Interactive Calculation of Cross-Sectional Areas for Aircraft Design and Analysis

by

Michele Marie Grieshaber

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

APPROVED:

J. Robert Mahan, Chairman

Arvid Myklebust

R. W. Walters

August 5, 1988

Blacksburg, Virginia

Interactive Calculation of Cross-Sectional Areas for Aircraft Design and Analysis

by

Michele Marie Grieshaber

J. Robert Mahan, Chairman

Mechanical Engineering

(ABSTRACT)

Discussed is the development of two computer programs for the calculation of cross-sectional areas for existing aircraft and aircraft concepts. The first program allows the user to display aircraft described in Hess format, to restructure the data in a manner compatible with computational fluid dynamic codes for predicting viscous flow drag, and to compute and display the axial distribution of cross-sectional area normal to the free-stream airflow. The second program is an extension of an existing conceptual design code called ACSYNT/VPI. The extension allows calculation of areas of intersection between a cutting plane tilted at an arbitrary angle with the aircraft axis and the aircraft. The design of both codes is described in depth, and user's guides for the resulting programs are included in the appendices.

Acknowledgements

Toute d'abord je voudrais remercier le Pr. J. R. Mahan de m'avoir convaincue de poursuivre des études élevées. I greatly appreciate all the encouragement he's given me concerning my studies both here and in France.

I would also like to thank Dr. A. Myklebust for all the support and guidance during the past year. Without him I wouldn't have been able to rewind 10000 Maniacs so quickly.

Thanks also go to Dr. R. W. Walters for serving on my committee and acting as the aerodynamic expert for this thesis.

Furthermore, I owe much to both Steve Wampler and S. Jayaram, Jayaram Sankar, Rayjam Snakbar, for enlightening me with wisdom and for helping me to fix my programs when they didn't run.

And finally I thank my father and mother for their love and support.

Table of Contents

List of Illustrations

1.0 INTRODUCTION

1.1 Overview

This thesis addresses two problems. The first is that of the display, reordering and processing of aircraft geometry data supplied by the NASA Langley Research Center in the Hess format. Such data files will be referred to as "Hess files" in this thesis in honor of J. L. Hess, who first suggested this particular format for aircraft geometry data [1]. Processing in this case refers to the calculation of the area enclosed by the wetted perimeter of the aircraft after first defining this perimeter in terms of the locus of points common to an intersection between a normal cutting plane and the aircraft skin. Computation of these point-defined perimeters and corresponding cross-sectional areas is accomplished either automatically at specified intervals of intersection or with user assistance at specified axial locations. The second problem deals with the computation of areas formed by cutting planes and their axial distribution for aircraft concepts defined by ACSYNT/VPI, a highly interactive aircraft conceptual design code currently under development at VPI&SU. The two problems are related since they both involve the computation of areas formed by cutting planes

which intersect the longitudinal axis of the aircraft. Therefore, procedures developed during solution of the first problem are often applicable to the solution of the second problem.

The data contained in Hess files represent coordinates on the surface of a component. These are arranged as consecutive cross-sections along the axis of the component. The locus of points defining the cross-section perimeter is formed by passing a cutting plane through each component with an orientation perpendicular to the local axis of the component. For example, fuselage data are ordered in cross-sections perpendicular to the central longitudinal axis of the aircraft, while wing data are arranged on the perimeter of cross-sections perpendicular to an axis running from the root to the tip of the wing. While this ordering of the data is convenient for inviscid flow calculations involving the solution of mixed hyperbolic-elliptical equations, it presents a problem to those interested in performing viscous flow analyses on the aircraft, which involve the solution of elliptic and parabolic equations. Most computational fluid dynamics (CFD) algorithms require that the cross-sectional data be arranged in a structured order perpendicular to the direction of free-stream air flow. Due to the fact that many CFD codes need data structured in this manner, and considering that the Hess files contain valuable geometric data, the first part of this thesis deals not only with the development of an interactive graphics program for displaying aircraft represented by Hess files but also with the restructuring of Hess data to meet the needs of CFD codes. As an added feature, the ability to calculate the area of a specific cross-section by both user-assisted and automatic means has also been provided. Area distribution as a function of the distance along the aircraft axis is then displayed in graphical format.

ACSYNT/VPI is the product of a grant from NASA Ames Research Center to Virginia Polytechnic Institute and State University. The program merges a highly sophisticated aircraft conceptual design code with an interactive graphical interface. The conceptual design code, known as ACSYNT (AirCraft SYNThesis), has been developed by engineers at NASA Ames. The program allows for conceptual design studies of advanced aircraft and may be useful in assessing the performance of aircraft available to potentially unfriendly governments.

The task of enhancing ACSYNT's capabilities to include a user interface for viewing and modification of aircraft geometries, as well as for interactive analysis of the resulting aircraft, was

begun in the summer of 1987 by a team of graduate students from the Department of Mechanical Engineering under the direction of Professors A. Myklebust and J. R. Mahan. The resulting interactive program, known as ACSYNT/VPI will take several years to complete. At the outset of the project, several short- and long-term goals were established in the form of software requirements. The reader may refer to [2] for a more complete description of the user and software requirements, as well as for an in-depth description of the functional capabilties of ACSYNT/VPI. A few of the software requirements targeted for completion during the first year of development are discussed in this thesis. These include the computation of cross-sectional areas and the capability to display the resulting output in graphical and tabular formats.

The ability to calculate cross-sectional areas is an important addition to the ACSYNT/VPI software. The cross-sectional areas are computed on the basis of wetted perimeters defined by the intersection of the aircraft with a normal cutting plane. Utilizing the information gained by computation of the area for a series of intersection locations, the designer can determine which portions of the aircraft should be reduced in area in order to decrease the aerodynamic drag of the aircraft.

1.2 Literature Review

In the following discussions of aircraft geometry a typical aircraft coordinate system is assumed wherein the x-axis is the axis of roll, the y-axis of pitch, and the z-axis of yaw.

1.2.1 Hess File Display and Intersection

No software is currently available which displays and reorders Hess format data. Hess format data are relatively simple to display provided one has access to a graphics display terminal and a means by which to convert numerical data into graphical output. The difficulty arises when it is

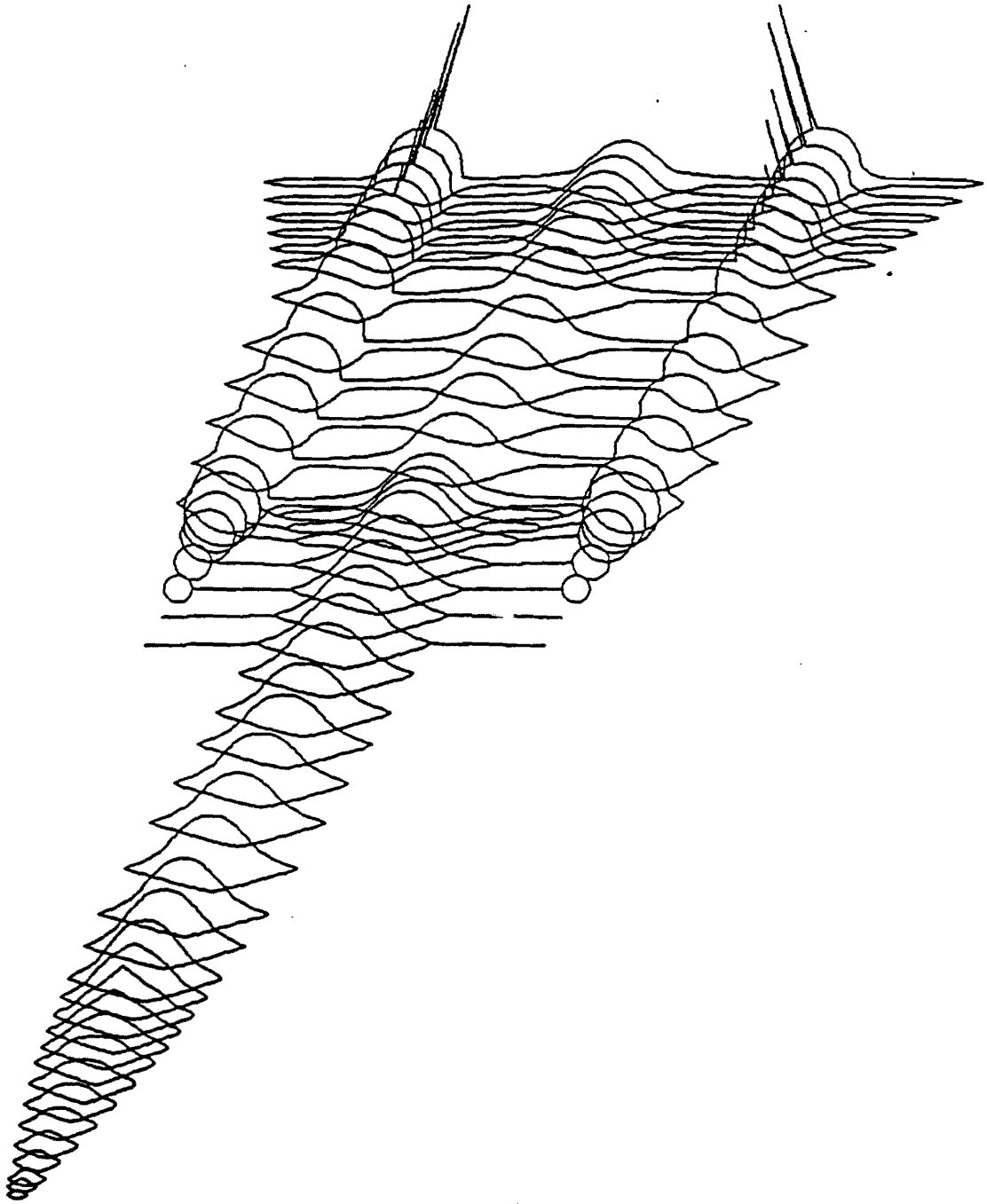

Fig. 1. Cross-Sectional Representation of the SR71 Aircraft [3] .

necessary to analyze the performance and aerodynamics of an aircraft described in Hess format. The reason is that most CFD codes involve implicit flow solutions and many of these require data in the form of surface coordinates along wetted perimeters perpendicular to the axis of the free-stream airflow. Figure 1 shows the method in which aircraft data are commonly ordered for use by CFD algorithms. Cross-sections normal to the free-stream airflow are needed because drag forces on the aircraft are the greatest for this orientation. But exactly why is it important to organize the data in a structured manner? The reason is that flowfield calculations are concerned with the principles of conservation of mass, momentum, and energy within a volume, V, surrounded by a surface, S. One example of this type of analysis by Walters, et al. [3], clearly illustrates the need for ordered data. According to [3], it is common to discretize the three-dimensional domain of an aircraft into hexagonal volumes (cells), each having quadrilateral cell faces. The governing conservation principles are then applied directly to each volume in order to analyze the flow field over the surface of the aircraft. Therefore, for these volumes to be defined, as well as for the computation of derivatives with respect to time and space which are required by the conservation equations, the cross-sections need to be ordered such that logical indices can determine their location.

The reordered data, now in terms of ordered sets of wetted perimeters, not only simplify CFD computations, but also facilitate the calculation of areas of the individual cross-sections. These area data can easily be displayed as an area distribution of the aircraft along the x-axis, provided one keeps track of the locations at which the cross-sectional areas were computed.

The importance of area distribution to the conceptual and final design of aircraft was first discovered during the early years of jet-powered flight [4] . At the end of World War II, there was a dramatic surge in the production of jet aircraft. However, these aircraft were all limited to subsonic flight speeds due to the large drag forces encountered approaching Mach 1. The planform and corresponding cross-sectional area distribution shown in Fig. 2 illustrate the cross-sectional area distributions of typical jet aircraft of that era. Note the steep slope of the distribution in the wing and tail areas. In contrast, for almost a century ballisticians were well aware of the fact that the speed of a supersonic bullet with a smooth variation of cross-sectional area was higher than projectiles with abrupt or discontinuous area distributions [5] . In 1954, R. Whitcomb, working

at NACA's Langley Aeronautical Laboratory, extended the ballistics theory to the design of supersonic aircraft [6] . Whitcomb reasoned that the cross-sectional area of an aircraft should be smooth, with no discontinuities. He found that the shock-wave formations about a relatively complex swept-wing/body combination at zero lift near the speed of sound are similar to those which occur for a body of revolution with the same axial development of cross-sectional area normal to the airstream. This meant that in the region of the wings and tail, the fuselage cross-sectional area should decrease in order to compensate for the additional areas added by these components. When this principle is incorporated into an aircraft design, a "coke bottle" fuselage shape results. The smoother transition in terms of area distribution along the axis of a modern supersonic aircraft, an F5-E, is depicted in Fig. 3. This result, known commonly as Whitcomb's Area Rule, successfully reduced the peak drag near Mach 1 such that supersonic flight of aircraft was possible by the mid-1950's [6].

While Whitcomb's Area Rule works well for aircraft undergoing transonic flight, a different but somewhat similar principle applies to supersonic flight. In order to extend into the supersonic region the idea that "smoother" bodies create less drag, a series of cross-sections must once again be considered. This time instead of creating the intersections with planes perpendicular to the free airstream, however, the intersections are created by planes tilted at an angle referred to as the Mach angle [7] . The rules for obtaining low wave drag state that each of the equivalent bodies produced by the oblique projections should be as smooth as possible. Wave drag is unique to supersonic flow and is associated with the energy radiated away from the vehicle in the form of pressure waves in much the same way as a fast-moving ship causes waves on the surface of the water. Therefore, the prediction of wave drag is significant in the design of supersonic aircraft. Where does the computation of area enter into this scenario? It is of both theoretical and practical interest to find area distributions of aircraft bodies which, given a set of constraints, have the lowest possible wave drag. Thus, there is a definite correlation between the area distribution and the wave drag of a vehicle in supersonic flight. Generally, the drag is not very sensitive to small departures from the optimum shape of aircraft bodies provided the area distribution is smooth [8].

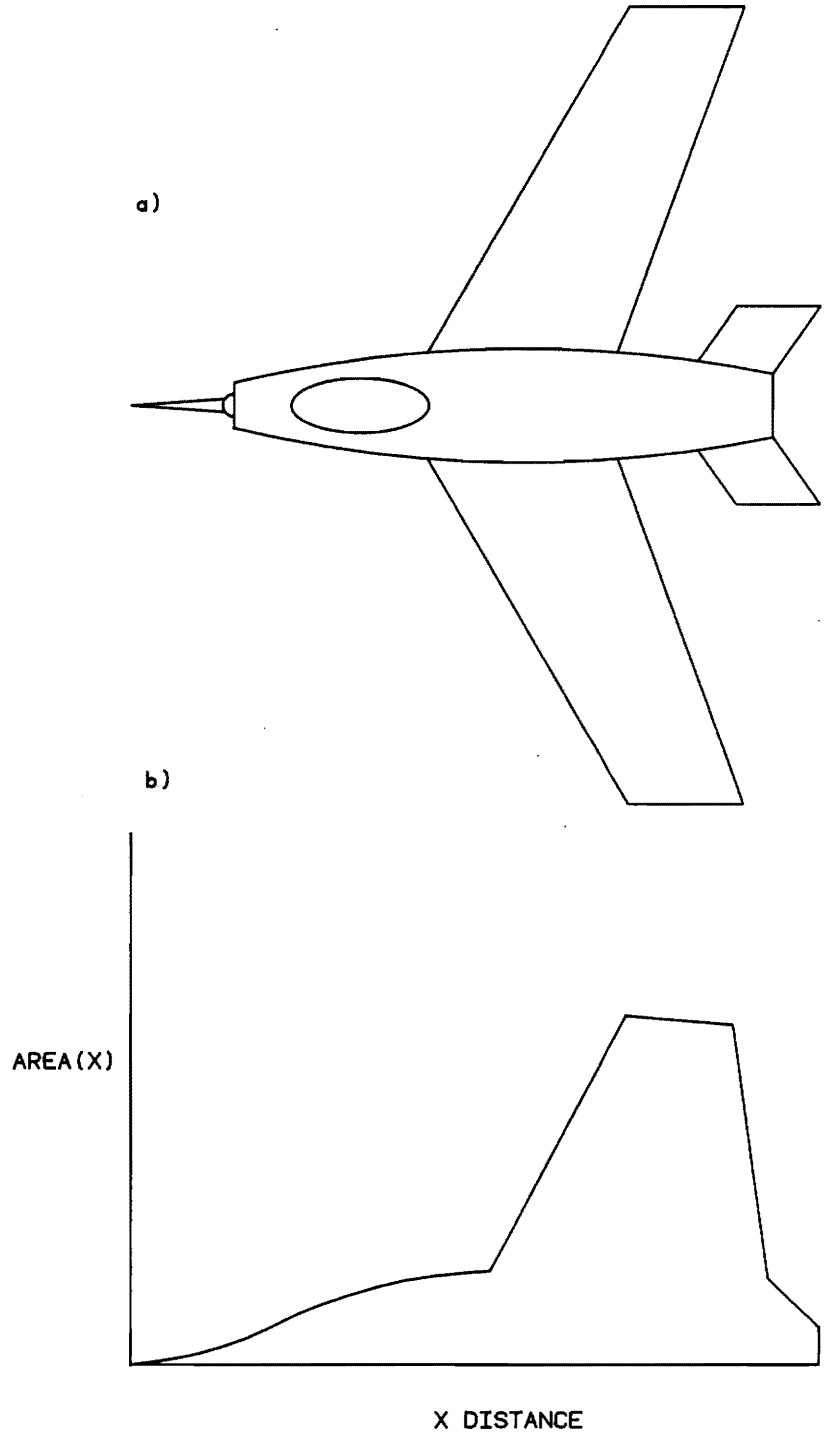

Fig. 2. Schematic of a)A Non-Area-Ruled Aircraft and b) the Corresponding Area Distribution.

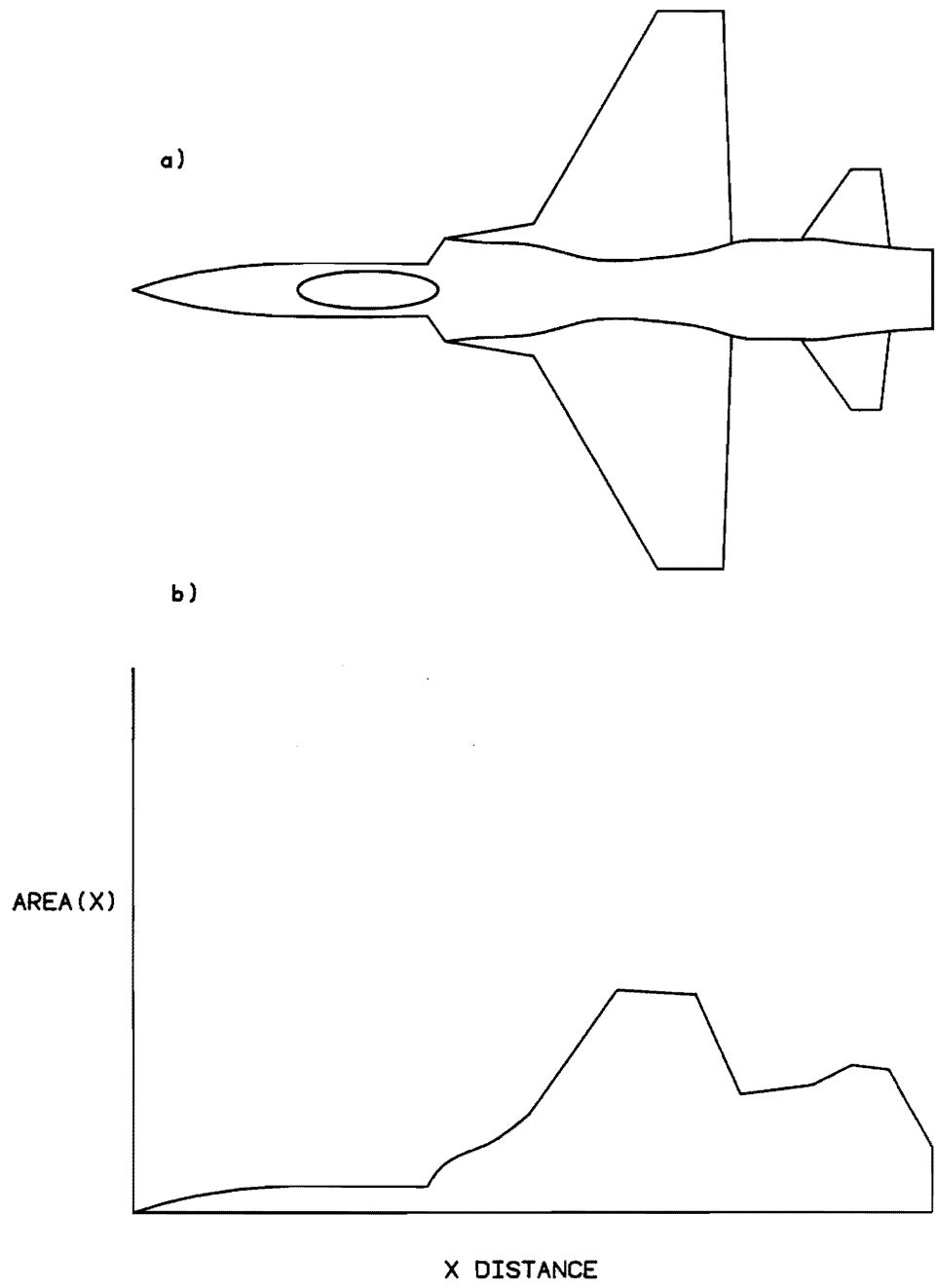

Fig. 3. Schematic of a)An Area-Ruled Aircraft and b)the Corresponding Area Distribution.

1.3 ACSYNT/VPI and Area Calculations

The ACSYNT code, which serves as a basis for the interactive code ACSYNT/VPI, encompasses many aircraft design disciplines. The merging of the analysis capabilities of ACSYNT with an interactive graphics interface results in a powerful design tool. In its mission-analysis mode, ACSYNT allows the user to enter geometric and engine parameters and then compute valuable information related to fuel consumption, range, stores, etc. With the addition of a graphical interface the user now has the added advantage that aircraft geometries may be visualized and modified directly on the screen. The first stage of the project to develop the graphics interface for ACSYNT was completed in June of 1988. Wampler's thesis [2], which describes in detail the functional capabilities of ACSYNT as well as the software design process used in the development of ACSYNT/VPI, should be read by anyone interested in the topics addressed by the present thesis.

There are other conceptual design codes which allow interactive design of aircraft. One such system is CDS (Configuration Development System) which was developed by and is used extensively at Rockwell International [9] . CDS permits the user to visualize the aircraft in three dimensions and enables the designer to add and arrange internal and external components. The program stores all surfaces and internal data as three-dimensional bodies of arbitrary shape described by cross-section. Furthermore, CDS can do a variety of on-line analyses such as wave-drag analysis, weight, balance and inertia calculations, and optimizations. It displays areas which could potentially increase wave-drag, and permits design-dependent drag optimization such as tail relocation. The main problem with CDS is that while its analysis capabilities are on the same level as those of ACSYNT, its interactive ability in terms of display and modification is not. For example, the designer must issue commands phrased as words, such as "display" and "move", in order to manipulate the visual data. ACSYNT/VPI on the other hand, allows the designer to utilize

valuator, string, and pick input devices to move and analyze the aircraft. These input devices are defined in Section 2.2.

A second code for aircraft conceptual design was developed by F. O. Smetana at North Carolina State University in 1984 [10]. Smetana and his students consider three modes for performance evaluation: take-off, point performance (assumes accelerations are zero), and path performance (assumes accelerations are nonzero). The aerodynamics, propulsive forces, and inertial characteristics of the aircraft, including its fuel consumption, are assumed known in the analysis. In ACSYNT, the optimization code which oversees the analysis portion of the program calculates the amount of fuel necessary for the mission as well as several other parameters particular to a given aircraft.

HESCAD, a program which enables users to perform interactive design and analysis of preliminary helicopter designs, was developed by L. J. Lu in 1985 at the Virginia Polytechnic Institute and State University [11]. The program is a merger of HESCOMP, a code written at Boeing Vertol for sizing and evaluation of helicopters, with a CADAM interface. It provides the designer with the tools to evolve and refine helicopter components through interactive, on-line, computer graphic display devices. A method is provided whereby results of analyses of the vehicle, such as weights, centers of gravity, moments and products of inertia, can be displayed directly on the screen. VASCOMP, a V/STOL aircraft sizing and performance program also written at Boeing Vertol is very similar to HESCOMP [12] .

2.0 HESS INTERSECTION

2.1 Goals

This chapter contains an overview of the design of the Hess Intersection program. The goals of the program are discussed in terms of user, software and hardware requirements. Also addressed is the design of individual modules. The primary goal of the Hess Intersection program is to allow the user to view and restructure data contained in Hess files. The reader is referred to Section 1.2 for a description of the Hess file format. An intersection algorithm which may be utilized either automatically (at preset intervals) or at user-specified axial locations rearranges the data. Also of interest to the designer is the computation of cross-sectional areas which, when plotted as a function of station along the x-axis, represent the area distribution of the aircraft. The principles involved in coding the area distribution algorithm are also used in the second portion of this thesis. The theory behind the coding of the program is described in the present chapter. Ideas are developed which aid in the understanding of Hess Intersection. Once again, a typical aircraft coordinate system is assumed where the x-axis defines the roll, the y-axis the pitch, and the z-axis the yaw. The

reader interested in specifics on how to use the program are referred to the Hess Intersection User's Guide, Appendix A.

2.2 *Workstation Requirements*

Before proceeding with an in-depth discussion of the coding process, an overview of the workstation requirements is necessary. The Hess Intersection code was written using the three-dimensional graphics standard called PHIGS (Programmer's Hierarchical Interactive Graphics Standard). PHIGS is basically a library of graphics routines which uses, in this case, the language of FORTRAN-77 to invoke the routines. A major advantage of using PHIGS is that it is a device-independent graphics system. This means that an applications program based on PHIGS may be transported easily from one graphics device to another without affecting program operation. Therefore, the first workstation requirement is that it must support PHIGS. Having established this limitation, specific requirements supported by PHIGS and used by this particular application are:

1. specified logical graphical input devices required:
   a. valuator
   b. choice
   c. pick
   d. string (keyboard)
2. specified graphical input modes required:
   a. request mode
   b. event mode

The requirements listed above ensure the interactive ability of the program and therefore must be respected. For the reader unfamiliar with the input devices listed above, valuators are logical input devices which return a real value that may be mapped to a dial box or thumb wheel. They often utilize potentiometers for continuous input. A choice device can be either a button box or a set

of keys which represent specific functions in the application and return an integer defining the location of the choice. A pick device is a logical device which can be mapped to a real device, such as a tablet and cursor, which returns a location and path indicative of the active view. A string device is a often a keyboard which allows for text string input [13]. The Template manual on **Understanding PHIGS** [14] describes the interactive ability of the input devices in this way; "Input devices return information to the application, and the application takes appropriate action--from defining to editing to displaying graphical data". The modes of these input devices determine when the application will process the input. In REQUEST mode, the applications program prompts the user for input, then waits until completion of the input task to continue. In EVENT mode, the user signals when input should occur and the application processes that input at the first convenient moment. A more complete description of PHIGS and its attributes is given in reference [14].

2.3 Design of the Hess Intersection Program

Throughout the design of the Hess Intersection code, adherence to the "top down" design process was a concern. The "top down" design process as defined by Tausworthe [15] is similar to deductive reasoning in a rational thought process. That is, there is a definite progression from an abstract problem statement to a very specific programming task. For example, the top module of the program, "HESSMAIN", has only six executable lines, three of which branch off into subroutines which contain normally no more than seven executable lines of code each. Those routines in turn call other subroutines and so forth until completion of the task. It is this hierarchical organization which makes the code modular and facilitates debugging. The hierarchical structure also allows the deductive thought process to continue by enabling the designer to consider the program one level at a time.

2.4 *User/Menu Interface*

The reader interested in specifics concerning the coding of this module is referred to Appendix B. The screen layout for the Hess Intersection program, shown in Fig. 4, consists of a top border which contains the program title, a right-handed menu area which contains the menu items, a large central workspace for display of the aircraft and output data, and a lower message area which may also be used for string input. Please note that the aircraft shown in the workspace is not an actual Hess model. The F-16 aircraft shown is used to demonstrate the location of the aircraft with respect to the rest of the menu area.

The menu structure for the program utilizes breadth instead of depth. All menu items are pickable; that is, they are activated when selected by a pick device. They are described below:

WHOLE AIRCRFT

- displays the entire aircraft for viewing purposes

HALF AIRCRFT

- displays half of the aircraft. This greatly reduces the time needed to reposition the aircraft because less data must be manipulated in the transformation.

DISPLAY PLANE

- renders the intersection plane visible to aid in manual intersection

NO-SHOW PLANE

- renders the intersection plane invisible to facilitate viewing of the aircraft

AUTO INTERSEC

- computes intersections between a plane of intersection and the aircraft at specified intervals

BEG MAN INT

- computes an intersection at an arbitrary location as specified by the user

END MAN INT

- signals that a manual intersection session has been completed

EXIT MAN INT

- allows user to exit the manual intersection mode

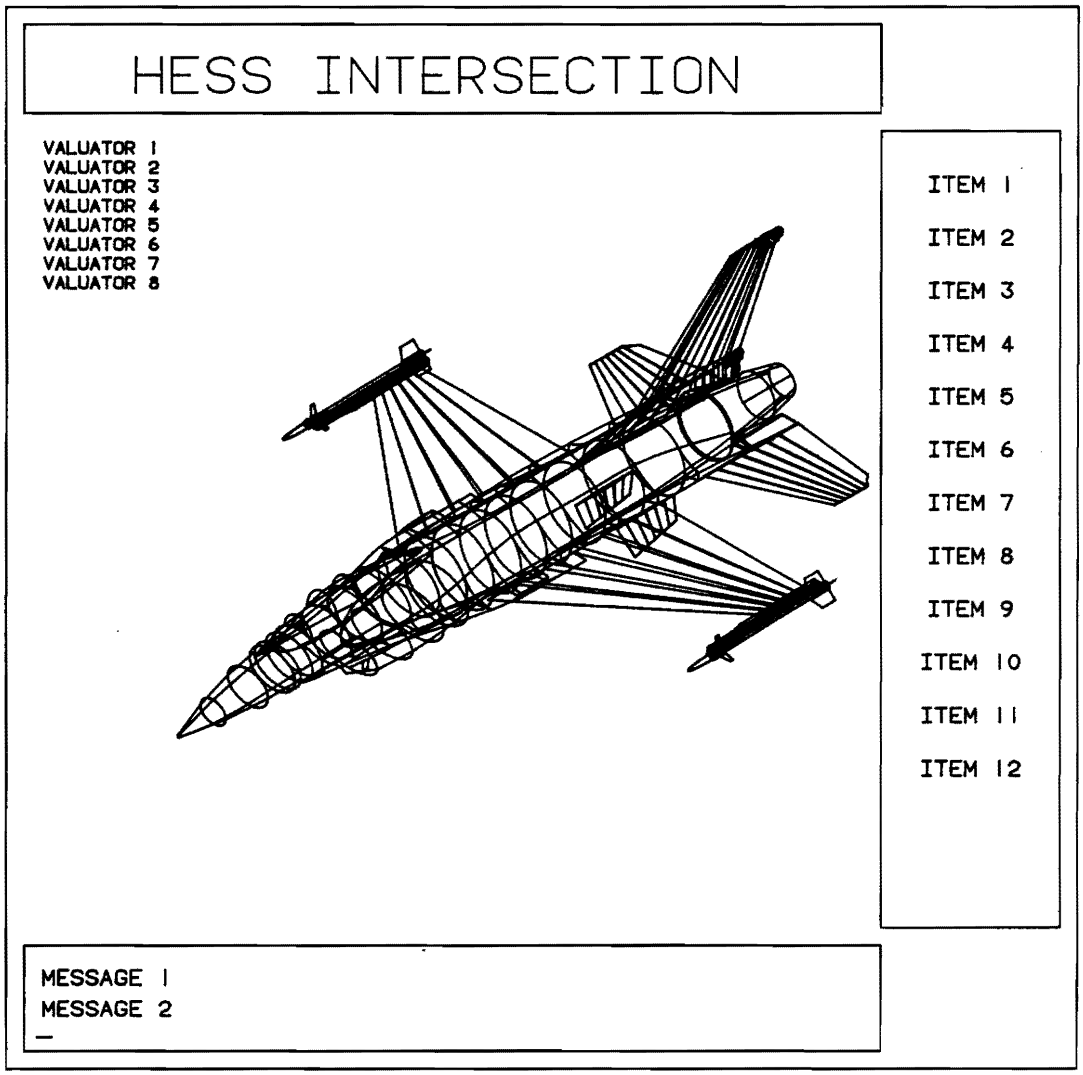

Fig. 4. Screen Layout for Hess Intersection Program.

AREA DISTRIB

- computes and displays a graph of the area distribution after having calculated an automatic intersection

FULL SCR

- allows full screen display of area distribution graph

1/4 SCR

- allows quarter screen display of area distribution graph

SAVE FILE

- overwrites a default file with user-specified filename and filetype

2.5 Reading in Hess Data

In order to display and manipulate Hess format data, they must first be read from an existing file. Upon entering the Hess Intersection program, the user is prompted to enter the filetype of the file which contains the Hess information. The Hess database filename must be FILE (a requirement to VM/VMS), whereas the filetype may be more descriptive, for example, F15HESS for a file representing an F-15. More specifically, the user's XEDIT file listing must contain a file of the form, following the previous example, FILE F15HESS. The program includes a routine to verify the existence of a file. **However, this routine is system dependent and replacement of portions of the routine is necessary if a system other than VM/CMS is being used.**

Upon establishing existence of the input file, the program creates a file located on logical unit number (LUN) 10 from which the data are read. A sample of the data contained in a Hess file is shown in Fig. 5. It is a requirement for purposes of this program that all Hess files follow this format. Hess files contain point data which are separated according to x,y,z-coordinate and status. The status of a data point is determined as shown below:

Status = 20

X	Y	Z	STATUS
116.29000	0.00000	105.35000	20 FUSELAGE 1-1
116.29000	0.00000	105.35000	
116.29000	0.00000	105.35000	
.	.	.	
.	.	.	
.	.	.	
150.00000	0.00000	121.36000	10 FUSELAGE 1-2
150.00000	1.05000	121.35000	
150.00000	2.08000	121.21000	
.	.	.	
.	.	.	
.	.	.	
196.50000	0.00000	135.17999	10 FUSELAGE 1-3
196.50000	1.69000	135.13000	
196.50000	3.79000	134.22000	
.	.	.	
.	.	.	
.	.	.	
.	.	.	
.	.	.	
.	.	.	
862.90002	70.00000	112.07000	20 HORZ TAIL 1-1
859.89001	70.00000	112.32000	
856.88000	70.00000	112.57000	
.	.	.	
.	.	.	
.	.	.	
870.90002	106.50000	112.03000	10 HORZ TAIL 1-2
868.77002	106.50000	112.16000	
866.63000	106.50000	112.29000	
.	.	.	
.	.	.	
.	.	.	
870.90002	106.50000	112.03000	20 HORZ TAIL 2-1
868.34998	106.50000	112.16000	
865.79999	106.50000	112.29000	
.	.	.	
.	.	.	
.	.	.	

Fig. 5. Sample Data from Hess File.

- this line of data begins a new network

Status = 10

- this line of data begins a new cross-section

Status = 30

- this line of data signals the end of the file

Status column is blank

- this line of data is part of the current cross-section and network

In the discussion which follows, the term "network" refers to the divisions in data structure as contained in the Hess file. These divisions represent a change in the spacing between cross-sections and/or the number of nodes per cross-section. The networks are elementary blocks of the aircraft structure, more elementary than components. Subdivision of components into networks is needed because some components may require more accurate surface definition in certain regions. In these locations the surface of the aircraft needs to be defined by a mesh of points which is more dense. For instance, the fuselage of the F-15 used to test the Hess Intersection code is comprised of networks one through six. This means that the spacing of cross-sections and nodal points is not uniform for the entire fuselage.

Upon reading a line of data, the program checks the status column to determine in which category the point belongs. Storage of the data points is in terms of a four-dimensional array which contains information regarding the x,y,z-coordinate, the node number of the point on the cross-section, the cross-section in the network, and the network. Figure 6 illustrates the hierarchical structure between the nodes, cross-sections, and networks as defined by Hess input. Due to the amount of data contained in the Hess files and considering that nearly all components of an aircraft are symmetric with respect to the x,z-plane, it is not difficult to understand why only half of the data needed to define an aircraft is stored in the Hess file. This presents little problem to the user wishing to display the entire vehicle, for a reflection about the plane of symmetry will produce a complete aircraft. In the interest of speed, computations such as intersections are performed using only half of the aircraft geometry as defined in Hess format.

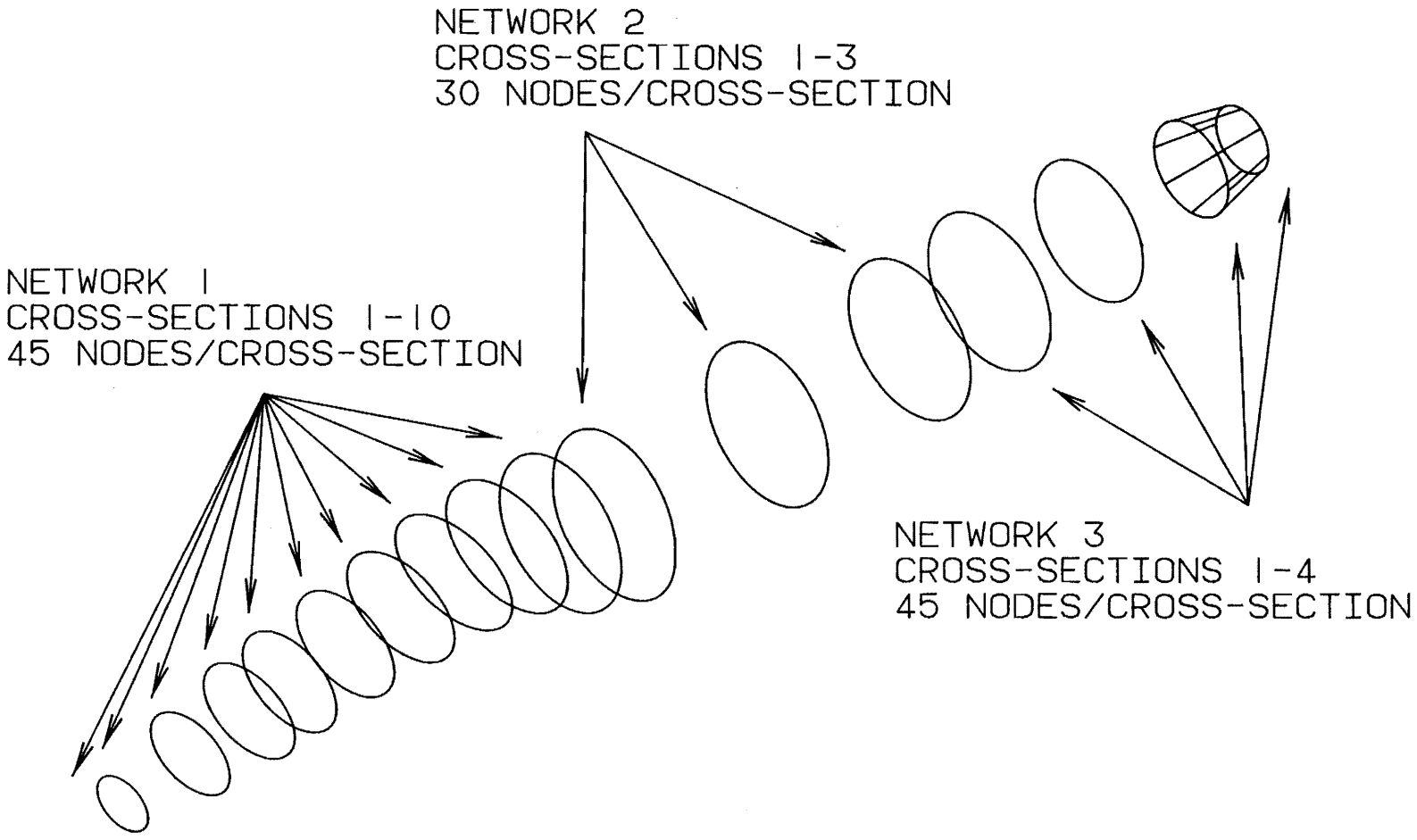

Fig. 6. Hierarchy of Hess Data.

2.6 *Hess File Display*

After reading in the Hess file and reordering the data in terms of the four-dimensional array described in the preceding section, the data are ready for display. As a simplification, the data are displayed using straight-line approximations of splines. This simplification is justifiable for several reasons. First, just as Simpson's Rule for the calculation of an integral increases in accuracy as the number of trapezoidal elements increases, segmental approximations of the spline approach the actual curvature as the number of divisions along the length of the spline increases. Second, computations involving the intersection between a plane of intersection and a straight line segment are computationally less extensive than those between an intersection plane and a spline. Finally, since PHIGS does not support the three-dimensional spline elements it is much simpler to utilize the existing three-dimensional line capabilities of the graphics standard. In order to best approximate the true shape of the aircraft using straight line segments the following procedure has been developed:

1. Consider each member of the ensemble of networks one at a time.
2. For each network hold the node number constant and pass from one cross-section to the next until the end. In this way a series of points representing a spline is identified for the particular network.
3. Send the collection of spline points to a routine which computes the tangent vectors for each point in the array.
4. Utilizing the array of points and their corresponding tangent vectors, divide the data into sets consisting of two points and their corresponding tangents.
5. With the data now arranged in two-point, two-tangent sets, approximate a cubic polynomial between the points.
6. Subdivide the cubic polynomial into a specified number of segments.
7. Draw straight line segments between the points of the divided cubic polynomial.

In this manner, the program ensures that the straight-line approximation of the aircraft model is still accurate.

2.7 *Intersection*

Before beginning the discussion on intersection, clarification of the meaning of the term "intersection" as used in this thesis is necessary. An "intersection" is the locus of all points common to the approximation of the surface describing the aircraft and a plane which intersects it normal to the x-axis.

Once the information contained in the Hess file has been displayed, the user has the option of specifying the intersection operation by automatic or user-assisted means. The algorithm which computes the intersection is the same in both cases. In order to decrease the amount of time necessary to compute an intersection, a procedure for determining which line segments will be intersected by the plane and which segments can be discarded has been developed. The procedure utilizes the fact that the intersection plane is defined in the y,z-plane and is therefore perpendicular to the x-axis. This implies that in order to determine the location of the intersection plane, one need only find the plane's location in terms of the x-coordinate. It also implies that any network whose boundaries do not contain the x-location of the intersection plane need not be considered as a candidate for intersection. The procedure is outlined below:

1. Examine each network to determine if the intersection plane is located within the minimum and maximum x-axis limits of the network.
2. Once a network which contains the intersection plane has been found, a second search begins. For the current network, the node numbers are held constant and the cross-sections are transversed from the first in the network to the last. In this manner creation of a spline along the local axis of the network results.
3. Each spline is then tested for intersection by determining if the intersection plane is located somewhere along the spline.
    a. If the x-coordinate of the point of intersection between the intersection plane and the spline lies within the x-range of the spline, then the next step is to determine exactly which two nodes of the spline bound the intersection plane.

b. The two boundary nodes are found using a half-interval search technique which uses the x-coordinates of the plane and spline array to determine the nodes closest to the plane.

c. Having found the two closest nodes on each side of the intersecting plane, calculation of the intersection between a plane and a line segment defined by the two nodal points commences.

There now remains the problem of finding the coordinates of intersection between a line segment and a plane of intersection. In order to present a clearer problem statement, it is first necessary to define the intersection plane. Figure 7 illustrates the intersection of a straight line and a plane [16] . The variables employed in the following discussion are defined in Fig. 7.

In general, a straight line segment may intersect a plane at one point or lie in the plane. If they intersect then, as shown in Fig. 7,

$$\mathbf{a} + u\mathbf{b} + w\mathbf{c} = \mathbf{d} + t\mathbf{e} \ . \tag{2.1}$$

Equation (2.1) represents three linear equations in three unknowns: u, w, and t . The symbols u and w represent the parametric lengths of the two vectors which define the intersection plane. The symbol t represents the parametric coefficient of the length of the line. In order to isolate t, form the scalar product between the cross product of the vectors **b** and **c** and Eq. (2.1). The resulting equation,

$$\mathbf{b} \times \mathbf{c} \cdot (\mathbf{a} + u\mathbf{b} + w\mathbf{c}) = \mathbf{b} \times \mathbf{c} \cdot (\mathbf{d} + t\mathbf{e}) \ , \tag{2.2}$$

may be solved for t, yielding

$$t = \frac{(\mathbf{b} \times \mathbf{c}) \cdot \mathbf{a} - (\mathbf{b} \times \mathbf{c}) \cdot \mathbf{d}}{(\mathbf{b} \times \mathbf{c}) \cdot \mathbf{e}} \ . \tag{2.3}$$

Once obtained, the variable t determines the distance along the line segment from $\mathbf{e}_1$ to $\mathbf{e}_2$ of the point of intersection.

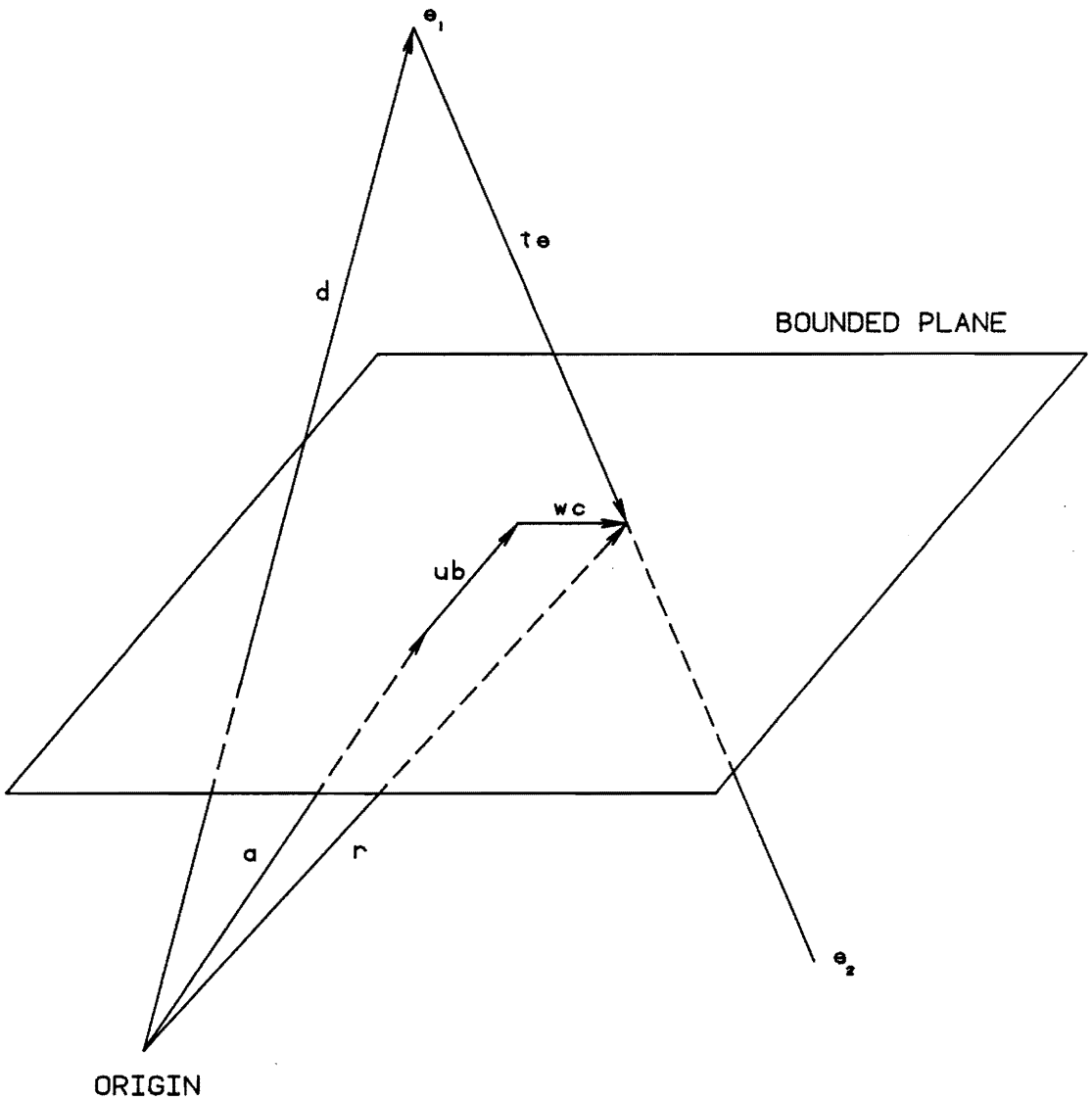

Fig. 7. Intersection Between a Plane and a Line Segment.

2.8 *Area Calculation*

At the conclusion of an intersection session; that is, when all intersection data have been computed for a specific location of the intersection plane; the cross-sectional area of each intersected network and the total area of the aircraft cross-section are calculated as a function of that axial location. This results in the creation of one data point for each network on the area distribution curve. When several intersection sessions have been completed there are sufficient data for the display of the area distribution curve.

The subroutine which computes the area for each network utilizes an algorithm for the calculation of area of a planar polygon developed by J. Sankar [17]. Sankar's code requires that the input data be a two-dimensional series of points. The basis of the algorithm is that straight-line approximations may be drawn between consecutive data points to form the perimeter of the polygon which approximates the wetted perimeter. From a central location situated within the polygon, triangular areas ressembling pie segments are formed by connecting the center point to each point on the perimeter of the polygon. The areas of these triangular regions are calculated one by one and added to or subtracted from the accumulating total area of the polygon, depending on the sign of the angle formed between lines drawn to the center from two consecutive perimeter points. This angle-sensitive addition or subtraction permits concave as well as convex polygons to be treated. This technique for area calculation will normally yield an underestimate of the actual area since the actual base of most of the "triangles" is convex rather than flat. However, because the underestimate is more or less consistent for all cross-sectional areas, the shape of the area distribution will not be distorted significantly. Of course, accuracy will increase as the density of points around the polygonal perimeter is increased.

This method of area calculation is easily adapted to the computation of cross-sectional areas for an aircraft. The intersection data are stored in terms of x,y,z-coordinate, node number, and network, and the area for each network is determined in the following manner:

1. Progress through all the networks one by one, each time holding the network number of the array constant.
2. Pass to the AREA routine all of the y,z-coordinates of the intersection points which belong to the current network.
3. Save the area data in terms of network and the intersection session number. The intersection session number is simply a counter which keeps track of the number of different intersection plane locations.

The total cross-sectional area for the aircraft for a specific intersection session number is then computed by adding the contributions of the individual networks.

2.9 Area Distribution

Having stored the intersection areas, it is now possible to create a graphical display of both the total cross-sectional area and the area of each network forming the cross-section as a function of location along the x-axis. Noting that the area distribution graph produced with Hess data only represents half of the aircraft, the program simply multiples the vertical scale of the area distribution by two in order to represent the area of the entire aircraft. This is justifiable since the area distribution is employed only as a relative measure of the total cross-sectional area with respect to a specific component or a group of specific components. Recalling that if the area of the aircraft can be kept relatively constant along the x-axis, the aircraft will experience much less drag in the transonic and supersonic flight regions, this tool serves as a means for the user to recognize those components of the aircraft which could be reduced in cross-sectional area to diminish wave drag.

2.10 Intersection Files

Creation of an intersection data file occurs in one of two possible modes: automatic and manual. The output file created is written on LUN 98 and has the filename and filetype of FILE DEFALT. This file will be overwritten for every intersection session which occurs in the automatic or manual mode. The contents of the intersection file are in term of the x,y,z-coordinates of the wetted perimeter of the aircraft, computed at specified x-intervals (in the case of automatic intersection), or at arbitrary x-locations (in the case of manual intersection). It is up to the user to decide whether or not to save the data created at the time of intersection. By invoking the SAVE FILE menu item the user can rename the file in both type and name and thereby recognize the specified file in the file listing after exiting the Hess Intersection program.

3.0 INTERSECTION IN ACSYNT/VPI

3.1 Goal

This chapter describes how the cross-sectional area calculation procedure developed for use with Hess files has been incorporated into ACSYNT/VPI. The object of the code discussed in this chapter is to define and compute the area of intersection between the aircraft and a plane located at any position along the x-axis and rotated at any angle about the y-axis. While perhaps not technically correct use of the word, these areas of intersection are also referred to as "cross-sections" in this thesis. The model of the aircraft contains components which overlap; therefore, the intersection data, which consist of an array of points, will also contain overlapping data. This means that the intersection algorithm requires user assistance to determine the local wetted perimeter of the aircraft. As before, a conventional aircraft coordinate system is assumed wherein the x-axis defines the roll, the y-axis the pitch, and the z-axis the yaw. The description of the code presented in this chapter is aimed at the reader interested in its modification or extension. For example, it will probably be desirable in the near future to automate the "user-assisted" proce-

dure for eliminating overlap of components. Eventually, cross-sectional area distribution calculations should be as automatic in ACSYNT/VPI as they currently are with Hess files. These refinements, however, are beyond the scope of the present thesis. For specific information on how to use the intersection portion of ACSYNT/VPI the reader is referred to the user's guide, Appendix D.

3.2 *Design Philosophy for Menu Modules*

Most of the discussion contained in Section 2.2, which adresses the requirements of the workstation, is also applicable to the discussion of the ACSYNT/VPI code. It is, however, recommended that those readers not familiar with the design and use of this latter code refer to S. Wampler's thesis [2] for necessary background information.

The coding of the menu modules for the intersection portion of ACSYNT/VPI follows the structure of the menu-based design developed by Wampler. Menu-based design refers to the philosophy that all pickable items on the screen, including the aircraft geometry and views, are treated as menu types [2]. It serves as a standard method for determining, based on a pick, the type of action to perform: execute a menu item, highlight a geometric component, or change the current view.

3.3 *Menu Design for ACSYNT/VPI Intersection*

The first step in the design phase for ACSYNT/VPI's intersection modules was to define the command menus. Before coding could begin, it was necessary to establish the design philosophy.

Also, considerable thought had to be given to deciding where in the code the intersection module should be appended. In this decision-making process, examination of the current modules proceeded starting with the top-most menu, MAIN MENU. The modules which comprise MAIN MENU are:

GEOMETRY
TRAJECTORY
AERODYNAMICS
PROPULSION
STABILITY
WEIGHTS
TAKE OFF AND LANDING
DESIGN
FILE

the GEOMETRY module was eventually chosen as the source for the intersection module. This is because the aircraft configuration is no longer visible once the user exits the GEOMETRY module. In order for the user to better visualize the location and orientation of the cutting plane with respect to the aircraft, it is necessary for the aircraft geometry to remain visible during the intersection process.

Extending this reasoning to the other analysis tools, a menu item named ANALYSIS has been added to the GEOMETRY menu. Figure 8 illustrates the possible paths through the menu structure of the program which lead to the intersection algorithm. When selected, ANALYSIS brings to the command portion of the menu screen the following menu items, the first three of which are not presently active under ACSYNT/VPI:

DRAG CONE AREAS

- computes surface area defined by the locus of all points common to a conical intersecting surface and the aircraft.

SURFACE AREAS

- calculates the total wetted surface area of the aircraft.

VOLUMES

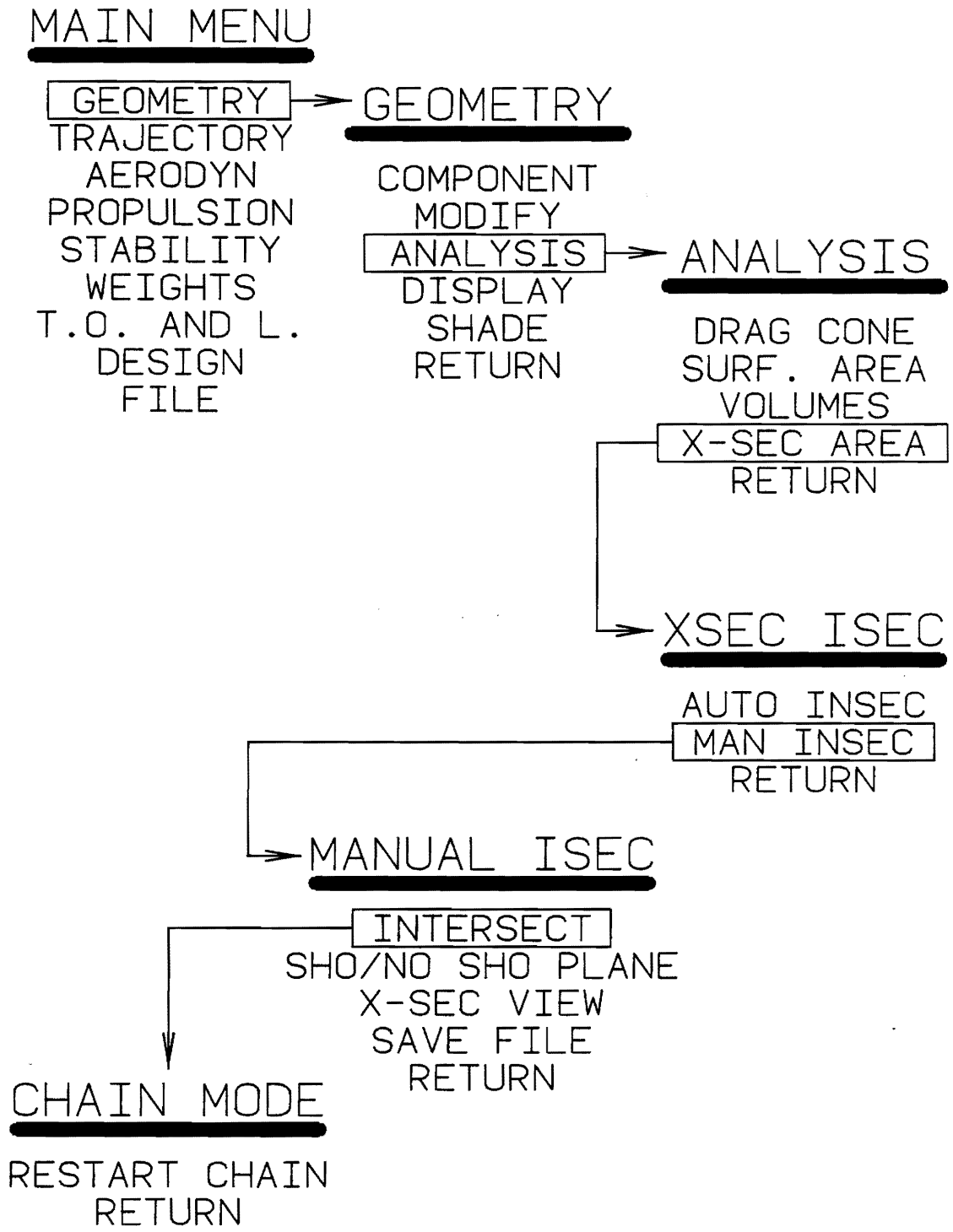

Fig. 8. Flow of Program Through the Intersection Menus.

- determines the volume of the aircraft.

X-SEC AREAS

- computes the area enclosed by a locus of intersection points which are determined by a planar intersection method.

Only the X-SEC AREAS menu item is currently executable. The other three are presently stub modules that return a message informing the user that they are not yet available. From the X-SEC ISEC menu, only two menu choices are presented. They are:

AUTO INSEC

- computes the locus of all points common to a cutting plane of variable orientation and the aircraft surface. The automatic portion of this module computes these intersections at intervals along the length of the aircraft which are predefined by the user. The area defined by this closed curve will be calculated later based on the points of intersection.

MAN INSEC

- computes the locus of all points common to a cutting plane of arbitrary orientation and the aircraft surface. User-assistance is required to position and orient the cutting plane, to determine the curve defined by this intersection, and to compute the area which it encloses.

Now consider the items within the MAN ISEC (manual intersection) menu. In the context of this code the term "Manual Intersection" means that user-assistance is required for the calculation process to proceed. Once the MAN ISEC module has been entered, the following options are available:

INTERSECT

- computes the locus of all points which define the intersection between the aircraft surface and the plane of intersection.

X-SEC VIEW

- allows the user to reposition the location of the default window which contains a front view of the aircraft before INTERSECT, and the locus of intersection points after INTERSECT.

SHO/NO SHO PLANE

- renders the intersection plane invisible if currently visible, and visible if currently invisible.

SAVE FILE

- enables the user to specify a filename and filetype for a file containing intersection point data.

With one exception none of the menu items previously listed as contained in the MANUAL INSEC menu invoke any submenus; instead they themselves perform the task described. The one exception is the INTERSECT menu item. This menu item can be considered as the most important due to its function. Once selected, INTERSECT puts the program into CHAIN MODE. The term "chain" refers to the method in which the region defined by points of intersection is enclosed before area calculation is performed. The user selects a starting point from which to begin chaining, then moves to another point and selects it, thereby drawing a straight line segment between the two consecutive points. This process is repeated until the "chain" enclosing the region is complete and the area can be calculated using the same technique as in Hess Intersection. CHAIN MODE is comprised of only one menu item:

RESTART CHAIN

- allows user to start the chaining process again by erasing all previously chained line segments and choosing a new point from which to begin a new chain.

3.4 Manual Intersection

Now that the menu path which leads to the manual intersection module has been established, the design of the manual intersection module can be described in detail. The manual intersection module is the first active module contained in the ANALYSIS/X-SEC AREA/ submenu. Its menu heading is entitled MANUAL ISEC. All other menus leading to MANUAL ISEC simply invoke other command menus or messages of information. The consequences of selecting MAN INSEC in the X-SEC AREA menu are:

1. Numerical input is established for control of the intersection plane.

2. A default window appears which contains a view normal to the plane of intersection.
3. The plane of intersection is visible in a default location which is situated in the y,z-plane at the origin of the coordinate system of the aircraft.
4. The numerical input is initialized as 0 along the x-axis and 0 degrees of rotation about the y-axis.
5. A submenu containing the options for intersection, default window relocation, plane invisibility, and save file is created.

There are two parameters which may be changed to manipulate the intersection plane: the location of the plane along the x-axis and the orientation of the plane with respect to the y-axis. These two parameters affect the local transformation matrix of the intersection plane. In this case, the local transformation matrix changes the orientation and location of the defined intersection plane. The plane, defined initially as bounded and lying in the y,z-plane, is multiplied by translation and rotation values contained within the matrix which locate the plane in the proper position as determined by the user through numerical input. The current values of translation and rotation are stored in the array which contains the valuator values. The location or rotation of the plane may be changed at any time when inside the MANUAL ISEC menu. This is possible due to event mode input which recognizes the valuator or string input as an event, and when evaluated produces a new transformation matrix which affects the postion of the intersection plane.

3.4.1 Intersection

The function of the manual intersection module is to perform an intersection between a plane and the aircraft and calculate the area of the region enclosed by the points of intersection as determined by the user. Upon choosing the INTERSECT option within the MANUAL ISEC menu, a locus of intersection points is computed. In order to fully understand how the intersection is effected, a brief discussion on the geometry of the aircraft described by ACSYNT/VPI is needed.

The geometry displayed by ACSYNT/VPI is derived from a component cross-section model which defines each component of the aircraft as a series of parametric cross-sections. An illustration

of a cross-sectional description of an aircraft is shown in Fig. 9. The program then generates a surface model of the aircraft by lofting surface patches across the cross-sections which define each component. Lofting is achieved by fitting a parametric surface patch between two consecutive cross-sectional descriptions. Reference [2] provides a complete discussion of the lofting techniques which applies to this modeling process, beginning on page 131. The final aircraft model is composed of a collection of these patches which define the surface. The problem with using this surface model to compute planar intersections is that no provision has been made in ACSYNT/VPI for determining surface intersections among the various components. The significance of this is that when a component intersects another component, for instance a wing and the fuselage, it is not possible to determine which surface patches are exterior to the aircraft and which are interior. Therefore, if intersected by a plane, the resulting cross-section would contain not only a closed curve on the exterior surface of the aircraft, but also line segments internal to the aircraft. This produces difficulties when calculating the area of the intersected region.

In view of the difficulty described above, a second method was considered for computing areas of intersection. This method involved defining the intersection between a plane and a wireframe representation of the aircraft. The wireframe aircraft model generated by ACSYNT/VPI is related to the surface model of the aircraft. It begins with the same cross-sectional component representation, but instead of lofting surfaces between the cross-sections, the wireframe model approximates the curves of the surfaces with polylines. The coordinates of the polylines are determined from the surface model of the aircraft, since the data given in ACSYNT contain no information about surfaces. The problem associated with using this wireframe model as the basis for intersection calculations is that no database contains the polyline coordinates. If there is no database, then there is no information with which to compute an intersection. In the interest of conserving virtual memory capacity it was decided that any attempt to create another database should be avoided if at all possible. (Virtual memory refers to the amount of memory which "appears" to be accessible to the user. Current virtual memory capacity for the IBM 4341 used in development of this program is 16 MegaBytes with 12 MB available to applications programs).

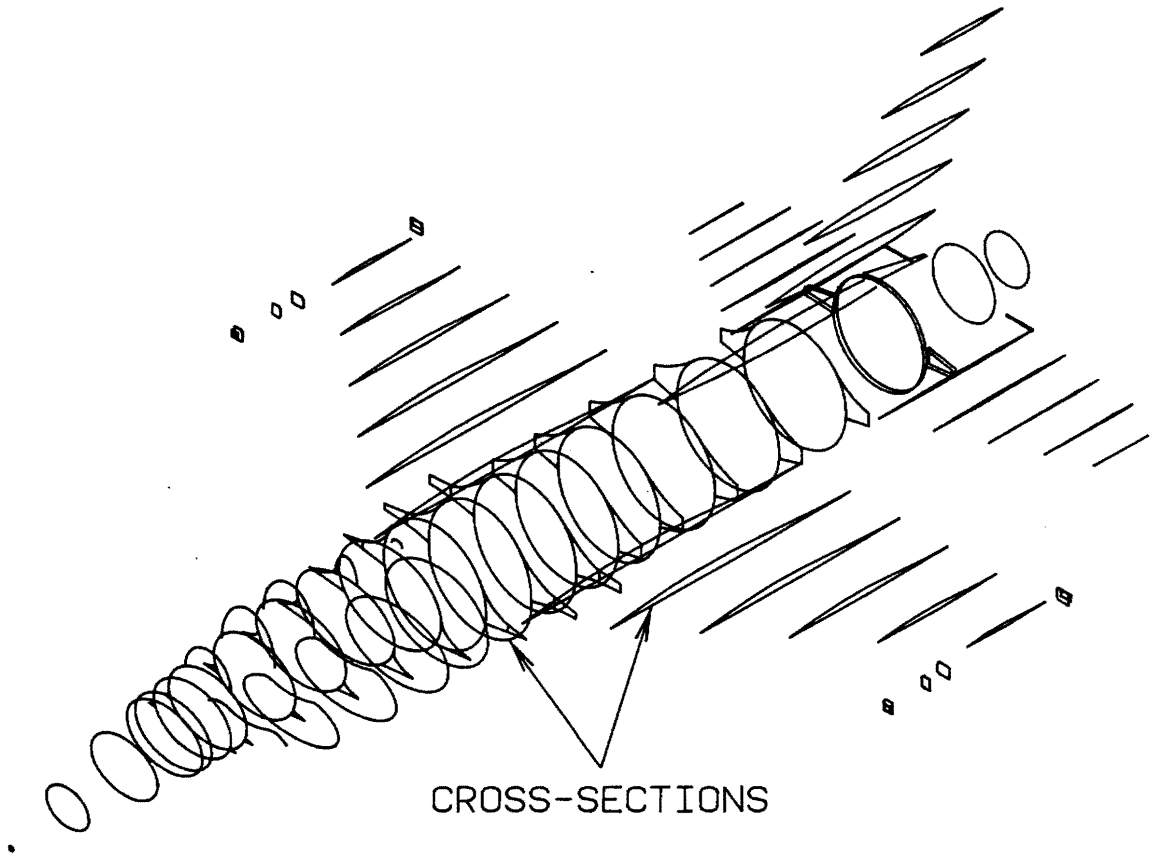

Fig. 9. Cross-Sectional Representation Of An F-16 In Terms Of Components [2] .

A solution to this dilemma is to utilize the existing tiling database to obtain information with which to compute the intersection. "Tiling" refers to the process in which polygons are generated on the surface of each component for purposes of shading. By calling a routine named XSTILE, the aircraft geometry is fitted with polygons and the tile data blocks are loaded. The common block which contains these data is /POLDAT/. The main array of interest within /POLDAT/ is XYZSUB, which is a three-dimensional array containings information on each polygon in the model in terms of the x,y,z-coordinate of each vertex of the polygon. By considering only the edges of the polygons, the intersection method developed in Hess Intersection may be used to calculate the intersection between a line segment and a plane.

As in the Hess Intersection code, a method designed to determine which line segments are likely candidates for intersection was developed. It utilizes the fact that although the intersection plane may not lie perpendicular to the x-axis, the edges of the plane may be projected to form a bounded region within which intersection must occur. The projected intersection boundary is presented in Fig. 10. The figure shows clearly that any line segment, formed by the connection of two polygonal vertices, which does not intersect the boundary volume also will fail to intersect the plane. Each polygon which tiles the aircraft must be tested for intersection, one at a time, following the search procedure outlined below:

1. Connect consecutive vertices of the current polygon and check to see if each edge of the polygon intersects the boundary volume.
2. If a line segment intersects the boundary volume, then a secondary test is performed which determines if the segment will intersect the plane.

The test mentioned in Step 2 above consists of calculating the scalar product of a vector from a point in the center of the intersection plane to a segment endpoint, and a vector which lies on the plane itself. The vector test method is illustrated in Fig. 11. The product is computed for both endpoints. If both products are of the same sign then the two points are on the same side of the intersection plane and no intersection with the plane is possible. Referring to Fig. 11,

$$\text{sign}\,(\mathbf{V}_C \cdot \mathbf{V}_{C1B}) = \text{sign}\,(\mathbf{V}_C \cdot \mathbf{V}_{C2b}) \quad . \tag{3.1}$$

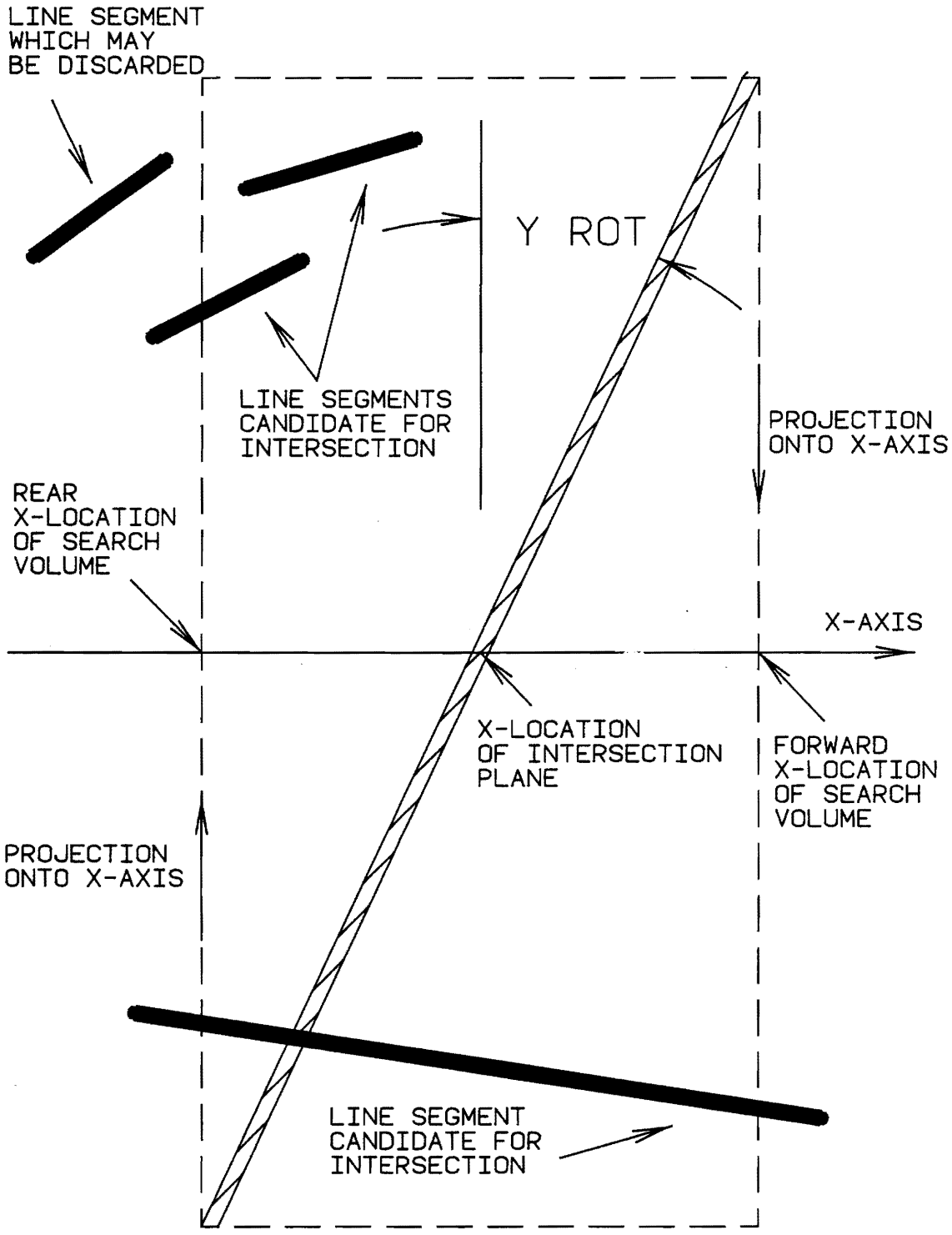

Fig. 10. Sample Search Volume of Intersection Plane.

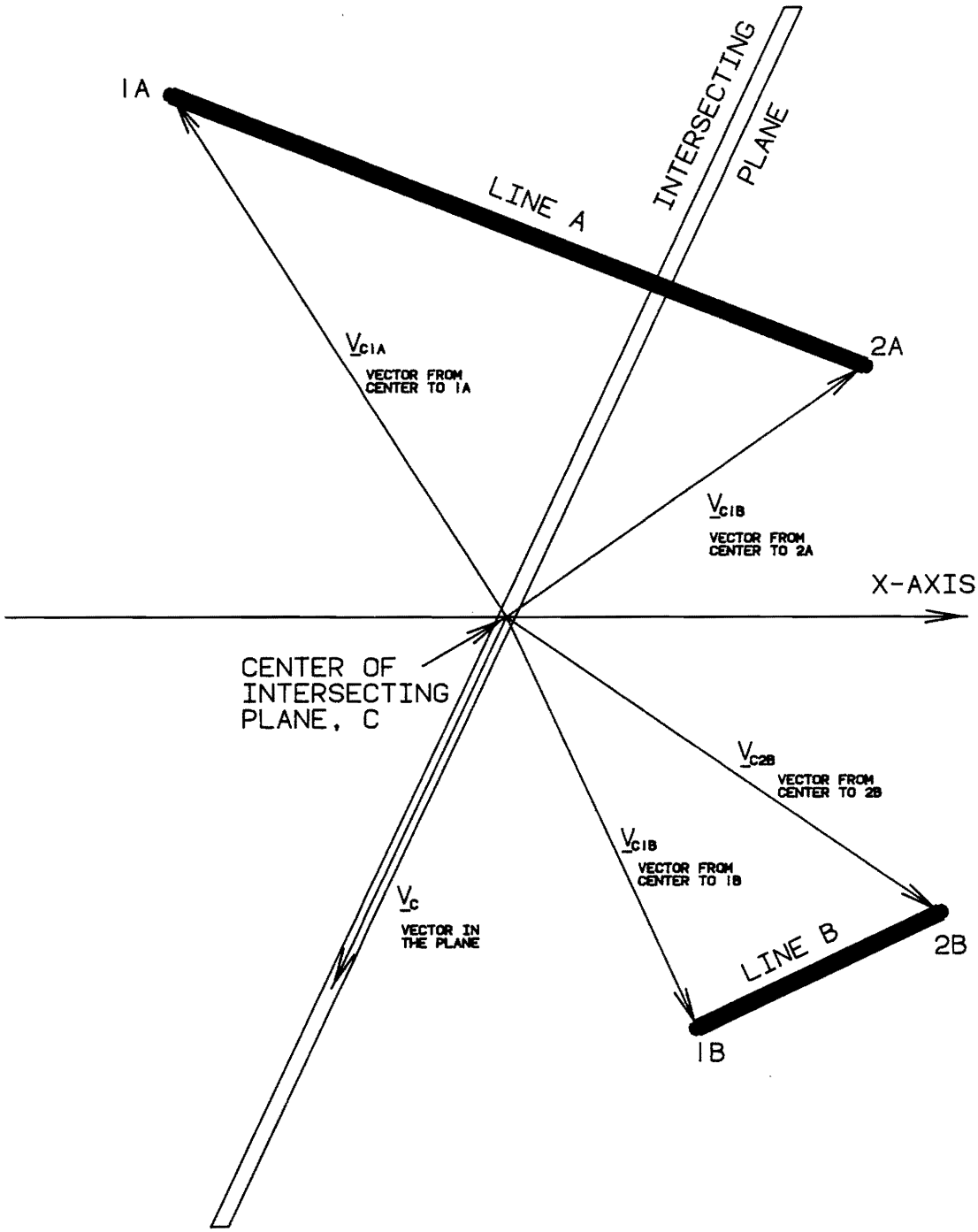

Fig. 11. Scalar Product Comparison to Determine Intersection Probability.

If the products are of opposite sign an intersection will occur as seen in Fig. 11 where

$$\text{sign}\,(\mathbf{V}_C \cdot \mathbf{V}_{C1A}) \neq \text{sign}\,(\mathbf{V}_C \cdot \mathbf{V}_{C2A}) \tag{3.2}$$

means the line segment and the plane intersect. And if the products are equal to zero, the line segment lies in the plane of intersection and the two endpoints are returned as the points of intersection. For the case of an intersection plane with zero degrees of rotation with respect to the y-axis, the boundary volume test, Step 1 above, is bypassed and only the secondary scalar product test is performed.

The problem of intersection area computation is handled in the same manner as described in Section 2.8. It is interesting to note at this point that the user may increase or decrease the density of the intersection points which will be present during the chaining process by re-tiling the aircraft with more or fewer tiles. In this case, re-tiling must be done before entering into the MANUAL ISEC menu.

3.4.2 Chain Mode

When all line segment intersections with the plane have been identified for a given position and orientation of the intersection plane, the user enters CHAIN MODE. "Chain mode" refers to the method by which the user assists the program in determining which sequence of intersection points represents the perimeter of the aircraft. Once CHAIN MODE is invoked the default view containing a front-view of the aircraft geometry is replaced with a locus of points which represents the intersection between the cutting plane and the tile edges. The chaining algorithm prompts the user to select a starting location for a "chain" which will enclose a region whose area will be calculated upon completion of the chain. This area will be approximately equal to the area of intersection between the intersecting plane and the aircraft. When a point is selected, the CHAIN

routine creates an intermediate holding array which acts as a pointer into the intersection array. The pointer acts as an index into the array much like a bookmark saves the current page in a book. The CHAIN routine also connects the selected point with the previous point in the holding array with a line segment. A counter is employed to keep track of the number of points which comprise the chain. The counter increments for every new chain point chosen, and decrements when a chain point or line segment is chosen which is the same as the most current point or line segment in the chain. In this manner the user is able to retract a bad choice and continue. The method the routine uses to keep track of which point has been selected may be outlined as follows:

1. Upon completion of the intersection, each point in the intersection array is given a pick identification number before it is displayed.
2. A root menu is created of type other than command, geometry, view, or standard. When an element, in this case an intersection point, is chosen from this new menu, a number is returned in the pick path which identifies the menu type. The pick path informs the program as to what menu the element came from and the identification number of the element. This is a feature of the menu-based design process and is discussed thoroughly in [2] in terms of creation of menu roots. In this manner the program can determine if the points within the intersection default view were chosen.
3. When a point is selected, a counter containing the number of picked points is incremented by one. An intermediate array which contains the pick identifier of the intersection point chosen is formed as a function of the counter which keeps track of the number of points in the chain. The current count number is the input into the intermediate array, and pick identification number is the output.
4. If the chain point counter is of value greater than one, the program obtains the pick identifier for the previous point. The pick identifier corresponds to the location of the point within the intersection array. Since two consecutive points and their coordinates are now established, a line segment joining them may be drawn. The line segment receives the pick identifier of the

previous point in the chain and can therefore be erased if chosen after being drawn. Refer to Fig. 12 for an example of this sequence.

5. If the user wishes to delete a line segment or a series of line segments in order to reach a point from which to restart all that needs to be done is to retrace the steps back through the sequence until a satisfactory point has been reached. Alternatively, if the user chooses the same point that was just selected, the line would be erased and the point would be deleted from the hold array.

Upon close inspection of the intersection points during the chaining process, one may notice that what from a distance appears to be a single point may actually be a pair of intersection points. This is due to the fact that during the tiling process many of the tiles do not meet at a common edge, but instead overlap. Therefore, when the tiled aircraft surface and the cutting plane are intersected, there exists the possibility that two points representing the same intersection will lie close together. This does not present a real problem when chaining since either of the two points will produce an area well within the accuracy of the method. The only possible problem this could cause is that, if the first point chosen is not the same as the last point chosen, the perimeter will not be closed and the area will not be computed. In this case it is easier to select the first line segment in the chain since its pick identifier is the same as that of the first point. In this manner the chain is closed and the area is calculated. Upon closure of the chained region, the program calculates the area of the polygon formed in the same manner as discussed in Section 2.8.

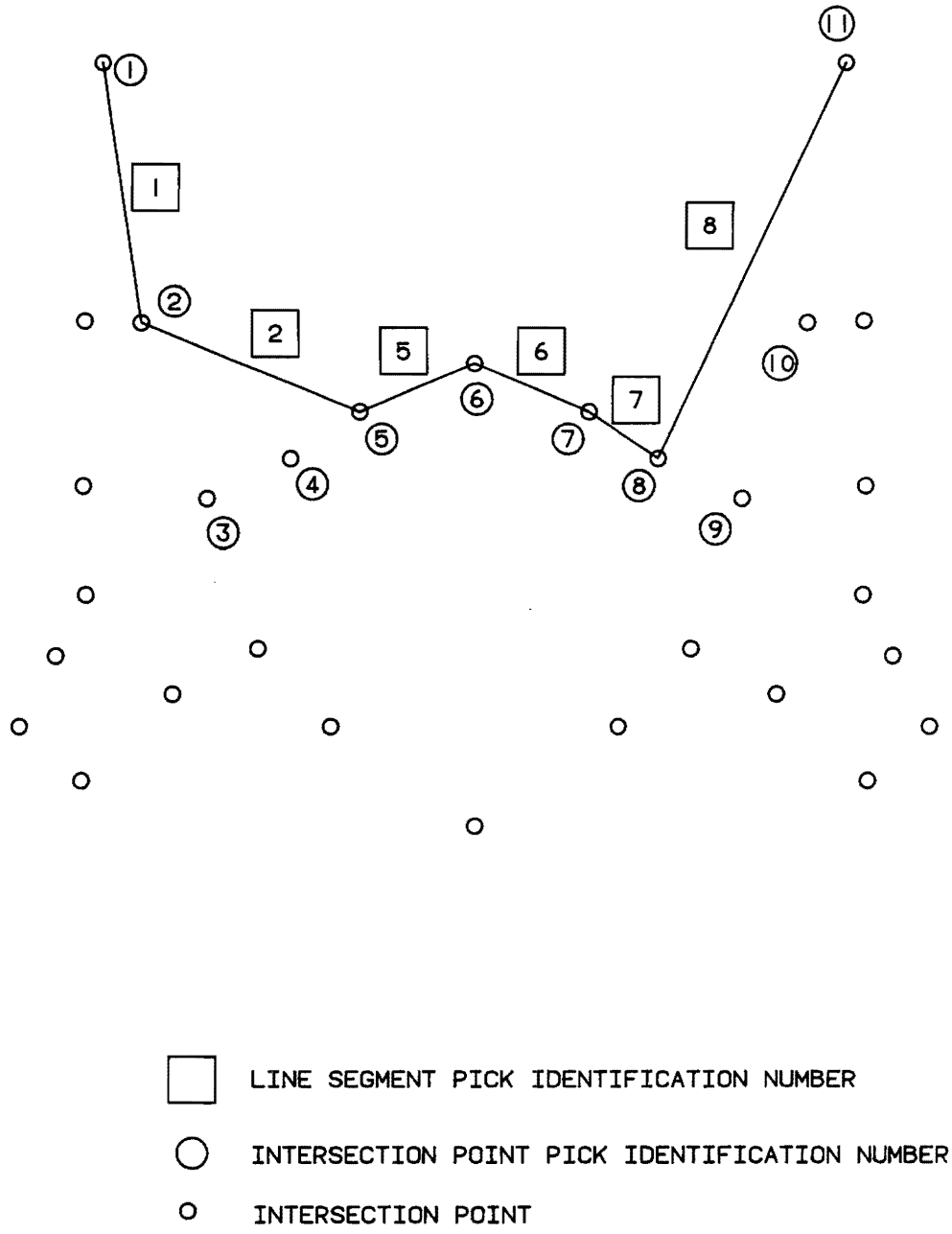

Fig. 12. Sample Chaining Sequence Including Pick Identification Numbers of Points and Lines.

4.0 RESULTS

4.1 Overview

This chapter illustrates the functional capabilities of both the Hess Intersection and ACSYNT/VPI intersection codes.

4.2 Results for Hess Intersection

Recall that in Chapter 2 the goals of the Hess Intersection program were stated to be:

- To display an aircraft described by data provided in Hess format.
- To restructure the data in a manner compatible with most CFD algorithms.
- To compute the cross-sectional area for each network of the aircraft lying in a given cross-section, as well as for the entire aircraft cross-section.

- To display the cross-sectional areas for each network within a cross-section as well as the total cross-sectional area as a function of position along the x-axis. This is known as the area distribution of the aircraft.

All of the goals listed above have been achieved. It is important to remember that the file containing intersection data, which are created either by automatic or manual means, only represents half of the aircraft. If needed, data describing the entire aircraft are easily created due to the symmetry of the aircraft. Once again, assuming the conventional aircraft coordinate system, the intersection data may be reflected with respect to the x,z-plane, across which symmetry of the aircraft is assumed, simply by copying the data for a given cross-section and changing the sign of the y-coordinate for every point which lies in the cross-section. In this manner a data file representing the entire aircraft is created.

The menu layout facilitates the use of the program by providing all menu options at the same level. This means that when a menu item is selected, no sub-menu structure is invoked; rather, the function is performed immediately. In this manner, problems concerning the depth of the menu structure are avoided. For example, the user need not dive several menu levels deep to perform a specific task. Thus, the user is less likely to experience any problems which may arise from having to find his way back to the main menu.

Figure 13 shows the menu screen containing an F-15 fighter aircraft. Note the reduction in fuselage area in the region of the wings due to area ruling. The area distribution for the F-15 of Fig. 13 is displayed in Fig. 14. Each color in the area distribution graph represents a separate network. Networks one through six comprise the fuselage of the aircraft, seven the wings, eight and nine the horizontal stabilizers, and ten the vertical stabilizers. The displayed data result when the automatic intersection mode is invoked using increments of ten inches along the aircraft axis. The aircraft and its corresponding area distribution are shown together in Fig. 15.

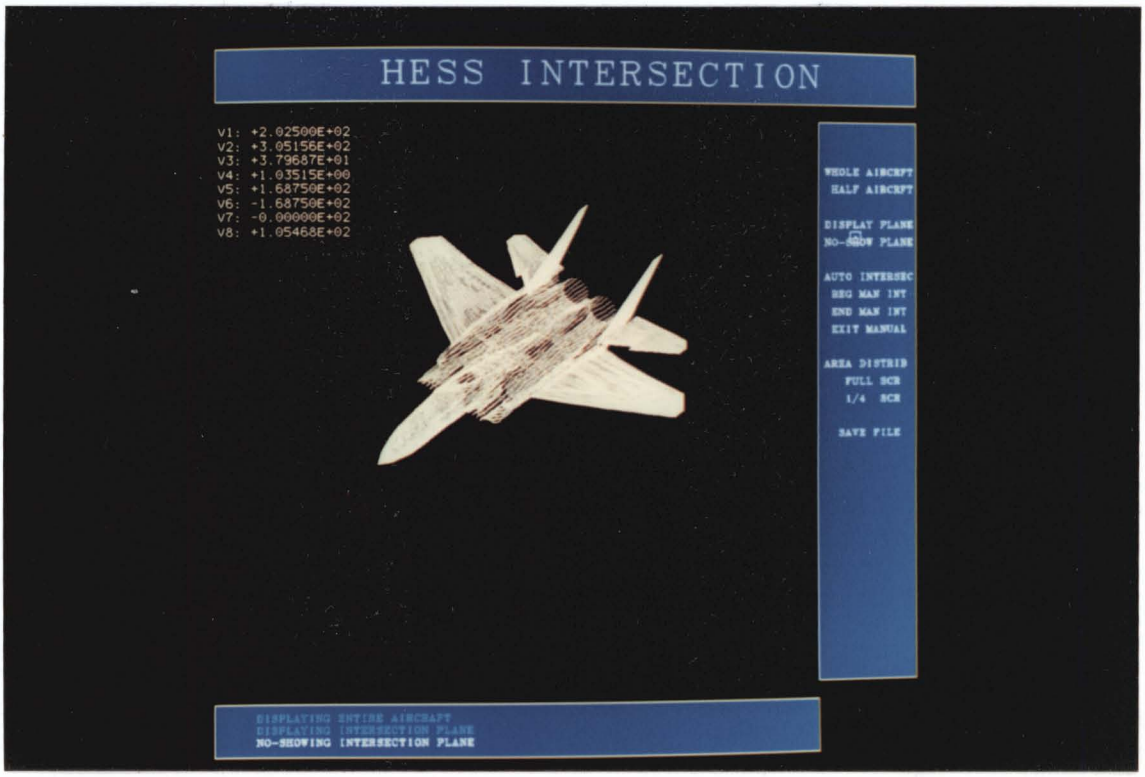

Fig. 13. Hess Intersection Menu Screen Displaying an F-15 Aircraft.

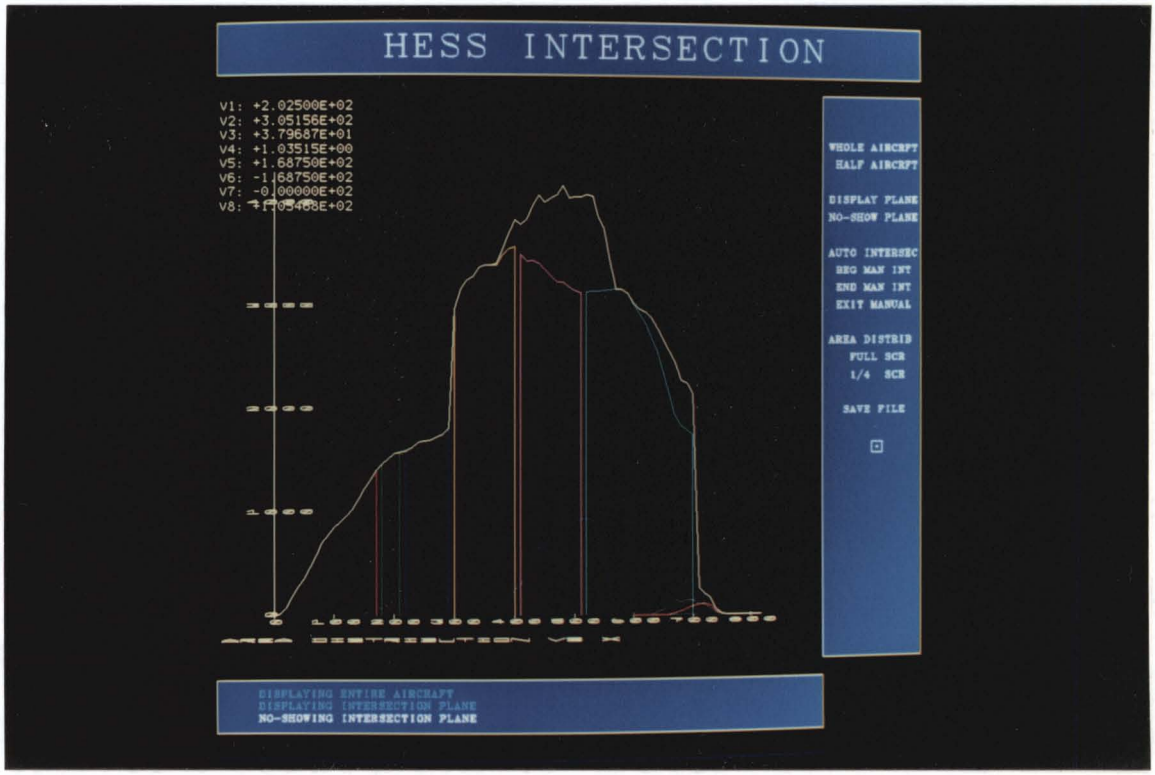

Fig. 14. Area Distribution of an F-15 Aircraft.

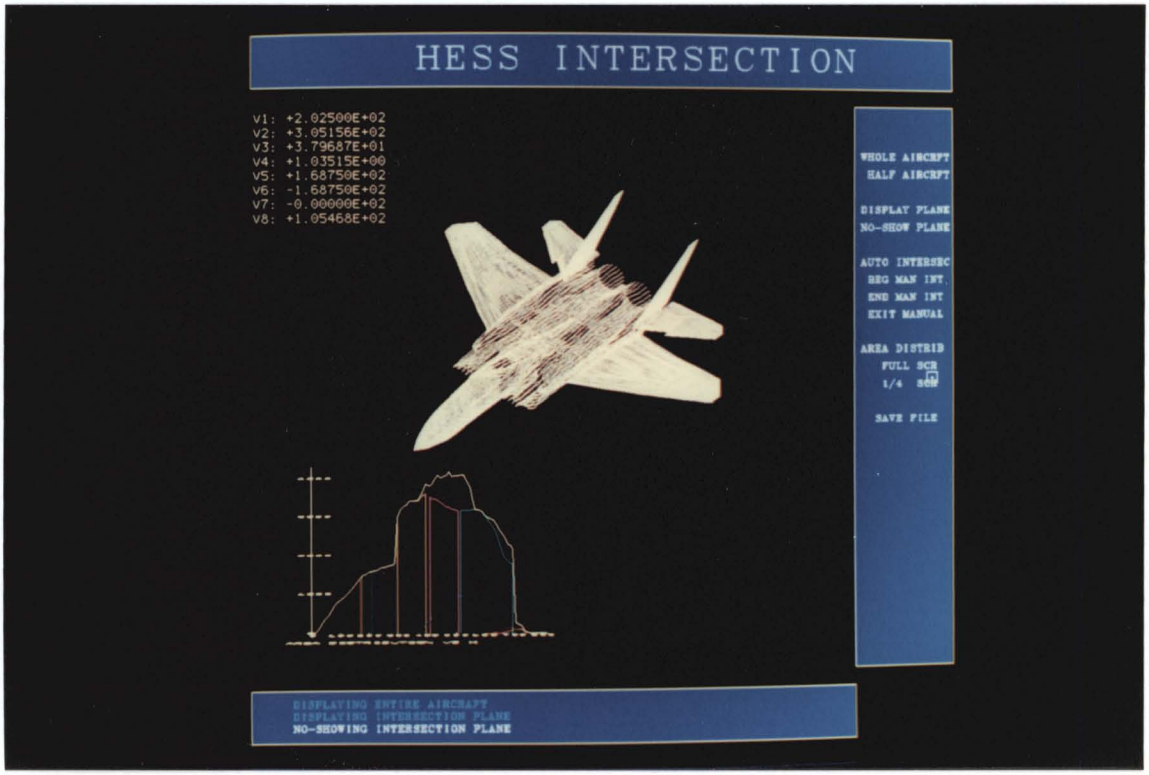

Fig. 15. F-15 Aircraft In Shown With Its Area Distribution.

4.3 Results for ACSYNT/VPI

The intersection capabilities incorporated into the ACSYNT/VPI program provide the user with one more piece of information during the design procedure. The MANUAL INTERSECTION module allows the user to interactively determine the area of intersection between a plane tilted at any angle with respect to the y-axis and the aircraft at any location along the length of the aircraft. The area may be computed for one component or for the entire perimeter of the aircraft cross-section, as specified by the user. This information is useful for determining which portions of the aircraft are likely candidates for reduction in cross-sectional area in order to reduce the drag on the aircraft. As previously discussed, the area distribution of the aircraft cross-sectional area as a function of station along the x-axis directly influences the amount of wave drag experienced during transonic and supersonic flight. In Fig. 16, the locus of points formed upon intersection is displayed in the lower left window of the screen. It is now up to the user to determine which set of points represents the perimeter of the aircraft. The designer does this by chaining the points together until the region of interest is closed, after which the code automatically computes the enclosed area. An example of the chaining process and area calculation is given in Fig. 17. The area for separate components may be computed by enclosing one component, calculating its area, and then selecting the RESTART CHAIN option which enables the user to begin anew on another component.

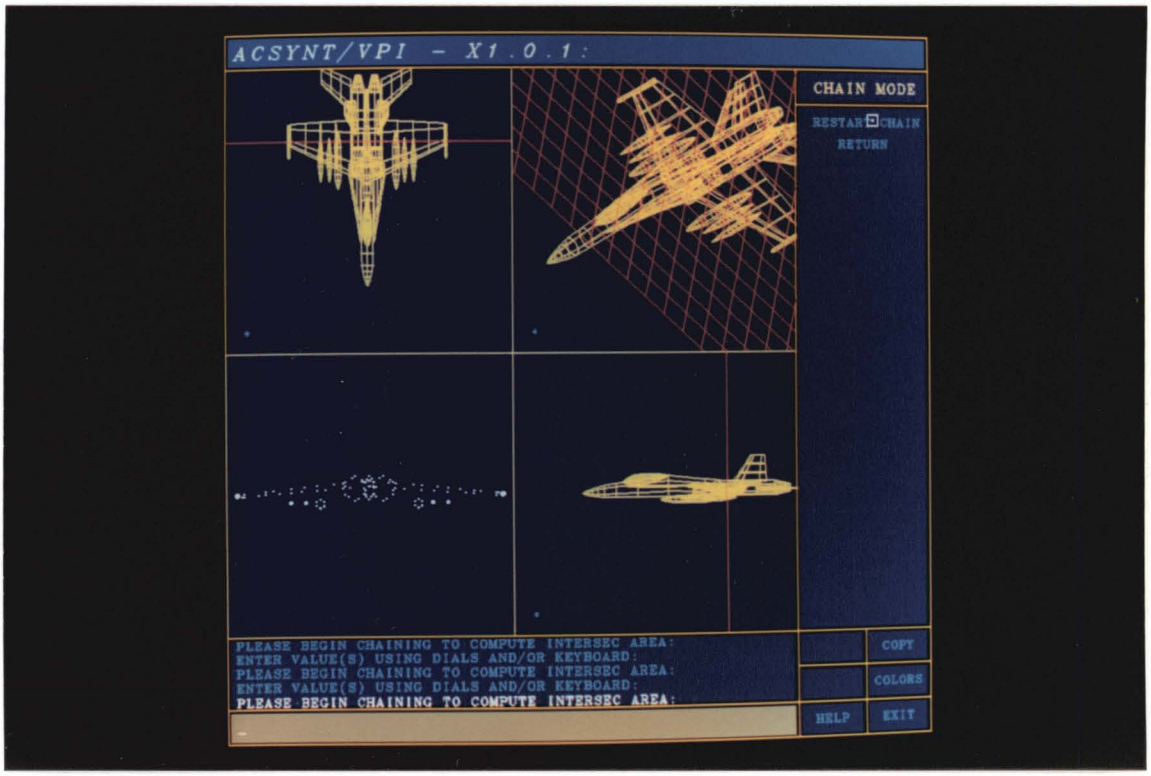

Fig. 16. Intersection Point Data Displayed For An F-18.

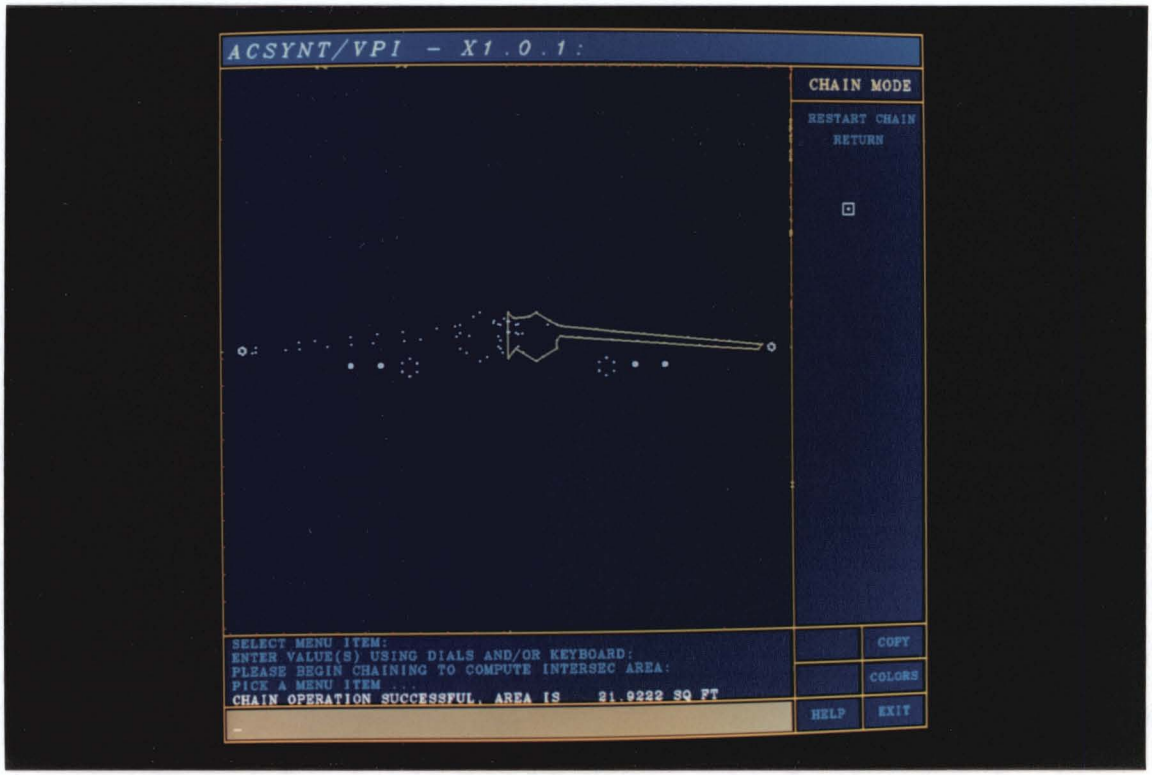

Fig. 17. Enclosed Cross-Sectional Region and Resulting Area.

5.0 RECOMMENDATIONS

5.1 Overview

This chapter presents recommendations for further development of the Hess and ACSYNT/VPI intersection codes.

5.2 Recommendations for Hess Intersection

The following recommendations are made for the improvement of the Hess Intersection code.

1. A testing sequence should be developed to check the flexibility of the code. During the development of the code, only one Hess file was used as a test case. Because of this, other file

data may contain node, cross-section, or network numbers which exceed the present capacity in the array which stores the Hess data for the program.

2. A method to decrease or to at least better utilize the current arrays which store Hess data should be developed. This is necessary in the case where a merger between the Hess file read and display module and ACSYNT/VPI is considered. Since Hess Intersection was intended to be a stand-alone program, there really was no need to minimize the required virtual memory space during its development. However, considering the present size of ACSYNT/VPI, it is not certain that a Hess display module would fit easily into the current structure of the code.
3. A view perpendicular to the axis of intersection should appear in the manual intersection mode. This would allow the user to visualize the intersection and enable him to make an informed decision as to whether or not this intersection contains valuable information.

5.3 Recommendations for ACSYNT/VPI

The following recommendations are made for the improvement of the ACSYNT/VPI code.

1. High priority should be given to providing an automatic intersection algorithm. This algorithm should include some means for calculating the area of the cross-section as defined by its cross-sectional perimeter. This may be achieved by addition of component area, but this approach will yield an exaggerated area due to overlapping components as, for example, in the case of the wings and the fuselage. Ideally, determination of these perimeters should be achieved by some type of surface/surface intersection which would allow only the true surface of the aircraft to be considered in the cross-sectional area calculations.
2. Area distribution should be added to the capabilities of the program. The data for area distribution are best acquired using an automatic method for determining intersection areas. This is because a minimum number of intersections is guaranteed using this method, whereas in the

manual intersection mode the user normally computes only one or two series of intersections, which is certainly not sufficient for an accurate area distribution. Once the data for an automatic series of intersections are obtained, a manner for storing the area data, both by component and location along the x-axis, is required. Once this is accomplished graphical display of area distributions is possible.

3. Cone intersection areas should be computed in addition to planar intersection areas. Supersonic wave drag computational algorithms require this type of data as opposed to simple planar intersections. As an approach to this problem, one may consider treating the cone of intersection as approximated by an envelope of planes whose areas are suitably averaged. The intersection problem which remains is that of two intersecting bounded planes (polygon/polygon).
4. Calculation of the external (surface) area of the aircraft should also be implemented. In essence, the solution to this problem is already available. When the aircraft is tiled, the entire surface is covered with polygonal approximations of the surface. These polygonal data are then stored in the common block called /POLDAT/. All one need do is access this block of information and loop through each polygon, each time computing its surface area by means of the AREA algorithm, written by J. Sankar, which appears in Appendix C. When all of the polygonal areas are added together, the aircraft surface area results. There is a problem that all polygons which tile the aircraft are not necessarily external to the aircraft. This fact would tend to exaggerate the actual surface area of the aircraft. However, this problem will be eliminated once a method for determing intersections between surfaces of the aircraft is developed. This suggests one final recommendation.
5. An algorithm should be developed which computes intersections between the components of the aircraft. Presently, no such module exists which can ensure tangential continuity between intersecting surfaces of components. This is a problem in the calculation of the surface area of the aircraft since many components overlap. In this case, computation of the surface area based on component surface area will be produce an exaggerated estimate.

6.0 REFERENCES

1. Halsey, N. D., and J. L. Hess, "A Geometric Package for Generation of Input Data for A Three-Dimensional Potential Flow Program", **NASA CR-2962**, June, 1978.

2. Wampler, S. G. , **Development of a CAD System for Automated Conceptual Design of Supersonic Aircraft**, Thesis -- Master of Science in Mechanical Engineering, VPI&SU, Blacksburg, Virginia, May 1988.

3. Walters, R. W., T. Reu, J. L. Thomas, and W. D. McGrory, **Zonal Techniques for Flowfield Simulation About Aircraft**, to be presented at the Symposium on Advances and Trends in Computational Structural Mechanics and Fluid Dynamics, October 17-19, 1988.

4. Whitcomb, R. T. "A Study of the Zero-Lift Drag Rise Characteristics of Wing-Body Combinations Near the Speed of Sound", **NACA RM L52H08**, 1952.

5. Anderson, John D. Jr., **Fundamentals of Aerodynamics**, McGraw Hill Book Co., New York, 1984.

6. Bertin J. J., and M. L. Smith, **Aerodynamics for Engineers**, Prentice Hall, Englewood Cliffs, N.J., 1979.

7. Jones, R. T., "Theory Of Wing-Body Drag At Supersonic Speeds", **NACA Report 1284**, 1953.

8. Ashley, H. and M. Landahl, **Aerodynamics of Wings and Bodies**, Dover Publications, Inc., New York, 1965.

9. Raymer, Daniel P., "CDS Grows New Muscles," **Astronautics and Aeronautics**, June 1982, pp. 22-31.

10. Smetana, Frederick O., **Computer Assisted Analysis of Aircraft Performance, Stability and Control**, McGraw Hill Book Co., New York, 1984.

11. Lu, Liang-Ju, **HESCAD - An Interface Between HESCOMP and CADAM for the Generation of Helicopter Models**, Thesis -- Master of Science in Mechanical Engineering, VPI&SU, Blacksburg, Virginia, December 1985.

12. Schoen, A. H. and J. S. Wisniewski, "User's Manual for VASCOMP, The V/STOL Aircraft Sizing and Performance Computer Program", Boeing Vertol Company, **Report D8-0375**, Volume VI, Rev. 2, September 1973.

13. Myklebust, Arvid, **Computer Aided Design I** ,VPI&SU, Class notes published by Kinkos Copies, Winter 1987.

14. Brown, Maxine D., Michael Heck, Editor, **Understanding PHIGS**, Megatek Corporation, 1985.

15. Tausworthe, Robert C., **Standardized Development of Computer Software - Part I Methods**, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1977.

16. Mortenson, M. E., **Geometric Modeling**, John Wiley & Sons, New York, 1985.

17. Sankar, Jayaram, Personal communication concerning development of the AREA algorithm, VPI&SU, May, 1988.

APPENDIX A. HESS INTERSECTION USER'S GUIDE

A.1 Display of a File In Hess Format

Purpose:

to allow the user to read in and display an aircraft described in Hess format.

Description:

Upon entering the Hess Intersection program, the user is prompted to enter the name of the desired file.

```
_
_
ENTER NAME OF FILE:
```

Any file the user wishes to analyze must be of filename FILE. The filetype may be any descriptive name for the file. The user need only enter the filetype since FILE is considered the default filename.

If the program cannot locate the file entered by the user, a message returns which states that no such file exists and another name is requested.

A.1.1 Manipulating the Aircraft and Intersection Plane

Purpose:
to allow the user to rotate and translate the aircraft and to reposition the intersection plane along the length of the aircraft.

Description:
Both the aircraft and plane of intersection can be manipulated using valuators to control their position. Valuators 1-7 affect the aircraft and valuator 8 the intersection plane. Their functions are as follows:

Valuator 1

- rotates the aircraft about the axis of roll (x-axis).

Valuator 2

- rotates the aircraft about the axis of pitch (y-axis).

Valuator 3

- rotates the aircraft about the axis of yaw (z-axis).

Valuator 4

- enables user to magnify or diminish the size of the viewed aircraft.

Valuator 5

- translates the aircraft in the x-direction on the screen.

Valuator 6

- translates the aircraft in the y-direction on the screen.

Valuator 7

- translates the aircraft into and out of the screen in the z-direction.

Valuator 8

- translates the intersection plane along the x-axis of the aircraft.

A.1.2 Displaying the Entire Aircraft

Purpose:
to allow the user to view the entire aircraft if the data contained in the Hess file represents only one-half of the aircraft.

Menu Path:
WHOLE AIRCRFT

Description:

Once the desired file has been read in, the aircraft is displayed on the screen. Typically, the Hess file only contains data which describe half of the aircraft, assuming that the aircraft is symmetric with respect to the x- or roll-axis. The resulting partial aircraft can be beneficial when relocating the aircraft since there are fewer data to transform. A suggested method for displaying the aircraft in a new position is to manipulate the partial aircraft, then select WHOLE AIRCRFT once the aircraft is positioned as desired. When WHOLE AIRCRFT is selected, the following message appears:

```
select menu item:
select menu item:
DISPLAYING ENTIRE AIRCRAFT
```

Please note that the aircraft may be relocated when the entire aircraft is on screen, but more time is needed for the aircraft to respond to valuator input.

Example:

1. Rotate the partial aircraft into a new position.
2. Select the WHOLE AIRCRFT menu item.
3. Notice how nice the entire aircraft looks.

A.1.3 Display of Half of the Aircraft

Purpose:

to allow the user view and manipulate only half of the aircraft.

Menu Path:

HALF AIRCRFT

Description:

With the selection of HALF AIRCRFT the user can get rid of the entire aircraft and manipulate half as much data.

```
select menu item:
select menu item:
DISPLAYING HALF AIRCRAFT
```

Example:

1. Rotate the partial aircraft into a new position.
2. Select the WHOLE AIRCRFT menu item.
3. Select the HALF AIRCRAFT menu item to regain the partial aircraft.

4. Relocate aircraft.

A.1.4 Show and No-Show of the Intersection Plane

Purpose:
to allow the user to alternately render the intersection plane invisible and visible.

Menu Path:
DISPLAY PLANE and NO-SHOW PLANE

Description:
Upon entering the Hess Intersection program, the intersection plane is set invisible. The DISPLAY PLANE option allows the user to view the intersection plane. If the plane of intersection is visible to the user, selecting the NO-SHOW PLANE menu item causes the intersection plane to be invisible.

Example:

1. enter the Hess Intersection program (note that the intersection plane is invisible).
2. select the DISPLAY PLANE menu item. The plane will appear.
3. select the NO-SHOW PLANE option once again, and notice the plane will disappear.

A.1.5 Automatic Intersection in Hess Intersection

Purpose:
to allow the user to compute cross-sectional intersections between a cutting plane normal to the free-stream airflow and the skin of the aircraft.

Menu Path:
AUTO INTERSEC

Description:
With the selection of AUTO INTERSEC the user is prompted to enter an interval (in inches) between successive intersections. Once this is done, the program automatically computes a series of intersections between the cutting plane and the aircraft. The location of the cutting plane is determined by adding the amount of the specified interval to the previous planar location.

```
displaying entire aircraft:
displaying half aircraft:
PLEASE ENTER INTERSECTION INTERVAL:
```

The series of intersections produce intersection points which lie in planar cross-sections. These points represent the wetted perimeter of the aircraft. The data are stored in such a way that the corresponding cross-sectional area of each component in the cross-section can be determined. By adding all the areas of the components, the total cross-sectional area for the aircraft may be calculated.

After the series of intersections have been completed, the user can save the data by selecting SAVE FILE, and/or use the data to compute the area distribution graph of the aircraft. The latter is accomplished by the AREA DISTRIB menu option.

Example:

1. select the AUTO INTERSEC menu item.
2. enter ten inches as the intersection interval.
3. upon completion of the intersections, choose the AREA DISTRIB option to view the cross-sectional area of each network as a function of distance along the length of the aircraft.

A.1.6 User-Assisted Intersections

Purpose:

to allow the computation of cross-sectional intersections between a cutting plane normal to the free-stream airflow and the skin of the aircraft at a location determined by the user.

Menu Path:

BEG MAN INT

Description:

Before selecting BEG MAN INT the intersection plane should be set as visible. This facilitates the viewing of the location of the intersection. With the selection of BEG MAN INT the user is prompted to position the intersection plane as desired and to select the choice device to indicate that the plane is ready for an intersection.

```
enter file name:
displaying intersection plane:
HIT CHOICE BUTTON TO SIGNIFY INTERSECTION OR EXIT MANUAL:
```

The user is now free to relocate the intersection plane. Once the plane is in position the selection of a choice device writes the points of intersection to a default file. The file may be renamed upon exiting the manual intersection mode. In order to close the default file, END MAN INT must be chosen. When chosen, END MAN INT brings up the following message:

```
hit choice button to signify intersection:
go for another intersection or end manual:
MANUAL INTERSECTION SESSION ENDED:
```

The user has exited the manual intersection mode and may now proceed with saving the file.

If instead of selecting the END MAN INT option the user selects EXIT MANUAL, the manual intersection mode is aborted. EXIT MANUAL may only be chosen after BEG MAN INT is active. It is an alternative exit from the mode and is chosen instead of END MAN INT when no file data are necessary.

It is recommended that the user-assisted intersections be used only for the generation of file data. AREA DISTRIB does not work well if only a few intersections are computed, especially if they are not in ascending order in terms of x-location.

Example:

1. select the DISPLAY PLANE option to make the intersection plane visible.
2. select the BEG MAN INT menu item.
3. position the intersection plane at around 100 in. using valuator 8.
4. select a choice button to signify intersection.
5. select END MAN INT to close the default file.
6. choose the SAVE FILE menu item and enter TEST FILE.
7. upon exiting the program check for the file in the file listing.

A.1.7 Area Distribution

Purpose:

to allow the user to display a graph of the aircraft's area distribution.

Menu Path:

AREA DISTRIB

Description:

After a series of intersections have been computed using the AUTO INTERSEC option (manual can be used too), selection of AREA DISTRIB creates a graph of the area distribution. The area distribution represents the area of each network as a function of the distance along the x-axis. Each network's area profile is represented by a different color. The total aircraft area is shown in white and is computed by adding together the areas of all networks at a specific x-location. There are two options which exist under the heading of AREA DISTRIB. They are FULL SCREEN and 1/4 SCREEN. These options allow the area distribution graph to be displayed in either the full screen or the lower left-hand quarter of the screen.

Example:

1. select AUTO INTERSEC.
2. set the intersection interval to ten inches.
3. select the AREA DISTRIB menu item.

4. select the 1/4 SCREEN menu option. Note that both the aircraft and the area distribution are visible.
5. chose FULL SCREEN. Note the area distribution fills the screen once again.

A.1.8 Save Intersection Data File

Purpose:
to allow the user to save intersection points by cross-section.

Menu Path:
SAVE FILE

Description:
When either the AUTO INTERSEC or MAN INTERSEC options are chosen, a series of intersections which generate data intersection point data are computed. These data are written to a default file which is renamed upon selection of the "SAVE FILE" option.

```
displaying half aircraft:
displaying entire aircraft:
displaying half aircraft:
ENTER FILENAME AND FILETYPE OF FILE:
```

If no save file command has been issued upon exiting Hess Intersection, the most current default file will named FILE DEFALT in the file listing.

Example:

1. position intersection plane at 55.00 in. along the x-axis.
2. select the AUTO INTERSEC menu item.
3. select SAVE FILE.
4. respond to prompts by typing <JUNK, your initials>. **Make sure the filename and filetype entered are in capital letters.**
5. after exiting ACSYNT/VPI check the file listing for the file.

A.1.9 Exiting Hess Intersection

Purpose:
to allow the user to exit the program.

Menu Path:
the user will notice there is no menu item for exiting. When the user has finished with the

Hess Intersection program and wishes to exit, selection of one of the buttons of the choice device enables him to do so.

APPENDIX B. HESS INTERSECTION CODE

The following is a listing of the code for the Hess Intersection program.

```
************************************************************************
************************************************************************
*  THIS IS THE MAIN PROGRAM FOR THE DETERMINATION OF PLANAR INTERSECTION
*  WITH AIRCRAFT DESCRIBED IN A HESS FILE
************************************************************************
************************************************************************
      PROGRAM HESSMA
      INTEGER ERRFIL,BUFA,WKID,CONNID,WTYPE
      PARAMETER ( WKID = 1)
      REAL VALUE(8)
      INTEGER MAINVW
      PARAMETER (ERRFIL = 50,MAINVW = 1)


C  OPEN PHIGS
      CALL POPPH(ERRFIL,-1)

C  OPEN WORKSTATION, SET STATIC MENU, AND SET WINDOWS AND VALUATORS
      CALL STMNU(WKID,CONNID,WTYPE,VALUE)

C CREATE INTERSECTION PLANE AND SET IT INVISIBLE
      CALL CRIPL(1,VALUE,WKID)

C  WAIT FOR CHOICE INPUT THEN END
      CALL INPRTN(WKID,MAINVW,VALUE)

C  CLOSE THE WORKSTATION
      CALL PCLWK(WKID)

C  CLOSE PHIGS
      CALL PCLPH

      STOP
      END
************************************************************************
************************************************************************
* SUBROUTINE: STMNU(WKID,CONNID,WTYPE,VALUE)
*
************************************************************************
* DESCRIPTION:
*     OPENS THE WORKSTATION AND DOES THE INITIALIZATION IE...
*     SETS UP COLOR TABLE, DRAWS MENU AND MSG AREAS, INITIALIZES
*     DEVICES.
*
*     INPUT:
*          WKID -- WORKSTATION IDENTIFIER
```

```
*          CONNID -- CONNECTION IDENTIFIER
*          WTYPE -- WORKSTATION TYPE
*     OUTPUT:
*          VALUE -- ARRAY CONTAINING VALUATOR VALUES
*
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/16/88
*************************************************************************
*************************************************************************
      SUBROUTINE STMNU(WKID,CONNID,WTYPE,VALUE)
      INTEGER WKID,CONNID,WTYPE,NCOL,NUMVAL
      PARAMETER (NUMVAL = 8)
      REAL VALUE(*)


C FIND OUT WORKSTATION TYPE ETC...
      CALL GWINFO(WTYPE,CONNID)

C  OPEN WORKSTATION
      CALL POPWK(WKID,CONNID,WTYPE)

C  SET UP COLOR TABLE
      CALL SETCOL(WKID)

C  DRAW MENU AREA AND WORK AREA
      CALL MENDR(WKID,WTYPE,CONNID)

C INITIALIZE VALUATORS
      CALL INITIN(WKID,0,NUMVAL,VALUE)

C SET UP VIEW FOR PLANE AND AIRCRAFT
      CALL SETVUE(WKID,1)

      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: SETCOL(WKID)
*
*************************************************************************
* DESCRIPTION:
*     THIS SUBROUTINE SETS THE WORKSTATION COLOR TABLE
*
*     INPUT:
*          WKID -- WORKSTATION IDENTIFIER
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/16/88
*************************************************************************
*************************************************************************
      SUBROUTINE SETCOL(WKID)
      INTEGER WKID,IVAL(3),NCOL
      PARAMETER (NCOL = 15)
      REAL RGB(3,NCOL)

      DATA RGB / 1.00,  1.00, 1.00,
     $           1.00 , 0.00, 0.00,
     $           0.00,  1.00, 0.00,
     $           0.00,  0.00, 1.00,
     $           1.00,  1.00, 0.00,
     $           1.00,  0.00, 1.00,
     $           0.00,  1.00, 1.00,
     $           0.25,  0.42, 0.75,
     $           0.50,  0.30, 0.87,
     $           0.84,  0.36, 0.00,
     $           0.00,  0.40, 0.00,
```

```
     /
     $          0.00,  0.00, 0.40,
     $          0.70,  0.70, 0.70,
     $          0.50,  0.50, 0.50,
     $          0.30,  0.30, 0.30/
C
C THE COLORS ARE AS FOLLOWS

*  0   BLACK
*  1   WHITE
*  2   RED
*  3   BLUE
*  4   GREEN
*  5   YELLOW
*  6   MAGENTA
*  7   TOURQUOISE
*  8   SKY BLUE
*  9   PURPLE
*  10  ORANGE
*  11  DK. GREEN
*  12  DK. BLUE
*  13  LT. GRAY
*  14  MED. GRAY
*  15  DARK GRAY
*
C  SET COLOR VALUES
      DO 10 I = 1,NCOL

C SET COLOR REPRESENTATION
         CALL PSCR(WKID,I,RGB(1,I),RGB(2,I),RGB(3,I))
   10 CONTINUE
      RETURN
      END
*******************************************************************************
*******************************************************************************
* SUBROUTINE: MENDR(WKID,WTYPE,CONNID)
*
*******************************************************************************
* DESCRIPTION:
*     THIS SUBROUTINE DRAWS THE TITLE AREA, MENU AREA, WORK AREA, AND
*     MESSAGE AREA
*
*     INPUT:
*          WKID -- WORKSTATION IDENTIFIER
*          CONNID -- CONNECTION IDENTIFIER
*          WTYPE -- WORKSTATION TYPE
*******************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/16/88
*******************************************************************************
*******************************************************************************
      SUBROUTINE MENDR(WKID,WTYPE,CONNID)
      INTEGER WKID,WTYPE,CONNID,PRIOR,VIEW
      PARAMETER(PRIOR = 1,VIEW = 0)
      INTEGER TAID,MENID,WORKID,MESSID
      PARAMETER (TAID = 1,MENID = 2,WORKID = 3,MESSID = 5)
      INTEGER TXIDT,TXIDMN,TXIDMS
      PARAMETER(TXIDT = 6,TXIDMN = 7,TXIDWK = 8,TXIDMS = 9)
      INTEGER RED,GREEN,BLUE,YELLOW,MAGENT,TOURQ,WHITE,LTGRAY
      INTEGER MDGRAY,DKGRAY,SKYBL,PURP,ORAN,DKGRE,DKBLUE
      INTEGER BLACK
      PARAMETER (WHITE = 1,RED =2,GREEN =3,BLUE=4,YELLOW=5,MAGENT=6)
      PARAMETER (TOURQ = 7,LTGRAY = 8,MDGRAY = 9, DKGRAY = 10)
      PARAMETER (SKYBL = 11,PURP = 12,ORAN = 13,DKGRE = 14, DKBLUE = 15)
      PARAMETER (BLACK = 0)
      REAL TC(2),TOP(2),MENC(2),MENOP(2),WKC(2),WKOP(2),MESSC(2)
      REAL MESSOP(2)
      REAL PXT,PYT,PXMEN,PYMEN,PXMESS,PYMESS,SPAMEN,SPAMES
```

```
      REAL CHARTX,CHARMN,CHARMS
      INTEGER NTXTIT,NTXMEN,NTXMES
      PARAMETER (NTXTIT = 1,NTXMEN = 16,NTXMES = 1)
      CHARACTER*30 TXTIT(NTXTIT),TXMEN(NTXMEN),TXMESS(NTXMES),TXWK(1)

      DATA TC/-1.0,0.85/,TOP/1.0,1.00/
      DATA MENC/0.7,-0.80/,MENOP/1.0,0.80/
      DATA WKC/-1.0,-0.80/,WKOP/0.65,0.80/
      DATA MESSC/-1.0,-1.0/,MESSOP/0.70,-0.85/

      DATA PXT/-0.53/,PYT/0.9/,CHARTX/0.07/
      DATA PXMEN/0.72/,PYMEN/0.65/,CHARMN/0.02/,SPAMEN/0.05/
      DATA PXMESS/-0.9/,PYMESS/-0.92/,CHARMS/0.025/,SPAMES/0.05/

      TXTIT(1) = 'HESS INTERSECTION'
      TXMEN(1) = 'WHOLE AIRCRFT'
      TXMEN(2) = ' HALF AIRCRFT'
      TXMEN(3) = '              '
      TXMEN(4) = 'DISPLAY PLANE'
      TXMEN(5) = 'NO-SHOW PLANE'
      TXMEN(6) = '              '
      TXMEN(7) = 'AUTO INTERSEC'
      TXMEN(8) = ' BEG MAN INT '
      TXMEN(9) = ' END MAN INT '
      TXMEN(10) = ' EXIT MANUAL '
      TXMEN(11) = '             '
      TXMEN(12) = 'AREA DISTRIB '
      TXMEN(13) = '   FULL SCR  '
      TXMEN(14) = '   1/4  SCR  '
      TXMEN(15) = '             '
      TXMEN(16) = '  SAVE FILE  '
      TXMESS(1) = ' '
      TXWK(1) = ' '

C SET UP TITLES AREA
      CALL MENUAR(WKID,WTYPE,TAID,TC,TOP,BLUE,YELLOW,VIEW)
      CALL MENTXT(TXIDT,NTXTIT,TXTIT,PXT,PYT,CHARTX,0.,2,0)
      CALL ACTMNU(WKID,0,TAID,TXIDT,PRIOR)

C SET UP MENU AREA
      CALL MENUAR(WKID,WTYPE,MENID,MENC,MENOP,BLUE,YELLOW,VIEW)
      CALL MENTXT(TXIDMN,NTXMEN,TXMEN,PXMEN,PYMEN,CHARMN,SPAMEN,2,1)
      CALL ACTMNU(WKID,VIEW,MENID,TXIDMN,PRIOR)

C SET UP WORK AREA
      CALL MENUAR(WKID,WTYPE,WORKID,WKC,WKOP,BLACK,YELLOW,2)

C SET UP MESSAGE AREA
      CALL MENUAR(WKID,WTYPE,5,MESSC,MESSOP,BLUE,YELLOW,VIEW)
      CALL MENTXT(TXIDMS,NTXMES,TXMESS,PXMESS,PYMESS,CHARMS,SPAMES
     $           ,5,0)
      CALL ACTMNU(WKID,VIEW,5,TXIDMS,PRIOR)

      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: MENUAR(WKID,WTYPE,STRID,CORN,OPCORN,COLFA,COLP,VIEW)
*
*************************************************************************
* DESCRIPTION:
*  THIS SUBROUTINE SETS UP A MENU AREA
*
*     INPUT:
*          WKID -- WORKSTATION IDENTIFIER
*          WTYPE -- WORKSTATION TYPE
*          STRID -- STRUCTURE ID OF MENU
```

```
*           CORN -- BOTTOM CORNER OF MENU BOX
*           OPCORN -- OPPSITE CORNER OF MENU BOX
*           COLFA -- COLOR OF BOX FILL AREA
*           COLFA -- COLOR OF BOX BORDER POLYLINE
*           VIEW -- VIEW ID FOR MENU STUFF
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/16/88
*************************************************************************
*************************************************************************
      SUBROUTINE MENUAR(WKID,MTYPE,STRID,CORN,OPCORN,COLFA,COLP,VIEW)

      INTEGER STRID,COLFA,COLP
      REAL CORN(2),OPCORN(2)
      REAL PXA(5),PYA(5)
      INTEGER PJTYPE,VIEW

C OPEN A STRUCTURE FOR THE MENU
      CALL POPST(STRID)

C SET VIEW ID
      CALL PSVWI(VIEW)

C SET FILL AREA INTERIOR COLOR
      CALL PSICI(COLFA)

C SET INTERIOR STYLE AS SOLID
      CALL PSIS(1)

C SET UP POLYGON FOR FILL AREA
      DO 10 I = 1,2
          PXA(I) = CORN(1)
          PXA(I+2) = OPCORN(1)
   10 CONTINUE
      PYA(1) = CORN(2)
      PYA(4) = CORN(2)
      PYA(2) = OPCORN(2)
      PYA(3) = OPCORN(2)

      CALL PFA(4,PXA,PYA)

C TRIM IT WITH A POLYLINE BORDER
      CALL PSPLCI(COLP)
      PXA(5) = PXA(1)
      PYA(5) = PYA(1)
      CALL PPL(5,PXA,PYA)

C  PUT IN LABEL SO YOU CAN FIND THE PLACE TO EXECUTE TEXT
      CALL PLB(1)
      CALL PPL(4,PXA,PYA)
      CALL PLB(2)

C CLOSE STRUCTURE
      CALL PCLST


      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE:
*     MENTXT(STIDTX,NTXT,TXTSTR,PXST,PYST,CHSIZE,SPACE,FONT,FLAG)
*
*************************************************************************
* DESCRIPTION:
*  THIS SUBROUTINE SETS UP A MENU TEXT
*
```

```
*     INPUT:
*          STIDTX -- MENU TEXT STRUCTURE ID
*          NTXT -- NUMBER OF TEXT ELEMENTS
*          TXTSTR -- ARRAY CONTAINING TEXT STRING
*          PXST -- X LOCATION OF START OF TEXT
*          PYST -- Y LOCATION OF START OF TEXT
*          CHSIZE -- CHARACTER SIZE
*          SPACE -- CHARACTER SPACING
*          FONT -- TYPE O'FONT
*          FLAG -- DETERMINES IF MENU ITEMS SHOULD BE PICKABLE OR NOT
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/16/88
*************************************************************************
*************************************************************************
      SUBROUTINE MENTXT(STIDTX,NTXT,TXTSTR,PXST,PYST,CHSIZE,SPACE
     $,FONT,FLAG)
      INTEGER STIDTX,NTXT,FLAG,FONT,MENU
      PARAMETER (MENU = 1)
      REAL PXST,PYST,CHSIZE
      CHARACTER*(*) TXTSTR(*)

C  OPEN STRUCTURE IN WHICH TO PUT TEXT
      CALL POPST(STIDTX)

C GIVE MENU TEXT A CLASS NAME
      CALL PADS(1,MENU)

C SET CHARACTER FONT
      CALL PSTXFN(FONT)

C  SET CHARACTER HEIGHT
      CALL PSCHH(CHSIZE)

C  PUT TEXT IN STRUCTURE
C  AND SET PICK ID'S
      DO 10 I = 1,NTXT
         IF (FLAG .EQ.1) THEN
            CALL PSHPID(I)

         END IF
         CALL PTX(PXST,PYST,TXTSTR(I))
         PXST = PXST
         PYST = PYST - SPACE
   10 CONTINUE

C CLOSE STRUCTURE
      CALL PCLST
      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: INITIN(WKID,VIEW,NUMVAL,VALUE)
*
*************************************************************************
* DESCRIPTION:
*     THIS SUBROUTINE ACTIVATES AN ALREADY CREATED MENU STRUCTURE AND
*     PLACES TEXT IN IT.
*
*     INPUT:
*          WKID -- WORKSTATION IDENTIFIER
*          VIEW -- MENU STRUCTURE VIEW ID
*          NUMVAL -- NUMBER OF VALUATORS
*          VALUE -- ARRAY CONTAINING THE CURRENT VALUES OF THE VALUATORS
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/16/88
```

```
*************************************************************************
*************************************************************************
      SUBROUTINE INITIN(WKID,VIEW,NUMVAL,VALUE)

      INTEGER WKID,KERROR,UNITS,NUMVAL,ECHO,ESWTCH,EVENT,VALS,REQ
      PARAMETER (ECHO=1,VALS=8,ESWTCH=2,EVENT=2,REQ = 0)
      REAL AREA(6),HIVAL(VALS),LOVAL(VALS),IVALUE(VALS)
      REAL VALUE(*)
      REAL RX,RY
      INTEGER LX,LY,IPPD,ON
      PARAMETER (IPPD = 1,ON = 1)
      INTEGER PP(3,IPPD)
      REAL EVOL(6)
      CHARACTER*80 DATREC,ISTR

      DATA IVALUE/0.0,0.0,0.0,1.000,0.0,0.0,0.0,0.0/
      DATA HIVAL/360.,360.,360.,2.,400.,400.,400.,900./
      DATA LOVAL/0.,0.,0.,0.1,-400.,-400.,-400.,0./

C INQUIRE ECHO AREA OF VALUATOR
C     CALL GPQADS(WKID,KERROR,UNITS,CSIZE,ASIZE)
      CALL PQDSP(WTYPE,KERROR,UNITS,RX,RY,LX,LY)

C CONVERT DISPLAY SIZE INTO ECHO AREA
      AREA(1)=0.
      AREA(2)=0.84*RX
      AREA(3)=0.+0.89*RY
      AREA(4)=0. + 0.99*RY
      AREA(5)=0.
      AREA(6)=0.84*RX


C INITIALIZE VALUATORS
      DO 10 J=1,NUMVAL
         AREA(3)=AREA(3)-0.02*RY
         CALL PINVL3(WKID,J,IVALUE(J),ECHO,AREA,LOVAL(J)
     +    ,HIVAL(J),0,DATREC)
C PUT VALUATORS IN EVENT MODE
         CALL PSVLM(WKID,J,EVENT,ECHO)
   10 CONTINUE

C INITIALIZE CHOICE DEVICE
C PUT CHOICE IN EVENT MODE
      CALL GPCHMO(WKID,1,3,2)

C INITIALIZE PICK DEVICE
C  SET PICK MODE TO EVENT
      CALL PSHPM(WKID,1,EVENT,ECHO)

C MAKE MENU ITEMS PICKABLE
      CALL PSHPFT(WKID,1,1,1,0,0)

C INITIALIZE VALUE ARRAY WHICH CONTAINS VALUATOR INFORMATION
      DO 20 I=1,NUMVAL
         VALUE(I)=IVALUE(I)
   20 CONTINUE

      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: SETVUE(WKID,VIEWID)
*
*************************************************************************
* DESCRIPTION:
*     SETS UP THE PRIMARY VIEW FOR THE PROGRAM
*
```

```
*     INPUT:
*          WKID -- INTEGER, WORKSTATION IDENTIFIER
*          VIEWID -- VIEW ID FOR PRIMARY VIEW (CONTAINS AIRCRAFT)
*
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/20/88
*************************************************************************
*************************************************************************
      SUBROUTINE SETVUE(WKID,VIEWID)
      REAL VWLIM(4),PROLIM(6),PJRX,PJRY,PJRZ,VPLD,FPLD,BPLD
      REAL VLIM(4),PLIM(6)
      REAL VIM(4),PIM(6)
      INTEGER PARALL,VIEWID,WKID
      PARAMETER (PARALL = 0)
      DATA PJRX/400./,PJRY/0./,PJRZ/600./
      DATA VPLD/910./,FPLD/900./,BPLD/-900./
      DATA VWLIM/0.,800.,-400.,400./
      DATA PROLIM/0.,0.7,0.1,0.8,0.,1.0/


C SET VIEW MAPPING
      CALL PSVMP3(WKID,VIEWID,VWLIM,PROLIM,PARALL,PJRX,PJRY,PJRZ,
     $            VPLD,FPLD,BPLD)

C SET VIEW CHARACTERISTICS
      CALL PSVWCS(WKID,VIEWID,0,0,0,1,0)


      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: CRIPL(VIEWID,VALUE,WKID)
*
*************************************************************************
* DESCRIPTION:
*     DRAWS THE INTERSECTION PLANE AND THEN SETS ITS INVISIBILITY
*     FILTER ON
*
*     INPUT:
*        VIEWID -- VIEW IN WHICH TO SHOW THE INTERSECTION PLANE
*        VALUE -- ARRAY CONTAINING CURRENT VALUATOR VALUES
*        WKID -- WORKSTATION IDENTIFIER
*
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 5/06/88
*************************************************************************
*************************************************************************
      SUBROUTINE CRIPL(VIEWID,VALUE,WKID)
      INTEGER VIEWID,WKID,ES
      REAL VALUE(8)

C CALL THE ROUTINE WHICH DRAWS THE INTERSECTION PLANE
      CALL PLANE(WKID,VALUE,VIEWID)

C SET THE INTERSECTION PLANE INVISIBLE
      CALL PSIVFT(WKID,1,2,0,ES)

      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: PLANE(WKID,VALUE,AIRVID)
*
*************************************************************************
```

```
* DESCRIPTION: THIS SUBROUTINE DETERMINES THE LOCATION AND APPEARANCE
*              OF THE INTERSECTION PLANE
*
*     INPUT:
*        WKID -- WORKSTATION IDENTIFIER
*        VALUE -- ARRAY CONTAINING VALUATOR VALUES
*        AIRVID -- VIEW ID WHICH CONTAINS THE AIRCRAFT
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 5/2/88
*************************************************************************
*************************************************************************
      SUBROUTINE PLANE(WKID,VALUE,AIRVID)
      REAL VALUE(*),A,B,C,D
      PARAMETER(A = -250.0, B = -200.0, C = 250.0, D = 255.0)
      INTEGER NUMINC,TOURQ,AIRVID,CLINT,PLID
      PARAMETER (NUMINC = 50,TOURQ = 7,CLINT = 2,PLID = 20)
      REAL PXA(100),PYA(100),PZA(100)
      REAL VW(4),PRO(6),XFORM(4,4)
      INTEGER BFTRAN,AFTRAN,BFINV
      PARAMETER (BFTRAN = 90, AFTRAN = 91,BFINV=89)
      DATA VW/0.,800.,-400.,400./
      DATA PRO/0.,0.9,0.1,0.9,0.,1.0/

C OPEN STRUCTURE FOR INTERSECTION PLANE
      CALL POPST(PLID)

C SET VIEW INDEX TO BE SAME AS AIRCRAFT STRUCTURE
      CALL PSVWI(AIRVID)

C SET PICK IDENTIFIER FOR PLANE
      CALL PSHPID(100)

C SET CLASS NAMES FOR ATTRIBUTES OF PLANE STRUCTURE
      CALL PADS(1,CLINT)

C SET FIRST LABEL BEFORE TRANSFORMATION
      CALL PLB(BFTRAN)

C CALCULATE TRANSFORMATION MATRIX, TRANSLATION IN X DIRECTION
      CALL PTR3(0.,0.,0.,XFORM)

C SET LOCAL TRANSFORM
      CALL PSLMT3(XFORM,2)

C SET SECOND LABEL
      CALL PLB(AFTRAN)

C SET POLYLINE COLOR
      CALL PSPLCI(TOURQ)

C DRAW POLYLINE MESH WHICH REPS A RECTANGULAR PORTION OF THE PLANE
      CALL DRMESH(NUMINC, VALUE, C, A, D, B)

C CLOSE STRUCTURE
      CALL PCLST

C POST ROOT
      CALL PPORT(WKID,PLID,.8)

      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: DRMESH(NUMINC, VALUE, C, A, D, B)
*
*************************************************************************
```

```
* DESCRIPTION:
*     DRAWS THE MESH WHICH REPRESENTS THE PLANE OF INTERSECTION
*
*     INPUT:
*        NUMINC -- NUMBER OF LINES WHICH REP THE MESH
*        VALUE -- ARRAY CONTAINING VALUATOR VALUES
*        A,B,C,D -- PARAMETERS OF THE INTERSECTION PLANE
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 5/2/88
*************************************************************************
*************************************************************************
      SUBROUTINE DRMESH( NUMINC, VALUE, C, A, D, B)
      INTEGER NUMINC
      REAL VALUE(*), C, A , D, B, U, W
      REAL PXA(100),PYA(100),PZA(100)

      DO 10 I = 1,NUMINC
         U = FLOAT(I)/FLOAT(NUMINC)
         DO 20 J = 1,NUMINC
            W = FLOAT(J)/FLOAT(NUMINC)

            PXA(J) = 100. + VALUE(8)
            PYA(J) = (C-A)*U + A
            PZA(J) = (D-B)*W + B
   20    CONTINUE
         CALL PPL3(NUMINC,PXA,PYA,PZA)
   10 CONTINUE

      DO 30 I = 1,NUMINC
         W = FLOAT(I)/FLOAT(NUMINC)
         DO 40 J = 1,NUMINC
            U = FLOAT(J)/FLOAT(NUMINC)

            PXA(J) = 100.
            PYA(J) = (C-A)*U + A
            PZA(J) = (D-B)*W + B
   40    CONTINUE
         CALL PPL3(NUMINC,PXA,PYA,PZA)
   30 CONTINUE

      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: INPRTN(WKID,VIEW,VALUE)
*
*************************************************************************
* DESCRIPTION:
*     WAITS FOR EVENT INPUT AND THEN DECIDES WHAT TO DO WITH IT BASED
*     ON THE INPUT DEVICE
*
*     INPUT:
*          WKID -- WORKSTATION IDENTIFIER
*          VIEW -- VIEW IDENTIFIER OF THE PRIMARY VIEW
*          VALUE -- INPUT ARRAY OF VALUATOR VALUES
*     OUTPUT:
*          VALUE -- OUTPUT ARRAY OF UPDATED VALUATOR VALUES
*
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 3/16/88
*************************************************************************
*************************************************************************
      SUBROUTINE INPRTN(WKID,VIEW,VALUE)

      INTEGER WKID,CLASS,IDNR,VALTR,PLID,PICK
```

```
      REAL SECS,VALUE(*),PTDATA(3,100,50,10),INTARR(3,100,10)
      PARAMETER(SECS=3600.0,VALTR=3,PLID = 20,PICK = 5)
      INTEGER COUNT,VIEW

C INITIALIZE COUNT TO ZERO
      COUNT = 0

C UPDATE WORKSTATION
  100 CALL PUWK(WKID,1)
      COUNT = COUNT + 1

C IF COUNT IS ONE THEN IT'S THE FIRST TIME THRU THE PROGRAM
      IF(COUNT .EQ. 1) THEN
         CALL MSG(WKID,'ENTER FILE NAME')
         GO TO 100

      ELSE IF (COUNT.EQ.2) THEN
         CALL INAIR(VIEW,WKID,PTDATA,COUNT)
         GO TO 100
      END IF

C WAIT FOR EVENT TO OCCUR
      CALL PWAIT(SECS,KWKID,CLASS,IDNR)

C CHECK TO SEE WHAT DEVICE WAS USED FOR INPUT

C IF IT WAS A VALUATOR SEE WHICH ONE
C VALUATORS 1-7 ARE USED TO CALCULATE A GLOBAL X-FORM
      IF (CLASS .EQ. VALTR) THEN

         IF (IDNR .NE. 8) THEN
            CALL PVLIN(WKID,IDNR,VIEW,VALUE)
         ELSE

C VALUATOR 8 IS USED TO LOCALLY MOVE THE INTERSECTION PLANE
            CALL DELXF(VALUE,PLID,IDNR)
         END IF
         GO TO 100

C SEE IF IT WAS A PICK
      ELSE IF(CLASS .EQ. PICK) THEN
         CALL DETPIK(WKID,PTDATA,INTARR,VALUE)
         GO TO 100

C IF IT WAS A CHOICE DEVICE THEN END
      END IF

      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: INAIR(VIEWID,WKID,PTDATA)
*
*************************************************************************
* DESCRIPTION:
*     READS IN AND DISPLAYS AIRCRAFT DATA
*
*     INPUT:
*        VIEWID -- AIRCRAFT VIEW ID
*        WKID -- WORKSTATION IDENTIFIER
*     OUTPUT:
*        PTDATA -- ARRAY CONTAINING AIRCRAFT GEOMETRY DATA
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 05/06/88
*************************************************************************
*************************************************************************
```

```
      SUBROUTINE INAIR(VIEWID,WKID,PTDATA)
      INTEGER VIEWID,WKID,LOSTR,STID
      REAL PTDATA(3,100,50,10)
      CHARACTER*7 STR
      PARAMETER (STID = 10)

C CREATE AND READ DATA FILE CONTAINING HESS INFORMATION
      CALL INFILE(WKID,LOSTR,STR,PTDATA)

C DRAW THE AIRCRAFT
      CALL HESSDR(PTDATA,STID,VIEWID,WKID,.8)

      RETURN
      END
*******************************************************************************
*******************************************************************************
* SUBROUTINE: INFILE(WKID,LOSTR,STR,PTDATA)
*
*******************************************************************************
* DESCRIPTION:
*     CREATES AND READS A SPECIFIED DATA FILE
*
*     INPUT:
*          WKID -- WORKSTATION IDENTIFIER
*     OUTPUT:
*          LOSTR -- LENGTH OF CHARACTER STRING READ FROM TERMINAL
*          STR -- CHARACTER STRING READ FROM TERMINAL
*          PTDATA -- ARRAY CONTAININ AIRCRAFT DATA (XYZ,NODE,XS,NET)
*
*******************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 05/06/88
*******************************************************************************
*******************************************************************************
      SUBROUTINE INFILE(WKID,LOSTR,STR,PTDATA,COUNT)
      INTEGER WKID,LOSTR,COUNT,RC
      REAL PTDATA(3,100,50,*)
      CHARACTER*(*) STR

C READ IN A STRING
      CALL STRIN(WKID,LOSTR,STR)

C CHECK TO SEE IF FILE EXISTS
      CALL CHECK(STR,RC)

C IF FILE IS OKAY THEN
      IF (RC .EQ. 0) THEN

C OPEN FILE UNIT 10 AND CREATE A FILE FROM WHICH THE HESS DATA MAY
C BE READ
         OPEN(FILE = STR,UNIT = 10)

C READ IN DATA FROM THE HESS FILE
         CALL RDHESS(PTDATA,WKID)

C CLOSE FILE
         CLOSE(UNIT = 10)

C ELSE IF FILE NO GOOD
      ELSE
         COUNT = 0
         CALL MSG(WKID,'NO SUCH FILE, TRY AGAIN')
      END IF

      RETURN
      END
*******************************************************************************
```

```
************************************************************************
* SUBROUTINE: CHECK(STR,RC)
*
************************************************************************
* DESCRIPTION:
*     DETERMINES IF A FILE EXISTS OR NOT
*
*     INPUT:
*          STR  -- STRING IDENTIFYING THE FILE NAME
*     OUTPUT:
*          RC -- RETURN CODE
*                0 - FILE EXISTS
*                1 - FILE DOES NOT EXIST
************************************************************************
************************************************************************
*
*  NOTE: THIS ROUTINE IS NOT DEVICE INDEPENDENT!!!!!!!!!!!!!!!!
*
*
************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 05/06/88
************************************************************************
************************************************************************
      SUBROUTINE CHECK(STR,RC)
      CHARACTER*(*) STR
      INTEGER RC
      LOGICAL FLAG

C     INQUIRE IF FILE EXISTS
      INQUIRE(FILE = STR, EXIST = FLAG)

C     IF EXISTS THEN
      IF (FLAG) THEN
         RC = 0

C     IF NOT
      ELSE
         RC = 1
      END IF

      RETURN
      END
************************************************************************
************************************************************************
* SUBROUTINE: STRIN (WKID,LOSTR,STR)
*
************************************************************************
* DESCRIPTION:
*     READS IN A STRING FROM THE TERMINAL
*
*     INPUT:
*          WKID -- WORKSTATION IDENTIFIER
*     OUTPUT:
*          LOSTR -- LENGTH OF RETURNED STRING
*          STR -- THE RETURNED STRING
*
************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 05/06/88
************************************************************************
************************************************************************
      SUBROUTINE STRIN(WKID,LOSTR,STR)
      INTEGER WKID,LOSTR
      CHARACTER*(*) STR

C REQUEST STRING
```

```
      CALL PRQST(WKID,1,STAT,LOSTR,STR)

      RETURN
      END
*****************************************************************************
*****************************************************************************
* SUBROUTINE: RDHESS(PTDATA,WKID)
*
*****************************************************************************
* DESCRIPTION: READS DATA IN FROM A HESS FILE
*      INPUT:
*         WKID -- WORKSTATION IDENTIFIER
*      OUTPUT:
*         PTDATA -- ARRAY CONTAINING AIRCRAFT DATA
*
*****************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/05/88
*****************************************************************************
*****************************************************************************
      SUBROUTINE RDHESS(PTDATA,WKID)

      COMMON /DPNTS/ NODE,XS,NET,NNODE,NXS
      INTEGER NODE,XS,NET,NNODE(20),NXS(20)

      COMMON /BORDER/ MAX,MIN
      REAL MAX(20),MIN(20)

      INTEGER STAT,WKID
      REAL XYZ(3),PTDATA(3,100,50,10)
      CHARACTER*80 STRG

C SET MAX AND MIN ARRAYS AS ZERO
      DO 10 I = 1,20
         MAX(I) = 0.
         MIN(I) = 1000.
   10 CONTINUE

C INITIALIZE NUMBER OF X-SECTIONS

      XS = 1

C INITIALIZE NUMBER OF NETWORKS
      NET = 0

C INITIALIZE NUMBER OF NODES
      NODE = 0

C READ IN DATA AS AN ALPHANUMERIC STRING
  200 READ(10,1000,END = 300) STRG
 1000 FORMAT(A)

C CONVERT ALPHANUMERIC TO REAL AND INTEGER DATA STRING
      READ(STRG,2000)XYZ(1),XYZ(2),XYZ(3),STAT
 2000 FORMAT(3(1X,F9.5),1X,I2)

C CALL SUBROUTINE TO DETERMINE STATUS OF DATA LINE JUST READ
      CALL STATUS(STAT,XYZ,PTDATA)

C READ NEXT LINE OF DATA
      GO TO 200
  300 RETURN
      END
*****************************************************************************
*****************************************************************************
* SUBROUTINE: STATUS(STAT,XYZ,PTDATA)
```

```
*
*************************************************************************
* DESCRIPTION: DETERMINES THE STATUS OF A LINE OF DATA READ FROM
*              THE HESS FILE.
*
*     INPUT:
*        STAT -- DETERMINES THE STATUS OF THE DATA LINE
*         XYZ -- INPUT DATA LINE FROM HESS FILE
*    OUTPUT:
*        PTDATA -- ARRAY CONTAINING AIRCRAFT DATA
*
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/15/88
*************************************************************************
*************************************************************************
      SUBROUTINE STATUS(STAT,XYZ,PTDATA)

      COMMON /DPNTS/ NODE,XS,NET,NNODE,NXS
      INTEGER NODE,XS,NET,NNODE(20),NXS(20)

      COMMON /BORDER/ MAX,MIN
      REAL MAX(20),MIN(20)

      REAL PTDATA(3,100,50,*),XYZ(*)
      INTEGER STAT


C DETERMINE THE STATUS OF THE LINE OF DATA
C
C IF STATUS IS 0 IT'S PART OF THE CURRENT X-SECTION
      IF(STAT .EQ. 0) THEN
         NODE = NODE + 1
C CALL SUBROUTINE TO PUT POINTS IN ANOTHER ARRAY
         CALL PTGET(XYZ,PTDATA)

C SET UP AN ARRAY WHICH CONTAINS THE MAX AND MIN X LOCATION POINTS
C IN A NETWORK
         IF(PTDATA(1,NODE,XS,NET) .GT. MAX(NET))
     $      MAX(NET) = PTDATA(1,NODE,XS,NET)
         IF(PTDATA(1,NODE,XS,NET) .LT. MIN(NET))
     $      MIN(NET) = PTDATA(1,NODE,XS,NET)



C IF STAT IS 10 THEN IT'S A NEW X-SECTION
      ELSE IF (STAT .EQ. 10) THEN
          NODE = 1
          XS = XS + 1
          CALL PTGET(XYZ,PTDATA)

C IF STAT IS 20 THEN IT'S A NEW NETWORK
      ELSE IF (STAT .EQ. 20) THEN
         IF (NET .NE. 0) THEN



C SET NODE AND CROSS-SECTION STUFF FOR THE PREVIOUS NETWORK
         NNODE(NET) = NODE
         NXS(NET) = XS

C RESET VALUES TO CONTINUE COMPILING PT ARRAY
         NODE = 1
         XS = 1

         ELSE
            NODE = NODE + 1
```

```
         END IF


         NET = NET + 1
         CALL PTGET(XYZ,PTDATA)


C IF STAT IS 30 THEN IT'S THE END OF THE DATA FILE

      ELSE IF (STAT .EQ. 30) THEN
         NODE = NODE + 1
         NNODE(NET) = NODE
         NXS(NET) = XS
         CALL PTGET(XYZ,PTDATA)
      END IF

      RETURN
      END
*******************************************************************************
*******************************************************************************
* SUBROUTINE: PTGET(XYZ,PTDATA)
*
*******************************************************************************
* DESCRIPTION: PUTS DATA POINTS READ FROM HESS FILE INTO NEW ARRAY.
*              ARRAY IS IN TERMS OF XYZ, NODE NUMBER, CROSS-SECTION,
*              AND NETWORK
*     INPUT:
*         XYZ -- INPUT DATA LINE FROM HESS FILE
*    OUTPUT:
*         PTDATA -- ARRAY CONTAINING AIRCRAFT DATA
*
*******************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/15/88
*******************************************************************************
*******************************************************************************
      SUBROUTINE PTGET(XYZ,PTDATA)
      REAL XYZ(*),PTDATA(3,100,50,*)

      COMMON /DPNTS/ NODE,XS,NET,NNODE,NXS
      INTEGER NODE,XS,NET,NNODE(20),NXS(20)


C PUT POINTS READ FROM HESS FILE INTO A NEW ARRAY IN TERMS OF NODE #,
C X-SECTION, AND NETWORK

      DO 10 I = 1,3
         PTDATA(I,NODE,XS,NET) =  XYZ(I)
   10 CONTINUE

      RETURN
      END
*******************************************************************************
*******************************************************************************
* SUBROUTINE: HESSDR(PTDATA,STID,VIEWID,WKID,PRIOR)
*
*******************************************************************************
* DESCRIPTION: CREATES AND DISPLAYS THE AIRCRAFT STRUCTURE
*
*     INPUT:
*        PTDATA -- ARRAY CONTAINING AIRCRAFT DATA
*        STID -- AIRCRAFT STRUCTURE ID
*        VIEWID -- AIRCRAFT VIEW ID
*        WKID -- WORKSTATION IDENTIFIER
*        PRIOR -- STRUCTURE PRIORITY
*
*******************************************************************************
```

```
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/27/88
*************************************************************************
*************************************************************************
      SUBROUTINE HESSDR(PTDATA,STID,VIEWID,WKID,PRIOR)
      REAL PTDATA(3,100,50,10),PRIOR
      INTEGER STID,VIEWID,WKID,WORKID


C OPEN STRUCTURE IN WHICH TO PUT AIRCRAFT
      CALL POPST(STID)

C EMPTY STRUCTURE
      CALL PEMST(STID)

C SET VIEW ID
      CALL PSVWI(VIEWID)

C CALL ROUTINE TO DRAW AIRCRAFT
      CALL SPLDRW(PTDATA)

C CLOSE STRUCTURE
      CALL PCLST

C POST ROOT
      CALL PPORT(WKID,STID,PRIOR)


      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: SPLDRW(PTDATA)
*
*************************************************************************
*************************************************************************
*   THIS SUBROUTINE TAKES THE DATA WHICH WAS READ FROM THE HESS FILE
*   AND BREAKS IT DOWN INTO CHUNKS SO THAT IT MAY BE DRAWN.  THE CHUNKS
*   CONSIST OF NETWORKS BROKEN INTO SPLINES WHICH ARE DRAWN FROM NODE
*   ONE ON CROSS-SECTION ONE TO NODE ONE ON CROSS-SECTION TWO... UNTIL
*   THE LAST CROSS-SECTION IN THE NETWORK.  THIS SUBROUTINE TAKES THE
*   DATA AND BREAKS IT INTO NETWORK SIZE SECTIONS.
*
*      INPUT:
*         PTDATA -- ARRAY CONTAINING AIRCRAFT DATA
*
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/27/88
*************************************************************************
*************************************************************************
      SUBROUTINE SPLDRW(PTDATA)
      REAL PTDATA(3,100,50,*)

      COMMON /DPNTS/ NODE,XS,NET,NNODE,NXS
      INTEGER NODE,XS,NET,NNODE(20),NXS(20)

C  SEND DATA TO ANOTHER ARRAY ONE NETWORK AT A TIME
      DO 10 I = 1,NET

C RECALL NUMBER OF X-SECS AND NODES PER X-SEC FOR THIS NETWORK
         NUMNET = I
         NUMNO = NNODE(I)
         NUMXS = NXS(I)

C  CALL SUBROUTINE TO PUT NETWORK DATA INTO A NEW ARRAY
         CALL NEWRAY(PTDATA,NUMNO,NUMXS,NUMNET)
```

```
   10 CONTINUE

      RETURN
      END
*******************************************************************************
*******************************************************************************
* SUBROUTINE: NEWRAY(PTDATA,NUMNO,NUMXS,NUMNET)
*
*******************************************************************************
*******************************************************************************
*  THIS SUBROUTINE CONVERTS THE ARRAY OF NETWORK DATA INTO A FORM
*  WHICH CAN BE PASSED TO A 3D SPLINE ROUTINE FOR CALCULATION OF THE
*  TANGENT VECTORS FOR EACH POINT IN THE ARRAY.  THE DATA AND THE NEW
*  ARRAY WHICH CONTAINS THE TANGENT INFO ARE THEN PASSED TO A CUBIC
*  ROUTINE WHICH DRAWS THE STUFF.
*
*      INPUT:
*         PTDATA -- ARRAY CONTAINING AIRCRAFT DATA
*         NUMNO -- NUMBER OF NODES PER XSEC FOR THIS NETWORK
*         NUMXS -- NUMBER OF XSECS FOR THIS NETWORK
*         NUMNET -- NETWORK NUMBER
*
*******************************************************************************
*******************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/27/88
*******************************************************************************
*******************************************************************************
      SUBROUTINE NEWRAY(PTDATA,NUMNO,NUMXS,NUMNET)
      REAL PTDATA(3,100,50,*),SPLPTS(3,50),TANVEC(3,50),S(50)
      REAL A(50),B(50),R(50)
      INTEGER NUMNO,NUMXS,SPDIM,ND,NUM
      PARAMETER (SPDIM = 3, ND = 0, NUM = 4)


         MST = 1

C  PUT NETWORK ARRAY INTO AN ARRAY THAT CAN BE PASSED TO SPLINE ROUTINE
      DO 30 I = 1,NUMNO
            DO 20 J = MST,NUMXS
               DO 10 K = 1,3

                  SPLPTS(K,J) = PTDATA(K,I,J,NUMNET)

   10          CONTINUE
   20       CONTINUE

C  CALL SPLINE ROUTINE AND THEN WITH THE TAN VECTOR INFO CALL A ROUTINE
C  WHICH BREAKS SPLINE INTO CHUNKS CONSISTING OF TWO POINTS APIECE
         NSPL = NUMXS
         CALL SPL3D(NSPL,SPDIM,ND,SPLPTS,TANVEC,S,A,B,R,*1000)
         CALL BAYS(SPLPTS,TANVEC,NSPL,NUM)

   30 CONTINUE
      RETURN

 1000 WRITE(*,*)'ERROR IN SPLINE 3D'

      RETURN
      END
*******************************************************************************
*******************************************************************************
      SUBROUTINE SPL3D(N,M,ND,D,V,S,A,B,R,*)                          00000100
C   RELEASE 20.1                                                      00000120
C   COPYRIGHT 1985 BY CADAM INC.                                      00000140
C   BURBANK, CALIFORNIA 91504                                         00000160
C   ALL RIGHTS RESERVED                                               00000180
```

```
C                                                                       00000200
C**-----------------------------------------------------------------**00000300
C**  SPL3D CALCULATES AN 'N'-POINT, 'M'-DIMENSIONAL PARAMETRIC SPLINE   00000400
C**   CURVE.  THE INPUT POINTS ARE LOADED INTO THE D ARRAY BEFORE THE   00000500
C**   CALL.  UPON COMPLETION OF THE CALCULATIONS, THE V ARRAY WILL CON- 00000600
C**   TAIN THE TANGENT VECTORS AND THE S ARRAY WILL CONTAIN THE CHORD   00000700
C**   MAGNITUDES FOR EACH BAY OF THE SPLINE CURVE.                      00000800
C**                                                                     00000900
C**-----------------------------------------------------------------**00001000
C**   M IS THE DIMENSION OF THE SPLINE                                  00001100
C**   N IS THE NUMBER OF POINTS                                         00001200
C**   D CONTAINS THE COORDINATES OF THE POINTS                          00001300
C**   V CONTAINS THE VECTORS                                            00001400
C**   S CONTAINS THE DISTANCES                                          00001500
C**   ND IS AN INDICATOR OF END VECTOR INPUTS:                          00001600
C**      ND < 0  CLOSED END SPLINE                                      00001700
C**      ND = 0  NO END VECTORS                                         00001800
C**      ND = 1  BEGINNING VECTOR ONLY                                  00001900
C**      ND = 2  END VECTOR ONLY                                        00002000
C**      ND = 3  BEGINNING AND END VECTORS                              00002100
C**                                                                     00002200
C**                                                                     00002300
C**-----------------------------------------------------------------**00002400
C**                                                                     00002500
      REAL D(3,*),V(3,*),S(N),A(N),B(N),R(N),EPSLON,DIST                00002600
      REAL F,T,TN,TSQR,TNSQR,V1SUM,VNSUM                                00002700
C**                                                                     00002800
C**                                                                     00002900


      EPSLON = 0.00005                                                  00003000
      NM1 = N - 1                                                       00003100
      NM2 = N - 2                                                       00003200
C**-----------------------------------------------------------------**00003300
C**      S IS AN INPUT FOR DIMENSION (M=1)                            **00003400
C**-----------------------------------------------------------------**00003500
      IF (M .EQ. 1)                          GO TO 106                  00003600
C**                                                                     00003700
C**-----------------------------------------------------------------**00003800
C**      COMPUTE CHORD MAGNITUDES.                                      00003900
C**-----------------------------------------------------------------**00004000
      DO   10  I = 2, N                                                 00004100
       DIST = 0.0                                                       00004200
       DO    5  L = 1, M                                               00004300
        DIST = DIST + (D(L,I) - D(L,I-1))**2                            00004400
    5  CONTINUE                                                         00004500
       IF(DIST .EQ. 0.0) RETURN1                                        00004600
       S(I-1) = SQRT(DIST)                                              00004700
   10 CONTINUE                                                          00004800
C**                                                                     00004900
C**-----------------------------------------------------------------**00005000
C**      2 POINT CASE HANDLE END VECTORS HERE                         **00005100
C**-----------------------------------------------------------------**00005200
      IF (N .GT. 2)  GO TO  100                                         00005300
      IF (ND .NE. 0) GO TO   20                                         00005400
C**                                                                     00005500
C**  COMPUTE STRAIGHT LINE IF NO VECTORS GIVEN. (ND = 0)                00005600
C**                                                                     00005700
      DO   15  L = 1, 2                                                 00005800
       DO   15  I = 1, M                                                00005900
        V(I,L) = D(I,2) - D(I,1)                                        00006000
   15 CONTINUE                                                          00006100
      GO TO  118                                                        00006200
C**                                                                     00006300
C**  COMPUTE A VECTOR IF ONLY ONE IS GIVEN. (ND = 1 OR 2)               00006400
C**                                                                     00006500
   20 IF (ND .LT. 0)  RETURN1                                           00006600
```

```
      IF (ND .EQ. 3)  GO TO 118                                          00006700
      J = 1 + ND/2                                                       00006800
      K = 1 + 1/ND                                                       00006900
C**                                                                      00007000
C**  FORM CHORD .DOT. END VECTOR                                         00007100
C**                                                                      00007200
      T = 0.0                                                            00007300
      DIST = 0.0                                                         00007400
      DO   25  I = 1, M                                                  00007500
       T = T + (D(I,2) - D(I,1))*V(I,J)                                  00007600
       DIST= DIST + (D(I,2)-D(I,1))**2                                   00007700
   25 CONTINUE                                                           00007800
      IF (ABS(T) .LT. EPSLON) GO TO 35                                   00007900
      T = .5*SQRT(DIST)/T                                                00008000
      DO   30  I = 1, M                                                  00008100
        V(I,K) = (D(I,2) - D(I,1)) - T*V(I,J)                            00008200
   30 CONTINUE                                                           00008300
      GO TO  118                                                         00008400
C**                                                                      00008500
C**  REVERSE THE GIVEN END VECTOR FOR THE OTHER END IFF CHORD .DOT. END  00008600
C**   IMPLIES 90 DEGREES.                                                00008700
C**                                                                      00008800
   35 DO   40  I = 1, M                                                  00008900
        V(I,K) = -V(I,J)                                                 00009000
   40 CONTINUE                                                           00009100
      GO TO  118                                                         00009200
C**                                                                      00009300
C**--------------------------------------------------------------------**00009400
C**  COMPUTE CONSTANTS FOR SPLINES WITH MORE THAN 2 POINTS.              00009500
C**--------------------------------------------------------------------**00009600
C**                                                                      00009700
  100 CONTINUE                                                           00009800
      IF (ND .LT. 0) GO TO 106                                           00009900
      IF (ND .GE. 3) GO TO  118                                          00010000
      IF (M .EQ. 1) GO TO  118                                           00010100
      T = S(1)/(S(1) + S(2))                                             00010200
      TSQR = T*T                                                         00010300
      TN = S(NM1)/(S(NM2)+S(NM1))                                        00010400
      TNSQR = TN*TN                                                      00010500
C**                                                                      00010600
C**  COMPUTE END VECTORS BASED ON CURVED ENDS IFF UNSPECIFIED.           00010700
C**                                                                      00010800
      DO  105  I = 1, M                                                  00010900
       IF (ND .NE. 1)                                                    00011000
     A V(I,1) = (D(I,3)*TSQR + D(I,1)*(1.0 - TSQR) - D(I,2))/(TSQR-T)    00011100
       IF (ND .NE. 2)                                                    00011200
     A V(I,N) = -(D(I,NM2)*TNSQR+D(I,N)*(1.0-TNSQR)-D(I,NM1))/(TNSQR-TN)00011300
  105 CONTINUE                                                           00011400
      GO TO 118                                                          00011500
C**                                                                      00011600
C**--------------------------------------------------------------------**00011700
C**  SPECIAL CASE FOR CLOSED SPLINE                                      00011800
C**--------------------------------------------------------------------**00011900
  106 IF (ND .GE. 0) GO TO 110                                           00011910
      IF (N .LE. 2) RETURN1                                              00011920
      T = S(NM1) / (S(NM1)+S(1))                                         00012000
      TSQR = T*T                                                         00012100
      DO 108 I = 1, M                                                    00012200
      V(I,1) = (D(I,2)*TSQR+D(I,NM1)*(1.0-TSQR)-D(I,1)) / (TSQR-T)       00012300
      V(I,1) = V(I,1) + 2.0*T*(D(I,2)-D(I,NM1)-V(I,1))                   00012400
      V(I,N) = V(I,1)                                                    00012500
  108 CONTINUE                                                           00012600
      GO TO 118                                                          00012700
C**                                                                      00012800
C**--------------------------------------------------------------------**00012900
C**      HANDLE CASES WHEN M = 1   HERE                                **00013000
C**--------------------------------------------------------------------**00013100
```

```
  110 CONTINUE                                                          00013200
      IF (N .EQ. 2 .AND. ND .GE. 3) GO TO 500                           00013300
      IF(ND .LT. 0) RETURN1                                             00013400
      IF (N .GT. 2) GO TO 111                                           00013500
      IF (ND .NE. 0) GO TO 115                                          00013600
      V(1,1)=(D(1,2)-D(1,1))/S(1)                                       00013700
      V(1,2)=V(1,1)                                                     00013800
      GO TO 500                                                         00013900
  111 CONTINUE                                                          00014000
      IF (N .EQ. 2) GO TO 500                                           00014100
      DO 115 I = 2 , NM1                                                00014200
      V(1,I) = (D(1,I+1)-D(1,I-1)) / (S(I-1)+S(I))                      00014300
  115 CONTINUE                                                          00014400
      IF (ND .EQ. 3) GO TO 500                                          00014500
      IF (ND .NE. 1)                                                    00014600
     & V(1,1) = (3.0*(D(1, 2 )-D(1,1))-V(1, 2 )*S( 1 ))/(2.0*S( 1 ))    00014700
      IF (ND .NE. 2)                                                    00014800
     & V(1,N) = (3.0*(D(1,N)-D(1,N-1))-V(1,N-1)*S(N-1))/(2.0*S(N-1))    00014900
      GO TO 500                                                         00015000
C**-------------------------------------------------------------------**00015100
C**       HANDLE CASES WHEN M IS GREATER THAN 1                       **00015200
C**       (NORMALIZE END VECTORS)                                     **00015300
C**-------------------------------------------------------------------**00015400
  118 CONTINUE                                                          00015500
      V1SUM = 0.0                                                       00015600
      VNSUM = 0.0                                                       00015700
      DO  120  I = 1, M                                                 00015800
      V1SUM = V1SUM + V(I,1)*V(I,1)                                     00015900
  120 VNSUM = VNSUM + V(I,N)*V(I,N)                                     00016000
      V1SUM = SQRT(V1SUM)                                               00016100
      VNSUM = SQRT(VNSUM)                                               00016200
      DO  122  I = 1, M                                                 00016300
      V(I,1) = V(I,1)/V1SUM                                             00016400
      V(I,N) = V(I,N)/VNSUM                                             00016500
  122 CONTINUE                                                          00016600
C**                                                                     00016700
C**                                                                     00016800
  124 IF (N .EQ. 2) RETURN                                              00016900
      R(1) = 0.0                                                        00017000
      DO 160 I = 1, M                                                   00017100
       DO  125  ID = 1, NM1                                             00017200
        A(ID) = D(I,ID+1) - D(I,ID)                                     00017300
        B(ID) = 3.*A(ID)                                                00017400
  125  CONTINUE                                                         00017500
       B(NM1) = B(NM1) - V(I,N)*S(NM1)                                  00017600
       DO  130  ID = 2, NM1                                             00017700
        R(ID) = S(ID-1)/S(ID)                                           00017800
        V(I,ID) = 2.*B(ID-1) - 3.*A(ID-1) + R(ID)*R(ID)*B(ID)           00017900
  130  CONTINUE                                                         00018000
      V(I,2) = V(I,2) - V(I,1)*S(1)                                     00018100
  160 CONTINUE                                                          00018200
C**                                                                     00018300
      IF (NM1 .EQ. 2) GO TO  200                                        00018400
      DO  180  ID = 2, NM2                                              00018500
       F = 2.*R(ID)*(1. + R(ID)) - R(ID-1)                              00018600
       DO  170  I = 1, M                                                00018700
        V(I,ID) = V(I,ID)/F                                             00018800
        V(I,ID+1) = V(I,ID+1) - V(I,ID)                                 00018900
  170  CONTINUE                                                         00019000
       R(ID) = R(ID)*R(ID)*R(ID+1)/F                                    00019100
  180 CONTINUE                                                          00019200
C**                                                                     00019300
  200 F = (2. + 2.*R(NM1))*R(NM1) - R(NM2)                              00019400
      DO   205  L = 1, M                                                00019500
       V(L,NM1) = V(L,NM1)/F                                            00019600
  205 CONTINUE                                                          00019700
C**                                                                     00019800
```

```
      IF (NM1 .EQ. 2) GO TO  225                                        00019900
      DO  220  K = 2, NM2                                               00020000
       ID = NM2 + 2 - K                                                 00020100
       DO  220  I = 1, M                                                00020200
        V(I,ID) = V(I,ID) - V(I,ID+1)*R(ID)                             00020300
  220 CONTINUE                                                          00020400
C**                                                                     00020500
C**  APPLY THE HOMOGENEOUS COORDINATE TO THE VECTORS.                   00020600
C**                                                                     00020700
 225  DO  260  I = 1, M                                                00020800
      DO 260 ID = 2, NM1                                                00020900
        V(I,ID) = V(I,ID)/S(ID)                                         00021000
  260 CONTINUE                                                          00021100
C**                                                                     00021200
  500 CONTINUE                                                          00021300
      RETURN                                                            00021400
      END                                                               00021500
*************************************************************************
*************************************************************************
* SUBROUTINE: BAYS(SPLPTS,TANVEC,NNODE,NUM)
*
*************************************************************************
*************************************************************************
*   THIS SUBROUTINE RECEIVES AN ARRAY OF POINTS AND AN ARRAY OF THEIR
*   CORRESPONDING TANGENT VECTORS(AS DETERMINED BY A 3D SPLINE ROUTINE)
*   AND BREAKS DOWN THE DATA INTO TWO CONSECUTIVE POINTS AND THE
*   CORRESPONDING TAN VECTORS OF THOSE TWO POINTS.
*   THE SMALLER RESULTING ARRAY IS THEN PASSED TO A ROUTINE WHICH
*   CALCULATES A CUBIC BETWEEN THOSE TWO POINTS.
*
*      INPUT:
*
*  SPLPTS  IS THE INPUT ARRAY OF POINTS FROM ONE SPLINE OF ONE NETWORK
*  TANVEC  IS THE INPUT ARRAY OF TANGENT VECTORS OF THE POINTS IN
*          THE ARRAY SPLPTS
*  NNODE   IS THE NUMBER OF POINTS WHICH MAKE UP THE SPLINE
*  NUM     IS THE NUMBER OF POINTS ON THE CUBIC NECESSARY IN THE
*          STRAIGHT LINE APPROXIMATION OF THE CURVE
*
*    LOCAL STUFF:
*  ENDPTS  IS THE ARRAY OF ENDPOINTS OF THE CUBIC TO BE CALCULATED
*          AND DIVIDED INTO LINEAR SEGMENTS OF NUMBER NUM-1
*  PU0     IS THE TANGENT OF THE FIRST ENDPT OF THE CUBIC
*  PU1     IS THE TANGENT OF THE SECOND ENDPT OF THE CUBIC
*************************************************************************
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/27/88
*************************************************************************
*************************************************************************
      SUBROUTINE BAYS(SPLPTS,TANVEC,NNODE,NUM)
      REAL SPLPTS(3,*),TANVEC(3,*),ENDPTS(3,2),PU0(3),PU1(3)
      INTEGER NUM
C
      DO 10 I = 1,NNODE-1
         DO 20 J = 1,3

            ENDPTS(J,1) = SPLPTS(J,I)
            ENDPTS(J,2) = SPLPTS(J,I+1)

            PU0(J) = TANVEC(J,I)
            PU1(J) = TANVEC(J,I+1)

   20    CONTINUE

C  NOW CALL ROUTINE TO CALCULATE AND DRAW CUBIC APPROXIMATION
         CALL CUBCRV(ENDPTS,PU0,PU1,NUM)
```

```
C  LOOP THRU UNTIL FINISHED WITH THIS SPLINE

   10 CONTINUE

      RETURN
      END
*******************************************************************************
*******************************************************************************
* SUBROUTINE: CUBCRV(ENDPTS,PU0,PU1,NUM)
*
*******************************************************************************
*******************************************************************************
*   THIS SUBROUTINE CALCULATES A CUBIC CURVE GIVEN THE INPUT OF THE
*   TWO ENDPOINTS AND THEIR RESPECTIVE TANGENT VECTORS.
*   IT ALSO DRAWS THE APPROXIMATED CUBIC USING LINE SEGMENTS DETERMINED
*   BY A SPECIFIED NUMBER OF NODE POINTS(NUM)
*
* INPUT:
*
* ENDPTS  THE INPUT ARRAY OF THE ENDPOINTS OF THE CUBIC SEGMENT
* P0U     THE INPUT TANGENT OF THE FIRST POINT OF THE CUBIC SEGMENT
* P1U     THE INPUT TANGENT OF THE LAST POINT OF THE CUBIC SEGMENT
* NUM     THE INPUT NUMBER OF POINTS THE CUBIC MUST BE LATER BROKEN
*         INTO IN ORDER TO BE APPROXIMATED BY STRAIGHT LINES DRAWN
*         WITH PHIGS
*******************************************************************************
*******************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/27/88
*******************************************************************************
*******************************************************************************
      SUBROUTINE CUBCRV(ENDPTS,P0U,P1U,NUM)
      REAL ENDPTS(3,2),P0U(3),P1U(3)
      REAL B(4,3),M(4,4),PROD1(4,3)
      REAL PP(3,50)
      INTEGER NUM
      DATA M/2.,-3.,0.,1.,-2.,3.,0.,0.,1.,-2.,1.,0.,1.,-1.,0.,0./


C PUT ENDPOINTS AND TANGENTS INTO ARRAY OF GEOM VARIABLES

      DO 10 I = 1,3
         B(1,I) = ENDPTS(I,1)
         B(2,I) = ENDPTS(I,2)
         B(3,I) = P0U(I)
         B(4,I) = P1U(I)
   10 CONTINUE

C IF THE EQUATION OF THE CUBIC IS P = W*M*B WHERE W IS THE 1X4
C U MATRIX, M IS THE 4X4 COEFFICIENT MATRIX AND B IS THE MATRIX
C OF GEOMETRIC COEFFS(4X3)
C
C NOW CALCULATE AN INTERMEDIATE MATRIX CALLED PROD1 = M*B
      CALL MTXMLT(M,B,4,4,3,PROD1)

C  NOW CALL A SUBROUTINE WHICH CALCULATES THE CUBIC AND DIVIDES IT INTO
C  SOME NUMBER OF SEGMENTS AS SPECIFIED BY NUM -- NUMBER OF SEGS WILL
C  BE NUM-1
      CALL DIVCRV(PROD1,NUM,PP)

C  NOW PASS THOSE POINTS TO A ROUTINE WHICH WILL DRAW THE 3D LINE
C  SEGMENTS
      CALL DRWSEG(PP,NUM)

      RETURN
```

```
      END
*****************************************************************************
*****************************************************************************
* SUBROUTINE: DIVCRV(PROD1,NUM,PP)
*
*****************************************************************************
*****************************************************************************
*   THIS SUBROUTINE CALCULATES THE POINTS NECESSARY FOR THE STRAIGHT
*   LINE APPROXIMATION OF A CUBIC
*
*  INPUT:
*
* PROD1   IS THE INTERMEDIATE MATRIX PRODUCT OF THE BLENDING COEFF
*         MATRIX AND THE GEOMETRIC COEFF MATRIX
* NUM     IS THE SPECIFIED NUMBER OF POINTS INTO WHICH THE CUBIC WILL
*         WILL BE DIVIDED
* PP      IS THE OUTPUT ARRAY OF POINTS ON THE CUBIC WHICH WILL BE
*         PASSED TO A PHIGS 3D LINE ROUTINE
*****************************************************************************
*****************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/27/88
*****************************************************************************
*****************************************************************************
      SUBROUTINE DIVCRV(PROD1,NUM,PP)
      REAL PROD1(4,*),PP(3,*)
      REAL P(1,3),W(4)
      INTEGER NUM

C  CALCULATE THE MATRIX WHICH CONTAINS THE PARAMETER U --
C  U GOES FROM 0 TO 1 BUT WILL BE SUBDIVIDED INTO NUM PARTS
      DO 10 I = 0,NUM-1
         U = FLOAT(I)/FLOAT(NUM-1)

C  NOW PUT THE U'S INTO THE ARRAY AND MULT BY PROD1 TO GET THE RESULTING
C  POINTS ON THE CUBIC
         W(1) = U**3
         W(2) = U**2
         W(3) = U
         W(4) = 1.

C  DO THE MATRIX MULTIPLICATION
         CALL MTXMLT(W,PROD1,1,4,3,P)

C  WRITE THE PRODUCT ,P, INTO ANOTHER MATRIX TO PASS OUT

         DO 20 J = 1,3
            PP(J,I+1) = P(1,J)
   20    CONTINUE
   10 CONTINUE

      RETURN
      END
*****************************************************************************
*****************************************************************************
* SUBROUTINE: DRWSEG(PP,NUM)
*
*****************************************************************************
*****************************************************************************
*  THIS SUBROUTINE DRAWS THE LINE SEGMENTS WHICH APPROXIMATE A CUBIC
*  GIVEN THE POINTS ON THE CUBIC WHICH ARE USED AS APPROXIMATIONS
*
*  INPUT:
*
* PP  IS THE INPUT ARRAY OF POINTS THAT NEED TO BE FED TO PHIGS FOR
*     THE STRAIGHT LINE APPROXIMATIONS
* NUM IS THE NUMBER OF POINTS IN THE CUBIC SEGMENT
```

```
*
**********************************************************************
**********************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/27/88
**********************************************************************
**********************************************************************
      SUBROUTINE DRWSEG(PP,NUM)
      REAL PP(3,*),PX(50),PY(50),PZ(50)
      INTEGER NUM

C  FOR PHIGS CALL THE POINTS MUST BE IN AN X,Y AND Z ARRAY

      DO 10 I = 1,NUM
         PX(I) = PP(1,I)
         PY(I) = PP(2,I)
         PZ(I) = PP(3,I)
   10 CONTINUE

C NOW CALL THE PHIGS ROUTINE
      CALL PPL3(NUM,PX,PY,PZ)

      RETURN
      END
**********************************************************************
**********************************************************************
* SUBROUTINE: PVLIN(WKID,IDNR,VIEW,VALUE)
*
**********************************************************************
* DESCRIPTION:
*     PROCESSES VALUATOR INPUT
*
*     INPUT:
*          WKID -- WORKSTATION IDENTIFIER
*          IDNR -- DEVICE NUMBER OR INPUT DEVICE
*          VIEW -- VIEW IDENTIFIER OF THE PRIMARY VIEW
*          VALUE -- INPUT ARRAY OF VALUATOR VALUES
*     OUTPUT:
*          VALUE -- OUTPUT ARRAY OF UPDATED VALUATOR VALUES
*
**********************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 3/16/88
**********************************************************************
**********************************************************************
      SUBROUTINE PVLIN(WKID,IDNR,VIEW,VALUE)

      INTEGER WKID,IDNR,VIEW
      REAL VALUE(*),MATD(4,4)


C GET VALUATOR INPUT
      CALL PGTVL(VALUE(IDNR))

C GET VALUATOR MATRIX
      CALL GTVMAT(WKID,VIEW,VALUE,MATD)

C SET VIEW MATRIX
      CALL PSVWM3(WKID,VIEW,MATD)

      RETURN
      END
**********************************************************************
**********************************************************************
**********************************************************************
* SUBROUTINE: GTVMAT(WKID,VIEW,VALUE,MATD)
*
```

```
*************************************************************************
* DESCRIPTION: CALCULATES THE GLOBAL TRANSFORMATION MATRIX FOR A VIEW
*
* INPUT:
*      WKID  -- WORKSTATION ID
*      VIEW  -- VIEW ID OF VIEW IN WHICH GLOBAL X-FORM IS TAKING PLACE
*      VALUE -- ARRAY OF VALUATOR VALUES
*
* OUTPUT:
*      MATD  -- TRANSFORMATION MATRIX
*
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 2/20/88
*************************************************************************
*************************************************************************
      SUBROUTINE GTVMAT(WKID,VIEW,VALUE,MATD)

      INTEGER WKID,VIEW
      REAL VALUE(*),MATD(4,4),MATRX(4,4),MATRY(4,4)
      REAL MATRZ(4,4),MATSC(4,4),MATTR(4,4),MATA(4,4),MATB(4,4)
      REAL MATC(4,4),ROTX,ROTY,ROTZ,SCALE(3),TRAN(3)
      REAL PI,CONVER
      PARAMETER (PI=3.14159,CONVER=PI/180.)

C CALCULATE MATRICES FOR THE DIFFERENT VALUATOR CHOICES
C ROTATION ABOUT X
         ROTX = VALUE(1)*CONVER
         CALL PROX(ROTX,MATRX)

C ROTATION ABOUT Y
         ROTY = VALUE(2)*CONVER
         CALL PROY(ROTY,MATRY)

C ROTATION ABOUT Z
         ROTZ = VALUE(3)*CONVER
         CALL PROZ(ROTZ,MATRZ)

C SCALE(ZOOM)
         CALL PSC3(VALUE(4),VALUE(4),VALUE(4),MATSC)

C TRANSLATION
         CALL PTR3(VALUE(5),VALUE(6),VALUE(7),MATTR)

C CONCATENATE IN ORDER ROTATE, SCALE,TRANSLATE
      CALL PCOM3(MATRX,MATRY,MATA)
      CALL PCOM3(MATA,MATRZ,MATB)
      CALL PCOM3(MATB,MATSC,MATC)
      CALL PCOM3(MATC,MATTR,MATD)

      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: DELXF(VALUE,PLID,IDNR)
*
*************************************************************************
* DESCRIPTION: THIS SUBROUTINE DELETES THE TRANSLATION MATRIX FROM
*              THE INTERSECTION PLANE STRUCTURE AND INSERTS A NEW
*              TRANSLATION MATRIX BASED ON THE VALUE OF VALUATOR 8
*      INPUT:
*         VALUE -- ARRAY CONTAINING VALUATOR VALUES
*         PLID -- STRUCTURE ID OF THE AIRCRAFT
*         IDNR -- INPUT DEVICE NUMBER
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 5/3/88
```

```
************************************************************************
************************************************************************
      SUBROUTINE DELXF(VALUE,PLID,IDNR)
      REAL VALUE(*),XFORM(4,4)
      INTEGER PLID,IDNR

C GET VALUATOR VALUE
      CALL PGTVL(VALUE(IDNR))

C OPEN INTERSECTION PLANE STRUCTURE
      CALL POPST(PLID)

C DELETE ELEMENT BETWEEN FIRST AND SECOND LABELS
      CALL PDELLB(90,91)

C CALCULATE NEW TRANSLATION MATRIX
      CALL PTR3(VALUE(IDNR),0.,0.,XFORM)

C INSERT NEW TRANSFORMATION MATRIX
      CALL PSLMT3(XFORM,2)

C CLOSE STRUCTURE
      CALL PCLST

      RETURN
      END
************************************************************************
************************************************************************
* SUBROUTINE: DETPIK(WKID,PTDATA,INTARR,VALUE)
*
************************************************************************
* DESCRIPTION: DETERMINES WHAT WAS PICKED AND WHAT TO DO WITH IT
*
*      INPUT:
*         WKID -- WORKSTATION IDENTIFIER
*         PTDATA -- ARRAY CONTAINING DATA WHICH REPRESENTS THE AIRCRAFT
*         VALUE -- ARRAY CONTAINING VALUATOR VALUES
*
*     OUTPUT:
*         INTARR -- ARRAY CONTAINING THE POINTS OF INTERSECTION
*
************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 5/4/88
************************************************************************
************************************************************************
      SUBROUTINE DETPIK(WKID,PTDATA,INTARR,VALUE)
      INTEGER IPPD,STAT,PPD,WKID
      PARAMETER (IPPD = 20)
      INTEGER PP(3,IPPD)
      REAL PTDATA(3,100,50,*),INTARR(3,100,*),VALUE(*)
      REAL ARARR(10,400),TAREA(400)

C GET PICK
      CALL PGTHP(IPPD,STAT,PPD,PP)

C IF IT'S PICK #1 THEN
C     DISPLAY THE WHOLE AIRCRAFT
      IF (PP(2,2) .EQ. 1) THEN
         CALL MSG(WKID,'DISPLAYING ENTIRE AIRCRAFT')
         CALL WHOLE(WKID)

C     DISPLAY HALF OF THE AIRCRAFT
      ELSE IF (PP(2,2) .EQ. 2) THEN
         CALL MSG(WKID,'DISPLAYING HALF OF THE AIRCRAFT')
         CALL HALF(WKID)
```

```
C     DISPLAY THE INTERSECTION PLANE
      ELSE IF (PP(2,2) .EQ. 4) THEN
         CALL MSG(WKID,'DISPLAYING INTERSECTION PLANE')
         CALL PSIVFT(WKID,0,1,1,2)

C     NO-SHOW THE INTERSECTION PLANE
      ELSE IF (PP(2,2) .EQ. 5) THEN
         CALL PSIVFT(WKID,1,2,0,1)
         CALL MSG(WKID,'NO-SHOWING INTERSECTION PLANE')

C     CALCULATE THE INTERSECTION AUTOMATICALLY
      ELSE IF (PP(2,2) .EQ. 7) THEN
         CALL MSG(WKID,'COMPUTING AUTOMATIC INTERSECTION')
         CALL PUWK(WKID,1)
         CALL AUTOIN(WKID,PTDATA,ARARR,TAREA)

C     CALCULATE THE INTERSECTION WITH USER ASSISTANCE
      ELSE IF (PP(2,2) .EQ. 8) THEN
         CALL BMAN(WKID,PTDATA,VALUE,ARARR,TAREA)

C     WRITE MESSAGE
      ELSE IF (PP(2,2) .EQ. 9) THEN
         CALL MSG(WKID,'YOU MUST SELECT BEG MANUAL FIRST')

C     WRITE MESSAGE
      ELSE IF (PP(2,2) .EQ. 10) THEN
         CALL MSG(WKID,'YOU MUST SELECT BEG MANUAL FIRST')

C     DISPLAY THE GRAPHICAL AREA DISTRIBUTION
      ELSE IF (PP(2,2) .EQ. 12) THEN
         CALL DISPAR(WKID,ARARR,TAREA)

C     DISPLAY AREA DISTRIB. IN FULL SCREEN
      ELSE IF (PP(2,2) .EQ. 13) THEN
         CALL FULLVW(WKID)

C     DISPLAY AREA DISTRIB. IN 1/4 SCREEN
      ELSE IF (PP(2,2) .EQ. 14) THEN
         CALL QRTVW(WKID)

C     CREATE A FILE WHICH CONTAINS INTERSECTION DATA
      ELSE IF (PP(2,2) .EQ. 16) THEN
         CALL MSG(WKID,'PLEASE ENTER NEW FN AND FT')
         CALL PUWK(WKID,1)
         CALL SAVEFL(WKID)
      ELSE
         WRITE(*,*)'YOU MESSED UP YOUR PICK'
      END IF

      RETURN
      END
*****************************************************************
*****************************************************************
* SUBROUTINE: WHOLE(WKID)
*
*****************************************************************
* DESCRIPTION:
*     DISPLAYS AND ALLOWS MANIPULATION OF THE ENTIRE AIRCRAFT
*     INPUT:
*        WKID -- WORKSTATION IDENTIFIER
*
*****************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/07/88
*****************************************************************
*****************************************************************
      SUBROUTINE WHOLE(WKID)
```

```
      INTEGER WKID,ALLAIR ,STID
      REAL MATRIX(4,4)
      PARAMETER (ALLAIR = 21,STID = 10)

C UN-POST PREVIOUS HALF STRUCTURE
      CALL PUPORT(WKID,STID)

C OPEN NEW STRUCTURE
      CALL POPST(ALLAIR)

C INSERT VIEW ID
      CALL PSVWI(1)

C EXECUTE THAT STRUCTURE
      CALL PEXST(STID)

C REFLECT THE DATA POINTS
      CALL GTSMMT(2,MATRIX)

C INSERT MATRIX INTO THE STRUCTURE
      CALL PSLMT3(MATRIX,0)

C EXECUTE THAT STRUCTURE AGAIN
      CALL PEXST(STID)

C CLOSE THE STRUCTURE
      CALL PCLST

C POST ROOT
      CALL PPORT(WKID,ALLAIR,.9)

      RETURN
      END
*********************************************************************
*********************************************************************
* SUBROUTINE: HALF(WKID)
*
*********************************************************************
* DESCRIPTION:
*     RESTORES THE ORIGINAL HALF OF THE AIRCRAFT STRUCTURE
*     INPUT:
*        WKID -- WORKSTAION IDENTIFIER
*
*********************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/07/88
*********************************************************************
*********************************************************************
      SUBROUTINE HALF(WKID)
      INTEGER WKID,STID,ALLAIR
      PARAMETER (STID = 10, ALLAIR = 21)

C UNPOST THE WHOLE AIRCRAFT STRUCTURE
      CALL PUPORT(WKID,ALLAIR)

C POST THE HALF STRUCTURE
      CALL PPORT(WKID,STID,0.9)

      RETURN
      END
*********************************************************************
*********************************************************************
* SUBROUTINE: AUTOIN(WKID,PTDATA,ARARR,TAREA)
*
*********************************************************************
* DESCRIPTION:
```

```
*     READS IN DELTA AREA OVER LENGTH CRITERIA FROM SCREEN AND
*     DETERMINES IF INTERSECTION CRITERIA HAS BEEN MET
*     INPUT:
*        WKID -- WORKSTATION IDENTIFIER
*        PTDATA -- ARRAY CONTAINING AIRCRAFT DATA
*     OUTPUT:
*       ARARR -- ARRAY CONTAINING AREA DATA BY COMPONENT
*       TAREA -- ARRAY CONTAINING AREA DATA FOR ENTIRE AIRCRAFT
*************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/07/88
*************************************************************
*************************************************************
      SUBROUTINE AUTOIN(WKID,PTDATA,ARARR,TAREA)
      INTEGER WKID,STRN,CHOICE,INCR
      REAL TIME,XB(2),PTDATA(3,100,50,*),INTARR(3,100,10)
      REAL ARARR(10,400),TAREA(*),CRIT
      CHARACTER*20 STR
      PARAMETER(STRN = 6,CHOICE = 4,INCR = 10)

      COMMON /INSECT/ INTCNT,INTNUM
      INTEGER INTCNT,INTNUM(800)

      COMMON /LOCN/ LOC
      REAL LOC(800)

      COMMON /INTPT/ QPT,QCNT
      REAL QPT(3,800)
      INTEGER QCNT

      COMMON /MAXGRF/ MAXAR
      REAL MAXAR

      STR = '                    '
      MAXAR = 0.
C WRITE A MESSAGE TO ENTER THE DELTA AREA OVER LENGTH CRITERIA
      CALL MSG(WKID,'COMPUTING AUTOMATIC INTERSECTION')
      CALL MSG(WKID,'ENTER DELTA AREA CRITERIA')
      CALL PUWK(WKID,1)

C OPEN A FILE FOR INTARR DATA
      OPEN(FILE = 'DEFALT',UNIT = 98)

C AWAIT STRING INPUT
 100  CALL STRIN(WKID,LOSTR,STR)
           READ(STR,*) CRIT

C NOW CALL AUTO AND DO SOME INTERSECTIN
C BUT FIRST DETERMINE THE MAX AND MIN X LOCATIONS OF THE AIRCRAFT
C (THE AIRCRAFT GOES FROM WHAT TO WHAT?)
      CALL XBOUND(XB)

C INITIALIZE THE INTCNT COUNTER
      INTCNT = 0
      XLOC = XB(1)

C LOOP THRU THE INTERSECTIONS OF THE AIRCRAFT
      DO 10 I = INT(XB(1)),INT(XB(2)),INCR
         INTCNT = INTCNT + 1
         RINCR = FLOAT(INCR)
   99    CALL INBND(PTDATA,XLOC,INTARR,ARARR,TAREA)

C IF NOT THE FIRST TIME THRU THEN
         IF (INTCNT .NE. 1) THEN
            TDIFF = ABS(TAREA(INTCNT) - TAREA(INTCNT - 1))
```

```
C SEE IF TDIFF MEETS THE AREA CRITERIA ,IF YES WRITE INT DATA TO A FILE
C IF NO RE-EVALUATE THE X LOCATION AND RECALCULATE INTERSECTION
            IF(TDIFF .GT. CRIT) THEN
               XLOC = XLOC - RINCR/2.
               RINCR = RINCR/2.
               GO TO 99
            END IF
         END IF

C WRITE INTERSECTION DATA TO A FILE IF THE AREA CRITERIA WAS NOT VIOLATE
      DO 22 L = 1,QCNT
   22    WRITE(98,101)(QPT(LL,L),LL = 1,3)
  101    FORMAT(1X,3(F9.4))

C CALCULATE THE NEXT X LOCATION
      XLOC = XLOC + FLOAT(INCR)
   10 CONTINUE

C CLOSE INTARR DATA FILE
      CLOSE(UNIT = 98)

      RETURN
      END
*****************************************************************
*****************************************************************
* SUBROUTINE: XBOUND(XB)
*
*****************************************************************
* DESCRIPTION:
*     DETERMINES THE BEGINNING AND ENDING X LOCATIONS ON THE AIRCRAFT
*     OUTPUT:
*        XB -- ARRAY CONTAINING THE MIN AND MAX EXTENTS OF AIRCRAFT
*****************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/07/88
*****************************************************************
*****************************************************************
      SUBROUTINE XBOUND(XB)
      REAL XB(2)


      COMMON /DPNTS/ NODE,XS,NET,NNODE,NXS
      INTEGER NODE,XS,NET,NNODE(20),NXS(20)

      COMMON /BORDER/ MAX,MIN
      REAL MAX(20),MIN(20)

C FIND THE MAX MAX AND THE MIN MIN
         XB(1) = MIN(1)
         XB(2) = MAX(1)
      DO 10 I = 2,NET
         IF(MIN(I) .LT. XB(1))  XB(1) = MIN(I)
         IF(MAX(I) .GT. XB(2))  XB(2) = MAX(I)
   10 CONTINUE
      RETURN
      END
*****************************************************************
*****************************************************************
* SUBROUTINE: BMAN(WKID,PTDATA,VALUE,ARARR,TAREA)
*
*****************************************************************
* DESCRIPTION:
*     COMPUTES A MANUAL INTERSECTION FOR THE AIRCRAFT
*     INPUT:
*        WKID -- WORKSTATION IDENTIFIER
*        PTDATA -- ARRAY CONTAINING AIRCRAFT DATA
*        VALUE -- ARRAY CONTAINING VALUATOR VALUES
```

```
*     OUTPUT:
*       ARARR -- ARRAY CONTAINING AREA DATA BY COMPONENT
*       TAREA -- ARRAY CONTAINING AREA DATA FOR ENTIRE AIRCRAFT
*
*****************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/07/88
*****************************************************************
*****************************************************************
      SUBROUTINE BMAN(WKID,PTDATA,VALUE,ARARR,TAREA)
      REAL PTDATA(3,100,50,*),VALUE(8),ARARR(10,*),TAREA(*)
      REAL TIME,INTARR(3,100,10)
      INTEGER WKID,PLID,VAL,PICK,CHOICE,PP(3,20),STAT,PPD,IPPD
      PARAMETER(TIME = 3600.,PLID = 20,VAL = 3,CHOICE = 4,PICK = 5)
      PARAMETER(IPPD = 20)
      CHARACTER*6 STRING(1)

      COMMON /INSECT/ INTCNT,INTNUM
      INTEGER INTCNT,INTNUM(800)

      COMMON /MAXGRF/ MAXAR
      REAL MAXAR

      COMMON /INTPT/ QPT,QCNT
      REAL QPT(3,800)
      INTEGER QCNT

C INITIALIZE INTERSECTION COUNT AND MAXIMUM AREA
      INTCNT = 0
      MAXAR = 0.

C OPEN FILE IN WHICH TO STORE INTERSECTION ARRAY
      STRING(1) = 'DEFALT'
      OPEN(FILE = STRING(1),UNIT = 98)

C AWAIT FOR CHOICE INPUT
      CALL MSG(WKID,'HIT CHOICE BUTTON TO SIGNIFY INTERSECTION')
  100 CALL PUWK(WKID,1)
      CALL PWAIT(TIME,KWKID,ICL,IDNR)

C IF IT WAS VALUATOR INPUT DO SOMETHING WITH IT
      IF(ICL .EQ. VAL) THEN
         IF(IDNR .EQ. 8) THEN
            CALL DELXF(VALUE,PLID,IDNR)
         END IF
         GO TO 100

C IF IT WAS A PICK THEN IT HAS TO BE END MANUAL
      ELSE IF(ICL. EQ. PICK) THEN
         CALL PGTHP(IPPD,STAT,PPD,PP)
         IF(PP(2,2) .EQ. 9) THEN
            CLOSE(UNIT = 98)
            CALL MSG(WKID,'MANUAL INTERSECTION SESSION ENDED')
            RETURN
         ELSE IF (PP(2,2) .EQ. 10) THEN
            CALL MSG(WKID,'YOU HAVE EXITED MANUAL INTERSECTION')
            RETURN
         ELSE
            CALL MSG(WKID,'PLEASE END MANUAL SESSION OR EXIT IT')
            GO TO 100
         END IF

C IF IT WAS A CHOICE THEN
      ELSE IF(ICL .EQ. CHOICE) THEN

         XLOC = VALUE(8)
         INTCNT = INTCNT + 1
```

```
         CALL INBND(PTDATA,XLOC,INTARR,ARARR,TAREA)
         WRITE(98,22)QPT
   22    FORMAT(1X,3(F9.4))
         CALL MSG(WKID,'GO FOR ANOTHER INTERSECTION OR END MANUAL')
         GO TO 100
      END IF


      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: DISPAR(WKID,ARARR,TAREA)
*
*************************************************************************
* DESCRIPTION:
*     DISPLAYS THE GRAPH OF AREA DISTRIBUTION AS A FUNCTION OF
*     STATION ALONG THE MAIN AXIS OF THE AIRCRAFT
*     INPUT:
*        WKID -- WORKSTATION IDENTIFIER
*     OUTPUT:
*       ARARR -- ARRAY CONTAINING AREA DATA BY COMPONENT
*       TAREA -- ARRAY CONTAINING AREA DATA FOR ENTIRE AIRCRAFT
*
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/20/88
*************************************************************************
*************************************************************************
      SUBROUTINE DISPAR(WKID,ARARR,TAREA)
      INTEGER WKID
      REAL ARARR(10,400),TAREA(400)
      REAL VLIM(4),PLIM(4),XB(2)

      COMMON /MAXGRF/ MAXAR
      REAL MAXAR

C FIND THE AIRCRAFT MAX AND MIN SO YOU CAN SET UP THE GRAPH AREA
      CALL XBOUND(XB)

C SET UP VIEW IN WHICH TO PUT THE AREA DISTRIBUTION GRAPHS
C SET UP WINDOW AND VIEWPORT LIMITS FOR A 2D VIEW
      VLIM(1) = (XB(1) - 100.)
      VLIM(2) = (XB(2) + 100.)
      VLIM(3) = (-100.)
      VLIM(4) = (MAXAR + 200.)

      PLIM(1) = 0.
      PLIM(2) = 0.8
      PLIM(3) = 0.15
      PLIM(4) = 0.8

C SET VIEW MAPPING
      CALL PSVMP(WKID,3,VLIM(1),VLIM(2),VLIM(3),VLIM(4),
     $           PLIM(1),PLIM(2),PLIM(3),PLIM(4))

C SET VIEW CHARACTERISTICS
      CALL PSVWCS(WKID,3,0,0,0,1,0)

C EMPTY THE GRAPH STRUCTURE
      CALL PEMST(300)

C DRAW BACKGROUND FILL AREA FOR VIEW IN BLACK
      CALL DRBACK(WKID,VLIM)

C DRAW STRUCTURE WHICH CONTAINS THE GRAPHICAL DATA FOR AREA DISTRIBUTION
      CALL DRAREA(WKID,VLIM,ARARR,TAREA)
```

```
      RETURN
      END
*****************************************************************************
*****************************************************************************
* SUBROUTINE: DRBACK(WKID,VLIM)
*
*****************************************************************************
* DESCRIPTION:
*     DRAWS A BLACK BACKGROUND FILL AREA ON WHICH TO PLACE THE
*     GRAPH OF AREA DISTRIBUTION
*     INPUT:
*        WKID -- WORKSTATION IDENTIFIER
*        VLIM -- LIMITS OF THE BACKGROUND WINDOW
*
*****************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/20/88
*****************************************************************************
*****************************************************************************
      SUBROUTINE DRBACK(WKID,VLIM)
      INTEGER WKID
      REAL VLIM(4),PX(4),PY(4)

C OPEN A STRUCTURE IN WHICH TO PLACE THE BACKGROUND FILL AREA
      CALL POPST(301)

C SET VIEW ID
      CALL PSVWI(3)

C DRAW FILL AREA IN BLACK
      CALL PSIS(1)
      CALL PSICI(0)

C DEFINE POINTS WHICH DEFINE FILL AREA
      PX(1) = VLIM(2)
      PX(2) = VLIM(1)
      PX(3) = VLIM(1)
      PX(4) = VLIM(2)

      PY(1) = VLIM(3)
      PY(2) = VLIM(3)
      PY(3) = VLIM(4)
      PY(4) = VLIM(4)

      CALL PFA(4,PX,PY)

C CLOSE STRUCTURE
      CALL PCLST

C POST IT AND GIVE IT A LOWER PRIORITY THAN THE GRAPH ITSELF
      CALL PPORT(WKID,301,.5)

      RETURN
      END
*****************************************************************************
*****************************************************************************
* SUBROUTINE: DRAREA(WKID,VLIM,ARARR,TAREA)
*
*****************************************************************************
* DESCRIPTION:
*     DRAWS THE AREA DISTRIBUTION FROM THE ARARR AREA ARRAY CREATED
*     DURING INTERSECTION
*     INPUT:
*        WKID -- WORKSTATION IDENTIFIER
*       VLIM -- WINDOW LIMITS
*       ARARR -- ARRAY CONTAINING AREA DATA BY COMPONENT
*       TAREA -- ARRAY CONTAINING AREA DATA FOR ENTIRE AIRCRAFT
```

```
*
**********************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/20/88
**********************************************************************
**********************************************************************
      SUBROUTINE DRAREA(WKID,VLIM,ARARR,TAREA)
      INTEGER WKID
      REAL ARARR(10,400),TAREA(400),VLIM(4)


      INTEGER CURNET,VIEWID
      REAL PX(400),PY(400)

      COMMON /DPNTS/ NODE,XS,NET,NNODE,NXS
      INTEGER NODE,XS,NET,NNODE(20),NXS(20)

      COMMON /INSECT/ INTCNT,INTNUM
      INTEGER INTCNT,INTNUM(800)

      COMMON /LOCN/ LOC
      REAL LOC(400)

C CREATE A GRAPH OF AREA AS A FUNCTION OF STATION ALONG THE X AXIS

C OPEN A STRUCTURE IN WHICH TO PUT THE GRAPH
         CALL POPST(300)
C SET VIEW ID
         CALL PSVWI(3)

C CALL ROUTINE TO DRAW AND LABEL AXES
         CALL DRAWAX(VLIM,ARARR)

C KEEP CROSS-SECTION CONSTANT AND GO FROM 1 TO # OF INTERSECTIONS
      DO 10 I = 1,NET
         DO 20 J = 1,INTCNT
            PX(J) = LOC(J)
C SET POLYLINE COLOR
            CALL PSPLCI(I+1)

C CREATE POINTS TO PUT INTO GRAPH
            PY(J) = ARARR(I,J)

            IF((J .GT. 2).AND.(J .LT. INTCNT-1)) THEN
               IF((PY(J) .EQ. 0.) .AND.(PY(J-1) .NE. 0.))THEN
                  PX(J) = LOC(J-1)
               ELSE IF((PY(J) .EQ. 0.).AND.(ARARR(I,J+1) .NE. 0.))THEN
                  PX(J) = LOC(J+1)
               ENDIF
            END IF
   20    CONTINUE

C MAKE A LINE
      CALL PPL(INTCNT,PX,PY)
   10 CONTINUE

C DRAW THE GRAPH REPRESENTING THE TOTAL AREA DISTRIBUTION CURVE
      CALL PSPLCI(1)
      DO 30 J = 1,INTCNT
         PY(J) = TAREA(J)
         PX(J) = LOC(J)
   30 CONTINUE
      CALL PPL(INTCNT,PX,PY)

C CLOSE STRUCTURE
      CALL PCLST
```

```
C POST STRUCTURE
      CALL PPORT(1,300,.9)


      RETURN
      END
*********************************************************************
*********************************************************************
* SUBROUTINE: DRAWAX(VLIM,ARARR)
*
*********************************************************************
* DESCRIPTION:
*     DRAWS AND LABELS AXES FOR THE AREA DISTRIBUTION GRAPH
*     INPUT:
*        VLIM -- WINDOW LIMITS
*        ARARR -- ARRAY WHICH CONTAINS AREA DATA BY COMPONENT
*********************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/04/88
*********************************************************************
*********************************************************************
      SUBROUTINE DRAWAX(VLIM,ARARR)
      REAL ARARR(10,*),PX(400),PY(400),VLIM(4)
      REAL CHH,CHXP,SPACE
      INTEGER INTER
      CHARACTER*5 CHAR(10),CHARY(8)
      CHARACTER*37 DESCRP

      COMMON /INSECT/ INTCNT,INTNUM
      INTEGER INTCNT,INTNUM(800)

      COMMON /LOCN/ LOC
      REAL LOC(400)

      DATA CHAR/'  0  ',
     $          ' 100 ',
     $          ' 200 ',
     $          ' 300 ',
     $          ' 400 ',
     $          ' 500 ',
     $          ' 600 ',
     $          ' 700 ',
     $          ' 800 ',
     $          ' 900 '/

      DATA CHARY/'  0  ',
     $           ' 1000',
     $           ' 2000',
     $           ' 3000',
     $           ' 4000',
     $           ' 5000',
     $           ' 6000',
     $           ' 7000'/


C SET POLYLINE COLOR TO WHITE
      CALL PSPLCI(1)
      MM = 0
      M = 0
C DRAW THE X AXIS
      PY(1) = 0.
      PY(2) = 0.
      PX(1) = LOC(1)
      PX(2) = VLIM(2) - 50.
      CALL PPL(2,PX,PY)

C DRAW THE Y AXIS
```

```
      PX(2) = LOC(1)
      PY(2) = VLIM(4) - 75.
      CALL PPL(2,PX,PY)

C SET THE CHARACTER HEIGHT, SPACING, AND EXPANSION FACTORS
C     CHH = VLIM(4) /(140.)
      CHH = 35.
C     READ(*,*)CHH
      CALL PSCHH(CHH)

C     CHXP = VLIM(2) / 150.
C     CHXP = 0.2
C     READ(*,*)CHXP
C     CALL PSCHXP(CHXP)

C     SPACE = VLIM(2) /75.
C     READ(*,*)SPACE
C     CALL PSCHSP(SPACE)
C MARK OF 50 UNIT INCREMENTS
      DO 10 I = INT(LOC(1)),INT(VLIM(2) - 50.),100

         DO 20 J = 1,2

            PY(1) = 15.
            PY(2) = -15.
            PX(J) = FLOAT(I)
   20    CONTINUE
         CALL PPL(2,PX,PY)

C PUT NUMBERS ON THE AXES
         M = M+1
         DY = -70.
         DX = FLOAT(I) - VLIM(2)/13.5
         CALL PTX(DX,DY,CHAR(M))

   10 CONTINUE

C MARK OFF UNITS OF 1000 FOR THE Y AXIS
      DO 60 I = 0,INT(VLIM(4)),1000
         DO 70 J = 1,2

            PY(J) = FLOAT(I)
            PX(1) = 15.
            PX(2) = -15.
   70    CONTINUE
C        CALL PPL(2,PX,PY)
C PUT NUMBERS ON THE Y AXIS
         MM = MM + 1
         DX = VLIM(1) + 20.
         DY = FLOAT(I)
         CALL PTX(DX,DY,CHARY(MM))

   60 CONTINUE


C LABEL THE X AXIS
      X = (VLIM(2) - VLIM(1))/2.
      Y = VLIM(3) + 25
C     READ(*,*)X,Y
      X = 25.
      Y = -250.

C MAKE SURE TEXT IS CENTERED
      CALL PSTXAL(2,3)

C WRITE GRAPH DESCRIPTION
      DESCRP = 'AREA DISTRIBUTION VS X'
```

```
      CALL PTX(X,Y,DESCRP)

      CALL PUWK(1,1)
      RETURN
      END
*****************************************************************************
*****************************************************************************
* SUBROUTINE: FULLVW(WKID)
*
*****************************************************************************
* DESCRIPTION:
*     DISPLAYS THE FULL SCREEN VIEW OF THE AREA DISTRIBUTION GRAPH
*     INPUT:
*        WKID -- WORKSTATION IDENTIFIER
*****************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/21/88
*****************************************************************************
*****************************************************************************
      SUBROUTINE FULLVW(WKID)
      INTEGER WKID
      REAL PLIM(4),VLIM(4)
      DATA PLIM/0.,0.8,0.15,0.8/,VLIM/-100.,900.,-100.,4500./


C REDEFINE THE VIEW MAPPING FOR FULL SCREEN
      CALL PSVMP(WKID,3,VLIM(1),VLIM(2),VLIM(3),VLIM(4),
     $           PLIM(1),PLIM(2),PLIM(3),PLIM(4))

      RETURN
      END
*****************************************************************************
*****************************************************************************
* SUBROUTINE: QRTVW(WKID)
*
*****************************************************************************
* DESCRIPTION:
*     DISPLAYS THE 1/4 SCREEN VIEW OF THE AREA DISTRIBUTION GRAPH
*     INPUT:
*        WKID -- WORKSTATION IDENTIFIER
*
*****************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/21/88
*****************************************************************************
*****************************************************************************
      SUBROUTINE QRTVW(WKID)
      INTEGER WKID
      REAL PLIM(4),VLIM(4)
      DATA PLIM/0.,0.4,0.15,0.4/,VLIM/-100.,900.,-100.,4500./


C REDEFINE THE VIEW MAPPING FOR FULL SCREEN
      CALL PSVMP(WKID,3,VLIM(1),VLIM(2),VLIM(3),VLIM(4),
     $           PLIM(1),PLIM(2),PLIM(3),PLIM(4))

      RETURN
      END
*****************************************************************************
*****************************************************************************
* SUBROUTINE: SAVEFL(WKID)
*
*****************************************************************************
* DESCRIPTION:
*     RENAMES THE DEFALT INTERSECTION FILE  TO BE A USER SPECIFIED
*     FILENAME.  THIS ROUTINE IS NOT DEVICE INDEPENDENT AND USES
*     THE IBM EXEC RENAME TO CHANGE THE FILENAME.
```

```
*
*     INPUT:
*        WKID -- WORKSTATION IDENTIFIER
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/21/88
*************************************************************************
*************************************************************************
      SUBROUTINE SAVEFL(WKID)
      INTEGER WKID,IER,LEN
      PARAMETER (IER = 1000)
      CHARACTER*20 STR
      CHARACTER*45 STRING

C QUERY USER FOR FILENAME
      CALL STRIN(WKID,LOSTR,STR)
      WRITE(*,*)STR
C CREATE A STRING TO GIVE TO SYSCAL FOR RENAMING THE FILE
      LEN = 27 + LOSTR
      STRING = 'EXEC RENAME FILE DEFALT A '
      STRING(27:LEN) = STR(1:LOSTR)

C SEND IT TO SYSCAL
      CALL SYSCAL(STRING,LEN,IER)

      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: INBND(PTDATA,XLOC,INTARR,ARARR,TAREA)
*
*************************************************************************
* DESCRIPTION:
*     THIS SUBROUTINE CALCULATES THE POINTS OF INTERSECTION BETWEEN
*     A PLANE OF INTERSECTION AND THE AIRCRAFT GEOMETRY.
*
*     INPUT:
*        PTDATA -- ARRAY CONTAINING AIRCRAFT DATA
*        XLOC -- LOCATION ALONG X AXIS OF INTERSECTION PLANE
*     OUTPUT:
*        INTARR -- ARRAY CONTAINING INTERSECTION DATA
*        ARARR -- ARRAY CONTAINING AREA INFO BY COMPONENT
*        TAREA -- ARRAY CONTAINING AREA INFO FOR ENTIRE AIRCRAFT
*
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 05/15/88
*************************************************************************
*************************************************************************
      SUBROUTINE INBND(PTDATA,XLOC,INTARR,ARARR,TAREA)
      REAL PTDATA(3,100,50,*),XLOC
      REAL INTARR(3,100,*),ARARR(10,400),TAREA(400)

      COMMON /INSECT/ INTCNT,INTNUM
      INTEGER INTCNT,INTNUM(800)

      COMMON /LOCN/ LOC
      REAL LOC(800)

      COMMON /INTPT/ QPT,QCNT
      REAL QPT(3,800)
      INTEGER QCNT

C USE COUNTER TO KEEP TRACK OF THE NUMBER OF INTERSECTIONS THAT HAVE
C OCCURED, INITIALIZE IT TO ZERO IN THE ROUTINE WHICH CALLS THIS
      QCNT = 0
```

```
C CREATE ARRAY TO KEEP TRACK OF THE X LOCATION OF THE INTERSECTION
      LOC(INTCNT) = XLOC

C CALL SUBROUTINE TO DETERMINE THE BOUNDING POINTS OF THE INTERSECTION
C PLANE
      CALL BOUNDS(PTDATA,XLOC,INTARR,ARARR,TAREA)

      RETURN
      END
*****************************************************************
*****************************************************************
* SUBROUTINE: BOUNDS(PTDATA,XLOC,INTARR,ARARR,TAREA)
*
*****************************************************************
* DESCRIPTION:
*     THIS SUBROUTINE SEARCHES TO DETERMINE WHICH NETWORK(S) BOUNDS
*     THE INTERSECTION PLANE AND THEN SEARCHES TO DETERMINE WHICH
*     SPLINES IN THE NETWORK ARE INVOLVED
*
*     INPUT:
*        PTDATA -- ARRAY CONTAINING AIRCRAFT DATA
*        XLOC -- LOCATION ALONG X AXIS OF INTERSECTION PLANE
*     OUTPUT:
*        INTARR -- ARRAY CONTAINING INTERSECTION DATA
*        ARARR -- ARRAY CONTAINING AREA INFO BY COMPONENT
*        TAREA -- ARRAY CONTAINING AREA INFO FOR ENTIRE AIRCRAFT
*****************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 05/15/88
*****************************************************************
*****************************************************************
      SUBROUTINE BOUNDS(PTDATA,XLOC,INTARR,ARARR,TAREA)
      REAL PTDATA(3,100,50,*),XLOC,INTARR(3,100,*)
      REAL ARARR(10,400),TAREA(400)

      COMMON /DPNTS/ NODE,XS,NET,NNODE,NXS
      INTEGER NODE,XS,NET,NNODE(20),NXS(20)

      COMMON /INSECT/ INTCNT,INTNUM
      INTEGER INTCNT,INTNUM(800)

C SET COUNTER FOR THE NUMBER OF NETWORKS INTERSECTED TO ZERO
      NETCNT = 0

C CALL ROUTINE TO SEARCH NETWORKS
      CALL NETBND(NETCNT,PTDATA,XLOC,INTARR,ARARR,TAREA)

      RETURN
      END
*****************************************************************
*****************************************************************
* SUBROUTINE: NETBND(NETCNT,PTDATA,XLOC,INTARR,ARARR,TAREA)
*
*****************************************************************
* DESCRIPTION:
*     SEARCHES THE NETWORKS TO DETERMINE WHICH ONES ARE AFFECTED
*     BY THE INTERSECTION
*
*     INPUT:
*        NETCNT -- CURRENT NETWORK NUMBER
*        PTDATA -- ARRAY CONTAINING AIRCRAFT DATA
*        XLOC -- LOCATION ALONG X AXIS OF INTERSECTION PLANE
*     OUTPUT:
*        INTARR -- ARRAY CONTAINING INTERSECTION DATA
*        ARARR -- ARRAY CONTAINING AREA INFO BY COMPONENT
*        TAREA -- ARRAY CONTAINING AREA INFO FOR ENTIRE AIRCRAFT
*
```

```
*****************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 5/16/88
*****************************************************************
*****************************************************************
      SUBROUTINE NETBND(NETCNT,PTDATA,XLOC,INTARR,ARARR,TAREA)
      REAL PTDATA(3,100,50,*),BND(2),XLOC
      REAL INTARR(3,100,*),TAREA(400),ARARR(10,400)

      COMMON /DPNTS/ NODE,XS,NET,NNODE,NXS
      INTEGER NODE,XS,NET,NNODE(20),NXS(20)

      COMMON /INSECT/ INTCNT,INTNUM
      INTEGER INTCNT,INTNUM(800)

      COMMON /BORDER/ MAX,MIN
      REAL MAX(20),MIN(20)

C INITIALIZE THE TOTAL AREA AS ZERO
      TAREA(INTCNT) = 0.

C LOOP THRU THE NETWORKS
      DO 10 I = 1,NET

      BND(1) = MIN(I)
      BND(2) = MAX(I)

C      SEE IF INT PLANE IS WITHIN THE CURRENT NETWORK
        IF ((XLOC .LE. BND(2)).AND.(XLOC .GE. BND(1))) THEN

C     IF X LOCATION IS BOUNDED THEN THE NETWORK HAS BEEN INTERSECTED
            NETCNT = NETCNT +1


C NOW LOOP THRU THE SPLINES CONTAINED IN THE INTERSECTED NETWORK
C SET THE SPLINE COUNTER TO ZERO
            SPLCNT = 0

C CALL SUBROUTINE TO LOOP THRU THE SPLINES
            NT = I

C  SEARCH SPLINES AND FINDAR BY COMPONENT AND FOR ENTIRE CRAFT
            CALL SPLBND(SPLCNT,PTDATA,NETCNT,XLOC,NT,INTARR)
            CALL FINDAR(SPLCNT,NT,INTARR,ARARR)
            CALL TOTAR(NT,ARARR,TAREA)

C  CHECK TO SEE IF THE SLOPE WAS NEGATIVE
         ELSE IF ((XLOC .GE. BND(2)).AND.(XLOC .LE. BND(1))) THEN
            NETCNT = NETCNT + 1

            SPLCNT = 0
            NT = I

C  SEARCH SPLINES AND FINDAR BY COMPONENT AND FOR ENTIRE CRAFT
            CALL SPLBND(SPLCNT,PTDATA,NETCNT,XLOC,NT,INTARR)
            CALL FINDAR(SPLCNT,NT,INTARR,ARARR)
            CALL TOTAR(NT,ARARR,TAREA)

         END IF
   10 CONTINUE

C GATHER AN ARRAY REPRESENTING THE NUMBER OF NETWORKS INTERSECTED FOR
C A PARTICULAR INTERSECTION INTCNT
      INTNUM(INTCNT) = NETCNT

      RETURN
```

```
      END
*****************************************************************
*****************************************************************
* SUBROUTINE: FINDAR(SPLCNT,NT,INTARR,ARARR)
*
*****************************************************************
* DESCRIPTION:
*     FINDS THE AREA OF A CROSS-SECTIONAL COMPONENT OF THE AIRCRAFT
*     FOR A GIVEN INTERSECTION
*     INPUT:
*        SPLCNT -- NUMBER OF INTERSECTED SPLINES
*        NT -- NETWORK NUMBER
*     OUTPUT:
*        INTARR -- ARRAY CONTAINING INTERSECTION DATA
*        ARARR -- ARRAY CONTAINING AREA INFO BY COMPONENT
*****************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 05/26/88
*****************************************************************
*****************************************************************
      SUBROUTINE FINDAR(SPLCNT,NT,INTARR,ARARR)
      INTEGER SPLCNT,NT
      REAL INTARR(3,100,*),ARARR(10,400),YZ(2,100)

      COMMON /INSECT/ INTCNT,INTNUM
      INTEGER INTCNT,INTNUM(800)


C LOOP THRU THE POINTS AND PASS THEM TO AREA ROUTINE
         DO 20 J = 1,SPLCNT
            DO 30 K = 1,2
               YZ(K,J) = INTARR(K+1,J,NT)
   30       CONTINUE
   20    CONTINUE

C CALL AREA ALGORITHM
      CALL AREA(SPLCNT,YZ,A)

C PUT AREA IN ARRAY WHICH STORES AREAS BY NETWORK AND INTERSECTION
      ARARR(NT,INTCNT) = A

      RETURN

      END
*****************************************************************
*****************************************************************
* SUBROUTINE: TOTAR(NT,ARARR,TAREA)
*
*****************************************************************
* DESCRIPTION:
*     DETERMINES THE TOTAL AREA FOR A SPECIFIC INTERSECTION
*
*     INPUT:
*        NT -- CURRENT NETWORK NUMBER
*        ARARR -- ARRAY CONTAINING AREA DATA BY COMPONENT
*     OUTPUT:
*        TAREA -- ARRAY CONTAINING TOTAL AREA
*
*****************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 06/07/88
*****************************************************************
*****************************************************************
      SUBROUTINE TOTAR(NT,ARARR,TAREA)
      INTEGER NT
      REAL ARARR(10,400),TAREA(400)
```

```
      COMMON /MAXGRF/ MAXAR
      REAL MAXAR


      COMMON /INSECT/ INTCNT,INTNUM
      INTEGER INTCNT,INTNUM(800)

      TAREA(INTCNT) = TAREA(INTCNT) + ARARR(NT,INTCNT)

C DETERMINE THE MAXIMUM GRAPH AREA
      IF(MAXAR .LT. TAREA(INTCNT)) MAXAR = TAREA(INTCNT)

      RETURN
      END
*****************************************************************
*****************************************************************
* SUBROUTINE: SPLBND(SPLCNT,PTDATA,NETCNT,XLOC,NT,INTARR)
*
*****************************************************************
* DESCRIPTION:
*     SEARCHES SPLINE BY SPLINE FOR THOSE SPLINES WHICH ARE INTER-
*     SECTED BY THE PLANE OF INTERSECTION
*
*     INPUT:
*        SPLCNT -- NUMBER OF SPLINES IN XSEC IN NETWORK THAT WERE INT.
*        NETCNT -- CURRENT NETWORK NUMBER
*        PTDATA -- ARRAY CONTAINING AIRCRAFT DATA
*        XLOC -- LOCATION ALONG X AXIS OF INTERSECTION PLANE
*        NT -- CURRENT NETWORK NUMBER
*     OUTPUT:
*        INTARR -- ARRAY CONTAINING INTERSECTION DATA
*
*****************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 05/15/88
*****************************************************************
*****************************************************************
      SUBROUTINE SPLBND(SPLCNT,PTDATA,NETCNT,XLOC,NT,INTARR)
      REAL PTDATA(3,100,50,*),SBND(2),XLOC,INTARR(3,100,*)
      INTEGER SPLCNT,NT,NS

      COMMON /DPNTS/ NODE,XS,NET,NNODE,NXS
      INTEGER NODE,XS,NET,NNODE(20),NXS(20)

      COMMON /INSECT/ INTCNT,INTNUM
      INTEGER INTCNT,INTNUM(800)

      COMMON /INTPT/ QPT,QCNT
      REAL QPT(3,800)
      INTEGER QCNT

C LOOP THRU THE SPLINES IN THE NETWORK
      DO 10 J = 1,NNODE(NT)
         NS = J

C     SET SPLINE BOUNDARIES ACCORDING TO NETWORK
         SBND(1) = PTDATA(1,J,1,NT)
         SBND(2) = PTDATA(1,J,NXS(NT),NT)

C     DETERMINE IF THE SPLINE'S SLOPE IS POS OR NEG
         SLOPE = SBND(1) - SBND(2)
         IF (SLOPE .GT. 0.) THEN

C     SEE IF SLOPE IS INTERSECTED
            IF((XLOC. LE. SBND(1)).AND.(XLOC .GE. SBND(2))) THEN
               SPLCNT = SPLCNT + 1
```

```
C       GATHER SPLINE DATA AS A FUNCTION OF THE NUMBER OF THE SPLINE
C       IN THE NETCNT NETWORK OF THE INTCNT INTERSECTION

C       CALL THE INTERSECTION ALGORITHM

              CALL INSEC(SPLCNT,XLOC,NS,NT,NETCNT,PTDATA,INTARR)
            END IF

          ELSE IF ((XLOC .GE. SBND(1)).AND.(XLOC .LE. SBND(2)))THEN
             SPLCNT = SPLCNT + 1

             CALL INSEC(SPLCNT,XLOC,NS,NT,NETCNT,PTDATA,INTARR)
            END IF

   10 CONTINUE

      QCNT = QCNT + SPLCNT

      RETURN
      END
*****************************************************************
*****************************************************************
*  SUBROUTINE: INSEC(SPLCNT,XLOC,NS,NT,NETCNT,PTDATA,INTARR)
*
*****************************************************************
* DESCRIPTION:
*     FINDS THE INTERSECTION BETWEEN A SPLINE AND A PLANE OF
*     INTERSECTION
*
*     INPUT:
*          SPLCNT -- NUMBER OF SPLINES INTERSECTED
*          XLOC -- X LOCATION OF THE INTERSECTION PLANE
*          NS -- SPLINE NUMBER IN PTDATA ARRAY
*          NT -- NETWORK NUMBER IN PTDATA ARRAY
*          NETCNT -- THE CURRENT NETWORK USED FOR INTERSECTION
*          PTDATA -- ARRAY CONTAINING THE POINTS WHICH DEFINE
*                    THE AIRCRAFT
*     OUTPUT:
*          INTARR -- OUTPUT ARRAY OF INTERSECTION POINTS
*
*****************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 05/23/88
*****************************************************************
*****************************************************************
      SUBROUTINE INSEC(SPLCNT,XLOC,NS,NT,NETCNT,PTDATA,INTARR)
      INTEGER NS,NT,NETCNT,SPLCNT,PNUM(2),NPTS
      REAL PTDATA(3,100,50,*),XLOC,PT(2,2),P(3,100)
      REAL B1(3),B2(3),INTARR(3,100,*),Q(3)

      COMMON /INTPT/ QPT,QCNT
      REAL QPT(3,800)
      INTEGER QCNT
      COMMON /DPNTS/ NODE,XS,NET,NNODE,NXS
      INTEGER NODE,XS,NET,NNODE(20),NXS(20)

      COMMON /INSECT/ INTCNT,INTNUM
      INTEGER INTCNT,INTNUM(800)

C SEND DATA TO HALF INTERVAL SEARCH ROUTINE TO DETERMINE THE TWO CLOSEST
C POINTS TO THE INTERSECTION PLANE
      NPTS = NXS(NT)
      DO 10 I = 1,NPTS
         P(1,I) = PTDATA(1,NS,I,NT)
         P(2,I) = PTDATA(2,NS,I,NT)
         P(3,I) = PTDATA(3,NS,I,NT)
   10 CONTINUE
```

```
C CALL THE ROUTINE WHICH DOES THE HALF-INTERVAL SEARCH AND PASS IT
C POINTS THRU WHICH TO SEARCH
      CALL HLFINT(NPTS,P,XLOC,PT,PNUM)

C DETERMINE THE TWO POINTS TO PASS TO THE INTERSECTION ALGORITHM
      IF(PNUM(1) .NE. PNUM(2)) THEN
         DO 20 I = 1,3
            B1(I) = PTDATA(I,NS,PNUM(1),NT)
            B2(I) = PTDATA(I,NS,PNUM(2),NT)
   20    CONTINUE

C CALL INTERSECTION ROUTINE TO PASS BACK THE POINT OF INTERSECTION
         CALL INTRTN(XLOC,B1,B2,Q)

C IF ONLY ONE POINT WAS PASSED BACK FROM THE HALF INTERVAL SEARCH
C THEN IT IS THE POINT OF INTERSECTION
      ELSE IF(PNUM(1) .NE. 0) THEN
         DO 30 I = 1,3
            Q(I) = PTDATA(I,NS,PNUM(1),NT)
   30    CONTINUE
      END IF

C NOW PUT THE POINT OF INTERSECTION IN A NEW ARRAY CALLED INTARR
      DO 40 I = 1,3
         INTARR(I,SPLCNT,NT) = Q(I)
         QPT(I,SPLCNT + QCNT) = Q(I)
   40 CONTINUE

      RETURN
      END
*******************************************************************************
*******************************************************************************
* SUBROUTINE: HLFINT(NPTS,P,PLOC,PT,PNUM)
*
*******************************************************************************
* DESCRIPTION: PERFORMS A HALF INTERVAL SEARCH TO LOCATE PLOC BETWEEN
*              TWO GIVEN POINTS
*
*              NPTS -- THE NUMBER OF POINTS IN SEARCH
*              P(2,NPTS) -- THE ARRAY OF POINTS USED IN SEARCH
*              PLOC(1) -- THE POINT TO BE LOCATED
*              PT(2,2) -- THE OUTPUT ARRAY OF POINTS WHICH BOUND PLOC
*              PNUM(2) -- ARRAY OF THE POINT NUMBERS WHICH BOUND
*******************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/25/88
*******************************************************************************
*******************************************************************************
      SUBROUTINE HLFINT(NPTS,P,PLOC,PT,PNUM)
      INTEGER NPTS,DET,PNUM(2),IFLAG
      REAL P(3,*),PLOC,PT(2,2),PP(3,100)

C SET COUNTER AND FLAG
      M = 2
      IFLAG = 0

C IF THE POINTS ARE IN DESCENDING ORDER THEN REORDER THEM INTO ASCENDING
C ORDER
      IF(P(1,NPTS) - P(1,1) .LT. 0.)THEN
         DO 21 I = 1,NPTS
            PP(1,I) = P(1,I)
   21    CONTINUE


         DO 22 I = 1,NPTS
            P(1,I) = PP(1,NPTS+1-I)
```

```
   22    CONTINUE

      IFLAG = 1

      END IF


C CALCULATE THE FIRST ARRAY POINT IN SEARCH(HALF INTERVAL POINT)
      DET = ((FLOAT(NPTS)/FLOAT(M)) + 0.5)


C INCREMENT COUNTER
  100 M = M+2

C SEE IF PLOC IS ABOVE OR BELOW HALF INTERVAL POINT
      IF (PLOC .GT. P(1,DET)) THEN

         IF ((P(1,DET) .LT. PLOC).AND.(PLOC .LT. P(1,DET+1))) THEN
            IF(IFLAG .EQ. 1)THEN
               PNUM(1) = NPTS  - DET
               PNUM(2) = NPTS + 1 - DET
            ELSE
               PNUM(1) = DET
               PNUM(2) = DET+1
            END IF
         ELSE
            CALL PGREAT(NPTS,M,DET)
            GO TO 100
         END IF
      ELSE IF (PLOC .LT. P(1,DET)) THEN
         IF ((P(1,DET-1) .LT. PLOC).AND.(PLOC .LT. P(1,DET))) THEN
            IF(IFLAG .EQ. 1)THEN
               PNUM(1) = NPTS + 1 - DET
               PNUM(2) = NPTS + 2 - DET
            ELSE
               PNUM(1) = DET-1
               PNUM(2) = DET
            END IF
         ELSE
            CALL PLESS(NPTS,M,DET)

            GO TO 100
         END IF
      ELSE IF(PLOC .EQ. P(1,DET)) THEN
            IF(IFLAG .EQ. 1)THEN
               PNUM(1) = NPTS + 1 - DET
               PNUM(2) = NPTS + 1 - DET
            ELSE
               PNUM(1) = DET
               PNUM(2) = DET
            END IF
C PUT IN A TERM FOR THE CONDITION THAT THE POINT OF LOCATION IS NOT IN
C THE INTERVAL SPECIFIED
      ELSE
         PNUM(1) = 0
         PNUM(2) = 0

      END IF
      RETURN
      END

*******************************************************************************
*******************************************************************************
* SUBROUTINE: PLESS(NPTS,M,DET)
*
*******************************************************************************
```

```
* DESCRIPTION: CALLED BY MAIN SUBROUTINE HLFINT IT SETS UP THE NEW
*              HALF INTERVAL BELOW THE PREVIOUS HALF POINT
*
*              NPTS -- NUMBER OF POINTS IN SEARCH
*              M -- COUNTER VALUE
*              DET -- ARRAY VALUE OF NEW HALF POINT
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/25/88
*************************************************************************
*************************************************************************
      SUBROUTINE PLESS(NPTS,M,DET)
      INTEGER NPTS,M,DET,INCR

C CALCULATE NEW HALF INTERVAL WIDTH
      INCR = ((FLOAT(NPTS)/FLOAT(M))+0.5)

C FIND NEW ARRAY VALUE OF HALF POINT
      DET = DET -INCR
      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: PGREAT(NPTS,M,DET)
*
*************************************************************************
* DESCRIPTION: CALLED BY MAIN SUBROUTINE HLFINT IT SETS UP THE NEW
*              HALF INTERVAL ABOVE THE PREVIOUS HALF POINT
*
*              NPTS -- NUMBER OF POINTS IN SEARCH
*              M -- COUNTER VALUE
*              DET -- ARRAY VALUE OF NEW HALF POINT
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 4/25/88
*************************************************************************
*************************************************************************
      SUBROUTINE PGREAT(NPTS,M,DET)
      INTEGER NPTS,M,DET,INCR

C CALCULATE NEW HALF INTERVAL WIDTH
      INCR = ((FLOAT(NPTS)/FLOAT(M))+0.5)

C FIND NEW ARRAY VALUE OF HALF POINT
      DET = DET + INCR
      RETURN
      END
*************************************************************************
*************************************************************************
* SUBROUTINE: INTRTN(XLOC,B1,B2,INTARR)
*
*************************************************************************
* DESCRIPTION:
*     CALCULATES THE POINT OF INTERSECTION BETWEEN A LINE AND A
*     PLANE OF INTERSECTION
*
*     INPUT:
*        XLOC -- X LOCATION OF THE INTERSECTION PLANE
*        B1 AND B2 -- ENDPNTS OF THE LINE SEGMENT WHICH WILL BE INTSCTD
*     OUTPUT:
*        INTARR -- ARRAY CONTAINING INTERSECTION DATA
*************************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 05/25/88
*************************************************************************
*************************************************************************
      SUBROUTINE INTRTN(XLOC,B1,B2,Q)
```

```
      REAL B1(3),B2(3),Q(3),PAR1(3),NOOM1,NOOM2
      REAL A(3),B(3),C(3),E(3),XLOC

      COMMON /DPNTS/ NODE,XS,NET,NNODE,NXS
      INTEGER NODE,XS,NET,NNODE(20),NXS(20)

      COMMON /INSECT/ INTCNT,INTNUM
      INTEGER INTCNT,INTNUM(800)


C DEFINE THE INTERSECTION PLANE IN NEW TERMS
      A(1) = XLOC
      A(2) = 250.
      A(3) = -200.

      B(1) = 0.
      B(2) = -550.
      B(3) = 0.

      C(1) = 0.
      C(2) = 0.
      C(3) = 455.

C DEFINE E AS THE DIFFERENCE BETWEEN B1 AND B2
      DO 10 I = 1,3
         E(I) = B2(I) - B1(I)
   10 CONTINUE

C MAKE E,B AND C INTO UNIT VECTORS
      CALL VMAG(B,BMAG)
      CALL VMAG(C,CMAG)
      DO 12 I = 1,3
         B(I) = B(I)/BMAG
         C(I) = C(I)/CMAG
   12 CONTINUE

C FIND THE PARAMETRIC VARIABLE FOR THE EQUATION OF THE LINE
C WHICH DEFINES THE POINT OF INTERSECTION
      CALL VPROD(B,C,PAR1)
      CALL SPROD(PAR1,A,NOOM1)
      CALL SPROD(PAR1,B1,NOOM2)
      CALL SPROD(PAR1,E,DENOM)

      T = (NOOM1 - NOOM2)/DENOM
C USE THE LINE EQUATION TO FIND THE POINT OF INTERSECTION
      DO 20 I = 1,3
         Q(I) = B1(I) + T*E(I)
   20 CONTINUE
      RETURN
      END
*****************************************************************
*****************************************************************
* SUBROUTINE: VPROD(V1,V2,V3)
*
*****************************************************************
*     THIS SUBROUTINE CALCULATES THE CROSS PRODUCT OF TWO THREE
*     COMPONENT VECTORS V1 AND V2
*     INPUT:
*        V1 -- VECTOR 1
*        V2 -- VECTOR 2
*     OUTPUT:
*        V3 -- VECTOR REPRESENTING THE CROSS PROD OF V1 AND V2
*****************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 03/15/88
*****************************************************************
*****************************************************************
```

```
      SUBROUTINE VPROD(V1,V2,V3)

      REAL V1(3),V2(3),V3(3)

C CALCULATE THE COMPONENTS OF VECTOR THREE BY DEFINITION OF
C CROSS PRODUCT
      V3(1)=V1(2)*V2(3)-V2(2)*V1(3)
      V3(2)=V2(1)*V1(3) - V1(1) * V2(3)
      V3(3)=V1(1)*V2(2)-V2(1)*V1(2)

      RETURN
      END
*****************************************************************
*****************************************************************
* SUBROUTINE: SPROD(V1,V2,S)
*
*****************************************************************
*     THIS SUBROUTINE CALCULATES THE DOT PRODUCTS OF VECTORS V1 AND
*     V2 AND RETURNS THE SCALAR S
*     INPUT:
*        V1 -- VECTOR 1
*        V2 -- VECTOR 2
*     OUTPUT:
*        S -- VECTOR REPRESENTING THE DOT PROD OF V1 AND V2
*****************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 03/15/88
*****************************************************************

      SUBROUTINE SPROD(V1,V2,S)
      REAL V1(3),V2(3),S

      S = 0.
      DO 10 I = 1,3

         S = V1(I)*V2(I) + S

   10 CONTINUE
      RETURN
      END
*****************************************************************
*****************************************************************
* SUBROUTINE: VMAG(V, MAG)
*
*****************************************************************
*   THIS SUBROUTINE CALCULATES THE MAGNITUDE OF VECTOR V
*     INPUT:
*        V -- THE VECTOR
*     OUTPUT:
*        MAG -- ITS MAGNITUDE
*****************************************************************
* CODED BY: MICHELE GRIESHABER
* DATE    : 03/15/88
*****************************************************************
*****************************************************************
      SUBROUTINE VMAG(V,MAG)
      REAL V(3),MAG
*     V(3) IS THE INPUT VECTOR
*     MAG IS THE MAGNITUDE OF VECTOR V

C CALCULATE THE MAGNITUDE OF VECTOR V
      MAG = (V(1)**2 + V(2)**2 + V(3)**2)**0.5

      RETURN
      END
```

APPENDIX C. ALGORITHM FOR CALCULATION OF AREA

The following is a listing of the code for the AREA algorithm written by J. Sankar.

```
************************************************************
*  SUBROUTINE FOR CALCULATION OF AREA
*
*  WRITTEN BY JAYARAM SANKAR
*
************************************************************

      SUBROUTINE AREA (NP,XY,A)

************************************************************
*  SUBROUTINE FOR CALCULATING THE AREA OF A PLANE PLOYGON
*  NP = NO. OF POINTS
*  XY(2,NP) = ARRAY CONTAINING X,Y COORDINATES
*  A = AREA
*
************************************************************



      REAL A,XY(2,*),PRVVEC(2),CURVEC(2),PRVANG,CURANG,DA
      REAL PI,XMIN,YMIN
      INTEGER NP, ICOUNT

      A = 0.
      PI = 4.0*ATAN(1.0)
C     WRITE(6,*) PI
      XY(1,NP +1) = XY(1,1)
      XY(2,NP +1) = XY(2,1)

      XMIN = XY(1,1)
      YMIN = XY(2,1)

      DO 5 I = 2,NP
         IF(XY(1,I).LT.XMIN) THEN
            XMIN = XY(1,I)
            YMIN = XY(2,I)
         ENDIF
5     CONTINUE

      DO 7 I = 1,NP+1
         XY(1,I) = XY(1,I) - XMIN
```

```
         XY(2,I) = XY(2,I) - YMIN
7     CONTINUE


      PRVVEC(1) = XY(1,1)
      PRVVEC(2) = XY(2,1)

      IF (XY(1,1).EQ.0.) THEN
         IF(XY(2,1).GT.0.) PRVANG = PI/2.0
         IF(XY(2,1).LT.0.) PRVANG = -PI/2.0
      ELSE
         PRVANG = ATAN2(XY(2,1),XY(1,1))
      ENDIF


      DO 10 ICOUNT = 2,NP + 1

         CURVEC(1) = XY(1,ICOUNT)
         CURVEC(2) = XY(2,ICOUNT)

         IF (CURVEC(1).EQ.0.) THEN
            IF(CURVEC(2).GT.0.) CURANG = PI/2.0
            IF(CURVEC(2).LT.0.) CURANG = -PI/2.0
         ELSE
            CURANG = ATAN2(CURVEC(2),CURVEC(1))
         ENDIF

         DA = ABS(0.5*(PRVVEC(1)*CURVEC(2) - PRVVEC(2)*CURVEC(1)))

         IF (CURANG.GT.PRVANG) A = A + DA
         IF (CURANG.LT.PRVANG) A = A - DA

         PRVANG = CURANG
         PRVVEC(1) = CURVEC(1)
         PRVVEC(2) = CURVEC(2)

10    CONTINUE

      A = ABS(A)

      RETURN
      END
```

APPENDIX D. ACSYNT/VPI USER'S GUIDE FOR INTERSECTION

D.1 User-Assisted Calculation of Cross-Sectional Area

Purpose:
to allow the user to specify a cross-sectional region on which to perform an area calculation.

Menu Path:
MAIN(GEOMETRY (ANALYSIS (XSEC AREA (MANUAL INSEC))))

Description:
With the selection of the "MANUAL INSEC" menu item, the following messages will appear:

```
select menu item:
select menu item:
enter values for x translation and y rotation of intersection plane:
enter value(s) using dials and/or keyboard:
OR CHOOSE A MENU ITEM:
_    0.00,  0.00
```

To perform an intersection, enter a value for the translation along the x axis which is greater than zero. Y rotations are valid between -90 degrees and 90 degrees.

Upon entering the "MANUAL ISEC" menu, a default window will appear in the upper right-hand corner of the screen. This default window contains a front view of the current aircraft when in "MANUAL ISEC", but on selection of the "INTERSECT" menu item, the intersection points for the current intersection plane location and orientation will appear in place of the geometry. Location of the default window may be changed by selecting the "X-SEC VIEW" menu item. This menu item allows the user to choose a new location for the default window. It also presents the user with the option of zooming the window to full-screen or allows the user to discard the view entirely. If the default view

has been discarded and the "INTERSECT" option is chosen, a new default view will appear which contains the points of intersection. This view will be full-screen.

When the "INTERSECT" menu item is chosen and the default window is filled with intersection points, the proximity and density of the points with respect to one another is such that often it may be necessary to zoom closer to chain accurately. The zoom option may be invoked in any of the intersection-related menus by changing the view in which the zoom is desired to the current view and using the geometry valuators to zoom in on the points of interest. The current view definition will change with the selection of either the view marker located in the lower left corner of the view, or by choosing the edge of a view, taking care that no geometry is chosen in the process.

Selecting the "RETURN" menu item on the "MANUAL ISEC" menu will drop the intersection plane from the view and re-enter the "ANALYSIS" menu. More information concerning intersection is provided in the following sections:

- "Manual Cross-Sectional Intersection"
- "Manipulation of the Default View" on page 117
- "Show and No-Show of the Intersection Plane" on page 118
- "Save Intersection Data File (Preliminary)" on page 119

D.1.1 Manual Cross-Sectional Intersection

Purpose:

to allow manual area determination of a specified cross-section.

Menu Path:

MAIN(GEOMETRY (ANALYSIS (XSEC AREA (MANUAL INSEC (INTERSECT)))))

Description:

With the selection of the "INTERSECT" menu item, the intersection plane location and orientation are assumed to be set in the desired positions. "INTERSECT" calculates the points of intersection between the intersection plane and the aircraft geometry. The resulting intersection points are then displayed in the default window, whose location may have been changed using the "XSEC VIEW" command in the "MANUAL ISEC" menu. If the view had been discarded, a new full screen default view will appear which contains the points.

```
select menu item:
enter values for (1) x translation and (2) y rotation of intersec plane:
enter value(s) using dials and/or keyboard:
or choose a menu item:
PLEASE BEGIN CHAINING OR SELECT MENU ITEM:
_
```

The chaining process is necessary to determine which areas the user wishes to compute. Upon entering the "CHAIN MODE" menu, the user selects the first point of the component or outline that he desires. He then continues to select one point at a time until finished. As the user works his way around the outline of the aircraft, line segments are drawn between consectutive points. If at any time the user would like to back up his se-

lection he merely retraces his steps back to the point at which he would like to restart by selecting line segments in order of those most previously drawn. Once he reaches the point from which he would like to begin anew, the user continues forward by choosing a new point on the aircraft cross-section.

Calculation of the area occurs when the user selects the first point he started with as the last point in the chain. Equivalently he could select the first line segment drawn to close the chain. Immediately after closing the chain, the area of the resulting section appears in the message area in terms of square feet.

The chaining mode may be used in two ways. First, if it is desirable to determine the cross-sectional area for the profile of the aircraft, all the user need do is select points around the perimeter. In the case where there are components that are entirely independent of the main body of the aircraft (eg. the case where a pod is intersected at a point where it is not attatched to the body), a solution is to chain along the main portion of the aircraft until a point is reached where one can bridge to the independent component. The chaining routine will draw a line segment from the main body to the independent body. Continue to chain around the independent component until complete. The next step is to close the independent body's profile by selecting its first point. This will have the effect of closing the independent body's area. One then reselects the bridge segment to return to the main body and resumes chaining there as if having never left the main body.

In the second case, one desires to calculate the area of individual components. All one need do is to completely chain around one component, await calculation of its area, select the "RESTART CHAIN" menu item and begin chaining around another component. This process may be repeated as required.

Example:

1. While in the "MANUAL ISEC" menu, position the intersection plane as desired. Numerical input position 1 is used for x translation of the plane and position 2 is used for rotation of the plane about the y axis.
2. Select the "INTERSECT" menu item.
3. Choose the view which contains the intersection point data in order to make it the current view.
4. Use the valuators to zoom in on the region of interest (valuator 7 is used for the zoom option)
5. Begin chaining points to enclose the desired area.
6. When the last point chosen and the first point chosen are concurrent, the area of the section appears in the message area in terms of square feet.

D.1.2 Manipulation of the Default View

Purpose:

to allow the user to reposition or discard the default view which contains a front view of the aircraft before intersection and the intersection data points after intersection.

Menu Path:

MAIN(GEOMETRY (ANALYSIS (XSEC AREA (MANUAL INSEC (XSEC VIEW)))))

Description:

With the selection of the "XSEC VIEW" menu item, the user can relocate or discard the default view used to contain a front view of the aircraft before intersection and the cross-section points after intersection.

```
select menu item:
enter values for (1) x translation and (2) y rotation of intersec plane:
enter value(s) using dials and/or keyboard:
or choose a menu item:
CHOOSE A NEW XSEC WINDOW OPTION OR SELECT RETURN:
_
```

There are several possibilities available to relocate the quarter-screen default window: "LOWER LEFT", "LOWER RIGHT", "UPPER RIGHT", and "UPPER LEFT". Two other menu item, "FULL SCREEN", and "DISCARD" enable the user to increase the size of the view to fill the screen or to delete the view entirely.

Example:

1. once inside the "MANUAL ISEC" menu, select the "XSEC VIEW" menu item.
2. select the "FULL SCREEN" menu item
3. select the "UPPER LEFT" menu item
4. select the "DISCARD" menu item
5. exit the "XSEC VIEW" menu

D.1.3 Show and No-Show of the Intersection Plane

Purpose:

to allow the user to alternately render the intersection plane invisible and visible.

Menu Path:

MAIN(GEOMETRY (ANALYSIS (XSEC AREA (MANUAL INSEC (SHO/NO SHO PLANE)))))

Description:

Upon entering the "MANUAL ISEC" menu, the intersection plane is set as visible. The "SHO/NO SHO PLANE" option allows the user to remove the intersection plane from display. If the plane of intersection is not visible to the user, selecting the "SHO/NO SHO PLANE" menu item causes the intersection plane to visible.

Example:

1. enter the "MANUAL ISEC" menu (note that the intersection plane is visible).

2. select the "SHO/NO SHO PLANE" menu item. The plane will dissappear.
3. select the "SHO/NO SHO PLANE" option once again. Notice the plane will reappear.

D.1.4 Save Intersection Data File (Preliminary)

Purpose:
to allow the user to save intersection related data, such as intersection points by cross-section and their corresponding areas.

Menu Path:
MAIN(GEOMETRY (ANALYSIS (XSEC AREA (MANUAL INSEC (SAVE FILE)))))

Description:
Upon entering the "MANUAL ISEC" menu, the user may perform a series of intersections which generate data in the form of cross-sectional areas and their corresponding points. These intersections are described in detail in the section entitled "Manual Cross-Sectional Intersection" on page 116. These data are written to a default file which is renamed upon selection of the "SAVE FILE" option.

```
or choose a menu item:
please begin chaining or select menu item:
chaining operation complete, area is XXX.XXX sq ft.:
select menu item:
enter filename and filetype of file:
_
```

If no save file command has been issued upon leaving the "MANUAL ISEC" menu, the default file will be deleted in order to conserve disk space.

Example:

1. enter the "MANUAL ISEC" menu.
2. position intersection plane at 55.00 ft along the x-axis with 0 degrees of rotation.
3. select the "INTERSECT" menu item.
4. chain closed a small region of area in the "CHAIN MODE" menu.
5. exit the "CHAIN MODE" menu and select "SAVE FILE".
6. respond to prompts by typing <JUNK, your initials>. **Make sure the filename and filetype entered are in capital letters.**
7. after exiting ACSYNT/VPI check the file listing for the file and remark upon the contents.

APPENDIX E. ACSYNT/VPI INTERSECTION CODE

The following is a listing of the code for the intersection portion of ACSYNT/VPI.

```
C=====================================================================
C                        M O D U L E   A M A N U
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL AMANU(WKID)
C=====================================================================
C INPUT PARAMETERS:
C     WKID - INTEGER, WORKSTATION IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     NVALS  - INTEGER, NUMBER OF VALUATORS
C     PREC   - INTEGER, ARRAY OF PRECISION VALUES FOR VALUATORS
C     IVAL   - REAL, ARRAY CONTAINING VALUES OF THE VALUATORS
C     MINVAL - REAL, ARRAY CONTAINING THE MINIMUM VALUATOR VALUES
C     MAXVAL - REAL, ARRAY CONTAINING THE MAXIMUM VALUATOR VALUES
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     AIRCRAFT MANUAL INTERSECTION MODULE
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
```

```
C     OPNUM - OPENS NUMBER INPUT
C     MMANU  - MANUAL INTERSECTION MENU DRIVER
C     CLNUM - CLOSE NUMBER INPUT
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================

      SUBROUTINE AMANU(WKID)
      INTEGER WKID,NVALS
      PARAMETER(NVALS = 2)
      INTEGER PREC(NVALS)
      REAL IVAL(NVALS), MAXVAL(NVALS), MINVAL(NVALS)

      DATA PREC/4,4/
      DATA IVAL/0.,0./
      DATA MINVAL/-10.,0./
      DATA MAXVAL/150.,90./

C     OPEN NUMBER INPUT
      CALL OPNUM(WKID)

C     GEOMETRY MENU
      CALL MMANU(WKID,NVALS, PREC, IVAL, MINVAL, MAXVAL)

C     CLOSE NUMBER INPUT
      CALL CLNUM(WKID)

      RETURN
      END

C=====================================================================
C                      M O D U L E  M M A N U
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU.2
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL MMANU(WKID,NVALS, PREC, IVAL, MINVAL, MAXVAL)
C=====================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C     NVALS  - INTEGER, NUMBER OF VALUATORS
C     PREC   - INTEGER, ARRAY OF PRECISION VALUES FOR VALUATORS
C     IVAL   - REAL, ARRAY CONTAINING VALUES OF THE VALUATORS
C     MINVAL - REAL, ARRAY CONTAINING THE MINIMUM VALUATOR VALUES
C     MAXVAL - REAL, ARRAY CONTAINING THE MAXIMUM VALUATOR VALUES
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
```

```
C     TITLE  - CHARACTER, MENU TITLE
C     NITEMS - INTEGER, NUMBER OF ITEMS IN THE MENU
C     ITEMS(NITEMS) - CHARACTER, MENU ITEMS
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     MANUAL INTERSEC MENU DRIVER
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     NEWMNU - DISPLAYS A NEW MENU
C     XSVWDF - SETS UP DEFAULT GEOMETRY AND INTERSECTION VIEW
C     SPLANE - CREATES THE DEFINITION OF THE INTERSECTION PLANE
C     EXMANU - EXECUTES GEOMETRY MENU
C     EPLANE - DELETES THE INTERSECTION PLANE STRUCTURE UPON LEAVING
C     XSVRST - RESETS THE VIEWS USED FOR GEOMETRY AND INTERSECTION
C     OLDMNU - DISPLAYS PREVIOUS MENU
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:45:24)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================

      SUBROUTINE MMANU(WKID, NVALS, PREC, IVAL, MINVAL, MAXVAL)
      REAL IVAL(*), MINVAL(*), MAXVAL(*)
      INTEGER WKID, NVALS, PREC(*)

      CHARACTER*12 TITLE
      PARAMETER(TITLE = 'MANUAL ISEC ')

      INTEGER NITEMS
      PARAMETER(NITEMS = 5)
      CHARACTER*16 ITEMS(NITEMS)


C     DEFINE MENU ITEMS
      DATA ITEMS / '    INTERSECT   ',
     >             '   X-SEC VIEW   ',
     >             'SHO/NO SHO PLANE',
     >             '    SAVE FILE   ',
     >             '     RETURN     ' /

C     DRAW NEW MENU
      CALL NEWMNU(WKID, TITLE, NITEMS, ITEMS)

C     SET UP DEFAULTS FOR THE XSEC WINDOW
      CALL XSVWDF(WKID)

C     DRAW THE INTERSECTION PLANE STRUCTURE
      CALL SPLANE(WKID,IVAL)

C     EXECUTE MANUAL INTERSECTION MENU
      CALL EXMANU(WKID, NVALS, PREC, IVAL, MINVAL, MAXVAL)

C     ERASE THE INTERSECTION PLANE STRUCTURE
      CALL EPLANE(WKID)

*     RESET THE VIEWS USED FOR THE CROSS SECTION
      CALL XSVRST (WKID)
```

```
C     DISPLAY OLD MENU
      CALL OLDMNU(WKID)

      RETURN
      END

C=====================================================================
C                      M O D U L E   S P L A N E
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL SPLANE(WKID,IVAL)
C=====================================================================
C INPUT PARAMETERS:
C     WKID - INTEGER, WORKSTATION IDENTIFIER
C     IVAL - ARRAY CONTAINING CURRENT VALUATOR VALUES
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     ITEM  - INTEGER, NUMBER OF STRUCTURE TO USE
C     STID  - INTEGER, STRUCTURE ID TO PUT PLANE INTO
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     DRAWS THE INTERSECTION PLANE
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     GTNVWS - GETS NUMBER OF ACTIVE VIEWS
C     GTIVWS  - GETS STRUCTURE IDS OF ELEMENTS IN VIEWS
C     POPST - PHIGS OPEN STRUCTURE
C     DPLANE- DRAWS THE INTERSECTION PLANE
C     PCLST- PHIGS CLOSE STRUCTURE
C     PSIVFT- PHIGS INVISIBILITY FILTER
C     UPPLVW- UPDATES VIEWS WITH PLANE
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/18/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE SPLANE(WKID,IVAL)
      INTEGER WKID,MGEOM,NW,ERRIND,ITEM,NITEMS,STID
      REAL PRIOR,IVAL(*)
```

```
      PARAMETER (MPLAN = 3,PRIOR = 1., ITEM = 4, NITEMS = 1,ITEM2 = 2)
      INTEGER VIEW, VIEW2
      PARAMETER (VIEW = 11, VIEW2 = 12)

C     GET NUMBER OF VIEWS
      CALL GTNVWS(NW,ERRIND)

C     LOOP THRU NUMBER OF VIEWS
      DO 10 I = 2, NW+1

C     GET THE STRUCTURE IDS
         CALL GTIVWS(ITEM,NITEMS,I,STID,ERRIND)

C     OPEN EACH STRUCTURE
         CALL POPST(STID)

C     DRAW THE INTERSECTION PLANE
         CALL DPLANE(WKID,I,IVAL)

C     CLOSE THE STRUCTURE
         CALL PCLST

C    SET THE PLANE TO BE INITIALLY INVISIBLE
C        CALL PSIVFT(WKID,1,254,0,0)
   10 CONTINUE

C     UPDATE XSEC VIEW
      CALL UPPLVW(WKID,VIEW,VIEW2,IVAL(1),IVAL(2))

      RETURN
      END
C=====================================================================
C                        M O D U L E  D P L A N E
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL SPLANE(WKID,VWID,IVAL)
C=====================================================================
C INPUT PARAMETERS:
C     WKID - INTEGER, WORKSTATION IDENTIFIER
C     VWID - INTEGER, VIEW ID OF ACTIVE VIEW
C     IVAL - ARRAY CONTAINING CURRENT VALUATOR VALUES
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     A,B,C,D - REAL, BOUNDARIES OF THE INTERSECTION PLANE
C     PLNAME - INTEGER, SET NAME OF PLANE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     DRAWS THE PLANE OF INTERSECTION
C=====================================================================
C EXAMPLE(S):
C     N/A
```

```
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     PADS - PHIGS ADD NAMES TO SET
C     PLTRAN  - TRANSFORMS PLANE
C     PLDRU - DRAW MESH OF PLANE IN U DIRECTION
C     PLDRW - DRAW MESH OF PLANE IN W DIRECTION
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/18/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE DPLANE(WKID,VWID,IVAL)
      INTEGER WKID,VWID,PLNAME
      REAL A,B,C,D,IVAL(*)
      PARAMETER (A = -50., B = -50., C = 50., D = 50., PLNAME = 254)


C SET CLASS NAMES FOR ATTRIBUTES OF PLANE STRUCTURE
      CALL PADS(1,PLNAME)

C SET THE GLOBAL TRANSFORMATION MATRIX FOR THE PLANE
      CALL PLTRAN(IVAL)

C DRAW MESH FOR PLANE IN THE U DIRECTION
      CALL PLDRU(A,B,C,D)

C DRAW MESH FOR PLANE IN THE W DIRECTION
      CALL PLDRW(A,B,C,D)

      RETURN
      END
C=====================================================================
C                        M O D U L E  P L T R A N
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL PLTRAN(IVAL)
C=====================================================================
C INPUT PARAMETERS:
C     IVAL - ARRAY CONTAINING CURRENT VALUATOR VALUES
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     XFORM - TRANSFORMATION ARRAY FOR TRANSLATION
C     YXFORM - TRANSFORMATION ARRAY FOR Y ROTATION
C     FFORM - FINAL TRANSFORMATION ARRAY
```

```
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     TRANSFORMS THE INTERSECTION PLANE
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     PLB - PHIGS LABEL
C     PTR3  - PHIGS X TRANSLATION
C     PROY - PHIGS Y ROTATION
C     PCOM3 - PHIGS CONCATENATE MATRICES
C     PSLMT3- PHIGS SET LOCAL TRANSFORMATION MATRIX
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/18/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE PLTRAN(IVAL)
      INTEGER BFTRAN,AFTRAN
      REAL XFORM(4,4),IVAL(*),YXFORM(4,4),FFORM(4,4)
      PARAMETER (BFTRAN = 1, AFTRAN = 2)

C SET FIRST LABEL BEFORE XFORM
      CALL PLB(BFTRAN)

C CALCULATE XFORM MATRIX FOR X TRANSLATION
      CALL PTR3(IVAL(1),0.,0.,XFORM)

C CALCULATE THE XFORM MATRIX FOR Y ROTATION
      CALL PROY(IVAL(2),YXFORM)

C PUT THEM TOGETHER
      CALL PCOM3(YXFORM,XFORM,FFORM)

C SET LOCAL XFORM MATRIX
      CALL PSLMT3(FFORM,2)

C SET SECOND LABEL
      CALL PLB(AFTRAN)

      RETURN
      END
C=====================================================================
C                       M O D U L E   P L D R U
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL PLDRU(A,B,C,D)
C=====================================================================
C INPUT PARAMETERS:
C     A,B,C,D - REAL, VALUES WHICH DEFINE THE BOUNDARIES OF THE PLANE
C=====================================================================
C OUTPUT PARAMETERS:
```

```
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     NUMINC - INTEGER, NUMBER OF LINES WHICH DEFINE PLANE MESH
C     PXA,PYA,PZA - X,Y,Z-COORDS OF THE LINE POINTS
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     DRAWS THE PLANE MESH IN THE DIRECTION OF THE U PARAMETER
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     PSPCLI - PHIGS SET COLOR INDEX
C     PPL3   - PHIGS 3D LINE
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/18/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE PLDRU(A,B,C,D)
      REAL A,B,C,D
      INTEGER NUMINC
      PARAMETER(NUMINC = 20)
      REAL PXA(NUMINC),PYA(NUMINC),PZA(NUMINC)

C MAKE A WHITE MESH
      CALL PSPLCI(1)

C DRAW MESH LINES EVERY NUMINC/100 FEET
      DO 10 I = 1,NUMINC
         U = FLOAT(I)/FLOAT(NUMINC)

            DO 20 J = 1,NUMINC
               W = FLOAT(J)/FLOAT(NUMINC)

               PXA(J) = 0.
               PYA(J) = (C-A) * U + A
               PZA(J) = (D-B) * W + B
   20       CONTINUE

C DRAW 3D POLYLINES TO REPRESENT THE MESH
      CALL PPL3(NUMINC,PXA,PYA,PZA)

   10 CONTINUE

      RETURN
      END
C=====================================================================
C                        M O D U L E  P L D R W
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
```

```
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL PLDRW(A,B,C,D)
C=====================================================================
C INPUT PARAMETERS:
C     A,B,C,D - REAL, VALUES WHICH DEFINE THE BOUNDARIES OF THE PLANE
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     NUMINC - INTEGER, NUMBER OF LINES WHICH DEFINE PLANE MESH
C     PXA,PYA,PZA - X,Y,Z-COORDS OF THE LINE POINTS
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     DRAWS THE PLANE MESH IN THE DIRECTION OF THE W PARAMETER
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     PSPCLI - PHIGS SET COLOR INDEX
C     PPL3   - PHIGS 3D LINE
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/18/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE PLDRW(A,B,C,D)
      REAL A,B,C,D
      INTEGER NUMINC
      PARAMETER(NUMINC = 20)
      REAL PXA(NUMINC),PYA(NUMINC),PZA(NUMINC)

C DRAW MESH LINES EVERY NUMINC/100 FEET
      DO 10 I = 1,NUMINC
         W = FLOAT(I)/FLOAT(NUMINC)

            DO 20 J = 1,NUMINC
               U = FLOAT(J)/FLOAT(NUMINC)

               PXA(J) = 0.
               PYA(J) = (C-A) * U + A
               PZA(J) = (D-B) * W + B
   20       CONTINUE

C DRAW 3D POLYLINES TO REPRESENT THE MESH
      CALL PPL3(NUMINC,PXA,PYA,PZA)

   10 CONTINUE
```

```
      RETURN
      END
C=======================================================================
C                      M O D U L E   E P L A N E
C=======================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=======================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C     CALL EPLANE(WKID)
C=======================================================================
C INPUT PARAMETERS:
C     WKID - INTEGER, WORKSTATION IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C LOCAL VARIABLES:
C     NVIEWS - INTEGER, NUMBER OF VIEWS FROM WHICH TO REMOVE PLANE
C     STID - INTEGER, STRUCTURE ID OF THE PLANE IN ALL ITS VIEWS
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     REMOVES THE PLANE FROM THE VIEWS ON SCREEN
C=======================================================================
C EXAMPLE(S):
C     N/A
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C MODULES CALLED:
C     GTNVWS - GETS NUMBER OF VIEWS THE PLANE CURRENTLY EXISTS IN
C     GTIVWS - GETS STRUCTURE IDS OF THE PLANE IN ALL VIEWS
C     PEMST - PHIGS EMPTY STRUCTURE
C=======================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/18/88 (15:26:51)
C=======================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=======================================================================
      SUBROUTINE EPLANE(WKID)
      INTEGER WKID,NVIEWS,ERRIND,ITEM,NITEMS,STID
      PARAMETER(ITEM = 4, NITEMS = 1)

C     GET NUMBER OF VIEWS
      CALL GTNVWS(NVIEWS,ERRIND)

C     LOOP THRU VIEWS
      DO 10 I = 2,NVIEWS+1

C     GET STRUCTURE ID
         CALL GTIVWS(ITEM, NITEMS, I, STID, ERRIND)

C     EMPTY THE STRUCTURE
```

```
         CALL PEMST(STID)

   10 CONTINUE

      RETURN
      END
C=====================================================================
C                      M O D U L E   E X M A N U
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU.2.4
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL EXMANU(WKID, NVALS, PREC, IVAL, MINVAL, MAXVAL)
C=====================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C     NVALS  - INTEGER, NUMBER OF VALUATORS
C     PREC   - INTEGER, ARRAY OF PRECISION VALUES FOR VALUATORS
C     IVAL   - REAL, ARRAY CONTAINING VALUES OF THE VALUATORS
C     MINVAL - REAL, ARRAY CONTAINING THE MINIMUM VALUATOR VALUES
C     MAXVAL - REAL, ARRAY CONTAINING THE MAXIMUM VALUATOR VALUES
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     PREV - LOGICAL, PREVIOUS MENU FLAG
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     EXECUTES MANUAL INTERSECTION MENU
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     MSG    - WRITES AND SCREEN MESSAGE
C     INNMIN - INITIALIZES MENU INPUT
C     INMANU - GETS AND PROCESSES GEOMETRY MENU INPUT
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (19:18:18)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================

      SUBROUTINE EXMANU(WKID, NVALS, PREC, IVAL, MINVAL, MAXVAL)
      REAL IVAL(*), MINVAL(*), MAXVAL(*)
      INTEGER WKID, NVALS, PREC(*)

      LOGICAL PREV
```

```
C     ASSUME PREVIOUS MENU FLAG IS FALSE
      PREV = .FALSE.

C     WRITE SCREEN MESSAGE
      CALL MSG(WKID, 'SELECT MENU ITEM:')

C     INITIALIZE MENU INPUT
  100 CALL INMMIN(WKID, NVALS, PREC, IVAL)

C     GET AND PROCESS MANUAL INTERSECTION MENU INPUT
      CALL INMANU(WKID, PREV, NVALS, IVAL, MINVAL, MAXVAL)

      IF (.NOT. PREV) THEN
        GOTO 100
      ENDIF

      RETURN
      END

C=======================================================================
C                      M O D U L E   I N M A N U
C=======================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU.2.4.4
C=======================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C     CALL INMANU(WKID, PREV, NVALS, IVAL, MINVAL, MAXVAL)
C=======================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C     NVALS  - INTEGER, NUMBER OF VALUATORS
C     PREC   - INTEGER, ARRAY OF PRECISION VALUES FOR VALUATORS
C     IVAL   - REAL, ARRAY CONTAINING VALUES OF THE VALUATORS
C     MINVAL - REAL, ARRAY CONTAINING THE MINIMUM VALUATOR VALUES
C     MAXVAL - REAL, ARRAY CONTAINING THE MAXIMUM VALUATOR VALUES
C=======================================================================
C OUTPUT PARAMETERS:
C     PREV - LOGICAL, PREVIOUS MENU FLAG
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C LOCAL VARIABLES:
C     ICL  - INTEGER, INPUT DEVICE CLASS
C     IDNR - INTEGER, INPUT DEVICE NUMBER
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     GETS AND PROCESSES MANUAL INTERSECTION MENU INPUT
C=======================================================================
C EXAMPLE(S):
C     N/A
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C MODULES CALLED:
C     GTMNIN - GET MENU INPUT
C     PMANU  - PROCESS AUTO INSEC MENU INPUT
C     PIPLVL - PROCESS INTERSECTION PLANE VALUATOR INPUT
```

```
C     PIPLST - PROCESS INTERSECTION PLANE STRING INPUT
C     MSG    - WRITES A SCREEN MESSAGE
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (13:26:41)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================

      SUBROUTINE INMANU(WKID, PREV, NVALS, IVAL, MINVAL, MAXVAL)
      REAL IVAL(*), MINVAL(*), MAXVAL(*)
      INTEGER WKID,NVALS
      LOGICAL PREV

      INTEGER ICL, IDNR, PPICK, PVALUA , PSTRIN
      PARAMETER(PVALUA = 3, PPICK = 5,PSTRIN = 6)

C     GET MENU INPUT
      CALL GTMNIN(WKID, ICL, IDNR)

      IF (ICL .EQ. PPICK) THEN
C       PROCESS MANUAL INTERSECTION MENU ITEM
        CALL PMANU(WKID, PREV, NVALS, IVAL, MINVAL, MAXVAL)
      ELSE IF (ICL .EQ. PVALUA) THEN
C       PROCESS INTERSECTION PLANE VALUATOR
        CALL PIPLVL(WKID, IDNR, NVALS, IVAL, MINVAL, MAXVAL)
      ELSE IF (ICL .EQ. PSTRIN) THEN
C       PROCESS INTERSECTION PLANE STRING INPUT
        CALL PIPLST(WKID, NVALS, IVAL, MINVAL, MAXVAL)
      ELSE
C       WRITE ERROR MESSAGE
        CALL MSG(WKID, 'PICK A MENU ITEM ...')
      ENDIF

      RETURN
      END

C=====================================================================
C                     M O D U L E  P I P L V L
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU.2.4.4.3
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL PIPLVL(WKID, IDNR, NVALS, IVAL, MINVAL, MAXVAL)
C=====================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C     IDNR   - INPUT VALUATOR DEVICE NUMBER
C     NVALS  - NUMBER OF VALUATORS IN USE
C     IVAL   - CURRENT VALUE OF THE VALUATOR
C     MINVAL - MINIMUM VALUE OF THE VALUATOR
C     MAXVAL - MAXIMUM VALUE OF THE VALUATOR
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
```

```
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     GETS AND PROCESSES INTERSECTION PLANE VALUATOR INPUT
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     PNMVL  - PROCESS VALUATOR INPUT
C     MSG    - WRITES A SCREEN MESSAGE
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (13:26:41)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE PIPLVL(WKID, IDNR, NVALS, IVAL, MINVAL, MAXVAL)
      INTEGER WKID, IDNR, NVALS
      REAL IVAL(*), MINVAL(*), MAXVAL(*)

C     PROCESS NUMBER VALUATOR
      CALL PNMVL(WKID, IDNR, NVALS, IVAL, MINVAL, MAXVAL)

C     SEND VALUATOR INFO TO INTERSECTION PLANE ROUTINE
      CALL PLDEF(WKID, IVAL(1), IVAL(2))

      RETURN
      END
C=====================================================================
C                        M O D U L E   P I P L S T
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU.2.4.4.4
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL PIPLST(WKID, NVALS, IVAL, MINVAL, MAXVAL)
C=====================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C     NVALS  - NUMBER OF VALUATORS IN USE
C     IVAL   - CURRENT VALUE OF THE VALUATOR
C     MINVAL - MINIMUM VALUE OF THE VALUATOR
C     MAXVAL - MAXIMUM VALUE OF THE VALUATOR
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
```

```
C LOCAL VARIABLES:
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     GETS AND PROCESSES INTERSECTION PLANE STRING INPUT
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     PNMST  - PROCESS VALUATOR INPUT
C     MSG    - WRITES A SCREEN MESSAGE
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (13:26:41)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE PIPLST(WKID, NVALS, IVAL, MINVAL, MAXVAL)
      INTEGER WKID, NVALS
      REAL IVAL(*), MINVAL(*), MAXVAL(*)

C     PROCESS NUMBER STRING
      CALL PNMST(WKID, NVALS, IVAL, MINVAL, MAXVAL)

C     SEND VALUATOR INFO TO INTERSECTION PLANE ROUTINE
      CALL PLDEF(WKID, IVAL(1), IVAL(2))

      RETURN
      END
C=====================================================================
C                        M O D U L E  P M A N U
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU.2.4.4.2
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C       CALL PMANU(WKID, PREV, NVALS, IVAL, MINVAL, MAXVAL)
C=====================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C     NVALS  - INTEGER, NUMBER OF VALUATORS
C     PREC   - INTEGER, ARRAY OF PRECISION VALUES FOR VALUATORS
C     IVAL   - REAL, ARRAY CONTAINING VALUES OF THE VALUATORS
C     MINVAL - REAL, ARRAY CONTAINING THE MINIMUM VALUATOR VALUES
C     MAXVAL - REAL, ARRAY CONTAINING THE MAXIMUM VALUATOR VALUES
C=====================================================================
C OUTPUT PARAMETERS:
C     PREV - LOGICAL, PREVIOUS MENU FLAG
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     MTYPE  - INTEGER, MENU TYPE PICKED BY USER (RETURNED BY GTMNTI)
```

```
C     ITEM   - INTEGER, MENU ITEM PICKED BY USER (RETURNED BY GTMNTI)
C     CURMNU - CHARACTER, CURRENT MENU IDENTIFIER
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     PROCESSES MANUAL INTERSECTION MENU INPUT
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     PIMANU - PROCESSES PICKED GEOMETRY MENU ITEM
C     PRSTMN - PROCESSES PICKED STANDARD MENU ITEM
C     PRVWPK - PROCESSES A PICKED VIEW MENU ITEM
C     MSG    - WRITES A MESSAGE ON THE SCREEN
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (13:40:55)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================

      SUBROUTINE PMANU(WKID, PREV, NVALS, IVAL, MINVAL, MAXVAL)
      INTEGER NVALS
      REAL IVAL(*), MINVAL(*), MAXVAL(*)
      INTEGER WKID
      LOGICAL PREV

      INTEGER MTYPE, ITEM, MREG, MSTND, MVIEW
      PARAMETER(MREG = 1, MSTND = 2, MVIEW = 5)
      CHARACTER*4 CURMNU
      PARAMETER(CURMNU = 'MANU')

C     GET MENU TYPE AND ITEM
      CALL GTMNTI(MTYPE, ITEM)

      IF (MTYPE .EQ. MREG) THEN
C       PROCESS MENU ITEM
        CALL PIMANU(WKID, ITEM, PREV, NVALS, IVAL, MINVAL, MAXVAL)
      ELSE IF (MTYPE .EQ. MSTND) THEN
C       PROCESS STANDARD MENU ITEM
        CALL PRSTMN(WKID, CURMNU, ITEM, PREV)
      ELSE IF (MTYPE .EQ. MVIEW) THEN
C       PROCESS VIEW PICK
        CALL PRVWPK(WKID,ITEM)
      ELSE
C       WRITE ERROR MESSAGE
        CALL MSG(WKID, 'PICK A MENU ITEM ...')
      ENDIF

      RETURN
      END

C=====================================================================
C                      M O D U L E  P I M A N U
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU.2.4.4.2.2
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
```

```
C=====================================================================
C CALLING SEQUENCE:
C     CALL PIMANU(WKID, ITEM, PREV, NVALS, IVAL, MINVAL, MAXVAL)
C=====================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C     ITEM   - INTEGER, MENU ITEM PICKED
C     NVALS  - INTEGER, NUMBER OF VALUATORS
C     PREC   - INTEGER, ARRAY OF PRECISION VALUES FOR VALUATORS
C     IVAL   - REAL, ARRAY CONTAINING VALUES OF THE VALUATORS
C     MINVAL - REAL, ARRAY CONTAINING THE MINIMUM VALUATOR VALUES
C     MAXVAL - REAL, ARRAY CONTAINING THE MAXIMUM VALUATOR VALUES
C=====================================================================
C OUTPUT PARAMETERS:
C     PREV - LOGICAL, PREVIOUS MENU FLAG
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     PROCESSES A MANUAL INTERSECTION MENU PICK
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     CLNUM - CLOSES NUMERICAL INPUT
C     ACINT - COMPUTE AN INTERSECTION
C     AXSVW - CROSS-SECTION VIEW
C     ASHPL - SHO INTERSECTION PLANE
C     ASVFL - SAVE FILE
C     OPNUM - OPENS NUMERICAL INPUT
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (09:19:01)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================

      SUBROUTINE PIMANU(WKID, ITEM, PREV, NVALS, IVAL, MINVAL, MAXVAL)
      REAL IVAL(*), MINVAL(*), MAXVAL(*)
      INTEGER WKID, ITEM, NVALS
      LOGICAL PREV

C     CLOSE NUMBER INPUT
      CALL CLNUM(WKID)

      IF (ITEM .EQ. 1) THEN
C       COMPUTE AN INTERSECTION
        CALL ACINT(WKID, IVAL)
      ELSE IF (ITEM .EQ. 2) THEN
C       XSEC VIEW
        CALL AXSVW(WKID)
      ELSE IF (ITEM .EQ. 3) THEN
C       SHO/NO SHO INTERSECTION PLANE
```

```
        CALL ASHPL(WKID)
      ELSE IF (ITEM .EQ. 4) THEN
C       SAVE FILE
        CALL ASVFL(WKID)
      ELSE IF (ITEM .EQ. 5) THEN
        PREV = .TRUE.
      ENDIF

C     OPEN NUMBER INPUT
      CALL OPNUM(WKID)

      RETURN
      END
C=======================================================================
C                        M O D U L E   A C I N T
C=======================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  ACINT
C=======================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C     CALL ACINT(WKID, IVAL)
C=======================================================================
C INPUT PARAMETERS:
C     WKID - INTEGER, WORKSTATION IDENTIFIER
C     IVAL - REAL, ARRAY CONTAINING CURRENT VALUATOR VALUES
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C LOCAL VARIABLES:
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     INTERSECTION CHAINING MODULE
C=======================================================================
C EXAMPLE(S):
C     N/A
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C MODULES CALLED:
C     GTIVWS - QUERIES FOR ACTIVE VIEWS
C     XSVWST - BRINGS UP A X-SEC VIEW
C     XSTILE - TILES THE AIRCRAFT GEOMETRY
C     INSECT - COMPUTES INTERSECTION BETWEEN INTERSECT PLANE AND TILES
C     MCINT - INTERSECTION MENU DRIVER
C     XSVWDL - DELETES AN X-SEC VIEW
C=======================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=======================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=======================================================================
```

```
      SUBROUTINE ACINT(WKID,IVAL)
      INTEGER WKID
      REAL INTARR(3,900),IVAL(*)
      INTEGER VWITM,NVWITM,VIEWID,ACTIVE,ERRIND
      PARAMETER (VWITM = 1, NVWITM = 1, VIEWID = 11)


C     QUERY VIEW TO CHECK IF X-SEC VIEW IS ACTIVE
      CALL GTIVWS(VWITM,NVWITM,VIEWID,ACTIVE,ERRIND)

C     IF X-SEC VIEW DISCARDED, MAKE IT APPEAR
      IF (ACTIVE .EQ. 0) THEN
         CALL XSVWST(WKID)
      END IF

C     TILE THE AIRCRAFT GEOMETRY(LOAD THE TILE DATA BLOCKS)
      CALL XSTILE

C     COMPUTE INTERSECTION
      CALL INSECT(IVAL,INTARR)

C     CHAIN MODE MENU
      CALL MCINT(WKID, INTARR)

C     IF X-SEC VIEW HAD BEEN DISCARDED, DISCARD IT ONCE AGAIN
      IF (ACTIVE .EQ. 0) THEN
         CALL XSVWDL(WKID)
      END IF

      RETURN
      END
C=======================================================================
C                        M O D U L E   I N S E C T
C=======================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  ACINT.3
C=======================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C     CALL INSECT( IVAL, INTARR)
C=======================================================================
C INPUT PARAMETERS:
C     IVAL - REAL, ARRAY CONTAINING CURRENT VALUATOR VALUES
C=======================================================================
C OUTPUT PARAMETERS:
C     INTARR - REAL, ARRAY CONTAINING THE POINTS OF INTERSECTION
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C LOCAL VARIABLES:
C   BOUND - ARRAY CONTAINING THE EXTREME X VALUES OF THE INT PLANE
C   PLLEN - LENGTH OF INTERSECTION PLANE ABOVE X AXIS
C   PT    - ARRAY CONTAINING COORDS OF THE LINE SEGMENT
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     INTERSECTION MODULE
C=======================================================================
C EXAMPLE(S):
C     N/A
C=======================================================================
```

```
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C MODULES CALLED:
C     POLYPT - CONNECTS THE VERTICES OF THE POLYGONS TO FORM A LINE
C     TSTBND - TESTS LINES TO SEE IF THEY INTERSECT THE PLANE
C=======================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=======================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=======================================================================
      SUBROUTINE INSECT(IVAL,INTARR)
      REAL IVAL(*),INTARR(3,900),PLLEN,BOUND(2),PT(3,2),PI
C     THE PARAMETER PLLEN REPRESENTS THE LENGTH OF THE INTERSECTION
C     PLANE ABOVE THE X AXIS.
      PARAMETER (PLLEN = 50.,PI = 3.141579)

      COMMON /POLDAT/ NPARTS,PRTLMT,NSUBT,NPSUB,XYZSUB
      INTEGER NPART, NPSUB(20000), PRTLMT(2,100), NSUBT, VERT
      REAL XYZSUB(3,20000,5)

      COMMON /INTRSC/ NUMINT
      INTEGER NUMINT

      COMMON /CHNDAT/ NPTCHD,HOLDAR
      INTEGER NPTCHD,HOLDAR(900)

C     INITIALIZE THE NUMBER OF INTERSECTIONS AND NUMBER OF CHAINED PTS
      NUMINT = 0
      NPTCHD = 0

C     NOW SET UP THE X BOUNDARIES FOR THE SEARCH VOLUME
      ADDX = PLLEN * COS((90.-IVAL(2)) * PI / 180.)
      BOUND(1) = IVAL(1) - ADDX
      BOUND(2) = IVAL(1) + ADDX


C     LOOP THRU THE POLYGONS THAT TILE THE AIRCRAFT
      DO 10 IPOL = 1,NSUBT

         VERT = NPSUB(IPOL)

C     LOOP THRU THE VERTICES
         DO 20 IPOINT = 1,VERT

C     CONNECT THE VERTICES OF THE POLYGON TO DETERMINE IF THEY'RE
C     WITHIN RANGE OF THE INTERSECTION
            CALL POLYPT(PT,VERT,IPOINT,IPOL)

C     TEST VERTICE DATA TO SEE IF WITHIN BOUNDARIES
            CALL TSTBND(IVAL,BOUND,PT,INTARR)

   20    CONTINUE

   10 CONTINUE

      RETURN
      END
C=======================================================================
C                        M O D U L E   P O L Y P T
C=======================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
```

```
C   PART #:
C MODULE #:  ACINT.3.6
C=======================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C     CALL POLYPT (PT, VERT, IPOINT, IPOL)
C=======================================================================
C INPUT PARAMETERS:
C     VERT - TOTAL NUMBER OF VERTICES IN POLYGON
C     IPOINT - CURRENT VERTICE NUMBER
C     IPOL  - CURRENT POLYGON NUMBER
C=======================================================================
C OUTPUT PARAMETERS:
C     PT - ARRAY CONTAINING THE ENDS OF THE LINES SEG AT EDGE OF POLY
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C LOCAL VARIABLES:
C   XYZSUB- ARRAY CONTAINING COORDS AND # OF VERTICES FOR EACH POLYGON
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     CREATES NEW ARRAY WHICH REPRESENTS THE BORDERS OF THE POLYGON
C=======================================================================
C EXAMPLE(S):
C     N/A
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C MODULES CALLED:
C     NONE
C=======================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=======================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=======================================================================
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     CREATES NEW ARRAY WHICH REPRESENTS THE BORDERS OF THE POLYGON
C=======================================================================
      SUBROUTINE POLYPT(PT,VERT,IPOINT,IPOL)
      REAL PT(3,*), VERT
      INTEGER IPOINT, IPOL

      COMMON /POLDAT/ NPARTS,PRTLMT,NSUBT,NPSUB,XYZSUB
      INTEGER NPART, NPSUB(20000), PRTLMT(2,100), NSUBT
      REAL XYZSUB(3,20000,5)

C     CREATE NEW ARRAY CALLED PT
      DO 10 J = 1,3

C     CONNECT EACH VERTEX  TO FOLLOWING VERTEX UNLESS IT'S THE LAST,
C     THEN CONNECT LAST TO FIRST VERTEX
         IF (IPOINT .LT. VERT) THEN
            PT(J,1) = XYZSUB(J,IPOL,IPOINT)
            PT(J,2) = XYZSUB(J,IPOL,IPOINT + 1)

         ELSE
```

```
            PT(J,1) = XYZSUB(J,IPOL,IPOINT)
            PT(J,2) = XYZSUB(J,IPOL,1)
          END IF
   10 CONTINUE

      RETURN
      END
C=====================================================================
C                        M O D U L E   T S T B N D
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  ACINT.3.7
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL TSTBND (IVAL, BOUND, PT, INTARR)
C=====================================================================
C INPUT PARAMETERS:
C     IVAL - REAL, ARRAY CONTAINING CURRENT VALUATOR VALUES
C     BOUND - REAL, ARRAY CONTAINING THE EXTREME X VALUES OF INT PLANE
C     PT - REAL, ARRAY CONTAINING ENDPTS OF THE POLYGONAL LINE SEGMENT
C=====================================================================
C OUTPUT PARAMETERS:
C     INTARR - REAL, ARRAY CONTAINING THE POINTS OF INTERSECTION
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C   NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C   PERFORMS A GENERAL TEST ON POLYGON DATA TO DETERMINE IF IT IS
C   WITHIN INTERSECTION BOUNDARIES OR NOT
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     TSTINT - PERFORMS A SECONDARY TEST ON LINE SEGMENTS
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE TSTBND(IVAL,BOUND,PT,INTARR)
      REAL IVAL(*), BOUND(*), PT(3,*),INTARR(3,*)


C     KEEP DATA IN CONSIDERATION IF AT LEAST ONE VERTEX IS WITHIN
C     THE INTERSECTION BOUNDARY
      IF(IVAL(2) .NE. 0.) THEN
         IF((PT(1,1) .GE. BOUND(1)) .OR. (PT(1,2) .GE. BOUND(1))) THEN
```

```
            IF((PT(1,1) .LE. BOUND(2)) .OR.
     $          (PT(1,2) .LE. BOUND(2))) THEN

C              CALL TSTINT(IVAL,PT,INTARR)

            END IF
          END IF

C     IF THE INTERSECTION PLANE IS VERTICAL SEND THE DATA ONTO THE
C     NEXT LEVEL
      ELSE
         CALL TSTINT(IVAL,PT,INTARR)
      END IF

      RETURN
      END
C=======================================================================
C                       M O D U L E   T S T I N T
C=======================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:
C=======================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C     CALL TSTINT (IVAL, PT, INTARR)
C=======================================================================
C INPUT PARAMETERS:
C     IVAL - REAL, ARRAY CONTAINING CURRENT VALUATOR VALUES
C     PT - REAL, ARRAY CONTAINING ENDPTS OF THE POLYGONAL LINE SEGMENT
C=======================================================================
C OUTPUT PARAMETERS:
C     INTARR - REAL, ARRAY CONTAINING THE POINTS OF INTERSECTION
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C LOCAL VARIABLES:
C   A, B, AND C - ARE VECTORS WHICH DEFINE THE SURFACE OF THE INT PLANE
C   NORM - VECTOR NORMAL TO THE INTERSECTION PLANE
C   P10 - VECTOR BETWEEN PT 1 OF THE LINE AND THE CENTER OF THE PLANE
C   P20 - VECTOR BETWEEN PT 2 OF THE LINE AND THE CENTER OF THE PLANE
C   VEC1 AND VEC2 - SCALAR PRODUCTS OF PT VECTORS AND NORM OF THE PLANE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     PERFORMS A SECONDARY TEST ON THE POLYGONAL DATA WHICH IS CANDIDATE
C     FOR INTERSECTION
C=======================================================================
C EXAMPLE(S):
C     N/A
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C MODULES CALLED:
C     TSTINT - PERFORMS A SECONDARY TEST ON LINE SEGMENTS
C=======================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=======================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
```

```
C     VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C     BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE TSTINT(IVAL,PT,INTARR)
      REAL IVAL(*), PT(3,*), INTARR(3,*)
      REAL PWID,PLEN,PI
      REAL B(3), A(3), C(3), BMAG, CMAG, NORM(3)
      REAL NOOM1, NOOM2, DENOM, P10(3), P20(3), P0(3)
      PARAMETER(PWID = 50., PLEN = 50., PI = 3.1415927)

      COMMON /INTRSC/ NUMINT
      INTEGER NUMINT

C     DEFINE TWO UNIT VECTORS ON THE PLANE'S SURFACE
      A(1) = IVAL(1)
      A(2) = PWID
      A(3) = 0.

      B(1) = 0.
      B(2) = -PWID
      B(3) = 0.

      C(1) = 0.
      C(2) = 0.
      C(3) = PLEN/COS(IVAL(2) * PI /180.)

C     CALL ROUTINE TO PERFORM VECTOR OPERATIONS
      CALL VECOPS(IVAL, PT, P10, P20, B, C, NORM, VEC1, VEC2)

C     SEE IF BOTH POINTS LIE ON THE PLANE, IF YES RETURN BOTH POINTS
C     AS PART OF THE INTERSECTION
      IF((VEC1 .EQ. 0.) .AND. (VEC2 .EQ. 0.)) THEN

C        INCREMENT NUMBER OF INTERSECTION POINTS BY ONE
         NUMINT = NUMINT + 1

C        CONSIDER THEM AS POINTS IN THE INTERSECTION ARRAY
         DO 10 I = 1,3
            INTARR(I,NUMINT) = PT(I,1)
            INTARR(I,NUMINT+1) = PT(I,2)
   10    CONTINUE

C        INCREMENT NUMBER OF INTERSECTION POINTS AGAIN
         NUMINT = NUMINT + 1

C     IF THEY ARE ON OPP SIDES CALC THE INTERSECTION
      ELSE IF(((VEC1 .GE. 0.) .AND. (VEC2 .LE. 0.)) .OR.
     $((VEC2 .GE. 0.) .AND. (VEC1 .LE. 0.))) THEN


C     INCREMENT THE NUMBER OF INTERSECTIONS
         NUMINT = NUMINT + 1

C     CALL INTERSECTION ALGORITHM
         CALL DOINT(PT,A,NORM,INTARR)

      END IF

      RETURN
      END
C=====================================================================
C                       M O D U L E   V E C O P S
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:
```

```
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL VECOPS(IVAL, PT, P10, P20, B, C, NORM, VEC1, VEC2)
C=====================================================================
C INPUT PARAMETERS:
C     IVAL - REAL, ARRAY CONTAINING CURRENT VALUATOR VALUES
C     PT - REAL, ARRAY CONTAINING ENDPTS OF THE POLYGONAL LINE SEGMENT
C     B, AND C - ARE VECTORS WHICH DEFINE THE SURFACE OF THE INT PLANE
C     P10 - VECTOR BETWEEN PT 1 OF THE LINE AND THE CENTER OF THE PLANE
C     P20 - VECTOR BETWEEN PT 2 OF THE LINE AND THE CENTER OF THE PLANE
C=====================================================================
C OUTPUT PARAMETERS:
C     VEC1 AND VEC2 - SCALAR PRODUCTS OF PT VECTORS AND NORM OF THE PLANE
C     NORM - VECTOR NORMAL TO THE INTERSECTION PLANE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C   NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     PERFORMS A BUNCH OF VECTOR OPERATIONS FOR THE INTERSECTION
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     DOT - CALCULATES THE DOT PRODUCT OF TWO VECTORS
C     CROSS - CALCULATES THE CROSS PRODUCT OF TWO VECTORS
C     INTVEC - MORE INTERMEDIATE VECTOR CALCULATIONS
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE VECOPS(IVAL, PT, P10, P20, B, C, NORM, VEC1, VEC2)
      REAL IVAL(*), PT(3,*)
      REAL B(3), C(3), NORM(3)
      REAL P10(3), P20(3), P0(3)


C     PERFORM SOME INTERMEDIATE VECTOR CALCS
      CALL INTVEC(IVAL,PT,P10,P20,B,C)

C     CALCULATE THE VECTOR NORMAL TO THE INTERSECTION PLANE
      CALL CROSS(B,C,NORM)

C     CALCULATE THE DOT PRODUCT OF P10 AND P20 WITH THE NORMAL VECTOR
C     TO DETERMINE IF THEY ARE ON OPPOSITE SIDES OF THE INTERSECTION
C     PLANE
      CALL DOT(P10,NORM,VEC1)
      CALL DOT(P20,NORM,VEC2)

      RETURN
```

```
      END
C=======================================================================
C                        M O D U L E   I N T V E C
C=======================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:
C=======================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C     CALL INTVEC(IVAL,PT,P10,P20,B,C)
C=======================================================================
C INPUT PARAMETERS:
C     IVAL - REAL, ARRAY CONTAINING CURRENT VALUATOR VALUES
C     PT - REAL, ARRAY CONTAINING ENDPTS OF THE POLYGONAL LINE SEGMENT
C     B, AND C - ARE VECTORS WHICH DEFINE THE SURFACE OF THE INT PLANE
C     P10 - VECTOR BETWEEN PT 1 OF THE LINE AND THE CENTER OF THE PLANE
C     P20 - VECTOR BETWEEN PT 2 OF THE LINE AND THE CENTER OF THE PLANE
C=======================================================================
C OUTPUT PARAMETERS:
C     P10 - VECTOR BETWEEN PT 1 OF THE LINE AND THE CENTER OF THE PLANE
C     P20 - VECTOR BETWEEN PT 2 OF THE LINE AND THE CENTER OF THE PLANE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C LOCAL VARIABLES:
C   NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     PERFORMS A BUNCH OF VECTOR OPERATIONS FOR THE INTERSECTION
C=======================================================================
C EXAMPLE(S):
C     N/A
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C MODULES CALLED:
C     VECVAL - CALCULATES THE MAGNITUDE OF A VECTOR
C=======================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=======================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=======================================================================
      SUBROUTINE INTVEC(IVAL,PT,P10,P20,B,C)
      REAL IVAL(*),PT(3,*),P10(*),P20(*),B(*),C(*)
      REAL BMAG,CMAG,P0(3)

C     DEFINE POINT ON THE PLANE WHICH IS STATIC NO MATTER WHAT
C     ORIENTATION THE PLANE IS IN
      P0(1) = IVAL(1)
      P0(2) = 0.
      P0(3) = 0.

C     CALCULATE THE MAGNITUDES OF B AND C SO AS TO NORMALIZE THEM
      CALL VECVAL(B,BMAG)
      CALL VECVAL(C,CMAG)
```

```
C     PERFORM A SECONDARY CHECH TO DETERMINE IF INDEED, THERE WILL BE
C     AN INTERSECTION BETWEEN LINE AND PLANE
      DO 10 I = 1,3
         P10(I) = PT(I,1) - P0(I)
         P20(I) = PT(I,2) - P0(I)
         B(I) = B(I)/BMAG
         C(I) = C(I)/CMAG
   10 CONTINUE

      RETURN
      END
C=======================================================================
C                        M O D U L E   D O I N T
C=======================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:
C=======================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C     CALL DOINT (PT, A, NORM, INTARR)
C=======================================================================
C INPUT PARAMETERS:
C     PT - REAL, ARRAY CONTAINING ENDPTS OF THE POLYGONAL LINE SEGMENT
C     A - IS A VECTOR WHICH DEFINE THE SURFACE OF THE INT PLANE
C     NORM - VECTOR NORMAL TO SURFACE OF INTERSECTION PLANE
C=======================================================================
C OUTPUT PARAMETERS:
C     INTARR - ARRAY WHICH CONTAINS THE POINTS OF INTERSECTION
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C LOCAL VARIABLES:
C     E - VECTOR WHICH DEFINES THE LINE SEGMENT WHICH WILL BE INTESECTED
C     P1 - NEW ARRAY WHICH CONTAINS THE COORDS OF PT1 OF THE LINE SEG
C     NOOM1 - PART OF THE NUMERATOR OF THE GOVERNING INT EQUATION
C     NOOM2 - PART OF THE NUMERATOR OF THE GOVERNING INT EQUATION
C     DENOM - THE DENOMINATOR OF THE GOVERNING INT EQUATION
C     T - PARAMETRIC LOCATION OF THE INTERSECTION (TELLS WHERE PLANE
C         HITS THE LINE SEGMENT)
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     PERFORMS THE INTERSECTION BETWEEN A LINE SEG AND A PLANE
C=======================================================================
C EXAMPLE(S):
C     N/A
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C MODULES CALLED:
C     DOT - DOT PRODUCT OF TWO VECTORS
C=======================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=======================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
```

```
C    BLACKSBURG, VA  24061
C=======================================================================
      SUBROUTINE DOINT(PT,A,NORM,INTARR)
      REAL PT(3,*), A(*), NORM(*), INTARR(3,*),E(3),P1(3), NOOM1, NOOM2

      COMMON /INTRSC/ NUMINT
      INTEGER NUMINT

C     DEFINE THE VECTOR E AS THE DIFFERENCE BETWEEN PT1 AND PT2
      DO 10 I = 1,3
         E(I) = PT(I,2) - PT(I,1)
         P1(I) = PT(I,1)
   10 CONTINUE

C     FIND THE PARAMETRIC VALUE OF THE INTERSECTION POINT ALONG THE
C     LENGTH OF THE LINE SEGMENT
      CALL DOT(NORM,A,NOOM1)
      CALL DOT(NORM,P1,NOOM2)
      CALL DOT(NORM,E,DENOM)

      T = (NOOM1 - NOOM2)/DENOM


C     USE THE EQUATION OF THE LINE TO FIND THE POINT OF INTERSECTION
      DO 33 I = 1,3
         INTARR(I, NUMINT) = P1(I) + T*E(I)
   33 CONTINUE

      RETURN
      END

C=======================================================================
C                        M O D U L E   M C I N T
C=======================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  ACINT.4
C=======================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C     CALL MCINT(WKID, INTARR)
C=======================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     INTARR - REAL, ARRAY CONTAINING THE POINTS OF INTERSECTION
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C LOCAL VARIABLES:
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     INTERSEC MENU DRIVER
C=======================================================================
C EXAMPLE(S):
C     N/A
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
```

```
C MODULES CALLED:
C     NEWMNU - CREATES INTERSECTION MENU ITEMS
C     PUTINT - PLACES INTERSECTION PTS IN A STRUCTURE
C     EXCINT - EXECUTES INTERSECTION MENU ITEM
C     EPTST - DELETES INTERSECTION PT STRUCTURE
C     OLDMNU - REPLACES OLD MENU UPON LEAVING
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:45:24)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================

      SUBROUTINE MCINT(WKID, INTARR)
      INTEGER WKID
      REAL INTARR(3,*)

      CHARACTER*12 TITLE
      PARAMETER(TITLE = ' CHAIN MODE ')

      INTEGER NITEMS
      PARAMETER(NITEMS = 3)
      CHARACTER*16 ITEMS(NITEMS)

C     DEFINE MENU ITEMS
      DATA ITEMS / '    AUTO CHAIN  ',
     >             '  RESTART CHAIN ',
     >             '     RETURN     ' /

C     DRAW NEW MENU
      CALL NEWMNU(WKID, TITLE, NITEMS, ITEMS)

C     SET UP DEFAULT VIEW #11 STUFF HERE AND DROP INPUT MODE
      CALL PUTINT(WKID,INTARR)

C     EXECUTE GEOMETRY MENU
      CALL EXCINT(WKID, INTARR)

C     GET RID OF VIEW STUFF AND THE STRUCTURE WHICH CONTAINS THE POINTS
C     REINSTATE THE INPUT STUFF
      CALL EPTST(WKID)

C     DISPLAY OLD MENU
      CALL OLDMNU(WKID)

      RETURN
      END

C=====================================================================
C                       M O D U L E   P U T I N T
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL PUTINT(WKID,INTARR)
C=====================================================================
C INPUT PARAMETERS:
C     WKID - INTEGER, WORKSTATION IDENTIFIER
C     INTARR - ARRAY CONTAINING INTERSECTION POINTS
```

```
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     STID - STRUCTURE ID IN WHICH TO PUT INTERSECTION POINTS
C     VIEW - VIEW ID FOR THE INTERSECTION POINTS
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     CREATES A MENU TYPE AND SUBSTRUCTURE FOR THE INTERSECTION PTS
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     GTIVWS - GETS THE STRUCTURE ID FOR VIEW 11
C     POPST - PHIGS OPEN STRUCTURE
C     PSVWI - PHIGS ASSOCIATE STRUCTURE WITH VIEW
C     PSLN - PHIGS SET POLYLINE TYPE
C     PSPCLI- PHIGS SET POLYLINE COLOR
C     INTSTR- PUT POINTS IN A STRUCTURE
C     PCLST- PHIGS CLOSE STRUCTURE
C     PTROOT - CREATES A ROOT MENU TYPE FOR STRUCTURE
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE PUTINT(WKID,INTARR)
      INTEGER WKID, ITEM, NITEM, STID, ERRIND, VIEW
      PARAMETER( ITEM = 4, NITEM = 1,VIEW = 11)
      REAL INTARR(3,*)

C     GET THE SHADING STRUCTURE CORRESP TO THE VIEW 11
      CALL GTIVWS(ITEM, NITEM, VIEW, STID, ERRIND)

C     OPEN THE STRUCTURE AND MAKE POINTS
      CALL POPST(STID)

C     ASSOCIATE THIS STRUCTURE WITH VIEW 11
      CALL PSVWI(VIEW)

C     SET THE POLYLINE TYPE
      CALL PSLN (1)

C     SET THE POLYLINE COLOR INDEX
      CALL PSPLCI (15)

C     PUT THE INTERSECTION POINTS IN A STRUCTURE
      CALL INTSTR(WKID, INTARR)

C     CLOSE THE STRUCTURE
      CALL PCLST
```

```
C     CREATE A ROOT STRUCTURE TO PUT THE POINTS INTO
      CALL PTROOT(STID,VIEW,WKID)

      RETURN
      END
C=======================================================================
C                        M O D U L E   I N T S T R
C=======================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=======================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C     CALL INTSTR(WKID,INTARR)
C=======================================================================
C INPUT PARAMETERS:
C     WKID - INTEGER, WORKSTATION IDENTIFIER
C     INTARR - ARRAY CONTAINING INTERSECTION POINTS
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C LOCAL VARIABLES:
C     NUMINT - NUMBER OF INTERSECTION POINTS
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     PLACES INTERSECTION POINTS IN A STRUCTURE WHICH IS THE CHILD OF
C     A ROOT STRUCTURE ASSOCIATED WITH VIEW 11
C=======================================================================
C EXAMPLE(S):
C     N/A
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C MODULES CALLED:
C     PADS - PHIGS ADD NAMES TO SET
C     PSMK - PHIGS SET POLYMARKER TYPE
C     PSHPID - PHIGS SET PICK FILTER
C     PLB - PHIGS LABEL
C     PSPCLI- PHIGS SET POLYLINE COLOR
C     PPM3 - PHIGS CREATE POLYMARKER
C=======================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=======================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=======================================================================
      SUBROUTINE INTSTR(WKID, INTARR)
      INTEGER WKID, WHITE,RED
      PARAMETER( WHITE = 7,RED = 1)
      REAL INTARR(3,*)

      COMMON /INTRSC/ NUMINT
      INTEGER NUMINT
```

```
C     GIVE THE POINTS A CLASS NAME
         CALL PADS(1,103)

C     MAKE THE POLYMARKERS INTO POINTS
      CALL PSMK(1)

C     GIVE EACH POINT IN THE STRUCTURE THE SAME PICKID AS ITS
C     NUMBER INCREMENT IN TERMS OF INTERSECTION POINT NUMBER
      DO 10 I = 1,NUMINT

C     SET PICK ID SAME AS INT PT NUMBER
         CALL PSHPID(I)

C     ADD A LABEL TO THE STRUCTURE
         CALL PLB(I)

C     SET POLYMARKER COLOR TO WHITE
         CALL PSPMCI(WHITE)

C     DRAW A 3D POINT
         CALL PPM3(1,INTARR(1,I), INTARR(2,I), INTARR(3,I))
   10 CONTINUE

      RETURN
      END
C=====================================================================
C                        M O D U L E   P T R O O T
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL PTROOT(STID,VIEW,WKID)
C=====================================================================
C INPUT PARAMETERS:
C     STID - INTEGER, STRUCTURE IDENTIFIER
C     VIEW - INTEGER, VIEW IDENTIFIER
C     WKID - INTEGER, WORKSTATION IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     NUMINT - NUMBER OF INTERSECTION POINTS
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     CREATES A ROOT AND MENU TYPE WHICH CONTAINS THE INT PTS
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     GTIVWS - FINDS THE ROOT STRUCTURE ID
```

```
C     CRMNRT - CREATES A MENU ROOT
C     PPORT - PHIGS POST ROOT
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE PTROOT(STID, VIEW, WKID)
      INTEGER STID, VIEW, ITEM, NITEMS, ERRIND, NSTID, TYPE, WKID
      REAL PRIOR
      PARAMETER( ITEM = 2, NITEMS = 1,TYPE = 6, PRIOR = 1.0)

C     QUERY TO FIND THE ROOT STRUCTURE
      CALL GTIVWS(ITEM, NITEMS, VIEW, NSTID, ERRIND)

C     CREATE A MENU ROOT OF TYPE = 6
      CALL CRMNRT(NSTID, VIEW, TYPE, STID)

C     POST ROOT
      CALL PPORT(WKID, NSTID, PRIOR)

      RETURN
      END
C=====================================================================
C                       M O D U L E  E P T S T
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL EPTST(WKID)
C=====================================================================
C INPUT PARAMETERS:
C     NONE
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     STID - STRUCTURE IDS OF THE INTERSECT PT STRUCTURE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     EMPTIES THE STRUCTURE CONTAINING THE INTERSECTION POINTS
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     GTIVWS - FINDS THE ROOT STRUCTURE ID
C     PEMST - PHIGS EMPTY STRUCTURE
```

```
C     CRROOT - CREATES A ROOT WHICH CONTAINS THE AIRCRAFT
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE EPTST(WKID)
      INTEGER ITEM, NITEMS, ERRIND, STID,WKID,ITEM1
      PARAMETER (ITEM = 4, NITEMS = 1,ITEM1 = 2)

C     GET THE SUBSTRUCTURE ID
      CALL GTIVWS(ITEM, NITEMS, 11, STID, ERRIND)

C     EMPTY SUBSTRUCTURE
      CALL PEMST(STID)

C     GET THE ROOT STRUCTURE ID
      CALL GTIVWS(ITEM1,NITEMS,11,STID,ERRIND)

C     CREATE THE ROOT CONTAINING THE AIRCRAFT
      CALL CRROOT(WKID,STID,11)

      RETURN
      END
C=====================================================================
C                      M O D U L E  E X C I N T
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  ACINT.4.3
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL EXCINT(WKID, INTARR)
C=====================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C     INTARR - REAL, ARRAY CONTAINING THE POINTS OF INTERSECTION
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     PREV - LOGICAL, PREVIOUS MENU FLAG
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     EXECUTES CHAIN MODE IN THE INTERSECTION MODULE
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C=====================================================================
```

```
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (19:18:18)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================

      SUBROUTINE EXCINT(WKID, INTARR)
      INTEGER WKID
      REAL INTARR(3,*)

      LOGICAL PREV

C     ASSUME PREVIOUS MENU FLAG IS FALSE
      PREV = .FALSE.

C     WRITE SCREEN MESSAGE
      CALL MSG(WKID, 'PLEASE BEGIN CHAINING TO COMPUTE INTERSEC AREA:')

C     GET AND PROCESS INTERSECTION MENU INPUT
 100  CALL INCINT(WKID, PREV, INTARR)

      IF (.NOT. PREV) THEN
        GOTO 100
      ENDIF

      RETURN
      END

C=====================================================================
C                      M O D U L E   I N C I N T
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  ACINT.4.3.3
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL INCINT(WKID, PREV, INTARR)
C=====================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C     PREV - LOGICAL, PREVIOUS MENU FLAG
C     INTARR - REAL, ARRAY CONTAINING THE POINTS OF INTERSECTION
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     GETS AND PROCESSES MANUAL INTERSECTION MENU INPUT
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
```

```
C=====================================================================
C MODULES CALLED:
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (13:26:41)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================

      SUBROUTINE INCINT(WKID, PREV, INTARR)
      INTEGER WKID
      REAL INTARR(3,*)
      LOGICAL PREV

      INTEGER ICL, IDNR, PPICK, PVALU
      PARAMETER(PPICK = 5, PVALU= 3)

C     GET MENU INPUT
      CALL GTMNIN(WKID, ICL, IDNR)

      IF (ICL .EQ. PPICK) THEN
C       PROCESS MANUAL INTERSECTION MENU ITEM
        CALL PCINT(WKID, PREV, INTARR)
      ELSE IF (ICL .EQ. PVALU) THEN
C       PROCESS THE GEOMETRY VALUATORS
        CALL PGMVL(WKID, IDNR)
      ELSE
C       WRITE ERROR MESSAGE
        CALL MSG(WKID, 'PICK A MENU ITEM ...')
      ENDIF

      RETURN
      END

C=====================================================================
C                      M O D U L E   P C I N T
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  ACINT.4.3.3.2
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL PCINT(WKID, PREV, INTARR)
C=====================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C     PREV - LOGICAL, PREVIOUS MENU FLAG
C     INTARR - REAL, ARRAY CONTAINING THE PTS OF INTERSECTION
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     MTYPE  - INTEGER, MENU TYPE PICKED BY USER (RETURNED BY GTMNTI)
C     MJUNK  - INTEGER, MENU TYPE REPRESENTING THE INTERSECTION STRUC
C     ITEM   - INTEGER, MENU ITEM PICKED BY USER (RETURNED BY GTMNTI)
```

```
C     CURMNU - CHARACTER, CURRENT MENU IDENTIFIER
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     PROCESSES MANUAL INTERSECTION MENU INPUT
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     GTMNTI - GETS MENU TYPE AND MENU ITEM PICKED BY THE USER
C     PICINT - PROCESSES PICKED INTERSECTION MENU ITEM
C     PRSTMN - PROCESSES PICKED STANDARD MENU ITEM
C     CHAIN  - BEGINS THE CHAINING PORTION OF OUR PROGRAM
C     MSG    - WRITES A MESSAGE ON THE SCREEN
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (13:40:55)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================

      SUBROUTINE PCINT(WKID, PREV, INTARR)
      REAL INTARR(3,*)
      INTEGER WKID
      LOGICAL PREV

      INTEGER MTYPE, ITEM, MREG, MSTND, MVIEW
      PARAMETER(MREG = 1, MSTND = 2, MVIEW = 5, MJUNK = 6)
      CHARACTER*4 CURMNU
      PARAMETER(CURMNU = 'CINT')

C     GET MENU TYPE AND ITEM
      CALL GTMNTI(MTYPE, ITEM)

      IF (MTYPE .EQ. MREG) THEN
C       PROCESS MENU ITEM
        CALL PICINT(WKID, INTARR,ITEM, PREV)
      ELSE IF (MTYPE .EQ. MSTND) THEN
C       PROCESS STANDARD MENU ITEM
        CALL PRSTMN(WKID, CURMNU, ITEM, PREV)
      ELSE IF (MTYPE .EQ. MVIEW) THEN
C       PROCESS VIEW PICK
        CALL PRVWPK(WKID,ITEM)
      ELSE IF (MTYPE .EQ. MJUNK) THEN
C       DO THE CHAINING PROCESS
        CALL CHAIN(WKID,ITEM,INTARR)
      ELSE
C       WRITE ERROR MESSAGE
        CALL MSG(WKID, 'PICK A MENU ITEM OR PICK POINT TO CHAIN.')
      ENDIF

      RETURN
      END

C=====================================================================
C                      M O D U L E   P I C I N T
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  ACINT.4.3.3.2.2
```

```
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL PICINT(WKID, INTARR,ITEM, PREV)
C=====================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C     ITEM   - INTEGER, MENU ITEM PICKED
C=====================================================================
C OUTPUT PARAMETERS:
C     PREV - LOGICAL, PREVIOUS MENU FLAG
C     INTARR - REAL, ARRAY CONTAINING THE POINTS OF INTERSECTION
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     PROCESSES A CHAIN MODE MENU PICK
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     ARSCH - RESTARTS CHAIN MODE
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (09:19:01)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================

      SUBROUTINE PICINT(WKID, INTARR,ITEM, PREV)
      REAL INTARR(3,*)
      INTEGER WKID, ITEM
      LOGICAL PREV


      IF (ITEM .EQ. 1) THEN
C       START AUTOMATIC CHAINING OPERATION
        CALL AACHN(WKID, INTARR)

      ELSE IF (ITEM .EQ. 2) THEN
C       RESTART CHAINING OPERATION
        CALL ARSCH(WKID, INTARR)
      ELSE IF (ITEM .EQ. 2) THEN

C       GO BACK TO PREVIOUS MENU
        PREV = .TRUE.
      ENDIF

      RETURN
      END
C=====================================================================
C                       M O D U L E   C H A I N
```

```
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL CHAIN(WKID,PIKID,PTARR)
C=====================================================================
C INPUT PARAMETERS:
C     WKID - INTEGER, WORKSTATION IDENTIFIER
C     PIKID - INTEGER, ID OF PICKED POINT
C     PTARR - ARRAY CONTAINING INTERSECTION POINTS
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     HOLDAR - ARRAY WHICH ACTS AS A POINTER INTO INTERSECT ARRAY
C     NPTCHD - INTEGER, NUMBER OF POINTS IN THE CHAIN TO DATE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     ENCLOSES A REGION FOR CALCULATION OF AREA
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     CHNPIK- DECIDES WHETHER TO INCREMENT OR DECREMENT THE HOLD ARRAY
C     CALCAR - CALCULATES THE AREA OF AN ENCLOSED REGION
C     MSG - WRITES OUT AREA TO SCREEN
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE CHAIN(WKID,PIKID,PTARR)
      INTEGER WKID,PIKID
      REAL PTARR(3,*)
      CHARACTER*52 STRING

      COMMON /CHNDAT/ NPTCHD,HOLDAR
      INTEGER NPTCHD, HOLDAR(900)

C     CALL ROUTINE WHICH TAKES THE PICKID AND DETEMINES WHICH POINT
C     IT WAS WHICH WAS PICKED
      CALL CHNPIK(WKID,PIKID,PTARR)

C     SEE IF THE LAST POINT PICKED WASN'T THE SAME AS THE FIRST
      IF (NPTCHD .GT. 2) THEN
         IF(HOLDAR(NPTCHD) .EQ. HOLDAR(1)) THEN
```

```
C     CALCULATE THE AREA OF THE INTERSECTION
            CALL CALCAR(PTARR,A)

C     WRITE RESULTING AREA TO SCREEN
            WRITE(STRING,20)'CHAIN OPERATION SUCCESSFUL, AREA IS ',A,
     $                      ' SQ FT'
   20       FORMAT(A36,F10.4,A6)
            CALL MSG(WKID,STRING)

         END IF

      END IF

      RETURN
      END
C=====================================================================
C                        M O D U L E  C H N P I K
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL CHNPIK(WKID,PIKID,PTARR)
C=====================================================================
C INPUT PARAMETERS:
C     WKID - INTEGER, WORKSTATION IDENTIFIER
C     PIKID - INTEGER, ID OF PICKED POINT
C     PTARR - ARRAY CONTAINING INTERSECTION POINTS
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     HOLDAR - ARRAY WHICH ACTS AS A POINTER INTO INTERSECT ARRAY
C     NPTCHD - INTEGER, NUMBER OF POINTS IN THE CHAIN TO DATE
C     FLAG - DETERMINES WHETHER TO INCREMENT OR DECREMENT POINT
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     DETERMINES WHETHER TO INCR OR DECR A POINT AND CALCS AREA WHEN DON
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     YINTPT- DRAWS LINE FROM POINT TO POINT OR ERASES IT AS NEEDED
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
```

```
      SUBROUTINE CHNPIK(WKID,PIKID,PTARR)
      INTEGER PIKID,YELLOW,WKID,FLAG, DECR, INCR
      REAL PTARR(3,*)
      PARAMETER (YELLOW = 15, DECR = 0, INCR = 1)

      COMMON /CHNDAT/ NPTCHD, HOLDAR
      INTEGER NPTCHD, HOLDAR(900)

C     INCREMENT THE NUMBER OF PICKED POINTS BY ONE
      NPTCHD = NPTCHD + 1

C     ADD THE PICKED POINT TO THE HOLD ARRAY
          HOLDAR(NPTCHD) = PIKID

C     IF NOT THE FIRST POINT, CHECK TO SEE IF IT IS THE SAME
C     AS THE PREVIOUS POINT
C     CHECK IT
          IF((NPTCHD .GT. 2) .AND.
     $    (HOLDAR(NPTCHD) .EQ. HOLDAR(NPTCHD-2))) THEN

C     IF SAME AS PREVIOUS POINT, THROW THEM BOTH OUT AND DECREMENT
C     THE HOLD ARRAY
              FLAG = DECR

C     CHANGE THE POINTS COLOR BACK TO RED
              CALL YINTPT(PIKID,PTARR,FLAG)

C     SUBTRACT 2 FROM NUMBER OF CHAINED POINTS
              NPTCHD = NPTCHD - 2

C      ELSE IF THE SAME LINE HAS NOT BEEN CHOSEN
       ELSE

C      REDRAW PICKED POINT IN YELLOW
          FLAG = INCR
          CALL YINTPT(PIKID,PTARR,FLAG)


       END IF

       RETURN
       END
C=====================================================================
C                        M O D U L E   Y I N T P T
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C      CALL YINTPT(PIKID,PTARR,FLAG)
C=====================================================================
C INPUT PARAMETERS:
C      PIKID - INTEGER, ID OF PICKED POINT
C      PTARR - ARRAY CONTAINING INTERSECTION POINTS
C      FLAG - INTEGER, SET AS EITHER INCREMENT OR DECREMENT
C=====================================================================
C OUTPUT PARAMETERS:
C      NONE
C=====================================================================
C COMMON INPUTS:
C      NONE
C=====================================================================
C COMMON OUTPUTS:
```

```
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     STID - INTEGER STRUCTURE ID
C     HOLDAR - ARRAY WHICH ACTS AS A POINTER INTO INTERSECT ARRAY
C     NPTCHD - INTEGER NUMBER OF POINTS IN CHAIN
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     INCREMENTS OR DECREMENTS LINE AND POINT AS NECESSARY
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     GTIVWS - FINDS THE STRUCTURE ID FOR CURRENT VIEW
C     POPST - PHIGS OPEN STRUCTURE
C     PSEPLB - PHIGS SET POINTER AT LABEL
C     POSEP - PHIGS OFFSET POINTER
C     PTDEC - DECREMENTS POINT AND ERASES LINE
C     PTINCR - INCREMENTS POINT AND DRAWS LINE
C     PCLST- PHIGS CLOSE STRUCTURE
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE YINTPT(PIKID,PTARR,FLAG)
      INTEGER ITEM, NITEMS, STID, ERRIND, PIKID, YELLOW,FLAG
      INTEGER DECR,INCR, WHITE
      REAL PTARR(3,*)

      PARAMETER (ITEM = 4, NITEMS = 1, YELLOW = 15, DECR = 0, INCR = 1)
      PARAMETER (WHITE = 7)

      COMMON /CHNDAT/ NPTCHD, HOLDAR
      INTEGER NPTCHD, HOLDAR(900)


C     FIND THE STRUCTURE ID OF THE INTERSECTION POINTS
      CALL GTIVWS(ITEM, NITEMS, 11, STID, ERRIND)

C     REOPEN THE STRUCTURE AND DRAW THE POINTS OVER AGAIN
      CALL POPST(STID)

C     SET POINTER AT LABEL
      CALL PSEPLB(HOLDAR(NPTCHD))

C     OFFSET LABEL BY ONE
      CALL POSEP(1)

C     DELETE THE FOLLOWING ELEMENT WHICH SETS THE COLOR AS WHITE
      CALL PDEL

C     HIGHLIGHT THE PICKED POINT BY CHANGING THE COLOR TO YELLOW
C     IF INCR OR BACK TO WHITE IF DECR
      IF(FLAG .EQ. DECR) THEN
         CALL PTDEC(WHITE,PIKID)

      ELSE
         CALL PTINCR(YELLOW, PTARR)
```

```
      END IF

C     CLOSE STRUCTURE
      CALL PCLST

      RETURN
      END
C=======================================================================
C                        M O D U L E   P T D E C
C=======================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=======================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C     CALL PTDEC(WHITE,PIKID)
C=======================================================================
C INPUT PARAMETERS:
C     WHITE - INTEGER, THE COLOR WHITE
C     PIKID - INTEGER, ID OF PICKED POINT
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C LOCAL VARIABLES:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     DECREMENTS LINE AND POINT
C=======================================================================
C EXAMPLE(S):
C     N/A
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C MODULES CALLED:
C     PSPMCI - PHIGS SET POLYMARKER COLOR INDEX
C     PSEPLB- PHIGS SET POINTER AT LABEL
C     PDEL - PHIGS DELETE ELEMENT
C=======================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=======================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=======================================================================
      SUBROUTINE PTDEC(WHITE,PIKID)
      INTEGER WHITE, PIKID

C      RESET POINT TO WHITE
       CALL PSPMCI(WHITE)

C      SET AT LABEL
       CALL PSEPLB(PIKID + 1)
```

```
C       OFFSET LABEL BY ONE
        CALL POSEP(-2)

C       DELETE THE FOLLOWING ELEMENT WHICH IS THE POLYLINE
        CALL PDEL

        RETURN
        END
C=====================================================================
C                        M O D U L E   P T I N C R
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL PTINCR(YELLOW,PTARR)
C=====================================================================
C INPUT PARAMETERS:
C     YELLOW- INTEGER, THE COLOR YELLOW
C     PTARR - ARRAY CONTAINING THE INTERSECTION POINTS
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     X,Y,Z - COORDS OF 3D LINE
C     HOLDAR - ARRAY WHICH ACTS AS POINTER INTO INTERSECTION ARRAY
C     NPTCHD - INTEGER, NUMBER OF POINTS IN CHAIN
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     INCREMENTS LINE AND POINT
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     PSPMCI - PHIGS SET POLYMARKER COLOR INDEX
C     PSEPLB- PHIGS SET POINTER AT LABEL
C     POSEP- PHIGS OFFSET POINTER
C     PPL3 - PHIGS 3D LINE
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE PTINCR(YELLOW,PTARR)
      INTEGER YELLOW
      REAL X(2), Y(2), Z(2), PTARR(3,*)
```

```
      COMMON /CHNDAT/ NPTCHD, HOLDAR
      INTEGER NPTCHD, HOLDAR(900)

C     SET COLOR TO YELLOW AND CALCULATE LINE END POINTS
       CALL PSPMCI(YELLOW)
       IF (NPTCHD .GT. 1) THEN

          X(1) = PTARR(1,HOLDAR(NPTCHD-1))
          X(2) = PTARR(1,HOLDAR(NPTCHD))

          Y(1) = PTARR(2,HOLDAR(NPTCHD-1))
          Y(2) = PTARR(2,HOLDAR(NPTCHD))

          Z(1) = PTARR(3,HOLDAR(NPTCHD-1))
          Z(2) = PTARR(3,HOLDAR(NPTCHD))

C      SET POINTER AT LABEL
       CALL PSEPLB(HOLDAR(NPTCHD - 1) + 1)

C      OFFSET LABEL BY ONE
       CALL POSEP(-2)

          CALL PPL3 (2, X, Y, Z)

       ENDIF

       RETURN
       END
C=====================================================================
C                        M O D U L E  C A L C A R
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:  AMANU
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL CALCAR(INTARR,A)
C=====================================================================
C INPUT PARAMETERS:
C     INTARR - ARRAY CONTAINING THE INTERSECTION POINTS
C=====================================================================
C OUTPUT PARAMETERS:
C     A - REAL, AREA
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     X,Y - COORDS FED TO AREA ROUTINE
C     HOLDAR - ARRAY WHICH ACTS AS POINTER INTO INTERSECTION ARRAY
C     NPTCHD - INTEGER, NUMBER OF POINTS IN CHAIN
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     CALCULATES AREA OF ENCLOSED REGION
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
```

```
C MODULES CALLED:
C     AREA - CALCULATES THE AREA OF AN N-SIDED POLYGON
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (15:26:51)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE CALCAR(INTARR,A)
      REAL INTARR(3,*),A,XY(2,900)
      COMMON /CHNDAT/ NPTCHD, HOLDAR
      INTEGER NPTCHD, HOLDAR(900)

C     IF THE INTERSECTION PLANE IS VERTICAL THEN
C     IF(IVAL(2) .EQ. 0.) THEN

C     FEED THE POINTS TO A ROUTINE WHICH WILL FIND THE AREA
      DO 10 I = 1,NPTCHD
         XY(1,I) = INTARR(2,HOLDAR(I))
         XY(2,I) = INTARR(3,HOLDAR(I))
   10 CONTINUE

C     CALCULATE THE AREA FORMED DURING THE CHAINING ROUTINE
      CALL AREA(NPTCHD,XY,A)

      RETURN
      END
C=====================================================================
C                       M O D U L E  A R S C H
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:  VT-A___
C MODULE #:  _
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL ARSCH (WKID,PTARR)
C=====================================================================
C INPUT PARAMETERS:
C     WKID - INTEGER, WORKSTATION IDENTIFIER
C     PTARR - ARRAY CONTAINING POINTS OF INTERSECTION
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     ALLOWS USER TO RESTART THE CHAIN OF INTERSECTION POINTS
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
```

```
C=====================================================================
C MODULES CALLED:
C     GTIVWS - GETS THE STRUCTURE ID FOR VIEW CONTAINING INT. POINTS
C     PEMST - PHIGS EMPTY STRUCTURE
C     PUTINT - PUTS INTERSECTION POINTS INTO A STRUCTURE
C     MSG - WRITES A MESSAGE
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (10:29:17)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================

      SUBROUTINE ARSCH(WKID,PTARR)
      INTEGER WKID,WHITE, ITEM, NITEMS, ERRIND, STID
      REAL PTARR(3,*)
      PARAMETER (WHITE = 7, ITEM = 4, NITEMS = 1)

      COMMON /CHNDAT/ NPTCHD,HOLDAR
      INTEGER NPTCHD, HOLDAR(900)

C     FIND THE STRUCTURE WHICH CONTAINS THE INTERSECTION POINTS
      CALL GTIVWS(ITEM, NITEMS, 11, STID, ERRIND)

C     EMPTY THE STRUCTURE
      CALL PEMST (STID)

C     RECREATE THE POINTS
      CALL PUTINT ( WKID, PTARR)


C     RE-INITIALIZE NUMBER OF CHAINED POINTS AS ZERO
      NPTCHD = 0

C     WRITE MSG TO RESTART CHAINING
      CALL MSG(WKID, 'PLEASE BEGIN CHAIN AGAIN')

      RETURN
      END
C=====================================================================
C                      M O D U L E  P L D E F
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL PLDEF(WKID,XLOC,YROT)
C=====================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C     XLOC   - REAL, LOCATION OF INTERSECTION PLANE ALONG X AXIS
C     YROT   - REAL, ROTATION OF INTERSECTION PLANE WRT Y AXIS
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
```

```
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     NVW - INTEGER, NUMBER OF VIEWS CURRENTLY SHOWN
C     ITEM - WHAT TYPE OF STRUCTURE ID TO USE
C     NITEMS - NUMBER OF ITEMS TO RETURN
C     STID - STRUCTURE ID
C     ERRIND - ERROR INDICATOR
C     VIEW - VIEW ID FOR STRUCTURE OF POINTS
C     VIEW2 - VIEW ID FOR BLACK BACKGROUND
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     DEFINES INTERSECTION PLANE
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     GTMVWS - GETS NUMBER OF VIEWS
C     GTIVWS - GETS STRUCTURE IDS FOR THOSE VIEWS
C     POPST - PHIGS OPEN STRUCTURE
C     PLXFRM - CHANGE THE XFORM MATRICES OF THE INTERSECTION PLANE
C     PCLST - PHIGS CLOSE STRUCTURE
C     UPPLVW - UPDATES THE VIEW CONTAINING THE INTERSECTION DATA PTS
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (09:19:01)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
      SUBROUTINE PLDEF(WKID,XLOC,YROT)
      INTEGER WKID,NVW,ITEM, NITEMS, STID,ERRIND,VIEW, VIEW2
      REAL XLOC,YROT
      PARAMETER (ITEM = 4, NITEMS = 1, VIEW = 11, VIEW2 = 12)

C     GET NUMBER OF VIEWS
      CALL GTNVWS(NVW,ERRIND)

C     LOOP THRU VIEWS
      DO 10 I = 2,NVW + 1

C     GET THE STRUCTURE IDS
         CALL GTIVWS(ITEM,NITEMS,I,STID,ERRIND)

C     OPEN PLANE STRUCTURE
         CALL POPST(STID)

C     CHANGE THE PLANE XFORMS
         CALL PLXFRM(WKID,XLOC,YROT)

C     CLOSE THE STRUCTURE
         CALL PCLST

   10 CONTINUE

C     UPDATE THE XSEC VIEW
      CALL UPPLVW(WKID,VIEW, VIEW2, XLOC,YROT)


      RETURN
```

```
      END
C=======================================================================
C                        M O D U L E   P L X F R M
C=======================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:
C=======================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C     CALL PLXFRM(WKID, XLOC, YROT)
C=======================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C     XLOC   - REAL, LOCATION OF INTERSECTION PLANE ALONG X AXIS
C     YROT   - REAL, ROTATION OF INTERSECTION PLANE WRT Y AXIS
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C LOCAL VARIABLES:
C     XFORM - TRANSLATION MATRIX
C     YXFORM - Y ROTATION MATRIX
C     FFORM - RESULTING XFORM MATRIX
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     REDEFINES THE INTERSECTION PLANE XFORM MATRICES
C=======================================================================
C EXAMPLE(S):
C     N/A
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C MODULES CALLED:
C     PDELLB - PHIGS DELETE LABEL
C     PTR3 - PHIGS XFORM MATRIX FOR TRANSLATION
C     PROY - PHIGS XFORM MATRIX FOR Y ROTATION
C     PCOM3 - PHIGS CONCATENATE XFORM MATRICES
C     PSLMT3 - PHIGS SET LOCAL XFORM MATRIX
C=======================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (09:19:01)
C=======================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=======================================================================
      SUBROUTINE PLXFRM(WKID, XLOC, YROT)
      INTEGER WKID
      REAL XLOC,YROT
      REAL XFORM(4,4),YXFORM(4,4),FFORM(4,4)
      REAL PI
      PARAMETER (PI = 3.1415927)


C     DELETE STUFF BETWEEN LABELS 1 AND 2
      CALL PDELLB(1,2)
```

```
C     NOW ADD IN THE NEW XFORMATION MATRICES
      CALL PTR3(XLOC,0.,0.,XFORM)

      YROT = YROT*PI/180.

C     ROTATION
      CALL PROY(YROT,YXFORM)

C     ADD EM
      CALL PCOM3(XFORM,YXFORM,FFORM)

C     SET LOCAL XFORM MATRIX
      CALL PSLMT3(FFORM,2)

      YROT = YROT *180./PI

      RETURN
      END
C=====================================================================
C                      M O D U L E  U P P L V W
C=====================================================================
C  PROJECT:  NASA/AMES
C  VERSION:  X1.0.0
C   PART #:
C MODULE #:
C=====================================================================
C COMPILER:  FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C     CALL UPPLVW(WKID,VIEW,VIEW2, XLOC,YROT)
C=====================================================================
C INPUT PARAMETERS:
C     WKID   - INTEGER, WORKSTATION IDENTIFIER
C     VIEW   - VIEW WHICH CONTAINS THE STRUCTURE
C     VIEW2  - VIEW WHICH CONTAINS THE BACKGROUND
C     XLOC   - REAL, LOCATION OF INTERSECTION PLANE ALONG X AXIS
C     YROT   - REAL, ROTATION OF INTERSECTION PLANE WRT Y AXIS
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C LOCAL VARIABLES:
C     PWID - REAL, DEFINES HALF THE WIDTH OF THE INTERSECTION PLANE
C     PLEN - REAL, DEFINES HALF THE LENGTH OF THE INTERSECTION PLANE
C     B AND C - VECTORS WHICH DEFINE THE INTERSECTION PLANE
C     NORM - REAL, ARRAY WHICH CONTAINS THE VECTOR NORMAL TO INT PLANE
C     LOC - REAL, ARRAY CONTAINS THE LOCATION OF THE CENTER OF PLANE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     UPDATES THE VIEW WHICH CONTAINS THE INTERSECTION PLANE
C=====================================================================
C EXAMPLE(S):
C     N/A
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C MODULES CALLED:
C     CROSS - CALCULATES CROSS PRODUCT OF TWO VECTORS
C     STNMVW - SETS THE NORMAL VECTOR OF THE VIEW
```

```
C     CLIPVW - SETS UP THE CLIPPING PLANES OF THE VIEW
C     ST3DVW - SET UP A 3D VIEW
C=====================================================================
C CODED BY:  MICHELE GRIESHABER
C     DATE:  06/16/88 (09:19:01)
C=====================================================================
C DEVELOPMENT SITE:
C    THE MECHANICAL ENGINEERING DEPARTMENT
C    VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY
C    BLACKSBURG, VA  24061
C=====================================================================
C=====================================================================
C FUNCTIONAL DESCRIPTION - DUMMY STUB FOR UPDATING THE XSEC VIEW
C=====================================================================
      SUBROUTINE UPPLVW(WKID,VIEW,VIEW2, XLOC,YROT)
      INTEGER WKID,VIEW, VIEW2
      REAL PWID,PLEN,XLOC,YROT
      PARAMETER(PWID = 50., PLEN = 50.)
      REAL B(3),C(3),NORM(3),LOC(3)

C     CALCULATE THE NORMAL VECTOR OF THE INTERSECTION PLANE
C     BUT FIRST DEFINE TWO VECTORS IN THE PLANE
      B(1) = 0.
      B(2) = -PWID
      B(3) = 0.

      C(1) = 0.
      C(2) = 0.
      C(3) = PLEN/COS(YROT * 3.141579/180.)

C     FIND THE VECTOR NORMAL TO THE PLANE
      CALL CROSS(B,C,NORM)

C     DEFINE THE CENTER LOCATION OF THE PLANE
      LOC(1) = XLOC
      LOC(2) = 0.
      LOC(3) = 0.

C     SET THE NORMAL OF THE VIEW
      CALL STNMVW(VIEW,LOC,NORM)

C     SET UP CLIPPING PLANES
      CALL CLIPVW(VIEW)

C     SET 3 D VIEW BACKGROUND
      CALL ST3DVW(WKID,VIEW2)

C     SET 3D VIEW
      CALL ST3DVW(WKID,VIEW)

      RETURN
      END
```

VITA

Michele Grieshaber was born in State College, Pennsylvania, in 1965. As a child she was a complete bully. No kidding. In 1983 she came to Virginia Polytechnic Institute and State University to study engineering. During her senior year at VPI she was one of four students chosen to participate in the first exchange between VPI and the University of Technology of Compiegne, France. After her graduation, she remained at VPI to pursue a Master's Degree in Mechanical Engineering. Next year she plans to work with engineers from Dassault, a French aerospace company, while studying at Ecole Centrale des Arts et Manufactures, Paris, as an ITT Fellow.

Michele Marie Grieshaber

Michele Marie Grieshaber