is the average number of polygons in a JWR

$\bar{N}_I$ is the average number of intersections per line

$\bar{N}_{P_{SWR}}$ is the average number of polygons in an SWR

Since we usually make up figures from quadrilaterals, let $\bar{N}_E = 4$. Our experience has indicated that $\bar{N}_I$ is between 1 and 2, so let's take $\bar{N}_I = 2$. Then we may write the last two terms of Eqn (4-17) as

$$4 \cdot N_L \cdot [\bar{N}_{P_{JWR}} + 3/4 \cdot \bar{N}_{P_{SWR}}]$$

Since an average SWR is much smaller than an average JWR, $\bar{N}_{P_{SWR}} << \bar{N}_{P_{JWR}}$ and we can ignore $\bar{N}_{P_{SWR}}$. (Again, experience indicates $\bar{N}_{P_{SWR}} \simeq 3$) If the number of polygons is very large,

$$\bar{N}_{P_{JWR}} \doteq N_P/M^2$$

so

$$N_{CLIP} \doteq M^2 \cdot [4 \cdot N_P + 2 \cdot N_L] + 4 \cdot N_L \cdot N_P/M^2$$

Finally, for figures made up of quadrilaterals,

$$N_P \doteq N_L/2$$

and we have

$$N_{CLIP} \doteq 4 \cdot N_L \cdot [M^2 + N_L/M^2]$$

N_{CLIP} is minimized if $M = N_L^{1/4}$. With this choice,

$$N_{CLIP} \doteq 8 \cdot N_L^{3/2}$$

so

$$T \simeq N_L^{3/2}$$

Numerical experiments gave the final, practical choice of

$$M = \text{int}\,[0.5 + N_L^{1/4}]$$

It is worthwhile to compare this with the case in which intermediate windows are not used. In this case

$$N_{CLIP} \doteq N_L \cdot \bar{N}_E \cdot N_P + (\bar{N}_I + 1) \cdot N_L \cdot \bar{N}_{P_{SWR}}$$

which, using the same arguments as above, gives

$$N_{CLIP} \doteq 2 \cdot N_L^2$$

The ratio of times

$$(T_{IWR}/T_O) \doteq 4 \cdot N_L^{-1/2}$$

clearly shows the advantage of using intermediate windows, since there may be hundreds or thousands of lines in a figure.

4.3.5 Data Formats

Data are input to the program in three lists:

1. A point list which contains the perspective coordinates (x/z, y/z, z) of pertinent points in the figure. (Perspective coordinates are used throughout the code).

2. A line list which contains the identifying subscripts of the two points in the point list which are the end points of each line in the figure.

3. A polygon list which contains the identifying subscripts of the lines which form the edges of each polygon surface element in the figure.

After the data are input, the initialization section of the code rearranges the elements of the polygon list so that the edges of each polygon are in order around the circumference. Simultaneously, it appends to the line list additional lines so that each polygon edge is a distinct line not used for any other polygon edge. This is done so that the end points of an edge may be switched to provide an orderly progression of vertices around the polygon. That is, the second end point of one edge must be the first end point of the next edge, and so on. The "windowing" technique requires this ordering. These additional lines are not used in forming the "I" and "J" line lists.

4.3.6 Windowing

A "windowing" technique is used to perform much of the LXL algorithm - processing lines for the IWR's and JWR's, and processing polygons for the JSR's, SWR's, and line segment visibility. The windowing method we use is borrowed, essentially intact, from Warnock's code as presented in Reference 2.

The windowing sections perform two basic tasks:

1. For a line - determine if any part of it lies inside the window.

2. For a polygon - determine if any part of any edge lies inside the window and, if not, whether the polygon "surrounds" the window. The first part of the polygon task is the same as the first task.

Most of the windowing tasks are performed in a sub-program named CLIP. The first section of CLIP, "ICLIP", determines if a line or edge intersects the window. The second section, "ANGLE", finds the angle between a line's end points, measured from the center of the window.

ICLIP divides the screen plane into 9 regions by extending the sides of the window. Each region is assigned a unique 4 bit code as shown in Figure 23:

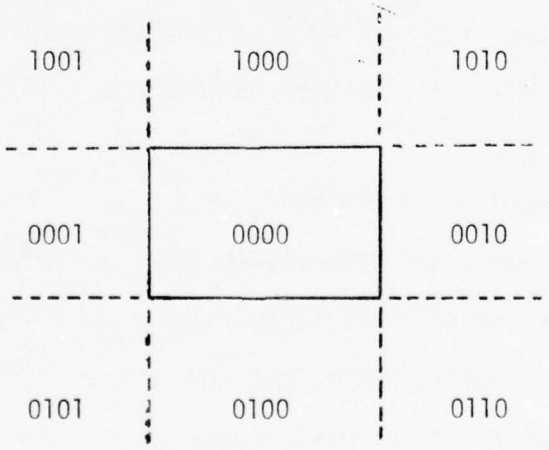


FIGURE 23. ICLIP Codes for Regions of Screen Plane

The following steps are then performed:

a) Determine the codes of both end-points of the line. Call them C_1 and C_2.

b) If C_1 and C_2 are both 0000, then the line lies wholly within the window, so we can set the indicator "ICLIP" to .TRUE. and exit to the calling program. Otherwise -

c) If either C_1 or C_2 is 0000, then the line intersects the window frame once. We must then determine the intersection point before setting "ICLIP" to .TRUE. and exiting. Otherwise -

d) If the Boolean product $C_1 \cdot C_2$ is not 0000, then the line cannot cross the window, so we can set "ICLIP" to .FALSE. and exit. (This is why the 4-bit code was chosen.) Otherwise -

e) Clip off any parts of the line that do not fall between the vertical window frame edges and go through steps a) - d) with the remaining line segment. Then, if we do not exit -

f) Clip off any parts of the segment that do not fall between the horizontal window frame edges. Then go through steps a) - d) with this remaining line segment. An exit will occur. For example, consider the three lines, A, B and C, in Figure 24a:

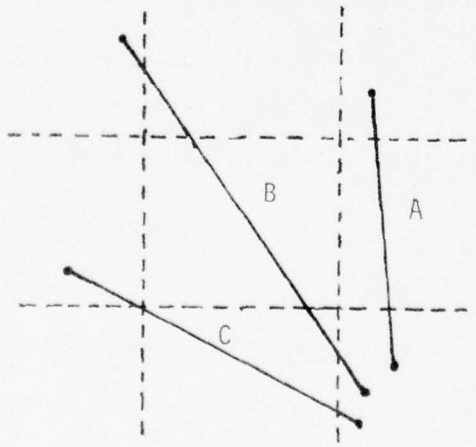


FIGURE 24a. Example of ICLIP Operation

Line A is eliminated by Step d) because the two's bits for both end points are "on."

Lines B and C are not immediately eliminated by Steps a) - d) and so are clipped by Step d) to give the segments shown in Figure 24b.

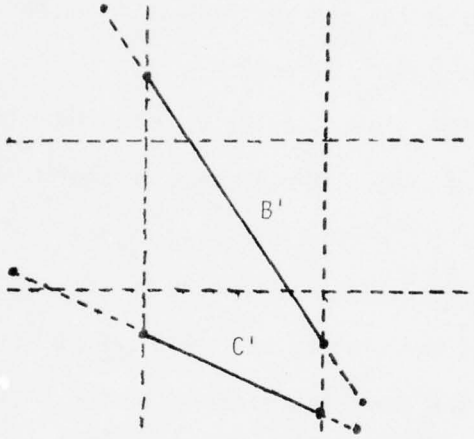


FIGURE 24b. Example of ICLIP Operation (Cont.)

Now segment B' is eliminated by Step d) - The end points have the same code. Segment C' is not eliminated so we go to Step e) which gives the segment C'' shown in Figure 24c.

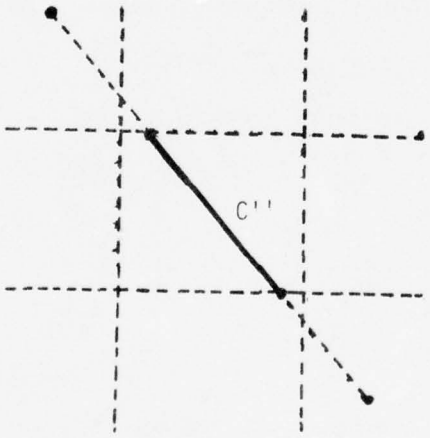


FIGURE 24c. Example of ICLIP Operation (Concl.)

Segment C'' is properly identified by Step b).

The reason for the "ANGLE" section is the following:

If the sides of a polygon are taken, in order, proceeding from vertex to vertex, the the sum of the angles subtended by successive vertices and an arbitrary point will be 0° if the point lies outside the polygon and ±360° if the point lies inside.

Determining the angle subtended by the edge if it does not intersect the window is performed by the "ANGLE" section in a rather crude manner since we don't need the actual angle, only a consistent system of counters which will enable us to determine whether a polygon completely covers the window.

To do this, we again subdivide the screen plane as above and assign to the subdivisions the counters shown in Figure 25.

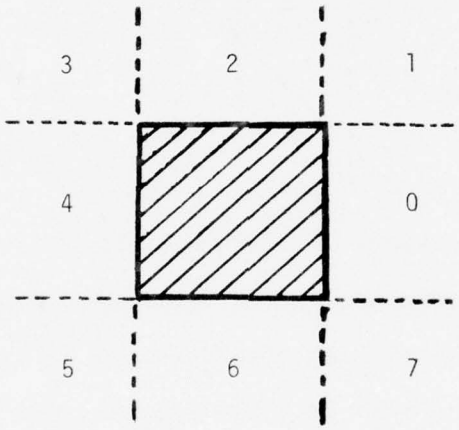


FIGURE 25. ANGLE Codes for Regions of Screen Plane

Then, basically, we define the angle subtended by a line as the difference between the counters for the regions in which the line's end points lie. For example, a line with its first end point in Region 1 and its second end point in Region 4 subtends an angle of 3.

This simple definition must be adjusted because it allows angles of magnitude greater than 4 (which represents 180°) -

for example, a line running from Region 7 to Region 0. This is corrected by adding or subtracting 8 (360°) to produce a result between -4 and +4.

A second problem arises if the end points of the line lie in diagonally opposite regions (1 and 5, or 3 and 7). For example, consider Lines A and B in Figure 26:

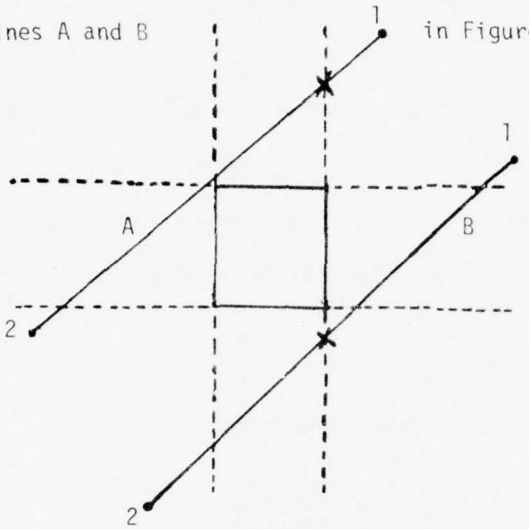


FIGURE 26. Example of ANGLE Ambiguity

Both lines are calculated to have the same angle (4). However, the angle for line B should be -4. This problem is due to the fact that we can't tell which side of the window the lines fall on. The solution to this ambiguity is to cut the line into two segments and sum the angles of the segments. The best way to do this is to cut the line at one of the window frame lines; say, the right-hand vertical frame line as shown. Doing this to line B gives, for the segment from end 1 to the frame line, an angle of -3 and, for the rest of the line, an angle of -1. Then the total angle is -4, which is correct.

(If we sum the angles subtended by the edges of a polygon for a particular window, the sum will be either 0 or $\pm$ 8. If +8,

the polygon covers the window, if 0, it doesn't.

The CLIP subprogram is used as follows:

a) For lines, only the "ICLIP" section is used.

b) For edges of polygons, the "ICLIP" section is executed and only if ICLIP is returned .FALSE., then the "ANGLE" section is executed.

4.3.7 Non-Planar Quadrilaterals

In order to find out whether a polygon hides a point of interest (e.g., the center of a line segment) we must find out if the intersection with the polygon's surface of a ray from the observer through the point of interest is beyond the point. For non-quadrilateral polygons, this is easy since they must be planar. However, for convenience, we have allowed quadrilaterals to be non-planar, which complicates things a bit.

We usually prefer that solid figures be constructed of quadrilateral surface elements, as opposed to, e.g., triangles. But quadrilaterals are not, in general, planar. Therefore, in order to find the distance to the surface of a quadrilateral, we must first decide on a convention for defining the shape of the surface.

We choose to define the quadrilateral surface to be generated by straight lines as illustrated in Figure 27:

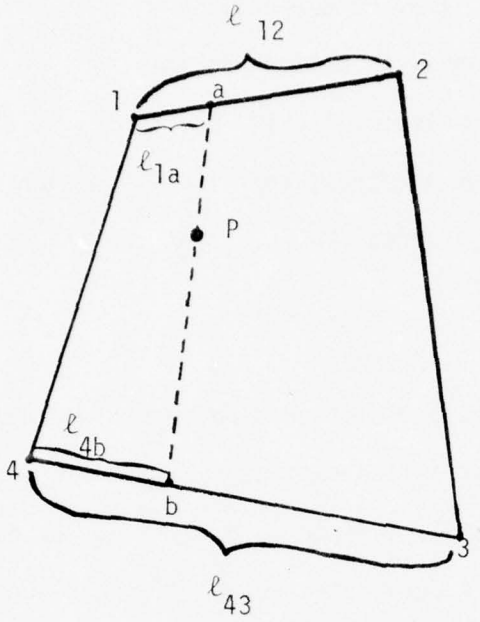


FIGURE 27. Definition of Quadrilateral Surface

A straight line (ab) lies in the surface if points a and b are defined such that

$$\ell_{1a} / \ell_{12} = \ell_{4b} / \ell_{43}$$

It can easily be shown that the same surface is defined if equivalent straight lines are drawn between the other two sides. Thus, this definition does not depend on the choice of sides.

With this surface shape, we can now determine the distance to a point p on the surface. From the Figure 27, we have

$$\begin{aligned} \vec{r}_a &= \vec{r}_1 + \alpha\,(\vec{r}_2 - \vec{r}_1) \\ \vec{r}_b &= \vec{r}_4 + \alpha\,(\vec{r}_3 - \vec{r}_4) \\ \vec{r}_p &= \vec{r}_b + \beta\,(\vec{r}_a - \vec{r}_b) \end{aligned} \qquad (4\text{-}19)$$

where $\vec{r}$ is the vector from the viewpoint to the point subscripted and

$$\alpha = \ell_{1a}/\ell_{12}$$

$$\beta = \ell_{bp}/\ell_{ba}$$

The vector equations represent nine equations with nine unknowns; x_a, y_a, z_a, x_b, y_b, z_b, α, β, z_p.

The solution of these equations is complicated by the fact that we are given, not x_p and y_p, but the projection of $\underline{p}$ on the screen plane, $\tilde{x}_p = x_p/z_p$ and $\tilde{y}_p = y_p/z_p$. If we eliminate $\vec{r}_a$ and $\vec{r}_b$, and let

$$\vec{r}_p' = \vec{r}_p/z_p \qquad (\tilde{x}_p, \tilde{y}_p, 1)$$

then equations (4-19) reduce to the non-linear vector equation

$$z_p \vec{r}_p' = \beta [\alpha\vec{D} + (\vec{r}_1 - \vec{r}_4)] + \alpha (\vec{r}_3 - \vec{r}_4) + \vec{r}_4$$

where, for convenience, $\vec{D} = -\vec{r}_1 + \vec{r}_2 - \vec{r}_3 + \vec{r}_4$. Now β is eliminated by taking the vector product of both sides of this equation with

$$\alpha\vec{D} + (\vec{r}_1 - \vec{r}_4)$$

which gives, after a bit of manipulation,

$$z_p \vec{r}_p' \times [\alpha\vec{D} + (\vec{r}_1 - \vec{r}_4)] = [\alpha(\vec{r}_3 - \vec{r}_4) + \vec{r}_4] \times [\alpha(\vec{r}_2 - \vec{r}_1) + \vec{r}_1] \quad (4\text{-}20)$$

Next, the left-hand side of the equation is eliminated by taking the scalar product of both sides and r_p to give

$$\vec{r}_p' \cdot \left\{ [\alpha(\vec{r}_3 - \vec{r}_4) + \vec{r}_4] \times [\alpha(\vec{r}_2 - \vec{r}_1) + \vec{r}_1] \right\} = 0 \qquad (4\text{-}21)$$

This is a quadratic equation* for α which is solved for the root,

---

* This equation may be written in terms of $\vec{r}_a$ and $\vec{r}_b$ as

$\vec{r}_p' \cdot (\vec{r}_b \times \vec{r}_a) = 0$ which is a mathematical statement of the fact that the vector from the viewpoint to point $\underline{p}$ must lie in the plane defined by the vectors from the viewpoint to point $\underline{b}$ and to point $\underline{a}$.

$0 \leq \alpha \leq 1$. The distance to the surface, z_p, is then found by solving any one of the three scalar equations represented by the vector equation (4-20).

(A twisted quadrilateral, viewed from some directions can have two intersections with a line-of-sight. In this case, both roots of equation (4-20) will be between 0 and 1. The solution which gives the smaller distance from the viewpoint is used.)

4.3.8 Discussion of LXL

The LXL algorithm, as outlined above, was coded in FORTRAN to run on a CDC 6600 computer system. As initially coded, it was found to execute an order of magnitude faster than Warnock's algorithm for the same figure. An additional doubling in speed was achieved by recoding selected subprograms, including "CLIP," in CDC COMPASS Assembly Language. This allowed us to take full advantage of the central processing unit's parallel architecture.

As an example, the surface of the aircraft in Figure 28 contains 1100 lines and 690 polygons.† It took 75 CPU seconds to process. (As a point of interest, the CLIP subprogram was called 300,000 times.)

The use of non-planar quadrilaterals is both a strength and a weakness of LXL. It allows the user flexibility in defining figures and satisfies our preference for figures with quadrilateral surface elements. It does, however, have some drawbacks, the most serious of which is the fact that a non-planar quadrilateral may hide a line which none of its edges intersect in the screen plane.

---

†The data lists for this drawing were borrowed from Capt R. R. Black of the Air Force Flight Dynamics Laboratory.

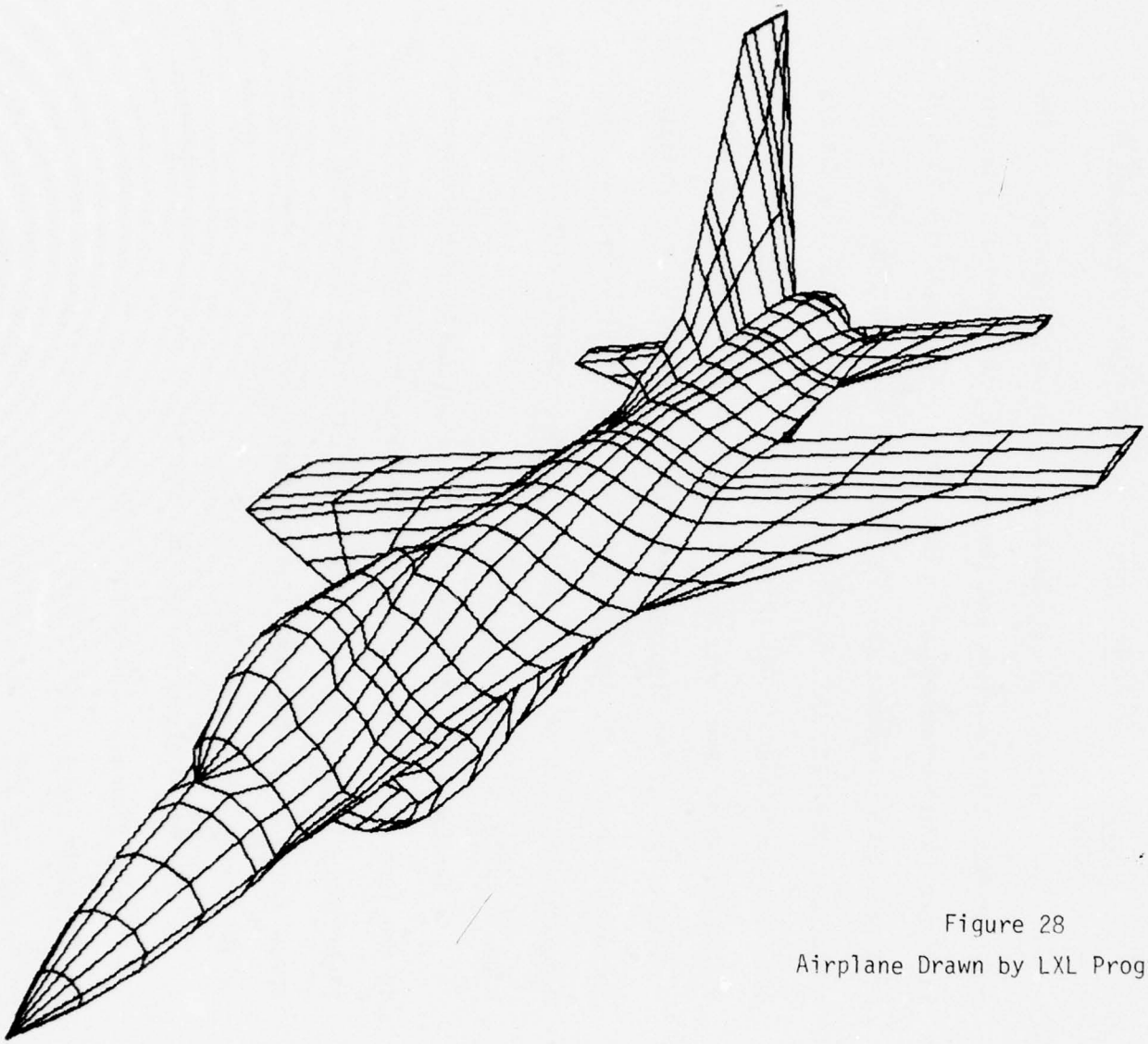


Figure 28
Airplane Drawn by LXL Program

The code cannot handle this. We do not think this problem is significant if we limit ourselves to quadrilaterals which are nearly planar. In fact, we haven't noticed any occurrences of this in any of our experiments.

Appendix F contains a listing of the LXL code and a sample data set.

4.4 Comparison of Performances: Roberts Algorithm vs. LXL Algorithm

In an attempt to determine the relative performances of the code for Robert's algorithm and the LXL code, the turbine blade drawing in Figure 17 was processed using both codes. The resultant drawings were identical in appearance. The LXL code was about 25% faster in execution. However, this is misleading for two reasons: 1) the LXL code obtained a significant speed advantage because assembly language code was used in some sections - Robert's algorithm was coded in FORTRAN only. 2) For figures constructed principally of quadrilaterals, the execution time for the LXL code is proportional to $N_{LINES}^{1.5}$, for Robert's algorithm, $N_{LINES}^{1.1}$. This means that Robert's algorithm has the edge in speed, at least potentially.

The LXL algorithm does, however, enjoy considerable advantages in the form of the input data and in the ease of understanding the algorithm. The simple form of the input data could offset the execution speed disadvantage when complicated figures are to be processed (such as the aircraft of Figure 28).

Robert's algorithm has, potentially, a major speed advantage for complex figures. (We estimate the execution time for Robert's algorithm of the aircraft figure to be about 40 seconds - compared to 75 seconds for LXL.) Unfortunately, this speed advantage is offset by

the considerable amount of work (and ingenuity) required of the user to break up the figure into convex polyhedra.

As might be expected, the authors were unable to agree on which of the two algorithms was "better."

References

1. B. R. Heap, et al, Three Contouring Algorithms, NPL-DNAM-81, National Physical Lab, Teddington, England, December 1969.

2. William M. Newman and Robert F. Sproull, Principles of Interactive Computer Graphics, New York, McGraw-Hill Book Company, 1973.

3. L. G. Roberts, Machine Perception of Three Dimensional Solids, in Optical and Electro-Optical Information Processing, Tippet, et al editors, MIT Press, 1965.

APPENDIX A

Listings of LEVEL1 and LEVEL2

The basic versions of the two contouring subroutines are listed herein. Also listed is a driver program, ONE, which was used to create Figures 2 and 3.

The listings which appear in this and the other appendices contain occasional continuation lines marked with a dollar sign. These were created by the listing program to maintain the right-hand margin and do not appear in the actual code. A line containing two or more statements separated by dollar signs is real, however; CDC FORTRAN Extended accepts this as well as statements of the form A = B = C = D = 0.0 and symbolic names of up to seven characters. The forms READ* and PRINT* are list-directed (format-free) input-output, used primarily for interactive programs.

```
      PROGRAM ONE (PLOT,OUTPUT=402B,TAPE6=OUTPUT)
      DIMENSION X(6),Y(5),Z(6,5),H(7)
      LOGICAL U(6,5)
      DATA X / 0.0,1.0,2.0,3.0,4.0,5.0 /
      DATA Y / 0.0,1.0,2.0,3.0,4.0 /
      DATA (Z(I,1),I=1,6) / 2.0,3.0,6.0,7.0,4.0,3.0 /
      DATA (Z(I,2),I=1,6) / 4.0,5.0,9.0,7.0,6.0,4.0 /
      DATA (Z(I,3),I=1,6) / 4.0,6.0,8.0,5.0,6.0,5.0 /
      DATA (Z(I,4),I=1,6) / 3.0,5.0,7.0,6.0,6.0,3.0 /
      DATA (Z(I,5),I=1,6) / 3.0,4.0,6.0,5.0,4.0,2.0 /
      DATA H / 2.5,3.5,4.5,5.5,6.5,7.5,8.5 /
      U(1,1) = .FALSE.
      CALL PLOT (2.0,4.0,-3)
      CALL PLOTPM ("PM PLAIN PAPER PLEASE.",3)
      I = 6
      J = 5
      K = 7
      IGO = 1
      GO TO 3
    1 CALL LEVEL1 (X,Y,Z,H,U,I,J,K)
      CALL PLOT (8.5,0.0,-3)
      IGO = 2
      GO TO 3
    2 CONTINUE
      CALL LEVEL2 (X,Y,Z,H,U,I,J,K)
      CALL PLOT (8.5,0.0,-3)
      CALL SYMBOL (0.0,0.5,0.105,"FINISHED",90.0,8)
      CALL PLOTE
      STOP
C          DRAW BOUNDARY
    3 CALL PLOT (X(6),Y(1),2)
      CALL PLOT (X(6),Y(5),2)
      CALL PLOT (X(1),Y(5),2)
      CALL PLOT (X(1),Y(1),2)
      GO TO (1,2), IGO
      END
```

```
      SUBROUTINE LEVEL1 (XM,YM,MESH,HGTS,UNUSED,P,Q,R)
      INTEGER P,Q,R
      REAL MESH
      COMMON /CUE/ OPEN,FIRST,LAST,H,POLAR
      DIMENSION XM(P),YM(Q),HGTS(R),MESH(P,Q),UNUSED(P,Q)
      LOGICAL FIRST,LAST,OPEN,UNUSED,POLAR
C
C          THIS PROCEDURE IS USED FOR CONTOURING OVER A
C          RECTANGULAR MESH.  A MORE DETAILED DESCRIPTION
C          IS GIVEN IN SECTION 4 OF THE ORIGINAL REPORT.
C
C          ADAPTED FROM ALGOL PROCEDURE CONTOUR1
C          B.R. HEAP, ET AL 'THREE CONTOURING ALGORITHMS',
C          NPL-81, NATIONAL PHYSICAL LAB,
C          TEDDINGTON ENGLAND, DEC. 1969, P. 14
C
      POLAR = UNUSED(1,1)
      JM = Q-1
      IM = P-1
C
C          EACH CONTOUR HEIGHT IS DEALT WITH IN TURN.
C
      DO 7 K=1,R
      H = HGTS(K)
C
C          THE ARRAY UNUSED IS SET UP FOR THIS HEIGHT.
C
      DO 1 J=2,JM
      DO 1 I=2,IM
      UNUSED(I,J) = MESH(I-1,J) .LT. H .AND. MESH(I,J) .GE.
     $  H
    1 CONTINUE
C
C          THE BOUNDARY OF THE MESH IS SCANNED FOR THE
C          BEGINNING OF ANY OPEN CONTOUR OF HEIGHT H.
C
      OPEN = .TRUE.
      DO 2 I=2,P
      IF(MESH(I-1,1) .LT. H .AND. MESH(I,1) .GE. H)  CALL
     1 FOLLOW (I,1,-1,0,XM,YM,MESH,P,Q,UNUSED)
    2 CONTINUE
      DO 3 J=2,Q
      IF(MESH(P,J-1) .LT. H .AND. MESH(P,J) .GE. H)  CALL
     1 FOLLOW (P,J,0,-1,XM,YM,MESH,P,Q,UNUSED)
    3 CONTINUE
      DO 4 L=1,IM
      I = P-L
      IF(MESH(I+1,Q) .LT. H .AND. MESH(I,Q) .GE. H)  CALL
     1 FOLLOW (I,Q,1,0,XM,YM,MESH,P,Q,UNUSED)
    4 CONTINUE
```

```
      DO 5 L=1,JM
      J = Q-L
      IF(MESH(1,J+1) .LT. H .AND. MESH(1,J) .GE. H)  CALL
     1 FOLLOW (1,J,0,1,XM,YM,MESH,P,Q,UNUSED)
    5 CONTINUE
C
C          THE ARRAY 'UNUSED' IS SCANNED FOR THE
C          BEGINNING OF ANY CLOSED CONTOUR OF HEIGHT H.
C
      OPEN = .FALSE.
      DO 6 L=2,JM
      J = Q-L+1
      DO 6 M=2,IM
      I = P-M+1
      IF(UNUSED(I,J))  CALL FOLLOW (I,J,-1,0,XM,YM,MESH,P,Q
     $ ,UNUSED)
    6 CONTINUE
    7 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FOLLOW (IF,JF,IAF,JAF,XM,YM,MESH,P,Q,UNUSE
     $ D)
      INTEGER P,Q,TEMP
      LOGICAL OPEN,FIRST,LAST,UNUSED,POLAR
      REAL MESH
      DIMENSION XM(P),YM(Q),MESH(P,Q),UNUSED(P,Q)
C
C          THE PROCEDURE STARTS BY SETTING UP THE INITIAL
C          VALUES OF VARIOUS QUANTITIES.
C
      COMMON /CUE/ OPEN,FIRST,LAST,H,POLAR
      FIRST = .TRUE.
      LAST  = .FALSE.
      I  = IF
      J  = JF
      IA = IAF
      JA = JAF
      Z  = MESH(I,J)
      ZA = MESH(I+IA,J+JA)
C
C          NEXT POINT
C
C          THE POSITION OF THE POINT T WHERE THE CONTOUR
C          CUTS BETWEEN (I,J) AND (I+IA,J+JA) IS
C          CALCULATED USING INVERSE LINEAR
C          INTERPOLATION.
C
    1 T = 0.0
      IF(Z .NE. ZA)  T = (Z-H)/(Z-ZA)
      X = XM(I)-T*(XM(I)-XM(I+IA))
      Y = YM(J)-T*(YM(J)-YM(J+JA))
C
C          TESTS ARE NOW MADE TO SEE IF T IS THE LAST POINT
C          ON THE CONTOUR.
C
      IF(OPEN)  GO TO 2
      IF(IA .EQ. -1 .AND. .NOT. UNUSED(I,J))  LAST = .TRUE.
      GO TO 3
    2 IF(FIRST)  GO TO 4
      IF(JA .EQ. 0 .AND. (J .EQ. 1 .OR. J .EQ. Q))  LAST =
     $ .TRUE.
      IF(JA .NE. 0 .AND. (I .EQ. 1 .OR. I .EQ. P))  LAST =
     $ .TRUE.
    3 IF(LAST)  GO TO 4
C
C          THE MARKER IN THE ARRAY UNUSED
C          IS CANCELLED IF NECESSARY.
C
      IF(IA .EQ. -1)  UNUSED(I,J) = .FALSE.
C
```

```
C          NEW POINT
C
C          THE COORDINATES OF T ARE OUTPUT
C          TO THE PROCEDURE DRAW.
C
    4 CALL DRAW (X,Y,POLAR,FIRST,LAST)
      IF(LAST)  RETURN
C
C          TESTS ARE NOW MADE TO SEE WHICH OF THE MESH
C          LINES THE CONTOUR CROSSES NEXT.
C
C          THE VALUES OF Z, ZA, I, IA, AND JA ARE
C          ADJUSTED BEFORE GOING ON TO FIND A NEW POINT T.
C
      ZB = MESH(I+JA,J-IA)
      IF(ZB .GE. H)  GO TO 5
      ZA = ZB
      TEMP = IA
      IA = JA
      JA = -TEMP
      GO TO 7
    5 ZC = MESH(I+IA+JA,J-IA+JA)
      IF(ZC .GE. H)  GO TO 6
      Z  = ZB
      ZA = ZC
      I  = I+JA
      J  = J-IA
      GO TO 7
    6 Z  = ZC
      I  = I+IA+JA
      J  = J-IA+JA
      TEMP = JA
      JA = IA
      IA = -TEMP
    7 FIRST = .FALSE.
      GO TO 1
      END
```

```
      SUBROUTINE LEVEL2 (XM,YM,MESH,HGTS,UNUSED,P,Q,R)
      INTEGER P,Q,R
      REAL MESH
      COMMON /CUE/ OPEN,FIRST,LAST,H,POLAR
      LOGICAL FIRST,LAST,OPEN,UNUSED,POLAR
      DIMENSION XM(P),YM(Q),HGTS(R),MESH(P,Q),UNUSED(P,Q)
C
C         THIS PROCEDURE IS USED FOR CONTOURING OVER A
C         RECTANGULAR MESH, AND IS A MORE ELABORATE ROUTINE
C         THAN CONTOUR 1.  A MORE DETAILED DESCRIPTION IS
C         GIVEN IN SECTION 5 OF THE ORIGINAL REPORT.
C
C         ADAPTED FROM ALGOL PROCEDURE CONTOUR2,
C       B.R. HEAP  ET AL, 'THREE CONTOURING ALGORITHMS',
C         NPL - DNAM - 81, NATIONAL PHYSICAL LAB.,
C         TEDDINGTON, ENGLAND, DEC. 1969, P 14.
C
      POLAR = UNUSED(1,1)
      JM = Q-1
      IM = P-1
C
C         EACH CONTOUR HEIGHT IS DEALT WITH IN TURN.
C
      DO 7 K=1,R
      H = HGTS(K)
C
C         THE ARRAY UNUSED IS SET UP FOR THIS HEIGHT.
C
      DO 1 J=2,JM
      DO 1 I=2,IM
      UNUSED(I,J) = MESH(I-1,J) .LT. H .AND. MESH(I,J) .GE.
     $  H
    1 CONTINUE
C
C         THE BOUNDARY OF THE MESH IS SCANNED FOR THE
C         BEGINNING OF ANY OPEN CONTOUR OF HEIGHT H.
C
      OPEN = .TRUE.
      DO 2 I=2,P
      IF(MESH(I-1,1) .LT. H .AND. MESH(I,1) .GE. H)  CALL
     1 PURSUE (I,1,-1,0,XM,YM,MESH,P,Q,UNUSED)
    2 CONTINUE
      DO 3 J=2,Q
      IF(MESH(P,J-1) .LT. H .AND. MESH(P,J) .GE. H)  CALL
     1 PURSUE (P,J,0,-1,XM,YM,MESH,P,Q,UNUSED)
    3 CONTINUE
      DO 4 L=1,IM
      I = P-L
      IF(MESH(I+1,Q) .LT. H .AND. MESH(I,Q) .GE. H)  CALL
     1 PURSUE (I,Q,1,0,XM,YM,MESH,P,Q,UNUSED)
```

```
    4 CONTINUE
      DO 5 L=1,JM
      J = Q-L
      IF(MESH(1,J+1) .LT. H .AND. MESH(1,J) .GE. H)  CALL
     1 PURSUE (1,J,0,1,XM,YM,MESH,P,Q,UNUSED)
    5 CONTINUE
C
C          THE ARRAY UNUSED IS SCANNED FOR THE BEGINNING
C        OF ANY CLOSED CONTOUR OF HEIGHT H.
C
      OPEN = .FALSE.
      DO 6 L=2,JM
      J = Q-L+1
      DO 6 M=2,IM
      I = P-M+1
      IF(UNUSED(I,J))  CALL PURSUE (I,J,-1,0,XM,YM,MESH,P,Q
     $ ,UNUSED)
    6 CONTINUE
    7 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PURSUE (IF,JF,IAF,JAF,XM,YM,MESH,P,Q,UNUSE
     $ D)
      INTEGER P,Q,TEMP
      LOGICAL OPEN,FIRST,LAST,UNUSED,RIGHT,POLAR
      REAL MESH
      DIMENSION XM(P),YM(Q),MESH(P,Q),UNUSED(P,Q)
C
C         THIS PROCEDURE IS USED TO FOLLOW A CONTOUR OF A
C         GIVEN HEIGHT THROUGH THE MESH.
C
      COMMON /CUE/ OPEN,FIRST,LAST,H,POLAR
C
C         THE PROCEDURE BEGINS BY SETTING UP THE INITIAL
C         VALUES OF VARIOUS QUANTITIES.
C
      FIRST = .TRUE.
      LAST  = .FALSE.
      I  = IF
      J  = JF
      IA = IAF
      JA = JAF
      Z  = MESH(I,J)
      ZA = MESH(I+IA,J+JA)
C
C         NEXT POINT.
C
    1 T = 0.0
      IF(Z .NE. ZA)  T = (Z-H)/(Z-ZA)
C
C          THE POSITION OF THE POINT WHERE THE CONTOUR
C         CROSSES OA (SEE FIG. 5) IS CALCULATED USING
C         INVERSE LINEAR INTERPOLATION.
C
      X = XM(I)-T*(XM(I)-XM(I+IA))
      Y = YM(J)-T*(YM(J)-YM(J+JA))
C
C         TESTS ARE NOW MADE TO SEE IF T IS THE LAST POINT
C         ON THE CONTOUR.
C
      IF(OPEN)  GO TO 2
      IF(IA .EQ. -1 .AND. .NOT. UNUSED(I,J))  LAST = .TRUE.
      GO TO 3
    2 IF(FIRST)  GO TO 4
      IF(JA .EQ. 0 .AND. (J .EQ. 1 .OR. J .EQ. Q))  LAST =
     $ .TRUE.
      IF(JA .NE. 0 .AND. (I .EQ. 1 .OR. I .EQ. P))  LAST =
     $ .TRUE.
    3 IF(LAST)  GO TO 4
C
C         THE MARKER IN THE ARRAY 'UNUSED' IS CANCELLED
```

```
C          IF NECESSARY.
C
      IF(IA .EQ. -1)  UNUSED(I,J) = .FALSE.
C
C          NEW POINT.
C
C          THE COORDINATES OF T ARE OUTPUT TO THE
C          PROCEDURE DRAW.
C
    4 CALL DRAW (X,Y,POLAR,FIRST,LAST)
      IF(LAST)  RETURN
      FIRST = .FALSE.
C
C          THE HEIGHTS OF THE POINTS B AND C (SEE FIG. 5)
C          ARE EXTRACTED AND THE HEIGHT OF D CALCULATED AS
C          THE AVERAGE OF THE HEIGHTS OF O, A, B, AND C.
C
      ZB = MESH(I+JA,J-IA)
      ZC = MESH(I+IA+JA,J-IA+JA)
      ZD = 0.25*(Z+ZA+ZB+ZC)
C
C          A TEST IS MADE TO SEE WHICH OF THE LINES OD OR AD
C          THE CONTOUR CROSSES.
C
      IF(ZD .GE. H)  GO TO 7
C
C          THE CONTOUR CROSSES OD AND THE COORDINATES OF THE
C          POINT WHERE THIS HAPPENS ARE CALCULATED USING
C          INVERSE LINEAR INTERPOLATION AND ARE OUTPUT TO
C          THE PROCEDURE DRAW.
C
      RIGHT = .TRUE.
      T = 0.0
      IF(Z .NE. ZD)  T = 0.5*(Z-H)/(Z-ZD)
      X = XM(I)-T*(XM(I)-XM(I+IA+JA))
      Y = YM(J)-T*(YM(J)-YM(J+JA-IA))
      CALL DRAW (X,Y,POLAR,FIRST,LAST)
C
C          A TEST IS NOW MADE TO SEE WHETHER THE CONTOUR
C          CROSSES OB AS IN CASE (II) OF FIG. 6.  IF SO, A
C          JUMP IS MADE TO THE LABEL 'TURN RIGHT'.
C
      IF(ZB .LT. H)  GO TO 8
C
C          TOP RIGHT.
C
    5 T = 0.0
      IF(ZB .NE. ZD)  T = 0.5*(ZB-H)/(ZB-ZD)
C
C          THE CONTOUR CROSSES BD AND THE COORDINATES OF
```

```
C          POINT WHERE THIS HAPPENS ARE CALCULATED USING
C          INVERSE LINEAR INTERPOLATION AND ARE OUTPUT TO
C          THE PROCEDURE DRAW.
C
      X = XM(I+JA)-T*(XM(I+JA)-XM(I+IA))
      Y = YM(J-IA)-T*(YM(J-IA)-YM(J+JA))
      CALL DRAW (X,Y,POLAR,FIRST,LAST)
C
C          A TEST IS MADE TO SEE IF THE SITUATION IS AS IN
C          CASE (V) OF FIG. 6.  IF SO A JUMP IS MADE TO THE
C          LABEL 'TURN RIGHT'.
C
      IF(.NOT. RIGHT)  GO TO 8
C
C          A TEST IS MADE TO SEE IF THE CONTOUR CROSSES BC
C          AS IN CASE (IV) OF FIG. 6.  IF SO, A JUMP IS MADE
C          TO THE LABEL 'STRAIGHT'.
C
      IF(ZC .LT. H)  GO TO 9
C
C          TOP LEFT.
C
    6 T = 0.0
C
C          THE CONTOUR CROSSES CD AND THE COORDINATES OF THE
C          POINT WHERE THIS HAPPENS ARE CALCULATED USING
C          INVERSE LINEAR INTERPOLATION AND ARE OUTPUT TO
C          THE PROCEDURE DRAW.
C
      IF(ZD .NE. ZC)  T = (ZD-H)/(ZD-ZC)
      X = 0.5*(XM(I)*(1.0-T)+XM(I+IA+JA)*(1.0+T))
      Y = 0.5*(YM(J)*(1.0-T)+YM(J+JA-IA)*(1.0+T))
      CALL DRAW (X,Y,POLAR,FIRST,LAST)
C
C          A TEST IS MADE TO SEE WHETHER THE SITUATION IS AS
C          IN CASE (VI) OF FIG. 6.  IF SO, A JUMP IS MADE
C          TO THE LABEL 'TURN LEFT'.
C
      IF(RIGHT)  GO TO 10
C
C          A TEST IS MADE TO SEE IF THE CONTOUR CROSSES BC
C          AS IN CASE (III) OF FIG. 6.  IF SO, A JUMP IS
C          MADE TO THE LABEL 'STRAIGHT';  OTHERWISE, A JUMP
C          IS MADE TO THE LABEL 'TOP TIGHT'.
C
      IF(ZB .LT. H)  GO TO 5
      GO TO 9
    7 RIGHT = .FALSE.
C
C          THE CONTOUR CROSSES AD AND THE COORDINATES OF THE
```

```
C          POINT WHERE THIS HAPPENS ARE CALCULATED USING
C          INVERSE LINEAR INTERPOLATION AND ARE OUTPUT TO
C          THE PROCEDURE DRAW.
C
      T = 0.0
      IF(ZA .NE. ZD)  T = 0.5*(ZA-H)/(ZA-ZD)
      X = XM(I+IA)-T*(XM(I+IA)-XM(I+JA))
      Y = YM(J+JA)-T*(YM(J+JA)-YM(J-IA))
      CALL DRAW (X,Y,POLAR,FIRST,LAST)
C
C          A TEST IS MADE TO SEE IF THE CONTOUR CROSSES AC
C          AS IN CASE (I) OF FIG. 6.  IF SO, A JUMP IS MADE
C          TO THE LABEL 'TURN LEFT'; OTHERWISE, A JUMP IS
C          MADE TO THE LABEL 'TOP LEFT'.
C
      IF(ZC .LT. H)  GO TO 6
      GO TO 10
C
C          TURN RIGHT.
C
    8 ZA = ZB
C
C          THE CONTOUR EXITS FROM THE BASIC RECTANGLE BY
C          CROSSING OB.  THE VALUES OF ZA, IA, AND JA ARE
C          UPDATED BEFORE GOING ON TO FIND A NEW POINT T.
C
      TEMP = IA
      IA = JA
      JA = -TEMP
      GO TO 1
C
C          STRAIGHT.
C
    9 Z  = ZB
C          THE CONTOUR EXITS FROM THE BASIC RECTANGLE BY
C          CROSSING BC.  THE VALUES OF Z, ZA, I, AND J ARE
C          UPDATED BEFORE GOING ON TO FIND A NEW POINT T.
C
      ZA = ZC
      I  = I+JA
      J  = J-IA
      GO TO 1
C
C          TURN LEFT.
C
   10 Z  = ZC
C
C          THE CONTOUR EXITS FROM THE BASIC RECTANGLE BY
C          CROSSING AC.  THE VALUES OF Z, I, J, IA, AND JA
C          ARE UPDATED BEFORE GOING ON TO FIND
```

```
C          A NEW POINT T.
C
      I   = I+IA+JA
      J   = J-IA+JA
      TEMP = JA
      JA = IA
      IA = -TEMP
      GO TO 1
      END
```

```
      SUBROUTINE DRAW (X,Y,POLAR,FIRST,LAST)
      LOGICAL POLAR,FIRST,LAST
      IF(POLAR)  GO TO 2
      XP = X
      YP = Y
    1 IF(FIRST)  CALL PLOT (XP,YP,3)
      CALL PLOT (XP,YP,2)
      RETURN
    2 XP = X*COS(Y)
      YP = X*SIN(Y)
      GO TO 1
      END
```

APPENDIX B

Some Special Versions of Subroutine DRAW

Four modifications of DRAW are listed here to show how easily one can adapt the basic contouring package. The first version was used to produce the Joukowski airfoil shown in Figure 5; Figure 4 was produced with the basic package using the same data. The second version untransforms streamlines from the X,η plane to the X,Y plane and draws them, as in Figure 6. The third version does not draw the contours directly but rather accumulates the coordinates in X, Y, and Z arrays until the contour is complete, then passes the transformations described in Section 3. Thus was Figure 6 produced. The fourth version shows how symbols might be added to the first and last points on the contours. The common block COUNT would be added to LEVEL1 or LEVEL2 and the variable NCT made to correspond to the loop counter K.

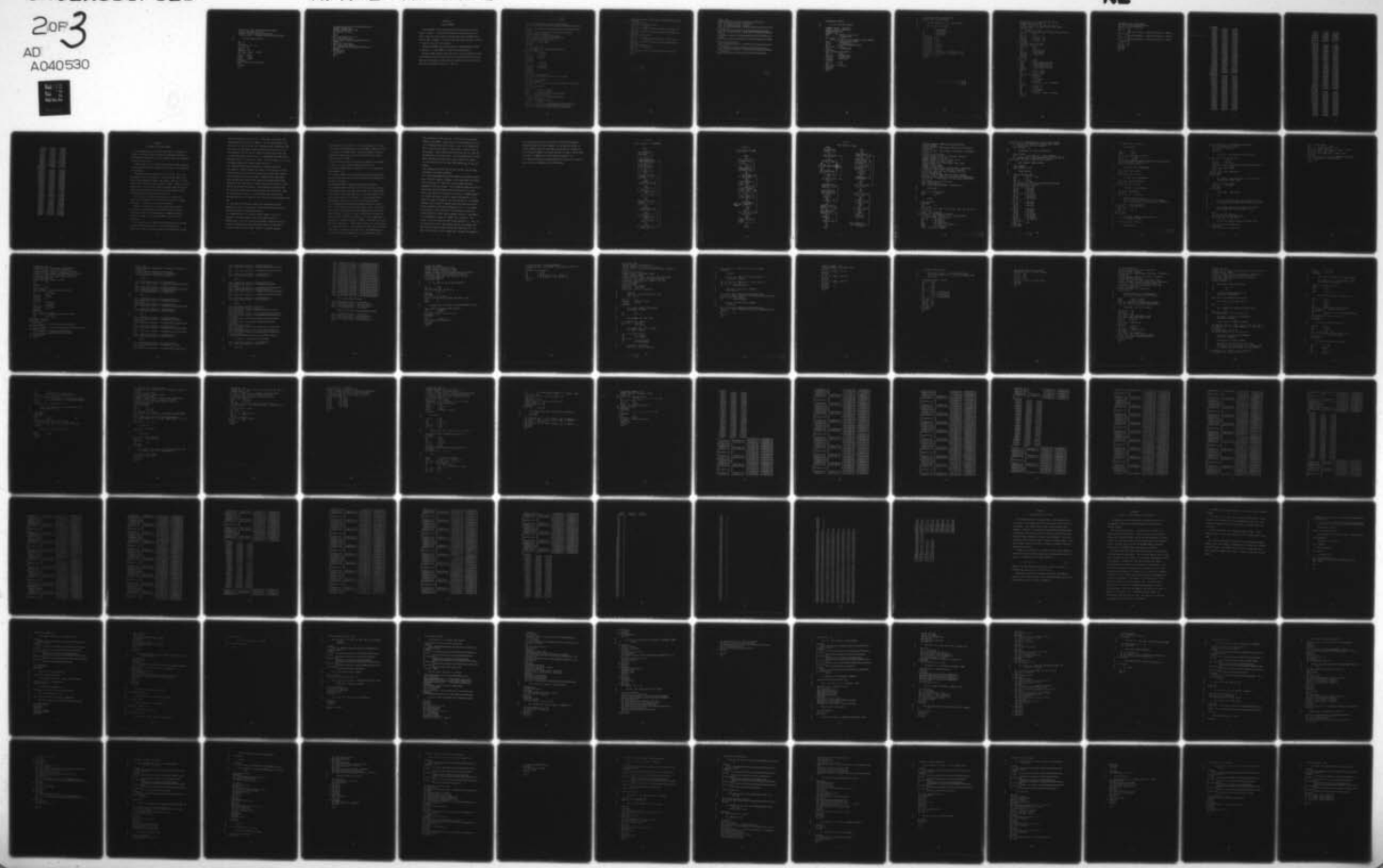
```
      SUBROUTINE DRAW (X,Y,POLAR,FIRST,LAST)
      COMMON /TRFORM/ EX,EY,SA,CA,RAM
C
C          MODIFIED FOR JOUKOWSKI TRANSFORMATION.
C
      LOGICAL POLAR,FIRST,LAST
      IF(POLAR)  GO TO 2
      I = 2
      IF ((X*X+Y*Y) .LT. RAM)  I=3
      A    = X+EX
      B    = Y+EY
      D    = A*A+B*B
      IF(D) 3,4
    3 XP   = A+(A*CA-B*SA)/D
      YP    = B-(A*SA+B*CA)/D
      GO TO 1
    4 XP   = A
      YP   = B
    1 IF(FIRST)  CALL PLOT (XP,YP,3)
      CALL PLOT (XP,YP,I)
      IF(.NOT. LAST)  RETURN
      PRINT 101
  101 FORMAT(1H )
      RETURN
    2 XP = X*COS(Y)
      YP = X*SIN(Y)
      GO TO 1
      END
```

```
      SUBROUTINE DRAW (AX,AY,POLAR,FIRST,LAST)
      COMMON /0/ X(61),YL(61),YU(61),SY
      LOGICAL POLAR,FIRST,LAST
C
      PS      = ATKN(X,YU,61,2,AX)
      SS      = ATKN(X,YL,61,2,AX)
      XP      = AX
      YP      = (SS + AY*(PS - SS))*SY
      IF (FIRST) 1,2
    1 CALL PLOT (XP,YP,3)
      RETURN
    2 CALL PLOT (XP,YP,2)
      RETURN
      END
```

AD-A040 530 AIR FORCE AERO PROPULSION LAB WRIGHT-PATTERSON AFB OHIO F/G 12/1
CONTOURING AND HIDDEN-LINE ALGORITHMS FOR VECTOR GRAPHIC DISPLA--ETC(U)
JAN 77 J S PETTY, K D MACH

UNCLASSIFIED AFAPL-TR-77-3 NL

2 OF 3
AD
A040530



```
      SUBROUTINE DRAW (X,Y,POLAR,FIRST,LAST)
      LOGICAL POLAR,FIRST,LAST
      COMMON /XYDRAW/ XC(100),YC(100),KC
      COMMON /MORE/   PX(100),PY(100),PZ(100),ZCONT
C
C          ACCUMULATING VERSION
C
      XP = X
      YP = Y
      IF (FIRST)  KC = 0
      KC = KC + 1
      XC(KC) = XP
      YC(KC) = YP
      IF (.NOT. LAST)  RETURN
      DO 1 I = 1,KC
      PY(I)  = YC(I)
      PX(I)  = XC(I)
      PZ(I)  = ZCONT
    1 CONTINUE
      CALL DISPLAY (PX,PY,PZ,KC)
      KC = 0
      RETURN
      END
```

```
      SUBROUTINE DRAW (X,Y,POLAR,FIRST,LAST)
      COMMON /COUNT/ NCT
      LOGICAL POLAR,FIRST,LAST
      IF (POLAR) GO TO 3
      XP=X
      YP=Y
    1 IF (.NOT.FIRST) GO TO 2
      CALL PLOT (XP,YP,3)
      CALL SYMBOL (XP,YP,0.105,NCT,0.0,-1)
      RETURN
    2 CALL PLOT (XP,YP,2)
      IF (.NOT.LAST) RETURN
      CALL SYMBOL (XP,YP,0.105,NCT,0.0,-1)
      RETURN
    3 XP=X*COS(Y)
      YP=X*SIN(Y)
      GO TO 1
      END
```

APPENDIX C

Program SHOWOFF

Program SHOWOFF and its subroutines were used to draw the turbine blade in Figure 11. The BLOCK DATA subroutine contains data only for the fir tree; the blade coordinate data was read from a permanent disk file via TAPE 4, though it could have been read from card input. The data is listed at the end of this appendix.

Subroutine PREPARE sets up the coordinate transformation described in Section 3.3., using ROTMAT to compute the rotation matrix.

Subroutine RAM transforms the input data to the eye coordinate system, using subroutine PHI to perform the matrix multiplication, then converts the result to perspective coordinates and computes the scale factors for plotting as described in Sections 3.4 and 3.5.

```
      PROGRAM SHOWOFF (INPUT,OUTPUT,TAPE4,PLOT)
      COMMON XB(5,27),YB(5,27),ZB(5,27),RXB(5,27),RYB(5,27)
     $ ,
     1 RZB(5,27),RX(21,2),RY(21,2),RZ(21,2),X(27),YP(27),YS
     $ (27)
      COMMON /OBJECT/ XFT(21,2),YFT(21,2),ZFT(21,2)
      COMMON /VIEW/ XV,YV,ZV
      COMMON /FRAME/ XF,YF,AF,BF,CF,DF
      COMMON /HOW/ PRSP,XMIN,XMAX,YMIN,YMAX
      DATA XF,YF,PRSP /2*6.0,.TRUE. /
      XSCALE(X)  = DF + CF*X
      YSCALE(Y)  = BF + AF*Y
      READ 4
      DO 3 K     = 1,5
      READ (4,100) Z
      READ (4,100) (X(I),YS(I),YP(I),I=1,27)
  100 FORMAT (3E10.5)
      DO 1 I     = 1,27,2
      J          = 1 + I/2
      XB(K,J)    = X(I)
      YB(K,J)    = YP(I)
      ZB(K,J)    = Z
    1 CONTINUE
      DO 2 I     = 28,53,2
      J          = 1 + I/2
      XB(K,J)    = X(54-I)
      YB(K,J)    = YS(54-I)
      ZB(K,J)    = Z
    2 CONTINUE
    3 CONTINUE
      CALL PLOT (1.0,2.5,-3)
      CALL PLOTPM ("PM PLAIN PAPER, PLEASE.",3)
    4 CONTINUE
      PRINT *, "VIEWPOINT:"
      READ *, XV,YV,ZV
      IF (XV .EQ. 0.0 .AND. YV .EQ. 0.0 .AND. ZV .EQ. 0.0)
     $ GO TO 8
      PRINT *, XV,YV,ZV
      CALL PREPAR
      XMAX       = YMAX = -1.0E6
      XMIN       = YMIN = 1.0E6
      CALL RAM (XFT,YFT,ZFT,RX,RY,RZ,21,2,21)
      PRINT *, "BETWEEN RAMS"
      CALL RAM (XB,YB,ZB,RXB,RYB,RZB,5,27,5)
C         DRAW FIGURE
      PRINT *, "BEGINNING FIGURE"
      DO 5 I     = 1,20
      CALL PLOT (XSCALE(RX(I+1,2)),YSCALE(RY(I+1,2)),3)
      CALL PLOT (XSCALE(RX(I,2)),YSCALE(RY(I,2)),2)
      CALL PLOT (XSCALE(RX(I,1)),YSCALE(RY(I,1)),2)
```

```
      CALL PLOT (XSCALE(RX(I+1,1)),YSCALE(RY(I+1,1)),2)
    5 CONTINUE
C
C         DRAW BLADE
      PRINT *, "BEGINNING BLADE"
      DO 6 I      = 1,27
      CALL PLOT (XSCALE(RXB(1,I)),YSCALE(RYB(1,I)),3)
      DO 6 K      = 2,5
      CALL PLOT (XSCALE(RXB(K,I)),YSCALE(RYB(K,I)),2)
    6 CONTINUE
      PRINT *, "MID BLADE"
      DO 7 K      = 1,5
      CALL PLOT (XSCALE(RXB(K,1)),YSCALE(RYB(K,1)),3)
      DO 7 I      = 2,27
      CALL PLOT (XSCALE(RXB(K,I)),YSCALE(RYB(K,I)),2)
    7 CONTINUE
      PRINT *, "FINISHED"
      CALL PLOT (8.5,0.0,-3)
      GO TO 4
    8 CALL SYMBOL (0.5,0.0,0.105,"FINISHED",90.0,8)
      CALL PLOTE
      STOP
      END
```

```
      BLOCK DATA
      COMMON /OBJECT/ XFT(21,2),YFT(21,2),ZFT(21,2)
      DATA (XFT(I,1),I=1,21) / 21*-0.1 /
      DATA (XFT(I,2),I=1,21) / 21*1.2  /
      DATA (YFT(I,1),I=1,21) / 2*0.9,0.85,0.75,0.8,0.7,0.75
     $ ,0.65,0.7,
     1 0.6,0.5,0.4,0.45,0.35,0.4,0.3,0.35,0.25,2*0.2,0.9 /
      DATA (YFT(I,2),I=1,21) / 2*0.48052,0.43052,0.33052,0.
     $ 38052,
     1 0.28052,0.33052,0.23052,0.28052,0.18052,0.08052,-0.0
     $ 1948,
     2 0.03052,-0.06948,-0.01948,-0.11948,-0.06948,-0.16948
     $ ,
     3 2*-0.21948,0.48052 /
      DATA (ZFT(I,1),I=1,21) / 10.0,2*9.5,9.45,9.4,9.35,9.3
     $ ,9.25,9.2,
     1 2*9.15,9.2,9.25,9.3,9.35,9.4,9.45,2*9.5,2*10.0 /
      DATA (ZFT(I,2),I=1,21) / 10.0,2*9.5,9.45,9.4,9.35,9.3
     $ ,9.25,9.2,
     1 2*9.15,9.2,9.25,9.3,9.35,9.4,9.45,2*9.5,2*10.0 /
      END
```

```
      SUBROUTINE PREPARE
C
C          SET UP ROTATION MATRIX
C
      COMMON /VIEW/    XV,YV,ZV
      COMMON /MATRIX/ RMAT(4,4)
      DIMENSION FM(3,3)
      LOGICAL ZERO
      ZERO        = ABS(YV) .LT. 1.0E-5
      F           = -SIGN(1.0,YV)
      IF (ZERO)  F = 1.0
C         ROTATE SO THAT ZE AXIS POINTS AT OLD ORIGIN
      HYP         = SQRT(XV*XV + YV*YV)
      ALPHA       = -ATAN(ZV/HYP)
      BETA        = 1.5707963*SIGN(1.0,XV)
      IF (ZERO)  GO TO 1
      BETA        = -ATAN(XV/YV)
    1 CONTINUE
      CALL ROTMAT (ALPHA,BETA)
      DO 2 J      = 1,3
      FM(J,1)     = F*RMAT(J,1)
      FM(J,2)     = F*RMAT(J,3)
      FM(J,3)     = RMAT(J,2)
    2 CONTINUE
      DO 3 I      = 1,3
      DO 3 J      = 1,3
      RMAT(I,J)   = FM(I,J)
    3 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ROTMAT (ALPHA,BETA)
      COMMON /MATRIX/ RMAT(4,4)
C
C          ASSUMES ARGUMENTS TO BE IN RADIANS.
C
C          LEFT - HANDED VERSION
C
      SA          = SIN(ALPHA)
      CA          = COS(ALPHA)
      SB          = SIN(BETA)
      CB          = COS(BETA)
      RMAT(1,1)   = CB
      RMAT(1,2)   = 0.0
      RMAT(1,3)   = SB
      RMAT(2,1)   = SA*SB
      RMAT(2,2)   = CA
      RMAT(2,3)   = -SA*CB
      RMAT(3,1)   = -SB*CA
      RMAT(3,2)   = SA
      RMAT(3,3)   = CA*CB
      RMAT(1,4)   = RMAT(2,4) = RMAT(3,4) = 0.0
      RMAT(4,1)   = RMAT(4,2) = RMAT(4,3) = 0.0
      RMAT(4,4)   = 1.0
      RETURN
      END
```

```
      SUBROUTINE RAM (X,Y,Z,RX,RY,RZ,M,N,NT)
      COMMON /VIEW/ XV, YV, ZV
      COMMON /FRAME/ XF, YF, AF, BF, CF, DF
      COMMON /HOW/ PRSP, XMIN, XMAX, YMIN, YMAX
      LOGICAL PRSP
C           ROTATE AND SCALE
      REAL X(M,N),Y(M,N),Z(M,N),RX(M,N),RY(M,N),RZ(M,N)
      DO 1 J      = 1,N
      DO 1 I      = 1,M
      RX(I,J)     = X(I,J) - XV
      RY(I,J)     = Y(I,J) - YV
      RZ(I,J)     = Z(I,J) - ZV
    1 CONTINUE
      CALL PHI (RX,RY,RZ,M,N)
      IF (.NOT. PRSP) GO TO 3
      DO 2 J      = 1,N
      DO 2 I      = 1,M
      OZ          = 1.0/RZ(I,J)
      RX(I,J)     = RX(I,J)*OZ
      RY(I,J)     = RY(I,J)*OZ
      RZ(I,J)     = - OZ
    2 CONTINUE
    3 DO 4 J      = 1,N
      DO 4 I      = 1,NT
      YMIN        = AMIN1(YMIN,RY(I,J))
      YMAX        = AMAX1(YMAX,RY(I,J))
      XMIN        = AMIN1(XMIN,RX(I,J))
      XMAX        = AMAX1(XMAX,RX(I,J))
    4 CONTINUE
      DY          = YMAX - YMIN
      DX          = XMAX - XMIN
      IF (DX .GT. DY) GO TO 5
      RXF         = XF*DX/DY
      AF          = YF/DY
      BF          = - AF*YMIN
      CF          = RXF/DX
      DF          = 0.5*(XF - RXF) - CF*XMIN
      RETURN
    5 RXF         = YF*DY/DX
      CF          = XF/DX
      DF          = - CF*XMIN
      AF          = RXF/DY
      BF          = 0.5*(YF - RXF) - AF*YMIN
      RETURN
      END
```

```
      SUBROUTINE PHI (X,Y,Z,M,N)
      DIMENSION X(M,N),Y(M,N),Z(M,N)
      COMMON /MATRIX/ RMAT(4,4)
      DO 1 I = 1,M
      DO 1 J = 1,N
      XP      = RMAT(1,1)*X(I,J) + RMAT(1,2)*Y(I,J) + RMAT(1
     $ ,3)*Z(I,J)
      YP      = RMAT(2,1)*X(I,J) + RMAT(2,2)*Y(I,J) + RMAT(2
     $ ,3)*Z(I,J)
      ZP      = RMAT(3,1)*X(I,J) + RMAT(3,2)*Y(I,J) + RMAT(3
     $ ,3)*Z(I,J)
      X(I,J) = XP
      Y(I,J) = YP
      Z(I,J) = ZP
    1 CONTINUE
      RETURN
      END
```

10.00000		
.00573	.33619	.33619
.01087	.36977	.32202
.02174	.38457	.33102
.03261	.39871	.33959
.04348	.41223	.34775
.05435	.42516	.35551
.10870	.48153	.38908
.16305	.52572	.41504
.21740	.55929	.43447
.27175	.58334	.44804
.32610	.59862	.45617
.38045	.60570	.45909
.43480	.60495	.45687
.48915	.59652	.44946
.54350	.58083	.43667
.59785	.55759	.41815
.65220	.52680	.39339
.70655	.48826	.36179
.76090	.44160	.32285
.81525	.38632	.27660
.86960	.32212	.22370
.92395	.24993	.16505
.97830	.17157	.10143
1.03265	.08881	.03349
1.06526	.03762	-.00914
1.07613	.02035	-.02365
1.08153	-.01095	-.01095
11.31200		
-.05794	.38100	.38100
-.05367	.41870	.36277
-.04237	.43489	.37327
-.03107	.45031	.38316
-.01976	.46498	.39251
-.00846	.47895	.40134
.04805	.53916	.43879
.10456	.58536	.46683
.16108	.61952	.48702
.21759	.64298	.50030
.27410	.65668	.50718
.33061	.66127	.50792
.38713	.65719	.50255
.44364	.64472	.49088
.50015	.62397	.47250
.55666	.59494	.44673
.61318	.55747	.41271
.66969	.51126	.36972
.72620	.45584	.31791
.78271	.39049	.25832
.83923	.31464	.19219

.89574	.22862	.12049
.95225	.13365	.04398
1.00876	.03133	-.03680
1.04267	-.03291	-.03715
1.05397	-.05473	-.10423
1.05902	-.09059	-.09059
12.62500		
-.12328	.44971	.44971
-.11826	.48846	.43195
-.10653	.50393	.44050
-.09479	.51874	.44870
-.08306	.53289	.45639
-.07132	.54642	.46367
-.01265	.60518	.49429
.04603	.65042	.51652
.10470	.68345	.53141
.16338	.70526	.53957
.22205	.71657	.54129
.28073	.71790	.53664
.33940	.70962	.52547
.39808	.69197	.50737
.45675	.66506	.48169
.51543	.62890	.44757
.57410	.58339	.40429
.63278	.52831	.35202
.69145	.46330	.29180
.75013	.38786	.22492
.80880	.30175	.15241
.86748	.20527	.07506
.92615	.09932	-.00658
.98483	-.01493	-.09206
1.02003	-.08692	-.14506
1.03177	-.11143	-.15300
1.03702	-.14995	-.14995
13.93800		
-.18997	.54827	.54827
-.18285	.58577	.53267
-.17058	.59915	.53767
-.15852	.61193	.54239
-.14635	.62411	.54682
-.13418	.63572	.55098
-.07334	.68555	.55780
-.01251	.72256	.57837
.04833	.74777	.58300
.10917	.76194	.58182
.17001	.76568	.57482
.23084	.75944	.56179
.29168	.74357	.54239
.35252	.71832	.51608
.41336	.68382	.48219

.47419	.64016	.44015
.53503	.58731	.38991
.58587	.52520	.33219
.65671	.45364	.25798
.71754	.37243	.19824
.77838	.28158	.12374
.83922	.18138	.04501
.90006	.07241	-.03747
.96089	-.04458	-.12336
.99740	-.11828	-.17641
1.00956	-.14339	-.19434
1.01535	-.18250	-.18250
15.25000		
-.25800	.65510	.65510
-.24739	.58762	.63853
-.23479	.59711	.63902
-.22219	.70620	.63935
-.20959	.71489	.63953
-.19699	.72318	.63954
-.13399	.75866	.63725
-.07099	.78429	.63098
-.00799	.80020	.62063
.05501	.80550	.60609
.11801	.80326	.58716
.18101	.79057	.56358
.24401	.76849	.53504
.30701	.73706	.50115
.37001	.69632	.46152
.43301	.64628	.41580
.49601	.58696	.36391
.55901	.51834	.30600
.62201	.44041	.24251
.68501	.35341	.17394
.74801	.25799	.10081
.81101	.15512	.02357
.87401	.04593	-.05741
.93701	-.06850	-.14130
.97481	-.13924	-.19396
.98741	-.16313	-.21159
.99397	-.20093	-.20093

APPENDIX D

The Hidden-Line program DRACULA

D.1 Program DRACULA, which was used to draw Figures 17 through 22, is listed on the following pages along with its subroutines (except for RAM, PREPARE, ROTMAT and PHI, which are identical with those in Appendix C) and all necessary data.

DRACULA begins by calling BLDATA, which reads the blade coordinates and plane equations from mass storage on logical unit 4. This operation is not repeated.

Then DRACULA reads the viewpoint (e.g. XV, YV, ZV: 12, 12, 20), the number of tiers to be drawn (NT: 0 thru 4) and the number of segments of the uppermost tier (NSEC: even numbers 2 thru 26). XF and YF are the frame sizes for the final drawing, in inches. PRINTM is a logical variable which if set to .TRUE. causes the rotated volume matrix to be printed and the logical variable WHITE signals the operator to mount plain paper instead of graph paper on the plotter.

The variables NT and NSEC are checked to be sure they do not exceed 4 and 26 respectively, then MANY counts the number of segments to be drawn. The least is 5, for the fir tree alone.

DRACULA calls PREPARE to set up the rotation matrix, ROTPLN to transform the plane equations and transform and scale the line coordinates, WINDOW to assign windows to each segment, and WINLIST to print the window list. Subroutine TIMREM is used to time the computation.

The preliminaries over, DRACULA begins selecting lines to be drawn. Subroutine FINDM decodes the line list entries and returns the segment number in NE and the plane numbers in MV and NV, which are used

for the preliminary visibility test. If the line is not a back line, DRACULA prepares two vectors of length 4 (VB for the beginning of the line and VE for the end of the line) and passes them to subroutine HIDE. The logical variable NEWSEG signals HIDE that the given line is on the same segment as the last and the old DO list can be used or that this is a new segment which needs a new list. When drawing the blade proper, DRACULA calls VSET to prepare the vectors. After all segments of the blade are drawn, DRACULA calls TOPPER to draw a top on it.

D.2 Subroutine BLDATA reads the data file listed at the end of this appendix. Blade coordinates are given as five sets of an X array and two Y arrays, one each for the pressure and suction sides, running from the leading to the trailing edges. BLDATA selects every other X point and steps from front to rear taking Y's from the pressure side, then back to the front taking Y's from the suction side to form a closed loop at each of the five sections. After reading the coordinate data for a section, BLDATA reads the plane equations for that section. When all five sections are read, it reads the matrix and line lists for the blade and finally it reads the line lists for the trailing edge and the top.

D.3 The BLOCK DATA subroutine contains the coordinates and plane equations for the fir tree as well as the plane translation matrix and the plane perspective transformation matrix.

D.4 Subroutine DOLIST is called by HIDE to prepare a list of all segments in the same windows as the current segment, which is denoted by NB. The first loop finds the windows (columns of array IN) occupied by segment NB and the second loop copies all other entries in those columns into array JLIST. MEMSET is a machine language

subroutine used to set JLIST to 0. It can be replaced by a DO loop.

D.5 Subroutine FINDM decodes the line list numbers supplied in L1 and L2 and returns the segment number in NB and the numbers of the two planes whose intersection is the line in MV and NV in the form of column subscripts in FVMAT.

D.6 Subroutine HIDE tests each line given it against all the bodies in JLIST. It is more fully described in Section 4.2.10.

D.7 Function INTHERE returns a value of 1, 2, 3, or 4 according to which window Y is in.

D.8 Subroutine MMIP is used to multiply the 4 by 548 volume matrix by the rotation matrix and store the result back in the volume matrix in order to save space.

D.9 Subroutine MMPY is a general purpose matrix multiplier.

D.10 Subroutine ROTPLN transforms the fir tree and blade coordinates using subroutine RAM. The largest and smallest X and Y from the two sets of data end up in XMIN, XMAX, etc. and are used by RAM, to compute the scale factors as in Section 3.5. The plane equations are transformed using matrices 4-14, 4-15, and the rotation matrix.

D.11 Subroutine SEEK performs the operations described in Section 4.2.7. SD contains the S and D vectors from eq. (4-6). J1 marks the first column of the current body in the rotated volume matrix and J2 marks the number of columns. JS then is the offset or distance from the first column of RVMAT. For instance, if the given segment were represented by columns 48 through 52 of RVMAT, J1 would be 48, J2 would be 5, and JS would be 47. Further details of SEEK are in Section 4.2.10.

D.12 Subroutine TOPPER is called by DRACULA to draw a cap on the blade. If the fourth tier was the last drawn, IW is 1 and TOPPER draws the cap using the top list; otherwise IW is 2 and TOPPER forms the cap

from the bottoms of the next tier. If a partial tier was drawn, DRACULA calls TOPPER a second time to cap the uncovered lower tier.

D.13 Subroutine VALID tests a given t-α pair in the inequalities (4-8). NVQ counts the number of times VALID is called; 50,000 is not uncommon. Usually VALID returns early from the first loop and therefore does not use much time. L1 and L2 mark the pair of lines whose intersection is the current t and α; they are not tested. If t and α survive the first loop, VALID updates t_{min} or t_{max} and returns.

D.14 Subroutine VSET forms the VB and VE vectors from the blade coordinates as directed by DRACULA.

D.15 Subroutine WINDOW assigns each segment of the object to one or more of the four subareas (windows) of the viewing area (see Figure 22) by comparing all Y coordinates of that segment with the Y boundaries of the four windows. The X boundaries need not be checked because the scaling procedure centers the object left to right. Results are recorded in the array IN such that if a segment appears in a window, the segment number is stored in IN (NW,NS), i.e., if segment 53 appears in window 3, IN (3,53) contains 53. CDC FORTRAN Extended allows arbitrarily complex subscripts, hence the peculiar looking statements just before statements 2 and 3. It all amounts to obtaining a Y value from the fir tree coordinates, which are stored sequentially rather than by segment, scaling it, and feeding it to INTHERE which returns an integer from 1 through 4. This integer is our first subscript; the second subscript is L. Two loops are necessary for the fir tree because the upper two segments have four corners on the end planes and the lower three have six. The logic for the blade is similar except that it handles the segments in

pairs (see Section 4.2.10) and it enters a value for the segments immediately above the current segment, for use when drawing the cap. Finally, WINDOW checks for cases where the two ends of a segment are not in adjacent windows, i.e., one end could be in window 1 and the other in window 3. WINDOW fills in the intervening spaces in IN.

D.16 Subroutine WINLIST prints the window list (array IN) so that the user can see which segments are in which windows.

FIGURE 29

Flow Chart of DRACULA

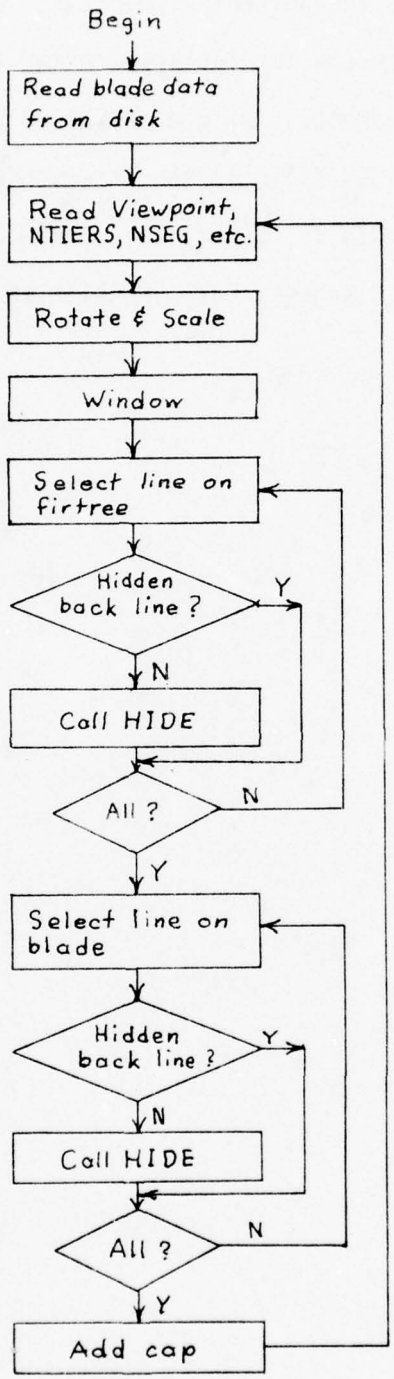


FIGURE 30

Flow Chart of HIDE

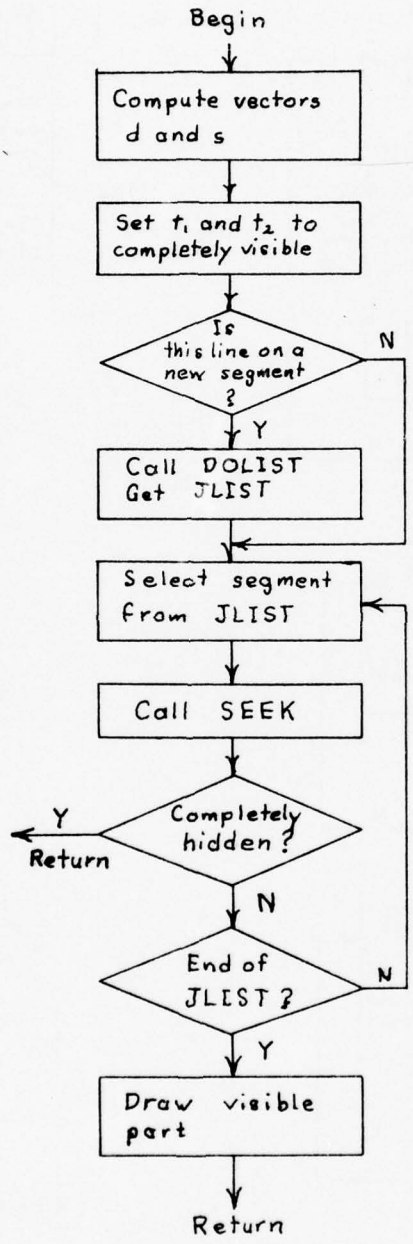
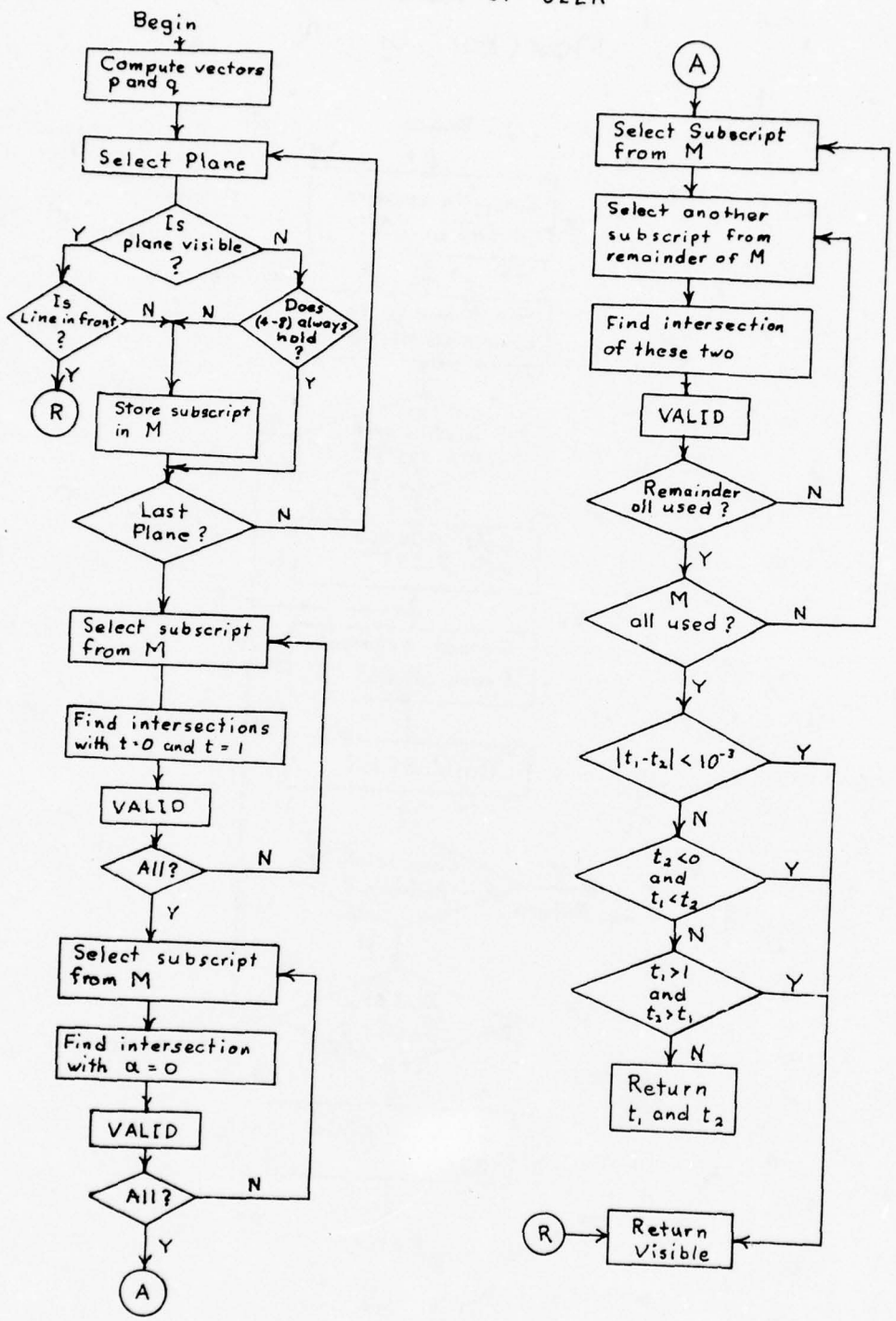


FIGURE 31

Flow Chart of SEEK



```
      PROGRAM DRACULA (INPUT,OUTPUT,TAPE4,PLOT,
     $ TAPE6=OUTPUT)
      COMMON /OBJECT/ VMAT(4,35), XFT(21,2), YFT(21,2),
     $ ZFT(21,2)
      COMMON /LISTS/ MLIST(109,2), LLIST(20,6), LLISTB(4,
     $ 13,8),
     1 LTE(4,2), LTOP(13,4)
      COMMON /BLADE/ XB(5,27), YB(5,27), ZB(5,27)
      COMMON /VIEW/ XV, YV, ZV
      COMMON /FRAME/ XF, YF, AF, BF, CF, DF
      COMMON /EDGE/ NB, VB(4), VE(4)
      COMMON /HOW/ PRSP, XMIN, XMAX, YMIN, YMAX
      COMMON /NVQAL/ NVQ
      COMMON /BODY/ NEWSEG
      COMMON // RVMAT(4,548), BLMAT(4,512), RXB(5,27),
     1 RYB(5,27), RZB(5,27), RX(21,2), RY(21,2),
     2 RZ(21,2), PQ(2,8), M(6), SD(2,4), J1, J2,
     3 T1N, T2N, MANY, NT
      LOGICAL PRSP, HIDDEN, PRINTM, WHITE, NEWSEG
      NAMELIST /STAR/ XV,YV,ZV,NT,NSEC,XF,YF,PRINTM,WHITE
      HIDDEN(M,N) = RVMAT(3,M) .LT. 0.0 .AND. RVMAT(3,N)
     $ .LT. 0.0
      DATA XV,YV,ZV,NSEC / 3*0.0,0 /
      DATA XF,YF / 2*0.0 /
      DATA PRINTM,WHITE,NVIEWS / 2*.FALSE.,0 /
C
C           INITIALIZE
C
      CALL PLOT (1.0,1.0,-3)
      PRSP        = .TRUE.
      NT          = 0
C
C           BEGIN
C
      CALL BLDATA
    1 CONTINUE
      NVQ         = 0
      READ STAR
      IF (XV .EQ. 0.0 .AND. YV .EQ. 0.0 .AND. ZV .EQ. 0.0)
     $ GO TO 30
      PRINT 100, XV,YV,ZV
  100 FORMAT (*1 VIEWPOINT: *,3F10.5)
      IF (WHITE .AND. NVIEWS .LT. 1) CALL PLOTPM (
     $ "PM PLAIN PAPER PLEASE.",3)
      NVIEWS      = NVIEWS + 1
      NT          = MIN0(NT,4)
      NSEC        = MIN0(NSEC,25)
      MANY        = 26*(NT - 1) + NSEC + 5
      MANY        = MAX0(MANY,5)
      PRINT 101, NT,NSEC,MANY
```

```
  101 FORMAT (*0 DRAWING*,I3,* TIERS OF THE BLADE*/
     1 *  AND*,I3* SECTIONS OF THE UPPERMOST TIER*/
     2 *  FOR A TOTAL OF*,I5* BODIES*)
      CALL PREPARE
C
C          TRANSFORM AND SCALE COORDINATES
C
      CALL ROTPLN
      LM          = MLIST(MANY,1) + MLIST(MANY,2) - 1
      IF (PRINTM) PRINT 102, ((RVMAT(I,J),I=1,4),J=1,LM)
  102 FORMAT (*0 VOLUME MATRIX*/(1H 4E15.7))
C
C          SORT SEGMENTS INTO WINDOWS
C
      CALL WINDOW
      CALL WINLIST (MANY)
C
C          DRAW FIRTREE.
C
      CALL TIMREM (TLEFT)
      VB(4)       = VE(4) = 1.0
      MB          = 0
      DO 6 I      = 1,20
      DO 5 J      = 1,2,2
      CALL FINDM (LLIST(I,J),LLIST(I,J+1),NB,MV,NV)
      IF (HIDDEN(MV,NV)) GO TO 5
      IF (J .GT. 4) GO TO 3
      VB(1)       = RX(I,1)
      VB(2)       = RY(I,1)
      VB(3)       = RZ(I,1)
      IF (J .GT. 2) GO TO 2
      VE(1)       = RX(I+1,1)
      VE(2)       = RY(I+1,1)
      VE(3)       = RZ(I+1,1)
      GO TO 4
    2 VE(1)       = RX(I,2)
      VE(2)       = RY(I,2)
      VE(3)       = RZ(I,2)
      GO TO 4
    3 VB(1)       = RX(I,2)
      VB(2)       = RY(I,2)
      VB(3)       = RZ(I,2)
      VE(1)       = RX(I+1,2)
      VE(2)       = RY(I+1,2)
      VE(3)       = RZ(I+1,2)
    4 NEWSEG      = NB .NE. MB
      CALL HIDE
      MB          = NB
    5 CONTINUE
    6 CONTINUE
```

```
      IF (MANY .LE. 5) GO TO 23
C
C           DRAW BLADE.
C
      MB            = 0
      DO 13 K       = 1,NT
      IMAX          = 13
      IF (K .EQ. NT) IMAX = NSEC/2
      DO 12 I       = 1,IMAX
      DO 11 J       = 1,7,2
      CALL FINDM (LLIST3(K,I,J),LLIST3(K,I,J+1),NB,MV,NV)
      IF (HIDDEN(MV,NV)) GO TO 11
      IF (J .GT. 1) GO TO 7
C
C           PRESSURE SIDE, HORIZONTAL
C
      CALL VSET (K,I,K,I+1)
      GO TO 10
    7 IF (J .GT. 3)  GO TO 8
C
C           SUCTION SIDE, HORIZONTAL
C
      CALL VSET (K,29-I,K,27-I)
      GO TO 10
    8 IF (J .GT. 5) GO TO 9
C
C           PRESSURE SIDE, VERTICAL
C
      CALL VSET (K,I,K+1,I)
      GO TO 10
    9 IF (I .EQ. 1)  GO TO 11
C
C           SUCTION SIDE, VERTICAL
C           FOR I = 1, IT IS THE SAME AS THE PRESSURE
C           SIDE (I.E. LEADING EDGE) AND IS NOT REPEATED.
C
      CALL VSET (K,29-I,K+1,29-I)
   10 NEWSEG        = NB .NE. MB
      CALL HIDE
      MB            = NB
   11 CONTINUE
   12 CONTINUE
C
C      LAST VERTICAL BOUNDARY (TRAILING EDGE
C           ON COMPLETED TIER).
C
      IF (K .EQ. NT) GO TO 14
C
C           TRAILING EDGE
C
```

```
   13 CALL FINDM (LTE(K,1),LTE(K,2),NB,MV,NV)
      IF (HIDDEN(MV,NV)) GO TO 18
      CALL VSET (K,14.K+1,14)
      CALL HIDE
      GO TO 18
C
C           VERTICAL LINES FROM NEXT BODY IN TIER
C
   14 IF (IMAX .EQ. 13) GO TO 13
      I           = IMAX + 1
      DO 17 J     = 5, ,2
      NEWSEG      = J .EQ. 5
      CALL FINDM (LLISTB(K,I,J),LLISTB(K,I,J+1),NB,MV,NV)
      IF (J .GT. 5) GO TO 15
      CALL VSET (K,I,K+1,I)
      GO TO 16
   15 CALL VSET (K,28-I,K+1,28-I)
   16 CALL HIDE
   17 CONTINUE
C
C           HORIZONTAL LINE, PRESSURE TO SUCTION SIDE,
C           AT BOTTOM OF NEXT BODY.
C
      CALL VSET (K,28-I,K,I)
      NEWSEG      = .FALSE.
      CALL HIDE
   18 CONTINUE
C
C           SIDES DONE.  NOW FINISH.
C
C           TOP
C
C
C           IF LESS THAN FOUR TIERS ARE DRAWN, THE CAP IS
C           FORMED FROM THE BOTTOMS OF THE NEXT TIER (KTOP =
C     &           1).
C
C           IF ALL FOUR TIERS ARE PRESENT, THE CAP IS
C           FORMED FROM THE TOP LIST (KTOP = 2).
C
      KTOP        = 1
      IF (NT .GE. 4) KTOP = 2
      CALL TOPPER (NTP1,1,IMAX,KTOP)
      IF (IMAX .GE. 13) GO TO 29
C
C           PUT TOP ON UNCOVERED PART OF LOWER TIER.
C
      CALL TOPPER (NT,IMAX+1,13,1)
   29 CONTINUE
      CALL PLOT (XF + 3.0,0.0,-3)
```

```
      CALL TIMREM (TNOW)
      TH          = TLEFT - TNOW
      PRINT *, "HIDDEN TIME = ", TH
      PRINT *, "VALID WAS CALLED ", NVQ, " TIMES"
      IF (TNOW .GT. 5.0) GO TO 1
      PRINT *, "LESS THAN FIVE SECONDS LEFT."
   30 CONTINUE
      CALL SYMBOL (0.5,0.5,0.105,"FINISHED",90.0,8)
      PRINT *, NVIEWS, " VIEWS DRAWN."
      CALL PLOTE
      END
```

```
      SUBROUTINE BLDATA
      COMMON /LISTS/ MLIST(109,2), LLIST(20,6),
     1 LLISTB(4,13,8), LTE(4,2), LTOP(13,4)
      COMMON /BLADE/ XB(5,27), YB(5,27), ZB(5,27)
      COMMON // RVMAT(4,548), BLMAT(4,512), RXB(5,27),
     1 RYB(5,27), RZB(5,27), RX(21,2), RY(21,2),
     2 RZ(21,2), PQ(2,8), M(8), SD(2,4), J1, J2,
     3 T1N, T2N, MANY, NT
      DIMENSION X(27), YS(27), YP(27)
      J1            = 1
      J2            = 128
      REWIND 4
      DO 3 K        = 1,5
      READ (4,100) Z
      READ (4,100) (X(I),YS(I),YP(I),I=1,27)
      DO 1 I        = 1,27,2
      J             = 1 + I/2
      XB(K,J)       = X(I)
      YB(K,J)       = YP(I)
      ZB(K,J)       = Z
    1 CONTINUE
      DO 2 I        = 29,53,2
      J             = 1 + I/2
      XB(K,J)       = X(54-I)
      YB(K,J)       = YS(54-I)
      ZB(K,J)       = Z
    2 CONTINUE
      IF (K .EQ. 5) GO TO 4
      READ (4,101) ((BLMAT(I,J),I=1,4),J=J1,J2)
      J1            = J1 + 128
      J2            = J2 + 128
  100 FORMAT (3F10.5)
  101 FORMAT (4E14.7)
    3 CONTINUE
    4 READ (4,102) ((MLIST(I,J),J=1,2),I=6,109)
  102 FORMAT (2I5)
      READ (4,103) (((LLISTB(K,J,I),I=1,8),J=1,13),K=1,4)
  103 FORMAT (8I5)
      READ (4,102) ((LTE(K,J),J=1,2),K=1,4)
      READ (4,104) ((LTOP(J,I),I=1,4),J=1,13)
  104 FORMAT (4I5)
      RETURN
      END
```

```
      BLOCK DATA
      COMMON /OBJECT/ VMAT(4,35),  XFT(21,2), YFT(21,2),
     1 ZFT(21,2)
      COMMON /TRFORM/ ZMAT(4,4), TNLT(4,4)
      COMMON /LISTS/ ILIST(109,2), LLIST(20,6),
     1 LLLSTB(4,13,8), LTE(4,2), LTOP(13,4)
C
C           BODY 1
C
      DATA (VMAT(I,1),I=1,4) / 2*0.0,1.0,-9.5 /
      DATA (VMAT(I,2),I=1,4) / -0.322677,-1.0,0.0,0.867732
     $ /
      DATA (VMAT(I,3),I=1,4) / 2*0.0,-1.0,10.0 /
      DATA (VMAT(I,4),I=1,4) / 0.322677,1.0,0.0,-0.167732 /
      DATA (VMAT(I,5),I=1,4) / -1.0,2*0.0,1.2 /
      DATA (VMAT(I,6),I=1,4) / 1.0,2*0.0,0.1 /
C
C           BODY 2
C
      DATA (VMAT(I,7),I=1,4) / 2*0.0,1.0,-9.45 /
      DATA (VMAT(I,8),I=1,4) / -0.161338,-0.5,1.0,-
     $ 9.091134 /
      DATA (VMAT(I,9),I=1,4) / 2*0.0,-1.0,9.5 /
      DATA (VMAT(I,10),I=1,4) / 0.161338,0.5,1.0,-9.008866
     $ /
      DATA (VMAT(I,11),I=1,4) / -1.0,2*0.0,1.2 /
      DATA (VMAT(I,12),I=1,4) / 1.0,2*0.0,0.1 /
C
C           BODY 3
C
      DATA (VMAT(I,13),I=1,4) / 2*0.0,1.0,-9.35 /
      DATA (VMAT(I,14),I=1,4) / -0.161338,-0.5,1.0,-
     $ 9.016134 /
      DATA (VMAT(I,15),I=1,4) / -0.322677,2*-1.0,10.167732
     $ /
      DATA (VMAT(I,16),I=1,4) / 2*0.0,-1.0,9.45 /
      DATA (VMAT(I,17),I=1,4) / 0.322677,1.0,-1.0,9.132266
     $ /
      DATA (VMAT(I,18),I=1,4) / 0.161338,0.5,1.0,-9.033866
     $ /
      DATA (VMAT(I,19),I=1,4) / -1.0,2*0.0,1.2 /
      DATA (VMAT(I,20),I=1,4) / 1.0,2*0.0,0.1 /
C
C           BODY 4
C
      DATA (VMAT(I,21),I=1,4) / 2*0.0,1.0,-9.25 /
      DATA (VMAT(I,22),I=1,4) / -0.161338,-0.5,1.0,-
     $ 8.991134 /
      DATA (VMAT(I,23),I=1,4) / -0.322677,2*-1.0,10.017732
     $ /
```

```
      DATA (VMAT(I,24),I=1,4) / 2*0.0,-1.0,9.35 /
      DATA (VMAT(I,25),I=1,4) / 0.322677,1.0,-1.0,8.902268
     & /
      DATA (VMAT(I,26),I=1,4) / 0.161338,0.5,1.0,-9.453866
     & /
      DATA (VMAT(I,27),I=1,4) / -1.0,2*0.0,1.2 /
      DATA (VMAT(I,28),I=1,4) / 1.0,2*0.0,0.1 /
C
C           BODY 5
C
      DATA (VMAT(I,29),I=1,4) / 2*0.0,1.0,-9.15 /
      DATA (VMAT(I,30),I=1,4) / -0.161338,-0.5,1.0,-
     & 8.896134 /
      DATA (VMAT(I,31),I=1,4) / -0.322677,2*-1.0,9.857732 /
      DATA (VMAT(I,32),I=1,4) / 2*0.0,-1.0,9.25 /
      DATA (VMAT(I,33),I=1,4) / 0.322677,1.0,-1.0,8.832268
     & /
      DATA (VMAT(I,34),I=1,4) / 0.161338,0.5,1.0,-9.303866
     & /
      DATA (VMAT(I,35),I=1,4) / -1.0,2*0.0,1.2 /
      DATA (VMAT(I,36),I=1,4) / 1.0,2*0.0,0.1 /
C
C           EDGES
C
      DATA (XFT(I,1),I=1,21) / 21*-0.1 /
      DATA (XFT(I,2),I=1,21) / 21*1.2  /
      DATA (YFT(I,1),I=1,21) / 2*0.9,0.85,0.75,0.8,0.7,
     & 0.75,0.65,0.7,
     1 0.6,0.5,0.45,0.35,0.3,0.4,0.3,0.35,0.25,2*0.2,0.9 /
      DATA (YFT(I,2),I=1,21) / 2*0.48052,0.43052,0.33052,
     & 0.38052,
     1 0.28052,0.33052,0.23052,0.20052,0.10052,0.08052,-
     & 0.01948,
     2 0.03052,-0.06948,-0.01948,-0.11948,-0.06948,-
     & 0.15948,
     3 2*-0.21948,0.43052 /
      DATA (ZFT(I,1),I=1,21) / 10.0,2*9.5,9.45,9.4,9.35,
     & 9.3,9.25,9.2,
     1 2*9.15,9.2,9.25,9.3,9.35,9.4,9.45,2*9.5,2*10.0 /
      DATA (ZFT(I,2),I=1,21) / 10.0,2*9.5,9.45,9.4,9.35,
     & 9.3,9.25,9.2,
     1 2*9.15,9.2,9.25,9.3,9.35,9.4,9.45,2*9.5,2*10.0 /
C
C           LOCATIONS OF BODY MATRICES IN VMAT
C
      DATA (MLIST(I,1),I=1,5) / 1,7,13,21,29 /
      DATA (MLIST(I,2),I=1,5) / 2*6,3*8 /
C
C           LINE LIST
C
```

```
      DATA (LLIST(1,I),I=1,6) / 12,16,12,13,12,15 /
      DATA (LLIST(2,I),I=1,6) / 11,16,11,12,11,15 /
      DATA (LLIST(3,I),I=1,6) / 22,26,11,11,22,25 /
      DATA (LLIST(4,I),I=1,6) / 33,38,21,22,33,37 /
      DATA (LLIST(5,I),I=1,6) / 32,38,32,33,32,37 /
      DATA (LLIST(6,I),I=1,6) / 43,48,31,32,43,47 /
      DATA (LLIST(7,I),I=1,6) / 42,48,42,43,42,47 /
      DATA (LLIST(8,I),I=1,6) / 53,58,41,42,53,57 /
      DATA (LLIST(9,I),I=1,6) / 52,58,52,53,52,57 /
      DATA (LLIST(10,I),I=1,6) / 51,58,51,52,51,57 /
      DATA (LLIST(11,I),I=1,6) / 56,58,56,51,55,57 /
      DATA (LLIST(12,I),I=1,6) / 55,58,55,55,55,57 /
      DATA (LLIST(13,I),I=1,6) / 46,48,41,46,46,47 /
      DATA (LLIST(14,I),I=1,6) / 45,48,45,46,45,47 /
      DATA (LLIST(15,I),I=1,6) / 36,38,31,36,36,37 /
      DATA (LLIST(16,I),I=1,6) / 35,38,35,36,35,37 /
      DATA (LLIST(17,I),I=1,6) / 24,25,21,24,24,25 /
      DATA (LLIST(18,I),I=1,6) / 11,16,11,11,11,15 /
      DATA (LLIST(19,I),I=1,6) / 14,15,14,11,14,15 /
      DATA (LLIST(20,I),I=1,6) / 13,16,13,14,13,15 /
C
C     PLANE TRANSLATION MATRIX
C
      DATA (TMLT(1,I),I=1,4) / 1.0,3*0.0 /
      DATA (TMLT(2,I),I=1,4) / 0.0,1.0,2*0.0 /
      DATA (TMLT(3,I),I=1,4) / 2*0.0,1.0,0.0 /
      DATA (TMLT(4,I),I=1,4) / 3*0.0,1.0 /
C
C     PLANE PERSPECTIVE TRANSFORMATION MATRIX
C
      DATA (ZMAT(1,I),I=1,4) / 1.0,3*0.0 /
      DATA (ZMAT(2,I),I=1,4) / 0.0,1.0,2*0.0 /
      DATA (ZMAT(3,I),I=1,4) / 3*0.0,-1.0 /
      DATA (ZMAT(4,I),I=1,4) / 2*0.0,1.0,0.0 /
      END
```

```
      SUBROUTINE DOLIST
      COMMON /EDGE/ NB, VB(4), VE(4)
      COMMON /WHICH/ IN(4,109), YW(5)
      COMMON /HLIST/ JLIST(109), JMAX
      COMMON // RVMAT(4,548), BLMAT(4,512), RXB(5,27),
     1 RYB(5,27), RZB(5,27), RX(21,2), RY(21,2),
     2 RZ(21,2), FQ(2,8), M(8), SB(2,4), J1, J2,
     3 T1N, T2N, MANY, NT
      DIMENSION JDO(4)
C
C         IS THE LINE IN TWO OR MORE WINDOWS?
C
      JJ          = 0
      DO 1 I      = 1,4
      IF (IN(I,NB) .LT. 1) GO TO 1
      JJ          = JJ + 1
      JDO(JJ)     = I
    1 CONTINUE
      IF (JJ .GT. 0) GO TO 2
      PRINT *, " NO ENTRIES FOUND FOR BODY ", NB
      STOP
    2 JMAX        = 0
C
C         ASSEMBLE LIST OF BODIES IN SAME WINDOWS AS LINE
C
      CALL MEMSET (JLIST(1),JLIST(109),0)
      DO 5 I      = 1,MANY
      DO 3 J      = 1,JJ
      IF (IN(JDO(J),I) .GT. 0) GO TO 4
    3 CONTINUE
      GO TO 5
    4 JMAX        = JMAX + 1
      JLIST(JMAX)= IN(JDO(J),I)
    5 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FIND4 (L1,L2,NB,MV,NV)
      COMMON /LISTS/ MLIST(109,2), LLIST(20,5), LLISTB(4,
     $ 13,8),
     1 LTE(4,2), LTOP(13,4)
      NB          = L1/10
      MB          = L2/10
      NV          = MLIST(NB,1) + L1 - NB*10 - 1
      MV          = MLIST(MB,1) + L2 - MB*10 - 1
      RETURN
      END
```

```
      SUBROUTINE HIDE
      COMMON /EDGE/ NB,VB(4),VE(4)
      COMMON /FRAME/ XF, YF, AF, BF, CF, DF
      COMMON /LISTS/ MLIST(109,2), LLIST(20,5), LLIST8(4,
     * 13,8),
     1 LTE(4,2), LTOP(13,4)
      COMMON /HLIST/ JLIST(109), JMAX
      COMMON /BODY/ NEWSEG
      COMMON // RVMAT(4,546), BLMAT(4,512), RXB(5,27),
     1 RYB(5,27), RZB(5,27), RX(21,2), RY(21,2),
     2 RZ(21,2), PQ(2,8), M(8), SD(2,4), J1, J2,
     3 T1N, T2N, MANY, NT
      LOGICAL NEWSEG
      XSCALE(X)  = DF + CF*X
      YSCALE(Y)  = BF + AF*Y
      POINT(J,T) = VE(J) + T*SD(1,J)
C
C           LOAD SD
C           ROW 1 IS THE DIFFERENCE, VB - VE.
C           ROW 2 IS VE.
C
      DO 1 J     = 1,4
      SD(1,J)    = VB(J) - VE(J)
      SD(2,J)    = VE(J)
    1 CONTINUE
C
C           TEST LINE AGAINST EACH VOLUME
C           EXCLUDING ITS OWN.
C
      T1         = 2.0
      T2         = -1.0
C
C          FIND WINDOW FOR THIS LINE
C
      IF (NEWSEG) CALL DOLIST
      DO 2 JTL   = 1,JMAX
      J          = JLIST(JTL)
C
C           NB DENOTES THE BODY TO WHICH
C           THE LINE BELONGS.
C
      IF (J .EQ. NB) GO TO 2
      J1         = MLIST(J,1)
      J2         = MLIST(J,2)
      CALL SEEK
      T1         = AMIN1(T1,T1N)
      T2         = AMAX1(T2,T2N)
C
C           WHIPPING A DEAD HORSE
C           LINE IS COMPLETELY HIDDEN.
```

```
C
      IF (T1 .LE. 0.0 .AND. T2 .GE. 1.0) RETURN
    2 CONTINUE
C
C           DISPLAY
C
C           IF T1 > 1.0 AND T2 < 0.0, THE LINE IS
C           COMPLETELY VISIBLE.
C
      IF (T1 .GT. 1.0 .AND. T2 .LT. 0.0) GO TO 5
      IF (T1 .LE. 0.0) GO TO 3
      T1          = AMIN1(T1,1.0)
C
C           LINE IS VISIBLE FOR T BETWEEN
C           ZERO AND T1.
C
      CALL PLOT (XSCALE(VE(1)),YSCALE(VE(2)),3)
      CALL PLOT (XSCALE(POINT(1,T1)),YSCALE(POINT(2,T1)),2)
    3 IF (T2 .GE. 1.0) RETURN
      T2          = AMAX1(T2,0.0)
C
C           LINE IS VISIBLE FOR T BETWEEN
C           T2 AND 1.
C
    4 CALL PLOT (XSCALE(VB(1)),YSCALE(VB(2)),3)
      CALL PLOT (XSCALE(POINT(1,T2)),YSCALE(POINT(2,T2)),2)
      RETURN
    5 T2          = 0.0
      GO TO 4
      END
```

```
      FUNCTION INTHERE (Y)
      COMMON /WHICH/ IN(4,109), Yw(5)
      INTHERE     = 0
      IF (Y .GT. YW(2)) GO TO 1
      INTHERE     = 1
      RETURN
    1 IF (Y .GT. YW(3)) GO TO 2
      INTHERE     = 2
      RETURN
    2 IF (Y .GT. YW(4)) GO TO 3
      INTHERE     = 3
      RETURN
    3 INTHERE     = 4
      RETURN
      END
```

```
      SUBROUTINE MMIF (A,B,N)
C
C          MULTIPLIES MATRIX A TIMES MATRIX B AND
C          STORES THE RESULT IN B.  A IS DIMENSIONED 4X4,
C          B IS 4XN.
C
      DIMENSION A(4,4), B(4,N)
      DO 2 J = 1,N
      BA     = 0.0
      BB     = 0.0
      BC     = 0.0
      BD     = 0.0
      DO 1 K = 1,4
      BA     = BA + A(1,K)*B(K,J)
      BB     = BB + A(2,K)*B(K,J)
      BC     = BC + A(3,K)*B(K,J)
      BD     = BD + A(4,K)*B(K,J)
    1 CONTINUE
      B(1,J) = BA
      B(2,J) = BB
      B(3,J) = BC
      B(4,J) = BD
    2 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MMPY (A,B,C,L,M,N)
      DIMENSION A(L,M), B(M,N), C(L,N)
      DO 1 I = 1,L
      DO 1 J = 1,N
      C(I,J) = 0.0
      DO 1 K = 1,M
      C(I,J) = C(I,J) + A(I,K)*B(K,J)
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ROTPLN
      COMMON /TRFORM/ ZMAT(4,4), TNLT(4,4)
      COMMON /VIEW/ XV,YV,ZV
      COMMON /OBJECT/ VMAT(4,35), XFT(21,2), YFT(21,2),
     1 ZFT(21,2)
      COMMON /BLADE/ XB(5,27), YB(5,27), ZB(5,27)
      COMMON /MATRIX/ RMAT(4,4)
      COMMON /HOW/ PRSP, XMIN, XMAX, YMIN, YMAX
      COMMON // RVMAT (4,540), BLMAT(4,512), RXB(5,27),
     1 RYB(5,27), RZB(5,27), RX(21,2), RY(21,2),
     2 RZ(21,2), PQ(2,8), M(8), SD(2,4), J1, J2,
     3 I1N, I2N, MANY, NT
      DIMENSION WAIT(4,4), PMAT(4,4)
      LOGICAL PRSP
C
C           TRANSFORM AND SCALE COORDINATES.
C           FIRST THE FIRTREE, THEN THE BLADE.
C
      XMIN        = YMIN = 1.0E5
      XMAX        = YMAX = -1.0E6
      CALL RAM (XFT,YFT,ZFT,RX,RY,RZ,21,2,21)
      CALL RAM (XB,YB,ZB,RXB,RYB,RZB,5,27,NT+1)
C
C           TRANSLATE AND ROTATE PLANES.
C
      TNLT(4,1)   = XV
      TNLT(4,2)   = YV
      TNLT(4,3)   = ZV
      CALL MMPY (RMAT,TNLT,WAIT,4,4,4)
      CALL MMPY (ZMAT,WAIT,PMAT,4,4,4)
      DO 1 J      = 1,36
      DO 1 I      = 1,4
      RVMAT(I,J) = VMAT(I,J)
    1 CONTINUE
      DO 2 J      = 37,548
      DO 2 I      = 1,4
      RVMAT(I,J) = BLMAT(I,J-36)
    2 CONTINUE
      CALL MMIP (PMAT,RVMAT,548)
      PRINT 100, ((PMAT(I,J),J=1,4),I=1,4)
  100 FORMAT (*0 PLANE ROTATION MATRIX*/
     1 (1H ,4E15.7))
      RETURN
      END
```

```
      SUBROUTINE SEEK
      COMMON /0/ T, ALPHA, TMIN, TMAX, KT, JTA, JS, L1, L2
      COMMON /BODY/ NEWSEG
      COMMON // RVMAT(4,540), BLMAT(4,512), RXB(5,27),
     1 RYB(5,27), RZB(5,27), RX(21,2), RY(21,2),
     2 RZ(21,2), PQ(2,3), M(6), SD(2,4), J1, J2,
     3 T1N, T2N, MANY, NT
      LOGICAL NEWSEG
C
C           MULTIPLY SD*VMAT, OBTAIN PQ
C
      JS          = J1 - 1
      N           = J2
C
C           N IS THE NUMBER OF PLANES ON
C           THE BODY IN QUESTION.
C
      CALL MMPY (SD,RVMAT(1,J1),PQ,2,4,N)
C
C           ROW 1 OF PQ IS Q, ROW 2 IS P
C
      JTA         = 0
C
C           SORT THROUGH PQ, RETAIN RELEVANT PAIRS
C
      DO 4 J      = 1,N
      IF (RVMAT(3,J+JS) .GT. 0.0) GO TO 2
C
C           IF THESE ARE TRUE, THE INEQUALITY
C           IS ALWAYS SATISFIED.
C
C           CHECK AGAINST INVISIBLE PLANES
C
      IF (PQ(2,J) .GE. 0.0 .AND. PQ(1,J) .GE. 0.0) GO TO 4
      IF (PQ(2,J) .GT. 0.0 .AND. PQ(1,J) .LT. 0.0 .AND.
     1 (PQ(2,J) +
     1 PQ(1,J)) .GE. 0.0) GO TO 4
      IF (RVMAT(3,J+JS) .LT. 0.0) GO TO 3
C
C           IF THESE ARE TRUE, THE INEQUALITY
C           IS NEVER SATISFIED.
C
C           CHECK AGAINST VISIBLE PLANES
C
C           IF THE TESTS IN STATEMENT 2 ARE TRUE,
C           THE LINE IS ON THE SURFACE OF A VISIBLE PLANE
C           OR BETWEEN THE VIEWER AND A VISIBLE PLANE.
C
    2 IF (PQ(2,J) .LE. 1.0E-4 .AND. (PQ(2,J) +
     1 PQ(1,J)) .LE. 1.0E-4) GO TO 10
```

```
    3 JTA          = JTA + 1
      M(JTA)       = J + JS
    4 CONTINUE
C
C
C           FIND INTERSECTIONS OF T - ALPHA LINES
C
C           IF TWO OR LESS T - ALPHA LINES EXIST,
C           NO SOLUTION IS POSSIBLE.
C
      IF (JTA .LT. 2) GO TO 10
      TMIN         = 2.0
      TMAX         = -2.0
      KT           = 0
      JMAX         = JTA - 1
C
C           INTERSECTIONS WITH T = 0 AND T = 1
C
      L2           = 0
      DO 6 J       = 1,JTA
      T            = 0.0
      MJ           = M(J)
      L1           = MJ
      ALPHA        = PQ(2,MJ-JS)/RVMAT(3,MJ)
      IF (ALPHA .LT. -1.0E-6) GO TO 5
      CALL VALID
    5 ALPHA        = ALPHA + PQ(1,MJ-JS)/RVMAT(3,MJ)
      T            = 1.0
      IF (ALPHA .LT. -1.0E-6) GO TO 6
      CALL VALID
    6 CONTINUE
C
C           INTERSECTIONS WITH ALPHA = 0
C
      ALPHA        = 0.0
      DO 7 J       = 1,JTA
      L1           = M(J)
      MJ           = M(J) - JS
      IF (ABS(PQ(1,MJ)) .LT. 1.0E-6) GO TO 7
      T            = -PQ(2,MJ)/PQ(1,MJ)
      CALL VALID
    7 CONTINUE
C
C           INTERSECTIONS WITH EACH OTHER
C
      DO 9 J       = 1,JMAX
      K1           = J + 1
      MJ           = M(J)
      L1           = MJ
      DO 8 K       = K1,JTA
      MK           = M(K)
```

```
      L2              = MK
      WW              = RVMAT(3,MJ)/RVMAT(3,MK)
      DENOM           = PQ(1,MJ-JS) - WW*PQ(1,MK-JS)
      IF (ABS(DENOM) .LT. 1.0E-6) GO TO 8
      T               = (WW*PQ(2,MK-JS) - PQ(2,MJ-JS))/DENOM
      ALPHA           = (T*PQ(1,MK-JS) + PQ(2,MK-JS))/RVMAT(3,
     $ MK)
      IF (ALPHA .LT. -1.0E-6) GO TO 8
C
C           THIS T AND ALPHA MUST ALSO SATISFY ALL THE
C           OTHER INEQUALITIES.
C
      CALL VALID
    8 CONTINUE
    9 CONTINUE
      T1N             = TMIN
      T2N             = TMAX
      IF (ABS(T1N-T2N) .LE. 1.0E-3 .OR.
     1 (T2N .LT. 0.0 .AND. T1N .LT. T2N) .OR.
     2 (T1N .GT. 1.0 .AND. T2N .GT. T1N)) GO TO 10
      RETURN
C
C           LINE IS COMPLETELY VISIBLE.
C
   10 T1N             = 2.
      T2N             = -2.0
      RETURN
      END
```

```
      SUBROUTINE TOPPER (K,IB,IE,IW)
      COMMON /LISTS/  LIST(100,2), LLIST(20,6), LLIST3(4,
     & 13,3),
     1 LIE(4,2), LTOP(13,4)
      COMMON /EDGE/ NB, VB(4), VE(4)
      COMMON /BODY/ NEWSEG
      COMMON // RVMAT(4,548), BLMAT(4,512), RXB(5,27),
     1 RYB(5,27), RZB(5,27), RX(21,2), RY(21,2),
     2 RZ(21,2), PQ(2,6), M(6), SS(2,4), J1, J2,
     3 T1N, T2N, MANY, NT
      LOGICAL NEWSEG
      MB          = 0
      DO 6 I      = IB,IE
      DO 5 J      = 1,3,2
      IF (IW .NE. 1) GO TO 1
      CALL FINDM (LLIST3(K,I,J),LLIST3(K,I,J+1),NB,MV,NV)
      GO TO 2
    1 CALL FINDM (LTOP(I,J),LTOP(I,J+1),NB,MV,NV)
      IF (RVMAT(3,MV) .LT. 0.0 .AND. RVMAT(3,NV) .LT. 0.0)
     & GO TO 5
    2 IF (J .GT. 1) GO TO 3
C
C         PRESSURE SIDE
C
      CALL VSET (K,I,K,I+1)
      GO TO 4
C
C         SUCTION SIDE
C
    3 CALL VSET (K,28-I,K,27-I)
    4 NEWSEG      = MB .NE. NB
      CALL HIDE
      MB          = NB
    5 CONTINUE
    6 CONTINUE
C
C         LINE FROM SUCTION SIDE TO PRESSURE SIDE IF THIS
C         CAP DOESN'T END AT THE TRAILING EDGE.
C
      IF (IE .GE. 13) RETURN
      CALL VSET (K,28-I,K,1)
      CALL HIDE
      RETURN
      END
```

```
      SUBROUTINE VALI
      COMMON /01/ T, ALPHA, TMIN, TMAX, KT, JTA, JS, L1, L2
      COMMON /NVQAL/ NVQ
      COMMON // RVMAT(4,548), BLMAT(4,512), RXB(5,27),
     1 RYB(5,27), RZB(5,27), RX(21,2), RY(21,2),
     2 RZ(21,2), PQ(2,6), M(5), SD(2,4), J1, J2,
     3 T1N, T2N, MANY, NT
      NVQ          = NVQ + 1
      DO 1 L       = 1,JTA
      IF (M(L) .EQ. L1 .OR. M(L) .EQ. L2) GO TO 1
      IF (PQ(2,M(L)-JS) + T*PQ(1,M(L)-JS) - ALPHA*RVMAT(3,
     2 M(L))
     1 .LT. -1.0E-6) RETURN
    1 CONTINUE
      KT           = KT + 1
      IF (T .GT. TMIN) GO TO 2
      TMIN         = T
      IF (KT - 1) 4,3,4
    2 IF (T .LE. TMAX) RETURN
    3 TMAX         = T
    4 RETURN
      END
```

```
      SUBROUTINE VSET (I,J,K,L)
      COMMON /EDGE/ N3, VB(4), VE(4)
      COMMON // RVMAT(4,548), BLMAT(4,512), RXB(5,27),
     1 RYB(5,27), RZB(5,27), RX(21,2), RY(21,2),
     2 RZ(21,2), FQ(2,6), M(8), SD(2,4), J1, J2,
     3 T1N, T2N, MANY, NT
      VB(1)       = RXB(I,J)
      VB(2)       = RYB(I,J)
      VB(3)       = RZB(I,J)
      VE(1)       = RXB(K,L)
      VE(2)       = RYB(K,L)
      VE(3)       = RZB(K,L)
      RETURN
      END
```

```
      SUBROUTINE WINDOW
      COMMON /FRAME/ XF, YF, AF, BF, CF, DF
      COMMON /WHICH/ IN(4,109), YW(5)
      COMMON // RVMAT(4,548), BLMAT(4,512), RX3(5,27),
     1 RY3(5,27), RZ3(5,27), RX(21,2), RY(21,2),
     2 RZ(21,2), FQ(2,8), M(8), SD(2,4), J1, J2,
     3 T1N, T2N, MANY, NT
      YSCALE(Y)   = BF + AF*Y
      KFIR(K,I)   = (K - 1)*(21 - I) + (2 - K)*I
      CALL MEMSET (IN(1),IN(435),0)
      YW(1)       = 0.
      YW(2)       = YF/4.0
      DO 1 I      = 3,5
      YW(I)       = YW(I-1) + YW(2)
    1 CONTINUE
C
C          FIR TREE
C
      DO 2 L      = 1,2
      I2          = 2**L
      I1          = I2 - 1
      DO 2 I      = I1,I2
      DO 2 J      = 1,2
      DO 2 K      = 1,2
C
C          INTHERE RETURNS A VALUE OF 1, 2, 3, OR 4
C
      IN(INTHERE(YSCALE(RY(KFIR(K,I),J))),L) = L
    2 CONTINUE
      DO 3 L      = 2,5
      I2          = 2**L
      I1          = I2 - 2
      DO 3 I      = I1,I2
      DO 3 J      = 1,2
      DO 3 K      = 1,2
      IN(INTHERE(YSCALE(RY(KFIR(K,I),J))),L) = L
    3 CONTINUE
      IF (MANY .LE. 5) GO TO 5
C
C          BLADE
C
      MAXL        = MIN0(MANY + 1,109)
      NSEC        = MANY - 5 - 26*(NT - 1)
      IF (NSEC .EQ. 25) MAXL = MANY - 1
      DO 4 L      = 6,MAXL,2
      K           = 1 + (L - 6)/26
      I           = 1 + (L - 5 - 26*(K - 1))/2
      DO 4 NK     = 1,2
      DO 4 NJ     = 1,2
      DO 4 NI     = 1,2
```

```
      KL          = IN HERE(YSCALE(RYB(K + 2 - NK,(2 - NJ)*
     2 (I + 2 - NI
     1 ) + (NJ - 1)*(_8 - (I + 2 - NI)) )))
      IN(KL,L)    = L
      IN(KL,L+1)  = L+1
      IF (K .EQ. 4 .O.. NK .EQ. 2) GO TO 4
      IN(KL,L+26) = L + 26
      IN(KL,L+27) = L + 27
    4 CONTINUE
    5 DO 7 L      = 1, MAXL
C
C     CHECK FOR CASES WHERE ONE BODY IS IN THREE OR
C          MORE
C     WINDOWS.
C
      IF (IN(1,L) .EQ. L .AND. IN(3,L) .EQ. L) IN(2,L) = L
      IF (IN(1,L) .NE. L .OR.  IN(4,L) .NE. L) GO TO 6
      IN(2,L)     = IN(3,L) = L
    6 IF (IN(2,L) .EQ. L .AND. IN(4,L) .EQ. L) IN(3,L) = L
    7 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE WINLIST (M)
      COMMON /WHICH/ IN(4,109), YW(5)
      PRINT 100
  100 FORMAT (*0  WINDOW LIST*)
      N             = M/30   &  L1 = 1   &  L2 = 30
      IF (N .LT. 1) GO TO 3
      DO 2 I        = 1,N
      DO 1 J        = 1,4
      PRINT 101, (IN(J,L),L=L1,L2)
  101 FORMAT (1H 30I4)
    1 CONTINUE
      PRINT 101
      L1            = L2 + 1   &  L2 = L2 + 30
    2 CONTINUE
    3 L2            = M
      DO 4 J        = 1,4
      PRINT 101, (IN(J,L),L=L1,L2)
    4 CONTINUE
      PRINT 101
      RETURN
      END
```

```
 10.00000
   .00573     .33619     .33619
   .01087     .36977     .32202
   .02174     .38457     .33102
   .03261     .39871     .33959
   .04348     .41223     .34775
   .05435     .42516     .35551
   .10870     .48153     .38908
   .16305     .52572     .41504
   .21740     .55929     .43447
   .27175     .58334     .44804
   .32610     .59862     .45617
   .38045     .60570     .45909
   .43480     .60495     .45687
   .48915     .59562     .44946
   .54350     .58083     .43607
   .59785     .55759     .41815
   .65220     .52680     .39339
   .70655     .48826     .35179
   .76090     .44160     .32285
   .81525     .38632     .27660
   .86960     .32212     .22370
   .92395     .24993     .16505
   .97830     .17157     .10143
  1.03265     .08881     .03349
  1.06526     .03762    -.00914
  1.07613     .02035    -.02365
  1.08153    -.01095    -.01095
0.             0.              .1000000E+01 -.1000000E+02
 .1747152E+02  .5410425E+02 -.1000000E+01 -.8289419E+01
 .1671367E+02 -.5530920E+01  .1000000E+01 -.8236329E+01
-.1646586E+02  0.            -.1000000E+01  .1035797E+02
 .1646586E+02-0.              .1000000E+01 -.1035797E+02
 .6250230E+02  .1258940E+03 -.1000000E+01 -.3303224E+02
 .1668181E+02 -.4819739E+01  .1000000E+01 -.8509135E+01
0.             0.            -.1000000E+01  .1131200E+02
-.2046483E+02  0.            -.1000000E+01  .1044491E+02
0.             0.              .1000000E+01 -.1000000E+02
-.1102407E+02  .1432536E+02 -.1000000E+01  .5497681E+01
 .1265675E+02 -.9947854E+01  .1000000E+01 -.6449511E+01
-.1528247E+02  0.            -.1000000E+01  .1066448E+02
 .2046483E+02-0.              .1000000E+01 -.1044491E+02
 .1528247E+02-0.              .1000000E+01 -.1066448E+02
-.1132597E+02  .1330978E+02 -.1000000E+01  .5863977E+01
 .1275309E+02 -.9582832E+01  .1000000E+01 -.6604173E+01
0.             0.            -.1000000E+01  .1131200E+02
-.2074636E+02  0.            -.1000000E+01  .1090205E+02
0.             0.              .1000000E+01 -.1000000E+02
-.9800359E+01  .1546527E+02 -.1000000E+01  .5048074E+01
 .1162251E+02 -.1093824E+02  .1000000E+01 -.5996275E+01
```

```
-.1021330E+02  0.             -.1000000E+01   .1111019E+02
 .2074636E+02 -0.              .1000000E+01  -.1090205E+02
 .1021330E+02 -0.              .1000000E+01  -.1111019E+02
-.9828770E+01   .1440123E+02  -.1000000E+01   .5465157E+01
 .1157669E+02  -.1058257E+02   .1000000E+01  -.6162559E+01
 0.             0.            -.1000000E+01   .1131200E+02
-.2163232E+02   0.            -.1000000E+01   .1235143E+02
 0.             0.             .1000000E+01  -.1000000E+02
-.7301239E+01   .1748501E+02  -.1000000E+01   .3990577E+01
 .9291097E+01  -.1298794E+02   .1000000E+01  -.4755859E+01
-.7747269E+01   0.            -.1000000E+01   .1168426E+02
 .2163232E+02  -0.             .1000000E+01  -.1235143E+02
 .7747269E+01  -0.             .1000000E+01  -.1168426E+02
-.7310243E+01   .1713201E+02  -.1000000E+01   .4145903E+01
 .9302574E+01  -.1308449E+02   .1000000E+01  -.4704353E+01
 0.             0.            -.1000000E+01   .1131200E+02
-.2329545E+02   0.            -.1000000E+01   .1506443E+02
 0.             0.             .1000000E+01  -.1000000E+02
-.4105718E+01   .2056643E+02  -.1000000E+01   .1957086E+01
 .5889131E+01  -.1627634E+02   .1000000E+01  -.2177101E+01
-.7950551E+01   0.            -.1000000E+01   .1259267E+02
 .2329545E+02  -0.             .1000000E+01  -.1506443E+02
 .7950551E+01  -0.             .1000000E+01  -.1259267E+02
-.3832004E+01   .2176310E+02  -.1000000E+01   .1338240E+01
 .5739630E+01  -.1745676E+02   .1000000E+01  -.1421731E+01
 0.             0.            -.1000000E+01   .1131200E+02
-.2523077E+02   0.            -.1000000E+01   .1822775E+02
 0.             0.             .1000000E+01  -.1000000E+02
-.1645528E+00   .2555270E+02  -.1000000E+01  -.1602715E+01
 .1250694E+01  -.2147716E+02   .1000000E+01   .2448806E+01
-.8164281E+01   0.            -.1000000E+01   .1354983E+02
 .2523077E+02  -0.             .1000000E+01  -.1822775E+02
 .8164281E+01  -0.             .1000000E+01  -.1354983E+02
 .1229046E+01   .3000413E+02  -.1000000E+01  -.4242376E+01
 .1128555E+00  -.2501187E+02   .1000000E+01   .5081862E+01
 0.             0.            -.1000000E+01   .1131200E+02
-.2752255E+02   0.            -.1000000E+01   .2196681E+02
 0.             0.             .1000000E+01  -.1000000E+02
 .6621431E+01   .3563149E+02  -.1000000E+01  -.9157984E+01
-.6987773E+01  -.3149133E+02   .1000000E+01   .1208896E+02
-.8390356E+01   0.            -.1000000E+01   .1456016E+02
 .2752255E+02  -0.             .1000000E+01  -.2196681E+02
 .8390356E+01  -0.             .1000000E+01  -.1456016E+02
 .1435308E+02   .5398286E+02  -.1000000E+01  -.2137360E+02
-.1268620E+02  -.4316056E+02   .1000000E+01   .2196390E+02
 0.             0.            -.1000000E+01   .1131200E+02
-.3026528E+02   0.            -.1000000E+01   .2644918E+02
 0.             0.             .1000000E+01  -.1000000E+02
 .2813100E+02   .7065250E+02  -.1000000E+01  -.3614103E+02
-.3020188E+02  -.6076151E+02   .1000000E+01   .4170633E+02
```

```
-.8628741E+01  0.             -.1000000E+01  .1562706E+02
 .3026528E+02-0.               .1000000E+01 -.2644918E+02
 .8628741E+01-0.               .1000000E+01 -.1562766E+02
 .5255341E+03  .9934959E+03   .1000000E+01 -.7435847E+03
-.1000780E+03 -.1701025E+03   .1000000E+01  .1448809E+03
 0.            0.             -.1000000E+01  .1131200E+02
-.3362378E+02  0.             -.1000000E+01  .3192943E+02
 0.            0.              .1000000E+01 -.1000000E+02
 .1418614E+03  .2186041E+03   .1000000E+01 -.1885186E+03
-.1198739E+05 -.1529377E+05   .1000000E+01  .1586493E+05
-.8861668E+01  0.             -.1000000E+01  .1675806E+02
 .3362378E+02-0.               .1000000E+01 -.3192943E+02
 .8861668E+01-0.               .1000000E+01 -.1675806E+02
 .3232368E+02  .3353610E+02   .1000000E+01 -.4703647E+02
-.6955018E+02 -.7734489E+02  -.1000000E+01  .9707624E+02
 0.            0.             -.1000000E+01  .1131200E+02
-.3780980E+02  0.             -.1000000E+01  .3876948E+02
 0.            0.              .1000000E+01 -.1000000E+02
 .3270531E+02  .3585544E+02   .1000000E+01 -.4646140E+02
-.6033626E+02 -.5489248E+02  -.1000000E+01  .8015040E+02
-.9149233E+01  0.             -.1000000E+01  .1795617E+02
 .3780980E+02-0.               .1000000E+01 -.3876948E+02
 .9149233E+01-0.               .1000000E+01 -.1795617E+02
 .2235116E+02  .2009506E+02   .1000000E+01 -.3393183E+02
-.3608588E+02 -.2888660E+02  -.1000000E+01  .5068523E+02
 0.            0.             -.1000000E+01  .1131200E+02
-.4320053E+02  0.             -.1000000E+01  .4756718E+02
 0.            0.              .1000000E+01 -.1000000E+02
 .2247233E+02  .1997827E+02   .1000000E+01 -.3401108E+02
-.3667806E+02 -.2648227E+02  -.1000000E+01  .5042571E+02
-.9434098E+01  0.             -.1000000E+01  .1922938E+02
 .4320053E+02-0.               .1000000E+01 -.4756718E+02
 .9434098E+01-0.               .1000000E+01 -.1922938E+02
 .1878057E+02  .1432143E+02   .1000000E+01 -.2982565E+02
-.2638283E+02 -.1647437E+02  -.1000000E+01  .3863692E+02
 0.            0.             -.1000000E+01  .1131200E+02
-.5036468E+02  0.             -.1000000E+01  .5927177E+02
 0.            0.              .1000000E+01 -.1000000E+02
 .1841651E+02  .1448560E+02   .1000000E+01 -.2943810E+02
-.2589428E+02 -.1681050E+02  -.1000000E+01  .3821655E+02
-.1160959E+02  0.             -.1000000E+01  .2236723E+02
 .5036468E+02-0.               .1000000E+01 -.5927177E+02
 .1160959E+02-0.               .1000000E+01 -.2236723E+02
 .1717696E+02  .1184428E+02   .1000000E+01 -.2818967E+02
-.2155114E+02 -.1169941E+02  -.1000000E+01  .3339770E+02
 0.            0.             -.1000000E+01  .1131200E+02
-.5807890E+02  0.             -.1000000E+01  .7186902E+02
 0.            0.              .1000000E+01 -.1000000E+02
 .1812510E+01  .1629346E+02   .1000000E+01 -.1178138E+02
-.2838833E+02 -.9509536E+01  -.1000000E+01  .4059870E+02
```

```
-.3376222E+02  0.              -.1000000E+01   .4651486E+02
 .5807880E+02-0.               .1000000E+01  -.7186902E+02
 .3376222E+02-0.               .1000000E+01  -.4651486E+02
 .3271563E+01   .1554944E+02   .1000000E+01  -.1336803E+02
-.2910072E+02  -.8248904E+01  -.1000000E+01   .4138298E+02
0.              0.              -.1000000E+01   .1131200E+02
 11.31200
  -.05794     .38100     .38100
  -.05367     .41870     .36277
  -.04237     .43489     .37327
  -.03107     .45031     .38316
  -.01976     .46498     .39251
  -.00846     .47895     .40134
   .04805     .53916     .43879
   .10456     .58536     .46683
   .16108     .61952     .48702
   .21759     .64298     .50030
   .27410     .65668     .50718
   .33061     .66127     .50792
   .38713     .65719     .50255
   .44364     .64472     .49088
   .50015     .62397     .47250
   .55666     .59494     .44673
   .61318     .55747     .41271
   .66969     .51126     .36972
   .72620     .45584     .31791
   .78271     .39049     .25832
   .83923     .31464     .19219
   .89574     .22662     .12049
   .95225     .13365     .04398
  1.00876     .03133    -.03680
  1.04267    -.03291    -.08715
  1.05397    -.05473    -.10423
  1.05902    -.09059    -.09059
0.              0.               .1000000E+01  -.1131200E+02
 .1797208E+02   .3619991E+02  -.1000000E+01  -.1438863E+01
 .1541228E+02  -.4452944E+01   .1000000E+01  -.8722441E+01
-.1622791E+02  0.              -.1000000E+01   .1062442E+02
 .1622791E+02-0.               .1000000E+01  -.1062442E+02
 .2224626E+02   .4072403E+02  -.1000000E+01  -.2946486E+01
 .1535883E+02  -.4744751E+01   .1000000E+01  -.8597802E+01
0.              0.              -.1000000E+01   .1262500E+02
-.2046446E+02  0.              -.1000000E+01   .1044492E+02
0.              0.               .1000000E+01  -.1131200E+02
-.9166674E+01   .1077227E+02  -.1000000E+01   .6902642E+01
 .1131531E+02  -.8502402E+01   .1000000E+01  -.7134935E+01
-.1513196E+02  0.              -.1000000E+01   .1101299E+02
 .2046446E+02-0.               .1000000E+01  -.1044492E+02
 .1513196E+02-0.               .1000000E+01  -.1101299E+02
-.8309586E+01   .1232002E+02  -.1000000E+01   .6312070E+01
```

```
 .1109551E+02 -.8992112E+01  .1000000E+01 -.6911600E+01
0.            0.            -.1000000E+01  .1262500E+02
-.2074250E+02 0.            -.1000000E+01  .1090213E+02
0.            0.             .1000000E+01 -.1131200E+02
-.8368509E+01  .1226164E+02 -.1000000E+01  .6333823E+01
 .1047230E+02 -.9573018E+01  .1000000E+01 -.6653806E+01
-.1001449E+02 0.            -.1000000E+01  .1179320E+02
 .2074250E+02-0.             .1000000E+01 -.1090213E+02
 .1001449E+02-0.             .1000000E+01 -.1179320E+02
-.8015531E+01  .1489112E+02 -.1000000E+01  .5163071E+01
 .1050374E+02 -.1023058E+02  .1000000E+01 -.6300786E+01
0.            0.            -.1000000E+01  .1262500E+02
-.2163097E+02 0.            -.1000000E+01  .1235137E+02
0.            0.             .1000000E+01 -.1131200E+02
-.6882717E+01  .1613007E+02 -.1000000E+01  .4564999E+01
 .8550396E+01 -.1202652E+02  .1000000E+01 -.5238627E+01
-.7557704E+01 0.            -.1000000E+01  .1252940E+02
 .2163097E+02-0.             .1000000E+01 -.1235137E+02
 .7557704E+01-0.             .1000000E+01 -.1252940E+02
-.6674694E+01  .2110117E+02 -.1000000E+01  .2110470E+01
 .8625102E+01 -.1293159E+02  .1000000E+01 -.4689951E+01
0.            0.            -.1000000E+01  .1262500E+02
-.2328840E+02 0.            -.1000000E+01  .1506330E+02
0.            0.             .1000000E+01 -.1131200E+02
-.4301575E+01  .2411528E+02 -.1000000E+01  .2602749E+00
 .5234642E+01 -.1592147E+02  .1000000E+01 -.2291559E+01
-.7750885E+01 0.            -.1000000E+01  .1343652E+02
 .2328840E+02-0.             .1000000E+01 -.1506330E+02
 .7750885E+01-0.             .1000000E+01 -.1343652E+02
-.2871875E+01  .3411079E+02 -.1000000E+01 -.5201127E+01
 .4968764E+01 -.1760521E+02  .1000000E+01 -.1112951E+01
0.            0.            -.1000000E+01  .1262500E+02
-.2522574E+02 0.            -.1000000E+01  .1822638E+02
0.            0.             .1000000E+01 -.1131200E+02
 .1681907E+01  .4105961E+02 -.1000000E+01 -.9973624E+01
 .9853423E-01 -.2183789E+02  .1000000E+01  .3001498E+01
-.7953719E+01 0.            -.1000000E+01  .1439112E+02
 .2522574E+02-0.             .1000000E+01 -.1822638E+02
 .7953719E+01-0.             .1000000E+01 -.1439112E+02
 .1073708E+02  .7964575E+02 -.1000000E+01 -.3287062E+02
-.1567677E+01 -.2647006E+02  .1000000E+01  .6690753E+01
0.            0.            -.1000000E+01  .1262500E+02
-.2750890E+02 0.            -.1000000E+01  .2196152E+02
0.            0.             .1000000E+01 -.1131200E+02
 .3412729E+02  .1283550E+03 -.1000000E+01 -.6640448E+02
-.1005009E+02 -.3419208E+02  .1000000E+01  .1504938E+02
-.8167963E+01 0.            -.1000000E+01  .1539721E+02
 .2750890E+02-0.             .1000000E+01 -.2196152E+02
 .8167963E+01-0.             .1000000E+01 -.1539721E+02
 .6936441E+02  .1875359E+03  .1000000E+01 -.1349154E+03
```

```
-.2025870E+02 -.5335185E+02  .1000000E+01  .3211034E+02
0.            0.            -.1000000E+01  .1262500E+02
-.3025346E+02 0.            -.1000000E+01  .2644327E+02
0.            0.             .1000000E+01 -.1131200E+02
 .5044907E+02  .9536954E+02  .1000000E+01 -.8160571E+02
-.4965340E+02 -.8440435E+02  .1000000E+01  .6619043E+02
-.8393531E+01 0.            -.1000000E+01  .1645875E+02
 .3025346E+02-0.             .1000000E+01 -.2644327E+02
 .8393531E+01-0.             .1000000E+01 -.1645875E+02
 .2532501E+02  .3339651E+02  .1000000E+01 -.4268741E+02
-.7151155E+03 -.1027535E+04 -.1000000E+01  .1022627E+04
0.            0.            -.1000000E+01  .1262500E+02
-.3359775E+02 0.            -.1000000E+01  .3191347E+02
0.            0.             .1000000E+01 -.1131200E+02
 .2673140E+02  .3185901E+02  .1000000E+01 -.4085582E+02
-.1280343E+03 -.1423635E+03 -.1000000E+01  .1691946E+03
-.8632479E+01 0.            -.1000000E+01  .1758091E+02
 .3359775E+02-0.             .1000000E+01 -.3191347E+02
 .8632479E+01-0.             .1000000E+01 -.1758091E+02
 .2118150E+02  .2209662E+02  .1000000E+01 -.3371874E+02
-.4781464E+02 -.4672369E+02 -.1000000E+01  .6733352E+02
0.            0.            -.1000000E+01  .1262500E+02
-.3778417E+02 0.            -.1000000E+01  .3875087E+02
0.            0.             .1000000E+01 -.1131200E+02
 .2255065E+02  .2027442E+02  .1000000E+01 -.3413372E+02
-.4562465E+02 -.3652234E+02 -.1000000E+01  .6109297E+02
-.8834829E+01 0.            -.1000000E+01  .1876841E+02
 .3778417E+02-0.             .1000000E+01 -.3875087E+02
 .8834829E+01-0.             .1000000E+01 -.1876841E+02
 .2054127E+02  .1729334E+02  .1000000E+01 -.3187446E+02
-.3299550E+02 -.2396795E+02 -.1000000E+01  .4654409E+02
0.            0.            -.1000000E+01  .1262500E+02
-.4314821E+02 0.            -.1000000E+01  .4752327E+02
0.            0.             .1000000E+01 -.1131200E+02
 .2160787E+02  .1647744E+02  .1000000E+01 -.3261277E+02
-.3412232E+02 -.2130783E+02 -.1000000E+01  .4665277E+02
-.9153015E+01 0.            -.1000000E+01  .2002796E+02
 .4314821E+02-0.             .1000000E+01 -.4752327E+02
 .9153015E+01-0.             .1000000E+01 -.2002796E+02
 .2070385E+02  .1528144E+02  .1000000E+01 -.3169931E+02
-.2854266E+02 -.1654635E+02 -.1000000E+01  .4070317E+02
0.            0.            -.1000000E+01  .1262500E+02
-.5030651E+02 0.            -.1000000E+01  .5921638E+02
0.            0.             .1000000E+01 -.1131200E+02
 .2153752E+02  .1485108E+02  .1000000E+01 -.3247425E+02
-.2934954E+02 -.1593291E+02 -.1000000E+01  .4138954E+02
-.1126845E+02 0.            -.1000000E+01  .2306128E+02
 .5030651E+02-0.             .1000000E+01 -.5921638E+02
 .1126845E+02-0.             .1000000E+01 -.2306128E+02
 .2121196E+02  .1438027E+02  .1000000E+01 -.3217584E+02
```


The FORMATs for the data inputs are listed at the end of SUBROUTINE

```
-.2633088E+02 -.1327289E+02 -.1000000E+01  .3332961E+02
0.            0.            -.1000000E+01  .1262500E+02
-.5799470E+02 0.            -.1000000E+01  .7178133E+02
0.            0.             .1000000E+01 -.1131200E+02
 .4407803E+01  .2094988E+02  .1000000E+01 -.1408210E+02
-.3459843E+02 -.9807297E+01 -.1000000E+01  .4706399E+02
-.3367530E+02 0.            -.1000000E+01  .4697482E+02
 .5799470E+02-0.             .1000000E+01 -.7178133E+02
 .3367530E+02-0.             .1000000E+01 -.4697482E+02
 .5752650E+01  .1998723E+02  .1000000E+01 -.1559353E+02
-.3455196E+02 -.9313626E+01 -.1000000E+01  .4705950E+02
0.            0.            -.1000000E+01  .1262500E+02
 12.62500
  -.12328     .44971        .44971
  -.11826     .46846        .43195
  -.10653     .50393        .44056
  -.09479     .51874        .44870
  -.08306     .53289        .45639
  -.07132     .54642        .46367
  -.01265     .60518        .49429
   .04603     .63042        .51652
   .10470     .68345        .53141
   .16338     .70526        .53957
   .22205     .71657        .54129
   .28073     .71790        .53664
   .33940     .70962        .52547
   .39808     .69197        .50737
   .45675     .66506        .48159
   .51543     .62890        .44757
   .57410     .58339        .40429
   .63278     .52831        .35202
   .69145     .46330        .29130
   .75013     .38786        .22492
   .80880     .30175        .15241
   .86748     .20527        .07506
   .92615     .09932       -.00658
   .98483    -.01493       -.09206
  1.02003    -.08692       -.14506
  1.03177    -.11143       -.16300
  1.03702    -.14995       -.14995
0.            0.             .1000000E+01 -.1262500E+02
 .1154448E+02  .2113333E+02 -.1000000E+01  .4544333E+01
 .1351688E+02 -.4175724E+01  .1000000E+01 -.9080774E+01
-.1573586E+02 0.            -.1000000E+01  .1094866E+02
 .1573586E+02-0.             .1000000E+01 -.1094866E+02
 .1166366E+02  .2122566E+02 -.1000000E+01  .4516353E+01
 .1309719E+02 -.4965503E+01  .1000000E+01 -.8727430E+01
0.            0.            -.1000000E+01  .1393800E+02
-.2046765E+02 0.            -.1000000E+01  .1044458E+02
0.            0.             .1000000E+01 -.1262500E+02
```

```
-.6308618E+01   .9353333E+01 -.1000000E+01   .7832239E+01
 .9291038E+01  -.7523719E+01  .1000000E+01  -.7840774E+01
-.1498516E+02  0.            -.1000000E+01   .1138033E+02
 .2046765E+02-0.              .1000000E+01  -.1044458E+02
 .1498516E+02-0.              .1000000E+01  -.1138033E+02
-.4322707E+01   .1149415E+02 -.1000000E+01   .7020141E+01
 .8626378E+01  -.8408645E+01  .1000000E+01  -.7427610E+01
0.             0.            -.1000000E+01   .1393800E+02
-.2074577E+02  0.            -.1000000E+01   .1090186E+02
0.             0.             .1000000E+01  -.1262500E+02
-.5676871E+01   .1054640E+02 -.1000000E+01   .7340208E+01
 .8630340E+01  -.8405896E+01  .1000000E+01  -.7428746E+01
-.9820494E+01  0.            -.1000000E+01   .1250077E+02
 .2074577E+02-0.              .1000000E+01  -.1090186E+02
 .9820494E+01-0.              .1000000E+01  -.1250077E+02
-.4148455E+01   .1443654E+02 -.1000000E+01   .5436683E+01
 .8406171E+01  -.9389169E+01  .1000000E+01  -.6473417E+01
0.             0.            -.1000000E+01   .1393800E+02
-.2163454E+02  0.            -.1000000E+01   .1235132E+02
0.             0.             .1000000E+01  -.1262500E+02
-.4479972E+01   .1416264E+02 -.1000000E+01   .5567777E+01
 .7246596E+01  -.1086480E+02  .1000000E+01  -.5358170E+01
-.7374747E+01  0.            -.1000000E+01   .1339714E+02
 .2163454E+02-0.              .1000000E+01  -.1235132E+02
 .7374747E+01-0.              .1000000E+01  -.1339714E+02
-.2797619E+01   .2239384E+02 -.1000000E+01   .1017600E+01
 .7208483E+01  -.1409605E+02  .1000000E+01  -.3745784E+01
0.             0.            -.1000000E+01   .1393800E+02
-.2329253E+02  0.            -.1000000E+01   .1506373E+02
0.             0.             .1000000E+01  -.1262500E+02
-.1962244E+01   .2330662E+02 -.1000000E+01   .4450784E+00
 .4618695E+01  -.1636556E+02  .1000000E+01  -.1923556E+01
-.7558140E+01  0.            -.1000000E+01   .1430328E+02
 .2329253E+02-0.              .1000000E+01  -.1506373E+02
 .7558140E+01-0.              .1000000E+01  -.1430328E+02
 .2939142E+01   .4372064E+02 -.1000000E+01  -.1169318E+02
 .3404272E+01  -.2312852E+02  .1000000E+01   .3192286E+01
0.             0.            -.1000000E+01   .1393800E+02
-.2523059E+02  0.            -.1000000E+01   .1822745E+02
0.             0.             .1000000E+01  -.1262500E+02
 .6675829E+01   .4952014E+02 -.1000000E+01  -.1566212E+02
-.1689448E+01  -.2852614E+02  .1000000E+01   .8191120E+01
-.7751343E+01  0.            -.1000000E+01   .1525581E+02
 .2523059E+02-0.              .1000000E+01  -.1822745E+02
 .7751343E+01-0.              .1000000E+01  -.1525581E+02
 .8331218E+02   .3125684E+03 -.1000000E+01  -.1798965E+03
-.9438926E+01  -.5194184E+02  .1000000E+01   .2743754E+02
0.             0.            -.1000000E+01   .1393800E+02
-.2751467E+02  0.            -.1000000E+01   .2196348E+02
0.             0.             .1000000E+01  -.1262500E+02
```

```
 .5547516E+03  .1486983E+04  .1000000E+01 -.9822725E+03
-.3149567E+02 -.8294472E+02  .1000000E+01  .5692387E+02
-.7954201E+01  0.           -.1000000E+01  .1625808E+02
 .2751467E+02-0.             .1000000E+01 -.2196348E+02
 .7954201E+01-0.             .1000000E+01 -.1625808E+02
 .3098206E+02  .8262287E+02  .1000000E+01 -.5694036E+02
-.2532044E+03 -.5156470E+03 -.1000000E+01  .4712123E+03
 0.           0.           -.1000000E+01  .1393800E+02
-.3026043E+02  0.           -.1000000E+01  .2644645E+02
 0.           0.            .1000000E+01 -.1262500E+02
 .3079852E+02  .4669517E+02  .1000000E+01 -.4318482E+02
-.7989471E+02 -.1147981E+03 -.1000000E+01  .1254652E+03
-.8168471E+01  0.           -.1000000E+01  .1731452E+02
 .3026043E+02-0.             .1000000E+01 -.2644645E+02
 .8168471E+01-0.             .1000000E+01 -.1731452E+02
 .2262629E+02  .2983247E+02  .1000000E+01 -.3767572E+02
-.3847274E+02 -.4350252E+02 -.1000000E+01  .6300808E+02
 0.           0.           -.1000000E+01  .1393800E+02
-.3360635E+02  0.           -.1000000E+01  .3191840E+02
 0.           0.            .1000000E+01 -.1262500E+02
 .2428277E+02  .2533187E+02  .1000000E+01 -.3680716E+02
-.3725938E+02 -.3640927E+02 -.1000000E+01  .5525642E+02
-.8394067E+01  0.           -.1000000E+01  .1842908E+02
 .3360635E+02-0.             .1000000E+01 -.3191840E+02
 .8394067E+01-0.             .1000000E+01 -.1842908E+02
 .2244016E+02  .2239415E+02  .1000000E+01 -.3467586E+02
-.3016067E+02 -.2745530E+02 -.1000000E+01  .4619964E+02
 0.           0.           -.1000000E+01  .1393800E+02
-.3779505E+02  0.           -.1000000E+01  .3875839E+02
 0.           0.            .1000000E+01 -.1262500E+02
 .2396263E+02  .2017372E+02  .1000000E+01 -.3508066E+02
-.3144381E+02 -.2284080E+02 -.1000000E+01  .4494836E+02
-.8633046E+01  0.           -.1000000E+01  .1960741E+02
 .3779505E+02-0.             .1000000E+01 -.3875839E+02
 .8633046E+01-0.             .1000000E+01 -.1960741E+02
 .2404592E+02  .2028333E+02  .1000000E+01 -.3516472E+02
-.2938481E+02 -.2077309E+02 -.1000000E+01  .4266152E+02
 0.           0.           -.1000000E+01  .1393800E+02
-.4316239E+02  0.           -.1000000E+01  .4753474E+02
 0.           0.            .1000000E+01 -.1262500E+02
 .2545493E+02  .1873825E+02  .1000000E+01 -.3607652E+02
-.3117826E+02 -.1807424E+02 -.1000000E+01  .4329588E+02
-.8885430E+01  0.           -.1000000E+01  .2085424E+02
 .4316239E+02-0.             .1000000E+01 -.4753474E+02
 .8885430E+01-0.             .1000000E+01 -.2085424E+02
 .2657598E+02  .2005933E+02  .1000000E+01 -.3710635E+02
-.3145340E+02 -.1829732E+02 -.1000000E+01  .4357286E+02
 0.           0.           -.1000000E+01  .1393800E+02
-.5032580E+02  0.           -.1000000E+01  .5923424E+02
 0.           0.            .1000000E+01 -.1262500E+02
```

```
 .2791757E+02  .1892621E+02  .1000000E+01 -.3835632E+02
-.3311073E+02 -.1669048E+02 -.1000000E+01  .4494821E+02
-.1094440E+02  0.           -.1000000E+01  .2378862E+02
 .5032580E+02 -0.            .1000000E+01 -.5923424E+02
 .1094440E+02 -0.            .1000000E+01 -.2378862E+02
 .2944375E+02  .2062800E+02  .1000000E+01 -.3966621E+02
-.3398201E+02 -.1734653E+02 -.1000000E+01  .4577991E+02
 0.           0.           -.1000000E+01  .1393800E+02
-.5802033E+02  0.           -.1000000E+01  .7180747E+02
 0.           0.            .1000000E+01 -.1262500E+02
 .9980721E+01  .3467739E+02  .1000000E+01 -.1777533E+02
-.4224144E+02 -.1138636E+02 -.1000000E+01  .5472284E+02
-.3313983E+02  0.           -.1000000E+01  .4699166E+02
 .5802033E+02 -0.            .1000000E+01 -.7180747E+02
 .3313983E+02 -0.            .1000000E+01 -.4699166E+02
 .1116405E+02  .3290553E+02  .1000000E+01 -.1926816E+02
-.4267423E+02 -.1192779E+02 -.1000000E+01  .5509046E+02
 0.           0.           -.1000000E+01  .1393800E+02
 13.93800
  -.18997     .54827     .54827
  -.18285     .58577     .53267
  -.17068     .59915     .53767
  -.15852     .61193     .54239
  -.14635     .62411     .54682
  -.13418     .63572     .55098
  -.07334     .68255     .56780
  -.01251     .72256     .57837
   .04833     .74777     .58300
   .10917     .76194     .58182
   .17001     .76568     .57482
   .23084     .75944     .56179
   .29168     .74357     .54239
   .35252     .71832     .51608
   .41336     .68382     .48219
   .47419     .64016     .44015
   .53503     .58731     .38931
   .59587     .52520     .33219
   .65671     .45364     .26798
   .71754     .37243     .19824
   .77838     .28158     .12374
   .83922     .18138     .04501
   .90006     .07241    -.03747
   .96089    -.04458    -.12336
   .99740    -.11828    -.17641
  1.00956    -.14339    -.19434
  1.01535    -.18250    -.18250
0.           0.            .1000000E+01 -.1393800E+02
 .1038135E+02  .1889210E+02 -.1000000E+01  .5552172E+01
 .1208858E+02 -.4583112E+01  .1000000E+01 -.9128750E+01
-.1502519E+02  0.           -.1000000E+01  .1137350E+02
```

```
 .1502519E+02-0.              .1000000E+01 -.1137350E+02
 .1596507E+02  .2304411E+02 -.1000000E+01  .4272791E+01
 .1109686E+02 -.8130875E+01  .1000000E+01 -.8370675E+01
0.             0.            -.1000000E+01  .1525000E+02
-.2046483E+02  0.            -.1000000E+01  .1044506E+02
0.             0.             .1000000E+01 -.1393800E+02
-.3932838E+01  .1045748E+02 -.1000000E+01  .7644059E+01
 .8220663E+01 -.8013189E+01  .1000000E+01 -.7733791E+01
-.1483492E+02  0.            -.1000000E+01  .1175691E+02
 .2046483E+02-0.              .1000000E+01 -.1044506E+02
 .1483492E+02-0.              .1000000E+01 -.1175691E+02
-.2825026E+00  .1395895E+02 -.1000000E+01  .8263621E+01
 .6836732E+01 -.9689856E+01  .1000000E+01 -.6883909E+01
0.             0.            -.1000000E+01  .1525000E+02
-.2074636E+02  0.            -.1000000E+01  .1090177E+02
0.             0.             .1000000E+01 -.1393800E+02
-.3400117E+01  .1183234E+02 -.1000000E+01  .6970231E+01
 .7667353E+01 -.9111221E+01  .1000000E+01 -.7129479E+01
-.9629358E+01  0.            -.1000000E+01  .1323178E+02
 .2074636E+02-0.              .1000000E+01 -.1090177E+02
 .9629358E+01-0.              .1000000E+01 -.1323178E+02
 .5851485E+00  .1940229E+02 -.1000000E+01  .2964293E+01
 .7018807E+01 -.1212296E+02  .1000000E+01 -.5112348E+01
0.             0.            -.1000000E+01  .1525000E+02
-.2163232E+02  0.            -.1000000E+01  .1235149E+02
0.             0.             .1000000E+01 -.1393800E+02
-.2127902E+01  .1703301E+02 -.1000000E+01  .4110595E+01
 .6443527E+01 -.1260019E+02  .1000000E+01 -.4827370E+01
-.7196139E+01  0.            -.1000000E+01  .1428579E+02
 .2163232E+02-0.              .1000000E+01 -.1235149E+02
 .7196139E+01-0.              .1000000E+01 -.1428579E+02
 .5730219E+01  .4344209E+02 -.1000000E+01 -.1166568E+02
 .6092357E+01 -.1847947E+02  .1000000E+01 -.4140530E+00
0.             0.            -.1000000E+01  .1525000E+02
-.2329545E+02  0.            -.1000000E+01  .1506387E+02
0.             0.             .1000000E+01 -.1393800E+02
 .2606082E+01  .3876626E+02 -.1000000E+01 -.8788684E+01
 .3180391E+01 -.2160748E+02  .1000000E+01  .2065720E+01
-.7370787E+01  0.            -.1000000E+01  .1519111E+02
 .2329545E+02-0.              .1000000E+01 -.1506387E+02
 .7370787E+01-0.              .1000000E+01 -.1519111E+02
 .2365983E+03  .6906897E+03  .1000000E+01 -.5661483E+03
 .8203016E+00 -.3377712E+02  .1000000E+01  .1178501E+02
0.             0.            -.1000000E+01  .1525000E+02
-.2523077E+02  0.            -.1000000E+01  .1322748E+02
0.             0.             .1000000E+01 -.1393800E+02
 .2300495E+03  .8630935E+03  .1000000E+01 -.5491721E+03
-.8475426E+01 -.4563976E+02  .1000000E+01  .2321404E+02
-.7554557E+01  0.            -.1000000E+01  .1614151E+02
 .2523077E+02-0.              .1000000E+01 -.1322748E+02
```

```
 .7554557E+01-0.              .1000000E+01 -.1614151E+02
 .2004933E+02  .4846921E+02  .1000000E+01 -.4607520E+02
-.3077252E+02 -.1115139E+03  .1000000E+01  .7795612E+02
0.             0.           -.1000000E+01  .1525000E+02
-.2752255E+02 0.            -.1000000E+01  .2196578E+02
0.             0.            .1000000E+01 -.1393800E+02
 .2098318E+02  .4241251E+02  .1000000E+01 -.4306249E+02
-.4260826E+03 -.8677110E+03  .1000000E+01  .7255456E+03
-.7747269E+01 0.            -.1000000E+01  .1714041E+02
 .2752255E+02-0.              .1000000E+01 -.2196578E+02
 .7747269E+01-0.              .1000000E+01 -.1714041E+02
 .1665510E+02  .2854384E+02  .1000000E+01 -.3458611E+02
-.6094806E+02 -.1064079E+03 -.1000000E+01  .1118953E+03
0.             0.           -.1000000E+01  .1525000E+02
-.3026528E+02 0.            -.1000000E+01  .2644846E+02
0.             0.            .1000000E+01 -.1393800E+02
 .1858275E+02  .2450111E+02  .1000000E+01 -.3343355E+02
-.4755127E+02 -.5994781E+02 -.1000000E+01  .7458731E+02
-.7950551E+01 0.            -.1000000E+01  .1819178E+02
 .3026528E+02-0.              .1000000E+01 -.2644846E+02
 .7950551E+01-0.              .1000000E+01 -.1819178E+02
 .1807607E+02  .2333352E+02  .1000000E+01 -.3270722E+02
-.3327985E+02 -.3834365E+02 -.1000000E+01  .5426332E+02
0.             0.           -.1000000E+01  .1525000E+02
-.3362378E+02 0.            -.1000000E+01  .3192773E+02
0.             0.            .1000000E+01 -.1393800E+02
 .2019496E+02  .2015356E+02  .1000000E+01 -.3260099E+02
-.3335146E+02 -.3035989E+02 -.1000000E+01  .4961270E+02
-.8164281E+01 0.            -.1000000E+01  .1929957E+02
 .3362378E+02-0.              .1000000E+01 -.3192773E+02
 .8164281E+01-0.              .1000000E+01 -.1929957E+02
 .2146057E+02  .2227384E+02  .1000000E+01 -.3400038E+02
-.2847542E+02 -.2448249E+02 -.1000000E+01  .4374431E+02
0.             0.           -.1000000E+01  .1525000E+02
-.3780980E+02 0.            -.1000000E+01  .3876807E+02
0.             0.            .1000000E+01 -.1393800E+02
 .2335161E+02  .1969766E+02  .1000000E+01 -.3455181E+02
-.2978067E+02 -.2105901E+02 -.1000000E+01  .4304847E+02
-.8390356E+01 0.            -.1000000E+01  .2046889E+02
 .3780980E+02-0.              .1000000E+01 -.3876807E+02
 .8390356E+01-0.              .1000000E+01 -.2046889E+02
 .2584742E+02  .2298359E+02  .1000000E+01 -.3690111E+02
-.2811593E+02 -.1942006E+02 -.1000000E+01  .4129118E+02
0.             0.           -.1000000E+01  .1525000E+02
-.4320053E+02 0.            -.1000000E+01  .4756443E+02
0.             0.            .1000000E+01 -.1393800E+02
 .2751829E+02  .2077058E+02  .1000000E+01 -.3792784E+02
-.2975532E+02 -.1730950E+02 -.1000000E+01  .4197295E+02
-.8628741E+01 0.            -.1000000E+01  .2170438E+02
 .4320053E+02-0.              .1000000E+01 -.4756443E+02
```

```
 .8628741E+01-0.              .1000000E+01 -.2170438E+02
 .3129069E+02  .2491864E+02  .1000000E+01 -.4116779E+02
-.3139983E+02 -.1865689E+02 -.1000000E+01  .4355066E+02
0.             0.           -.1000000E+01  .1525000E+02
-.5036468E+02 0.            -.1000000E+01  .5926924E+02
0.             0.            .1000000E+01 -.1393800E+02
 .3278380E+02  .2296801E+02  .1000000E+01 -.4258477E+02
-.3315892E+02 -.1692637E+02 -.1000000E+01  .4500865E+02
-.1063295E+02 0.            -.1000000E+01  .2454331E+02
 .5036468E+02-0.              .1000000E+01 -.5926924E+02
 .1063295E+02-0.              .1000000E+01 -.2454331E+02
 .3691069E+02  .2724715E+02  .1000000E+01 -.4594606E+02
-.3858837E+02 -.2100614E+02 -.1000000E+01  .4994143E+02
0.             0.           -.1000000E+01  .1525000E+02
-.5807890E+02 0.            -.1000000E+01  .7186579E+02
0.             0.            .1000000E+01 -.1393800E+02
 .1765391E+02  .5203409E+02  .1000000E+01 -.2236667E+02
-.4611845E+02 -.1289047E+02 -.1000000E+01  .5041185E+02
-.3236310E+02 0.            -.1000000E+01  .4079737E+02
 .5807880E+02-0.              .1000000E+01 -.7186579E+02
 .3236310E+02-0.              .1000000E+01 -.4679737E+02
 .1821143E+02  .5006184E+02  .1000000E+01 -.2329259E+02
-.4840599E+02 -.1503418E+02 -.1000000E+01  .6034328E+02
0.             0.           -.1000000E+01  .1525000E+02
 15.25000
  -.25800     .65510     .65510
  -.24739     .68762     .63853
  -.23479     .69711     .63902
  -.22219     .70620     .63935
  -.20959     .71489     .63953
  -.19699     .72318     .63954
  -.13399     .75866     .63725
  -.07099     .78429     .63098
  -.00799     .80020     .62063
   .05501     .80550     .60609
   .11801     .80326     .58716
   .18101     .79057     .55358
   .24401     .76849     .53504
   .30701     .73706     .50115
   .37001     .69632     .46152
   .43301     .64628     .41580
   .49601     .58696     .35391
   .55901     .51834     .30600
   .62201     .44041     .24251
   .68501     .35341     .17394
   .74801     .25799     .10081
   .81101     .15512     .02357
   .87401     .04593    -.05741
   .93701    -.06850    -.14180
   .97481    -.13924    -.19396
```

```
  .98741    -.16313    -.21159
  .99397    -.20093    -.20093
 37     4
 41     5
 46     5
 51     5
 56     5
 61     5
 66     5
 71     5
 76     5
 81     5
 86     5
 91     5
 96     5
101     5
106     5
111     5
116     5
121     5
126     5
131     5
136     5
141     5
146     5
151     5
156     5
161     4
165     4
169     5
174     5
179     5
184     5
189     5
194     5
199     5
204     5
209     5
214     5
219     5
224     5
229     5
234     5
239     5
244     5
249     5
254     5
259     5
264     5
269     5
```

274	5
279	5
284	5
289	4
293	4
297	5
302	5
307	5
312	5
317	5
322	5
327	5
332	5
337	5
342	5
347	5
352	5
357	5
362	5
367	5
372	5
377	5
382	5
387	5
392	5
397	5
402	5
407	5
412	5
417	4
421	4
425	5
430	5
435	5
440	5
445	5
450	5
455	5
460	5
465	5
470	5
475	5
480	5
485	5
490	5
495	5
500	5
505	5
510	5
515	5

520	5						
525	5						
530	5						
535	5						
540	5						
545	4						
62	61	63	61	62	64	63	64
82	81	83	81	82	85	83	85
102	101	103	101	102	105	103	105
122	121	123	121	122	125	123	125
142	141	143	141	142	145	143	145
162	161	163	161	162	165	163	165
182	181	183	181	182	185	183	185
202	201	203	201	202	205	203	205
222	221	223	221	222	225	223	225
242	241	243	241	242	245	243	245
262	261	263	261	262	265	263	265
282	281	283	281	282	285	283	285
302	301	303	301	302	305	303	305
322	321	323	321	322	324	323	324
342	341	343	341	342	345	343	345
362	361	363	361	362	365	363	365
382	381	383	381	382	385	383	385
402	401	403	401	402	405	403	405
422	421	423	421	422	425	423	425
442	441	443	441	442	445	443	445
462	461	463	461	462	465	463	465
482	481	483	481	482	485	483	485
502	501	503	501	502	505	503	505
522	521	523	521	522	525	523	525
542	541	543	541	542	545	543	545
562	561	563	561	562	565	563	565
582	581	583	581	582	584	583	584
602	601	603	601	602	605	603	605
622	621	623	621	622	625	623	625
642	641	643	641	642	645	643	645
662	661	663	661	662	665	663	665
682	681	683	681	682	685	683	685
702	701	703	701	702	705	703	705
722	721	723	721	722	725	723	725
742	741	743	741	742	745	743	745
762	761	763	761	762	765	763	765
782	781	783	781	782	785	783	785
802	801	803	801	802	805	803	805
822	821	823	821	822	825	823	825
842	841	843	841	842	844	843	844
862	861	863	861	862	865	863	865
882	881	883	881	882	885	883	885
902	901	903	901	902	905	903	905
922	921	923	921	922	925	923	925

```
 942  941  943  941  942  945  943  945
 962  961  963  961  962  965  963  965
 982  981  983  981  982  985  983  985
1002 1001 1003 1001 1002 1005 1003 1005
1022 1021 1023 1021 1022 1025 1023 1025
1042 1041 1043 1041 1042 1045 1043 1045
1062 1061 1063 1061 1062 1065 1063 1065
1082 1081 1083 1081 1082 1085 1083 1085
 313  312
 573  572
 833  832
1093 1092
 852  854  854  853
 872  874  874  873
 892  894  894  893
 912  914  914  913
 932  934  934  933
 952  954  954  953
 972  974  974  973
 992  994  994  993
1012 1014 1014 1013
1032 1034 1034 1033
1052 1054 1054 1053
1072 1074 1074 1073
1092 1094 1094 1093
```

APPENDIX E

Fitting Equations to Planes

The important concept to understand here is that quadrilaterals in 3-space are not necessarily planar, even though all four sides may be straight. For a simple proof, crease a sheet of paper along a diagonal. Therefore, we represent the quadrilateral surface elements of the bodies to be drawn by pairs of triangles, which must be planar (hence the wedge-shaped blade segments used by DRACULA). The choice of which pair of triangles to use to represent a quadrilateral is somewhat arbitary; there may be objects for which one choice is more convenient than the other.

Fitting a plane equation to a triangle is then a simple matter of solving the three simultaneous equations obtained when one substitutes the x, y, z coordinates of the three corners into

$$z = ax + by + d \qquad \text{(E-1)}$$

Special cases result when one encounters a plane of the form x = constant, but they are not a great difficulty.

Having decided how he will subdivide his object into segments, one can write a computer program to read object coordinates and create from them a data file like that in Appendix D.

APPENDIX F

Listings for the Line-Against-Line Algorithm

The operation of the LXL Algorithm is described in Section 4.3. The program listings on the following pages were written in CDC Fortran Extended.

Data are input on two files. From the first file, INPUT, are read the plot scale and offsets. SCALE is the size of the picture in plotter units and XOFF and YOFF are the location coordinates in plotter units of the lower left corner of the picture relative to the plotter origin. (Plotter units are 1/inch for CALCOMP plotters and 1024/screen width for TEKTRONIX 4010 series graphics terminals.)

The second input file, TAPE4, contains the point, line, and polygon lists which describe the objects. The first card image on this file contains the numbers of points (NPTS), lines (NLINES), and polygons (NPOLYS) in the picture. Following this card image are NPTS card images, each containing the perspective coordinates of a single point. (If the z-axis passes through the viewpoint and lies along the line of sight, the perspective coordinates of a point [x,y,z] in the scene are defined to be [x/z, y/z, z].) Next come NLINES card images, each identifying a line by its end points. For example, a line from point #1 in the point list to point #2 is identified by (1,2). The polygon data cards are last. For each of the NPOLYS polygons there is a set of data card images. The first card image of a set defines the number of edges n of the polygon. This is followed by n card images, each identifying an edge from the line list. The edges must be identified in sequence around the polygon's circumference.

The FORMATs for the data inputs are listed at the end of SUBROUTINE READINP.

The code, as listed here, uses an online CALCOMP Plotter. However, relatively minor changes in HIDE and LINEDRW permit the use of other plotters and graphics terminals (in particular, TEKTRONIC 4010 series terminals).

A small sample data list follows the program listing. (These data describe a 2" cube viewed from 10" on a 30° azimuth and 50° elevation.

For a two-fold decrease in execution time, the subprograms LINESX, NXTEDGE, ICLIP, PUSH, IWCODE, and IANGLE were also coded in CDC COMPASS Assembly Language. The COMPASS listings are not included here due to their length and computer specificity. Listings are available from the authors.

```
      PROGRAM LXL (INPUT,OUTPUT,TAPE4,TAPE5=INPUT,TAPE6=
     & OUTPUT,PLOT)
C
C          LINE BY LINE ALGORITHM FOR HIDDEN LINE REMOVAL
C          ==============================================
C
C          INITIALIZE PLOTTER
C
      CALL PLOT (0.0,0.0,-3)
C
C          READ INPUT DATA - POINT, LINE, & POLYGON LISTS
C
      CALL READINP
C
C          INITIALIZE LXL TABLES
C
      CALL INITIAL
C
C          MAIN ALGORITHM
C
      CALL HIDE
C
C          TERMINATE PLOT
C
      CALL PLOT (12.,0.0,-3)
      CALL SYMBOL (0.0,0.5,0.1,"FINISHED",90.0,8)
      CALL PLOTE
C
      STOP
C
      END
```

```
      SUBROUTINE READINP
C
C          READ INPUT DATA AND FORM LINKED LISTS
C
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   //  IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     4         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     6         AZ(1770),AXY(1770),AXYZ(1770)
C
      DO 1 I=1,400
      IEDLINK(I)=0
    1 CONTINUE
C
C          READ PLOT SCALE AND OFFSETS
C
      READ (5,8) SCALE,XOFF,YOFF
C
C          READ NUMBER OF POINTS, LINES, AND POLYGONS
C
      READ (4,9) NPTS,NEDGES,NPOLYS
      NLINES=NEDGES
C
C          READ POINT COORDINATE LIST
C
      READ (4,10) (XS(I),YS(I),ZS(I),I=1,NPTS)
C
C          READ LINE ENDPOINTS LIST
C
      READ (4,11) (IED1(I),IED2(I),I=1,NEDGES)
C
C          READ AND LINK POLYGON AND POLYGON EDGE LISTS
C
      DO 2 I=1,NEDGES
      IEDLINK(I)=-1
    2 CONTINUE
      IPPTR=0
      DO 7 I=1,NPOLYS
      IPLINK(I)=IPPTR
      IPPTR=I
```

```
      READ (4,12) J
      IPENO(I)=J
      IPEDGE(I)=K=IFINDED(NEDGES,M)
      IEDNO(K)=M
    3 CONTINUE
      IF (J.EQ.1) GO TO 4
      K=IEDLINK(K)=IFINDED(NEDGES,M)
      IEDNO(K)=M
      J=J-1
      GO TO 3
    4 CONTINUE
C
C         ARRANGE POLYGON EDGES AND VERTICES IN SEQUENCE IN
C     $         LINKED LISTS
C
      IEDLINK(K)=0
      K=IPEDGE(I)
      J=IED2(K)
      IF ((J.EQ.IED1(IEDLINK(K))).OR.(J.EQ.IED2(IEDLINK(K))
     $ )) GO TO 5
      IX=IED1(K) $ IED1(K)=IED2(K) $ IED2(K)=IX
    5 CONTINUE
      IF (K.EQ.0) GO TO 7
      J=IEDLINK(K)
      IF ((J.EQ.0).OR.(IED2(K).EQ.IED1(J))) GO TO 6
      IX=IED1(J) $ IED1(J)=IED2(J) $ IED2(J)=IX
    6 CONTINUE
      K=J
      GO TO 5
    7 CONTINUE
      RETURN
C
C
C         FORMAT FOR SCALE AND OFFSETS
C
    8 FORMAT (3F10.4)
C
C         FORMAT FOR NPTS, NLINES, AND NPOLYS
C
    9 FORMAT (3I5)
C
C         FORMAT FOR POINT COORDINATES
C
   10 FORMAT (3E23.15)
C
C         FORMAT FOR LINE LIST
C
   11 FORMAT (2I5)
C
C         FORMAT FOR NUMBER OF EDGES ON POLYGON
```

```
C
   12 FORMAT (I5)
C
C          FORMAT FOR POLYGON EDGE IDENT
C
C  13 FORMAT (5X,I5)
C
      END
```

```
      FUNCTION IFINDED (NEDGES,M)
C
C          READ THE NEXT POLYGON EDGE AND GIVE IT AN UNIQUE
C     $          STORAGE
C                LOCATION
C
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   // IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     4         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     6         AZ(1770),AXY(1770),AXYZ(1770)
C
C          READ NEXT POLYGON EDGE NUMBER
C
      READ (4,2) M
      IF (IEDLINK(M).EQ.-1) GO TO 1
C
C          THIS EDGE ALSO BELONGS TO ANOTHER POLYGON - GIVE
C     $          IT A
C                DIFFERENT STORAGE LOCATION
C
      NEDGES=NEDGES+1
      IED1(NEDGES)=IED1(M)
      IED2(NEDGES)=IED2(M)
      IFINDED=NEDGES
      RETURN
C
C          THIS EDGE HAS NOT BEEN USED PREVIOUSLY
C
    1 CONTINUE
      IFINDED=M
      RETURN
C
    2 FORMAT (5X,I5)
      END
```

```
      SUBROUTINE INITIAL
C
C          INITIALIZE LXL CONSTANTS AND TABLES
C
      DIMENSION X(65),Y(65),Z(65),ISIDE(65)
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   //  IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     4         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     6         AZ(1770),AXY(1770),AXYZ(1770)
      LOGICAL CHANGE
C
C          DEVELOP SCALE FACTORS AND OFFSETS
C
      XMIN=YMIN=ZMIN=1.E22 $ XMAX=YMAX=ZMAX=-1.E22
      DO 1 I=1,NPTS
      XMIN=AMIN1(XMIN,XS(I)) $ YMIN=AMIN1(YMIN,YS(I))
      XMAX=AMAX1(XMAX,XS(I)) $ YMAX=AMAX1(YMAX,YS(I))
      ZMIN=AMIN1(ZMIN,ZS(I)) $ ZMAX=AMAX1(ZMAX,ZS(I))
    1 CONTINUE
      SF=AMIN1(1./(XMAX-XMIN),1./(YMAX-YMIN))
      SZ=1./(ZMAX-ZMIN)
      DO 2 I=1,NPTS
      XS(I)=XS(I)*SF $ YS(I)=YS(I)*SF $ ZS(I)=ZS(I)*SZ
    2 CONTINUE
      XOFF=XOFF-XMIN*SF*SCALE $ YOFF=YOFF-YMIN*SF*SCALE
C
C          GENERATE PLANE EQUATIONS FOR PLANAR POLYGONS
C
      IOLDP=0
      IP=IPPTR
    3 CONTINUE
      IF (IP.EQ.0) GO TO 13
      NXTEDG=IPEDGE(IP)
      NSIDES=IPENO(IP)
      IF (NSIDES.LE.4) GO TO 6
      DO 4 I=1,NSIDES
      ISIDE(I)=NXTEDG
      ID=NXTEDGE(NXTEDG)
      X(I)=PX $ Y(I)=PY $ Z(I)=PZ
```

```
    4 CONTINUE
      S=0. $ JS=0
      X(NSIDES+1)=QX $ Y(NSIDES+1)=QY $ Z(NSIDES+1)=QZ
      NSM=NSIDES-1
      DO 5 I=1,NSM
      SI=ABS((X(I+1)-X(I))*(Y(I+2)-Y(I+1))-(Y(I+1)-Y(I))*
     $ (X(I+2)-X(I+1))
     1)
      IF (SI.LE.S) GO TO 5
      S=SI $ JS=I
    5 CONTINUE
      ID=NXTEDGE(ISIDE(JS))
      X1=X(JS)*Z(JS) $ Y1=Y(JS)*Z(JS) $ Z1=Z(JS)
      X2=X(JS+1)*Z(JS+1)-X1 $ Y2=Y(JS+1)*Z(JS+1)-Y1 $ Z2=
     $ Z(JS+1)-Z1
      X3=X(JS+2)*Z(JS+2)-X1 $ Y3=Y(JS+2)*Z(JS+2)-Y1 $ Z3=
     $ Z(JS+2)-Z1
      GO TO 7
    6 CONTINUE
      ID=NXTEDGE(NXTEDG)
      X1=PX*PZ $ Y1=PY*PZ $ Z1=PZ
      ID=NXTEDGE(NXTEDG)
      X2=PX*PZ-X1 $ Y2=PY*PZ-Y1 $ Z2=PZ-Z1
      X3=QX*QZ-X1 $ Y3=QY*QZ-Y1 $ Z3=QZ-Z1
    7 CONTINUE
      POLYA(IP)=Y3*Z2-Y2*Z3
      POLYB(IP)=X2*Z3-X3*Z2
      POLYC(IP)=X3*Y2-X2*Y3
      POLYD(IP)=-(POLYA(IP)*X1+POLYB(IP)*Y1+POLYC(IP)*Z1)
C
C         STORE NEAREST VERTEX OF EACH POLYGON
C
      ZMIN=1.E99
      NXTEDG=IPEDGE(IP)
    8 CONTINUE
      IF (NXTEDGE(NXTEDG).EQ.0) GO TO 9
      IF (ZMIN.GT.PZ) ZMIN=PZ
      GO TO 8
    9 CONTINUE
      PZMIN(IP)=ZMIN
      IF (POLYC(IP).NE.0.) GO TO 11
C
C         THIS POLYGON IS SEEN ON EDGE - ELIMINATE IT
C
      IF (IOLDP.EQ.0) GO TO 10
      IPLINK(IOLDP)=IPLINK(IP)
      GO TO 12
   10 CONTINUE
      IPPTR=IP
      GO TO 12
```

```
   11 CONTINUE
      IOLDP=IP
   12 CONTINUE
      IP=IPLINK(IP)
      GO TO 3
C
C          RELINK THE POLYGON LIST IN ORDER OF DISTANCE FROM
C     $          VIEWER
C
   13 CONTINUE
      CHANGE=.FALSE.
      IOLDP=0 $ IP=IPPTR
   14 CONTINUE
      IF (IP.EQ.0) GO TO 18
      J=IPLINK(IP)
      IF (.NOT.((J.NE.0).AND.(PZMIN(IP).LT.PZMIN(J)))) GO
     $ TO 17
      IF (IOLDP.EQ.0) GO TO 15
      IPLINK(IOLDP)=J
      GO TO 16
   15 CONTINUE
      IPPTR=J
   16 CONTINUE
      IPLINK(IP)=IPLINK(J)
      IPLINK(J)=IP
      CHANGE=.TRUE.
      IOLDP=J
      GO TO 14
   17 CONTINUE
      IOLDP=IP
      IP=IPLINK(IP)
      GO TO 14
   18 CONTINUE
      IF (CHANGE) GO TO 13
C
C          GENERATE THE EQUATIONS FOR THE LINES
C
      DO 20 I=1,NLINES
      IE1=IED1(I) $ IE2=IED2(I)
      AX(I)=XS(IE2)-XS(IE1) $ AY(I)=YS(IE2)-YS(IE1)
      AZ(I)=ZS(IE2)-ZS(IE1) $ AXY(I)=YS(IE2)*XS(IE1)-
     $ XS(IE2)*YS(IE1)
      IF (ABS(AX(I)).LT.ABS(AY(I))) GO TO 19
      IF (ABS(AX(I)).LT.1.E-4) AX(I)=1.E66
      ZZ=XS(IE2)*ZS(IE2)-XS(IE1)*ZS(IE1)
      IF (ABS(ZZ).LT.ZS(IE2)*1.E-10) ZZ=ZS(IE2)*1.E-10
      AZ(I)=(AZ(I)/ZZ).AND.-18
      AXYZ(I)=-ZS(IE2)*ZS(IE1)*AX(I)/ZZ
      GO TO 20
   19 CONTINUE
```

```
      IF (ABS(AY(I)).LT.1.E-4) AX(I)=1.E66
      ZZ=YS(IE2)*ZS(IE2)-YS(IE1)*ZS(IE1)
      IF (ABS(ZZ).LT.ZS(IE2)*1.E-10) ZZ=ZS(IE2)*1.E-10
      AZ(I)=(AZ(I)/ZZ).OR.1B
      AXYZ(I)=-ZS(IE2)*ZS(IE1)*AY(I)/ZZ
   20 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE HIDE
C
C          CODE FOR INTERMEDIATE SIZED WINDOWS
C
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   //  IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     4         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     6         AZ(1770),AXY(1770),AXYZ(1770)
      COMMON
     1   /LINKS/  JPLINK(690),ILLIST(1770),IJLLINK(1770),
     $ ILPTR,JLPTR,
     2         JPPTR
C
C          SET NUMBER OF INTERMEDIATE WINDOWS
C
      NW=(0.5+(FLOAT(NLINES))**0.25)
C
C          SIZE THE LARGE WINDOW RECTANGLE (LWR)
C
      WLX=WBY=1.E99 $ WRX=WTY=-1.E99
      DO 1 I=1,NPTS
      WLX=AMIN1(WLX,XS(I))
      WRX=AMAX1(WRX,XS(I))
      WBY=AMIN1(WBY,YS(I))
      WTY=AMAX1(WTY,YS(I))
    1 CONTINUE
      XMIN=WLX-1.E-6*ABS(WLX)
      DX=(WRX+1.E-6*ABS(WRX)-XMIN)/FLOAT(NW)
      YMIN=WBY-1.E-6*ABS(WBY)
      DY=(WTY+1.E-6*ABS(WTY)-YMIN)/FLOAT(NW)
C
C          PRESET LINE STATUS LIST
C
      DO 2 I=1,NLINES
      ILLIST(I)=0
    2 CONTINUE
C
C          SCAN BY INTERMEDIATE WINDOW RECTANGLES (IWR)
C
```

```
      DO 19 IJX=1,NW
      DO 19 IJY=1,NW
      WRX=XMIN+DX*FLOAT(IJX)
      WLX=WRX-DX
      WTY=YMIN+DY*FLOAT(IJY)
      WBY=WTY-DY
C
C          IDENTIFY "I" LINES AND FORM A LINKAGE LIST
C
      ILPTR=0
      DO 3 I=1,NLINES
      IF (ILLIST(I).NE.0) GO TO 3
      QX=XS(IED1(I)) $ QY=YS(IED1(I))
      PX=XS(IED2(I)) $ PY=YS(IED2(I))
      IF (ICLIP(0).EQ.0) GO TO 3
      IJLLINK(I)=ILPTR $ ILPTR=I $ ILLIST(I)=1
    3 CONTINUE
      IF (ILPTR.EQ.0) GO TO 19
C
C          DETERMINE THE "J" WINDOW RECTANGLE (JWR)
C
      WLX=WBY=1.E99 $ WRX=WTY=-1.E99
      I=ILPTR
    4 CONTINUE
      WLX=AMIN1(WLX,XS(IED1(I)),XS(IED2(I)))
      WRX=AMAX1(WRX,XS(IED1(I)),XS(IED2(I)))
      WBY=AMIN1(WBY,YS(IED1(I)),YS(IED2(I)))
      WTY=AMAX1(WTY,YS(IED1(I)),YS(IED2(I)))
      I=IJLLINK(I)
      IF (I.NE.0) GO TO 4
C
C          FIND "J" LINES AND FORM A LINKAGE LIST
C
      JLPTR=ILPTR
      DO 5 I=1,NLINES
      IF (ILLIST(I).EQ.1) GO TO 5
      QX=XS(IED1(I)) $ QY=YS(IED1(I))
      PX=XS(IED2(I)) $ PY=YS(IED2(I))
      IF (ICLIP(0).EQ.0) GO TO 5
      IJLLINK(I)=JLPTR
      JLPTR=I
    5 CONTINUE
C
C          FIND POLYGONS IN THIS WINDOW AND FORM A LINKAGE
C     $          LIST
C
      ISUR=JPPTR=0
      ZSUR=1.E99
      IP=IPPTR
    6 CONTINUE
```

```
      ITHETA=J
      NXTEDG=IPEDGE(IP)
    7 CONTINUE
      IF (NXTEDGE(NXTEDG).EQ.0) GO TO 9
      IF (ICLIP(1).EQ.0) GO TO 8
      JPLINK(IP)=JPPTR
      JPPTR=IP & ITHETA=-1
      GO TO 10
    8 CONTINUE
      ITHETA=ITHETA+IDELTA
      GO TO 7
    9 CONTINUE
      IF (IABS(ITHETA).NE.8) GO TO 10
      ZIP=GETZ(IP,WLX,WBY)
      IF (ZIP.GE.ZSUR) GO TO 10
      ZSUR=ZIP & ISUR=IP
   10 CONTINUE
      IP=IPLINK(IP)
      IF (IP.NE.0) GO TO 6
C
C          ELIMINATE ALL LINES AND POLYGONS BEYOND THE
C     &          NEAREST SURROUNDER
C
      IF ((ISUR.EQ.0).OR.(JPPTR.EQ.0)) GO TO 17
      IOLDP=0 & IP=JPPTR
   11 CONTINUE
      NXTEDG=IPEDGE(IP)
   12 CONTINUE
      NED=IEDNO(NXTEDG)
      IF (NXTEDGE(NXTEDG).EQ.0) GO TO 15
      IF (ICLIP(1).EQ.0) GO TO 12
      WXY=XX1
      IF ((AZ(NED).AND.1B).NE.0.) WXY=YY1
      ZVAL=AXYZ(NED)/(AZ(NED)*WXY-1.)
      IF (ZVAL.LE.GETZ(ISUR,XX1,YY1)) GO TO 15
   13 CONTINUE
      IF (ILLIST(NED).EQ.1) ILLIST(NED)=2
      ILLIST(NED)=ILLIST(NED)+10
      NED=IEDNO(NXTEDG)
      IF (NXTEDGE(NXTEDG).NE.0) GO TO 13
      J=JPLINK(IP)
      IF (IOLDP.EQ.0) GO TO 14
      JPLINK(IOLDP)=J
      GO TO 16
   14 CONTINUE
      JPPTR=J
      GO TO 16
   15 CONTINUE
      IOLDP=IP
   16 CONTINUE
```

```
      IP=JPLINK(IP)
      IF (IP.NE.0) GO TO 11
   17 CONTINUE
C
C          PROCESS "I" LINES AND DRAW VISIBLE PORTIONS
C
      IF (JPPTR.EQ.0) GO TO 19
      CALL LINEDRW
C
C          CHECK OFF LINES COMPLETED
C
      DO 18 I=1,NLINES
      IF (ILLIST(I).GE.10) ILLIST(I)=ILLIST(I)-10
      IF (ILLIST(I).EQ.1) ILLIST(I)=2
   18 CONTINUE
C
C          THIS INTERMEDIATE WINDOW IS PROCESSED
C
   19 CONTINUE
C
C          ALL DONE
C
      RETURN
      END
```

```
      SUBROUTINE LINEDRW
C
C          PROCESS THE LINES IN AN INTERMEDIATE WINDOW
C
      DIMENSION XI(102),YI(102),ZI(102)
      LOGICAL LINESX,LVIS,ICANCIT
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   //  IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     4         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     5         AZ(1770),AXY(1770),AXYZ(1770)
      COMMON
     1   /LINKS/  JPLINK(690),ILLIST(1770),IJLLINK(1770),
     $ ILPTR,JLPTR,
     2         JPPTR
C
C          CHECK AND DRAW LINE BY LINE
C
      I=ILPTR
    1 CONTINUE
C
C          IF THIS LINE IS SEEN END-ON, IGNORE IT
C
      IF (AX(I).EQ.1.E66) GO TO 15
      IE1=IED1(I) $ IE2=IED2(I)
      IF (LWINDOW(I,INTRSCT)-1) 15,3,2
C
C          NO INTERSECTORS - DRAW LINE
C
    2 CONTINUE
      CALL PLOT ((XOFF+SCALE*XS(IE1)),(YOFF+SCALE*YS(IE1)),
     $ 3)
      CALL PLOT ((XOFF+SCALE*XS(IE2)),(YOFF+SCALE*YS(IE2)),
     $ 2)
      GO TO 15
C
C          INTERSECTORS EXIST - SOLVE
C
    3 CONTINUE
C
```

```
C          FIND INTERSECTIONS ON LINE "I"
C
       XI(1)=XS(IE1) $ YI(1)=YS(IE1) $ ZI(1)=ZS(IE1)
       NINT=2
       J=JLPTR
     4 CONTINUE
       IF ((J.EQ.I).OR.(AX(J).EQ.1.E66)) GO TO 5
       IF (LINESX(I,J,XI(NINT),YI(NINT),ZI(NINT))) NINT=
      $ NINT+1
     5 CONTINUE
       J=IJLLINK(J)
       IF (J.NE.0) GO TO 4
       NINT=NINT-1
       IF (NINT.LE.2) GO TO 9
C
C          ARRANGE THE INTERSECTIONS IN ORDER FROM "IE1" TO
C     $          "IE2"
C
       IF (ABS(AX(I)).LT.ABS(AY(I))) GO TO 7
       NN=NINT-1
       DO 6 J=2,NN
       JP=J+1
       DO 6 K=JP,NINT
       IF (AX(I)*(XI(K)-XI(J)).GE.0.) GO TO 6
       X=XI(J) $ XI(J)=XI(K) $ XI(K)=X
       Y=YI(J) $ YI(J)=YI(K) $ YI(K)=Y
       Z=ZI(J) $ ZI(J)=ZI(K) $ ZI(K)=Z
     6 CONTINUE
       GO TO 9
     7 CONTINUE
       NN=NINT-1
       DO 8 J=2,NN
       JP=J+1
       DO 8 K=JP,NINT
       IF (AY(I)*(YI(K)-YI(J)).GE.0.) GO TO 8
       X=XI(J) $ XI(J)=XI(K) $ XI(K)=X
       Y=YI(J) $ YI(J)=YI(K) $ YI(K)=Y
       Z=ZI(J) $ ZI(J)=ZI(K) $ ZI(K)=Z
     8 CONTINUE
     9 CONTINUE
       XI(NINT+1)=XS(IE2) $ YI(NINT+1)=YS(IE2) $ ZI(NINT+1)=
      $ ZS(IE2)
C
C          DRAW VISIBLE SEGMENTS OF LINE "I"
C
       XJ=W=0.5*(XI(1)+XI(2)) $ YJ=0.5*(YI(1)+YI(2))
       IF ((AZ(I).AND.18).NE.0.) W=YJ
       ZJ=AXYZ(I)/(AZ(I)*W-1.)
       IF (.NOT.ICANCIT(INTRSCT,XJ,YJ,ZJ)) GO TO 10
       LVIS=.TRUE.
```

```
      XE=XI(1) $ YE=YI(1)
      GO TO 11
   10 CONTINUE
      LVIS=.FALSE.
   11 CONTINUE
      IF (NINT.EQ.1) GO TO 14
      DO 13 J=2,NINT
      XJ=W=0.5*(XI(J)+XI(J+1)) $ YJ=0.5*(YI(J)+YI(J+1))
      IF ((AZ(I).AND.1B).NE.0.) W=YJ
      ZJ=AXYZ(I)/(AZ(I)*W-1.)
      IF (ICANCIT(INTRSCT,XJ,YJ,ZJ)) GO TO 12
      IF (.NOT.LVIS) GO TO 13
      LVIS=.FALSE.
      CALL PLOT ((XOFF+SCALE*XE),(YOFF+SCALE*YE),3)
      CALL PLOT ((XOFF+SCALE*XI(J)),(YOFF+SCALE*YI(J)),2)
      GO TO 13
   12 CONTINUE
      IF (LVIS) GO TO 13
      XE=XI(J) $ YE=YI(J)
      LVIS=.TRUE.
   13 CONTINUE
   14 CONTINUE
      IF (.NOT.LVIS) GO TO 15
      CALL PLOT ((XOFF+SCALE*XE),(YOFF+SCALE*YE),3)
      CALL PLOT ((XOFF+SCALE*XS(IE2)),(YOFF+SCALE*YS(IE2)),
     $ 2)
   15 CONTINUE
      I=IJLLINK(I)
      IF (I.NE.0) GO TO 1
C
      RETURN
      END
```

```
      FUNCTION LWINDOW (I,INTRSCT)
C
C          FORM A LINKED LIST OF ALL POLYGONS WHICH
C     $          INTERSECT THE SWR
C
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   //  IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     +         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     6         AZ(1770),AXY(1770),AXYZ(1770)
      COMMON
     1   /LINKS/  JPLINK(690),ILLIST(1770),IJLLINK(1770),
     $ ILPTR,JLPTR,
     2         JPPTR
C
      LWINDOW=0
C
C          DEFINE SMALL WINDOW RECTANGLE LIMITS FOR LINE "I"
C
      XC=XS(IE1) $ YC=YS(IE1) $ ZC=ZS(IE1)
      XT=(XC+XS(IE2))*.5 $ YT=(YC+YS(IE2))*.5 $ ZT=(ZC+
     $ ZS(IE2))*0.499999
      IF (ABS(AX(I)).LT.ABS(AY(I))) GO TO 1
      EPX=ABS(AX(I))*1.E-5
      GO TO 2
    1 CONTINUE
      EPX=-ABS(AY(I))*1.E-5
    2 CONTINUE
      EPY=-EPX
      WLX=AMIN1(XS(IE2),XC)+EPX
      WRX=AMAX1(XS(IE2),XC)-EPX
      WBY=AMIN1(YS(IE2),YC)+EPY
      WTY=AMAX1(YS(IE2),YC)-EPY
C
C          INITIALIZE THINKER LISTS
C
      ISURNDR=INTRSCT=0
      ZMINMAX=AMAX1(ZC,ZS(IE2))
      ZSUR=1.E99
C
```

```
C          START LOOKING DOWN THE POLYGON LIST
C
      IP=JPPTR
C
C          LOOKER
C
    3 CONTINUE
      IF ((IP.EQ.0).OR.(PZMIN(IP).GT.ZMINMAX)) GO TO 9
C
C          DON'T CONSIDER POLYGONS HAVING LINE "I" AS AN
C     $          EDGE
C
      NXTEDG=IPEDGE(IP)
    4 CONTINUE
      IF (IEDNO(NXTEDG).EQ.I) GO TO 8
      NXTEDG=IEDLINK(NXTEDG)
      IF (NXTEDG.NE.0) GO TO 4
C
      ITHETA=0
      NXTEDG=IPEDGE(IP)
    5 CONTINUE
      IF (NXTEDGE(NXTEDG).EQ.0) GO TO 7
      IF (ICLIP(1).EQ.0) GO TO 6
      IPLIST(IP)=INTRSCT
      INTRSCT=IP
      ITHETA=-1
      GO TO 7
    6 CONTINUE
      ITHETA=ITHETA+IDELTA
      GO TO 5
    7 CONTINUE
      IF (IABS(ITHETA).NE.3) GO TO 8
      ZIP=GETZ(IP,XT,YT)
      IF (ZIP.LT.ZT) RETURN
      IF (ZIP.GE.ZSUR) GO TO 8
      ZSUR=ZIP
      ISURNDR=IP
    8 CONTINUE
      IP=JPLINK(IP)
      GO TO 3
    9 CONTINUE
C
C          THINKER
C
      IF (ISURNDR.EQ.0) GO TO 10
C
C          REMOVE ANY HIDDEN POLYGONS
C
      IDLOP=0 $ IP=INTRSCT
   10 CONTINUE
```

```
      IF (IP.EQ.0) GO TO 15
      NXTEDG=IPEDGE(IP)
   11 CONTINUE
      NED=IEDNO(NXTEDG)
      IF (NXTEDGE(NXTEDG).EQ.0) GO TO 13
      IF (IDLIP(1).EQ.0) GO TO 11
      WXY=XX1
      IF ((AZ(NED).AND.1B).NE.0.) WXY=YY1
      ZVAL=AXYZ(NED)/(AZ(NED)*WXY-1.)
      IF (ZVAL.LE.GETZ(ISURNDR,XX1,YY1)) GO TO 13
C
C          POLYGON COMPLETELY HIDDEN
C
      J=IPLIST(IP)
      IF (IOLDP.EQ.0) GO TO 12
      IPLIST(IOLDP)=J
      GO TO 14
   12 CONTINUE
      INTRSCT=J
      GO TO 14
   13 CONTINUE
      IOLDP=IP
   14 CONTINUE
      IP=IPLIST(IP)
      GO TO 10
   15 CONTINUE
      LWINDOW=2
      IF (INTRSCT.NE.0) LWINDOW=1
      RETURN
C
      END
```

```
      LOGICAL FUNCTION LINESX (I,J,XC,YC,ZC)
C
C          DOES LINE "J" CROSS IN FRONT OF LINE "I" ?
C
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   //  IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     4         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     6         AZ(1770),AXY(1770),AXYZ(1770)
C
      JE1=IED1(J) $ JE2=IED2(J)
      LINESX=.FALSE.
      IF (AMIN1(ZS(JE1),ZS(JE2)).GE.AMAX1(ZS(IE1),ZS(IE2)))
     $ RETURN
      D=AX(I)*AY(J)-AX(J)*AY(I)
      IF (ABS(D).LT.1.E-8) RETURN
      XC=-(AXY(I)*AX(J)-AXY(J)*AX(I))/D
      YC=-(AXY(I)*AY(J)-AXY(J)*AY(I))/D
      IF (ABS(AX(I)).LT.ABS(AY(I))) GO TO 1
      IF (((XC-XS(IE1))/AX(I).LE.1.E-5).OR.((XS(IE2)-XC)/
     $ AX(I).LE.1.E-5)
     1) RETURN
      ZC=AXYZ(I)/(AZ(I)*XC-1.)
      GO TO 2
    1 CONTINUE
      IF (((YC-YS(IE1))/AY(I).LE.1.E-5).OR.((YS(IE2)-YC)/
     $ AY(I).LE.1.E-5)
     1) RETURN
      ZC=AXYZ(I)/(AZ(I)*YC-1.)
    2 CONTINUE
      IF (ABS(AX(J)).LT.ABS(AY(J))) GO TO 3
      IF (((XC-XS(JE1))/AX(J).LE.1.E-5).OR.((XS(JE2)-XC)/
     $ AX(J).LE.1.E-5)
     1) RETURN
      DJ=AXYZ(J)/(AZ(J)*XC-1.)
      GO TO 4
    3 CONTINUE
      IF (((YC-YS(JE1))/AY(J).LE.1.E-5).OR.((YS(JE2)-YC)/
     $ AY(J).LE.1.E-5)
     1) RETURN
```

```
      DJ=AXYZ(J)/(AZ(J)*YC-1.)
    4 CONTINUE
      IF (ZC.LT.DJ) RETURN
      LINESX=.TRUE.
      RETURN
C
      END
```

```
      LOGICAL FUNCTION ICANCIT (IPPT,XC,YC,ZC)
C
C          IS THE POINT (XC,YC,ZC) VISIBLE?
C
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     2 XX2,YY2,IDELTA
      COMMON
     1   //  IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     2 NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     3 IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     4         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     5 POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     6 AY(1770),
     6         AZ(1770),AXY(1770),AXYZ(1770)
      DATA EPS/1.E-9/
C
      ICANCIT=.FALSE.
C
C          CONSTRUCT A TINY WINDOW ABOUT THE POINT
C
      WLX=XC-EPS $ WRX=XC+EPS
      WBY=YC-EPS $ WTY=YC+EPS
C
C          START LOOKING DOWN THE POLYGON LIST
C
      IP=IPPT
    1 CONTINUE
      IF (IP.EQ.0) GO TO 5
      IF (GETZ(IP,XC,YC).GT.ZC-1.E-7) GO TO 4
      ITHETA=0
      NXTEDG=IPEDGE(IP)
    2 CONTINUE
      IF (NXTEDG(NXTEDG).EQ.0) GO TO 3
      IF (ICLIP(1).NE.0) RETURN
      ITHETA=ITHETA+IDELTA
      GO TO 2
    3 CONTINUE
      IF (IABS(ITHETA).EQ.8) RETURN
    4 CONTINUE
      IP=IPLIST(IP)
      GO TO 1
    5 CONTINUE
      ICANCIT=.TRUE.
      RETURN
      END
```

```
      FUNCTION GETZ(IP,XP,YP)
C
C          FIND THE Z-VALUE AT (XP,YP) ON SURFACE OF POLYGON
C     $          IP
C
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   //  IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2          XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3          IED1(2760),IED2(2760),IEDLINK(2760),
     4          PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5          IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     6          AZ(1770),AXY(1770),AXYZ(1770)
C
      DATA IPSTORE /0/
C
C          IF POLYGON IS NOT A QUADRALATERAL, FIND Z THE
C     $          EASY WAY
C
      IF (IPENO(IP).EQ.4) GO TO 1
      GETZ=-POLYD(IP)/(POLYA(IP)*XP+POLYB(IP)*YP+POLYC(IP))
      RETURN
C
C          IF POLYGON IP WAS THE LAST QUADRALATERAL DONE,
C     $          SKIP THE
C             PRELIMINARY STUFF
C
    1 CONTINUE
      IF (IP.EQ.IPSTORE) GO TO 2
C
C          PRELIMINARY STUFF
C
      IPSTORE=IP
      JE1=IPEDGE(IP)
      JE2=IEDLINK(JE1) $ JE3=IEDLINK(JE2)
      IP1=IED1(JE1) $ IP2=IED2(JE1) $ IP3=IED2(JE2) $ IP4=
     $ IED2(JE3)
      XO=XS(IP4)*ZS(IP4) $ YO=YS(IP4)*ZS(IP4) $ ZO=ZS(IP4)
      XAO=XS(IP1)*ZS(IP1)-XO
      XCO=XS(IP3)*ZS(IP3)-XO
      DX=-XAO-XCO-XO+XS(IP2)*ZS(IP2)
      YAO=YS(IP1)*ZS(IP1)-YO
      YCO=YS(IP3)*ZS(IP3)-YO
```

```
      DY=-YAD-YCD-YD+YS(IP2)*ZS(IP2)
      ZAD=ZS(IP1)-ZD
      ZCD=ZS(IP3)-ZD
      DZ=-ZAD-ZCD-ZD+ZS(IP2)
      BX=YCD*DZ-ZCD*DY $ BY=ZCD*DX-XCD*DZ $ BZ=XCD*DY-YCD*
     $ DX
      EX=YAD*ZCD-ZAD*YCD-YD*DZ+ZD*DY
      EY=ZAD*XCD-XAD*ZCD-ZD*DX+XD*DZ
      EZ=XAD*YCD-YAD*XCD-XD*DY+YD*DX
      GX=YAD*ZD-ZAD*YD $ GY=ZAD*XD-XAD*ZD $ GZ=XAD*YD-YAD*
     $ XD
C
C          SET UP AND SOLVE QUADRATIC FOR ALPHA
C
    2 CONTINUE
      B=XP*BX+YP*BY+BZ
      E=XP*EX+YP*EY+EZ
      G=XP*GX+YP*GY+GZ
      IF (ABS(B).LT.1.E-8*ABS(E)) GO TO 5
      S=E/(B+B)
      SS=S**2+G/B
      IF (SS.LT.0.) GO TO 6
      ALPHA=S+SQRT(SS)
      IF (ALPHA.GT.1.) ALPHA=S+S-ALPHA
    3 CONTINUE
      XX=(DX-XP*DZ)*ALPHA+XAD-XP*ZAD
      YY=(DY-YP*DZ)*ALPHA+YAD-YP*ZAD
      IF ((ABS(XX)+ABS(YY)).LT.ZD*1.E-11) GO TO 7
      IF (ABS(XX).LT.ABS(YY)) GO TO 4
      GETZ=((BY*ALPHA-EY)*ALPHA-GY)/XX
      RETURN
    4 CONTINUE
      GETZ=-((BX*ALPHA-EX)*ALPHA-GX)/YY
      RETURN
    5 CONTINUE
      ALPHA=-G/E
      GO TO 3
C
C          THE POINT IS NOT ON THE POLYGON'S SURFACE
C
    6 CONTINUE
      GETZ=1.E22
      RETURN
C
C          TWO SOLUTIONS - TAKE THE NEAREST
C
    7 CONTINUE
      Z2=ZD+ALPHA*ZCD $ Z1=Z2+ZAD+ALPHA*DZ
      GETZ=AMIN1(Z1,Z2)
      RETURN
      END
```

```
      FUNCTION NXTEDGE (NXTEDG)
C
C          FIND ENDPOINTS OF NEXT EDGE IN LINKED LIST
C
      COMMON
     1   /CW1/ WLX,WKX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   //  IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     4         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     6         AZ(1770),AXY(1770),AXYZ(1770)
C
      IF (NXTEDG.EQ.0) GO TO 1
      I=IED1(NXTEDG)
      PX=XS(I)
      PY=YS(I)
      PZ=ZS(I)
      I=IED2(NXTEDG)
      QX=XS(I)
      QY=YS(I)
      QZ=ZS(I)
      NXTEDG=IEDLINK(NXTEDG)
      NXTEDGE=-1
      RETURN
C
C          NO MORE EDGES FOR THIS POLYGON
C
    1 CONTINUE
      NXTEDGE=0
      RETURN
C
      END
```

```
      FUNCTION ICLIP (D,IND)
C
C          CLIP SUBPROGRAMS - INCLUDE ICLIP, PUSH, IANGLE,
C     $          AND IWCODE
C
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   //  IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     4         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     6         AZ(1770),AXY(1770),AXYZ(1770),IC1,IC2
C
C          ICLIP SECTION
C
      IDELTA=0
      XX1=QX $ YY1=QY
      IC1=IWCODE(XX1,YY1)
      XX2=PX $ YY2=PY
      IC2=IWCODE(XX2,YY2)
    1 CONTINUE
      IF ((IC1.EQ.0).AND.(IC2.EQ.0)) GO TO 6
      IF ((IC1.AND.IC2).NE.0) GO TO 7
      IF (IC1.NE.0) GO TO 2
      IX=IC1 $ IC1=IC2 $ IC2=IX
      X=XX1 $ XX1=XX2 $ XX2=X
      X=YY1 $ YY1=YY2 $ YY2=X
    2 CONTINUE
      IF ((IC1.AND.1).EQ.0) GO TO 3
      CALL PUSH (0,WLX)
      GO TO 1
    3 CONTINUE
      IF ((IC1.AND.2).EQ.0) GO TO 4
      CALL PUSH (0,WRX)
      GO TO 1
    4 CONTINUE
      IF ((IC1.AND.4).EQ.0) GO TO 5
      CALL PUSH (1,WBY)
      GO TO 1
    5 CONTINUE
      IF ((IC1.AND.8).NE.0) CALL PUSH (1,WTY)
      GO TO 1
```

```
    6 CONTINUE
      ICLIP=-1
      RETURN
C
    7 CONTINUE
      IF (IND.EQ.0) GO TO 10
C
C          ANGLE SECTION
C
      IA1=IANGLE(QX,QY) $ IA2=IANGLE(XX1,YY1) $ IA3=
     $ IANGLE(PX,PY)
      IA1=IA1-IA2 $ IA2=IA2-IA3
      IF (IABS(IA1).LE.3) GO TO 8
      IF (IA1.LT.0) IA1=IA1+16
      IA1=IA1-8
    8 CONTINUE
      IF (IABS(IA2).LE.3) GO TO 9
      IF (IA2.LT.0) IA2=IA2+16
      IA2=IA2-8
    9 CONTINUE
      IDELTA=IA1+IA2
   10 CONTINUE
      ICLIP=0
      RETURN
C
      END
```

```
      SUBROUTINE PUSH (IAL,BT)
C
C          CLIP THE LINE AGAINST THE EXTENDED EDGES OF THE
C     $          WINDOW
C
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   // IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     4         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     6         AZ(1770),AXY(1770),AXYZ(1770),IC1,IC2
C
      IF (IAL.NE.0) GO TO 1
      YY1=(YY2-YY1)*(BT-XX1)/(XX2-XX1)+YY1
      XX1=BT
      GO TO 2
    1 CONTINUE
      XX1=(XX2-XX1)*(BT-YY1)/(YY2-YY1)+XX1
      YY1=BT
    2 CONTINUE
      IC1=IWCODE(XX1,YY1)
      RETURN
C
      END
```

```
      FUNCTION IWCODE (X,Y)
C
C          DETERMINE THE LOCATION CODE FOR POINT (X,Y)
C
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   //  IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     +         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     6         AZ(1770),AXY(1770),AXYZ(1770),IC1,IC2
      IWCODE=0
      IF (X.LT.WLX) IWCODE=IWCODE+1
      IF (X.GT.WRX) IWCODE=IWCODE+2
      IF (Y.LT.WBY) IWCODE=IWCODE+4
      IF (Y.GT.WTY) IWCODE=IWCODE+8
C
      RETURN
      END
```

AD-A040 530 AIR FORCE AERO PROPULSION LAB WRIGHT-PATTERSON AFB OHIO F/G 12/1
CONTOURING AND HIDDEN-LINE ALGORITHMS FOR VECTOR GRAPHIC DISPLA--ETC(U)
JAN 77 J S PETTY, K D MACH
AFAPL-TR-77-3

UNCLASSIFIED NL

3 OF 3
AD
A040530

END

DATE
FILMED
7-77

```
      FUNCTION IANGLE(X,Y)
C
C         DETERMINE THE ANGLE CODE FOR THE POINT (X,Y)
C
      COMMON
     1   /CW1/ WLX,WRX,WBY,WTY,QX,QY,QZ,PX,PY,PZ,XX1,YY1,
     $ XX2,YY2,IDELTA
      COMMON
     1   //  IE1,IE2,NLINES,NPOLYS,IPPTR,XOFF,YOFF,SCALE,
     $ NPTS,
     2         XS(1105),YS(1105),ZS(1105),IEDNO(2760),
     $ IPENO(690),
     3         IED1(2760),IED2(2760),IEDLINK(2760),
     4         PZMIN(690),POLYA(690),POLYB(690),POLYC(690),
     $ POLYD(690),
     5         IPEDGE(690),IPLIST(690),IPLINK(690),AX(1770),
     $ AY(1770),
     6         AZ(1770),AXY(1770),AXYZ(1770),IC1,IC2
C
      IF (X.GE.WLX) GO TO 1
      IANGLE=4
      IF (Y.GT.WTY) IANGLE=3
      IF (Y.LT.WBY) IANGLE=5
      RETURN
    1 CONTINUE
      IF (X.LE.WRX) GO TO 2
      IANGLE=0
      IF (Y.GT.WTY) IANGLE=1
      IF (Y.LT.WBY) IANGLE=7
      RETURN
    2 CONTINUE
      IF (Y.GT.WTY) IANGLE=2
      IF (Y.LT.WBY) IANGLE=6
      RETURN
C
      END
```

SAMPLE DATA FILES

FILE "INPUT"

```
8.0         1.0         1.0
```

FILE "TAPE4"

```
    8    12     6
 0.1671    -0.0362     10.1120
 0.0975     0.1443      9.4692
-0.0483     0.0438      8.3559
 0.0403    -0.1518      8.9987
 0.0347    -0.0314     11.6441
-0.0329     0.1242     11.0013
-0.1708     0.0370      9.8880
-0.0877     0.1297     10.5308
    1     2
    2     3
    3     4
    4     1
    5     6
    6     7
    7     8
    8     5
    1     5
    2     6
    3     7
    4     8
    4
          1
          2
          3
          4
    4
          5
          6
          7
          8
    4
          1
         10
          5
          9
    4
          2
```

```
      11
       6
      10
4
       3
      12
       7
      11
4
       4
       9
       8
      12
```

(FOR CONVENIENCE, HERE WE HAVE VIOLATED
THE FORMAT FOR THE POINT COORDINATES.)

☆ U. S. GOVERNMENT PRINTING OFFICE: 1977 — 757-001/335