NASA TECHNICAL MEMORANDUM

NASA TM X-3206

NASA TM X-3206

A COMPUTER PROGRAM FOR FITTING SMOOTH SURFACES TO AN AIRCRAFT CONFIGURATION AND OTHER THREE-DIMENSIONAL GEOMETRIES

Charlotte B. Craidon

Langley Research Center
Hampton, Va. 23665

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • JUNE 1975

1. Report No. NASA TM X-3206	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle A COMPUTER PROGRAM FOR FITTING SMOOTH SURFACES TO AN AIRCRAFT CONFIGURATION AND OTHER THREE-DIMENSIONAL GEOMETRIES		5. Report Date June 1975
		6. Performing Organization Code
7. Author(s) Charlotte B. Craidon		8. Performing Organization Report No. L-9934
		10. Work Unit No. 501-06-01-13
9. Performing Organization Name and Address NASA Langley Research Center Hampton, Va. 23665		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546		14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

A computer program that uses a three-dimensional geometric technique for fitting a smooth surface to the component parts of an aircraft configuration is presented. The resulting surface equations are useful in performing various kinds of calculations in which a three-dimensional mathematical description is necessary. Program options may be used to compute information for three-view and orthographic projections of the configuration as well as cross-section plots at any orientation through the configuration.

The aircraft (Harris) geometry input section of the program may be easily replaced with a surface point description in a different form so that the program could be of use for any three-dimensional surface equations.

17. Key Words (Suggested by Author(s))		18. Distribution Statement	
Mathematics Surface equations Plotting Computers Design		Unclassified – Unlimited New Subject Category 61	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 112	22. Price* $5.25

*For sale by the National Technical Information Service, Springfield, Virginia 22151

PAGE MISSING FROM AVAILABLE VERSION

CONTENTS

Page

A COMPUTER PROGRAM FOR FITTING SMOOTH SURFACES TO AN AIRCRAFT CONFIGURATION AND OTHER THREE-DIMENSIONAL GEOMETRIES

Charlotte B. Craidon
Langley Research Center

SUMMARY

A digital computer program (D3400) that uses a three-dimensional geometric technique for fitting a smooth surface to the component parts of an aircraft configuration is presented. The resulting surface equations are useful in performing various kinds of calculations in which a three-dimensional mathematical description is necessary.

Program options may be used to compute information for three-view and orthographic projections of the configuration as well as cross-section plots at any orientation through the configuration. These operations were implemented to validate the usefulness and versatility of the surface equations. Output from this program has been used to drive Calcomp, Gerber, and Varian plotters and for on-line display on a cathode-ray-tube device.

The aircraft (Harris) geometry input section of the program may be easily replaced with a surface point description in a different form. Therefore, the program could be of use for any three-dimensional surface equations.

At the present time, the program can only be applied to relatively smooth surfaces; that is, there must be no abrupt changes in curvature. This deficiency is overcome to some degree by using the airplane component parts or, stated another way, by using a collection of surfaces.

INTRODUCTION

Aerospace vehicles, automobiles, and ships are examples of objects which require smooth curved surfaces to establish their exterior shapes. An integral part of the design, development, and manufacture of these objects is the construction of surface models which can be analyzed for their interaction with the environment in which they are to operate. The most useful models from the point of view of versatility and exactness of definition are mathematical models.

The simplest mathematical model of a three-dimensional surface is a set of planes which are defined by points and approximate the curved surface. In order to obtain an accurate definition using a discrete set of planes, a large number of points on the surface must be defined. Preparing and manipulating the data which yield a planar approximation of a surface is laborious if an accurate definition is desired. Another difficulty with planar approximation occurs when cross sections or contours of curved surfaces are necessary. Planar approximation yields a very rough cross section or contour unless an extremely large number of points are used to define the surface.

In recent years a high-order accurate method for mathematical modeling of smooth three-dimensional surfaces has been developed. (See ref. 1.) This method is based on approximating an arbitrary surface by piecing together surface "patches." Each patch is defined by four boundary curves and is bicubic with respect to two parametric variables in the interior. A patch is therefore defined by four corner points, the first derivative of the corner points with respect to two parametric variables, and the cross derivatives of the corner points with respect to the parametric variables. The patch-equation definition yields a smooth representation of an arbitrary surface with relatively few points of definition. It also yields smooth approximations to cross sections and contour plots.

The purpose of this report is to describe a computer program which is based on the use of sets of bicubic patches to define a relatively smooth surface. (See ref. 2.) In particular the data description for the program is oriented toward aircraft configurations. This allows the organization of data for the various components to be identical with the data used for several standardized aerodynamic analysis computer programs. (See ref. 3.) The aircraft data description has become known as the Harris Wave Drag geometry.

The program can also be used to model arbitrary three-dimensional objects by using an alternate data input format. The data-point input to the program is not required to be equally spaced in any coordinate variable. However, there are some restrictions on the number of points in the descriptive lines for the same surface.

A three-dimensional parametric cubic spline technique is used to curve fit the input data points roughly describing the surface. From the curve fit, the derivatives of the surface patches with respect to the parametric variable at the corner points are established. The cross derivatives of the patch representations with respect to the parametric variables are not used in this program. The values of the corner points and the derivatives at the corner points constitute the information necessary to solve the patch equation. In this way 36 pieces of information are required to define a patch; however, only 12 pieces of information must be supplied as input. The remainder is determined from the spline fit. Appendix A describes the cubic spline fit technique and appendix B, the patch equations.

The entire aircraft geometry or other three-dimensional object is converted into surface patch form. Each patch definition is identical in matrix structure which simplifies the organization of the computer program and data base. (See appendix C.) All computations, such as rotations, are performed directly on patch equations rather than on interpolated x-, y-, and z-coordinates.

The computer program has the ability to display the orthographic projections of the input description of the surface and the enriched description of the surface based on the patch definition. (See appendix D.) The desired angles of orientation for viewing the surface are inputs to the computer program and the transformation based on these angles is applied to the patch definition. An option of the program tests the derivatives normal to the surface and deletes those points from the orthographic projection which are facing away from the observer. This option gives the program a partial hidden-line capability which works very well for convex closed surfaces. Figures 1 to 3 are examples of orthographic projections.

The program is also capable of producing plots of the surface coordinates of a cross section at any desired orientation. (See appendix E.) Figures 4 to 6 are examples of cross-section plots. The cross-section calculations consist of the simultaneous solution of the patch equations and the equation of a plane. The plane is defined by three points which are input to the program.

SYMBOLS

A,B,C,D	parametric cubic spline coefficients
a,b,c,d	plane equation coefficients
$\bar{B}$	boundary matrix
L	chord length
M	blending function matrix
$\bar{M}$	Mach number
N	surface normal vector
P	a vector whose components are functions of t
P_1,P_2,P_3	points used to define a plane
S	component of surface patch equation

T	diagonal vectors
t	independent variable in cubic equation
u,w	independent variables in patch equation
V	a vector whose components are functions of u and w
v	unit vector
x,y,z	coordinates of a point
θ	pitch angle
$\bar{\theta}$	roll angle for Mach plane orientation
ϕ	roll angle
ψ	yaw angle

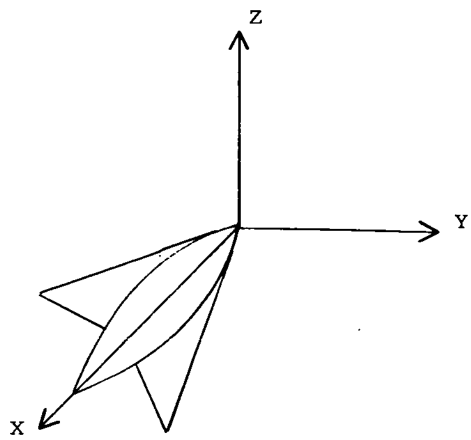
PROBLEM DESCRIPTION AND METHOD OF SOLUTION

The numerical model of the input airplane configuration is assumed to be symmetrical about the XZ-plane (positive Y-side) and may include any combination of components: wing, body, pods, fins, and canards. The wing is made up of airfoil sections, the fuselage is defined by either circular or arbitrary sections, the pods are defined similar to the fuselage, and fins and canards are defined similar to the wings.

The configuration is usually positioned with its nose at the origin and with the length of the body stretching in the positive X-direction.

The coordinate system used for this program is a right-handed Cartesian coordinate system as illustrated in sketch (a).

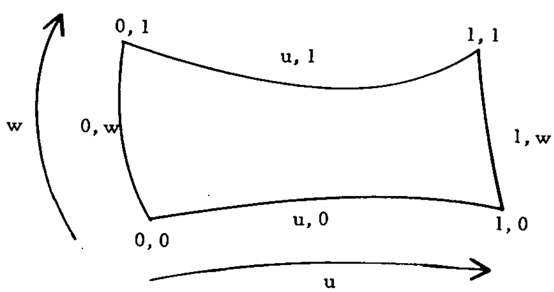
Since the modeling technique expects to approximate a smooth surface, sufficient input data points with no abrupt changes in curvature should be supplied. A three-dimensional parametric cubic spline technique is used for the patch boundary-curve definitions in which the coordinates are expressed as cubic functions of one variable. A series of adjacent polynomial segments between each given point is used to represent the curve. The length of each segment is used as the parameter and later normalized to 1. Linear segments are used when a line consists of less than three points.



Sketch (a)

The cubic spline curve fitting subroutine uses a technique from a paper by Timothy Johnson of Massachusetts Institute of Technology and is summarized in appendix A.

The x-, y-, and z-coordinates of a surface patch are each single-valued cubic functions of two parameters, u and w. The coefficients of these cubics are expressed in terms of end points and partial and cross derivatives with respect to the u and w parameters. The result is a parametric bicubic representation of three-dimensional surfaces. Sketch (b) shows a typical patch.



Sketch (b)

Each patch equation requires 48 pieces of information.

A summary of the bicubic surface patch equation form is given in appendix B and the storage file description of the surface patches is given in appendix C.

The orthographic projections illustrated in this report are created by applying the three-dimensional rotation equations directly to the patch equations describing the body surface for plotting the aircraft at any desired viewing angle. The rotated patch equations are projected into the two-dimensional patch form of the paper plane. An enriched surface may be obtained from the rotated and projected patch equations by holding u constant and varying w from 0.0 to 1.0, then by holding w constant and varying u from 0.0 to 1.0.

The orthographic plotting routine also includes a hidden-line option where the normal vectors are computed from the rotated and projected form of the patches. A positive normal vector indicates that the point is visible and a negative normal vector indicates that the vector points away from the viewer and thus is not visible. The method used for the orthographic projections is given in appendix D.

Another routine has been written to compute and plot the surface coordinates of a cross section through the body at any desired orientation. The calculations consist of the simultaneous solution of the patch equations and the equation of a plane. The method is described in appendix E.

PROGRAM DESCRIPTION

LABELED COMMON

The following list contains the FORTRAN variables appearing in labeled COMMON.

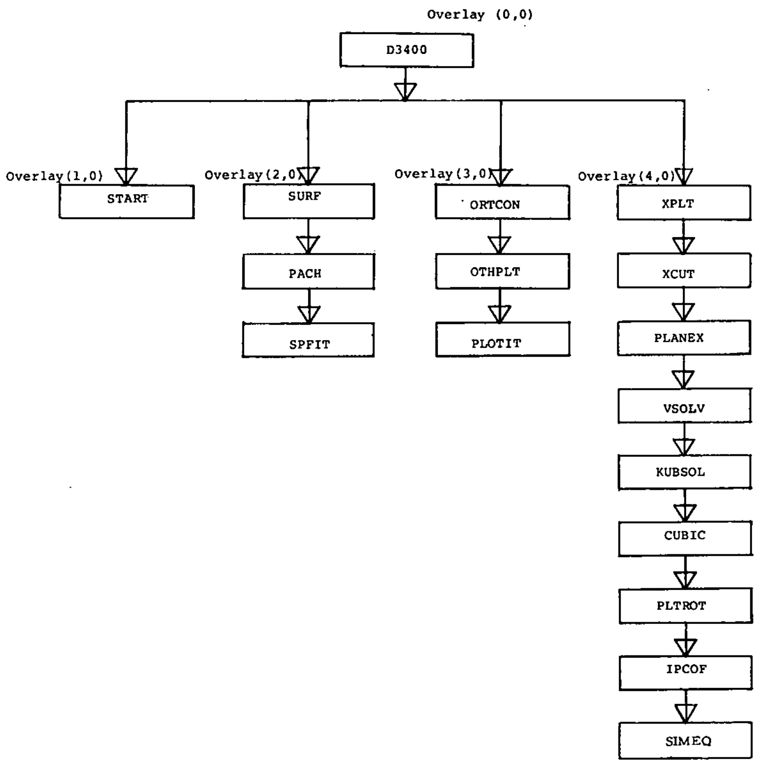
COMMON label	FORTRAN variable	Description
PATPLT		
	XMIN	Minimum x-value for plotting
	XMAX	Maximum x-value for plotting
	YMIN	Minimum y-value for plotting
	YMAX	Maximum y-value for plotting
	ZMIN	Minimum z-value for plotting
	ZMAX	Maximum z-value for plotting
	NOBJ	Total number of objects (or components) which could form an aircraft configuration
THREED		
	ABCDE(8)	Identification
	HORZ	X-axis of the paper plane
	VERT	Y-axis of the paper plane
	TEST1	Hidden-line option flag

COMMON label	FORTRAN variable	Description
	PHI	Roll angle
	THETA	Pitch angle
	PSI	Yaw angle
	PLOTSZ	Plot frame size
	TYPE	Type of plot desired
	NOU	Number of internal points for each patch in u-direction
	NOW	Number of internal points for each patch in w-direction
	ISIDE	Flag for plotting object or object and its mirror image
	KODE	Flag for plot-option branch
XSECT		
	ABCDE(8)	Identification
	PPL1(3)	Origin of cross-section plot and one point in three-point-plane definition
	PPL2(3)	Second point in three-point-plane definition; or X-intercept, roll angle, and Mach number in plane-angle definition
	PPL3(3)	Third point in three-point-plane definition
	PLOTSZ	Scale factor
	HPAGE	Horizontal paper origin
	VPAGE	Twice the vertical paper origin
	INP	Specifies kind of plane input
	NOU	Number of points to interpolate for each patch in u-direction
	NOW	Number of points to interpolate for each patch in w-direction
	ISIDE	Flag for examining object or object and its mirror image

COMMON label	FORTRAN variable	Description
	IPRIN	Print code
	KODE	Flag for plot-option branch
	XSTAT	X-intercept for Mach plane
	THETR	Roll angle for Mach plane orientation
	XMACH	Mach number

OVERLAY ARRANGEMENT

Program D3400 is set up in the overlay mode and sketch (c) illustrates the overlay arrangement.



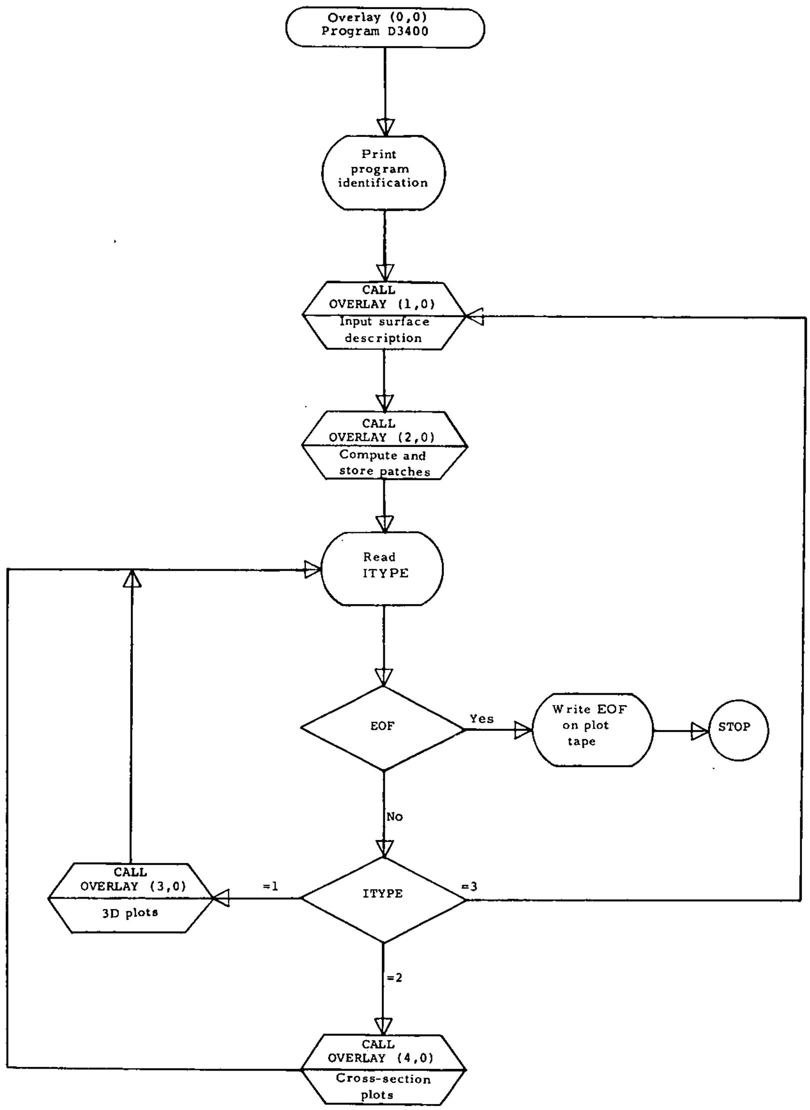
Sketch (c)

The control program (0,0) calls in the other parts of the program as they are needed. The initialization overlay (1,0) reads cards defining the body surface, converts the input to actual units, and temporarily stores the surface description as a series of lines. Cubic spline fairing is performed on curved lines defining the body surface in overlay (2,0) and surface patch equations are constructed and temporarily stored. Overlay (3,0) generates orthographic plot information using the patch equations, and overlay (4,0) generates cross-sectional plot information using the patch equations.

PROGRAMS AND SUBPROGRAMS

Program D3400

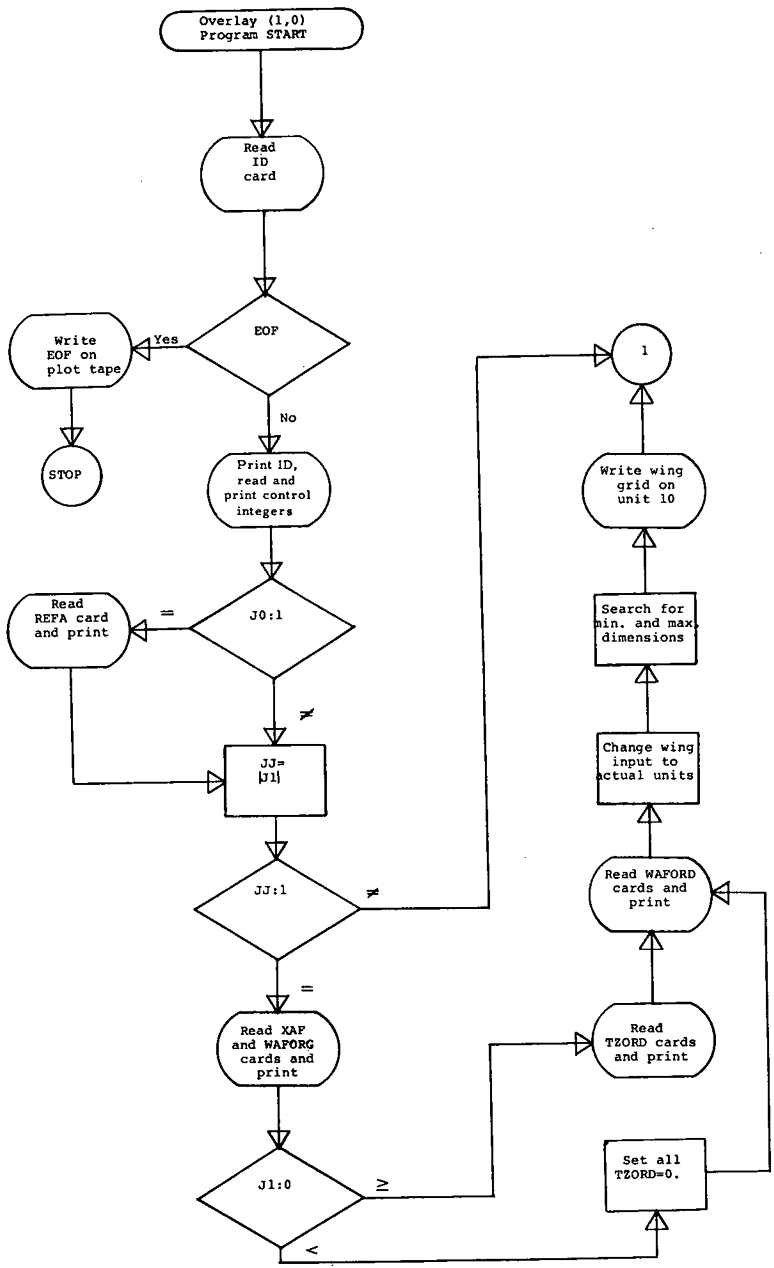
Program D3400 (overlay (0,0)) is the control program. This program initiates loading and execution of other parts of the program as required. The flow chart and the FORTRAN statements for this overlay are as follows:

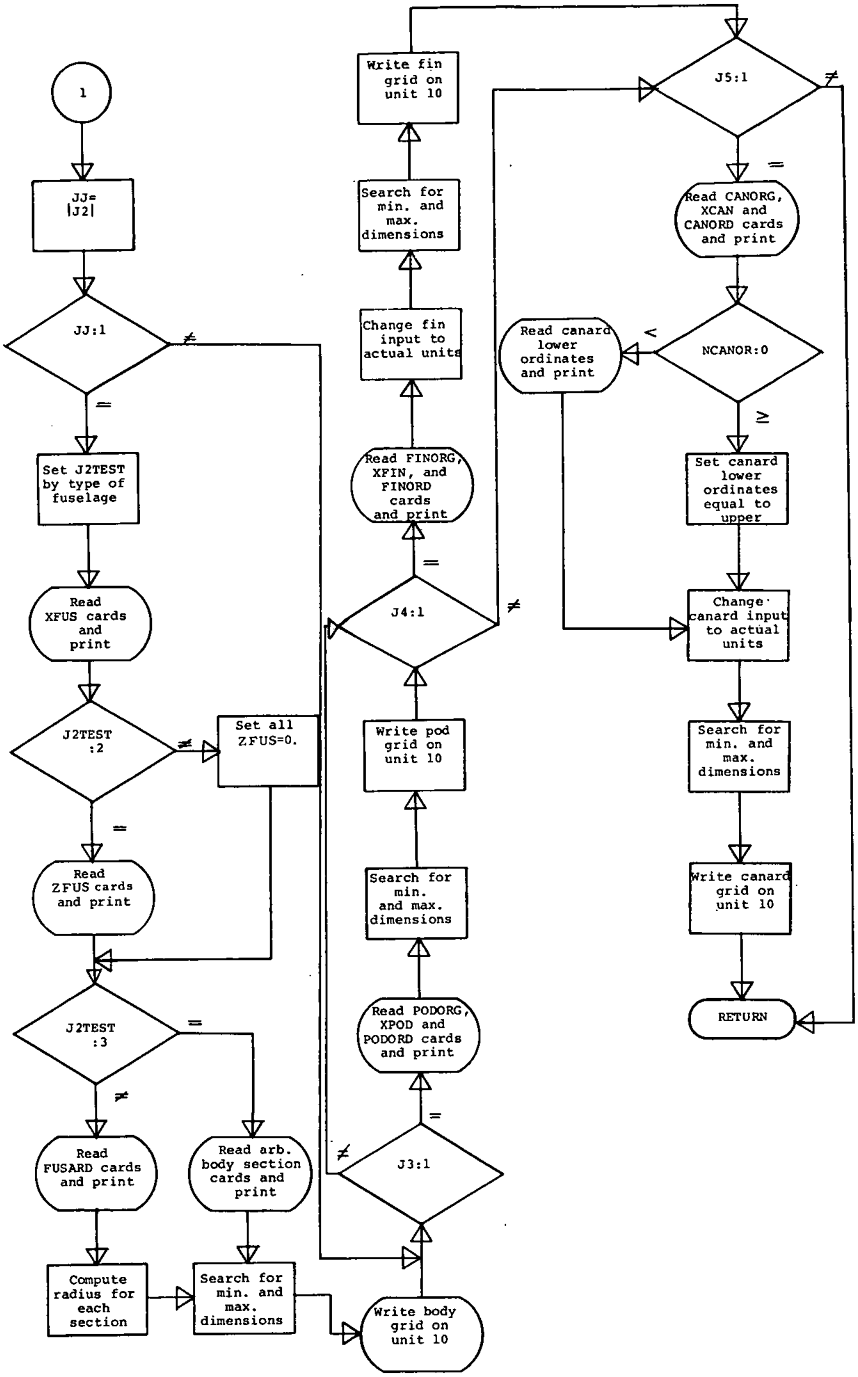


```
      OVERLAY(CBC,0,0)
      PROGRAM D3400(INPUT=201,OUTPUT=201,TAPE10=201,
     1TAPE5=INPUT,TAPE6=OUTPUT,TAPE7)
C
C          D3400 (SPADE) - SURFACE PATCH DEFINITION EQUATIONS
C              (CONVERTS A SURFACE POINT DESCRIPTION
C              TO THREE DIMENSIONAL SURFACE PATCH EQUATIONS)
C          PROGRAMER - CHARLOTTE B. CRAIDON
C
      COMMON/PATPLT/
     1XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,NOBJ
C
C
      CBC=3LCBC
      RECALL=6HRECALL
      CALL PSEUDO
      WRITE (6,20)
20    FORMAT (1H110X,24HPROGRAM D3400 (SPADE) - ,34HSURFACE PATCH DEFINI
     1TION EQUATIONS///)
C
C          INPUT SURFACE POINT DESCRIPTION AND PROCESS
C          FOR TAPE 10 AND FOR LABELED COMMON PATPLT
C
30    CONTINUE
      CALL OVERLAY (CBC,1,0,0)
C
C          COMPUTE AND STORE PATCH EQUATIONS
C
      CALL OVERLAY (CBC,2,0,0)
C
40    READ (5,50) ITYPE
50    FORMAT (I4)
      IF (ENDFILE 5) 90,60
60    GO TO (70,80,30), ITYPE
C
C          THREE DIMENSIONAL PLOTS
C
70    CONTINUE
      CALL OVERLAY (CBC,3,0,0)
      GO TO 40
C
C          CROSS SECTION PLOTS
C
80    CALL OVERLAY (CBC,4,0,0)
      GO TO 40
90    CALL NFRAME $ CALL CALPLT(0.,0.,999) $ STOP
C
C          END OF D3400
C
      END
```

Program START

Program START (overlay (1,0)) reads the configuration description cards and prints them, changes the input values to actual units where necessary, computes the minimum and maximum dimensions of the given configuration, and temporarily stores the airplane description as a series of lines. The flow chart and the FORTRAN statements for this overlay are as follows:




1
JJ= |J2|
JJ:1
=
≠
Set J2TEST by type of fuselage
Read XFUS cards and print
J2TEST :2
≠
Set all ZFUS=0.
=
Read ZFUS cards and print
J2TEST :3
=
≠
Read FUSARD cards and print
Read arb. body section cards and print
Compute radius for each section
Search for min. and max. dimensions
Write body grid on unit 10
J3:1
=
≠
Read PODORG, XPOD and PODORD cards and print
Search for min. and max. dimensions
Write pod grid on unit 10
J4:1
=
≠
Read FINORG, XFIN, and FINORD cards and print
Change fin input to actual units
Search for min. and max. dimensions
Write fin grid on unit 10
J5:1
=
≠
Read CANORG, XCAN and CANORD cards and print
NCANOR:0
<
≥
Read canard lower ordinates and print
Set canard lower ordinates equal to upper
Change canard input to actual units
Search for min. and max. dimensions
Write canard grid on unit 10
RETURN

```
      OVERLAY(CBC,1,0)
      PROGRAM START
C
C          INPUTS AIRCRAFT SURFACE DESCRIPTION,
C          FORMS INTO DESCRIPTIVE LINES WRITTEN ON TAPE 10,
C          AND COMPUTES MINIMUMS AND MAXIMUMS
C
      COMMON/PATPLT/
     1XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,NOBJ
C
      DIMENSION BLOCK (7500)
C
      DIMENSION XAF(30),WAFORG(20,4),WAFORD(20,3,30),TZORD(20,30)
      EQUIVALENCE (BLOCK,XAF),(BLOCK(31),WAFORG),
     1(BLOCK(111),WAFORD),(BLOCK(1911),TZORD)
C
      DIMENSION XFUS(30,4),ZFUS(30,4),FUSARD(30,4),FUSRAD(30,4),
     1SFUS(30,30,8)
      EQUIVALENCE (BLOCK,XFUS),(BLOCK(121),ZFUS),(BLOCK(241),FUSARD),
     1(BLOCK(361),FUSRAD),(BLOCK(241),SFUS)
C
      DIMENSION PODORG(9,3),XPOD(9,30),PODORD(9,30),XPOD1(9,30)
      EQUIVALENCE (BLOCK,PODORG),(BLOCK(28),XPOD),(BLOCK(298),PODORD),
     1(BLOCK(568),XPOD1)
C
      DIMENSION FINORG(6,2,4),XFIN(6,10),FINORD(6,2,10),
     1FINX2(6,2,10),FINX3(6,2,10)
      EQUIVALENCE (BLOCK,FINORG),(BLOCK(49),XFIN),(BLOCK(109),FINORD),
     1(BLOCK(229),FINX2),(BLOCK(349),FINX3)
C
      DIMENSION CANORG(2,2,4),XCAN(2,10),CANORD(2,2,10),
     1CANOR1(2,2,10),CANORX(2,2,10)
      EQUIVALENCE (BLOCK,CANORG),(BLOCK(17),XCAN),(BLOCK(37),CANORD),
     1(BLOCK(77),CANOR1),(BLOCK(117),CANORX)
C
      DIMENSION ABC(8),ABCD(8),ANSIN(30),ANCOS(30),NAME(2)
      DIMENSION NRADX(4),NFORX(4)
      DIMENSION ALRT (31,3)
      DATA PI/3.14159265/
C
      REWIND 10
10    FORMAT (8A10)
20    FORMAT (1X8A10)
30    FORMAT (10F7.0)
C
C          READ ID CARD AND CARD OF CONTROL INTEGERS
C
      READ (5,10) ABC
      IF (ENDFILE 5) 35,40
35    CALL NFRAME $ CALL CALPLT(0.,0.,999) $ STOP
40    CONTINUE
      WRITE (6,50) ABC
50    FORMAT (23X,34HAIRCRAFT CONFIGURATION DESCRIPTION//1X8A10/)
      READ (5,10) ABCD
      WRITE (6,60) ABCD
60    FORMAT (1X8A10/)
      DECODE (72,70,ABCD) J0,J1,J2,J3,J4,J5,J6,NWAF,NWAFOR,NFUS,(NRADX(I
     1),NFORX(I),I=1,4),NP,NPODOR,NF,NFINOR,NCAN,NCANOR
```

```
70    FORMAT (24I3)
C
      WRITE (10) ABC
      NOBJ=0
C
C         REFERENCE AREA
C
      IF (J0.NE.1) GO TO 80
      READ (5,10) ABCD
      WRITE (6,20) ABCD
C
C         WING
C
80    JJ=IABS(J1)
      IF (JJ.NE.1) GO TO 290
      N=IABS(NWAFOR)
      NREC=(N+9)/10
      I1=-9
      I2=0
      DO 90 NN=1,NREC
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      I1=I1+10
      I2=I2+10
      DECODE (70,30,ABCD) (XAF(I),I=I1,I2)
90    CONTINUE
      DO 100 I=1,NWAF
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      DECODE (28,30,ABCD) (WAFORG(I,J),J=1,4)
100   CONTINUE
      IF (J1.LT.0) GO TO 130
      DO 120 NN=1,NWAF
      I1=-9
      I2=0
      DO 110 N1=1,NREC
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      I1=I1+10
      I2=I2+10
      DECODE (70,30,ABCD) (TZORD(NN,I),I=I1,I2)
110   CONTINUE
120   CONTINUE
      GO TO 150
130   DO 140 I=1,NWAF
      DO 140 K=1,N
140   TZORD(I,K)=0.
150   L=1
      IF (NWAFOR.LT.0) L=2
      DO 170 NN=1,NWAF
      DO 170 K=1,L
      I1=-9
      I2=0
      DO 160 N1=1,NREC
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      I1=I1+10
      I2=I2+10
```

```
      DECODE (70,30,ABCD) (WAFORD(NN,K,I),I=I1,I2)
  160 CONTINUE
  170 CONTINUE
      IF (NWAFOR.LT.0) GO TO 190
      DO 180 NN=1,NWAF
      DO 180 K=1,N
  180 WAFORD(NN,2,K)=WAFORD(NN,1,K)
  190 CONTINUE
      NWAFOR=IABS(NWAFOR)
      NW=NWAFOR
      J1=IABS(J1)
C
C         CHANGE TO ACTUAL UNITS, COMPUTE MINIMUMS AND MAXIMUMS
C
      DO 210 I=1,NWAF
      E=.01*WAFORG(I,4)
      E3=WAFORG(I,3)
      DO 200 J=1,NWAFOR
      WAFORD(I,1,J)=E*WAFORD(I,1,J)+E3+TZORD(I,J)
      WAFORD(I,2,J)=-E*WAFORD(I,2,J)+E3+TZORD(I,J)
  200 WAFORD(I,3,J)=WAFORG(I,1)+E*XAF(J)
  210 CONTINUE
      XMIN=XMAX=WAFORG(1,1)
      YMAX=WAFORG(1,2)
      YMIN=WAFORG(1,2)
      ZMIN=ZMAX=WAFORD(1,1,1)
      DO 230 N=1,NWAF
      XMAX=AMAX1(XMAX,WAFORD(N,3,NW))
      XMIN=AMIN1(XMIN,WAFORD(N,3,1))
      YMAX=AMAX1(YMAX,WAFORG(N,2))
      YMIN=AMIN1(YMIN,WAFORG(N,2))
      DO 220 NN=1,NW
      ZMAX=AMAX1(ZMAX,WAFORD(N,1,NN))
      ZMIN=AMIN1(ZMIN,WAFORD(N,2,NN))
  220 CONTINUE
  230 CONTINUE
C
C         WRITE LINE TAPE
C
      NN=2
      NCOMP=1$NAME(1)=10HWING       $NAME(2)=10H
      WRITE (10) NN,NCOMP,NAME,NN,NN
      NOBJ=NOBJ+1
      DO 280 I=1,2
      WRITE (10) NWAF,NWAFOR,NN,NN,NN
      KKK=(I-1)*(NWAFOR+1)
      KK=(-1)**(I+1)
C
C         SETUP SPANWISE LINES
C
      DO 250 K=1,NWAFOR
      NN=KKK+KK*K
      DO 240 N=1,NWAF
      ALRT(N,1)=WAFORD(N,3,NN)
      ALRT(N,2)=WAFORG(N,2)
      ALRT(N,3)=WAFORD(N,I,NN)
  240 CONTINUE
      WRITE (10) ((ALRT(N,N3),N=1,NWAF),N3=1,3)
  250 CONTINUE
```

```
C
C          SETUP STREAMWISE LINES
C
      DO 270 NN=1,NWAF
      DO 260 K=1,NWAFOR
      N=KKK+KK*K
      ALRT(K,1)=WAFORD(NN,3,N)
      ALRT(K,2)=WAFORG(NN,2)
      ALRT(K,3)=WAFORD(NN,I,N)
260   CONTINUE
      WRITE (10) ((ALRT(N,N3),N=1,NW),N3=1,3)
270   CONTINUE
280   CONTINUE
C
C          FUSELAGE
C
290   JJ=IABS(J2)
      IF (JJ.NE.1) GO TO 590
      J2TEST=3
      IF (J2.EQ.-1.AND.J6.EQ.-1) J2TEST=1
      IF (J2.EQ.-1.AND.J6.EQ.0) J2TEST=2
      IF (J6.EQ.1) J2TEST=1
      J2=1
      DO 410 NFU=1,NFUS
      NRAD=NRADX(NFU)
      NFUSOR=NFORX(NFU)
      N=NFUSOR
      NREC=(N+9)/10
      I1=-9
      I2=0
      DO 300 N1=1,NREC
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      I1=I1+10
      I2=I2+10
      DECODE (70,30,ABCD) (XFUS(I,NFU),I=I1,I2)
300   CONTINUE
      IF (J2TEST.NE.2) GO TO 320
      I1=-9
      I2=0
      DO 310 N1=1,NREC
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      I1=I1+10
      I2=I2+10
      DECODE (70,30,ABCD) (ZFUS(I,NFU),I=I1,I2)
310   CONTINUE
      GO TO 340
320   DO 330 I=1,N
330   ZFUS(I,NFU)=0.
340   IF (J2TEST.NE.3) GO TO 380
      NCARD=(NRAD+9)/10
      DO 370 LN=1,N
      DO 360 K=1,2
      KK=K+(NFU-1)*2
      II=10
      I1=-9
      I2=0
```

```
      DO 350 NN=1,NCARD
      IF (NN.EQ.NCARD) II=MOD(NRAD,10)
      IF (II.EQ.0) II=10
      I1=I1+10
      I2=I2+II
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      DECODE (70,30,ABCD) (SFUS(I,LN,KK),I=I1,I2)
350   CONTINUE
360   CONTINUE
370   CONTINUE
      GO TO 410
380   I1=-9
      I2=0
      DO 390 N1=1,NREC
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      I1=I1+10
      I2=I2+10
      DECODE (70,30,ABCD) (FUSARD(I,NFU),I=I1,I2)
390   CONTINUE
      DO 400 I=1,N
400   FUSRAD(I,NFU)=SQRT(FUSARD(I,NFU)/PI)
C
410   CONTINUE
C
C          FUSELAGE MIN AND MAX
C
      IF (J1.NE.0) GO TO 430
      XMIN=XFUS(1,1)
      XMAX=XFUS(1,1)
      IF (J2TEST.EQ.3) GO TO 420
      YMIN=FUSRAD(1,1)
      YMAX=FUSRAD(1,1)
      ZMIN=-FUSRAD(1,1)+ZFUS(1,1)
      ZMAX=FUSRAD(1,1)+ZFUS(1,1)
      GO TO 430
420   YMAX=SFUS(1,1,1)
      YMIN=SFUS(1,1,1)
      ZMIN=SFUS(1,1,2)
      ZMAX=SFUS(1,1,2)
430   DO 470 N=1,NFUS
      NRAD=NRADX(N)
      NFUSOR=NFORX(N)
      XMIN=AMIN1(XMIN,XFUS(1,N))
      XMAX=AMAX1(XMAX,XFUS(NFUSOR,N))
      DO 460 NN=1,NFUSOR
      IF (J2TEST.EQ.3) GO TO 440
      YMAX=AMAX1(YMAX,FUSRAD(NN,N))
      YMIN=AMIN1(YMIN,FUSRAD(NN,N))
      ZMAX=AMAX1(ZMAX,FUSRAD(NN,N)+ZFUS(NN,N))
      ZMIN=AMIN1(ZMIN,-FUSRAD(NN,N)+ZFUS(NN,N))
      GO TO 460
440   KK=1+(N-1)*2
      DO 450 NR=1,NRAD
      YMIN=AMIN1(YMIN,SFUS(NR,NN,KK))
      YMAX=AMAX1(YMAX,SFUS(NR,NN,KK))
      ZMIN=AMIN1(ZMIN,SFUS(NR,NN,KK+1))
450   ZMAX=AMAX1(ZMAX,SFUS(NR,NN,KK+1))
```

```
460   CONTINUE
470   CONTINUE
C
C          WRITE LINE TAPE
C
      JJN=2$N1=1$NAME(1)=10HFUSELAGE   $NAME(2)=10H
      NOBJ=NOBJ+NFUS
      DO 580 NFU=1,NFUS
      NRAD=NRADX(NFU)
      NFUSOR=NFORX(NFU)
      WRITE (10) N1,JJN,NAME,N1,N1
      WRITE (10) NFUSOR,NRAD,N1,N1,N1
      NAN=NRAD
      IF (J2TEST.EQ.3) GO TO 490
      FANG=(NRAD-1)*2
      DELE=6.2831853/FANG
      DO 480 N=1,NAN
      E=N-1
      ANSIN(N)=SIN(E*DELE+4.712389)
480   ANCOS(N)=COS(E*DELE+4.712389)
490   CONTINUE
      KK=1+(NFU-1)*2
C
C          SETUP STREAMWISE LINES
C
      DO 530 N=1,NAN
      DO 520 NN=1,NFUSOR
      ALRT(NN,1)=XFUS(NN,NFU)
      IF (J2TEST.EQ.3) GO TO 500
      ALRT(NN,2)=FUSRAD(NN,NFU)*ANCOS(N)
      ALRT(NN,3)=FUSRAD(NN,NFU)*ANSIN(N)+ZFUS(NN,NFU)
      GO TO 510
500   ALRT(NN,2)=SFUS(N,NN,KK)
      ALRT(NN,3)=SFUS(N,NN,KK+1)
510   CONTINUE
520   CONTINUE
      WRITE (10) ((ALRT(N,N3),N=1,NFUSOR),N3=1,3)
530   CONTINUE
C
C          SETUP LINES AROUND BODY
C
      DO 570 N=1,NFUSOR
      DO 560 NN=1,NAN
      ALRT(NN,1)=XFUS(N,NFU)
      IF (J2TEST.EQ.3) GO TO 540
      ALRT(NN,2)=FUSRAD(N,NFU)*ANCOS(NN)
      ALRT(NN,3)=FUSRAD(N,NFU)*ANSIN(NN)+ZFUS(N,NFU)
      GO TO 550
540   ALRT(NN,2)=SFUS(NN,N,KK)
      ALRT(NN,3)=SFUS(NN,N,KK+1)
550   CONTINUE
560   CONTINUE
      WRITE (10) ((ALRT(N,N3),N=1,NAN),N3=1,3)
570   CONTINUE
580   CONTINUE
C
C          NACELLES
C
```

```
590   CONTINUE
      IF (J3.NE.1) GO TO 730
      N=NPODOR
      NREC=(N+9)/10
      DO 620 NN=1,NP
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      DECODE (21,30,ABCD) (PODORG(NN,I),I=1,3)
      I1=-9
      I2=0
      DO 600 N1=1,NREC
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      I1=I1+10
      I2=I2+10
      DECODE (70,30,ABCD) (XPOD(NN,I),I=I1,I2)
600   CONTINUE
      I1=-9
      I2=0
      DO 610 N1=1,NREC
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      I1=I1+10
      I2=I2+10
      DECODE (70,30,ABCD) (PODORD(NN,I),I=I1,I2)
610   CONTINUE
620   CONTINUE
C
C          COMPUTE ACTUAL X,MINIMUM,MAXIMUM
C
      DO 630 N=1,NP
      DO 630 NN=1,NPODOR
630   XPOD1(N,NN)=XPOD(N,NN)+PODORG(N,1)
      IF (J1.NE.0.OR.J2.NE.0) GO TO 640
      XMIN=XPOD1(1,1)
      XMAX=XPOD1(1,NPODOR)
      YMIN=PODORG(1,2)+PODORD(1,1)
      YMAX=PODORG(1,2)+PODORD(1,1)
      ZMIN=PODORG(1,3)-PODORD(1,1)
      ZMAX=PODORG(1,3)+PODORD(1,1)
640   DO 660 N=1,NP
      XMIN=AMIN1(XMIN,XPOD1(N,1))
      XMAX=AMAX1(XMAX,XPOD1(N,NPODOR))
      DO 650 NN=1,NPODOR
      YMIN=AMIN1(YMIN,PODORD(N,NN)+PODORG(N,2))
      YMAX=AMAX1(YMAX,PODORD(N,NN)+PODORG(N,2))
      ZMIN=AMIN1(ZMIN,PODORG(N,3)-PODORD(N,NN))
650   ZMAX=AMAX1(ZMAX,PODORG(N,3)+PODORD(N,NN))
660   CONTINUE
      DATA NAN2/4/,PIPL/4.712389/
      NANG1=NAN2+1
      NANG2=2*NAN2+1
      FANG=NAN2*2
      DELE=6.2831853/FANG
      DO 670 N=1,NANG2
      E=N-1
      EE=E*DELE
      ANSIN(N)=SIN(EE+PIPL)
670   ANCOS(N)=COS(EE+PIPL)
```

```
C
C          WRITE LINE TAPE
C
      JJN=3$NAME(1)=10HPODS       $NAME(2)=10H
      NOBJ=NOBJ+NP
      DO 720 NP1=1,NP
      I=2
      IF (PODORG(NP1,2).EQ.0.) I=1
      WRITE (10) 1,JJN,NAME,I,I
      DO 720 I=1,2
      IF (I.EQ.2.AND.PODORG(NP1,2).EQ.0.) GO TO 720
      WRITE (10) NPODOR,NANG1,I,I,I
C
C          SETUP STREAMWISE LINES
C
      DO 690 K=1,NANG1
      NN=(I-1)*NAN2+K
      DO 680 N=1,NPODOR
      ALRT(N,1)=XPOD(NP1,N)+PODORG(NP1,1)
      ALRT(N,2)=PODORD(NP1,N)*ANCOS(NN)+PODORG(NP1,2)
      ALRT(N,3)=PODORD(NP1,N)*ANSIN(NN)+PODORG(NP1,3)
680   CONTINUE
      WRITE (10) ((ALRT(N,N3),N=1,NPODOR),N3=1,3)
690   CONTINUE
C
C          SETUP LINES AROUND PODS
C
      DO 710 N=1,NPODOR
      DO 700 K=1,NANG1
      NN=(I-1)*NAN2+K
      ALRT(K,1)=XPOD(NP1,N)+PODORG(NP1,1)
      ALRT(K,2)=PODORD(NP1,N)*ANCOS(NN)+PODORG(NP1,2)
      ALRT(K,3)=PODORD(NP1,N)*ANSIN(NN)+PODORG(NP1,3)
700   CONTINUE
      WRITE (10) ((ALRT(K,N3),K=1,NANG1),N3=1,3)
710   CONTINUE
720   CONTINUE
C
C          FINS
C
730   CONTINUE
      IF (J4.NE.1) GO TO 890
      N=NFINOR
      DO 740 NN=1,NF
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      DECODE (56,30,ABCD) ((FINORG(NN,I,J),J=1,4),I=1,2)
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      DECODE (70,30,ABCD) (XFIN(NN,I),I=1,N)
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      DECODE (70,30,ABCD) (FINORD(NN,1,J),J=1,N)
740   CONTINUE
C
C          CHANGE TO ACTUAL UNITS, COMPUTE MINIMUMS AND MAXIMUMS
C
```

```
      DO 760 LQ=1,NF
      DO 760 I=1,2
      J=3-I
      E=.01*FINORG(LQ,J,4)
      E2=FINORG(LQ,J,2)
      DO 750 K=1,NFINOR
      EE=FINORD(LQ,1,K)*E
      FINORD(LQ,J,K)=E2+EE
      FINX2(LQ,J,K)=E2-EE
750   FINX3(LQ,J,K)=FINORG(LQ,J,1)+E*XFIN(LQ,K)
760   CONTINUE
C
      IF (J1.NE.0.OR.J2.NE.0.OR.J3.NE.0) GO TO 770
      XMIN=FINORG(1,1,1)
      XMAX=FINORG(1,1,1)
      YMIN=FINORG(1,1,2)
      YMAX=FINORG(1,1,2)
      ZMIN=FINORG(1,1,3)
      ZMAX=FINORG(1,1,3)
770   DO 780 N=1,NF
      ZMIN=AMIN1(ZMIN,FINORG(N,1,3))
      ZMAX=AMAX1(ZMAX,FINORG(N,2,3))
      DO 780 N2=1,2
      XMIN=AMIN1(XMIN,FINORG(N,N2,1))
      XMAX=AMAX1(XMAX,FINX3(N,N2,NFINOR))
      DO 780 NN=1,NFINOR
      YMIN=AMIN1(YMIN,FINX2(N,N2,NN))
      YMAX=AMAX1(YMAX,FINORD(N,N2,NN))
780   CONTINUE
C
C          WRITE LINE TAPE
C
      JJN=4$NAME(1)=10HFINS       $NAME(2)=10H
      NOBJ=NOBJ+NF
      NK2=2
      DO 880 NF1=1,NF
      I=2
      IF (FINORG(NF1,1,2).EQ.0.) I=1
      WRITE (10) I,JJN,NAME,I,I
      DO 870 NN2=1,2
      IF (NN2.EQ.2.AND.FINORG(NF1,1,2).EQ.0.) GO TO 870
      WRITE (10) NFINOR,NK2,I,I,I
      I1=1
      I2=2
      IF (NN2.EQ.1) GO TO 790
      I1=2$I2=1
790   CONTINUE
C
C          SETUP HORIZONTAL LINES
C
      DO 810 N=1,NFINOR
      ALRT(N,1)=FINX3(NF1,I1,N)
      ALRT(N,3)=FINORG(NF1,I1,3)
      IF (NN2.EQ.2) GO TO 800
      ALRT(N,2)=FINORD(NF1,I1,N)
      GO TO 810
800   ALRT(N,2)=FINX2(NF1,I1,N)
810   CONTINUE
```

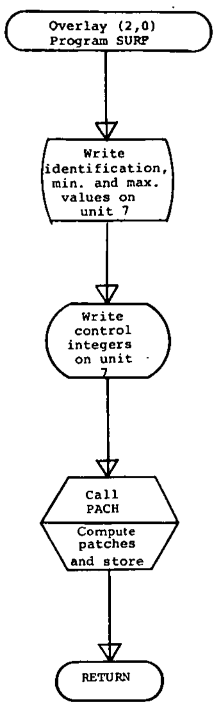
```
      WRITE (10) ((ALRT(N,N3),N=1,NFINOR),N3=1,3)
      DO 830 N=1,NFINOR
      ALRT(N,1)=FINX3(NF1,I2,N)
      ALRT(N,3)=FINORG(NF1,I2,3)
      IF (NN2.EQ.2) GO TO 820
      ALRT(N,2)=FINORD(NF1,I2,N)
      GO TO 830
820   ALRT(N,2)=FINX2(NF1,I2,N)
830   CONTINUE
      WRITE (10) ((ALRT(N,N3),N=1,NFINOR),N3=1,3)
C
C          SETUP VERTICAL LINES
C
      DO 860 NN=1,NFINOR
      ALRT(1,1)=FINX3(NF1,I1,NN)
      ALRT(2,1)=FINX3(NF1,I2,NN)
      ALRT(1,3)=FINORG(NF1,I1,3)
      ALRT(2,3)=FINORG(NF1,I2,3)
      IF (NN2.EQ.2) GO TO 840
      ALRT(1,2)=FINORD(NF1,I1,NN)
      ALRT(2,2)=FINORD(NF1,I2,NN)
      GO TO 850
840   ALRT(1,2)=FINX2(NF1,I1,NN)
      ALRT(2,2)=FINX2(NF1,I2,NN)
850   WRITE (10) ((ALRT(N,N3),N=1,2),N3=1,3)
860   CONTINUE
870   CONTINUE
880   CONTINUE
C
C          CANARDS
C
890   CONTINUE
      IF (J5.NE.1) GO TO 1080
      N=IABS(NCANOR)
      DO 920 NN=1,NCAN
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      DECODE (56,30,ABCD) ((CANORG(NN,I,J),J=1,4),I=1,2)
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      DECODE (70,30,ABCD) (XCAN(NN,I),I=1,N)
      READ (5,10) ABCD
      WRITE (6,20) ABCD
      DECODE (70,30,ABCD) (CANORD(NN,1,J),J=1,N)
      IF (NCANOR.LT.0) GO TO 910
      DO 900 J=1,N
900   CANOR1(NN,1,J)=CANORD(NN,1,J)
      GO TO 920
910   READ (5,10) ABCD
      WRITE (6,20) ABCD
      DECODE (70,30,ABCD) (CANOR1(NN,1,J),J=1,N)
920   CONTINUE
      NCANOR=IABS(NCANOR)
      NC=NCANOR
C
C          CHANGE TO ACTUAL UNITS, COMPUTE MINIMUMS AND MAXIMUMS
C
```

```
      DO 950 NN=1,NCAN
      DO 940 K=1,2
      I=3-K
      E=.01*CANORG(NN,1,4)
      E3=CANORG(NN,I,3)
      DO 930 J=1,NCANOR
      CANORD(NN,I,J)=E*CANORD(NN,1,J)+E3
      CANOR1(NN,I,J)=-E*CANOR1(NN,1,J)+E3
930   CANORX(NN,I,J)=CANORG(NN,I,1)+E*XCAN(NN,J)
940   CONTINUE
950   CONTINUE
      IF (J1.NE.0.OR.J2.NE.0.OR.J3.NE.0.OR.J4.NE.0) GO TO 960
      XMIN=CANORX(1,1,1)
      XMAX=CANORX(1,1,NCANOR)
      YMIN=CANORG(1,2,2)
      YMAX=CANORG(1,2,2)
      ZMIN=CANOR1(1,1,1)
      ZMAX=CANORD(1,1,1)
960   DO 990 NCA=1,NCAN
      YMIN=AMIN1(YMIN,CANORG(NCA,1,2))
      YMAX=AMAX1(YMAX,CANORG(NCA,2,2))
      DO 980 N2=1,2
      XMIN=AMIN1(XMIN,CANORX(NCA,N2,1))
      XMAX=AMAX1(XMAX,CANORX(NCA,N2,NCANOR))
      DO 970 NN=1,NCANOR
      ZMIN=AMIN1(ZMIN,CANOR1(NCA,N2,NN))
970   ZMAX=AMAX1(ZMAX,CANORD(NCA,N2,NN))
980   CONTINUE
990   CONTINUE
C
C          WRITE LINE TAPE
C
      JJN=5$NAME(1)=10HCANARDS    $NAME(2)=10H
      NOBJ=NOBJ+NCAN
      NK2=2
      DO 1070 NCA=1,NCAN
      WRITE (10) NK2,JJN,NAME,NK2,NK2
      DO 1060 I=1,2
      WRITE (10) NK2,NC,NK2,NK2,NK2
      KKK=(I-1)*(NC+1)
      KK=(-1)**(I+1)
C
C          SETUP SPANWISE LINES
C
      DO 1020 K=1,NC
      NN=KKK+KK*K
      DO 1010 N2=1,2
      ALRT(N2,1)=CANORX(NCA,N2,NN)
      ALRT(N2,2)=CANORG(NCA,N2,2)
      IF (I.EQ.2) GO TO 1000
      ALRT(N2,3)=CANORD(NCA,N2,NN)
      GO TO 1010
1000  ALRT(N2,3)=CANOR1(NCA,N2,NN)
1010  CONTINUE
      WRITE (10) ((ALRT(N2,N3),N2=1,2),N3=1,3)
1020  CONTINUE
```

```
C
C          SETUP TWO STREAMWISE LINES
C
      DO 1050 N2=1,2
      DO 1040 N=1,NC
      J=KKK+KK*N
      ALRT(N,1)=CANORX(NCA,N2,J)
      ALRT(N,2)=CANORG(NCA,N2,2)
      IF (I.EQ.2) GO TO 1030
      ALRT(N,3)=CANORD(NCA,N2,J)
      GO TO 1040
1030  ALRT(N,3)=CANORI(NCA,N2,J)
1040  CONTINUE
      WRITE (10) ((ALRT(N,N3),N=1,NC),N3=1,3)
1050  CONTINUE
1060  CONTINUE
1070  CONTINUE
1080  CONTINUE
      RETURN
C
C          END OF START
C
      END
```

Program SURF

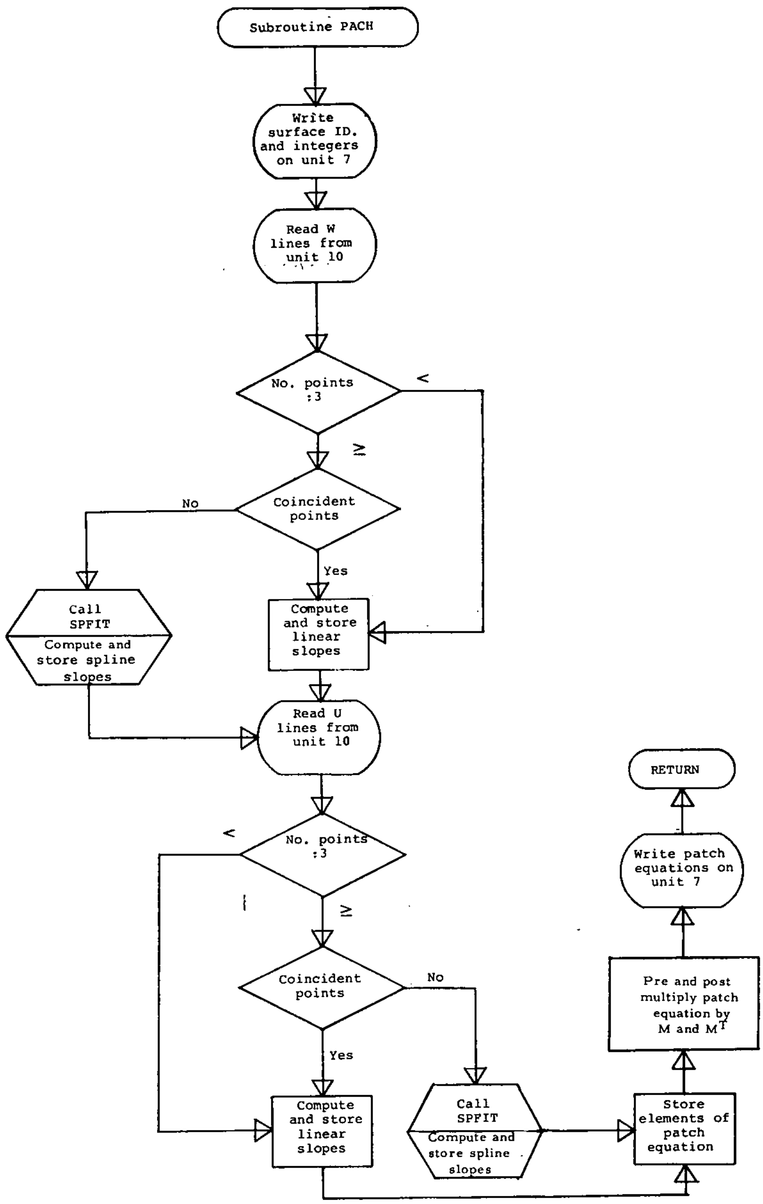
Program SURF (overlay (2,0)) is the control program for constructing surface patch equations. The flow chart and the FORTRAN statements for this overlay are as follows:




```
      OVERLAY(CBC,2,0)
      PROGRAM SURF
C
C          CALLS A SUBROUTINE TO COMPUTE PATCHES
C          AND CONTROLS WRITING OF PATCH TAPE
C
      COMMON/PATPLT/
     1XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,NOBJ
      DIMENSION ABC(8)
      REWIND 7
      REWIND 10
      READ (10) ABC
      CALL RECOUT (7,2,0,ABC,1,8,1)
      CALL RECOUT (7,1,0,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,NOBJ)
      DO 20 I=1,NOBJ
      READ (10) NSURF,J2,J3,J4,J5,J6
      CALL RECOUT (7,1,0,NSURF,J2,J3,J4,J5,J6)
      DO 10 N=1,NSURF
      READ (10) NCOL,NROW,N3,N4,N5
      CALL PACH (NCOL,NROW,N3,N4,N5)
   10 CONTINUE
   20 CONTINUE
      END FILE 7
      RETURN
      END
```

Subroutine PACH

Subroutine PACH computes surface patch equations from the given grid information describing a surface. The patch equations are stored for further use. The flow chart and the FORTRAN statements for this subroutine are as follows:



```
      SUBROUTINE PACH (NLD,NLS,L1,L2,L3)
C
C          CONSTRUCTS SURFACE PATCHES WITH THE COMPONENTS
C          EXPRESSED AS CUBIC FUNCTIONS OF TWO PARAMETERS (U AND W)
C          AND WRITES ON TAPE
C
      DIMENSION PATCH(4,4,3),COEF1(31,4,3),COEF2(31,4,3)
      DIMENSION SLOPE(31,31,3),XMAT(4,4),ALINE(31,3),ELEN(31),PAT(4,4)
      DATA (XMAT(I),I=1,16)/2.,-3.,0.,1.,-2.,3.,0.,0.,
     11.,-2.,1.,0.,1.,-1.,0.,0./
      DATA MAXN/31/,EPS/.00001/
      N1=NLD-1
      N2=NLS-1
      CALL RECOUT (7,1,0,N1,N2,L1,L2,L3)
C
C          COMPUTE PARAMETRIC SLOPES IN W DIRECTION
C
      DO 70 N=1,NLS
      READ (10) ((ALINE(NN,N3),NN=1,NLD),N3=1,3)
C
C          CHECK IF CUBIC FAIRING POSSIBLE
C
      IF (NLD.LT.3) GO TO 20
C
C          CHECK FOR A POINT
C
      NL=NLD-1
      DO 10 NN=1,NL
      E1=ABS(ALINE(NN,1)-ALINE(NN+1,1))
      E2=ABS(ALINE(NN,2)-ALINE(NN+1,2))
      E3=ABS(ALINE(NN,3)-ALINE(NN+1,3))
      IF (E1+E2+E3.LE.EPS) GO TO 20
10    CONTINUE
      GO TO 40
C
C          COMPUTE LINEAR SLOPES
C
20    DO 30 NN=1,NLD
      DO 30 N3=1,3
      SLOPE(NN,N,N3)=ALINE(2,N3)-ALINE(1,N3)
30    CONTINUE
      GO TO 70
C
C          COMPUTE CUBIC SPLINE SLOPES
C
40    CALL SPFIT (MAXN,NLD,ALINE,ELEN,COEF1,I1,0,I2,EK,CP,I3,I4)
      NL=NLD-1
      DO 50 NN=1,NL
      DO 50 N3=1,3
      SLOPE(NN,N,N3)=COEF1(NN,3,N3)
50    CONTINUE
      DO 60 N3=1,3
      SLOPE(NLD,N,N3)=3.*COEF1(NLD-1,1,N3)+2.*COEF1(NLD-1,2,N3)+COEF1(NL
     1D-1,3,N3)
60    CONTINUE
70    CONTINUE
C
```

```
C          COMPUTE PARAMETRIC SLOPES IN U DIRECTION
C          FORM PATCHES AND WRITE TAPE
C
      READ (10) ((ALINE(NN,N3),NN=1,NLS),N3=1,3)
C
C          CHECK IF CUBIC FAIRING POSSIBLE
C
      IF (NLS.LT.3) GO TO 90
C
C          CHECK FOR A POINT
C
      NL=NLS-1
      DO 80 NN=1,NL
      E1=ABS(ALINE(NN,1)-ALINE(NN+1,1))
      E2=ABS(ALINE(NN,2)-ALINE(NN+1,2))
      E3=ABS(ALINE(NN,3)-ALINE(NN+1,3))
      IF (E1+E2+E3.LE.EPS) GO TO 90
80    CONTINUE
      GO TO 110
C
C          COMPUTE LINEAR SLOPES
C
90    DO 100 NN=1,NLS
      DO 100 N3=1,3
      COEF1(NN,3,N3)=ALINE(2,N3)-ALINE(1,N3)
      COEF1(NN,4,N3)=ALINE(NN,N3)
100   CONTINUE
      GO TO 130
110   CONTINUE
C
C          COMPUTE CUBIC SPLINE SLOPES
C
      CALL SPFIT (MAXN,NLS,ALINE,ELEN,COEF1,K1,0,K2,EP,CP,K3,K4)
      DO 120 N3=1,3
      COEF1(NLS,3,N3)=3.*COEF1(NLS-1,1,N3)+2.*COEF1(NLS-1,2,N3)+COEF1(NL
     1S-1,3,N3)
      COEF1(NLS,4,N3)=COEF1(NLS-1,1,N3)+COEF1(NLS-1,2,N3)+COEF1(NLS-1,3,
     1N3)+COEF1(NLS-1,4,N3)
120   CONTINUE
130   CONTINUE
      DO 290 N=2,NLD
      READ (10) ((ALINE(NN,N3),NN=1,NLS),N3=1,3)
      IF (NLS.LT.3) GO TO 150
      NL=NLS-1
      DO 140 NN=1,NL
      E1=ABS(ALINE(NN,1)-ALINE(NN+1,1))
      E2=ABS(ALINE(NN,2)-ALINE(NN+1,2))
      E3=ABS(ALINE(NN,3)-ALINE(NN+1,3))
      IF (E1+E2+E3.LE.EPS) GO TO 150
140   CONTINUE
      GO TO 170
150   DO 160 NN=1,NLS
      DO 160 N3=1,3
      COEF2(NN,3,N3)=ALINE(2,N3)-ALINE(1,N3)
      COEF2(NN,4,N3)=ALINE(NN,N3)
160   CONTINUE
      GO TO 190
```

```
170    CALL SPFIT (MAXN,NLS,ALINE,ELEN,COEF2,K1,0,K2,EP,CP,K3,K4)
       DO 180 N3=1,3
       COEF2(NLS,3,N3)=3.*COEF2(NLS-1,1,N3)+2.*COEF2(NLS-1,2,N3)+COEF2(NL
      1S-1,3,N3)
       COEF2(NLS,4,N3)=COEF2(NLS-1,1,N3)+COEF2(NLS-1,2,N3)+COEF2(NLS-1,3,
      1N3)+COEF2(NLS-1,4,N3)
180    CONTINUE
C
C           STORE PATCHES
C
190    DO 270 L=2,NLS
       DO 210 N3=1,3
       DO 200 M=1,2
       MM=MOD(M,2)
       LL=L-MM
       PATCH(M,1,N3)=COEF1(LL,4,N3)
       PATCH(M,2,N3)=COEF2(LL,4,N3)
       PATCH(M,3,N3)=SLOPE(N-1,LL,N3)
       PATCH(M,4,N3)=SLOPE(N,LL,N3)
       PATCH(M+2,1,N3)=COEF1(LL,3,N3)
       PATCH(M+2,2,N3)=COEF2(LL,3,N3)
       PATCH(M+2,3,N3)=0.
       PATCH(M+2,4,N3)=0.
200    CONTINUE
210    CONTINUE
C
C           COMPUTE PATCH IN FORM OF S=MBM(TRANSPOSE) AND WRITE ON TAPE
C
       DO 260 N3=1,3
       DO 230 I4=1,4
       DO 230 J4=1,4
       SUM=0.
       DO 220 K4=1,4
220    SUM=SUM+XMAT(I4,K4)*PATCH(K4,J4,N3)
230    PAT(I4,J4)=SUM
C
       DO 250 I4=1,4
       DO 250 J4=1,4
       SUM=0.
       DO 240 K4=1,4
240    SUM=SUM+PAT(I4,K4)*XMAT(J4,K4)
250    PATCH(I4,J4,N3)=SUM
260    CONTINUE
       CALL RECOUT (7,2,0,PATCH,1,48,1)
270    CONTINUE
C
C           MOVE COEFFICIENTS
C
       DO 280 N3=1,3
       DO 280 N4=1,4
       DO 280 NN=1,NLS
       COEF1(NN,N4,N3)=COEF2(NN,N4,N3)
280    CONTINUE
290    CONTINUE
       RETURN
       END
```

Subroutine SPFIT

Subroutine SPFIT uses a parametric cubic spline curve fit technique with optional enrichment of the given input curve. The method is explained in appendix A. The description, flow chart, and the FORTRAN statements for this subroutine are as follows:

Language: FORTRAN

Purpose: SPFIT is a parametric cubic spline curve fit subroutine. Parametric coefficients are computed to approximate a cubic spline curve through a three-dimensional set of input points describing a curve, and, optionally an enriched curve is computed.

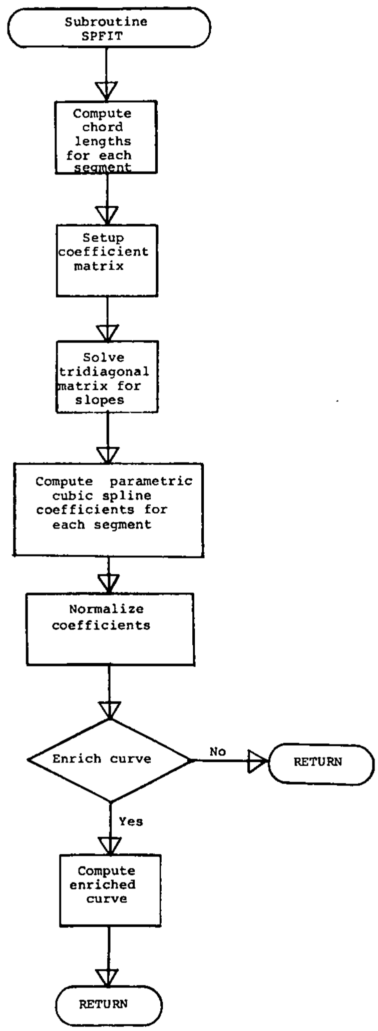
Use: CALL SPFIT (MAXN, N, PNT, ELEN, COEF, NFIT, MAXSP, II, EPS, CPT, K1, K2)

MAXN	The maximum number of input points allowed as stated in the dimension statement of the calling program.
N	The number of input points; $4 \leqq N \leqq MAXN$.
PNT	A two-dimensional array of the consecutive points describing the three-dimensional (X,Y,Z) input curve.
ELEN	A one-dimensional array used by the subroutine for the chord lengths between each consecutive pair of input points.
COEF	A three-dimensional array used by the subroutine for the parametric cubic spline coefficients.
NFIT	A number of interpolated points to be computed between each pair of given points as specified by the user.
MAXSP	The maximum number of points allowed in the enriched curve as stated in the dimension statement of the calling program. If MAXSP is 0, only the cubic spline coefficients are computed and the calculation of the enriched curve is omitted.
II	The total number of points in the enriched curve calculated by the subroutine.
EPS	A small number supplied by the user which is used to check the second derivative at each point of the faired curve. The point will be omitted if the absolute value of the second derivative is less than EPS. An EPS of 0.0 will cause all the interpolated points to be retained.
CPT	A two-dimensional array used by the program for storage of the enriched curve.

K1 An integer supplied by the user. If K1=1, retain all the input points. If K1=2, include input points in second derivative test.

K2 An integer supplied by SPFIT as an error code. If K2=1, normal return. If K2=2, error return when the number of interpolated points exceeds the allowable storage (MAXSP).

Restrictions: SPFIT has been written with a variable dimension statement, and the following must be dimensioned in the calling program: PNT(MAXN,3), ELEN(MAXN), COEF(MAXN,4,3), CPT(MAXSP,3). If the coefficient-only option is used (MAXSP=0), dummy entries for NFIT, II, EPS, CPT, K1, and K2 must be included in the calling sequence. The input curve must not include any consecutive duplicate points.



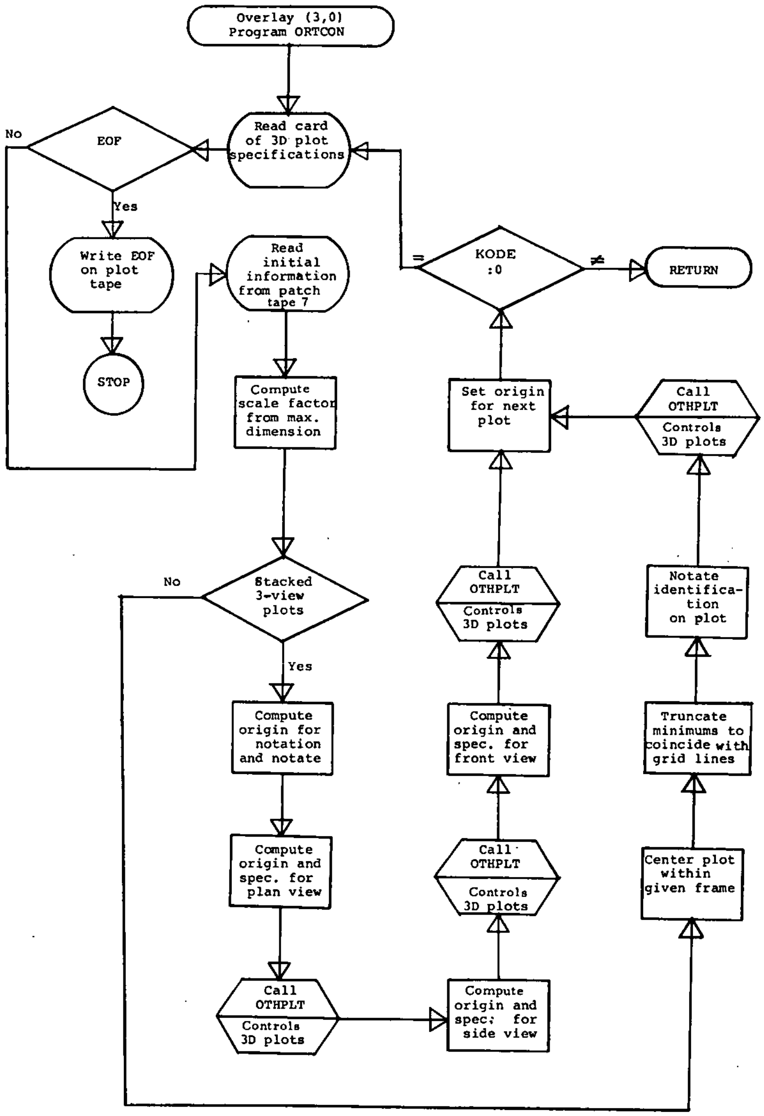
```
      SUBROUTINE SPFIT (MAXN,N,PNT,ELEN,COEF,NFIT,MAXSP,II,EPS,CPT,K1,K2
     1)
C
C          COMPUTES PARAMETRIC CUBIC SPLINE COEFFICIENTS TO
C          APPROXIMATE A SMOOTH CURVE THROUGH A 3D SET OF INPUT
C          POINTS AND OPTIONALLY COMPUTES AN ENRICHED CURVE
C          MAXN IS THE MAXIMUM NUMBER OF INPUT POINTS ALLOWED
C          N IS THE ACTUAL NO. OF INPUT POINTS
C          NFIT IS THE NUMBER OF DESIRED SPLINED POINTS BETWEEN GIVEN
C               POINTS
C          MAXSP IS THE MAXIMUM NUMBER OF SPLINED POINTS ALLOWED,
C               MAXSP=(MAXN-1)*(MAX,NFIT+1)+1 FOR EPS OF 0.
C               MAXSP=0 OMITS COMPUTATION OF ENRICHED CURVE
C          II IS THE NO. OF POINTS IN THE ENRICHED CURVE
C          K1 IS AN INTEGER SUPPLIED BY THE USER
C               K1=1,RETAIN ALL INPUT POINTS
C               K1=2,INCLUDE INPUT POINTS IN SECOND DERIVATIVE TEST
C          K2 IS AN INTEGER SUPPLIED BY SPFIT AS AN ERROR CODE
C               K2=1,NORMAL RETURN
C               K2=2,INCOMPLETE FAIRED CURVE WHEN MAXSP IS EXCEEDED
C
C          PROGRAMER - CHARLOTTE CRAIDON     2-1-71
C
      DIMENSION PNT(MAXN,3),ELEN(MAXN),COEF(MAXN,4,3),CPT(MAXSP,3)
      DIST(X1,Y1,Z1,X2,Y2,Z2)=SQRT((X2-X1)**2+(Y2-Y1)**2+(Z2-Z1)**2)
      N1=N-1
C
C          COMPUTE CHORD LENGTHS
C
      DO 10 NN=2,N
      ELEN(NN-1)=DIST(PNT(NN-1,1),PNT(NN-1,2),PNT(NN-1,3),PNT(NN,1),PNT(
     1NN,2),PNT(NN,3))
10    CONTINUE
C
C          SETUP COEFFICIENT MATRIX WITH UNCLAMPED END POINTS
C          (2ND DER=0. AT P1 AND PN)
C
      COEF(1,1,1)=0.
      COEF(1,1,2)=2.
      COEF(1,1,3)=1.
      COEF(N,1,1)=1.
      COEF(N,1,2)=2.
      COEF(N,1,3)=0.
      DO 20 NN=2,N1
      COEF(NN,1,1)=ELEN(NN)
      COEF(NN,1,2)=2.*(ELEN(NN-1)+ELEN(NN))
      COEF(NN,1,3)=ELEN(NN-1)
20    CONTINUE
C
C          SOLVE FOR SLOPES
C
      DO 60 I=1,3
      COEF(1,4,I)=(3./ELEN(1))*(PNT(2,I)-PNT(1,I))
      COEF(N,4,I)=(3./ELEN(N-1))*(PNT(N,I)-PNT(N-1,I))
      DO 30 NN=2,N1
      COEF(NN,4,I)=(3./(ELEN(NN-1)*ELEN(NN)))*(ELEN(NN-1)**2*(PNT(NN+1,I
     1)-PNT(NN,I))+ELEN(NN)**2*(PNT(NN,I)-PNT(NN-1,I)))
30    CONTINUE
```

```
C
C          SOLVE TRIDIAGONAL MATRIX
C
      COEF(1,2,1)=COEF(1,1,3)/COEF(1,1,2)
      COEF(1,3,I)=COEF(1,4,1)/COEF(1,1,2)
      DO 40 K=2,N
      KM1=K-1
      TEMP=COEF(K,1,2)-COEF(K,1,1)*COEF(KM1,2,1)
      COEF(K,2,1)=COEF(K,1,3)/TEMP
      COEF(K,3,I)=(COEF(K,4,1)-COEF(K,1,1)*COEF(KM1,3,I))/TEMP
40    CONTINUE
      DO 50 K=1,N1
      KK=N-K
      COEF(KK,3,I)=COEF(KK,3,I)-COEF(KK,2,1)*COEF(KK+1,3,I)
50    CONTINUE
60    CONTINUE
C
C          COMPUTE CUBIC COEFFICIENTS FOR EACH SEGMENT
C
      DO 70 NN=1,N1
      EL=1./ELEN(NN)
      EL2=EL*EL
      EL3=EL*EL2
      DO 70 I=1,3
      COEF(NN,4,I)=PNT(NN,I)
      E=PNT(NN+1,I)-PNT(NN,I)
      COEF(NN,2,I)=E*EL2*3.-EL*(2.*COEF(NN,3,I)+COEF(NN+1,3,I))
      COEF(NN,1,I)=-E*EL3*2.+EL2*(COEF(NN,3,I)+COEF(NN+1,3,I))
C
C          REFERENCE LENGTH TO 1.
C
      COEF(NN,1,I)=COEF(NN,1,I)/EL3
      COEF(NN,2,I)=COEF(NN,2,I)/EL2
      COEF(NN,3,I)=COEF(NN,3,I)/EL
70    CONTINUE
      IF (MAXSP.EQ.0) RETURN
      IF (K1.EQ.0) K1=1
      II=0
C
C          COMPUTE ENRICHED POINTS
C
      IFIT=NFIT+1
      XFIT=IFIT
      DELT=1./XFIT
      DO 110 NN=1,N1
      DO 100 NF=1,IFIT
      E=NF-1
      T=DELT*E
      IF (NN.EQ.1.AND.NF.EQ.1) GO TO 80
      IF (NF.EQ.1.AND.K1.EQ.1) GO TO 80
      T6=6.*T
      EX=ABS(T6*COEF(NN,1,1)+2.*COEF(NN,2,1))
      EY=ABS(T6*COEF(NN,1,2)+2.*COEF(NN,2,2))
      EZ=ABS(T6*COEF(NN,1,3)+2.*COEF(NN,2,3))
      EE=(EX+EY+EZ)/(ELEN(NN)*ELEN(NN))
      IF (EE.LT.EPS) GO TO 100
```

```
80    II=II+1
      IF (II.GT.MAXSP) GO TO 130
      T2=T*T
      T3=T*T2
      DO 90 I=1,3
90    CPT(II,I)=T3*COEF(NN,1,I)+T2*COEF(NN,2,I)+T*COEF(NN,3,I)+COEF(NN,4
     1,I)
100   CONTINUE
110   CONTINUE
      II=II+1
      IF (II.GT.MAXSP) GO TO 130
      DO 120 I=1,3
120   CPT(II,I)=PNT(N,I)
      K2=1
      RETURN
130   K2=2
      RETURN
      END
```

Program ORTCON

Program ORTCON (overlay (3,0)) is the control routine for the orthographic projections of the input body. This program reads the plot information card and prints it, computes scale factors, computes vertical offsets for three-view plots, and notates on the plot. The flow chart and the FORTRAN statements for this overlay are as follows:



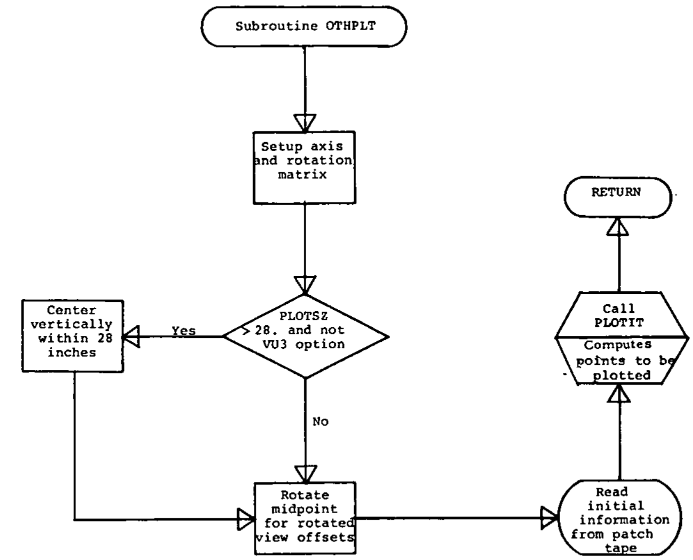
```
      OVERLAY(CBC,3,0)
      PROGRAM ORTCON
C
C          CONTROL ROUTINE FOR ORTHOGRAPHIC  PLOTS
C          OF A SURFACE OR OF A COLLECTION OF SURFACES
C
      COMMON /THREED/ABCDE(8),HORZ,VERT,TEST1,PHI,THETA,PSI,
     1PLOTSZ,TYPE,NOU,NOW,ISIDE,KODE
C
      DIMENSION ORG(3),ABC(8)
      DATA TYPEO/3HORT/,TYPEV/3HVU3/
C
C
C          READ PLOT CARD
C
      WRITE (6,10)
10    FORMAT (1H126X,27HTHREE DIMENSIONAL PLOT DATA//)
20    CONTINUE
      READ (5,30) ABCDE
30    FORMAT (8A10)
      IF (ENDFILE 5) 35,40
35    CALL NFRAME $ CALL CALPLT(0.,0.,999) $ STOP
40    WRITE (6,50) ABCDE
50    FORMAT (1X,8A10/)
      DECODE (72,60,ABCDE) HORZ,VERT,TEST1,PHI,THETA,PSI,PLOTSZ,TYPE,NOU
     1,NOW,ISIDE,KODE
60    FORMAT (2A2,A3,3F5.0,25X,F5.0,A3,3I3,7X,I1)
      IF (ISIDE.EQ.0) ISIDE=1
C
C          READ PATCH TAPE
C
      REWIND 7
      CALL RECIN (7,2,IC,ABC,1,8,1)
      IF (ENDFILE 7) 70,90
70    WRITE (6,80)
80    FORMAT (1H1/38H END OF FILE ENCOUNTERED ON PATCH TAPE)
      STOP
90    CONTINUE
      CALL RECIN (7,1,IC,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,NOBJ)
      IF (ISIDE.EQ.2) YMIN=-YMAX
C
C          FIND SCALE FACTOR FROM MAXIMUM DIMENSION
C
      XDIS=XMAX-XMIN
      YDIS=YMAX-YMIN
      ZDIS=ZMAX-ZMIN
      DMAX=AMAX1(XDIS,YDIS,ZDIS)
      SCALE=DMAX/PLOTSZ
      IF (TYPE.NE.TYPEV) GO TO 140
C
C          3VU WHERE VIEWS ARE STACKED VERTICALLY
C
      ORG(1)=PHI
      ORG(2)=THETA
      ORG(3)=PSI
      PHI=THETA=PSI=0.
```

```
      YBIG=ORG(1)
      YORG=FLOAT(IFIX(YMAX/SCALE))+ORG(1)
      IF (YBIG.GT.ORG(2)) GO TO 100
      YBIG=ORG(2)
      YORG=FLOAT(IFIX(ZMAX/SCALE))+ORG(2)
  100 IF (YBIG.GT.ORG(3)) GO TO 110
      YBIG=ORG(3)
      YORG=FLOAT(IFIX(ZMAX/SCALE))+ORG(3)
  110 CALL CALPLT (0.,YORG,-3)
C
C          NOTATE ON 3VIEW PLOTS
C
      NCHAR=IFIX(6.*PLOTSZ)
      IF (NCHAR.GT.80) GO TO 120
      X=0.
      GO TO 130
  120 CONTINUE
      NDIF=(NCHAR-80)/2
      X=FLOAT(NDIF)/6.
      NCHAR=80
  130 CALL NOTATE (X,0.,.2,ABC,0.,NCHAR)
      YSAV=YMIN
      XMIN=YMIN=ZMIN=0.
      HORZ=1HX$VERT=1HY
      YORG=ORG(1)-YORG-1
      CALL CALPLT (0.,YORG,-3)
      CALL OTHPLT (XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,NOBJ,XMID,YMID,ZMID,SCA
     1LE)
      VERT=1HZ
      YORG=ORG(2)-ORG(1)
      CALL CALPLT (0.,YORG,-3)
      CALL OTHPLT (XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,NOBJ,XMID,YMID,ZMID,SCA
     1LE)
      HORZ=1HY
      YORG=ORG(3)-ORG(2)
      YMIN=(FLOAT(IFIX(YSAV/SCALE)-1))*SCALE
      CALL CALPLT (0.,YORG,-3)
      CALL OTHPLT (XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,NOBJ,XMID,YMID,ZMID,SCA
     1LE)
      X=FLOAT(IFIX(PLOTSZ+6.))
      Y=1.-ORG(3)
      GO TO 160
  140 CONTINUE
C
C          CENTER PLOT
C
      XMID=.5*(XMAX+XMIN)
      YMID=.5*(YMAX+YMIN)
      ZMID=.5*(ZMAX+ZMIN)
      XFIX=.5*(DMAX-XDIS)
      XMIN=XMIN-XFIX
      XMAX=XMAX+XFIX
      YFIX=.5*(DMAX-YDIS)
      YMIN=YMIN-YFIX
      YMAX=YMAX+YFIX
      ZFIX=.5*(DMAX-ZDIS)
      ZMIN=ZMIN-ZFIX
      ZMAX=ZMAX+ZFIX
```

```
C
C          ADJUST MINIMUMS FOR GRID LINES
C
      XMIN=FLOAT(IFIX(XMIN/SCALE))*SCALE
      YMIN=FLOAT(IFIX(YMIN/SCALE))*SCALE
      ZMIN=FLOAT(IFIX(ZMIN/SCALE))*SCALE
C
C          NOTATE ID ON PLOT
C
      X=0.
      NCHAR=IFIX(11.*PLOTSZ)+3
      IF (NCHAR.LE.80) GO TO 150
      NDIF=(NCHAR-80)/2
      X=FLOAT(NDIF)/11.
      NCHAR=80
150   CALL NOTATE (X,.8,.1,ABC,0.,NCHAR)
      CALL NOTATE (X,.5,.1,ABCDE,0.,NCHAR)
C
C          ORTHOGRAPHIC
C
      CALL OTHPLT (XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,NOBJ,XMID,YMID,ZMID,SCA
     1LE)
      X=FLOAT(IFIX(PLOTSZ+2.))
      Y=0.
C
C          END OF COMPLETE PLOT
C
160   CONTINUE
      CALL CALPLT(X,Y,-3) $CALL NFRAME
      IF (KODE.EQ.0) GO TO 20
      RETURN
C
C          END OF ORTCON
C
      END
```

Subroutine OTHPLT

Subroutine OTHPLT determines the specified axis system and paper plane, sets up the rotation matrix, and establishes the necessary offsets for proper plot placement. The flow chart and the FORTRAN statements for this subroutine are as follows:



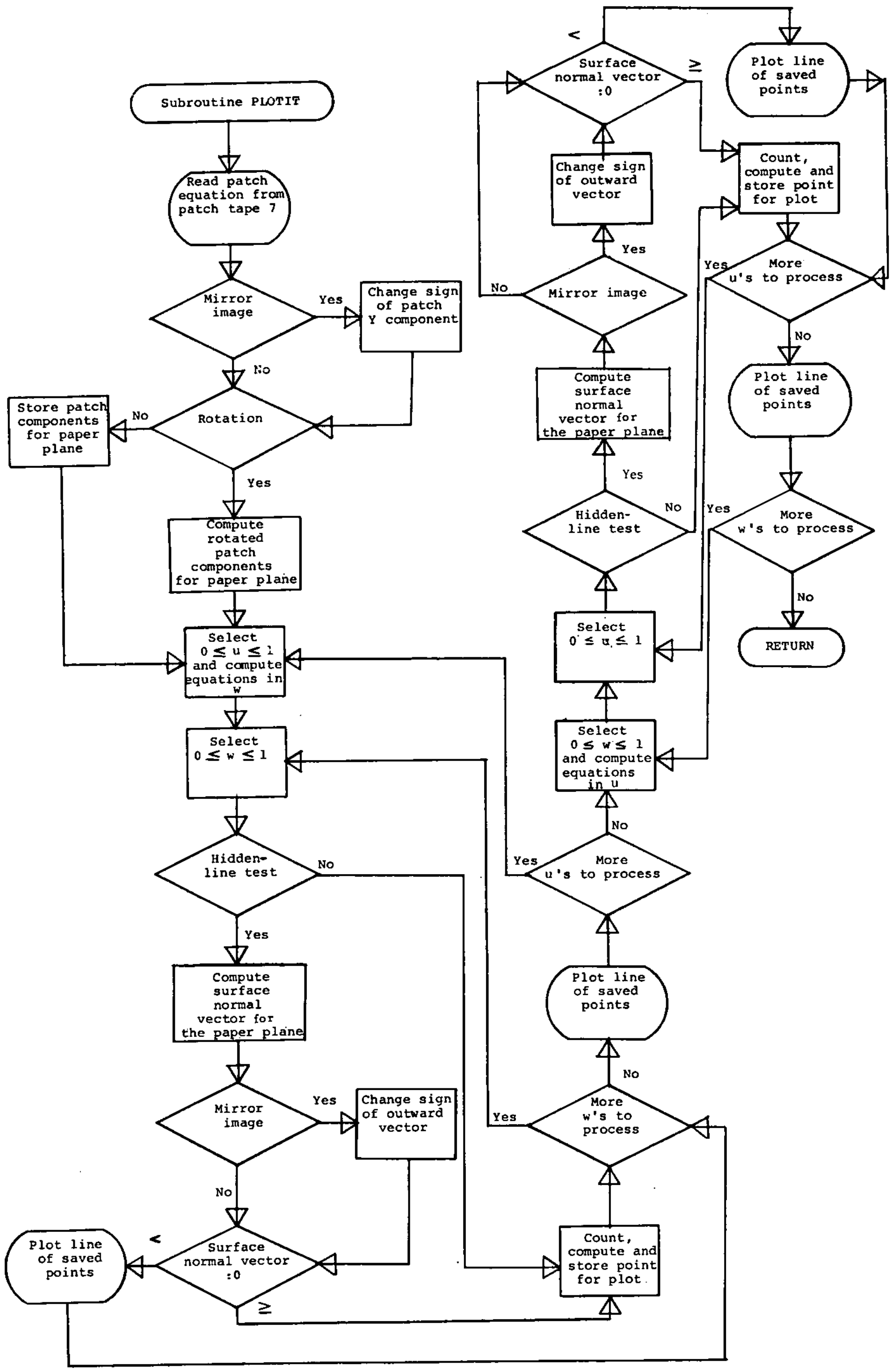

```
      SUBROUTINE OTHPLT (XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,NOBJ,XMID,YMID,ZM
     1ID,SCALE)
C
C         ORTHOGRAPHIC PROJECTIONS
C
C
      COMMON/THREED/ABCDE(8),HORZ,VERT,TEST1,PHI,THETA,PSI,
     1PLOTSZ,TYPE,NOU,NOW,ISIDE,KODE
C
      DIMENSION A(2,3),NAME(2),ABC(8)
C
      DATA XSEE/2HX /,YSEE/2HY /,ZSEE/2HZ /,
     1XINTST/3HOUT/,CONV/.017453293/,NUM2/2/,NAN2/24/
C
C         INITIALIZE
C
      ITEST1=1
      ITEST2=1
      IF (XINTST.NE.TEST1) ITEST1=0
      IF (PSI.EQ.0..AND.THETA.EQ.0..AND.PHI.EQ.0.) ITEST2=0
      PHI=CONV*PHI
      THETA=CONV*THETA
      PSI=CONV*PSI
C
C         SETUP AXIS
```

```
C
      SINPSI=SIN(PSI)
      SINTHE=SIN(THETA)
      SINPHI=SIN(PHI)
      COSPSI=COS(PSI)
      COSTHE=COS(THETA)
      COSPHI=COS(PHI)
      IF (XSEE.NE.HORZ) GO TO 20
C
C          USE X FOR HORIZONTAL VARIABLE
C
      IF (ITEST2.EQ.0) GO TO 10
      A(1,1)=COSTHE*COSPSI
      A(1,2)=-SINPSI*COSPHI+SINTHE*COSPSI*SINPHI
      A(1,3)=SINPSI*SINPHI+SINTHE*COSPSI*COSPHI
10    HMIN=XMIN
      HMAX=XMAX
      HMID=XMID
      IHORZ=1
      GO TO 60
20    IF (YSEE.NE.HORZ) GO TO 40
C
C          USE Y FOR HORIZONTAL VARIABLE
C
      IF (ITEST2.EQ.0) GO TO 30
      A(1,1)=COSTHE*SINPSI
      A(1,2)=COSPSI*COSPHI+SINTHE*SINPSI*SINPHI
      A(1,3)=-COSPSI*SINPHI+SINTHE*SINPSI*COSPHI
30    HMIN=YMIN
      HMAX=YMAX
      HMID=YMID
      IHORZ=2
      GO TO 60
C
C          USE Z FOR HORIZONTAL VARIABLE
C
40    CONTINUE
      IF (ITEST2.EQ.0) GO TO 50
      A(1,1)=-SINTHE
      A(1,2)=COSTHE*SINPHI
      A(1,3)=COSTHE*COSPHI
50    HMIN=ZMIN
      HMAX=ZMAX
      HMID=ZMID
      IHORZ=3
60    IF (XSEE.NE.VERT) GO TO 80
C
C          USE X FOR VERTICAL VARIABLE
C
      IF (ITEST2.EQ.0) GO TO 70
      A(2,1)=COSTHE*COSPSI
      A(2,2)=-SINPSI*COSPHI+SINTHE*COSPSI*SINPHI
      A(2,3)=SINPSI*SINPHI+SINTHE*COSPSI*COSPHI
70    VMIN=XMIN
      VMAX=XMAX
      VMID=XMID
      IVERT=1
      GO TO 120
80    IF (YSEE.NE.VERT) GO TO 100
```

```
C
C         USE Y FOR VERTICAL VARIABLE
C
      IF (ITEST2.EQ.0) GO TO 90
      A(2,1)=COSTHE*SINPSI
      A(2,2)=COSPSI*COSPHI+SINTHE*SINPSI*SINPHI
      A(2,3)=-COSPSI*SINPHI+SINTHE*SINPSI*COSPHI
90    VMIN=YMIN
      VMAX=YMAX
      VMID=YMID
      IVERT=2
      GO TO 120
C
C         USE Z FOR VERTICAL VARIABLE
C
100   CONTINUE
      IF (ITEST2.EQ.0) GO TO 110
      A(2,1)=-SINTHE
      A(2,2)=COSTHE*SINPHI
      A(2,3)=COSTHE*COSPHI
110   VMIN=ZMIN
      VMAX=ZMAX
      VMID=ZMID
      IVERT=3
120   CONTINUE
C
C         CENTER WITHIN PAGE SIZE IF SIZE GREATER THAN 28 INCHES
C
      IF (PLOTSZ.GT.28..AND.TYPE.NE.3HVU3) VMIN=-13.*SCALE+FLOAT(IFIX(VM
     1ID/SCALE))*SCALE
C
C         ROTATE MIDPOINT TO PLACE ROTATED VIEW CORRECTLY
C
      IF (ITEST2.EQ.0) GO TO 130
      AMID1=A(1,1)*XMID+A(1,2)*YMID+A(1,3)*ZMID
      AMID2=A(2,1)*XMID+A(2,2)*YMID+A(2,3)*ZMID
      HMIN=HMIN-HMID+AMID1
      VMIN=VMIN-VMID+AMID2
130   CONTINUE
C
C         BEGIN PLOTTING
C
      DO 160 ISI=1,ISIDE
      REWIND 7
      CALL RECIN (7,2,IC,ABC,1,8,1)
      CALL RECIN (7,1,IC,H1,H2,H3,H4,H5,H6,I7)
      DO 150 J=1,NOBJ
      CALL RECIN (7,1,IC,NSURF,J3,NAME(1),NAME(2),J4,J5)
      DO 140 N=1,NSURF
      CALL RECIN (7,1,IC,ND1,NS1,J3,J4,J5)
      CALL PLOTIT (ND1,NS1,ISI,ITEST,ITEST1,ITEST2,IHORZ,IVERT,HMIN,VMIN
     1,SCALE,A)
140   CONTINUE
150   CONTINUE
160   CONTINUE
      RETURN
C
C         END OF OTHPLT
C
      END
```

Subroutine PLOTIT

Subroutine PLOTIT reads patch equations from tape and rotates them, computes enriched surfaces, and does a visibility test if desired. The flow chart and the FORTRAN statements for this subroutine are as follows:



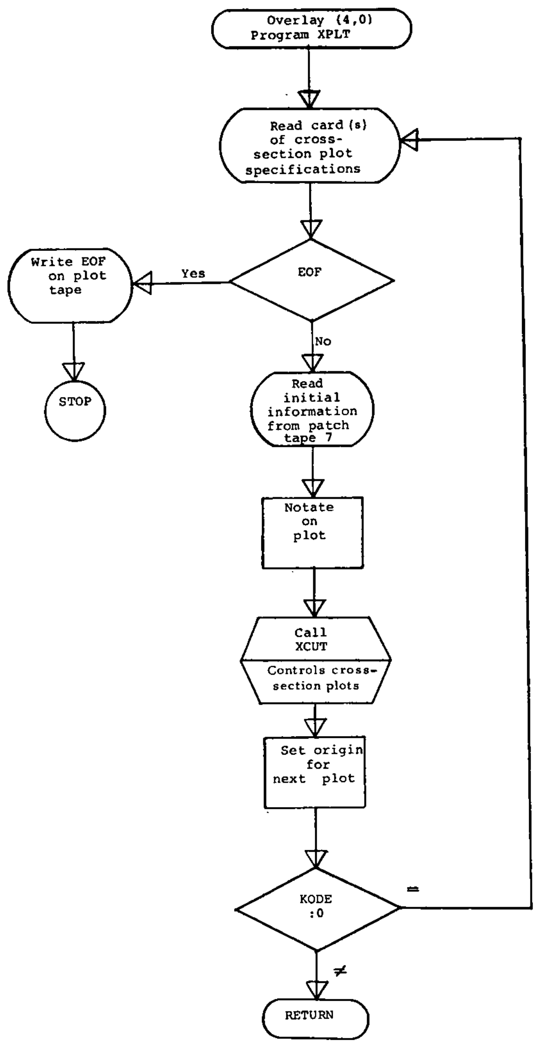
```
      SUBROUTINE PLOTIT (ND1,NS1,ISI,ITEST,ITEST1,ITEST2,IHORZ,IVERT,HMI
     1N,VMIN,SCALE,A)
C
C          READS PATCHES FROM TAPE,
C          MANIPULATES IN SPECIFIED MANNER AND PLOTS
C
      DIMENSION PATCH(4,4,3),PAT(4,4,2),A(2,3),PLPAT(4,2),
     1PLINE(54,2)
      DIMENSION VEC(4,2),VPAT(4)
      COMMON/THREED/ABCDE(8),HORZ,VERT,TEST1,PHI,THETA,PSI,
     1PLOTSZ,TYPE,NOU,NOW,ISIDE,KODE
C
      NNU=NOU+2
      NNW=NOW+2
      FU=NOU+1
      FW=NOW+1
      DU=1./FU
      DW=1./FW
      NPAT=ND1*NS1
      DO 230 N=1,NPAT
      CALL RECIN (7,2,IC,PATCH,1,48,1)
      IF (ISI.EQ.1) GO TO 20
C
C          CHANGE Y SIGN
C
      DO 10 I4=1,4
      DO 10 J4=1,4
      PATCH(I4,J4,2)=-PATCH(I4,J4,2)
10    CONTINUE
20    CONTINUE
C
C          ROTATE PATCHES
C
      IF (ITEST2.EQ.1) GO TO 40
      DO 30 I4=1,4
      DO 30 J4=1,4
      PAT(I4,J4,1)=PATCH(I4,J4,IHORZ)
      PAT(I4,J4,2)=PATCH(I4,J4,IVERT)
30    CONTINUE
      GO TO 80
40    CONTINUE
      DO 70 I4=1,4
      DO 70 J4=1,4
      DO 60 K2=1,2
      PAT(I4,J4,K2)=0.
      DO 50 N3=1,3
      PAT(I4,J4,K2)=PAT(I4,J4,K2)+A(K2,N3)*PATCH(I4,J4,N3)
50    CONTINUE
60    CONTINUE
70    CONTINUE
80    CONTINUE
C
C          PLOT IN W DIRECTION
C
      DO 150 NU=1,NNU
      EU=NU-1
      U=EU*DU
```

```
      DO 90 J4=1,4
      DO 90 K2=1,2
      PLPAT(J4,K2)=((U*PAT(1,J4,K2)+PAT(2,J4,K2))*U+PAT(3,J4,K2))*U+PAT(
     14,J4,K2)
      VEC(J4,K2)=(3.*U*PAT(1,J4,K2)+2.*PAT(2,J4,K2))*U+PAT(3,J4,K2)
90    CONTINUE
      NIT=0
      DO 140 NW=1,NNW
      EW=NW-1
      W=EW*DW
      IF (ITEST1.EQ.0) GO TO 120
C
C          COMPUTE DV/DU AND DV/DW
C
      DO 100 J=1,2
      VPAT(J)=((W*VEC(1,J)+VEC(2,J))*W+VEC(3,J))*W+VEC(4,J)
      VPAT(J+2)=(3.*PLPAT(1,J)*W+2.*PLPAT(2,J))*W+PLPAT(3,J)
100   CONTINUE
      VNORM=VPAT(1)*VPAT(4)-VPAT(2)*VPAT(3)
      IF (ISI.EQ.1) VNORM=-VNORM
      IF (VNORM.GE.0.) GO TO 120
      IF (NIT.GT.1) GO TO 110
      NIT=0
      GO TO 140
110   PLINE(NIT+1,1)=HMIN$PLINE(NIT+1,2)=VMIN
      PLINE(NIT+2,1)=PLINE(NIT+2,2)=SCALE
      CALL LINE (PLINE(1,1),PLINE(1,2),NIT,1,0,0,0)
      NIT=0
      GO TO 140
120   NIT=NIT+1
      DO 130 K2=1,2
      PLINE(NIT,K2)=((W*PLPAT(1,K2)+PLPAT(2,K2))*W+PLPAT(3,K2))*W+PLPAT(
     14,K2)
130   CONTINUE
140   CONTINUE
      IF (NIT.LE.1) GO TO 150
      PLINE(NIT+1,1)=HMIN$PLINE(NIT+1,2)=VMIN
      PLINE(NIT+2,1)=PLINE(NIT+2,2)=SCALE
      CALL LINE (PLINE(1,1),PLINE(1,2),NIT,1,0,0,0)
150   CONTINUE
C
C          PLOT IN U DIRECTION
C
      DO 220 NW=1,NNW
      EW=NW-1
      W=EW*DW
      DO 160 J4=1,4
      DO 160 K2=1,2
      PLPAT(J4,K2)=((W*PAT(J4,1,K2)+PAT(J4,2,K2))*W+PAT(J4,3,K2))*W+PAT(
     1J4,4,K2)
      VEC(J4,K2)=(3.*W*PAT(J4,1,K2)+2.*PAT(J4,2,K2))*W+PAT(J4,3,K2)
160   CONTINUE
      NIT=0
      DO 210 NU=1,NNU
      EU=NU-1
      U=EU*DU
      IF (ITEST1.EQ.0) GO TO 190
```

```
C
C          COMPUTE DV/DU AND DV/DW
C
      DO 170 J=1,2
      VPAT(J)=(3.*PLPAT(1,J)*U+2.*PLPAT(2,J))*U+PLPAT(3,J)
      VPAT(J+2)=((U*VEC(1,J)+VEC(2,J))*U+VEC(3,J))*U+VEC(4,J)
  170 CONTINUE
      VNORM=VPAT(1)*VPAT(4)-VPAT(2)*VPAT(3)
      IF (ISI.EQ.1) VNORM=-VNORM
      IF (VNORM.GE.0.) GO TO 190
      IF (NIT.GT.1) GO TO 180
      NIT=0
      GO TO 210
  180 PLINE(NIT+1,1)=HMIN$PLINE(NIT+1,2)=VMIN
      PLINE(NIT+2,1)=PLINE(NIT+2,2)=SCALE
      CALL LINE (PLINE(1,1),PLINE(1,2),NIT,1,0,0,0)
      NIT=0
      GO TO 210
  190 NIT=NIT+1
      DO 200 K2=1,2
      PLINE(NIT,K2)=((U*PLPAT(1,K2)+PLPAT(2,K2))*U+PLPAT(3,K2))*U+PLPAT(
     14,K2)
  200 CONTINUE
  210 CONTINUE
      IF (NIT.LE.1) GO TO 220
      PLINE(NIT+1,1)=HMIN$PLINE(NIT+1,2)=VMIN
      PLINE(NIT+2,1)=PLINE(NIT+2,2)=SCALE
      CALL LINE (PLINE(1,1),PLINE(1,2),NIT,1,0,0,0)
  220 CONTINUE
  230 CONTINUE
      RETURN
      END
```

Program XPLT

Program XPLT (overlay (4,0)) is the control routine for cross-section plots through the input body. The program reads the plot information card and prints it and notates on the plot. The flow chart and the FORTRAN statements for this overlay are as follows:



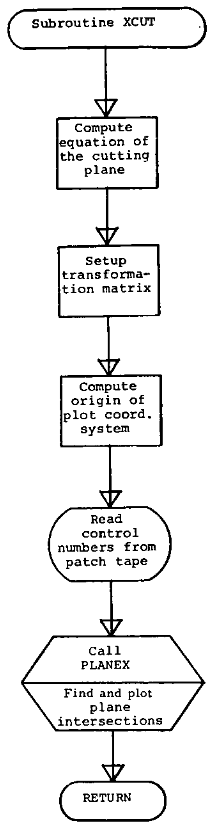
```
      OVERLAY(CBC,4,0)
      PROGRAM XPLT
C
C          CONTROL PROGRAM FOR CROSS
C          SECTIONAL OR CONTOUR PLOTS
C
      COMMON/XSECT/ABCDE(8),PPL1(3),PPL2(3),PPL3(3),
     1PLOTSZ,HPAGE,VPAGE,INP,NOU,NOW,ISIDE,IPRIN,
     2KODE,XSTAT,THETR,XMACH
      DIMENSION ABC(8),ABCD(8)
C
C          READ PLOT CARD(S) AND PRINT
C
      WRITE (6,10)
10    FORMAT (1H127X,25HCROSS SECTIONAL PLOT DATA//)
20    CONTINUE
      READ (5,30) ABCDE
30    FORMAT (8A10)
      IF (ENDFILE 5) 35,40
35    CALL NFRAME $ CALL CALPLT(0.,0.,999) $ STOP
40    CONTINUE
      WRITE (6,50)
50    FORMAT (/26X,30H*******  PLOT CARD(S)   *******//)
      WRITE (6,60) ABCDE
60    FORMAT (1X8A10)
      DECODE (80,70,ABCDE) X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,PLOTSZ,HPAGE,VPAGE
     1,INP,NOU,NOW,ICUT,ISIDE,IPRIN,KODE
70    FORMAT (10F6.0,2F3.0,A3,2I3,I2,3I1)
      IF (ICUT.EQ.0) GO TO 90
      READ (5,30) ABCD
      WRITE (6,60) ABCD
      DECODE (21,80,ABCD) DX,DY,DZ,IH
80    FORMAT (3F6.0,I3)
90    CONTINUE
      IF (INP.EQ.3HANG) GO TO 110
      WRITE (6,100) X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,PLOTSZ,HPAGE,VPAGE
100   FORMAT (//13X,13HCUTTING PLANE/6X,1HX11X,1HY11X,1HZ/3F12.5/3F12.5/
     13F12.5//6X,5HSCALE7X,5HHPAGE7X,5HVPAGE/F12.5,2F12.2)
      GO TO 130
110   WRITE (6,120) X1,Y1,Z1,X2,Y2,Z2,PLOTSZ,HPAGE,VPAGE
120   FORMAT (//13X,13HCUTTING PLANE/6X,2HX010X,2HY010X,2HZ0/3F12.5/6X,1
     1HX11X,5HTHETA7X,4HMACH/3F12.5//6X,5HSCALE7X,5HHPAGE7X,5HVPAGE/F12.
     25,2F12.2)
130   CONTINUE
      WRITE (6,140) NOU,NOW,ICUT,ISIDE,IPRIN
140   FORMAT (6X,28HNOU  NOW  ICUT  ISIDE  IPRIN/I9,I5,I6,2I7)
      IF (ISIDE.EQ.0) ISIDE=1
      IF (IPRIN.EQ.0) IPRIN=1
      HSAV=HPAGE$IF(IH.EQ.0)HPAGE=0.
      NCUT=ICUT+1
      IF (ICUT.NE.0) GO TO 150
      DX=0.$DY=0.$DZ=0.
      GO TO 200
150   WRITE (6,160) DX,DY,DZ
160   FORMAT (6X,2HDX10X,2HDY10X,2HDZ/3F12.5)
      IF (IH.NE.0) GO TO 180
      WRITE (6,170)
170   FORMAT (27H OVERLAID PLOTS WITH IH = 0)
      GO TO 200
```

```
180    WRITE (6,190) IH
190    FORMAT (24H SPACED PLOTS WITH IH = ,I3)
C
C          LOOP FOR INCREMENTED CUTS
C
200    DO 350 N=1,NCUT
       IF (N.EQ.1) GO TO 290
       IF (INP.EQ.3HPNT) GO TO 250
       X2=X2+DX$Y2=Y2+DY$Z2=Z2+DZ
       DECODE (10,210,ABCDE(2) )TEMP1
210    FORMAT (A8)
       DECODE (10,220,ABCDE(4) )TEMP2
220    FORMAT (6XA4)
       ENCODE (40,230,ABCDE(2) )TEMP1,X2,Y2,Z2,TEMP2
230    FORMAT (A8,3F6.2,A4)
       WRITE (6,240) X2,Y2,Z2
240    FORMAT (//7X,25HINCREMENTED CUTTING PLANE/6X,1HX11X,5HTHETA7X,4HMA
      1CH/3F12.5)
       GO TO 290
250    X1=X1+DX$Y1=Y1+DY$Z1=Z1+DZ
       X2=X2+DX$Y2=Y2+DY$Z2=Z2+DZ
       X3=X3+DX$Y3=Y3+DY$Z3=Z3+DZ
       DECODE (10,260,ABCDE(6) )TEMP1
260    FORMAT (4XA6)
       ENCODE (60,270,ABCDE(1) )X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,TEMP1
270    FORMAT (9F6.2,A6)
       WRITE (6,280) X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3
280    FORMAT (//26H INCREMENTED CUTTING PLANE/6X,1HX11X,1HY11X,1HZ/3F12.
      15/3F12.5/3F12.5)
290    CONTINUE
       PPL1(1)=X1$PPL1(2)=Y1$PPL1(3)=Z1
       PPL2(1)=X2$PPL2(2)=Y2$PPL2(3)=Z2
       PPL3(1)=X3$PPL3(2)=Y3$PPL3(3)=Z3
       IF (N.EQ.NCUT) HPAGE=HSAV
C
C          READ PATCH TAPE
C
       REWIND 7
       CALL RECIN (7,2,IC,ABC,1,8,1)
       IF (ENDFILE 7) 300,320
300    WRITE (6,310)
310    FORMAT (1H1/38H END OF FILE ENCOUNTERED ON PATCH TAPE)
       STOP
320    CONTINUE
C
C          NOTATE
C
       X=0.
       IF (HPAGE.EQ.0.) GO TO 330
       NCHAR=IFIX(11.*HPAGE)+2
       IF (NCHAR.GT.80) NCHAR=80
       CALL NOTATE (X,.8,.1,ABC,0.,NCHAR)
       IF (ICUT.NE.0) CALL NOTATE (X,.6,.1,ABCD,0.,NCHAR)
       CALL NOTATE (X,.4,.1,ABCDE,0.,NCHAR)
330    CONTINUE
       Y=FLOAT(IFIX(.5*VPAGE))+1.
       CALL CALPLT (X,Y,-3)
       IF (HPAGE.NE.0.) CALL NOTATE (0.,0.,2.,3,0.,-1)
```

```
      CALL XCUT
      X=0.
      Y=-Y
      CALL CALPLT (0.,0.,3)
      CALL CALPLT (X,Y,-3)
      IF (HPAGE.EQ.0.) GO TO 340
      X=HPAGE+2.
      Y=0.
      CALL CALPLT (X,Y,-3)
      CALL NFRAME
340   CONTINUE
350   CONTINUE
      IF (KODE.EQ.0) GO TO 20
      RETURN
C
C          END OF XPLT
C
      END
```

Subroutine XCUT

Subroutine XCUT sets up the transformation matrix and the origin of the plot coordinate system for the cross-section plots. The flow chart and the FORTRAN statements for this subroutine are as follows:




```
      SUBROUTINE XCUT
C
C          CROSS SECTIONAL OR CONTOUR PLOTS
C
      COMMON/XSECT/ABCDE(8),PPL1(3),PPL2(3),PPL3(3),
     1PLOTSZ,HPAGE,VPAGE,INP,NCU,NOW,ISIDE,IPRIN,
     2KODE,XSTAT,THETR,XMACH
      DIMENSION A(3,3),NAME(2),ABC(8)
      DIMENSION XPRM(3),PNT1(3),PNT2(3),PNT3(3),PNT4(3)
      EQUIVALENCE (PNT1,PPL1),(PNT2,PPL2),(PNT3,PPL3)
C
C           INITIALIZE
C
      SCALE=1./PLOTSZ
      XB=PPL1(1)$YB=PPL1(2)$ZB=PPL1(3)
      IF (INP.EQ.3HANG) GO TO 20
      M=IPCOF(ACOEF,B,C,D,PPL1,PPL2,PPL3)
      IF (M.EQ.1) GO TO 30
```

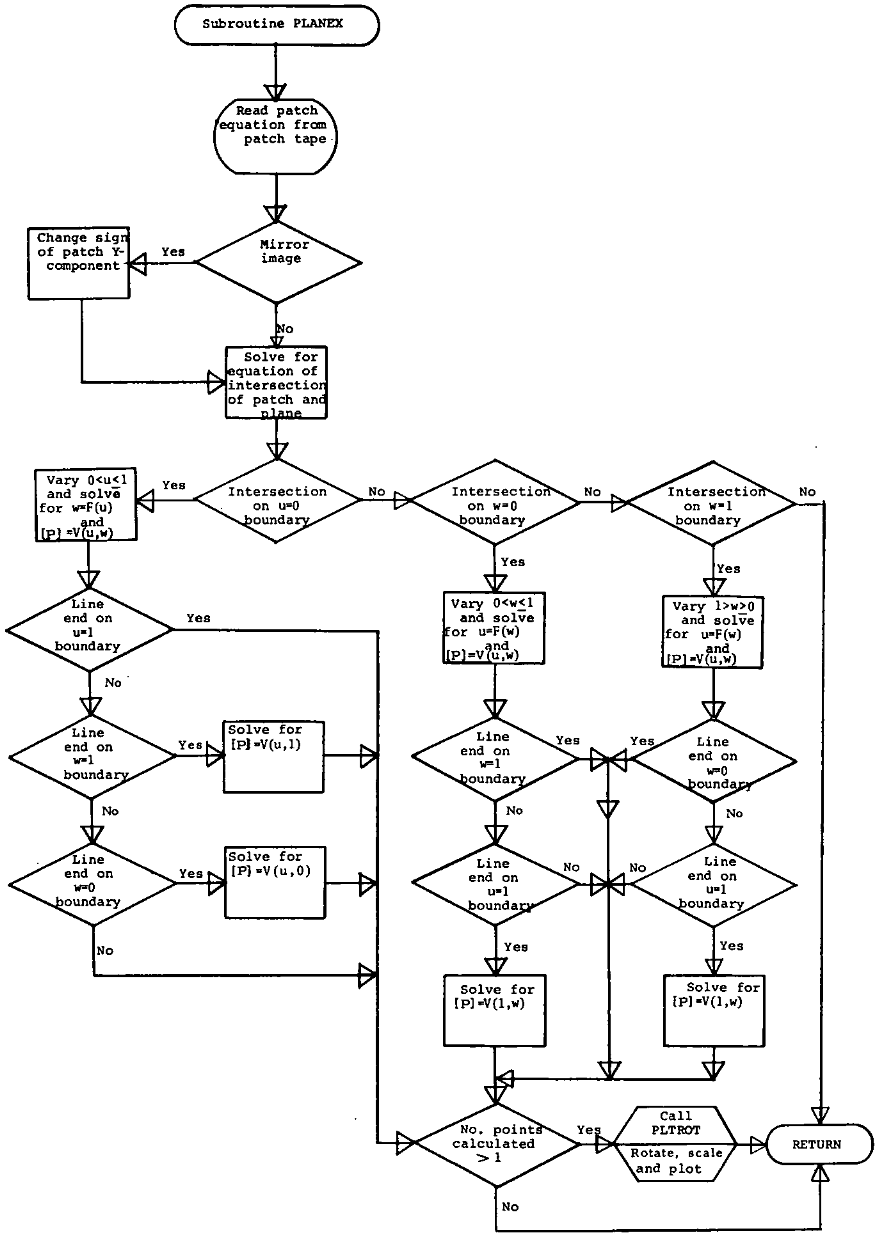
```
      WRITE (6,10)
10    FORMAT (//19H NO PLANE DESCRIBED//)
      RETURN
20    XSTAT=PPL2(1)
      THETR=PPL2(2)*.01745329252
      XMACH=PPL2(3)
      Y=-Y

      THETR=PPL2(2)*.01745329252
      XMACH=PPL2(3)
      BETA=SQRT(XMACH**2-1.)
      ACOEF=1.
      B=-BETA*COS(THETR)
      C=-BETA*SIN(THETR)
      D=XSTAT
      PNT1(1)=XSTAT
      PNT1(2)=0.
      PNT1(3)=0.
      PNT2(2)=PNT3(3)=0.
      PNT2(3)=PNT3(2)=1.
      PNT2(1)=PNT1(1)-C
      PNT3(1)=PNT1(1)-B
30    CONTINUE
      PNT4(1)=PNT3(1)$PNT4(2)=PNT3(2)$PNT4(3)=PNT3(3)
C
C         COMPUTE VECTORS
C
      T1X=PNT3(1)-PNT1(1)$T1Y=PNT3(2)-PNT1(2)$T1Z=PNT3(3)-PNT1(3)
      T2X=PNT4(1)-PNT2(1)$T2Y=PNT4(2)-PNT2(2)$T2Z=PNT4(3)-PNT2(3)
      FNX=T2Y*T1Z-T1Y*T2Z
      FNY=T1X*T2Z-T2X*T1Z
      FNZ=T2X*T1Y-T1X*T2Y
      UN=SQRT(FNX*FNX+FNY*FNY+FNZ*FNZ)
      UNX=FNX/UN
      UNY=FNY/UN
      UNZ=FNZ/UN
      UT1=SQRT(T1X*T1X+T1Y*T1Y+T1Z*T1Z)
      UT1X=T1X/UT1
      UT1Y=T1Y/UT1
      UT1Z=T1Z/UT1
      UT2X=UNY*UT1Z-UNZ*UT1Y
      UT2Y=UNZ*UT1X-UNX*UT1Z
      UT2Z=UNX*UT1Y-UNY*UT1X
C
C         SETUP TRANSFORMATION MATRIX
C
      A(1,1)=UNX$A(1,2)=UNY$A(1,3)=UNZ
      A(2,1)=UT1X$A(2,2)=UT1Y$A(2,3)=UT1Z
      A(3,1)=UT2X$A(3,2)=UT2Y$A(3,3)=UT2Z
C
C         SET ORIGIN OF NEW COORD. SYSTEM
C
      XPRM(1)=A(1,1)*XB+A(1,2)*YB+A(1,3)*ZB
      XPRM(2)=A(2,1)*XB+A(2,2)*YB+A(2,3)*ZB
      XPRM(3)=A(3,1)*XB+A(3,2)*YB+A(3,3)*ZB
      HMIN=0.
      VMIN=0.
      WRITE (6,40) ACOEF,B,C,D
```

```
40    FORMAT (33H EQUATION OF THE PLANE AX+BY+CZ=D/3H A=E15.8,4X,2HB=,E1
     15.8,4X,2HC=,E15.8,4X,2HD=,E15.8)
C
C          BEGIN COMPUTING AND PLOTTING LINES OF INTERSECTION
C
      DO 80 ISI=1,ISIDE
      REWIND 7
      CALL RECIN (7,2,IC,ABC,1,8,1)
      CALL RECIN (7,1,IC,H1,H2,H3,H4,H5,H6,NOBJ)
      DO 70 J=1,NOBJ
      CALL RECIN (7,1,IC,NSURF,JJ,NAME(1),NAME(2),JJ,JJ)
      WRITE (6,50) NAME
50    FORMAT (1X2A10)
      DO 60 N=1,NSURF
      CALL RECIN (7,1,IC,ND1,NS1,J3,J4,J5)
      CALL PLANEX (ND1,NS1,ISI,ACOEF,B,C,D,HMIN,VMIN,A,SCALE,XPRM,NAME)
60    CONTINUE
70    CONTINUE
80    CONTINUE
      RETURN
C
C          END OF XCUT
C
      END
```

Subroutine PLANEX

Subroutine PLANEX reads patch equations from tape and finds the patch equation of the line of intersection of a plane and the patch. A line of intersection through the patch is computed if such a line exists. The flow chart and the FORTRAN statements for this subroutine are as follows:



```
      SUBROUTINE PLANEX (ND1,NS1,ISI,ACOEF,B,C,D,HMIN,VMIN,A,SCALE,XPRM,
     1NAME)
C
C          READS PATCHES FROM TAPE,
C          SOLVES FOR INTERSECTIONS BY SPECIFIED PLANE
C          AND PLOTS
C
      DIMENSION PATCH(4,4,3),A(3,3),ALINE(52,3),
     1GMAT(4,4),GVEC(4),VEC(4,3)
      DIMENSION XPRM(3),NAME(2)
C
      COMMON/XSECT/ABCDE(8),PPL1(3),PPL2(3),PPL3(3),
     1PLOTSZ,HPAGE,VPAGE,INP,NOU,NOW,ISIDE,IPRIN,
     2KODE,XSTAT,THETR,XMACH
C
      NNU=NOU+2
      NNW=NOW+2
      FU=NOU+1
      FW=NOW+1
      DU=1./FU
      DW=1./FW
      NPAT=ND1*NS1
      DO 370 NNPAT=1,NPAT
      CALL RECIN (7,2,IC,PATCH,1,48,1)
      IF (ISI.EQ.1) GO TO 20
C
C          CHANGE Y SIGN
C
      DO 10 I4=1,4
      DO 10 J4=1,4
      PATCH(I4,J4,2)=-PATCH(I4,J4,2)
10    CONTINUE
20    CONTINUE
C
C          SOLVE FOR G MATRIX
C
      DO 30 N4=1,4
      DO 30 I4=1,4
      GMAT(N4,I4)=0.
      GMAT(N4,I4)=ACOEF*PATCH(N4,I4,1)+B*PATCH(N4,I4,2)+C*PATCH(N4,I4,3)
30    CONTINUE
C
C          SOLVE FOR W WITH U=0.
C
      U=0.
      DO 40 I=1,4
40    GVEC(I)=GMAT(4,I)
      GVEC(4)=GVEC(4)-D
      GUESS=0.
      IKODE=KUBSOL(GUESS,W,GVEC)
      GO TO (50,150,150), IKODE
50    NPT=1
      CALL VSOLV (X,Y,Z,U,W,PATCH)
      ALINE(NPT,1)=X$ALINE(NPT,2)=Y$ALINE(NPT,3)=Z
      DO 140 N=2,NNU
      E=N-1
      U=E*DU
```

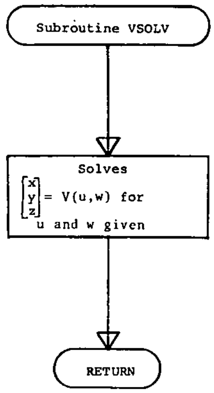
```
C
C          SOLVE FOR CUBIC IN W
C
      DO 60 J4=1,4
      GVEC(J4)=((U*GMAT(1,J4)+GMAT(2,J4))*U+GMAT(3,J4))*U+GMAT(4,J4)
60    CONTINUE
      GVEC(4)=GVEC(4)-D
      GUESS=W
      IKODE=KUBSOL(GUESS,W,GVEC)
      GO TO (130,70,70), IKODE
C
C          TRY U FOR W=1. ON OUTER BOUNDARY
C
70    W=1.
      DO 90 I=1,4
      GVEC(I)=0.
      DO 80 J=1,4
80    GVEC(I)=GVEC(I)+GMAT(I,J)
90    CONTINUE
      GVEC(4)=GVEC(4)-D
      GUESS=1.
      IKODE=KUBSOL(GUESS,U,GVEC)
      GO TO (120,100,100), IKODE
C
C          TRY U FOR W=0. ON OUTER BOUNDARY
C
100   W=0.
      DO 110 I=1,4
110   GVEC(I)=GMAT(I,4)
      GVEC(4)=GVEC(4)-D
      GUESS=1.
      IKODE=KUBSOL(GUESS,U,GVEC)
      GO TO (120,360,360), IKODE
120   CALL VSOLV (X,Y,Z,U,W,PATCH)
      NPT=NPT+1
      ALINE(NPT,1)=X$ALINE(NPT,2)=Y$ALINE(NPT,3)=Z
      GO TO 360
130   CALL VSOLV (X,Y,Z,U,W,PATCH)
      NPT=NPT+1
      ALINE(NPT,1)=X$ALINE(NPT,2)=Y$ALINE(NPT,3)=Z
140   CONTINUE
      GO TO 360
C
C          SOLVE FOR U WITH W=0.
C
150   W=0.
      DO 160 I=1,4
160   GVEC(I)=GMAT(I,4)
      GVEC(4)=GVEC(4)-D
      GUESS=0.
      IKODE=KUBSOL(GUESS,U,GVEC)
      GO TO (170,250,250), IKODE
170   NPT=1
      CALL VSOLV (X,Y,Z,U,W,PATCH)
      ALINE(NPT,1)=X$ALINE(NPT,2)=Y$ALINE(NPT,3)=Z
      DO 240 N=2,NNW
      E=N-1
      W=E*DW
```

```
C
C         SOLVE FOR CUBIC IN U
C
      DO 180 J4=1,4
      GVEC(J4)=((W*GMAT(J4,1)+GMAT(J4,2))*W+GMAT(J4,3))*W+GMAT(J4,4)
180   CONTINUE
      GVEC(4)=GVEC(4)-D
      GUESS=U
      IKODE=KUBSOL(GUESS,U,GVEC)
      GO TO (230,190,190), IKODE
C
C         TRY W FOR U=1. ON OUTER BOUNDARY
C
190   U=1.
      DO 210 I=1,4
      GVEC(I)=0.
      DO 200 J=1,4
200   GVEC(I)=GVEC(I)+GMAT(J,I)
210   CONTINUE
      GVEC(4)=GVEC(4)-D
      GUESS=1.
      IKODE=KUBSOL(GUESS,W,GVEC)
      GO TO (220,360,360), IKODE
220   CALL VSOLV (X,Y,Z,U,W,PATCH)
      NPT=NPT+1
      ALINE(NPT,1)=X$ALINE(NPT,2)=Y$ALINE(NPT,3)=Z
      GO TO 360
230   CALL VSOLV (X,Y,Z,U,W,PATCH)
      NPT=NPT+1
      ALINE(NPT,1)=X$ALINE(NPT,2)=Y$ALINE(NPT,3)=Z
240   CONTINUE
      GO TO 360
C
C         SOLVE FOR U WITH W=1.
C
250   W=1.
      DO 270 I=1,4
      GVEC(I)=0.
      DO 260 J=1,4
260   GVEC(I)=GVEC(I)+GMAT(I,J)
270   CONTINUE
      GVEC(4)=GVEC(4)-D
      GUESS=0.
      IKODE=KUBSOL(GUESS,U,GVEC)
      GO TO (280,370,370), IKODE
280   NPT=1
      CALL VSOLV (X,Y,Z,U,W,PATCH)
      ALINE(NPT,1)=X$ALINE(NPT,2)=Y$ALINE(NPT,3)=Z
      DO 350 N=2,NNW
      E=N-1
      W=1.-E*DW
C
C         SOLVE FOR CUBIC IN U
C
      DO 290 J4=1,4
      GVEC(J4)=((W*GMAT(J4,1)+GMAT(J4,2))*W+GMAT(J4,3))*W+GMAT(J4,4)
```

```
290    CONTINUE
       GVEC(4)=GVEC(4)-D
       GUESS=U
       IKODE=KUBSOL(GUESS,U,GVEC)
       GO TO (340,300,300), IKODE
C
C           TRY W FOR U=1. ON OUTER BOUNDARY
C
300    U=1.
       DO 320 I=1,4
       GVEC(I)=0.
       DO 310 J=1,4
310    GVEC(I)=GVEC(I)+GMAT(J,I)
320    CONTINUE
       GVEC(4)=GVEC(4)-D
       GUESS=1.
       IKODE=KUBSOL(GUESS,W,GVEC)
       GO TO (330,360,360), IKODE
330    CALL VSOLV (X,Y,Z,U,W,PATCH)
       NPT=NPT+1
       ALINE(NPT,1)=X$ALINE(NPT,2)=Y$ALINE(NPT,3)=Z
       GO TO 360
340    CALL VSOLV (X,Y,Z,U,W,PATCH)
       NPT=NPT+1
       ALINE(NPT,1)=X$ALINE(NPT,2)=Y$ALINE(NPT,3)=Z
350    CONTINUE
C
C           SCALE,PRINT AND PLOT
C
360    CONTINUE
       IF (NPT.LE.1) GO TO 370
       CALL PLTROT (NPT,A,ALINE,HMIN,VMIN,SCALE,IPRIN,XPRM,NAME)
370    CONTINUE
       RETURN
C
C           END OF PLANEX
C
       END
```

Subroutine VSOLV

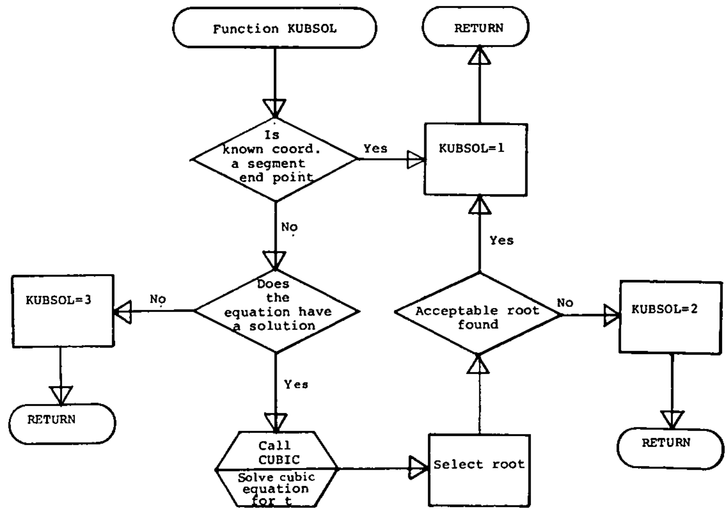
Subroutine VSOLV evaluates a patch equation for x-, y-, and z-coordinates with u and w given. The flow chart and the FORTRAN statements for this subroutine are as follows:




```
      SUBROUTINE VSOLV (X,Y,Z,U,W,PATCH)
C
C         SOLVES FOR V(U,W) FROM PATCH EQUATION
C
      DIMENSION VEC(4,3),PATCH(4,4,3),V(3)
C
      DO 10 J4=1,4
      DO 10 K3=1,3
      VEC(J4,K3)=((U*PATCH(1,J4,K3)+PATCH(2,J4,K3))*U+PATCH(3,J4,K3))*U+
     1PATCH(4,J4,K3)
10    CONTINUE
      DO 20 K3=1,3
      V(K3)=((W*VEC(1,K3)+VEC(2,K3))*W+VEC(3,K3))*W+VEC(4,K3)
20    CONTINUE
      X=V(1)$Y=V(2)$Z=V(3)
      RETURN
      END
```

Function KUBSOL

Function KUBSOL selects the required real root from the roots of a cubic equation. The flow chart and the FORTRAN statements for this function are as follows:

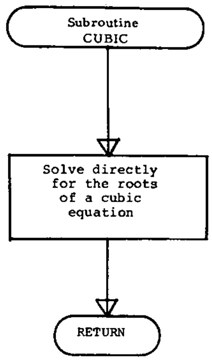



```
      FUNCTION KUBSOL(TL,T,COEFFS)
C
C         FINDS THE ROOTS OF A CUBIC AND SELECTS THE REQUIRED REAL
C         ROOT BETWEEN 0. AND 1. CLOSEST TO A GIVEN ESTIMATE
C
C         THE ROUTINE SETS
C          KUBSOL=1    SUCCESS
C          KUBSOL=2    NO ACCEPTABLE ROOT
C          KUBSOL=3    ERROR
C         TL IS ESTIMATE FOR SELECTION OF REQUIRED ROOT IF ALL ARE REAL
C           AND BETWEEN 0. AND 1.
C         T IS REQUIRED ROOT WITH KUBSOL=1
C         COEFFS(I) ARE THE COEFFICIENTS AT**3+BT**2+CT+D=0.
C
      DIMENSION COEFFS(4)
      COMPLEX ROOTS(3),FTEM(8)
      DATA EPS/1.E-6/,EP1/1.E-5/
C
C         CHECK FOR GIVEN VARIABLE ON SEGMENT END POINTS
C
      T=0.
      IF (ABS(COEFFS(4)).LE.EPS) GO TO 90
      T=1.
      CCC=COEFFS(1)+COEFFS(2)+COEFFS(3)+COEFFS(4)
      IF (ABS(CCC).LE.EPS) GO TO 90
```

```
C
C         SOLVE CUBIC EQUATION FOR T
C
      IF (ABS(COEFFS(1)).GT.EP1) GO TO 40
      IF (ABS(COEFFS(2)).GT.EP1) GO TO 30
      IF (ABS(COEFFS(3)).GT.EP1) GO TO 20
10    KUBSOL=3
      RETURN
20    T=-COEFFS(4)/COEFFS(3)
      ROOTS(1)=ROOTS(2)=ROOTS(3)=T
      IF (T.LT.0..OR.T.GT.1.) GO TO 100
      GO TO 90
30    TEMP=COEFFS(3)**2-4.*COEFFS(2)*COEFFS(4)
      IF (TEMP.LT.0.) GO TO 10
      TMP=SQRT(TEMP)
      ROOTS(1)=ROOTS(2)=(-COEFFS(3)+TMP)/(2.*COEFFS(2))
      ROOTS(3)=(-COEFFS(3)-TMP)/(2.*COEFFS(2))
      GO TO 50
40    CONTINUE
      CALL CUBIC (COEFFS,ROOTS)
C
C         SELECT DESIRED ROOT
C
      TT=REAL(ROOTS(1))
      T=TT
      IF (AIMAG(ROOTS(1)).EQ.0..AND.TT.GE.0..AND.TT.LE.1.) GO TO 60
50    CONTINUE
      TT=REAL(ROOTS(2))
      T=TT
      IF (AIMAG(ROOTS(2)).EQ.0..AND.TT.GE.0..AND.TT.LE.1.) GO TO 80
      TT=REAL(ROOTS(3))
      T=TT
      IF (AIMAG(ROOTS(3)).EQ.0..AND.TT.GE.0..AND.TT.LE.1.) GO TO 90
      GO TO 100
60    IF (AIMAG(ROOTS(2)).NE.0.) GO TO 90
      TT=REAL(ROOTS(2))
      IF (TT.LT.0..OR.TT.GT.1.) GO TO 80
      PRINT 70, T,TT,TL,COEFFS,ROOTS
70    FORMAT (//40H TWO ACCEPTABLE ROOTS FOUND, ROOTS ARE  ,2E17.8/34H E
     1STIMATE FOR SELECTION OF ROOTS  ,E17.8/15H COEFFICIENTS =,4E17.8/8
     2H ROOTS =,6E17.8//)
      IF (ABS(TL-TT).LT.ABS(TL-T)) T=TT
80    IF (AIMAG(ROOTS(3)).NE.0.) GO TO 90
      TT=REAL(ROOTS(3))
      IF (TT.LT.0..OR.TT.GT.1.) GO TO 90
      PRINT 70, T,TT,TL,COEFFS,ROOTS
      IF (ABS(TL-TT).LT.ABS(TL-T)) T=TT
90    CONTINUE
      KUBSOL=1
      RETURN
100   CONTINUE
      KUBSOL=2
      RETURN
      END
```

Subroutine CUBIC

Subroutine CUBIC uses the direct solution for the roots of the cubic equation: $AX^3 + BX^2 + CX + D = 0$. The flow chart and the FORTRAN statements for this subroutine are as follows:

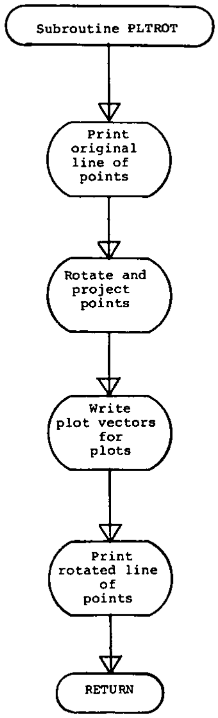



```
      SUBROUTINE CUBIC (A,X)
C
C
C         PROGRAMMER  HSING-SHIH, CHANG
C
C
      COMPLEX X
      DIMENSION X(3)
      DIMENSION A(4),XR(3), XI(3),AQ(3)
C
C         SOLVE THE CUBIC EQUATION F(X)=0
C
C         F(X)=A(1)*X**3+A(2)*X**2+A(3)*X+A(4)
C
C         QUADRATOON F(X)=0
C
C         F(X)=AQ(1)*X**2+AQ(2)*X+AQ(3)
C
C         F(X)=X**2+B2*X+B3
      IPATH=2
      EX=1./3.
      IF (A(4)) 20,10,20
10    XR(1)=0.
      GO TO 150
20    A2=A(1)*A(1)
      Q=(27.*A2*A(4)-9.*A(1)*A(2)*A(3)+2.*A(2)**3)/(54.*A2*A(1))
      IF (Q) 40,30,50
30    Z=0.
      GO TO 140
40    Q=-Q
      IPATH=1
50    P=(3.*A(1)*A(3)-A(2)*A(2))/(9.*A2)
      ARG=P*P*P+Q*Q
```

```
      IF (ARG) 60,70,80
60    Z=-2.*SQRT(-P)*COS(ATAN(SQRT(-ARG)/Q)/3.)
      GO TO 120
70    Z=-2.*Q**EX
      GO TO 120
80    SARG=SQRT(ARG)
      IF (P) 90,100,110
90    Z=-(Q+SARG)**EX-(Q-SARG)**EX
      GO TO 120
100   Z=-(2.*Q)**EX
      GO TO 120
110   Z=(SARG-Q)**EX-(SARG+Q)**EX
120   GO TO (130,140), IPATH
130   Z=-Z
140   XR(1)=(3.*A(1)*Z-A(2))/(3.*A(1))
150   AQ(1)=A(1)
      AQ(2)=A(2)+XR(1)*A(1)
      AQ(3)=A(3)+XR(1)*AQ(2)
C
      B2=AQ(2)/AQ(1)
      B3=AQ(3)/AQ(1)
      X1=-B2/2.
      DISC=X1*X1-B3
      IF (DISC.LT.0.0) 160,170
160   X2=SQRT(-DISC)
      XR(2)=X1
      XR(3)=X1
      XI(2)=X2
      GO TO 200
170   IF (DISC.EQ.0.0) 180,190
180   X2=0.
      XR(2)=X1+X2
      XR(3)=X1-X2
      XI(2)=0.
      GO TO 200
190   X2=SQRT(DISC)
      XR(2)=X1+X2
      XR(3)=X1-X2
      XI(2)=0.
200   XI(1)=0.
      XI(3)=-XI(2)
      X(1)=CMPLX(XR(1),XI(1))
      X(2)=CMPLX(XR(2),XI(2))
      X(3)=CMPLX(XR(3),XI(3))
      RETURN
      END
```

Subroutine PLTROT

Subroutine PLTROT rotates and translates points defining a cross section, generates plot instructions, and prints the points. The flow chart and the FORTRAN statements for this subroutine are as follows:

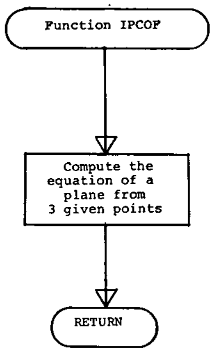



```
      SUBROUTINE PLTROT (NPT,A,ALINE,HMIN,VMIN,SCALE,IPRIN,XP,NAME)
C
C         ROTATES A SET OF 3D POINTS INTO A SPECIFIED PLANE
C         AND GENERATES A CALCOMP PLOT TAPE
C
      DIMENSION A(3,3),ALINE(52,3),RLINE(54,3)
      DIMENSION XP(3),NAME(2)
      DATA EPS/.00000001/
C
      N=1
10    N=N+1
      T1=ABS(ALINE(N-1,1)-ALINE(N,1))
      T2=ABS(ALINE(N-1,2)-ALINE(N,2))
      T3=ABS(ALINE(N-1,3)-ALINE(N,3))
      IF (.NOT.(T1.LE.EPS.AND.T2.LE.EPS.AND.T3.LE.EPS)) GO TO 30
      DO 20 NK=N,NPT
      DO 20 N3=1,3
20    ALINE(NK-1,N3)=ALINE(NK,N3)
      NPT=NPT-1
      N=N-1
```

```
30    IF (N.NE.NPT) GO TO 10
      IF (NPT.LE.1) RETURN
      GO TO (70,40,70,40), IPRIN
40    CONTINUE
      DO 50 N=1,NPT
      DO 50 N3=1,3
50    RLINE(N,N3)=ALINE(N,N3)/SCALE
      WRITE (6,60) ((ALINE(I,J),J=1,3),(RLINE(I,J),J=1,3),I=1,NPT)
60    FORMAT (//44X,37HCOORDINATES OF POINTS OF INTERSECTION/27X,8HORIGI
     1NAL52X,6HSCALED//7X,1HX19X,1HY19X,1HZ19X,1HX19X,1HY19X,1HZ/(6E20.8
     2))
70    CONTINUE
      DO 100 N=1,NPT
      RLINE(N,1)=0.
      RLINE(N,2)=0.
      RLINE(N,3)=0.
      DO 90 I=1,3
      DO 80 J=1,3
80    RLINE(N,I)=RLINE(N,I)+A(I,J)*ALINE(N,J)
90    RLINE(N,I)=RLINE(N,I)-XP(I)
100   CONTINUE
      RLINE(NPT+1,2)=HMIN
      RLINE(NPT+1,3)=VMIN
      RLINE(NPT+2,2)=SCALE
      RLINE(NPT+2,3)=SCALE
      CALL LINE (RLINE(1,2),RLINE(1,3),NPT,1,0,0,0)
      GO TO (150,150,110,110), IPRIN
110   CONTINUE
      DO 130 N=1,NPT
      DO 120 N3=1,3
120   ALINE(N,N3)=RLINE(N,N3)/SCALE
130   CONTINUE
      WRITE (6,140) ((RLINE(I,J),J=1,3),(ALINE(I,J),J=1,3),I=1,NPT)
140   FORMAT (//33X,59HCOORDINATES OF POINTS OF INTERSECTION ROTATED INT
     1O YZ PLANE/27X,8HORIGINAL52X,6HSCALED//7X,1HX19X,1HY19X,1HZ19X,1HX
     219X,1HY19X,1HZ/(6E20.8))
150   CONTINUE
      RETURN
C
C          END OF PLTROT
C
      END
```

Function IPCOF

Function IPCOF finds the coefficients of a plane from three given points. The flow chart and the FORTRAN statements for this function are as follows:




```
      FUNCTION IPCOF (A,B,C,D,P1,P2,P3)
      DIMENSION  P1(3),P2(3),P3(3),AMAT(3,3),BMAT(3),IPIVOT(3)
C
C         FINDS THE COEFFICIENTS OF A PLANE IN THE
C         FORM AX+BY+CZ=D
C
      DATA EPS/1.E-8/
C
      D=1.
      DO 50 I=1,4
      DO 10 N3=1,3
      AMAT(1,N3)=P1(N3)
      AMAT(2,N3)=P2(N3)
      AMAT(3,N3)=P3(N3)
10    BMAT(N3)=1.
      II=I-1
      IF (II.EQ.0) GO TO 30
      DO 20 N3=1,3
      BMAT(N3)=-AMAT(N3,II)
      AMAT(N3,II)=-1.
20    CONTINUE
30    CONTINUE
      CALL SIMEQ (AMAT,3,BMAT,1,DET,IPIVOT,3,ISCALE)
      IF (ABS(DET).LE.EPS) GO TO 50
      IF (I.EQ.1) GO TO 40
      D=BMAT(II)
      BMAT(II)=1.
40    A=BMAT(1)
      B=BMAT(2)
      C=BMAT(3)
      IPCOF=1
      RETURN
50    CONTINUE
      IPCOF=2
      RETURN
C
C     ----END OF IPCOF------------------------------------
C
      END
```

Langley Library Subroutine SIMEQ

Language: FORTRAN

Purpose: SIMEQ solves the matrix equation $AX = B$ where A is a square coefficient matrix and B is a matrix of constant vectors. The solution to a set of simultaneous equations and the determinant may be obtained. If the user wants the determinant only, use DETEV for savings in time and storage.

Use: CALL SIMEQ (A, N, B, M, DETERM, IPIVOT, NMAX, ISCALE)

A	A two-dimensional array of the coefficients.
N	The order of A; $1 \leqq N \leqq NMAX$.
B	A two-dimensional array of the constant vectors B. On return to calling program, X is stored in B.
M	The number of column vectors in B.
DETERM	Gives the value of the determinant by the following formula: $DET(A) = \left(10^{100}\right)^{ISCALE}(DETERM)$
IPIVOT	A one-dimensional array of temporary storage used by the routine.
NMAX	The maximum order of A as stated in dimension statement of calling program.
ISCALE	A scale factor computed by subroutine to keep results of computation within the floating-point word size of the computer.

Restrictions: Arrays A, B, and IPIVOT are dimensioned with variable dimensions in the subroutine. The maximum size of these arrays must be specified in a DIMENSION statement of the calling program as: A (NMAX, NMAX), B (NMAX, M), IPIVOT (NMAX). The original matrices, A and B, are destroyed. They must be saved by the user if there is further need for them. The determinant is set to zero for a singular matrix.

Method: Jordan's method is used through a succession of elementary transformations: $l_n, l_{n-1}, \ldots, l_1$. If these transformations are applied to a matrix B of constant vectors, the result is X where $AX = B$. Each transformation is selected so that the largest element is used in the pivotal position.

Accuracy: Total pivotal strategy is used to minimize the rounding errors; however, the accuracy of the final results depends upon how well-conditioned the original matrix is.

Reference: (a) Fox, L.: An Introduction to Numerical Linear Algebra. Oxford Univ. Press, c.1965.

Storage: 432_8 locations.

Subroutine date: August 1, 1968.

The FORTRAN statements for this subroutine are as follows:

```
      SUBROUTINE SIMEQ (A,N,B,M,DETERM,IPIVOT,NMAX,ISCALE)
C     SOLUTION OF SIMULTANEOUS LINEAR EQUATIONS
C     *** DOCUMENT DATE 08-01-68   SUBROUTINE REVISED 08-01-68 *********
C
      DIMENSION IPIVOT(N),A(NMAX,N),B(NMAX,M)
      EQUIVALENCE (IROW,JROW),(ICOLUM,JCOLUM),(AMAX,T,SWAP)
C
C     INITIALIZATION
C
10    ISCALE=0
      R1=10.0**100
      R2=1.0/R1
      DETERM=1.0
      DO 20 J=1,N
20    IPIVOT(J)=0
      DO 380 I=1,N
C
C     SEARCH FOR PIVOT ELEMENT
C
      AMAX=0.0
      DO 70 J=1,N
      IF (IPIVOT(J)-1) 30,70,30
30    DO 60 K=1,N
      IF (IPIVOT(K)-1) 40,60,390
40    IF (ABS(AMAX)-ABS(A(J,K))) 50,60,60
50    IROW=J
      ICOLUM=K
      AMAX=A(J,K)
60    CONTINUE
70    CONTINUE
      IF (AMAX) 90,80,90
80    DETERM=0.0
      ISCALE=0
      GO TO 390
90    IPIVOT(ICOLUM)=IPIVOT(ICOLUM)+1
C
C     INTERCHANGE ROWS TO PUT PIVOT ELEMENT ON DIAGONAL
C
      IF (IROW-ICOLUM) 100,140,100
100   DETERM=-DETERM
      DO 110 L=1,N
      SWAP=A(IROW,L)
      A(IROW,L)=A(ICOLUM,L)
110   A(ICOLUM,L)=SWAP
      IF (M) 140,140,120
120   DO 130 L=1,M
      SWAP=B(IROW,L)
      B(IROW,L)=B(ICOLUM,L)
130   B(ICOLUM,L)=SWAP
140   PIVOT=A(ICOLUM,ICOLUM)
      IF (PIVOT) 150,80,150
C
C     SCALE THE DETERMINANT
C
150   PIVOTI=PIVOT
      IF (ABS(DETERM)-R1) 180,160,160
```

```
160   DETERM=DETERM/R1
      ISCALE=ISCALE+1
      IF (ABS(DETERM)-R1) 210,170,170
170   DETERM=DETERM/R1
      ISCALE=ISCALE+1
      GO TO 210
180   IF (ABS(DETERM)-R2) 190,190,210
190   DETERM=DETERM*R1
      ISCALE=ISCALE-1
      IF (ABS(DETERM)-R2) 200,200,210
200   DETERM=DETERM*R1
      ISCALE=ISCALE-1
210   IF (ABS(PIVOTI)-R1) 240,220,220
220   PIVOTI=PIVOTI/R1
      ISCALE=ISCALE+1
      IF (ABS(PIVOTI)-R1) 270,230,230
230   PIVOTI=PIVOTI/R1
      ISCALE=ISCALE+1
      GO TO 270
240   IF (ABS(PIVOTI)-R2) 250,250,270
250   PIVOTI=PIVOTI*R1
      ISCALE=ISCALE-1
      IF (ABS(PIVOTI)-R2) 260,260,270
260   PIVOTI=PIVOTI*R1
      ISCALE=ISCALE-1
270   DETERM=DETERM*PIVOTI
C
C     DIVIDE PIVOT ROW BY PIVOT ELEMENT
C
      DO 290 L=1,N
      IF (IPIVOT(L)-1) 280,290,390
280   A(ICOLUM,L)=A(ICOLUM,L)/PIVOT
290   CONTINUE
      IF (M) 320,320,300
300   DO 310 L=1,M
310   B(ICOLUM,L)=B(ICOLUM,L)/PIVOT
C
C     REDUCE NON-PIVOT ROWS
C
320   DO 380 L1=1,N
      IF (L1-ICOLUM) 330,380,330
330   T=A(L1,ICOLUM)
      DO 350 L=1,N
      IF (IPIVOT(L)-1) 340,350,390
340   A(L1,L)=A(L1,L)-A(ICOLUM,L)*T
350   CONTINUE
      IF (M) 380,380,360
360   DO 370 L=1,M
370   B(L1,L)=B(L1,L)-B(ICOLUM,L)*T
380   CONTINUE
390   RETURN
      END
```

PROGRAM USE

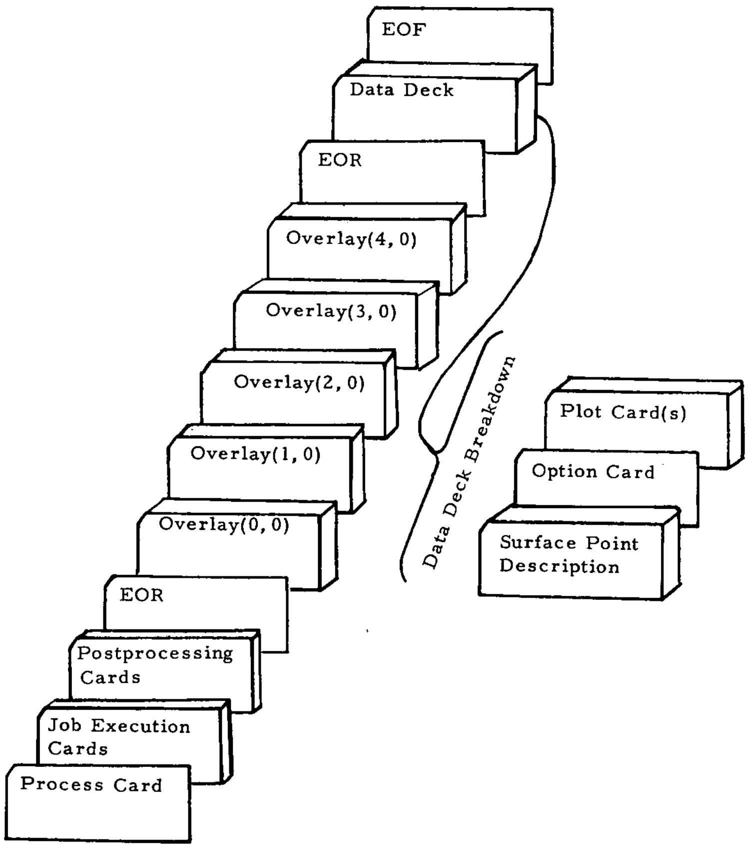
PROGRAM IDENTIFICATION

This program is for fitting smooth surfaces to the component parts of an aircraft configuration using a three-dimensional modeling technique called Coon's patches (ref. 1). It is identified as program D3400.

PROGRAM SETUP FOR A COMPILE AND EXECUTE

This section describes the input data requirements, limitations, and the punched card formats. The program will end normally if there are no input cards at the beginning of a READ sequence.

The input data cards are assembled with the program decks in the order illustrated in sketch (d).



Sketch (d)

DESCRIPTION OF INPUT DATA CARDS

Configuration

The form for the airplane configuration input has become known throughout the aircraft industry as the Harris Wave Drag geometry input and is identical to that described in reference 3.

Since the airplane has to be symmetrical about the XZ-plane, only half of the airplane need be described to the computer. The convention used in presenting the input data is that the half of the airplane on the positive Y-side of the XZ-plane is presented. The program then uses this information to construct the complete airplane if required. The number of input cards depends on the number of components used to describe the configuration and the amount of detail used to describe each component. It is not possible to change only part of a configuration in a succeeding case. The complete configuration must be input each time. The method of input is by FORTRAN "READ" statements.

Card 1 – Identification.- Card 1 contains any desired identifying information in columns 1 to 80.

Card 2 – Control integers.- Card 2 contains 24 integers, each punched right-justified in a 3-column field. Columns 73 to 80 may be used in any desired manner. An identification of the card columns, the name used by the source program, and a description of each integer are given in the following table:

Columns	FORTRAN name	Description
01 to 03	J0	If J0 = 0, no reference area If J0 = 1, reference area to be read
04 to 06	J1	If J1 = 0, no wing data If J1 = 1, cambered wing data to be read If J1 = -1, uncambered wing data to be read
07 to 09	J2	If J2 = 0, no fuselage data If J2 = 1, data for arbitrarily shaped fuselage to be read If J2 = -1, data for circular fuselage to be read (with J6 = 0, fuselage will be cambered; with J6 = -1, fuselage will be symmetrical with XY-plane; with J6 = 1, entire configuration will be symmetrical with XY-plane)

Columns	FORTRAN name	Description
10 to 12	J3	If J3 = 0, no pod data If J3 = 1, pod data to be read
13 to 15	J4	If J4 = 0, no fin data If J4 = 1, fin data to be read
16 to 18	J5	If J5 = 0, no canard data If J5 = 1, canard data to be read
19 to 21	J6	Simplification code: If J6 = 0, indicates a cambered circular or arbitrary fuselage if J2 $\neq$ 0 If J6 = 1, complete configuration is symmetrical with respect to XY-plane, which implies uncambered circular fuselage if there is a fuselage If J6 = -1, indicates uncambered circular fuselage with J2 $\neq$ 0
22 to 24	NWAF	Number of airfoil sections used to describe the wing; $4 \leqq$ NWAF $\leqq 20$
25 to 27	NWAFOR	Number of ordinates used to define each wing airfoil section; $4 \leqq$ NWAFOR $\leqq 30$
28 to 30	NFUS	Number of fuselage segments; $1 \leqq$ NFUS $\leqq 4$
31 to 33	NRADX(1)	Number of points used to represent half-section of first fuselage segment; if fuselage is circular, the program computes indicated number of y- and z-ordinates; $4 \leqq$ NRADX(1) $\leqq 30$
34 to 36	NFORX(1)	Number of stations for first fuselage segment; $4 \leqq$ NFORX(1) $\leqq 30$
37 to 39 40 to 42	NRADX(2) NFORX(2)	Same as NRADX(1) and NFORX(1), but for second fuselage segment
43 to 45 46 to 48	NRADX(3) NFORX(3)	Same as NRADX(1) and NFORX(1), but for third fuselage segment
49 to 51 52 to 54	NRADX(4) NFORX(4)	Same as NRADX(1) and NFORX(1), but for fourth fuselage segment

Columns	FORTRAN name	Description
55 to 57	NP	Number of pods described; NP $\leqq$ 9
58 to 60	NPODOR	Number of stations at which pod radii are to be specified; 4 $\leqq$ NPODOR $\leqq$ 30
61 to 63	NF	Number of fins (vertical tails) described; NF $\leqq$ 6
64 to 66	NFINOR	Number of ordinates used to define each fin airfoil section; 4 $\leqq$ NFINOR $\leqq$ 10
67 to 69	NCAN	Number of canards (horizontal tails) described NCAN $\leqq$ 2
70 to 72	NCANOR	Number of ordinates used to define each canard airfoil section; 4 $\leqq$ NCANOR $\leqq$ 10; if NCANOR is given a negative sign, the program will expect to read lower ordinates also; otherwise, airfoil is assumed to be symmetrical

Cards 3, 4, . . . – remaining data input cards.- The remaining data input cards contain a detailed description of each component of the airplane. Each card contains up to 10 values, each value punched in a 7-column field with a decimal and may be identified in columns 73 to 80. The cards are arranged in the following order: reference area, wing data cards, fuselage data cards, pod (or nacelle) data cards, fin (vertical tail) data cards, and canard (or horizontal tail) data cards.

Reference area card: The reference area value is punched in columns 1 to 7 and may be identified as REFA in columns 73 to 80. This value is not used by the program but may be present in an already existing input deck.

Wing data cards: The first wing data card (or cards) contains the locations in percent chord at which the ordinates of all the wing airfoils are to be specified. There will be exactly NWAFOR locations in percent chord given. Each card may be identified in columns 73 to 80 by the symbol XAFj where j denotes the number of the last location in percent chord given on that card. For example, if NWAFOR = 16, there are 16 ordinates to be specified for every airfoil, and two data cards will be required. The first XAF card is identified as XAF 10 and the second as XAF 16.

The next wing data cards (there will be NWAF cards) each contain four numbers which give the origin and chord length of each of the wing airfoils that is to be specified. The cards representing the most inboard airfoil are given first, followed by the cards for successive airfoils. The information is arranged on each card as follows:

Columns	Description
1 to 7	x-ordinate of airfoil leading edge
8 to 14	y-ordinate of airfoil leading edge
15 to 21	z-ordinate of airfoil leading edge
22 to 28	Airfoil streamwise chord length
73 to 80	Card identification, WAFORGj where j denotes the particular airfoil; for example, WAFORG1 denotes first (most inboard) airfoil

If a cambered wing has been specified, the next set of wing data cards is the mean camber line (TZORD) cards. The first card contains up to 10 Δz values, referenced to the z-ordinate of the airfoil leading edge, at each of the specified percents of chord for the first airfoil. If more than 10 values are to be specified for each airfoil (there will be NWAFOR values), the remaining values are continued on successive cards. The remaining airfoils are described in the same manner, data for each airfoil starting on a new card, and the cards arranged in the order which begins with the most inboard airfoil and proceeds to the outboard. Each card may be identified in columns 73 to 80 as TZORDj, where j denotes the particular airfoil.

Next are the wing airfoil ordinate (WAFORD) cards. The first card contains up to 10 half-thickness ordinates of the first airfoil expressed as percent chord. If more than 10 ordinates are to be specified for each airfoil (there will be NWAFOR values), the remaining ordinates are continued on successive cards. The remaining airfoils are each described in the same manner, and the cards are arranged in the order which begins with the most inboard airfoil and proceeds to the outboard. Each card may be identified in columns 73 to 80 as WAFORDj, where j denotes the particular airfoil.

Fuselage data cards: The first card (or cards) specifies the x-values of the fuselage stations of the first segment. There will be NFORX(1) values and the cards may be identified in columns 73 to 80 by the symbol XFUSj where j denotes the number of the last fuselage station given on that card.

If the fuselage is circular and cambered, the next set of cards specifies the z-locations of the center of the circular sections. There will be NFORX(1) values and the cards may be identified in columns 73 to 80 by the symbol ZFUSj where j denotes the number of the last fuselage station given on that card.

If the fuselage is circular, the next card (or cards) gives the fuselage cross-sectional areas, and may be identified in columns 73 to 80 by the symbol FUSARDj where j denotes the number of the last fuselage station given on that card. If the fuselage is of arbitrary shape, the y-ordinates for a half-section are given (NRADX(1) values) and identified in columns 73 to 80 as Yi where i is the station number. Following these are the corresponding z-ordinates (NRADX(1) values) for the half-section identified in columns 73 to 80 as Zi where i is the station number. Each station will have a set of Y and Z cards, and the convention of ordering the ordinates from bottom to top is observed.

For each fuselage segment a new set of cards as described must be provided. The segment descriptions should be given in order of increasing values of x.

Pod data cards: The first pod or nacelle data card specifies the location of the origin of the first pod. The information is arranged on the card as follows:

Columns	Description
1 to 7	x-ordinate of origin of first pod
8 to 14	y-ordinate of origin of first pod
15 to 21	z-ordinate of origin of first pod
73 to 80	Card identification, PODORGj where j denotes pod number

The next pod input data card (or cards) contains the x-ordinates, referenced to the pod origin, at which the pod radii (there will be NPODOR of them) are to be specified. The first x-value must be zero, and the last x-value is the length of the pod. These cards may be identified in columns 73 to 80 by the symbol XPODj where j denotes the pod number. For example, XPOD1 represents the first pod.

The next pod input data cards give the pod radii corresponding to the pod stations that have been specified. These cards may be identified in columns 73 to 80 as PODRj where j denotes the pod number.

For each additional pod, new PODORG, XPOD, and PODR cards must be provided. Only single pods are described but the program assumes that if the y-ordinate is not zero an exact duplicate is located symmetrically with respect to the XZ-plane; a y-ordinate of zero implies a single pod.

Fin data cards: Exactly three data input cards are used to describe a fin. The information presented on the first fin data input card is as follows:

Columns	Description
1 to 7	x-ordinate of lower airfoil leading edge
8 to 14	y-ordinate of lower airfoil leading edge
15 to 21	z-ordinate of lower airfoil leading edge

Columns	Description
22 to 28	Chord length of lower airfoil
29 to 35	x-ordinate of upper airfoil leading edge
36 to 42	y-ordinate of upper airfoil leading edge
43 to 49	z-ordinate of upper airfoil leading edge
50 to 56	Chord length of upper airfoil
73 to 80	Card identification, FINORGj where j denotes fin number

The second fin data input card contains up to 10 locations in percent chord (exactly NFINOR of them) at which the fin airfoil ordinates are to be specified. The card may be identified in columns 73 to 80 as XFINj where j denotes the fin number.

The third fin data input card contains the fin airfoil half-thickness ordinates expressed in percent chord. Since the fin airfoil must be symmetrical, only the ordinates on the positive y-side of the fin chord plane are specified. The card identification, FINORDj, may be given in columns 73 to 80, where j denotes the fin number.

For each fin, new FINORG, XFIN, and FINORD cards must be provided.

Only single fins are described but the program assumes that if the y-ordinate is not zero an exact duplicate is located symmetrically with respect to the XZ-plane; a y-ordinate of zero implies a single fin.

Canard data cards: If the canard (or horizontal tail) airfoil is symmetrical, exactly three cards are used to describe a canard, and the input is given in the same manner as for the fin. If, however, the canard airfoil is not symmetrical (indicated by a negative value of NCANOR), a fourth canard data input card will be required to give the lower ordinates. The information presented on the first canard data input card is as follows:

Columns	Description
1 to 7	x-ordinate of inboard airfoil leading edge
8 to 14	y-ordinate of inboard airfoil leading edge
15 to 21	z-ordinate of inboard airfoil leading edge
22 to 28	Chord length of inboard airfoil
29 to 35	x-ordinate of outboard airfoil leading edge
36 to 42	y-ordinate of outboard airfoil leading edge
43 to 49	z-ordinate of outboard airfoil leading edge

Columns	Description
50 to 56	Chord length of outboard airfoil
73 to 80	Card identification, CANORGj where j denotes the canard number

The second canard data input card contains up to 10 locations in percent chord (exactly NCANOR of them) at which the canard airfoil ordinates are to be specified. The card may be identified in columns 73 to 80 as XCANj where j denotes the canard number.

The third canard data input card contains the upper half-thickness ordinates, expressed in percent chord, of the canard airfoil. This card may be identified in columns 73 to 80 as CANORDj where j denotes the canard number. If the canard airfoil is not symmetrical, the lower ordinates are presented on a second CANORD card. The program expects both upper and lower ordinates to be punched as positive values in percent chord.

For another canard, new CANORG, XCAN, and CANORD cards must be provided.

Alternate Surface-Description Input

The surface-description method used in this report is not restricted to any particular shape. An airplane configuration was chosen for the current application because of its complexity and an anticipated need. The program input is best thought of as a set of points describing a surface or surfaces.

To use the program for a different input form, the user may substitute another overlay (1,0) for preprocessing input data in the proper form to be used by the program. If desired to print this input data, provisions to do so should be provided for in overlay (1,0).

Overlay (1,0) must:

1. Store in labeled COMMON

    COMMON/PATPLT/XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX, NOBJ

    The minimums and maximums define a box in which the shape to be plotted orthographically will fit. These values are not used for the cross-section plots, but space must be allowed so that NOBJ will occupy the correct position.

    NOBJ is the number of objects (or components) each made up of a number of surfaces, all of which could form a body.

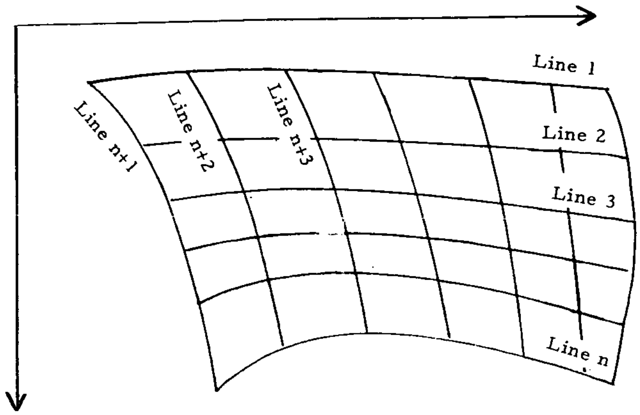
2. Write binary tape 10 in the following format. FORTRAN names from the given program are used for illustration.

Record	Name	Purpose
1	ABC(8)	Identification
2	(For $NOBJ_1$)	
	NSURF	Number of surfaces in object
	M1	Not used
	M2(2)	Name of object (for printing only)
	M3	Not used
	M4	Not used
3	(For $NSURF_1$)	
	NCOL	Number of columns of grid points, ≦31
	NROW	Number of rows of grid points, ≦31
	N3	Not used
	N4	Not used
	N5	Not used
4	((ALINE(N,N3), N=1, NCOL), N3=1,3)	NROW lines of NCOL points (x,y,z)
5	((ALINE(N,N3), N=1, NROW), N3=1,3)	NCOL lines of NROW points (x,y,z)
6	(Repeat records 3, 4, and 5 for the number of surfaces given in record 2)	
.		
.		
.		

(Repeat record 2 for each NOBJ as given in labeled COMMON PATPLT and repeat records 3, 4, and 5 as required.)

(Although the dummy variables are not used at the present time, they must be written on tape.)

Great care must be taken to describe the grid of input points in exactly the manner specified so that in a collection of surfaces the outward normal vectors will be consistent. For the current application, the rectangular grid of values (not necessarily rectangular in shape) does not require any specific corner as the starting place. However, points must be given as lines in a rowwise direction then as lines in a columnwise direction as illustrated in sketch (e)



Sketch (e)

3. If it is desired to increase the number of points in the rows and/or columns and it is not feasible to break a surface down into more than one surface, increase all dimensions of 31 to the maximum of NCOL or NROW in:

   (a) Overlay (1,0), program START

   (b) Overlay (2,0), subroutine PACH. Also in subroutine PACH, change the dimensions of the SLOPE array to SLOPE$\left(NCOL_{max}, NROW_{max}, 3\right)$ and change MAXN in the data statement to the largest of $NCOL_{max}$ or $NROW_{max}$.

Option Card

The option card indicates to the program the next kind of input to be read.

Column	FORTRAN name	Description
4	ITYPE	If ITYPE = 1, Read orthographic-projection plot card
		If ITYPE = 2, Read cross-section plot card
		If ITYPE = 3, Read another set of geometry cards

Plot Cards

A single card contains all the necessary information for one plot. The available options and the necessary input for each are described in the following sections.

Orthographic projections (ITYPE = 1).- For one orthographic projection, the card should be set up as follows:

Columns	FORTRAN name	Description
1	HORZ	"X", "Y", or "Z" for horizontal axis
3	VERT	"X", "Y", or "Z" for vertical axis
5 to 7	TEST1	Word "OUT" for deletion of hidden lines; otherwise, leave blank
8 to 12	PHI	Roll angle, degrees
13 to 17	THETA	Pitch angle, degrees
18 to 22	PSI	Yaw angle, degrees
48 to 52	PLOTSZ	Plot frame size (scale factor is computed by using PLOTSZ and maximum dimension of configuration, i.e., the maximum dimension, usually the body length, will be scaled to exactly PLOTSZ)
53 to 55	TYPE	Word "ORT"
56 to 58	NOU	Number of lines, originating on the $w = 0.0$ boundary, computed within each patch for enriching the surface grid in the w-direction, $NOU \leqq 50$
59 to 61	NOW	Number of lines, originating on the $u = 0.0$ boundary, computed within each patch for enriching the surface grid in the u-direction, $NOW \leqq 50$
64	ISIDE	If ISIDE = 0 or 1, plot described body If ISIDE = 2, plot described body and mirror image
72	KODE	If KODE = 0, continue to read plot cards If KODE $\neq$ 0, return and read option card after this plot

Plan, front, and side views (stacked) (ITYPE = 1).- For plan, front, and side views stacked one above the other in a pleasing-to-the-eye arrangement, the card should be set up as follows:

Columns	FORTRAN name	Description
5 to 7	TEST1	Word "OUT" for deletion of hidden lines; otherwise leave blank

Columns	FORTRAN name	Description
8 to 12	PHI	y-origin on paper of plan view, inches
13 to 17	THETA	y-origin on paper of side view, inches
18 to 22	PSI	y-origin on paper of front view, inches
48 to 52	PLOTSZ	Plot frame size (scale factor is computed using PLOTSZ and maximum dimension of configuration, i.e., the maximum dimension, usually the body length, will be scaled to exactly PLOTSZ)
53 to 55	TYPE	Word "VU3"
56 to 58	NOU	Number of lines, originating on the $w = 0.0$ boundary, computed within each patch for enriching the surface grid in the w-direction, NOU $\leqq$ 50
59 to 61	NOW	Number of lines, originating on the $u = 0.0$ boundary, computed within each patch for enriching the surface grid in the u-direction, NOW $\leqq$ 50
64	ISIDE	If ISIDE = 0 or 1, plot described body If ISIDE = 2, plot described body and mirror image
72	KODE	If KODE = 0, continue to read plot cards If KODE $\neq$ 0, return and read option card after this plot

Cross-section plots where the method of input for defining the plane is by three input points (ITYPE = 2).- For cross-section plots where the method of input for defining the plane is by three input points (ITYPE = 2), the input card should be set up as follows:

Columns	FORTRAN name	Description
1 to 6	PPL1(1)	x_1, also used as x_0
7 to 12	PPL1(2)	y_1, also used as y_0
13 to 18	PPL1(3)	z_1, also used as z_0

Columns	FORTRAN name	Description
19 to 24	PPL2(1)	x_2
25 to 30	PPL2(2)	y_2
31 to 36	PPL2(3)	z_2
37 to 42	PPL3(1)	x_3
43 to 48	PPL3(2)	y_3
49 to 54	PPL3(3)	z_3
55 to 60	PLOTSZ	Scale factor, inches/unit
61 to 63	HPAGE	Horizontal paper origin, inches (a value of 0.0 will overlay the plots)
64 to 66	VPAGE	Vertical paper origin will be half of VPAGE, inches
67 to 69	INP	Word "PNT"
70 to 72	NOU	Number of u-values between 0.0 and 1.0, together with the $u = 0.0$ and $u = 1.0$ values, where interpolated values of w are to be computed, NOU $\leqq 50$
73 to 75	NOW	Number of w-values between 0.0 and 1.0, together with the $w = 0.0$ and $w = 1.0$ values, where interpolated values of u are to be computed, NOW $\leqq 50$
76 to 77	*ICUT	Number of additional cross-section plots desired

*For ICUT $\neq 0$, a second input card is required:

Columns	FORTRAN name	Description
1 to 6	DX	ΔX for x_1, x_2, and x_3
7 to 12	DY	ΔY for y_1, y_2, and y_3
13 to 18	DZ	ΔZ for z_1, z_2, and z_3
21	IH	If IH = 0, plots will be overlaid with HPAGE applied only after last plot If IH $\neq 0$, normal HPAGE spacing between each plot

Columns	FORTRAN name	Description
78	ISIDE	If ISIDE = 0 or 1, examine given body If ISIDE = 2, examine given body and its mirror image
79	IPRIN	If IPRIN = 0 or 1, do not print intersection points If IPRIN = 2, print original and original rotated points of intersection If IPRIN = 3, print scaled and scaled rotated points of intersection If IPRIN = 4, print points of intersection· original and original rotated, scaled and scaled rotated
80	KODE	If KODE = 0, continue to read plot cards If KODE ≠ 0, return and read option card after this plot

Cross-section plots where the method of input for defining the plane is by specifying a Mach number to define a Mach angle, an angle to define the orientation of the Mach angle, and the plane intercept on the X-axis (ITYPE = 2).- For cross-section plots where the method of input for defining the plane is by specifying a Mach number to define a Mach angle, an angle to define the orientation of the Mach angle, and the plane intercept on the X-axis (ITYPE = 2), the input card should be set up as follows:

Columns	FORTRAN name	Description
1 to 6	PPL1(1)	X_0
7 to 12	PPL1(2)	Y_0
13 to 18	PPL1(3)	Z_0
19 to 24	PPL2(1)	X-intercept
25 to 30	PPL2(2)	Roll angle of plane, degrees
31 to 36	PPL2(3)	Mach number, >1.0
55 to 60	PLOTSZ	Scale factor, inches/unit
61 to 63	HPAGE	Horizontal paper origin, inches (a value of 0.0 will overlay the plots)
64 to 66	VPAGE	Vertical paper origin will be half of VPAGE, inches

Columns	FORTRAN name	Description
67 to 69	INP	Word "ANG"
70 to 72	NOU	Number of u-values between 0.0 and 1.0, together with the $u = 0.0$ and $u = 1.0$ values, where interpolated values of w are to be computed, NOU $\leqq 50$
73 to 75	NOW	Number of w-values between 0.0 and 1.0, together with the $w = 0.0$ and $w = 1.0$ values, where interpolated values of u are to be computed, NOW $\leqq 50$
76 to 77	*ICUT	Number of additional cross-section plots desired
78	ISIDE	If ISIDE = 0 or 1, examine given body If ISIDE = 2, examine given body and its mirror image
79	IPRIN	If IPRIN = 0 or 1, do not print intersection points If IPRIN = 2, print original and original rotated points of intersection If IPRIN = 3, print scaled and scaled rotated points of intersection If IPRIN = 4, print points of intersection: original and original rotated, scaled and scaled rotated
80	KODE	If KODE = 0, continue to read plot cards If KODE $\neq$ 0, return and read option card after this plot

*For ICUT $\neq$ 0, a second input card is required:

Columns	FORTRAN name	Description
1 to 6	DX	Δ for X-intercept
7 to 12	DY	Δ for roll angle
13 to 18	DZ	Δ for Mach number
21	IH	If IH = 0, plots will be overlaid with HPAGE applied only after last plot If IH $\neq$ 0, normal HPAGE spacing between each plot

DESCRIPTION OF PROGRAM OUTPUT

The program output includes the input data printout, the printout of points of intersection for cross-section plots, and a plot vector file to be postprocessed for off-line machine plots.

Input Data Printout

The card images of all the input data – configuration description and plot cards – are printed. Cards for a sample case input deck are listed in table I.

Cross-Section Plot Printout

The points of intersection of the cutting plane and the given body may be printed as specified on the cross-section plot card. The actual points in the original coordinate system, the actual points in the scaled coordinate system, the rotated and projected points in the original coordinate system, and the rotated and projected points in the scaled coordinate system may be chosen for printing or printing of the intersection points may be omitted entirely. Since the plane of intersection is always transformed into the YZ-plane, the rotated x-coordinates should be all zeros. The equation of the plane is also printed.

Plot Vector File

A plot vector file is generated during execution of both plot options – three-dimensional orthographic or cross sectional. The plot vector file can be postprocessed on the same computer run or at a later time for the desired plotting device.

MACHINE SETUP

This program was written in FORTRAN Version IV for Control Data series 6000 computer systems with the Scope 3 operating system and library tape as modified for the Langley computer facility. Tape unit 5 is used for input, unit 6 for output, and units 7 and 10 for intermediate storage. Approximately 60000 octal locations of core storage are required and the processing of information for one plot is less than 1 minute of computer time.

OPERATIONAL DETAILS

The graphic output system at Langley Research Center is in two parts: (1) a device independent graphic language which produces a plot vector file containing only plotting commands, and (2) a set of postprocessors which format the output of the graphic language to a particular graphic device or devices.

Subroutines PSEUDO, NFRAME, NOTATE, and LINE are the basic subroutines used from the graphic software package. Subroutine PSEUDO causes the necessary parameters and linkage to be set up to output a device independent plot vector file for postprocessing for a particular plotting device. Subroutine NFRAME provides a means of executing a frame advance movement. Alphanumeric information for annotation and labeling is drawn by subroutine NOTATE. Subroutine LINE draws a continuous line through a set of successive data points where the minimum values and scale factors are stored at the end of the data arrays.

CONCLUDING REMARKS

A computer program has been written to define mathematically an arbitrary curved surface in surface-patch-equation form. Although the program is oriented toward aircraft configurations, it can be used to model mathematically any three-dimensional object by using an alternate data input format. Parametric cubic spline curves are fitted to the input data points to define the boundary-curve slopes. These slopes and the corner points are the necessary components of the surface patch equations. Program options include the application of three-dimensional rotation equations directly to the patch equations for plotting the surface at any desired viewing angle and the solution of a number of patches and a plane for generating cross-section or contour plots of an object or surface. Output from this program has been used to drive Calcomp, Gerber, and Varian plotters. The program has also been used for on-line display on a cathode-ray-tube device.

Langley Research Center,
National Aeronautics and Space Administration,
Hampton, Va., February 28, 1975.

APPENDIX A

PARAMETRIC CUBIC SPLINE SPACE CURVES

The derivatives in u and w at the corners of the surface patches are required for the surface equation. The grid points which determine the boundary curves are fitted in the w-direction and then in the u-direction for computation of the derivatives. A parametric cubic spline curve fit technique by Timothy E. Johnson of Massachusetts Institute of Technology is used.

Parametric cubics are the lowest order polynomial with the property of being able to twist through space. The spline curve exhibits the usual meaning of smoothness by minimizing curvature. Parametric curves are not sensitive to infinite slopes.

For purposes of completeness a brief description of the method follows as well as an explanation of the calling sequence for Subroutine SPFIT which utilizes this method.

A series of adjacent polynomial segments between each pair of given points is used to represent the curve.

The component cubic parametric equations are

$$X(t) = A_x t^3 + B_x t^2 + C_x t + D_x$$

$$Y(t) = A_y t^3 + B_y t^2 + C_y t + D_y$$

$$Z(t) = A_z t^3 + B_z t^2 + C_z t + D_z$$

where all operations are performed once on each component equation.

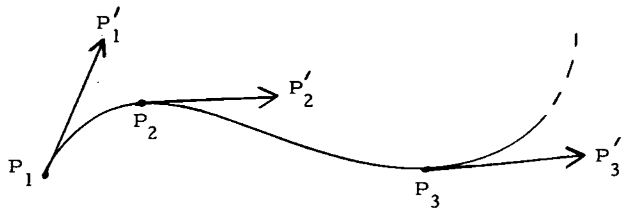
The component equations will be represented by

$$P(t) = At^3 + Bt^2 + Ct + D \tag{1}$$

The tangent parametric slopes at each point define the coefficients of the cubic segments and must be determined so that the tangent slopes give a smooth curve.

The given points $P_1, P_2, \ldots, P_n$ have the corresponding tangent slopes $P'_1, P'_2, \ldots, P'_n$ and the independent variable t varies so that $0 \leq t \leq L$ where L is the chord length between the given points. (See sketch (f).)

APPENDIX A



Sketch (f)

The boundary conditions for one cubic segment $P(t)$ can be written as

$$P(0) = P_1$$

$$P(L_1) = P_2$$

$$\frac{dP(t)}{dt}_{t=0} = P_1'$$

$$\frac{dP(t)}{dt}_{t=L_1} = P_2'$$

The coefficients of equation (1) can be written in terms of the end points and parametric slopes for a given segment

$$P(0) = D_1 = P_1$$

$$\frac{dP(t)}{dt}_{t=0} = C_1 = P_1'$$

and

$$P(L_1) = AL_1^{\ 3} + BL_1^{\ 2} + P_1'L_1 + P_1 = P_2$$

$$\frac{dP(t)}{dt}_{t=L_1} = 3AL_1^{\ 2} + 2BL_1 + P_1' = P_2'$$

Solving the last two equations yields

APPENDIX A

$$A_1 = \frac{2\left(P_1 - P_2\right)}{{L_1}^3} + \frac{P_1'}{{L_1}^2} + \frac{P_2'}{{L_1}^2}$$

$$B_1 = \frac{3\left(P_2 - P_1\right)}{{L_1}^2} - \frac{2P_1'}{L_1} - \frac{P_2'}{L_1}$$

The coefficients for each adjoining segment are found in the same manner.

The adjacent cubic segments can be related by setting the second derivatives equal at the common points.

For equal second derivatives at P_2

$$\left.\frac{d^2P(t)}{dt^2}\right|_{t=L_1} = \left.\frac{d^2P(t)}{dt^2}\right|_{t=0}$$

$$6A_1L_1 + 2B_1 = 2B_2 \tag{2}$$

Substituting the expressions for A and B into equation (2) and collecting terms give in terms of the unknown slopes:

$$L_2P_1' + 2\left(L_2 + L_1\right)P_2' + L_1P_3' = \frac{3}{L_1L_2}\left[{L_1}^2\left(P_3 - P_2\right) + {L_2}^2\left(P_2 - P_1\right)\right]$$

This equation may be written over all segments in matrix notation:

$$\begin{bmatrix} L_2 & 2(L_1 + L_2) & L_1 & \bigcirc & \\ & L_3 & 2(L_2 + L_3) & L_2 & \\ & & L_4 & 2(L_3 + L_4) & L_3 \\ & & & & \cdot \\ & \bigcirc & & \cdot & \cdot \end{bmatrix} \times \begin{bmatrix} P_1' \\ P_2' \\ P_3' \\ \cdot \\ \cdot \\ P_n' \end{bmatrix} = \begin{bmatrix} 3/L_1L_2\left[{L_1}^2(P_3 - P_2) + {L_2}^2(P_2 - P_1)\right] \\ 3/L_2L_3\left[{L_2}^2(P_4 - P_3) + {L_3}^2(P_3 - P_2)\right] \\ \cdot \\ \cdot \\ \cdot \\ 3/L_{n-2}L_{n-1}\left[L_{n-2}^2(P_n - P_{n-1}) + L_{n-1}^2(P_{n-1} - P_{n-2})\right] \end{bmatrix}$$

APPENDIX A

Since there are two more unknowns than there are equations, the second derivatives at P_1 and P_N are made equal to zero. The Thomas Algorithm, which is equivalent to Gaussian Elimination without pivoting, is used for solving the tridiagonal matrix. The cubic coefficients are normalized so that the parametric space for each curve segment varies from 0 to 1:

$$A' = AL^3$$

$$B' = BL^2$$

$$C' = CL$$

APPENDIX B

BICUBIC SURFACE PATCHES

This appendix is a brief description of the surface patch method used in program D3400.

The x-, y-, and z-coordinates of surface points are functions of two variables u and w:

$$X = f(u,w)$$

$$Y = g(u,w)$$

$$Z = h(u,w)$$

$$V = (X\ Y\ Z)$$

The vector $V(u,w)$ is a function of the two variables, u and w, which take on only the values between 0.0 and 1.0.

The method builds a surface by joining surface "patches." A surface patch can be thought of as a portion of a surface bounded by four space curves: (0,w), (1,w), (u,0), and (u,1).

The surface patch equation used is

$$V(u,w) = UM\bar{B}M^{t}W^{t}$$

where

$$U = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

$$W = \begin{bmatrix} w^3 & w^2 & w & 1 \end{bmatrix}$$

$$\bar{B} = \begin{bmatrix} V(0,0) & V(0,1) & V_w(0,0) & V_w(0,1) \\ V(1,0) & V(1,1) & V_w(1,0) & V_w(1,1) \\ V_u(0,0) & V_u(0,1) & V_{uw}(0,0) & V_{uw}(0,1) \\ V_u(1,0) & V_u(1,1) & V_{uw}(1,0) & V_{uw}(1,1) \end{bmatrix}$$

This 4×4 matrix is called a boundary matrix as it contains only geometric properties from the boundary curves: corner coordinates, corner slopes, and corner twists. (The corner twists are all made equal to zero in the present application.) These quantities are constants and grouped systematically in the matrix.

The blending function matrix M provides a blending effect from one surface patch into an adjoining patch

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

Since the matrix product MBM^t is constant, the pre- and post-multiplications are performed and the equation stored as $S = MBM^t$ for further computer processing. It is a simple matter, then, for writing an expression for u with w held fixed or u held fixed and w varying.

For a detailed development of the surface patch equation, see reference 1. Reference 2 presents the bicubic form of the surface patch equation as used in this report.

APPENDIX C

DESCRIPTION OF FILE AND METHOD OF STORAGE FOR SURFACE PATCH EQUATIONS

Program D3400 computes and stores by columns of the surface grid the patch matrix equations using the previously stored lines of grid points. A disk file is used for temporary storage of the elements of all the patch matrix equations.

The elements and certain other information are written in binary on disk unit 7 in the following format. FORTRAN names from the given program are used for illustration.

Record	Name	Purpose
1	ABC(8)	Identification
2	XMIN	Minimum X of the entire body
	XMAX	Maximum X of the entire body
	YMIN	Minimum Y of the entire body
	YMAX	Maximum Y of the entire body
	ZMIN	Minimum Z of the entire body
	ZMAX	Maximum Z of the entire body
	NOBJ	Total number of objects (or components) to form the complete body
3	(for $NOBJ_1$)	
	NSURF	Number of surfaces in object
	M1	Not used
	M2(2)	Name of object (for printing only)
	M3	Not used
	M4	Not used
4	(for $NSURF_1$)	
	N1	Number of rowwise patches (NCOL - 1)
	N2	Number of columnwise patches (NROW - 1)
	N3	Not used
	N4	Not used
	N5	Not used
5	PATCH (4,4,3)	48 elements of a patch equation are written by columns as one record
.		
.		
.		

(Record 5 is repeated for N1 × N2 patches taken by columns over the surface grid)

(Records 4, 5, and so forth are repeated for each NSURF from record 3)

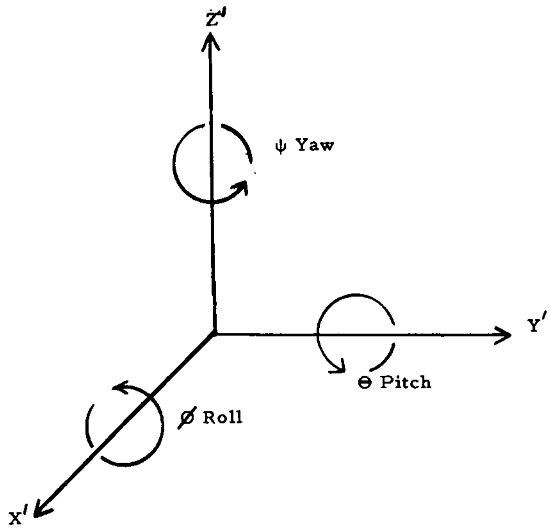
(Repeat record 3 for each NOBJ as given in record 2 and repeat records 4, 5, and so forth as required)

APPENDIX D

ORTHOGRAPHIC PROJECTIONS USING SURFACE PATCH EQUATIONS

The orthographic projections illustrated in this report are created by rotating each patch element to the desired viewing angle, transforming the rotated patch into a coordinate system in the plane of the paper, and computing actual coordinates from the patch equation. The body coordinate system is coincident with the fixed system in the plane of the paper when all the rotation angles are zero; for example, the configuration X-axis and Y-axis would coincide with the paper for plots in the X'Y' paper plane.

The rotations of the body and its coordinate system to give a desired viewing angle are specified by angles of yaw, pitch, and roll (ψ, θ, and ϕ), shown in sketch (g).



Sketch (g)

The equations used to transform the patch equations defining the body with a set of rotation equations (ψ, θ, and ϕ) into the desired paper plane are

$$S'_x = S_x(\cos\theta\cos\psi) + S_y(-\sin\psi\cos\phi + \sin\theta\cos\psi\sin\phi)$$
$$+ S_z(\sin\psi\sin\phi + \sin\theta\cos\psi\cos\phi)$$

$$S'_y = S_x(\cos\theta\sin\psi) + S_y(\cos\psi\cos\phi + \sin\theta\sin\psi\sin\phi)$$
$$+ S_z(-\cos\psi\sin\phi + \sin\theta\sin\psi\cos\phi)$$

$$S_z' = S_x(-\sin\theta) + S_y(\cos\theta\sin\phi) + S_z(\cos\theta\cos\phi)$$

Each element of the patch equation is rotated to the desired viewing angle and then transformed into a coordinate system in the plane of the paper. Only two of the preceding equations, determined by the desired paper plane, are used and result in a two-component patch equation.

The body surface plots may be enriched to any desired degree. By assigning u a value between 0.0 and 1.0, we have two cubic equations in w which may be easily solved by varying w from 0.0 to 1.0 for the rotated and projected coordinate values of a line across the patch. In a similar manner, w may be assigned a value from 0.0 to 1.0 resulting in cubic equations in u. By varying u from 0.0 to 1.0, coordinate values are generated across the patch in the u-direction.

An optional hidden-line test is incorporated into the program which may be used to provide the capability of deleting most elements on the surface of the configuration which would not be seen by a viewer. No provision is made for deleting portions of an element or components hidden by other components.

The surface vector normal to the paper plane is computed at each interpolated point. (See ref. 1.) If the surface vector is positive, the computed point faces the viewer and is visible; if the surface vector is negative the computed point is not visible and is not plotted. With

$$U = \begin{bmatrix} X_u & Y_u & Z_u \end{bmatrix} = \begin{bmatrix} \dfrac{\partial X}{\partial u} & \dfrac{\partial Y}{\partial u} & \dfrac{\partial Z}{\partial u} \end{bmatrix}$$

$$W = \begin{bmatrix} X_w & Y_w & Z_w \end{bmatrix} = \begin{bmatrix} \dfrac{\partial X}{\partial w} & \dfrac{\partial Y}{\partial w} & \dfrac{\partial Z}{\partial w} \end{bmatrix}$$

For one surface, the surface normal vector is

$$N = U \times W = \begin{bmatrix} J_x & J_y & J_z \end{bmatrix}$$

where

$$J_x = \begin{vmatrix} Y_u & Z_u \\ Y_w & Z_w \end{vmatrix}$$

$$J_y = \begin{vmatrix} Z_u & X_u \\ Z_w & X_w \end{vmatrix}$$

$$J_z = \begin{vmatrix} X_u & Y_u \\ X_w & Y_w \end{vmatrix}$$

The visibility of the point under consideration is determined by evaluating only the Jacobian outwardly normal to the paper plane at the point.

APPENDIX E

CROSS-SECTION OR CONTOUR PLOTS USING SURFACE PATCH EQUATIONS

By using both the surface patch equation in the form

$$V(u,w) = USW^T$$

where the components of the 4×4 matrix S are $\left[S_x \quad S_y \quad S_z\right]$ (see appendix B) and the equation of a plane

$$ax + by + cz - d = 0.0$$

and equation of the intersection of the plane and the patch surface may be written. Substitute $x = US_xW^T$, $y = US_yW^T$, and $z = US_zW^T$ in the preceding equation so that

$$U\left[aS_x + bS_y + cS_z\right]W^T - d = 0.0$$

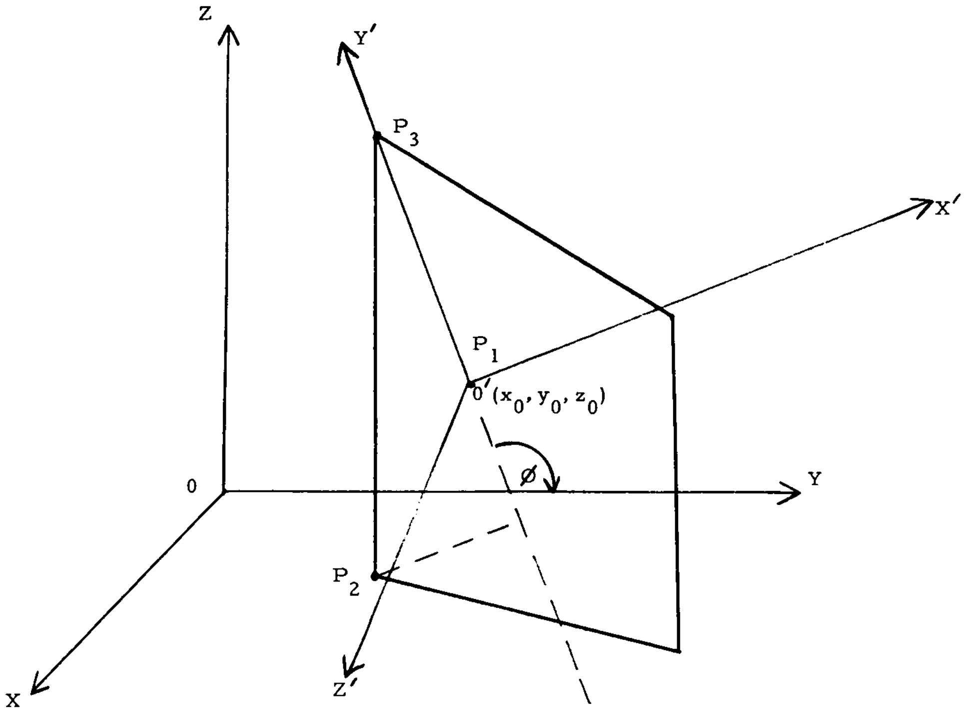
The matrix $\left[G\right] = \left[aS_x + bS_y + cS_z\right]$ is composed of constant elements and may be evaluated for an equation in the two variables u and w

$$UGW^T = d$$

By assigning values to w, a series of cubic equations in u are solved for points on the intersection curve of the surface with the plane or values may be assigned to u for cubic equations in w to solve for the intersection curve. (See ref. 1.) Only one solution for a curve of intersection is allowed.

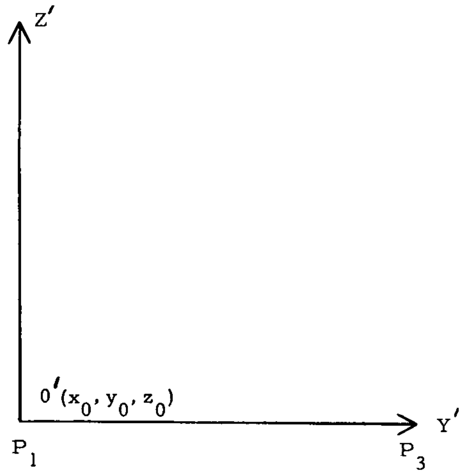
The points of the curve of intersection are then rotated and translated so that the plane of intersection coincides with the YZ-plane of the paper. The method for rotation and translation follows.

Sketch (h) illustrates the transformation system established by the 3 input points used to define the cutting plane.



Sketch (h)

The translated and rotated coordinate system used for plotting is shown in sketch (i).



Sketch (i)

Sketch (j) illustrates the choosing of three points on a cutting plane when the method of defining the plane is by the input of an X-intercept, a roll angle $\overline{\theta}$, and a Mach number.

XGF	One-quarter of Mach cone
EFGHIJ	Lies in plane parallel to YZ-plane

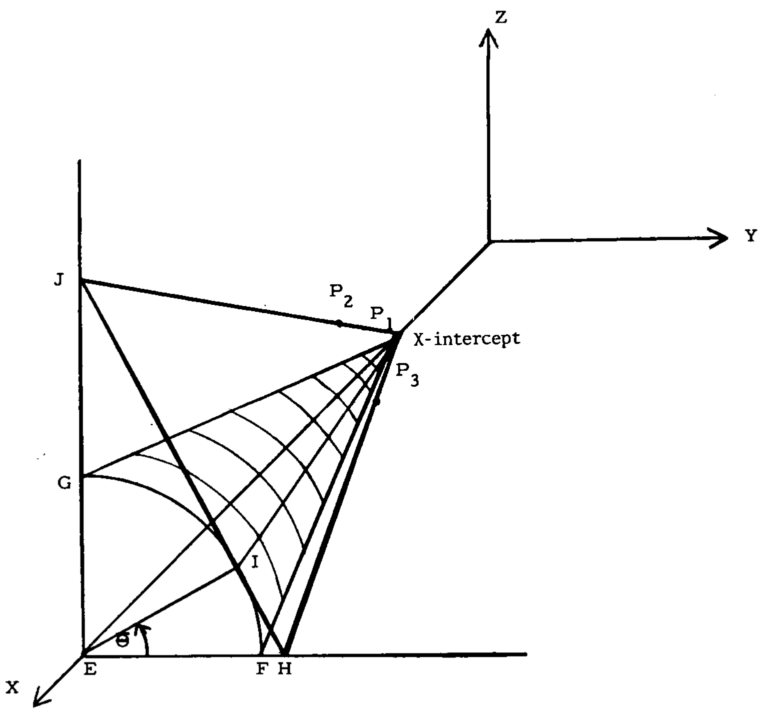
JXH	Mach plane, tangent to cone along IX
IE	Projection onto plane JHE of normal to Mach plane at I

To locate three points on the plane:

Point	X	Y	Z
P_1	X	0.0	0.0
P_2	X - C	0.0	1.0
P_3	X - B	1.0	0.0

where the equation of the plane is expressed as $AX + BY + CZ = D$ with

$$\beta = \sqrt{\overline{M}^2 - 1}, \quad A = 1.0, \quad B = -\beta \cos \overline{\theta}, \quad C = -\beta \sin \overline{\theta}, \text{ and } \quad D = X$$



Sketch (j)

By using the three input coordinates that define the cutting plane, two diagonal vectors may be formed with the components:

$$T_{1X} = x_3 - x_1 \qquad T_{1Y} = y_3 - y_1 \qquad T_{1Z} = z_3 - z_1$$

$$T_{2X} = x_3 - x_2 \qquad T_{2Y} = y_3 - y_2 \qquad T_{2Z} = z_3 - z_2$$

The cross products of the diagonal vectors give

$$N_X = T_{2Y}T_{1Z} - T_{1Y}T_{2Z}$$

$$N_Y = T_{1X}T_{2Z} - T_{2X}T_{1Z}$$

$$N_Z = T_{2X}T_{1Y} - T_{1X}T_{2Y}$$

which when divided by $N = \sqrt{N_X^2 + N_Y^2 + N_Z^2}$ yield the unit normal vector n of the cutting plane

$$n_X = \frac{N_X}{N}$$

$$n_Y = \frac{N_Y}{N}$$

$$n_Z = \frac{N_Z}{N}$$

To determine the coordinate system for plotting, two unit vectors lying in the cutting plane are needed

$$v_{1X} = \frac{T_{1X}}{T} \qquad v_{1Y} = \frac{T_{1Y}}{T} \qquad v_{1Z} = \frac{T_{1Z}}{T}$$

where

$$T = \sqrt{T_{1X}^2 + T_{1Y}^2 + T_{1Z}^2}$$

and

$$v_{2X} = n_Y v_{1Z} - n_Z v_{1Y}$$

$$v_{2Y} = n_Z v_{1X} - n_X v_{1Z}$$

$$v_{2Z} = n_X v_{1Y} - n_Y v_{1X}$$

The transformation matrix becomes

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

where

$$a_{11} = n_X \qquad a_{12} = n_Y \qquad a_{13} = n_Z$$

$$a_{21} = v_{1X} \qquad a_{22} = v_{1Y} \qquad a_{23} = v_{1Z}$$

$$a_{31} = v_{2X} \qquad a_{32} = v_{2Y} \qquad a_{33} = v_{2Z}$$

The origin of the reference coordinate system is transferred to the first point given in defining the cutting plane

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = [A] \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$$

The transformation from the reference to the paper coordinate system is accomplished by:

REFERENCES

1. Coons, Steven A.: Surfaces for Computer-Aided Design of Space Forms. MAC-TR-41 (Contract No. AF-33(600)-42859), Massachusetts Inst. Technol., June 1967. (Available from DDC as AD 663 504.)

2. Eshleman, A. L.; and Meriwether, H. D.: Graphic Applications to Aerospace Structural Design Problems. Douglas Paper 4650, McDonnell Douglas Corp., Sept. 1967.

3. Craidon, Charlotte B.: Description of a Digital Computer Program for Airplane Configuration Plots. NASA TM X-2074, 1970.

TABLE I.- SAMPLE CASE INPUT CARDS

```
                    SIMPLE CAMBERED CIRCULAR BODY
  1  1 -1  1  1          6  5  1  5  6                    1  4  1  5
9494.
0.      20.     50.     70.     100.                                      REFA
82.30   5.05    0.      180.100                                           XAF      5
114.1999.90     -.45    142.351                                           WAFORG  1
157.98  19.80   -1.85   98.570                                            WAFORG  3
239.18  46.20   -2.80   36.719                                            WAFORG  5
269.23  59.40   -4.30   15.670                                            WAFORG  9
282.00  66.00   -4.40   7.400                                             WAFORG11
3.6     2.75    -3.45   -6.8    -9.4                                      WAFORG12
0.      1.35    -1.2    -3.45   -6.8                                      TZORD   1
0.      .72     .0875   -.7825  -2.173                                    TZORD   3
0.      .248    .311    .2995   .2385                                     TZORD   5
0.      -.043   -.090   -.110   -.1224                                    TZORD   9
0.      -.0325  -.075   -.100   -.1324                                    TZORD  11
0.      1.069   1.518   1.451   0.                                        TZORD  12
0.      .889    1.272   1.136   0.                                        WAFORD  1
0.      .886    1.294   1.087   0.                                        WAFORD  3
0.      .880    1.375   1.155   0.                                        WAFORD  5
0.      .880    1.375   1.155   0.                                        WAFORD  9
0.      .880    1.375   1.155   0.                                        WAFORD11
0.      70.     130.    200.    260.    312.                              WAFORD12
7.4     7.4     1.25    -7.45   -10.2   -10.2                             XFUS     6
0.      96.     98.     79.     62.     0.                                ZFUS     6
241.0   31.75   -3.60                                                     AFUS     6
0.      12.     24.     34.5                                              PODORG  2
2.292   2.79    3.08    3.1                                               XPOD
252.0   47.0    -2.95   35.3    285.36  47.0    6.31    4.77              PODR
0.      30.     50.     70.     100.                                      FINORG  2
0.      .76     .977    .927    0.                                        XFIN
   1                                                                      FINORD
        10.  4.    8.                           15.  VU3          2
X Z     -45. 10.   -30.                         15.  ORT          2
X Z     -45. 10.   -30.                         15.  ORT   5   6  2
X Z OUT-45.  10.   -30.                         15.  ORT   5   6  2        1
   2
200.  0.     0.     200.  0.     1.    200.  1.    0.     .30    0. 16.PNT
200.  0.     0.     200.  0.     1.    200.  1.    0.     .30    20.16.PNT   8   8
-10.  0.     0.     0.    45.    1.2                      .1     0. 16.ANG   8   8202
2.5
-10.  0.     0.     55.   45.    1.2                      .1     20.16.ANG   8   8482
5.
```

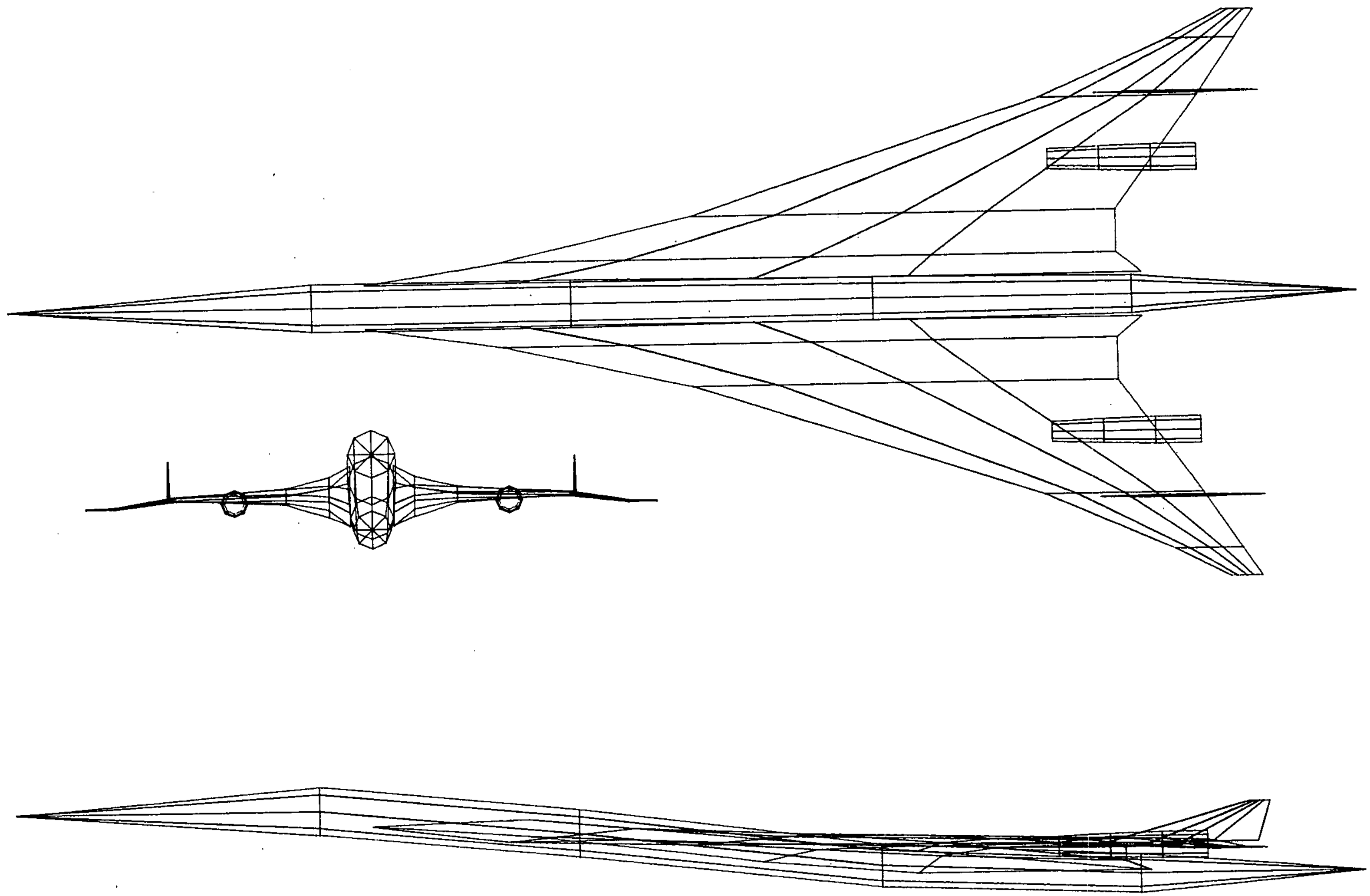
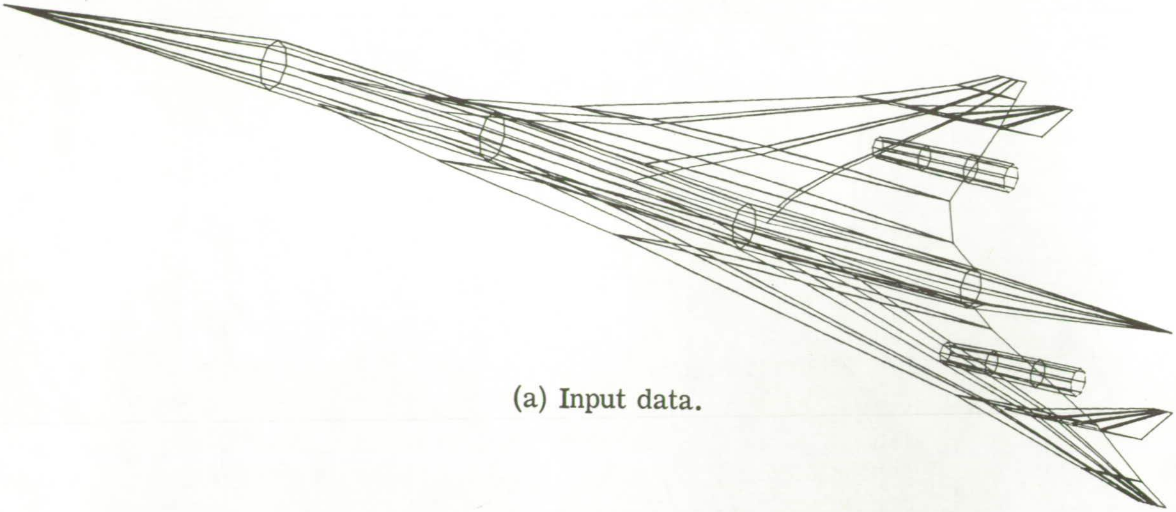
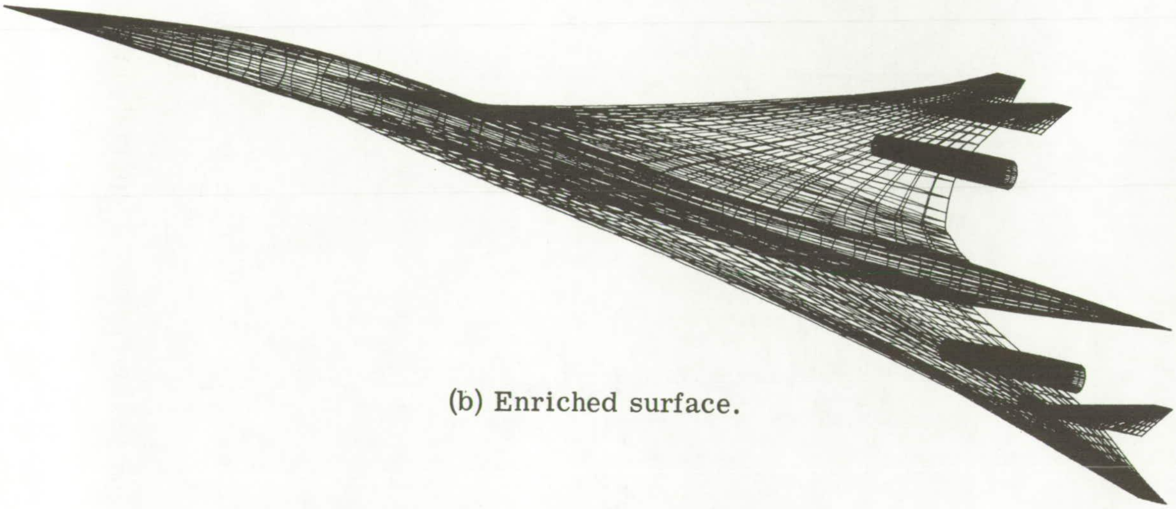


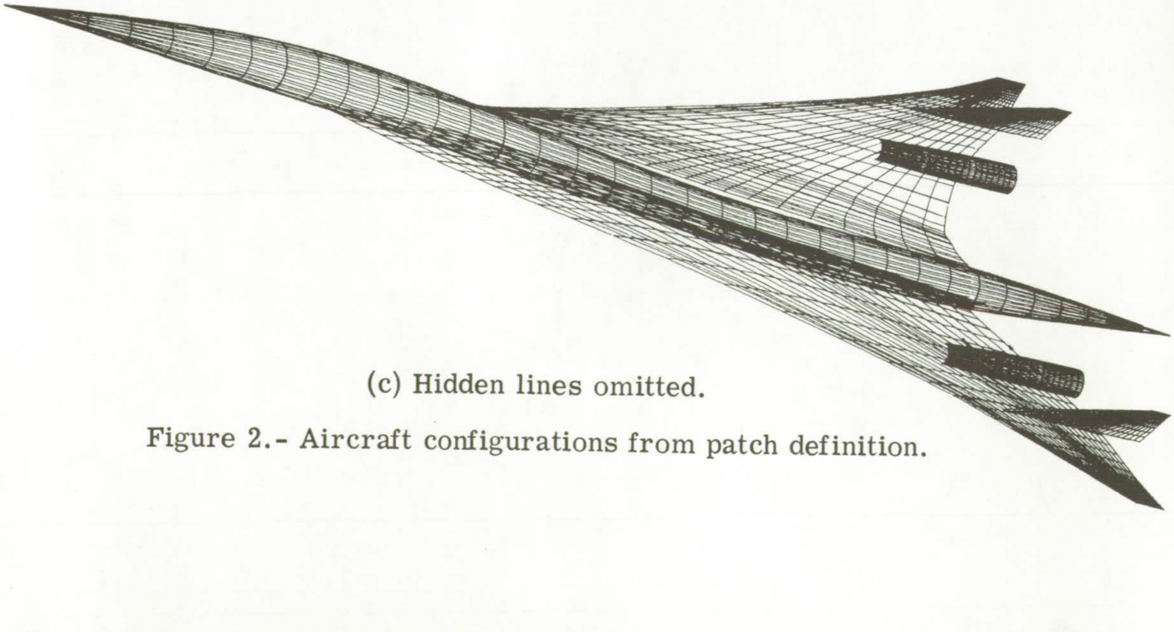
Figure 1.- A sample input configuration – simple cambered circular body.



(a) Input data.

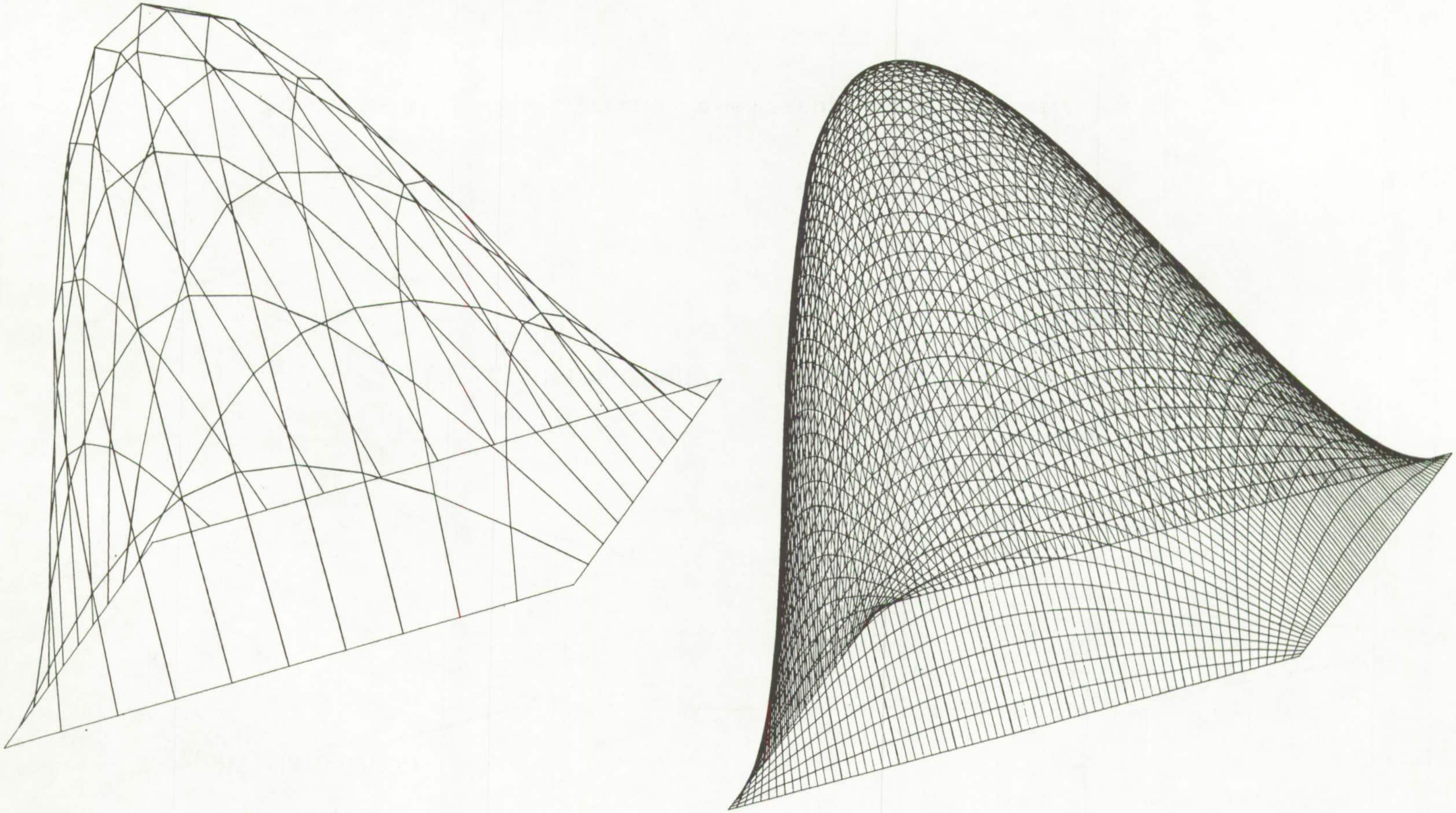


(b) Enriched surface.



(c) Hidden lines omitted.

Figure 2.- Aircraft configurations from patch definition.



(a) Input data.

(b) Enriched surface.

Figure 3.- Alternate input from patch definition.

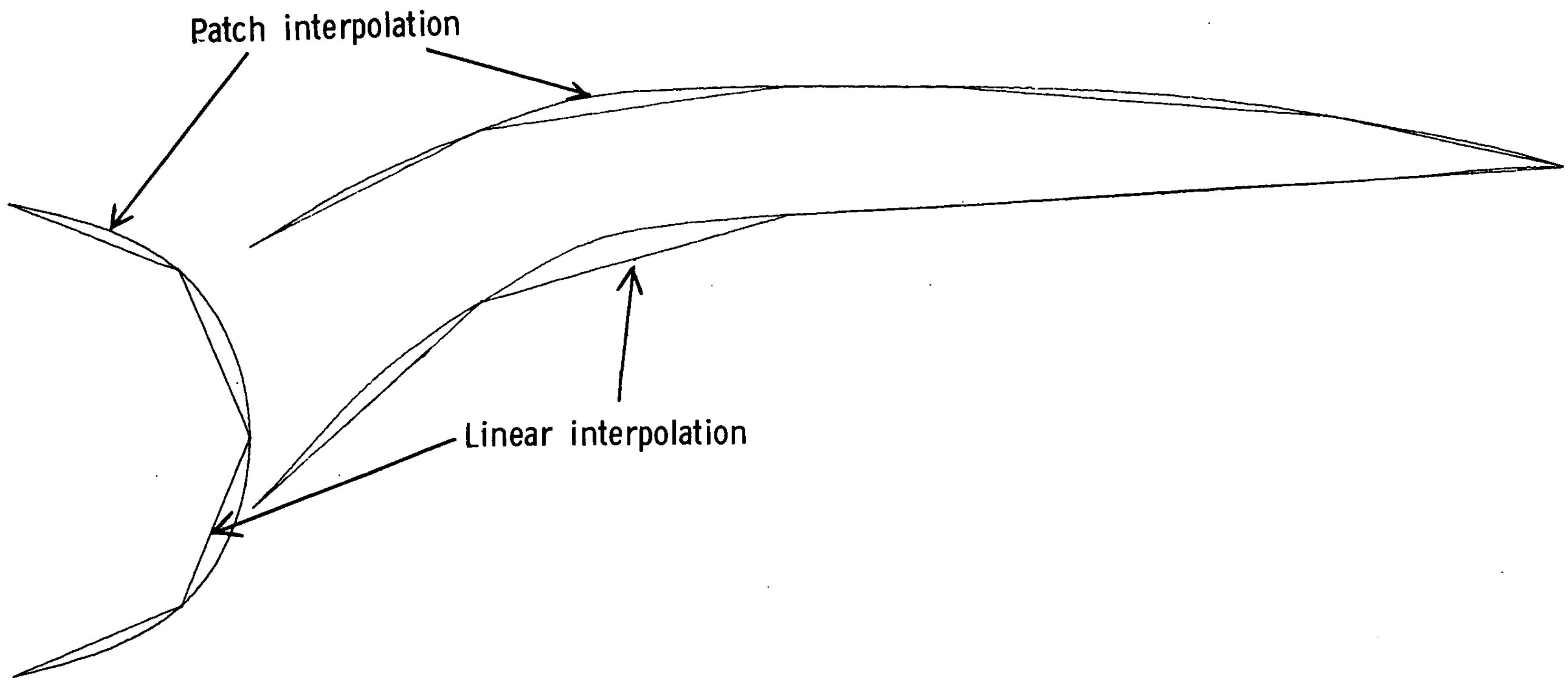


Figure 4.- Aircraft configuration cross section from patch definition.

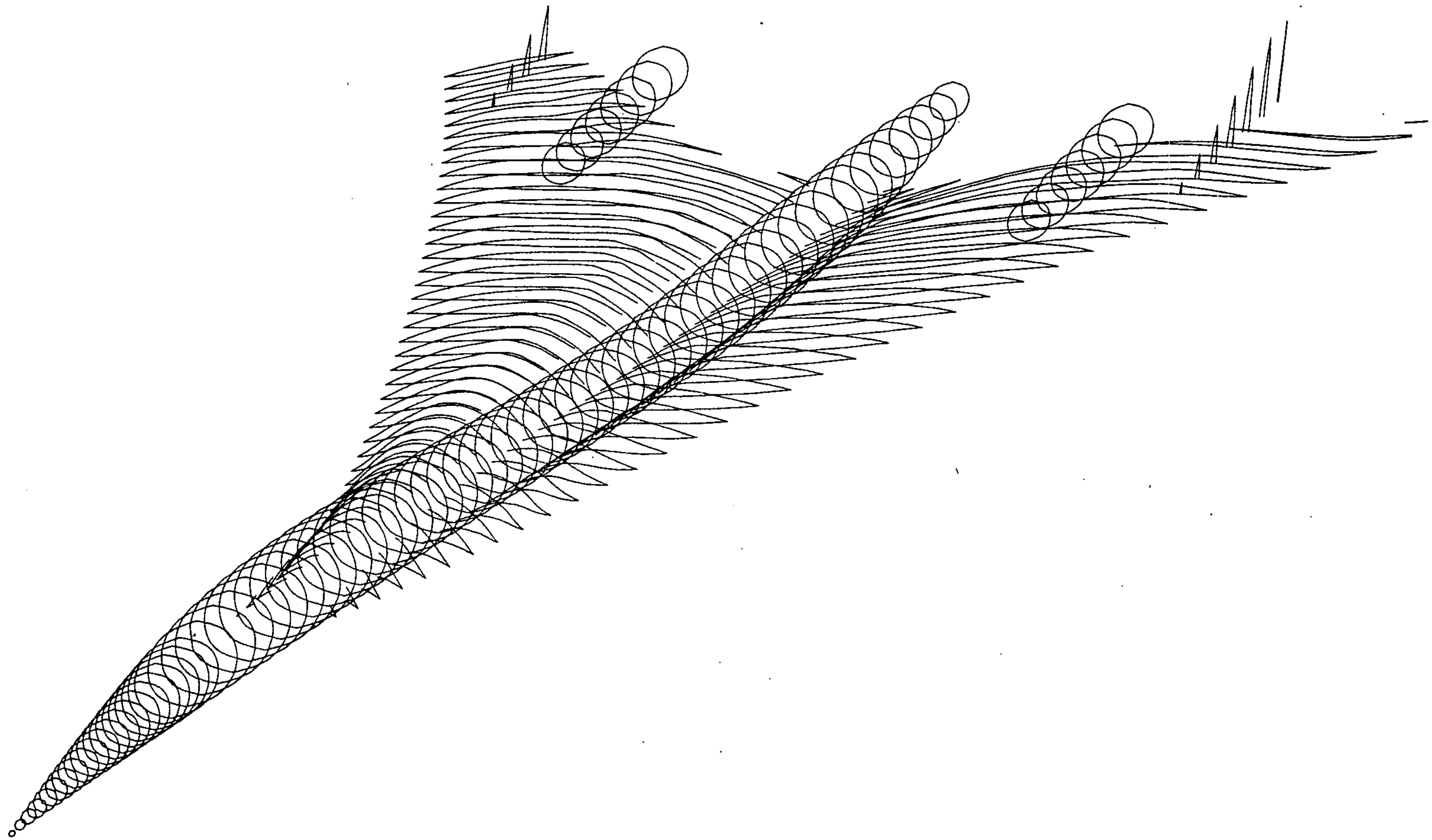
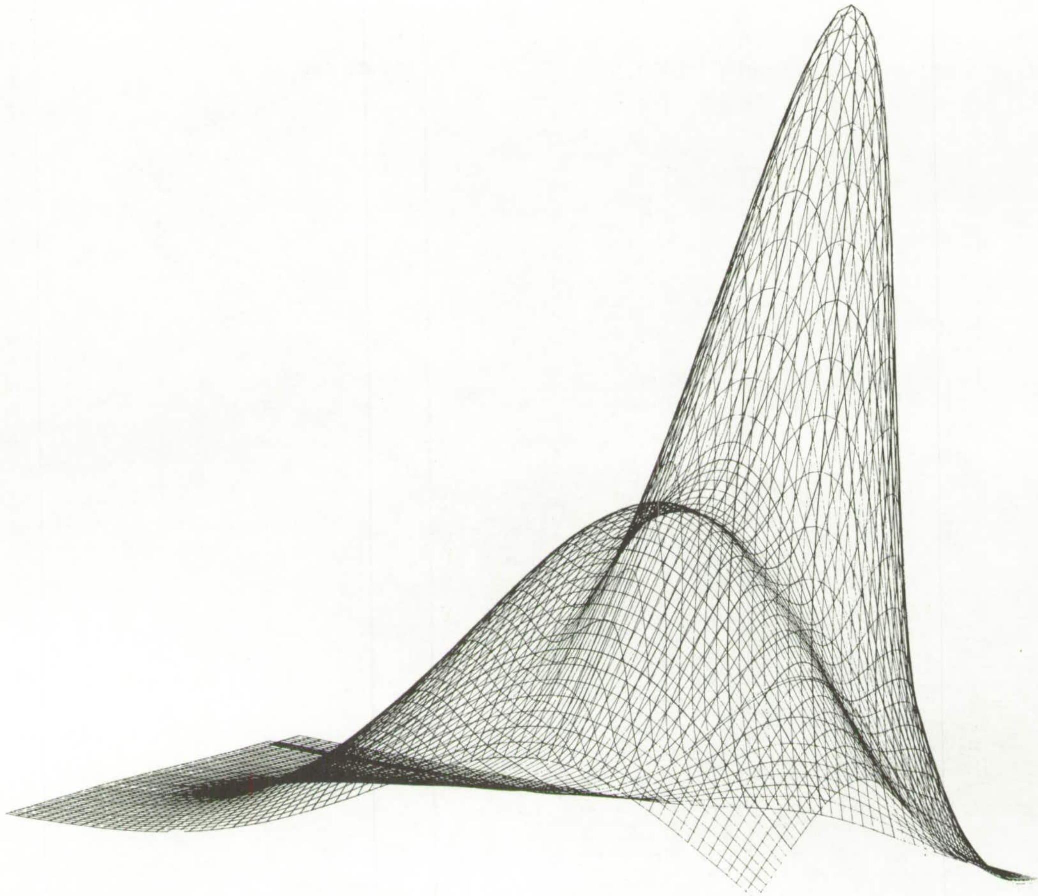


Figure 5.- Aircraft configuration cross sections with constant reference.



(a) Input data.

(b) Contour data.

Figure 6.- Optional input cross sections with constant reference.

NASA-Langley, 1975 L-9934

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WASHINGTON, D.C. 20546

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE $300

SPECIAL FOURTH-CLASS RATE
BOOK

POSTAGE AND FEES PAID
NATIONAL AERONAUTICS AND
SPACE ADMINISTRATION
451

U.S.MAIL

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Return

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons. Also includes conference proceedings with either limited or unlimited distribution.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include final reports of major projects, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546