Interactive design and NC machining of free form surfaces /

Gregory Charles Loney
Lehigh University

INTERACTIVE DESIGN AND NC MACHINING OF FREE FORM SURFACES

BY

GREGORY CHARLES LONEY

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Mechanical Engineering

Lehigh University

1985

CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering.

12/09/85
Date

Tulga Özsoy
Professor in Charge

F. Erdogan
Chairman of Department

ACKNOWLEDGEMENTS

I wish to thank Dr. Tulga Ozsoy for unending support and guidance during the course of the '3D Space Curves for NC Machining' project. He contributed the graphics package among other routines running CISPA and most importantly provided invaluable technical consultation aiding in the completion of this phase of the software.

I would like to acknowledge the support of Bridgeport TEXTRON who provided the funds for this project and especially Marvin Kreithen, Peter Voytershark and Lena Fishman for their suggestions and encouragement.

I thank Mustafa Kayrak for his contributions to CISPA's error handling, entity codes, and entity name recoginition. I would also like to thank Rich Smith for his work in documentation and test, and David Tress for his efforts in porting CISPA to the TEKTRONIX[1] 4107.

I would also like to thank my family for their encouragement and prayer.

Finally, thanks to my friends Sia, Jack, Drew, and Joan for making graduate life such a pleasure.

1. TEKTRONIX is a trademark of Tektronix, Inc.

TABLE OF CONTENTS

TABLE OF CONTENTS continued

LIST OF FIGURES

LIST OF FIGURES continued

ABSTRACT

Advances in computer technology, such as computational speed and display processing has allowed computer-aided design/computer-aided manufacturing (CAD/CAM) to move into manufacturing industry at even greater rates than expected. As this process takes place more emphasis on user-friendly, interactive, design intuitive systems will be necessary, taking operation of such systems, from a complicated expert level to a level that is self guided and forgiving of a untrained user.

CISPA, Computer Interactive Surfaces Pre-APT, was designed with the above ideas in mind. This system utilizes a menu-driven front-end with graphical feedback to guide a user through curve and free form surface definition resulting in a mathematical model which may be used to generate NC (numerically controlled) machine paths for milling of 3D shapes. This front end was built upon routines taken from the execution phase of APT 4, a processor designed to handle free form curves and surfaces.

Improvements were made in the definition and calculation of step size basing the input on geometric attributes rather than abstract algebraic quantities. Routines were also added to supplement error handling in the APT complex and increase flexibility in milling surfaces, by roughing-to-depth.

1. INTRODUCTION

1.1 BACKGROUND

Computer-Aided Design/Computer-Aided Manufacturing (CAD/CAM) (1,2) has long been extolled as a boom to design and manufacturing productivity. Computer graphics help designers visualize their creations without the expense of prototype development. Since most engineering drawing utilizes analytical functions such as lines, circles, ellipses, etc. the rendering of 3-D objects in a 3-D wire frame format is not a difficult task in computer graphics, however free-form surfaces, by definition cannot be described be a single mathematical relationship. These surfaces appear in a great many applications in the aerospace, automobile and shipbuilding industries. With the advent of automated assembly precluding the use of fewer and fewer fastening elements more complicated parts (probably plastic) will appear utilizing free form surfaces.

At the present the greatest need in surface definition seems to be in the numerically controlled (NC) machining (3) of complicated parts, although it is easy to imagine the need for free form surfaces in robotics (painting and welding applications), art and in the visualization of large amounts of discrete data generated by today's finite elements code. The manufacturing of parts with blended surfaces means the creation of a mold or die. For example, in the production of an investment casting this usually means hand sculpting the shape out of wax. Clearly if there is a need for tolerance control or design iteration this hand sculpting method becomes unacceptable. Furthermore, traditional ways of representing surfaces using graphical methods such as lofting, or multiple orthogonal projections, have great drawbacks in the effectiveness of visualization and become inefficient as complexity increases.

With the advent of the computer, computational methods were devised dividing the surface into an assembly of curvilinear quadrilateral patches, each patch mathematically defined by one formula. One of the earliest contributions to surface definition was made by Ferguson (1963) (4) who described the patch system in terms of parametric rather than cartesian coordinates. This coordinate definition is now in standard use for the following reasons: 1) it enables twisted space curves in three dimensions to have a simple mathematical representation, 2) it avoids problems which arise when representing curves with vertical tangents in a fixed coordinate system, 3) it enables coordinate transformations such as translation, rotation, or perspective to be performed and 4) parametric forms of curves and surfaces are easily digitized for display. In short, by using parametric forms independance is gained from a particular system of coordinates.

A general theory of surface patches was described by Coons (1964) (5) which showed how four boundary curves can be blended into a smooth patch using general blending functions allowing any order of continuity between patches. The properties of spline curves were investigated and found to be useful in representing smooth curves and surfaces. Derived from the physical spline, a long narrow strip of wood used by a loftsman to fair in curves between data points, a spline curve is represented by an nth-order polynomial. B-splines suggested by Schoenberg (1946) (6) and later applied to curve and surface definition by Reisenfeld(1974) were found to be useful for their properties of being locally non-zero.

The introduction of the control polygon by Bezier (1971) (7) helped bring designers the intuition needed in the efficient control of surface geometry. The input to Bezier's system is an open polygon made up of straight lines. The vertices of this polygon are used in obtaining the curve definition. The resulting curve is a loose approximation of the input polygon. Modifying the polygon modifies the curve in a predictable way. Surfaces are defined in a similar way by inputing open polyhedrons.

Gordon and Reisenfield (1974) (8) made use of the control polygon in their work with B-splines exploiting the non-global behavior of the B-spline basis. Multiple coincident vertices are used to pull the curve closer to the control polygon, while increasing the order of the spline polynomial tends to pull the curve away from the polygon. Local control is added by changing vertex location. This concept is extended to B-spline surfaces and prove to be useful as an interactive tool for surface definition.

All of the preceding falls under a rather new field called computational geometry (9). Computation not done by hand but by computer. Surface geometry, such as surface normal, curvature, etc. are easily calculated once a mathematical description is found.

The final result is usually a large array of points which are connected together in a wire frame representation. To aid in the visualization of the surface, computer graphics are implemented and hidden line removal, coloring of critical areas based on extreme curvature, surface texturing, perspective and view control may be invoked (10).

1.2 PROBLEM STATEMENT

With the decline in expense of computer hardware and improvements in speed and storage of such machines, computer-aided manufacturing can be introduced to virtually any size company as long as the need exists. The real problem in the utilization of such technology seems to be in the price, support and friendliness of the software running the CAM system. Specifically in NC machining applications a file is usually submitted to a post processor in APT (Automatically Programmed Tool) or like language, errors may be flagged if the language is not used properly, however an improper cutting cycle may be introduced, not detected until NC machining has started. Ideally, a part may be represented by a 3D graphics display alerting the user to impossible geometry or undesirable design characteristics. If such a system is used in design with iteration through a series of configurations, an iterative means of input with real time results displayed becomes necessary. In summation, it seems that there are currently three distinct levels of input used in geometry and NC machining parameters: 1) language input, 2) language input with graphical verification, 3) highly interactive input with graphical verification (usually menu-driven).

As mentioned earlier the need for representing sculptured surfaces for machining, robotic manipulator trajectories, etc. has generated a good deal of research with McAuto's UNIGRAPHICS II (TM), Dassault System's CATIA (TM), Applicon's APPLICON (TM) being notable examples of working interactive systems.

The problem initially brought to Lehigh University's CAD lab by Bridgeport-TEXTRON Controls Division was rather vaguely put as 'the development of methods for implementing NC milling strategies for 3D free form curves and surfaces'. This request was more concretely interpreted as : develop a system where by 3D space curves and surfaces may be created, and later used to define NC paths. This system should be inexpensive, yet reliable, easy to use and run on existing hardware contained in Lehigh's CAD lab, namely a DEC VAX 11/780 mini-computer and DEC VS11 graphics terminal. Future developments would find this software ported to a micro-computer aiding in portablity throughout a plant and further decrease hardware costs. In addition the first prototype should be running in a period of time of one to two years.

1.3 APPROACH TO PROBLEM

Creating original software from scratch allowing definition of the myriad of shapes used in NC machining proved to be a position of last resort. Instead a survey was started of available sculptured surface software contained in the public domain.

Computer-Aided Manufacturing-International, Inc. (CAM-I) was chosen as a likely source for code that might be used in part or in whole. CAM-I is a non-profit organization formed in 1972 to further cooperative research and development efforts of companies with common interests in computer-aided manufacturing. The membership encompasses government, industrial and educational organizations who jointly contribute to the improved productivity of industry by identifying need, creating specifications and evaluating solutions through prototype development. Sculptured surfaces research by CAM-I has progressed through several different stages culminating in CAM-I's newest release, Sculptured Surfaces SSR1. SSR1 was rejected in favor of a more compact processor called CASPA03 (Computer-Aided Sculptured Pre-APT, Version 3) because of SSR1's size, cost and hardware requirements. CASPA supports many of the surfaces contained in SSR1 and seemed to be of a more manageable size important if the software would later be run on a micro-computer. The CASPA system also provides an integrated graphics control permitting immediate plotting of sculptured geometry and CASPA generated cutter positions, from any desired viewpoint. Input is in a fixed format using keywords. These keywords are matched to a symbol table to determine appropriate action. This language oriented input was viewed as slow and error prone and would later be replaced with an interactive front end.

The problem approach would comprise of buying inexpensive code, namely CASPA (cost, $500.00) from which certain routines would be used in whole or modified in part to work with supplementary portions developed to add flexibility, speed and more user-friendliness creating a new sculptured surfaces definition package. These new features would be based on an interchange of suggestions and ideas leading from quarterly meetings and progress reports with Bridgeport-TEXTRON.

1.4 ORGANIZATION OF THESIS

Contained in this thesis is a thorough description of the prototype software called CISPA (Computer-aided Interactive Surfaces Pre-APT) including algorithmic discussion, database organization and program operation.

Chapter 2 discusses project history, followed by Chapter 3 which features an explanation of the mathematics of surfaces and synthetic curves used in CASPA. Newly developed algorithms achieving step size based on surface tolerance, rough-to-depth cutting and picking entities with general flow charts included, are discussed in Chapter 4. Organization of the CISPA data structure ends the chapter.

Chapter 5 presents program operation and program structure information: the type of hardware required to run CISPA, the software: graphics driver, APT routines under CASPA, etc., and description of the major modules. A summary and conclusion is included in Chapter 6. A case study, blank common data structure and menu flowchart with brief discriptions are contained in Appendices A, B, and C. For further study a users manual (11) with subroutine documentation and detailed explanation of menu options has been compiled separate from this thesis.

2. PROJECT HISTORY

Research into sculptured surfaces started by appraising the public domain for inexpensive software, thus saving time and duplication of work. It was discovered that CAM-I had a very powerful sculptured surfaces processor called SSR1 (SSR2 release followed), upon further investigation it was found that about ten years worth of effort had produced other software embodied in SSR1. The first prototype, a complete experimental APT 4 based system, produced was SSX1 (Sculptured Surfaces Experimental 1) in 1970. SSX2 (12) followed in the same fashion allowing milling of patches based on a Coons canonical form and definition of surfaces using a mesh of points. Progress continued based on SSX2, through SSX6, SSX6A, and SSX7 (13) to SSX8X (14), where by now, synthetic curves utilizing a combine feature allowing spline and conics to be combined together under one canonical form, surfaces of revolution, cylindrical surfaces, surfaces through general curves, full transform capability of sculptured geometry defined by forming various cross products of synthetic curves, and regional milling features had been added to SSX2. CASPA03 (15) was developed to provide a vehicle for evaluating new sculptured surface capabilities while supporting all of the sculptured surface definitions of the APT-based SSX6A system. More extensions were added to the SSX8X software and appear as SSR1 these include routines for extracting parametric coordinate values from the last extrinsic geometric construction with a curve or surface, provision of the ability to extract values out of the canonical form of geometric entities, and incorporation of a number of arithmetic subroutines, which can also be nested, for use in part programs where a numerical value is allowed. The APT 4 source code is required for all of these systems except CASPA.

From evaluating several of these systems it was determined that CASPA should be purchased for the project for the following reasons 1) compact size (~10,500 lines, 90 subroutines), 2) written in Fortran IV, 3) supported a large array of surface types and 4) could be used as a pre-processor to the APT 4 system (see reference 15 and Figure 2-1). SSR1 is a very powerful system, but needed to be run on an IBM 3033 machine with a FORTRAN H extended compiler none of which was at our disposal. CASPA seemed an good start for the prototype interactive sculptured surfaces processor.

Since the hardware requirements of the project were to run the system on available hardware (DEC VAX 11/780 and DEC VS11 terminal) inquiries were made into whether a VAX runnable CASPA was on the market. Software found available, however, was being privately marketed and was too expensive to merit the purchase.

The conversion of CASPA to the VAX required a change from IBM's representation of character data as EBCDIC code to ASCII code. This can be done by reading each EBCDIC character finding the corresponding

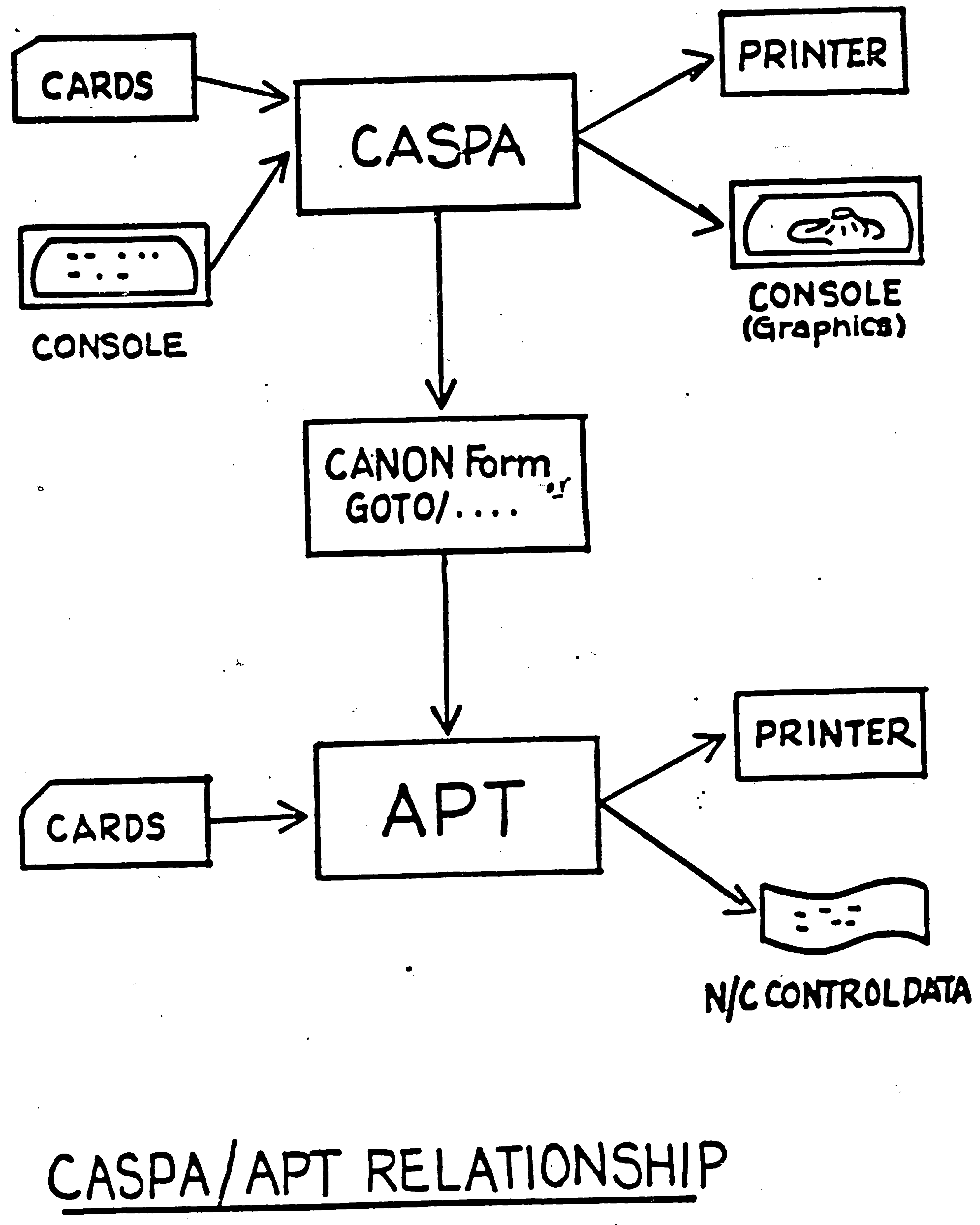


FIGURE 2-1. CASPA/APT RELATIONSHIP (REFERENCE 15)

ASCII code for that character, and writing that ASCII code to a new file.

Computer dependance as discussed in the CASPA Users Manual(15) is broken down into four catagories:

-- computer dependance in FORTRAN routines,

-- assembly language routines,

-- APT 4 compatability,

-- graphics compatability.

The CASPA system is made up of about 90 FORTRAN subroutines and four assembly language routines. The total source consists of about 10,500 lines. The CASPA language interpreter and graphics routines occupy about 25% of the code, these routines are written in single precision FORTRAN IV and presented no problem in transfer to the VAX. The assembly routines were re-written in VAX FORTRAN using the VAX dependant ENCODE statement, these routines convert floating point and integer values to character format.

APT 4 compatability is achieved in CASPA by interface routines: TSCURV, TSSURF, DSCURV, DSSURF and SPUNCH they convert between CASPA's single precision input to APT 4's required double precision storage.

The APT sculptured surface definition routines are led by APT107 (surface processor) and APT108 (synthetic curve processor). These routines proved to be portable to the VAX, but it should be noted that they are written in double precision.

CASPA utilizes CALCOMP graphics routines for display. These routines were replaced by GRAPH3D (16) (formerly VS113D) graphics package. GRAPH3D, created at Lehigh's CAD lab, allows drawing, display of alphanumeric data, model orientation routines and display control essential for production of an interactive front-end.

Valid keywords for CASPA are stored in a symbol table and are initialized upon execution in the block data program NMBLK. Since the VAX differs from the IBM 360/370 machine in the way it converts character BCD (binary coded decimal) to integer values, the search routine SWORDS was changed to let the word search proceed correctly.

After getting CASPA running on the VAX test programs were written to check the systems performance. During this test phase two observations were made: one, certain kinds of surface definition did not work and would have to be de-bugged, two, CASPA would need a new front end to make part definition an interactive process, and additional improvements in error handling, surface tolerancing and provision for rough to depth cutting would have to be made.

De-bugging the code, proved to be a simple task occupying a month, considerably more time would be spent on the front end. It was noted that the CASPA interpreter and interface (to the APT complex) routines formed a front end and that if the data transfer through the interface routines could be understood the CASPA front end could be replaced by the planned interactive menu-driven front end.

Once this transfer mechanism was understood work proceeded chronologically as follows:

- menu-driven input, for creating points, by existing point, absolute coordinates and cursor position,
- menu-driven input for all synthetic curves supported by

SSX6,

- naming, modifying and selecting these synthetic curves by cursor,
- menu-driven input of all surfaces (except PNTVEC, BEZIER and PNTSON) available in SSX6,
- improved error handling and error flagging,
- automatic saving of input information for curves and points,
- retrieval of curves by name, deletion of curves and points by name or by cursor selection,
- cursor selection of entities from any view,
- NC milling module in menu driven form including machining parameters, ie. tool diameter, offset surface, etc.,
- improved step size calculation fixing step size as a function of chordal deviation (see figure 2-2) and fixing the number of zig-zag passes as a function of scallop height (see figure 2-3)

  giving more user control over surface tolerance,
- rough-to-depth cutting of surface allowing cutting to depth in incremental z layers, menu-driven input with system defaults (see figure 2-4),
- naming and saving of tool paths on external file in binary to be written to a CLSF when ready for post-processing,
- CISPA ported to a Tektronix 4107 graphics display and run on a Micro VAX I a much less expensive hardware configuration compared to the VAX 11/780 VS11 terminal set-up,

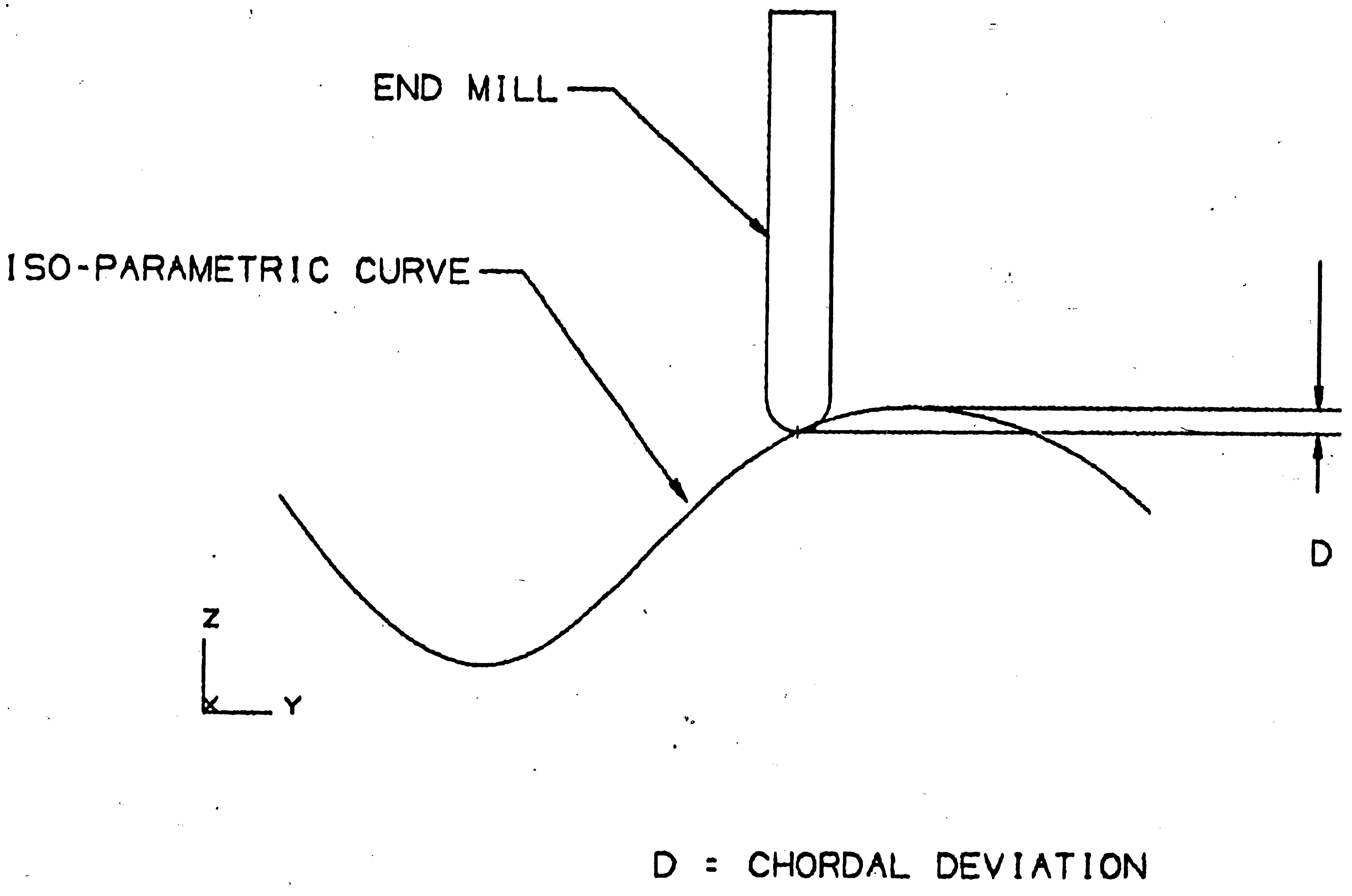


FIGURE 2-2. CHORDAL DEVIATION

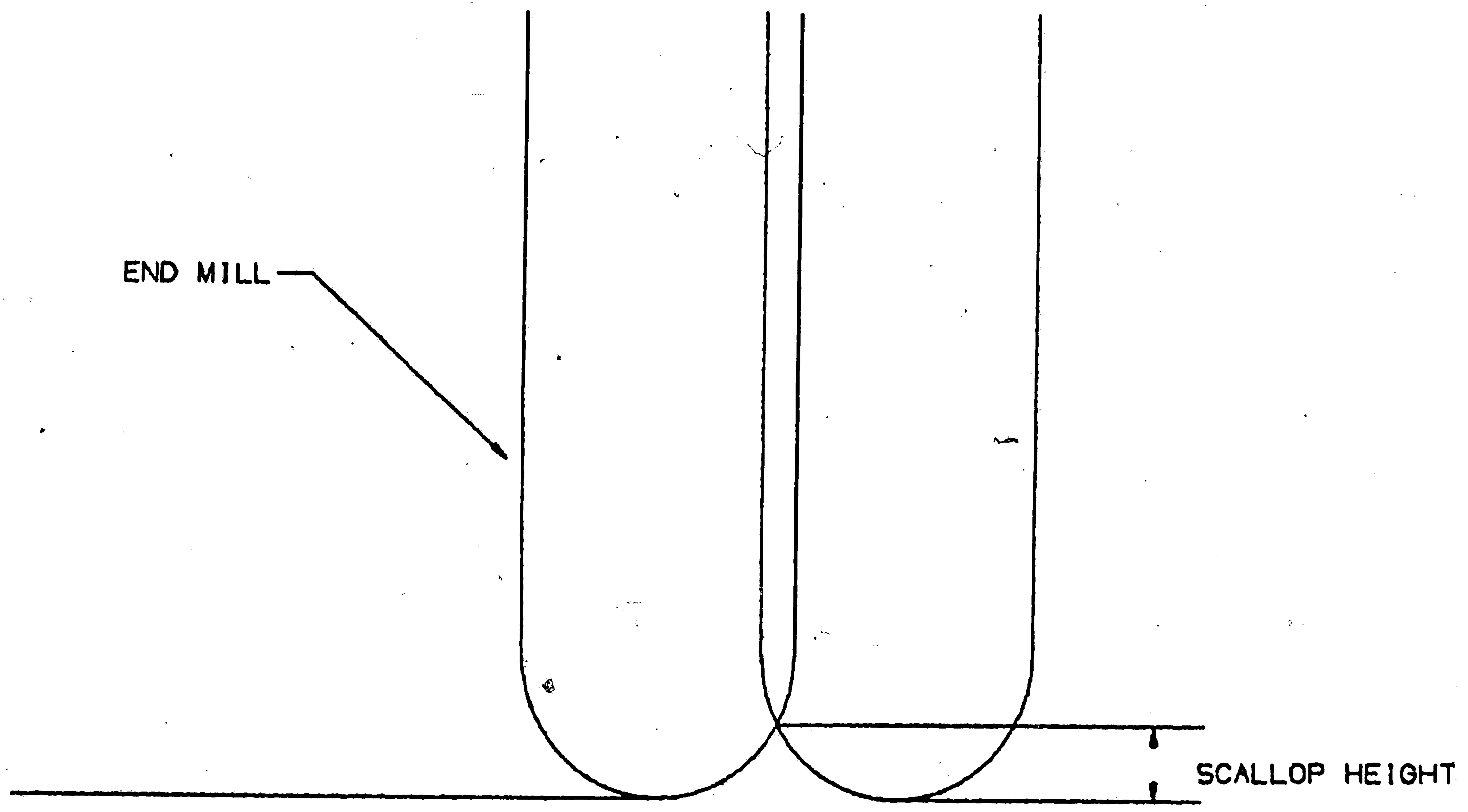


FIGURE 2-3. SCALLOP HEIGHT

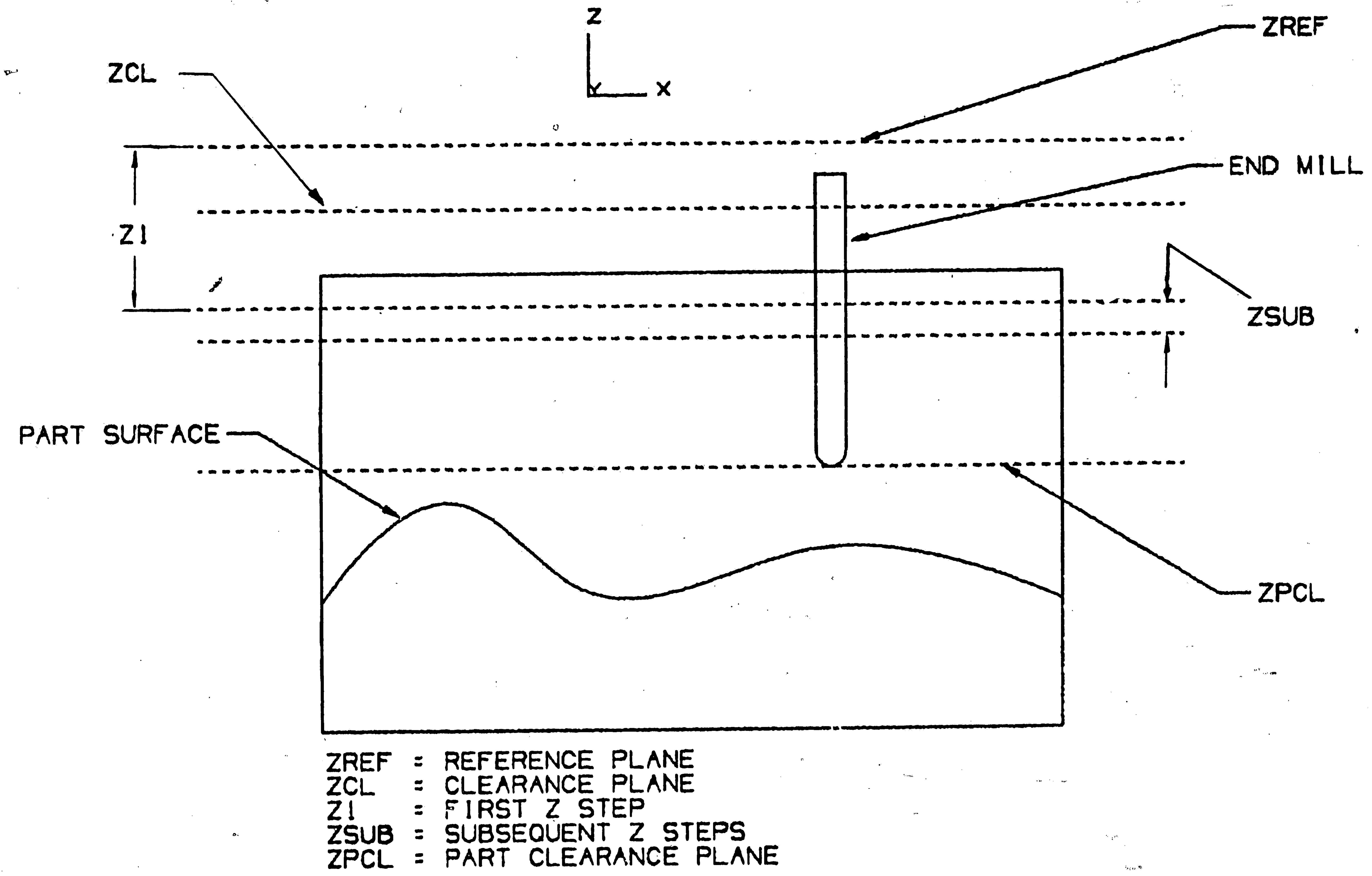


FIGURE 2-4. ROUGH-TO-DEPTH CUTTING

- production of a system users manual with subroutine descriptions (only for non-CASPA routines), program operation section, and two example tutorials.

The addition of the front end and additional features essentially eliminated CASPA creating a new package called CISPA for Computer-aided Interactive Surfaces Pre-APT system, although CASPA may still be used with the use of a software switch. The rest of this thesis will detail the CISPA design encompassed in the above list.

3. MATHEMATICS OF CURVES AND SURFACES

3.1 INTRODUCTION

The heirarchy of geometric elements contained in CISPA is seen as:

- points
- simple curves
- combined simple curves (synthetic curves)
- flow curves
- patches
- sculptured surfaces

That is, the design strategy proceeds by defining points, fitting curves to these points, if need be, combining these simple curves, arranging them in space and fitting a surface to this curve structure.

The underlying mathematics allowing definition of space forms follows with a discussion of rational parametric cubic equations representing conics, cubic polynomials and their use in representing 3D space curves. Surface mathematics are presented with discussion of bi-cubic patches. Some consideration of more efficient storage and simplified representation of the patch canonical form ends the chapter.

3.2 REPRESENTING CONIC SEGMENTS BY MEANS OF RATIONAL PARAMETRIC CUBIC EQUATIONS

The convention used in the manipulation of geometric elements in projective space rather than Euclidean space permeates most computational geometry (17). The justification for this representation stems from the fact that certain well-known geometric elements are more conveniently manipulated as projective elements. In Euclidean plane geometry a point is represented by a pair of coordinates as (x,y). In projective space the point receives one more coordinate, (x,y,w), this triplet is also called the homogeneous representation of a point. There is no unique homogeneous coordinate representation of a point in 2D space, that is (x,y,w) and (kx,ky,kw), $k \neq 0$ represent the same point. The point (x,y,1) or multiples thereof map to the original Euclidean space (x,y). The points (x,y,0) are called points at infinity. The point (0,0,0) has no meaning in projective space. The idea of projective representation is that parallel lines <u>do intersect</u> at the points at infinity. For example, consider parallel lines

$$y = x$$

$$y = x + 1 \quad ,$$

rewrite as

$$(x \ y \ 1) \begin{bmatrix} 1 & 1 & 0 \\ -1 & -1 & 0 \\ 0 & 1 & 1 \end{bmatrix} = (0 \ 0 \ 1)$$

The 3 x 3 matrix will be called **M** and is singular and thus non-invertible, re-write in alternate form as (see ref. 7)

$$x - y = 0$$
$$x - y + 1 = 0$$
$$x = x$$

or

$$(x\ y\ 1)\begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix} = (0\ 0\ x)$$

M is now invertible

$$(x\ y\ 1) = (0\ 0\ x)\begin{bmatrix} 0 & -1 & -1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$(x\ y\ 1) = (x\ x\ 0) = x(1\ 1\ 0).$$

The intersection point is (1 1 0), since x is some non-zero multiplicative constant.

The use of homogeneous coordinates give representation of points at infinity and allow the capability of generalized transformations, as will be shown next.

Projective transformation in a plane is shown by mutiplying a 3 x 3 matrix to the original point

$$(x^*\ y^*\ w^*) = (x\ y\ w)\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \qquad (3.1)$$

or

$$(x^* \ y^* \ w^*) = (x \ y \ w)\mathbf{A}$$

If **A** is non-singular, then **A** provides a one to one transformation of projective space to projective space. Note that the Euclidean rotation and translation matrix is a subset of the general matrix **A** where

$$\mathbf{A} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ x_0 & y_0 & 1 \end{bmatrix} \qquad (3.2)$$

Applying this matrix to a point rotates the point through θ and translates x_0 and y_0 units along the x and y axes respectively. The new point is expressed in the familiar form

$$\left(\frac{x^*}{w^*}\ \frac{y^*}{w^*}\ 1\right) = \left(\frac{x}{w}\cos\theta - \frac{y}{w}\sin\theta + x_0,\ \frac{x}{w}\sin\theta + \frac{y}{w}\cos\theta + y_0,\ 1\right) \qquad (3.3)$$

This transformation matrix is uniquely determined by the mapping of four points (P_1,P_2,P_3,P_4), the first three non-collinear and no two points identical into $(P_1^*,P_2^*,P_3^*,P_4^*)$.
Mapping the first three yields

$$(k_1x_1^* \ k_1y_1^* \ k_1w_1^*) = (x_1 \ y_1 \ w_1)\mathbf{A}$$
$$(k_2x_2^* \ k_2y_2^* \ k_2w_2^*) = (x_2 \ y_2 \ w_2)\mathbf{A}$$
$$(k_3x_3^* \ k_3y_3^* \ k_3w_3^*) = (x_3 \ y_3 \ w_3)\mathbf{A}$$

Solving for matrix **A**

$$\mathbf{A} = \mathbf{B}^{-1}\mathbf{Q}\mathbf{C} \qquad (3.4)$$

where

$$\mathbf{B} = \begin{bmatrix} x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \\ x_3 & y_3 & w_3 \end{bmatrix} , \qquad \mathbf{Q} = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{bmatrix}$$

and

$$\mathbf{C} = \begin{bmatrix} x_1^* & y_1^* & w_1^* \\ x_2^* & y_2^* & w_2^* \\ x_3^* & y_3^* & w_3^* \end{bmatrix} .$$

The matrix $\mathbf{A}$ is determined above except for the three constants in $\mathbf{Q}$, k_1, k_2, and k_3. Invoking the condition that the fourth point P_4 is mapped into P_4^* fully determines the matrix:

$$(x_4^* \; y_4^* \; w_4^*) = (x_4 \; y_4 \; w_4)\mathbf{A} \quad . \tag{3.5}$$

Substitute (3.4) into (3.5) and solve the three simultaneous equations for $k_{1,2,3}$, using these values in (3.4) yields transformation matrix $\mathbf{A}$.

The fact that $\mathbf{A}$ exists and is unique stems from the assumption that the first three points are not collinear making each of the matrices involved in the product in equation (3.4) non-singular.

The transformation matrix just discussed is useful in representing conics. For example, $p(u)=(u^2 \;\; u \;\; 1)$ where u is a real number, represents a parabola parametrically in a projective plane. Note points (fig. 3-1) p_0 at $u=0$, and p_1 at $u=1$ as well as the parametric mid-point p_m, $u=.5$ and p_t, the intersection of tangents at p_0 and p_1.

From projective geometry it is known that any conic can be represented as a projective transformation of the conic:

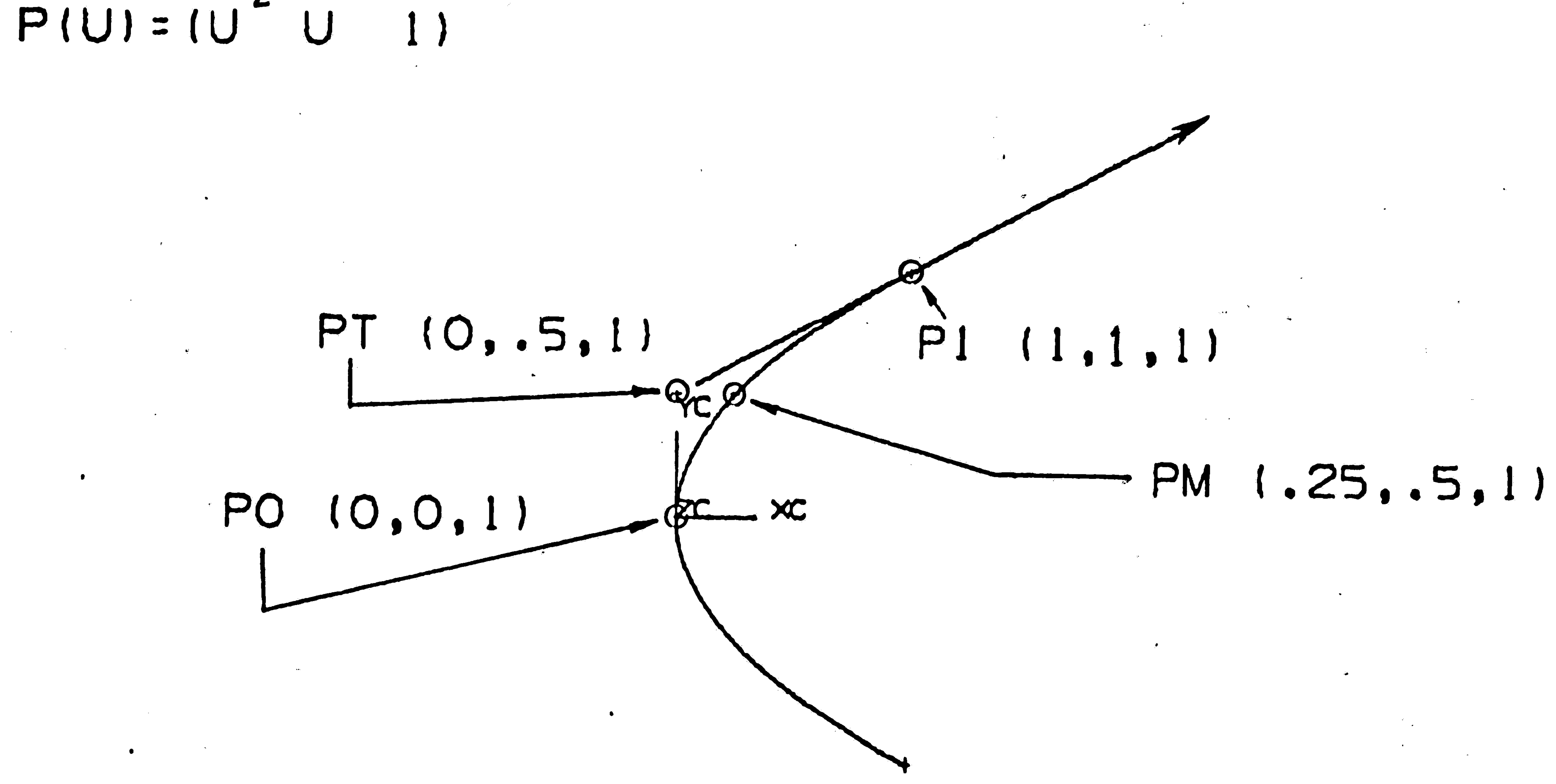


FIGURE 3-1. PARABOLA $p(u) = (u^2\ u\ 1)$

$$(u^2 \quad u \quad 1)^* = (u^2 \quad u \quad 1) \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad (3.6)$$

Conics, in general, can be represented by rational parametric forms. That is if

$$(x \ y \ w) = (u^2 \quad u \quad 1)\mathbf{A}$$

then

$$\frac{x}{w} = \frac{a_{11}u^2 + a_{21}u + a_{31}}{a_{13}u^2 + a_{23}u + a_{33}} \qquad (3.7)$$

and

$$\frac{y}{w} = \frac{a_{12}u^2 + a_{22}u + a_{23}}{a_{13}u^2 + a_{23}u + a_{33}} \qquad (3.8)$$

To determine the above coefficients so as to represent a conic, assume that p_0^*, p_1^*, p_m^* the parametric midpoint and intersection of tangents p_t are known for a give conic segment, the equation (3.4) becomes

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1/2 & 1 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{bmatrix} \begin{bmatrix} x_0^* & y_0^* & w_0^* \\ x_t^* & y_t^* & w_t^* \\ x_1^* & y_1^* & w_1^* \end{bmatrix} \qquad (3.9)$$

Solving for $k_{1,2,3}$ will map the simple parabola to the four known points.

Using the fourth constraint

$$(1/4 \quad 1/2 \quad 1)\mathbf{A} = (x_m^* \quad y_m^* \quad w_m^*)$$

or
$$(1 \quad 2 \quad 4)\mathbf{A} = (x_m^* \quad y_m^* \quad w_m^*) \tag{3.10}$$

then **A** is determined.

As an example, consider representing the semi-circle in figure 3-2 in rational parametric form. The points p_1 , p_0 and p_t are substituted into the **B** matrix and inverted, resulting in,

$$\mathbf{B}^{-1} = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} .$$

Equation (3.9) becomes

$$\mathbf{A} = \begin{bmatrix} k_1 & -2k_2 & k_3 \\ -2k_1 & 2k_2 & 0 \\ k_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \tag{3.11}$$

Note that the tangents at p_0 and p_1 form parallel lines, however their intersection exists at a point at infinity, (0 1 0).

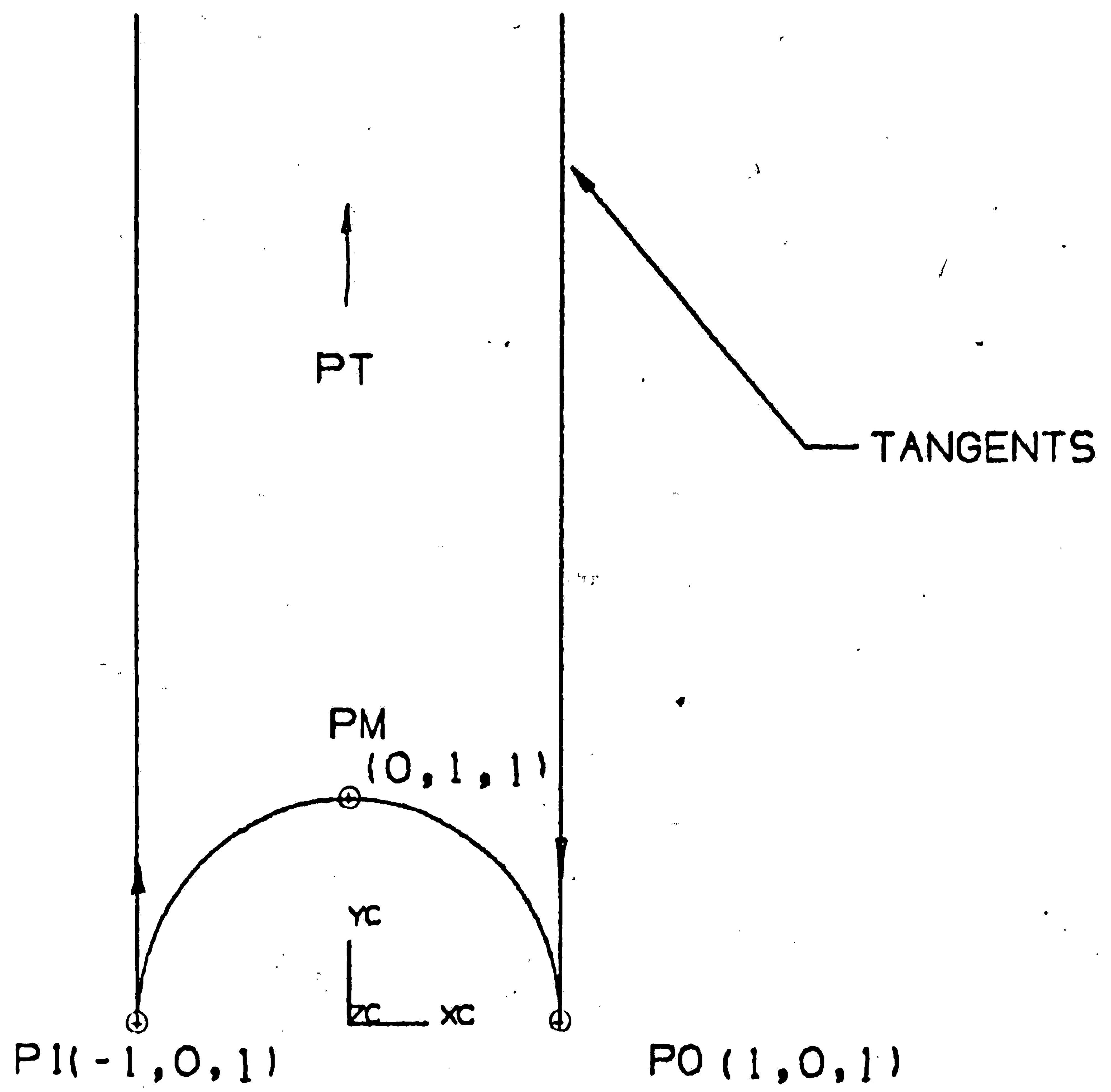
To determine the k's apply equation (3.10),

$$(1 \ 2 \ 4)\begin{bmatrix} k_1 & -2k_2 & k_3 \\ -2k_1 & 2k_2 & 0 \\ k_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} = (0 \ 1 \ 1)$$

which when the three simultaneous equations are solved

$$k_1 = k_2 = k_3 = 1/2 .$$

The transformed point is then



PT IS A POINT AT INFINITY (0,1,0)

FIGURE 3-2. SEMI-CIRCLE

$$(x^* \quad y^* \quad w^*) = (u^2 \quad u \quad 1)\begin{bmatrix} 0 & -2 & 2 \\ -2 & 2 & -2 \\ 1 & 0 & 1 \end{bmatrix}$$

The circular arc may then be written in Cartesian space as

$$x = \frac{x^*}{w^*} = \frac{-2u + 1}{2u^2 - 2u + 1} \quad \text{and} \quad y = \frac{y^*}{w^*} = \frac{-2u^2 + 2u}{2u^2 - 2u + 1} .$$

The general expression for a conic may be written as a rational quadratic after following the steps below. First solve for the unknown k's

$$(k_1 \quad 2k_2 \quad k_3)\begin{bmatrix} x_0^* & y_0^* & w_0^* \\ x_t^* & y_t^* & w_t^* \\ x_1^* & y_1^* & w_1^* \end{bmatrix} = (x_m^* \quad y_m^* \quad w_m^*)$$

where $(x_0^* \ y_0^* \ w_0^*)$ and $(x_1^* \ y_1^* \ w_1^*)$ are the endpoints of the conic section and $(x_t^* \ y_t^* \ w_t^*)$ is the intersection point for the tangents at the end points. The point $(x_m^* \ y_m^* \ w_m^*)$ is the parametric midpoint. The matrix **A** is then

$$\mathbf{A} = \begin{bmatrix} k_1 & -2k_2 & k_3 \\ -2k_1 & 2k_2 & 0 \\ k_1 & 0 & 0 \end{bmatrix}\begin{bmatrix} x_0^* & y_0^* & w_0^* \\ x_t^* & y_t^* & w_t^* \\ x_1^* & y_1^* & w_1^* \end{bmatrix} .$$

The equation of the desired conic as a rational quadratic is then given by

$$(x^* \quad y^* \quad w^*) = (u^2 \quad u \quad 1)\mathbf{A} \quad .$$

The problem with the above formulation is in the division of x^* and y^* by w^*, if w^* is zero a singularity occurs and the computation breaks down. This happens when using an identical point as the starting and ending point as in the representation of a closed conic.

One more drawback is the point at infinity resulting from the two tangents at each endpoint of the semi-circle, in practice the CASPA sculptured surfaces processor cannot handle this case. If this configuration is input the user will receive a CASPA error statement.

3.3 PARAMETRIC CUBIC CURVES

Frequently in design, a smooth curve must be fit to a stream of points. Again the curve takes a parametric form and uses projective geometric concepts.

Many curves cannot be described by a simple analytical expression and are defined in a piece-wise manner. Continuity across the joins between pieces can be built into the parameterized form of the curve. Definition of curve segments require some sort of parametric form that is easily differentiable to determine tangents, normals, curvature, etc. . Polynomial equations are an obvious choice. It is found in practice that a high degree polynomial may exhibit unwanted oscillations and also require more storage (large number of coefficients). However, the degree must be high enough for complex curve representation, and must also satisfy certain continuity requirements. The cubic polynomial is utilized as a compromise. The following development is taken from Ferguson (4) and Faux and Pratt (9). The segment takes the form

$$\mathbf{r} = \mathbf{r}(u) = \mathbf{a}_0 + \mathbf{a}_1 u + \mathbf{a}_2 u^2 + \mathbf{a}_3 u^3 \tag{3.11}$$

The vectors $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2$ and $\mathbf{a}_3$ must be solved for to determine the segment. Four conditions are needed, usually $\mathbf{r}$, and the first derivatives $\dot{\mathbf{r}}$, at each endpoint. If the parameter u is assigned the values u=0 and u=1 at the endpoints of the segment, the coefficients are

$$\begin{aligned}
\mathbf{a}_0 &= \mathbf{r}(0) \\
\mathbf{a}_1 &= \dot{\mathbf{r}}(0) \\
\mathbf{a}_2 &= 3[\mathbf{r}(1)-\mathbf{r}(0)] - 2\dot{\mathbf{r}}(0) - \dot{\mathbf{r}}(1) \\
\mathbf{a}_3 &= 2[\mathbf{r}(0)-\mathbf{r}(1)] - \dot{\mathbf{r}}(0) - \dot{\mathbf{r}}(1) \quad .
\end{aligned} \tag{3.12}$$

Substituting into (3-11) the expression for r results

$$\begin{aligned}
\mathbf{r} = \mathbf{r}(u) = \mathbf{r}(0)(1 - 3u^2 + 2u^3) + \mathbf{r}(1)(3u^2 - 2u^3) + \\
\dot{\mathbf{r}}(0)(u - 2u^2 + u^3) + \dot{\mathbf{r}}(1)(-u^2 + u^3) \quad .
\end{aligned} \tag{3.13}$$

The expression for r may be written in matrix form as r=UCS where UCS denotes the product of the three matrices below:

$$\mathbf{r}(u) = \overset{\mathbf{U}}{(1 \quad u \quad u^2 \quad u^3)} \overset{\mathbf{C}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & -3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}} \overset{\mathbf{S}}{\begin{bmatrix} \mathbf{r}(0) \\ \mathbf{r}(1) \\ \mathbf{r}(0) \\ \mathbf{r}(1) \end{bmatrix}} \tag{3.14}$$

The above formulation is called a Ferguson curve.

3.4 COMPOSITE PARAMETRIC CUBIC CURVES

The curve segment given in equation (3.14) is used to piece together a composite curve. The composite curve may have any desired degree of continuity across joins, however, if a cubic parametric form is used, then the degree of continuity is limited to curvature or C2. Refering to figure (3-3), the segment 1 and segment 2, form a composite curve of two pieces with C2 continuity if

$$\ddot{r}^{(1)}(1) = \ddot{r}^{(2)}(0) \qquad (3.15)$$

assuming tangents are equal in magnitude as well as direction (see ref. 9, pg. 169 for further discussion of curvature continuity between joins). Taking two derivatives of equation (3.13) for each segment and introducing the compatability across the join the result is

$$\begin{aligned} 6r^{(1)}(0) - 6r^{(1)}(1) + 2\dot{r}^{(1)}(0) + 4\dot{r}^{(1)}(1) &= -6r^{(2)}(0) \\ &+ 6r^{(2)}(1) - 4\dot{r}^{(2)}(0) - 2\dot{r}^{(2)}(1) \ . \qquad (3.16) \end{aligned}$$

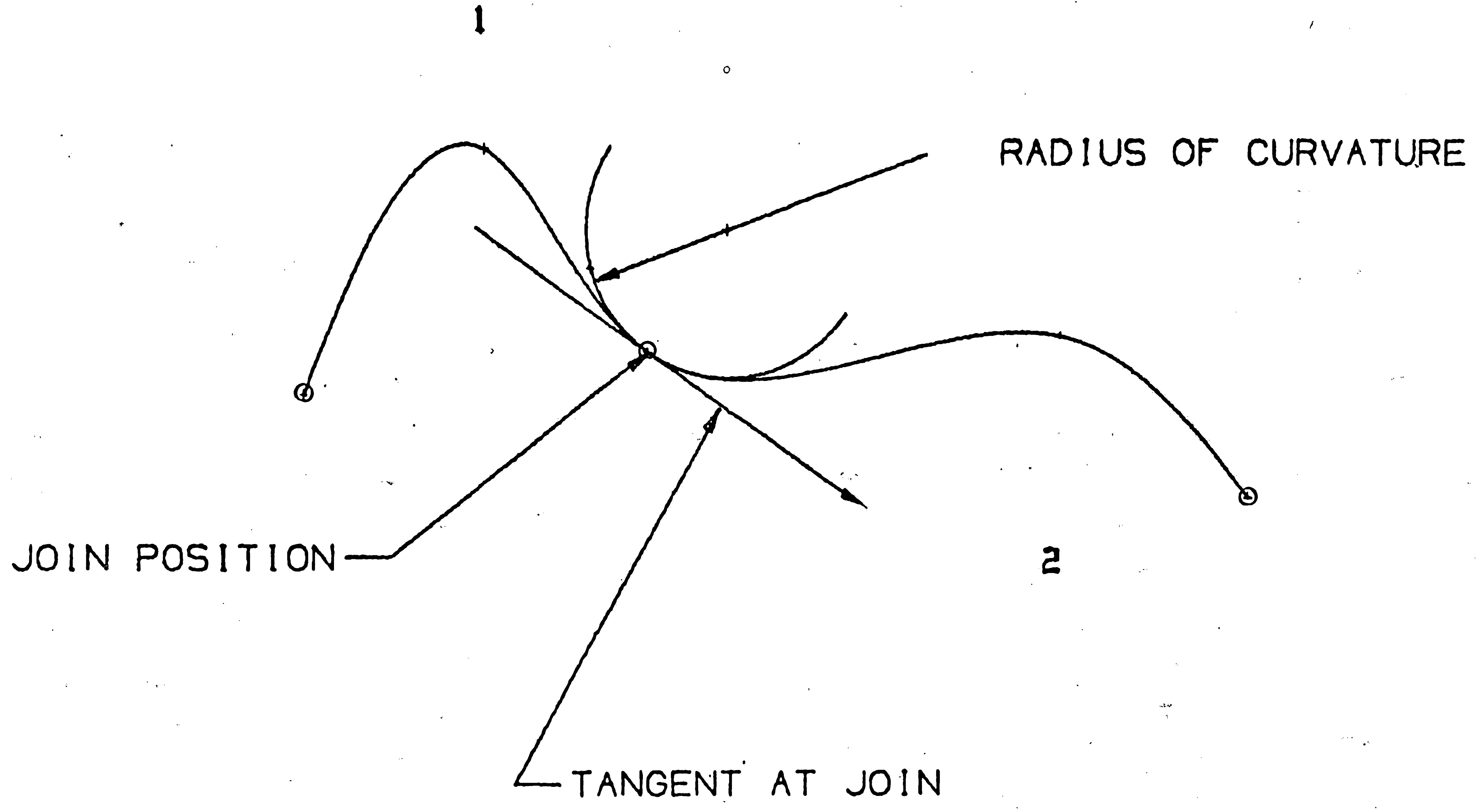


FIGURE 3-3. COMPOSITE CURVE

This simplifies because of the earlier assumption that tangents are equal. If the curve is fit through a series of points $(\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_n)$ with tangents at these points $(\mathbf{t}_0, \mathbf{t}_1, \ldots, \mathbf{t}_n)$ then equation (3.16) may be written as

$$\mathbf{t}_{i-1} + 4\mathbf{t}_i + \mathbf{t}_{i+1} = 3(\mathbf{r}_{i+1} - \mathbf{r}_{i-1}) \text{ , } i=1,2,\ldots,n-1 \qquad (3.17)$$

This recurrance relation between the tangents at three successive points, allows the determination of all internal tangents, given the tangents at the endpoints $\mathbf{t}_0$, $\mathbf{t}_n$, in terms of the positional data. The parameter u is usually based on chord length.

3.5 PARAMETRIC BI-CUBIC PATCHES

The type of surface used in CISPA under the APT complex is called a Ferguson surface, because the surface is made up of a network of Ferguson curves. The Ferguson surface is a special case of the Coons generalized form and often denoted as a tensor-product (or Cartesian product) patch, for the following form

$$\mathbf{r}(u,v) = \mathbf{F}(u)\mathbf{Q}\mathbf{F}^T(v) \text{ .} \qquad (3.18)$$

This simplification from the generalized form occurs because the boundary curves are defined using the same blending functions as are used in the construction of the surface patch.

The function r(u,v) is bi-variate in u and v, and dependant on the matrix product formed by $\mathbf{FQF}^T$ where

$$F(u) = \{\alpha_0(u) \quad \alpha_1(u) \quad \beta_0(u) \quad \beta_1(u)\}$$

and

$$\mathbf{Q} = \begin{bmatrix} r(0,0) & r(0,1) & r_v(0,0) & r_v(0,1) \\ r(1,0) & r(1,1) & r_v(1,0) & r_v(1,1) \\ r_u(0,0) & r_u(0,1) & r_{uv}(0,0) & r_{uv}(0,1) \\ r_u(1,0) & r_u(1,1) & r_{uv}(1,0) & r_{uv}(1,1) \end{bmatrix}$$

The F vector is made up of interpolating functions and are chosen to satisfy the following conditions

$$\begin{gathered} \alpha_0(0) = 1\ , \quad \alpha_0(1) = 0, \\ \alpha_1(0) = 0, \quad \alpha_1(1) = 1, \\ \alpha_0'(0) = \alpha_0'(1) = \alpha_1'(0) = \alpha_1'(1) = 0 \\ \beta_0(0) = \beta_0(1) = \beta_1(0) = \beta_1(1) = 0 \\ \beta_0'(0) = 1, \quad \beta_0(1) = 0, \\ \beta_1'(0) = 0, \quad \beta_1'(1) = 0 \quad . \end{gathered} \tag{3.19}$$

This patch exhibits C1 continuity. See Forrest (18) for a general approach to determine the blending functions for patches with Cn continuity.

One set of polynomial blending functions satisfying the group of conditions (3.19) and also yielding aesthetically pleasing results are the Hermite polynomials of order 1:

$$\alpha_0(u) = 1 - 3u^2 + 2u^3, \qquad \alpha_1(u) = 3u^2 - 2u^3 \qquad (3.20)$$
$$\beta_0(u) = u - 2u^2 + u^3, \qquad \beta_1(u) = -u^2 + u^3$$

Note that these functions form the same basis as the one used in the formulation of the Ferguson curves, equation (3.13).

Matrix **Q** contains geometric data from the patch corners, partitioning the matrix simplifies analysis of this data,

$$\mathbf{Q} = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right] .$$

The **A** array contains positional information, it is possible that two of the four points may be identical, yielding as degenerate case a triangular patch. The **B** and **C** partitions hold tangent information in the v and u directions respectively, for each corner of the patch. Finally, partition **D** holds the parametric cross derivatives called 'twist vectors'. These vectors do not necessarily measure twist in a surface but depend greatly on the way the surface is parameterized. Since **Q** must be input, the question arises how to specify **D**?

Frequently, **D** is set to [0], the null matrix, however this may lead to local flattening of the generated surface near the patch corners (18). By specifying C2 continuity, that is curvature continuity across patch boundaries the **D** matrix is determined. The reason being that the parametric cross derivatives must be matched to adjacent patches. The derivation follows much the same method as those resulting in equation set (3.12). See reference (9, Sect. 7.2.1) for more information on Ferguson patches with C2 continuity.

Since **r**(u,v) is a vector comprised of four elements, each patch will necessate the storage of four **Q** matrices. The surface definition processor distinguishes special sub-types of **Q** which permit less storage and greater computational speed. Accordingly, each matrix is assigned a flag based on the following distinction.

TYPE 1 Constant Matrix

The general form is

$$\mathbf{A} = \begin{bmatrix} k_1 & k_1 \\ k_1 & k_1 \end{bmatrix} \qquad \mathbf{B} = \mathbf{C} = \mathbf{D} = [0]$$

in this case

$$\mathbf{r}(u,v) = k_1 \quad \text{for all } u,v.$$

TYPE 2 Linear matrix

The linear matrix is of the form

$$\mathbf{A} = \begin{bmatrix} x_1 & x_3 \\ x_2 & x_4 \end{bmatrix}; \quad \mathbf{B} = \begin{bmatrix} x_3-x_1 & x_3-x_1 \\ x_4-x_2 & x_4-x_2 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} x_2-x_1 & x_4-x_3 \\ x_2-x_1 & x_4-x_3 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} (x_4-x_2)-(x_3-x_1) & (x_4-x_2)-(x_3-x_1) \\ (x_4-x_2)-(x_3-x_1) & (x_4-x_2)-(x_3-x_1) \end{bmatrix}$$

Upon doing the matrix multiplication the expression for **r**(u,v) becomes

$$\mathbf{r}(u,v) = [x_1(1-u) + x_2u](1-v) + [x_3(1-u) + x_4u]v$$

This sub-type requires storage of only four numbers and a flag, in this simple form computation time is greatly improved.

TYPE 3 Zero Twist Matrix

Here the parametric cross derivatives are all set to zero yielding

$$\mathbf{D} = [0].$$

A,**B**, and **C** are non-zero, 12 entries are stored.

TYPE 4 Full Storage

The full **Q** matrix is stored.

4. CISPA'S ALGORITHMIC WORK AND DATA STRUCTURE

4.1 INTERACTIVE FEATURES

CISPA is, essentially, a menu-driven front-end for a set of APT routines used in APT 4 for the creation of curve and surface geometry. Enhanced features for user-friendliness include several methods for creating points, curves and surfaces, supported by APT 4, picking, modifying, saving, deleting and naming entities, improved error recovery and error flagging, advanced viewing features using GRAPH3D graphics package and increased flexibility in creating the CLSF (cutter location source file) to be used in NC machining. Surface tolerance control expressed as cutter step size may be based on chordal deviation from the exact surface, additionally scallop height may be input to fix the number of zig-zag passes needed to machine the surface. Zig-zag paths may be reviewed and saved on external files.

The menus are designed, allowing the user to move along a decision path based on graphical feedback from the display monitor. The main menu displays options to either submit a language input file, start an interactive session or exit the program. Under the interactive module full use of defaults, colors, line fonts and symbols, are utilized to guide the user through point, curve and surface design. The display functions both as the entity graphics display and message monitor. Graphics occupy the viewport with menus, system inquiries, warning or error statements super-imposed on the left side of the screen. These messages may be blanked if the user wishes. See Appendix C for menu flow charts and refer to 'CISPA Package Manual' (11) for details.

At each decision node a global menu is activated by pressing the 'G' key on the keyboard. Contained in the global menu are the options, redraw, view manipulations, clear graphics or dialog and hard copy. Redraw repaints everything in the picture file that has not been deleted. View manipulations allows the user to enter view control or image control menus. Under view control the model may be rotated, screen rotated or eye point changed to aid in visualization. The ORIENT option facilitates view transformation by entering single key inputs to change rotation, scaling and translation parameters. For example, by pressing the 'x' key the model is rotated about the x axis by a pre-defined increment. Clear graphics and clear dialog are self-explanatory. Hard copy generates a picture on the line printer using the image currently on the screen. Besides the global menu switch 'G' there are other global characters - '/', a slash character or a '!' , an exclamation mark character back step through the menus. In general, a '/', returns to a major decision point and an '!', back steps one in the menu structure. A 'CR', carriage return, is used as a entry complete switch, the 'SB', space bar, functions as the selection key when picking entities by cursor, a 'r', the letter r, key is utilized as a global reject. Under the 'enter point' menu a '$' , the dollar sign key, indicates an alternate action is desired, that is if one is presently entering points by picking existing points, an alternate method of entering points may be selected by hitting the '$' key dropping control back to the enter point menu. User-friendliness of CISPA is augmented by an error-handling and error recovery system. In the event a system error (host computer) occurs, the user may re-enter the CISPA program and continue work since all input and canonical information is saved externally. A list of error statements is kept in a file, ERROR.DAT. The record numbers of which are stored in an array defined and initialized at the beginning of the program by ERR_INIT. The record numbers are linked to error codes that are flagged during program operation. If an error code is flagged the code is matched with the record number and the error description is read from ERROR.DAT. Three fields are used in the data file: a one integer error status code (0 to 4), where 0 is a fatal error, 1 is a warning or non-fatal error, the remainder is currently unused; a seven integer error code; and a character string with provision for continuation. The write statement to the screen starts with an 'E' or 'W' depending on the error status, continues with the error code and a descriptive text statement. The error code and status code are retained in a common block so that error conditions may be handled more flexibly. Control is returned to the user to continue, make modifications or start the interactive session over.

All curves and surfaces have default or user input names to aid in retrieving, selecting or deleting entities. The names are associated with storage locations holding canonical forms on the external file, FOR004.DAT. The names are limited to four charaters and may not be duplicated. The storage location will be refered to as an entity number. These numbers are used to link start and end locations of original input stored on external file CSAVESET.DAT, and also reference the start and end locations of the discrete plot data in the picture file for efficient display processing. Importantly this number identifies the entity and points associated with that entity.

The default name is a concatenation of a descriptive letter ('C' for curve, 'S' for surface, 'P' for path) and a number which is incremented with the naming of each entity. The number must be represented as a character value to perform the concatenation. This is done with the character function CHAR provided in FORTRAN, which converts an ASCII integer value to a character value. All name table data is converted to an integer form for storage and this is accomplished by an internal read statement (18). Figure 4-1 shows the simplified routine for the above.

All input from the terminal is guarded against extraneous data entry. For example, if numeric data is expected and characters are inadvertantly typed in, the user will be prompted to re-enter given a data type mismatch. This is best exemplified in the routine READ_WORDS, which reads a list of names for the retrieve function. READ_WORDS examines a long character string (300 characters) a character at a time counting a space or a comma as a delimiter, parses the list and returns a double precision array of the names contained in the list. The following flags are set depending on character or more precisely the ASCII equivalents encountered:

```
      SUBROUTINE DEF_NAME(NAME,NUM,LETTER)
      CHARACTER*20 PROMPT
      CHARACTER*1  NAMENUM(4),LETTER
      CHARACTER*4  NMTEM
      INTEGER*4    NAME,NUM,LNM
      DATA PROMPT /'  ENTER THE ENTITY NUMBER('/
      NAMENUM(1) = LETTER
      NAMENUM(3) = ' '
      NAMENUME(4)= NAMENUM(3)
      IF(NUM.LT.10) THEN
      NAMENUM(2) = CHAR(48 + NUM)
      ELSE
      NAMENUM(2) = CHAR(48 + NUM/10)
      NAMENUM(3) = CHAR(48 + JMOD(NUM,10))
      ENDIF
      NMTEM      = NAMENUM(1)//NAMENUM(2)//NAMENUM(3)//NAMENUM(4)
      READ(NMTEM,10) LNM
      WRITE(6,20) PROMPT,NAME
      READ(5,30) NCHARS,NAME
      IF(NCHARS.EQ.0) NAME = LNM
 10   FORMAT(A4)
 20   FORMAT(A25,A3,'):',$)
 30   FORMAT(Q,A8)
      RETURN
      END
```

FIGURE 4-1. SIMPLIFIED DEFAULT NAMES ALGORITHM

- occurance of a special character before the first word,
- occurance of any special character anywhere in the string,
- occurance of a comma after the last word,
- occurance of a numeric digit as the first character of any word, is ignored, but sets a flag,
- number of words or names exceeds a certain limit,
- the number of characters entered in the word list is zero.

Special characters are those ASCII characters not alpha-numeric.

Interaction is enhanced by taking advantage of cursor selection of entities. This selection mode is used in point creation, in picking of curves in synthetic curve and surface creation, and deleting entities. Verification of a correct selection is accomplished by re-plotting a point or curve in blue, indicating it has been picked. If the entity is picked by mistake the next operation should be to reject the pick by hitting the 'r' key. The curve is re-plotted in its original color, and a correct selection may occur. The underlying stategy for all pick routines is shown in the flow diagram, figure 4-2. Referring to figure 4-2, note that information about a previous pick is dependant on what one does presently, in this way data is stored temporarily as a candidate pick and sent to worker routines only if the next pick is concerned with transmitting new information (entry complete or new selection). As a study, examine the routine FINDJPT, this routine allows selection of junction points on a synthetic curve for re-parameterization of the curve segment or what will later be thought of as specification of a flow segment. The

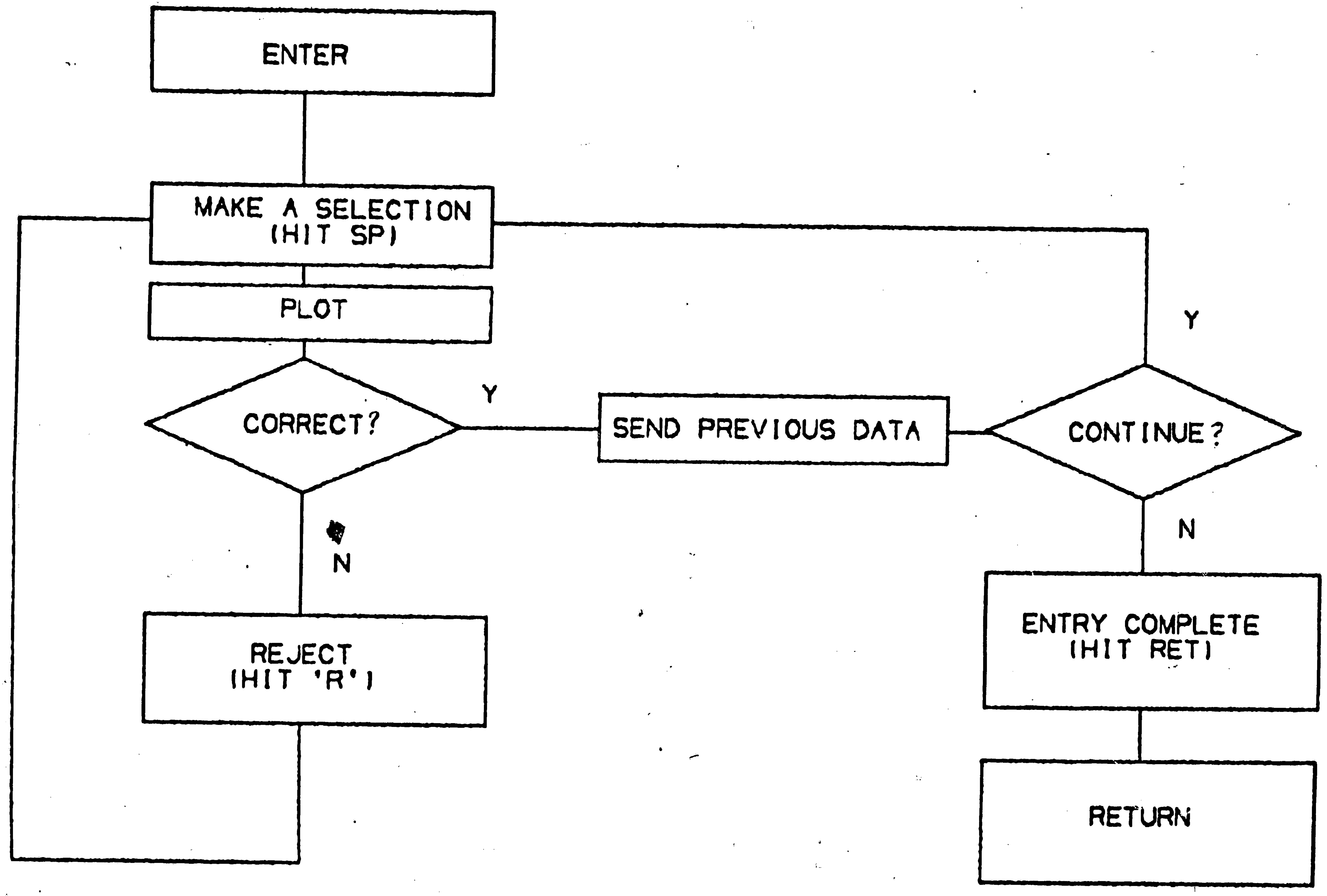


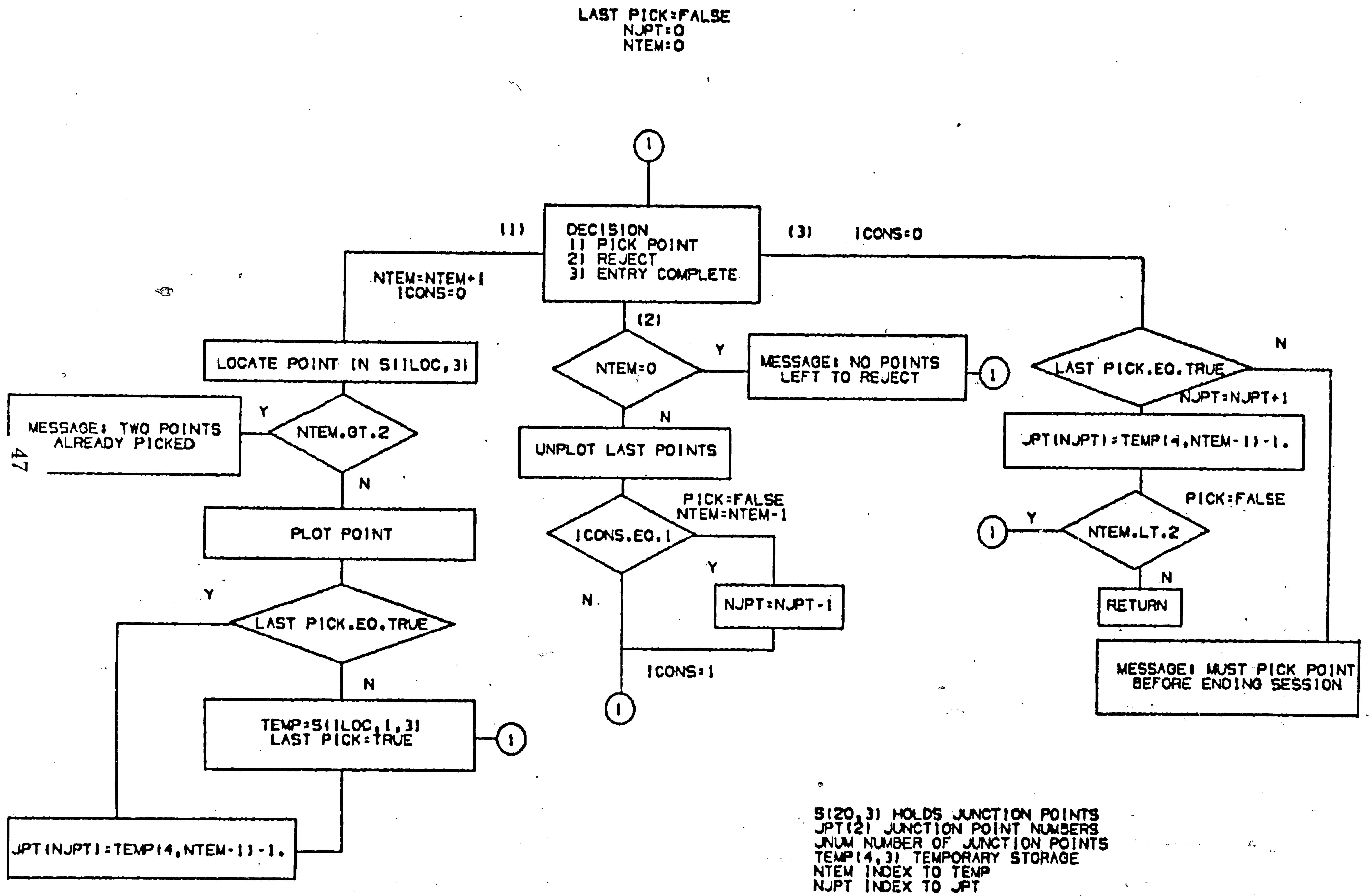
FIGURE 4-2. STRATEGY USED IN PICKING ROUTINES

junction points are contained in an array which is defined by one pass through the picture file. Cursor location is compared to the array data.

A flow structure is defined by describing the parameterization for each segment while moving from one end of the curve to the other. This FINDJPT routine is flow charted in figure 4-3. Note the storage of candidate data in array TEMP, the pointer to TEMP is NTEM, the desired data is stored in JPT with pointer NJPT. Points may be rejected, as long as there are points to reject, by backstepping the pointer NTEM. Data in JPT will be overwritten. This routine could be used for the general task of picking things that are presented in some chained sequence.

4.2 VARIABLE CUTTER STEP SIZE ALGORITHM

In chapter 3, a surface patch definition was derived as a function of two parameters u and v. Holding one parameter constant defines an isoparametric curve on the surface patch and if one of the parameters is set to zero or one the curve becomes exactly one of the boundary curves surrounding the patch. If both parameters are held constant, a point is specified on the surface patch. To mill a surface parametrically, cutter location points are generated using the patch definition and current machining parameters, stored in a file and submitted to the NC machine processor. The ball end cutter tip is



S(20,3) HOLDS JUNCTION POINTS
JPT(2) JUNCTION POINT NUMBERS
JNUM NUMBER OF JUNCTION POINTS
TEMP(4,3) TEMPORARY STORAGE
NTEM INDEX TO TEMP
NJPT INDEX TO JPT

FIGURE 4-3. FLOW CHART OF ROUTINE FINDJPT

directed to move in straight line motions to each point. The fewer the points the more rough the surface will appear. On the other hand, it's undesirable to create huge point files, for reasons of storage, computation and milling time. CISPA uses two methods of calculating cutter location points: chordal deviation tolerance and iso-parametric step size in the primary direction (u) and scallop height tolerance and iso-parametric step size in the cross direction (v).

The iso-parametric step size definition requires only input of percent change along each direction. For example, if the iso-parametric step is specified as .1 (10%) in the u and v directions, then u moves from 0 to 1 in steps of .1 and v moves from 0 to 1 in steps of .1 creating 11 points per pass, 11 passes, thus 121 points entered into the CLSF for one patch. The problem with this type of step definition is that it does not take into account areas of local flattening where fewer points are required, plus it does not base the input on geometric quantities that are familiar, such as surface tolerance. With the above in mind, algorithms were formulated basing the primary step on maximum chordal deviation (see figure 2-2). The input being a tolerance not to be exceeded-- the distance from the ball end cutter tip to the exact surface. Referring to figure 4-4, a straight line segment is drawn from A(0) to B(1) the chordal deviation is calculated, if the deviation is greater than the input tolerance, the new end points A(0) and P(r(u)) are used and checked for tolerance. The curve is subdivided further until all straight line segments are within the specified deviation. To calculate the maximum

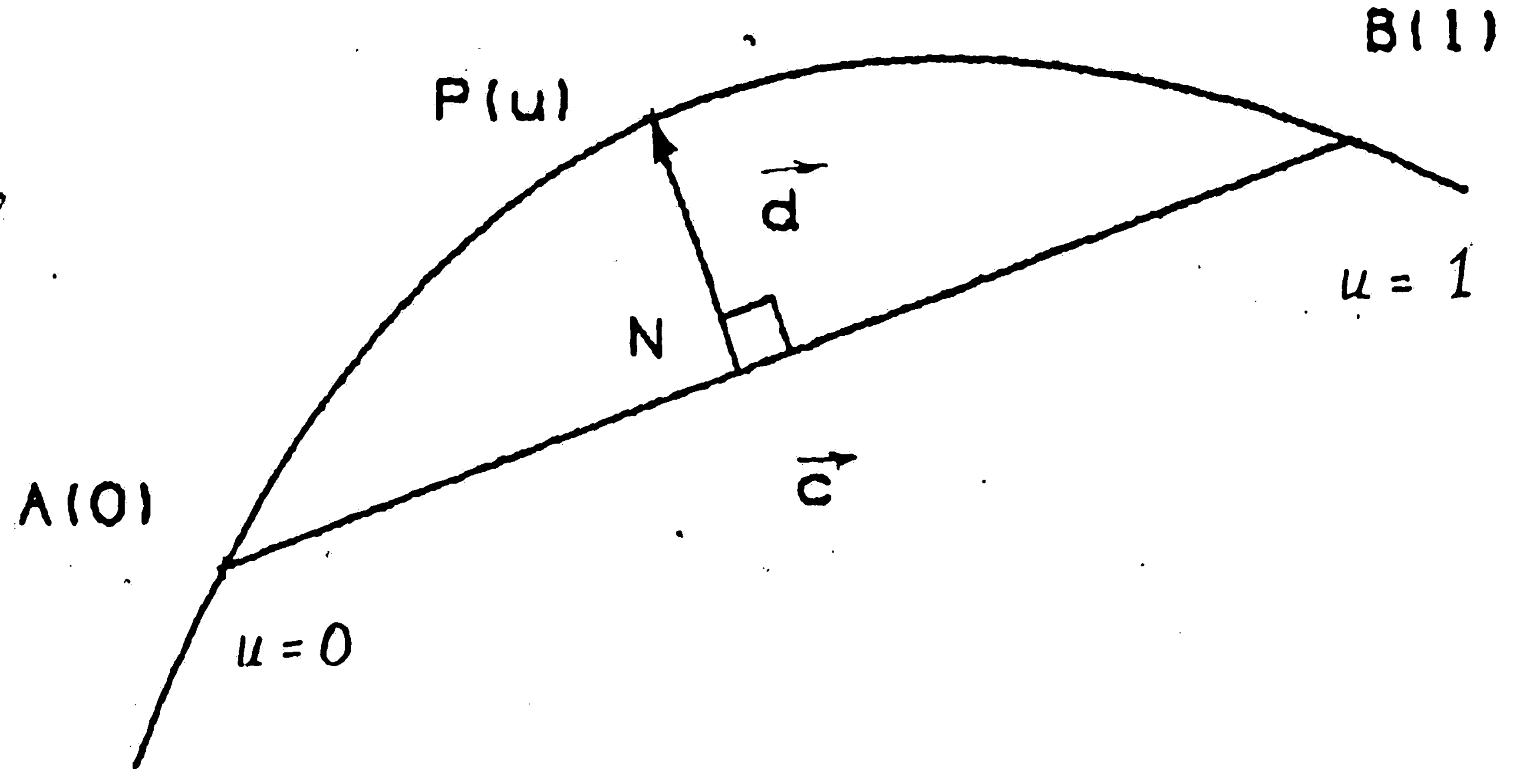


FIGURE 4-4. CHORDAL DEVIATION NOTATION

normal distance, δ , the curve $\mathbf{r}(u)$ must be known. Since the patch definition exists, holding v constant, lets

$$\mathbf{r}_i(u) = \mathbf{r}(u,v_i) \quad .$$

To calculate the maximum normal distance from the chord joining A(0) and B(1) then the condition, u=0 and u=1 at the respective end points must be satisfied. Any curve segment which is parameterized between $u_0 \le u \le u_1$ may be transformed to new parameter u' where $0 \le u' \le 1$ by the following transformation

$$u = (1-u')u_0 + u'u_1 \quad . \qquad (4.1)$$

If the above equation is written as

$$u = u_0 + \Delta u_0 u', \text{ where } \Delta u_0 = u_1 - u_0$$

then

$$u^n = \sum_{r=0}^{n} {}^nC_r u_0^{\,r} (u'\Delta u_0)^{n-r}$$

for n=3, in matrix form

$$[1 \quad u \quad u^2 \quad u^3] = [1 \quad u' \quad u'^2 \quad u'^3] \begin{bmatrix} 1 & u_0 & u_0^2 & u_0^3 \\ 0 & \Delta u_0 & 2u_0\Delta u_0 & 3u_0^2\Delta u_0 \\ 0 & 0 & \Delta u_0^2 & 3u_0\Delta u_0^2 \\ 0 & 0 & 0 & \Delta u_0^3 \end{bmatrix}$$

or **U=U'T.**

The chord AB will be noted by vector **c** , shown in figure 4-4. Let d(u) represent the perpendicular from AB to general point P(**r**(u)) of the curve. The equation

$$\mathrm{d} = \mathbf{r}(u) - \mathbf{r}(0) - \lambda\mathbf{c} \qquad (4.2)$$

where **c** and **d** are orthogonal or

$$\mathbf{c} \bullet \mathrm{d} = 0. \qquad (4.3)$$

Using (4.3) in (4.2)

$$c\bullet[r(u) - r(0)] = \lambda|c|^2$$

$$d = r(u)-r(0) - \frac{[r(u)-r(0)]\bullet c}{|c|^2}c \quad .$$

Which can be written

$$d = P\ [r(u)-r(0)] \qquad (4.4\ )$$

where P is called the projection matrix $I - cc^T/c^Tc$ (9).

For the cubic polynomial derived in Chapter 3, in the equivalent Bezier's cubic form, equation (4.4) becomes

$$d = 3u(1-u)P[(1-u)(r_1-r_0) + u(r_2-r_0)] \qquad (4.5\ )$$

where $r_0 = r(0)$, $r_1 = r(0) + \dot{r}(0)/3$.
$r_3 = r(1)$, $r_2 = r(1) + \dot{r}(1)/3$.

The vectors r_0, r_1, r_2 and r_3 are the control points for the equivalent cubic Bezier curve.

At the maximum perpendicular, the curve tangent is perpendicular to d , so that

$$\dot{r}^Td = \dot{r}^TP[r(u)-r(0)] = 0. \qquad (4.6\ )$$

Although this polynomial equation is of order 2n-1, for a curve of degree n, the factors u(1-u) occur in (4.5) and thus two roots are known, only 2n-3 roots must be solved for.

For the cubic curve, n=3 and there are three roots of the equation

$$[(1-u)^2(r_1-r_0) + 2u(1-u)(r_2-r_1) + u^2(r_3-r_2)]^TP$$
$$x[(1-u)(r_1-r_0) + u(r_2-r_0)] = 0 \quad . \qquad (4.7\ )$$

At a true maximum, $\ddot{r}^T d < 0$, so the condition

$$[(1-u)(r_2-2r_1+r_0) + u(r_3-2r_2+r_1)]^T P \times [(1-u)(r_1-r_0) + u(r_2-r_0)] < 0 \qquad (4.7a)$$

must be satisfied at each root calculated in equation (4.7).

If $c = \{c_1\ c_2\ c_3\} = r_3-r_0$

then matrix **P** is

$$P = \begin{bmatrix} 1-c_1^2/c_n^2 & -c_1c_2/c_n^2 & -c_1c_3/c_n^2 \\ -c_1c_2/c_n^2 & 1-c_2^2/c_n^2 & -c_2c_3/c_n^2 \\ -c_1c_3/c_n^2 & -c_2c_3/c_n^2 & 1-c_3^2/c_n^2 \end{bmatrix}$$

where $c_n^2 = c_1^2 + c_2^2 + c_3^2$.

Figure 4-5 shows a flow diagram of steps taken to generate the CLSF for a surface given the maximum chordal deviation criteria.

Starting at one corner of the surface, a patch is loaded, v is held at one value while chords are approximated in the u direction, the point located during the sub-division process are stored, by storing the parameter u_j associated with the points in USUB. USUB is a one-dimensional array holding the u parameters (u_j's) fixing the points on the iso-parametric curve that will later be used in the representation of the curve by straight line segments. The points are written to the CLSF in binary once a new patch is encountered, moving along the u direction. Each patch has topological information connected with it by referring to array TOP, when the last chord approximating $r_i(u,v_i)$ is detected a new patch may be loaded depending

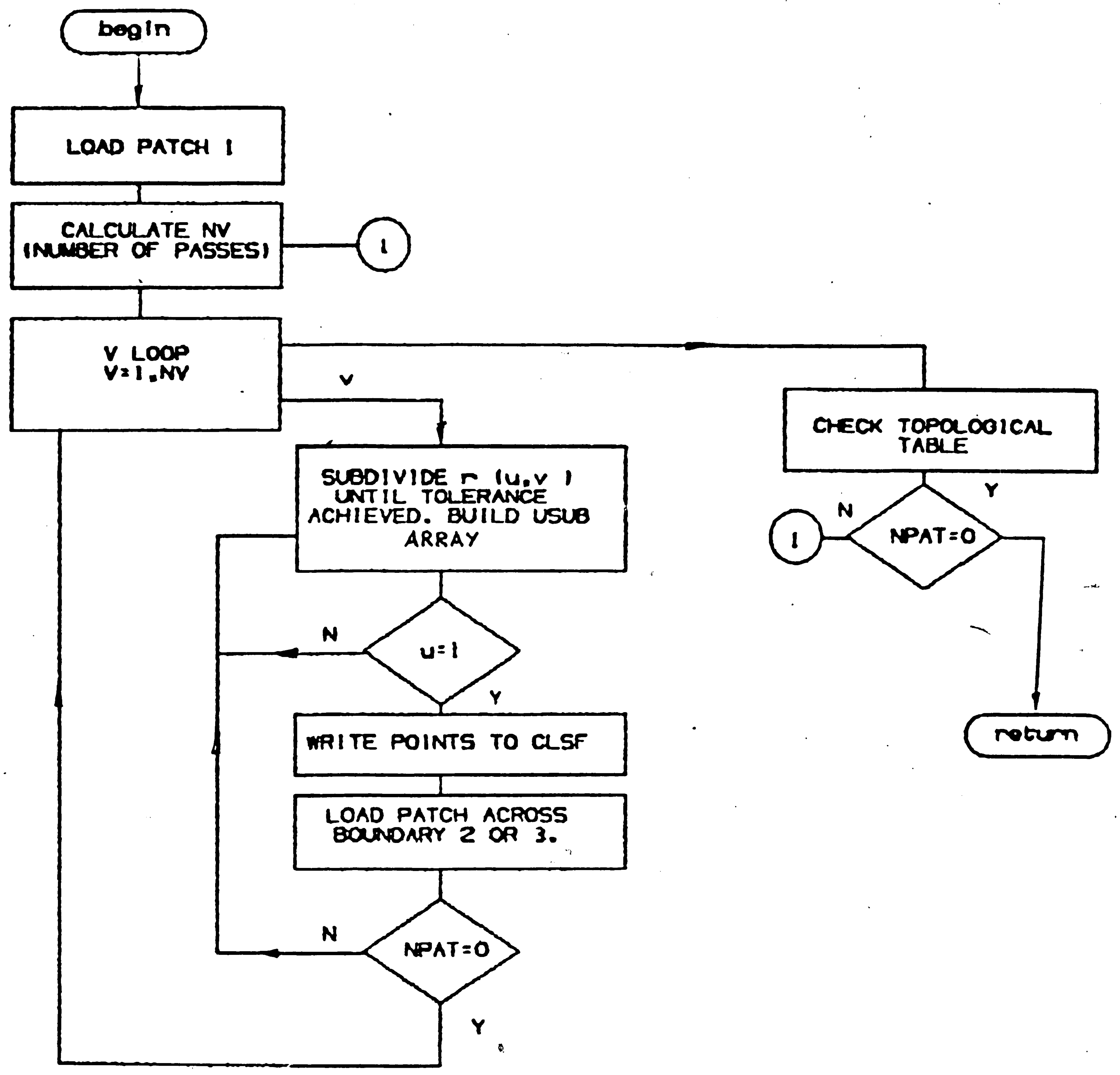

begin
LOAD PATCH 1
CALCULATE NV (NUMBER OF PASSES)
1
V LOOP V=1,NV
V
SUBDIVIDE r (u,v) UNTIL TOLERANCE ACHIEVED. BUILD USUB ARRAY
N
u=1
Y
WRITE POINTS TO CLSF
LOAD PATCH ACROSS BOUNDARY 2 OR 3.
N
NPAT=0
Y
CHECK TOPOLOGICAL TABLE
Y
N
1
NPAT=0
return

FIGURE 4-5. FLOW CHART FOR SURFACE TOLERANCE

on the information in TOP (13). Specifically, if the patch number is zero, no further milling takes place along that direction and either the mill operation is deemed finished or a side step in the v direction occurs.

The roots for the cubic equation $Pu^3 + Qu^2 + Ru + S = 0$ resulting from the chordal deviation calculation are found by employing routine GCUBIC. This routine uses a classic closed form solution of a cubic equation. If the discriminant is positive, an algebraic solution method is used, and, if negative a trigonometric solution is utilized.

The result of this algorithm was a very significant reduction in the number of points needed to generate the surface without loss of accuracy. In one case (the yacht hull shown in figure 4-12) the two methods were compared, iso-parametric and chordal deviation both sharing the same maximum allowable tolerance. The variable step calculation gave a 68% reduction in the CLSF, trading off computational time, which was doubled.

4.3 SIDE STEP BASED ON SCALLOP HEIGHT

To relate the side step or step size in the v direction, to a surface attribute, definition was based on scallop height, see figure 2-3. The input more precisely is the maximum scallop height appearing on the surface after milling.

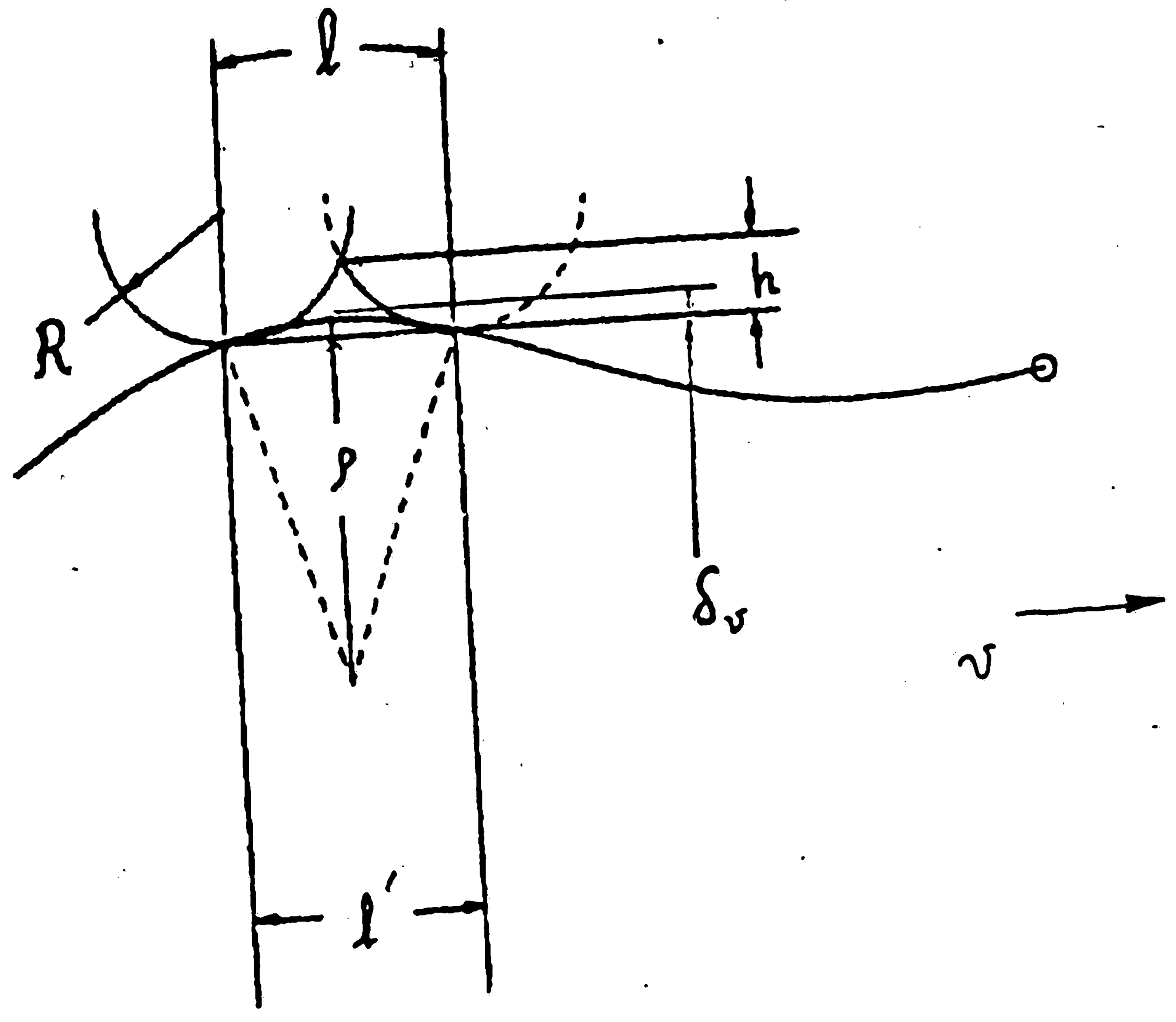


FIGURE 4-6a. GEOMETRY NEEDED FOR SCALLOP HEIGHT CALCULATION

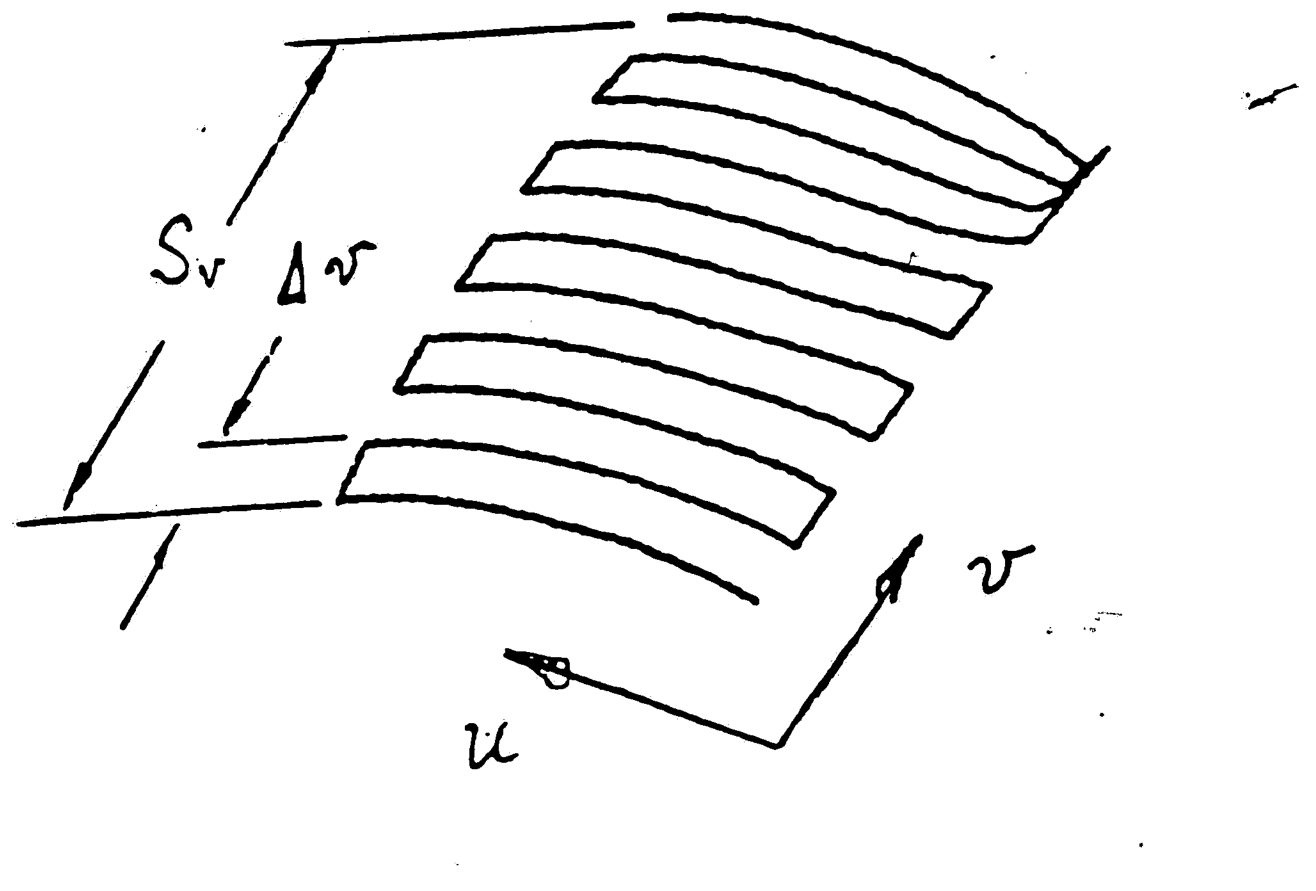


FIGURE 4-6b. PATCH SHOWN WITH u_i CURVES AND S_v SHOWN

Figure 4-6a shows the geometry needed for this calculation, where R is the ball end mill radius, h is scallop height, and l is the side step. The relation is then

$$l' = 2\sqrt{h(2R-h)} \tag{4.8 }$$

and the fractional parametric step is

$$\Delta v = 1/S_v \tag{4.9 }$$

S_v is the total arc length given constant u, that is

$$S_v = \int_0^1 \left| \dot{r}(u_i,v) \right| \, dv \quad . \tag{4.10}$$

Figure 4-6b shows the above analysis.

For small steps, the curve may be approximated by its osculating circle, radius ρ . The chordal deviation for a side step based on scallop height, is calculated using Pythagoras' theorem

$$\delta_v = \rho - \sqrt{\rho^2 - (l'/2)^2} \quad , \tag{4.11}$$

where l' is the chord length shown in figure 4-6a. The arc length l is related to the angle between the end points of chord l'

$$l = \rho\theta$$

and then

$$l' = 2\rho\sin(\theta/2) = 2\rho\sin(l/2\rho) \quad . \tag{4.12}$$

The chordal deviation is formed by substituting equation (4.12) into (4.11)

$$\delta_v = \rho[1-\cos(l/2\rho)] \tag{4.13}$$

for $l/2\rho$ small, $\cos(l/2\rho)$ may be replaced by

$$\cos(l/2\rho) = 1- l^2/8\rho^2$$

and

$$\delta_v = h(2R-h)/2\rho \tag{4.14}$$

Finally, the dimensionless equation is

$$\frac{\delta_v}{h} = \frac{R}{\rho} - \frac{1}{2\rho}h \qquad (4.15)$$

frequently the term R/ρ is less than one to avoid gouging the surface, therefore δ_v/h is less than one meaning the chordal deviation on a side step is less than the given scallop height tolerance. The restriction on input h is $h<R$ meaning the step is small enough to ensure that the surface will be milled entirely.

The integration in equation (4.10) is done with three point Gaussian quadrature, see reference (20) for details.

Since the arc length may vary as u changes parameter value, each patch is sampled at $u=0, .25, .5, .75$, and 1. and the greatest number of passes calculated is chosen as the value needed to achieve the given scallop height tolerance or less. In practice, the patches are not so severely deformed that this procedure errors, however, more checks for arc length may be implemented at the expense of computational time.

The method for calculating scallop height relies on evaluation of five sample arc lengths per patch. More accurate assessment of scallop height may be undertaken by taking into account surface normals and exact point of tangency at the surface for each point. Since a milled surface must be finished eventually, it seems unreasonable to use local checking of scallop height for the little accuracy gained in surface tolerance.

Shown in figure 4-7 is a general flow chart of how scallop height is used to digitize the surface for NC path generation.

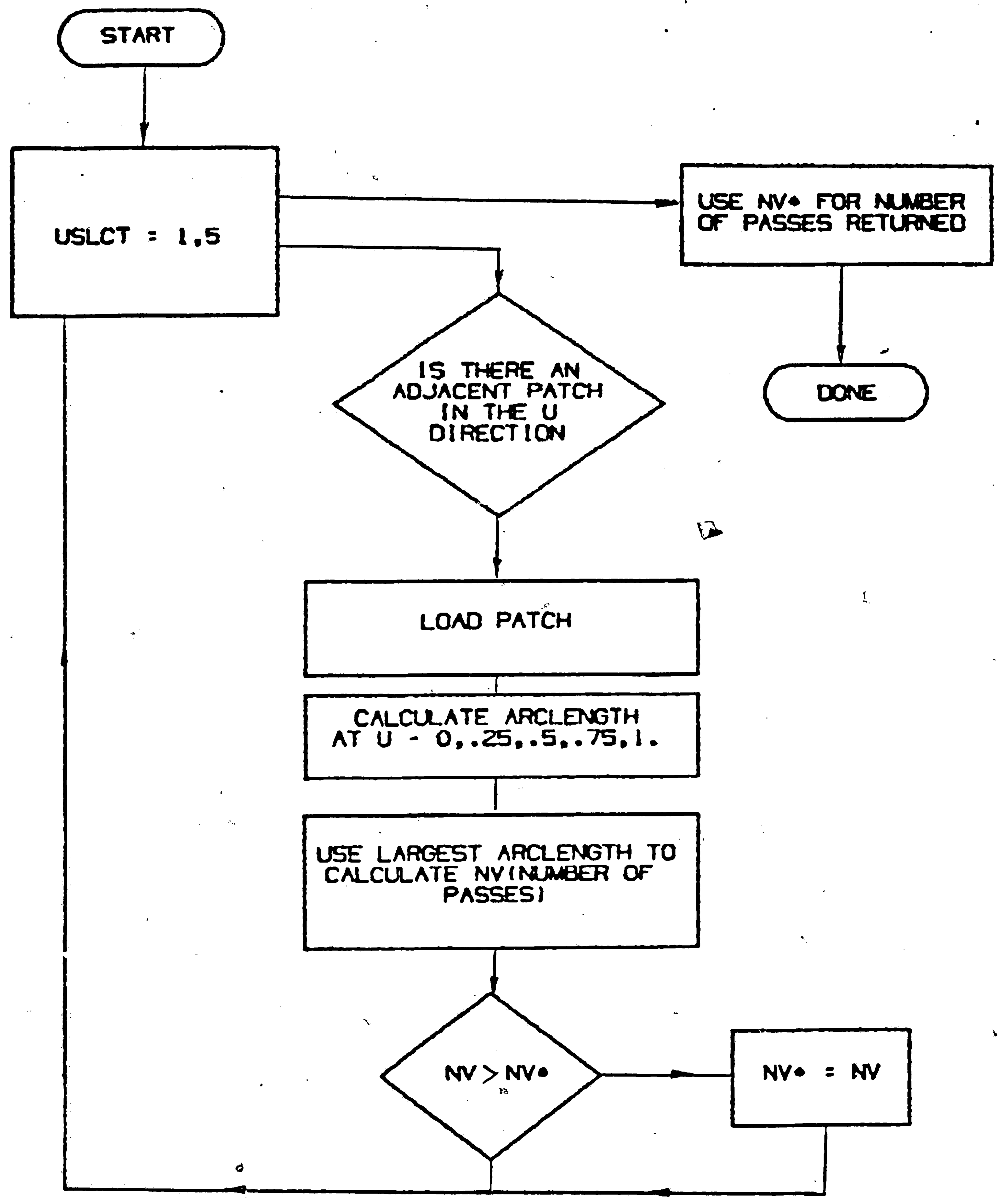

START
USLCT = 1,5
USE NV* FOR NUMBER OF PASSES RETURNED
DONE
IS THERE AN ADJACENT PATCH IN THE U DIRECTION
LOAD PATCH
CALCULATE ARCLENGTH AT U - 0,.25,.5,.75,1.
USE LARGEST ARCLENGTH TO CALCULATE NV(NUMBER OF PASSES)
NV > NV*
NV* = NV

FIGURE 4-7. FLOW CHART OF SCALLOP HEIGHT CALCULATION

4.4 ROUGH TO DEPTH MILLING

While associating step size to surface attributes such as tolerance allows more efficient surface representation, it also reduces the number of points in the CLSF without sacrificing accuracy. Hueristically, points are put where there is greater curvature, therefore more line segments are needed to give a good approximation to the exact surface. Reduction in points is particularly useful when roughing-to-depth, see figure 2-4. The rough-to-depth module in CISPA allows the milling of a surface in incremental layers, where a direct plunge to surface by the cutter would certainly damage the tool, machine or both.

In actuality, the cutter removes material in layers until the surface is encountered, the cutter rides up over the surface and continues to cut within the layer. Successive layers are milled, the cutter rides up on the surface until the adjacent upper layer is detected where the cutter moves to a clearance plane, makes a rapid move to the next location above a point on the surface within the cutting layer and milling continues. By making rapid moves, previously milled areas are not re-milled, thus saving time. The path is written to a file named CLSF.DAT on logical unit 12. The path on each layer is defined by projecting a previously defined path onto a cutting plane or critical plane and flagging locations where the surface z coordinates are above this plane. Figure 4-8 shows a simplified flow diagram detailing the logic, in routine RTD.

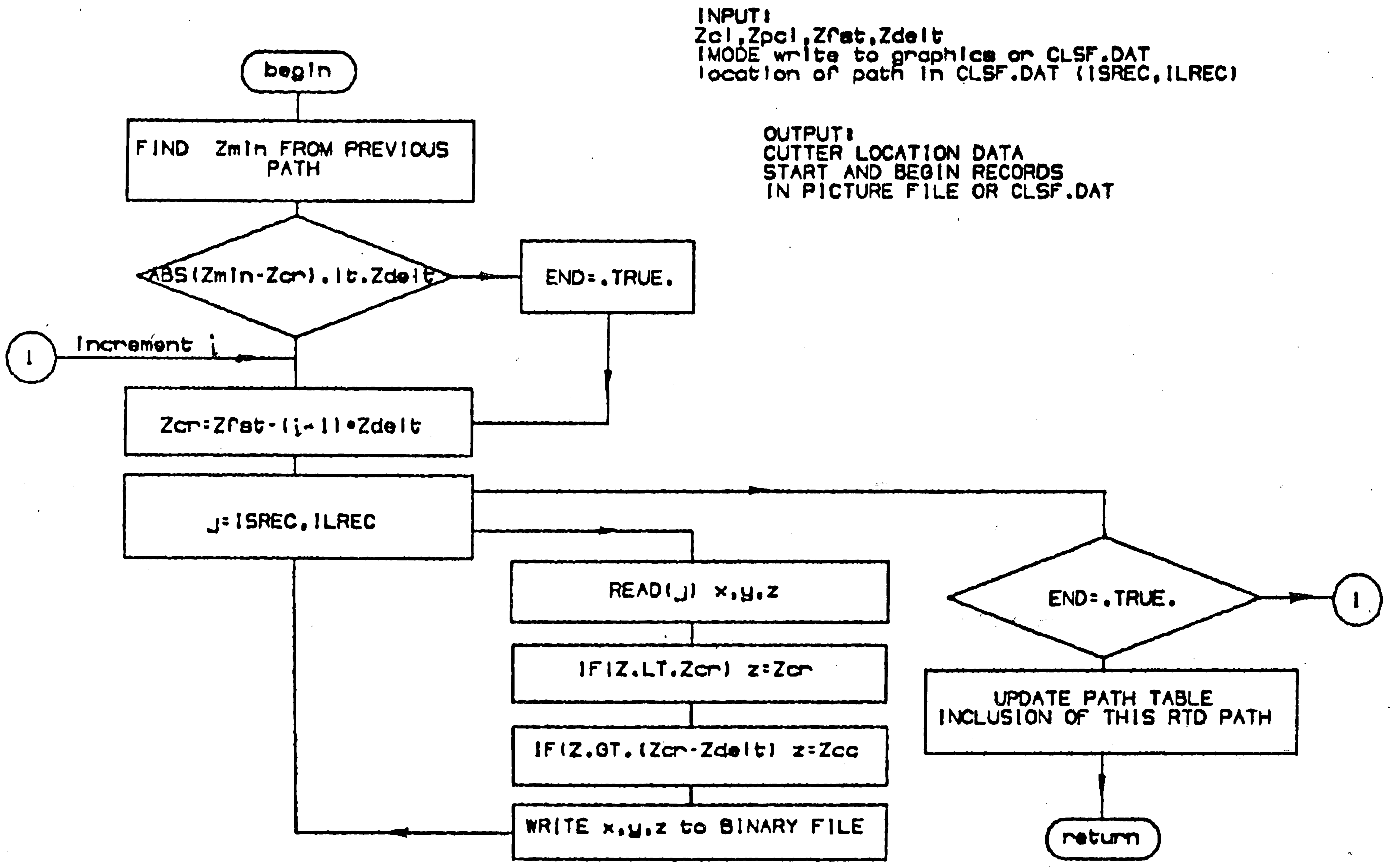


FIGURE 4-8. FLOW CHART FOR RTD CUTTING

RTD is more complicated by searching a point ahead to indicate whether the path is leaving the part surface moving to the clearance plane, the plotting font turns to a dashed line to easier visualize cutter motion. This flag will later be used to signify a change in feedrates since the cutter is on a clearance plane where rapid moves may be employed. A line printer plot of a surface with a RTD path is shown in figure 4-9.

4.5 DATA STRUCTURES

Four important data structures will be discussed, data in the blank common block, use of data in a large temporary array TEM, storage of input data on external file, CSAVESET and storage of NC path data in binary form on CLSF. The additional explanation of entity codes and how they are used in the picture file follows.

CASPA transfers data to the APT surface processor through the blank common. If a curve or a surface is input the data is placed in the blank common and pushed through the processing routines. The blank common is a one-dimensional array (5000 double precision entries) used as a data stream into either APT108 or APT107. The data for a curve or a surface is added to the COMMON block with header information contained in locations 40 to 48, and either storage location of a canonical form or geometric data specified from 49 on depending on the need. For example, if CURSEG or SPLINE type curves

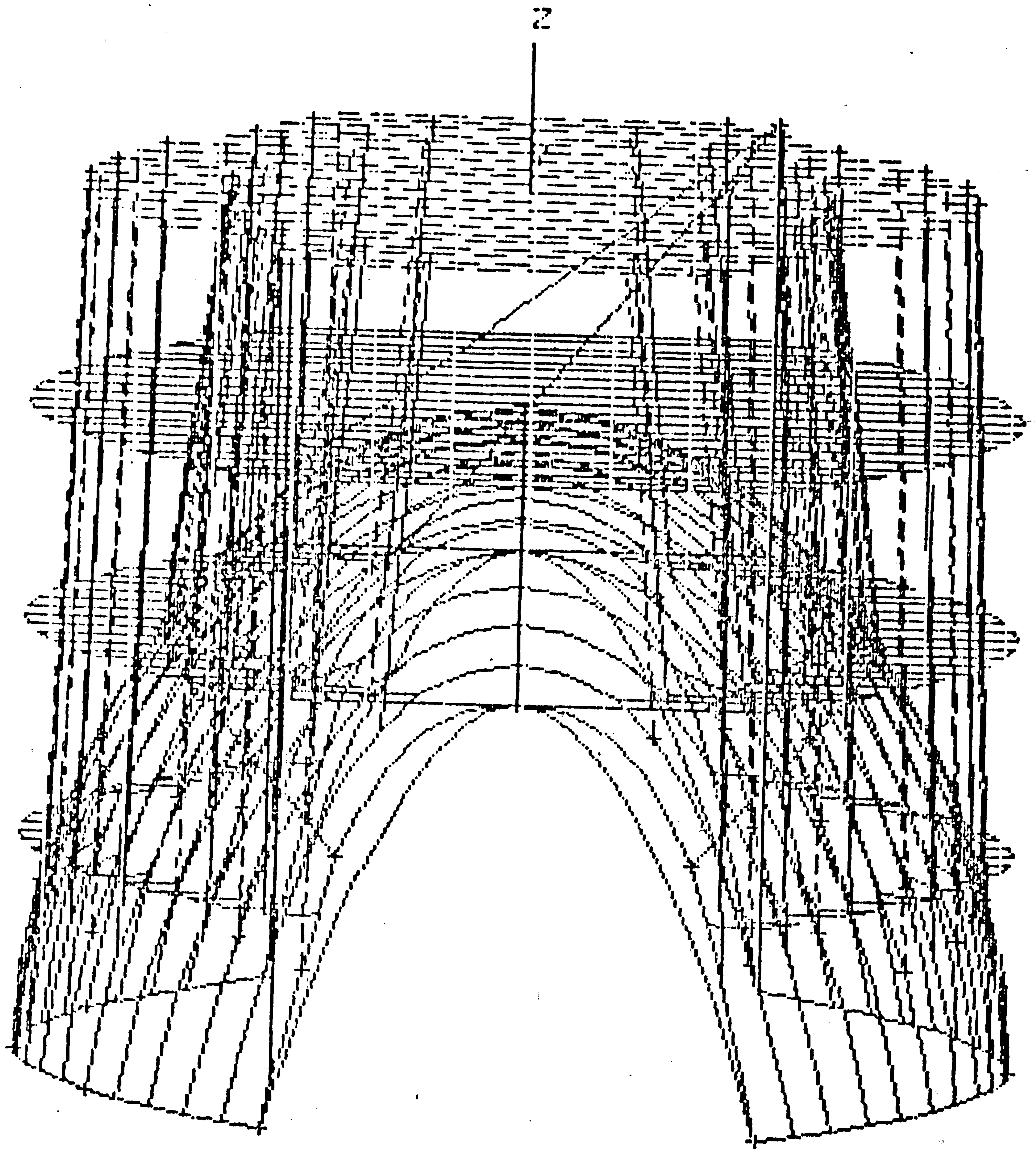


FIGURE 4-9. PLOT OF RTD PATH

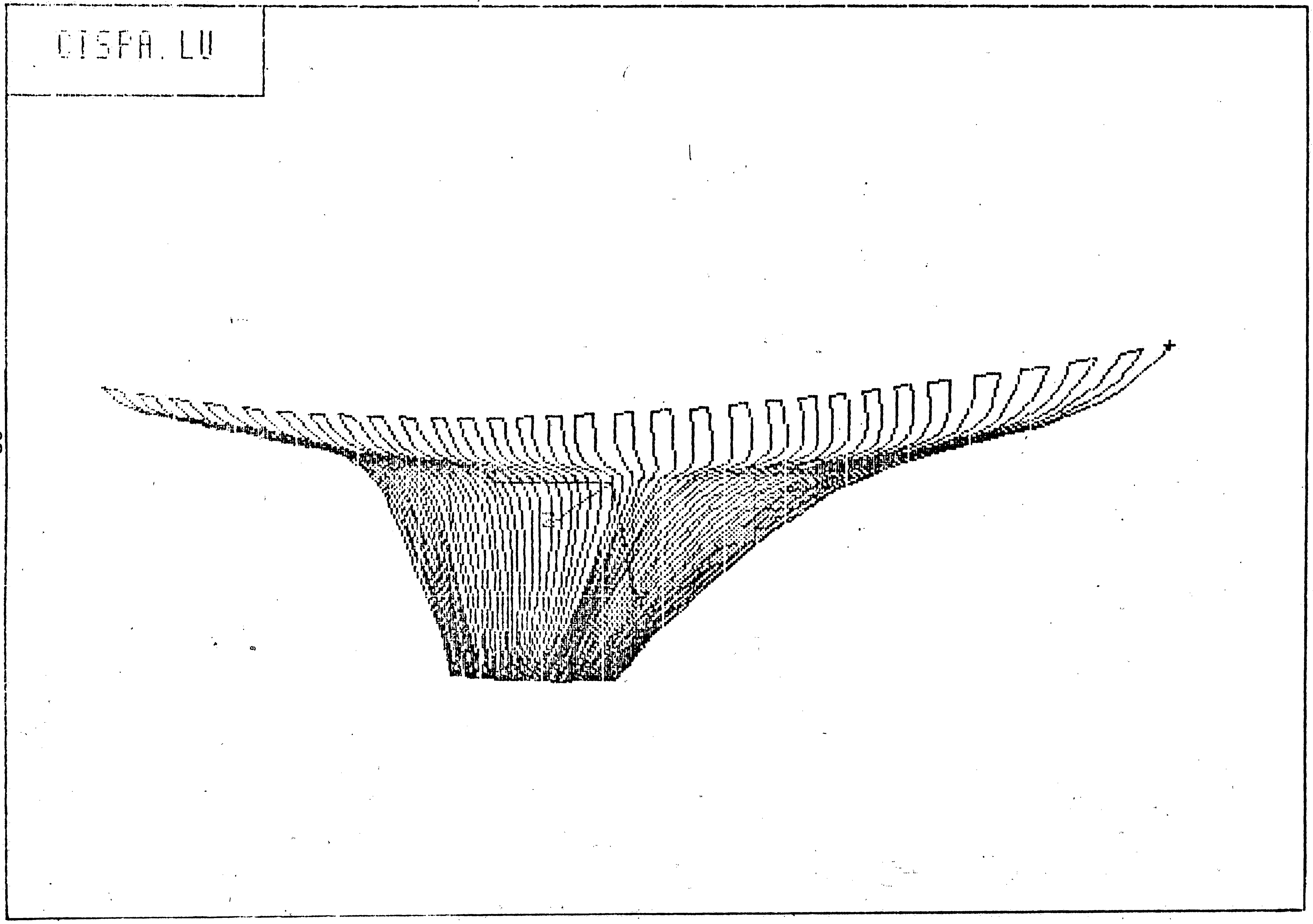


FIGURE 4-10. YACHT HULL

are chosen then blocks of data are stored for each point in the curve, these blocks comprise position coordinates, tangent, normal, cross-tangent, weighting or limit values depending on user specification. Each of the aforementioned value types has a APT class code connected with it, part of this code is used to specify what the following numerical values will be used for in upcoming calculations by the surface processor. In a COMBIN operation, which combines two curves with positional continuity, locations and size of each curve canonical form are placed in the common block. Generally, the information needed for surface generation is canonical form locations of curves building the surface. The notable exception being the MESH of curves type surface. Here a user inputs a rectangularly ordered set of N by M points in space which lie on the intended surface, splines are interpolated in each direction, finally parametric bi-cubic patches are calculated to fill the space between each grouping of four points, all achieved with C1 continuity. The data filling the blank common in this case are blocks of data for each spline in the primary direction. By understanding the format in which data should be placed into the blank common, freedom is gained to use a completely different input method, yet access all of the curve and surface definition types supported by CAM-I's SSX6A. Examples of different curve and surface definition as structured into the blank common, with descriptive notes are contained in Appendix B.

The TEM buffer is a temporary buffer holding information in much the same format as the common area. This buffer is, as of now, used strictly for curve data, it allows storage of the maximum amount of data for each point, that is all point constraints could be used and stored here, if certain constraints are not input a dummy value or default remains in place. This TEM buffer is necessary under the MODIFY option. The curve may be modified by changing point location or constraints associated with the point, though at the MODIFY stage no new points may be added or deleted. Once processing is decided upon this TEM buffer is condensed with applicable data read into the COM array which is in turn processed by the APT routines. See figure 4-11 for the structure of the TEM buffer. Note that for each vector constraint a couplet of information is needed-- a code and the vector itself. A code of 776. indicates that no vector follows and that the couplet of information should be skipped during the condensation operation in TEM_COM.

Curve input, as discussed above, is saved in CSAVESET.DAT and when retrieved shoved through the surface processor to obtain the canonical form and picture information. This is done so that the input may be graphically shown and, if need be, modified . Saving a canonical form stores none of this input data. By recomputing the canonical form certain arrays and flags are initialized. The computation of a curve canonical form is quite easy, actually more calculations are needed to create the digitized form of the curve for the picture file but this would have to be done anyway since saving the picture file is inefficient.

		NUMBER OF POINTS					
DESCRIPTION		1	2	......	19	20	
point number	1	1	2	......	19	20	690
APT code, point	2	19.					691
	3	x_1					.
coordinate	4	y_1					.
	5	z_1					.
ISUB code	6	ISUB					
NAME	7	TANGENT					
APT code, vector	8	20.					
	9	x					
direction	10	y					
	11	z					
ISUB code	12	ISUB					
NAME	13	CROSS					
APT code, vector	14	20.					
	15	x					
direction	16	y					
	17	z					
ISUB code	18	ISUB					
NAME	19	NORMAL					
APT code, vector	20	20.					
	21	x					
direction	22	y					
	23	z					
ISUB code	24	ISUB					
NAME	25	WEIGHT					
APT code, value	26	21.					
weight value	27	w					.
ISUB code	28	ISUB					.
NAME	29	LIMIT					.
APT code, value	30	21.					719
limit value	31	l					720

FIGURE 4-11. STRUCTURE OF THE TEM BUFFER

CSAVESET.DAT is an unformatted binary file, meaning that the data is transferred between main memory and the file with no conversion - the bit patterns representing the data are transfered completely intact. This I/O transfer is more efficient in terms of execution time, and usually in terms of file space required. This file starts with header information and then curve input is stored in sequence in much the same format as entailed by the COM space. Each record in this file is 8 bytes long, one double precision value. The first 51 values are used for the header, the first entry holding the number of curves saved and deleted. The next 50 records contain begin records for the curve data following the header, each curve number is equal to the header record number minus one. At initialization of the program, records 2 through 51 are read into array ICATALOG and serve as a catalog to locations of curve input on the external file CSAVESET.DAT, the index of ICATALOG serves as the identifying curve number. Each curve has a block of information with an identifying header, and exactly the information needed in the TEM or COM array to process the curve. SPLINE and CURSEG definitions keep the TEM form and COMBIN and FLOW definitions have the COM form. To retrieve curves, each saved curve name is listed and the user is asked to enter one or more names for retrieval, each of these names is linked with its curve number. ICATALOG is referenced, the curve number as the index and the start location on CSAVESET is obtained to read the data which will be processed to yield a canonical form and picture data. If a curve is deleted its ICATALOG value is made negative, flagging the fact that it need not be plotted and is no longer active. Being a simple data structure no provision is made to pack the data in CSAVESET, the deleted curve data is retained but is not accessible. Since ICATALOG has 50 double precision words, 50 saved or deleted curves may be stored, enough for many deletions during a session. The curve header information is simply: the curve name, the position of the next curve data block, and the type of curve (SPLINE,CURSEG, etc.). Pictorial representation of this structure appears in figure 4-12.

The other file created also unformatted contains NC path cutter locations. This file CLSF.DAT has a simple header and coordinate information following. The header starts with the total number of paths contained in the file and then four pieces of information for each record, the name of the path, the start and end records, and the surface name related to this path. Blocks of coordinate data follow after the 21 record header. As in the initialization of CSAVESET the header for CLSF is read into PCATALOG, a 20x4 integer array, and used to locate paths for ASCII text conversion or rough-to-depth cutting. At the end of the session this PCATALOG array, likely updated, will replace the old header. The coordinate information is the cutter location x,y,z, data with the point count occupying the fourth location.

The last topic of discussion is the entity code definition in the picture file. When curves, surfaces or the calculated NC path cutter locations are rendered for display the end points of the straight line segments representing the curve or surface grid are stored in the

1	NSAVE	number of curves saved and deleted
2	52	location of first curve saved
3	.	
.	.	
.	.	
.	.	
51	nnn	location of 50th curve saved
52	$NAME_1$	name of first curve saved
53	$NEXT_1$	next location of a saved curve end record of this data block = $NEXT_1 - 1$
54	$ARG1_1$	type of curve saved -- SPLINE, CURSEG, COMBIN, FLOW.
55	.	
.	.	
.	.	
.	.	
$NEXT_1-1$	$NAME_2$	name of second curve saved
$NEXT_1$	$NEXT_2$	next location of a saved curve
$NEXT_1+1$	ARG1	type of curve saved
.	.	
.	.	
.	.	
.	.	
.	.	

FIGURE 4-12. CSAVESET DATA STRUCTURE

picture file. The picture file contains three dimensional absolute coordinates. Every point in the picture file (the file plotted during a display operation) has an entity code, this code references the type of entity (point,curve,surface), the entity number and location of the entity in the TEM file if applicable. The entity code is three integers encoded into one number

NSS.TT

where

$$N = \begin{cases} 0 & \text{point} \\ 1 & \text{curve} \\ 2 & \text{surface} \end{cases}$$

SS = a two digit number identifying the entity, location of canonical form, many times refered to as the curve or surface number, points have an entity number of zero,

and TT = location in the TEM buffer, needed for modification of curves to re-structure inputs connected to the point, this value must be either 0 or 3 + 31n where n= 0,1,2,3,....,N , N is the number of points in a curve.

The picture file or graphics point file saves three dimensional points and also instructions for generating certain strokes (solid lines, dashed lines, dashed points or just points) plus allowing input of a symbol at the point, as well as the entity code explained above. The field is formatted as

F1 = graphics point stroke control
(see reference 15)

F2-F4 = x,y,z coordinates

F5 = a symbol or four character word to
centered at x,y,z

F6 = entity code.

The field F5 is not used at this time in the interactive mode, however, while in CASPA symbols or words may be input into the graphics point file.

5. PROGRAM STRUCTURE/PROGRAM OPERATION

5.1 INTRODUCTION

A general description of program operation/structure is presented in this chapter.

First, a brief discussion of the hardware running CISPA, the host computer and display terminals. Secondly, a description of the various outside software that has been incorporated into CISPA. Finishing with an overview of the program operation and structure with some discussion of the major modules embodied in CISPA.

5.2 HARDWARE

Currently, CISPA runs on a host computer, the VAX 11/780 (VAX is a trademark of Digital Equipment Corporation), with 32 bit addressing and a virtual memory operating system, eliminating the need for overlay organization. Backing storage is accomplished by magnetic disc. A magnetic tape drive is available for back-up operations.

The VAX 11/780 supports may engineering applications programs and claims to support as many as 100 interactive users. CISPA may also be run on the Micro VAX I, a 32 bit machine with Micro VMS operating system. The Micro VAX I is physically much smaller then the VAX

11/780 (fits under a desk) and is a fraction of the cost of the larger VAX ($20 000 compared to .5 million).

The graphics may be displayed on either a DEC VS11 or Tektronix 4107 terminal. The VS11 is a high performance color graphics system with joystick assembly, supported by the VAX 11/780. The terminal uses a 60 hz. raster scan display with a 512x512 pixel resolution. The raster scan technology permits selective erasure and a rapid refresh rate lending itself to an interactive environment. As discussed in the 'VS11 Installation Guide' (21), the basic components of the system are: a high speed display processor, image memory modules and sync generator/cursor controls. By using a high speed microprocessor as the display processor and an image memory to store pixel data output, the need for storage of a image data file in the host central processing unit (CPU) is eliminated.

The Tektronix 4107 terminal has about the same resolution (640x480) and also incorporates a 60 hz. raster scan display, but is comparatively inexpensive and based on current technology, see Tektronix 4107/4109 Programmers Reference (22) for more information.

5.3 SOFTWARE

The graphics driver used in CISPA is called by GRAPH3D, permitting display on both of the terminals mentioned above. This package formerly VS113D, contains basic plotting, elementary and advanced graphics routines, transformations for real time motion and interactive input. These graphics routines may be called as subroutines once the GRAPH3D library is properly linked.

The routines in this package are divided into groups as:

1. initialization routines- initialize VS11 or 4107 driver, set window, set viewport, and display window borders,
2. basic drawing routines- to draw points and lines,
3. high level drawing routines- to draw circles, arcs, symbols and display the absolute coordinate axes,
4. alphanumeric data display routines- to display alphanumeric data,
5. model orientation routines- to rotate, translate, scale the user-defined pictures,
6. display control routines- to change colors, erase, blink pictures and dump the display file to the terminal,
7. user-computer interactions- allows user-computer interactions with the tracking cross shaped cursor and keyboard.

The routines specifically used are lower level graphics routines permitting plotting of points and symbols, drawing of lines and cursor control. More information on GRAPH3D may be found in the reference manual 'GRAPH3D Graphics Package' (16).

Software borrowed from the CAM-I CASPA package included the routines associated with curve and surface definition, reading or writing of the canonical form and inclusion of the name table and name search routines. Refer to figure 5-1 for the subroutine structure of CISPA.

Most of the routines from CASPA are taken directly from the APT4 execution phase. The sculptured surface and synthetic curve portion form most of these routines, falling generally under APT107 and APT108.

APT107 is the main routine for definition of sculptured surfaces. This routine recieves input mostly in blank common and some information in the calling arguments. The routine re-organizes and checks input data to some extent and then calls programs for each sub-type of surface definition. Upon specification of the patch coefficients, the topology table is completed by a call to TOPGEN, the surface canonical form data can be displayed by a call to SSPICT and its stored and saved by a call to APT094. Note figure 5-2 for subroutine structure under the Sculptured Surface Definition processor.

Synthetic curve definition is accomplished by APT108 yielding a synthetic curve canonical form. The majority of input is recieved in the blank common. The program first checks for FLOW/SEGMENTATION data in the input stream (23). If FLOW data is encountered it is structured into a buffer area, checked for validity, and removed from the input stream. Next simple curve definitions (CURSEG and SPLINE) are processed, this includes calls to SCURV to process a SPLINE or CURSEG definition. A COMBIN definition is processed within APT108 using APT094 to fetch input curves, and CNCURV to validate matching

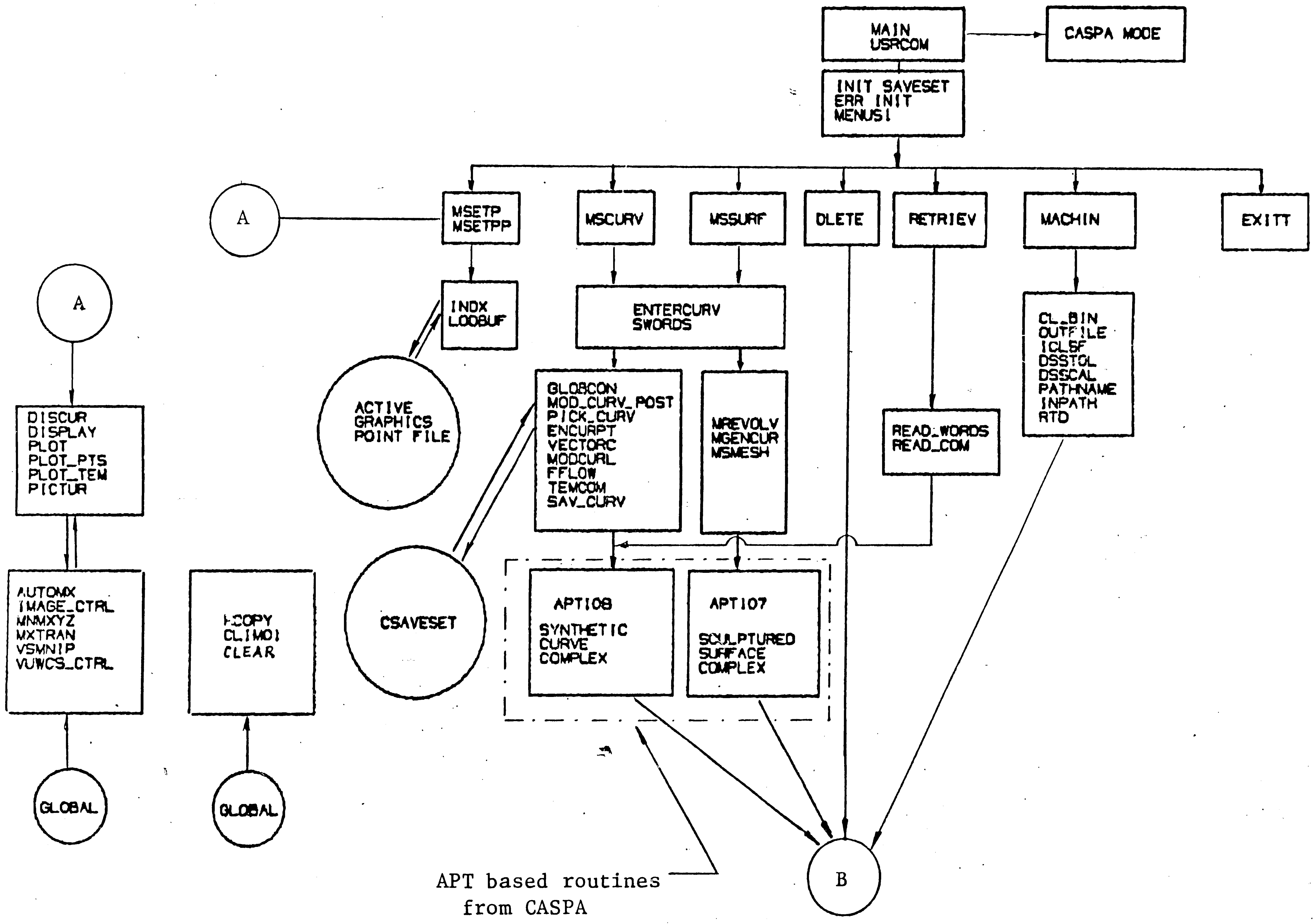


FIGURE 5-1. SUBROUTINE CALL STRUCTURE FOR CISPA

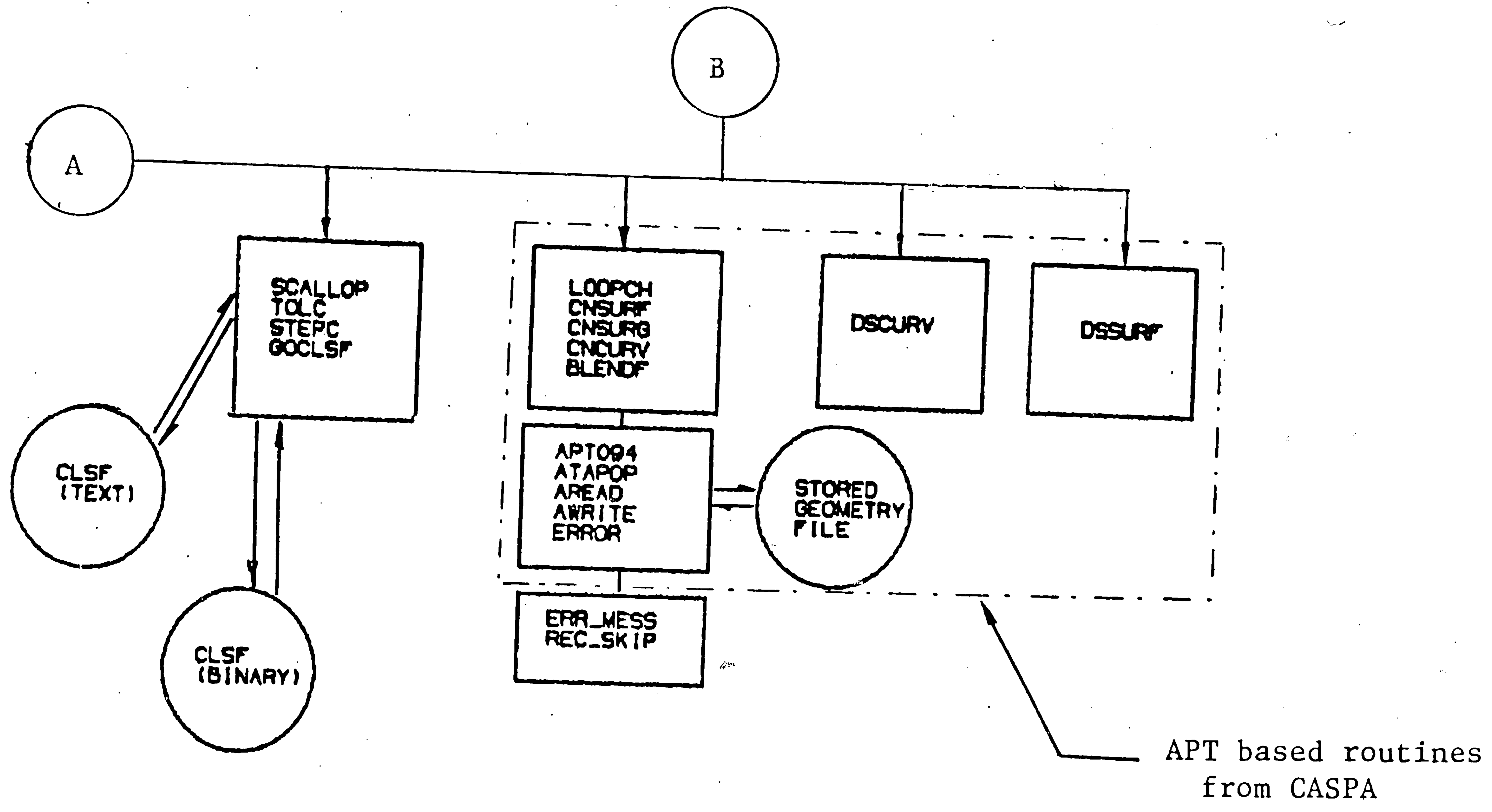


FIGURE 5-1 CONTINUED. SUBROUTINE CALL STRUCTURE FOR CISPA

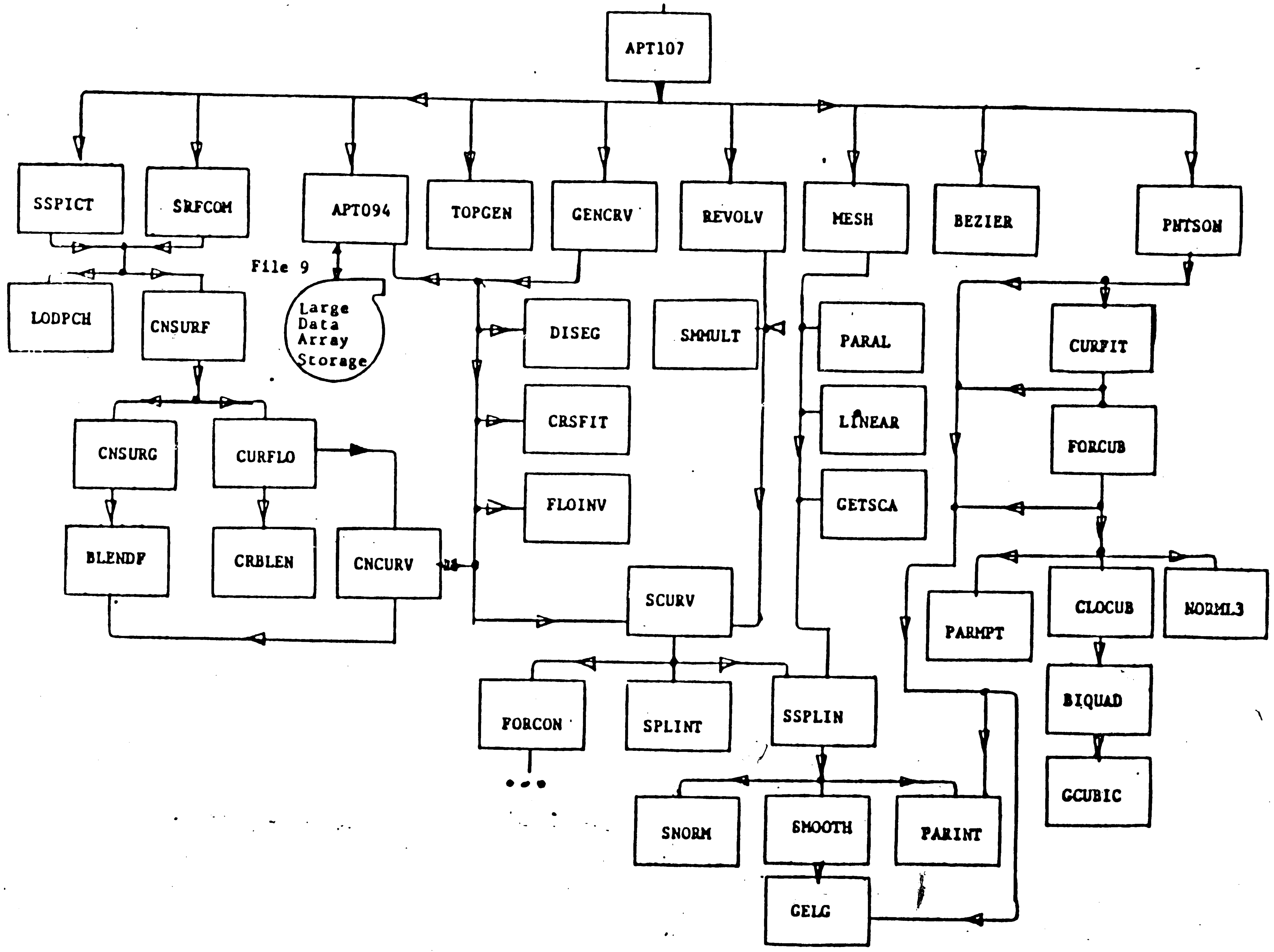


FIGURE 5-2. SUBROUTINE CALL STRUCTURE FOR APT107 (REFERENCE 14)

conditions at junction points. Upon completion of simple curve processing any specified flow rates are imposed on the simple curve and the entire curve data structure for FLOW and SEGMENTATION is constructed in APT108. Help from routines ANGSEG, ARCSEG and CHDSEG in determining parameterization based on angular, arc or chordal rates respectively, may be necessary. The synthetic curve is verified by a call to SCPICT and saved on external file by a call to APT094. Figure 5-3 shows the interrelationship between routines making up the synthetic curve processing complex.

The naming of curves and surfaces is retained by using the same name table and name search routine SWORDS with changes made appropriately as discussed in Chapter 2. The SWORDS routines is not associated with APT4.

Reading and writing of canonical forms, as mentioned earlier, is managed by APT094. Curves and surfaces are digitized for picture representation by DSCURV and DSSURF respectively. These two routines are not included in the APT4 processor. The point data generated by these routines along with the graphics codes are dispatched to the picture file by a call to SETP.

5.4 MAJOR MODULES - CISPA

At this point in time, CISPA consists of four major software functions:

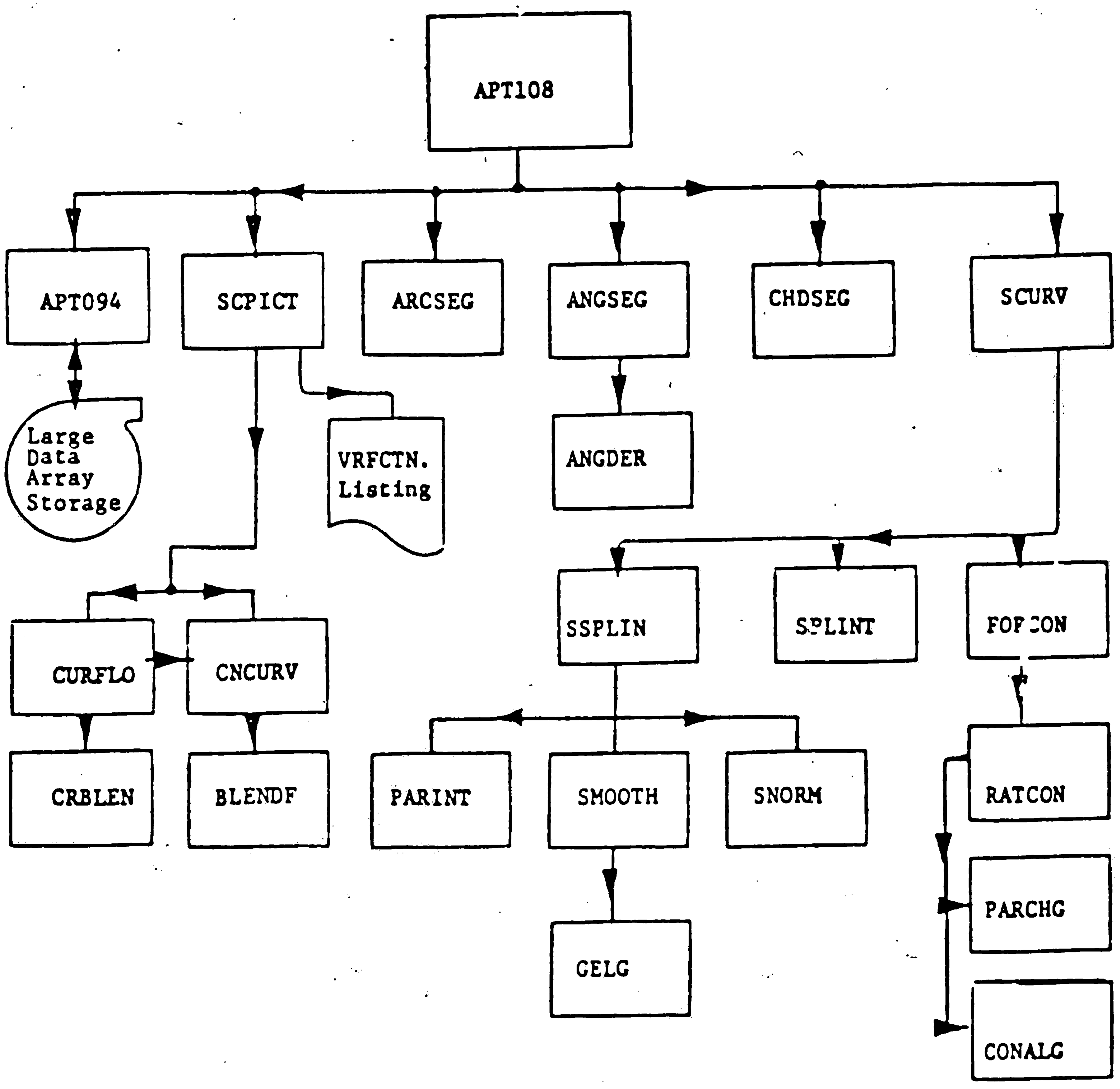


FIGURE 5-3. SUBROUTINE CALL STRUCTURE FOR APT108 (REFERENCE 14)

1) geometry creation,

2) entity management,

3) cutter path generation,

4) display and view control.

Geometry creation contains the whole of the APT sculptured and synthetic geometry complex, plus the front end routines needed to dispatch data in an intermediate form to the blank common for manipulation by the APT complex. The two major modules belonging to this front end are MSCURV and MSSURF. They both provide a translating/switching function. MSCURV displays menus guiding the user to definition of the curve types supported by APT4. Calls are made to GLOBCON to establish global constraints (weights and limits) (23) on all subsequent point input. Simple curves are defined by points entered through ENCURPT using absolute coordinates, existing point or screen position. VECTORC is called if vector constraints are desired at these points. COMBIN curves are managed in MSCURV with use of PICKCURV to select curves used in the combine function. Simple curves may be modified with a call to MODCURL, before processing the input and MOD_CURV_POST is called if modification of simple curves are desired after processing. FLOW and SEGMENTATION input is realized through the routine FFLOW. Curve data in the temporary buffer, TEM is read into the common blank before a call to APT108 to start synthetic curve processing. Depending on whether the curve is successfully defined the curve is saved on external file CSAVESET.DAT on logical unit 11 by executing routine SAV_CURV.

The surface input translator calls routines for the surface sub-type desired. The general strategy in this routine is to pick curves forming the basis of the surface and dispatch the locations of the canonical forms to the proper places in the common blank.

Entity management, may more aptly be named curve management. The two major modules used are RETRIEV and DLETE. The routine RETRIEV serves the function of retrieving a curves input in intermediate form, sending it through the synthetic curve processor thus obtaining the picture data for display. Calls to READ_WORDS which parses a string of characters to obtain the list of names to retrieve, and READ_COM to read the input from external storage accomplish these actions.

The deletion of points and curves is executed by DLETE. This routine allows selection of point or curve by cursor so a flag may be set indicating that the canonical form is undefined. The entity is also unplotted and flags are set in a table holding record locations in the picture file used for selective plotting.

Cutter path generation is headed by routine MACHIN which duties include saving or deleting the unformatted CLSF (cutter location source file), writing this file into a a standard format, such as an APT file, allowing specification of machining parameters, such as, tool radius,step size, cutter offset, etc. and parameter information used in rough-to-depth cutting. Important routines called are DSSTOL for calculation of variable step size in the primary direction based on chordal deviation, DSSCAL, calculation of the cutter side step based on scallop height, and RTD determining the cutter locations at user specified cutting planes. Plotting functions are accomplished by calls to DISCUR, DISPLAY and PICTUR once the entity is digitized and view control is not necessary. Accessing the global menu by hitting the 'G' key switches to routine VSMNIP. This routine serves as a switch to transfer the control to the routines for display and view manipulations under the GRAPH3D package. These routines provide full control of view and image manipulations in a specified window. They have recently been made accessable through the GRAPH3D package for application oriented programs. The functions they provide are discussed in detail in the 'CISPA.LU Package Manual' (11). The routines HCOPY, CLIMO1, and CLEAR are routines from the GRAPH3D package (16).

5.5 PROGRAM OPERATION

The following presentation is a brief overview of program operation, see (11) for more information.

The two types of program operation are the original CASPA command mode and the newly implemented menu-driven mode. The command mode utilizes keywords which are matched to a symbol table to determine the action taken (15). In the menu-driven mode, a menu of available options is presented to the user at every decision node. The user selects an option through a key hit specifying an option number or special action. Display and option selection from the active menu or global menu is controlled by subroutine FINDOP. FINDOP calls routines GLOBME, to switch control to the global menu and READC1 which reads the key input. The global menu contains options that can be accessed at any time during the menu-driven mode of operation. Global options such as redraw, view control, image control, blanking/unblanking, clear graphics and hardcopy may need to be executed frequently. Appendix C, contains a menu structure detailing what has been explained here in words.

Geometry creation is divided into point, curve and surface creation. Points are defined by entering the absolute coordinates, x,y,z, in units of inches.

Curves are input, by specifying whether or not global constraints are imposed on succeeding point input. Points are then entered by selecting existing points, entering absolute coordinates, or screen position. These points should be input in a certain order, from one end of the curve to the other. Point constraints are imposed by inputing direction and chosing tangent, cross or normal vector options. In addition, weight and limit constraints may be entered controlling the local nature of the curve at the chosen point. The CURSEG definition generates a conic through the given input points and imposed constraints. The SPLINE definition fits a cubic polynomial through points positioned in 3-D space. The COMBIN option combines the two previously mentioned simple curve types under one name. The FLOW option allows parameterization of an existing synthetic curve on the basis of arc length, chord length, angle or Coons parameter (existing parameterization). Finally, selecting MODIFY adds flexibility to the design session by viewing curves and then modifying point postion and constraint data to suit the users needs.

The surface creation option supports ruled surfaces, surfaces of revolution, free-form doubly curved surfaces (GENCUR) and mesh of points type surfaces.

By chosing the entity management option, the retrieve or delete modules may be chosen. All retrievable curves will be listed, where the curve name is entered to access the stored input. All curves are stored in CSAVESET.DAT, the user is asked upon exiting the interactive mode, whether the session should be saved. Based on the user response then CSAVESET.DAT is saved or deleted.

Both points and synthetic curves may be deleted by selecting from display or by name. Both points and curves will be erased and made unselectable in the picture file.

Under NC path generation, machining parameters may be input, as well as parameters used for RTD machining. File handling of CLSF in binary form allowing saving, deletion or writing this file to an ASCII text file. This file may later be postprocessed according to the NC machine used.

6. CONCLUSION

6.1 SUMMARY

CISPA is a complete interactive CAD/CAM package where one may define a surface, generate a tool path, and create the cutter location source file needed to mill the surface using an NC machine.

Since the scope of such a project is large software in the public domain was sought with plans to modify the software to suit the projects needs. An evolution of CAM-I's sculptured surfaces program was presented and CASPA was chosen based on its cost, ease of implementation and support of a wide range of curves and surfaces. Limitations were identified in CASPA and addressed after CASPA had been ported from the IBM 360/370 to the CAD lab's VAX 11/780.

A presentation of curve and surface mathematics was included in Chapter 3. The mapping of points in Euclidean space to projective space was found useful in applying 3D transformations to geometry as well as representing 2D conics as a parametric cubic rational form. Fitting splines to a stream of points positioned in three dimensional space was presented as well as, a particular case of the general Coons surface patch, the bi-cubic patch. The importance of a one parameter definition for curves and a two parameter formulation for sufaces became obvious because:

- it enables twisted space cuves in three dimensions to have a simple mathematical form,
- it avoids problems which arise when representing curves with vertical tangents in a fixed coordinate system,
- it enables coordinate transformations such as translation, rotation, or perspective to be performed,
- parameterized forms are easily digitized for display.

In other words, parametric forms allow independance from a particular coordinate system. This presentation is applicable in that the APT complex uses these definitions in representing curve and surface geometry.

Particular algorithms of interest were presented next, with special attention paid to picking of entities by cursor and certain attributes of the interactive menu-driven front end, and algorithms pertaining to step size based on geometric attributes rather than on abstract algebraic quantities. This chapter exhibits much of the original work done on CISPA. Data structures designed to facilitate transfer to intermediate form used in processing, saving curves and holding temporary input was explained to help the reader better understand CISPA's data allocation.

Program structure/operation constitute discussions on hardware, software, program structure and operation. The VAX 11/780 - VS11 configuration give good response for the complicated calculations entailed by surface computation. The VS11 and Tektronix 4107 both raster scan displays allow selective erasure and rapid refresh, well suited for the interactive philosophy used in CISPA. The main outside sources of software are from the previously discussed CASPA and GRAPH3D. It is pointed out that most of CASPA is a group of APT routines computing the curves and surfaces. It is this group which is used in CISPA. GRAPH3D is responsible for the graphics portion of CISPA. Program structure and operation are discussed briefly, see 'CISPA Package Manual' (11) for a more thorough presentation.

6.2 LIMITATIONS

The most limiting aspect of CISPA is in the definition of boundary curves to create surfaces. This is most evident when design iteration is heavily used. To modify a surface one must first modify boundary curves and then process the surface given the changed network of boundary curves. Clearly a more efficient method is to modify the surface directly. To do this in a geometrically intuitive manner, use of the Bezier polygonal structure, discussed in Chapter 1, may be introduced even though using the Coon's bi-cubic patch. Manipulation of the control vertices of this structure, possibly in real time (neccesatating new display hardware) would add greatly to the design process.

One drawback in the use of the bi-cubic patch is the absence of local control of surface geometry, although the use of weights and limits in curve definition help control aesthetic attributes such as 'smoothness' and 'fairness'.

Use of symmetry or similarity in the construction of boundary curves is currently not being used to advantage. For example, translating, rotating, and scaling curve shapes make much less input necessary and are needed elements for many 3D representations.

Blanking, saving and deleting surfaces should be implemented. Currently, the primary cutting direction is set by default. For milling purposes it is sometimes desirable to make this user-defined.

6.3 FUTURE STUDY

Since the '3D Space Curves used in NC Machining' project will be continued, much of the previous section will be implemented.

Blanking surfaces, curves, and tool paths with more efficient methods for selecting surfaces will be implemented in the near future.

Other future plans include:

- transformation of curves allowing translation, rotation and scaling of curves speeding surface definition and giving more accuracy to input sessions,
- saving of surface canonical forms and input information,
- deleting of surfaces
- utilizing a ring type data structure for more efficient storage of input and canonical form data,

- surface machining with user-defined primary direction indicated by graphics symbols,
- command mode to menu-driven mode data transfer (data compatability in both directions),
- command file creation- automatic command file created, upon request, as user moves through menu-driven mode,
- contours of Gaussian curvature, displayed in color, if desired,
- utilization of the Bezier polygonal structure for doubly-curved surfaces for more intuitive modification of surfaces.

REFERENCES

1. Groover, M.P. and Zimmers, E.W., "CAD/CAM Computer-Aided Design and Manufacturing", Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1984.
2. Pao, Y.C., "Elements of Computer-Aided Design and Manufacturing", John Wiley and Sons, Inc., 1984.
3. Pressman R.S. and Williams J.E., "Numerical Control and Computer-Aided Manufacturing", John Wiley and Sons, Inc., 1977.
4. Ferguson, J.C., "Multivariable Curve Interpolation", Journal ACM, vol. II, no. 2, pp.221-228, 1964.
5. Coons, S.A., "Surfaces for Computer Aided Design of Space Forms", Report MAC-TR-41, Project MAC, MIT, 1967.
6. Schoenberg, I.J., "Contributions to the Problem of Approximation of Equidistant Data by Analytical Functions", Q. Appl. Math, vol. 4, pp.45-99;112-141, 1946.
7. Rogers, D.F. and Adams, J.A., "Mathematical Elements for Computer Graphics", McGraw-Hill Book Co., 1976.
8. Gordan, W.J. and Riesenfeld, R.F., "Berstein-Bezier Methods for the Computer-Aided Design of Free Form Curves and Surfaces", Journal ACM, vol. 21, no.2, pp.293-310, 1974.
9. Faux, I.D. and Pratt, M.J., "Computational Geometry for Design and Manufacture", John Wiley and Sons, New York, 1979.

10. Foley, J.D. and Van Dam, A., "Fundamentals of Interactive Computer Graphics", Addison-Wesley Publishing Co., 1983.

11. Loney, G.C., Ozsoy, T.M. and Smith, R., "CISPA.LU Package Manual", CAD Laboratory, Department of Mechanical Engineering and Mechanics, Lehigh University, 1985.

12. CAM-I, "Sculptured Surfaces SSX2 Release", Publication No. PS-71-SS-06, Computer Aided Manufacturing International, Inc., Arlington, Texas, 1971.

13. CAM-I, "Consolidation of User Documentation for the CAM-I Sculptured Surfaces SSX6, SSX6A, and SSX7 System Releases", Publication No. PS-77-SS-06, Computer Aided Manufacturing International, Inc., Arlington, Texas, 1977.

14. CAM-I, "Sculptured Surfaces SSX8X Release Documentation", Publication No. PS-78-SS-02, Computer Aided Manufacturing International, Inc., Arlington, Texas, 1978.

15. CAM-I, "Documentation for Computer Aided Sculptured Pre-APT System (CASPA.03)", Publication No. PS-78-SS-01, Computer Aided Manufacturing International, Inc., Arlington, Texas, 1978.

16. Ozsoy, T.M., Bhalla, S., Summer, R., and Tress, D., "GRAPH-3D Graphics Package", CAD Laboratory, Department of Mechanical Engineering and Mechanics, Lehigh University, 1985.

17. CAM-I, "Mathematics for Representing Conics as Coon's Rational Parametric Cubic Curves", Publication No. R-75-SS-04, Computer-Aided Manufacturing, International, Inc., Arlington, Texas, 1975.

18. Forrest, A.R., "On Coons and Other Methods for the Representation of Curved Surfaces", Computer Graphics and Image Processing, vol. I, pp. 341-359, 1972.

19. Digital Equipment Corporation, "VAX-11 FORTRAN Language Reference Manual", Digital Equipment Corporation, Maynard, MA., 1979.

20. Froberg, Carl-Erik, "Introduction to Numerical Analysis", 2nd edition, Svenska Bokforlaget Bonniers, 1969.

21. Digital Equipment Corporation, "VS11 Installation Guide", Digital Equipment Corporation, Maynard, MA., 1979.

22. Tektronix, Inc., "4107/4109 Programmers Reference", Tektronix, Inc. Beaverton, OR., 1983.

23. CAM-I, "CAM-I Sculptured Surfaces Users Course", Publication No. TM-77-SS-01, Computer Aided Manufacturing International, Inc., Arlington, Texas, 1977.

SUPPLEMENTARY REFERENCES

Wagener, Jerrold L., "Fortran 77, Principles of Programming", John Wiley and Sons, 1980.

Middlebrooks, Charlotte, "VAX Fortran", Reston Publishing Co., Reston, VA., 1984.

Barnhill, R.E. and Boehm, W., "Surfaces in Computer Aided Geometric Design", North Holland Publishing Co., 1983.

McAUTO, "Graphics Machining Module", McAUTO, St. Louis, MO.,1980.

Kreysig, E., "Advanced Engineering Mathematics", John Wiley and Sons, 1979.

Giloi, W.K., "Interactive Computer Graphics", Prentice-Hall, Inc., Englewood Cliffs, NJ., 1978.

APPENDIX A. CASE STUDY

CASE STUDY

A brief example follows showing the steps needed to create a surface using the prototype software. The part modeled may be a molding for a cover to some mechanism and is provided to show the ease of inputing doubly curved surfaces. Reference 11 'CISPA Package Manual' contains two, more detailed examples.

1. Enter the CISPA program by picking the option MENU DRIVEN MODE and create ponts under GEOMETRY CREATION, POINT CREATION (Figure A1). The view may be changed by utilizing view manipulations under the global menu.
2. Synthetic curves are fit in the primary direction using the SPLINE definition, under the synthetic curve option, and inputing the proper tangential directions as constraints (Figure A2).
3. The curve structure is completed by defining synthetic curves in the secondary direction. The two end boundaries are a combination of spline and straight line segments (Figure A3).
4. These curves are re-parameterized using the FLOW option (attachment of a flow structure) and the primary curves and

secondary curves are selected for the GENCUR surface definition under SURFACE CREATION.

5. Backstepping to the main menu the machining module is called and an NC path is generated with surface tolerance in the primary direction, .01-in., and a scallop height of .001-in.. The radius of the ballend mill is input as .25-in. (Figure A5).
6. The path is written to an ASCII text file for post processing depending on the type of NC machine used.

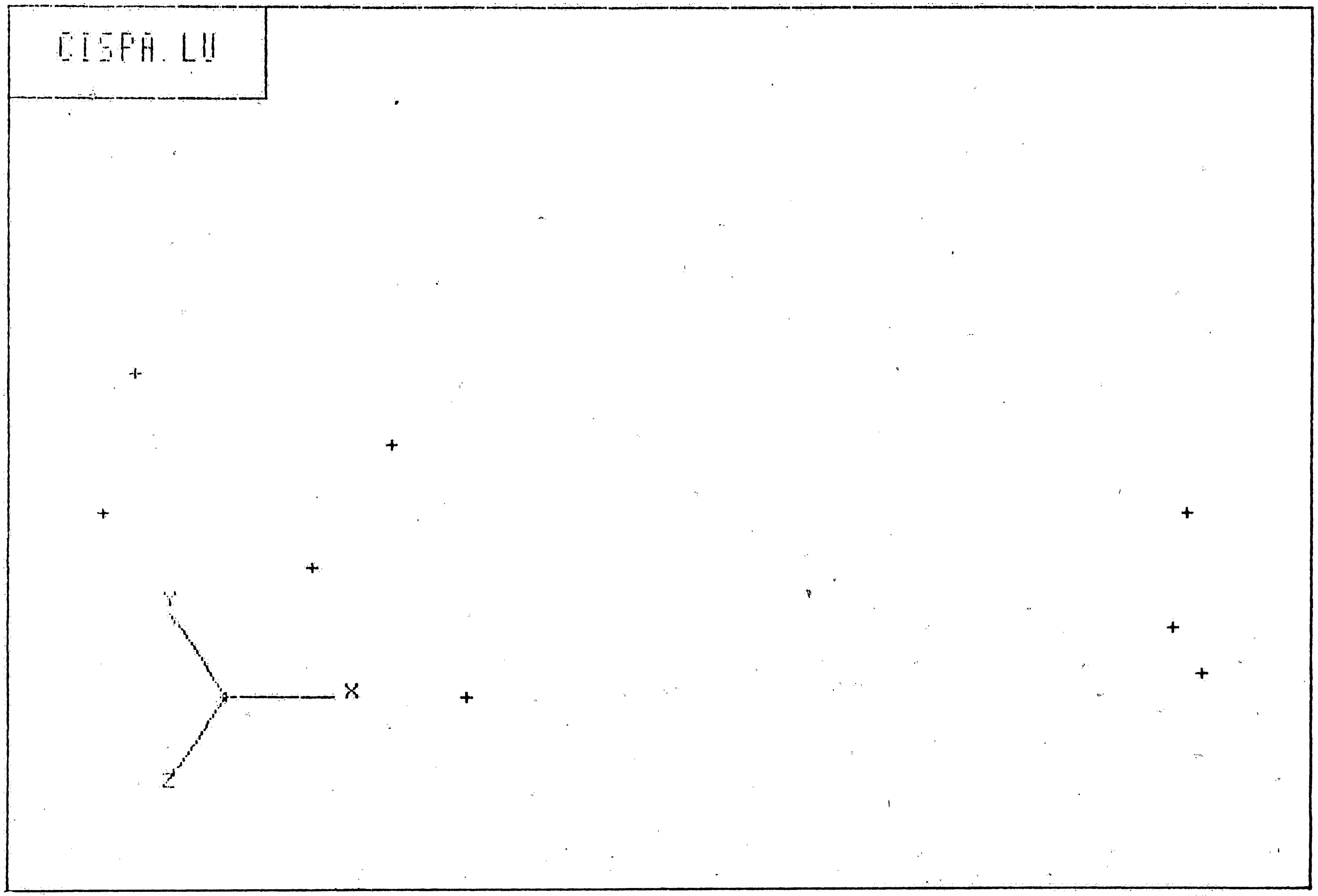


FIGURE A1. POINT INPUT FOR CASE STUDY

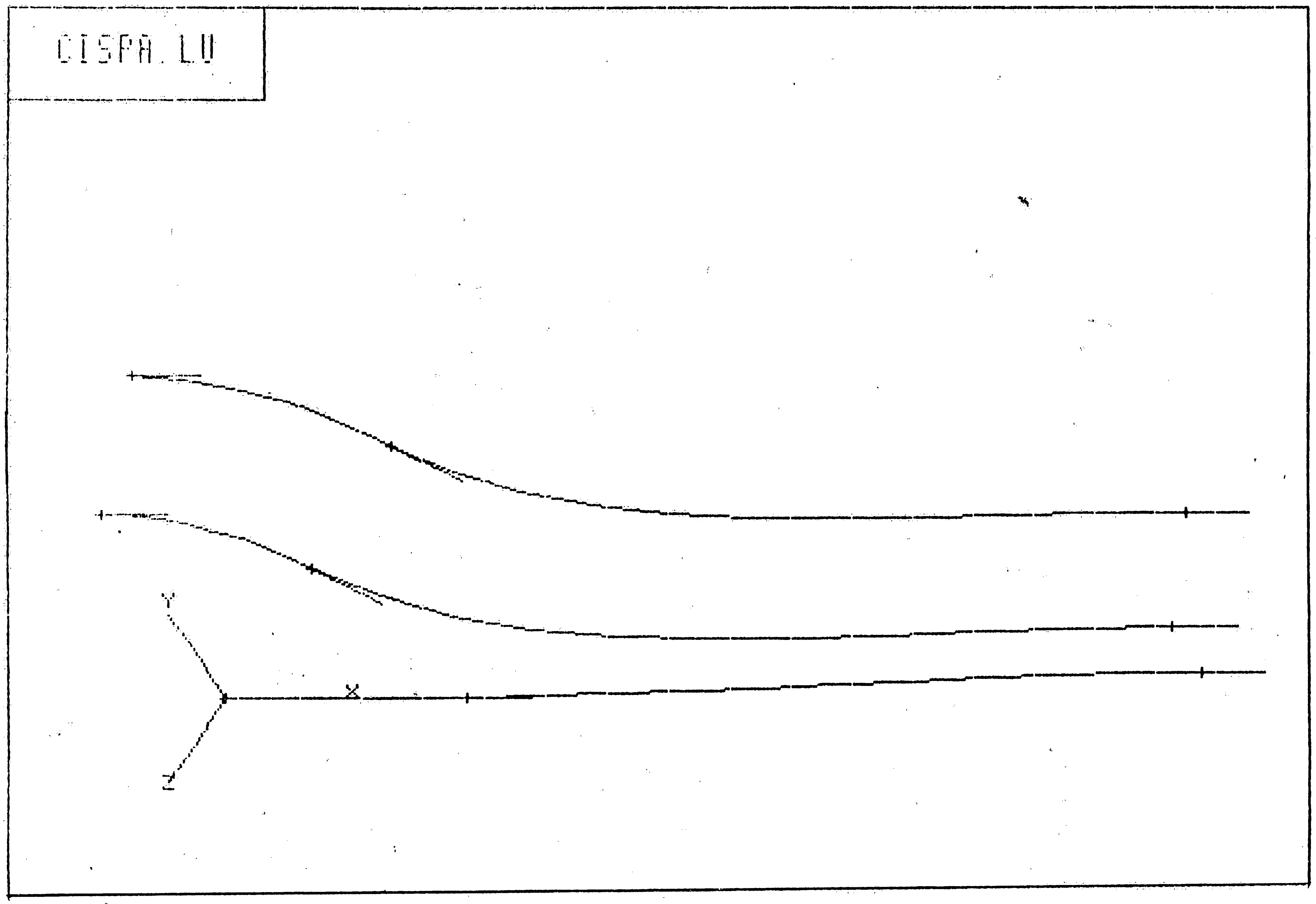


FIGURE A2. PRIMARY CURVES WITH TANGENT CONSTRAINTS

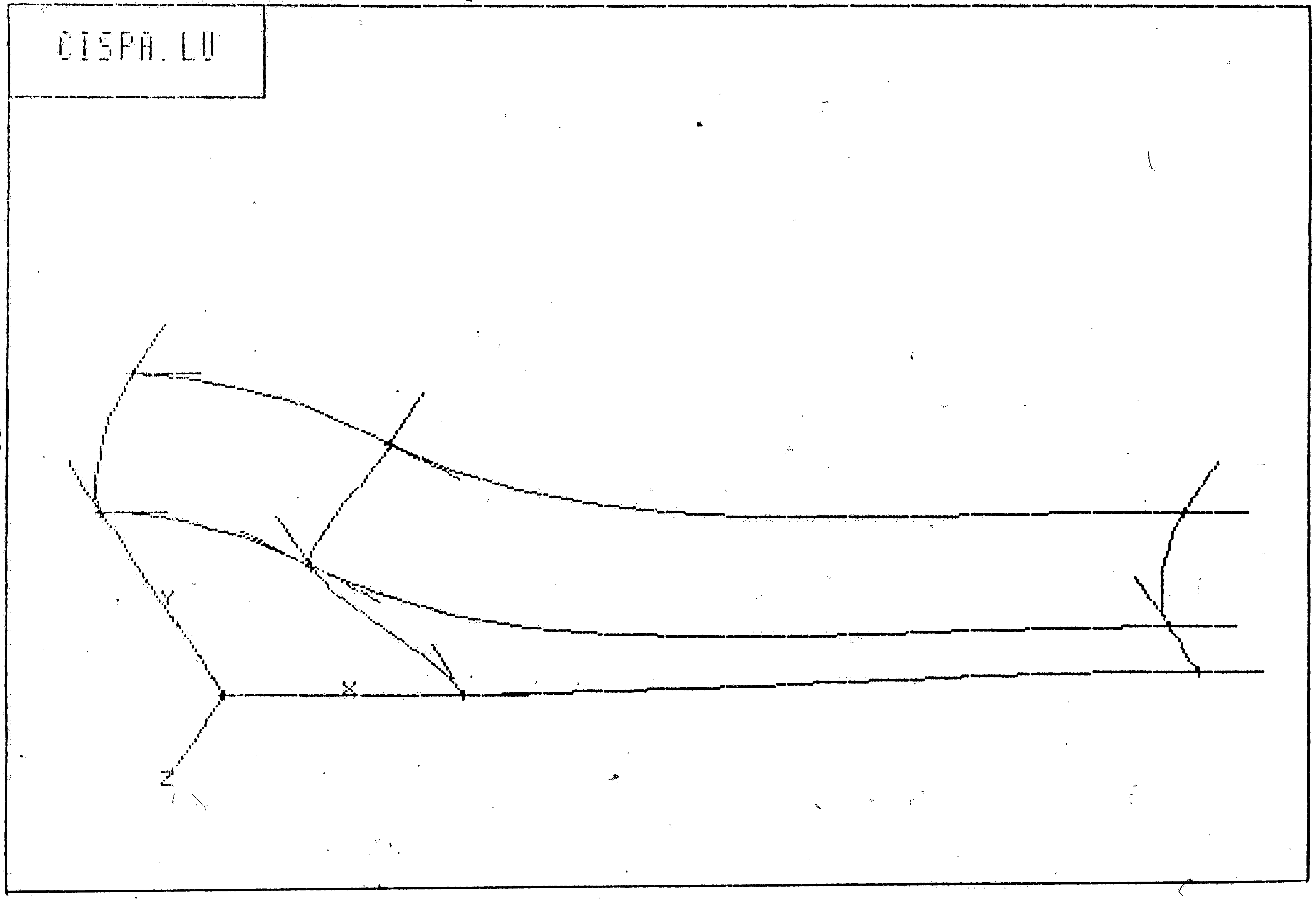


FIGURE A3. PRIMARY AND SECONDARY CURVES WITH CONSTRAINTS

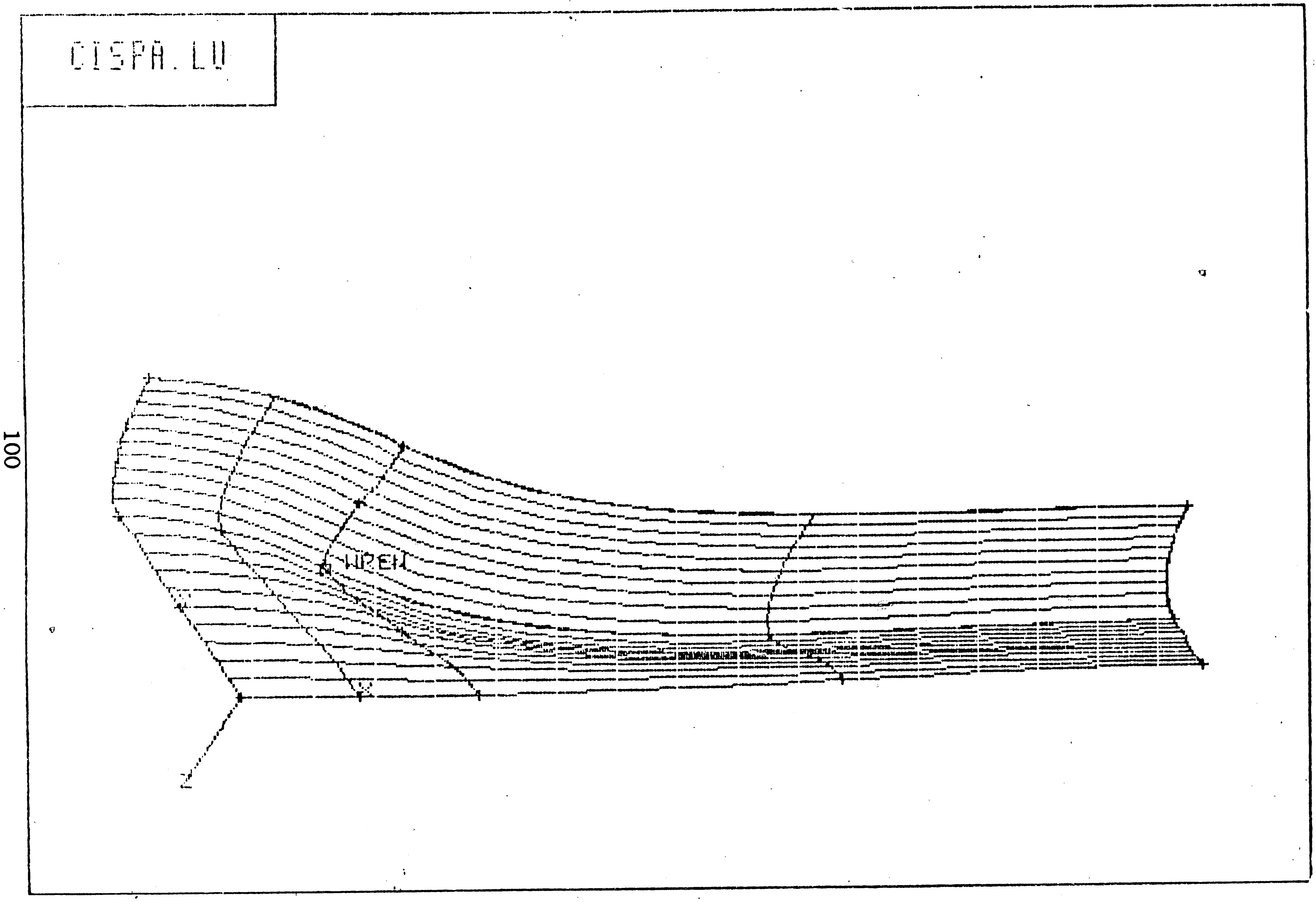


FIGURE A4. DOUBLY CURVES SURFACE - GENCUR DEFINITION

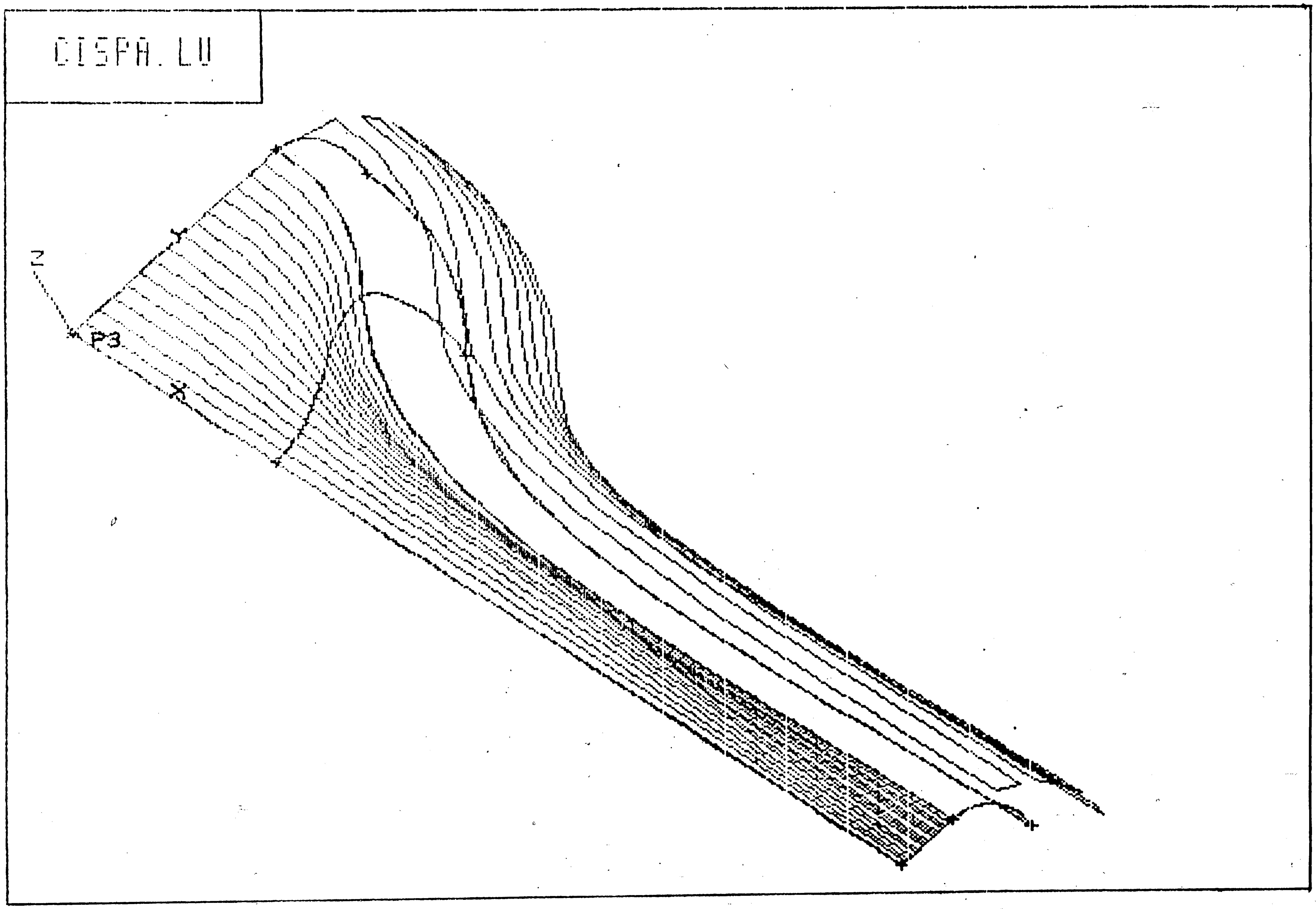


FIGURE A5. NC TOOL PATH CUTTER OFFSET IS .25-IN. FOR SURFACE

APPENDIX B. BLANK COMMON DESCRIPTION

BLANK COMMON DESCRIPTION 1

LOC	VALUE	DESCRIPTION
41	0.00	
42	2.00	CODE FOR CURSEG DEFINITION
43	3.00	NUMBER OF POINTS
44	28	NUMBER OF ELEMENTS IN BLANK COMMON
45	0.00	
46	0.00	
47	0.00	
48	0.00	
49	CURSEG	HOLLERITH CONSTANT FOR DEFINITION
50	19.	POINTS TO FOLLOW
51	0.	
52	0.	COORDINATES
53	0.	
54	19.	POINTS TO FOLLOW
55	10.	
56	5.	COORDINATES
57	0.	
58	19.	POINTS TO FOLLOW
59	20.	
60	-5.	COORDINATES
61	0.0	
62	136.	ISUB FOR TANGENT- CODE FOR TANGENT CONSTRAINT
63	TANSPL	HOLLERITH CONSTANT FOR TANSPL
64	20.	VECTORS TO FOLLOW
65	0.00	
66	-1.0	DIRECTION
67	0.00	

APPENDIX B. CURSEG DEFINITION

BLANK COMMON DESCRIPTION 2

LOC	VALUE	DESCRIPTION
41	0.00	
42	0.00	
43	0.00	CODE FOR COMBIN DEFINITION
44	2.00	TWO SIMPLE CURVES COMBINED
45	12.0	NUMBER OF ELEMENTS IN BLANK COMMON
46	0.00	
47	0.00	
48	0.00	
49	1.0	LOCATION OF CANONICAL FORM OF FIRST CURVE
50	48.0	RECORDSIZE
51	2.0	LOCATION OF CANONICAL FORM OF SECOND CURVE
52	48.0	RECORDSIZE

APPENDIX B. COMBIN DEFINITION

BLANK COMMON DESCRIPTION 3

LOC	VALUE	DESCRIPTION
41	0.00	
42	0.00	
43	0.00	
44	0.00	
45	21.0	NUMBER OF ELEMENTS IN BLANK COMMON
46	0.00	
47	0.00	
48	0.00	
49	RULED	HOLLERITH CONSTANT FOR RULED DEFINITION
50	1.0	LOCATION OF CANONICAL FORM OF CURVE USED IN RULED SURFACE
51	96.0	RECORDSIZE
52	146.0	ISUB FOR AXIS- CODE FOR AXIS
53	AXIS	HOLLERITH CONSTANT FOR AXIS
54	19.0	POINTS TO FOLLOW
55	0.00	
56	0.00	COORDINATES
57	0.00	
58	0.00	
59	19.0	POINTS TO FOLLOW
60	1.0	
61	0.0	COORDINATES
62	0.0	

APPENDIX B. RULED SURFACE DEFINITION

BLANK COMMON DESCRIPTION 4

LOC	VALUE	DESCRIPTION
41	0.00	
42	0.00	
43	3.00	NUMBER OF SPLINES
44	3.033	SURFACE SIZE PARAMETER
45	63.0	NUMBER OF ELEMENTS IN BLANK COMMON
46	0.00	
47	0.00	
48	0.00	
49	SPLINE	HOLLERITH CONSTANT FOR SPLINE
50	19.0	POINTS TO FOLLOW
51	2.0	
52	3.0	COORDINATE
53	0.0	
54	19.0	POINTS TO FOLLOW
55	4.0	
56	3.0	COORDINATE
57	0.0	
58	136.	ISUB FOR TANGENT CONSTRAINT
59	TANSPL	HOLLERITH CONSTANT
60	20.0	VECTOR TO FOLLOW
61	1.0	
62	0.0	DIRECTION
63	0.0	
64	138.	ISUB FOR WEIGHT CONSTRAINT
65	WEIGHT	HOLLERITH CONSTANT
66	21.0	VALUE TO FOLLOW
67	.80	WEIGHT VALUE
68	19.0	POINT TO FOLLOW
69	9.	
70	3.	COORDINATE
71	0.	
72	SPLINE	HOLLERITH CONSTANT - NEW SPLINE DEFINITION TO FOLLOW
73	19.	POINT TO FOLLOW
74	3.0	
75	5.5	COORDINATE
76	1.0	
77	19.	POINT TO FOLLOW
78	5.	
79	6.5	COORDINATE
80	1.0	

APPENDIX B. MESH SURFACE DEFINITION

MESH SURFACE DEFINITION CONTINUED...

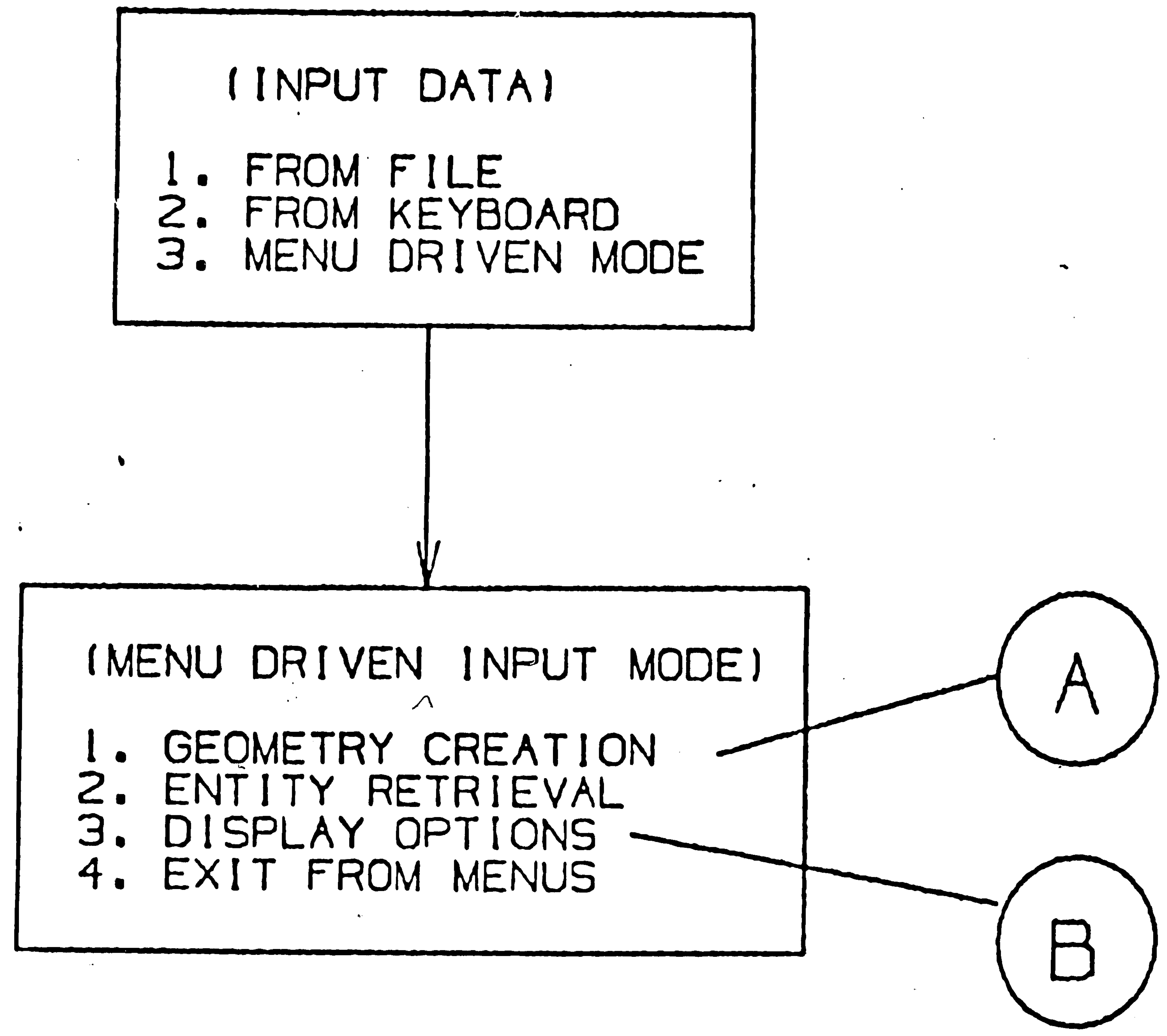
LOC	VALUE	DESCRIPTION
81	19.	POINT TO FOLLOW
82	10.	
83	5.5	COORDINATE
84	1.0	
85	13.0	ISUB FOR NORMAL VECTOR CONSTRAINT
86	NORMAL	HOLLERITH CONSTANT
87	20.0	VECTOR TO FOLLOW
88	.08	
89	.124	DIRECTION
90	.98	
91	SPLINE	HOLLERITH CONSTANT - NEW SPLINE DEFINITION TO FOLLOW
92	19.	POINT TO FOLLOW
93	4.0	
94	7.0	COORDINATE
95	0.0	
96	19.	POINT TO FOLLOW
97	6.	
98	7.	COORDINATE
99	0.	
100	19.	POINT TO FOLLOW
101	11.	
101	7.	COORDINATE
102	0.	

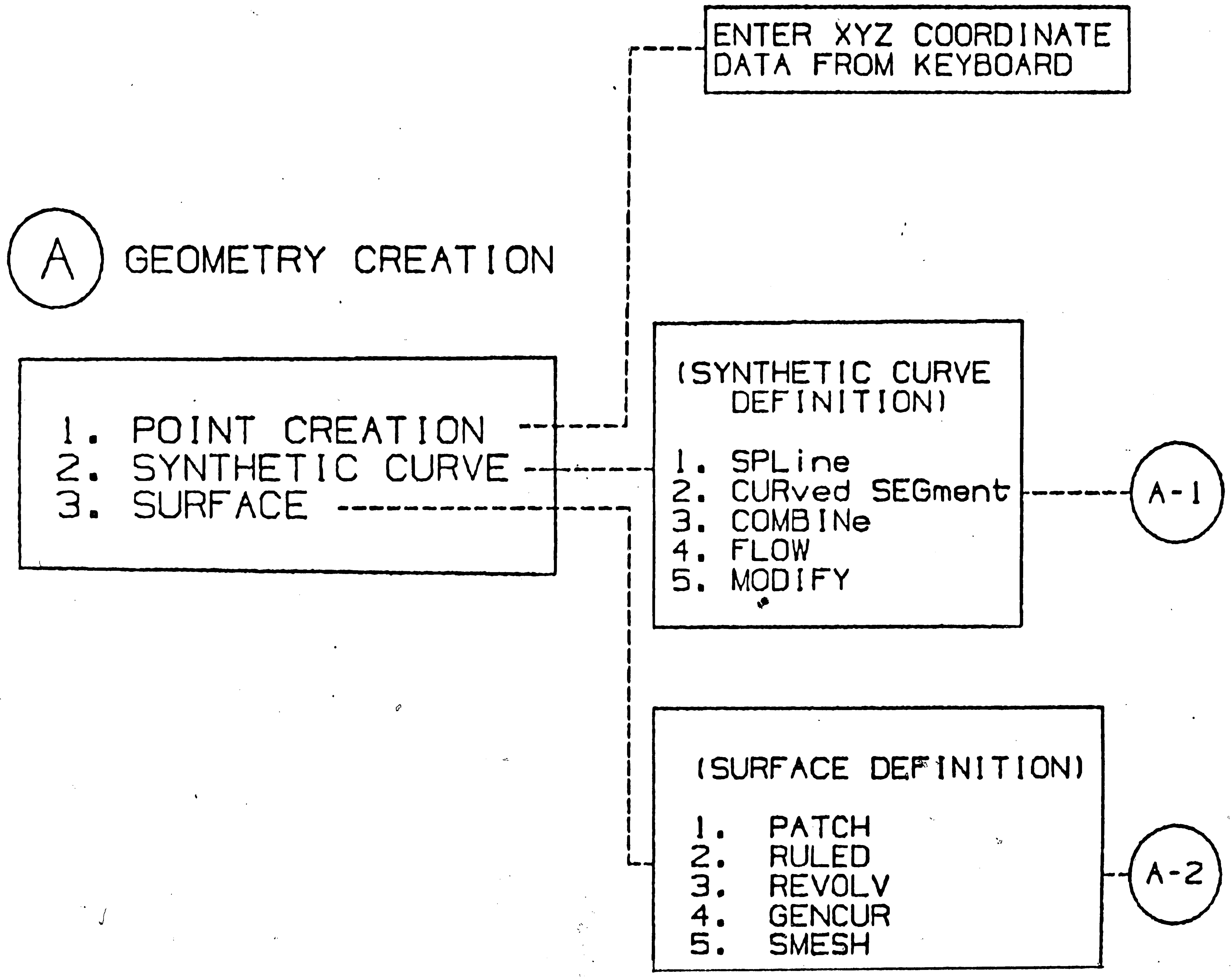
BLANK COMMON DESCRIPTION 5

LOC	VALUE	DESCRIPTION
41	0.00	
42	0.00	
43	0.00	
44	0.00	
45	22.0	NUMBER OF ELEMENTS IN BLANK COMMON
46	0.00	
47	0.00	
48	0.00	
49	12.	LOCATION OF CANONICAL FORM --LOC=12
50	104.	RECORDSIZE
51	15.	LOCATION OF CANONICAL FORM --LOC=15
52	104.	RECORDSIZE
53	18.	LOCATION OF CANONICAL FORM --LOC=18
54	104.	RECORDSIZE
55	0.0	ISUB FOR CROSS (NOT USED)
56	CROSS	HOLLERITH CONSTANT-- CROSS CURVES TO FOLLOW
57	3.	LOCATION OF CANONICAL FORM --LOC=3
58	132.	RECORDSIZE
59	9.	LOCATION OF CANONICAL FORM --LOC=9
60	104.	RECORDSIZE
61	6.	LOCATION OF CANONICAL FORM --LOC=6
62	112.	RECORDSIZE

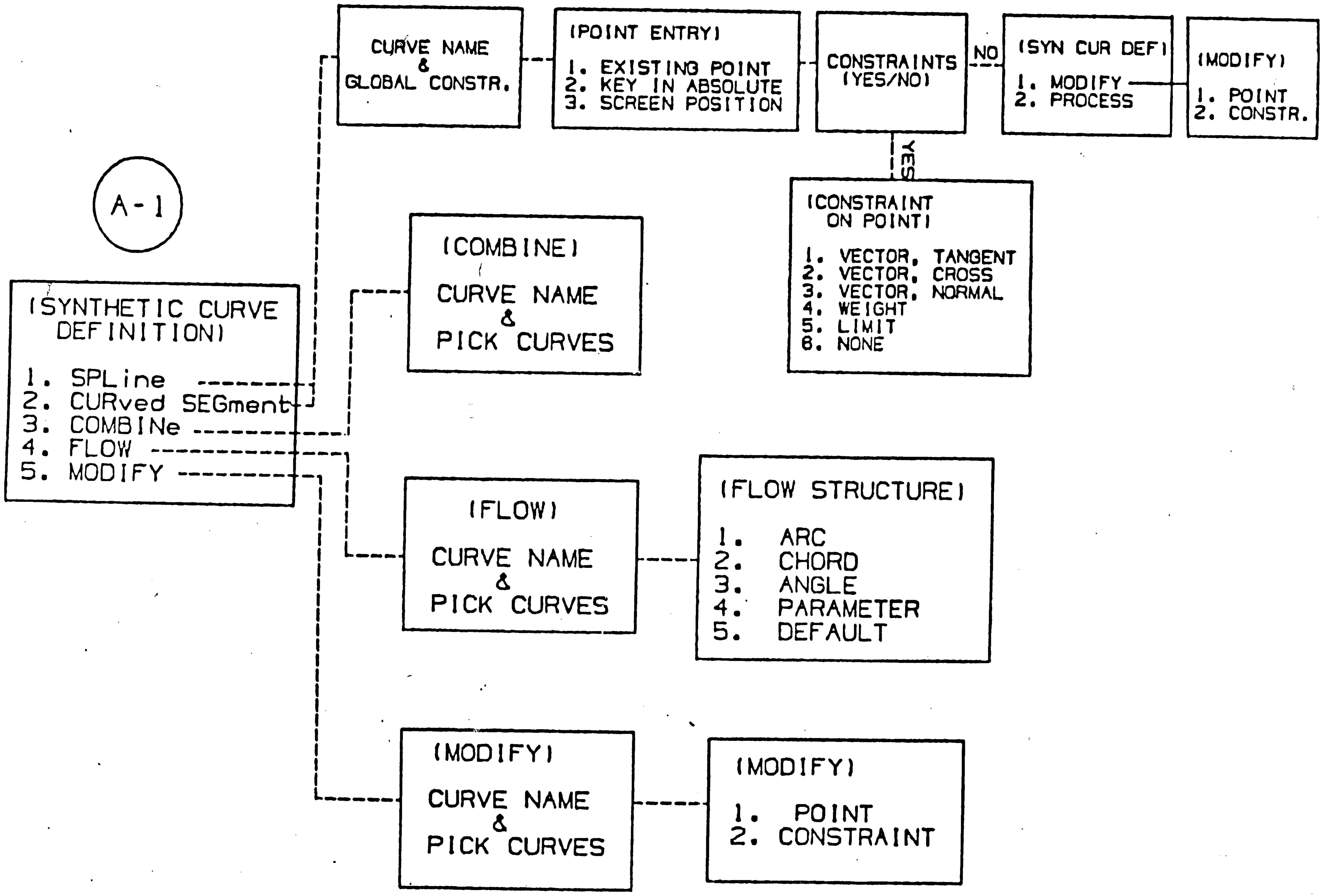
APPENDIX C. MENU TREE FOR PROGRAM OPERATION

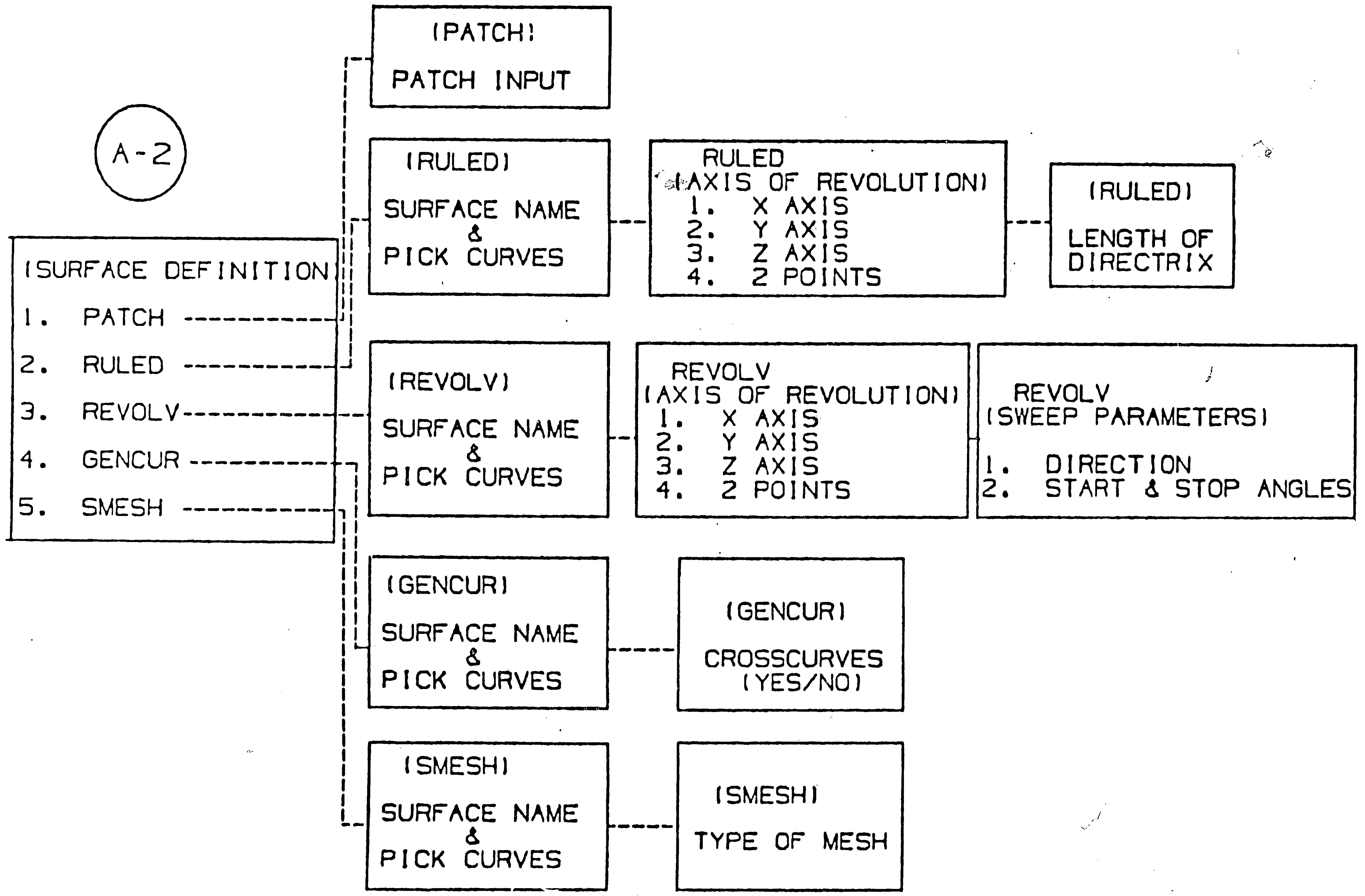
CISPA.LU MENU STRUCTURE



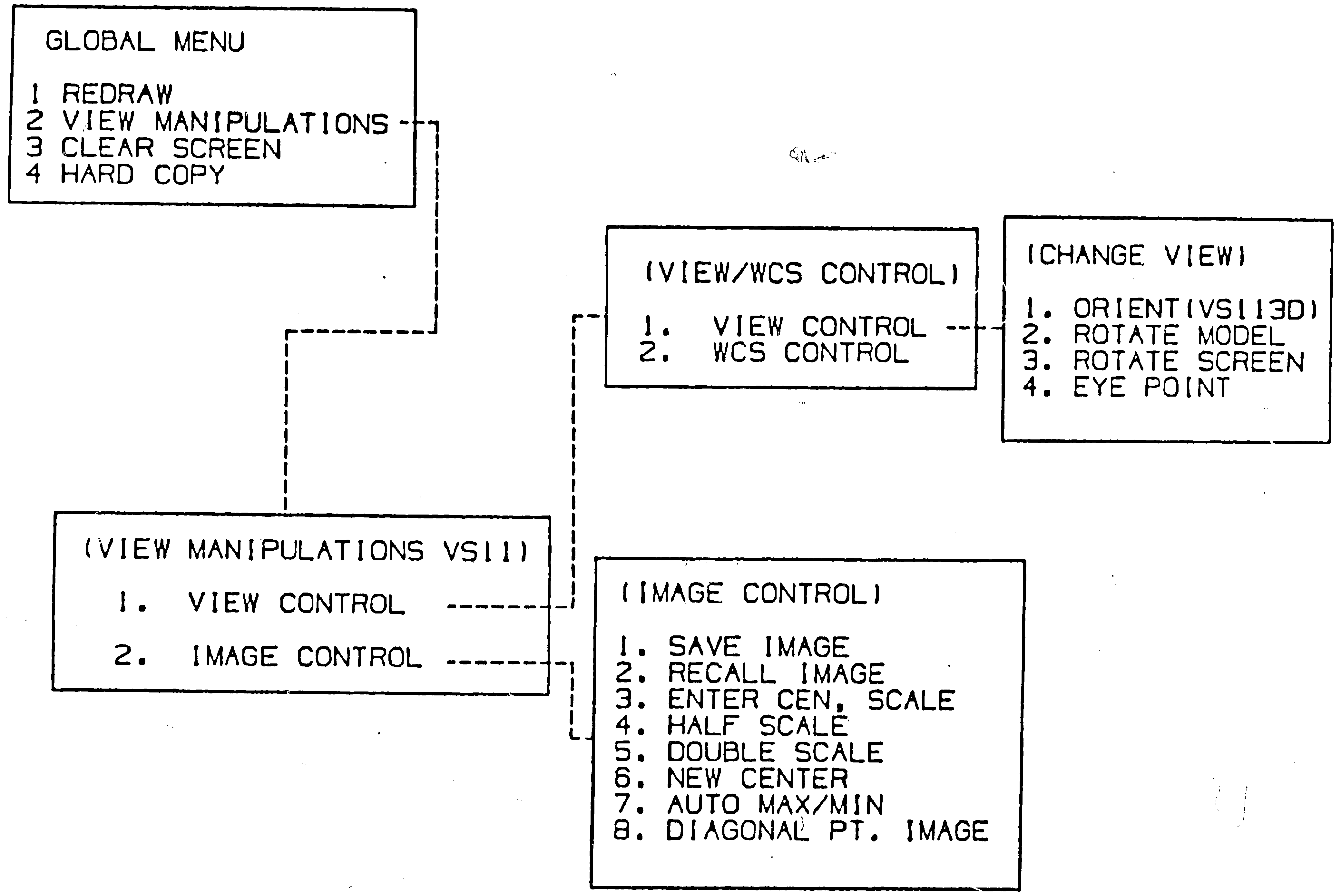

A
GEOMETRY CREATION
1. POINT CREATION
2. SYNTHETIC CURVE
3. SURFACE
ENTER XYZ COORDINATE DATA FROM KEYBOARD
(SYNTHETIC CURVE DEFINITION)
1. SPLine
2. CURved SEGment
3. COMBINe
4. FLOW
5. MODIFY
A-1
(SURFACE DEFINITION)
1. PATCH
2. RULED
3. REVOLV
4. GENCUR
5. SMESH
A-2

A-1

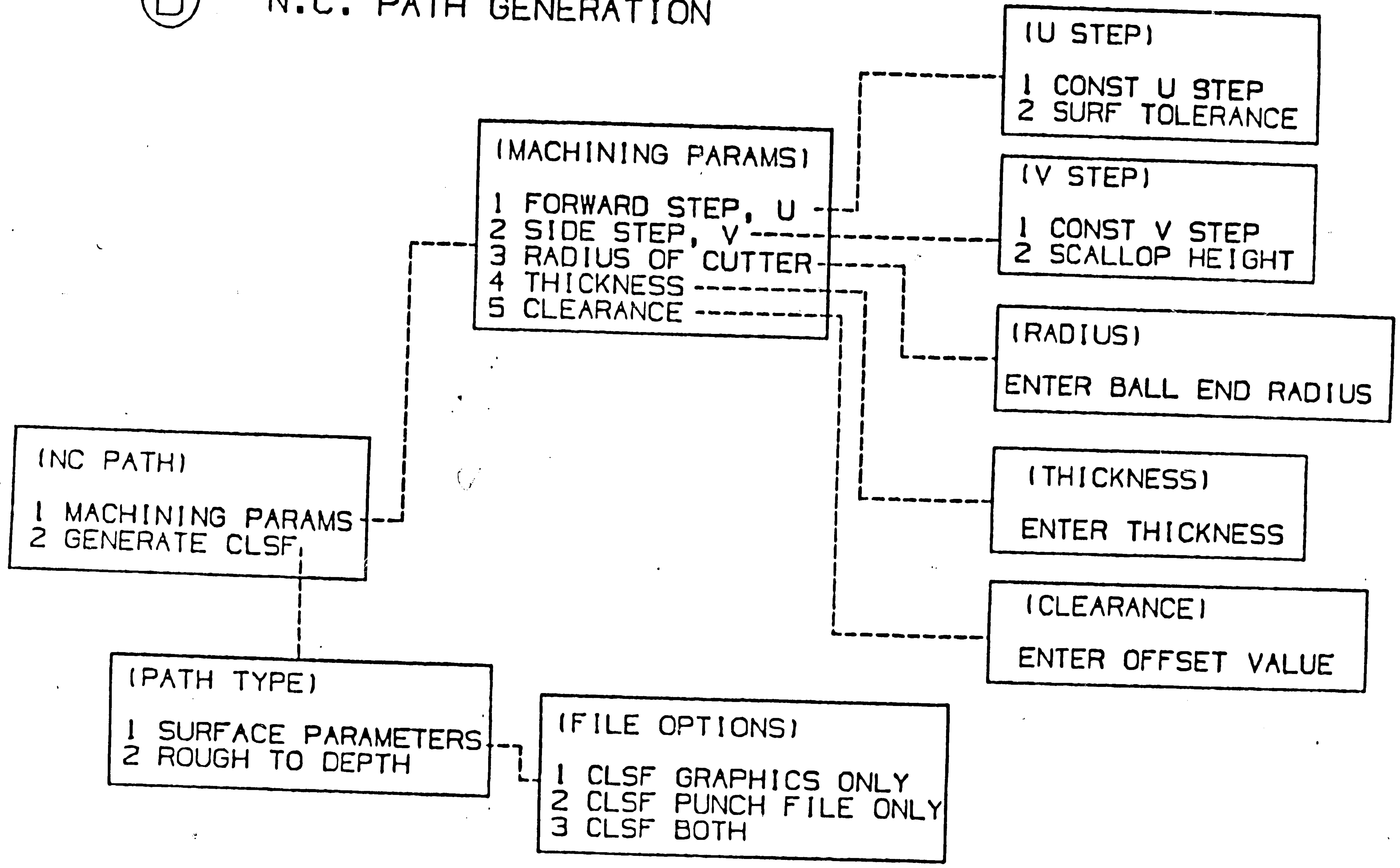

(SYNTHETIC CURVE DEFINITION)
1. SPLine
2. CURved SEGment
3. COMBINe
4. FLOW
5. MODIFY
CURVE NAME & GLOBAL CONSTR.
(POINT ENTRY)
1. EXISTING POINT
2. KEY IN ABSOLUTE
3. SCREEN POSITION
CONSTRAINTS (YES/NO)
NO
(SYN CUR DEF)
1. MODIFY
2. PROCESS
(MODIFY)
1. POINT
2. CONSTR.
YES
(CONSTRAINT ON POINT)
1. VECTOR, TANGENT
2. VECTOR, CROSS
3. VECTOR, NORMAL
4. WEIGHT
5. LIMIT
6. NONE
(COMBINE)
CURVE NAME & PICK CURVES
(FLOW)
CURVE NAME & PICK CURVES
(FLOW STRUCTURE)
1. ARC
2. CHORD
3. ANGLE
4. PARAMETER
5. DEFAULT
(MODIFY)
CURVE NAME & PICK CURVES
(MODIFY)
1. POINT
2. CONSTRAINT


A-2
(SURFACE DEFINITION)
1. PATCH
2. RULED
3. REVOLV
4. GENCUR
5. SMESH
(PATCH)
PATCH INPUT
(RULED)
SURFACE NAME & PICK CURVES
RULED (AXIS OF REVOLUTION)
1. X AXIS
2. Y AXIS
3. Z AXIS
4. 2 POINTS
(RULED)
LENGTH OF DIRECTRIX
(REVOLV)
SURFACE NAME & PICK CURVES
REVOLV (AXIS OF REVOLUTION)
1. X AXIS
2. Y AXIS
3. Z AXIS
4. 2 POINTS
REVOLV (SWEEP PARAMETERS)
1. DIRECTION
2. START & STOP ANGLES
(GENCUR)
SURFACE NAME & PICK CURVES
(GENCUR)
CROSSCURVES (YES/NO)
(SMESH)
SURFACE NAME & PICK CURVES
(SMESH)
TYPE OF MESH

(C) GLOBAL MENU



Ⓑ N.C. PATH GENERATION



VITA

Gregory Charles Loney was born April 24, 1959 in San Francisco, California. He obtained a Bachelor's of Science degree, in Mechanical Engineering from the University of California, Santa Barbara in 1981.

Astro Aerospace Corporation, Santa Barbara, California employed him as a mechanical engineer where he designed and conducted tests on deployable structures used in the aerospace industry.

He decided to continue his education in 1984 at Lehigh University where he could pursue his interests in control and identification of systems, CAD/CAM, study of dynamical systems and structural mechanics, working toward a Master of Science in mechanical engineering.

His parents are Kevin and Mary Ann Loney of Los Altos, California.