K. DIGGES

AFAMRL-TR-81-111
Volume I

ARTICULATED TOTAL BODY (ATB) "VIEW" PROGRAM SOFTWARE REPORT, PART I, PROGRAMMER'S GUIDE

BRUCE D. LEETCH
WILLIAM L. BOWMAN

SYSTEMS RESEARCH LABORATORIES
2800 INDIAN RIPPLE ROAD
DAYTON, OHIO 45440

JUNE 1983

Distribution limited to U.S. Government Agencies only, official operational use, June 1983. Other requests for this document must be referred to AFAMRL/BBM.

SUBJECT TO EXPORT CONTROLS

This document contains information for manufacturing or using munitions of war. Exporting this information or releasing it to foreign nationals living in the United States without first obtaining an export license violates the International Traffic In Arms Regulations, 22 USC 2778, such a violation is punishable by up to 2 years in prison and by a fine of $100,000.

AIR FORCE AEROSPACE MEDICAL RESEARCH LABORATORY
BIODYNAMICS AND BIOENGINEERING DIVISION
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFAMRL-TR-81-111 Volume I	2. GOVT ACCESSION NO.
	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): ARTICULATED TOTAL BODY (ATB) "VIEW PROGRAM SOFTWARE REPORT, PART I, PROGRAMMER'S GUIDE	5. TYPE OF REPORT & PERIOD COVERED
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Bruce D. Leetch, William L. Bowman	8. CONTRACT OR GRANT NUMBER(s): F33615-81-C-0500
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Systems Research Labatories, Inc. 2800 Indian Ripple Road Dayton, OH 45440	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62202F; 7231-15-02
11. CONTROLLING OFFICE NAME AND ADDRESS: AFAMRL, Biodynamics and Bioengineering Division, AMD, AFSC Wright-Patterson, Air Force Base, OH 45433	12. REPORT DATE: JUNE 1983
	13. NUMBER OF PAGES: 202
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

DISTRIBUTION LIMITED TO U.S. GOVERNMENT AGENCIES ONLY, OFFICIAL OPERATIONAL USE, JUNE 1983. OTHER REQUEST FOR THIS DOCUMENT MUST BE REFERRED TO AFAMRL/BBM

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

18. SUPPLEMENTARY NOTES

SUBJECT TO EXPERT CONTROLS

AFAMRL CONTACT: Mr. R. Rasmussen, AFAMRL/BBM, extension 513-255-3667

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

Computer Simulation
Human Body Model
Aerodynamic Forces
Mathematical
Ejection
Harness Restraint System
Three Dimensional Dynamics
Computer Graphics
Crash Victim Simulation

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

The Articulated Total Body Model (ATBM) is used by the AFAMRL to study the biomechanics of pilot-seat ejection. The VIEW program provides a graphical representation of the simulation output from the ATBM.

The VIEW input data consist of two files. The first file, output from the ATBM, consists of body element and contact plane information. The second file contains control data necessary for running VIEW and additional planar surfaces for visual enhancement of the plots.

20. Abstract (continued)

VIEW output consists of two files. The first file is a listing file containing an echo of the input data and debug messages. The second file is the graphics output of VIEW. VIEW graphics can be output to four different devices: single color and multicolor Calcomp plotters and single color and multicolor graphics terminals.

This document is Part I of a two-part set. This part (Programmer's Guide) contains a detailed software description of VIEW. It is designed for the experienced FORTRAN programmer trying to understand the theory and structure of the VIEW program itself. In the Programmer's Guide, the structure and purpose of each program module is delineated and the meaning of each variable is described. In addition, the more complex technical and mathematical considerations governing the programming rationale are presented in four appendices.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

LIST OF ILLUSTRATIONS

1.0 INTRODUCTION

The Articulated Total Body Model (ATBM) computer program simulates gross human body motion in response to internally or externally applied forces.

Program VIEW was written to provide visual computer-aided data analysis for users of the ATBM program. It plots a pictorial representation of simulation-generated body position data at any constant time step. These plots, since they are projected views as seen by a camera, can be directly compared with pictures taken from tests using dummies or human subjects. The plots from the VIEW program provide an efficient method for preparing and checking initial position data used for input to the ATBM program. Still pictures of the initial position of the pilot and cockpit configuration can be overlaid with plots from program VIEW that represent body position data at time step zero. The input data can then be adjusted until the desired positions of the body segment contact ellipsoids are obtained.

Plots from program VIEW approximate pictures taken with a camera because of the projection technique used by the program. Three-dimensional objects are projected through a point onto a projection plane. This is similar to 3D objects being projected through a lens onto a film plane. The viewpoint used by program VIEW must be placed at the corresponding position and orientation of the camera lens to reproduce the same aspect. Then, by selecting the correct scale factor, plots from the program VIEW can be overlaid with pictures from the camera. This provides a convenient means for pictorally comparing simulated and photographic experimental data.

The basic graphics elements used in program VIEW are ellipsoids and polygons. These basic elements correspond to the ellipsoids and contact planes used in the ATBM program. The VIEW program has the capability to plot convex polygons with up to four sides which allows the plotting of realistic figures. The graphic elements for VIEW are defined by data stored on logical unit number (LUN) 1 which is a data file generated by the ATBM simulation program. VIEW reads the ATBM input file to obtain linear position and orientation data for the elements at any requested constant time step.

Each element is represented by a set of contour lines that are projected to form an image on the projection plane. Each contour line consists of a set of connected vectors in three-space that projects to a set of connected vectors on the projection plane.

The first step in generating the 3D ellipsoid images is to define contour lines for ellipsoids. This is done by setting up a grid mesh in the local X-Y plane of the ellipsoid. The Z values for the contour vectors are calculated using a constant X value and changing Y by constant increments. This results in ellipsoidal contour lines that are concentric about the local X axis of the ellipsoid. Since these contour lines are fixed to the local coordinate system of the ellipsoid, any rotation of the ellipsoid will be seen as rotation of the projected contour lines. Polygon contour lines are defined by a set of vectors in the global reference frames, each of which represents a side of the polygon. These vectors are projected in the same manner as ellipsoidal vectors. The result is a plot of the projected contour of the polygon.

Hidden line routines are included in the VIEW program. These routines eliminate sections of contour lines that are hidden from a viewer positioned at the viewpoint. Some lines are hidden because they are on the surface of an ellipsoid that faces away from the viewer. Other lines are hidden because another element blocks that contour line from the viewer. The routines that check for hidden lines are formulated to handle elements that are imbedded in other elements. For example, an ellipsoid can be imbedded in another ellipsoid. The contour lines will be drawn only up to the intersection of the surfaces.

2.0 INSTALLING THE VIEW PROGRAM

The VIEW program source code contains 1717 lines of FORTRAN code, consisting of a mainline and 36 subroutines and functions. The subroutines and functions are in alphabetical order. VIEW was originally developed on a CDC 6600 and has since been implemented on a Perkin-Elmer 3242 computer and an IBM 370 computer. The version used for development of this document was the Perkin-Elmer version. All Calcomp plotting calls are standard except the call to the initialization subroutine PLOTS which will be discussed later. All other subroutines and functions are either contained within the program or are standard intrinsic FORTRAN functions (SQRT, COS, etc.).

2.1 POSSIBLE PROGRAM MODIFICATIONS

Although the VIEW program source code may compile and execute properly on the target system, slight modifications may be necessary due to compiler differences. They include the following:

a. In certain FORTRAN compilers, seven character subroutine names are allowed. VIEW has incorporated this feature. If the FORTRAN compiler on the target system does not allow this, subroutines BUILDIE, CONVREC, LSEGINT, OVERLAP, and all references to these subroutines must be trimmed to six characters or less.

b. This version of VIEW was implemented on a 32 bit machine. The character packing per word factor was set to 4 (A4). If the target computer is other than a 32 bit machine, the character packing factor will have to be changed (16 bit to A2, 60 bit to A7, etc.). This is done by altering the format with statement label 200 in the mainline and the format with statement label 200 in subroutine TITLE. Allow for 8 bits per character.

c. Hollerith strings in this version of VIEW are enclosed by single quotes ('. . .'). On other target systems, new delimiter characters may be needed (e.g., *. . .*, ". . .," etc.).

d. In READ statements, the END = feature was used to check for end-of-file on the input files. If this feature is not in the target system FORTRAN compiler, an alternative means of checking for end-of-file will have to be used (e.g., EOF function on CDC systems).

e. The call to the Calcomp subroutine PLOTS in the mainline is for a Perkin-Elmer system [call PLOTS(0, 0, LUPLOT) where LUPLOT is the logical unit number of the plotting device]. Consult your system operator for the protocol used on the target system call to PLOTS.

f. The subroutine NFRAME is designed for a Grinnell color graphics subsystem on a Perkin-Elmer 3240. If this option is chosen for the VIEW program, consult your system operator for the protocol needed to interface to your graphics system.

3.0 VIEW PROGRAM MAINLINE DESCRIPTION

The general functions of the VIEW program are listed in the numbered blocks in the flow chart of the main routine (Section 5.1). A general description of each block will be given. For further information on a particular block, see the subroutine descriptions later in this report.

Block No. 1 calls subroutine INPUT to read data from the input control file and the ATBM input file. The input control file defines parameters for selecting what data are to be plotted. The ATBM input file contains data generated from the ATBM program on unit 1 that defines object sizes, orientations, and positions.

Block No. 2 calls subroutine CONVREC to convert rectangles used as contact planes in the ATBM program to polygons used in the VIEW program and to transfer planar coordinates for all polygons to the inertial reference system. See the discussion of CONVREC (Section 4.4) for further information.

Block Nos. 3, 4, and 6 are used to decrease the computation time of the hidden line subroutines. Without these three blocks, the hidden line subroutines would have to check out every point or 'vector head' against all objects to determine if that point was hidden or not hidden. The subroutines contained within these three blocks project all objects onto the projection plane and generate perimeters for all the shadows of the projected objects.

In Block No. 3, subroutine PRJPLY projects the ATBM and input polygons onto the projected plane. This is the form needed by the analysis performed by subroutine BUILDIE in Block No. 6.

In Block No. 4, subroutine PRJELR performs the projection of the ellipsoids. Ellipsoids, in general, project as complex, nonelliptical shadows. If the viewpoint coordinate system points directly at the center of the ellipsoid, the ellipsoid projects to an elliptical shadow. The subroutine assumes that this is the case. PRJELR projects all ellipsoids as elliptical shadows circumscribed with rectangles to conform with the format of subroutine BUILDIE.

In Block No. 5, subroutine POLYD is called to generate the directional cosine matrices for all polygons. See the description of POLYD (Section 4.22) for further information.

In Block No. 6, subroutine BUILDIE takes the polygons formed by the third and fourth blocks to build an array that defines object overlap. BUILDIE starts with the first object and records the object numbers of all polygons that overlap with the first object. Then the second object is selected and so forth until all objects and their overlapping objects are stored in the array IE.

Blocks No. 7 and No. 8 call subroutines PSE and PLPLN to check each vector against all the blocking objects recorded in the array IE with the hidden line subroutines. If the vector is not hidden, it is projected through a point onto the projection plane. The resultant vector is plotted. Note that while projections performed by PRJPLY and PRJELR are only on the perimeter of the shadow of each object, in PSE and PLPLN, the projections are of vectors that make up the contours of the objects. The program returns to Block No. 1 to get the next data set.

4.0 VIEW PROGRAM SUBPROGRAM DESCRIPTIONS

No.	Name	Type
1	MAIN	Main Program
2	BUILDIE	Subroutine
3	CLIP	Subroutine
4	CONVREC	Subroutine
5	CROSS	Subroutine
6	DET	Real Function
7	DOT	Subroutine
8	DOTT	Subroutine
9	DRCYPR	Subroutine
10	ELIPSN	Subroutine
11	EXTEND	Subroutine
12	GENDCM	Subroutine
13	HIDE	Subroutine
14	HYDE	Subroutine
15	INPUT	Subroutine
16	LSEGINT	Subroutine
17	MAT	Subroutine
18	NFRAME	Subroutine
19	OVERLAP	Subroutine
20	PLPLN	Subroutine
21	PNTPLT	Subroutine
22	POLYD	Subroutine
23	PREPLT	Subroutine
24	PRJELR	Subroutine
25	PRJPLY	Subroutine
26	PSE	Subroutine
27	ROT	Subroutine
28	SOLVA	Subroutine
29	SOLVR	Subroutine
30	TITLE	Subroutine
31	TPOINT	Subroutine
32	TRANS1	Subroutine
33	XINTCP	Real Function

34	XYZ	Subroutine
35	YINTCP	Real Function
36	XYZ	Subroutine
37	Z	Subroutine

4.1 MAIN PROGRAM

a. Purpose

Main program for the VIEW plotting package providing graphical representation of the ATB model output. Controls the initialization, data input, data processing, and plotting output functions of the program.

b. Subroutines Required

BUILDIE, CONVREC, DOT, ELIPSN, INPUT, MAT, NEWPEN, NFRAME, NUMBER, PLOT, PLOTS, PLPLN, POLYD, PRJELR, PRJPLY, PSE, SYMBOL, TITLE

c. Labeled Common Blocks Used

ATB, DBUG, ELLIPSE, INTERS, PLTT, POLYGON, VIEWP

d. Input or Argument Parameters

Input cards 1.0, 3.0, and 4.0

e. Optional Output

IDEBUG (1) = 1, print NIE array
IDEBUG (2) = 2, print IE array

f. Procedure

1. Read input cards 1.0, 3.0, and 4.0. Call subroutine INPUT to get ATB data.

2. Call subroutine CONVREC to convert ATB rectangles to VIEW polygon and to transform all polygon data from the segment coordinate system to the inertial coordinate system.

3. Call subroutine PRJPLY to project polygons onto the projected plane.

4. Call subroutine PRJELR to circumscribe ellipsoids with rectangles and project them to the projected plane.

5. Call subroutine POLYD to generate directional cosine matrices.

6. Call subroutine BUILDIE to define object overlap.

7. Call subroutines PSE and PLPLN to plot ellipsoids and polygons, then go get next data set.

4.2 SUBROUTINE BUILDIE

a. Purpose

The BUILDIE subroutine is called by the main routine to build the IE and NIE arrays. These arrays are used by the hidden line routines to determine if any objects block the point being plotted. This subroutine uses the data stored in the CONVEC array. See Figure 4.1 for a pictorial example of the data used by BUILDIE routine. The rectangles represent the projected ellipsoids of the ATB program. The ellipses are circumscribed with rectangles because algorithms dealing with polygon overlap require less memory and computation time than with algorithms that look for ellipse overlap. The rectangles shown in Figure 4.1 are sufficient to assure that no object overlap will be missed. The area inside the rectangles is larger than the area inside the circumscribed ellipse. While there is a chance that the BUILDIE routine would find two objects to overlap that really just missed, such a result will not make the plot incorrect; it just increases the computation time of the hidden line routines.

b. Subroutines Called

OVERLAP

c. Labeled Common Blocks Used

ELLIPSE, INTERS, POLYGON, REMOVE

d. Input or Argument Parameters

None

e. Optional Output

None

f. Procedure

The flow of BUILDIE starts with initializing the NIE and IE arrays. If the number of segments is equal to zero, the routine goes on and examines the polygons. If there are segments, the first one is selected, and immediately its own segment number is inserted in the IE array to indicate that it can hide portions of itself. Next, any other segment that is found to overlap with the first segment is recorded in the IE array. The fact of the overlap is recorded two places, once in the portion of the array for segment No. 1, and once in the portion of the array

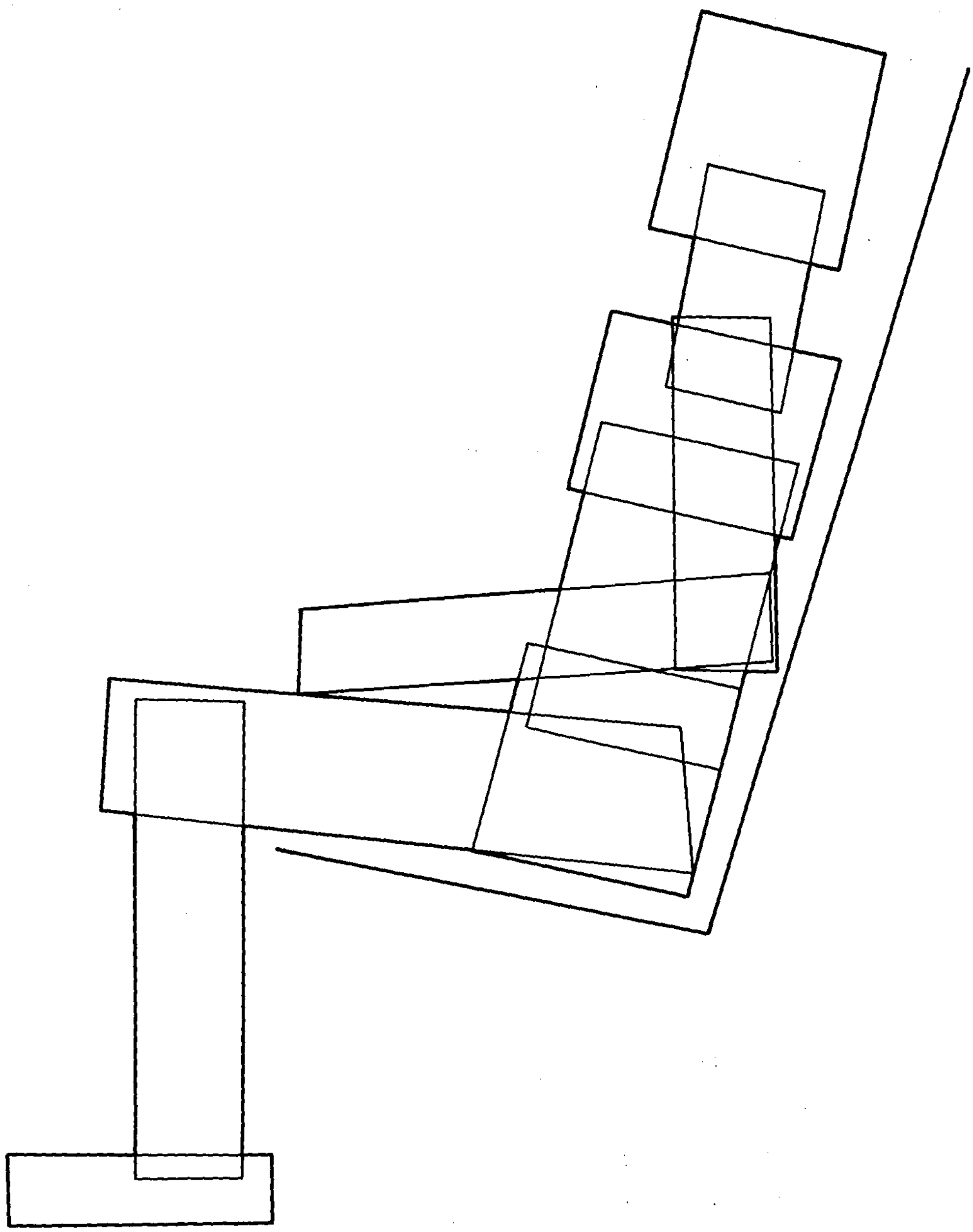


Figure 4.1 Example of Data Used by BUILDIE

that represents the other segment. In this way all segments are checked against each other until all segment-segment overlaps are recorded in the IE array. After the segment-segment overlaps are recorded, they are recorded in the IE array. Segment numbers are always less than 31, and polygon numbers are always greater than 30.

4.3 SUBROUTINE CLIP

a. Purpose

The purpose of this subroutine is to correctly execute the first draw that crosses a boundary (X or Y) of the valid plotting region. What is meant by "correctly execute" is that the intended draw extends to the boundary, but not beyond it. All further plotting after this draw is "clipped," i.e., not drawn, until the pen returns to the valid plotting region. The valid plotting region is described by the variables XMIN, XMAX, YMIN, and YMAX. CLIP also takes care of the special circumstance that the point being clipped is the first point in a line segment.

b. Subroutines Called

PLOT, XINTCP, YINTCP

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

X	-- X Coordinate of Point to be Plotted
Y	-- Y Coordinate of Point to be Plotted
XSAV	-- X Coordinate of Last Point Plotted
YSAV	-- Y Coordinate of Last Point Plotted
XMIN	-- X Coordinate of Left Side of Plotting Region

XMAX	--	X Coordinate of Right Side of Plotting Region
YMIN	--	Y Coordinate of Bottom of Plotting Region
YMAX	--	Y Coordinate of Top of Plotting Region
IPEN	--	Calcomp Pen Control Variable
IPLOT	--	Flag that Tells CLIP if Last Pen Move Was Clipped (IPLOT=Ø) or Not Clipped (IPLOT=1)

e. Optional Output

None

f. Procedure

The first function of CLIP is to handle the special circumstance that the point being clipped is the origin of a line. The reason this merits special treatment is that there is no previous point in the line from which a new point can be interpolated. The subroutine must wait until the next call to do this. So, the first step in the subroutine is to check the last call flag and see if it is set (LCALL = 1). If it is set, it means that on the previous call, a clip was performed on the origin of a line segment. What CLIP does is determine the X and Y coordinates to move the pen. The X coordinate is either the X intercept if the Y coordinate was off the plotter or the X boundary value if the Y was on the plotter. The Y coordinate is either the Y intercept value if the X coordinate was off the plotter or the Y boundary value if the X coordinate was on the plotter. CLIP moves the pen (up, IPEN = 3) to the point denoted by the X and Y coordinates, resets the previous X and Y coordinate variables (XSAV and YSAV), and clears the last call flag (LCALL = 0). If LCALL was not set at the beginning of the subroutine, CLIP skips this portion of the code.

The second step of this first function is to check if the current point being clipped is the origin of a line. If this is true (IPLOT = 1 and IPEN = 3), the coordinates of the point are saved in XLSAV and YLSAV, and the last call flag is set to one (LCALL = 1). If this second step was executed, the subroutine exits here. If not, it continues.

The second function of CLIP starts here. The subroutine determines if the clip flag has been set (IPLOT = 1). This is the flag that enables only the valid portion of the first segment clipped to be plotted. IPLOT is cleared after plotting and is kept clear until the pen returns to a valid X or Y coordinate. It is then reset by a draw in the valid plotting region. The X and Y coordinates are calculated by looking at the XOFF and YOFF flags and using either XINTCP and YINTCP functions or the X and Y boundaries. A line is drawn to that point. The clip flag IPLOT is cleared and the subroutine exits.

4.4 SUBROUTINE CONVREC

a. Purpose

The purpose of subroutine CONVREC is as follows. The ATB simulation program outputs plane information in the form of three plane equations that define boundary planes that bound the rectangle. The coordinate system used for defining these planes can be any one of the following three.

1. A rectangle defined in the inertial reference frame.

2. A rectangle defined in the vehicle reference frame.

3. A rectangle defined in the segment reference frame.

The VIEW plotting package works with polygons in an inertial reference frame. Therefore, each of the three possibilities described above must be converted to the format used internally in the VIEW plotting package.

Subroutine CONVREC converts rectangles in the ATB simulation format to polygons in the VIEW plotting format. It then transforms the vectors to the vertices of all the polygons from the coordinate system of the segment they are tied to the inertial reference system.

b. Subroutines Called

DET, DOT

c. Labeled Common Blocks Used

ATB, CONECT, DBUG, ELLIPSE, POLYGON

d. Input or Argument Parameters

None

e. Optional Output

IDEBUG(4) = 1; print reference segment number (ISG), plane number, and plane vectors in inertial reference.

f. Procedure

The equations that represent the rectangle in the ATB output are as follows:

1. A0X + B0Y + C0Z = D0

2. A1X + B1Y + C1Z = D1

3. A2X + B2Y + C2Z = D2

Solving these equations simultaneously gives a point of intersection of all three planes (see Figure 4.2).

This point is the point P1 used by the VIEW program for defining polygons. The VIEW program defines polygons by specifying the position of all corners. $\vec{r}$ is the normal to plane number 3, and $\vec{r}_3$ is the normal to the plane number 2. The length of $\vec{r}_2$ and $\vec{r}_3$ are obtained from the output of the ATB simulation program. Each corner of the polygon is found by adding the contour vectors of the rectangle in succession. This is done for all polygons read from the ATB input file.

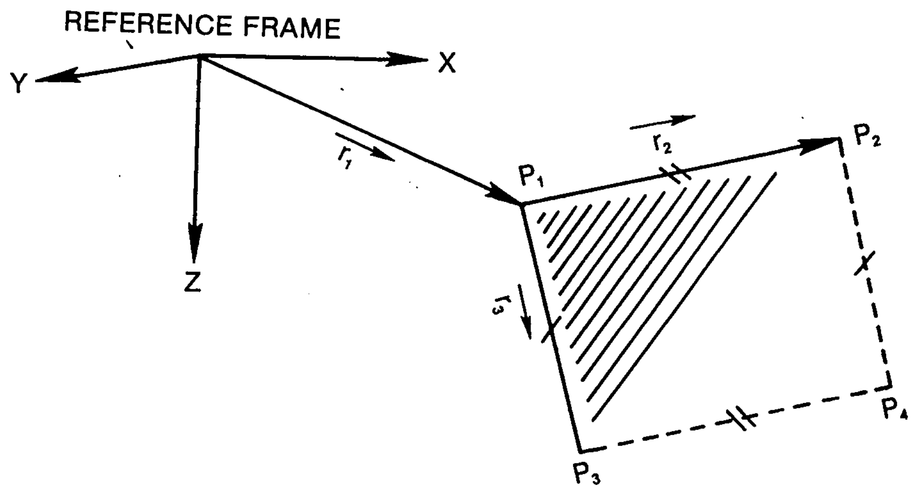


Figure 4.2 Point of Intersection for all Three Planes in CONVREC

4.5 SUBROUTINE CROSS

a. Purpose

Computes vector cross product $\vec{C} = \vec{A} \times \vec{B}$.

b. Subroutines Required

None

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

A, B, C: Arrays, each consisting of three elements, that represent vectors where the cross product is defined by $\vec{C} = \vec{A} \times \vec{B}$.

e. Optional Output

None

f. Procedure

Computes each element of $\vec{C}$ by

$$c_1 = a_2 b_3 - a_3 b_2$$

$$c_2 = a_3 b_1 - a_1 b_3$$

$$c_3 = a_1 b_2 - a_2 b_1$$

4.6 FUNCTION DET

a. Purpose

DET finds the determinant of the 3 × 3 square array passed to it.

b. Subroutines Called

None

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

A11	--	Values Representing Square, 3 × 3 Array
A12	--	
A13	--	
A21	--	
A22	--	
A23	--	
A31	--	
A32	--	
A33	--	

e. Optional Output

None

f. Procedure

The determinant is calculated in the following manner:

DET = A11*(A22*A33-A23*A32) - A12*(A21*A33-A23*A31) +
A13*(A21*A32-A22*A31)

4.7 SUBROUTINE DOT

a. DOT performs matrix multiplication C = $\underline{A}'\underline{B}$. If $\vec{A}$ and $\vec{B}$ are vectors, C is the dot product $\vec{A} \cdot \vec{B}$.

b. Subroutines Required

None

c. Labeled Common Blocks Used

None

d. Input or Argument parameters

A	- Matrix of Size (L,N)
B	- Matrix of Size (L,M)
C	- Product Matrix of Size (N,M)
N,M,L	- Sizes of Matrices A,B,C

e. Optional Output

None

f. Procedure

Each element C(I,J) of the product matrix C is computed by:

$$C(I,J) = \sum_{K=1}^{L} A(K,I) * B(K,J) \quad \text{for } I = 1,N \text{ and } J = 1,M$$

4.8 SUBROUTINE DOTT

a. Purpose

DOTT performs the matrix multiply $\underline{C} = \underline{AB'}$

b. Subroutine Called

None

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

A	- Matrix of Size (N,L)
B	- Matrix of Size (M,L)
C	- Matrix of Size (N,M)
N,M,L	- Sizes of Matrices A,B,C

e. Optional Output

None

f. Procedure

Each element C(I,J) of the product matrix C is computed as:

$$C(I,J) = \sum_{K=1}^{L} A(I,K) * B(J,K) \quad \text{for } I = 1,N \text{ and } J = 1,M$$

4.9 SUBROUTINE DRCYPR

a. Purpose

Sets up direction cosine matrix $\underline{D}$ for rotation angles A given in degrees about local the x, y, or z axis of the segment in question as indicated by I1, I2, or I3.

b. Subroutines Required

MAT, ROT

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

D	--	3 × 3 direction cosine matrix to be computed.
A	--	3 rotation angles given in degrees.
ID	--	3 integers (I1, I2, and I3) that indicate axis of rotation for each of the three angles in A (1, 2, or 3 indicates x, y, or z axis, respectively).

e. Optional Output

None

f. Procedure

Computes as a matrix product

$$\underline{D} = \underline{D}_{I3}\underline{D}_{I2}\underline{D}_{I1}$$

where each D_I is one of the following

$$\underline{D}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix}$$

$$\underline{D}_2 = \begin{pmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{pmatrix}$$

$$\underline{D}_3 = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

depending as I1, I2, and I3 have values of 1, 2, or 3.

Note: (1) For the normal sequence of yaw (Ψ), pitch (θ), and roll (ϕ), let ID = 3, 2, 1 to obtain

$$\underline{D} = \underline{D}_{\phi}\ \underline{D}_{\theta}\ \underline{D}_{\psi} = \underline{D}_1\ \underline{D}_2\ \underline{D}_3$$

(2) For the reverse sequence as required by Subroutine INITIAL prior to Version 18 of the ATB program, let ID = 1, 2, 3 to obtain

$$\underline{D} = \underline{D}_{\psi}\ \underline{D}_{\theta}\ \underline{D}_{\phi} = \underline{D}_3\ \underline{D}_2\ \underline{D}_1$$

(3) For Euler angles, precession (ϕ), nutations (θ), and spin (ψ), let ID = 3, 1, -3 to obtain

$$\underline{D} = \underline{D}_{\psi}\ \underline{D}_{\theta}\ \underline{D}_{\phi} = \underline{D}_3\ \underline{D}_1\ \underline{D}_3$$

4.10 SUBROUTINE ELIPSN

a. Purpose

Subroutine ELIPSN generates the X1 array of contour vectors for a quarter of the ellipsoid. These contours are used by subroutine PSE to generate complete contours for the entire ellipsoid.

b. Subroutines Called

SQRT

c. Labeled Common Blocks Used

ELLIPSE

d. Input of Argument Parameters

INDEX -- Number of Steps Plus One

X1 -- Array Containing a Quarter of the Ellipsoid

IN -- Array Containing Number of Points in each Contour Array

e. Optional Output

None

f. Procedure

ELIPSN is called once for each ellipsoid. For each step for ellipsoid, calculate X and Y coordinates. Calculate test variable [TEST = $(1-X^2*SIMP1) - Y^2*SIMP2$]. If result is negative, it means point is not on the ellipsoid. In this case, Y coordinate is calculated as

$$\sqrt{\frac{1-X^2 * SIMP1}{SIMP2}}$$

and Z is set to zero. The results are stored in X1. If TEST was positive, Z is calculated and the results are stored in X1.

4.11 SUBROUTINE EXTEND

a. Purpose

EXTEND is a subroutine used in conjunction with the hidden line routines. This routine removes large gaps that could exist around boundaries where contour lines are hidden and not hidden. The gaps are caused by dividing up contour lines into a set of vectors. Once it is determined that a particular vector crosses a boundary, only a portion of the vector must be plotted. If the entire vector were left unplotted, a gap would exist around these boundaries. The size of the gaps would then cover the range from very small to the size of the vector. The EXTEND subroutine is called whenever a vector passes through one of these boundaries. EXTEND finds the portion of the vector not hidden and returns this information to the plotting subroutine.

b. Subroutines Called

HIDE, HYDE

c. Labeled Common Blocks Used

ELLIPSE, INTERS, PLTT

d. Input or Argument Parameters

P -- Contains X, Y, and Z coordinates of the two end points of the line.

I -- Points to unhidden point location in P array.

J -- Points to hidden point location in P array.

e. Optional Output

None

f. Procedure

EXTEND starts by finding the midpoint for a line between the X, Y, and Z coordinates found in array P. This midpoint is passed to either subroutine HYDE (ellipsoids) or HIDE (polygons) to check if contour line is hidden. If it is hidden, the midpoint coordinates are loaded into row J of array P. If it is not hidden, the midpoint is loaded into row I of array P.

4.12 SUBROUTINE GENDCM

a. Purpose

The name of GENDCM stands for generate direction cosine matrix. This subroutine creates a direction cosine matrix from information that is readily available to the user of program VIEW. GENDCM requires two position vectors to define the viewpoint position and orientation. Vector $\vec{VP}$ defines the position of the viewpoint in the reference frame of the segment to which the viewpoint is attached. Vector $\vec{RA}$ defines a point where the viewpoint coordinate system is aimed. The aiming of the viewpoint coordinate system is determined by the direction of the positive Z axis. The viewpoint always looks down the positive Z axis.

b. Subroutines Called

SQRT

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

CAMERA	--	Position vector for the viewpoint in the reference frame of the segment to which the viewpoint is attached (X, Y, Z coordinates).
FOCUS	--	Position vector for point which viewpoint Z axis is aimed (X, Y, Z coordinates in the reference frame of the segment to which the viewpoint is attached).
D	--	Direction cosine matrix for viewpoint (transformed from the inertial coordinate system to the viewpoint coordinate system).

e. Optional Output

None

f. Procedure

GENDCM assumes the $\hat{X}$ axis of the viewpoint coordinate system remains parallel to the X-Y plane of the inertial reference frame. Also, $\hat{X}$ axis must be normal to the $\hat{Z}$ axis in the X-Y plane of the inertial reference frame. $\hat{Z}$ in the X-Y plane is given by

$$\left(\frac{Z_1}{|\vec{Z}|}, \frac{Z_2}{|\vec{Z}|}, 0\right)$$

and

$$\left(\frac{Z_2}{|\vec{N}|}, -\frac{Z_1}{|\vec{N}|}, 0\right)$$

must be $(\hat{Z}_2, -\hat{Z}_1, 0)$. Now both the $\hat{X}$ and $\hat{Z}$ vectors of the viewpoint coordinate system are determined. The $\hat{Y}$ is obtained by crossing $\hat{Z}$ into $\hat{X}$. The direction cosine matrix has the following form.

$$\begin{bmatrix} Z(2)/XNORM & -Z(1)/XNORM & 0.0 \\ Z(1)*Z(3)/XNORM & Z(2)*Z(3)/XNORM & -XNORM \\ Z(1) & Z(2) & Z(3) \end{bmatrix}$$

where XNORM is $|\vec{N}|$, the length of $\vec{N}$.

4.13 SUBROUTINE HIDE

a. Purpose

The purpose of HIDE is to determine if a point is being hidden by another polygon. The subroutine corresponds with subroutine HYDE (HYDE checks ellipsoids).

b. Subroutines Called

DOT, MAT, TPOINT, TRANS1

c. Labeled Common Blocks Used

ELLIPSE, POLYGON

d. Input or Argument Parameters

KK	--	Polygon number to check blocking.
P3	--	Contains position vector of point to check.
IFLAG	--	Flag passed back to caller indicating hidden (IFLAG=1) or not hidden (IFLAG=2).

e. Optional Output

None

f. Procedure

The subroutine first determines if the projected point and the projected polygon overlap. If the projected point is outside of the projected polygon, the point cannot be hidden by the polygon being examined. If the projected point does lie within the projected polygon, it must be determined which is closer to the viewpoint. If the point is closer than the polygon, that point is not hidden by the polygon being examined. Appendix C, "Intersection of a Three Space Vector and Plane," describes in detail how to determine whether the plane or point is closer to the viewpoint.

The overlap on the projection plane of a projected point and a projected polygon is determined by subroutine TPOINT. TPOINT returns a flag called IFLAG that indicates overlap or no overlap.

4.14 SUBROUTINE HYDE

a. Purpose

The purpose of HYDE is to determine if a point is hidden by an ellipsoid. If the point is hidden, variable IFLAG is returned as one; if not hidden, IFLAG is set to two.

b. Subroutines Called

ABS, SQRT, DOT, DOTT, MAT, XYZ, YZ, Z

c. Labeled Common Blocks Used

ELLIPSE

d. Input or Argument Parameters

N	--	Possible hiding ellipsoid number (I)
R	--	Vector to plotting point (I)
IFLAG	--	Flag indicating hidden (IFLAG=1) or not (IFLAG=2) (O)

e. Optional Output

None

f. Procedure

The equations used by subroutine HYDE (and accompanying subroutines XYZ, YZ, and Z) can be found in Appendix A, "Hidden Line Problem Between Two Ellipsoids." Refer to this appendix for further information on HYDE.

4.15 SUBROUTINE INPUT

a. Purpose

INPUT has two separate functions. The first is to read initial data from the input control file, read polygon data from the ATB input file, and to initialize variables. This is all done on the first call. The second function, performed on all subsequent calls, is to read ellipsoid data from the ATB input file.

b. Subroutines Called

DOT, DRCYPR, GENDCM

c. Labeled Common Blocks Used

ATB, CONECT, DBUG, ELLIPSE, INTERS, PLTT, POLYGON, REMOVE, VIEWP

d. Input or Argument Parameters

CTIME	--	Current time of program passed from main.
Input Cards	--	6.0, 6.1, 7.0, 7.1, 7.2, 8,0, 9.0 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, 15.1

e. Optional Output

IDEBUG(3) = 1; print viewpoint position vector, viewpoint direction cosine matrix, NSTEPS from cards 7.0 and 8.0, segment inertial position vector, segment inertial direction cosine matrix, NSEG, NPL, SFACTR, OFSETX, and OFSETY.

f. Procedure

During the first call, cards 6.0, 6.1, 7.0, 7.1, 7.2, 8.0, 9.0, 10.0, 11.0, 12.0, 13.0, 14.0, 15.0, and 15.1 are read from the input control file. Polygon data are read from the ATB input file. The direction cosine matrix is initialized in three different ways, depending on the input flag ICODE. When ICODE equals zero, the roll, pitch, and yaw angles are found in the RA array (card 15.0); and subroutine DRCYPR calculates the direction cosine matrix. When ICODE equals one, the direction cosine matrix is supplied as input (card 15.1). When ICODE is equal to two, subroutine GENDCM generates the direction cosine matrix. The subroutine exits.

On all subsequent calls, INPUT starts at FORTRAN statement #600. Segment data are read from the ATB input file. This reading is continued until the time of the data is greater than or equal to the current time of the program (CTIME). When this time is reached, the current location of the contact ellipsoid is calculated. If the time interval of the ATBM simulation data is greater than the DT of the view run, variable IFLAG is set to ten. This is done to signal the program to continue incrementing CTIME without making plots until CTIME reaches the next available time point in the simulation data. The subroutine then exits.

4.16 SUBROUTINE LSEGINT

a. Purpose

LSEGINT is called by subroutine OVERLAP to check for two lines intersecting. The line segments are represented by two end points for each line.

b. Subroutines Called

None

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

P1	--	X and Y coordinates for line No. 1, end point No. 1
P2	--	X and Y coordinates for line No. 1, end point No. 2
R1	--	X and Y coordinates for line No. 2, end point No. 1
R2	--	X and Y coordinates for line No. 2, end point No. 2
IFLag	--	Flag passed back to OVERLAP to show intersection (IFLAG=1) or no intersection (IFLAG=0)

e. Optional Output

None

f. Procedure

LSEGINT checks for six separate cases:

Case 1 -- Regular configuration
Case 2 -- One line is vertical
Case 3 -- Both lines are vertical
Case 4 -- Both lines are horizontal
Case 5 -- Both lines have the same nonzero slope
Case 6 -- One line is vertical, the other is horizontal

It first checks for Cases 3 and 4. If either of these cases are true, the subroutine returns. For Cases 1, 2, 5, and 6, refer to Appendix D, "Intersection of Line Segments in a Plane." The code of subroutine LSEGINT follows the equations and text found in the appendix.

4.17 SUBROUTINE MAT

a. Purpose

Performs the matrix multiple $\underline{C} = \underline{AB}$.

b. Subroutines Called

None

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

A -- Matrix of Size (LL,MM)
B -- Matrix of Size (MM,NN)
C -- Product Matrix of Size (LL,NN)

LL, MM, NN	--	Sizes of Matrices A, B, C
JA, JB, JK	--	First Dimension of A, B, C in Calling Subroutine

e. Optional Output

None

f. Procedure

Each element (C(I,J) of the product matrix C is computed by:

$$C(I,J) = \sum_{K=1}^{MM} A(I,K) * B(K,J) \text{ for } I = 1, LL \text{ and } J = 1, NN$$

4.18 SUBROUTINE NFRAME

a. Purpose

NFRAME performs all end of frame operations necessary for a Grinnell graphics system. The code furnished in this version of the VIEW program is for a Perkin-Elmer 3242 running under OS/32.

b. Subroutines Called

DOLWH, PLOTS

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

None

e. Optional Output

None

f. Procedure

Again, this subroutine is designed for a Perkin-Elmer 3242 computer and a particular graphics system, a Grinnell GMR-27. If a different graphics system is to be utilized, this subroutine will need to be changed. Immediately upon entering NFRAME, subroutine DOLWH is called. DOLWH stands for digital output with handshake. An end-of-frame halfword (FFFF) is output to the Grinnel system. Subroutine PLOTS is called to initialize the next frame. The subroutine then exits.

4.19 SUBROUTINE OVERLAP

a. Purpose

This subroutine is called by BUILDIE to check two objects for overlap. The objects are defined by the first two arguments in the call to OVERLAP. At the point where OVERLAP is called, all objects have a projected polygon representation, and OVERLAP determines if these two polygons overlap.

b. Subroutines Called

LSEGINT, TPOINT

c. Labeled Common Blocks Used

POLYGON

d. Input or Argument Parameters

III	--	Object No. 1 to be Tested
KKK	--	Object No. 2 to be Tested
MFLAG	--	Flag indicating overlap or not. MFLAG = 0, no overlap; MFLAG = 1, overlap.

e. Optional Output

None

f. Procedure

There are two basic checks employed by this subroutine to determine object overlap. First is the POINT INSIDE A POLYGON TEST. This test starts with a point, usually a corner of a polygon, and tests are made to see if that point lies inside or outside the polygon. Once a point is found to be within the other polygon, MFLAG is set equal to 1 to indicate object overlap, and OVERLAP returns to the calling program. If no points lie inside the polygons being tested, the INTERSECTING LINE SEGMENT TEST is used. This test checks for line segments of one polygon intersecting line segments of the other polygon. Again, if intersection is found, MFLAG is set equal to 1, and OVERLAP returns to BUILDIE. If no intersection is found, the two objects must not overlap, and MFLAG is set equal to 0 and OVERLAP returns to BUILDIE.

4.20 SUBROUTINE PLPLN

a. Purpose

This subroutine is called by the Main program to set up arrays containing polygon data for subroutine PNTPLT to plot.

b. Subroutines Called

NEWPEN, PNTPLT

c. Labeled Common Blocks Used

DBUG, ELLIPSE, PLTT, POLYGON, REMOVE

d. Input or Argument Parameters

SEG	--	Array containing vectors representing sides of the polygons.
INDEX2	--	Array size for SEG.

e. Optional Output

None

f. Procedure

Array SEG must have the vectors that represent the sides of the polygons. Each side of the polygon is represented by a series of short vectors, even though one long vector could be plotted. A series of short vectors is used so current algorithms can be used for the hidden line segment problem. The vectors in array SEG are in the inertial reference frame. After loading data into SEG, it is sent to subroutine PNTPLT to plot. After doing this for all planes, PLPLN exits.

4.21 SUBROUTINE PNTPLT

a. Purpose

The purpose of PNTPLT is to plot all the points passed to it in array SEG. PNTPLT is called by subroutines PSE to plot ellipsoids and by PLPLN to plot polygons. PNTPLT is where all calls to the Calcomp plotting subroutines are made.

b. Subroutines Called

CLIP, EXTEND, HIDE, HYDE, PLOT, PREPLT, TRANS1

c. Labeled Common Blocks Used

ELLIPSE, INTERS, PLTT

d. Input or Argument Parameters

SEG	--	Array containing points to be plotted.
IPEN	--	Calcomp pen control variable.
INDEX2	--	Maximum number of points to plot.
NPTS	--	Number of points to plot.
CARD 16.0	--	XMIN, XMAX from input control file.

e. Optional Output

None

f. Procedure

PNTPLT takes each point and checks to see if the point is hidden. This is accomplished by calling subroutine HYDE to check for possible blocking ellipsoids and by calling subroutine HIDE to check for possible blocking polygons. After it is determined that the point is or is not hidden, PNTPLT checks to see if a boundary is crossed by the vector originating on the last point and ending on the current point. A boundary is indicated whenever the state of the last point and the current point is different. If the last point was not hidden and the current point is hidden, the vector goes from a not hidden zone into a hidden zone. The boundary between the two zones is the boundary that is checked for at this stage of subroutine PNTPLT. If a boundary is present, subroutine EXTEND is called to extend a new vector from the not hidden point up to the boundary. This new vector is in the same direction as the vector that crosses the boundary.

Once a nonhidden vector is established, subroutine TRANS1 is called to transform the vector into the viewpoint reference frame. Subroutine PNTPLT projects the vector that is now in the viewpoint system. This projection is similar to the projection of a lens onto a projection plane and is represented by the following equations:

$$X' = \frac{(SFACTOR) * (X)}{Z}$$

$$Y' = \frac{(SFACTOR) * (Y)}{Z}$$

where X' and Y' are the plotting coordinates on the projection plane.

X, Y, Z are the coordinates of the position vector in the viewpoint coordinate system.

SFACTOR is the scale factor for the plot.

Once the plotting coordinates of the vector have been calculated, the X and Y coordinates are checked to see if the plot move will take the pen off the boundaries of the plotter. The X coordinate is compared to the variables XMIN and XMAX and the Y coordinate is compared to YMIN and YMAX. These variables represent the bottom, top, left, and right boundary values of the plotting region and are defined in subroutine. If X = Y coordinate is off the plotter, subroutine CLIP is called to compensate for what the pen was to have done. The X and Y coordinates of this intended pen move are saved and the pen variable is set to lift (IPEN = 3) to prevent any drawing until the pen returns to the plotting region. If the X and Y coordinates were in the plotting region, subroutine PREPLT is called. PREPLT completes any pen moves made during clipping. The Calcomp subroutine PLOT is then called to perform the pen move or draw.

4.22 SUBROUTINE POLYD

a. Purpose

The purpose of POLYD is to generate direction cosine matrices for all polygons.

b. Subroutines Called

CROSS, SQRT

c. Labeled Common Blocks Used

ELLIPSE, POLYGON

d. Input or Argument Parameters

None

e. Optional Output

None

f. Procedure

POLYD generates a direction cosine matrix for all polygons in View. The approach taken here is simply to cross two adjacent sides of the polygon to obtain the normal to the polygon surface--this is one coordinate vector. One of the sides used in the cross product is picked as another coordinate vector. The third coordinate vector is obtained by crossing the first two. A direction cosine matrix is obtained by placing these three vectors in a matrix. (For the following equations, see Figure 4.3.)

$$\vec{r}_1 = \overline{P_2 - P_1} \times \overline{P_3 - P_1}$$

$$\hat{r}_1 = \frac{\vec{r}_1}{|\vec{r}_1|} = a_1X + b_1Y + c_1Z$$

$$\hat{r}_2 = \frac{\overline{P_2 - P_1}}{|\overline{P_2 - P_1}|} = a_2X + b_2Y + c_2Z$$

$$\hat{r}_3 = \hat{r}_1 \times \hat{r}_2 = a_3X + b_3Y + c_3Z$$

The direction cosine matrix is given by

$$\underline{D} = \begin{bmatrix} \hat{X} \cdot \hat{r}_1 & \hat{Y} \cdot \hat{r}_1 & \hat{Z} \cdot \hat{r}_1 \\ \hat{X} \cdot \hat{r}_2 & \hat{Y} \cdot \hat{r}_2 & \hat{Z} \cdot \hat{r}_2 \\ \hat{X} \cdot \hat{r}_3 & \hat{Y} \cdot \hat{r}_3 & \hat{Z} \cdot \hat{r}_3 \end{bmatrix}$$

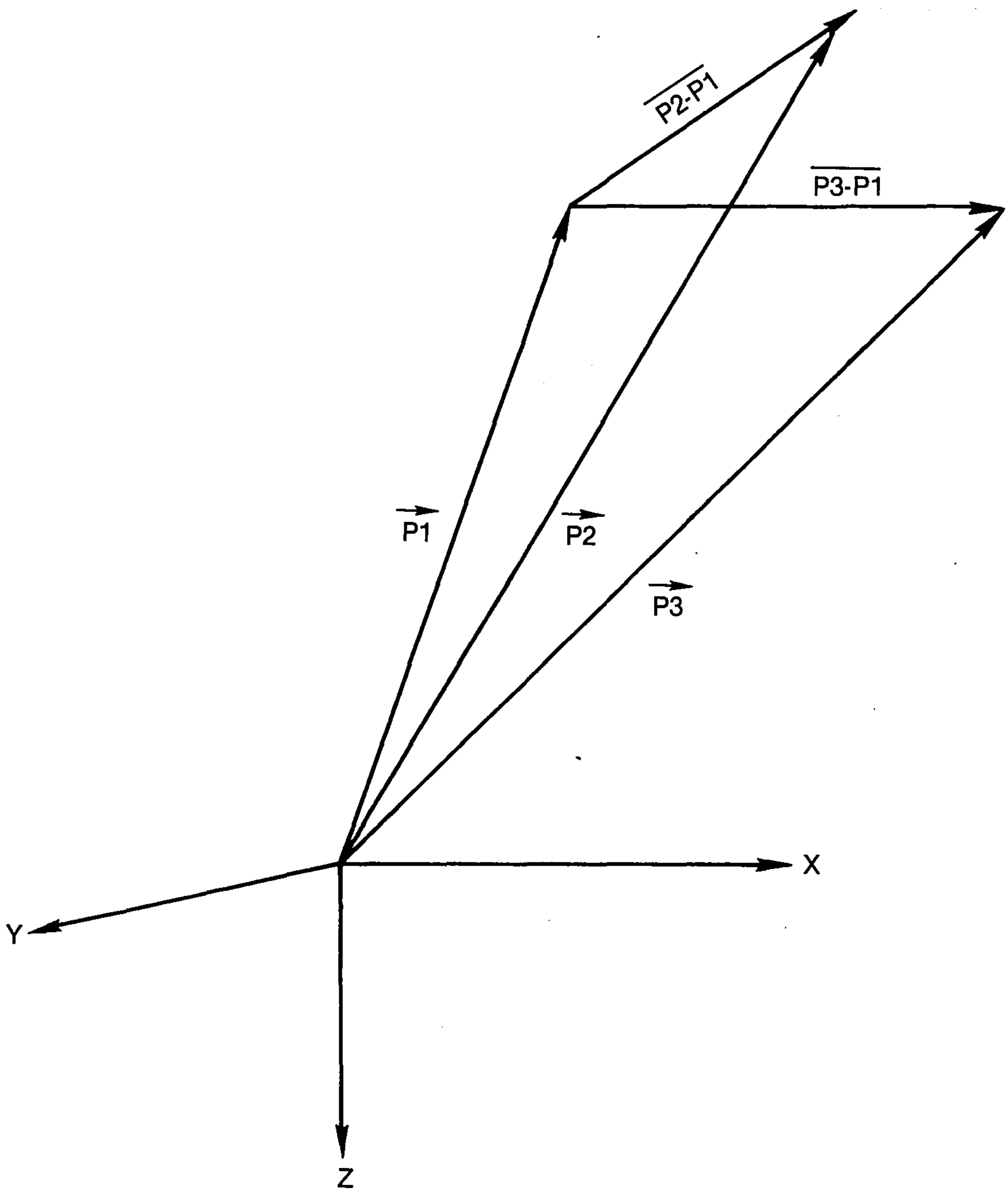


Figure 4.3 Pictorial Representation of Equations in POLYD

Since $\hat{X} \cdot \hat{Y} = 0$ and $\hat{X} \cdot \hat{Z} = 0$ and $\hat{Y} \cdot \hat{Z} = 0$

$$\underline{D} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}$$

This direction cosine matrix can be used to transform from the inertial reference system to the local reference system.

4.23 SUBROUTINE PREPLT

a. Purpose

The purpose of this subroutine is to position the pen before a call to PLOT with pen down if that move would cause the pen to exceed the specified plotting region. The correct position for the pen is at the point denoted by the coordinates of an X or Y intercept between saved and present points at the X or Y boundary, and the X or Y boundary (Figure 4.4).

b. Subroutines Called

PLOT, XINTCP, YINTCP

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

X	--	X coordinate of present point
Y	--	Y coordinate of present point

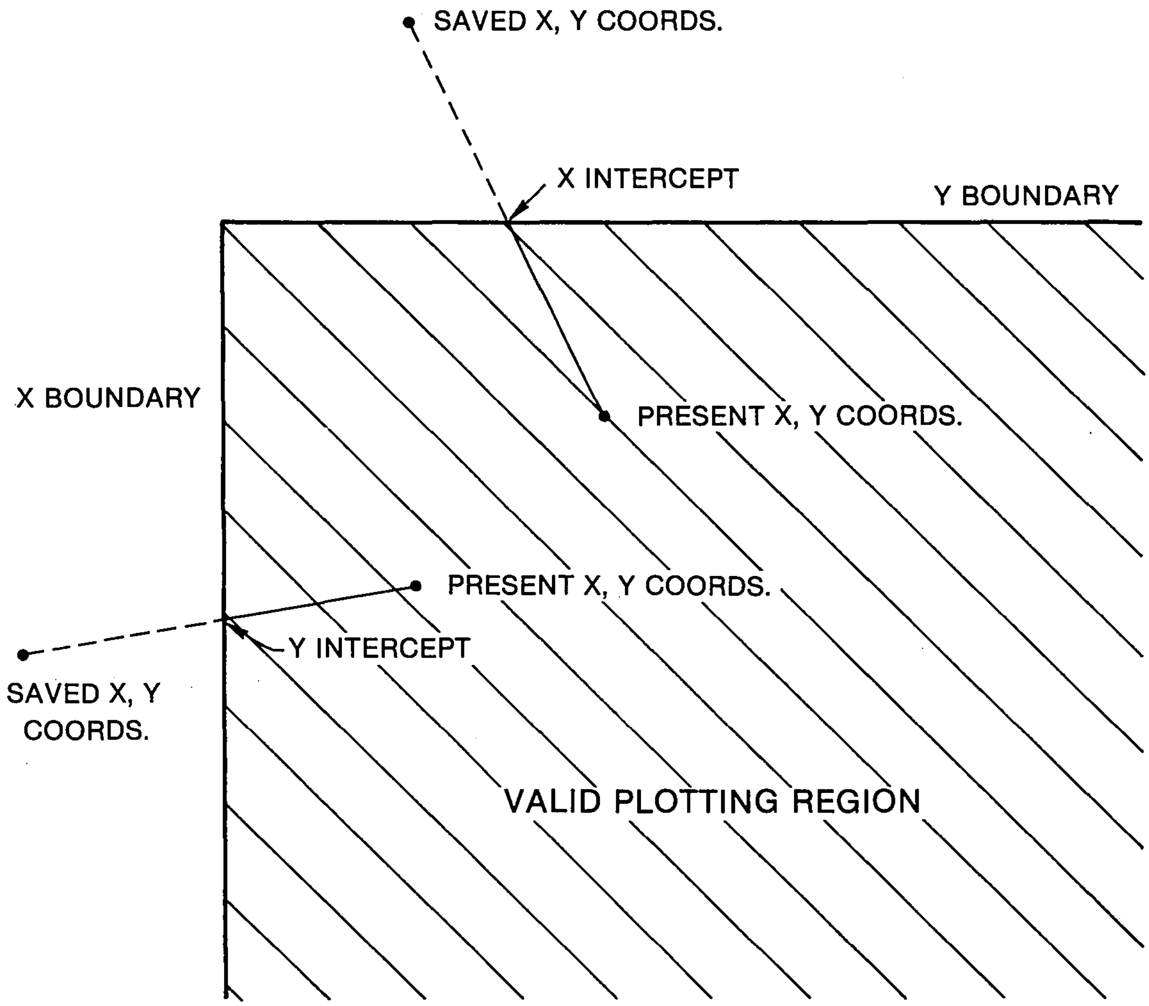


Figure 4.4 Figure for Variables in PREPLT

XSAV	--	X coordinate of previous point
YSAV	--	Y coordinate of previous point
XMIN	--	X coordinate of left side of plotting region
XMAX	--	X coordinate of right side of plotting region
YMIN	--	Y coordinate of bottom of plotting region
YMAX	--	Y coordinate of top of plotting region
IPEN	--	Calcomp pen control value
NEWPEN	--	Saved value of previous IPEN move

e. Optional Output

None

f. Procedure

The method in which PREPLT knows if the previous draw was across the border is a combination of two reasons. The first is the fact that the subroutine is being called. That is, the program flow of subroutine PNTPLT (the only subroutine that calls PREPLT) will only call PREPLT if the present X and Y coordinates are on the plotter. The second reason is determined by checking flag NEWPEN. NEWPEN is a variable that has saved the previous pen move. It is also set to a negative value if the pen move was out of the plotting region. So, if NEWPEN is equal to -2 (negative meaning clipped, 2 meaning pen down), then we want to continue. If it is not equal to -2, the subroutine exits. If continued, PREPLT determines to which coordinates to move the pen. If the X coordinate is outside the valid plotting region, the Y coordinate is set to the Y intercept between the present and saved coordinates at the X boundary. If the X coordinate is inside the valid plotting region, the Y coordinate is set to the Y boundary value. If the Y coordinate is outside the valid plotting region, the X coordinate is set to the X intercept value between the present and saved points at the Y boundary. If the Y coordinate was inside the valid plotting region, the X coordinate is set to the X boundary value. PREPLT moves the pen (up) to this spot denoted by the just calculated X and Y coordinates. Variable IPEN is set to 2 and the subroutine exits.

4.24 SUBROUTINE PRJELR

a. Purpose

This subroutine projects ellipsoids onto the projection plane and circumscribes the projected shadow of the ellipsoid with a rectangle. This rectangle is used later in the program by the BUILDIE routine which checks for overlaps of objects on the projection plane. The information obtained from these operations is used by the hidden line routines to decrease computation time.

The PRJELR routine assumes that an ellipsoid projects onto the projection plane as an ellipse, but in general this is not the case. The assumption used in writing the VIEW program is that when the viewpoint is sufficiently far away from the subject and the viewpoint coordinate system is looking almost directly at the subject, all ellipsoids will project approximately as ellipses.

b. Subroutines Called

ABS, SQRT, DOT, DOTT, MAT, SOLVA, SOLVR

c. Labeled Common Blocks Used

ELLIPSE, POLYGON

d. Input or Argument Parameters

None

e. Optional Output

None

f. Procedure

Immediately upon entering, PRJELR checks the number of segments. If zero, the subroutine exits. If nonzero, it continues. The following procedure is done for each segment. The A array is transformed into the viewpoint reference frame. Subroutine SOLVR is called three times to get the vectors $\vec{r}_1$, $\vec{r}_2$, and $\vec{r}_3$ as described in Appendix B. Vectors $\vec{r}_1$, $\vec{r}_2$, and $\vec{r}_3$ are projected onto the projection plane. Subroutine SOLVA is called to get the coefficients of the ellipse matrix. Eigenvalues LAMDA1 and LAMDA2 are solved. The two Eigenvalues are constructed and normalized. The SIGN and CONVEC arrays are set up. After doing this procedure for each segment, the subroutine exits.

4.25 SUBROUTINE PRJPLY

a. Purpose

PRJPLY is called by the Main program to project polygons in three space to the two-dimensional space of the projection plane.

b. Subroutines Called

MAT

c. Labeled Common Blocks Used

ELLIPSE, POLYGON

d. Input or Argument Parameters

None

e. Optional Output

None

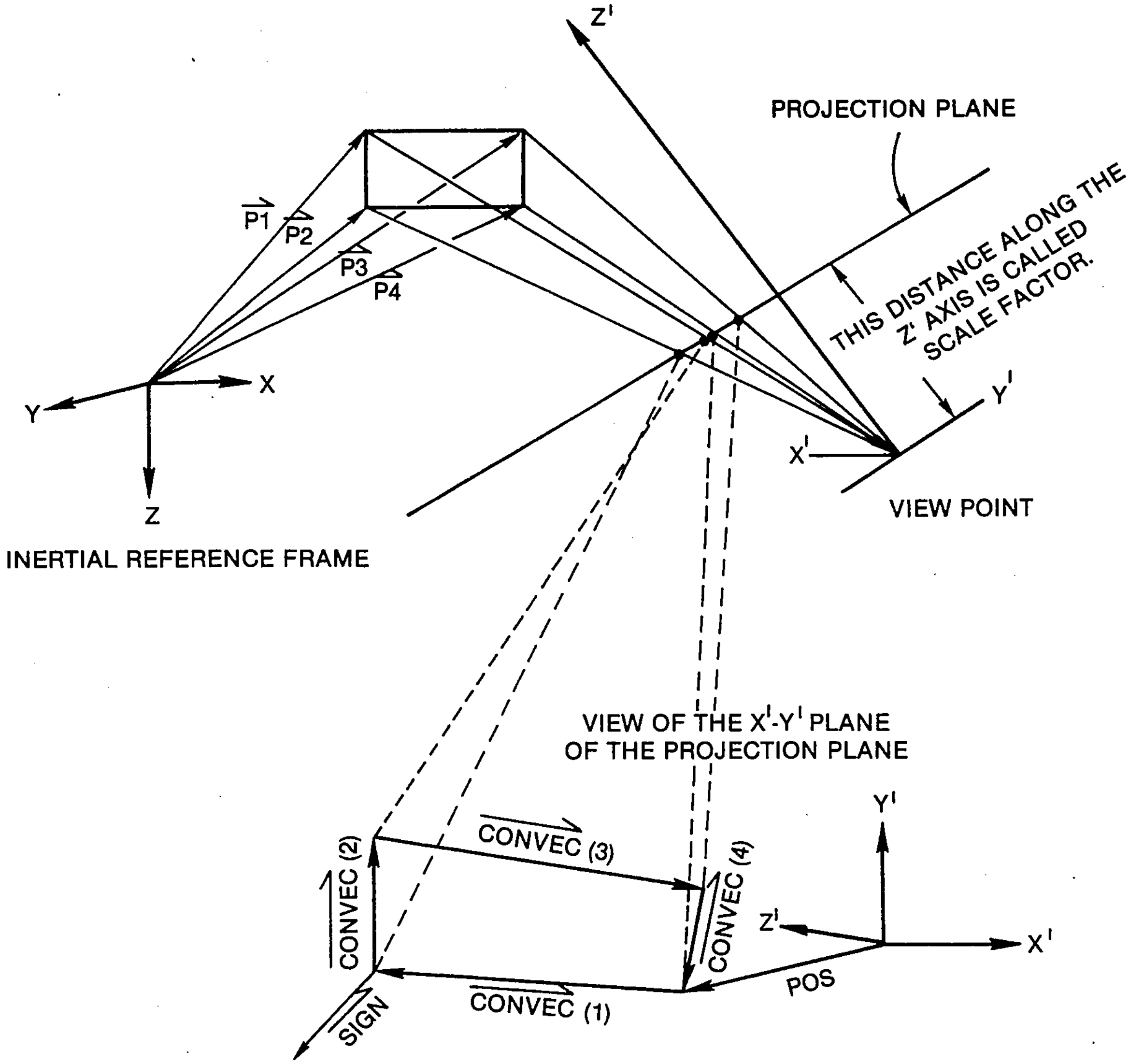
f. Procedure

PRJPLY works with one plane at a time and fills the POS, CONVEC, and SIGN arrays for that plane; then the next plane is used until all planes have been converted. A pictorial representation of the meaning of the vectors in the arrays is given in Figure 4.5. The $\vec{P}$ vectors are given in the inertial reference frame. These vectors define the corners of the polygon in three space. These $\vec{P}$ vectors are projected onto the projection plane, and the CONVEC array is generated. The CONVEC array consists of contour vectors of the projected polygon. The POS array contains a vector that defines one corner of the projected polygon from a coordinate system aligned with the viewpoint coordinate system. The SIGN array contains the result of CONVEC(1) crossed with CONVEC(2). The CONVEC and SIGN arrays are used by the BUILDIE block.

4.26 SUBROUTINE PSE

a. Purpose

This subroutine is called from the Main program to set up arrays containing semiellipsoid data for subroutine PNTPLT to plot. Subroutine PSE is called twice for each ellipsoid, once to plot its top half, the second time to plot the bottom half.



$\overrightarrow{\text{SIGN}} = \overrightarrow{\text{CONVEC (1)}} \times \overrightarrow{\text{CONVEC (2)}}$

Figure 4.5 Meaning of Vectors in Arrays in PRJPLY

b. Subroutines Called

PNTPLT

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

X1	--	Semiellipsoid contour array.
IN	--	Number of points in each quarter contour saved in array X1.
SEG	--	Array containing a complete contour.
INDEX	--	Number of steps plus one.
INDEX2	--	Maximum number of points any complete contour can have.
IHALF	--	Flag controlling which ellipsoid half is to be plotted. IHALF = 1, semiellipsoid with X > 0 is plotted. IHALF = 2, semiellipsoid with X < 0 is plotted.

e. Procedure

Contours are plotted in sequence starting with the contour that is represented by just a point (i.e., Y = 0, Z = 0, X > 0) until all contours have been plotted. The last point plotted is represented by a point (i.e., Y = 0, Z = 0, X < 0).

4.27 SUBROUTINE ROT

a. Purpose

Computes rotation matrix $\underline{A}$ for angle TH about X, Y, or Z axes as L = 1, 2, or 3.

b. Subroutines Called

COS, SIN

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

A -- 3 × 3 rotation matrix to be computed.

L -- 1, 2, or 3 indicating rotation about X, Y, or Z axes, respectively.

TH -- Angle of rotation θ, in radians.

e. Optional Output

None

f. Procedure

1. For L = 1, computes

$$\underline{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix}$$

2. For L = 2, computes

$$\underline{A} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix}$$

3. For L = 3, computes

$$\underline{A} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Note: Special tests are performed to insure that cos θ and sin θ are exactly 0 or ±1 for values of θ that are multiples of $\frac{\pi}{2}$ to correct for small errors introduced by the SIN and COS routines.

4.28 SUBROUTINE SOLVA

a. Purpose

SOLVA solves three equations simultaneously and returns the components of [α]. See Appendix B, "Discussion of Equations Used by PRJELR" for more information.

b. Subroutines Called

None

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

R	--	Contains values of r_1, r_2, and r_3 on the projection plane.
AA11	--	α_{11} component of [α]
AA22	--	α_{22} component of [α]
AA12	--	α_{12} component of [α]

e. Optional Output

None

f. Procedure

See Appendix B, "Discussion of Equations Used by PRJELR" for procedural information.

4.29 SUBROUTINE SOLVR

a. Purpose

SOLVR solves a set of simultaneous equations to find the components of vector $\vec{r}$ that satisfy the properties needed to determine the equation of the projected ellipse. For more information, refer to Appendix B, "Discussion of Equations Used by PRJELR."

b. Subroutines Called

SQRT

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

CALL SOLVR (A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, SS, R1, R3)

Case No. 1	Case No. 2	Case No. 3
$A_1 = A'_{11}$	$A_1 = A'_{12}$	$A_1 = A'_{11} + A'_{12}$
$A_2 = A'_{21}$	$A_2 = A'_{22}$	$A_2 = A'_{21} + A'_{22}$

$A_3 = A'_{31}$	$A_3 = A'_{32}$	$A_3 = A'_{31} + A'_{32}$
$A_4 = A'_{13}$	$A_4 = A'_{13}$	$A_4 = A'_{13}$
$A_5 = A'_{23}$	$A_5 = A'_{23}$	$A_5 = A'_{23}$
$A_6 = A'_{33}$	$A_6 = A'_{33}$	$A_6 = A'_{33}$
$A_7 = A'_{11}$	$A_7 = A'_{22}$	$A_7 = A'_{11} + 2A'_{12} + A'_{22}$
$A_8 = A'_{13}$	$A_8 = A'_{23}$	$A_8 = A'_{13} + A'_{23}$
SS = SS	SS = SS	SS = SS
R1 = X component of r_1	R1 = Y component of r_2	R1 = X and Y component of r_3
R3 = Z component of r_1	R1 = Z component of r_2	R3 = Z component of r_3

e. Optional Output

None

f. Procedure

Refer to Appendix B, "Discussion of Equations Used by PRJELR" for procedural information.

4.30 SUBROUTINE TITLE

a. Purpose

The purpose of this subroutine is to read the title data cards and write them to the plot file.

b. Subroutines Called

NEWPEN, NFRAME, PLOT, SYMBOL

c. Labeled Common Blocks Used

DBUG

d. Input or Argument Parameters

Input cards 2.0 and 2.1 (20)

e. Optional Output

None

f. Procedure

After initializing some variables, the pen is moved (up) to point 0,0 and is defined as the origin. Data card 2.0 is read. If number of title frames equal to zero, the subroutine returns. The following procedure is repeated for each title frame. The set of 20 data card 2.1's are read in and SYMBOL is called to write them to the plot file. Depending on the value of DEVFLG, either PLOT or NFRAME is called to ready the plot file for the first flame. After doing all the frames, the subroutine exits.

4.31 SUBROUTINE TPOINT

a. Purpose

This subroutine tests a point against a polygon, both being on the projection plane. The results of the tests indicate if the point lies inside or outside of the polygon. IN is a flag that is returned to the calling program to indicate the final result.

b. Subroutines Called

None

c. Labeled Common Blocks Used

POLYGON

d. Input or Argument Parameters

PP2	--	Point on the projection plane.
I	--	Polygon number on the projection plane.
IN	--	Flag returned telling if point was inside (IN=1) or outside (IN=2) polygon.

e. Optional Output

None

f. Procedure

The test used is called the CROSS PRODUCT TEST (see Figure 4.6). This name is appropriate since the test is based upon the sign of the result of a cross product between two vectors on the projection plane. One vector represents a side of the polygon, and the other vector always extends from the base of the side vector to the point being tested. Examples of these vectors are given in Figure 4.6. The vector that is represented by points AB is crossed into the vector represented by points AE. This process is continued until all sides have been crossed with a vector to the point being examined. In the case of point number 2, the following cross products would be examined.

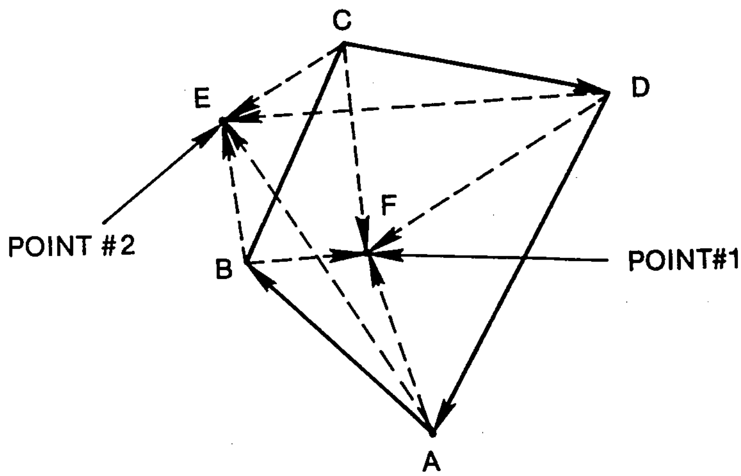


Figure 4.6 Cross-Product Test with Convex Polygon in TPOINT

$$\overrightarrow{AB} \times \overrightarrow{AE},\ \overrightarrow{BC} \times \overrightarrow{BE},\ \overrightarrow{CD} \times \overrightarrow{CE},\ \overrightarrow{DA} \times \overrightarrow{DE}$$

Notice that the first cross product gives a different sign than the second. If a point is outside the polygon, there must be a change in the resulting sign of the cross products. If the point is inside the polygon, the following cross products need to be examined.

$$\overrightarrow{AB} \times \overrightarrow{AF},\ \overrightarrow{BC} \times \overrightarrow{BF},\ \overrightarrow{CD} \times \overrightarrow{CF},\ \overrightarrow{DA} \times \overrightarrow{DF}$$

Notice that all cross products have the same sign. This will be true for any polygon except for concave polygons. In the case of concave polygons (see Figure 4.7), even though the point is inside the polygon, the test would indicate that the point was outside the polygon because there is a sign change between

$$\overrightarrow{AB} \times \overrightarrow{AE} \text{ and } \overrightarrow{BC} \times \overrightarrow{BE}$$

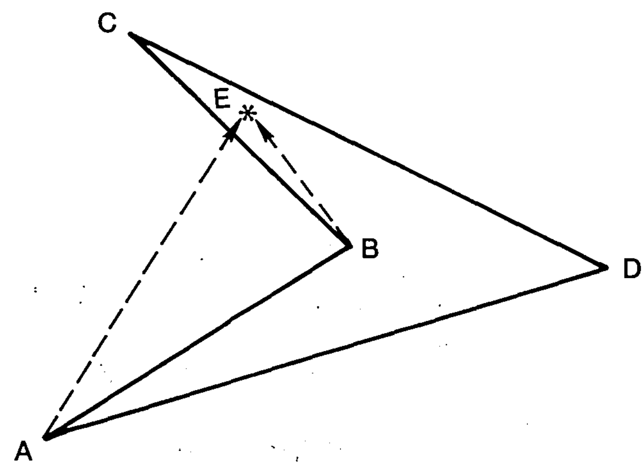


Figure 4.7 Cross-Product Test with Concave Polygon in TPOINT

Therefore, concave polygons are ruled out. Concave polygon shapes can be included only when that concave polygon is represented by convex polygons.

4.32 SUBROUTINE TRANS1

a. Purpose

The purpose of TRANS1 is to transform the input vector $\vec{R}$ into the viewpoint reference frame.

b. Subroutines Called

DOTT, MAT

c. Labeled Common Blocks Used

ELLIPSE, VIEWP

d. Input or Argument Parameters

R	--	Input vector.
P	--	Output vector (input vector transformed to Viewpoint Reference Frame).

e. Optional Output

None

f. Procedure

The vectors used by TRANS1 are shown in Figure 4.8. The output of TRANS1 is vector $\vec{P}$ in the Viewpoint Reference Frame. The other vectors are in the reference frame as specified below.

$\vec{R}$ in Local Reference Frame of Ellipsoid.
This vector is a position vector for surface points of an ellipsoid, and it originates at the center of the ellipsoid.

$\overrightarrow{SEGLP}$ in Inertial Reference Frame.
This vector is a position vector for the center of the contact ellipsoid associated with $\vec{R}$.

$\overrightarrow{VP}$ in the Inertial Reference Frame.
This vector is a position vector for the origin of the Viewpoint Coordinate System.

$\vec{P}$ can be found from vectors $\vec{R}$, $\overrightarrow{SEGLP}$, and $\overrightarrow{VP}$ once they are all in the Viewpoint Reference Frame. The following equations transform these vectors into the Viewpoint Reference Frame.

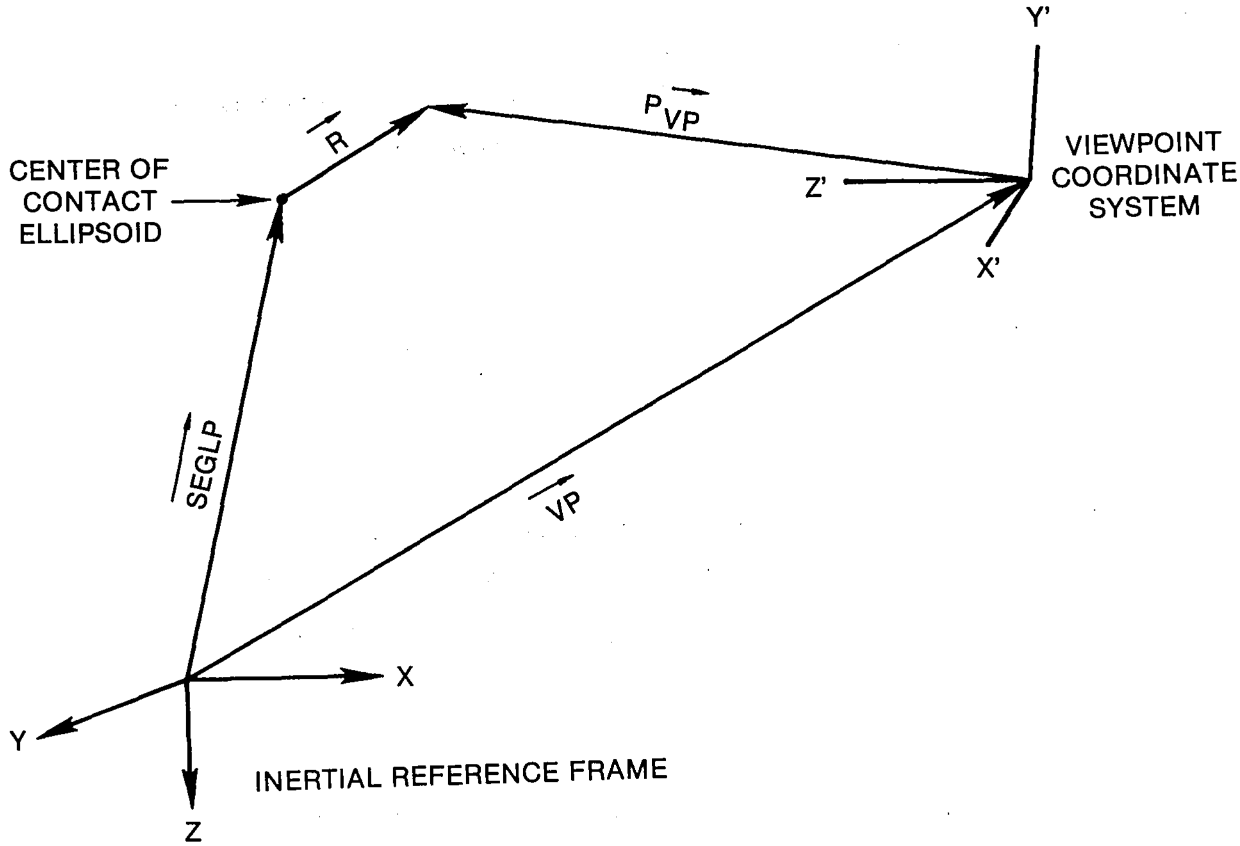


Figure 4.8. Vectors Used by TRANS1

$$\overrightarrow{R2} = [\underline{DVP}] \left[\underline{D}^T\right] \vec{R}$$

$$\overrightarrow{SEGLP2} = [\underline{DVP}] \overrightarrow{SEGLP}$$

$$\overrightarrow{VP2} = [\underline{DVP}] \overrightarrow{VP}$$

Note: [DVP] is the direction cosine matrix that transforms from the inertial to the viewpoint frame of reference.
[D] is the direction cosine matrix that transforms from the inertial to the viewpoint frame of reference.

Then $\vec{P}$ is given by

$$\vec{P} = \overline{SEGLP\vec{2}} + \overline{R\vec{2}} - \overline{VP\vec{2}}$$

IF IELP is greater than 30, TRANS1 must work with a vector on a polygon and not an ellipsoid. The vectors that define the sides of the polygon are always in the Inertial Reference Frame; therefore, DVP is placed in DD matrix, and the same equations are used to find $\vec{P}$.

4.33 FUNCTION XINTCP

a. Purpose

The purpose of XINTCP is as follows: given two sets of X and Y coordinates and a Y coordinate that falls between them, XINTCP calculates the X coordinate at the given Y value.

b. Subroutines Called

None

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

X	--	X coordinate of end point No. 1.
Y	--	Y coordinate of end point No. 1.
XSAV	--	X coordinate of end point No. 2.
YSAV	--	Y coordinate of end point No. 2.
YTEMP	--	Y coordinate of the point at which the caller wants the X coordinate.

e. Optional Output

None

f. Procedure

XINTCP has two input pairs of coordinates. In this discussion they will be referred to the saved (XSAV, YSAV) and present (X,Y) coordinates (see Figure 4.9). XINTCP first calculates the differences between the saved and present X and Y coordinates. These differences are called X1 and Y1. It then calculates the scaling factor SFACTR as

$$\frac{X1}{Y1}$$

If Y1 is equal to zero, PFACTR is set to zero. The difference between the saved and intercept Y value is calculated (Y2). This difference is multiplied by PFACTR and added to the saved X value. This is the X intercept value. It is set equal to XINTCP and the function returns.

4.34 SUBROUTINE XYZ

a. Purpose

XYZ solves the equation for an ellipse to find a point on that ellipse that will return a vector. It is used when M_1 and M_2 are not equal to zero. Refer to Appendix A, "Hidden Line Problem Between Two Ellipsoids."

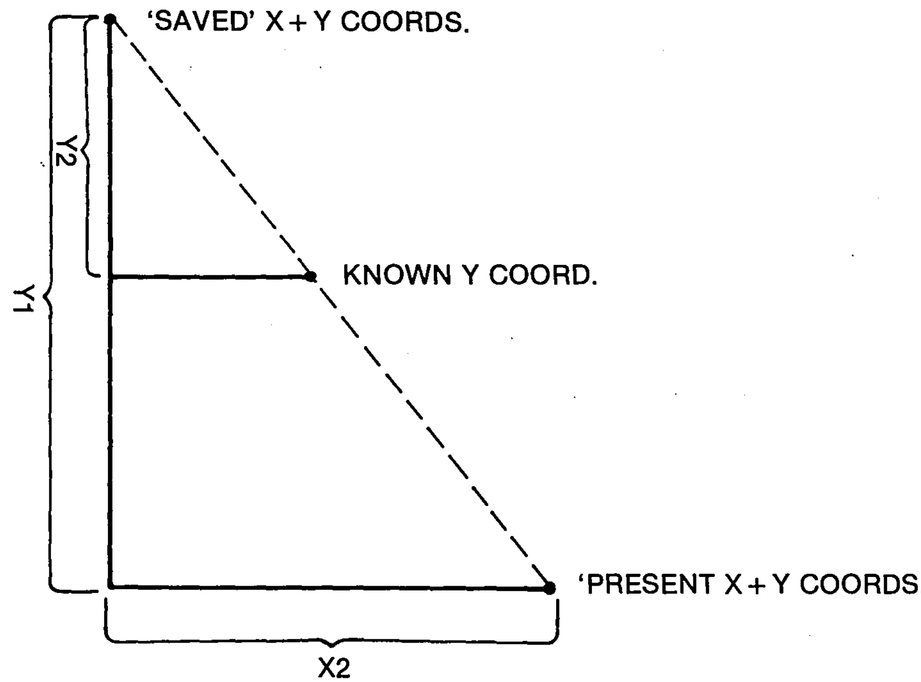


Figure 4.9. Variables Used by XINTCP

b. Subroutines Called

SQRT

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

MU	--	See Appendix A equations for definitions.
A	--	
B	--	
C	--	
S	--	
M	--	
JFLAG	--	Flag passed back as either zero when equation solved or one when not solved.

e. Optional Output

None

f. Procedure

Follow equations in Appendix A for procedural information.

4.35 FUNCTION YINTCP

a. Purpose

The purpose of YINTCP is as follows: given two sets of X and Y coordinates and an X coordinate that falls between them, YINTCP calculates the Y coordinate at the given X value.

b. Subroutines Called

None

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

X	--	X coordinate of point No. 1.
Y	--	Y coordinate of point No. 1.
XSAV	--	X coordinate of point No. 2.
YSAV	--	Y coordinate of point No. 2.
XTEMP	--	X coordinate of the point at which the caller wants the Y coordinate.

e. Optional Output

None

f. Procedure

YINTCP has two input pairs of coordinates. In this discussion they will be referred to as the present (X,Y) and saved (XSAV, YSAV) coordinates (see Figure 4.10). YINTCP first calculates the differences between the present and saved X and Y coordinates. These differences are called X1 and Y1. It then calculates the scaling factor SFACTR as

$$\frac{Y1}{X1}$$

If Y1 is equal to zero, PFACTR is set to zero. The difference between the saved and intercept X values is calculated (X2). The difference (X2) is multiplied by PFACTR and added to the saved Y coordinate. This is the Y intercept value. It is set equal to YINTCP and the function returns.

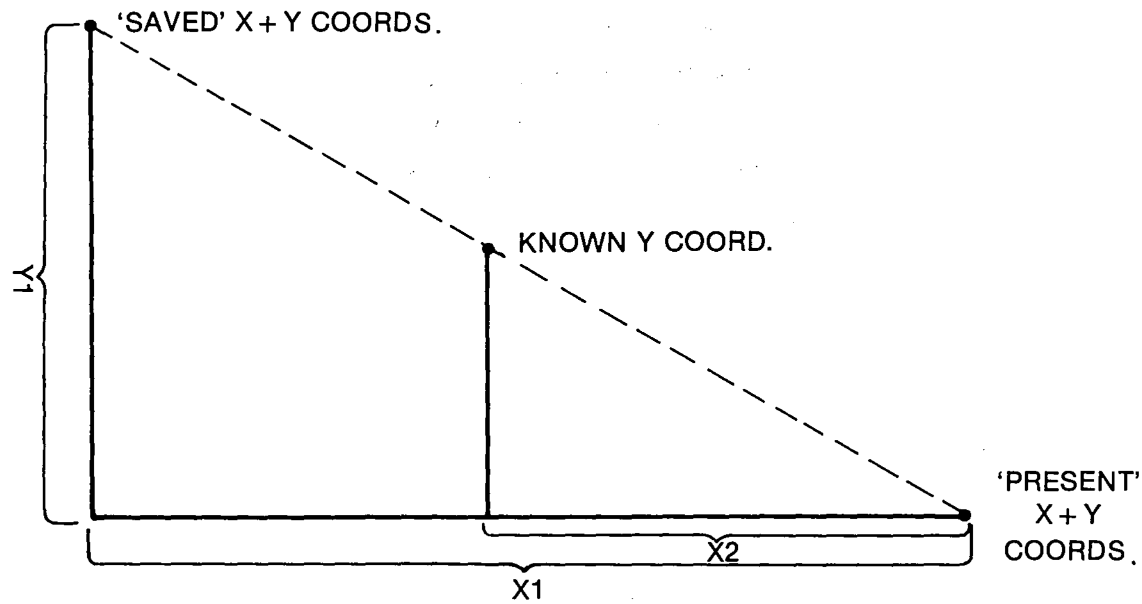


Figure 4.10. Variables used by YINTCP

4.36 SUBROUTINE YZ

a. Purpose

YZ solves the equation for an ellipse to find a point on that ellipse that will return a vector. Is used when $M_1 = 0$ and $M_2 = 0$. Refer to Appendix A, "Hidden Line Problem Between Two Ellipsoids."

b. Subroutines Called

SQRT

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

MV	--	See Appendix A equations for definitions.
A	--	
B	--	
C	--	
S	--	
M	--	
JFLAG	--	Flag passed back as either zero when equation solved or one if not solved.

e. Optional Output

None

f. Procedure

Follow equations in Appendix A for procedural information (Case No. 2).

4.37 SUBROUTINE Z

a. Purpose

Z solves the equation for an ellipse to find a point on that ellipse that will return a vector. Is used when both M_1 and M_2 are equal to zero. Refer to Appendix A, "Hidden Line Problem Between Two Ellipsoids."

b. Subroutines Called

SQRT

c. Labeled Common Blocks Used

None

d. Input or Argument Parameters

MV	--	See Appendix A equations for definitions.
A	--	
B	--	
C	--	
S	--	
M	--	
JFLAG	--	Flag passed back as either zero when equation solved or one if not solved.

e. Input or Argument Parameters

None

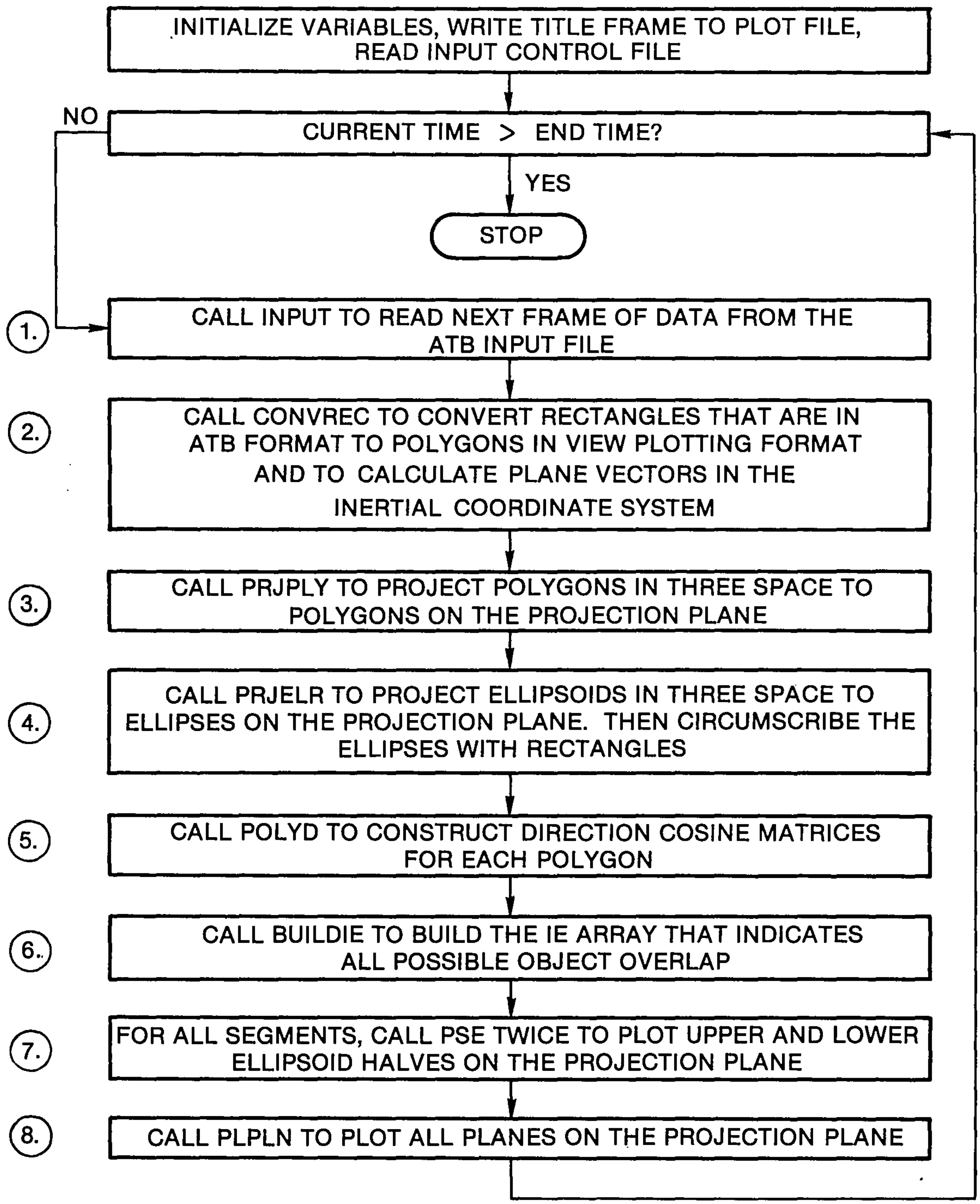
f. Procedure

Follow equations in Appendix A for procedural information (Case No. 1).

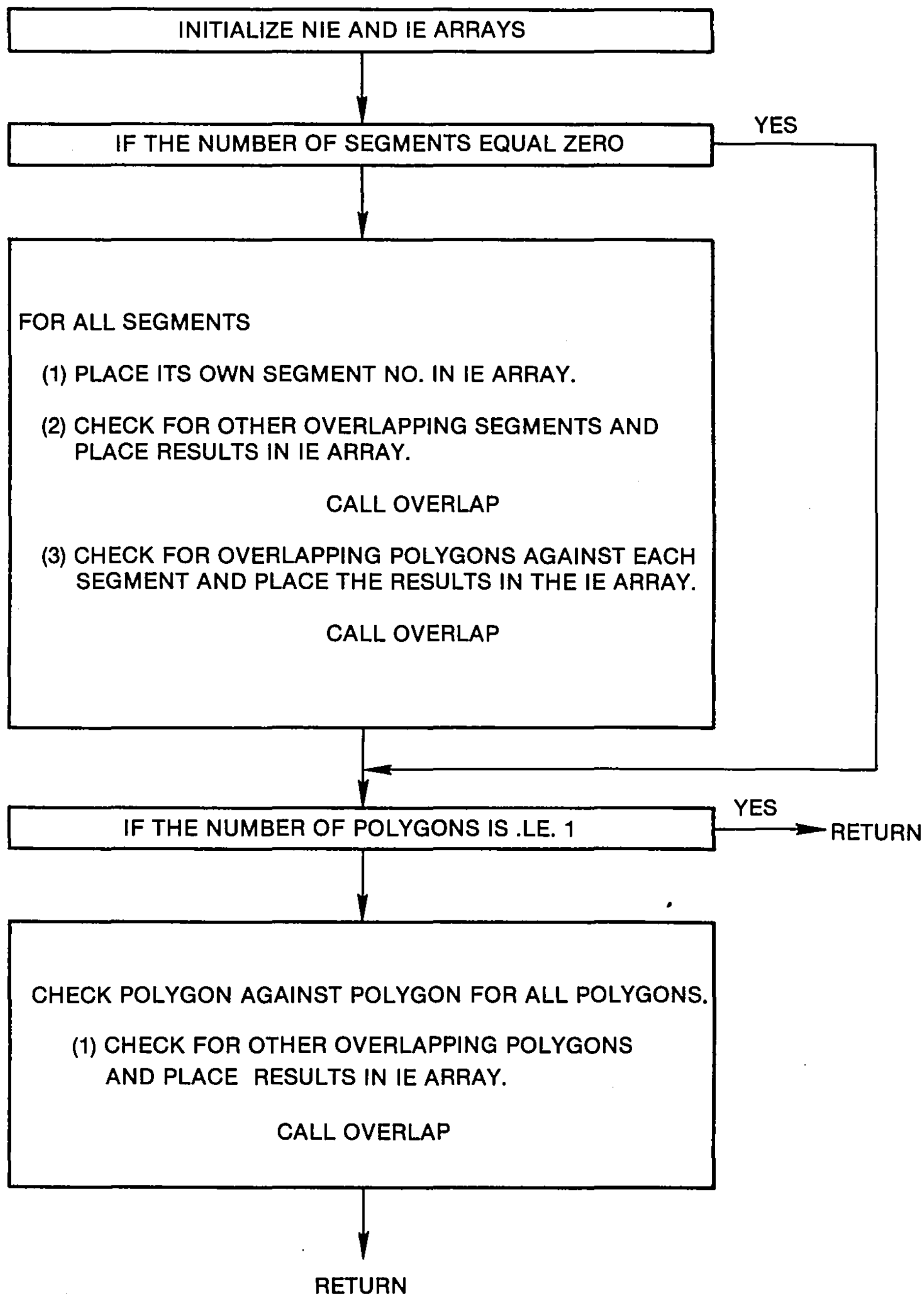
5.0 VIEW PROGRAM FLOWCHARTS

Contained in this section are the flowcharts for selected program modules. They are not direct, line-by-line flowcharts, but more of a general flow description of critical program modules.

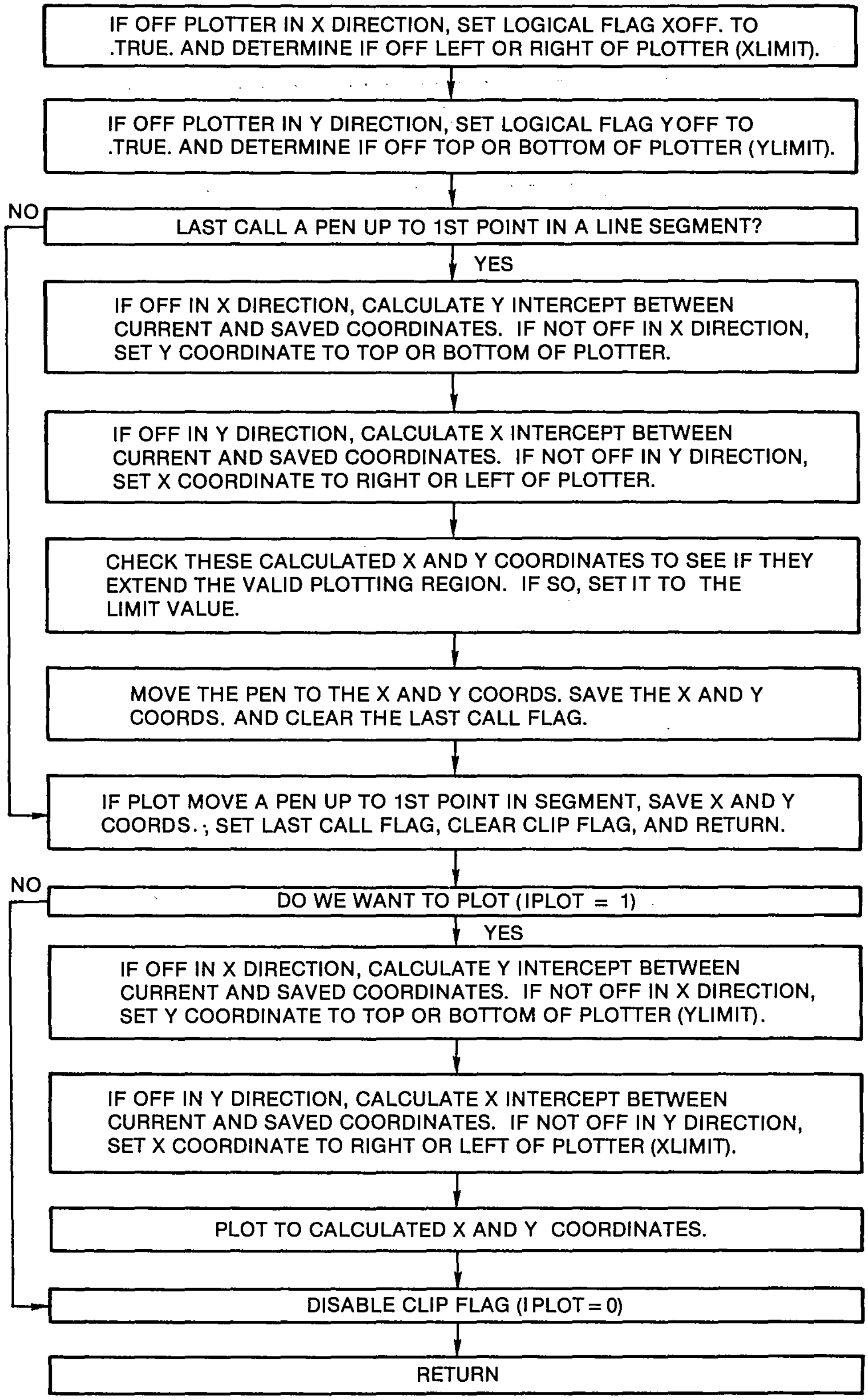
5.1 VIEW MAIN


INITIALIZE VARIABLES, WRITE TITLE FRAME TO PLOT FILE, READ INPUT CONTROL FILE
NO
CURRENT TIME > END TIME?
YES
STOP
1.
CALL INPUT TO READ NEXT FRAME OF DATA FROM THE ATB INPUT FILE
2.
CALL CONVREC TO CONVERT RECTANGLES THAT ARE IN ATB FORMAT TO POLYGONS IN VIEW PLOTTING FORMAT AND TO CALCULATE PLANE VECTORS IN THE INERTIAL COORDINATE SYSTEM
3.
CALL PRJPLY TO PROJECT POLYGONS IN THREE SPACE TO POLYGONS ON THE PROJECTION PLANE
4.
CALL PRJELR TO PROJECT ELLIPSOIDS IN THREE SPACE TO ELLIPSES ON THE PROJECTION PLANE. THEN CIRCUMSCRIBE THE ELLIPSES WITH RECTANGLES
5.
CALL POLYD TO CONSTRUCT DIRECTION COSINE MATRICES FOR EACH POLYGON
6.
CALL BUILDIE TO BUILD THE IE ARRAY THAT INDICATES ALL POSSIBLE OBJECT OVERLAP
7.
FOR ALL SEGMENTS, CALL PSE TWICE TO PLOT UPPER AND LOWER ELLIPSOID HALVES ON THE PROJECTION PLANE
8.
CALL PLPLN TO PLOT ALL PLANES ON THE PROJECTION PLANE

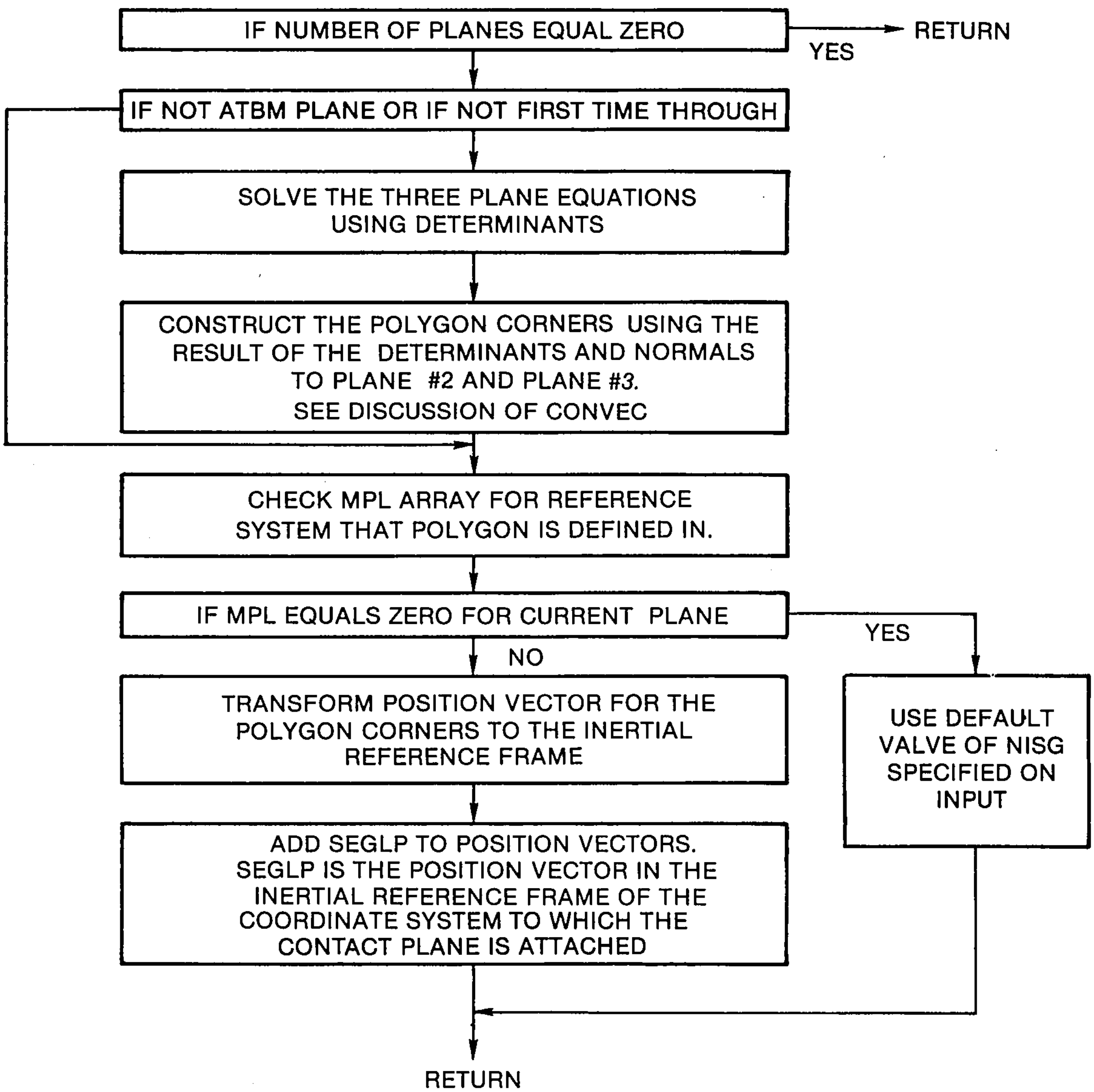
5.2 BUILDIE


INITIALIZE NIE AND IE ARRAYS
IF THE NUMBER OF SEGMENTS EQUAL ZERO
YES
FOR ALL SEGMENTS
(1) PLACE ITS OWN SEGMENT NO. IN IE ARRAY.
(2) CHECK FOR OTHER OVERLAPPING SEGMENTS AND PLACE RESULTS IN IE ARRAY.
CALL OVERLAP
(3) CHECK FOR OVERLAPPING POLYGONS AGAINST EACH SEGMENT AND PLACE THE RESULTS IN THE IE ARRAY.
CALL OVERLAP
IF THE NUMBER OF POLYGONS IS .LE. 1
YES
RETURN
CHECK POLYGON AGAINST POLYGON FOR ALL POLYGONS.
(1) CHECK FOR OTHER OVERLAPPING POLYGONS AND PLACE RESULTS IN IE ARRAY.
CALL OVERLAP
RETURN

5.3 CLIP

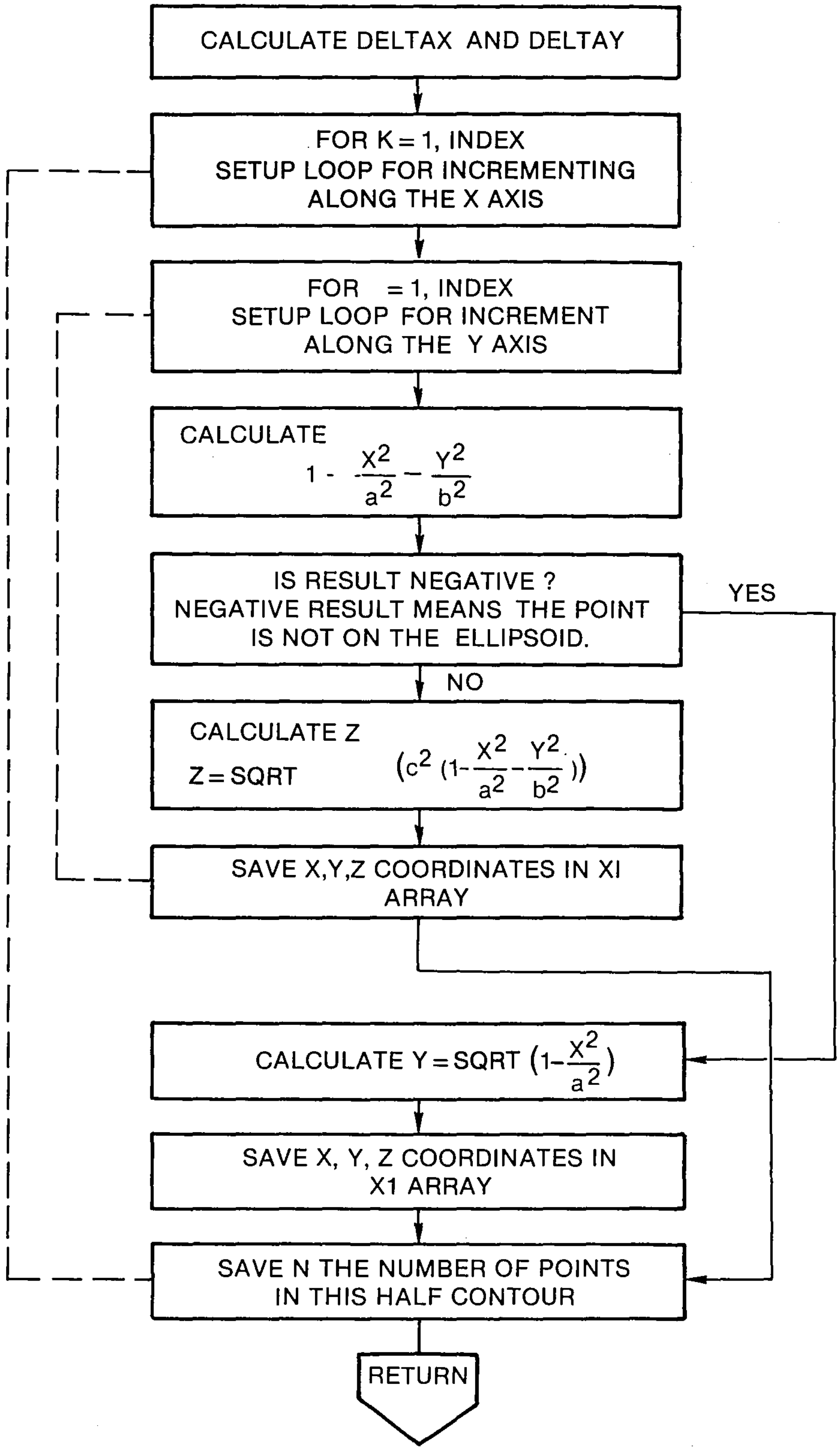

IF OFF PLOTTER IN X DIRECTION, SET LOGICAL FLAG XOFF. TO .TRUE. AND DETERMINE IF OFF LEFT OR RIGHT OF PLOTTER (XLIMIT).
IF OFF PLOTTER IN Y DIRECTION, SET LOGICAL FLAG YOFF TO .TRUE. AND DETERMINE IF OFF TOP OR BOTTOM OF PLOTTER (YLIMIT).
NO
LAST CALL A PEN UP TO 1ST POINT IN A LINE SEGMENT?
YES
IF OFF IN X DIRECTION, CALCULATE Y INTERCEPT BETWEEN CURRENT AND SAVED COORDINATES. IF NOT OFF IN X DIRECTION, SET Y COORDINATE TO TOP OR BOTTOM OF PLOTTER.
IF OFF IN Y DIRECTION, CALCULATE X INTERCEPT BETWEEN CURRENT AND SAVED COORDINATES. IF NOT OFF IN Y DIRECTION, SET X COORDINATE TO RIGHT OR LEFT OF PLOTTER.
CHECK THESE CALCULATED X AND Y COORDINATES TO SEE IF THEY EXTEND THE VALID PLOTTING REGION. IF SO, SET IT TO THE LIMIT VALUE.
MOVE THE PEN TO THE X AND Y COORDS. SAVE THE X AND Y COORDS. AND CLEAR THE LAST CALL FLAG.
IF PLOT MOVE A PEN UP TO 1ST POINT IN SEGMENT, SAVE X AND Y COORDS.; SET LAST CALL FLAG, CLEAR CLIP FLAG, AND RETURN.
NO
DO WE WANT TO PLOT (IPLOT = 1)
YES
IF OFF IN X DIRECTION, CALCULATE Y INTERCEPT BETWEEN CURRENT AND SAVED COORDINATES. IF NOT OFF IN X DIRECTION, SET Y COORDINATE TO TOP OR BOTTOM OF PLOTTER (YLIMIT).
IF OFF IN Y DIRECTION, CALCULATE X INTERCEPT BETWEEN CURRENT AND SAVED COORDINATES. IF NOT OFF IN Y DIRECTION, SET X COORDINATE TO RIGHT OR LEFT OF PLOTTER (XLIMIT).
PLOT TO CALCULATED X AND Y COORDINATES.
DISABLE CLIP FLAG (IPLOT = 0)
RETURN

5.4 CONVREC



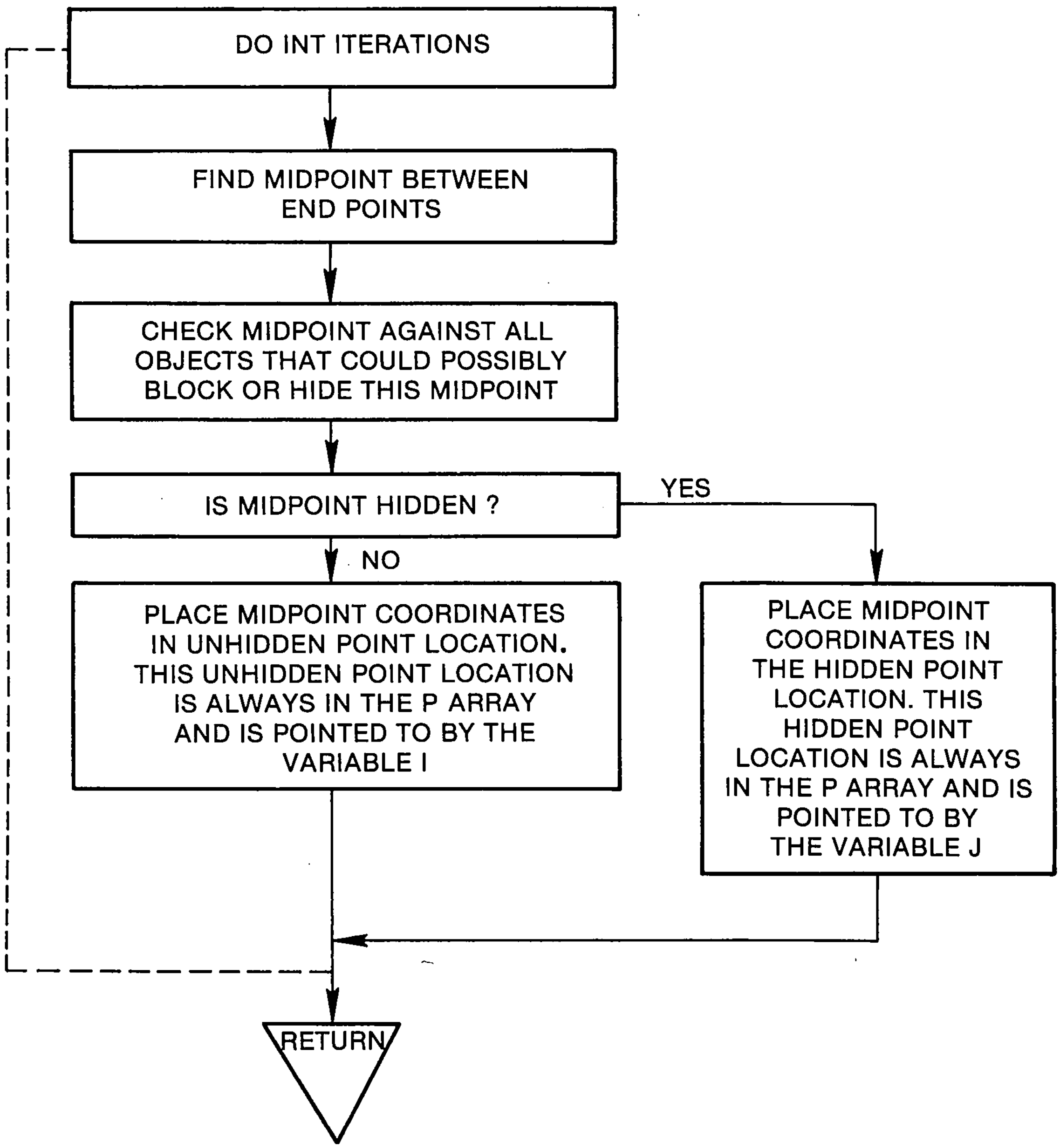
IF NUMBER OF PLANES EQUAL ZERO
YES
RETURN
IF NOT ATBM PLANE OR IF NOT FIRST TIME THROUGH
SOLVE THE THREE PLANE EQUATIONS USING DETERMINANTS
CONSTRUCT THE POLYGON CORNERS USING THE RESULT OF THE DETERMINANTS AND NORMALS TO PLANE #2 AND PLANE #3. SEE DISCUSSION OF CONVEC
CHECK MPL ARRAY FOR REFERENCE SYSTEM THAT POLYGON IS DEFINED IN.
IF MPL EQUALS ZERO FOR CURRENT PLANE
YES
NO
TRANSFORM POSITION VECTOR FOR THE POLYGON CORNERS TO THE INERTIAL REFERENCE FRAME
USE DEFAULT VALVE OF NISG SPECIFIED ON INPUT
ADD SEGLP TO POSITION VECTORS. SEGLP IS THE POSITION VECTOR IN THE INERTIAL REFERENCE FRAME OF THE COORDINATE SYSTEM TO WHICH THE CONTACT PLANE IS ATTACHED
RETURN

5.5 ELIPSN

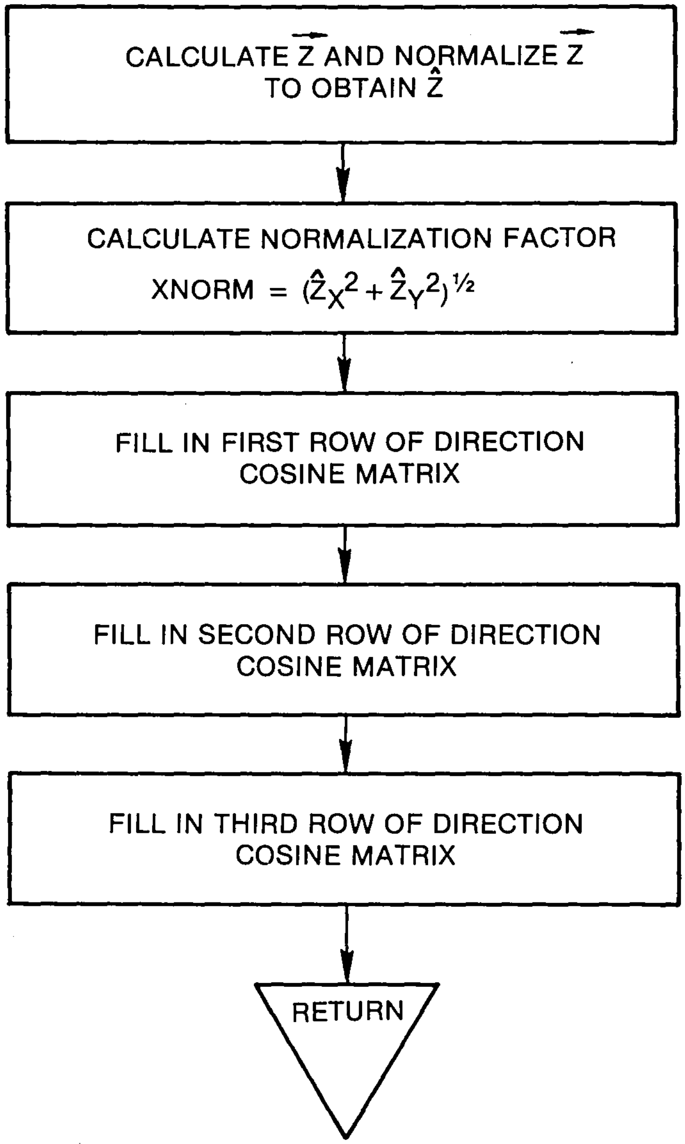


CALCULATE DELTAX AND DELTAY
FOR K = 1, INDEX
SETUP LOOP FOR INCREMENTING ALONG THE X AXIS
FOR = 1, INDEX
SETUP LOOP FOR INCREMENT ALONG THE Y AXIS
CALCULATE
$1 - \frac{X^2}{a^2} - \frac{Y^2}{b^2}$
IS RESULT NEGATIVE ?
NEGATIVE RESULT MEANS THE POINT IS NOT ON THE ELLIPSOID.
YES
NO
CALCULATE Z
Z = SQRT $(c^2 (1-\frac{X^2}{a^2}-\frac{Y^2}{b^2}))$
SAVE X,Y,Z COORDINATES IN XI ARRAY
CALCULATE Y = SQRT $(1-\frac{X^2}{a^2})$
SAVE X, Y, Z COORDINATES IN X1 ARRAY
SAVE N THE NUMBER OF POINTS IN THIS HALF CONTOUR
RETURN

5.6 EXTEND

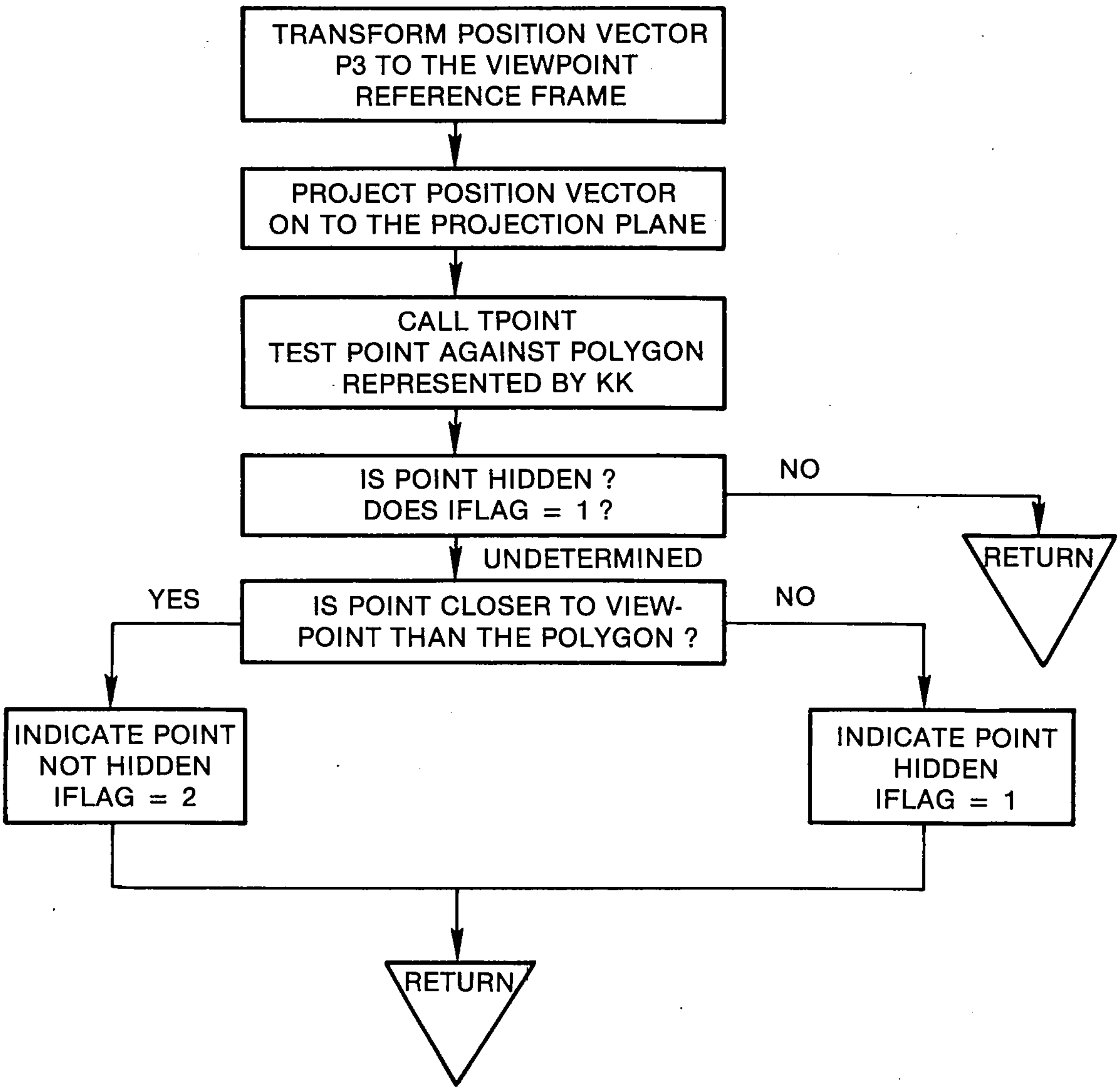

DO INT ITERATIONS
FIND MIDPOINT BETWEEN END POINTS
CHECK MIDPOINT AGAINST ALL OBJECTS THAT COULD POSSIBLY BLOCK OR HIDE THIS MIDPOINT
IS MIDPOINT HIDDEN ?
YES
NO
PLACE MIDPOINT COORDINATES IN UNHIDDEN POINT LOCATION. THIS UNHIDDEN POINT LOCATION IS ALWAYS IN THE P ARRAY AND IS POINTED TO BY THE VARIABLE I
PLACE MIDPOINT COORDINATES IN THE HIDDEN POINT LOCATION. THIS HIDDEN POINT LOCATION IS ALWAYS IN THE P ARRAY AND IS POINTED TO BY THE VARIABLE J
RETURN

5.7 GENDCM



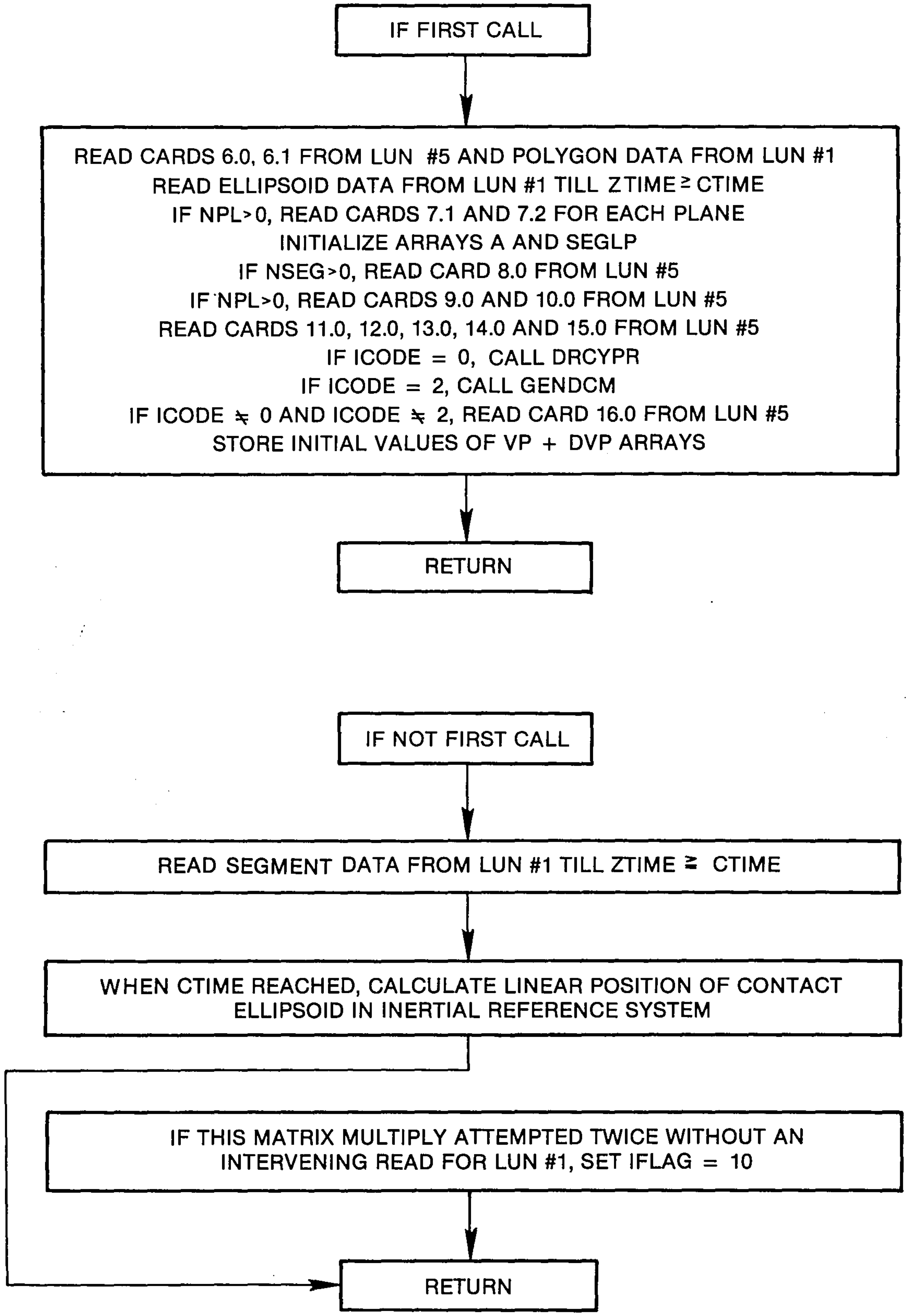
CALCULATE $\vec{Z}$ AND NORMALIZE $\vec{Z}$ TO OBTAIN $\hat{Z}$
CALCULATE NORMALIZATION FACTOR
XNORM = $(\hat{Z}_X^2 + \hat{Z}_Y^2)^{1/2}$
FILL IN FIRST ROW OF DIRECTION COSINE MATRIX
FILL IN SECOND ROW OF DIRECTION COSINE MATRIX
FILL IN THIRD ROW OF DIRECTION COSINE MATRIX
RETURN

5.8 HIDE

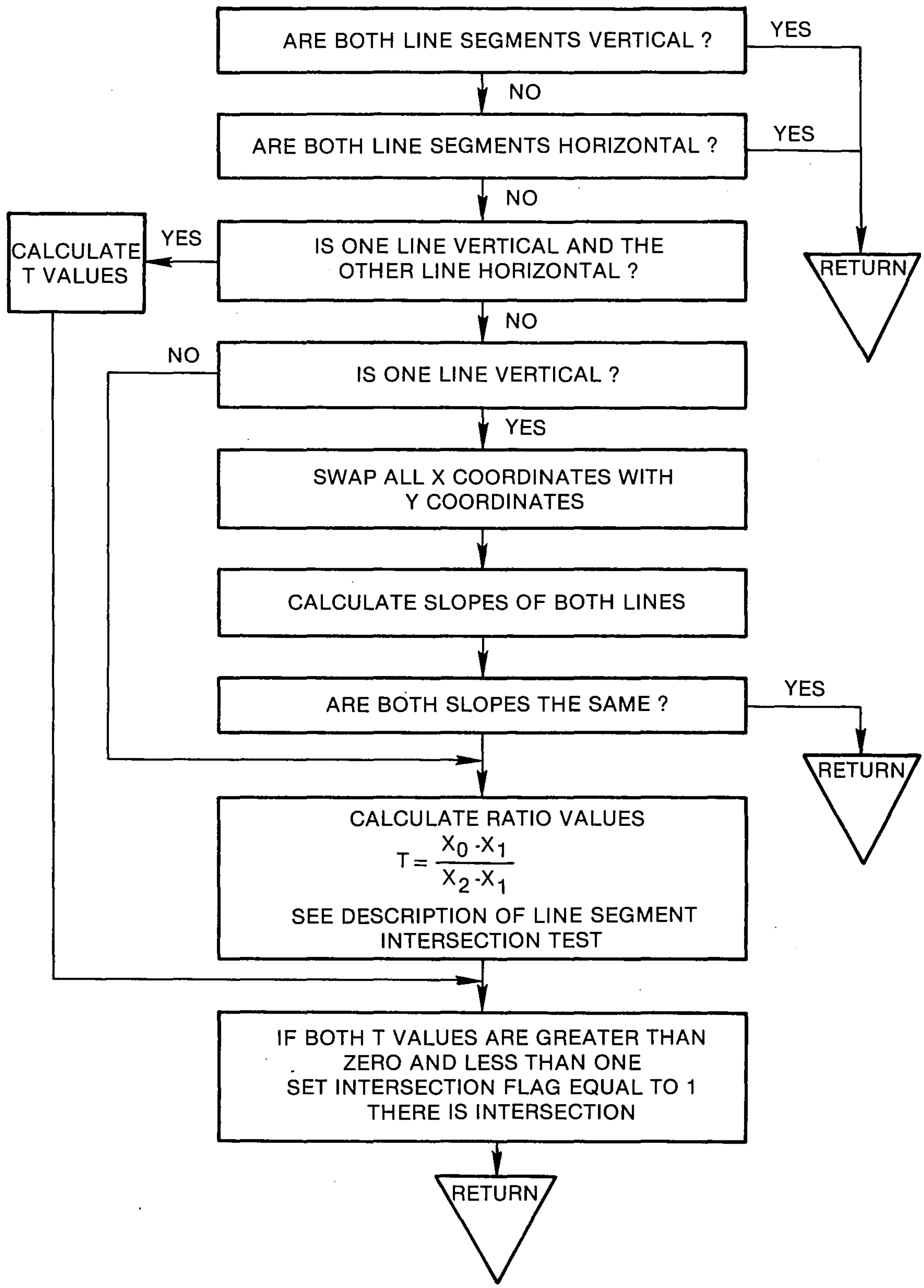


TRANSFORM POSITION VECTOR P3 TO THE VIEWPOINT REFERENCE FRAME
PROJECT POSITION VECTOR ON TO THE PROJECTION PLANE
CALL TPOINT TEST POINT AGAINST POLYGON REPRESENTED BY KK
IS POINT HIDDEN ? DOES IFLAG = 1 ?
NO
RETURN
UNDETERMINED
YES
IS POINT CLOSER TO VIEW-POINT THAN THE POLYGON ?
NO
INDICATE POINT NOT HIDDEN IFLAG = 2
INDICATE POINT HIDDEN IFLAG = 1
RETURN

5.9 INPUT

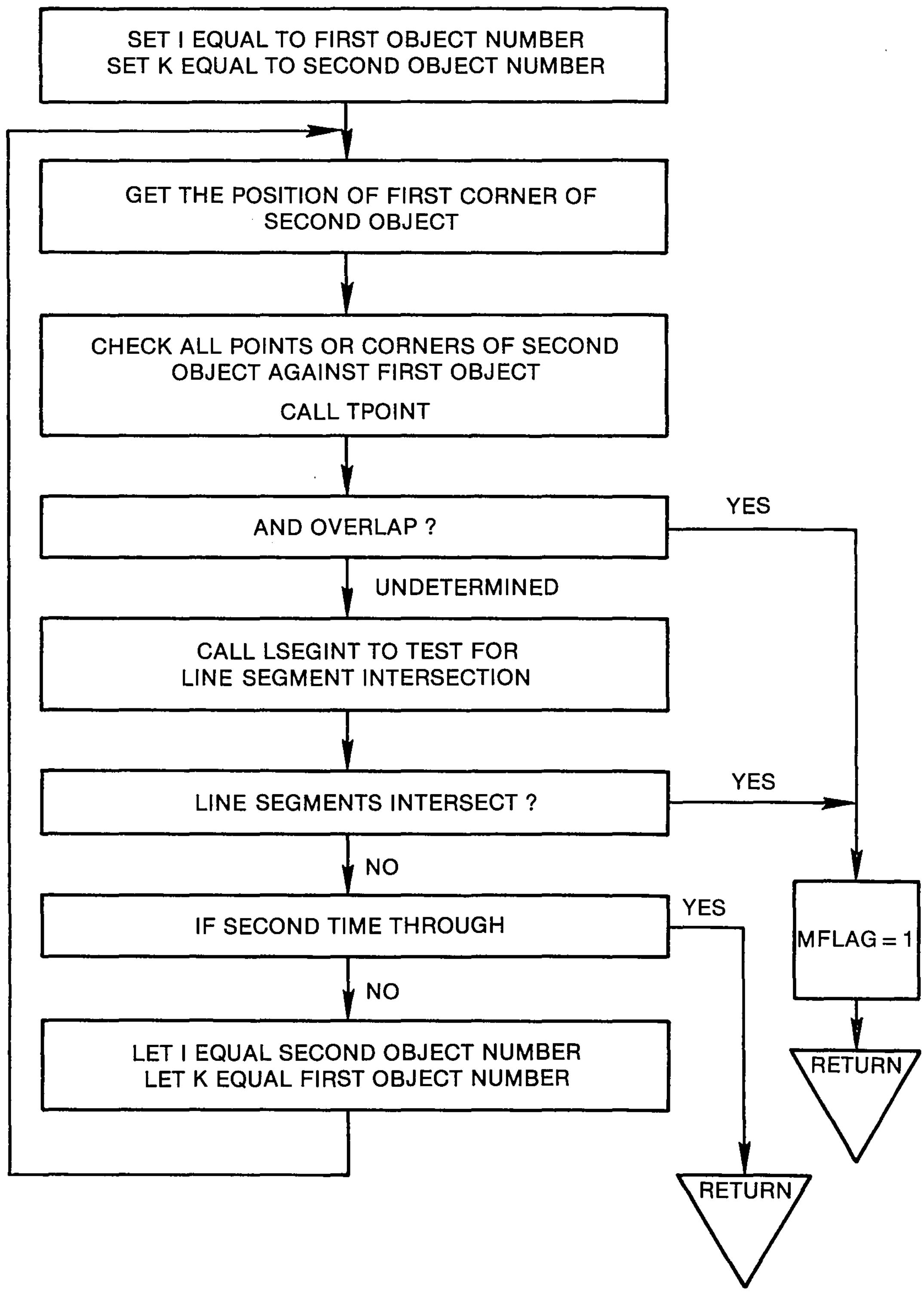

IF FIRST CALL
READ CARDS 6.0, 6.1 FROM LUN #5 AND POLYGON DATA FROM LUN #1
READ ELLIPSOID DATA FROM LUN #1 TILL ZTIME ≥ CTIME
IF NPL>0, READ CARDS 7.1 AND 7.2 FOR EACH PLANE
INITIALIZE ARRAYS A AND SEGLP
IF NSEG>0, READ CARD 8.0 FROM LUN #5
IF NPL>0, READ CARDS 9.0 AND 10.0 FROM LUN #5
READ CARDS 11.0, 12.0, 13.0, 14.0 AND 15.0 FROM LUN #5
IF ICODE = 0, CALL DRCYPR
IF ICODE = 2, CALL GENDCM
IF ICODE ≠ 0 AND ICODE ≠ 2, READ CARD 16.0 FROM LUN #5
STORE INITIAL VALUES OF VP + DVP ARRAYS
RETURN
IF NOT FIRST CALL
READ SEGMENT DATA FROM LUN #1 TILL ZTIME ≥ CTIME
WHEN CTIME REACHED, CALCULATE LINEAR POSITION OF CONTACT ELLIPSOID IN INERTIAL REFERENCE SYSTEM
IF THIS MATRIX MULTIPLY ATTEMPTED TWICE WITHOUT AN INTERVENING READ FOR LUN #1, SET IFLAG = 10
RETURN

5.10 LSEGINT

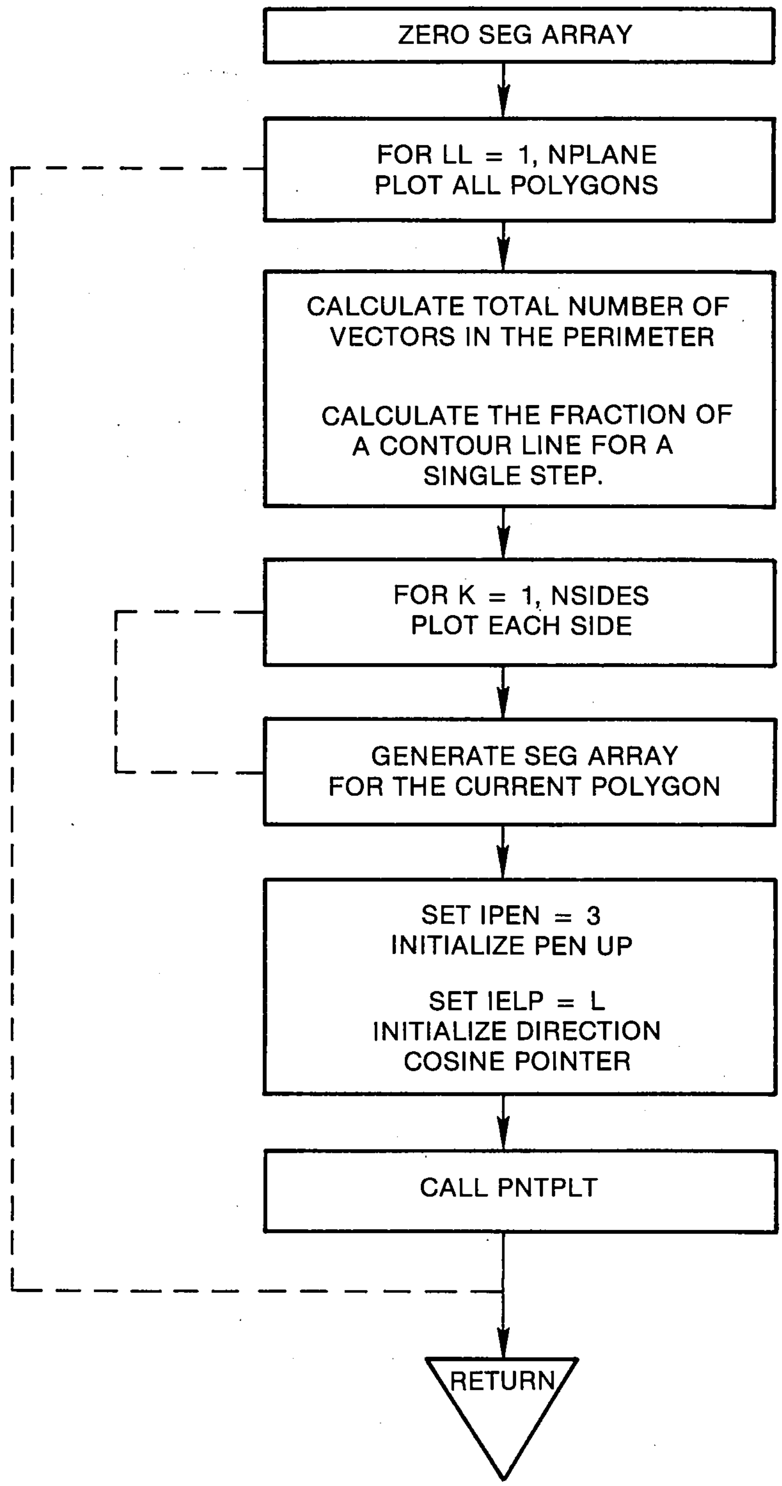


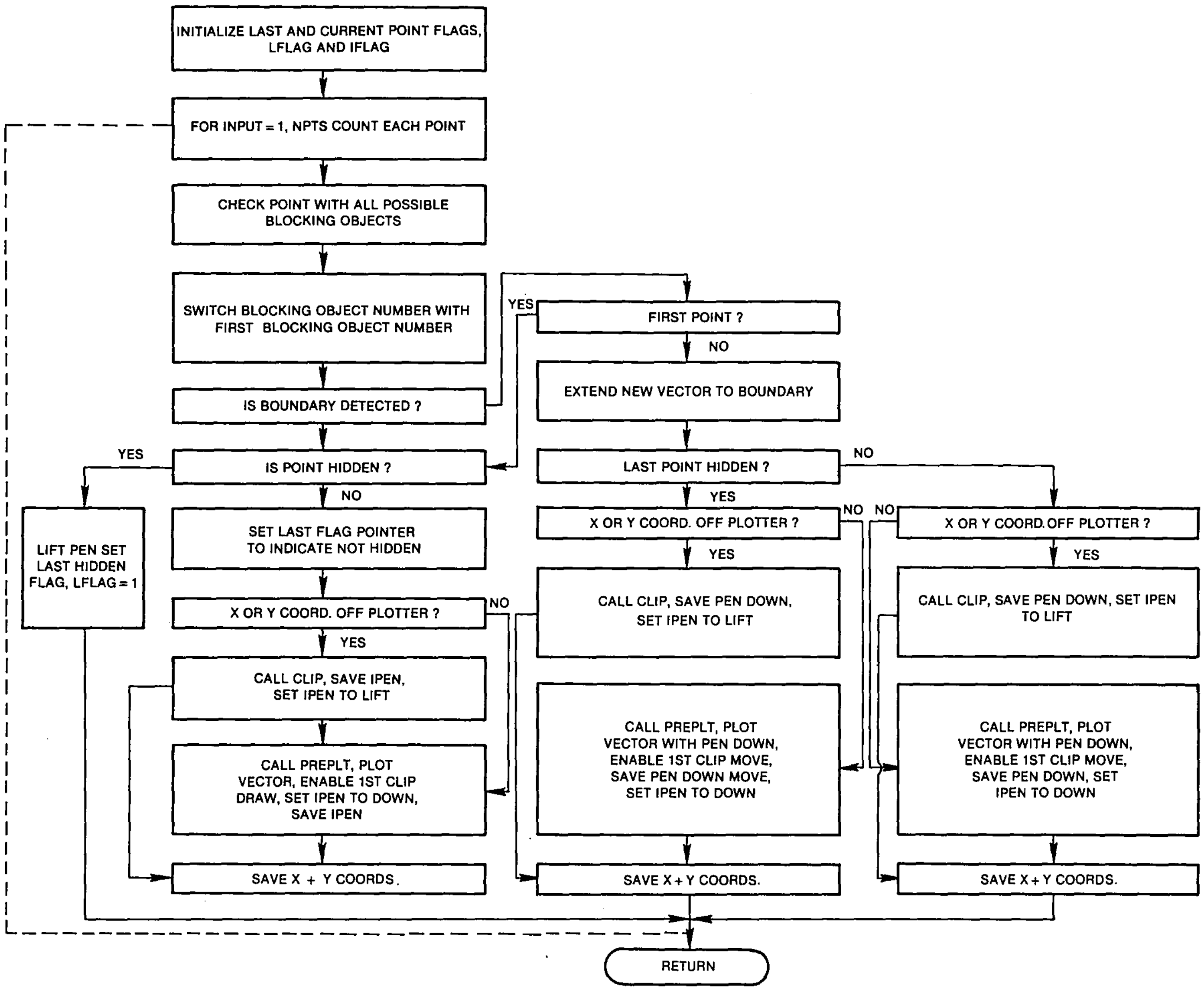
ARE BOTH LINE SEGMENTS VERTICAL ?
YES
NO
ARE BOTH LINE SEGMENTS HORIZONTAL ?
YES
NO
CALCULATE T VALUES
YES
IS ONE LINE VERTICAL AND THE OTHER LINE HORIZONTAL ?
RETURN
NO
NO
IS ONE LINE VERTICAL ?
YES
SWAP ALL X COORDINATES WITH Y COORDINATES
CALCULATE SLOPES OF BOTH LINES
ARE BOTH SLOPES THE SAME ?
YES
RETURN
CALCULATE RATIO VALUES
$T = \frac{X_0 - X_1}{X_2 - X_1}$
SEE DESCRIPTION OF LINE SEGMENT INTERSECTION TEST
IF BOTH T VALUES ARE GREATER THAN ZERO AND LESS THAN ONE SET INTERSECTION FLAG EQUAL TO 1 THERE IS INTERSECTION
RETURN

5.11 OVERLAP

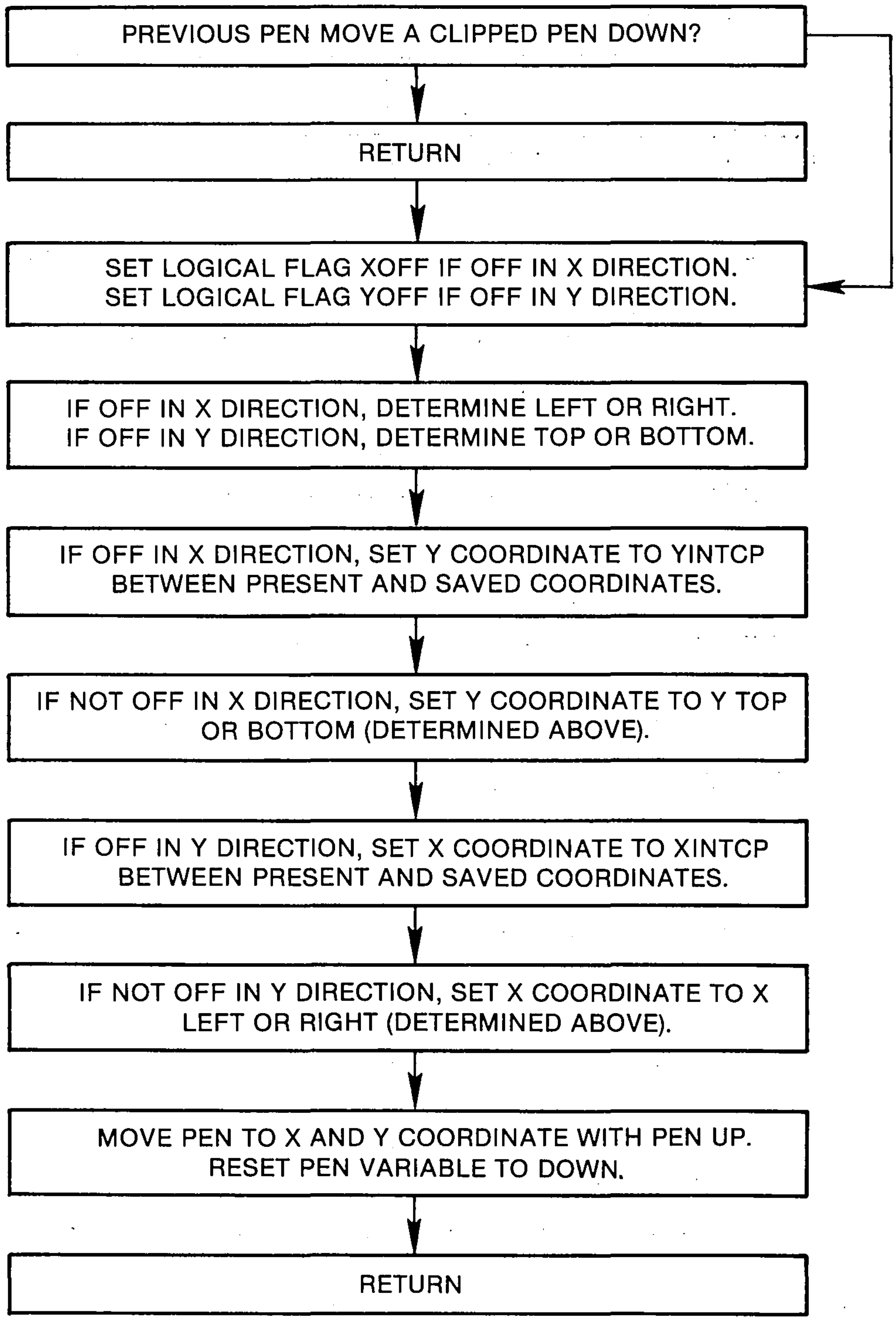

SET I EQUAL TO FIRST OBJECT NUMBER
SET K EQUAL TO SECOND OBJECT NUMBER
GET THE POSITION OF FIRST CORNER OF SECOND OBJECT
CHECK ALL POINTS OR CORNERS OF SECOND OBJECT AGAINST FIRST OBJECT
CALL TPOINT
AND OVERLAP ?
YES
UNDETERMINED
CALL LSEGINT TO TEST FOR LINE SEGMENT INTERSECTION
LINE SEGMENTS INTERSECT ?
YES
NO
IF SECOND TIME THROUGH
YES
NO
LET I EQUAL SECOND OBJECT NUMBER
LET K EQUAL FIRST OBJECT NUMBER
MFLAG = 1
RETURN
RETURN

5.12 PLPLN

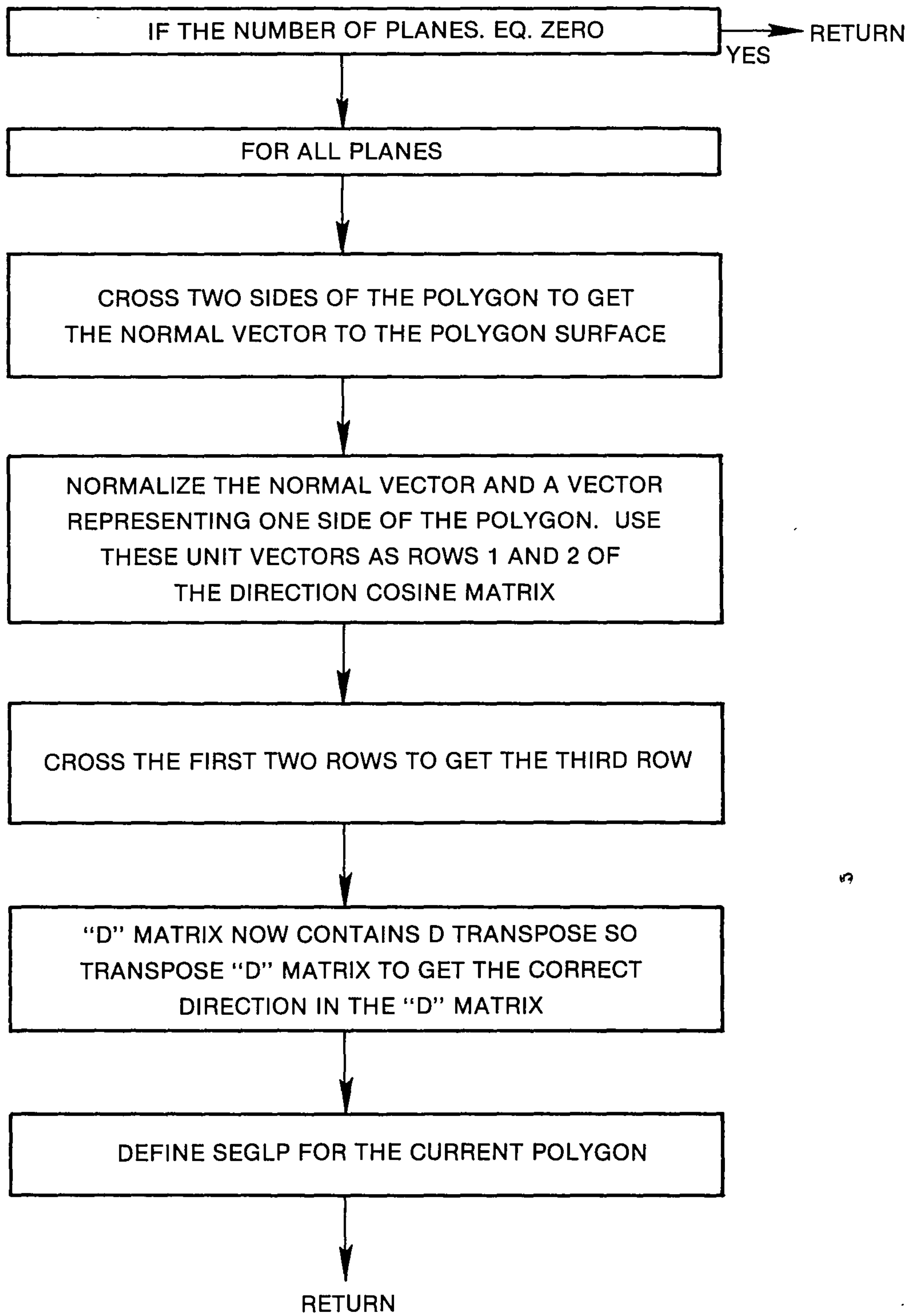

ZERO SEG ARRAY
FOR LL = 1, NPLANE
PLOT ALL POLYGONS
CALCULATE TOTAL NUMBER OF VECTORS IN THE PERIMETER
CALCULATE THE FRACTION OF A CONTOUR LINE FOR A SINGLE STEP.
FOR K = 1, NSIDES
PLOT EACH SIDE
GENERATE SEG ARRAY FOR THE CURRENT POLYGON
SET IPEN = 3
INITIALIZE PEN UP
SET IELP = L
INITIALIZE DIRECTION COSINE POINTER
CALL PNTPLT
RETURN


INITIALIZE LAST AND CURRENT POINT FLAGS, LFLAG AND IFLAG
FOR INPUT = 1, NPTS COUNT EACH POINT
CHECK POINT WITH ALL POSSIBLE BLOCKING OBJECTS
SWITCH BLOCKING OBJECT NUMBER WITH FIRST BLOCKING OBJECT NUMBER
IS BOUNDARY DETECTED ?
YES
FIRST POINT ?
NO
EXTEND NEW VECTOR TO BOUNDARY
IS POINT HIDDEN ?
YES
NO
LIFT PEN SET LAST HIDDEN FLAG, LFLAG = 1
SET LAST FLAG POINTER TO INDICATE NOT HIDDEN
X OR Y COORD. OFF PLOTTER ?
NO
YES
CALL CLIP, SAVE IPEN, SET IPEN TO LIFT
CALL PREPLT, PLOT VECTOR, ENABLE 1ST CLIP DRAW, SET IPEN TO DOWN, SAVE IPEN
SAVE X + Y COORDS.
LAST POINT HIDDEN ?
NO
YES
X OR Y COORD. OFF PLOTTER ?
NO
YES
CALL CLIP, SAVE PEN DOWN, SET IPEN TO LIFT
CALL PREPLT, PLOT VECTOR WITH PEN DOWN, ENABLE 1ST CLIP MOVE, SAVE PEN DOWN MOVE, SET IPEN TO DOWN
SAVE X + Y COORDS.
NO
X OR Y COORD. OFF PLOTTER ?
YES
CALL CLIP, SAVE PEN DOWN, SET IPEN TO LIFT
CALL PREPLT, PLOT VECTOR WITH PEN DOWN, ENABLE 1ST CLIP MOVE, SAVE PEN DOWN, SET IPEN TO DOWN
SAVE X + Y COORDS.
RETURN

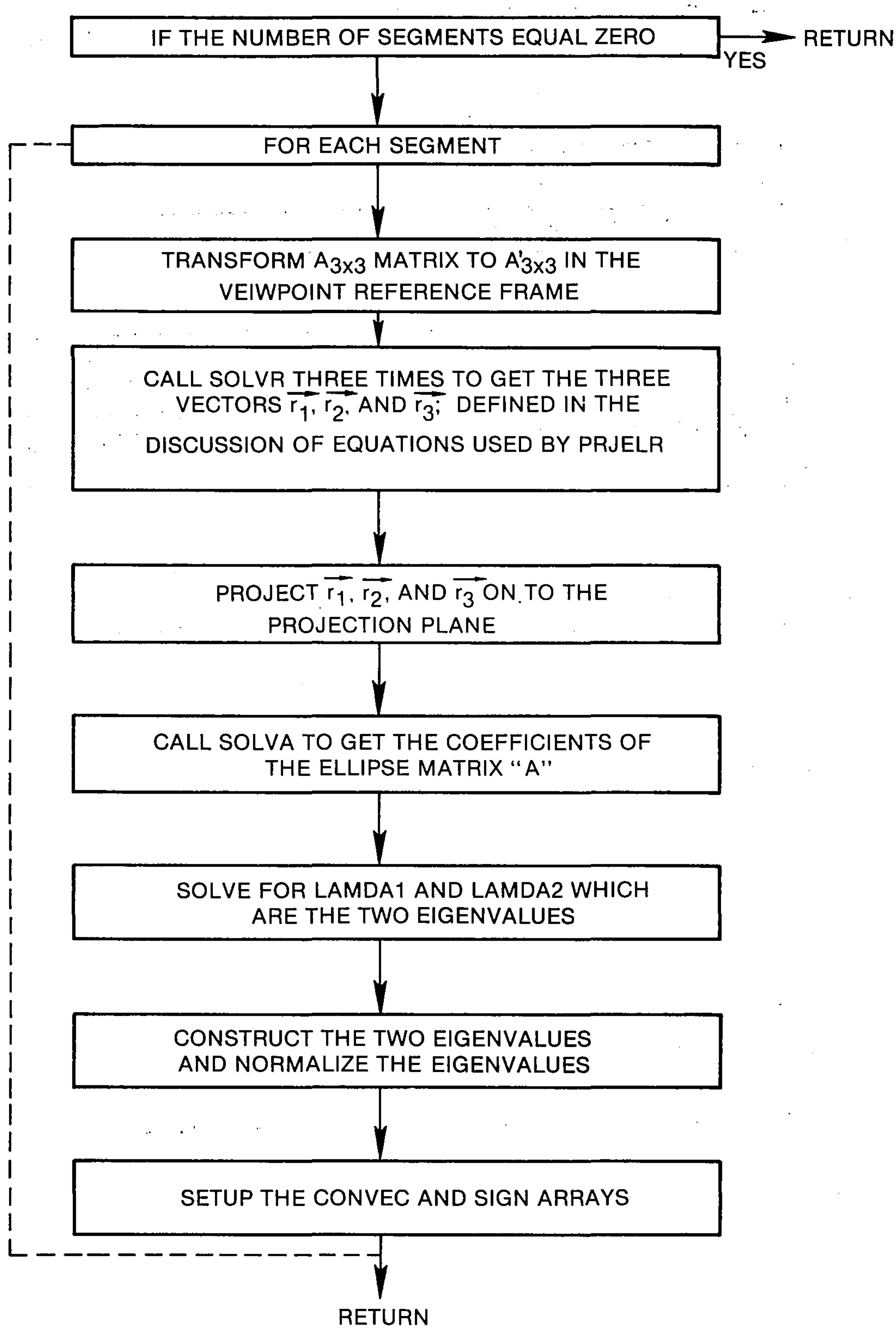
5.14 PREPLT


PREVIOUS PEN MOVE A CLIPPED PEN DOWN?
RETURN
SET LOGICAL FLAG XOFF IF OFF IN X DIRECTION.
SET LOGICAL FLAG YOFF IF OFF IN Y DIRECTION.
IF OFF IN X DIRECTION, DETERMINE LEFT OR RIGHT.
IF OFF IN Y DIRECTION, DETERMINE TOP OR BOTTOM.
IF OFF IN X DIRECTION, SET Y COORDINATE TO YINTCP BETWEEN PRESENT AND SAVED COORDINATES.
IF NOT OFF IN X DIRECTION, SET Y COORDINATE TO Y TOP OR BOTTOM (DETERMINED ABOVE).
IF OFF IN Y DIRECTION, SET X COORDINATE TO XINTCP BETWEEN PRESENT AND SAVED COORDINATES.
IF NOT OFF IN Y DIRECTION, SET X COORDINATE TO X LEFT OR RIGHT (DETERMINED ABOVE).
MOVE PEN TO X AND Y COORDINATE WITH PEN UP.
RESET PEN VARIABLE TO DOWN.
RETURN

5.15 POLYD

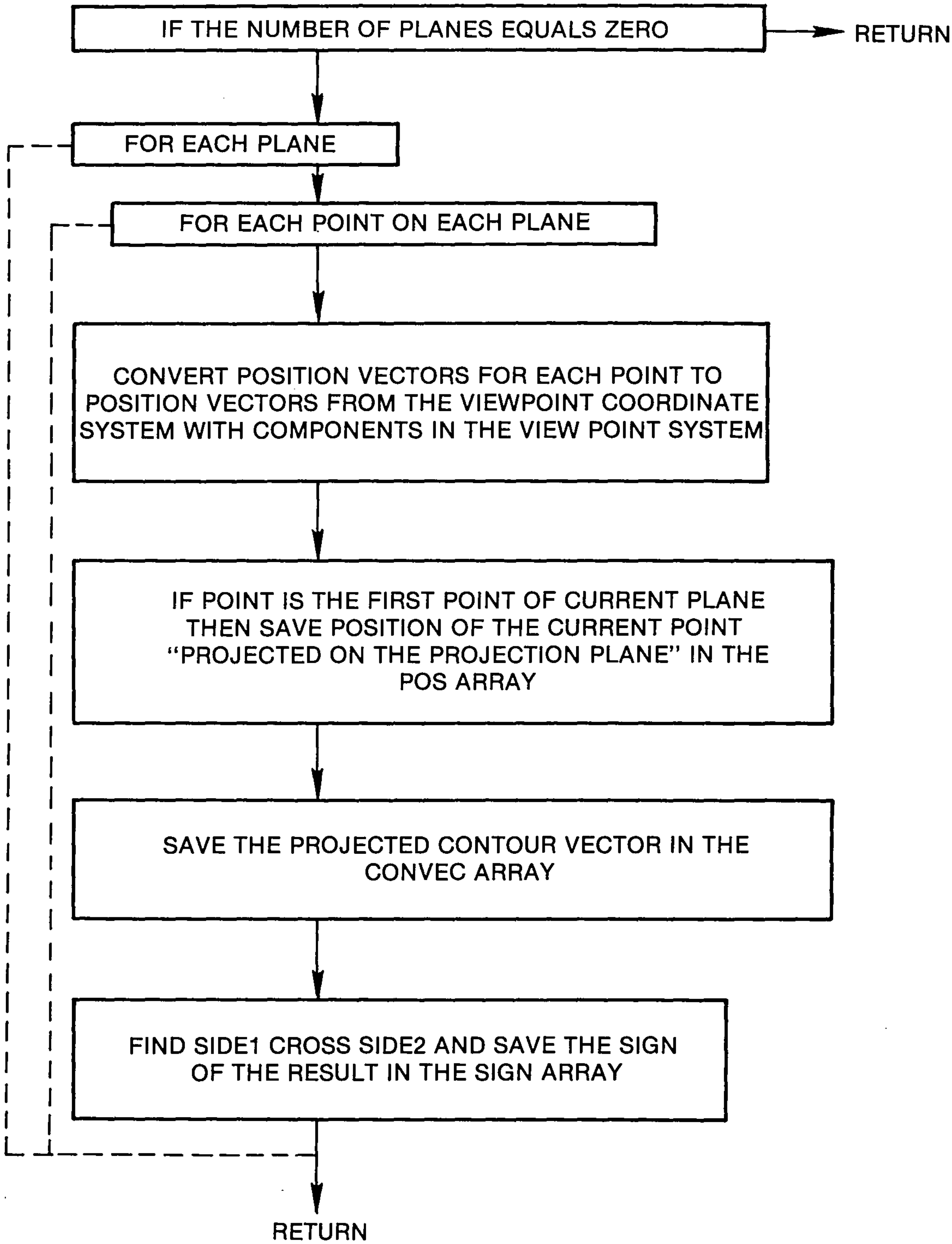

IF THE NUMBER OF PLANES. EQ. ZERO
YES
RETURN
FOR ALL PLANES
CROSS TWO SIDES OF THE POLYGON TO GET THE NORMAL VECTOR TO THE POLYGON SURFACE
NORMALIZE THE NORMAL VECTOR AND A VECTOR REPRESENTING ONE SIDE OF THE POLYGON. USE THESE UNIT VECTORS AS ROWS 1 AND 2 OF THE DIRECTION COSINE MATRIX
CROSS THE FIRST TWO ROWS TO GET THE THIRD ROW
"D" MATRIX NOW CONTAINS D TRANSPOSE SO TRANSPOSE "D" MATRIX TO GET THE CORRECT DIRECTION IN THE "D" MATRIX
DEFINE SEGLP FOR THE CURRENT POLYGON
RETURN

5.16 PRJELR

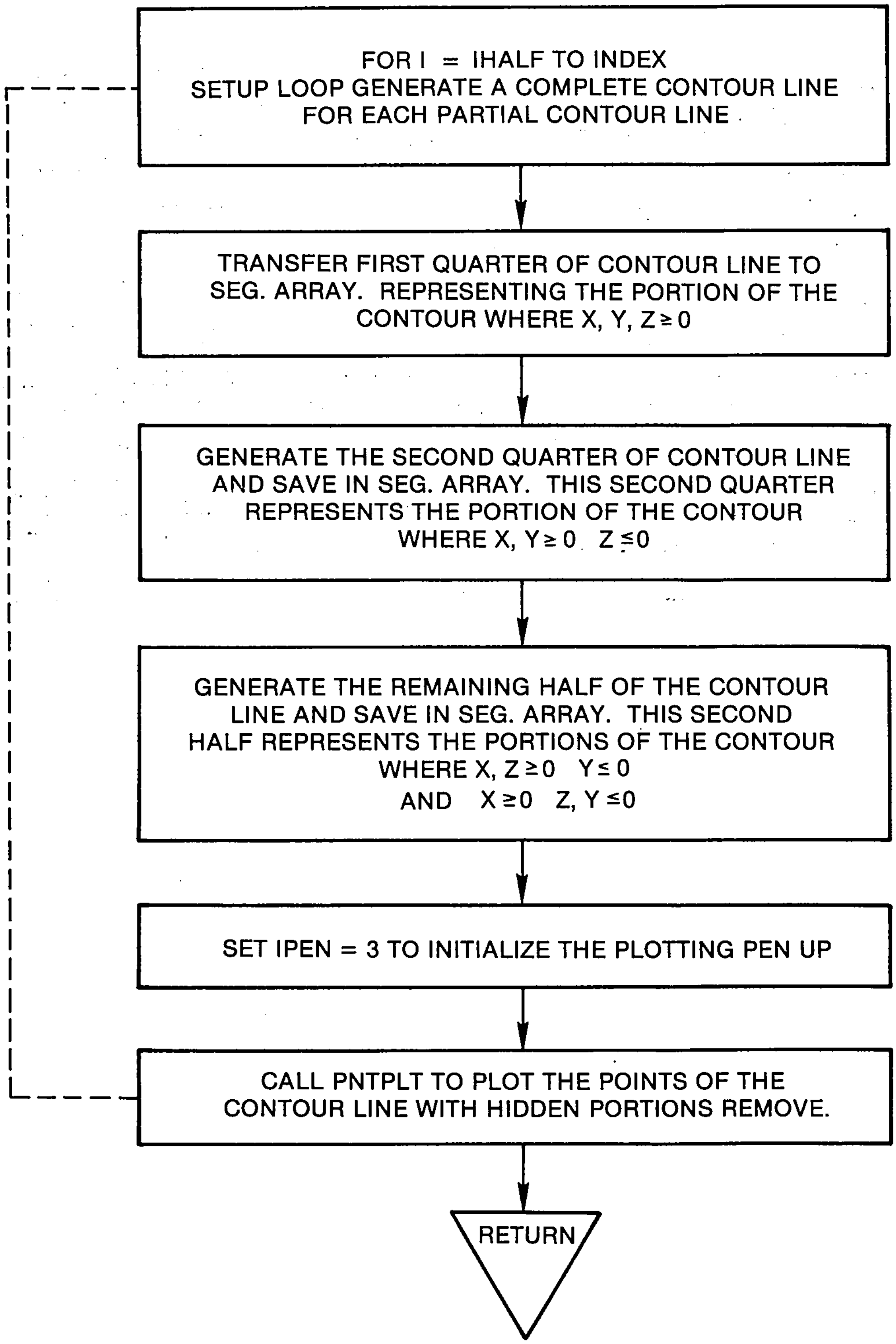


IF THE NUMBER OF SEGMENTS EQUAL ZERO
YES
RETURN
FOR EACH SEGMENT
TRANSFORM A_{3x3} MATRIX TO A'_{3x3} IN THE VEIWPOINT REFERENCE FRAME
CALL SOLVR THREE TIMES TO GET THE THREE VECTORS $\vec{r_1}$, $\vec{r_2}$, AND $\vec{r_3}$; DEFINED IN THE DISCUSSION OF EQUATIONS USED BY PRJELR
PROJECT $\vec{r_1}$, $\vec{r_2}$, AND $\vec{r_3}$ ON TO THE PROJECTION PLANE
CALL SOLVA TO GET THE COEFFICIENTS OF THE ELLIPSE MATRIX "A"
SOLVE FOR LAMDA1 AND LAMDA2 WHICH ARE THE TWO EIGENVALUES
CONSTRUCT THE TWO EIGENVALUES AND NORMALIZE THE EIGENVALUES
SETUP THE CONVEC AND SIGN ARRAYS
RETURN

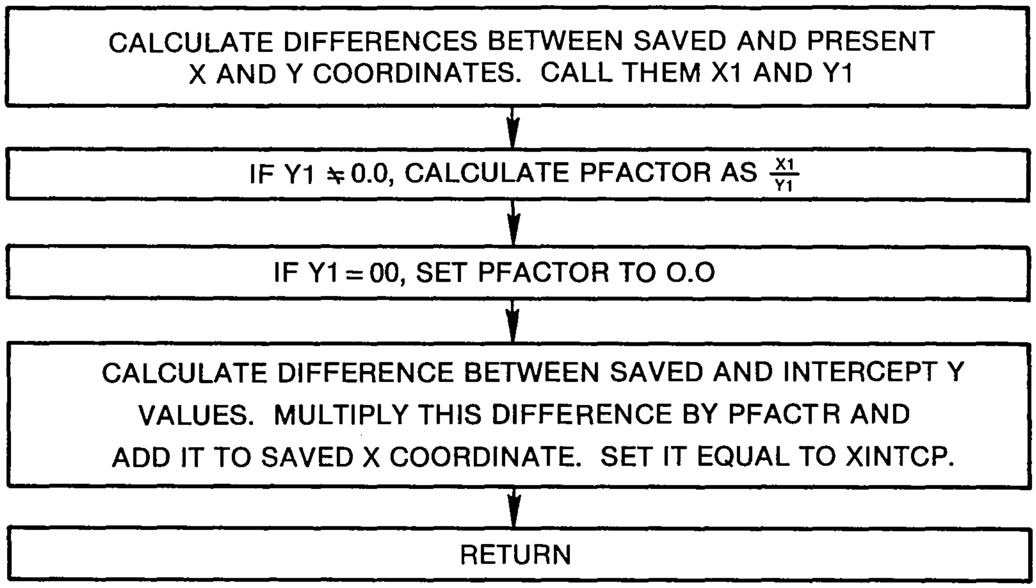
5.17 PRJPLY


IF THE NUMBER OF PLANES EQUALS ZERO
RETURN
FOR EACH PLANE
FOR EACH POINT ON EACH PLANE
CONVERT POSITION VECTORS FOR EACH POINT TO POSITION VECTORS FROM THE VIEWPOINT COORDINATE SYSTEM WITH COMPONENTS IN THE VIEW POINT SYSTEM
IF POINT IS THE FIRST POINT OF CURRENT PLANE THEN SAVE POSITION OF THE CURRENT POINT "PROJECTED ON THE PROJECTION PLANE" IN THE POS ARRAY
SAVE THE PROJECTED CONTOUR VECTOR IN THE CONVEC ARRAY
FIND SIDE1 CROSS SIDE2 AND SAVE THE SIGN OF THE RESULT IN THE SIGN ARRAY
RETURN

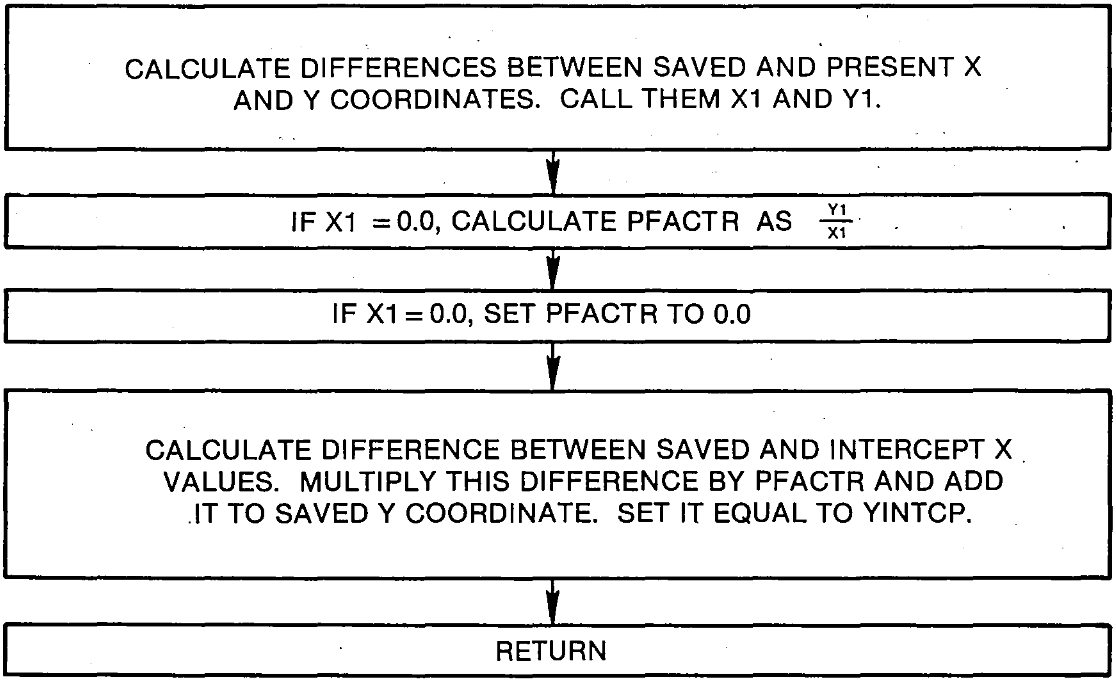
5.18 PSE


FOR I = IHALF TO INDEX
SETUP LOOP GENERATE A COMPLETE CONTOUR LINE
FOR EACH PARTIAL CONTOUR LINE
TRANSFER FIRST QUARTER OF CONTOUR LINE TO
SEG. ARRAY. REPRESENTING THE PORTION OF THE
CONTOUR WHERE X, Y, Z ≥ 0
GENERATE THE SECOND QUARTER OF CONTOUR LINE
AND SAVE IN SEG. ARRAY. THIS SECOND QUARTER
REPRESENTS THE PORTION OF THE CONTOUR
WHERE X, Y ≥ 0 Z ≤ 0
GENERATE THE REMAINING HALF OF THE CONTOUR
LINE AND SAVE IN SEG. ARRAY. THIS SECOND
HALF REPRESENTS THE PORTIONS OF THE CONTOUR
WHERE X, Z ≥ 0 Y ≤ 0
AND X ≥ 0 Z, Y ≤ 0
SET IPEN = 3 TO INITIALIZE THE PLOTTING PEN UP
CALL PNTPLT TO PLOT THE POINTS OF THE
CONTOUR LINE WITH HIDDEN PORTIONS REMOVE.
RETURN

5.19 XINTCP


CALCULATE DIFFERENCES BETWEEN SAVED AND PRESENT X AND Y COORDINATES. CALL THEM X1 AND Y1
IF Y1 ≠ 0.0, CALCULATE PFACTOR AS X1/Y1
IF Y1 = 00, SET PFACTOR TO O.O
CALCULATE DIFFERENCE BETWEEN SAVED AND INTERCEPT Y VALUES. MULTIPLY THIS DIFFERENCE BY PFACTR AND ADD IT TO SAVED X COORDINATE. SET IT EQUAL TO XINTCP.
RETURN

5.20 YINTCP



CALCULATE DIFFERENCES BETWEEN SAVED AND PRESENT X AND Y COORDINATES. CALL THEM X1 AND Y1.
IF X1 =0.0, CALCULATE PFACTR AS $\frac{Y1}{X1}$
IF X1 = 0.0, SET PFACTR TO 0.0
CALCULATE DIFFERENCE BETWEEN SAVED AND INTERCEPT X VALUES. MULTIPLY THIS DIFFERENCE BY PFACTR AND ADD IT TO SAVED Y COORDINATE. SET IT EQUAL TO YINTCP.
RETURN

6.0 VIEW PROGRAM VARIABLE GLOSSARIES BY PROGRAM MODULE

This section contains lists of selected variables contained in VIEW. The lists contain all pertinent variables and exclude temporary variables such as DO loop indices and intermediate, calculated variables. This section is organized in the following manner; common block variables, main program variables, and variables for each subroutine, subroutines in alphabetical order. The variables for all common blocks have been combined and listed in alphabetical order. For which common block the variable belongs, see Section 7, cross-reference Chart 7.3. The variable descriptions are alphabetized within each subroutine. The variable description consists of the variable name; its dimension, if any; FORTRAN data type; and a short description.

6.1 COMMON BLOCK VARIABLE DEFINITIONS

A(3,3,30)	Real Array. Contains the ellipsoid matrices used in the following equation: $\vec{r} \cdot [\underline{A}]\, \vec{r} = 1$
BDRS	Integer scalar. Constant that represents a particular graphics output system. Used with DEVFLG. Initialized in Main.
CONVEC(2,5,90)	Real array. Contains vectors on the projection plane that represent the projected polygons and rectangles that circumscribe projected ellipsoids on the projection plane. Initialized in subroutine PRJPLY.
D(3,3,90)	Real array. Contains directional cosine matrices for segments (third subscript value 1-30) and polygons (31-90). Data for the segments is read from the ATB input file in subroutine INPUT and data for polygons is initialized in subroutine POLYD. Transforms from the inertial coordinate system to the local geometric coordinate sytem of the segment.
DEVFLG	Integer scalar. Represents VIEW program output device number. Read from input control file in Main.

DVP(3,3)	Real array. Contains the direction cosine matrix for the viewpoint coordinate system that transforms from the inertial reference system to the viewpoint coordinate system. Initialized in subroutine INPUT and modified in Main.
DVPØ(3,3)	Real array. Contains the original direction cosine matrix that transforms from the segment reference system to which the viewpoint is attached, to the viewpoint coordinate system. Initialized in subroutine INPUT.
ICOLOR (91)	Integer array. Contains color numbers for each ellipsoid (subscript values 1-30), polygon (31-90) and title (91). Read from input control file in subroutine INPUT.
IDEBUG(80)	Integer array. Contains debugging printer control flags. Read from input control file in Main.
IE(90,90)	Integer array. Contains object numbers that overlap with a given object. Second subscript values 1-30 represent ellipsoids, 31-90 represents polygons. Initialized in subroutine BUILDIE.
IELP	Integer scalar. Represents ellipsoid or polygon that is currently being plotted. Initialized and modified in Main and subroutine PLPLN.
IFLAG	Integer scalar. Flag indicating state of subroutine INPUT. Initialized and modified in Main and INPUT.
INT	Integer scalar. Number of iterations used in subroutine EXTEND for finding the edge of a hidden line segment. INT = 4 is recommended. Read from input control file in subroutine INPUT.
IREMOV(30)	Integer array. Contains numbers that represent polygons to be removed from View program output. Read from input control file in subroutine INPUT.

IVP	Integer scalar. Segment number representing ellipsoid, vehicle, or ground to which the viewpoint and focal point are attached.
MPL(3,5,60)	Integer array. Contains segment and ellipsoid identification numbers for each plane segment contact as defined in the ATB program. MPL is read in from both the ATB input file and the input control file.
NIE(90)	Integer array. Contains the number of overlapping ellipsoids and polygons for a given ellipsoid or polygon. Subscript values 1-30 represent ellipsoids, 31-90 represent polygons. Initialized in subroutine BUILDIE.
NISG	Integer scalar. Defines the reference frame for unattached planes. Read from input control file in subroutine INPUT.
NP	Integer scalar. Represents the number of ATB input polygons. Read from ATB input file in subroutine INPUT.
NPLANE	Integer scalar. Represents total number of polygons (ATB input plus VIEW input planes). Initialized in subroutine INPUT.
NPPP(90)	Integer array. Contains number of points per polygon. This equals number of sides plus one. Initialized in subroutine PRJELR for rectangles about ellipsoids and read from input control file in subroutine INPUT for polygons.
NPREM	Integer scalar. Represents the number of polygons to be removed from plot output. Read from input control file in subroutine INPUT.
NSEG	Integer scalar. Total number of segments read from ATB input file in subroutine INPUT.
NSTEPS(90)	Integer array. Contains number of steps (grid points along an axis) for each ellipsoid and polygon. In the case of the polygons, NSTEPS represents the number of vectors on any one side of the polygon. Read from input control file in subroutine INPUT.

OFLINE	Integer scalar. Constant that represents off-line condition. Used with DEVFLG. Initialized in Main.
OFSETX	Real scalar. Offset variable to move plot in X direction on the plotting area. Read from input control file in subroutine INPUT.
OFSETY	Real scalar. Offset variable to move plot in Y direction on the plotting area. Read from input control file in subroutine INPUT.
ONLINE	Integer scalar. Constant that represents online condition. Used with DEVFLG. Initialized in Main.
P(3,4,60)	Real array. Contains vectors that represent the polygon sides in the inertial reference frame. Initialized in subroutine CONVREC.
PL(17,30)	Real array. Contains polygon parameters for ATB input polygons. Read from ATB input file.
POS(2,90)	Real array. Contains position vectors for the polygons represented by the CONVEC array. These position vectors originate from the viewpoint origin on the projection plane to a corner of the polygon stored in the CONVEC array. POS vectors are in the viewpoint coordinate system. POS is initialized in subroutine PRJPLY.
PO(3,4,60)	Real array. Contains position vectors of the polygon vertices in reference to the MPL coordinate system. ATB input polygons are initialized in subroutine CONVREC, and VIEW input polygons are read from input control file in subroutine INPUT.
RA(3)	Real array. Contains three different kinds of data depending on the value of ICODE. These data are: ICODE = 0 — Roll, pitch, and yaw angles are supplied in RA array. Initialized in subroutine DRYCYPR.

	ICODE = 1	Direction cosine matrix supplied as input. The RA array is the first row of the direction cosine matrix. The second and third rows are on the next record. This data is read from the input control file in subroutine INPUT.
	ICODE = 2	Point at which viewpoint Z-axis to aim is supplied in the RA array. Read from input control file in subroutine INPUT.

SEGLP(3,90) — Real array. Contains position vectors for ellipsoids and polygons. These vectors go from the inertial reference origin to the center of the contact ellipsoid. In the case of a polygon, this vector points to one corner of the polygon. SEGLP is in the inertial reference frame. It is read from the ATB input file in subroutine INPUT.

SFACTR — Real scalar. Reprsents scale factor for plot. Is the Z coordinate of the projection plane in the viewpoint reference frame. Read from input control file in subroutine INPUT.

SIGN(90) — Real array. Contains cross product of side 1 and side 2 of a polygon. Initialized in subroutine PRJPLY.

TERM — Integer scalar. Constant that represents terminal as output device. Used with DEVFLG. Initialized in Main.

TIME — Real scalar. Time (seconds) of the current segment data set. Read from ATB input file in subroutine INPUT.

VP(3) — Real array. Position vector for the viewpoint coordinate system. This vector is in the inertial reference frame. Read from input control file in subroutine INPUT and modified in Main.

VPO(3)	Real array. Contains the coordinates of the viewpoint in the coordinate system to which the viewpoint is attached. These are the original values read into array VP. Initialized in subroutine INPUT.
VP2(3)	Real array. Contains the viewpoint position vector in the viewpoint coordinate system.
ZTIME	Double precision scalar. Represents time of current data set (TIME) rounded to the nearest microsecond. This was done in order to get equal comparisons between input data time and program calculated time. Modified in subroutine INPUT.

6.2 MAIN PROGRAM

CTIME	Double precision scalar. Represents current time of the program rounded to the nearest microsecond.
DTIME	Double precision scalar. Represents delta time or what time step value CTIME will be incremented. Read from record 4.0 of input control file.
ETIME	Double precision scalar. Represents end time of the program. Read from record 4.0 of input control file.
ID(10)	Real array. Contains 40 character title strip. Read from record 3.0 of input control file.
LUPLOT	Integer scalar. Represents logical unit number for the output plotting file.
NF	Integer scalar. Frame counter that is written every frame to output list file.
STIME	Double precision scalar. Represents start time of the program or when to begin plotting. Read from card 4.0 of input control file.
XTIME	Real scalar. Temporary variable that is the current program time expressed in milliseconds for plotting.

WORK(10000)	Real array. Temporary array space allocated for subroutines ELIPSN, PSE, and PLPLN.

6.3 SUBROUTINE BUILDIE

IOBJ	Integer scalar. Equal to present object number plus one.
KPLANE	Integer scalar. Equal to total number of polygons plus 30.
MFLAG	Integer scalar. Flag passed back from subroutine OVERLAP telling whether or not there is a overlap condition.
MPLANE	Integer scalar. Equal to total number of polygons plus 29.

6.4 SUBROUTINE CLIP

IPEN	Integer scalar. Calcomp pen value passed through argument list from subroutine PNTPLT.
IPLOT	Integer scalar. Flag that tells CLIP if last pen move was clipped (IPLOT=0), or plotted (IPLOT=1).
LCALL	Integer scalar. Flag that tells CLIP if last call to CLIP was a pen up to first point in line. LCALL=0 means it was, LCALL=1 means that it was not.
X	Real scalar. X coordinate of point to be plotted passed through argument list from PNTPLT.
XLIMIT	Real scalar. If X coordinate off plotting region, this variable contains value of which boundary (left or right) it is off.
XLSAV	Real scalar. X coordinate of first point in line that was clipped in last call to CLIP.
XLTEMP	Real scalar. Value of X coordinate to move to if previous call to CLIP was a pen up to first point in segment.

XMAX	Real scalar. Constant that represents right side X limit of plotting region. Read from record 16.0 in input control file.
XMIN	Real Scalar. Constant that represents left side X limit of plotting region. Read from record 16.0 in input control file.
XOFF	Logical scalar. Is .FALSE. if X coordinate within valid plotting region, .TRUE. if outside plotting region.
XSAV	Real scalar. X coordinate of last plot move. Passed through argument list from PNTPLT.
XTEMP	Real scalar. X coordinate of end point of line segment to be clipped.
Y	Real scalar. Y coordinate of point to be plotted. Passed through argument list from subroutine PNTPLT.
YLIMIT	Real scalar. If Y coordinate of plotting region, this variable contains value of which boundary (top or bottom) it is off.
YLSAV	Real scalar. Y coordinate of first point in line that was clipped in last call to CLIP.
YLTEMP	Real scalar. Value of YMIN or YMAX, depending on if first point in line clipped was off top or bottom of plotter.
YMAX	Real scalar. Constant that represents upper (top) limit of the plotting region. Defined in subroutine PNTPLT.
YMIN	Real scalar. Constant that represents lower (bottom) limit of the plotting region. Defined in subroutine PNTPLT.
YOFF	Logical scalar. Is .FALSE. if Y coordinate within valid plotting region, .TRUE. if outside plotting region.
YSAV	Real scalar. Represents Y coordinate of previous point plotted. Passed through parameter list from subroutine PNTPLT.

YTEMP	Real scalar. Value of YMIN or YMAX, depending on if point clipped is off top or bottom of the plotter.

6.5 SUBROUTINE CONVREC

DDD	Real scalar. Intermediate variable used in calculating data for PØ array.
DX(3)	Real array. Contains results from determinant function DET for solving the three equations CONVREC uses.
ISG	Integer scalar. Segment number that defines the coordinate system for plotting that particular plane.
NPPPP	Integer scalar. Number of points in the polygon. Loaded from array NPPP.
R(3)	Real array. Contains matrix multiple of portions of arrays DVP and PØ.

6.6 SUBROUTINE CROSS

A(3)	Real array. Vector in $\vec{C}=\vec{A}*\vec{B}$.
B(3)	Real array. Vector in $\vec{C}=\vec{A}*\vec{B}$.
C(3)	Real array. Result vector in $\vec{C}=\vec{A}*\vec{B}$.

6.7 FUNCTION DET

A11	--	Real scalars. Values Representing 3 × 3, square array.
A12	--	
A13	--	
A21	--	
A22	--	
A23	--	
A31	--	
A32	--	
A33	--	
DET	--	Real scalar. Determinant of input array.

6.8 SUBROUTINE DOT and DOTT

A(L,3)	Real array. Array A in matrix multiply C=AB.
B(1,3)	Real array. Array B in matrix multiply C=AB.
C(N,M)	Real array. Output of DOT(T), in array C matrix multiply C=AB.
L	Integer scalar. First subscript value for arrays A and B.
M	Integer scalar. Second subscript value for array C.
N	Integer scalar. First subscript value for array C.

6.9 SUBROUTINE DRCYPR

A(3)	Real array. Contains rotation angles (degrees).
D(3,3)	Real array. Output of DRCYPR, contains direction cosine matrix.
I1	Integer scalar. Axis of rotation for first angle (X=1, Y=2, Z=3).
I2	Integer scalar. Axis of rotation for second angle (X=1, Y=2, Z=3).
I3	Integer scalar. Axis of rotation for third angle (X=1, Y=2, Z=3).
M	Integer scalar. Constant of 6, size of T array.
N	Integer scalar. Constant of 3, size of A array.
P	Real scalar. Pitch angle in radians.
R	Real scalar. Roll angle in radians.
RADIAN	Real scalar. Constant for degrees to radians conversion.

T(6,3)	Real array. Temporary buffer space used for matrix multiple.
Y	Real scalar. Yaw angle in radians.

6.10 SUBROUTINE ELIPSN

DELTAX	Real scalar. Semiaxes length in X direction divided by number of steps.
DELTAY	Real scalar. Semiaxes length in Y direction divided by number of steps.
IN(INDEX)	Integer array. Number of points in each contour array.
INDEX	Integer scalar. Number of steps plus one.
N	Integer scalar. Number of points in this half contour that is about the X axis.
SIMP1	Real scalar. $\frac{1}{a^2}$ where "a" is the semiaxes length in the X direction.
SIMP2	Real scalar. $\frac{1}{b^2}$ where "b" is the semiaxes length in the Y direction.
SIMP3	Real scalar. $\frac{1}{c^2}$ where "c" is the semiaxes length in the Z direction.
TEMP	Real scalar. The value of $1\text{-}X^2$*SIMP1.
TEST	Real scalar. The value of TEMP-Y^2*SIMP2.
X	Real scalar. Current X coordinate, starts at zero.
X1(3, INDEX, INDEX2)	Real array. Semiellipsoid contour array.

Y	Real scalar. Current Y coordinate, starts at zero.
Z	Real scalar. Current Z coordinate, starts at length "C."

6.11 SUBROUTINE EXTEND

I	Integer scalar. Points to unhidden point location in P array.
IFLAG	Integer scalar. Flag passed back from subroutines HIDE and HYDE telling whether or not the point is hidden.
J	Integer scalar. Points to hidden point location in P array.
KK	Integer scalar. Overlapping object number that overlaps with present object, from array IE.
N	Integer scalar. Denotes midpoint coordinates, either I or J, depending on hidden or not.
NUM	Integer scalar. Represents the number of overlapping objects with a particular object.
P(3,2)	Real array. Contains X, Y, and Z coordinates of the end points of the line.
P3(3)	Real array. Temporary buffer containing coordinates of the midpoint of the line.

6.12 SUBROUTINE GENDCM

CAMERA(3)	Real array. Position vector for the viewpoint in the reference sysem of the segment to which the viewpoint is attached.
D(3,3)	Real array. Direction cosine matrix for viewpoint. This direction cosine matrix transforms from the inertial reference frame to the local reference frame.

FOCUS(3)	Real array. Position vector for point which viewpoint Z axis is aimed.
SUM	Real scalar. Running summation of values in Z array.
XNDRM	Real scalar. Normalization factor.
Z(3)	Real array. Contains differences between matching values in CAMERA and FOCUS arrays.

6.13 SUBROUTINE HIDE

IFLAG	Integer scalar. Flag passed back to caller indicating hidden (IFLAG=1), or not hidden (IFLAG=2).
KK	Integer scalar. Polygon number to check blocking on.
NPRIME(3)	Real array. Contains matrix multiply of arrays DVP and PPRIME.
P3(3)	Real array. Contains position vector of point.
P4(3)	Real array. Contains P3 transformed to viewpoint reference system.
P5(3)	Real array. Contains matrix multiple of NPRIME and PPRIME arrays.
P6(3)	Real array. Contains matrix multiple of NPRIME and P4 arrays.
P7(2)	Real array. Contains position vector P4 projected on the projection plane.
PP(3)	Real array. Contains polygon sides P minus the viewpoint array VP.

6.14 SUBROUTINE HYDE

IFLAG	Integer Scalar. Flag passed back to caller to indicate whether or not the point was hidden. IFLAG=1 means hidden, IFLAG=2, not hidden.

N	Integer scalar. Possible hiding ellipsoid number.
R(3)	Real array. Vector to plotting point in local reference frame of ellipsoid or polygon of which it is a part.

6.15 SUBROUTINE INPUT

BD(24, 40)	Real array. Contains contact ellipsoid parameters for each contact ellipsoid. The rotation of the contact ellipsoid relative to the segment coordinate system is already incorporated in the values of this array. Read from ATB input file.
CTIME	Double precision scalar. Current program time (seconds) calculated in Main program.
DD (3)	Real array. Contains the offset vector (in the inertial coordinate system) of the contact ellipsoid from the segment c.g.
ICODE	Integer scalar. Input flag controlling the generation of the direction cosine matrix. Read from input control file.
ISW1	Integer scalar. Flag controlling whether plots are to be made when the time of the VIEW run is less than the time interval of the simulation data. If this is true, only the available simulation data will be plotted.
NFAST	Integer scalar. Represents number of segments to be removed used in ATB simulation. Read from input control file.
NSIDES	Integer scalar. Number of sides in a polygon. Loaded from NPPP array.
NSP	Integer scalar. Defines number of supplemental planes being input from input control file.

6.16 SUBROUTINE LSEGINT

IFLAG	Integer scalar. Flag indicating intersection (IFLAG=1), or no intersection (IFLAG=0).
M(2)	Real array. Contain the slopes of the two lines.
P(4,2)	Real array. Contains coordinates of lines P1-P2 and R1-R2.
P1(2)	Real array. Contains X and Y coordinate of one end point of line P1-P2.
P2(2)	Real array. Contains X and Y coordinate of the other end point of line P1-P2.
R1(2)	Real array. Contains X and Y coordinate of one end point of line R1-R2.
R2(2)	Real array. Contains X and Y coordinate of the other end point of line R1-R2.
T(2)	Real array. Contains T ratio values, where

$$T = \frac{X0 - X1}{X2 - X1}$$

6.17 SUBROUTINE MAT

A(JA,1)	Real array. Array in matrix multiply C=AB.
B(JB,1)	Real array. Array in matrix multiply C=AB.
C(JC,1)	Real array. Array in matrix multiply C=AB.
JA	Integer scalar. First subscript value for array A.
JB	Integer scalar. First subscript value for array B.
JC	Integer scalar. First subscript value for array C.
LL	Integer scalar. Size of array A.

MM	Integer scalar. Size of array B.
NN	Integer scalar. Size of array C.
S	Real scalar. Running summation of matrix multiple.

6.18 SUBROUTINE NFRAME

ENDFRA	Integer scalar. End-of-frame halfword to Grinnell graphics system.
LU	Integer scalar. Dummy argument for PLOTS.
M	Interger scalar. Dummy argument for PLOTS.
MASK	Integer scalar. Halfword mask defining what corresponding bit in ENDFRA to change.
N	Integer scalar. Dummy argument for PLOTS.
STATUS	Integer scalar. Contains return status of the request.

6.19 SUBROUTINE OVERLAP

I	Integer scalar. Denotes object No. 1 to be tested.
III	Integer scalar. Same as variable I.
K	Integer scalar. Denotes object No. 2 to be tested.
KKK	Integer scalar. Same as variable K.
MFLAG	Integer scalar. Indicates overlap or not. MFLAG=0 indicates no overlap and MFLAG=1 indicates overlap.
NPTS1	Integer scalar. Number of sides plus one for object No. 1.
NPTS2	Integer scalar. Number of sides plus one for object No. 2.

P1(2)	Real array. Contains X and Y coordinates on projection plane of first end point of first line segment.
P2(2)	Real array. Contains X and Y coordinates on projection plane of second and point of first line segment.
R1(2)	Real array. Contains X and Y coordinates on projection plane of first end point of second line segment.
R2(2)	Real array. Contains X and Y coordinates on projection plane of second end point of second line segment.

6.20 SUBROUTINE PLPLN

A	Real scalar. The fraction of a contour line that a single step represents.
I1	Integer scalar. Starting point number for the current side.
I2	Integer scalar. Ending point number for the current side.
INDEX2	Integer scalar. Variable telling how large to dimension SEG in subroutine PNTPLT.
IPEN	Integer scalar. Calcomp pen control variable.
NSIDES	Integer scalar. Number of points for present polygon.
NUM	Integer scalar. Total number of points that make up the contour of a plane.
SEG(3,3000)	Real array. Array of vectors that make up the sides of a polygon.

6.21 SUBROUTINE PNTPLT

IFLAG	Integer scalar. Flag passed back from subroutines HYDE and HIDE telling whether or not the line is hidden.

INDEX2	Integer scalar. Input argument telling how large to dimension array SEG. Calculated in Main as: (NSTEPS for that object × 4) + 1.
INUM	Integer scalar. Represents the number of objects which overlap with a particular object.
IPEN	Integer scalar. Pen value sent to the Calcomp subroutine PLOT. Either pen up (3) or pen down (2).
IPLOT	Integer scalar. Flag sent to subroutine CLIP to plot only first line segment after clipping determined.
KK	Integer scalar. Overlapping object number that overlaps with present object, from array IE.
LFLAG	Integer scalar. Flag set which tells whether or not last point was hidden. LFLAG=1 means point was hidden and LFLAG=2 means not hidden.
NEWPEN	Integer scalar. Variable contains value of last pen move. The number is positive if the pen move was performed and negative if the pen move was clipped.
NPTS	Integer scalar. Number of points to plot, from subroutines PLPLN and PSE.
P(3)	Real array. Contains coordinates of end point of line returned by subroutine EXTEND.
PP(3,2)	Real array. Contains coordinates of end points of line sent to subroutine EXTEND.
PPP(3)	Real array. Coordinates of point currently being plotted, converted to viewpoint reference frame.
SEG(3,INDEX2)	Real array. Contains coordinates of points to be plotted, passed through argument list by subroutines PSE and PLPLN.
X	Real scalar. Final X coordinate of point to be plotted.

XMAX	Real scalar. Constant that represents the right limit of the plotting region. Defined in data card 16.0 in input control file.
XMIN	Real scalar. Constant that represents the left limit of the plotting region. Defined in data card 16.0 in input control file.
XSAV	Real scalar. Represents X coordinate of previous point plotted.
Y	Real scalar. Final Y coordinate of point to be plotted.
YMAX	Real scalar. Constant that represents upper (top) limit of the plotting region.
YMIN	Real scalar. Constant that represents lower (bottom) limit of the plotting region.
YSAV	Real scalar. Represents Y coordinate of previously plotted point.

6.22 SUBROUTINE POLYD

D1(810)	Real array. Equivalent with array D, contains directional cosine matrices for the polygons.
INDEX(G)	Real array. Contains constants used for transposing data elements in D to get correct direction in matrix D.
J	Integer scalar. Calculated variable that converts a 3-dimensional subscript value to a single dimensional one.
NUM	Integer scalar. Offset used to put data from P array into SEGLP array.
SUMD1	Real scalar. Magnitude squared of the $\hat{x}$ coordinate axis of the polygon.
SUMD2	Real scalar. Magnitude squared of the $\hat{y}$ coordinate axis of the polygon.

6.23 SUBROUTINE PREPLT

IPEN	Integer scalar. Variable that denotes Calcomp pen move value from PNTPLT.
NEWPEN	Integer scalar. Saved value of the previous pen (IPEN) move.
X	Real scalar. X coordinate of first end point of line X,Y - XSAV, YSAV.
XLIMIT	Real scalar. If X coordinate outside valid plotting region is equal to boundary value (left or right), pen is off.
XMAX	Real scalar. Constant that represents right limit of valid plotting region. Defined in subroutine PNTPLT.
XMIN	Real scalar. Constant that represents left limit of valid plotting region. Defined in subroutine PNTPLT.
XOFF	Logical scalar. Is .FALSE. if X coordinate in plotting region, .TRUE. if outside.
XSAV	Real scalar. X coordinate of second end point of line X, Y - XSAV, YSAV.
XTEMP	Real scalar. X coordinate of the intercept point at YTEMP between X, Y and XSAV, YSAV.
Y	Real scalar. Y coordinate of the first end point of line X, Y - XSAV, YSAV.
YLIMIT	Real scalar. If Y coordinate outside valid plotting region, is equal to boundary value (top or bottom) pen is off.
YMAX	Real scalar. Constant that represents upper (top) limit of the plotting region. Defined in subroutine PNTPLT.
YMIN	Real scalar. Constant that represents lower (bottom) limit of the plotting region. Defined in subroutine PNTPLT.
YOFF	Logical scalar. Is .FALSE. if Y coordinate in plotting region, .TRUE. if outside.

YSAV	Real scalar. Y coordinate of the second end point of line X, Y, and XSAV, YSAV.
YTEMP	Real scalar. Y coordinate (either YMIN or YMAX) where pen went out of the plotting region.

6.24 SUBROUTINE PRJELR

A11	Real scalar. Represents the α_{11} component of $[\alpha]$.
A12	Real scalar. Represents the α_{12} component of $[\alpha]$.
A22	Real scalar. Represents the α_{22} component of $[\alpha]$.
DD(3,3)	Real array. Array containing intermediate values for transposing A array to A'.
DDD(3,3)	Real array. Contains values of the A' array.
LAMDA1	Real scalar. Eigenvalue used to circumscribe ellipse with rectangle.
LAMDA2	Real scalar. Eigenvalue used to circumscribe ellipse with rectangle.
M1	Real scalar. Value representing major axis vector.
M2	Real scalar. Value representing minor axis vector.
R(3,3)	Real array. Contains X, Y, and Z components of $\vec{r_1}$, $\vec{r_2}$, and $\vec{r_3}$.
R2(2,3)	Real array. Contains values of $\vec{r_1}$, $\vec{r_2}$, and $\vec{r_3}$ on the projection plane.
RX1	Real scalar. r_x value for $\vec{r_1}$.
RX2	Real scalar. r_x value for $\vec{r_2}$.
RY1	Real scalar. r_y value for $\vec{r_1}$.
RY2	Real scalar. r_y value for $\vec{r_2}$.

S(3)	Real array. Contains matrix multiple of DVP and SS.
SS(3)	Real array. Contains position vectors for individual ellipsoid minus inertial reference frame.

6.25 SUBROUTINE PRJPLY

NPTS	Integer scalar. Number of points in the polygon. Taken from array NPPP.
PP1(3)	Real array. Vector from the viewpoint origin to the point (vertex) in the inertial coordinate system.
PP2(3)	Real array. Vector from the viewpoint origin to the point (vertex) in the inertial coordinate system.
PPP2(3)	Real array. Vector from the viewpoint origin to the point (vertex) in the viewpoint coordinate system.

6.26 SUBROUTINE PSE

IHALF	Integer scalar. Flag that indicates which half of the ellipsoid to plot. IHALF=1, semiellipsoid with $X > 0$ plotted, IHALF=2, semiellipsoid with $X < 0$ plotted.
IN	Integer scalar. Number of points in each quarter contour saved in array X1.
INDEX	Integer scalar. Number of steps plus one.
INDEX2	Integer scalar. Maximum number of points any complete contour can have.
IPEN	Integer scalar. Calcomp pen control variable.
KK	Integer scalar. Variable in DO 60 loop which runs the loop backwards.
LINE	Integer scalar. Varaible in DO 100 loop which runs the loop backwards.

NPT	Integer scalar. Number of points in semiellipsoid to transfer to array SEG.
NPTS	Integer scalar. Number of points in semiellipsoid.
SEG(3,INDEX2)	Real array. Array containing a complete contour.
X1(3,INDEX,INDEX)	Real array. Semiellipsoid contour array.

6.27 SUBROUTINE ROT

A(M,3)	Real array. Output of ROT, rotation matrix to be computed.
C	Real scalar. Cosine of angle of rotation.
L	Integer scalar. Variable telling which axis to rotate about (X=1, Y=2, Z=3).
M	Integer scalar. First subscript value for array A.
S	Real scalar. Sine of angle of rotation.
TH	Real scalar. Angle of rotation (radians).

6.28 SUBROUTINES SOLVA, SOLVR

See Appendix B, Discussion of Equations used by PRJELR. The variables correspond to the equations found in this appendix.

6.29 SUBROUTINE TITLE

ID(10,20)	Integer array. Contains text for title frame.
NFRAME	Integer scalar. Number of title frames to plot.

NLINE	Integer scalar. Constant that represents number of lines in each title frame.
SIZE	Real scalar. X coordinate of where to start plotting title frame.
X	Real scalar. Height of characters in title frame.
Y	Real scalar. Y coordinate of where to start plotting title frame.

6.30 SUBROUTINE TPOINT

I	Integer scalar. Denotes polygon on projection plane.
IN	Integer scalar. Flag passed back to IN=1 caller, IN=1 means point inside polygon, means outside the polygon.
NPTS1	Integer scalar. Number of points for polygon I.
PP1(3)	Real array. Contains position vector for polygon I.
PP2(3)	Real array. Point on the projection plane to be tested.
R(3)	Real array. Contains differences between PP1 and PP2.
SIGN2	Real scalar. Test variable created like array SIGN.

6.31 SUBROUTINE TRANS1

DD(3,3)	Real array. Contains matrix multiple of DVP and parts of D.
P(3)	Real array. Output of TRANS1, it is position vector of an ellipsoid transformed to the viewpoint reference frame.
R(3)	Real array. Input to TRANS1, contains position vector for surface points of an ellipsoid in the segment coordinate system.

R2(3)	Real array. Contains the position vector of an ellipsoid in the viewpoint coordinate system.
SEGLP2(3)	Real array. Contains the location of the segment c.g. in the viewpoint coordinate system.

6.32 FUNCTION XINTCP

PFACTR	Real scalar. The slope of the line X, Y - XSAV, YSAV.
X	Real scalar. X coordinate of first end point of line X, Y - XSAV, YSAV.
X1	Real scalar. Difference between X and XSAV.
XINTCP	Real scalar. X coordinate value of YTEMP in line X, Y - XSAV, YSAV.
XSAV	Real scalar. X coordinate of second end point of line X, Y - XSAV, YSAV.
Y	Real scalar. Y coordinate of first end point of line X, Y - XSAV, YSAV.
Y1	Real scalar. Difference between Y and YSAV.
Y2	Real scalar. Difference between YTEMP and YSAV.
YSAV	Real scalar. Y coordinate of second end point of line X, Y - XSAV, YSAV.
YTEMP	Real scalar. Y coordinate value (between Y and YSAV) for which the caller wants the X coordinate.

6.33 SUBROUTINE XYZ

See Appendix B, "Hidden Line Problem Between Two Ellipsoids." Variables in XYZ correspond directly to the equations in this appendix.

6.34 FUNCTION YINTCP

PFACTR	Real scalar. The slope of the line X, Y - XSAV, YSAV.
X	Real scalar. X coordinate of the first end point of the line X, Y - XSAV, YSAV.
X1	Real scalar. Difference between X and X1.
X2	Real scalar. Difference between XTEMP and XSAV.
XSAV	Real scalar. X coordinate of second end point of line X, Y - XSAV, YSAV.
XTEMP	Real scalar. X coordinate value (between X and XSAV) for which the caller wants the Y coordinate.
Y	Real scalar. Y coordinate of first end point of line X, Y - XSAV, YSAV.
Y1	Real scalar. Difference between Y and YSAV.
YINTCP	Real scalar. Y coordinate value at XTEMP in line X, Y - XSAV, YSAV.
YSAV	Real scalar. Y coordinate of second end point of line X, Y - XSAV, YSAV.

6.35 SUBROUTINE YZ, Z

See Appendix B, "Hidden Line Problem Between Two Ellipsoids." Variables in YZ and Z correspond directly to the equations in this appendix.

7.0 SUBROUTINE, COMMON BLOCK, AND VARIABLE CROSS-REFERENCE CHARTS

This section contains three cross-reference charts: subprograms called by other subprograms, common blocks used by subprograms, and variables contained within each common block.

7.1 Subprogram Cross-Reference Chart

CALLING ROUTINE / CALLED ROUTINE	FREQ	MAIN	BUILDIE	CLIP	CONVREC	CROSS	DET	DOT	DOTT	DRCYPR	ELIPSN	EXTEND	GENDCM	HIDE	HYDE	INPUT	LSEGINT	MAT	NFRAME	OVERLAP	PLPLN	PNTPLT	POLYD	PREPLT	PRJELR	PRJPLY	PSE	ROT	SOLVA	SOLVR	TITLE	TPOINT	TRANS1	XINTCP	XYZ	YINTCP	YZ	Z
BUILDIE	1	1																																				
CLIP	3																					3																
CONVREC	1	1																																				
CROSS	2																						2															
DET	4				4																																	
DOT	9	1			1									2	2	2									1													
DOTT	3														1										1								1					
DRCYPR	1															1																						
ELIPSN	1	1																																				
EXTEND	1																					1																
GENDCM	1															1																						
HIDE	2										1											1																
HYDE	2										1											1																
INPUT	1	1																																				
LSEGINT	1																			1																		
MAT	16	2								2				2	4										3	1							2					
NFRAME	2	1																													1							
OVERLAP	3		3																																			
PLPLN	1	1																																				
PNTPLT	2																				1						1											
POLYD	1	1																																				
PREPLT	3																					3																
PRJELR	1	1																																				
PRJPLY	1	1																																				
PSE	2	2																																				
ROT	3								3																													
SOLVA	1																								1													
SOLVR	3																								3													
TITLE	1	1																																				
TPOINT	2													1						1																		
TRANS1	3													1								2																
XINTCP	3			2																				1														
XYZ	1														1																							
YINTCP	3			2																				1														
YZ	1														1																							
Z	1														1																							

7.2 Common Block Cross-Reference Chart

CALLING ROUTINE \ COMMON BLOCK	ATB	CONECT	DBUG	ELLIPSE	INTERS	PLTT	POLYGON	REMOVE	VIEWP
MAIN	*		*	*	*	*	*		*
BUILDIE				*	*		*	*	
CLIP									
CONVREC	*	*	*	*			*		
CROSS									
DET									
DOT									
DOTT									
DRCYPR									
ELIPSN				*					
EXTEND				*	*	*			
GENDCM									
HIDE				*			*		
HYDE				*					
INPUT	*	*	*	*	*	*	*	*	*
LSEGINT									
MAT									
NFRAME									
OVERLAP							*		
PLPLN			*	*		*	*	*	
PNTPLT				*	*	*			
POLYD				*			*		
PREPLT									
PRJELR				*			*		
PRJPLY				*			*		
PSE									
ROT									
SOLVA									
SOLVR									
TITLE			*						
TPOINT							*		
TRANS1				*					*
XINTCP									
XYZ									
YINTCP									
YZ									
Z									

7.3 Variable Cross-Reference Chart

COMMON BLOCK / VARIABLE	ATB	CONECT	DBUG	ELLIPSE	INTERS	PLTT	POLYGON	REMOVE	VIEWP
A				*					
BDRS			*						
CONVEC							*		
D				*					
DEVFLG			*						
DVP				*					
DVPO									*
ICOLOR						*			
IDEBUG			*						
IE					*				
IELP				*					
IFLAG							*		
INT						*			
IREMOV								*	
IVP									*
MPL		*							
NIE					*				
NISG			*						
NP		*							
NPLANE							*		
NPPP							*		
NPREM								*	
NSEG				*					
NSTEPS				*					
OFLINE			*						
OFSETX						*			
OFSETY						*			
ONLINE			*						
P							*		
PL	*								
POS							*		
PO							*		
RA				*					
SEGLP				*					
SFACTR						*			
SIGN							*		
TERM			*						
TIME						*			
VP				*					
VPO									*
VP2									*
ZTIME						*			

8.0 VIEW PROGRAM SOURCE LISTING

```
      PROGRAM VIEW
      COMMON/PLTT/SFACTR,INT,TIME,ICOLOR(91),OFSETX,OFSETY,ZTIME
      COMMON/INTERS/ NIE(90),IE(90,90)
      COMMON/ELLIPSE/NSTEPS(90),IELP,A(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      COMMON/POLYGON/NPLANE,IFLAG,NPPP(90),PO(3,4,60),P(3,4,60),
     *CONVEC(2,4,90),POS(2,90),SIGN(90)
      COMMON/ATB/PL(17,30)
      COMMON/VIEWP/VPO(3),DVPO(3,3),IVP,VP2(3)
      COMMON/DBUG/IDEBUG(50),NISG,DEVFLG,ONLINE,TERM,BDRS,OFLINE
      DIMENSION WORK(10000)
      DIMENSION ID(10)
      INTEGER DEVFLG,ONLINE,TERM,EDRS,OFLINE
      DOUBLE PRECISION ZTIME,CTIME,STIME,DTIME,ETIME
C
C     PROGRAM VIEW              VERSION 1.1          MARCH 9,1983
C     WRITTEN BY SRL IN SUPPORT OF THE
C     MATHEMATICS AND ANALYSIS BRANCH OF AMRL
C     AT WPAFB.
C
C     STORED ON TAPE1 THAT IS OUTPUT FROM THE ATBM VERSION V5D.
C
C     THIS PROGRAM USES CONTOUR LINES TO REPRESENT THE 3-D PROPERTIES OF
C     THE DATA ON TAPE1. THE CONTOUR LINES ARE PLOTTED ON PAPER THAT
C     REPRESENTS THE PROJECTION PLANE. THE POINTS THAT COMPOSE A CONTOUR
C     LINE IN 3-SPACE ARE PROJECTED THROUGH A POINT ON TO THE PROJECTION
C     PLANE.
C
C     CURRENTLY THERE ARE TWO CLASSES OF OBJECTS THAT ARE PLOTTED USING
C     CONTOUR LINES.
C        CLASS 1 - ELLIPSOIDS
C     ELLIPSOIDS ARE USED TO REPRESENT BODY SEGMENTS. THIS PROGRAM ALLOWS
C     ELLIPSOIDS TO BE IMBEDDED IN OTHER ELLIPSOIDS.
C
C        CLASS 2 - CONVEX POLYGONS
C     CONVEX POLYGONS ARE USED TO REPRESENT OBJECTS THAT CAN BE DEFINED
C     BY A SET OF PLANES. ALL POLYGONS DEFINED BY THE INPUT MUST BE
C     CONVEX POLYGONS; CONCAVE POLYGONS CAN BE OBTAINED USING A
C     COMBINATION OF CONVEX POLYGONS.
C
C     THE HIDDEN LINE ROUTINES CHECK FOR POINTS HIDDEN BY
C     ELLIPSOIDS OR POLYGONS. THESE ROUTINES MUST CHECK
C     FOR ANY POSSIBLE OBJECT THAT MAY BE BLOCKING THE
C     CURRENT POINT AS SEEN FROM THE VIEWPOINT.
C     IN ORDER TO ELIMINATE CHECKING ALL OBJECTS FOR
C     EACH POINT, SUBROUTINES ARE INCLUDED IN THE
C     VIEW PROGRAM THAT DETECT OBJECT OVERLAP ON THE
C     PROJECTION PLANE. BEFORE THE PLOTTING PHASE OF THE
C     VIEW PROGRAM, OBJECTS WHICH OVERLAP EACH OTHER ON
C     THE PROJECTION PLANE ARE RECORDED IN THE IE ARRAY.
C     DURING THE PLOTTING PHASE OF THE VIEW PROGRAM, THE
C     IE ARRAY IS REORDERED TO DECREASE THE SEARCH TIME
C     FOR OBJECTS THAT MAY BE BLOCKING THE CURRENT POINT
C     BEING PLOTTED. THE ASSUMPTION USED HERE IS - IF AN
C     OBJECT BLOCKED THE PREVIOUS POINT ON A CONTOUR LINE
C     THEN THAT OBJECT PROBABLY BLOCKS THE NEXT POINT
C     ON THE CONTOUR LINE.
C
      LUPLOT=8
      ONLINE=1
      TERM=2
      NF=0
```

```
      OFLINE=3
      BDRS=4
      READ(5,130) DEVFLG
      WRITE(6,130) DEVFLG
130   FORMAT(I1)
C
      IF (DEVFLG.EQ.1.OR.DEVFLG.EQ.3) CALL PLOTS(0,0,LUPLOT)
      IF (DEVFLG.EQ.2.OR.DEVFLG.EQ.4) CALL PLOTS(0,0,LUPLOT)
C
      IFLAG = 0
      CALL TITLE
      READ(5,200) (ID(I),I=1,10)
      WRITE(6,200) (ID(I),I=1,10)
  200 FORMAT(10A4)
      READ(5,150) STIME,DTIME,ETIME
150   FORMAT(3D10.0)
      CTIME=STIME-DTIME
      ITIME=CTIME*1000000.D0
      CTIME=ITIME/1000000.D0
      READ(5,125) IDEBUG
      WRITE(6,125) IDEBUG
 125  FORMAT(80I1)
  100 CONTINUE
      IFLAG = IFLAG + 1
      CTIME=CTIME+DTIME
      ITIME=CTIME*1000000.D0
      CTIME=ITIME/1000000.D0
      IF(CTIME.GT.ETIME) CALL PLOT(0.,0.,999)
      IF(CTIME.GT.ETIME) STOP
      CALL INPUT(CTIME)
      IF(IFLAG.EQ.10) GO TO 100
      IF(DEVFLG.EQ.OFLINE.OR.DEVFLG.EQ.BDRS) CALL NEWPEN(ICOLOR(91))
      XTIME=ZTIME*1000.0
      NF=NF+1
      WRITE(6,1000)NF
1000  FORMAT(' MAIN - PROCESSING FRAME #',I4)
      CALL PLOT(0.,0.,-3)
      CALL SYMBOL(.5,10.0,.335,ID,0.,35)
      CALL SYMBOL(.5,9.0,.335,'TIME(MSEC)',0.,10)
      CALL NUMBER(3.85,9.0,.335,XTIME,0.,-1)
      CALL MAT(DVP0,D(1,1,IVP),DVP,3,3,3,3,3,3)
      CALL DOT(D(1,1,IVP),VP0,VP,3,1,3)
      DO 10 K=1,3
   10 VP(K)=VP(K)+SEGLP(K,IVP)
      CALL MAT(DVP,VP,VP2,3,3,1,3,3,3)
      CALL CONVREC
      CALL PRJPLY
      CALL PRJELR
      CALL POLYD
      CALL BUILDIE
      IF(IDEBUG(1).EQ.1) WRITE(6,350) (NIE(I),I=1,90)
      IF(IDEBUG(2).EQ.1) WRITE(6,350) ((IE(I,J),I=1,90),J=1,90)
  350 FORMAT(270(30(1X,I2)/))
      DO 30 IK=1,NSEG
      IF(DEVFLG.EQ.OFLINE.OR.DEVFLG.EQ.BDRS) CALL NEWPEN(ICOLOR(IK))
      IELP=IK
      INDEX=NSTEPS(IK)+1
      INDEX2=4*INDEX-3
      IX1=1
      IIN=3*INDEX*INDEX+IX1
      ISEG=IIN+INDEX
      CALL ELIPSN(INDEX,WORK(IX1),WORK(IIN))
      CALL PSE(WORK(IX1),WORK(IIN),WORK(ISEG),INDEX,INDEX2,1)
```

```
      CALL PSE(WORK(IX1),WORK(IIN),WORK(ISEG),INDEX,INDEX2,2)
   30 CONTINUE
      CALL PLPLN(WORK,INDEX2)
      IF(DEVFLG.EQ.BDRS) CALL NFRAME
      IF(DEVFLG.EQ.OFLINE.OR.DEVFLG.EQ.ONLINE) CALL PLOT (12.,0.0,-3)
      IF(DEVFLG.EQ.TERM) CALL PLOT(0.,0.,-3)
      GO TO 100
      END
```

```
      SUBROUTINE BUILDIE
C
C     ONCE THIS SUBROUTINE IS CALLED ALL OBJECTS ARE REPRESENTED BY
C     POLYGONS PROJECTED ON THE VIEWPOINT PROJECTION PLANE.
C     THIS SUBROUTINE WILL BUILD THE IE AND NIE ARRAYS.
C     NEE(K) REPRESENTS THE NEMBER OF ENTRIES IN THE IE(I,K) ARRAY FOR
C     OBJECT K.
C     THE IE(I,K) ARRAY CONTAINS OBJECT NUMBERS FOR OBJECTS THAT OVERLAP
C     IN THE PROJECTION PLANE.
C     FOR EXAMPLE, IE(1,2) MIGHT CONTAIN A 3 WHICH MEANS OBJECT 3
C     OVERLAPS WITH OBJECT 2.
      COMMON/POLYGON/NPLANE,IFLAG,NPPP(90),PO(3,4,60),P(3,4,60),
     *CONVEC(2,4,90),POS(2,90),SIGN(90)
      COMMON/ELLIPSE/NSTEPS(90),IELP,A(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      COMMON/INTERS/ NIE(90),IE(90,90)
      COMMON/REMOVE/NPREM,IREMOV(30)
      DO 5 I=1,90
      NIE(I) = 0
      DO 5 K=1,90
    5 IE(I,K) = 0
      IF(NSEG .EQ. 0) GO TO 60
      DO 55 I=1,NSEG
      IF(NIE(I) .NE. 0)  GO TO 10
      NIE(I) = 1
      IE(1,I) = I
   10 IOBJ = I + 1
      IF(I.EQ.NSEG) GO TO 31
      DO 30  K=IOBJ,NSEG
      CALL OVERLAP(I,K,MFLAG)
      IF(MFLAG .EQ. 0)  GO TO 30
C
C     YES, THERE IS OVERLAP BETWEEN I AND K
C
      IF(NIE(K) .NE. 0)  GO TO 20
      NIE(K) = 1
      IE(1,K) = K
   20 NIE(K) = NIE(K) + 1
      NIE(I) = NIE(I) + 1
      IE(NIE(I),I) = K
      IE(NIE(K),K) = I
   30 CONTINUE
   31 CONTINUE
      IF(NPLANE.EQ.0) GO TO 55
      IOBJ = NPLANE + 30
      DO 50  K=31,IOBJ
      DO 200 LT=1,NPREM
      IF(K-30.EQ.IREMOV(LT)) GO TO 50
200   CONTINUE
      CALL OVERLAP(I,K,MFLAG)
      IF(MFLAG .EQ. 0)  GO TO 50
C
C     YES, THERE IS OVERLAP BETWEEN I AND K
C
      NIE(K) = NIE(K) + 1
      NIE(I) = NIE(I) + 1
      IE(NIE(I),I) = K
      IE(NIE(K),K) = I
   50 CONTINUE
   55 CONTINUE
C
C     NOW CHECK PLANE AGAINST PLANE
```

```
C
   60 IF(NPLANE .LE. 1)  RETURN
      MPLANE=NPLANE+29
      KPLANE=NPLANE+30
      DO 100 I=31,MPLANE
      DO 300 LT=1,NPREM
      IF(I-29.EQ.IREMOV(LT)) GO TO 100
300   CONTINUE
      IOBJ = I + 1
      DO 400 K=IOBJ,KPLANE
      CALL OVERLAP(I,K,MFLAG)
      IF(MFLAG .EQ. 0)  GO TO 400
C
C     YES, THERE IS OVERLAP BETWEEN I AND K
C
      NIE(K) = NIE(K) + 1
      NIE(I) = NIE(I) + 1
      IE(NIE(I),I) = K
      IE(NIE(K),K) = I
  400 CONTINUE
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CLIP(X,Y,XSAV,YSAV,XMIN,XMAX,YMIN,YMAX,IPEN,IPLOT)
C    ***********************************************
C
C   THIS SUBROUTINE CLIPS PLOTTING OFF BOTH ENDS OF THE CALCOMP DRUM
C
C    ***********************************************
C
      LOGICAL XOFF,YOFF
      DATA LCALL/0/
C
C   DETERMINE IF X AND/OR Y CLIPPING AND IF OFF TOP,BOTTOM,LEFT,
C   OR RIGHT OF PLOTTER
C
      XOFF=.FALSE.
      YOFF=.FALSE.
      IF (X.LT.XMIN.OR.X.GT.XMAX) XOFF=.TRUE.
      IF (Y.LT.YMIN.OR.Y.GT.YMAX) YOFF=.TRUE.
      IF (X.LT.XMIN) XLIMIT=XMIN
      IF (X.GT.XMAX) XLIMIT=XMAX
      IF (Y.LT.YMIN) YLIMIT=YMIN
      IF (Y.GT.YMAX) YLIMIT=YMAX
C
C   IF PREVIOUS CALL TO CLIP WAS A PEN UP TO 1ST POINT IN SEGMENT,
C   INTERPOLATE USING NEW AND SAVED COORD.'S, MOVE PEN, RESET SAVE
C   VALUES FOR X+Y POINTS, AND CONTINUE
C
      IF (LCALL.EQ.0) GO TO 10
          IF (XOFF) YLTEMP=YINTCP(X,Y,XSAV,YSAV,XLIMIT)
          IF (.NOT.XOFF) YLTEMP=YLIMIT
          IF (YOFF) XLTEMP=XINTCP(X,Y,XSAV,YSAV,YLIMIT)
          IF (.NOT.YOFF) XLTEMP=XLIMIT
          IF (XLTEMP.LT.XMIN) XLTEMP=XMIN
          IF (XLTEMP.GT.XMAX) XLTEMP=XMAX
          IF (YLTEMP.LT.YMIN) YLTEMP=YMIN
          IF (YLTEMP.GT.YMAX) YLTEMP=YMAX
          CALL PLOT(XLTEMP,YLTEMP,3)
          XSAV=XLTEMP
          YSAV=YLTEMP
          LCALL=0
C
C   IF 1ST POINT OF SEGMENT AND PEN UP, SAVE THESE COORD.'S, SET
C   FLAG, AND EXIT
C
10    CONTINUE
      IF (IPLOT.NE.1.OR.IPEN.NE.3) GO TO 20
          XLSAV=X
          YLSAV=Y
          LCALL=1
          IPLOT=0
          RETURN
C
C   DO WE WANT TO PLOT?
C
20    CONTINUE
      IF (IPLOT.NE.1) GO TO 30
C
C   DETERMINE X AND Y COORDINATES TO PLOT TO
C
      IF (XOFF) YTEMP=YINTCP(X,Y,XSAV,YSAV,XLIMIT)
      IF (.NOT.XOFF) YTEMP=YLIMIT
      IF (YOFF) XTEMP=XINTCP(X,Y,XSAV,YSAV,YLIMIT)
      IF (.NOT.YOFF) XTEMP=XLIMIT
```

```
C
C  PLOT ONLY THE FIRST SEGMENT AFTER CLIPPING DETERMINED, IGNORING
C  ALL SEGMENTS AFTER UNLESS PEN TO BE LIFTED.
C
      CALL PLOT(XTEMP,YTEMP,IPEN)
30    CONTINUE
      IPLOT=0
      RETURN
      END
```

```
      SUBROUTINE CONVREC
C
C      THIS SUBROUTINE CONVERTS RECTANGLES IN THE ATB
C     SIMULATION FORMAT TO POLYGONS IN THE VIEW PLOTTING FORMAT.
C
      COMMON/ATB/PL(17,30)
      DIMENSION R(3)
      DIMENSION DX(3)
      COMMON/POLYGON/NPLANE,IFLAG,NPPP(90),PO(3,4,60),P(3,4,60),
     *CONVEC(2,4,90),POS(2,90),SIGN(90)
      COMMON/CONECT/ NP,MPL(3,5,60)
      COMMON/DBUG/IDEBUG(80),NISG,DEVFLG,ONLINE,TERM,BDRS,OFLINE
      COMMON/ELLIPSE/NSTEPS(90),IELP,A(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      INTEGER DEVFLG,ONLINE,TERM,BDRS,OFLINE
      IF(NPLANE.EQ.0) RETURN
      IF(IDEBUG(4).NE.0) WRITE(6,50)
   50 FORMAT(1H1,'PLANE INFORMATION',/,1H ,17(1H*))
      DO 100 J=1,NPLANE
      IF(J.GT.NP) GO TO 15
      IF(IFLAG.NE.1) GO TO 15
      DDD=DET(PL(1,J),PL(2,J),PL(3,J),PL(8,J),PL(9,J),PL(10,J),
     -        PL(13,J),PL(14,J),PL(15,J))
      DX(1) = DET(PL(4,J),PL(2,J),PL(3,J),PL(11,J),PL(9,J),PL(10,J),
     -            PL(16,J),PL(14,J),PL(15,J))
      DX(2) = DET(PL(1,J),PL(4,J),PL(3,J),PL(8,J),PL(11,J),PL(10,J),
     -            PL(13,J),PL(16,J),PL(15,J))
      DX(3) = DET(PL(1,J),PL(2,J),PL(4,J),PL(8,J),PL(9,J),PL(11,J),
     -            PL(13,J),PL(14,J),PL(16,J))
      DO 10  I=1,3
      TEMP=DX(I)/DDD
      PO(I,1,J)=TEMP
      PO(I,2,J)=PL(I+7,J)*PL(12,J)+TEMP
      PO(I,3,J)=PL(I+12,J)*PL(17,J)+PO(I,2,J)
   10 PO(I,4,J)=PL(I+12,J)*PL(17,J)+TEMP
   15 CONTINUE
      ISG=MPL(1,1,J)
      IF(ISG.EQ.0) ISG=NISG
      IF(IDEBUG(4).NE.0) WRITE(6,200) ISG
  200 FORMAT(1X,'ISG=',I2)
      DO 20 L=1,4
      CALL DOT(D(1,1,ISG),PO(1,L,J),R,3,1,3)
      DO 20 I=1,3
      P(I,L,J)=R(I)+SEGLP(I,ISG)
   20 CONTINUE
      IF(J.LE.NP) NPPP(J+30)=4
      IF(IDEBUG(4).NE.0) WRITE(6,3000)
 3000 FORMAT(1X,30(1H*))
      IF(IDEBUG(4).NE.0) WRITE(6,2000) J
 2000 FORMAT(3X,'PLANE NUMBER = ',I3)
      JJ=J
      NPPPP=NPPP(J+30)
      IF(IDEBUG(4).NE.0) WRITE(6,1000)((P(I,K,JJ),I=1,3),K=1,NPPPP)
 1000 FORMAT(3X,F7.2,3X,F7.2,3X,F7.2)
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CROSS(A,B,C)
C     COMPUTES VECTOR CROSS PRODUCT C=AXB
      DIMENSION A(3),B(3),C(3)
      C(1)=A(2)*B(3)-A(3)*B(2)
      C(2)=A(3)*B(1)-A(1)*B(3)
      C(3)=A(1)*B(2)-A(2)*B(1)
      RETURN
      END
```

```
      FUNCTION DET(A11,A12,A13,A21,A22,A23,A31,A32,A33)
      DET=A11*(A22*A33-A23*A32)-A12*(A21*A33-A23*A31)
     1+A13*(A21*A32-A22*A31)
      RETURN
      END
```

```
      SUBROUTINE DOT(A,B,C,N,M,L)
C                                                      REV 03 05/31/73
C     PERFORMS MATRIX MULTIPLICATION C = A'B.
C     IF A AND B ARE VECTORS, C IS THE DOT PRODUCT A.B
C
C       ARGUMENTS:
C         A: MATRIX OF SIZE (L,N).
C         B: MATRIX OF SIZE (L,M).
C         C: PRODUCT MATRIX OF SIZE (N,M).
C         N,M,L: SIZES OF MATRICES A,B,C.
C
C       (NOTE: SUBROUTINE ASSUMES THAT THE FIRST DIMENSION
C              OF A,B AND C IN THE CALLING PROGRAM IS L,L AND N.)
C
      DIMENSION A(L,1),B(L,1),C(N,1)
      DO 10 I=1,N
      DO 10 J=1,M
      C(I,J) = 0.0
      DO 10 K=1,L
   10 C(I,J) = C(I,J) + A(K,I)*B(K,J)
      RETURN
      END
```

```
      SUBROUTINE DOTT(A,B,C,N,M,L)
C                                                   REV 01 11/20/72
C     PERFORMS MATRIX MULTIPLICATION C = AB'
C     WHERE DIMENSIONS ARE A(N,L) , B(M,L) AND C(N,M).
C
      DIMENSION A(N,1),B(M,1),C(N,1)
      DO 10 I=1,N
      DO 10  J=1,M
      C(I,J)=0.
      DO 5  K=1,L
    5 C(I,J)= A(I,K)*B(J,K)+C(I,J)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DRCYPR (D,A,I1,I2,I3)
C                                                     REV 03 07/08/74
C     SETS UP 3X3 DIRECTION COSINE MATRIX FOR GIVEN YAW,PITCH AND ROLL.
C
C       ARGUMENTS:
C         D: 3X3 DIRECTION COSINE MATRIX TO BE COMPUTED.
C         A: ARRAY OF LENGTH 3 CONTAINING ROTATATION ANGLES (DEGREES).
C        I1: AXIS OF ROTATION FOR 1ST ANGLE (1,2,3 = X,Y,Z)
C        I2: AXIS OF ROTATION FOR 2ND ANGLE (1,2,3 = X,Y,Z)
C        I3: AXIS OF ROTATION FOR 3RD ANGLE (1,2,3 = X,Y,Z)
C
      DIMENSION D(3,3),A(3),T(6,3)
      RADIAN=.0174532925199433
      Y = A(1)*RADIAN
      P = A(2)*RADIAN
      R = A(3)*RADIAN
      M = 6
      N = 3
      DO 10 I=1,3
      DO  5 J=1,3
      D(I,J)=0.
    5 T(I,J)=0.
      T(I,I)=1.
   10 D(I,I)=1.
      IF(Y.EQ.0.)GO TO 20
      CALL ROT(T,I1,Y,M)
      DO 15 I=1,3
      DO 15 J=1,3
   15 D(I,J)=T(I,J)
   20 IF(P.EQ.0.0)GO TO 30
      CALL ROT(T(4,1),I2,P,M)
      CALL MAT(T(4,1),T(1,1),D(1,1),3,3,3,M,M,N)
      DO 25 I=1,3
      DO 25 J=1,3
   25 T(I,J)=D(I,J)
   30 IF(R.EQ.0.0) GO TO 40
      CALL ROT(T(4,1),I3,R,M)
      CALL MAT(T(4,1),T(1,1),D(1,1),3,3,3,M,M,N)
   40 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ELIPSN(INDEX,X1,IN)
C    ***********************************************
C
C     THIS SUBROUTINE GENERATES 1/4 OF THE CONTOUR LINES FOR AN ELLIPSOID
C
C    ***********************************************
      DIMENSION X1(3,INDEX,INDEX),IN(INDEX)
      COMMON/ELLIPSE/NSTEPS(90),IELP,A(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      DELTAX=SQRT(1/A(1,1,IELP))/NSTEPS(IELP)
      DELTAY=SQRT(1/A(2,2,IELP))/NSTEPS(IELP)
      SIMP1 = A(1,1,IELP)
      SIMP2 = A(2,2,IELP)
      SIMP3 = A(3,3,IELP)
      DO 101 L=1,INDEX
      X=(L-1)*DELTAX
      N=0
      DO 50 K=1,INDEX
      Y=(K-1)*DELTAY
      TEMP = 1-X*X*SIMP1
      TEST = TEMP - Y*Y*SIMP2
      N=N+1
      IF(TEST.LT.0.0) GO TO 100
      Z = SQRT(TEST/SIMP3)
      X1(1,N,L)=X
      X1(2,N,L)=Y
      X1(3,N,L)=Z
   50 CONTINUE
      GO TO 101
  100 Y = SQRT(TEMP/SIMP2)
      X1(1,N,L)=X
      X1(2,N,L)=Y
      X1(3,N,L)=0.0
  101 IN(L)=N
      RETURN
      END
```

```
      SUBROUTINE EXTEND(P,I,J)
      COMMON/INTERS/ NIE(90),IE(90,90)
      COMMON/ELLIPSE/NSTEPS(90),IELP,A(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      DIMENSION P(3,2),P3(3)
      COMMON/PLTT/SFACTR,INT,TIME,ICOLOR(91),OFSETX,OFSETY,ZTIME
C         ******************************************
C
C         THIS SUBROUTINE FINDS A MIDPOINT FOR A LINE THAT
C        BEGINS ON P(1) AND ENDS ON P(2).
C         THIS NEW POINT IS CHECKED BY SUBROUTINE HYDE.
C         IF IT IS HIDDEN THEN P(I)=P3
C         IF IT IS NOT HIDDEN THEN P(J)=P3
C         THIS ALGORITHM IS ITERATED INT TIMES.
C         UPON LEAVING EXTEND- P(I) WILL CONTAIN THE RESULT.
C
C         NOTE: P ARRAY IS CHANGED BY THIS SUBROUTINE.
C
C         ******************************************
      DO 3 IN=1,INT
      DO 1 L=1,3
    1 P3(L)=(P(L,2)+P(L,1))/2.0
      NUM=NIE(IELP)
      DO 2 IM=1,NUM
      KK=IE(IM,IELP)
      IF(KK.LE.30) CALL HYDE(KK,P3,IFLAG)
      IF(KK.GT.30) CALL HIDE(KK,P3,IFLAG)
      IF(IFLAG.EQ.1) GO TO 10
    2 CONTINUE
   10 N=I
      IF(IFLAG.EQ.1) N=J
      DO 3 L=1,3
    3 P(L,N)=P3(L)
      RETURN
      END
```

```
      SUBROUTINE GENDCM(CAMERA,FOCUS,D)
C    **********************************************
C     THIS SUBROUTINE GENERATES A DIRECTION COSINE MATRIX.
C
      DIMENSION CAMERA(3),FOCUS(3),Z(3),D(3,3)
      SUM = 0.0
      IF (FOCUS(1).NE.0.0.OR.FOCUS(2).NE.0.0.OR.
     *    CAMERA(1).NE.0.0.OR.CAMERA(2).NE.0.0) GO TO 50
      DO 40 I=1,3
      DO 40 J=1,3
      D(I,J)=0.0
40    CONTINUE
      D(1,2)=1.
      D(2,1)=1.
      D(3,3)=-1.
      GO TO 999
50    CONTINUE
C
      DO 100  I=1,3
      Z(I) = FOCUS(I) - CAMERA(I)
  100 SUM = SUM + Z(I)*Z(I)
      SUM = SQRT(SUM)
      DO 200  I=1,3
  200 Z(I) = Z(I)/SUM
C
      XNORM = SQRT(Z(1)*Z(1) + Z(2)*Z(2))
C
C FILL IN FIRST ROW OF D
C
       D(1,1) = Z(2)/XNORM
      D(1,2) = -Z(1)/XNORM
      D(1,3) = 0.0
C
C FILL IN SECOND ROW OF D
C
      D(2,1) = Z(1)*Z(3)/XNORM
      D(2,2) = Z(2)*Z(3)/XNORM
      D(2,3) = -XNORM
C
C FILL IN THIRD ROW OF D
C
      DO 300  I=1,3
  300 D(3,I) = Z(I)
999   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE HIDE(KK,P3,IFLAG)
      COMMON/POLYGON/NPLANE,KFLAG,NPPP(90),PO(3,4,60),P(3,4,60),
     *CONVEC(2,4,90),POS(2,90),SIGN(90)
      COMMON/ELLIPSE/NSTEPS(90),IELP,A(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      DIMENSION P7(2) ,PP(3)
      DIMENSION P3(3),P4(3)
      REAL PPRIME(3),NPRIME(3)
      CALL TRANS1(P3,P4)
      P7(1)=P4(1)/P4(3)
      P7(2)=P4(2)/P4(3)
      CALL TPOINT(P7,KK,IFLAG)
      IF(IFLAG.EQ.2) RETURN
C
C        POINT IS INSIDE POLYGON CHECK TO SEE
C       IF POLYGON OR POINT IS CLOSER TO VIEWPOINT.
C               CALCULATE TAU.
C
      DO 5 I=1,3
      PPRIME(I)=D(1,I,KK)
      IF (PPRIME(I).EQ.0.0) PPRIME(I)=.000001
5     CONTINUE
      CALL MAT(DVP,PPRIME,NPRIME,3,3,1,3,3,3)
      DO 10 J=1,3
   10 PP(J)=P(J,1,KK-30)-VP(J)
      CALL MAT(DVP,PP,PPRIME,3,3,1,3,3,3)
      CALL DOT(NPRIME,PPRIME,P5,1,1,3)
      CALL DOT(NPRIME,P4,P6,1,1,3)
      IFLAG=2
      IF(P5/P6.GE..99999999) RETURN
      IFLAG=1
      RETURN
      END
```

```
      SUBROUTINE HYDE(N,R,IFLAG)
C
C         ****************************************
C
C         SUBROUTINE HYDE DETERMINES IF A POINT IS HIDDEN BY
C          ANOTHER ELLIPSOID.
C
C         ****************************************
C         *****************************************
C         N= POSSIBLE HIDING ELLIPSOID NUMBER.
C         R= VECTOR TO PLOTTING POINT.
C         IFLAG= FLAG THAT INDICATES HIDDEN LINE OR NOT.
C              IFLAG = 2 = NOT HIDDEN
C              IFLAG = 1 = HIDDEN
C         *****************************************
      COMMON/ELLIPSE/NSTEPS(90),IELP,AA(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      DIMENSION P1(3),P2(3),R2(3),S(3),V(3)
      DIMENSION MU(3),M(3,2),P(3)
      DIMENSION DD(3,3),VP1(3)
      DIMENSION R(3)
      REAL M , MU , MAG
C        ASSUME NOT HIDDEN.
      IFLAG=2
C
C   *****************************************************************
C
C         PUT SEGLP(N) IN N'S FRAME.
C         PUT SEGLP(M) IN N'S FRAME.
C         PUT R2 IN N'S FRAME.
C         PUT VIEW POINT IN N'S FRAME.
C
C   *****************************************************************
      CALL MAT(D(1,1,N),SEGLP(1,N),P1,3,3,1,3,3,3)
      CALL MAT(D(1,1,N),SEGLP(1,IELP),P2,3,3,1,3,3,3)
      IF(IELP .LE. 30)  GO TO 55
      DO 56  I=1,3
      DO 56  J=1,3
   56 DD(I,J) = D(I,J,N)
      GO TO 57
   55 CONTINUE
      CALL DOTT(D(1,1,N),D(1,1,IELP),DD,3,3,3)
   57 CONTINUE
      CALL MAT(DD,R,R2,3,3,1,3,3,3)
      CALL MAT(D(1,1,N),VP,VP1,3,3,1,3,3,3)
      MAG = 0.0
C
C   *****************************************************************
C
C         FIND VECTORS S,V,AND MU.
C         MU WILL BECOME A UNIT VECTOR IN VECTOR M'S DIRECTION
C           OR IN M'S OPPOSITE DIRECTION.
C
C   *****************************************************************
C
      DO 1 I=1,3
      S(I) = P2(I) + R2(I) - P1(I)
      V(I) = VP1(I) - P1(I)
      MU(I) = S(I) - V(I)
    1 MAG = MAG + MU(I)**2
      MAG = SQRT(MAG)
C
```

```
C     ******************************************************************
C
C          MAKE MU A UNIT VECTOR.
C
C     ******************************************************************
C
      DO 2 I=1,3
    2 MU(I) = MU(I) / MAG
      A = AA(1,1,N)
      B = AA(2,2,N)
      C = AA(3,3,N)
      IF(ABS(MU(1)).GT..000000001) GO TO 10
      IF(ABS(MU(2)).GT..000000001) GO TO 20
      CALL Z(MU,A,B,C,S,M,JFLAG)
   30 IF(JFLAG.EQ.1) RETURN
C
C     ******************************************************************
C
C          FIND P AND COMPARE M TO P TO DETERMINE WHAT POINT
C           IS CLOSER TO THE VIEW POINT.
C
C     ******************************************************************
C
      DO 3 I=1,3
    3 P(I)=  S(I) - V(I) - M(I,1)
      CALL DOT(P,M(1,1),RESLT1,1,1,3)
      CALL DOT(P,M(1,2),RESLT2,1,1,3)
      IF(N.EQ.IELP) GO TO 400
      IF(RESLT1.GT.0.000000001) IFLAG=1
   41 IF(RESLT2.GT.0.000000001) IFLAG=1
      RETURN
   10 CALL XYZ(MU,A,B,C,S,M,JFLAG)
      GO TO 30
   20 CALL YZ(MU,A,B,C,S,M,JFLAG)
      GO TO 30
  400 IF(ABS(RESLT2).GT.ABS(RESLT1)) GO TO 41
      RESLT2=RESLT1
      GO TO 41
      RETURN
      END
```

```
      SUBROUTINE INPUT(CTIME)
      COMMON/PLTT/SFACTR,INT,TIME,ICOLOR(91),OFSETX,OFSETY,ZTIME
      COMMON/INTERS/ NIE(90),IE(90,90)
      COMMON/ELLIPSE/NSTEPS(90),IELP,A(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      COMMON/POLYGON/NPLANE,IFLAG,NPPP(90),PO(3,4,60),P(3,4,60),
     *CONVEC(2,4,90),POS(2,90),SIGN(90)
      COMMON/ATR/PL(17,30)
      DIMENSION BD(24,40)
      DIMENSION DD(3)
      COMMON/DBUG/IDEBUG(80),NISG,DEVFLG,ONLINE,TERM,BORS,OFLINE
      COMMON/VIEWP/VPO(3),DVPO(3,3),IVP,VP2(3)
      COMMON/CONECT/ NP,MPL(3,5,60)
      COMMON /REMOVE/NPREM,IREMOV(30)
      DOUBLE PRECISION CTIME,ZTIME
      INTEGER DEVFLG
      IF(IFLAG.NE.1) GO TO 600
      READ(5,70) NFAST,NPREM,NISG
      WRITE(6,70) NFAST,NPREM,NISG
      READ(5,72)(IREMOV(I),I=1,NPREM)
      WRITE(6,72)(IREMOV(I),I=1,NPREM)
72    FORMAT(3(40I2/))
      READ(1,END=300) NSEG,NP,PL,BD, (((MPL(I,J,K),I=1,3),  J=1,5),K
     *=1,30)
C
   39 READ(1,END=700) TIME, ((SEGLP(I,J), I=1,3), J=1,30),
     *         (((D(I,J,K), I=1,3), J=1,3), K=1,30)
      ITIME=TIME*1000000.+.5
      ZTIME=ITIME/1000000.D0
C
70    FORMAT(3I2)
      IF(ZTIME.LT.CTIME) GO TO 39
      READ(5,40) NSP
   40 FORMAT(I2)
      IF(NSP.EQ.0) GO TO 46
      DO 45 L=1,NSP
      K=NP+L
      II=30+NP+L
      READ(5,41) NPPP(II),MPL(1,1,K)
   41 FORMAT(I1,I2)
      NSIDES=NPPP(II)
      DO 45 J=1,NSIDES
      READ(5,42) (PO(I,J,K),I=1,3)
   42 FORMAT(3F10.0)
   45 CONTINUE
   46 NPLANE=NP+NSP
      DO 100 J=1,NSEG
      DO 100 I=1,3
  100 A(I,I,J)=1.0/BD(I,J)**2
      DO 200 J=1,NSEG
      CALL DOT(D(1,1,J),BD(4,J),DD,3,1,3)
      DO 200 I=1,3
  200 SEGLP(I,J)=SEGLP(I,J)+DD(I)
      IF(IDEBUG(3).EQ.1) WRITE(6,6) NSEG,NPLANE
    6 FORMAT(1H1,'NUMBER OF SEGMENTS = ',I2,',  NUMBER OF PLANES = ',I2)
      NSEG = NSEG-NFAST
      READ(5,301) (ICOLOR(I),I=1,30)
  301 FORMAT(8(5X,I5))
      II=NPLANE+30
      READ(5,301) (ICOLOR(I),I=31,90)
      READ(5,301) ICOLOR(91)
      IF(IDEBUG(3).EQ.1) WRITE(6,71) NSEG
```

```
   71 FORMAT(1X,'THE NUMBER OF SEGMENTS TO  BE PLOTTED = ',I2)
      READ(5,1) (NSTEPS(IPP),IPP=1,NSEG)
      READ(5,1)  (NSTEPS(IPP+30),IPP=1,NPLANE)
    1 FORMAT(30I2)
      IF(IDEBUG(3).EQ.1) WRITE(6,2) (NSTEPS(IPP),IPP=1,NSEG)
    2 FORMAT(10X,'NUMBER OF DIVISIONS ALONG A RADIUS',/2X,30I3)
      IF(IDEBUG(3).EQ.1) WRITE(6,55) (NSTEPS(IPP+30),IPP=1,NPLANE)
   55 FORMAT(10X,'NUMBER OF DIVISIONS ALONG A SIDE ',/2X,30I3)
      READ(5,11) INT,SFACTR
   11 FORMAT(I3,7X,F10.2)
      WRITE(6,11) INT,SFACTR
      READ(5,901) OFSETX,OFSETY
      WRITE(6,901) OFSETX,OFSETY
901   FORMAT(2F10.0)
      IF(IDEBUG(3).EQ.1) WRITE(6,902) OFSETX,OFSETY
902   FORMAT(1X,'OFSETX= ',F10.3,4X,'OFSETY= ',F10.3)
      IF(IDEBUG(3).EQ.1) WRITE(6,12) SFACTR,INT
   12 FORMAT(1X,'SCALE FACTOR = ',F10.2,2X,' ITERATION NUMBER = ',I3)
      READ(5,13) VP,RA,IVP,ICODE
   13 FORMAT(6F10.0,2I10)
C
C   ICODE = 0 :  ROLL, PITCH, AND YAW ANGLES ARE SUPPLIED IN RA ARRAY.
C
C   ICODE = 1 : DIRECTION COSINE MATRIX SUPPLIED AS INPUT.RA ARRAY IS 1ST
C               ROW OF MATRIX.THE NEXT CARD CONTAINS THE 2ND AND 3RD ROWS
C
C   ICODE = 2 :  POINT AT WHICH VIEWPOINT Z-AXIS IS TO AIM IS SUPPLIED
C                IN RA ARRAY.
C
      IF(ICODE .NE. 0)  GO TO 500
      IF(IDEBUG(3).EQ.1) WRITE(6,4) VP,RA
    4 FORMAT(1X,'VIEWPOINT VECTOR (',F10.1,',',F10.1,',',F10.1,')'/1X
     1'ROTATION OF VIEWPOINT RELATIVE TO SEGMENT COORDINATE SYSTEM'/1X
     2'THESE ROTATIONS MUST BE DETERMINED BY PERFORMING ROLL MOTION FIRS
     3T'/1X'THEN A PITCH MOTION AND THEN THE YAW MOTION'/10X
     4'ROLL = ',F10.1,1X,'DEG.',5X,'PITCH = ',F10.1,1X,'DEG.',
     55X,'YAW = ',F10.1,1X,'DEG.')
      CALL DRCYPR(DVP,RA,1,2,3)
      GO TO 550
  500 IF(ICODE .EQ. 2)  CALL GENDCM(VP,RA,DVP)
      IF(ICODE .EQ. 2)  GO TO 550
      DO 501 JJJ=1,3
  501 DVP(1,JJJ) = RA(JJJ)
      READ(5,13) ((DVP(I,J),J=1,3),I=2,3)
      WRITE(6,14)((DVP(I,J),J=1,3),I=1,3)
 14   FORMAT(3(/' ',3(1X,F10.0)))
      IF(IDEBUG(3).EQ.1) WRITE(6,15) (VP(I),I=1,3)
   15 FORMAT(' VIEW POINT VECTOR   (',F10.1,',',F10.1,',',F10.1,').',//,
     -  ' VIEW POINT ORIENTATION DEFINED IN DIRECTION COSINE MATRIX FORM
     -.')
  550 CONTINUE
      DO 80 J=1,3
      VPO(J)=VP(J)
      DO 80 I=1,3
   80 DVPO(J,I)=DVP(J,I)
      IF(IFLAG .NE. 1)  RETURN
      IF(IDEBUG(3).EQ.1) WRITE(6,60)
   60 FORMAT(1X,'********************'/,1X,
     1'VIEWPOINT DIRECTION COSINE MATRIX')
      IF(IDEBUG(3).EQ.1) WRITE(6,53)((DVP(I,J),J=1,3),I=1,3)
      IF(IDEBUG(3).EQ.1) WRITE(6,54)
      DO 20 IKE=1,NSEG
      IF(IDEBUG(3).EQ.1) WRITE(6,50) IKE,(SEGLP(I,IKE),I=1,3)
```

```
   50 FORMAT(1X,'********************'/1X'SEGMENT # ',I3,5X,'SEGLP = ('
     -F10.3,2(',' F10.3),','//,1X,'A MATRIX',T50,'DIRECTION COSINE MATRIX
     -  '/)
      DO 400  III=1,3
      IF(IDEBUG(3).EQ.1) WRITE(6,51) (A(III,J,IKE),J=1,3),(D(III,J,IKE),
     +J=1,3)
  400 CONTINUE
   51 FORMAT(3(2X,F3.5),T50,3(2X,F9.6))
   53 FORMAT(3(2X,F9.6))
      IF(IDEBUG(3).EQ.1) WRITE(6,54)
   54 FORMAT(1X,/,1X'********************')
   20 CONTINUE
      RETURN
600   CONTINUE
      IF(ZTIME.LT.CTIME) GO TO 675
      IF(ISW1.EQ.0) GO TO 680
      DO 650 J=1,NSEG
      CALL DOT(D(1,1,J),BD(4,J),DD,3,1,3)
      DO 650 I=1,3
  650 SEGLP(I,J)=SEGLP(I,J)+DD(I)
      ISW1=0
      IFLAG=5
      RETURN
  675 READ(1,END=700) TIME,((SEGLP(I,J),I=1,3),J=1,30),
     *          (((D(I,J,K), I=1,3), J=1,3), K=1,30)
      ITIME=TIME*1000000.+.5
      ZTIME=ITIME/1000000.D0
C
      ISW1=1
      GO TO 600
  700 WRITE(6,720)
  720 FORMAT(1X,'END OF DATA REACHED.')
      ISW1=1
      STOP
  800 WRITE(6,820)
  820 FORMAT(1X,'NO DATA ON TAPE.')
      STOP
  680 IFLAG=10
      RETURN
      END
```

```
      SUBROUTINE LSEGINT(P1,P2,R1,R2,IFLAG)
C
C THIS SUBROUTINE DETERMINES IF TWO LINE SEGMENTS, P1P2 AND R1R2,
C   INTERSECT.
C ALL PARALLEL LINE SEGMENTS,WHETHER COINCIDENT OR NOT, ARE
C   CONSIDERED TO BE NON-INTERSECTING.
C CASE 1 IS CONSIDERED TO BE THE REGULAR CONFIGURATION.
C THE SPECIAL CASES ARE AS FOLLOWS:
C  2)  ONE LINE IS VERTICAL
C  3)  BOTH LINES ARE VERTICAL
C  4)  BOTH LINES ARE HORIZONTAL
C  5)  BOTH LINES HAVE THE SAME NON-ZERO SLOPE
C  6)  ONE LINE IS VERTICAL, THE OTHER IS HORIZONTAL
C
C
C IFLAG = 1 INDICATES INTERSECTION; IFLAG = 0 INDICATES NO INTERSECTION.
C
      DIMENSION  P1(2),P2(2),R1(2),R2(2),P(4,2),T(2)
      REAL M(2)
      IFLAG=0
C
C  SET UP ARRAYS
      DO 1  I=1,2
      P(1,I) = P1(I)
      P(2,I) = P2(I)
      P(3,I) = R1(I)
    1 P(4,I) = R2(I)
C
C  DETERMINE IF CASE 3
      IF(ABS(P(1,1)-P(2,1)).LT.1.E-11.AND.ABS(P(3,1)-P(4,1)).LT.
     +1.E-11) RETURN
C
C  DETERMINE IF CASE 4
      IF(ABS(P(1,2)-P(2,2)).LT.1.E-11.AND.ABS(P(3,2)-P(4,2)).LT.
     +1.E-11) RETURN
C
C  DETERMINE IF CASE 6
      DO 2  I=1,2
      J = 3-I
      IF(ABS(P(1,I)-P(2,I)).LT.1.E-11.AND.ABS(P(3,J)-P(4,J)).LT.
     +1.E-11) GO TO 10
    2 CONTINUE
C
C  DETERMINE IF CASE 2
      IF(ABS(P(1,1)-P(2,1)).LT.1.E-11.OR.ABS(P(3,1)-P(4,1)).LT.
     +1.E-11) GO TO 6
      GO TO 5
    6 DO 3  I=1,4
      TEMP = P(I,1)
      P(I,1) = P(I,2)
    3 P(I,2) = TEMP
C
C  REGULAR PROCEDURE
    5 DO 4  I=1,2
      I1 = 2*I
      I2 = 2*I - 1
    4 M(I) = (P(I1,2) - P(I2,2))/(P(I1,1) - P(I2,1))
C
C  CHECK FOR CASE 5
      IF(ABS(M(1)-M(2)).LT.1.E-11) RETURN
      X = (P(3,2) - P(1,2) + M(1)*P(1,1) - M(2)*P(3,1))/(M(1) - M(2))
      DO 7  I=1,2
```

```
    7 T(I) = (X - P(2*I-1,1))/(P(2*I,1) - P(2*I-1,1))
   20 IF(T(1).GT.0 .AND.  T(2).GT.0 .AND. T(1).LT.1 .AND. T(2).LT.1)
     -    IFLAG=1
      RETURN
C
C     CASE 6 PROCEDURE
   10 IF(ABS(P(1,1)-P(2,1)).LT.1.E-11) GO TO 11
      J=1
      I=2
      GO TO 12
   11 I=1
      J=2
   12 T(1) = (P(3,J) - P(1,J))/(P(2,J) - P(1,J))
      T(2) = (P(1,I) - P(3,I))/(P(4,I) - P(3,I))
      GO TO 20
      END
```

```
      SUBROUTINE MAT(A,B,C,LL,MM,NN,JA,JB,JC)
C                                                      REV 03 05/31/73
C     PERFORMS MATRIX MULTIPLICATION C = AB.
C
C       ARGUMENTS:
C         A: MATRIX OF SIZE (L,M).
C         B: MATRIX OF SIZE (M,N).
C         C: PRODUCT MATRIX OF SIZE (L,N).
C         L,M,N: SIZES OF MATRICES A,B,C.
C         LA,LB,LC: 1ST DIMENSION OF A,B,C IN CALLING PROGRAM.
C
      DIMENSION  A(JA,1),B(JB,1),C(JC,1)
      DO 20 L=1,LL
      DO 10 N=1,NN
      S = 0.0
      DO 5 M=1,MM
    5 S=S+A(L,M)*B(M,N)
      C(L,N)=S
   10 CONTINUE
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE NFRAME
C
C THIS ROUTINE PERFORMS THE END OF FPAME HANDLING FOR THE BDRS
C
      INTEGER*2 ENDFRA,MASK,STATUS
      DATA ENDFRA,MASK/ZFFFF,ZFFFF/
      CALL DOLWH(S,1,ENDFRA,MASK,STATUS)
      CALL PLOTS(M,N,LU)
      RETURN
      END
```

```
      SUBROUTINE OVERLAP(III,KKK,MFLAG)
      DIMENSION P1(2),P2(2),R1(2),R2(2)
      DIMENSION PP2(2)
      COMMON/POLYGON/NPLANE,IFLAG,NPPP(90),P0(3,4,60),P(3,4,60),
     *CONVEC(2,4,90),POS(2,90),SIGN(90)
C
C     OVERLAP TAKES OBJECTS I AND K AND TESTS FOR ANY OVERLAP ON THE
C     PROJECTION PLANE.
C     MFLAG WILL BE RETURNED TO INDICATE IF OVERLAP OR NOT.
C          MFLAG=0 MEANS NO OVERLAP
C          MFLAG=1 MEANS OVERLAP
C
      I = III
      K = KKK
    5 CONTINUE
      DO 10  J=1,2
   10 PP2(J) = POS(J,K)
C
C     GO AROUND THE RINGS
C
      NPTS1 = NPPP(I)
      NPTS2 = NPPP(K)
      DO 200  J=1,NPTS2
      CALL TPOINT(PP2,I,MFLAG)
      IF(MFLAG.EQ.1) RETURN
      DO 200 N=1,2
  200 PP2(N) = PP2(N) + CONVEC(N,J,K)
C
C         CHECKED ALL POINTS AND FOUND THEY WERE ALL OUTSIDE.
C
C  NEXT, CHECK FOR INTERSECTING LINE SEGMENTS.
C
      DO 60  II=1,2
      P1(II) = POS(II,I)
   60 R1(II) = POS(II,K)
      DO 61  L=1,NPTS1
      DO 62  II=1,2
   62 P2(II) = P1(II) + CONVEC(II,L,I)
      DO 63  J=1,NPTS2
      DO 64  II=1,2
   64 R2(II) = R1(II) + CONVEC(II,J,K)
      CALL LSEGINT(P1,P2,R1,R2,MFLAG)
      IF(MFLAG .EQ. 1)  RETURN
      R1(1) = R2(1)
   63 R1(2) = R2(2)
      P1(1) = P2(1)
   61 P1(2) = P2(2)
      IF(I .NE. III)  RETURN
      I = KKK
      K = III
      GO TO 5
      END
```

```
      SUBROUTINE PLPLN(SEG,INDEX2)
C    ***********************************************
C
C     THIS SUBROUTINE PLOTS THE PLANES.
C
C    ***********************************************
      COMMON/ELLIPSE/NSTEPS(90),IELP,AA(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      COMMON/POLYGON/NPLANE,IFLAG,NPPP(90),PO(3,4,60),P(3,4,60),
     *CONVEC(2,4,90),POS(2,90),SIGN(90)
      COMMON/DBUG/IDEBUG(80),NISG,DEVFLG,ONLINE,TERM,BDRS,OFLINE
      DIMENSION SEG(3,3333)
      INTEGER DEVFLG,ONLINE,TERM,BDRS,OFLINE
      COMMON/PLTT/SFACTR,INT,TIME,ICOLOR(91),OFSETX,OFSETY,ZTIME
      COMMON /REMOVE/NPREM,IREMOV(30)
      IF(NPLANE .EQ. 0)  RETURN
      SEG(1,1) = 0.0
      SEG(2,1) = 0.0
      SEG(3,1) = 0.0
      DO 500  LL=1,NPLANE
      DO 100 I=1,NPREM
      IF (LL.EQ.IREMOV(I)) GO TO 500
100   CONTINUE
      L = LL + 30
      IF(DEVFLG.EQ.OFLINE.OR.DEVFLG.EQ.BDRS) CALL NEWPEN(ICOLOR(L))
      NUM = NPPP(L)*NSTEPS(L) + 1
      A = 1./NSTEPS(L)
      NSIDES = NPPP(L)
      DO 400  K=1,NSIDES
      KK = K + 1
      IF(K .EQ. NSIDES)  KK=1
      I1 = (K-1)*NSTEPS(L) + 2
      I2 = I1 + NSTEPS(L) - 1
      DO 400  I=I1,I2
      DO 400  J=1,3
  400 SEG(J,I) = SEG(J,I-1) + A*(P(J,KK,LL)-P(J,K,LL))
      IPEN = 3
      IELP = L
      CALL PNTPLT(SEG(1,1),IPEN,INDEX2,NUM)
  500 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PNTPLT(SEG,IPEN,INDEX2,NPTS)
C    **********************************************
C
C     POINT PLOT SUBROUTINE.
C
C    **********************************************
      COMMON/PLTT/SFACTR,INT,TIME,ICOLOR(91),OFSETX,OFSETY,ZTIME
      COMMON/ELLIPSE/NSTEPS(90),IELP,A(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      COMMON/INTERS/ NIE(90),IE(90,90)
      DIMENSION P(3),PP(3,2),PPP(3)
      DIMENSION SEG(3,3333)
      DATA YMIN/0.0/,YMAX/11.0/,IPLOT/1/
      DATA IFIRST/0/
      DATA ITWO/2/,ITHREE/3/
      IF (IFIRST.EQ.0) READ(5,1000)XMIN,XMAX
1000  FORMAT(2F10.2)
      IF (IFIRST.EQ.0) WRITE(6,1001)XMIN,XMAX
1001  FORMAT(' XMIN,XMAX=',2(1X,F10.3))
      IFIRST=1
      LFLAG=2
      IFLAG=2
      NEWPEN=0
      DO 100 IPNT=1,NPTS
      INUM=NIE(IELP)
      IF(INUM.EQ.0) GO TO 61
      DO 60 K=1,INUM
      KK=IE(K,IELP)
      IF(KK.LE.30) CALL HYDE(KK,SEG(1,IPNT),IFLAG)
      IF(KK.GT.30) CALL HIDE(KK,SEG(1,IPNT),IFLAG)
      IF(IFLAG.EQ.1) GO TO 61
   60 CONTINUE
   61 IF(K.GT.INUM .OR. K.EQ.1)  GO TO 62
      ITEMP = IE(1,IELP)
      IE(1,IELP) = IE(K,IELP)
      IE(K,IELP) = ITEMP
   62 IF(IFLAG .NE. LFLAG)  GO TO 200
   70 IF(IFLAG.EQ.1) GO TO 400
      LFLAG=2
      CALL TRANS1(SEG(1,IPNT),PPP)
      X=-PPP(1)*SFACTR/PPP(3)+OFSETX
      Y=PPP(2)*SFACTR/PPP(3)+OFSETY
      IF (X.GE.XMIN.AND.X.LE.XMAX.AND.
     1    Y.GE.YMIN.AND.Y.LE.YMAX)    GO TO 71
           CALL CLIP(X,Y,XSAV,YSAV,XMIN,XMAX,YMIN,YMAX,IPEN,IPLOT)
           NEWPEN=-2
           IPEN=3
           GO TO 75
71    CONTINUE
           IF (IPNT.NE.1)
     1     CALL PREPLT(X,Y,XSAV,YSAV,XMIN,XMAX,YMIN,YMAX,IPEN,NEWPEN)
           CALL PLOT(X,Y,IPEN)
           IPLOT=1
           NEWPEN=2
           IPEN=2
75    CONTINUE
           XSAV=X
           YSAV=Y
  100 CONTINUE
      RETURN
  200 IF(IPNT.EQ.1) GO TO 70
      DO 250 IJ=1,3
```

```
      PP(IJ,1)=SEG(IJ,IPNT-1)
  250 PP(IJ,2)=SEG(IJ,IPNT)
      CALL EXTEND(PP,IFLAG,LFLAG)
      DO 260 IJ=1,3
  260 P(IJ)=PP(IJ,IFLAG)
      CALL TRANS1(P,PPP)
      X=-PPP(1)*SFACTR/PPP(3)+OFSETX
      Y=PPP(2)*SFACTR/PPP(3)+OFSETY
      IF(LFLAG.EQ.1) GO TO 350
      IF (X.GE.XMIN.AND.X.LE.XMAX.AND.
     1    Y.GE.YMIN.AND.Y.LE.YMAX)    GO TO 261
          CALL CLIP(X,Y,XSAV,YSAV,XMIN,XMAX,YMIN,YMAX,ITWO,IPLOT)
          NEWPEN=-3
          GO TO 265
261   CONTINUE
          IF (IPNT.NE.1)
     1    CALL PREPLT(X,Y,XSAV,YSAV,XMIN,XMAX,YMIN,YMAX,IPEN,NEWPEN)
          CALL PLOT(X,Y,2)
          IPLOT=1
          NEWPEN=3
265   CONTINUE
          IPEN=3
          XSAV=X
          YSAV=Y
          LFLAG=1
          GO TO 100
350   CONTINUE
      IF (X.GE.XMIN.AND.X.LE.XMAX.AND.
     1    Y.GE.YMIN.AND.Y.LE.YMAX)    GO TO 351
          CALL CLIP(X,Y,XSAV,YSAV,XMIN,XMAX,YMIN,YMAX,ITHREE,IPLOT)
          NEWPEN=-2
          IPEN=3
          GO TO 355
351   CONTINUE
          IF (IPNT.NE.1)
     1    CALL PREPLT(X,Y,XSAV,YSAV,XMIN,XMAX,YMIN,YMAX,IPEN,NEWPEN)
          CALL PLOT(X,Y,3)
          IPLOT=1
          NEWPEN=2
          IPEN=2
355   CONTINUE
          XSAV=X
          YSAV=Y
          GO TO 70
  400 IPEN=3
      LFLAG=1
      GO TO 100
      END
```

```
      SUBROUTINE POLYD
C
C     POLYD GENERATES DIRECTION COSINE MATRICIES FOR THE POLYGONS.
C        THE X AXIS OF THE POLYGON COORDINATE SYSTEM IS THE
C        NORMAL VECTOR TO THE POLYGON SURFACE.
C       Y VECTOR IS ALIGNED WITH ONE OF THE POLYGON SIDES.
C
      COMMON/ELLIPSE/NSTEPS(90),IELP,A(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      COMMON/POLYGON/NPLANE,IFLAG,NPPP(90),P0(3,4,60),P(3,4,60),
     *CONVEC(2,4,90),POS(2,90),SIGN(90)
      DIMENSION INDEX(6),D1(810)
      EQUIVALENCE (D,D1)
      DATA INDEX/3,6,7,1,2,5/
      DO 100 L=1,NPLANE
      J=9*L+262
      DO 20 I=1,3
      D1(J+I+2)=P(I,2,L)-P(I,1,L)
   20 D1(J+I+5)=P(I,3,L)-P(I,1,L)
      CALL CROSS(D1(J+3),D1(J+6),D1(J))
      SUMD1=0.0
      SUMD2=0.0
      DO 30 I=1,3
      SUMD1=SUMD1+D1(J+I-1)**2
   30 SUMD2=SUMD2+D1(J+I+2)**2
      SUMD1=SQRT(SUMD1)
      SUMD2=SQRT(SUMD2)
      DO 40 I=1,3
      D1(J+I-1)=D1(J+I-1)/SUMD1
   40 D1(J+I+2)=D1(J+I+2)/SUMD2
      CALL CROSS(D1(J),D1(J+3),D1(J+6))
      DO 50 I=1,3
      TEMP=D1(INDEX(I)+J)
      D1(INDEX(I)+J)=D1(INDEX(I+3)+J)
   50 D1(INDEX(I+3)+J)=TEMP
  100 CONTINUE
      DO 600  J=1,NPLANE
      NUM = J + 30
      DO 600  K=1,3
  600 SEGLP(K,NUM) = P(K,1,J)
      RETURN
      END
```

```
      SUBROUTINE PREPLT(X,Y,XSAV,YSAV,XMIN,XMAX,YMIN,YMAX,IPEN,NEWPEN)
C*************************************************************
C
C  THIS SUBROUTINE PERFORMS NECESSARY PEN MOVES FOR 1ST PLOT
C  AFTER CLIPPING
C
C*************************************************************
      LOGICAL XOFF,YOFF
      IF (NEWPEN.NE.-2) RETURN
C
C  OUTSIDE X AND/OR Y BOUNDARIES???
C
      XOFF=.FALSE.
      YOFF=.FALSE.
      IF (X.LT.XMIN.OR.XSAV.LT.XMIN.OR.
     1    X.GT.XMAX.OR.XSAV.GT.XMAX)   XOFF=.TRUE.
      IF (Y.LT.YMIN.OR.YSAV.LT.YMIN.OR.
     1    Y.GT.YMAX.OR.YSAV.GT.YMAX)   YOFF=.TRUE.
C
C  DETERMINE IF OFF TOP,BOTTOM,RIGHT,OR LEFT SIDE OF PLOTTER
C
      IF (X.LT.XMIN.OR.XSAV.LT.XMIN) XLIMIT=XMIN
      IF (X.GT.XMAX.OR.XSAV.GT.XMAX) XLIMIT=XMAX
      IF (Y.LT.YMIN.OR.YSAV.LT.YMIN) YLIMIT=YMIN
      IF (Y.GT.YMAX.OR.YSAV.GT.YMAX) YLIMIT=YMAX
C
C  GET X AND Y POINTS DEPENDING ON WHAT BOUNDARIES OVER
C
      IF (XOFF) YTEMP=YINTCP(X,Y,XSAV,YSAV,XLIMIT)
      IF (.NOT.XOFF) YTEMP=YLIMIT
      IF (YOFF) XTEMP=XINTCP(X,Y,XSAV,YSAV,YLIMIT)
      IF (.NOT.YOFF) XTEMP=XLIMIT
C
C  MOVE PEN UP TO THAT SPOT
C
      CALL PLOT(XTEMP,YTEMP,3)
      IPEN=2
      RETURN
      END
```

```
      SUBROUTINE PRJELR
C    ********************************************
C
C     THIS SUBROUTINE PROJECTS ELLIPSOIDS ONTO THE PROJECTION PLANE.
C
C    ********************************************
      COMMON/ELLIPSE/NSTEPS(90),IELP,A(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      COMMON/POLYGON/NPLANE,IFLAG,NPPP(90),PO(3,4,60),P(3,4,60),
     *CONVEC(2,4,90),POS(2,90),SIGN(90)
       DIMENSION DD(3,3),DDD(3,3),SS(3),S(3)
      DIMENSION R(3,3),R2(2,3)
      REAL LAMDA1,LAMDA2,M1,M2
      IF(NSEG .EQ. 0)  RETURN
      DO 100  I=1,NSEG
      CALL DOTT(D(1,1,I),DVP,DD,3,3,3)
      CALL MAT(A(1,1,I),DD,DDD,3,3,3,3,3,3)
      CALL DOT(D(1,1,I),DDD,DD,3,3,3)
      CALL MAT(DVP,DD,DDD,3,3,3,3,3,3)
      DO 10  K=1,3
   10 SS(K) = SEGLP(K,I) - VP(K)
      CALL MAT(DVP,SS,S,3,3,1,3,3,3)
      DO 30 II=1,3
      IF (S(II).EQ.0.0) S(II)=1.0
30    CONTINUE
C
      CALL SOLVR(DDD(1,1),DDD(2,1),DDD(3,1),DDD(1,3),DDD(2,3),DDD(3,3),
     *DDD(1,1),DDD(1,3),S,R(1,1),R(3,1))
      CALL SOLVR(DDD(1,2),DDD(2,2),DDD(3,2),DDD(1,3),DDD(2,3),DDD(3,3),
     *DDD(2,2),DDD(2,3),S,R(2,2),R(3,2))
      CALL SOLVR(DDD(1,1)+DDD(1,2),DDD(2,1)+DDD(2,2),DDD(3,1)+DDD(3,2),
     *DDD(1,3),DDD(2,3),DDD(3,3),DDD(1,1)+2.0*DDD(1,2)+DDD(2,2),
     *DDD(1,3)+DDD(2,3),S,R(1,3),R(3,3))
      R(2,1)=0.0
      R(1,2)=0.0
      R(2,3)=R(1,3)
      DO 15 IK=1,3
      DO 15 IJ=1,2
   15 R2(IJ,IK)=(S(IJ)+R(IJ,IK))/(S(3)+R(3,IK))-S(IJ)/S(3)
      CALL SOLVA(R2,A11,A22,A12)
      TEMP=(A11+A22)**2-4.0*(A11*A22-A12**2)
      IF(TEMP.LT.0.0) TEMP=0.0
      TEMP=SQRT(TEMP)
      LAMDA1=(A11+A22+TEMP)/2.0
      LAMDA2=(A11+A22-TEMP)/2.0
      RX1=A12
      RY1=LAMDA1-A11
      RX2=LAMDA2-A22
      RY2=A12
      LAMDA1=ABS(LAMDA1)
      LAMDA2=ABS(LAMDA2)
      M1=SQRT(1.0/(LAMDA1*(RX1**2+RY1**2)))
      M2=SQRT(1.0/(LAMDA2*(RX2**2+RY2**2)))
      RX1=RX1*M1
      RX2=RX2*M2
      RY1=RY1*M1
      RY2=RY2*M2
      CONVEC(1,1,I)=-2.0*RX1
      CONVEC(2,1,I)=-2.0*RY1
      CONVEC(1,2,I)=-2.0*RX2
      CONVEC(2,2,I)=-2.0*RY2
      DO 20  IJ=1,2
```

```
      CONVEC(IJ,3,I) = -CONVEC(IJ,1,I)
      CONVEC(IJ,4,I) = -CONVEC(IJ,2,I)
   20 POS(IJ,I) = CONVEC(IJ,3,I)/2.0 + CONVEC(IJ,4,I)/2.0+S(IJ)/S(3)
      NPPP(I)=4
      SIGN(I)=CONVEC(1,1,I)*CONVEC(2,2,I)-
     1CONVEC(2,1,I)*CONVEC(1,2,I)
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PRJPLY
C
C         ****************************************
C
C         THIS SUBROUTINE WILL SETUP THE CONVEC ARRAY.
C         IT ALSO SETS UP THE POS AND SIGN ARRAYS.
C
C           ARRAY P IS THE ORIGINAL POSITION VECTORS FOR THE POLYGONS
C            IN THE INERTIAL REFERENCE SYSTEM.
C           ARRAY CONVEC WILL CONTAIN THE CONTOUR VECTORS
C            FOR PROJECTED POLYGONS.
C           ARRAY POS WILL CONTAIN POSITION VECTORS FROM THE
C            PROJECTION PLANE ORIGIN TO POLYGON POINT # 1.
C           ARRAY SIGN WILL CONTAIN THE SIGN THAT RESULTS FROM
C            THE CROSS PRODUCT (P2-P1)X(P3-P2).
C
C         ****************************************
      COMMON/POLYGON/NPLANE,IFLAG,NPPP(90),PO(3,4,60),P(3,4,60),
     *CONVEC(2,4,90),POS(2,90),SIGN(90)
      COMMON/ELLIPSE/NSTEPS(90),IELP,A(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      DIMENSION PPP2(3),PP2(3),PP1(3)
      IF(NPLANE.EQ.0) RETURN
      DO 40 I=1,NPLANE
      II=I+30
      NPTS=NPPP(II)
      DO 35 K=1,NPTS
      DO 10 J=1,3
      PPP2(J)=P(J,K,I)-VP(J)
   10 CONTINUE
      CALL MAT(DVP,PPP2,PP2,3,3,1,3,3,3)
      IF(K.NE.1) GO TO 16
      DO 15 J=1,2
      POS(J,II)=PP2(J)/PP2(3)
   15 CONTINUE
      GO TO 25
   16 DO 20 J=1,2
   20 CONVEC(J,K-1,II)=PP2(J)/PP2(3)-PP1(J)/PP1(3)
   25 DO 30 J=1,3
   30 PP1(J)=PP2(J)
      IF(K.EQ.3) SIGN(II)=CONVEC(1,1,II)*CONVEC(2,2,II)-
     1CONVEC(2,1,II)*CONVEC(1,2,II)
   35 CONTINUE
      DO 40 J=1,2
      CONVEC(J,NPTS,II)=POS(J,II)-PP2(J)/PP2(3)
   40 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PSE(X1,IN,SEG,INDEX,INDEX2,IHALF)
      DIMENSION X1(3,INDEX,INDEX),IN(INDEX),SEG(3,3333)
C
C        THIS SUBROUTINE PLOTS A SEMIELLIPSOID.
C       THE HALF OF THE ELLIPSOID PLOTTED DEPENDS UPON IHALF.
C          IF IHALF = 1 X .GE. 0 IS PLOTTED.
C          IF IHALF = 2 X .LT. 0 IS PLOTTED.
      DO 100 I=IHALF,INDEX
      LINE=INDEX-I+1
      IF(IHALF.EQ.2) LINE=I
      NPTS=IN(LINE)
      DO 50 K=1,NPTS
      DO 50 J=1,3
      SEG(J,K)=X1(J,K,LINE)
      IF(IHALF.EQ.2.AND.J.EQ.1) SEG(J,K)=-SEG(J,K)
   50 CONTINUE
      N=NPTS
      IF(LINE.EQ.INDEX) GO TO 71
      NPT=NPTS-1
      DO 60 K=1,NPT
      KK=NPT-K+1
      N=N+1
      SEG(1,N)=SEG(1,KK)
      SEG(2,N)=SEG(2,KK)
   60 SEG(3,N)=-SEG(3,KK)
      NPT=N-1
      DO 70 K=1,NPT
      KK=NPT-K+1
      N=N+1
      SEG(1,N)=SEG(1,KK)
      SEG(2,N)=-SEG(2,KK)
   70 SEG(3,N)=SEG(3,KK)
   71 IPEN=3
      CALL PNTPLT(SEG,IPEN,INDEX2,N)
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ROT(A,L,TH,M)
C                                                          REV 01 06/10/72
C     COMPUTES ROTATION MATRIX A FOR ANGLE TH
C     ABOUT X,Y OR Z AXIS AS L = 1,2, OR 3.
C
C       ARGUMENTS:
C         A: 3X3 ROTATION MATRIX TO BE COMPUTED.
C         L: 1,2 OR 3 TO ROTATE ABOUT X,Y OR Z AXIS.
C         TH: ANGLE OF ROTATION IN RADIANS.
C         M: 1ST DIMENSION OF A IN CALLING PROGRAM.
C
      DIMENSION A(M,3)
      C= COS(TH)
      S= SIN(TH)
      IF (L.EQ.2) S = -S
      DO 30 I=1,3
      IF(I.EQ.3)GO TO 20
      DO 10 J=I,2
      A(I,J+1)=0.0
      A(J+1,I)=0.0
      IF(I+J+L.NE.5)GO TO 10
      A(I,J+1)=S
      A(J+1,I)=-S
   10 CONTINUE
   20 A(I,I)= C
      IF(I.EQ.L)A(I,I)=1.0
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SOLVA(R,AA11,AA22,AA12)
      DIMENSION R(2,3)
      A11=R(1,1)**2
      A12=2.0*R(2,1)*R(1,1)
      A13=R(2,1)**2
      A21=R(1,2)**2
      A22=2.0*R(2,2)*R(1,2)
      A23=R(2,2)**2
      A31=R(1,3)**2
      A32=2.0*R(2,3)*R(1,3)
      A33=R(2,3)**2
      DEL=A11*(A22*A33-A23*A32)-A12*(A21*A33-A23*A31)+
     1A13*(A21*A32-A22*A31)
      AA11=((A22-A12)*(A33-A23)-(A23-A13)*(A32-A22))/DEL
      AA12=((A23-A13)*(A31-A21)-(A21-A11)*(A33-A23))/DEL
      AA22=((A21-A11)*(A32-A22)-(A22-A12)*(A31-A21))/DEL
      RETURN
      END
```

```
      SUBROUTINE SOLVR(A1,A2,A3,A4,A5,A6,A7,A8,P,RX,RZ)
C
C        ******************************************
C
C        THIS SUBROUTINE WILL SOLVE A SET OF SIMULTANEOUS EQUATIONS
C      TO FIND COMPONETS OF VECTOR R THAT SATISFY THE PROPERTIES NEEDED
C     TO DETURMINE THE EQUATION OF THE PROJECTED ELLIPSE.
C
C              SEE WRITEUP.
C
C        ******************************************
      DIMENSION P(3)
      B=A1*P(1)+A2*P(2)+A3*P(3)
      D=A4*P(1)+A5*P(2)+A6*P(3)
      T1=A7*(D/B)**2+A6-2.0*A8*D/B
      T2=2.0*A7*D/(B)**2-2.0*A8/B
      T3=A7*(1/B)**2-1
        RZ=(-T2+SQRT(T2**2-4.0*T1*T3))/(2.0*T1)
      RX=-D*RZ/B-1.0/B
      RETURN
      END
```

```
      SUBROUTINE TITLE
      DIMENSION ID(10,20),ICOLOR(21)
      COMMON/DBUG/IDEBUG(30),NISG,DEVFLG,ONLINE,TERM,BDRS,OFLINE
      INTEGER ONLINE,DEVFLG,TERM,BDRS,OFLINE
      NLINE=20
      SIZE=.335
      X=1.375
      Y=10.0
C
C     INITIALIZE PLOTTING PACKAGE
C
      CALL PLOT(0.0,0.0,-3)
      READ(5,1) NFRME
 1    FORMAT(I2)
      IF(NFRME.EQ.0) RETURN
      DO 300 K=1,NFRME
      DO 50 I=1,NLINE
      READ(5,200) (ID(J,I),J=1,8),ICOLOR(I)
      WRITE(6,200) (ID(J,I),J=1,8),ICOLOR(I)
200   FORMAT(7A4,A2,I2)
 50   CONTINUE
      DO 100 I=1,NLINE
      Y=Y-.5
      IF(DEVFLG.EQ.OFLINE.OR.DEVFLG.EQ.BDRS) CALL NEWPEN(ICOLOR(I))
      CALL SYMBOL(X,Y,SIZE,ID(1,I),0.,30)
100   CONTINUE
      IF(DEVFLG.EQ.ONLINE) CALL PLOT(12.,0.,-3)
      IF(DEVFLG.EQ.TERM) CALL PLOT(0.0,0.0,-3)
      IF(DEVFLG.EQ.BDRS) CALL NFRAME
      IF(DEVFLG.EQ.OFLINE) CALL PLOT (14.,0.,-3)
300   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TPOINT(PP2,I,IN)
      DIMENSION PP2(3),R(3),PP1(3)
      COMMON/POLYGON/NPLANE,IFLAG,NPPP(90),PO(3,4,60),P(3,4,60),
     *CONVEC(2,4,90),POS(2,90),SIGN(90)
C
C       THIS SUBROUTINE TESTS A POINT ON THE PROJECTION PLANE
C      DEFINED BY PP2 AGAINST A POLYGON ON THE PROJECTION PLANE
C      DEFINED BY I. A FLAG 'IN' IS RETURNED TO INDICATE IF THE
C      POINT WAS INSIDE OR OUTSIDE THE POLYGON.
C
C             IN = 1 POINT WAS INSIDE POLYGON
C
C             IN = 2 POINT WAS OUTSIDE POLYGON
C
      IN=1
      NPTS1=NPPP(I)
      DO 20 JJ=1,2
   20 PP1(JJ)=POS(JJ,I)
      DO 100 L=1,NPTS1
      DO 30 N=1,2
   30 R(N)=PP2(N)-PP1(N)
      SIGN2=CONVEC(1,L,I)*R(2)-CONVEC(2,L,I)*R(1)
      IF(SIGN2*SIGN(I) .LT. 0.)  GO TO 150
      IF(ABS(SIGN2*SIGN(I)).LT.1.E-11) GO TO 150
      DO 100 N=1,2
  100 PP1(N)=PP1(N)+CONVEC(N,L,I)
      RETURN
  150 IN=2
      RETURN
      END
```

```
      SUBROUTINE TRANS1(R,P)
      COMMON/ELLIPSE/NSTEPS(90),IELP,A(3,3,30),SEGLP(3,90),VP(3),
     *D(3,3,90),DVP(3,3),RA(3),NSEG
      COMMON/VIEWP/VP0(3),DVP0(3,3),IVP,VP2(3)
      DIMENSION DD(3,3),P(3),R(3),R2(3),SEGLP2(3)
      IF(IELP .GT. 30)  GO TO 10
      CALL DOTT(DVP,D(1,1,IELP),DD,3,3,3)
   20 CONTINUE
      CALL MAT(DD,R,R2,3,3,1,3,3,3)
      CALL MAT(DVP,SEGLP(1,IELP),SEGLP2,3,3,1,3,3,3)
      DO 1  I=1,3
    1 P(I)=SEGLP2(I)+R2(I)-VP2(I)
      RETURN
   10 DO 11  I=1,3
      DO 11  J=1,3
   11 DD(I,J) = DVP(I,J)
      GO TO 20
      END
```

```
      REAL FUNCTION XINTCP(X,Y,XSAV,YSAV,YTEMP)
C    ********************************************
C
C  THIS FUNCTION CALCULATES THE X INTERCEPT AT YTEMP
C
C    ********************************************
      X1=X-XSAV
      Y1=Y-YSAV
      IF (Y1.NE.0.0) PFACTR=X1/Y1
      IF (Y1.EQ.0.0) PFACTR=0.0
      Y2=YTEMP-YSAV
      XINTCP=Y2*PFACTR+XSAV
      RETURN
      END
```

```
      SUBROUTINE XYZ(MU,A,B,C,S,M,JFLAG)
      DIMENSION MU(3),S(3),M(3,2)
      REAL M,MU
C
C    ******************************************************************
C
C        ALPHA AND BETA RELATE THE Y AND Z COMPONENTS OF M TO
C         THE X COMPONENT OF M.
C    ******************************************************************
C
      ALPHA = MU(2) / MU(1)
      BETA = MU(3) / MU(1)
C
C    ******************************************************************
C
C        SOLVE THE EQUATION FOR ELLIPSE 'N' TO FIND A POINT ON
C         THE ELLIPSE THAT WILL DETERMINE A VECTOR M.
C
C    ******************************************************************
C
      T1=A+B*ALPHA*ALPHA+C*BETA*BETA
      T2=2*A*S(1)+2*B*ALPHA*S(2)+2*C*BETA*S(3)
      T3=A*S(1)**2+B*S(2)**2+C*S(3)**2-1
      TEMP= T2 * T2 - 4 * T1 * T3
C
C    ******************************************************************
C
C        NO SOLUTION MEANS ELLIPSOID 'N' NOT TOUCHED BY LINE OF
C         SIGHT RAY.
C
C    ******************************************************************
C
      IF(TEMP.LT.0.0) GO TO 2
      TEMP=SQRT(TEMP)
C
C    ******************************************************************
C
C        FIND TWO POSSIBLE M VECTORS BECAUSE A RAY ENTERING A
C         SOLID MUST ALSO LEAVE.
C
C    ******************************************************************
C
      M(1,1)=(T2+TEMP)/(2*T1)
      M(2,1)=ALPHA*M(1,1)
      M(3,1)=BETA*M(1,1)
      M(1,2)=(T2-TEMP)/(2*T1)
      M(2,2)=ALPHA*M(1,2)
      M(3,2)=BETA*M(1,2)
      JFLAG=0
      RETURN
    2 JFLAG=1
      RETURN
      END
```

```
      REAL FUNCTION YINTCP(X,Y,XSAV,YSAV,XTEMP)
C    ***************************************************
C
C    THIS FUNCTION CALCULATES THE Y INTERCEPT AT XTEMP
C
C    ***************************************************
      X1=X-XSAV
      Y1=Y-YSAV
      IF (X1.NE.0.0) PFACTR=Y1/X1
      IF (X1.EQ.0.0) PFACTR=0.0
      X2=XTEMP-YSAV
      YINTCP=X2*PFACTR+YSAV
      RETURN
      END
```

```
      SUBROUTINE YZ(MU,A,B,C,S,M,JFLAG)
      DIMENSION MU(3),S(3),M(3,2)
      REAL MU,M
C         ******************************************
C
C         ALPHA RELATES THE Z COMPONET TO THE Y COMPONET
C
C         ******************************************
      ALPHA=MU(3)/MU(2)
C         ******************************************
C
C         SOLVE THE EQUATION FOR ELLIPSE 'N' WHEN X=0 TO
C       FIND A POINT ON THE ELLIPSE THAT WILL DETERMINE A VECTOR M.
C
C         ******************************************
      T1=B+C*ALPHA*ALPHA
      T2=2*B*S(2)+2*C*ALPHA*S(3)
      T3=A*S(1)**2+B*S(2)**2+C*S(3)**2-1
      TEMP=T2*T2-4*T1*T3
C         ******************************************
C
C         NO SOLUTION MEANS ELLIPSOID 'N' NOT TOUCHED BY LINE
C       OF SIGHT RAY.
C
C         ******************************************
      IF(TEMP.LT.0.0) GO TO 2
      TEMP=SQRT(TEMP)
C         ******************************************
C
C         FIND TWO POSSIBLE M VECTORS BECAUSE A RAY ENTERING
C       A SOLID MUST ALSO LEAVE.
C
C         ******************************************
      M(2,1)=(T2+TEMP)/(2*T1)
      M(1,1)=0.0
      M(3,1)=ALPHA*M(2,1)
      M(2,2)=(T2-TEMP)/(2*T1)
      M(1,2)=0.0
      M(3,2)=ALPHA*M(2,2)
      JFLAG=0
      RETURN
    2 JFLAG=1
      RETURN
      END
```

```
      SUBROUTINE Z(MU,A,B,C,S,M,JFLAG)
      DIMENSION MU(3),S(3),M(3,2)
      REAL M,MU
C        *****************************************
C
C        SOLVE THE EQUATION FOR ELLIPSE 'N' WHEN X=0 AND
C      Y=0 TO FIND A POINT ON THE ELLIPSE THAT
C      WILL DETERMINE A VECTOR M.
C
C        *****************************************
      T1=C
      T2=2*C*S(3)
      T3=A*S(1)**2+B*S(2)**2+C*S(3)**2-1
      TEMP=T2*T2-4*T1*T3
C        *****************************************
C
C        NO SOLUTION MEANS ELLIPSOID 'N' NOT TOUCHED BY LINE
C      OF SIGHT RAY.
C
C        *****************************************
      IF(TEMP.LT.0.0) GO TO 2
      TEMP=SQRT(TEMP)
C        *****************************************
C
C        FIND TWO POSSIBLE M VECTORS BECAUSE A RAY ENTERING
C      A SOLID MUST ALSO LEAVE.
C
C        *****************************************
      M(1,1)=0.0
      M(2,1)=0.0
      M(3,1)=(T2+TEMP)/(2*T1)
      M(1,2)=0.0
      M(2,2)=0.0
      M(3,2)=(T2-TEMP)/(2*T1)
      JFLAG=0
      RETURN
    2 JFLAG=1
      RETURN
      END
```

APPENDIX A

HIDDEN LINE PROBLEM BETWEEN TWO ELLIPSOIDS

Ellipsoids are plotted as a set of contour lines. These lines consist of a series of short vectors that are sequentially plotted to form a contour line. The hidden line problem can be reduced to the problem of finding what vectors are hidden from the viewpoint. This problem can be broken down further if it is assumed that each vector is short enough that only the point representing the head of the vector needs to be checked; thus the problem reduces to checking points to see if they are hidden from the viewpoint. Figure A.1 shows the coordinate systems and vectors used to solve the hidden point problem. The following equations are used to solve the hidden point problem using ellipsoids.

$$\overrightarrow{P_{21}} = [D_1]\ \overrightarrow{P_2}$$

$$\overrightarrow{P_{11}} = [D_1]\ \overrightarrow{P_1}$$

$$\overrightarrow{r_{21}} = [D_1]\ [D_2]^T\ \overrightarrow{r_2}$$

$$\vec{S} - \overrightarrow{r_{21}} - \overrightarrow{P_{21}} + \overrightarrow{P_{11}} = 0$$

$$\vec{S} = \overrightarrow{P_{21}} + \overrightarrow{r_{21}} - \overrightarrow{P_{11}}$$

$$\vec{S}_{\text{in ellipsoid No. 1 system}} = [D_1]\ (\overrightarrow{P_2}) + [D_1]\ [D_2]^T\ \overrightarrow{r_2} - [D_1]\ \overrightarrow{P_1}$$

$$\overrightarrow{VP_1} = [D_1]\ \overrightarrow{VP}$$

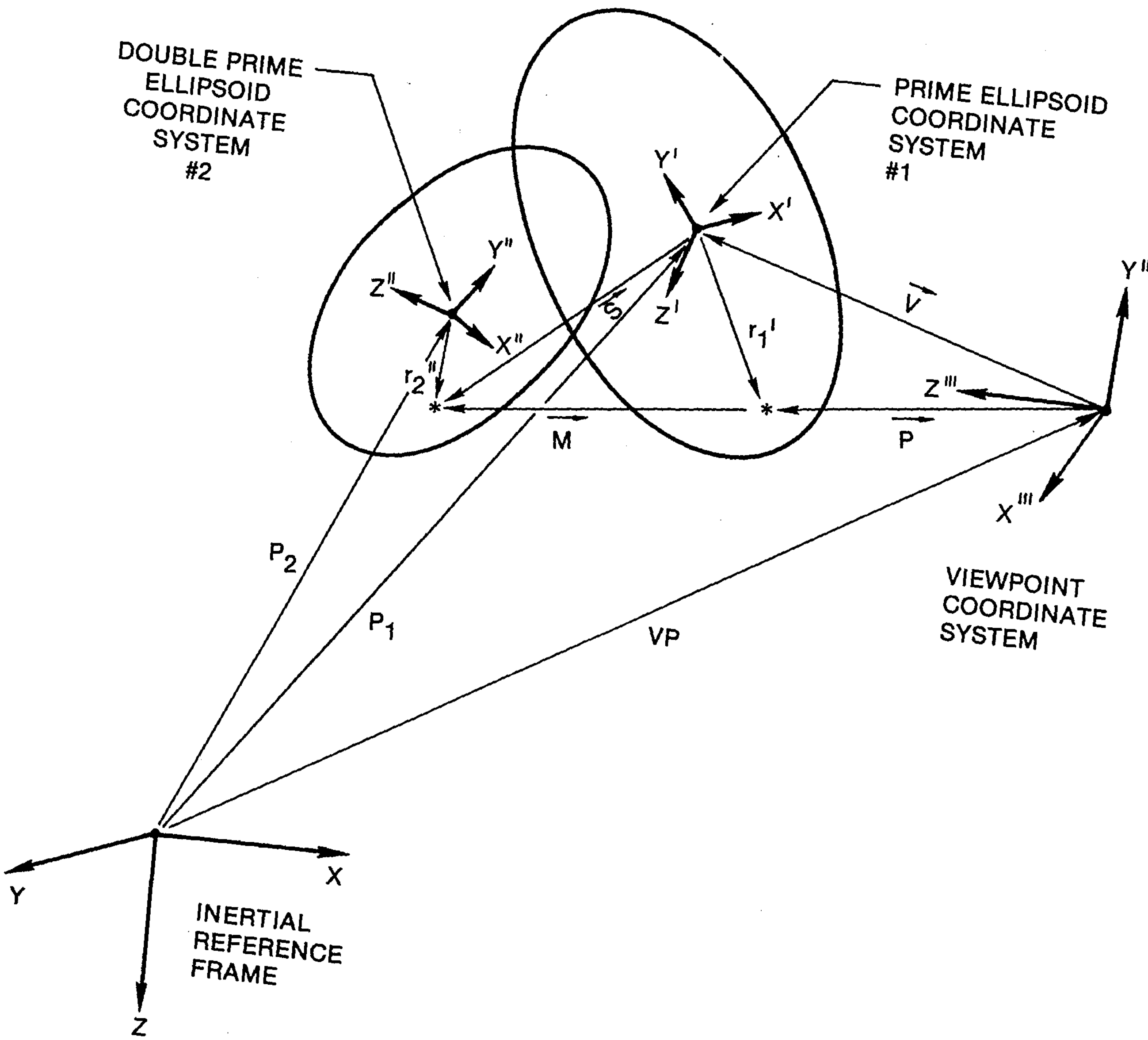


Figure A.1 Coordinate Systems and Vectors Used to Solve the Hidden Line Problem

$$\vec{V} - \overrightarrow{VP_1} + \overrightarrow{P_{11}} = 0$$

$$\vec{V} = \overrightarrow{VP_1} - \overrightarrow{P_{11}}$$

$$\vec{V}_{\text{in ellipsoid No. 1 system}} = [D_1]\,\overrightarrow{VP} - [D_1]\,\overrightarrow{P_1}$$

$$\vec{P} + \vec{M} = \vec{S} - \vec{V} \qquad \overrightarrow{r_1} = \vec{S} - \vec{M} \qquad \overrightarrow{r_1} = \vec{V} + \vec{P}$$

using the ellipsoid equation

$$\vec{r} \cdot [A]\,\vec{r} = 1$$

$$\vec{r}^{\,T}\,[A]\,\vec{r} = 1$$

$$(\vec{S} - \vec{M})^T\,[A]\,(\vec{S} - \vec{M}) = 1 \qquad A = \begin{bmatrix} \frac{1}{a^2} & 0 & 0 \\ 0 & \frac{1}{b^2} & 0 \\ 0 & 0 & \frac{1}{c^2} \end{bmatrix}$$

$$S = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} \qquad M = \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix}$$

Since $\vec{P}$ and $\vec{M}$ are in the same direction, $\vec{P} + \vec{M}$ is also in in the same direction, then a unit vector in the direction of M can be obtained from

$$\vec{P} + \vec{M} = \vec{S} - \vec{V}$$

therefore

$$\begin{pmatrix} \dfrac{S_1 - V_1}{MAG} \\ \\ \dfrac{S_2 - V_2}{MAG} \\ \\ \dfrac{S_3 - V_3}{MAG} \end{pmatrix} = \hat{\mu} = \begin{pmatrix} \mu_1 \\ \\ \mu_2 \\ \\ \mu_3 \end{pmatrix}$$

where

$$MAG = [(S_1 - V_1)^2 + (S_2 - V_2)^2 + (S_3 - V_3)^2]^{1/2}$$

giving $\hat{\mu}$ in the direction of $\vec{M}$ must obey the following relationship.

$$\frac{\mu_1}{\mu_2} = \frac{M_1}{M_2} \qquad \frac{\mu_1}{\mu_3} = \frac{M_1}{M_3}$$

$$M_2 = M_1 \frac{\mu_2}{\mu_1} = M_{1_\alpha} \qquad M_3 = M_1 \frac{\mu_3}{\mu_1} = M_{1_\beta}$$

back to

$$(\vec{S} - \vec{M})\ [A]\ (\vec{S} - \vec{M}) = 1$$

$$S - M = \begin{pmatrix} S_1 = M_1 \\ S_2 = \alpha M_1 \\ S_3 - \beta M_1 \end{pmatrix}$$

therefore

$$(\vec{S} - \vec{M})^T \; [A] \; (\vec{S} - \vec{M}) = 1$$

expands to

$$\frac{1}{a^2}(S_1 - M_1)^2 + \frac{1}{b^2}(S_2 - \alpha M_1)^2 + \frac{1}{c^2}(S_3 - \beta M_1)^2 = 1$$

let

$$A = \frac{1}{a^2} \qquad B = \frac{1}{b^2} \qquad C = \frac{1}{c^2}$$

$$A(S_1 - M_1)^2 + B(S_2 - \alpha M_1)^2 + C(S_3 - \beta M_1)^2 = 1$$

$$AS_1^2 + AM_1^2 - 2AS_1M_1 + BS_2^2 + B\alpha^2M_1^2 = 2BS_2\alpha M_1 + CS_3^2 +$$
$$C\beta^2M_1^2 - 2C\beta S_3M_1 = 1$$

$$(A + B\alpha^2 + C\beta^2)\,M_1^2 - (2AS_1 + 2BS_2 + 2C\beta S_3)\,M_1 + AS_1^2 + BS_2^2 +$$
$$CS_3^2 - 1 = 0$$

let

$$T_1 = A + B\alpha^2 + C\beta^2$$

$$T_2 = - (2AS_1 + 2BS_2\alpha + 2C\beta S_3)$$

$$T_3 = AS_22 + BS_2^2 + CS_3^2 -1$$

$$M_1 = \frac{-T_2 \pm \sqrt{T_2^2 - 4T_1T_3}}{2T_1}$$

only $T_2^2 - 4T_1T_3$ must be evaluated to test for M_1 on the ellipsoid.

If M_1 is real, then $\vec{M}$ is given by

$$\vec{M} = \begin{pmatrix} M_1 \\ \alpha M_1 \\ \beta M_1 \end{pmatrix}$$

$$\vec{P} = \vec{S} - \vec{V} - \vec{M}$$

If there is a real $\vec{M}$, then it must be determined if both $\vec{M}$s are in the opposite direction of $\vec{P}$.

Case No. 1--if both $\vec{M}$s are in the opposite direction of $\vec{P}$, then the point is not hidden.

Case No. 2--if either $\vec{M}$ is in the same direction as $\vec{P}$, then the point is hidden.

Using the dot product to determine the relative directions of $\vec{M}$ and $\vec{P}$ gives

$$\vec{P} \cdot \vec{M} = < 0 \text{ , the point is not hidden.}$$

$$\vec{P} = \vec{S} - \vec{V} - \vec{M}$$

$$(S_1 - V_1 - M_1, S_2 - V_2 - M_2, S_3 - V_3 - M_3) \qquad \begin{pmatrix} M_1 \\ M_2 \\ M_3 \end{pmatrix}$$

$$\begin{aligned} \vec{P} \cdot \vec{M} = {} & S_1M_1 - V_1M_1 - M_1M_1 \\ & + S_2M_2 - V_2M_2 - M_2M_2 \\ & + S_3M_3 - V_3M_3 - M_3M_3 \end{aligned}$$

If $|\vec{M}|$ is close to zero, the points on each ellipsoid are very close together, and it is not necessary to hide the point on the contour vector.

If μ_1 equals zero, then an α and β cannot be found with the given expressions. Therefore, the following cases must be examined.

Case No. 1

$$\mu_1 = 0 \qquad \mu_2 = 0$$

$$M_1 = 0$$

$$M_2 = 0$$

$$(\vec{S} - \vec{M})^T A(\vec{S} - \vec{M}) = 1$$

$$\vec{S} - \vec{M} = \begin{pmatrix} S_1 \\ S_2 \\ S^3 - M^3 \end{pmatrix}$$

$$AS_1{}^2 + BS_2{}^2 + C\ (S_3 - M_3)^2 = 1$$

$$AS_1{}^2 + BS_2{}^2 + CS_3{}^2 + CM_3{}^2 - 2SC_3M_3 = 1$$

$$T1 = C$$

$$T2 = -2CS_3$$

$$T3 = AS_1{}^2 + BS_2{}^2 + CS_3{}^2 - 1$$

Use these T values back in the quadratic formula used earlier.

Case No. 2

$\mu_1 = 0 \qquad \mu_2 \neq 0$

$$M_1 = 0$$

$$\vec{S} - \vec{M} = \begin{pmatrix} S_1 \\ S_2 - M_2 \\ S_3 - M_3 \end{pmatrix} \qquad \frac{M_3}{M_2} = \frac{\mu_3}{\mu_2} \qquad \alpha = \frac{\mu_3}{\mu_2}$$

$$M_3 = \alpha M_2$$

$$AS_1{}^2 + BS_2{}^2 + BM_2{}^2 = 2BS_2M_2 + CS_3{}^2 + C\alpha^2M_2{}^2 - 2C\alpha M_2S_2 = 1$$

$$T1 = B + C\alpha^2$$

$$T2 = 2BS_2 - 2C\alpha S_3$$

$$T3 = AS_1^2 + BS_2^2 + CS_3^2 - 1$$

Use these T values in the quadratic formula and proceed.

APPENDIX B
DISCUSSION OF EQUATIONS USED BY PRJELR

PRJELR stands for project ellipsoid routine. The function of PRJELR is to circumscribe a projected shadow of an ellipsoid with a rectangle. The resulting rectangle is used as a polygon by the overlap routines to determine what objects overlap after they are projected.

GENERAL APPROACH

1. Assume that all ellipsoids project an elliptical shadow. This is a good assumption when the viewpoint is far away from all objects and the viewpoint Z axis is nearly directly pointed at all ellipsoids.

2. Find three radial vectors of the ellipsoid pointing to a surface point that forms the contour of the projected shadow. Add three further conditions--one radial vector is in the X-Z plane of the viewpoint coordinate system, another radial vector is in the Y-Z plane of the viewpoint coordinate system, and one radial vector is in the X = Y and Z plane of the viewpoint coordinate system. These planes are defined as if the viewpoint coordinate system were at the center of the ellipse. This gives three radial vectors as indicated in Figure B.1, $\vec{r_1}$, $\vec{r_2}$, $\vec{r_3}$.

3. Project all three vectors onto the projection plane and solve for an ellipse matrix.

4. Circumscribe the resulting ellipse with a rectangle.

Take [A(3,3)] ellipsoid matrix and transform to [A'(3,3)] in the viewpoint coordinate system.

$$A' = [DVP]\ [D]^T\ [A]\ [D]\ [DVP]^T$$

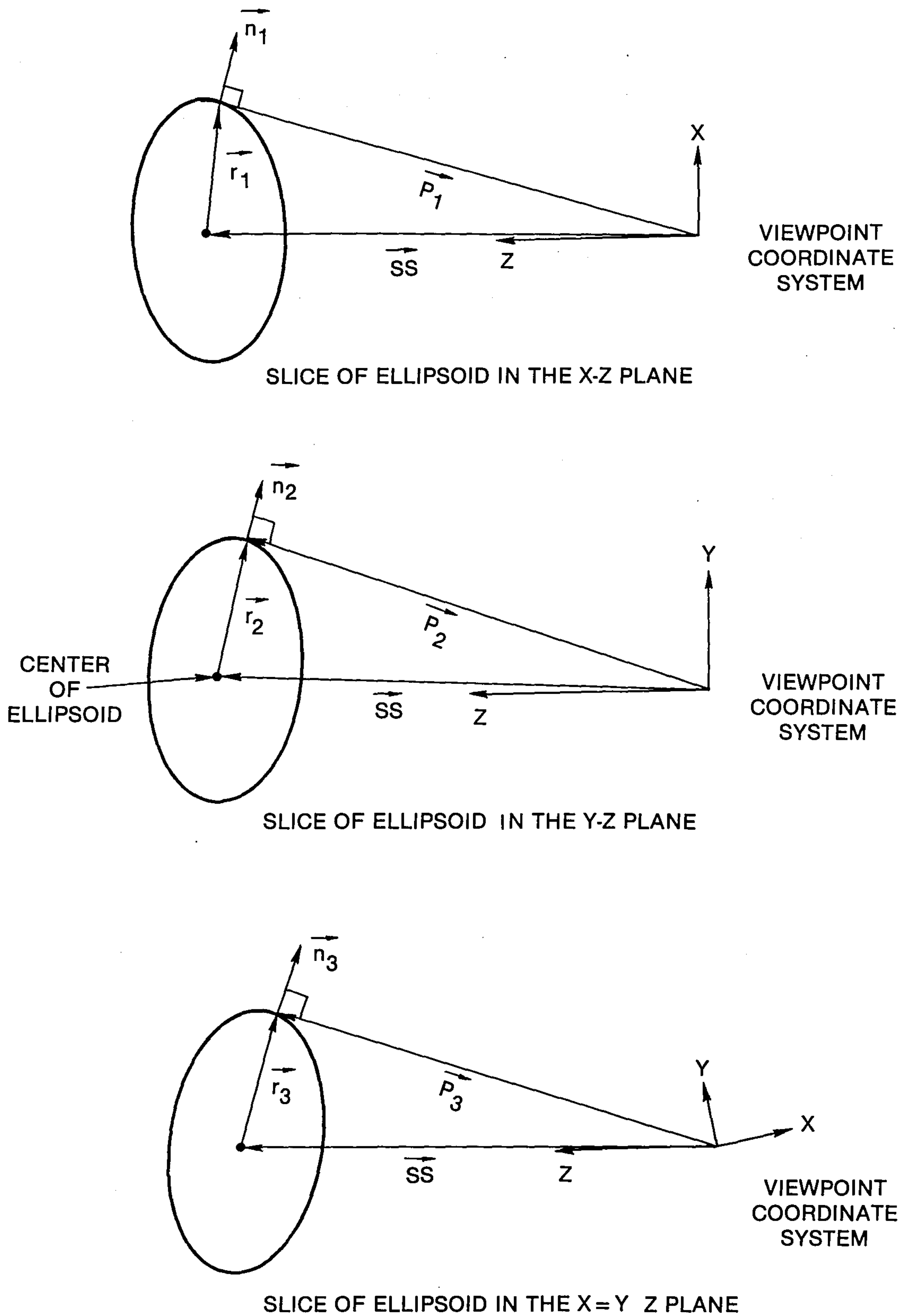


Figure B.1. Three Radial Vectors

where

[DVP] = direction cosine matrix that transforms from the inertial to the viewpoint frame of reference.

and

[D] = direction cosine matrix that transforms from the inertial to the ellipsoid frame of reference.

Find the following three vectors

$$\vec{r} = \begin{pmatrix} r_x \\ 0 \\ r_z \end{pmatrix} \quad \vec{r} = \begin{pmatrix} 0 \\ r_y \\ r_z \end{pmatrix} \quad \vec{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} \quad r_x = r_y$$

that also have the properties of being radial vectors defined by $\underline{A}'$ and that the associated vector $\vec{P}$ from the viewpoint to the tip of $\vec{r}$ is normal to the normal vector for the point defined by $\vec{r}$ on the ellipsoid. See Figure 9.

$$\vec{n} \cdot \vec{P} = 0 \qquad r^T\underline{A}'r = 1$$

$$\vec{n} = \underline{A}'r$$

$$\underline{A}'\vec{r} \cdot \vec{P} = \vec{P} \cdot \underline{A}'\vec{r} = 0 \qquad \vec{P} = \overrightarrow{SS} + \vec{r}$$

$$(\overrightarrow{SS} + \vec{r})^T\underline{A}'\vec{r} = 0$$

$$\vec{SS}^T \underline{A}' \vec{r} + \vec{r}^T \underline{A}' \vec{r} = 0 \qquad \vec{r}^T \underline{A}' \vec{r} = 1$$

$$\vec{SS}^T \underline{A}' \vec{r} = -1$$

This is the equation that subroutine SOLVR uses to return components for $\vec{r}$ that represent $\vec{r_1}$, $\vec{r_2}$, $\vec{r_3}$.

Let

Case No. 1 be $\vec{r_1}$

Case No. 2 be $\vec{r_2}$

Case No. 3 be $\vec{r_3}$

in all three cases

$$(SS_X, SS_Y, SS_Z) \begin{pmatrix} A'_{11} & A'_{12} & A'_{13} \\ A'_{21} & A'_{22} & A'_{23} \\ A'_{31} & A'_{32} & A'_{33} \end{pmatrix} r = -1$$

expanding the left hand part gives

$$(SS_X A'_{11} + SS_Y A'_{21} + SS_Z A'_{31}, SS_X A'_{12} + SS_Y A'_{22} + SS_Z A'_{32}$$

$$SS_X A'_{13} + SS_Y A'_{23} + SS_Z A'_{33}) \quad r = -1$$

With each case of r the expression reduces to

$$(\alpha_1 SS_X + \alpha_2 SS_Y + \alpha_3 SS_Z)\, r_{X/Y} + (\alpha_4 SS_X + \alpha_5 SS_Y + \alpha_6 SS_Z)\, r_Z = -1$$

$\alpha_4 = A'_{13}$

$\alpha_5 = A'_{23}$ true for all cases

$\alpha_6 = A'_{33}$

Case No. 1	Case No. 2	Case No. 3
$\alpha_1 = A'_{11}$	$\alpha_1 = A'_{12}$	$\alpha_1 = A'_{11} + A'_{12}$
$\alpha_2 = A'_{21}$	$\alpha_2 = A'_{22}$	$\alpha_2 = A'_{21} + A'_{22}$
$\alpha_3 = A'_{31}$	$\alpha_3 = A'_{32}$	$\alpha_3 = A'_{31} + A'_{32}$

Making another substitution, let

$$\beta_1 = (\alpha_1 SS_X + \alpha_2 SS_Y + \alpha_3 SS_Z)$$

$$\beta_2 = (\alpha_4 SS_X + \alpha_5 SS_Y + \alpha_6 SS_Z)$$

Then $r_{X/Y}$ can be solved for by the following equation

$$r_{X/Y} = -\frac{\beta_2}{\beta_1} r_Z - \frac{1}{\beta_1}$$

now $\vec{r_1}$, $\vec{r_2}$, and $\vec{r_3}$ is written

$$\vec{r_1} = \begin{pmatrix} -\frac{\beta_2}{\beta_1} r_Z - \frac{1}{\beta_1} \\ 0 \\ r_Z \end{pmatrix}$$

$$\vec{r_2} = \begin{pmatrix} 0 \\ -\frac{\beta_2}{\beta_1} r_Z - \frac{1}{\beta_1} \\ r_Z \\ -\frac{\beta_2}{\beta_1} r_Z - \frac{1}{\beta_1} \end{pmatrix}$$

$$\vec{r_3} = \begin{pmatrix} -\frac{\beta_2}{\beta_1} r_Z - \frac{1}{\beta_1} \\ r_Z \end{pmatrix}$$

Since

$$\vec{r}^T \underline{A}' \vec{r} = 1$$

r_Z can be found for any one of the three cases, if the above expression is expanded, the following general form results.

$$\gamma_1 r_{X/Y}^2 + \gamma_2 r_{X/Y} r_Z + \gamma_3 r_Z^2 = 1$$

Case No. 1

$$\gamma_1 = A'_{11} \qquad \gamma_2 = 2A'_{13} \qquad \gamma_3 = A'_{33}$$

Case No. 2

$$\gamma_1 = A'_{22} \qquad \gamma_2 = 2A'_{23} \qquad \gamma_3 = A'_{33}$$

Case No. 3

$$\gamma_1 = A'_{11} + 2A'_{12} + A'_{22} \qquad \gamma_2 = 2(A'_{13} + A'_{23}) \qquad \gamma_3 = A'_{33}$$

now by substituting

$$r_{X/Y} = -\frac{\beta_2}{\beta_1} r_Z - \frac{1}{\beta_1}$$

results in the following expression

$$\gamma_1 - \left(\frac{\beta_1}{\beta_1} r_Z - \frac{1}{\beta_1}\right)^2 + \gamma_2 - \frac{\beta_2}{\beta_1} r_Z - \frac{1}{\beta_1} r_z = \gamma_3 r_Z{}^2 = 1$$

expanding and combining like terms yields

$$\gamma_1 \left(\frac{\beta_2}{\beta_1}\right)^2 + \gamma_3 \qquad \gamma_2 \frac{\beta_2}{\beta_1} r_Z{}^2 + 2\gamma \; \frac{\beta_2}{\beta_1{}^2} - \gamma \, \beta\frac{1}{1}$$

$$\left(r_Z + \gamma_1 \frac{1}{\beta_1}\right)^2 \; -1 = 0$$

Let the coefficients for the terms r_Z^2, r_Z, and constant term be represented by T1, 5T2, and T3, respectively.

$$r_Z = -\frac{T_2 \pm T_2^2 - (4T1T3)^{1/2}}{2T1}$$

$$r_{Z/Y} = -\frac{\beta_2}{\beta_1} r_Z - \frac{\beta_2}{\beta_1}$$

Values for r_X, r_Y, and r_Z can be obtained depending upon what case is being solved.

SOLVR subroutine will return vector components for any of the three cases.

Definition of call to SOLVR subroutine

CALL SOLVR (A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, SS, R1, R3)

Case No. 1	Case No. 2	Case No. 3
$A_1 = A'_{11}$	$A_1 = A'_{12}$	$A_1 = A'_{11} + A'_{12}$
$A_2 = A'_{21}$	$A_2 = A'_{22}$	$A_2 = A'_{21} + A'_{22}$
$A_3 = A'_{31}$	$A_3 = A'_{32}$	$A_3 = A'_{31} + A_{32}$
$A_4 = A'_{13}$	$A_4 = A'_{13}$	$A_4 = A'_{13}$
$A_5 = A'_{23}$	$A_5 = A'_{23}$	$A_5 = A'_{23}$
$A_6 = A'_{33}$	$A_6 = A'_{33}$	$A_6 = A'_{33}$

$A_7 = A'_{11}$	$A_7 = A'_{22}$	$A_7 = A'_{11} + 2A'_{12} + A'_{22}$
$A_8 = A'_{13}$	$A_8 = A'_{23}$	$A_8 = A'_{13} + A'_{23}$
SS = SS	SS = SS	SS = SS
R1 = X component of r_1	R1 = Y component of r_2	R1 = X and Y component of r_3
R3 = Z component of r_1	R3 = Z component of r_2	R3 = Z component of r_3

After SOLVR constructs all three cases for vector $\vec{r}$, these three vectors are used to find an ellipse matrix [a] on the projection plane. The properties of the three $\vec{r}$ vectors are such that they satisfy the three-dimensional ellipsoid and when projected satisfy the ellipse on the projection plane.

Also the properties are such that the coefficients of the ellipse matrix can be found. This is accomplished by SOLVA subroutine.

Projection of $\vec{r}$ onto the Projection Plane

$$\vec{r'} = \vec{\rho'} - \vec{S'}\,\vec{\rho} = \left(\frac{S_X + r_X}{S_X + r_Z}, \frac{S_Y + r_Y}{S_Z + r_Z}\right)$$

$$\vec{S'} = \left(\frac{S_X}{S_Z}, \frac{S_Y}{S_Z}\right)$$

$$\vec{r'} = \left(\frac{S_X + r_X}{S_Z + r_Z} - \frac{S_X}{S_Z}, \frac{S_Y + S_Y}{S_Z + r_Z} - \frac{S_Y}{S_Z}\right)$$

$$r'^T \underline{a} r' = 1 \qquad [a] = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

Since there are three $\vec{r}$ vectors and only three of the four components of [a] are independent, the components of [a] are obtained by solving three equations simultaneously in subroutine SOLVA. SOLVA returns the components of [a].

To circumscribe the ellipse with a rectangle, the major and minor axis vectors must be found. These vectors are found by solving for the eigenvectors of [a].

$$\underline{a}\vec{r} = \lambda \vec{r}$$

This condition is true only for the vectors that represent the major and minor axis of the ellipse.

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix} \begin{pmatrix} r_X \\ r_Y \end{pmatrix} = \lambda \begin{pmatrix} r_X \\ r_Y \end{pmatrix}$$

$$\begin{pmatrix} a_{11}r_X + a_{12}r_Y \\ a_{12}r_X + a_{22}r_Y \end{pmatrix} = \begin{pmatrix} \lambda r_X \\ \lambda r_Y \end{pmatrix}$$

$$\lambda r_X - a_{11}r_X - a_{12}r_Y = 0$$

$$\lambda r_Y - a_{12}r_X - a_{22}r_Y = 0$$

$$(\lambda - a_{11})\, r_X - a_{12} r_Y = 0 \qquad [1]$$

$$(\lambda - a_{22})\, r_Y - a_{12} r_X = 0 \qquad [2]$$

The only way No. 1 and No. 2 can be zero is if

$$\vec{r_1} = \begin{pmatrix} a_{12} \\ \\ \lambda_1 - a_{11} \end{pmatrix} = \begin{pmatrix} r_X \\ \\ r_Y \end{pmatrix}$$

$$\vec{r_2} = \begin{pmatrix} \lambda_2 - a_{22} \\ \\ a_{12} \end{pmatrix} = \begin{pmatrix} r_X \\ \\ r_Y \end{pmatrix}$$

These are the major and minor axes vectors.

These vectors must also satisfy the ellipse equation

$$\underline{a}\vec{r} = \lambda\vec{r}$$

$$\vec{r}^T\underline{a}\vec{r} = 1$$

$$r^T\lambda\vec{r} = 1$$

since λ is a scalar

$$\lambda \vec{r}^T \vec{r} = 1$$

$$\lambda_x^2 + \lambda_y^2 = \frac{1}{\lambda}$$

$$|r_1|^2 = \frac{1}{\lambda}$$

Both eigenvalues can be found by solving

$$\begin{vmatrix} \lambda - a_{11} & -a_{12} \\ -a_{12} & \lambda - a_{22} \end{vmatrix} = 0$$

$$\lambda = \frac{a_{11} + a_{22} \pm [(a_{11} + a_{22})^2 - 4(a_{11}a_{22} - a_{12}^2)]^{1/2}}{2}$$

Using these two eigenvalues, $\vec{r_1}$ and $\vec{r_2}$ are determined and must be normalized by the relation

$$|\vec{r_1}|^2 = \frac{1}{\lambda}$$

These equations are used to circumscribe the ellipse with a rectangle.

$$\vec{P_1} = \vec{r_1} + \vec{r_2}$$

$$\vec{P_2} = -\vec{r_1} + \vec{r_2}$$

$P_3 = -\vec{r_1} - \vec{r_2}$

$P_4 = \vec{r_1} - \vec{r_2}$

CONVEC (I,1,K) $= \vec{P_2} - \vec{P_1} = \vec{r_2} - \vec{r_1} - \vec{r_1} - \vec{r_2} = -2\vec{r_1}$

CONVEC (I,2,K) $= \vec{P_3} - \vec{P_2} = \vec{r_2} - \vec{r_1} - \vec{r_2} + \vec{r_1} = -2\vec{r_2}$

CONVEC (I,3,K) $= \vec{P_4} - \vec{P_3} = \vec{r_1} - \vec{r_2} + \vec{r_2} + \vec{r_1} = 2\vec{r_1}$

CONVEC (I,4,K) $= \vec{P_1} - \vec{P_4} = \vec{r_1} + \vec{r_2} - \vec{r_1} + \vec{r_2} = 2\vec{r_2}$

I = 1,2 for vectors on the projection plane

K 1,2,3,4, ... 90

K is an ellipsoid number or polygon number.

APPENDIX C

INTERSECTION OF A THREE SPACE VECTOR AND PLANE

Figure C.1 shows the typical vector problem that represents the intersection of a three-space vector and a plane. The problem is to determine if the intersection point lies along vector $\vec{r}$ or beyond the tip of $\vec{r}$. The figure shows that if Tau is less than one, the intersection point lies between the two points VP and PT.

Tau is found with the following equations (where $\hat{N}$ is the normal unit vector to the plane).

$$\hat{N} \cdot \vec{C} = 0 \qquad \vec{C} = \vec{r}\tau - \vec{P}$$

$$\hat{N} \cdot (\vec{r}\tau - \vec{P}) = 0$$

$$\hat{N} \cdot \vec{r}\tau - \hat{N} \cdot P = 0$$

$$\tau = \frac{\hat{N} \cdot \vec{P}}{\hat{N} \cdot \vec{P}}$$

$\tau > 1$ point is not blocked by the intersection point.

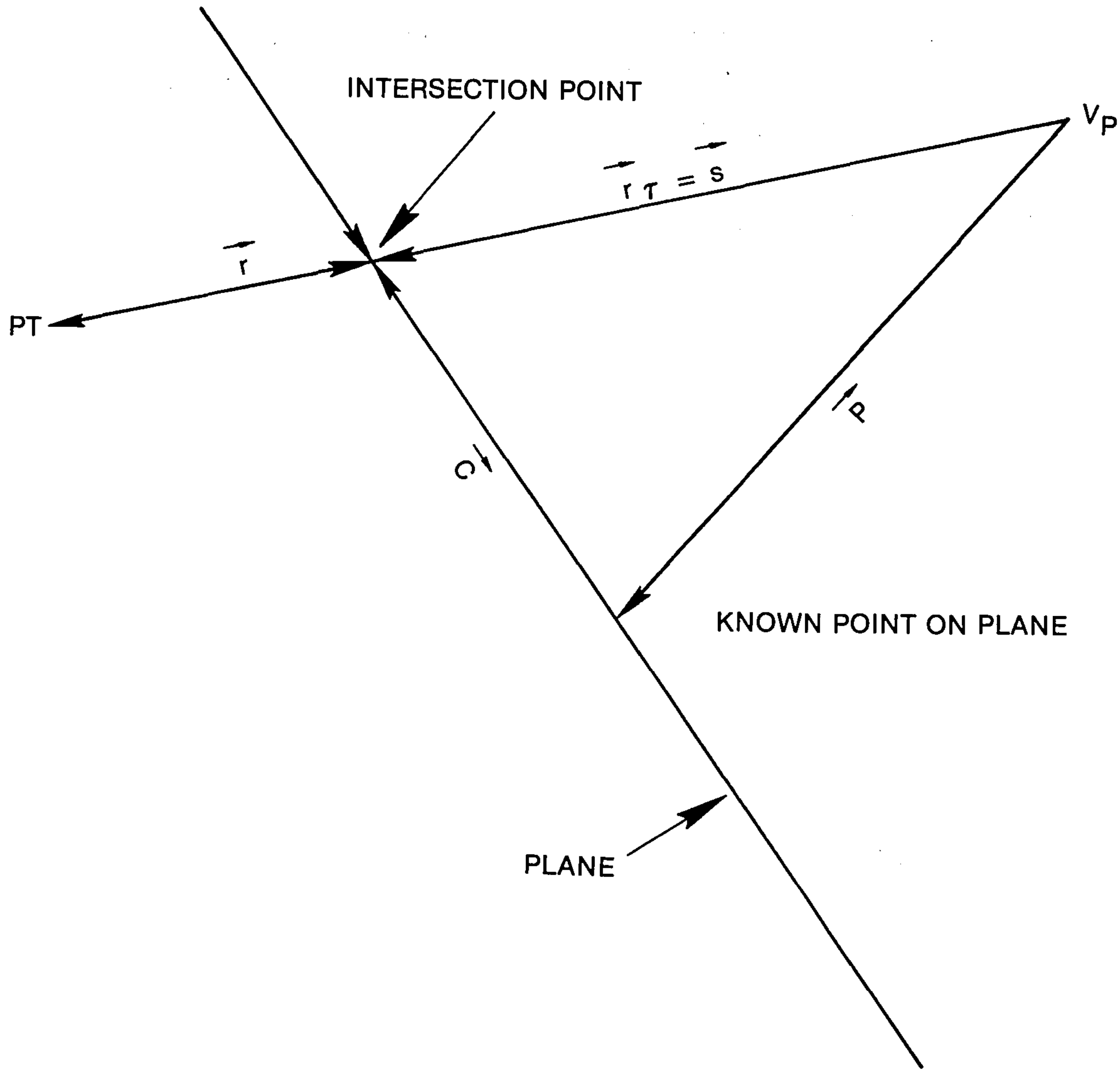


Figure C.1. Intersection of a Three Space Vector and a Plane

APPENDIX D
INTERSECTION OF LINE SEGMENTS IN A PLANE

Given two line segments,

$$\overline{P_1P_2} \quad \text{and} \quad \overline{P_3P_4}$$

where

$$P_i = (x_i, y_i)$$

Consider all parallel lines to be nonintersecting.

<u>The Regular Configuration</u> -- Neither of the line segments is vertical.

The line which contains $\overline{P_1P_2}$ has the equation

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{y - y_1}{x - x_1}$$

which simplifies to

$$y = (x - x_1)\, m_1 + y_1$$

where

$$m_1 = \frac{y_2 - y_1}{x_2 - x_1}$$

Likewise the line containing $\overline{P_3P_4}$ has the equation

$$y = (x - x_3)\, m_2 + y_3$$

where

$$m_2 = \frac{y_4 - y_3}{x_4 - x_3}$$

Since at the point of intersection

$$P_0 = (x_0, y_0)$$

$$y_0 = (x_0 - x_1)\, m_1 + y_1$$

$$y_0 = (x_0 - x_3)\, m_2 + y_3$$

then equating and solving for x_0 yields

$$x_0 = \frac{y_3 - y_1 + m_1 x_1 - m_2 x_3}{m_1 - m_2}$$

To determine if the point of intersection of the two lines is on each line segment, note

$$P_o \varepsilon P_i P_j \Leftrightarrow P_o = P_i + t(P_j - P_i) \quad \text{for } 0 < t < 1$$

Then let

$$t = \frac{x_0 - x_1}{x_2 - x_1}$$

$$s = \frac{x_0 - x_3}{x_4 - x_3}$$

then if $0 \leq t \leq 1$ and $0 \leq s \leq 1$, there is intersection.

But, since we will say that the two segments do not intersect if the point of intersection is one of the endpoints, then if $0 < t < 1$ and $0 < s < 1$, there is intersection.

If one of the lines is vertical, the regular procedure will not work since there will be a zero denominator in one of the m_i's. Therefore, provided the nonvertical segment is not horizontal, make the substitution

$$P_i' = y_i, x_i \qquad \text{for each } P_i = (x_i, y_i)$$

Using the P_i' endpoints, the regular procedure will then determine if there is an intersection.

The only case not covered, so far, is the case where one segment is vertical and the other is horizontal. Without loss of generality, assume $\overline{P_1P_2}$ is vertical and $\overline{P_3P_4}$ is horizontal.

In this case

$$P_0 = (x_1, y_3)$$

so

$$(x_1, y_3) = (x_1, y_1) + t\,[(x_1, y_2) - (x_1, y_1)] \Rightarrow$$

$$t = \frac{y_3 - y_1}{y_2 - y_1}$$

likewise

$$[(x_1, y_3) = (x_3, y_3) + s\,(x_4, y_3) - (x_3, y_3)] \Rightarrow$$

$$s = \frac{x_1 - x_3}{x_4 - x_3}$$

then if $0 < t < 1$ and $0 < a < 1$, there is an intersection.

☆U.S.Government Printing Office: 1983 — 659-005/2051