A SOFTWARE SYSTEM FOR INTERACTIVELY CREATING THREE DIMENSIONAL FREE FORM SURFACES

By

ALY A. BADAWY, B.SC., M.ENG.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree
Doctor of Philosophy

McMaster University
September 1980

THREE DIMENSIONAL FREE FORM SURFACE DESIGNS

To Professor J. N. Siddall

and to my wife Nadia

DOCTOR OF PHILOSOPHY (1980)
(Mechanical Engineering)

McMASTER UNIVERSITY
Hamilton, Ontario

TITLE: A Software System for Interactively Creating Three-Dimensional Free Form Surfaces

AUTHOR: Aly Ahmed Kamal Badawy,
B.Sc., (Alexandria University),
M.Eng., (McMaster University)

SUPERVISOR: Professor J. N. Siddall

NUMBER OF PAGES: ix, 208

ABSTRACT

A system for the interactive design of free form surfaces is presented. The system is best suited for creative design based on aesthetics, experience, or a number of empirical rules. The system provides the designer with a carefully integrated set of tools which permit a rapid and convenient creation of a curved 3-D surface of any type.

In addition to the uniqueness of the overall concept, there are several innovative features. These include definition of a patch by 16 surface points only; surface modification by dragging nodes to any desired location with the light pen; and a powerful technique for defining patches using plane curves on sections.

A new method for determining NC cutter path location is suggested using the developed system.

ACKNOWLEDGMENTS

I would like to express my most heartfelt thanks to my Graduate Committee consisting of Professor J. N. Siddall, Dr. M. Dokanish, Dr. H. Elmaragy and Dr. Solentsef, for their help and guidance.

I also owe a tremendous debt of gratitude to Professor Siddall. Without his concern, sensitivity and enthusiasm this work might never have been accomplished.

TABLE OF CONTENTS

Page

5.

APPENDICES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 The Role of Computer Graphics in Computer Aided Design

Design is regarded as the set of activities leading from the establishment of a product requirement to the generation of the information necessary for making the product. The design process itself differs widely from industry to industry. What fundamentally distinguishes CAD as a discipline from the ad hoc use of computers in the design process is that it involves the building of systems rather than disorderly collections of programs. This leads us to the more rigorous definition of CAD as: CAD is the integration of appropriate computer hardware and software modules to create design systems for particular requirements.

The role of computer graphics in this definition of CAD is a controversial topic. On the one hand there are those who consider the terms computer graphics and CAD almost synonymous, and on the other hand, those who see no use for computer graphics in CAD. To resolve some of this controversy we should concentrate on two issues:

- What use is computer graphics in design?
- How are designer's needs for computer graphics best provided?

Almost all engineering industries rely very heavily on drawings both for communication and as a means for information storage. The cost of preparing, codifying and storing drawings is high and is increasing rapidly. Computer graphics has helped some branches of industry to cut their costs significantly. Plotters and computer-output on microfilm (COM) devices are today heavily used in place of manual drafting. This application is sometimes called passive or non-interactive graphics. Input of the data from which drawings are plotted may, however, involve the use of interactive graphics.

Applications of interactive graphics fall into two categories:

1. Visual Scanning of Data

By presenting data graphically, CAD systems have made good use of designer's skills in pattern recognition, and in detecting special features in a design, such as poor fairing or a lack of clearance. This skill is particularly useful in detecting errors in large input data files, such as files representing input data for a three-dimensional finite element program.

2. Input of Design Data

Computer graphic techniques can help in defining and editing complex geometrical or topological relationships in the input data for CAD programs. Examples include the specifications of car body and aircraft component shapes for control programs, and the definition of electronic circuits for simulation purposes. Input of design data is generally facilitated by the use of graphic input devices. These are of three main types: light pens; tablets, which are a stylus working on a flat surface; and other devices, such as joysticks, tracker-ball and mouse, which do not attempt to simulate a pen or a pencil [1].

This thesis is heavily concerned with the second application of interactive graphics, which is the input of design data, although the developed system could be used also in visual scanning of data. The light pen is the main input device used, since it was proved to be in use longest and has the strongest following [2]. It has been proven in this thesis that the light pen is very easy to use especially when we use a menu of light button commands.

We come now to the second topic which has received particular attention in this thesis - the way designers communicate with computer graphics system, or man-machine interaction. We consider this to be an important topic because, without good communication, the designer will have difficulty using the system, and the system will be less effective. A truly effective communication cannot be established without considering flow of data in both directions (man-machine and machine-man), and it seems likely that improving the man's understanding of the picture will aid him in communication his ideas back to the computer. The light pen was chosen as an input device as it can be used to alter a picture dynamically. The operation of the programs throughout this thesis is based on the use of light buttons, words or symbols displayed on the screen which when selected with the light pen cause some appropriate program to be executed, or option to be chosen. No effort was made to design a special text command language to aid in man-machine communication through the use of the keyboard. This was due to the following reasons:

1. At the lowest level, the data rate from a graphical device is often much higher than from a keyboard. A light pen tracking cross can be sampled automatically on each refresh cycle.

2. The directness of graphics allows selection of either control items or names to be made from a set restricted to the valid possibilities. This makes graphic commands less error-prone.

3. With the use of graphic input, the number of options open per conscious action can be higher, implying that more information can be supplied to the program. This in turn means that commands can be concise without losing intelligibility.

From the previous general discussion, one can emphasize the value of computer graphics in engineering design.

1.2 Literature Survey on Three-Dimensional Interactive Surface Design

In three-dimensional interactive graphics, three particular issues have been the primary fields for research, those are, 3-D mathematics, 3-D graphic systems and 3-D hardware.

1. Three-Dimensional Mathematics:

By 3-D mathematics we mean the mathematical form selected to represent the shapes required by the design, either exactly or to a sufficiently close approximation.

The first useful description of a technique to represent free-form (3-D) surfaces [5] was based on the work of Professor Coons [6]. Coons divided the surface into smaller segments, called patches. In designing the patch three different entities have to be considered - points, slopes and twists, and the user typically has to supply numbers of three different orders.(*). Apart from the initial inconvenience, the effects of modifying slope and twist vectors is confusing (the surface bulges in an unexpected way). During this period (1967) Gordon [7] and Forrest [8] extended and refined the general approach to 3-D curves and surfaces using one and two-dimensional parametric cubics [9] which received considerable attention after cubic curves evolved as the most popular form. While many advantages were evident, the parametric methods were not without problems. A network of four-sided surface patches had to be formed into design surfaces. Position definitions of each corner was simple enough to achieve,

---

(*) See Appendix (I)

but the definition and manipulation of parametric derivatives were another problem. It seemed that potential users of the algorithms simply had to be in command of too much mathematics to be effective.

The early 1970's saw a new approach to surface patch models by Bezier [10], which offered more intuitive and more localized control of shape. Bezier's method in more general form [11] is the B-spline curve, which allows parametric derivatives or shape control through the use of so-called 'design points'. To control shape the designer manipulates a network of 3-D design points. Unfortunately, the design points are not on the surface and, therefore, the network only remotely resembles the intended design surface (Figure (1.1)). Moreover, in order to specify a Bezier patch, sixteen spatial points must be specified together with eight tangent vectors, which appears to be rather cumbersome.

Since 1970 up til now, the Coons patch proved to be the dominant technique for creating a 3-D surface. Almost all the publications on 3-D modeling [12, 13, 14, 15, ...etc.] were more or less based on the Coons patch. With virtually all of the established algorithms for free-form curve and surface design, the user (designer) is forced to contend in some manner with parametric tangents as boundary conditions. Unfortunately no special attempt was made to tackle this problem

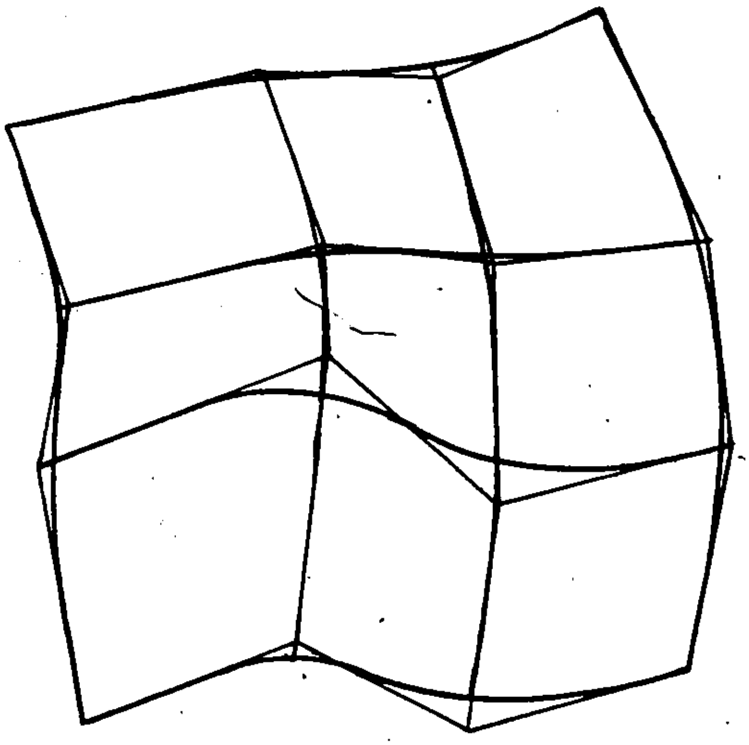


Figure (1.1) Bezier's patch defined by 16 points and 8 tangent vectors.

2. Three-Dimensional Interactive Graphic Systems:

Interactive computer graphics is usually thought to have been born with Ivan Sutherland's sketchpad in 1966 [3]. At that time, systems used very elaborate hardware, and the software was very hardware dependent [16, 17]. The system due to MacCalum [18] is noteworthy. The surfaces used were bicubic Coons patches. Design starts by causing the machine to read a tape of very approximate patch data. Thereafter design proceeds mainly by pointing with the light pen and 'dragging' part of the surface. Much work of the system is in the routines to decide what patch parameter is to be changed during dragging - e.g., a corner moved, a slope changed, etc. - following a pen hit on part of the displayed patch. The system lacks visualization aids and suffers from the readiness with which work is destroyed when errors are made.

Another interesting system is due to Armit [19, 20, 21]. By using typed commands the user creates a patch which appears on the screen with its corners labelled A, B, C, D and with its name shown. Patch modification commands are given in terms of the corner labels. There are over three hundred possible commands to allow patch modification, view modification and input/output. The significance of a language processor in 3-D design work was also demonstrated by Cordes and Brewer [22], who developed an interactive, user-oriented language called ICES/GETAM for 3-D data generation and manipulation using storage tube displays [1].

A different and more fundamental approach to language designs which can be used in 3-D work was reported by Kestner [23]. His language allows the user to deal directly with the mathematical constructs of curves and surfaces in analytical form. A user with a good and appropriate mathematical background would certainly find this system of utility.

Using a keyboard command interpreter as the primary input technique, Braid [24, 25] merged primitive objects (parallelepipeds, wedges, cylinder, etc.) to form more complicated objects. Volumes are added and/or subtracted until the desired shape is achieved. The method is not very effective in designing sculptured surfaces.

In 1975, Armit and Lemake [26] developed the ICON system for the interactive creation of data for the NASTRAN [27] suite of structured analysis programs. NASTRAN programs are used within Lloyd's Register of shipping to perform many kinds of static, elastic and dynamic analysis. The ICON system was designed to help in checking and modifying descriptions of finite element idealizations of ship hulls for input to the NASTRAN suite of structural analysis programs.

In 1976 Lacoste and Rothenberg [28] developed what they called 'Dialogue Programming', which uses interactive graphics and keyboard as input. Their objects were made from standard shapes or form elements, which were restricted to planes and cylinders. The system was coupled to EXAPT 2 for NC maching.

Pikler and Simon [29] designed a somewhat similar system to Lacoste's system, called 'Interactive Geometrical System Using GD'71'. They used interactive drawing on the screen to input geometrical definitions (points, slopes and twists). Their program was restricted to plane geometry only. The system is currently used in interactive die design and interactive lathe programming has also been implemented.

Seifert's [30] system PROREN, said to be in use in eight mechanical engineering firms/sites in West Germany, uses input from a keyboard, arbitrary views and arbitrary intersections can be plotted, and part description is done by adding/subtracting primitive solids which could intersect and overlap.

The DUCT system designed by Welbourne, Mathews, Gossling, et al. [31, 32] is now in use in small firms for the design of patterns, moulds and dies. The program is semi-interactive and uses Bezier [11] polynomial interpolation to define surfaces and curves of intersections. A good feature of the system is that cutter paths (for ball-end cutters) are generated directly from the design.

A system to aid interactive modeling (in 3-D) of a physical object was designed by England [33]. The system allows a user to fit a bi-cubic parametric spline surface to an object by superimposing stereoscopic views of the computer surface with stereoscopic television views of the object.

Voelcker, Requicha, et al.[34] designed the PADL system. PADL (part and assembly description language) is a language for defining solid objects via constructive solid geometry, in which complex solids may be defined as combinations of primitive solid building blocks. Unfortunately the system cannot handle sculptured surfaces.

3. Three-Dimensional Hardware:

There is now in existence an extensive family of devices to facilitate input of information to a computer. Attempts have been made to overcome the limiting features of 2-D displays and input devices by designing new 3-D hardware. These include the spark pen, marketed by Science Accessories Corporation, 3-wire wand, and the Twinklebox.

The spark pen consists of a hand held stylus which produces small electrical sparks. These generate acoustic wave fronts which are detected by three orthogonally mounted strip microphones. Strip microphones are mounted on long tubes and are sensitive to sound along their entire length. At the time each spark is generated, a counter is started. The counter is read as the wave front is detected at each microphone, thereby determining the time taken for the wave front to reach each microphone. Knowing the speed of sound, the position of the spark can be determined.

The 3-wire wand [35] employs three shaft encoders mounted at the vertices of a triangle on the ceiling. Each shaft encoder is fitted with a spring loaded pulley, around which a wire is wrapped. The three wires are joined together on a hand grip. By maintaining a measure of the lengths of the three wires, the computer can determine the position of the hand grip.

The spark pen and 3-wire wand both have severe limitations. Care must be taken to avoid obstructing the signals, either the sound waves or the wires. With these thoughts in mind the Twinklebox was developed by Robert Burton [36].

The Twinklebox is a device for sensing the positions of one or more small light sources. This is done using four scanners, one at each corner of the ceiling. Each scanner consists of a rotating disc around the edge of which radial slots have been cut. The axis of each disc points towards the center of the room. Consider positioning your eye behind a disc. As a slot passes your eye you see a planar slice through the room. If you were looking for a small light source, you would only see it when your eye, the slot, and the light source were all in the same plane. In the Twinklebox your eye is replaced by a photomultiplier which outputs a pulse when it sees a light. Two photomultipliers are used with each disc, subtending a right angle at the center of the disc, thereby giving two planes on which the light must lie. The four scanners therefore give eight planes to which a best fit point is found. Since only three planes are needed to define the position of a point, the system is highly redundant, thereby allowing up to five scanners to be obscured without ill effect. The Twinklebox is not entirely satisfactory due to the significant mechanical content of the device, which has caused accuracy problems. Also, four 17" discs with slots around the periphery, rotating at 3600 r.p.m., make a fairly efficient siren.

Using a 3-wire wand Clark [37] developed a system for real-time 3-D surface design based on B-splines. Clark reported that the 3-D head-mounted display was "somewhat cumbersome". Position sensing mechanisms for the wand head mounted display were reported to have unacceptable accuracy and resolution problems. From a system software point of view, Clark expressed his dissatisfaction with the B-spline algorithm since control of surfaces is managed by design points off the design surface.

1.3 Motivations and Aim of the Present Thesis

From the discussion of the last two sections, it is apparent that there are three major problems facing the existing 3-D free form (sculptured) surface design systems, these are:

1. The definition of a free form surface is not easy for designers, it usually involves the definition of slopes and twists.
2. Input of 3-D data is a rather cumbersome task and error prone.
3. The dynamic interaction between the designer and the systems is lost.

A software system for the interactive design of free form surfaces, that tackles these problems, has been developed in this thesis. The first problem was solved by the definition of a patch (parametric bi-cubic) by 16 surface points <u>only</u>; slopes and twists were hidden completely from the user. The second problem was solved using a powerful technique for defining patches using plane curves on sections via the light pen. The third problem was solved by a dynamic surface modification which uses dragging of nodes (any of the 16 points defining the patch) to any desired location with the light pen. These solutions were integrated into a carefully designed software system, engineered to be conveniently and creatively interactive. In addition to the uniqueness of the overall concept, a fully developed, ship hull design program was designed as a powerful application of the developed system. A new technique for the calculation of the cutter path for NC end milling (ball-end) machine, for manufacturing free form surfaces, using an optimization technique, is proposed. A proposal is also made for achieving the complete integration of CAD-CAM using intersection curves; these are used as an output of the CAD system and as input to the CAM system.

CHAPTER 2

A NEW APPROACH TO 3-D SURFACE DESIGN

In Chapter I it was demonstrated that in almost all of the mathematical representation algorithms for 3-D curves and surfaces, slopes and twists (second derivatives) were involved. In this chapter, however, we will mainly be concerned with the mathematical techniques employed in the parametric bi-cubic surface patch [38, 39, 40, 13], and its definition by 16 surface points only, rather than its definition by points, slopes and twists [6], or by points not lying on the surface [11].

2.1 Three Dimensional Parametric Cubic Curves

In the parametric representation of 3-D curves the x, y and z coordinates of any point lying on the curve can be expressed as

$$x = f(u), \quad y = g(u), \quad z = h(u)$$

where f, g and h are different functions in the parameter u. If we consider these functions to be polynomials of the third degree in the parameter u, we can express x, y and z as

$$\begin{aligned} x(u) &= A_1 u^3 + A_2 u^2 + A_3 u + A_4 \\ y(u) &= B_1 u^3 + B_2 u^2 + B_3 u + B_4 \\ z(u) &= C_1 u^3 + C_2 u^2 + C_3 u + C_4 \end{aligned} \tag{1}$$

In matrix notation, Equation (1) could be rewritten as:

$$\begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}^T = (u^3 \ u^2 \ u \ 1) \begin{bmatrix} A_1 & A_2 & A_3 & A_4 \\ B_1 & B_2 & B_3 & B_4 \\ C_1 & C_2 & C_3 & C_4 \end{bmatrix}^T \tag{2}$$

The parameter u could take any value, but for convenience we will consider $0.0 \leq u \leq 1.0$, where the first point on the curve will have $u = 0.0$ and the last point on the curve will have $u = 1$, Figure (2.1). Hence, setting $u = 0$ and $u = 1$ in Equation (2) will yield

$$[x(0) \quad y(0) \quad z(0)] = [A_4 \ B_4 \ C_4] = \overline{w}(0) \tag{3}$$

and

$$\begin{bmatrix} x(1) \\ y(1) \\ z(1) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{4} A_i \\ \sum_{i=1}^{4} B_i \\ \sum_{i=1}^{4} C_i \end{bmatrix} = \overline{w}(1)^T \tag{4}$$

Taking the parametric derivatives of Equation (2) at $u = 0$ and $u = 1$, we get

$$\begin{bmatrix} x'(0) \\ y'(0) \\ z'(0) \end{bmatrix} = \begin{bmatrix} A_3 \\ B_3 \\ C_3 \end{bmatrix} = \overline{w}'(0)^T \tag{5}$$

and

$$\begin{bmatrix} x'(1) \\ y'(1) \\ z'(1) \end{bmatrix} = \begin{bmatrix} 3A_1 + 2A_2 + A_3 \\ 3B_1 + 2B_2 + B_3 \\ 3C_1 + 2C_2 + C_3 \end{bmatrix} = \bar{w}'(1)^T$$

Equations (3), (4), (5) and (6) could be combined as:

$$\begin{bmatrix} \bar{w}(0) \\ \bar{w}(1) \\ \bar{w}'(0) \\ \bar{w}'(1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \\ A_4 & B_4 & C_4 \end{bmatrix} \tag{7}$$

If we only consider the x coordinate we get:

$$\begin{bmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix} = \bar{w}_x(u) \tag{8}$$

Solving the linear system (8) for the coefficients A_1, A_2 and A_4 produces

$$(A_1\ A_2\ A_3\ A_4)^T = \bar{M}\ [x(0)\ x(1)\ x'(0)\ x'(1)]^T \tag{9}$$

where $\bar{M}$ is the inverse of the matrix presented in Equation (8).

$$\bar{M} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

Substituting Equation (9) into Equation (2) for the coefficients A_1, A_2, A_3, A_4 and B_1, B_2, B_3, B_4 and C_1, C_2, C_3, C_4, we can totally represent the curve by

$$[x(u)\ y(u)\ z(u)] = (u^3\ u^2\ u\ 1)\overline{M} \begin{bmatrix} x(0) & y(0) & z(0) \\ x(1) & y(1) & z(1) \\ x'(0) & y'(0) & z'(0) \\ x'(1) & y'(1) & z'(1) \end{bmatrix} \qquad (11)$$

Thus the curve can be defined if we know the coordinates of the end points $x(0)$, $y(0)$, $z(0)$ and $x(1)$, $y(1)$, $z(1)$, and the parametric slopes at the same end points $x'(0)$, $y'(0)$, $z'(0)$ and $x'(1)$, $y'(1)$, $z'(1)$.

2.2 The Parametric Bi-Cubic Surface Patch

Differential geometry [13] rests on Gauss' concept of a surface as a continuous function of two parameters u, v such that

$$x = F_1(u,v), \quad y = F_2(u,v), \quad z = F_3(u,v)$$

Following the same argument of representing the parametric cubic curve, one can define the parametric bi-cubic surface patch as:

$$\overline{w}(u,v) = (u^3\ u^2\ u\ 1)\overline{M}\ \overline{B}\ \overline{M}^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix} \qquad (12)$$

where $\overline{M}$ was defined in Equation (10) and $\overline{B}$ will be called the geometry matrix, since its elements control the shape of the surface patch, as will be shown later. The notation for elements of the $\overline{B}$ matrix is as follows:

$$\overline{B} = \begin{bmatrix} W_{00} & W_{01} & W_{00v} & W_{01v} \\ W_{10} & W_{11} & W_{10v} & W_{11v} \\ W_{00u} & W_{01u} & W_{00uv} & W_{01uv} \\ W_{10u} & W_{11u} & W_{10uv} & W_{11uv} \end{bmatrix}$$

Figure (2.2) depicts the relation between 3-D space and the u, v parametric plane. The elements of the geometry matrix $\overline{B}$ are explained as:

$$W_{00} = [W(u,v)]_{\text{at } \substack{u=0 \\ v=0}} = \text{point data,}$$

$$W_{00u} = \left[\frac{\delta w(u,v)}{\delta u}\right] \text{ at } \substack{u=0 \\ v=0} = \text{slope data,}$$

$$W_{00uv} = \left[\frac{\delta w(u,v)}{\delta u \delta v}\right] \text{ at } \substack{u=0 \\ v=0} = \text{twist data}$$

Thus the patch is now fully described by three different entities - points, slopes and twists, all related to the corner points of the patch.

2.3 Defining a Parametric Bi-Cubic Patch

Many ways have been used to define and create a parametric bi-cubic patch; and each approach must provide sufficient data to determine the 48 coefficients implies by Equation (12), i.e., 16 coefficients for each, x, y and z. Position definition of each patch corner is simple enough for the designer to achieve, but the definition and manipulation of the parametric derivatives, contained in Equation (12), is much more difficult. From the interactive computer-aided graphics point of view, it would seem that users of such algorithms would have to be in command of too much mathematics for them to be effective. A more serious problem arises when the designer tries to modify his surface. Altering one or more parametric slope or twist will give unpredictable effects on the shape of the surface, i.e., the direct intuitive relationship between the designer and the surface design program is lost. One possible method of tackling these problems is to define the surface by only spatial points. Peters [13] used a grid of 16 points (planar or twisted) with the corresponding "u,v" values specified in advance to define a bi-cubic surface patch. The definition of these "u,v" values in advance might cause problems for the designer because it is difficult for him to do the transformation from the 3-D space to the "u,v" plane. It is preferable if the designer specifies only the spatial points and the program calculates the corresponding "u,v" values. In this thesis a method is suggested for defining the patch by only 16 points, and estimating the "u,v" values of these points.

Using a more compact form for Equation (12), $\overline{w}(u,v)$ can be expressed as:

$$\overline{w}(u,v) = (u^3\ u^2\ u\ 1)\ \overline{S} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix} \qquad (14)$$

where $\bar{S} = \bar{m}\ \bar{B}\ \bar{M}^T$

Expanding Equation (14) will yield

$$(u^3v^3)S_{11} + (u^3v^2)S_{12} + (u^3v)S_{13} + (u^3)S_{14} +$$
$$(u^2v^3)S_{21} + (u^2v^2)S_{22} + (u^2v)S_{23} + (u^2)S_{24} +$$
$$(uv^3)S_{31} + (uv^2)S_{32} + (uv)S_{33} + (u)S_{34} +$$
$$(v^3)S_{41} + (v^2)S_{42} + (v)S_{43} + S_{44} = W(u,v)$$

or

$$\bar{R}\ \bar{T} = \bar{G} \qquad (15)$$

where $\bar{R}$ is a 16 x 16 matrix of uv products (which are still unknown), $\bar{T}$ is a 16 x 1 vector of the unknown $\bar{S}$ elements, and $\bar{G}$ is a 16 x 1 vector of the given 16 data points. Hence the result is a linear simultaneous system of equations whose unknowns are the elements of the $\bar{T}$ vector and the elements of the $\bar{R}$ matrix.

In some systems the u, v values are assumed [33], say u = 0, 0.25, 0.5, .75 and v = 0, 0.25 0.5, .75, and the u,v products are precomputed. This method could be useful only if the data points are not scattered and the patch itself is very small, or if the data points are equally spaced. If the previous specifications are not satisfied, which is a more general case, a more rigorous approach should be followed.

Consider for example, Figure (2.3) in which the patch is presented with 16 spatial points lying on the surface of the patch. The points are first attached by straight lines (dotted); then by calculating the length of these line segments, an estimate for the corresponding "u,v" values can be obtained, as shown in Table (2.1).

From the values calculated in Table 2.1, we can go back and calculate the elements of 16 x 16 (R) matrix of Equation (15). Note that the solution to Equation (15) provides three $\bar{S}$ matrices, one for each of the coordinates x, y and z. The Gauss-Jordan elimination technique with maximum pivot strategy [41] handles this problem. After solving the linear system for the 48 unknown values of $\bar{S}$, we can then go back and get the coordinates (x, y and z) for any arbitrary point lying on the patch, using the expanded form of Equation (15). This can be done by fixing the "u" value and incrementing "v" and thus obtaining lines of constant "u". Similarly we fix "v" and increment "u" to get lines of constant "v". These lines are used to draw the patch on a graphics terminal CRT.

Throughout this thesis the prescribed method for creating a bi-cubic surface patch using a grid of 16 points as the only data, has been used. It has been proven that this algorithm can handle very complicated shapes. Some typical output photographs are shown on Figures (5.25) and (5.31), showing the 16 data points lying on the patch surface. It has been also proven (as will be demonstrated in the next chapters) that the model can fit large regions of a surface, so a smaller number of patches could be used to define the whole surface.

That makes the algorithm a powerful, realistic, efficient and accurate device for CAD interactive graphics.

Point Number	u value	v value
1	0.0	0.0
2	$L_1/(L_1+L_2+L_3)$	0.0
3	$(L_1+L_2)/(L_1+L_2+L_3)$	0.0
4	1.0	0.0
5	0.0	$L_4/(L_4+L_{11}+L_{18})$
6	$L_8/(L_8+L_9+L_{10})$	$L_5/(L_5+L_{12}+L_{19})$
7	$(L_8+L_9)/(L_8+L_9+L_{10})$	$L_6/(L_6+L_{13}+L_{20})$
8	1.0	$L_7/(L_7+L_{14}+L_{21})$
9	0.0	$(L_4+L_{11})/(L_4+L_{11}+L_{18})$
10	$L_{15}/(L_{15}+L_{16}+L_{17})$	$(L_{12}+L_5)/(L_5+L_{12}+L_{19})$
11	$(L_{15}+L_{16})/(L_{15}+L_{16}+L_{17})$	$(L_{13}+L_6)/(L_6+L_{13}+L_{20})$
12	1.0	$(L_{14}+L_7+L_{14}+L_{21})$
13	0.0	1.0
14	$L_{22}/(L_{22}+L_{23}+L_{24})$	1.0
15	$(L_{22}+L_{23})/(L_{22}+L_{23}+L_{24})$	1.0
16	1.0	1.0

Table (2.1) Estimate of the "u,v" values

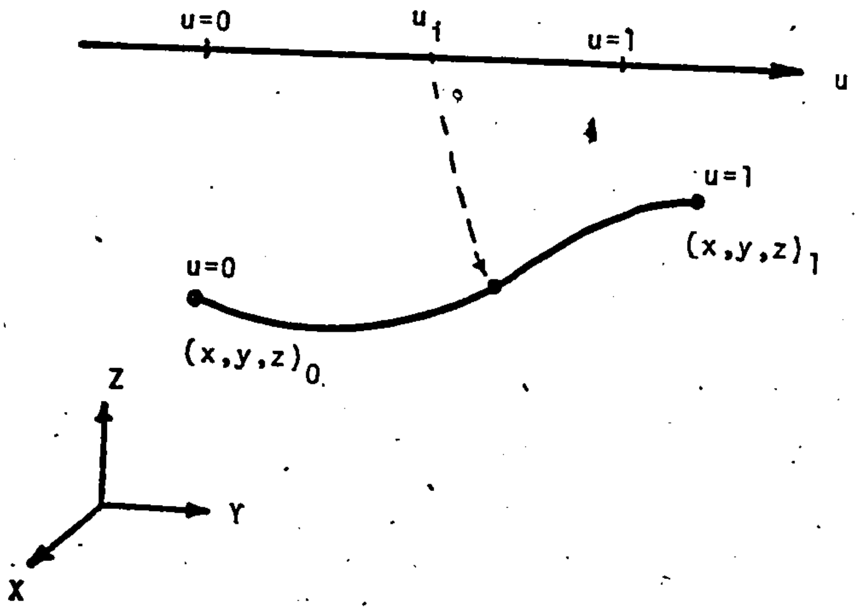


Figure (2.1) The correspondence between real x, y, z space and parametric space for parametric cubic space curve.

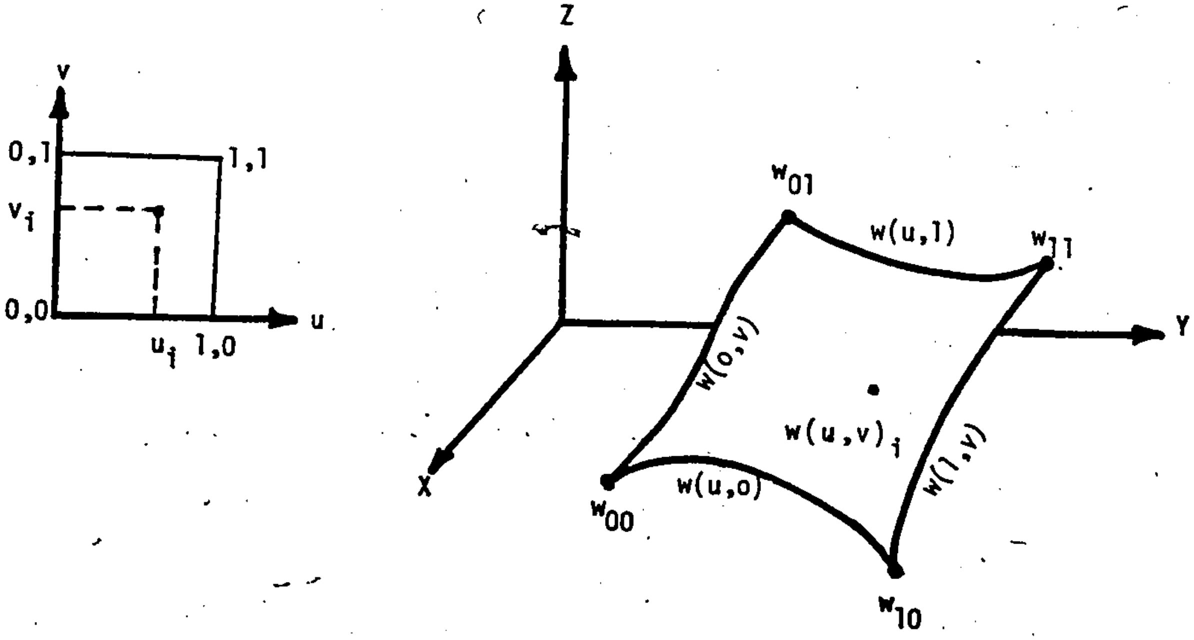


Figure (2.2) The correspondence between real x, y, z space and parametric space for parametric bi-cubic surface patch.

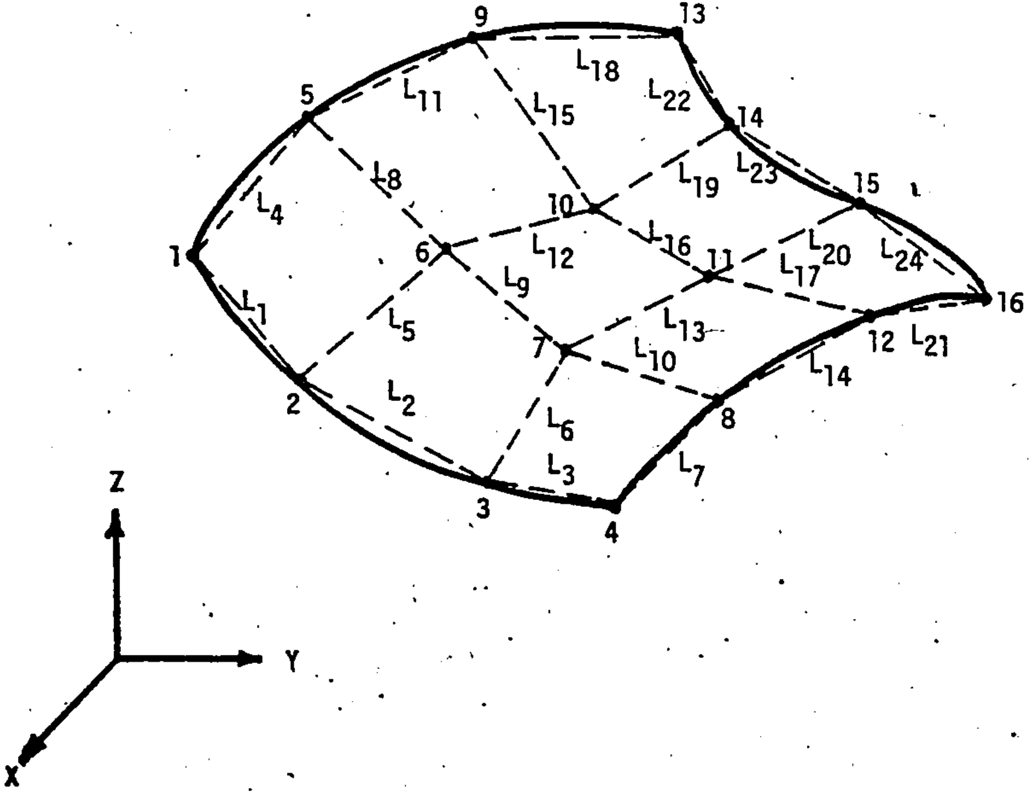


Figure (2.3) Parametric bi-cubic surface patch presented with 16 control points lying on the patch's surface.

CHAPTER 3

A SOFTWARE SYSTEM FOR INTERACTIVELY CREATING 3-D CURVED SURFACE DESIGNS

3.1 The Ideal System

The design of curved surfaces has always created difficulties for the engineer. Some types of curved surfaces, such as spheres, cones, cylinders, etc., can be represented very easily. However, this is not possible for free form shapes that are often used by designers.

In the last chapter we have investigated the mathematical modeling of such surfaces in a computer-amenable form. In this chapter we will confine our attention to the design and manipulation of these free form surfaces from the CAD graphics point of view.

Experience in CAD has indicated that the following software system specifications will ensure an ideal system for interactively creating 3-D curved surface designs:

(1) Initial input need only be a very rough approximation of the desired surface, and can be defined via numerical coordinate data, digitized data from a sketch, or data generated directly by an input device, using an assembly of patches.

(2) To define the surface, the designer need only define spatial points lying on the surface (nodes) and no slopes or twists of patches need be prescribed, i.e., no mathematical

knowledge of surface geometry is required.

(3) The initial input is displayed immediately.

(4) The assembly of patches can be displayed in any scale and orientation, with or without hidden lines, so that it can be readily visualized.

(5) Any surface can be created, including fully or partially closed surfaces, and fully smoothed or with discontinuities.

(6) Adjoining patches can have their junction smoothed by the computer, using a simple command.

(7) Transition or fairing patches between any adjacent but non-touching patches can be defined by a simple command.

(8) One or more nodes on any patch can be relocated using an input device (e.g., light pen). The surface is immediately redefined and displayed with the new points fully refaired.

(9) Any section through the surface can be defined and a true view obtained.

(10) The surface is defined numerically by the computer in a manner suitable for physical duplication, or interfacing with metalworking processors, finite element processors, modelling processors, and the like.

3.2 Objectives of Software Development

The main objective of this thesis was to design a system for interactively creating 3-D curved (sculptured) surface designs, which will fulfill the previously mentioned ten specifications.

A method suitable for the interactive design of free form surfaces is presented. The method is best suited for creative design based on aesthetics, experience, or a number of empirical rules. The method provides the designer with a carefully integrated set of tools which permit a rapid and convenient creation of a curved 3-D surface of any type.

3.3 General Algorithm for Surface Design

The general objective is to provide a facility on a typical minicomputer graphics system (Figure 3.1) which enables the user to do the subjective design of 3-D curved surfaces and make the definition and modification of such surfaces as direct and simple as possible. The basic tool is a surface patch defined by space coordinates. A "patch" is simply a relatively small segment (Figure 3.2) of a curved 3-D surface. Any surface is to be built up from a collection of such patches.

The basic tool of this software system is the definition of a patch by 16 space points lying on its surface as indicated in Figure 3.2. The patch was developed in Chapter 2.

The second primary tool is the facility to select any node and relocate it interactively. The system will then resmooth the surface to fit this new point. The patch to be distorted is first identified, and it may be a sub-patch or a multiple patch. Multiple nodes may be relocated before reconstructing the surface.

The third primary tool is the facility to create a patch

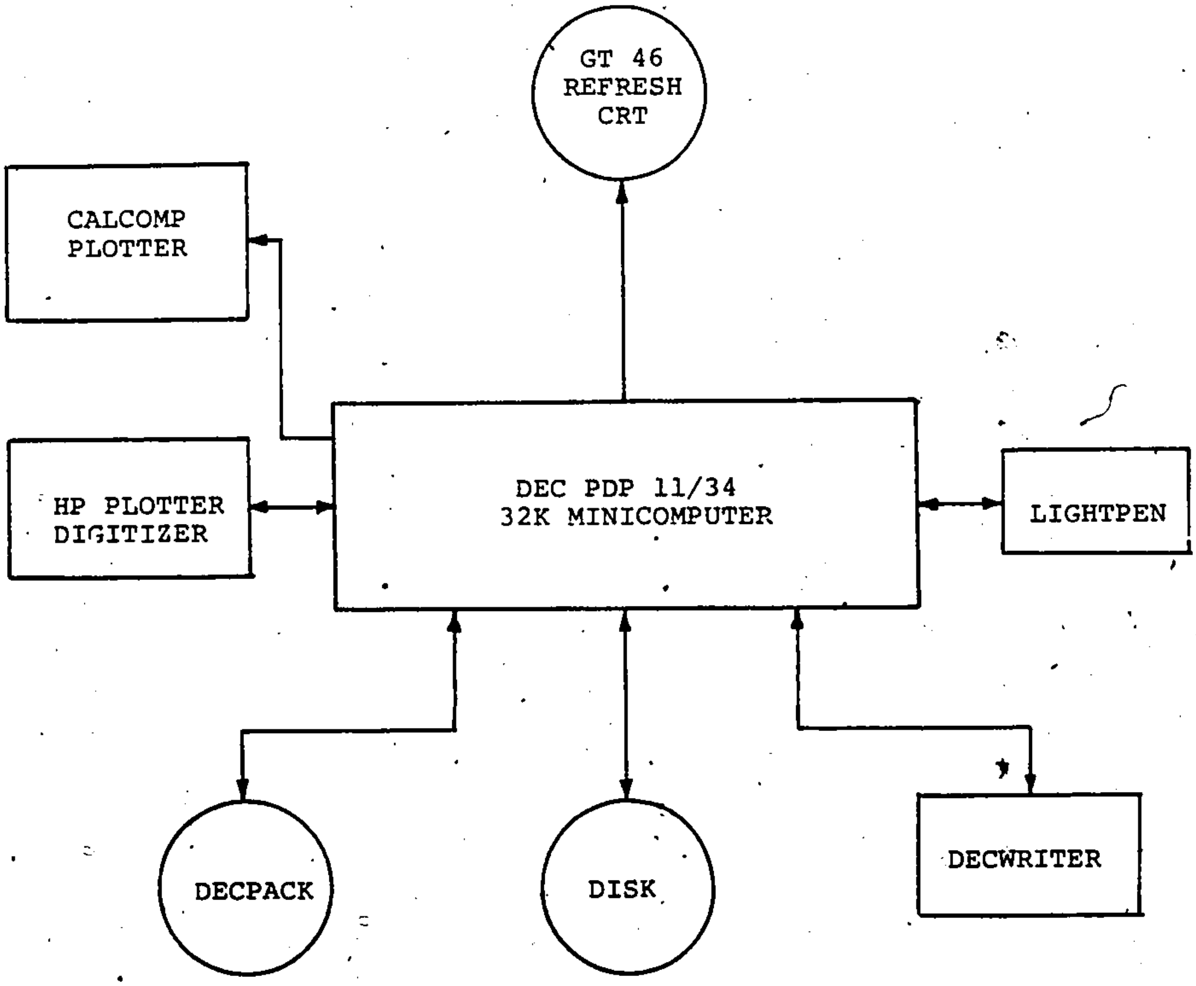


Figure 3.1 Interactive Graphic Hardware Configuration

between parallel or orthogonal sections. The user is asked to define four planes (parallel to x-y, y-z or z-x planes) and in each plane he is asked to define four points, i.e., define 16 points; and so define a patch. The program will then generate the surface patch which pass through these points. This facility is well illustrated in section 3.3.2.

This leads to the fourth primary tool, which is the facility to create a sub-patch as shown in Figure 3.3. A sub-patch is a patch created by splitting an already defined patch.

A final primary tool in building up the patch work surface is the facility to smooth two patches along the common borders. Each patch is independently smoothed as shown in Figure 3.4. At the junction point "A" there can be an undesirable second order discontinuity. A facility is provided for designating such a point a "smooth junction", if this is desired, and one of the two patches will be so modified.

Other important facilities are also provided by the system, like rotation, scaling and obtaining surfaces of revolutions.

3.3.1 Free Form Surface Design Program

A necessary element in any CAD or CAM system is to design an efficient and economical method for describing a 3-D part geometry in a form understandable by a computer.

There are several ways to provide this description. For example, a part can be described as an enormous number of coordinate points. This type of description, however, is

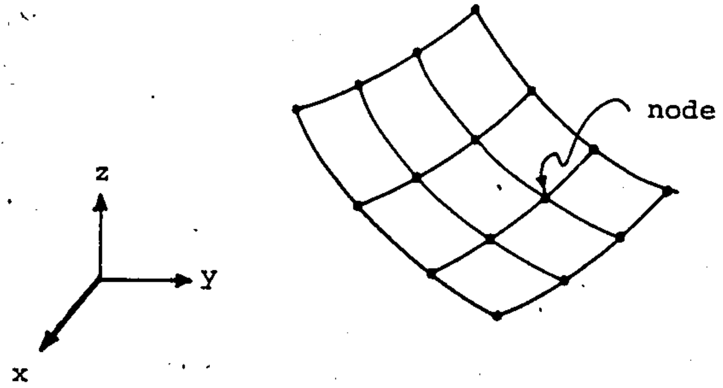


Figure 3.2 A Surface Patch Defined by 16 Points

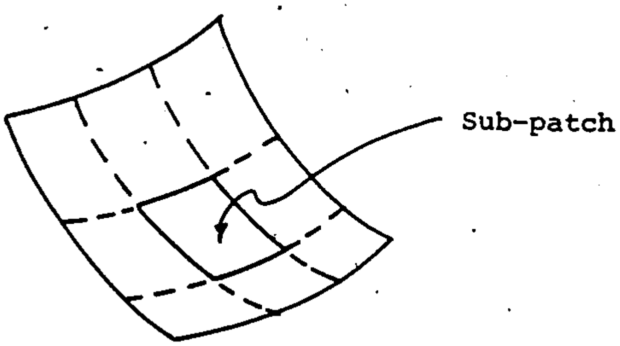


Figure 3.3 A Sub-patch

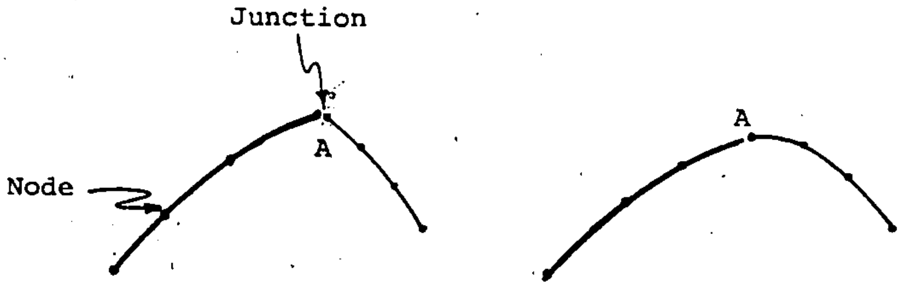


Figure 3.4 Section Through Two Adjacent Patches

inefficient.

Another technique is to use the so-called "user-oriented" NC programming language such as APT [42]. Such languages can describe geometric entities, including planes, cones, conic section and other complex surfaces, through simple English-like statements. We will not elaborate on such NC programming languages, but it should be noted that modeling of sculptured surfaces is difficult using such languages. In fact, the difficulty of modeling is one of the reasons that relatively few dies (for metal forming) are now cut by NC. Most of them are still made by tracer mills, where a stylus or tracer mechanism moves along a plaster or wooden model to guide a cutting tool that duplicates the shape in metal. The models, in turn, are made by hand in a laborious procedure that is anything but automated [43].

In this thesis, a completely different technique is used. Describing or actually designing 3-D geometry is done by defining selected coordinates (16 points per patch) and then having the computer blend them mathematically to generate the rest of the part configuration. In the following sections we will describe the various capabilities of the developed software system applying the previously mentioned technique.

3.3.2 Input of Initial Surface Data

Of all the input data needed for computer-aided design, the most difficult and time-consuming to produce is a description of part geometry. For this reason, the system provides the designer with three input options.

The design process begins as soon as the designer arrives at the computer with approximate patch coordinates and a layout of patches on a piece of paper. The design process can also begin with no prior surface design at all, i.e., it can be completely generated at the computer. As soon as the designer runs the program, a menu having the three input options will be displayed on the CRT as shown in Figure 3.5.

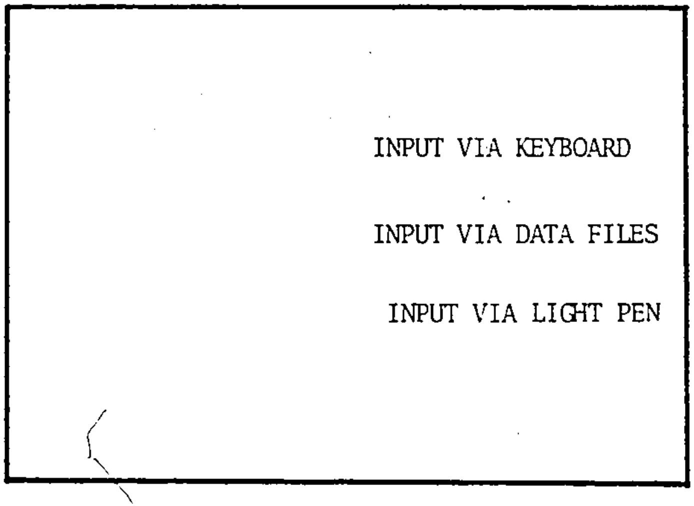
If we consider only one patch, the designer is required to define his surface patch through the use of any of the three options of Figure 3.5. As mentioned before, a patch is defined only by 16 x, y, z coordinate points.

If the designer hits the light button command "INPUT VIA KEYBOARD" with the light pen, the program asks him to enter 16 x, y, z coordinate points defining the patch via the keyboard. The patch is then displayed immediately.

The procedure is the same with the "INPUT VIA DATA FILES" command; the program asks for the data file name containing coordinate data describing the patch, the designer enters the name, the program displays the patch.

Finally if the user hits the "INPUT VIA LIGHT PEN" command with the light pen, the program will use the following steps:

(1) The X, Y, Z axis will be displayed on the CRT together with two light buttons on the menu area, "DEFINE WORKING PLANE" and "DONE" as shown in Figure 3.6.



INPUT VIA KEYBOARD

INPUT VIA DATA FILES

INPUT VIA LIGHT PEN

Figure (3.5) The three input options

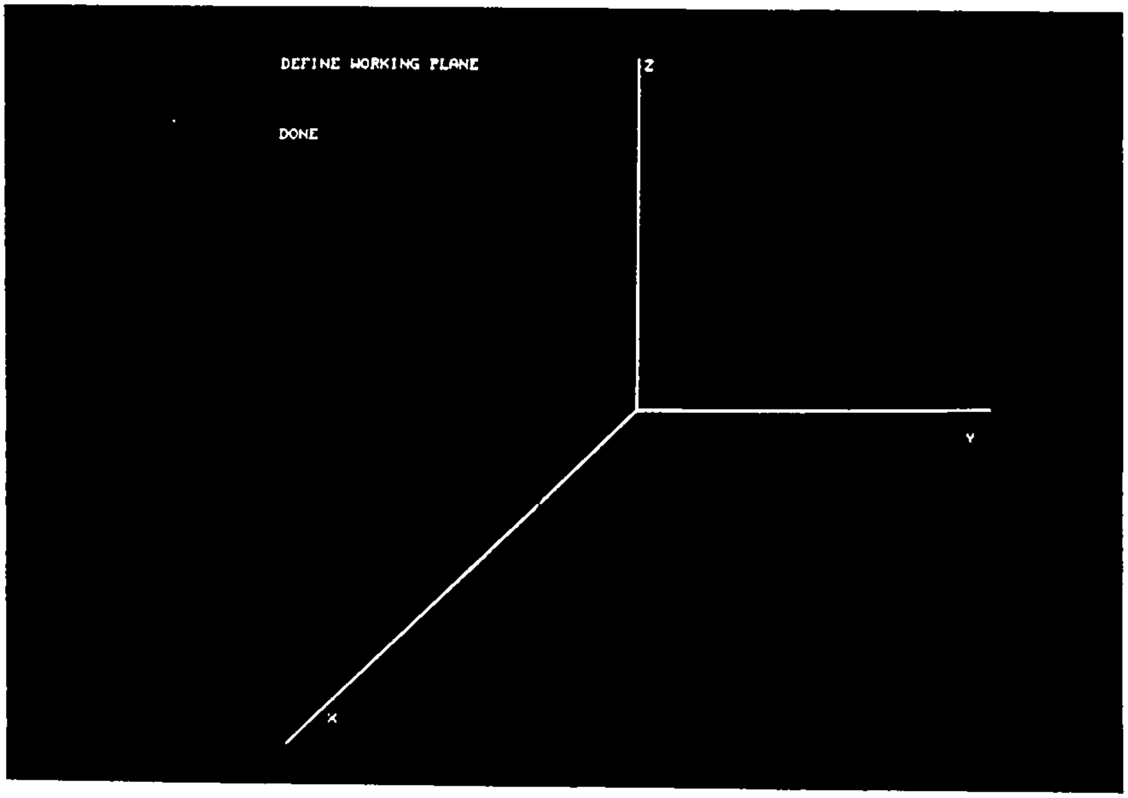


Figure (3.6) "DEFINE WORKING PLANS"
and "DONE" options

(2) To define a working plane, the designer hits the "DEFINE WORKING PLANE" option with the light pen; and the program types the following message: "POSITION THE TRACKING CROSS AT ANY POINT OF ANY ONE OF THE DISPLAYED X, OR Y, OR Z AXIS TO DEFINE THE LOCATION OF PLANE NUMBER 1".

(3) The user has positioned the tracking cross at point "A" of the X axis as shown in Figure 3.7.

(4) As soon as the first plane is displayed from point "A", another two menu light buttons will be displayed "POSITION" and "DONE" options. (Note that as the hit was on the X axis the working plane is parallel to the Y-Z plane).

(5) To define nodes of the patch, the user hits "POSITION" option with the light pen, the program types the following message: "POSITION TRACKING CROSS IN PLANE NUMBER 1, TO DEFINE PATCH OR PATCHES DATA POINTS, TO DO SO:
TRACK THE TRACKING CROSS WITH THE LP AND POSITION IT AT ANY POINT ON THE PLANE. HIT "RETURN" KEY ON THE DECWRITER.
REPEAT THE PREVIOUS PROCEDURE.
NOTE: TO DEFINE A PATCH YOU NEED TO POSITION TRACKING CROSS FOUR TIMES IN FOUR PLANES.

(6) Figure 3.8 shows the four defined points (nodes) in plane number 1.

(7) After defining the first four points of the patch the user can hit the "DONE" option. The program will erase "POSITION", "DONE" options and return to "DEFINE WORKING PLANE", "DONE" options. Again, the user could follow steps 2 to 7 to

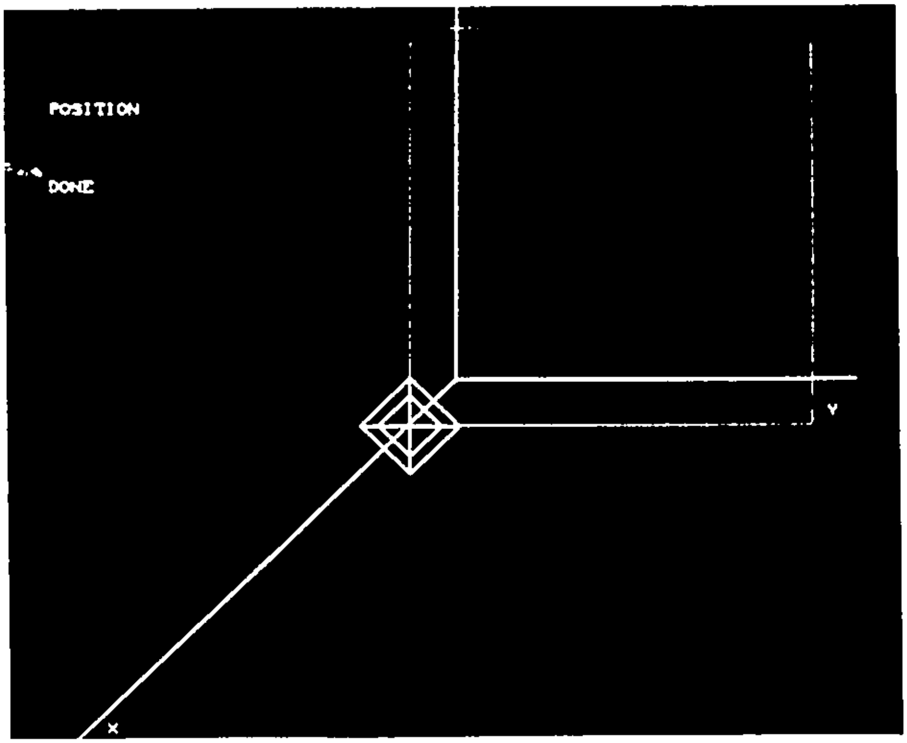


Figure (3.7) Tracking cross on the x-axis

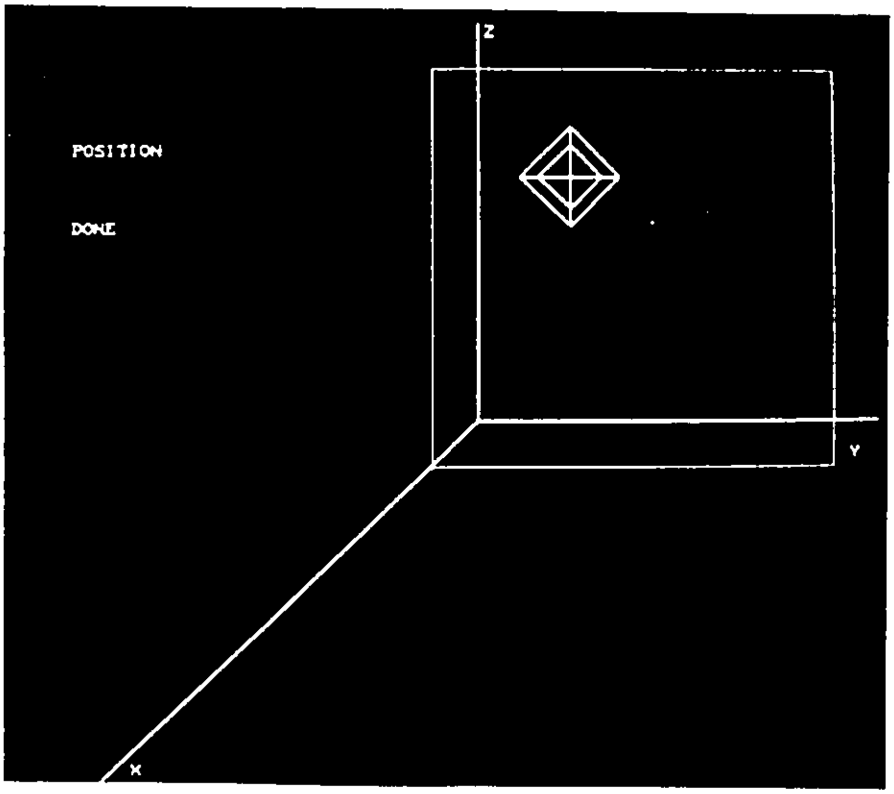


Figure (3.8) Four defined points in working plane number one define the second four points of the patch in a second plane parallel to the plane through point A, and so on until the user is finished with entering the 16 points defining the patch.

Figure 3.9 shows the 16 points defining the patch (four points per plane). Note that the intensity level increases from the first to last so that the user can easily decide on the working plane. Note also that the working planes do not have to be parallel, it could also be orthogonal.

(8) The user is ready now to hit the "DONE" option; the options "DEFINE", "DONE" will appear. Once more he could hit the "DONE" option, after which the patch surface will be dynamically drawn on the CRT. Figure 3.10 shows the typical patch surface.

(9) A surface manipulation menu will then automatically appear on the CRT, e.g., "SMOOTH", "BLEND", "ROTATE", "ERASE"...etc.

3.3.3 Patch Modification

If the user wants to modify the shape of the patch, all he needs to do is to point at the "MOD" option (in the menu are shown in Figure 3.10) and then track the tracking cross with the light pen and position it at the new desired position (Figure 3.11); the program then asks the user to point with the light pen at the point to be repositioned (point B). The program then dynamically erases the original patch and draws the new modified patch, (Figure 3.12). The patch to be modified can be identified by only pointing with the light pen at any part of it, and also multiple design points may be relocated before reconstructing the patch surface.

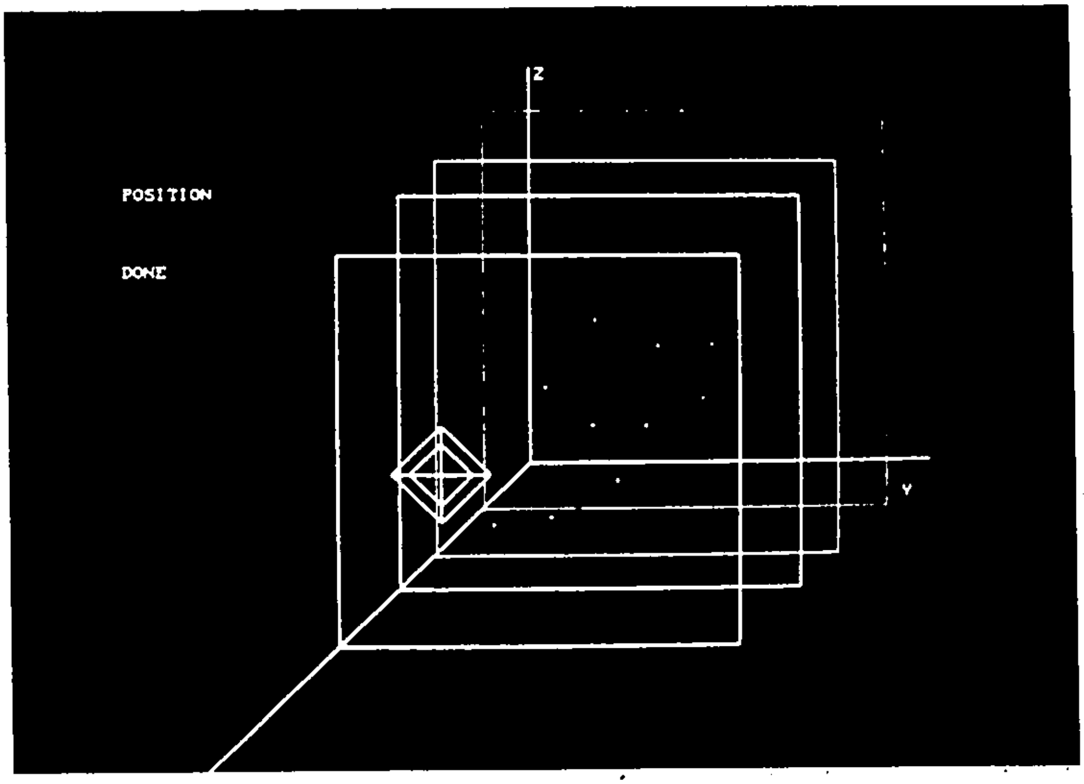


Figure (3.9) The 16 points defining the patch

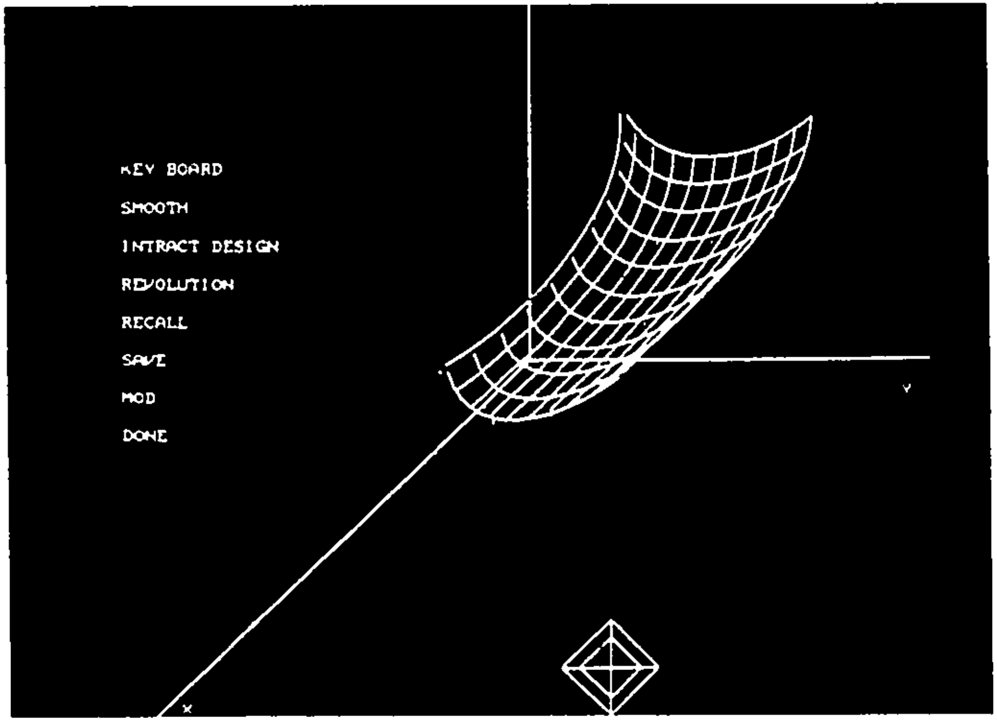


Figure (3.10) The generated patch

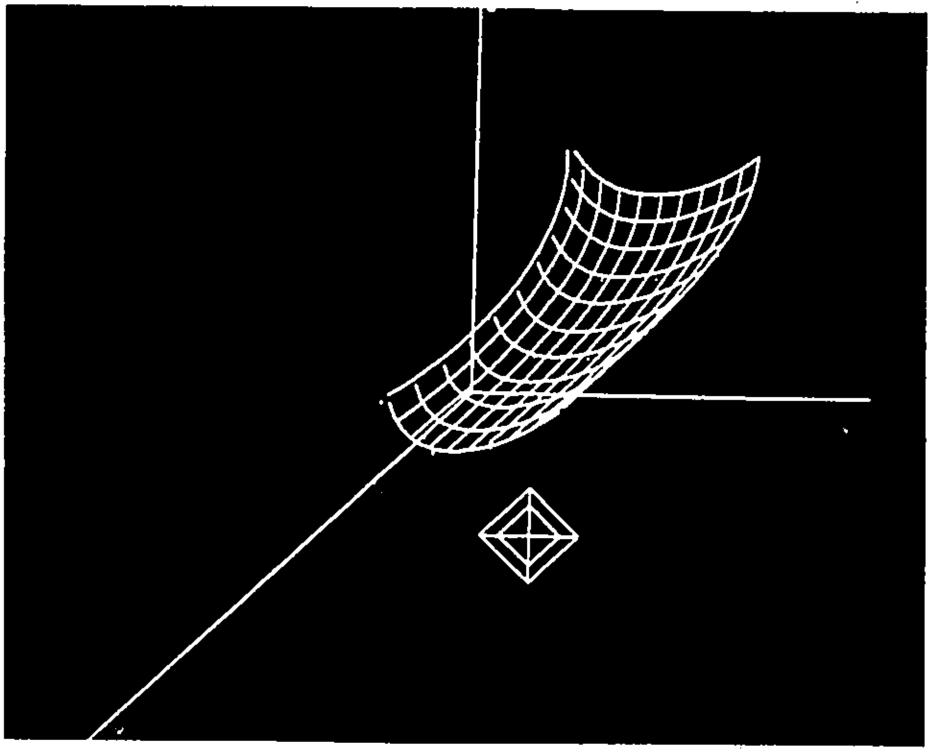


Figure (3.11) Tracking cross positioned at new designed location

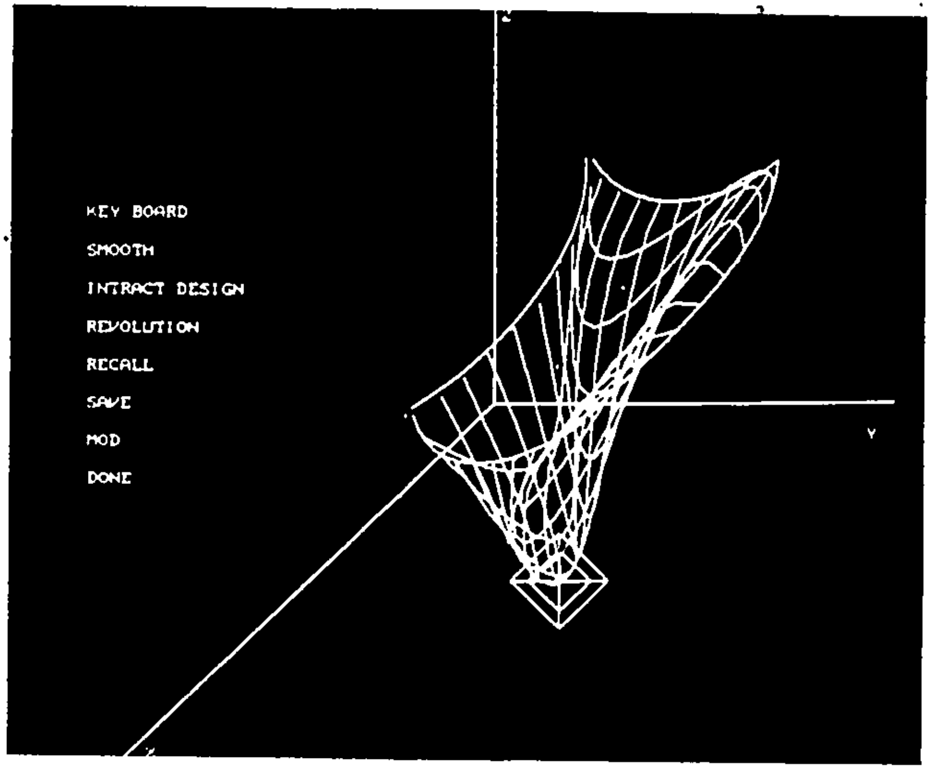


Figure (3.12) Patch after modification

3.3.4 Smoothing Two Patches at the Common Borders

Each patch is independently smoothed as shown in Figure 3.13. At the junction there can be an undesirable second order discontinuity. A facility is provided for designating such a border a "smooth junction" if this is desired, and any one of the two patches could be so modified to ensure "first and second order continuity" along the common borders*. The user first hits the "IDENTIFY" option, then hits the first patch (to be fixed), then hits the second patch (to be smoothed with the first patch). Execution is initiated by hitting the "SMOOTH" option, then the program smooths the two patches. Figure 3.14, shows the two patches after smoothing along the common borders.

3.3.5 Creating Transition Patches

The program can generate an in-between transition patch which ensures first and second order continuity across the adjoining edges of the surrounding patches**. The designer defines his first and second non touching patches, hits the "BLEND" menu option, and the program generates the transition patch immediately. Figure 3.15 shows the two non touching patches before blending and Figure 3.16 shows the two patches after blending with a transition patch.

---

* Appendix II

** Appendix III

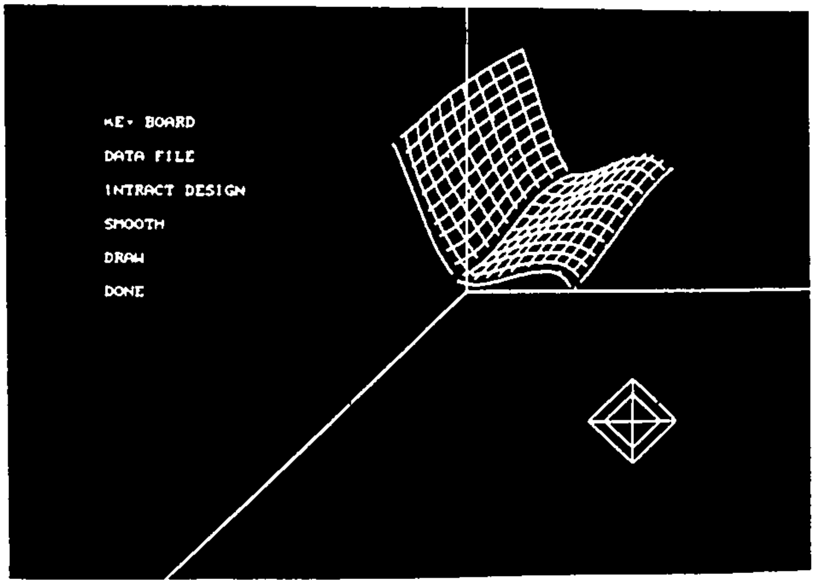


Figure (3.13) Two patches with discontinuity across the common border

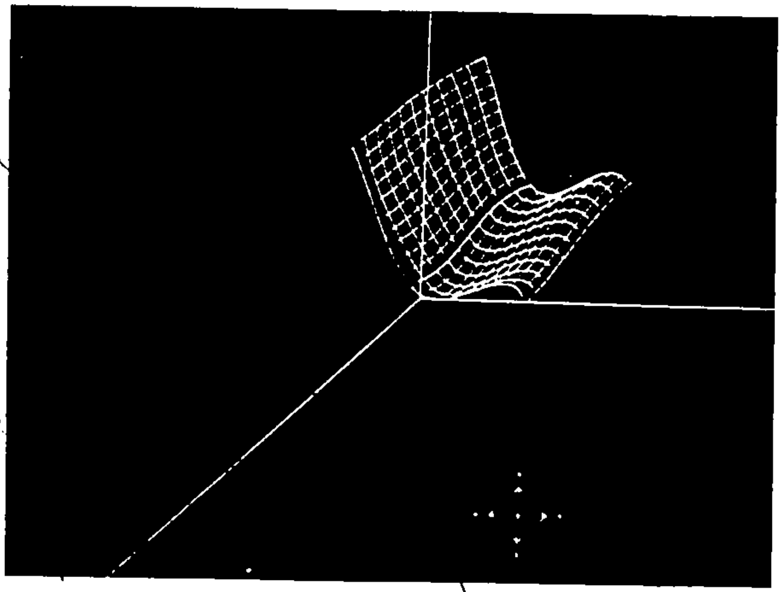


Figure (3.14) The two patches after "smoothing"

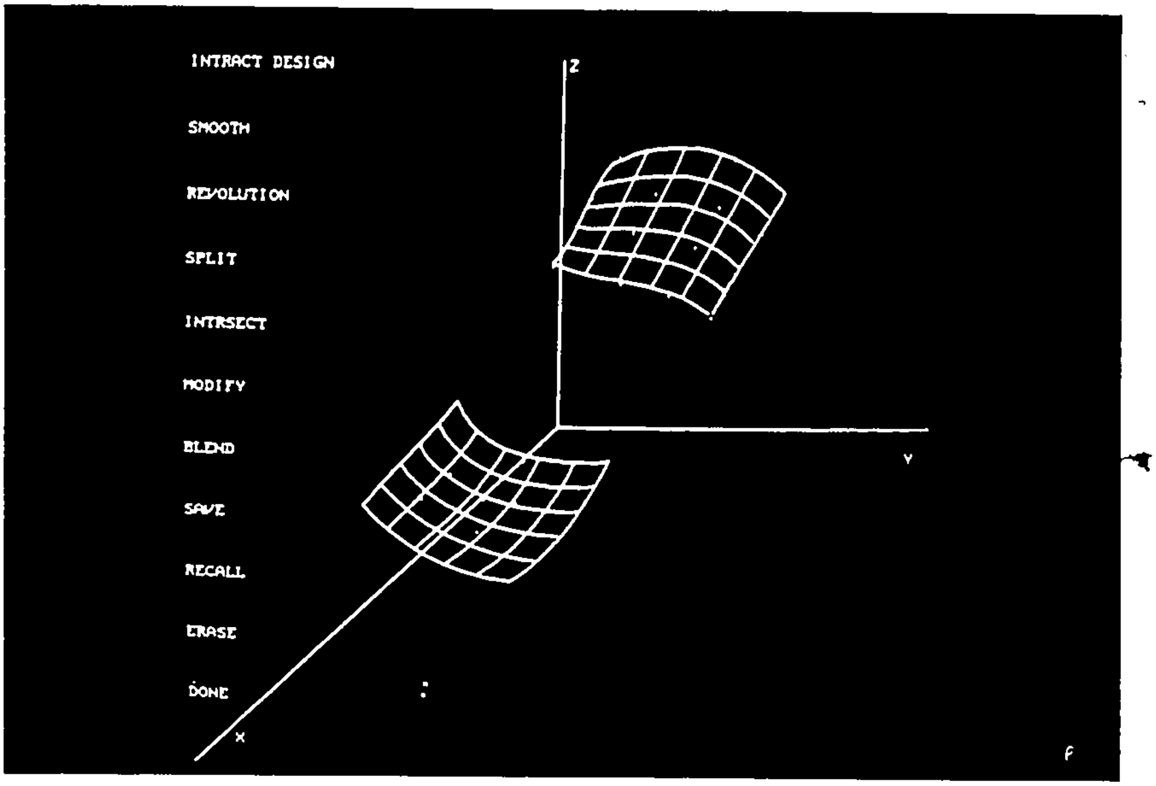


Figure (3.15) Two non-touching patches before "blending"

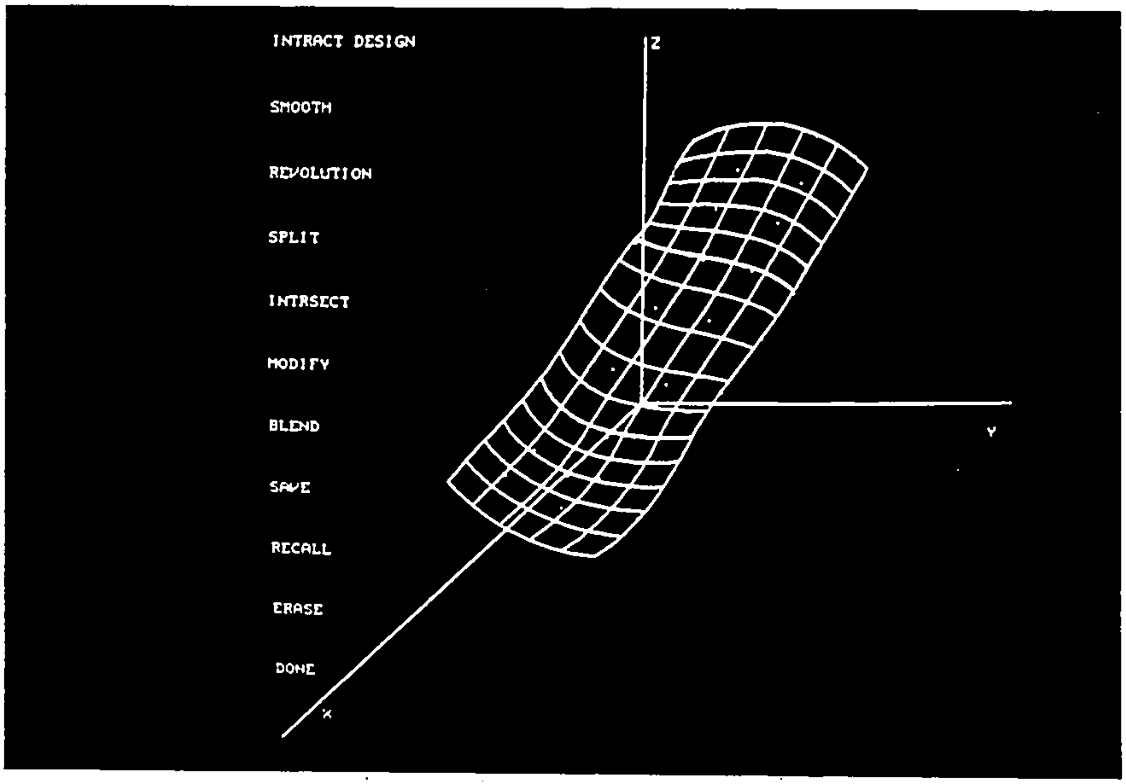


Figure (3.16). The two patches after blending

3.3.6 Creating Surfaces of Revolutions

Surfaces of revolutions can be created in a very simple way. As shown in Figure 3.17, the designer defines 6 points, the first two define the axis of revolution, the third, fourth, fifth and sixth define the surface as shown in Figure 3.18. The program uses points 3, 4, 5 and 6 to define the remaining 12 points defining the patch.

3.3.7 Creating Curves of Intersections with Planes

Curves of intersections between a patch and a plane can be obtained graphically. The designer, after defining his patch on CRT, hits the "INTRSC" menu option with the light pen. The program displays a representation of the patch on the U-V parametric plane (shown as a square on the upper left corner of the CRT in Figure 3.19, also the tracking cross will be displayed on the center of the CRT. The user is then asked to define the plane he wants to intersect with the patch; and, since we are working on the U-V parametric plane, the plane will be represented by a line. To define that line the user is asked to position the tracking object at two points on the borders of the U-V parametric plane, Figure 3.19. The program, then, dynamically draws the resulting continuous curve on the patch's surface, Figure 3.19. This way the user is only asked to define his plane graphically, rather than mathematically. Appendix IV illustrates the internal mathematics involved in generating these curves of intersection.

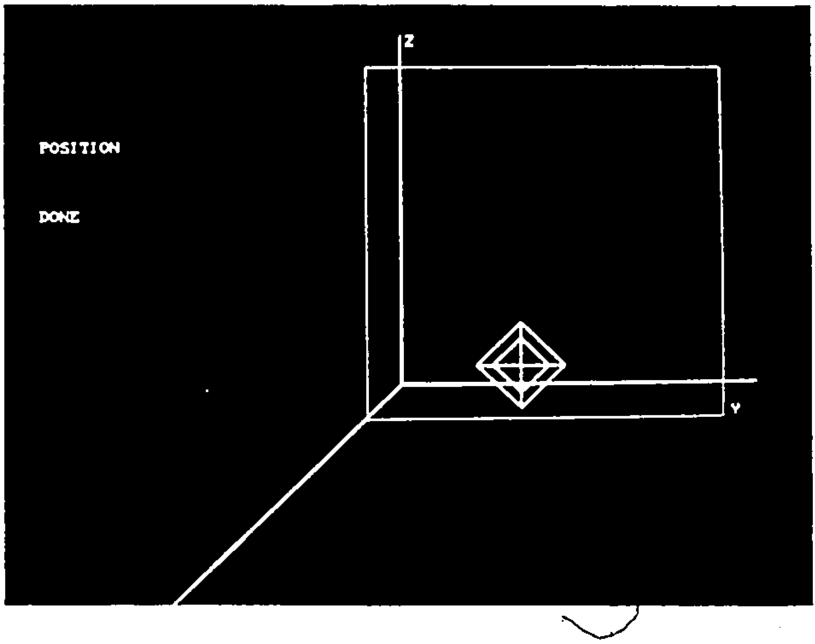


Figure (3.17) Six input points defining a surface of revolution

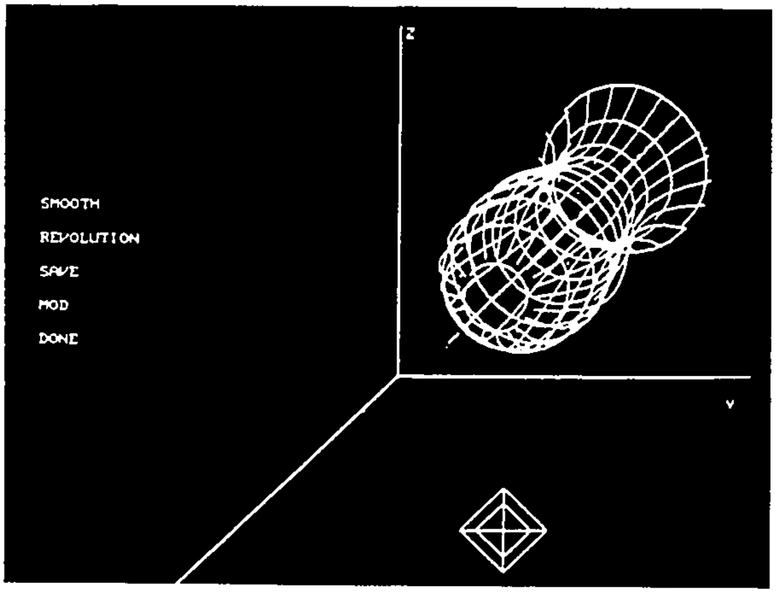


Figure (3.18) The generated surface of revolution

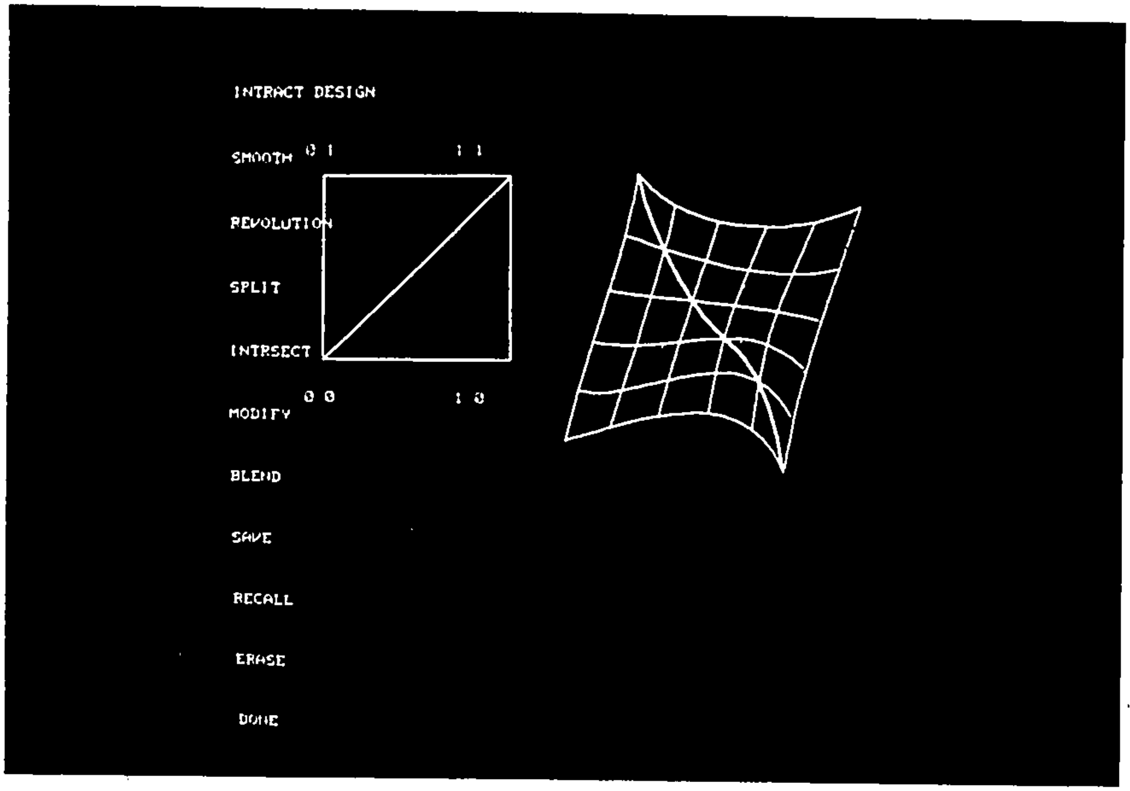


Figure (3.19) Intersection of a patch with a plane defined by a line in the U-W parametric plane

3.3.8 Patch Splitting

In engineering design of curved surfaces local refinements are necessary. A facility is provided to allow the designer to accomplish this task. The designer can split (subdivide) his patch to a smaller patch at the location he wants to do local refinements, then he can do whatever modifications he wants on the smaller patch. Using this technique the rest of the original patch surface would not be affected. Considering Figure 3.20, if the designer wants to create patch P_2, by subdividing patch P_1, all that he is asked to do is to define values for the u, parameters corresponding to the four corner points of patch P_2, i.e., u_1, v_1, u_2, v_2. The user first hits the "SPLIT" option on the CRT menu area, then the program responds: DEFINE THE PATCH TO BE SPLIT; the user hits the patch to be split with the light pen; the program responds: "ENTER U1, U2, V1, V2 VALUES"; the user enters (e.g.) 0.1, 0.4, 0.1, 0.4; the program dynamically generates the new patch (P_2) which has geometric properties similar to the given patch. Figure 3.21 shows a typical splitting operation with .1, .5, .1, .5 for U1, U2, V1, V2 respectively. The mathematics involved in creating the boundary (B) matrix of the patch generated by subdividing a given patch is illustrated in Appendix IV.

3.3.9 Hidden Line Removal

The algorithms available for elimination of hidden lines in 3-D are among the most interesting in graphics work.

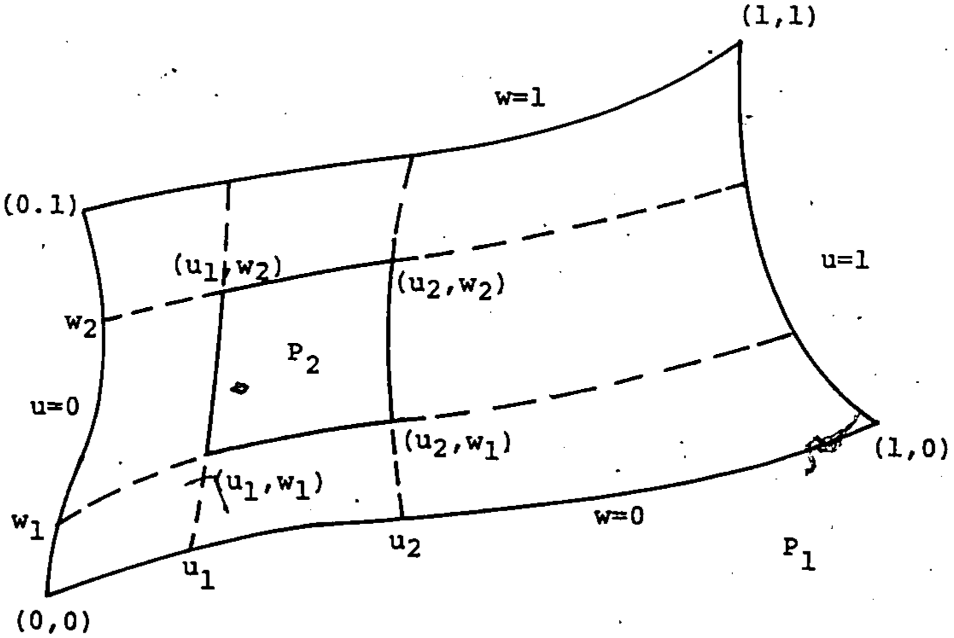


Figure 3.20 Geometry for Creating Patch P_2 by Splitting Patch P_1

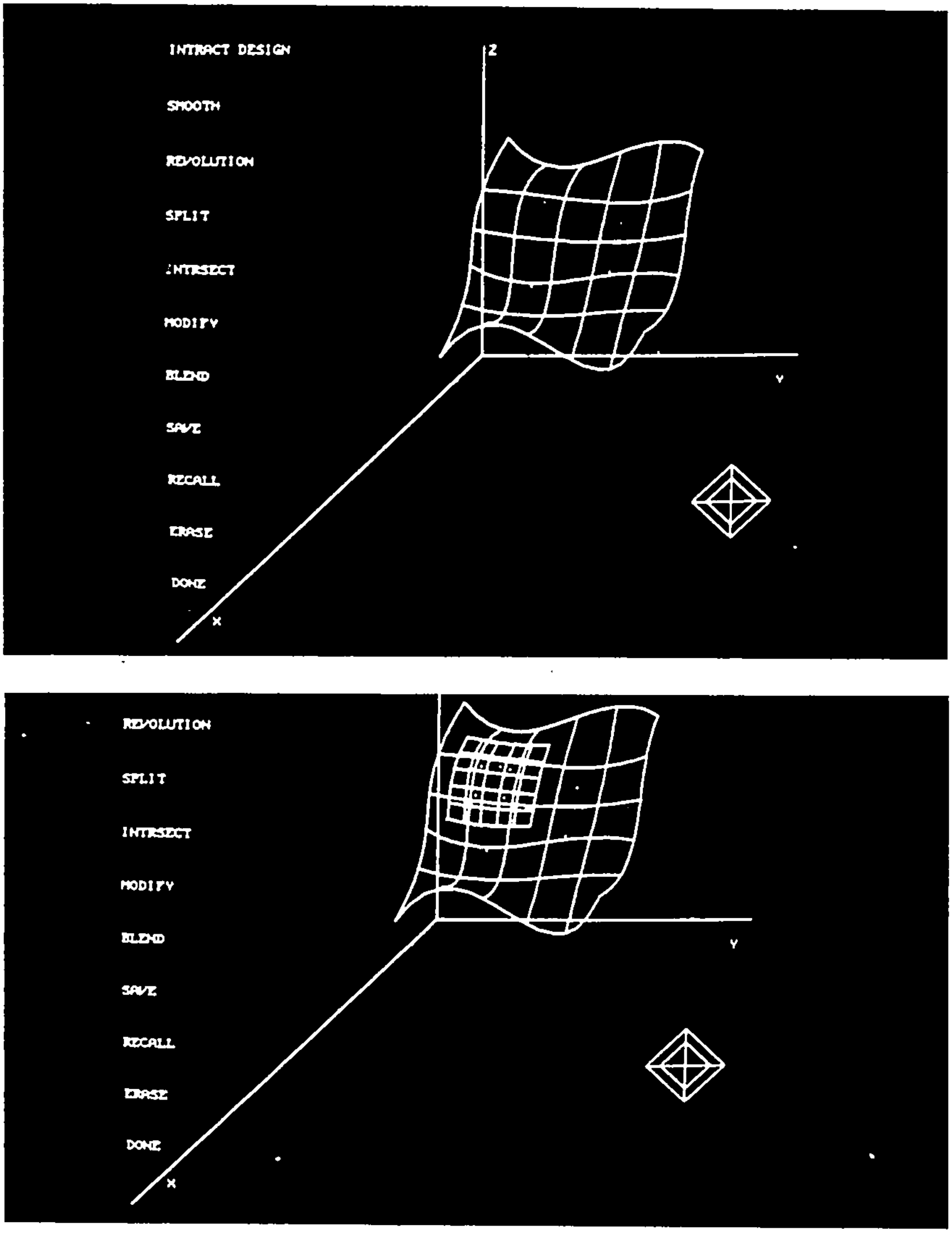


Figure (3.21) A patch before and after "splitting"

The resulting displays are very attractive views of the scene which aid the designer visualizing his design on a CRT graphics terminal. The single disadvantage of hidden-line elimination is that considerable computation is required to decide which lines are hidden. A great deal of effort has been applied to solving the hidden line problem and the result is a sizeable collection of algorithms [44, 45, 46].

In this thesis no special attempt has been made to design a special hidden line removal program, instead a program called "HIDE" designed by H. Williamson and modified by M. Vannier and M. Oliff [47] has been adapted to our 3-D surface design program. "HIDE" is a FORTRAN-callable subroutine used to generate a 2-dimensional representation of a 3-D figure or surface. Subroutine HIDE is called once for each line to be plotted. The first line is plotted in its entirety. Only that portion of subsequent lines that is visually above (optionally below) all previous lines will be plotted. The result is an orthographic projection with hidden lines eliminated. A typical design for a designed surface after hidden line removal is shown in Figure 3.22 .

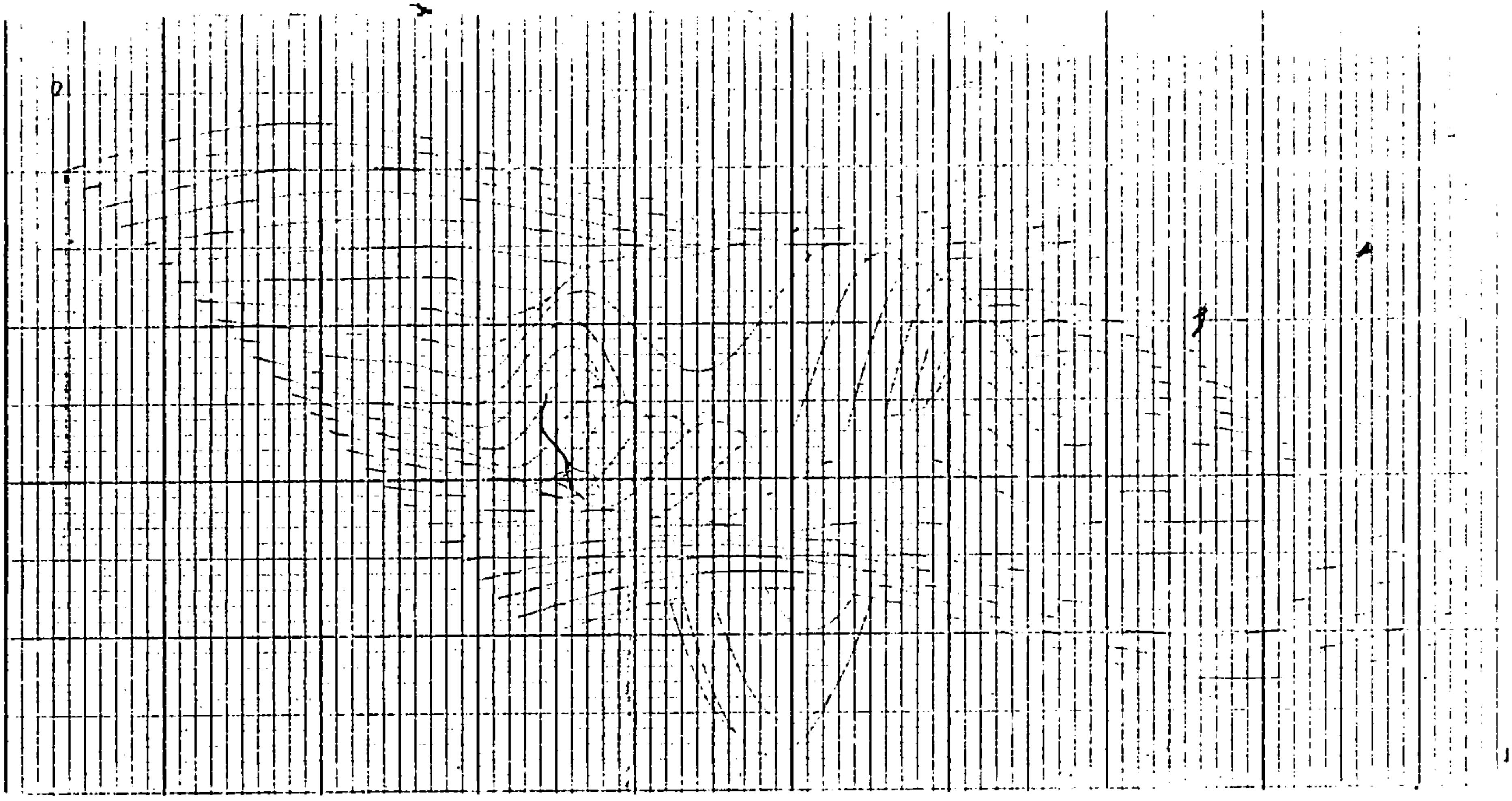


Figure (3.22) A designed surface with hidden lines removed; (hard copy from a CALCOMP drum plotter)

CHAPTER 4

SOFTWARE TECHNICAL INFORMATION

4.1 Introduction

Programs are designs, and software engineers began to realize that design techniques are very important in software design. Advanced programming techniques [75], have been used to develop the software of this thesis. Top-down and structured [76] programming techniques were used. The software system is composed of an executive or main-line program which acts as the control-reading in data, making major logical decisions, and calling subprograms which do various jobs in the free form surface design package. The software is also "modularized" to ensure the independency of modules. The structure of the program modules is described in hierarchical input-processing-output [75] charts, as shown in Appendix A. The function of each module is briefly described as well as the input and output of each.

Three main programs were designed to perform the design and manipulation of curved surfaces. Each program is capable of performing the design and manipulation of curved surfaces; the only distinction is the kind of input devices used. Program JOLIA was designed to accept input from the LP; program KEY to accept input from the keyboard; and program DATAFL to accept input from data points stored on data files.

A two-dimensional curve design program CURDES is also included in Appendix A. The program can be used for the interactive design of two-dimensional curves of any type.

4.2 Program Specifications

The specification is the document which fully defines the requirements of the design of the software. The program design was based on the following specifications:

1. The General Objectives of the Software Package

A CAD package is required for the interactive graphics design of a free form surface.

2. Technical Level of Users Related to Programming and Modeling Skills

User should not require any familiarity with programming skills or mathematical surface modeling techniques.

3. Input and Output Software Configuration Needed for Use of the Package

The only input data permitted is x, y, z coordinates of spatial points lying on the surface. The output should include data files defining the designed surface.

4. Input and Output Hardware to be Used

A PDP-11/34 mini-computer, with a refresh type (GT46) graphics terminal CRT. Input media is floppy discs or, DEC packs (RK05).

Input - DEC WRITER II, Light pen.

Output - CRT

5. Core Memory Available for the Package

32 K words.

CHAPTER 5

5.1 Ship Hull Design

The preliminary design of a ship, is common with that of most other engineering objects, involves careful. compromise between a number of factors in order to produce the most economical result satisfying the functional requirements of the design. In the design of large commercial ships the basic functional requirements are usually stated as the ability to take a given load a certain distance. Thus, in the initial specification of the required ship, usually the parameters available to the ship builder are speed, length, beam, draught, deadweight, together with the class of ship, whether it be a tanker or cargo ship. From such sparse data, the hull surface definition has to be built up. The only geometrical information that is given for the surface specification is the length, beam and draught. Making use of the previous data, the traditional method of hull design was to define the form approximately by a series of points through which the surface would pass [48]. Initially these might define a set of smooth curves representing vertical plane sections across the hull. If these curves were intersected by horizontal plane sections one could plot sets of points representing the shape of the hull at various waterplanes.

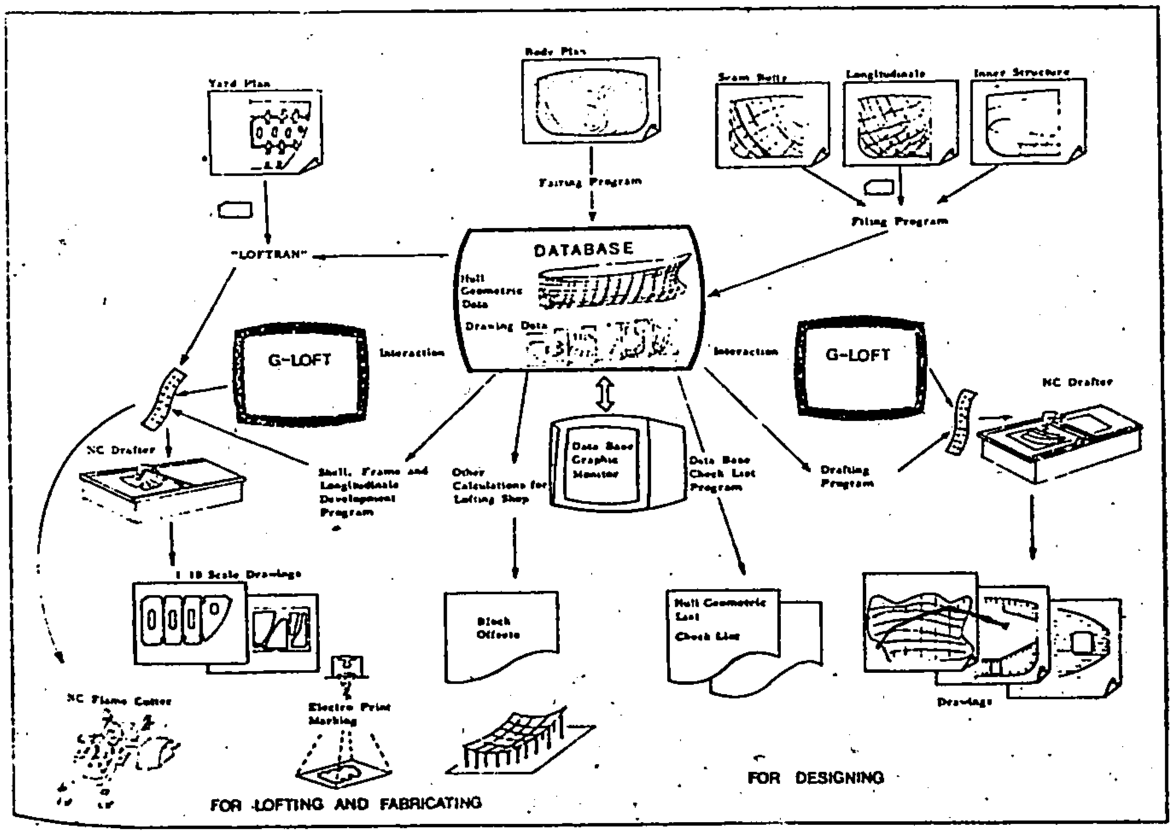


Figure (5.1) The NK-ASD CAD system for overall shipbuilding operations, from design to building

In general, curves through these points would not be smooth. The process of fairing then begins and consists of reconciling the two sets of curves until both are smooth and both represent intersections with the same surface. Usually this was a tedious and time consuming task. Most of this recent effort has been devoted to the 'hull fairing' problem in which a given lines plan is to be approximated mathematically, often being improved in fairness at the same time. D. Taylor [49] mathematically generated the hull forms in his standard series, defining sectional area curve and design waterline by fifth order polynomials in accordance with form parameters* he prescribed. Weinblum [50, 51, 52] extended the principles of parametric lines creation in connection with the systematic variation and hydrodynamic optimization of hull forms.

More recently two distinct goals have been pursued by numerous investigators: Lines creation by 'distortion of a parent hull form' versus lines creation from 'given hull form parameters'. In the former category significant contributions were made by Lackenby [53], Schneekluth [54], and Puchstein [55]. With the latter problem, good progress was also made over the last two decades owing to research efforts by Thieme [56], Miller and Kuo [57], Williams [58], Kwik [59, 60], Kuiper [61], and Reed and Nowacki [62] and others.

Following these traditional efforts for ship hull

---

* See Appendix B

design, MacCallum [63, 18] developed an excellent system for the preliminary design of ship hulls using interactive graphics. However, MacCallum's system had some disadvantages

1. The design consisted of an arrangement of surface patches joined along common boundaries and made continuous only to the first degree. In some cases a second degree continuity is necessary.

2. Curves of intersections (body plan) could not be obtained dynamically.

3. Design could not start from scratch; however it could be started using a library of topologies, i.e., standard types of bow forms, types of sterns, etc. The program can be left to construct an initial form by piecing together items from its library of standard forms. If the need arises to define a new topology the program is handicaped.

4. Shape modification was achieved through 'dragging' one of the corner points of a patch using a light pen, but any other control point on the patch's surface could not be so dragged. If this was required, the original patch was split at the required control point and then the regenerated patch was changed.

Another interesting system was designed by Yuille [64]. The user has to deal with parametric derivatives and cross derivatives, which makes the system difficult to use. The program is not suitable for making large changes to the shape of a patch in one step because it is not practical to move points on a patch very far in a given coordinate direction by altering values of the cross-derivatives [10]. One can, however, obtain good results by making a series of small changes in different directions.

Walker [65] argued that the best way to define a ship's hull is through the use of a close mesh of points rather than analytic surfaces.

Groot [66] presented a design method together with practical results of designed hull surfaces composed of simple analytical functions. Still, the method is not capable of designing any desired curved surface.

It is worthwhile here to discuss the CAD system for shipbuilding at the Japanese 'Chita' shipyard, the most modern shipyard in the world [68]. In 1978, the shipbuilding division of Nippon Kokan KK, one of the major shipbuilders in Japan, completed a comprehensive CAD system for shipbuilding operations from designing to building [67]. The NK-ASD system features are a database system, improved usage graphic display equipment and direct control of NC equipment. In the NK-ASD system the common data, such as hull configuration and structure, are filed in a database and can be extracted at will. Figure 5.1 shows a functional outline of the system. The major programs are as follows:

1. Fairing program for calculating the outer hull configuration.
2. A filing program that files the detailed structure of the vessel into the database.

3. LOFTRAN for producing the control tape for NC parts generation.

4. G-LOFT for parts generation by interactive graphics.

5. The database graphic monitor program.

6. The shell development program.

7. A program providing a checklist. The graphic display unit allows designers to extract information from the files. The G-LOFT program helps preparing drawings of design and parts generation. The graphic monitor program helps the user to study the contents of the database in numerical or graphic form. One of the great strengths of the NK-ASD system is that the information concerning the design is directly connected to the manufacturing processes.

A new method is introduced in this thesis for ship hull design. The method makes use of the free form surface design program explained in Chapter 3. When using the program one is concerned with direct manipulation of a surface from the start; and not with curves. The naval architect interacts with the computer, which stores the information and then calculates and draws curves in the current designed surface. The procedure, when using this program to design a new hull, is first to sketch the vessel and the outlines of the patches that will form a preliminary representation of its hull. A new surface is computed and stored with its patch corners at positions corresponding to those in the sketch. Actually, the designer can start designing his hull from scratch or by modifying a previously stored surface. The designer can easily obtain true views of plane cross sections across and along the new surface representing the hull. He can also commence an iterative procedure during which he repeatedly modifies the surface by dragging control points; and he then examines the shape of the plane cross sections; repeating this until he obtains the shape he requires.

In addition to the fairness of the hull form the program could be very easily extended to accommodate some analysis routines which will aid the naval architect in the design process, e.g.,

1. Subroutines to calculate the resistance of the underwater body to motion through the water.
2. Subroutines to calculate the flow (pattern) of water around the vessel and into the propellers.
3. Subroutines to calculate the displacement and center of buoyancy of the underwater form.
4. Subroutines to measure the static and dynamic stability of the ship.

In the next section we will describe the general algorithm of the ship hull design program from a user's point of view.

5.1.1 Ship Hull Design Algorithm

The general approach of the program will be as follows:

1. User inputs data from which major dimensions could be

calculated.

Service Speed (Knots)

Deadweight (tons)

Deadweight coefficient (deadweight/light weight + deadweight)

2. Program Calculates L, B, H, D as shown in Figure 5.2.

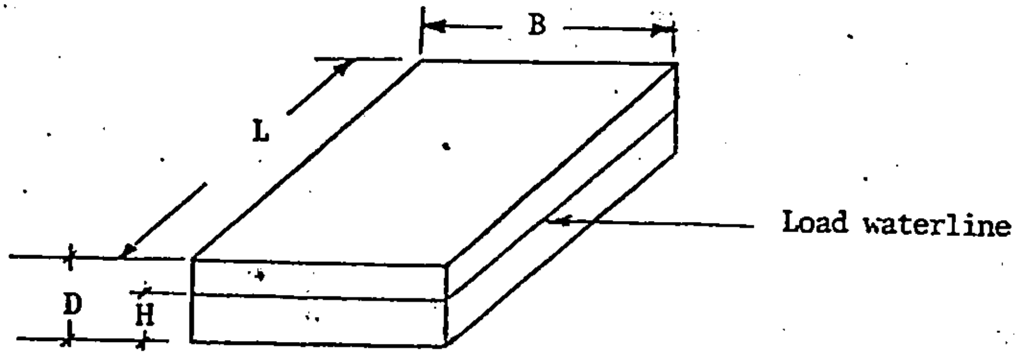


Figure 5.2 Box shaped vessel

(Note: User would be able to change any of these parameters).

3. Program displays the box shaped vessel showing the load waterline.

4. User is asked to divide the box into four major divisions, using the light pen and tracking object, as shown in Figure 5.3.

F - Forward
M - Middle
A - Aft
S - Stern

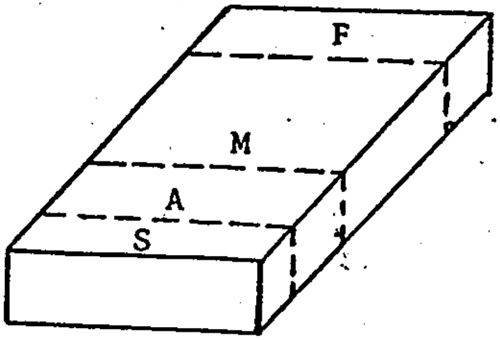


Figure 5.3 Divided vessel

The program returns the length of each division. If the user wants to change any length he could go back and repeat the previous step (Figure 5.3).

5. The user is asked to define curves (including the midship section curve) resulting from 'slicing' the ship with vertical planes through the two edges and the dotted lines of Figure 5.3.

5.1.2 Designing the Middle Body

1. As the shape of the midsection is characteristic of size and form as well as function, the design process should begin by designing the middle body (M).

The user is asked to define the shape of the midship section (using the light pen and tracking object the designer enters points defining the midship section curve, using the same algorithm for the free form surface design program described in Chpater 3, section 3.3.2). The boundary of the curve is the rectangle shown in Figure 5.4. Due to the symmetry of the ship along the centerline, only half of the curve need be defined, see Figure 5.5.

2. After the user defines the shape of the midship section the program returns the value of the "midship section coefficient" (C_m)

$$C_m = \frac{A_m}{A}$$

If the value of C_m is different than what the user

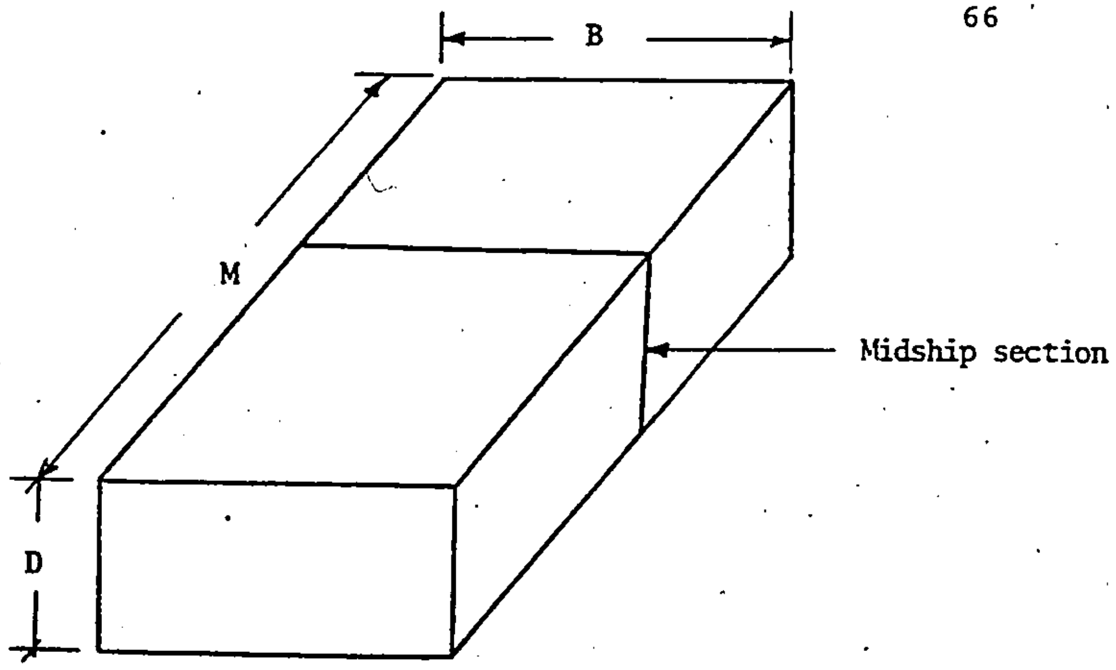


Figure 5.4 Middle body of a ship

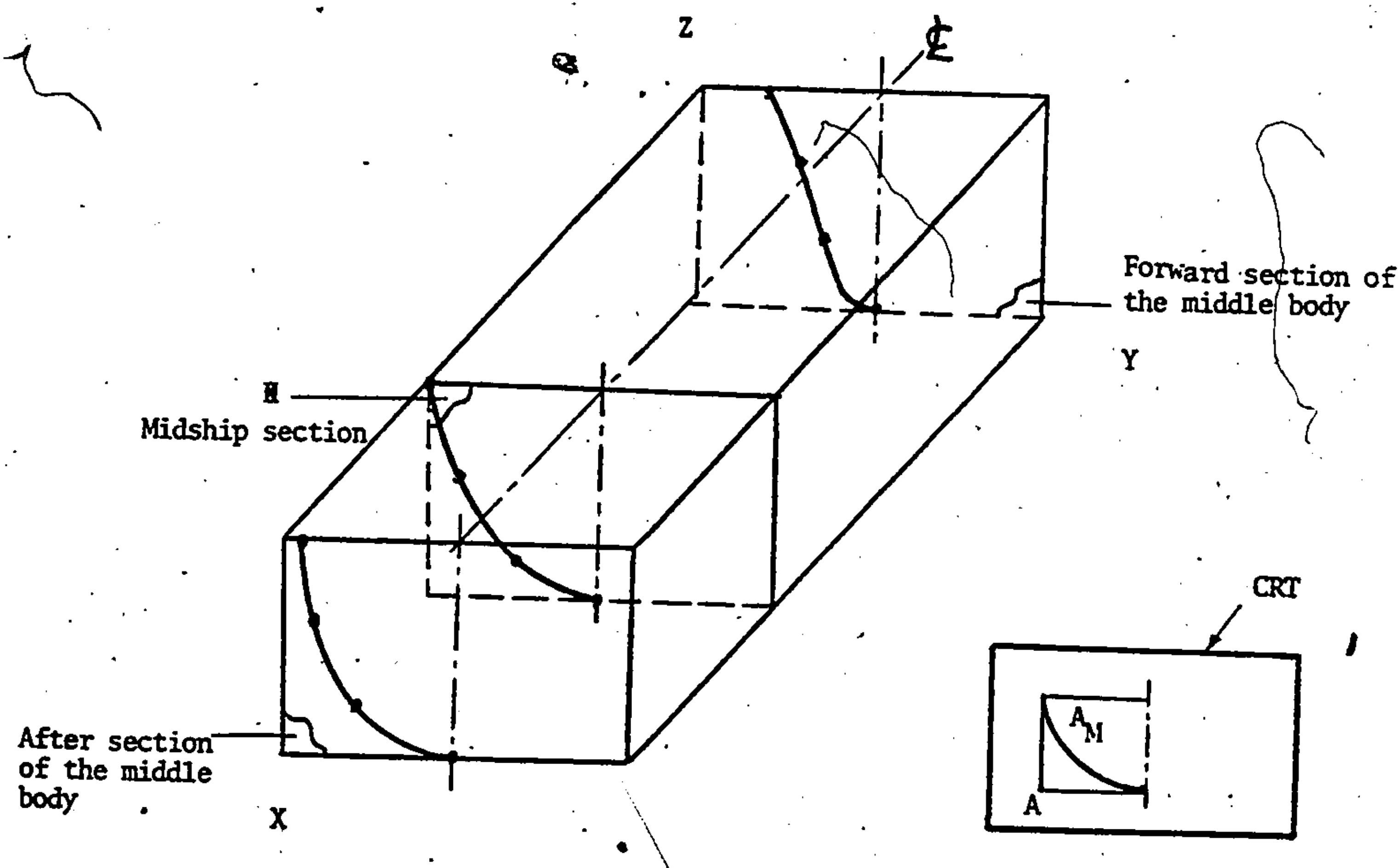


Figure 5.5 Middle body boundary curves

has in mind, he could go back and redesign the curve.

3. The user is then asked to define the curve of the forward section of the middle body using the same procedure as the previous step, see Figure 5.6.

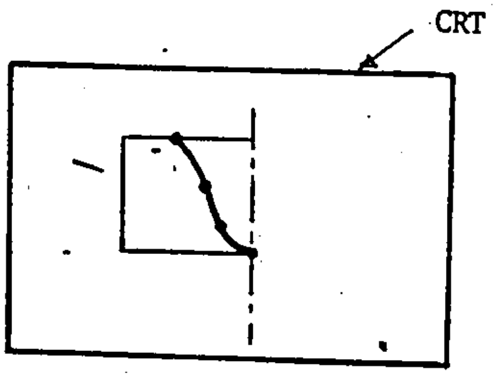


Figure 5.6 Curve of the forward section of the middle body

4. The two previously created curves are then plotted on the 3-D space on the Y-Z planes (Figure 5.7).

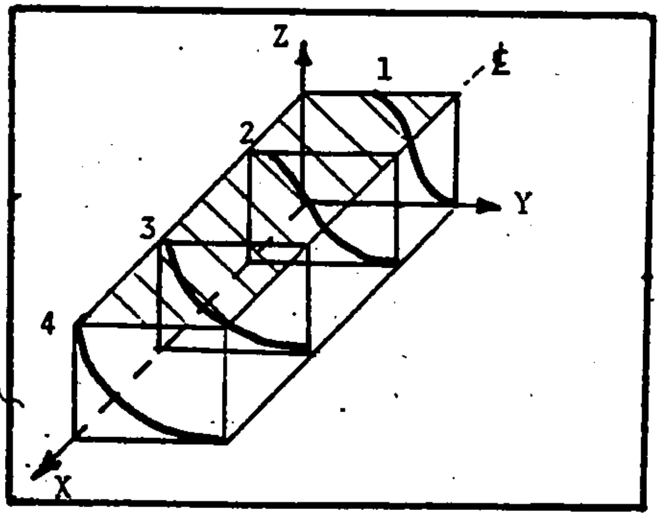


Figure 5.7 Forward part of middle body sectional curves

5. The user is asked to define (using the LP and tracking object) two additional curves in two parallel planes so that curves 1, 2, 3 and 4 (four defined points/plane) will now define the patch.

6. The resulting patch is then dynamically plotted on the CRT (see Figure 5.8).

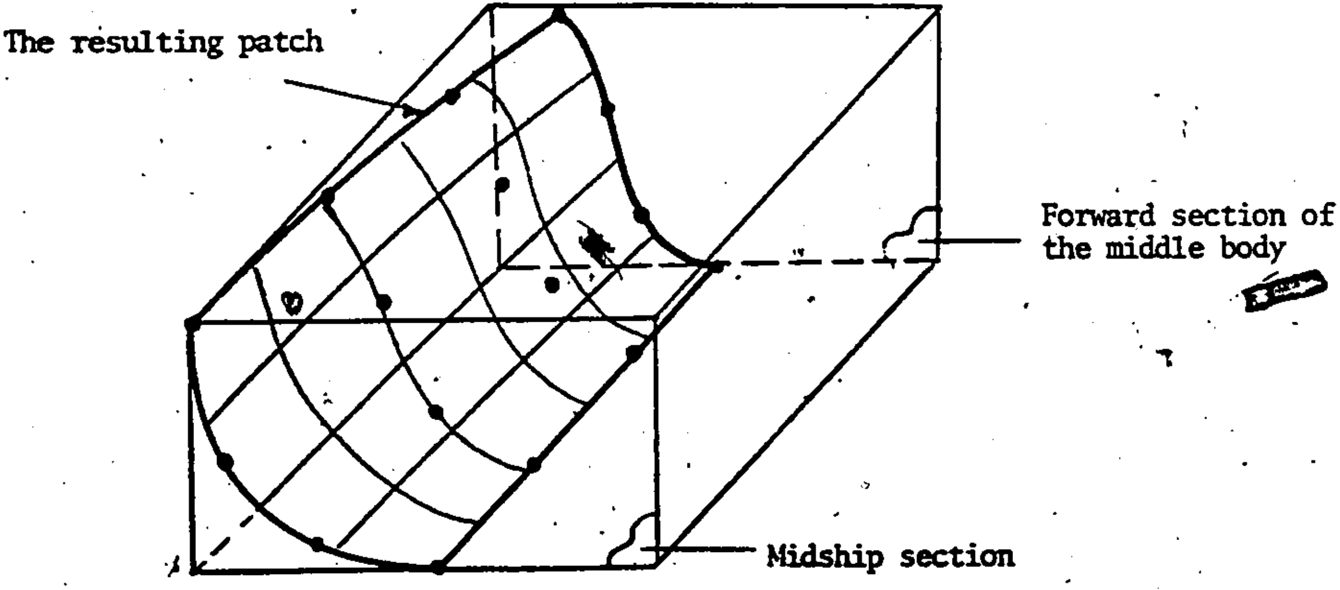


Figure 5.8 Resulting patch of the forward part of middle body

7. If the user is not satisfied with the shape of the patch, he could go back and change the position of any of the points defining the patch until he is convinced with the shape of the patch.

8. The same algorithm could be used to develop the surface of the afterpart of the middle body, making sure that continuity exists along the common border between the forward and afterparts (patches) of the middle body. The

resulting middle body will look like Figure 5.9.

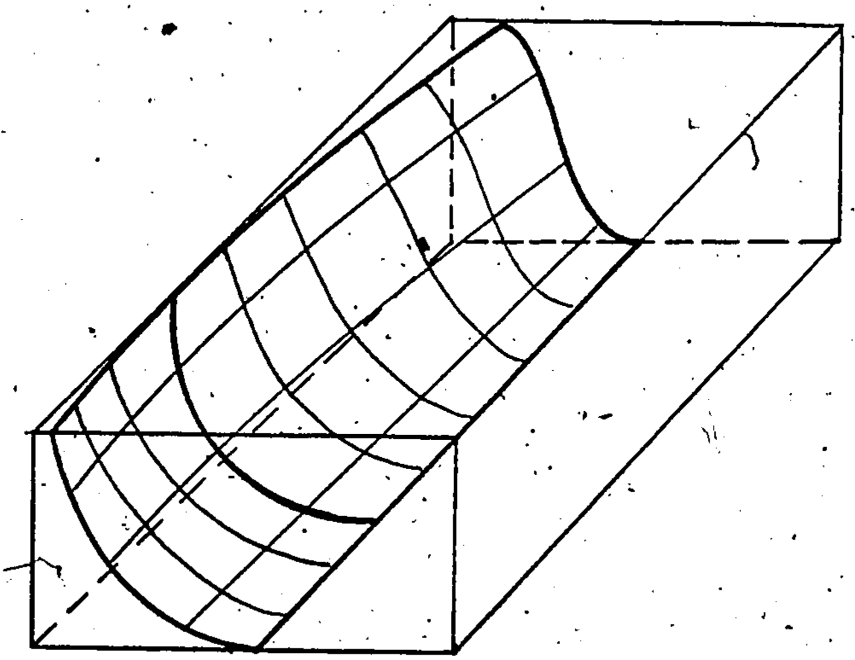


Figure 5.9 Middle body surface

5.1.3 Designing the Forward Part [F]

The user is asked now to define only three sections of the forward part (the after section of the forward body has been defined in designing the middle body), see Figure 5.9.

Notcie that the forward section of the forward body has a bulbous bow. Actually the user would be able to define any shape for the forward part.

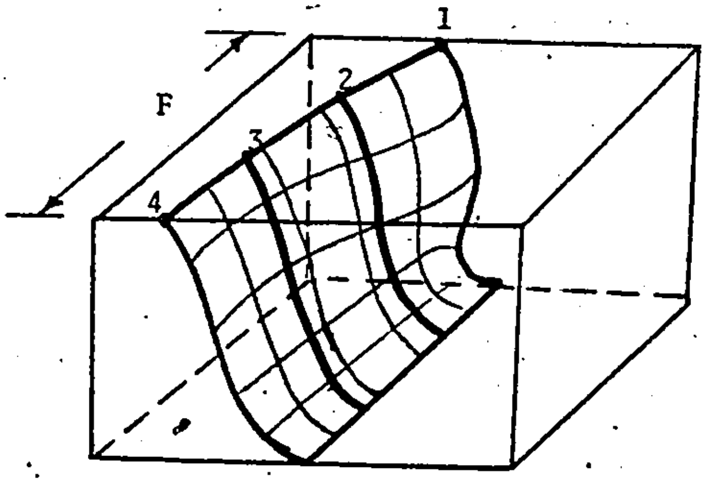


Figure 5.10 The Forward surface

5.1.4 Designing the Aft Part and Stern [A], [S]

The same procedure developed in the previous sections could be used to define the 'Aft', and 'Stern' parts of a ship.

Figure 5.11 shows a complete 'top-down' program design for ship hull surface design. A complete ship hull design using the prescribed algorithm is shown in Figure 5.29. True views of the curves of intersections, i.e., body plans are also shown in Figures 5.23 to 5.27.

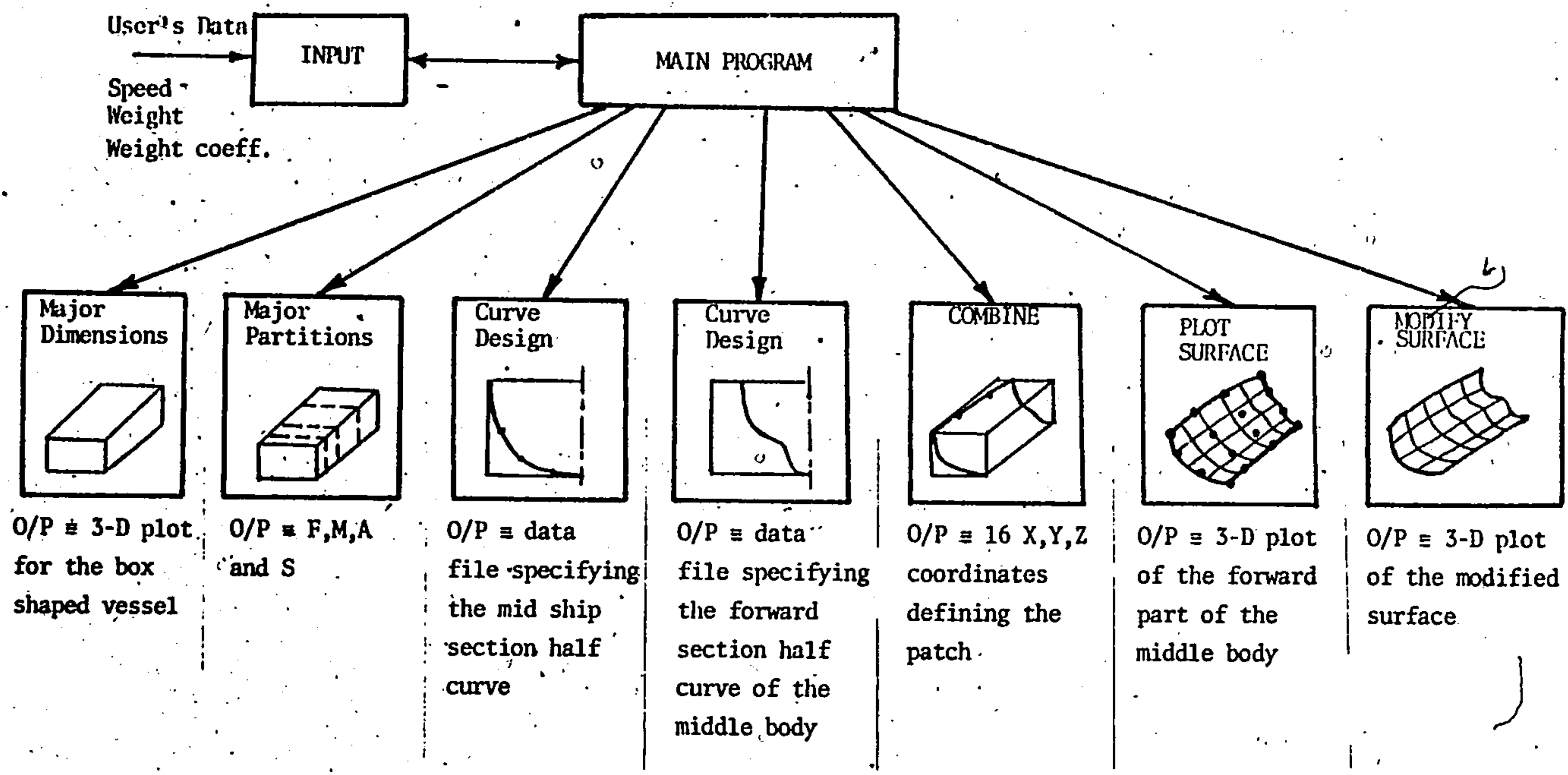


Figure (5.11) HIPO chart for the hull design program

5.2 Chair Design

The main steps taken to design a chair are as follows.

1. The industrial designer has some new idea for designing a chair. He sketches the new shape on a piece of paper as shown in Figure 5.12.

2. The designer begins by subdividing his surface into patches whenever he feels the need to define a certain part of the surface by a patch.

3. The divided chair would look like Figure 5.13, (note that the designer is still working on a piece of paper).

4. The designer is now ready to define to the computer each patch using the light pen input command.

5. To define patch number one, the user would define 16 points (four per working plane) as shown in Figure 5.14. The program will then display the surface fitting these points.

6. Using the same four working planes (parallel to X-Z plane), the user would continue defining patch number two. Note that point number one of patch 2 should be at the same position as point 4 of patch number 1, and similarly for points 5 and 8, 9 and 12, and 13 and 16 respectively.

7. After the user is finished with defining his five patches; using the prescribed four working planes; he can now alter the position of any point (node) and see the effect on the chair's shape until he is satisfied. He can then 'SAVE' the display and create a data file containing the X, Y, Z

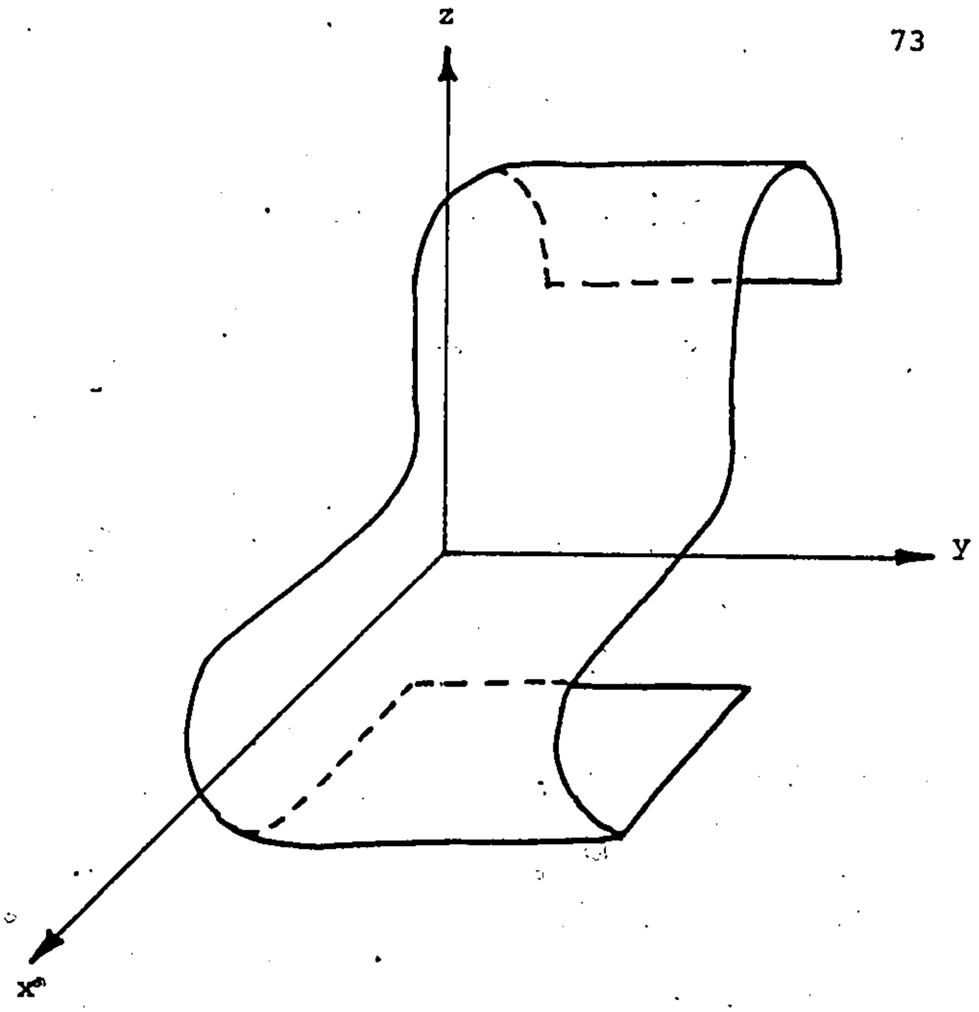


Figure (5.12) 3-D view of a chair as sketched by a designer

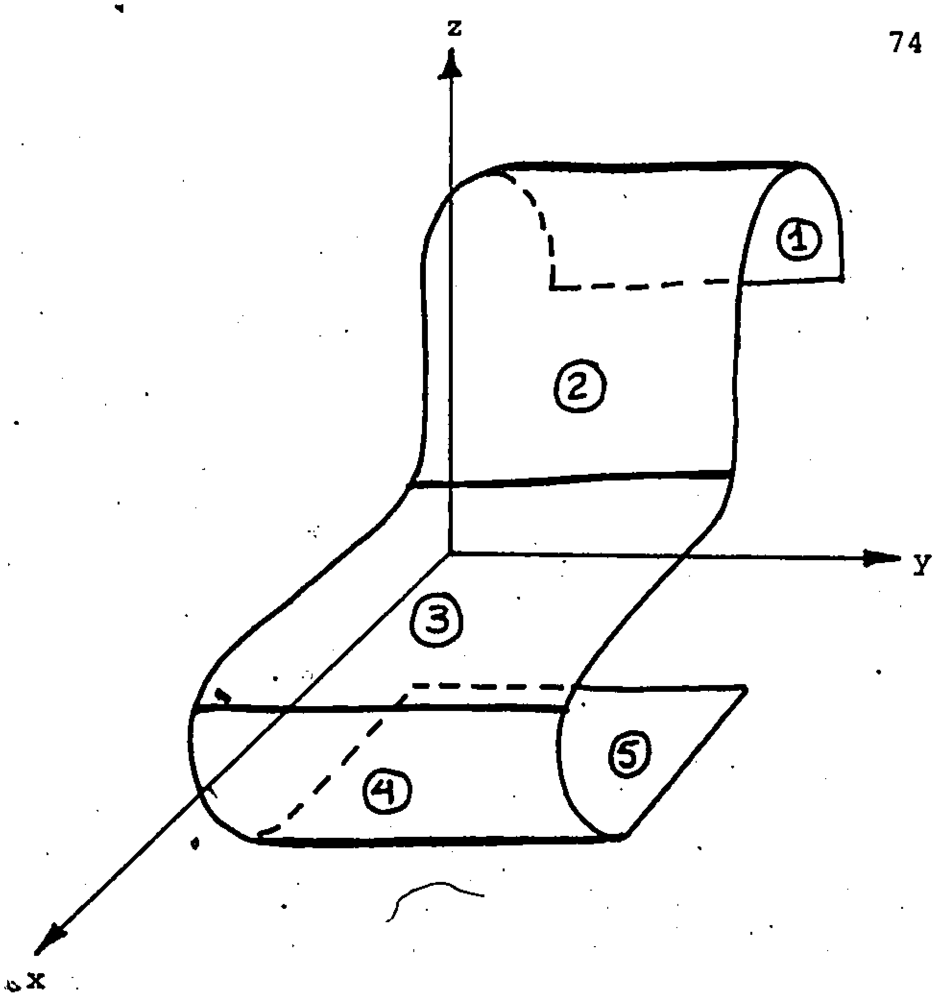


Figure (5.13) Chair divided into 5 patches, as sketched by the designer

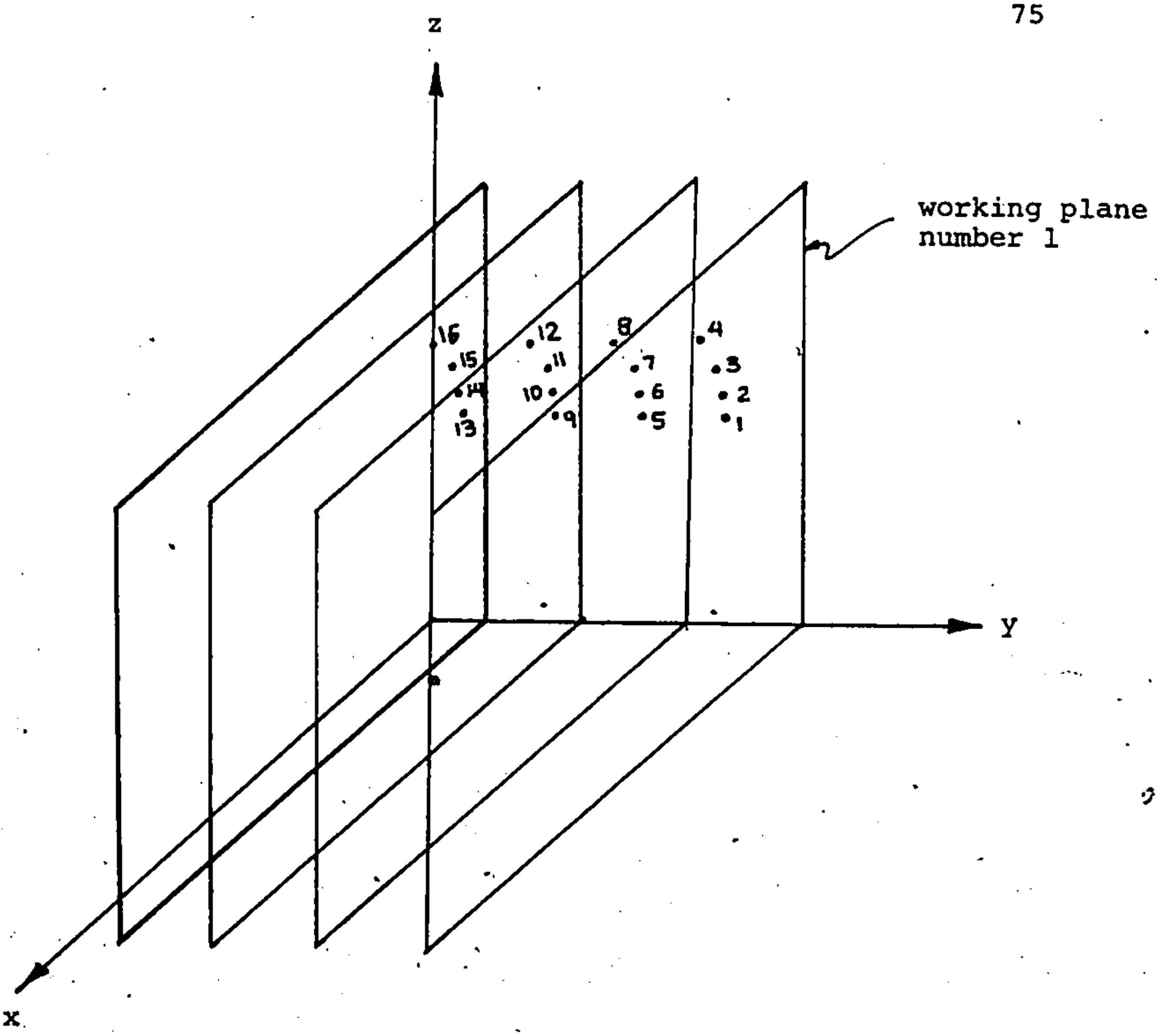


Figure (5.14) 16 points defining patch number one, 4 points per working plane coordinates of the points defining the chair's surface.

A typical chair design as designed on the CRT graphics terminal using the prescribed algorithm is shown in Figure 5.15.

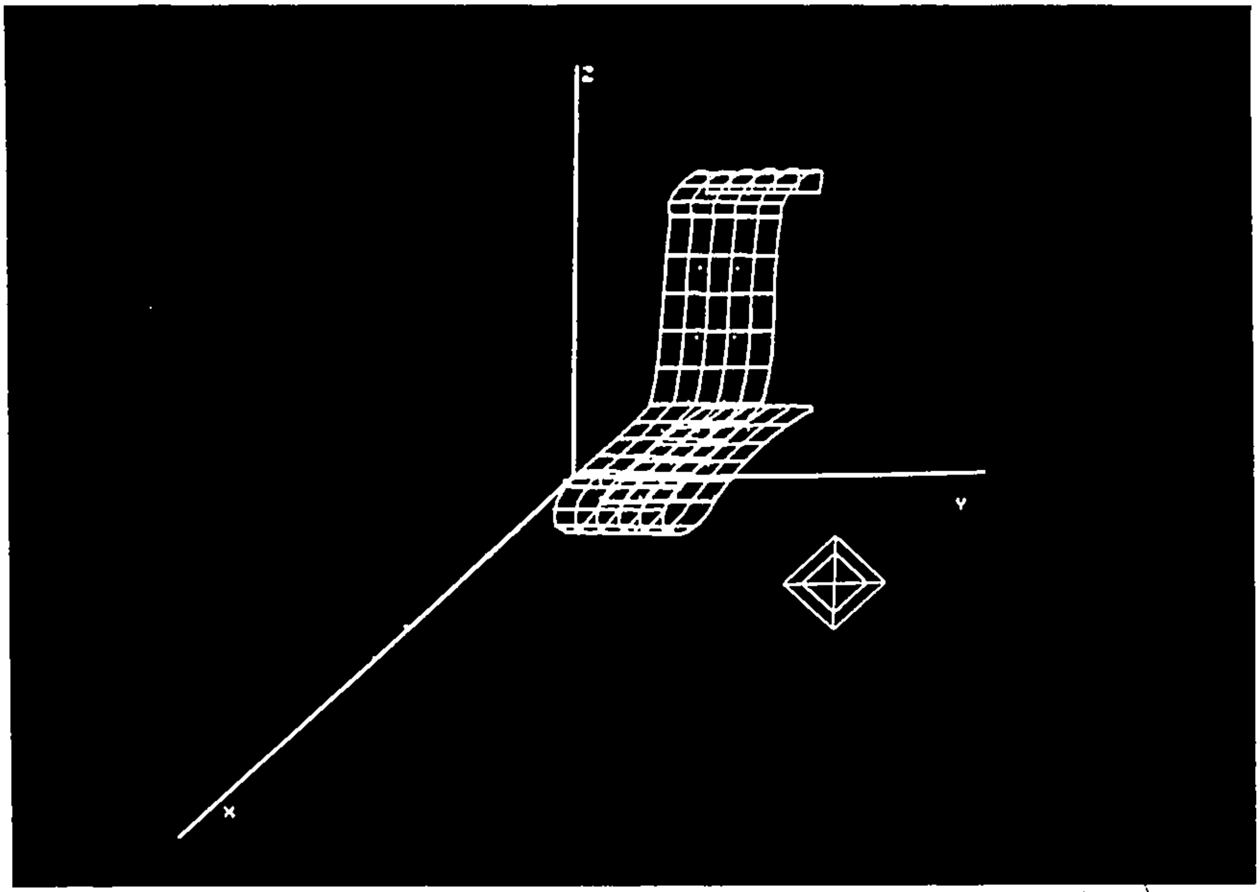


Figure 5.15 Three-dimensional view of chair design

5.3 Duct Design

A typical procedure for a duct design using digitized X, Y, Z coordinates as input data is explained in detail in the following section:

1. The designer draws, elevation and plan for the duct as shown in Figures 5.16 and 5.17.

2. From both figures the designer can easily obtain X, Y, Z coordinates of 16 control points of each patch as shown in Figure 5.16.

3. Using input via keyboard command, the user can now enter the 16 X, Y, Z coordinates of each of the four patches.

4. The program will dynamically generate the surface of the duct as shown in Figure 5.18.

5. The user can 'SAVE' the display and create a corresponding data file. The user can also obtain curves of intersections and get the X, Y, Z coordinates of the points defining the intersection curves.

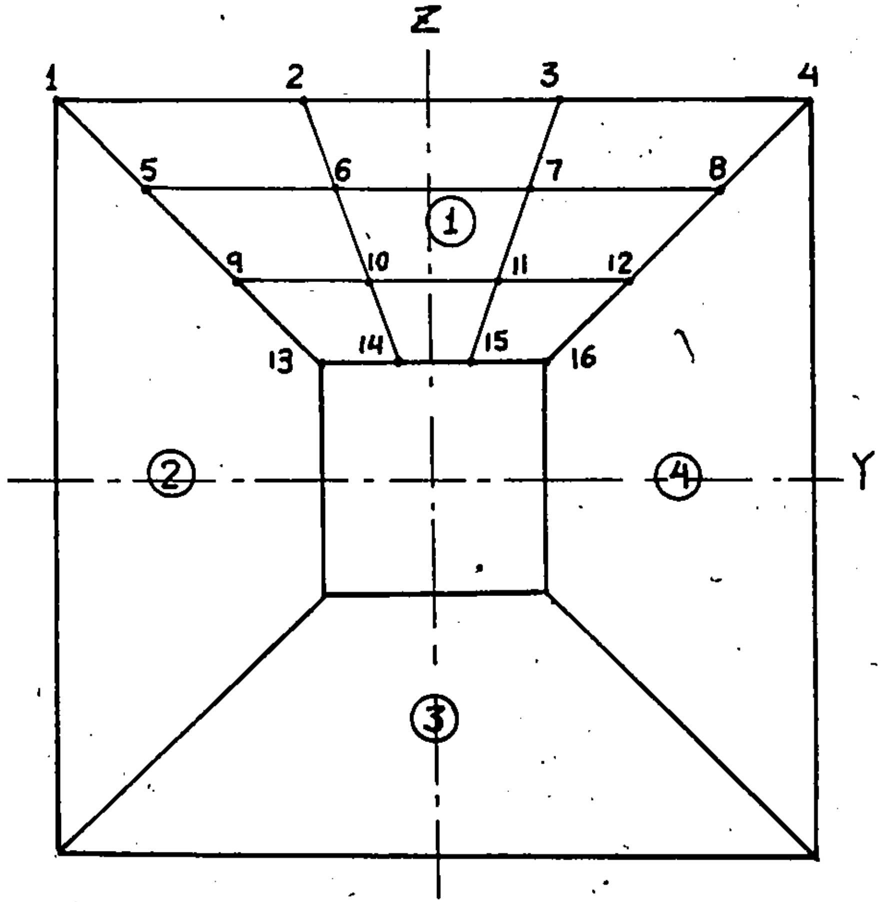


Figure 5.16 Elevation of the duct

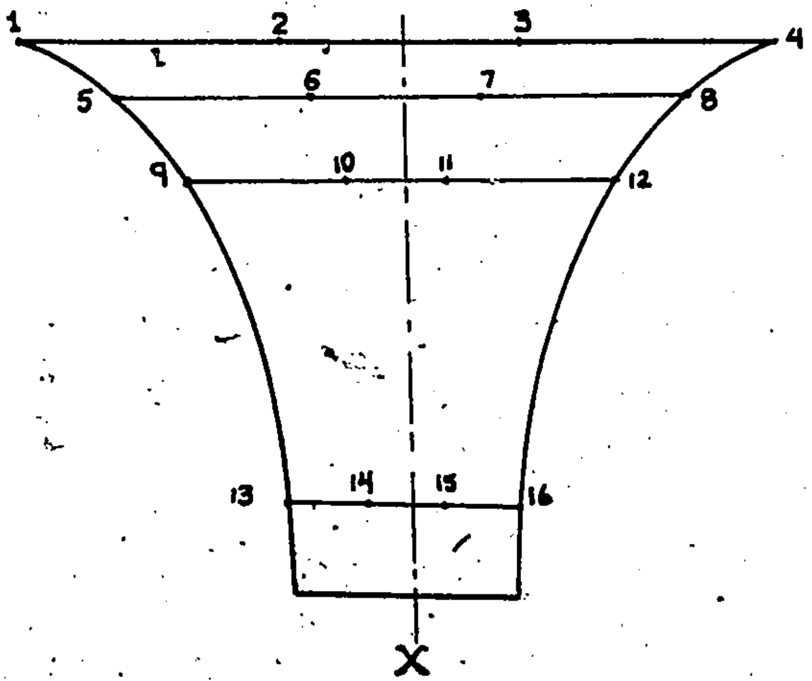


Figure 5.17 Plan of the duct

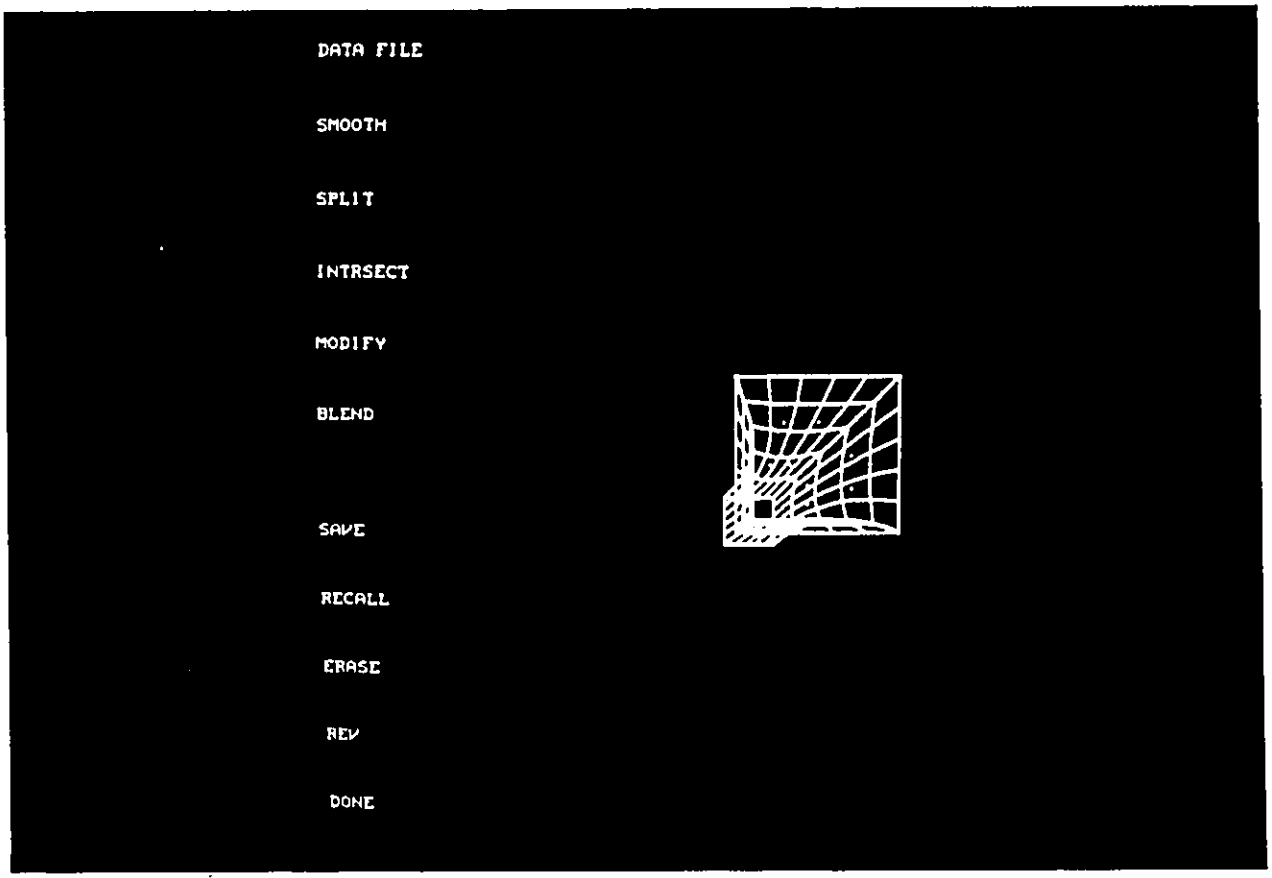


Figure (5.18) Duct surface as shown on CRT

5.4 Glass Container Designs

Simple round containers, such as bottles and wine glasses, can be completely specified with a single section line defining the profile of the container. The surface design program enables the designer to 'draw' the profile with a light pen (in reality, he is defining 4 points which construct a parametric cubic curve). This profile can be modified until the designer is satisfied and he can observe a pictorial view of the complete container, whilst he is working. The real power of this developed technique is in designing non-round containers from a predesigned round container (using control points dragging).

Designing round containers is very easy using the developed free form surface design program as will be proved in the following practical desing of a bottle:

1. The designer hits the 'REVOLUTION' option on the meanu area. The program responds by typing the following message: 'PLEASE ENTER NUMBER OF PATCHES'. Let us say that the user decided to use two patches to define his surface (neck and main body of the bottle). The user enters: 2.
2. The program responds: 'DEFINE WORKING PLANE, YOU SHOULD DEFINE 10 POINTS ON THAT PLANE. THE FIRST TWO DEFINE AXIS OF REVOLULTION, THE REST ARE USED TO DEFINE SURFACE'.
3. The user defines a working plane together with 10 points on that plane (as described in detail in Chapter 3, section 3.3.6).

4. The program dynamically generates the two patches defining the bottle's surface as shown in Figure 5.19.

Figure 5.20 shows a design for a non-round container, designed from a predesigned round container by dragging some of the control points defining the original surface.

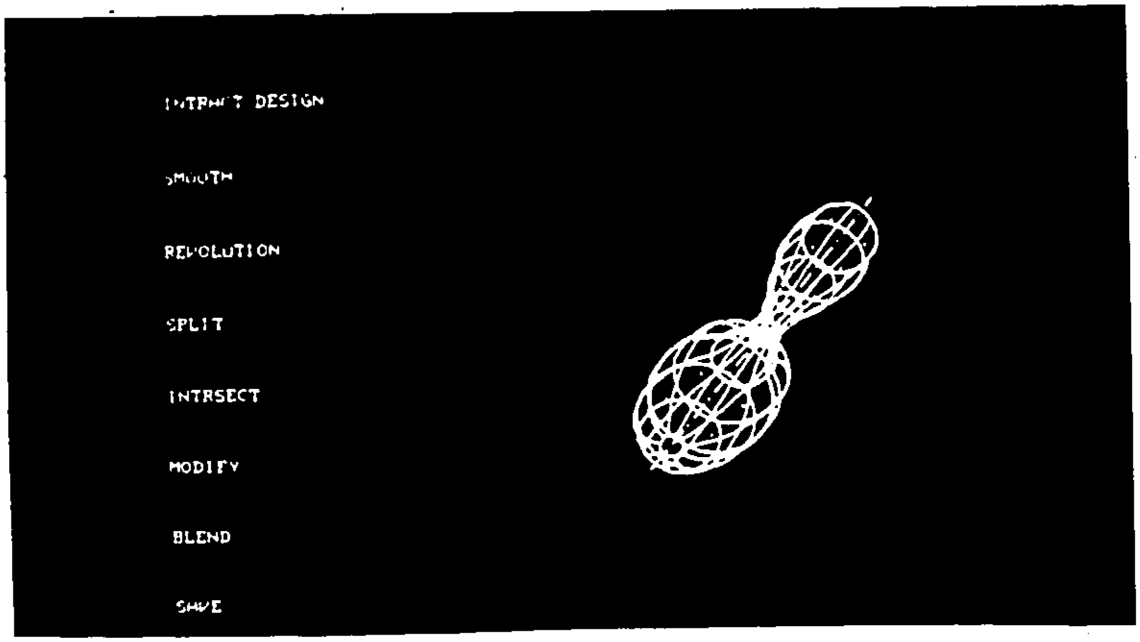


Figure (5.19) Design of a round container

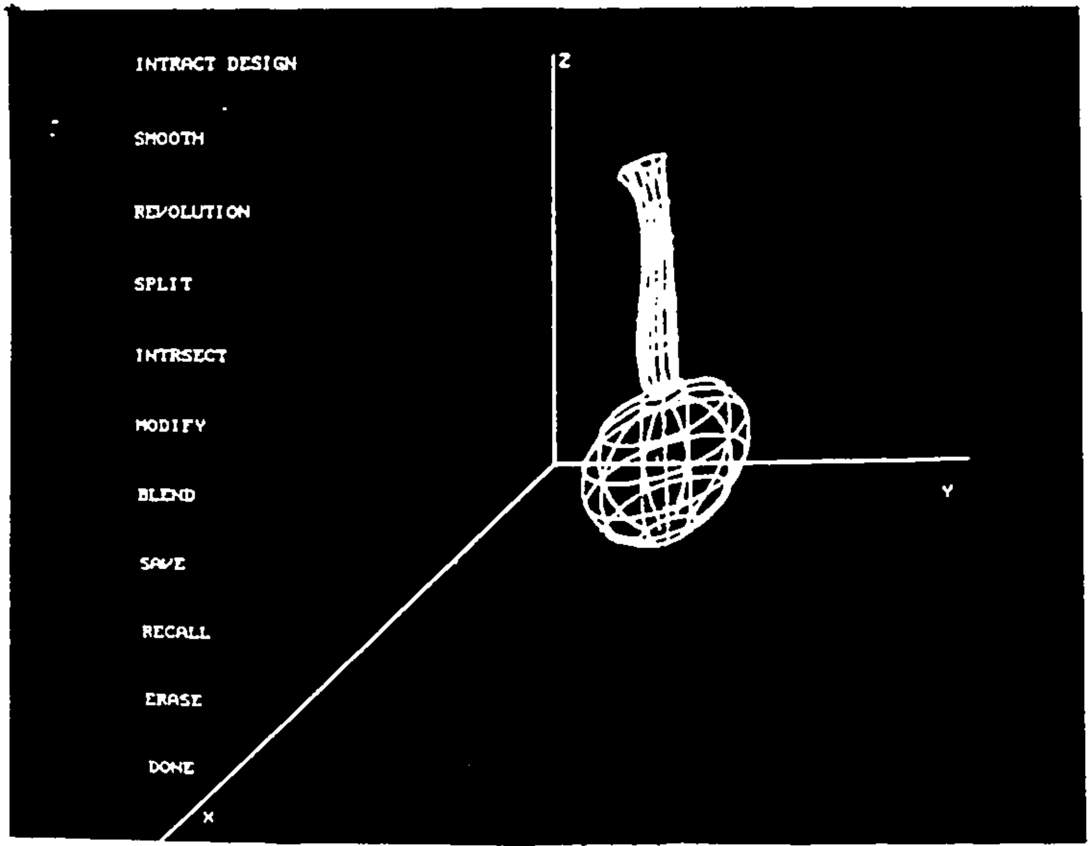


Figure (5.20) Design of a non-round container

5.5 Other Designs

Figure 5.21 shows a photo taken of the CRT graphics terminal, for a "nozzel" designed using only one patch.

Figure 5.22 shows a "smoking pipe", that was designed using only two patches.

Figures 5.23-5.29 show the different stages of a ship hull design.

The most interesting feature of the prescribed software free form surface design system is the short elapsed design time. The nozzel shown in Figure 5.21 was designed in less than one minute. The smoking pipe of Figure 5.22 was designed in less than 4 minutes. The ship hull of Figure 5.29 was designed in less than 10 minutes.

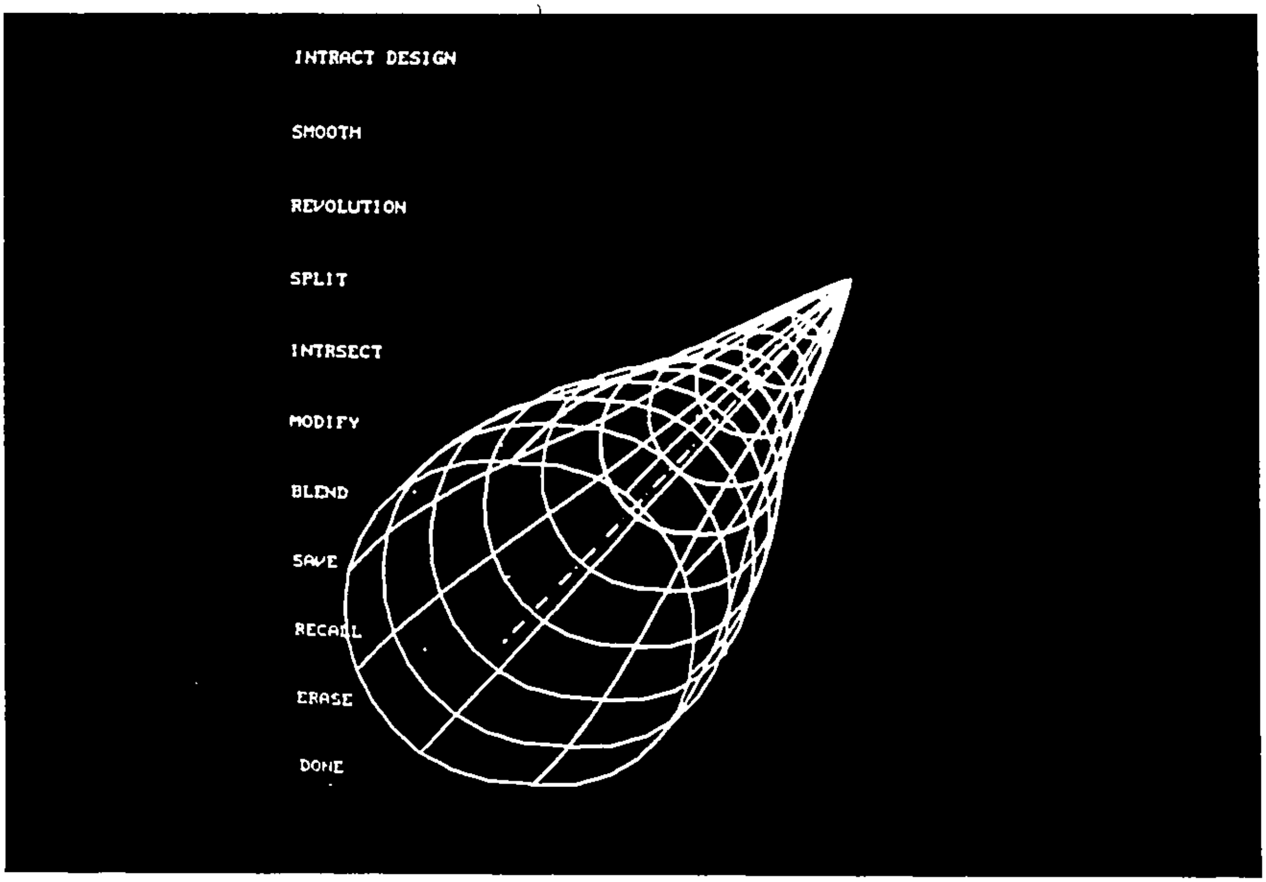


Figure (5.21) A nozzel design using only one patch

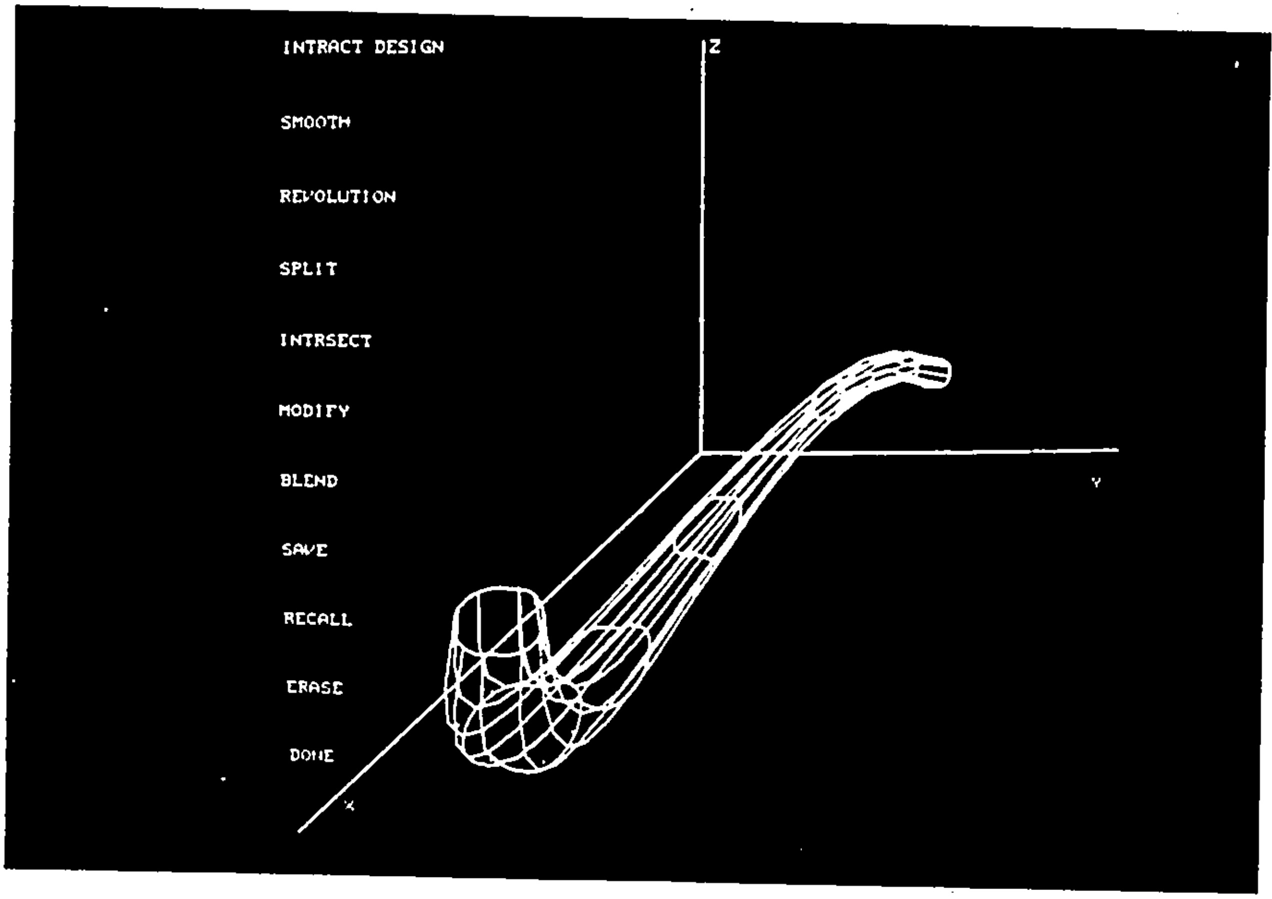


Figure (5.22) A smoking pipe design using only two patches

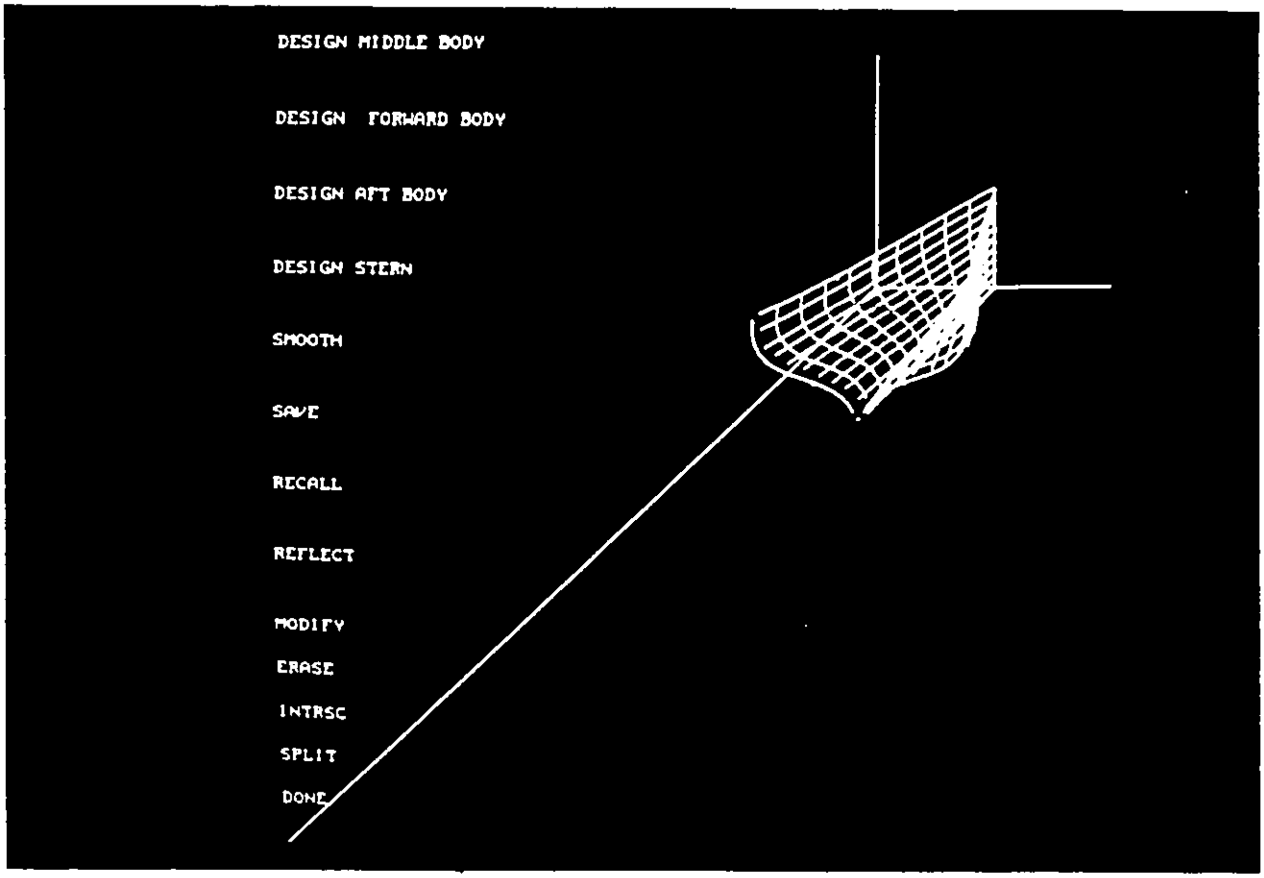


Figure (5.23) Design of the forward body of a ship

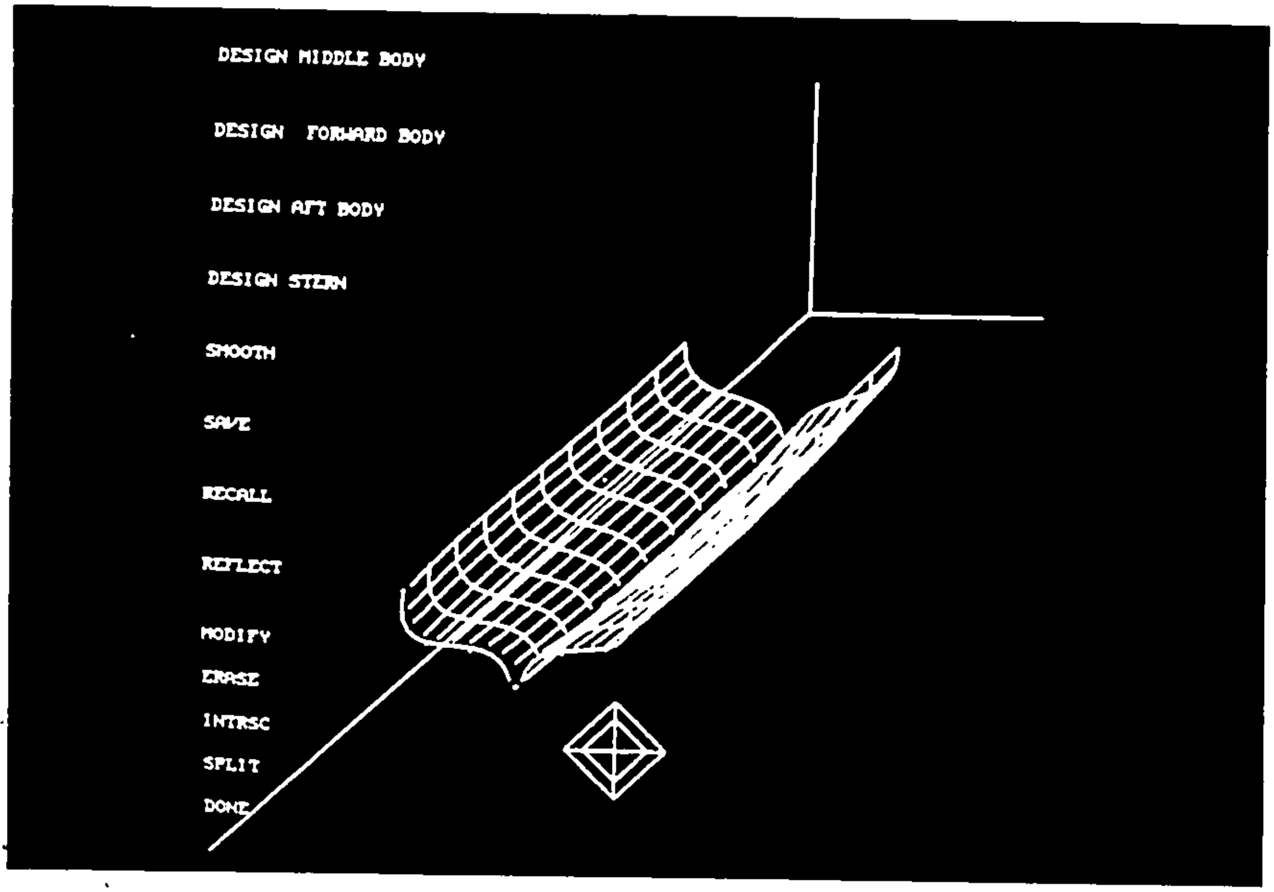


Figure (5.24) Design of the middle body of a ship

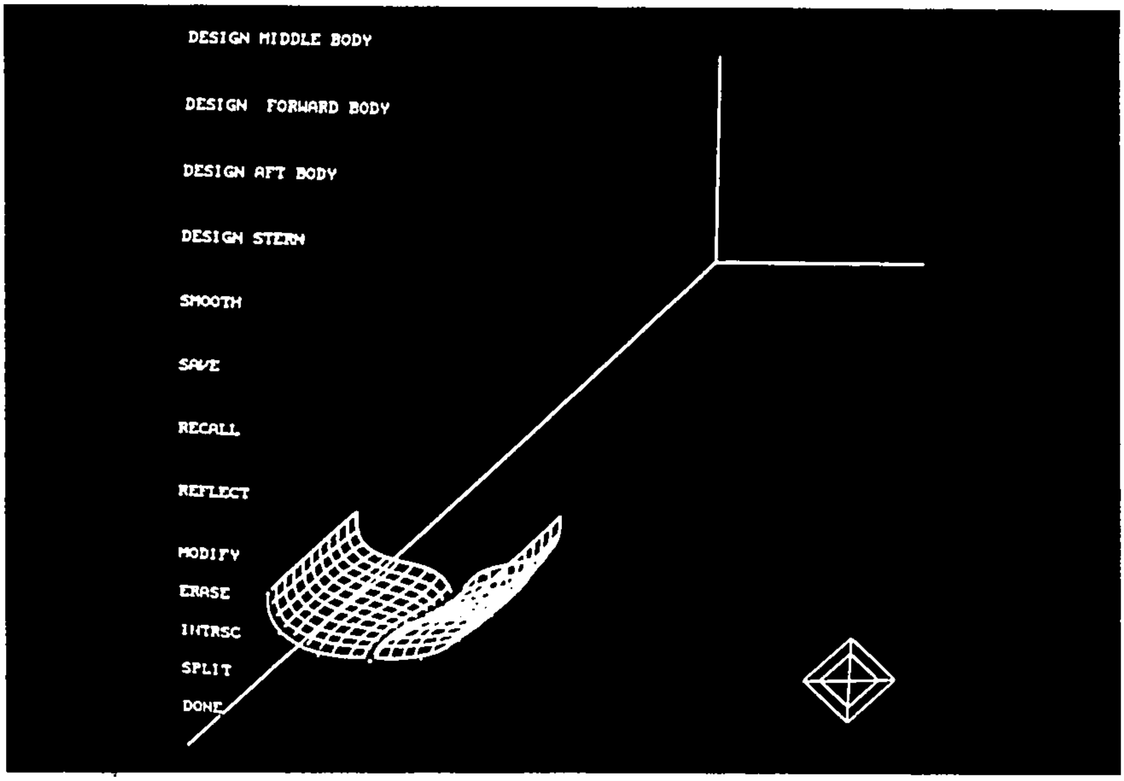


Figure (5.25) Design of the aft body of a ship

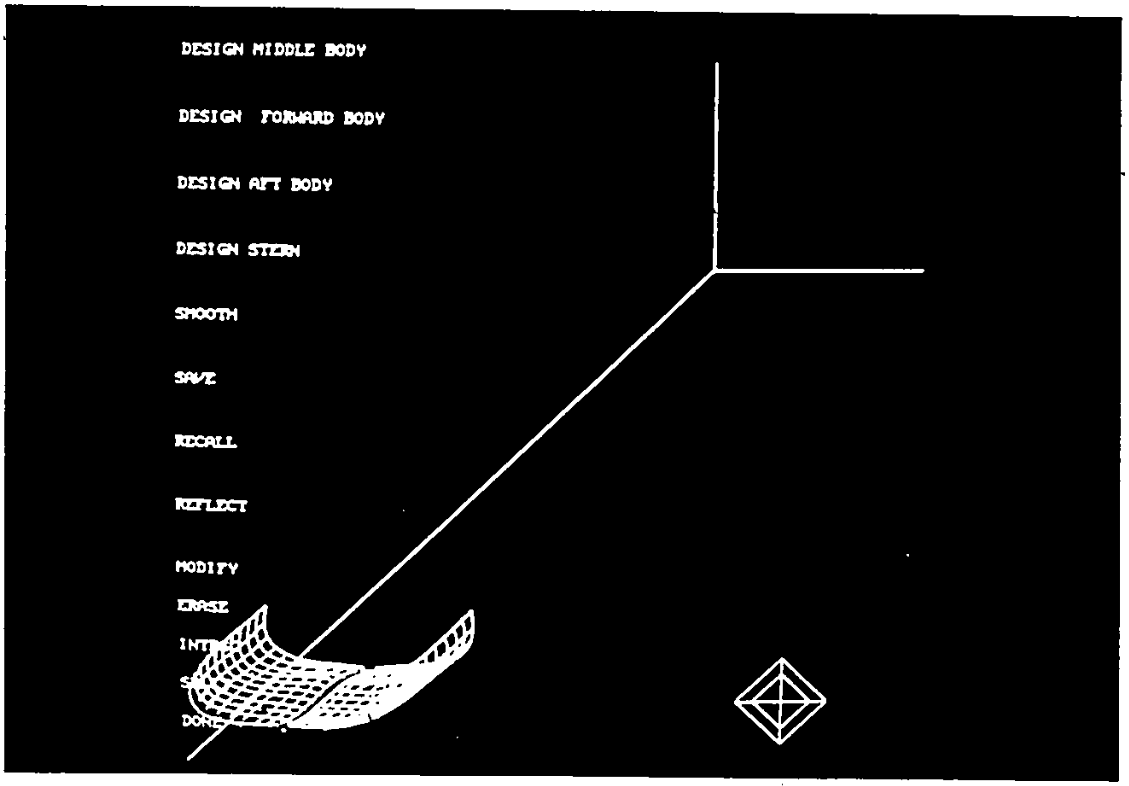


Figure (5.26) Design of the stern of a ship

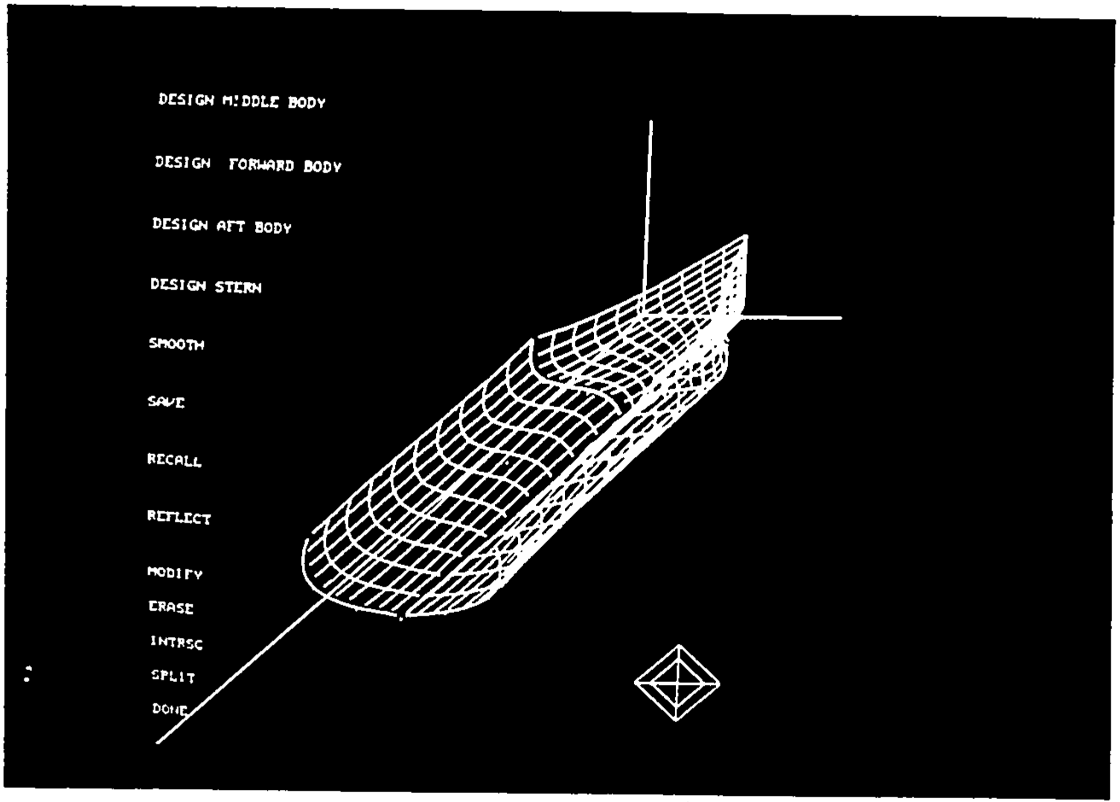


Figure (5.27) The forward and middle bodies combined together

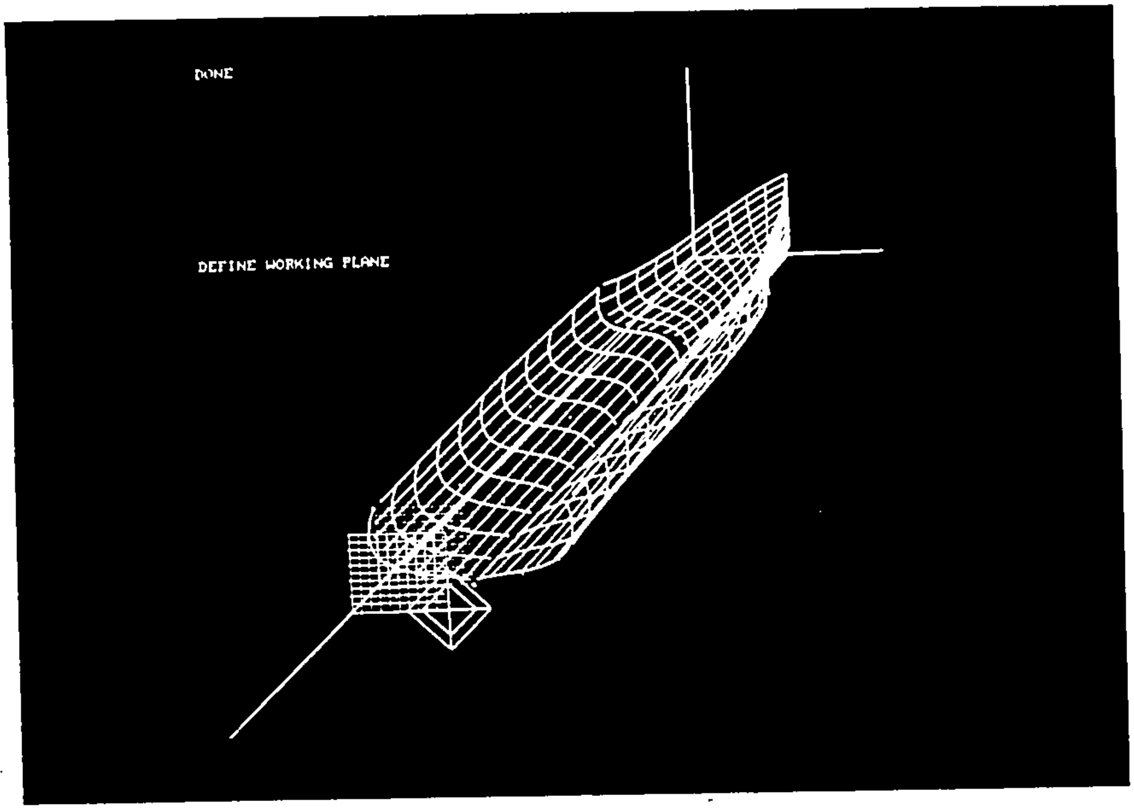


Figure (5.28) Designing the aft body as combined to middle and forward bodies of the ship

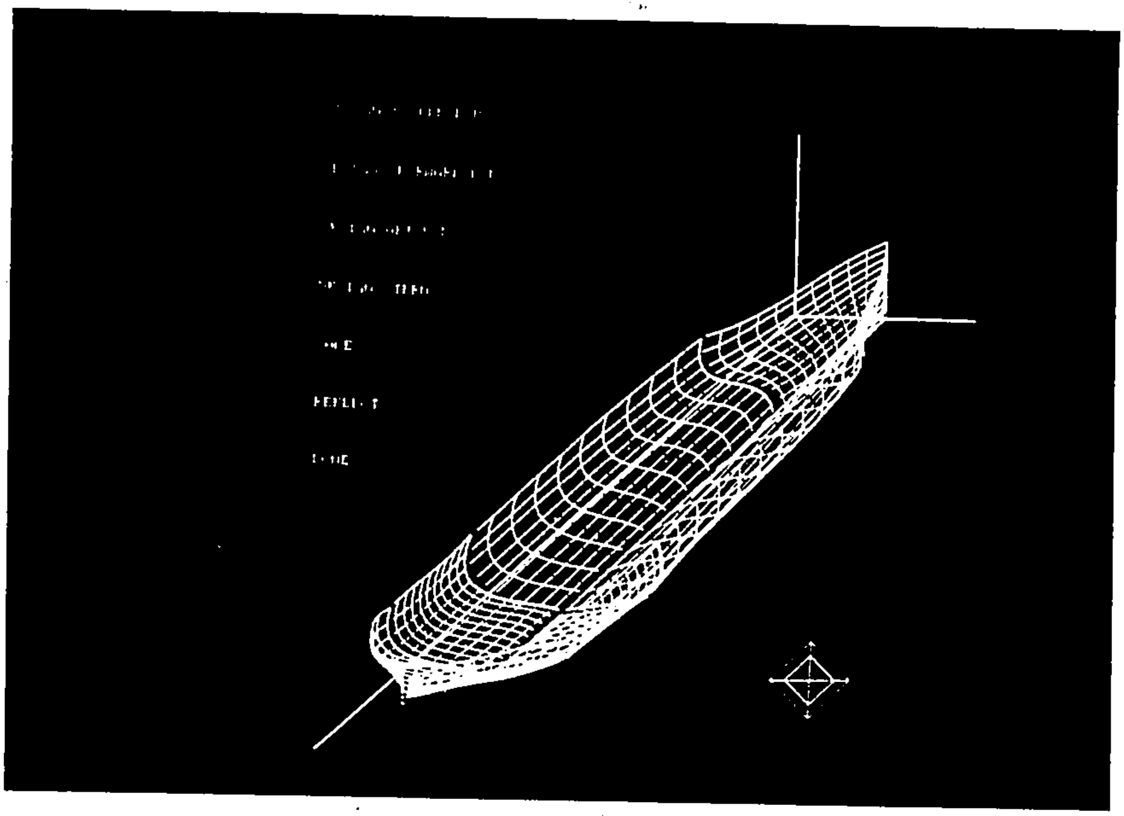


Figure (5.29) A complete ship hull design

CHAPTER 6

MANUFACTURING SCULPTURED SURFACES USING NC SYSTEMS

6.1 What is NC?

Before we discuss the possible ways of manufacturing sculptured surfaces designed using the CAD graphics system described in the previous chapters, we will have a quick look at NC [42]. Numerical control is not a kind of machine tool but a technique for controlling a wide variety of machines. It is a system that can interpret a set of prerecorded instructions in some symbolic format; it can cause the controlled machine to execute the instructions, and then can monitor the results so that the required precision and function are maintained. The numerical control system forms a communication link as shown in Figure (6-1). Symbolic instructions are input to an electronic control unit which decodes them, performs any logical operations required, and outputs precise instructions that control the operation of the machine. The feedback enables the controller to verify that the machine operation conforms to the symbolic input instructions.

In 1957, the first successful NC installations were being used in production; however, many users were experiencing difficulty in generating part programs for input to the

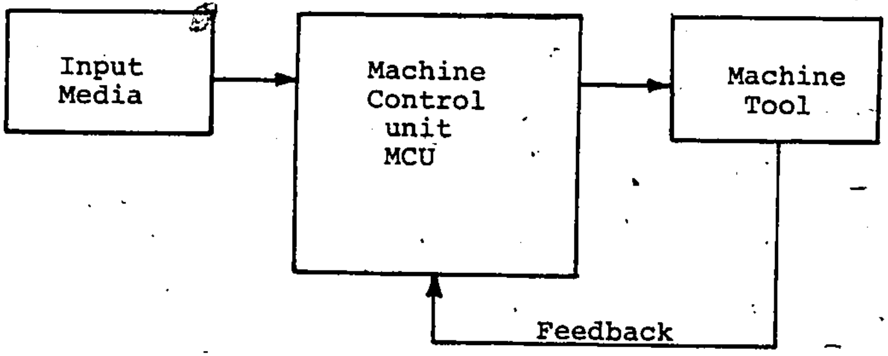


Figure (6-1) A simplified schematic of an NC system

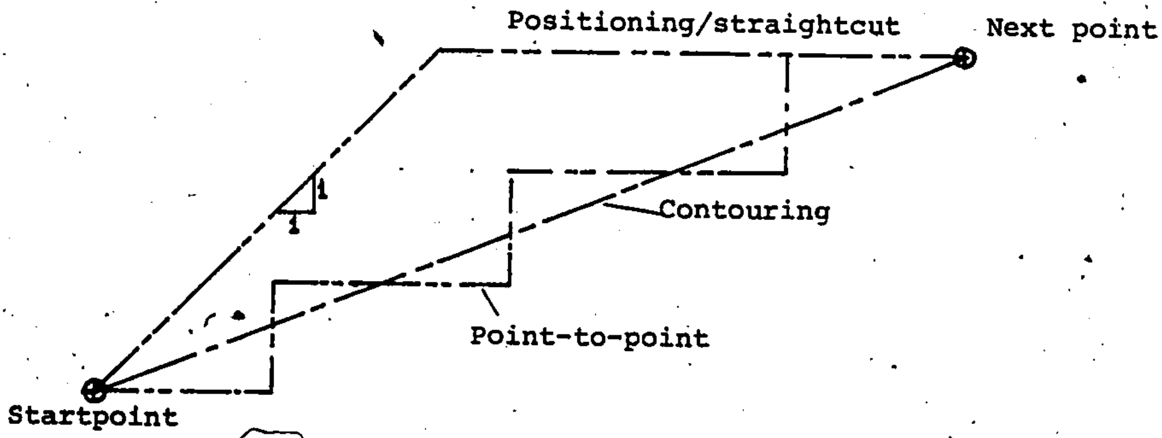


Figure (6-2) Comparison of control system paths machine controller. To remedy this situation, M.I.T. began the development of a computer based part programming language called APT - automatically programmed tools. The objective was to devise a symbolic language which would enable the part programmer to specify mathematical relationships in a straightforward manner. APT provides the programmer with three tools.

1. A geometry description capability that enables him to describe necessary calculations without having to execute them.
2. A method of describing tool motion.
3. A means for specifying inactive tool information such as feeds, speeds, and miscellaneous functions.

Although modern NC systems perform many functions, the most important controlled operation is dynamic positioning of the cutting tool with the use of system coordinates that are general enough to define any geometric motion. Points along the part profile* are defined by x, y, z coordinates and fed in sequence to the NC controller which generates the appropriate positioning commands. Positioning can be accomplished using two distinct methods, absolute positioning or incremental movement. The incremental system uses the change in x, y, z dimensions to specify position, whereas the absolute system uses coordinate values.

---

* In many cases offset points (cutter path) are required.

The path which the cutting tool follows as it traverses from point to point depends on the type of control system used. Figure (6-2) describes the different control system paths. The contouring controller, the one that we are interested in, generates a path between points by interpolating intermediate coordinates. All contouring systems have a linear interpolation capability (i.e., the ability to generate a straight line between two points).

A numerically controlled machine will function only if the proper instructions are developed and passed on the machine control. The process by which the symbolic NC instructions are transferred to the control unit is termed the "communication cycle". The cycle begins with the development of a set of NC instructions, called a "part program", that specifies positioning data and related machining functions in a machine readable format. The next step in the communication cycle is the physical transfer of the part program to the machine controller. The "communication medium" (usually a tape) transports a symbolically coded part program to the control unit. Even a relatively short set of NC instructions may contain hundreds, and possibly thousands, of alphanumeric characters and special symbols. For this reason a communication medium must represent a symbolic code in a compact form which can be easily deciphered by the machine control. Usually this communication medium (NC tape) is created using a special computer program [69] (post processor) called by the processor and used to convert cutter location (CL) into that medium that is understandable by an MCU (machine control unit). A typical post processor contains five elements: input, motion analysis, auxiliary functions, output and control and diagnostics.

The "input" element reads the cutter location data and miscellaneous information that is output from the processor. It verifies the format of the data and transfers appropriate values to other elements of the post processor.

The "motion analysis" element contains the dynamics and geometry sections. The geometry section performs coordinate transformations to convert the general CL data into specific machine tool coordinates. The geometry section insures that the machine's physical limits are not exceeded and that the tool does not cut into any part of the machine. Finally, it is the job of the geometry section to select proper linear and rotary motions and to insure that the resultant path is within tolerance.

The "dynamics" section of the motion analysis element calculates the appropriate tool velocity based on servo type, and the acceleration/deceleration characteristics of the machine tool.

The "auxiliary function" element provides for the output of miscellaneous and preparatory command codes, i.e., translates the machine control commands, e.g., COOLANT/ON, SPINDLE/ON, COOLANT/OFF, etc.

The "output" element of the post processor generates two types of output: (1) the actual numerical control blocks in a media form that can be either directly input to the MCU or easily converted into a form for direct input to the MCU and (2) computer printout of each NC block in a readable format.

The control and diagnostic element of the post processor is necessary to insure that a proper flow of information occurs in the program and that analysis errors are diagnosed and brought to the attention of the NC programmer.

The total picture from engineering drawing to finished product is shown in Figure (6-3) using an APT processor.

6.2 A Proposal for an Integrated CAD/CAM System

Our aim is to build a semiautomatic programming system that can handle both the design and manufacturing of sculptured surfaces.

In Section 6.3 of this thesis, we introduced a new technique for determining the CL. This method could be very easily incorporated in our interactive free form surface design program in the following fashion:

1. After the designer has designed his surface and he is satisfied with his design he can interactively create curves of intersections with the surface using the 'INTERSECT' light button command.

2. Depending upon the complexity of the designed surface,

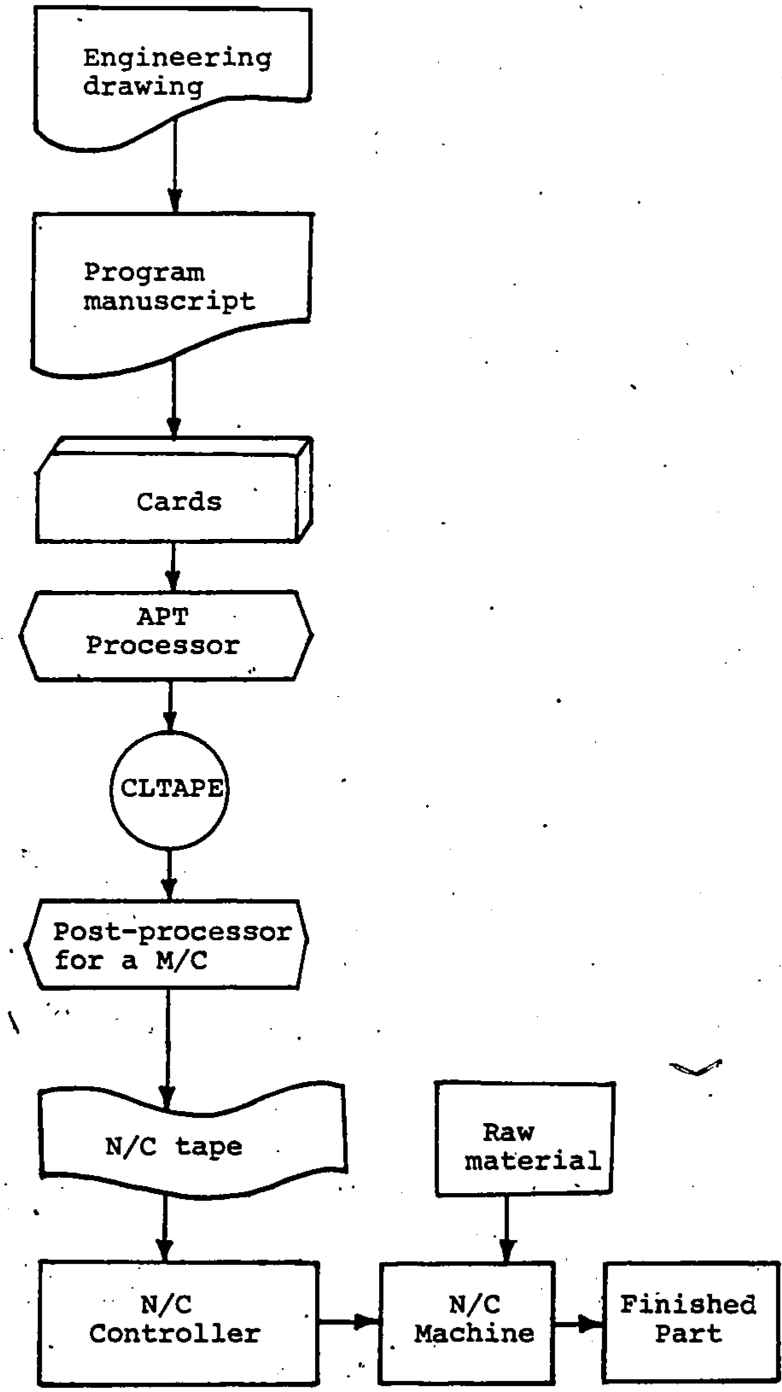


Figure (6.3) The total picture from engineering drawing to finished part the desired accuracy and the designer's experience, the designer can decide upon the number of points to be interpolated on a curve of intersection (one complete cutter pass or what is referred to as master dimension information MDI) and the number of these MDI's (Figure 6-4). Automatic selection of the number of points on an MDI and separation between successive MDI's requires a knowledge of the radius of curvature at various positions in the surface. A method for the evaluation of minimum curvature for a parametric cruve has been developed by Helpert [70] and involves an optimum search technique for determining the minimum value of a function. The consequent knowledge of minimum radius of curvature enables the maximum cutter size (diameter) to be used [71].

3. Using these data points, the CL's could be calculated as described in section 6.3 using an NC processor, e.g., NELAPT, APT [72] (an NC processor developed by the National Engineering Laboratory (NEL)).

As the engineering drawing is the major interface between design and manufacture it was logical to consider curves of intersections (MDI's or contours) as a possible digital interface between CAD and CAM [73]. The whole idea of using MDI's as an interface between CAD and CAM is shown in Figure (6-5).

Calculation of cutting conditions and determining technological data (optimized machining sequence, tool radius, feed, rotational spindle speeds, etc.) as a part of an

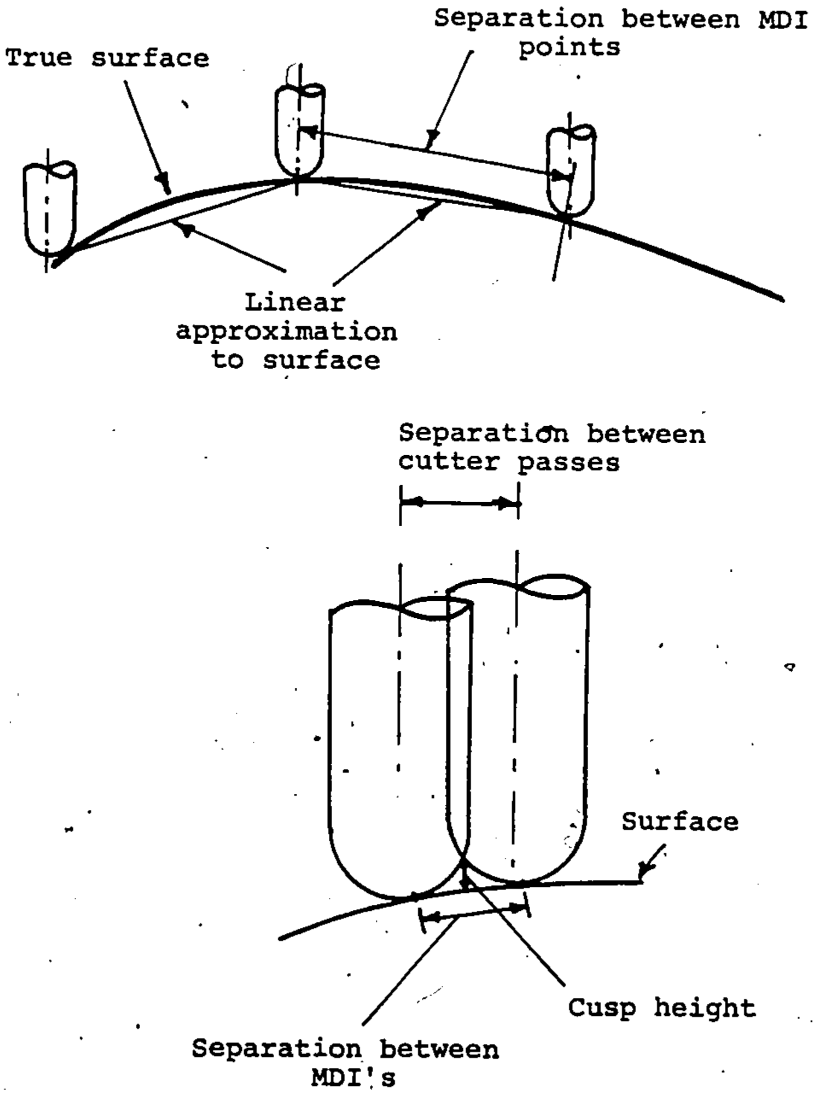


Figure (6.4) Separation between points and between cutter passes

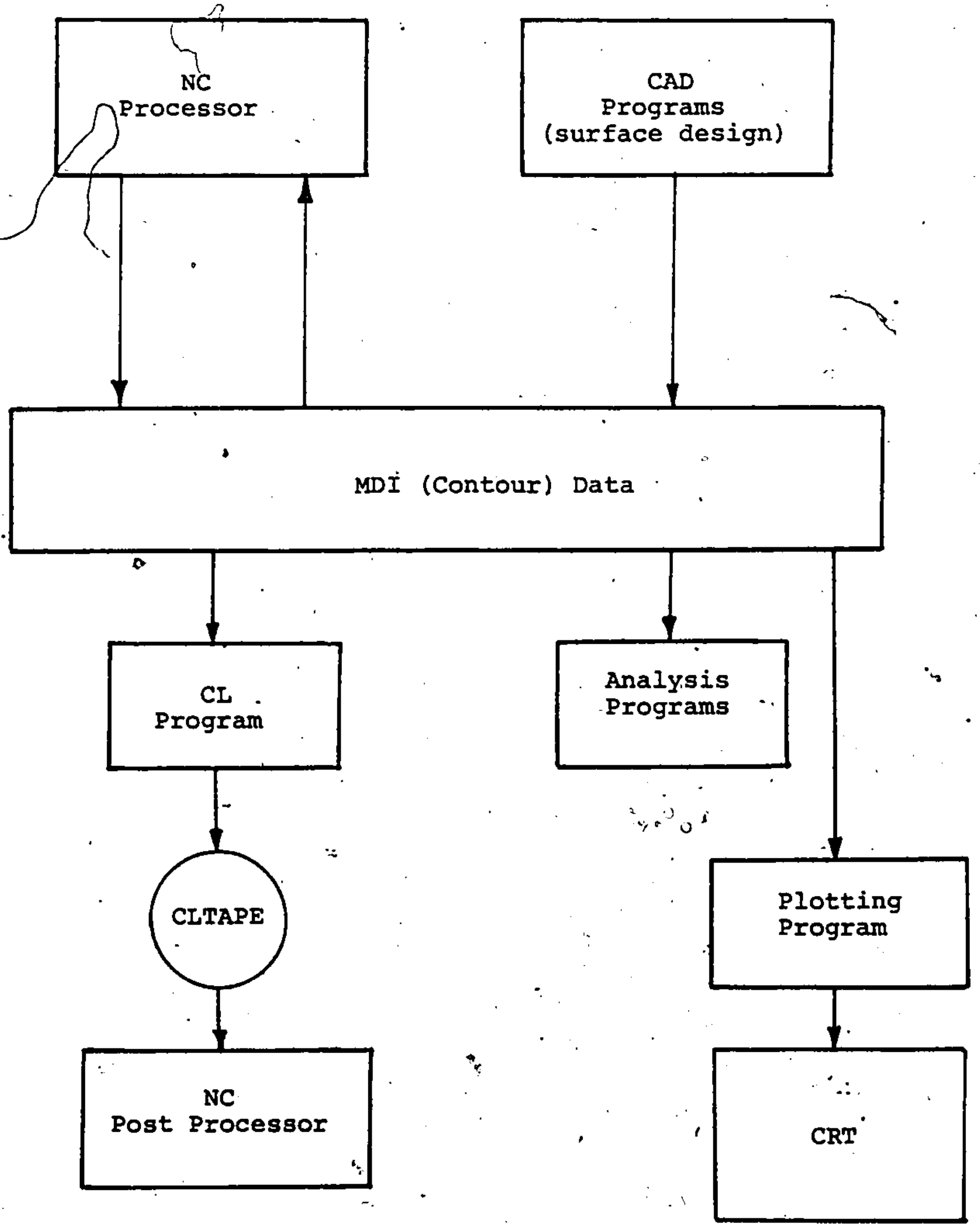


Figure (6.5) The MDI as an interface between CAD and CAM integrated information processing system could be achieved using an especially designed preprocessor [74]. The practical application of this technique using a minicomputer could be performed as follows. The preprocessor control program calls a predesigned milling optimization program which in turn will scan the geometrical data files. During the execution of the optimization program, the cutting conditions are computed via interaction with the machinability data, operation requirements, tooling available, etc., stored on the technological data files. The optimized cutting parameters should not be imposed on the user. The part programmer should make use of his experience and commonsense to select the cutting parameters and tool specifications from a displayed table containing the optimum values. A very good example of the preprocessor technique is given in Reference [74] in Chapter 4.

6.2.1 Advantages of the Proposed System

In contrast with processing in a batch environment (Figure 6-3), the proposed system will provide the user with the capability of generating the control tape interactively. In an interactive graphic environment, the on-line computer graphics will facilitate the part programmer's job and reduce the number of trial runs before producing a good part. Part-geometry input verification, via an interactive graphics terminal, can shorten and refine the procedure used to produce a part. Errors are detected before the actual processing begins. In the event of an error prediction the user can easily and dynamically modify and correct the errors via the refreshed CRT graphics terminal. To this can be added the ability to generate a graphical simulation of the cutting operation (trajectories of center of ball-end cutter) on a graphics terminal CRT.

6.3 A Proposed Method for Obtaining an NC Cutter Path for Milling a Three-Dimensional Curved Surface

We have developed an interactive graphics program which can generate any three-dimensional surface. One product of this system is the availability of any section through the surface. It further provides the coordinates of any point on the section where it transects the surface.

The algorithm used for our graphics system has special features which make it uniquely adaptable to this problem. Two possible modes could be used.

6.3.1 Mode 1 - Tangent Point Follows the Section Line

In this approach the nominal position of the cutter is offset so as to hold the tangent point on the section line. It is proposed that the cutter path be defined so as to cut along tangent lines corresponding to any convenient set of sections - usually a closely spaced set parallel to one coordinate plane.

Figure (6-6) illustrated such a transection, and Figure (6-7) shows a cross-section at point P_t parallel to the y-z plane. In order to establish tangency at P_t the cutter must be offset from the nominal position along the section an amount Δy. A similar offset Δx must occur in the x-z plane.

The two offsets, and the vertical location of the cutter centre point C, can be treated as optimization variables, and a nonlinear programming technique used to minimize the difference between the vector CP_t, R_v, and the cutter radius, R. At the correct location this criterion quality should reduce to nearly zero.

6.3.2 Mode 2 - Cutter Path Follows the Section Line

In this approach the cutter is maintained in the section, and the tangent path is allowed to wander as necessary. Only the height of the cutter need be determined.

This can be done by using our patch splitting facility to generate a new micro patch with 16 points in the quadrant containing the tangent point. This quadrant can be determined from the tangent lines T_1 and T_2, shown in Figure (6-8).

We designate the i^{th} point in the patch as P_i, and the distance from C to P_i as L_i. An optimization strategy is now used with two stages. For a given ΔR we identify the P_i which gives minimum L_i-R. We then use a second

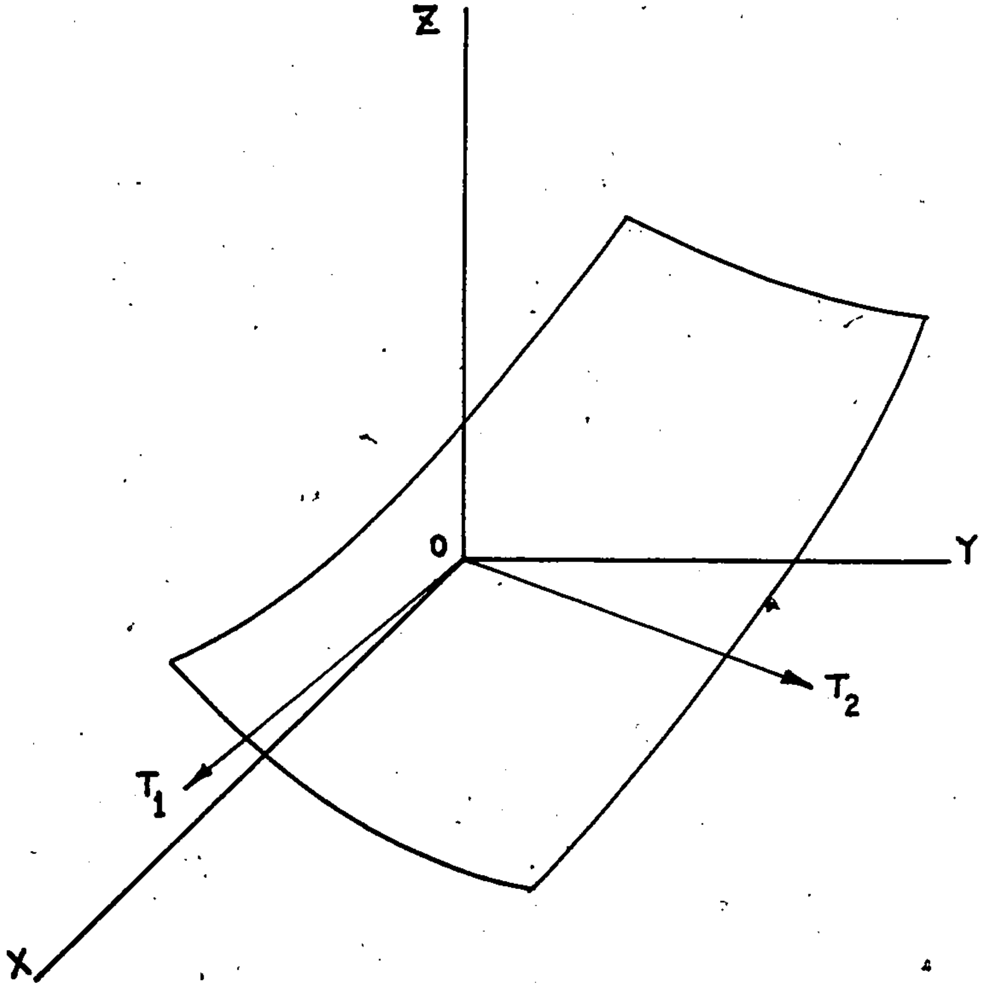


Figure (6.6) Three-dimensional surface

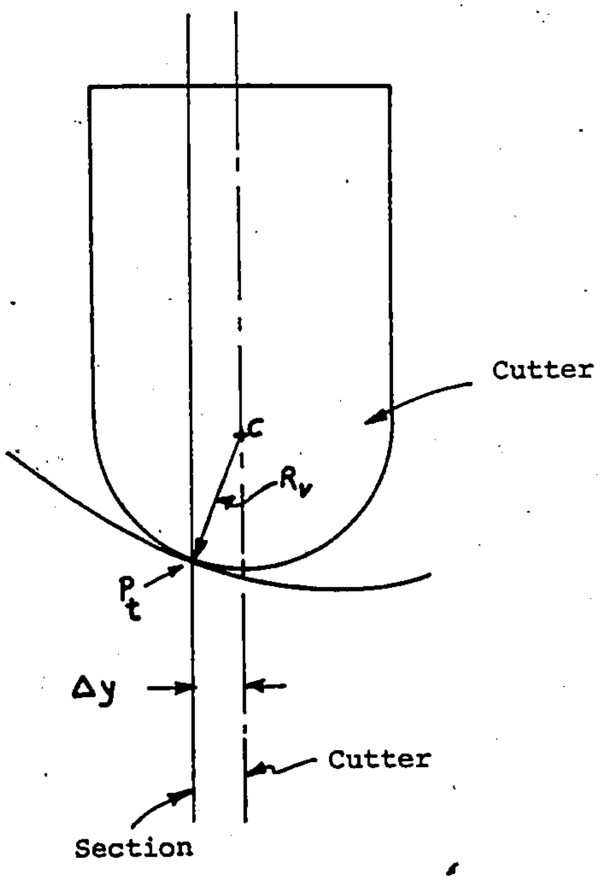


Figure (6.7) Section through surface parallel to y-3 plane one-dimensional search startegy to adjust ΔR so that $(L_i\text{-}R)$ is nearly zero. The mesh must be fine for adequate accuracy, and the tangent point may be off the patch. This will be observed if $(L_i\text{-}R)$ does not reduce to zero, and then a second adjacent patch can be used.

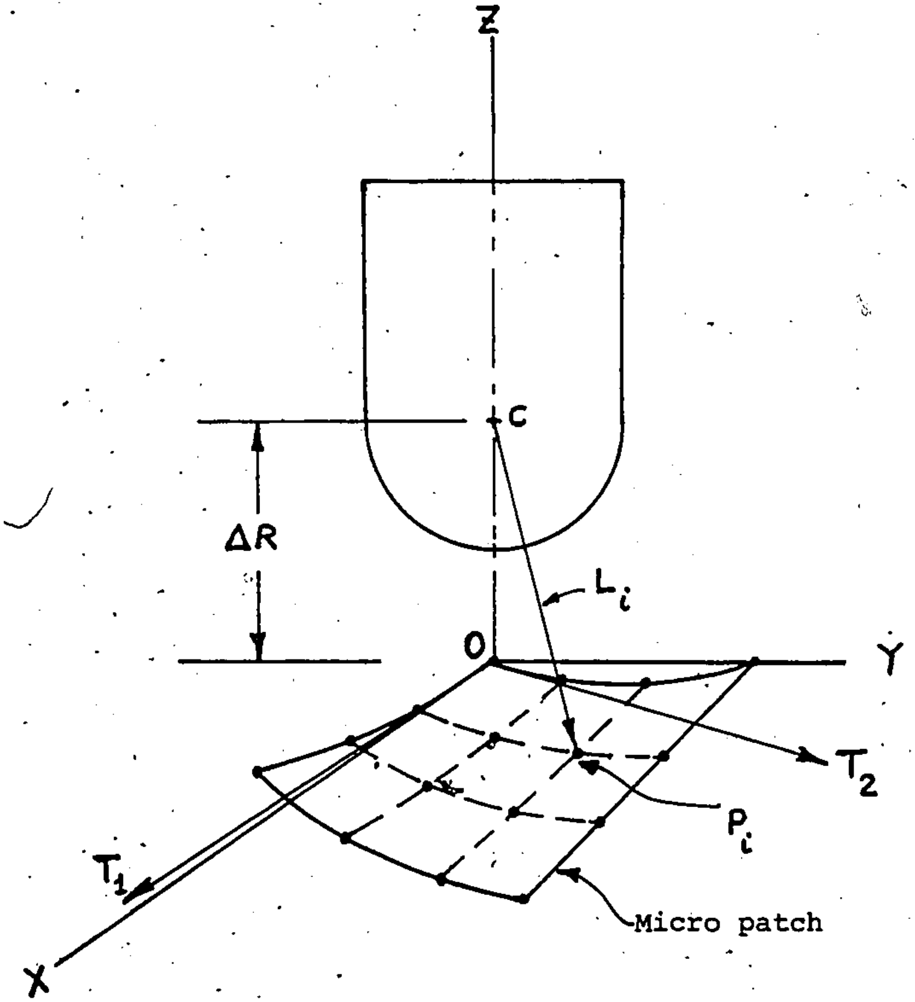


Figure (6.8) Micro patch used to determine ΔR

CHAPTER 7

CONCLUSION

Shape is one of the the most important variables in engineering design, and the computer aided design of 3-D shapes is an active field. In any computer-based shape handling system there must be two aspects: the definition of shape, and the interrogation of shape. These two aspects, although distinct, are completely interdependent and neither should be stressed to the exclusion of the other.

In Chapter 1 of this thesis we have investigated the already existing computer aided design graphics systems for interactively creating three dimensional curved surfaces. In almost all of these systems surface definition algorithms were developed from the Coon's patch * algorithm where, as was described earlier in this thesis, 48 coefficients are required to define a patch [77] . To define a patch, slope vectors and twist vectors are involved in the design process. The user has to define or give values to these vectors in order to generate the patch surface. Designers usually have great difficulty dealing with such slope and

---

* Appendix D

twist entities. On the other hand, surface modification requires redefinition of these entities. The effect of changing one element of the slope or twist vectors is unpredictable and consequently the interaction between the designer and the computer model was troublesome.

The first aim of this thesis was to tackle the previous problem. Surface patch definition was achieved using only a grid of 16 spatial points lying on the patch surface. Surface modification is done dynamically by relocating the position of any of the predefined mesh points (or what we called control points) using a refresh type graphics terminal CRT, [78].

One of the more difficult problems associated with the design of an interactive system is to make it easy to use - the more facilities the system has, the more difficult it is to make them available to the user. The free form surface design program developed in this thesis proved to be very easy and simple to use in creating three-dimensional surfaces. The mathematics involved in creating such surfaces is completely hidden from the user so that he can direct his attention only to the design process. The system provides the designer with a powerful and integrated set of design tools (smoothing, blending,...) which aids in the design and refinement of any curved surface. It is worth reviewing the traditional drawing board method of designing surfaces, in order to highlight the difficulties that have to be overcome by designers. As a special application, let's examine how a ship's hull might be tackled on a drawing board. The first stage would be to lay down the principal profiles and sections in a three-view drawing.

In many cases these would be laid down within specified constraints - length, breadth, depth, etc. - but the designer would be exercising considerable freedom of choice within these. The next stage is to add more sections to the drawing in order to specify the surface in more detail. This is done by using a graphical fairing technique that is essentially iterative. The designer constructs some diagonal planes which intersect the sections that he drew in the first stage and are so arranged that the section line is approximately normal to the diagonal plane .
The fairing procedure consists of fitting a spline curve through the known points of this diagonal line which is then used to interpolate the shape of intermediate sections. The final stage [79] is to construct these intermediate sections and then to construct the shapes of waterlines and buttocks. Very often these constructed sections show hollow and humps that should not be there. To remove them requires another cycle of the fairing process starting at Figure 7-2. There are obvious disadvantages associated with the traditional method of designing surfaces, especially ship hulls, based on the following reasons:

- Designing surfaces by hand requires design draughting skills of a high order. Given these skills it is still a long job to specify the surface in sufficient detail to manufacture the design.

- This type of procedure does not design the surface, it merely designs lines on the surface. Very often further constructions are necessary in order to obtain manufacturing information. This of course is time-consuming and could lead to inaccuracies.
- Complete freedom of choice is only being exercised at the first stage [77] . The design becomes more and more constrained as the design proceeds until the final stage is merely a graphical construction, with no scope for applying design talent.

The system developed in this thesis overcomes all the above objections. a special ship hull design program was designed and proved to be an efficient and easy to use design program. Aesthetic, geometric designs (bottles, furniture, pipes, nozzles, etc.) also proved to be a rich field for the applications of the system.

A new technique for generating the cutter path for NC manufacturing of free form surface was proposed in this thesis. A literature survey indicates that this technique could solve many problems in manufacturing sculptured surfaces. Chapter 6, which contains a description of this technique) could be a basis for further research. Chapter 6 of this thesis also suggests an interface between the CAD system developed in this thesis and any existing CAM system, in order to achieve an integrated CAD/CAM system, [80].

LIST OF REFERENCES

REFERENCES

1. Newman, W., and Sproull, R., Principles of Interactive Computer Graphics, McGraw-Hill Co., New York, 1973.

2. Hatvany, J. Newman, W. M. and Sabin, M. A., World Survey of Computer-Aided Design, CAD, Volume 9, No. 2, April 1977.

3. Sutherland, I. E., "A Man-Machine Graphical Communication System, M.I.T. Lincoln Laboratory, Technical Report No. 296, 1963.

4. Johnson, T. E., "SKETCHPAD III, Three-dimensional Graphical Communications with a Digital Computer", Technical Memorandum ESL-TM-176, M.I.T., 1963.

5. South, N. E., Johnson, W. L. and Sanders, J. W., "Analytic Surfaces for CAD", A paper presented at the Automotive Engineering Congress, Jan. 1966.

6. Coons, S. A., "Surfaces for Computer-Aided Design of Space Forms, M.I.T. Project MAC TR-41, June 1967.

7. Gordon, W. J., "Blending-Function Methods of Bivariate and Multivariate Interpolation and Approximation", General Motors Research Report, GMR-834 (updated 1970, GMR-834-B), GMC, Warren, Michigan (1968).

8. Forrest, A. R., "Curves and Surfaces for Computer-Aided Design", Ph.D. Thesis, Cambridge University, 1968.

9. Cohen, D. and Lee, T. M. P., "Fast Drawing of Curves for Computer Display", Spring-Joint Computer Conference, pp. 297-307, 1969.

10. Bezier, P., "Emploi des Machinen a Commande Numerique Masson et Cie, Paris, 1970.

11. Bezier, P., "Numberical Control in Automobile Design and Manufacture of Curved Surfaces", Curves and Surfaces in Engineering, I.P.C. Science and Technology Press, Guilford, England, 1972.

12. Ball, A. A., "A Simple Specification of the Parametric Cubic Segment", Computer and Graphics Journal, Volume 3, 1978.

13. Peters, G. J., "Parametric Bicubic Surfaces", in Harnhill, R. E. and Riesenfeld, R. F. (eds.), Computer Aided Geometric Design, Academic Press, New York (1974).

14. Closher, R. and Stowell, G., "Fast Generation and Manipulation of Curved Surfaces", ASME, paper 77-DET-106, 1977.

15. Barnhill, R. E., Brown, J. H. and Klucewicz, I. M., "A New Twist in Computer-Aided Geometric Design", Computer Graphics and Image Processing, 1978.

16. Flanagan, D. L. and Hefner, O. V., "Surface Moulding New Tool for the Engineer, Astronautics and Aeronautics, April 1967.

17. Chasen, S. H., "The Role of Man-Computer Graphics in the Design Process", Lockheed Georgia Research Laboratory, Feb. 1969.

18. MacCallum, K. J., "The Use of Interactive Graphics in Preliminary Ship Hull Design", International Symposium, Computer Graphics 70, Brunel Univ., April 1970.

19. Armit, A. P., "A Multipatch Design System for Coon's Patches", IEEE International Conference on Computer-Aided Design, Conference Publ. 51, Southampton, April 1969.

20. Armit, A. P., "A Language Processor for Interactive Work at the PDP-7", Cambridge Computer-Aided Design Group, Document 17, October 1968.

21. Armit, A. P., "Computer Systems for Interactive Design of Three-dimensional Shapes", Ph.D. Thesis, Cambridge University, 1970.

22. Cordes, R. M. and Brewer, J. A., "ICES/GETAM: A Basic Computer Graphics Problem Oriented Language", User's Manual, Louisiana State University, Baton Rouge, Louisiana, 1973.

23. Kestner, W., "A Dialogue System for Creating and Manipulating Graphical Symbols and Structures", Computer-Aided Design, 8, 2, pp. 101-110, 1976.

24. Braid, I., "Designing with Volume", Ph.D. Thesis, Computer Laboratory, University of Cambridge, England, 1970.

25. Braid, I., "Six Systems for Shape Design and Representation, A Review", Computer Laboratory, Cambridge University, CAD Group, Document No. 87 (1975).

26. Armit, A. P. and Lemke, H. U., "ICON: The Interactive Creation of NASTRAN data. A System description", Computer Aided Design, Vol. 7, No. 3 (July 1975), pp. 145-150.

27. McCormick, C. W. (ed.) "The NASTRAN User's Manual", National Aeronautics and Space Administration, 174.

28. Lacoste, J. P. and Rothenbury, R., "Communication in Computer Aided Design and Computer Aided Manufacturing", Proc. 3rd IFIP/IFAC Int. Conf., Programming Languages for Machine Tools (PROLAMAT '76), North Holland (1976).

29. Pickler, G. and Simon, V., "A General Dialogue System for Interactive Graphic Programming of NC Machines and CAD Systems", Proc. 3rd IFIP/IFAC Int. Conf. Programming Languages for Machine Tools (PROLAMAT '76), North Holland, (1976).

30. Seifert, H., "The Development of Graphic Software Systems for the Mechanical Engineering Design", Proc. Symp. Computer Aided Design in Mechanical Engineering, (1976), pp. 267-278.

31. Welbourne, D. B., Johnson, A. L., Morris, R. B., and Olding, P. W., "The Cambridge University Duct System for Pattern, Mould, and Die Making", Proc. Interactive Design Systems Conference, Stratford-on-Avon Computer Aided Design Centre, (1977).

32. Gossling, T. H., "The DUCT System of Design for Practical Objects", Int. Federation for the Theory of Machines and Mechanism Conf., (1976).

33. England, J. N., "A System for Interactive Modeling of Physical Curved Surface Objects", pp. 336-340, Computer Graphics, A Quarterly Report of SIGGRAPH-ACM, Vol. 12. No. 3, August 1978.

34. Voelcker, H., Requiche, A., Hartquist, E, Fisher, W., Metzzer, J., Tilove, R., Birrells, N., Hunt, W., Armstrong, G., Check, T., Moote, R. and McSweeney, J., "The PADL-10/2 System for Defining and Displaying Solid Objects", Production Automation Project, The University of Rochester, N.Y., pp. 257-263, SIGGRAPH-ACM, Vol. 12, No. 3, August 1978.

35. Sutherland, I. E., "A Head-Mounted 3D Display", Proc. Fall Joint Computer Conference, AFIPS Press, Montrale, New Jersey, pp. 757-764 (1968).

36. Burton, R., "Real-Time Measurement of Multiple 3D Positions", University of Utah Comp. Sci. Technical Report UTEC-CSC-72-122, Salt Lake City, Utah, June 1973.

37. Clark, J. H., "Designing Surfaces in 3D, Comm. ACM 19, 8, pp. 454-460, 1976.

38. Lavick, J. J., "Design Philosophies for a Man-Machine Engineering Environment, McDonnell Douglas Automation Report No. 6045, 1967.

39. Lavick, J. J., "Computer-Aided Design at McDonnel Douglas", Advanced Computer Graphics, Plenum Press, London and New York, 1971.

40. Lavick, J. J., and Martin, G. L., "Modern Techniques in Design", 1972, CAD/CAM Conference, Society of Manufacturing Engineers, Atlanta, Georgia, 1972.

41. Pennington, R. H., "Introductory Computer Methods and Numerical Analysis", MacMillan Company, Toronto, 1970.

42. Pressman, R. S. and Williams, J. E., "Numerical Control and CAM", John Wiley & Sons, N.Y., 1977.

43. Akgerman, N. and Altan, T., "NC's Growing Impact on Design", Battelle Columbus Laboratories Int. Report, Columbus, Ohio, 1976.

44. Roberts, L. G., "Machine Perception of 3D Solids", MIT Lincoln Laboratory, TR 315, May 1963.

45. Warnock, J. E., "A Hidden-Surface Algorithm for Computer Generated Half-tone Pictures", TR 4-15, Computer Science Department, University of Utah, 1969.

46. Watkins, G. S., "A Real-Time Visible Surface Algorithm", Computer Science Department, University of Utah, UTECH-CSC-70-101, June 1970.

47. Vannier, M. and Oliff, M., "Hidden Line Removal/Plotting Subprogram", Department of Diagnostic Radiology, University of Kentucky Medical Center, Lexington, May 1977.

48. Kuo, C., "Computer Methods for Ship Surface Design", American Elsevier Pub. Comp., Inc., N.Y., 1971.

49. Taylor, D., "Calculations for Ships' Forms and the Light Thrown by Model Experiment Upon Resistance, Propulsion and Rolling of Ships", Trans. International Engineering Congress, San Francisco, 1915.

50. Weinblum, G., "Contributions to the Theory of Ship Surface", Werft-Reederei-Hafen, 1929, pp. 462-466, and 1930, pp. 12-14.

51. Weinblum, "Exact Waterlines and Sectional Area Curves", Schiffbau, 1934, pp. 120-142.

52. Weinblum, G., "Systematic Development of Ship Forms", Trans. STG, 1953.

53. Lackenby, H., "On the Systematic Geometrical Variation of Ship Forms", Trans. INA, 1950.

54. Schneekluth, H., "Some Procedures and Approximation Formulas for Use in Lines Design", Schiffstechnick, 1959.

55. Puchstein, K., "Possibilities in Changing Ship Form Parameters by Geometric Distortion of the Hull", Schiffbauforschung, 1965.

56. Thieme, H., "About the Fundamentals for the Mathematical Lines Plan of Cargo Ship", Schiffstechnick, 1955/56.

57. Miller, N. and Kuo, C., "The Mathematical Fairing of Ship Lines", European Shipbuilding, 1963.

58. Williams, A., "Mathematical Representation of Ordinary Ship Forms", Schiff und Hafen, 1964.

59. Kwik, K., "Tabulated Functions for the Representation of Ship Lines and Stream Line Profiles, Report No. 124, Univ. of Hamburg, 1969.

60. Kwik, K., "On the Mathematical Representation of Ship Lines", Schiffstechnik, 1969.

61. Kuiper, G., "Preliminary Design of Ship Lines by Mathematical Methods", Journal of Ship Research, Vol. 14, No. 1, March 1970, pp. 52-66.

62. Reed, A., and Nowacki, H., "Interactive Creation of Fair Ship Lines", Journal of Ship Research, Vol. 18, No. 2, June 1974, pp. 96-112.

63. MacCallum, K, J., "Surfaces for Interactive Graphical Design", The Computer Journal, Vol. 13, No. 4, Nov. 1970.

64. Yuille, I. M., "Ship Design", Curved Surfaces in Engineering, A Conference Organized by the CAD Center, 15-17, March 1972, Churchill College, Cambridge, England, published by IPC Science and Technology Press Ltd.

65. Walker, L. F., "Curved Surfaces in Shipbuilding Design and Production", CAD Journal, pp. 57-62, 1976.

66. Groot, D. J., "Designing Curved Surfaces with Analytical Functions", CAD Journal, pp. 3-8, 1978.

67. C.A.D. in Industry, "Design to Building by C.A.D.", CAD Journal, p. 100, 1978.

68. Riesenfeld, R. F., "Computer-Aided Design in Japan's Shipbuilding Industry", Scientific Bulletin, Department of the Navy Office of Naval Research, Tokyo, Vol. 3, No. 3, pp. 43-48, July to Sept. 1978.

69. Harvie, R. E., "Understanding APT-Type Post Processors", The Expanding World of NC, Proceedings of the Ninth Annual Meeting and Technical Conference, Numerical Control Society, 1972, Chicago, Illinois, Edited by M. De Vries.

70. Helpert, E. P., "A Method for Determining the Maximum Normal Curvature of a Given Surface", RCA Report, PEM-2823, 9/12/66.

71. Yellowley, I., "The Selection of Cutter Diameter in Peripheral Milling", Metalworking Research Group, Report No. 82, McMaster University, 1977.

72. National Engineering Laboratory, 2C,L, Part Programming Reference Manual. NEL Report No. 543, National Engineering Laboratory, East Kilbride, Glasgow (1973).

73. Wilkinson, D. G., "The Use of Contours as an Interface Between CAD and CAM", CAD 74 Conference, Imperial College, London, 24-27, Sept. 1974.

74. De Smit, B., "A Minicomputer Preprocessor and System Macros for the APT Language", M.Eng. Thesis, McMaster University, Dec. 1977.

75. Knuth, D.E., "The Art of Computer Programming", Addison-Wesley, Reading, Massachusetts, 1977.

76. Siddall, J., "Software Engineering Concepts", Internal Report, McMaster University, 1978.

77. Badawy, A. and Siddall, J., "A Software System for Interactively Creating Three Dimensional Curved Surface Designes", Ninth International Conference, Mini and Micro Computers, Montreal, Canada, September 1979.

78. Badawy, A. and Siddall, J., "Use of Computer Graphics in Curve and Surface Design", Proceedings of the Seventh Canadian Congress of Applied Mechanics, Sherbrooke, May 1979.

79. Badawy, A. and Siddall, J., "Use of Computer Graphics in Ship Hull Design", Paper to be published in the Eleventh Annual Pittsburgh Conference on Modeling and Simulation, May 1980.

80. Akgerman, N. and Badawy, A., "Numerically Controlled Machining", Internal Report, Battelle Columbus Laboratories, Metal Processing, November 1979.

APPENDICES

120 c

APPENDIX A

USER'S MANUAL

Program JOLIA

Function

This program aids in the design of any curved surface using input data only from the light pen.

User's Manual

We will consider here a practical design of a smoking pipe using two patches. The main steps required to design the pipe are as follows:

1. User starts up the PDP-11/34 minicomputer.*
2. User runs the program, by typing.

   ```
   .RUN  RK2: JOLIA   < CR >
   ```

3. Program responds by typing a menu of design commands on the CRT graphic terminal, Figure (5.25).
4. User hits the 'INTACT DESIGN' command with the light pen.
5. Program responds by typing the following message; 'PLEASE ENTER THE NUMBER OF PATCHES'.
6. User enters

   ```
   2 <CR>
   ```

7. Program responds by typing the following message: 'PLEASE ENTER NUMBER OF PATCHES PER WORKING PLANE.

---

* H.A. ElMaraghy, "Operating Procedure of PDP 11/34 Minicomputer And GT-46 Graphic Terminal, McMaster Univ., 1977.

8. User enters

2 <CR> (i.e. the points defining the first and second patch lie on the same working planes).

9. Program responds by showing the X-Y-Z axis and a menu having two commands, 'DEFINE WORKING PLANE' and 'DONE'.

10. User hits 'DEFINE WORKING PLANE'option with the light pen.

11. Program responds by typing:

'DEFINE PATCH IN PLANE NUMBER 1'

'POSITION TRACKING OBJECT ON ANY POINT OF THE THREE AXIS TO DEFINE WORKING PLANE NUMBER 1'.

12. User positions the tracking cross on the z axis and hits 'RETURN' key on the TT.

13. Program respond by showing a working plane parallel to the x-y plane and passing with the point the user has positioned the tracking cross at, on the z axis, as shown in Figure (3.7).

14. User hits 'DONE' option with the LP.

15. Program erases the 'DEFINE WORKING PLANE' and 'DONE' options and shows another two options 'POSITION' and 'DONE'.

16. User tracks the tracking cross with the LP and position it (the tracking cross) at any point on the working plane, then hits the 'POSITION' option with LP.

17. Program responds by plotting a dot at the predefined position.

18. User repeats the 16 and 17 steps, seven more times (4 points per patch per plane); user should make sure that point number eight should lie exactly over point number one since the surface is to be closed as shown in Figure (5.25).

19. User hits the 'DONE' option.

20. Program respond by typing:
'DEFINE PATCH IN PLANE NUMBER 2' and shows the 'DEFINE WORKING PLANE' replacing the 'POSITION' option.

21. User continues defining the points defining the two patches at different working planes until he finishes defining 32 points (16 per patch), as illustrated in steps 1 to 20, as shown in Figure (5.25).

22. User hits the 'DONE' option two times.

23. Program responds by plotting the surface passing with the predefined 32 points as shown in Figure (5.25).

24. If the user is satisfied with his design he can hit the 'SAVE' option, the program will ask for a file name, and the user can enter any name, e.g. PIPE.

25. User can now hit the 'EXIT' option since he is finished with the design.

26. If the user wants to have the data describing his surface, he should enter the following command on the TT:

```
.TYPE  <CR>
```

computer will respond by typing

```
FILE NAME?
```

user enters PIPE <CR>

program types the x, y, z coordinates of the points defining the surface.

HIERARCHICAL CHART AND LISTING FOR PROGRAM JOLIA

A hierarchical input-processing-output HIPO chart for the program JOLIA is shown in Figure (A.1). The function of each module is briefly described as well as the input and output of each one. Figure (A.2) shows the three overlay regions of the program. A listing for the program is also included. The program modules were "linked" to the DEC-graphics library "GLIB", the linking procedure is included in the file "JOLIA.COM". A listing for that command file is shown in Figure (A.3). In case of any extension or motification of any of the program modules, the programmer should edit the modified module, compile it and then type:

```
@JOLIA.COM <CR>
```

to perform the linking operation.

Main Program (JOLIA)	Subprogram Level 1	Subprogram Level 2	Subprogram Level 3
- Accepts input data from light pen. - Calls MENUH to choose a design option.	INTACT Creates x, y, z, co-ordinates of a patch using input from LP.		
- First option is interactive design of curved surfaces using input from LP. A - calls INTACT to perform the first option B - calls DRW-JOL to draw the designed patch C - calls TEST 1 to create the boundary vectors of the designed patch.	Input - number of patches, number of patches per working plane, flag to define either curved surfaces or surfaces of revolutions, X-Y co-ordinate from LP. Output - x, y, z,co-ordinates of the control points defining the surface patch.		
- Second option is to smooth two patches at a common border. A - calls SMOOTH to perform the second option.	Calls MENUH to define either points or working planes.		MENUH used to return the tag of a hidden (by LP) subpicture
B - Calls DRW.JOL to draw the smoothed patch.	DRWJOL Used to draw the patch.		
- Third option is to create surfaces of revolutions. A - calls INTACT to define axis of rotation and the control points of the surface of revolution. B - Calls ROTATE to generate the rest of the points defining the surface from the control points defined in the previous step A. C - Calls DRW,JOL to draw the surface of revolution.	Input - array containing the x boundary vector, array containing the Y boundary vector, array containing the Z boundary vector, the tag of the subpicture containing the patch, flag to decide either to draw lines or lines and control points of a patch, number of patches per working plane.		
- Fourth option is to split a pre-defined patch.	Calls POINTS to create x, y, z coordinates of the control points of the patch.	POINTS used to calculate x, y, z coordinates of N points lying on a patch surface.	

B - Calls Test 1 to create the boundary vectors of the splitted patch.

C - Calls DRW.JOL to draw the splitted patch.

- Fifth option is to create curves of intersections of a patch with planes.

A - Calls INT-JOL to generate x, y, z coordinates of the curve of intersection.

- Sixth option is to modify a patch.

A - Calls LPEN to define the patch number (tag).

B - Calls TRAKXY to define x, y coordinates of the new point

C - Calls LPEN to define the number of the point to be relocated.

D - Calls TEST 1 to create the boundary vectors of the modified patch.

E - Calls DRWJO1 to draw the new (modified) patch.

- Seventh option is to blend two adjacent but non touching patches.

A - Calls GETB to get the blending functions of the first patch.

B - Calls GETB to get the blending functions of the second patch.

C - Calls BLEND to create the blending functions of the third (blended) patch.

D - Calls GETBX to create the boundary vectors of the third patch from the blending functions obtained from

Output - image of the patch on CRT.

TEST 1

Used to calculate the x, y and z boundary vectors of a patch defined by 16 x, y, z coordinates.

Input - x, y, z coordinates defining the patch.

Calls LENGTH to get the length between input points.
Calls CMATRX to define the elements of the matrix containing the U, W parametric products.
Calls SOLVE to solve the linear system of equation for boundary vectors.

Output - x, y and z boundary vectors defining the patch.

SMOOTH
used to smooth two adjacent patches along the common border.

Input - three arrays containing x, y and z boundary vectors of the patch before smoothing, number of defined patches to be smoothed, flag to determine whether to smooth along longitudinal or transverse borders.

Calls IDNTFY to define the tags of the two

Input - U, W arrays defining the parametric values of the points, boundary vector (x or y or z) defining the patch, number of points (N).
Output - array containing (x or y or z) coordinates of the points.

LENGTH
used to calculate the length between each successive points (16 points).

Input - x, y, z coordinates of 16 points.
Output - array (of length 16) containing the length between each 2 points.

CMATRX
used to calculate the C matrix (contains U, W parametric products) using the parametric bi-cubic patch expanded equation.

Input - two arrays containing U and W parametric values, number of data points.
Output - C matrix.

SOLVE
used to solve a

the blending functions obtained from the previous step C.
E - Calls DRWJOL to draw the third (blended patch).

- Eighth option is to save the display and create a corresponding data file.
  Calls OBSAVE to save the x, y, z coordinates of the displayed patches.
- Nineth option is to recall the saved display
  Calls GETSHP to recall x, y, z coordinates of the saved display.
- Tenth option is to erase a predefined patch.
  A - Calls LPEN to define the number (tag) of the patch to be erased.
  B - Calls ERAS to erase the patch.
- Eleventh option is to exit.
  Calls FREE to stop and exit from the program.

patches to be smoothed.

Calls GETB to calculate the blending functions of the two patches to be smoothed.

Calls GETBX to calculate the boundary vectors of the smoothed patch.

Output - x, y and z boundary vectors of the smoothed patch.

ROTATE

used to perform rotation of points about any arbitrary axis.
Input - number of x, y, z triplets, x, y, z coordinates of the points to be rotated, direction cosiness of axis of rotation, rotation angle in degrees.

Calls MU to multiply x, y, z coordinates matrix with transformation matrix.

Output - x, y, z coordinates of the rotated points.

GENJOL

used to create the x, y, z coordinates of a patch generated by the subdivision of a given patch.

Input - x, y, z boundary vectors of the patch to be splitted.

system of linear equations using Gauss elimination.

Input - The M by N matrix of the right hand side, the M by M coefficient matrix, the number of equations in the system, the number of right hand side vectors, relative tolerance.

Output - solution of the system, flag if zero there is no error, if -1 there is no solution.

IDNTFY

used to identify the patch's number.

Input - number of already defined patches.

Calls MENUH to identify the patch's number.
Output - the hidden (with LP) patch's number.

GETB

used to calculate the boundary matrix of a patch.

Input - x, y and z boundary vectors of the patch.

Calls MU to perform matrix multiplications.

MU

used to multiply t
general matrices.

Calls POINTS to calculate the x, y, z coordinates of the created patch.

Output - x, y, z coordinates of the patch created by subdividing (splitting) the given patch.

INTJOL

used to draw the resultant continuous curve from the intersection of a plane and a patch.

Input - x, y, z boundary vectors defining the patch.

Calls POINTS to calculate the x, y, z coordinates of the curve of intersection.

Output - the x, y, z coordinates of the curve of intersection, and an image of that curve on the patch at CRT.

BLEND

used to blend two non adjacent patches by creating an in-between patch, smoothed to the predefined two patches.

Input - boundary matrices of the two patches to be blended.

Output - boundary matrix of the in-between (generated) patch.

Output - x, y and z boundary matrices of the patch.

GETBX

used to calculate the boundary vectors of a patch.

Input - x, y and z boundary matrices of the patch.

Calls MU to perform matrix multiplication.

Output - x, y and z boundary vectors of the patch.

Input - name the first matrix, name of the second matrix, name of the resultant matrix, number of rows in the first and columns of the second matrix, number of rows in the second matrix, number of rows in the resultant matrix.

Output - elements of the resultant matrix.

OBSAVE

used to save the display in a data and a display file.

Input - x, y, z coordinates of the patch (or patches) to be saved.

Calls INFILE to define the file name.

Output - data file containing x, y, z coordinates of the patch to be saved.

GETSHP

used to recall a previously saved display.

Calls INFILE to define the name of the file to be recalled.

Output - image of the saved display on CRT.

INFILE

used to accept a file name.

Input - number of patches to be saved or recalled, name of the file.

Figure (A.1) a Hierarchical Chart For Program 'JOLIA'

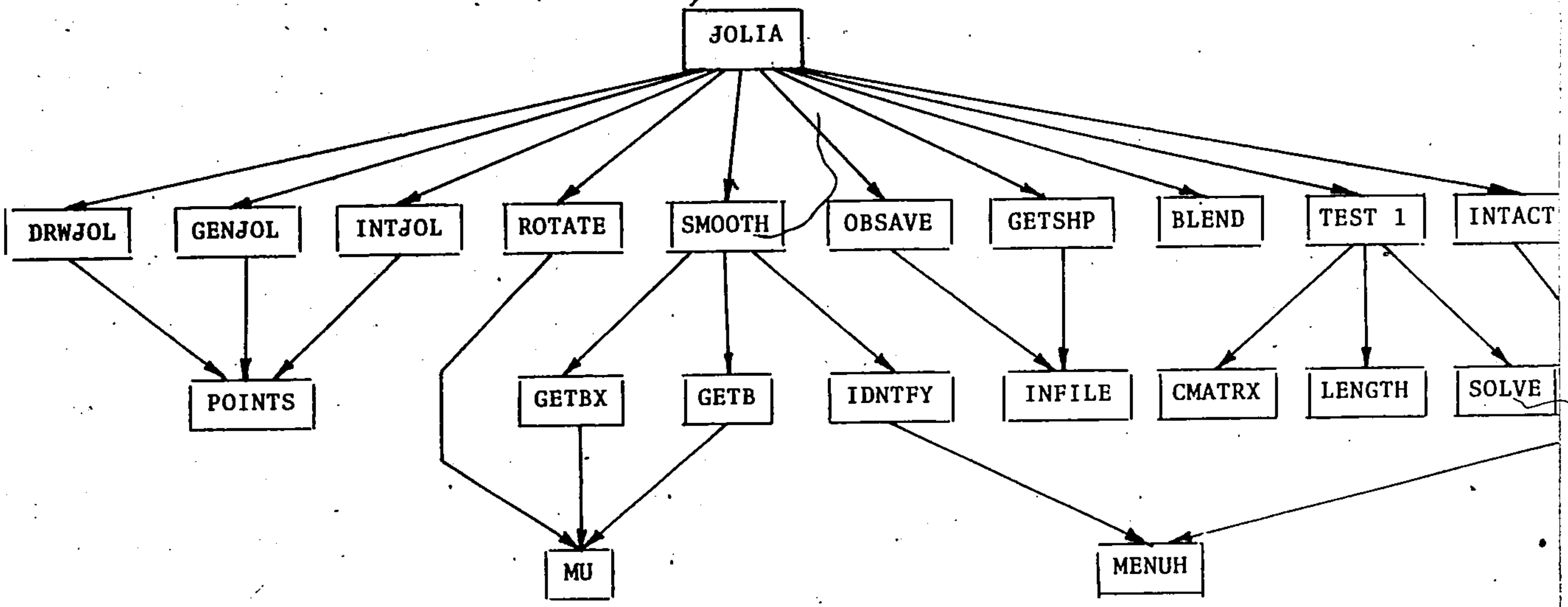


Figure (A.2) PROGRAM "JOLIA"

```
DIR ~U
TYPE
Files? JOLIA.COM
R LINK
RK2:JOLIA,JOLIA=RK2:JOLIA,RK0:GLIB,FORLIB//
RK2:GENJOL/O:1
RK2:INTACT/O:1
RK2:ROTATE/O:1
RK2:SMOOTH/O:1
RK2:BLEND/O:1
RK2:OBSAVE,GETSHP/O:1
RK2:INTJOL/O:1
RK2:TEST1/O:1
RK2:DRWJOL/O:1
RK2:GETB/O:2
RK2:GETBX/O:2
RK2:IDNTFY,INFILE/O:2
RK2:CMATRX/O:2
RK2:LENGTH/O:2
RK2:SOLVE/O:2
RK2:POINTS/O:2
RK2:MENUH,MU/O:3
//

GT OFF
RU JOLIA
```

Figure (A.3) Listing of JOLIA.COM

Program KEY

Function

This program aids in the preliminary design of any curved surface using input data only from the keyboard.

User's Manual

The program main flow from the user's point of view is as follows:

1 - User starts up the mini-computer.

2 - User runs the program, he types:

`RUN RK2:  KEY`

3 - Program responds by typing a menu of design commands on the CRT graphic terminal.

4 - User hits the 'KEY BOARD' light button with LP.

5 - Program responds by typing the following message:

'PLEASE ENTER THE NUMBER OF PATCHES'

6 - User enters the number of patches required to define the surface, e.g.

`1   <CR>`

7 - Program responds by typing:

'ENTER 16 x,y,z COORDINATES OF PATCH NUMBER 1'

8 - Enter the coordinates e.g.

```
10.,  100.,  50.   <CR>
20.,  30.,   40.   <CR>
501., 20.,   60.   <CR>
200., 300.,  121.
       .
       .
       .
```

9 - Program responds by displaying the patch of CRT, passing by the predifined points, together with the menu of design commands.

10 - User is ready now to make use of the design light button commands till he is satisfied with his design.

HIERARCHICAL CHART AND LISTING FOR PROGRAM KEY

A [HIPO] chart of program KEY is shown in Figure (A.4). Figure (A.5) shows the three overlay regions of the program. The modular concept has been preserved during the design stage of the programs presented in this thesis and that's why we made use of some of the modules already designed for program JOLIA, in program KEY.

Main Program (KEY)	Subprogram Level 1	Subprogram Level 2	Subprogram Level 3
- Accepts input data from key board. - Calls MENUH to choose a design option. - First option is to input data points defining the patch via key board.			
A - Calls KEYBRD to return to the main program the x, y, z coordinates of a patch or a group of patches, defined via key board.	KEYBRD used to return to the main program the x, y and z coordinates of a patch defined via keyboard.		
B - Calls TEST1 to create the x, y and z boundary vectors of the patch. C - Calls DRWJOL to draw the patch.	Output - number of patches, x, y and z coordinates of the patch.	LENGTH CMATRX SOLVE	
- Second option is to smooth two patches at a common border.	TEST1		
A - Calls SMOOTH to perform the second option.	DRWJOL	POINTS	
B - Calls DRWJOL to draw the smoothed patch.	SMOOTH	IDNTFY	
- Third option is to split a predefined patch.		GETB GETBX	MU
A - Calls GENJOL to create the control points of the splitted patch.	GENJOL		
B - Calls TEST1 to create the boundary vectors of the splitted (created) patch. C - Calls DRWJOL to draw the created patch. - Fourth option is to create curves of intersection of a patch with planes.			

A - Calls INTJOL to generate xm y, z coordinates of the curve of intersection.	INTJOL
- <u>Fifth option</u> is to modify a patch.	
A - Calls LPEN to define the patch number (tag).	
B - Calls TRAKXY to define x, y coordinates of the new point.	
C - Calls LPEN to define the number of the point to be relocated.	
D - Calls TEST1 to create the boundary vectors of the modified patch.	
E - Calls DRWJOL to draw the new (modified) patch.	
- <u>Sixth option</u> is to blend two adjacent but not touching patches with a third (created) patch.	
A - Calls GETB to get the blending functions (boundary matrices) of the first patch.	
B - Calls GETB to get the boundary matrices of the second patch.	
C - Calls BLEND to create the blending functions (boundary matrices) of the third (blended) patch.	BLEND
D - Calls GETBX to create the boundary vectors of the third patch.	
E - Calls DRWJOL to draw the third patch.	

- Seventh option is to save the display and create a corresponding data file. - Calls OBSAVE to save the x, y and z coordinates of the displayed patch (or patches).	OBSAVE	INFILE
- Eighth option is to recall the saved display. - Calls GETSHP to recall the x, y and z coordinates of the saved display.	GETSHP	
- Nineth option is to erase a predefined patch. A - Calls LPEN to define the number (tag) of the patch to be erased. B - Calls ERAS to erase the patch.		
- Tenth option is to stop and exit from the program.		

Figure (A.4) A Hierarchical Chart For Program 'KEY'

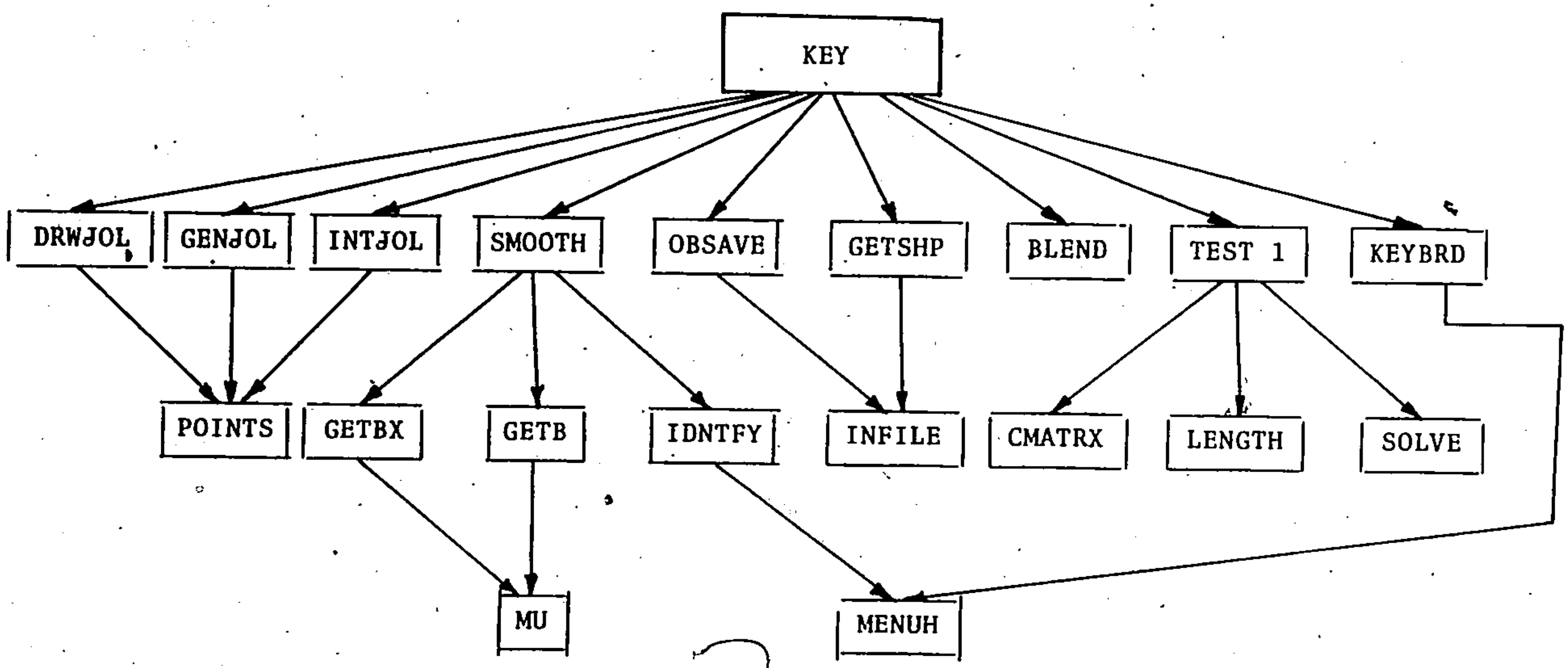


Figure (A.5) PROGRAM 'KEY'

Program DATAFL

Function

This program aids in the preliminary design of any curved surface using input data only from x,y,x co-ordinates stored on data files.

User's Manual

The program main flow from the user's point of view ~~is as~~ follows:

1 - User starts the mini-computer.

2 - User runs the program, type:

    • RUN DATAFL   <CR>

3 - Program responds by showing a menu of design commands.

4 - User hit the 'DATA FILE' light button with the LP.

5 - Program responds by typing the following message:

    'PLEASE ENTER THE NUMBER OF PATCHES (DATA FILES)'

6 - User enter the number of data files e.g.

    1   <CR>

7 - Program responds by typing:

    'ENTER THE NAME OF FILE NUMBER 1'

8 - User enters the file name (containing the data points defining the patch) e.g.

    PATCH 1   <CR>

9 - Program responds by displaying the patch surface and the design commands menu.

10 - User is now ready to modify the surface; using the light button commands; till he is satisfied with the design.

6-3-6 HIERARCHICAL CHART AND LISTING FOR PROGRAM DATAFL

A [HIPO] chart for program DATFL is shown in Figure (A.6). Figure (A-7) shows the three overlay regions of the program. Listing for program DATAFL, together with listing of the command file DATAFL.COM are also included.

MAIN PROGRAM (DATAFL)	SUBPROGRAM LEVEL 1	SUBPROGRAM LEVEL 2	SUBPROGRAM LEVEL 3
- Accepts input data from stored data files containing x,y,z coordinates defining the surface.			
- Calls MENUH to choose a design option.			MENUH
- First option is to input data points defining the surface patches via data files.			
A. Calls DATFIL to return to the main program the x,y,z coordinates of a patch or a group of patches, stored on data files.	DATAFIL used to return to the main program the x,y, and z coordinates of a patch or a group of patches, stored on data files.		
- The rest of this hierarchical chart is exactly the same as the one of program KEY, except that here we have an extra design option.	Input- number of data files (patches). Output- x,y,z coordinates stored on the data files.		

- Tenth option is to rotate the designed surface about any arbitrary axis.
  - A. Calls INTREV to return to the main program the x,y,z coordinates of the two points defining the axis of rotation.
  - B. Calls ROTATE to rotate the predicted points describing the surface about the axis of rotation.
  - C. Calls TEST 1 to create the new x,y and z boundary vectors of the rotated patch.
  - D. Calls DRWJOL to draw the rotated patch.
- Eleventh option is to exit and stop the program.

INTREV
used to return to the main program the x,y,z coordinates of the two points defining the axis of rotation.

Output- x,y,z coordinates of the two points defining the axis of rotation.

Figure (A.6) A Hierarchical Chart for Program DATAFL

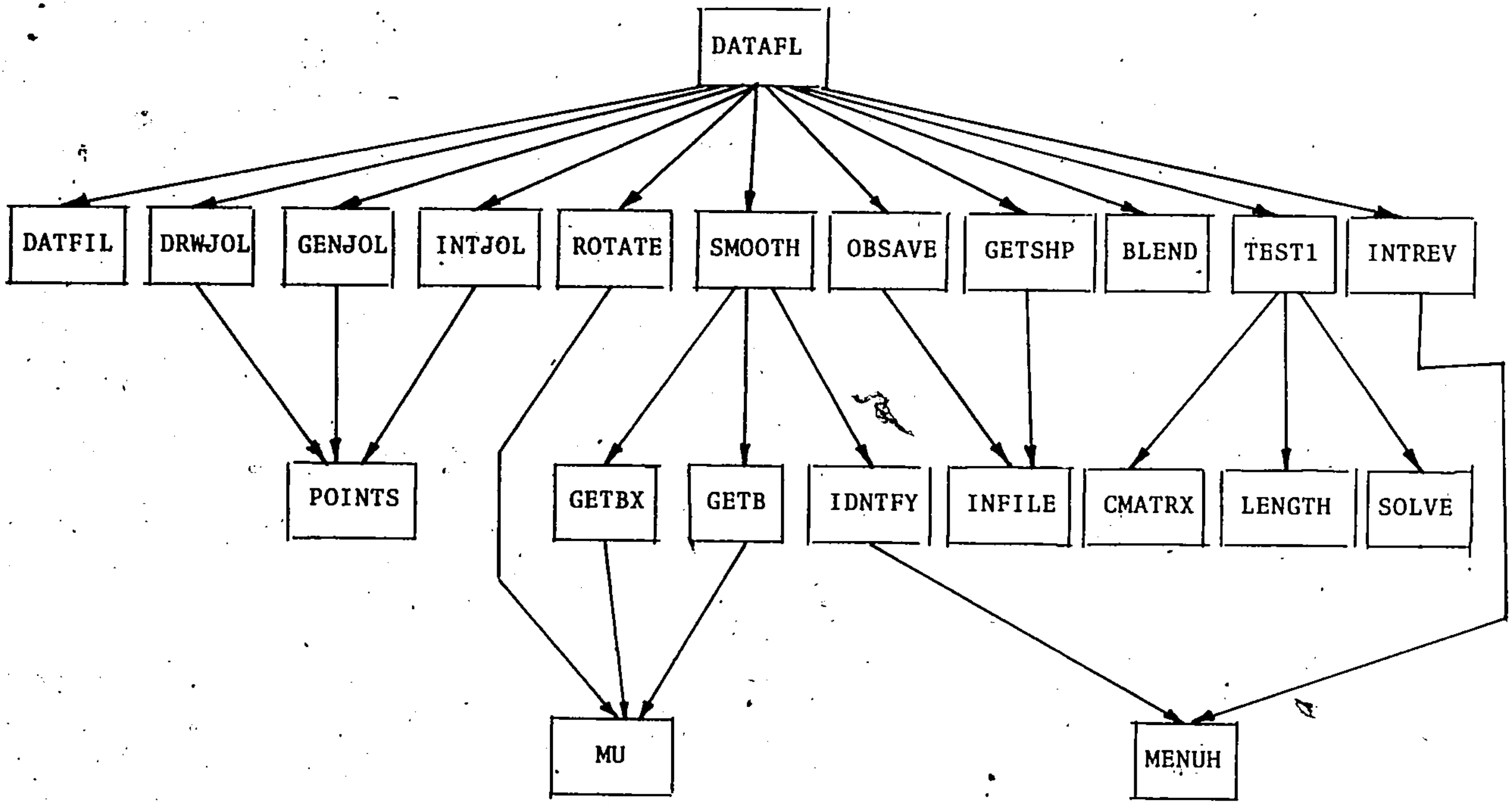


Figure (A.7) PROGRAM "DATAFL"

PROGRAM SEC1

FUNCTION

This program allows the user to perform shape modification of a certain patch. It also allows him to interactively translate his patch and to obtain the resultant continuous curve from the intersection of a plane with his patch.

USER'S MANUAL

1 - create your own data file containing the X, Y, Z coordinates of 16 points defining the patch, to do so

a - start up the computer

b -

```
b - . R EDIT
      * EWDXL: NAME. DAT $$
      * I  X1, Y1, Z1
           X2, Y2, Z2

          X16, Y16, Z16
   $$
   EX$$
```

Remarks

- NAME is the name of your data file
- The underlined commands are commands entered by the user through the DECWRITER,
- DX1 is a user floppy disk, if the user wants to load his data file on any other device he should replace DX1 by the device name (e.g. RK2 for a magnetic disk).

2 - Run program SEC1, write

```
. RUN RK2: SEC1
```

The program will respond by writing

```
THIS PROGRAM ALLOWS PATCH MODIFICATIONS
ENTER THE DATA FILE NAME
*DX1: NAME.DAT
```

3 - The program will respond by drawing the sixteen points with straight lines joining each successive point. A menu area will be seen on the left hand side of the CRT containing four options, move point, intersect, translate and done (see Figure A.8).

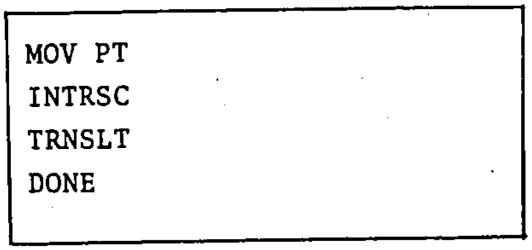


Figure (A.8) A typical plotting of a patch on the CRT showing the menu area

4 - The user is free now to choose anyone of the options on the menu area by just pointing at it with the light pen.

5 - To aid in understanding the option 'MOV PT' see SUBROUTINE MODPAT, 'INTRSC' see SUBROUTINE INTRSC, 'TRNSLT' see SUBROUTINE DRAWP and D3TRNS.

6 - At any time the user can exit by pointing at 'DONE' option.

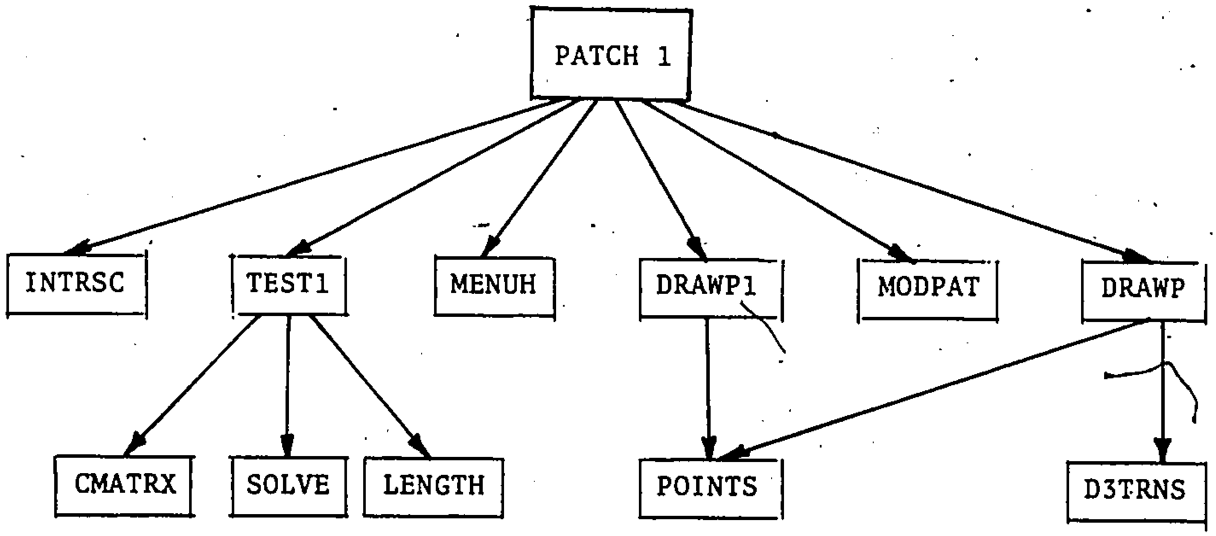


Figure (A.9) FLOW CHART FOR PROGRAM 'SEC1'

PROGRAM GENRAT

FUNCTION

This program allows the user to generate a patch from a given patch.

USER'S MANUAL

1. Start up the computer.
2. Run program GENRAT type

   .RUN RK2: GENRAT

3. The program will respond by asking the user to enter x, y, z coordinates of the original patch (i.e. define patch).

   'ENTER X, Y, Z COORDINATES'

4. The user should respond by typing the x, y, z coordinates of a 16 points defining his patch.
5. The program will then display the patch on the CRT in the 3-D space.
6. The program will ask the user to define the borders of the patch to be generated from the given patch.

   'ENTER U9L), U(2), W(1), W(2)'

7. The user should respond by supplying these values e.g.

   0.2, 0.6, 0.2, 0.6 <CR>

8. The program will generate the new patch on the original patch (see Figures A.9 and A.10).

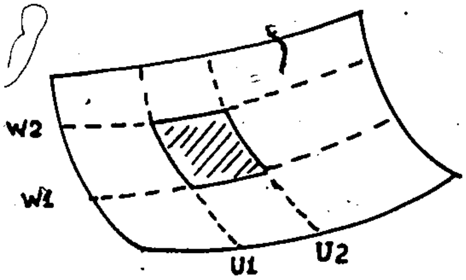


Figure (A.10) Geometry for Creating a patch by subdividing a given patch

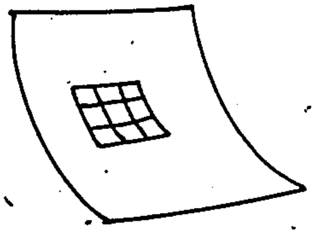


Figure (A.11) PC patch subdivision

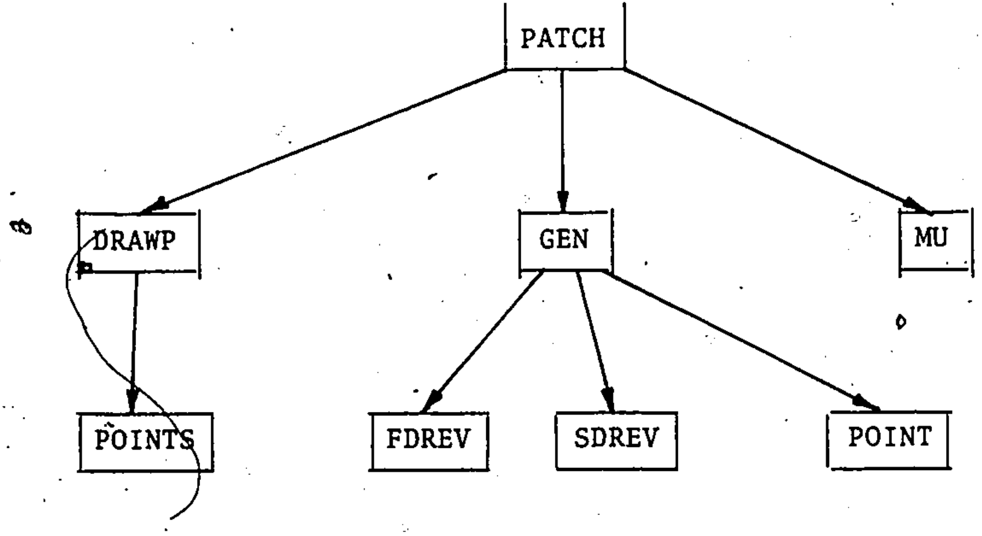


Figure (A.12) FLOW CHART OF PROGRAM 'GENRAT'

PROGRAM "HIDDN.SAV"

FUNCTION

This program performs a hidden line removal operation. User enters the patch file name, program displays the patch with hidden lines removed.

USER'S MANUAL

1. Create your own data file containing the X, Y, Z coordinates of 16 points defining the patch surface.
2. Run program HIDDN, type

```
.RUN RK2: HIDDN <CR>
```

The program will respond by typing,

```
'THIS PROGRAM SHOWS THE PATCH WITH HIDDEN LINES
REMOVED'.

'ENTER THE DATA FILE NAME'.
```

3. User enters the data file name, e.g.

```
FILNAM <CR>
```

4. Program immediately displays the patch with hidden lines removed.

APPENDIX (B)

CURVE DESIGN PROGRAM

INTRODUCTION

An interactive computer program was designed to aid in the use of computer graphics in the design of curves (two-dimension).

The main purpose is to simplify the design process so that the designer is not required to know things not particularly relevant to his role in the design process. The requirements of the formulation were that it must automatically maintain curve continuity and yet allow changes in shape information to be specified by moving control points (using LP) that affect the curve in an intuitive way.

CURVE DESIGN

To satisfy the real time constraint, the curves must be computable by a very fast algorithm. This requirement was satisfied by using the parametric basis - splines [B - spline] , [78] .

A B-spline curve "intuitively" mimics the shape of control polygon, which is an ordered sequence of points (1, 2, 3,..., N) as shown in Figure (B.3). The curve that follows the shape of this control polygon is composed of a sequence of spline segments. The designer enters the control points on the CRT (using LP), the program generates the piecewise curve using a continuous first derivative B-spline curve as shown in Figure (B.3).

A menu area on the CRT is assigned to give the user the ability to interact with the program as shown in Figure (B.1).

EXAMPLE ON HOW TO USE PROGRAM CURDES

1. START UP THE COMPUTER
2. Type

   .RUN RK2: CURDES
3. A menu containing different options will then be drawn on the CRT (Fig. (B.1).

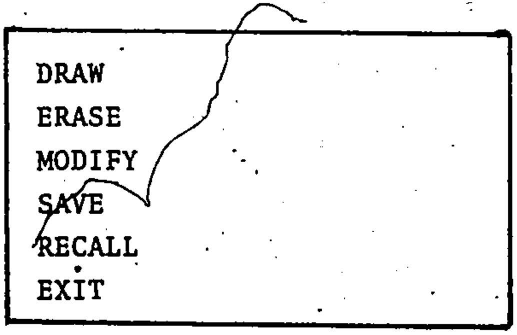


Figure (B.1) Menu Area

4. The user can now point at 'DRAW' with the light pen.
5. Another menu will be drawn on the CRT (Fig. B.2) and also a tracking object will be seen on the center of the CRT.

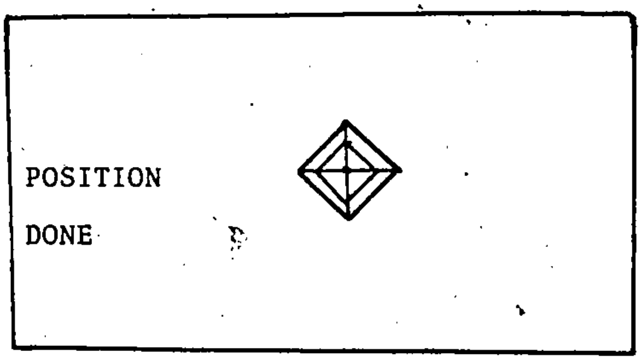


Figure (B.2) Menu area after the user has pointed at 'DRAW'

6. The user can position the tracking object at any point on the CRT and then press < CR > (RETURN KEY) to enter the control points that control the shape of 2-D curve, after he is done, he can point at 'DONE'.

The program then generates the curve automatically (Fig. (B.3)).

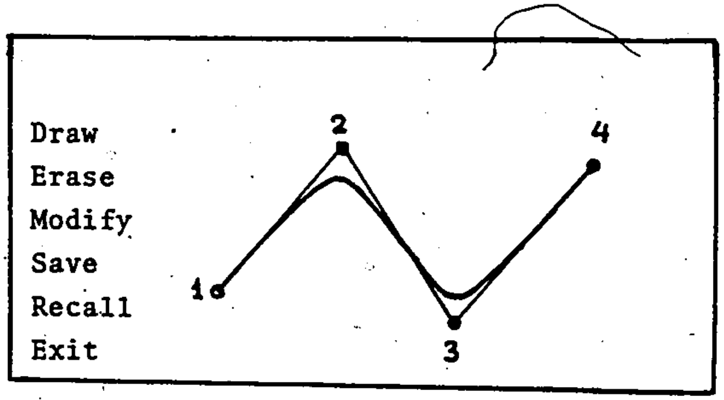
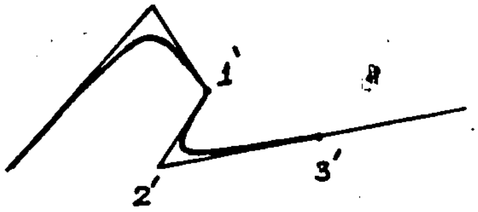


Figure (B.3) The control points (1, 2, 3, 4) and the resulting curve

7. The user can then choose any other option (including 'DRAW'); he can erase, modify, save and recall his curve. It is worth mentioning here that the 'MODIFY' option will allow the user to erase any <u>part</u> of his curve. Suppose that the user has designed the following curve (Figure B.4).

and he wants to modify that curve to the following (Figure B.5).



He can do that by splitting the lines at points 1 and 2 and then erasing the curve in between, then using the draw option to enter the new control points 1', 2' and 3'.

APPLICATIONS

The program can be used in an interactive way to design value curves [*] relating to consumer products. It can be also used to draw contour lines of any figure. See Figure (B.6).

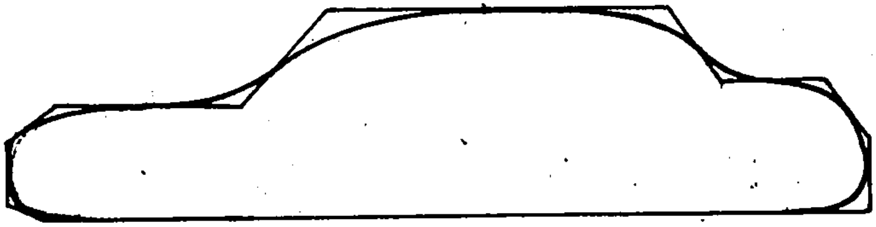


Figure (B.6) Demonstration of the possibilities of B-spline approximations

---

*J.N. Siddall, 'Value Theory and User Participation, Architectural Design, Vol. 42, No. 5 (May 1972), pp. 319-322.

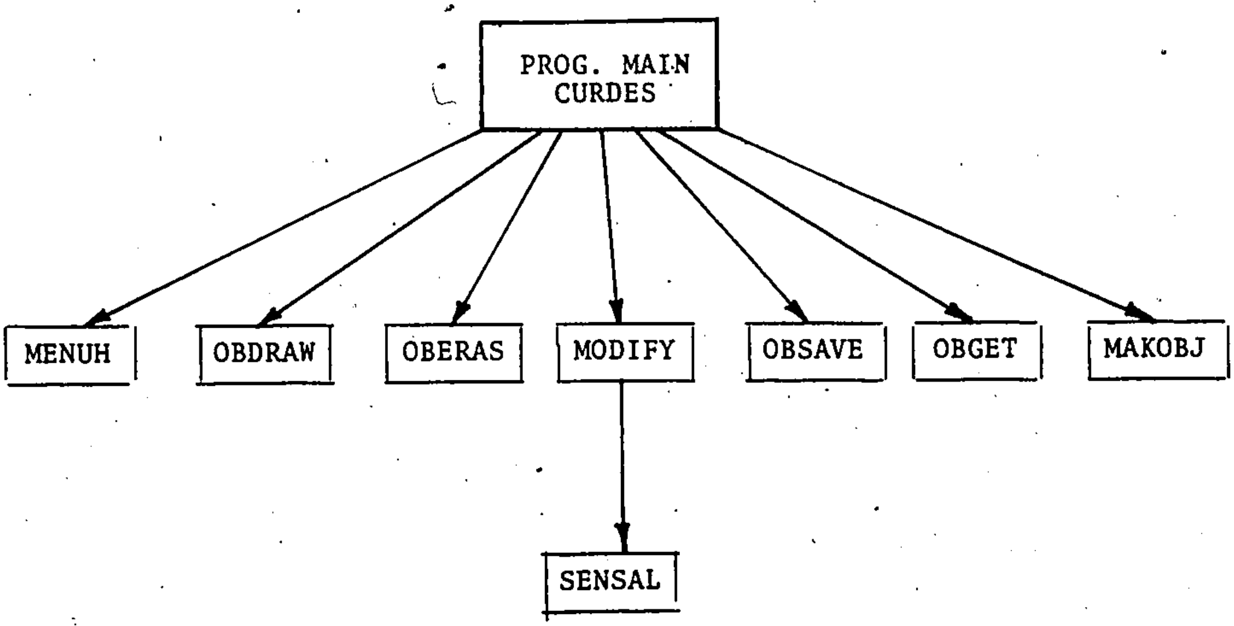


Figure (B.7) FLOW CHART FOR PROGRAM CURDES

PROGRAM CURDES

FUNCTION	- This program can be used to design curves in 2-D. The program sets a menu area on the screen showing the options of

DRAW
ERASE
MODIFY
SAVE
RECALL
EXIT

If the user points (with the light pen) at DRAW, a tracking (diamond shape) object will appear on the screen, and a message will be written on the Decwriter:

```
POSITION THE TRACKING OBJECT,
TYPE < CR>  WHEN DONE.
```

so the user can enter his control points (see Figure 3).

USAGE - .RUN RK2: CURDES < CR>

LANGUAGE - FORTRAN

SUBROUTINE MENUH (IT, M1, M2)

FUNCTION		- This SUBROUTINE detects a light pen hit on one of the options in the menu area.
USAGE		- CALL MENUH (IT, M1, M2)
PARAMETERS	IT	- Tag of the hidden subpicture
	M1	- Tag of the first option in the menu area.
	M2	- Tag of the last option in the menu area.
LANGUAGE		- FORTRAN

SUBROUTINE QBDRAW

FUNCTION - This subroutine draws a curve on the CRT, by positioning the control points.

USAGE - CALL OBDRAW.

LANGUAGE - FORTRAN

ALGORITHM - This subroutine uses the cubic spline algorith to draw a curve. For more information about cubic splines see reference [*] .

*George J. Peters, 'Interactive computer graphics applications of the parametric BI-CUVIC SURFACE', McDonnell Douglas Automation Company, St. Louis, Missouri.

SUBROUTINE OBERAS

FUNCTION	- This subroutine erases an object out of the CRT.
USAGE	- CALL OBERAS
LANGUAGE	- FORTRAN

SUBROUTINE MODIFY

FUNCTION - This subroutine modifies an object on the CRT. The user gets two options if a call is made to MODIFY, erasing any line and or, splitting a line, when he is done, he can point with the light pen on the option 'DONE', then a RETURN is made to the main program CURDES.

USAGE - CALL MODIFY

LANGUAGE - FORTRAN

SUBROUTINE OBSAVE

FUNCTION	- This subroutine saves the display in a file named by the user. When a call is made to OBSAVE, the program asks for a name to assign to the file to be saved.
USAGE	- CALL OBSAVE
LANGUAGE	- FORTRAN

SUBROUTINE OBGET

FUNCTION	- This subroutine recalls the file saved by OBSAVE, i.e., restores the display.
USAGE	- CALL OBGET
LANGUAGE	- FORTRAN

SUBROUTINE SENSAL

FUNCTION - This subroutine turns the light pen sensitivity on and off for all the objects.

USAGE - CALL SENSAL

LANGUAGE - FORTRAN

APPENDIX (C)

SHIP HULL FORM COEFFICIENTS

FORM COEFFICIENTS

In order to refer to certain proportions of ships; to compare them in form with regard to their actual dimensions or difference in dimensions; to be able to describe their shapes more precisely than "fat" or "thin", "full" or "fine"; there are certain geometric qualities that can be related as ratios or dimensionless coefficients. These coefficients of form are exceptionally useful in comparing certain performance characteristics associated with hydrodynamic phenomena.

In the following relationships, the symbols used are defined as follows:

L_{pp}	-	length between perpendiculars or designed waterline length
T	-	draft to the waterline, or draft
B	-	beam or breadth molded
	-	displacement volume at draft T
	-	area of midsection at draft T
	-	area of waterplane at draft T

The coefficients most commonly used by naval architects are as follows:

Midship section coefficient

$$C_m = \frac{Am}{BT}$$

Block coefficient

$$C_b = \frac{\Delta}{L_{pp}BT}$$

Prismatic coefficient

$$C_p = \frac{\Delta}{A_M L_{pp}} = \frac{\Delta}{C_M B T L_{pp}} = C_b/C_M$$

Waterline coefficient

$$C_{WP} = \frac{Aw}{BL_{pp}}$$

There are also certain commonly used ratios of dimensions, and these with their approximate range of values are:

Length - beam ratio

L_{pp}/B range, 3 to 12

Length - draft ratio

L_{pp}/T range, 7 to 30

Beam - draft ratio

B/T range, 1.8 to 4

Displacement-length ratio

Δ/L^3_{pp}

Displacement - length coefficient

$\Delta/(L_{pp}/100)^3$ range, 50 to 500

APPENDIX D

COON'S PATCHES

APPENDIX (D)

COON'S PATCHES

In Coon's notation points on the surface of a bicubic patch satisfy

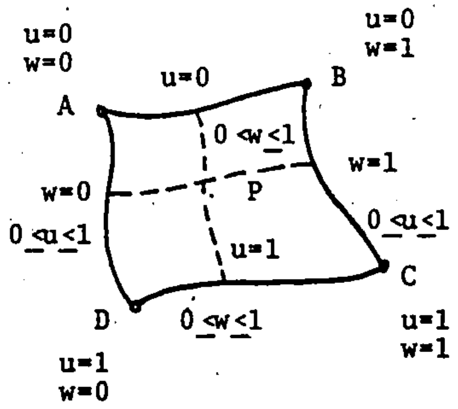
$$UW = U\ M\ B\ M^T\ W^T$$

$$= [u^3\ u^2\ u\ 1]\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & 3 & -2 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 00 & 01 & 00W & 01W \\ 10 & 11 & 10W & 11W \\ 00U & 01U & 00UW & 01UW \\ 10U & 11U & 10UW & 11UW \end{bmatrix}$$

$$\begin{bmatrix} 2 & -3 & 0 & 1 & W^3 \\ -2 & 3 & 0 & 0 & W^2 \\ 1 & -2 & 1 & 0 & W \\ 1 & -1 & 0 & 0 & 1 \end{bmatrix}$$

The so-called "boundary conditions matrix" is in fact a tensor - each entry above (e.g., 00 01W, etc.) is a vector. The <u>corners</u> (points A, B, C and D) of the patch (00 01 11 10) are vectors relative to the origin (and axes) of the design, the <u>slope</u> vectors (00W 01W 01U 11U), etc. are vectors relative to their respective patch corners. Further, the <u>twist</u> vectors are (best thought of) relative to the point on the parallelogram completed from respective slope vectors.

A qualitative description of Coon's patches may be useful. A patch has four boundary curve segments (edges) which meet at four points (patch corners) in the fashion suggested by Figure D.1.



Point P is at
$x = f(u,w)$
$y = g(u,w)$
$z = h(u,w)$

Figure D.1 Parameters on Coon's Patch

APPENDIX (E)

THE MATHEMATICAL FORMULATION OF SMOOTHING TWO PATCHES AT THE COMMON BORDERS

APPENDIX (E)

THE MATHEMATICAL FORMULATION OF SMOOTHING TWO PATCHES AT THE COMMON BORDERS

Consider Figure E.1 in which patches P_1 and P_2 have a common border and we wish to smooth the two patches such that first order $C^{(0)}$ and second order $C^{(1)}$ continuity exist across that common boundary. We know (from Chapter 2) that the PC bi-cubic surface patch is defined as

$$v(u,w) = (u^3\ u^2\ u\ 1)\ (M)\ (B)\ (M)^T \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix}$$

where (M) is a constant matrix defined in the equation and (B) is the boundary matrix

$$(B) = \left[\begin{array}{cc|cc} V_{00} & V_{01} & V_{00W} & V_{01W} \\ V_{10} & V_{11} & V_{10W} & V_{11W} \\ \hline V_{00U} & V_{01U} & V_{00UW} & V_{01UW} \\ V_{10U} & V_{11U} & V_{10UW} & V_{11UW} \end{array}\right]$$

From Figure E.1, to ensure the $C^{(0)}$ continuity between P_1 and P_2;

$\left[\frac{\partial v(u,w)}{\partial u}\right]_{u=1}$ of P_1 should be equal to

$\left[\frac{\partial v(u,w)}{\partial u}\right]_{u=0}$ of P_2.

That is

$$(u = 1, \text{ row } 2)_{P_1} = (u = 0, \text{ row } 1)_{P_2}$$

To ensure $C^{(1)}$ continuity between P_1 and P_2;

$\left[\frac{\partial^2 v(u,w)}{\partial u \partial w}\right]_{u=1}$ of P_1 should be equal to

$\left[\frac{\partial^2 v(u,w)}{\partial u \partial w}\right]_{u=0}$ of P_2

so that $(u = 1, \text{ row } 4)_{P_1} = (u = 0, \text{ row } 3)_{P_2}$

Therefore, adjacent patches have position and slope continuity if common position rows (or columns) are identical and if common slope rows (or columns) are multiples of each other. Graphically, the elements of interest in the (B) matrices for both patches of Figure E.1 are as shown in Figure E.2.

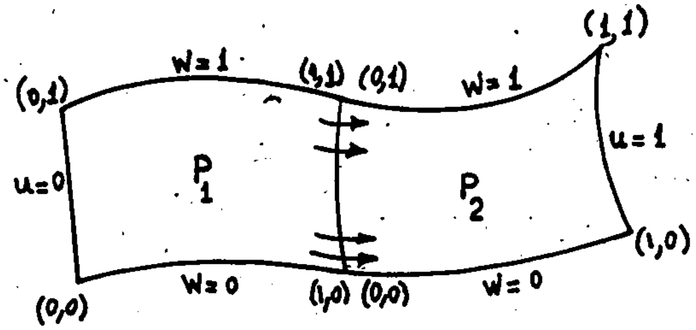


Figure E.1 Geometry for Smoothing Between Two Patches at a Common Border

				b_{21}	b_{22}	b_{23}	b_{24}
b_{21}	b_{22}	b_{23}	b_{24}				
				b_{41}	b_{42}	b_{43}	b_{44}
b_{41}	b_{42}	b_{43}	b_{44}				

Figure E.2 Graphical Representation of the Boundary (B) Matrices Elements of Interest of the Two Patches of Figure E.1

APPENDIX (F)

THE MATHEMATICAL FORMULATION OF BLENDING BETWEEN TWO NON-ADJACENT PATCHES

APPENDIX (F)

THE MATHEMATICAL FORMULATION OF BLENDING BETWEEN TWO NON-ADJACENT PATCHES

Figure F.1 shows patches P_1 and P_2 to be blended by patch P_3 such that it ensures $C^{(0)}$ and $C^{(1)}$ continuity at the respective common borders. The method applied here is the same as that used in Appendix E on smoothing between two patches at a common border, i.e., to ensure $C^{(0)}$ continuity the following relationship between the (B) matrices of P_1, P_2 and P_3 should be preserved.

$(u=1,\ \text{row } 2)_{P_1} = (u=0,\ \text{row } 1)_{P_3}$,

$(u=0,\ \text{row } 1)_{P_2} = (u=1,\ \text{row } 2)_{P_3}$

and to ensure $C^{(1)}$ continuity

$(u=1,\ \text{row } 4)_{P_1} = (u=0,\ \text{row } 3)_{P_3}$ and

$(u=0,\ \text{row } 3)_{P_2} = (u=1,\ \text{row } 4)_{P_3}$.

Graphically, the previous relationships are shown in Figure F-2.

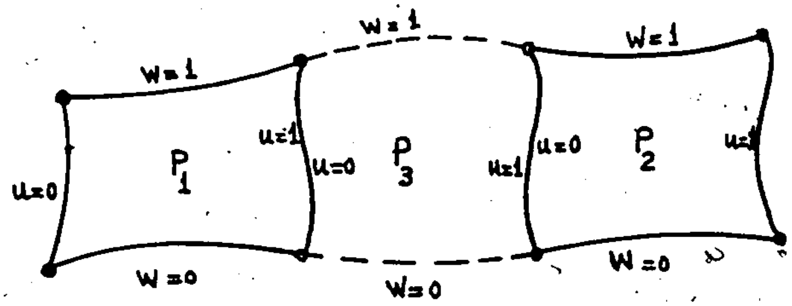


Figure F.1 Geometry for Blending Between Two Non-adjacent Patches

P_1				P_3				P_2			
								B_{11}	B_{12}	B_{13}	B_{14}
b_{21}	b_{22}	b_{23}	b_{24}	B_{11}	B_{12}	B_{13}	B_{14}				
				b_{41}	b_{42}	b_{43}	b_{44}	B_{31}	B_{32}	B_{33}	B_{34}
b_{41}	b_{42}	b_{43}	b_{44}	B_{31}	B_{32}	B_{33}	B_{34}				

Figure F.2 Boundary (B) Matrices Elements of Interest for Blending Between Two Non-adjacent Patches

APPENDIX (G)

THE MATHEMATICAL FORMULATION OF INTERSECTING A PATCH WITH A PLANE

APPENDIX (G)

THE MATHEMATICAL FORMULATION OF INTERSECTING A PATCH WITH A PLANE

Consider Figure G.1 in which a patch is represented in the u-w parametric plane. In order to generate cut points to be fitted later by a parametric cubic curve, we must calculate the corresponding u,w values of the cut plane, which is represented by a line on Figure G.1. We have

$$\Delta w = w_2 - w_1 \quad , \quad w_2 > w_1$$

where w_1 and w_2 are obtained by intersecting the patch's boundary with the plane. For any point on the curve of intersection, let

$$\text{DELTAW} = \frac{w-w_1}{w_2-w_1}$$

hence, $w = w_1 + \text{DELTAW} \cdot \Delta W$

for $w = w^*$

$$w^* = w_1 + \text{DELTAW} \cdot \Delta W$$

and u^* is produced by cutting $w = w^*$ by the plane. Hence, the point on the patch surface $v(u^*,w^*)$ is easily computed since we already know the patch's boundary matrix. The points produced in this fashion are then fitted with a parametric cubic curve to give the actual continuous curve of intersection.

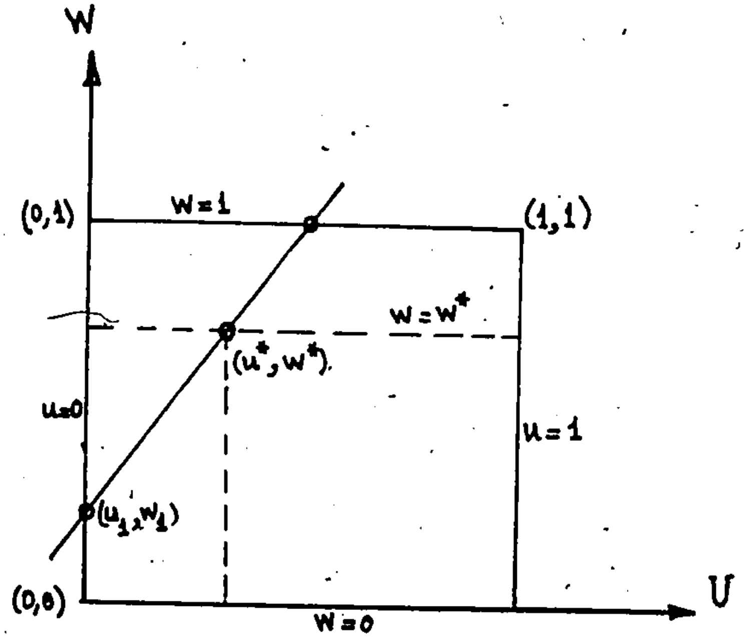


Figure G.1 Representation of a Patch and a Cut Plane in U-W Parametric Plane

APPENDIX (H)

CREATING BOUNDARY MATRIX (B) OF A PATCH GENERATED BY SPLITTING A GIVEN PATCH

APPENDIX (H)

CREATING BOUNDARY MATRIX (B) OF A PATCH GENERATED BY SPLITTING A GIVEN PATCH

Referring to Figure 3.22 of Chapter 3, section 3.3.8, our task is to obtain the boundary (B) matrix of patch P_2, knowing the boundary (B) matrix of patch P_1. Before we do that, let us first consider splitting a parametric cubic curve. Asume we wish to define a new curve from u_1 to u_2 on the original curve v(u) as shown on Figure H.1. Using the linear transformation

$$u = u_1 + t\ (u_2-u_1)$$

where t is equivalent to u, but for the new curve segment, then

$$\frac{du}{dt} = u_2-u_1$$

then at t=0, $u=u_1$; and at t=1, $u=u_2$

$$\frac{dV}{dt} = \frac{dV}{du}\frac{du}{dt} \tag{H.1}$$

Using Equation (H.1) and denoting the new curve by C(t), the geometric coefficients of the split curve in terms of the original curve are:

$$\begin{bmatrix} C(0) \\ C(1) \\ C'(0) \\ C'(1) \end{bmatrix} = \begin{bmatrix} V(u_1) \\ V(u_2) \\ (u_2-u_1)V'(u_1) \\ (u_2-u_1)V'(u_2) \end{bmatrix} \tag{H.2}$$

Note that C(0) and C(1) represent position data pertaining to end points of the split curve, and C'(0), C'(1) represent a parametric slope data at the corresponding end points.

Now, following the same argument with respec to the PC bi-cubic surface patch splitting, and referring to Figure 3.22 we have

$$[V(0,0)]_{P_2} = [V(u_1,w_1)]_{P_1}$$

and

$$[V(0,0)_u]_{P_2} = (u_2-u_1)[V(u_1,w_1)_u]_{P_1}$$

The corner cross-derivatives (twists) for P_2 are obtained by evaluating the cross-derivatives of P_1 at the given u, w values, e.g.,

$$[V(1,1)_{uw}]_{P_2} = \left[\frac{\partial^2 V(u,w)}{\partial u \partial w}\right]_{P_1 \substack{u=u_2 \\ w=w_2}}$$

$$[V(0,0)_{uw}]_{P_2} = \left[\frac{\partial^2 V(u,w)}{\partial u \partial w}\right]_{P_1 \substack{u=u_1 \\ w=w_1}}$$

$$[V(0,1)_{uw}]_{P_2} = \left[\frac{\partial^2 V(u,w)}{\partial u \partial w}\right]_{P_1 \substack{u=u_1 \\ w=w_2}}$$

and

$$[V(1,0)_{uw}]_{P_2} = \left[\frac{\partial^2 V(u,w)}{\partial u \partial w}\right]_{P_2 \substack{u=u_2 \\ w=w_1}}$$

Therefore, the boundary matrix (B) of the split patch can be easily obtained.

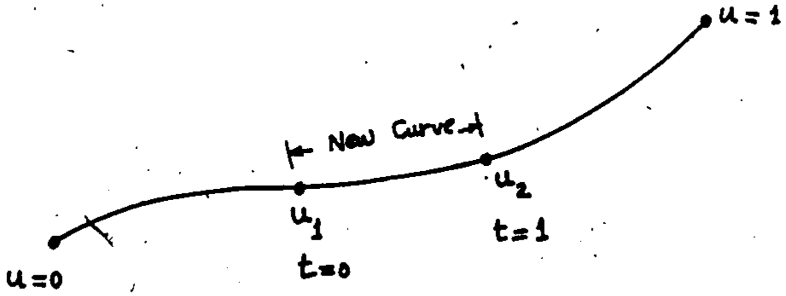


Figure E.1 Geometry for Splitting a Parametric Cubic Curve

181a

APPENDIX (I)

LISTINGS FOR THE GENERAL SUBROUTINES

APPENDIX (I)

GENRAL SUBROUTINES

1 - SUBROUTINE TEST 1
2 - SUBROUTINE CMATRX
3 - SUBROUTINE SOLVE
4 - SUBROUTINE LENGTH
5 - SUBROUTINE POINTS
6 - SUBROUTINE DRAWP
7 - SUBROUTINE D3TRNS
8 - SUBROUTINE MENUH
9 - SUBROUTINE MINV
10 - SUBROUTINE MU
11 - SUBROUTINE HIDE
12 - SUBROUTINE LOOKUP
13 - SUBROUTINE PDATAX
14 - SUBROUTINE GEN
15 - SUBROUTINE FDREV
16 - SUBROUTINE SDREV
17 - SUBROUTINE DRAWP1
18 - SUBROUTINE DRAWP2
19 - SUBROUTINE BLEND
20 - SUBROUTINE MODPAT
21 - SUBROUTINE INTRSC
22 - SUBROUTINE MODPAT
23 - SUBROUTINE BETA

24 - SUBROUTINE BETA1

25 - SUBROUTINE BXBYBZ

26 - SUBROUTINE TOTA

SUBROUTINE BLEND (BX, BBX, BY, BBY, BZ, BBZ, FBX, FBY, FBZ)

FUNCTION - This SUBROUTINE blends two non-adjacent patches.

USAGE - - CALL BLEND (BX, BBX, BY, BBY, BZ, BBZ, FBX, FBY, FBZ).

PARAMETERS BX - Input (B) matrix of patch P1. for the x-coordinates.

BBX - Input (B) matrix of patch P2 for the x-coordinates.

BY - Input (B) matrix of patch P1 for the y-coordinates.

BBY - Input (B) matrix of patch P2 for the y-coordinates.

BZ - Input (B) matrix of patch P1 for the z-coordinates.

BBZ - Input (B) matrix of patch P2 for the z-coordinates.

FBX - Output (B) matrix for patch P3 for the x-coordinates.

FBY - Output (B) matrix for patch P3 for the y-coordinates.

FBZ - Output (B) matrix for patch P3 for the z-coordinates.

LANGUAGE - FORTRAN

SUBROUTINE TEST1 (X, Y, Z, BX, BY, BZ)

FUNCTION - This SUBROUTINE calculates the BX, BY and BZ vectors of a patch defined by 16 x, y and z triplets.

USAGE - CALL TEST1 (X, Y, Z, BX, BY, BZ)

PARAMETERS
X - Input vector of length 16 of the data points defining the patch (x-coordinates)
Y - same as x, but (Y-coordinates)
Z - same as X and Y (Z-coordinates)
BX - Output BX vector of length 16
BY - Output BY vector of length 16
BZ - Output BZ vector of length 16

LANGUAGE - FORTRAN

SUBROUTINE CMATRX (U, W, C, N)

FUNCTION		- This SUBROUTINE calculates the C matrix of a patch.
USAGE		- CALL CMATRX (U, W, C, N)
PARAMETERS	U	- Input vector of U values of the transformation U-W plane (see Fig.2.2) of length N.
	W	- Input vector of W values of the transformation U-W plane (see Fig.2.2) of length N.
	C	- Output C matrix dimensioned N by N.
LANGUAGE		- FORTRAN

SUBROUTINE SOLVE (R, A, M, N, EPS, IER)

FUNCTION - This SUBROUTINE solves a general system of simultaneous linear equations.

USAGE - CALL SOLVE (R, AM, M, N, EPS, IER)

PARAMETERS R - The M by N matrix of right hand sides (destroyed), on return R contains the solution of the equations.

A - The M by M coefficient matrix (destroyed).

M - The number of equation in the system.

N - The number of right hand side vectors.

EPS - An input constant which is used as relative tolerance for test on loss of significance.

METHOD - Solution is done by means of Gauss-Elimination with complete pivoting.

SUBROUTINE LENGTH (L, X, Y, Z)

FUNCTION - This SUBROUTINE calculates the length between each successive points.

USAGE - CALL LENGTH (L, X, Y, Z).

PARAMETERS L - Output vector of the lengths between the points. If the number of points defining the patch is 16, L should be dimensioned by L(24) in the calling program.

X - Input vector of the X-coordinates of the points defining the patch.

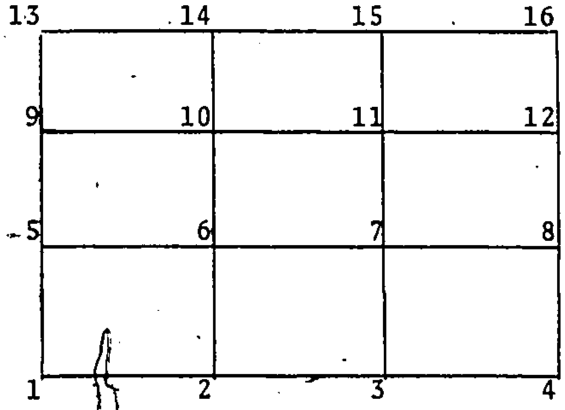
Y - Input vector of the Y-coordinates of the points defining the patch.

Z - Input vector of the Z-coordinates of the points defining the patch.

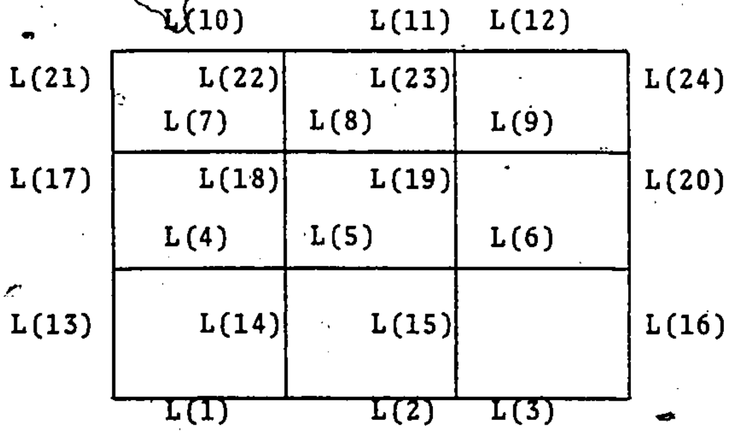
LANGUAGE - FORTRAN

EXAMPLE

If the points defining a patch were entered with the following ordering



the output lengths will have the following ordering.



SUBROUTINE POINTS (U, W, B, X, N)

FUNCTION - This SUBROUTINE calculates the coordinate (X <u>or</u> Y or Z) of N points lying on the surface of a patch; defined by the corresponding B vector (BX <u>or</u> BY or BZ); at a corresponding U and W values.

USAGE - CALL POINTS (U, W, BY, Y, 16)

PARAMETERS

U - vector of V values of length N

W - vector of W values of length N

B - BX or BY or BZ vector defining the patch.

X - output vector containing the X (or Y or Z depending on the call) coordinate values of the points lying on the surface.

LANGUAGE - FORTRAN

SUBROUTINE DRAWP (BX, BY, BZ, IFLAG)

FUNCTION - This SUBROUTINE draws the patch on the CRT.

USAGE - CALL DRAWP (BX, BY, BZ, IFLAG).

PARAMETERS

BX - Input boundary vector of a patch defined by 16 points.

BY - Input boundary vector of a patch defined by 16 points.

BZ - Input boundary vector of a patch defined by 16 points.

IFLAG = 0 only DRAWP draws the patch.

IFLAG = 1 DRAWP calls subroutine D3TRNS to perform translation of the patch to another location on the CRT.

LANGUAGE - FORTRAN

SUBROUTINE D3TRNS (N, X, Y, Z, L, M, NN)

FUNCTION - This SUBROUTINE performs a 3-D translation.

USAGE - CALL D3TRNS (N, X, Y, Z, L, M, NN)

PARAMETERS
N - Number of X, Y, Z triplets.
X - Array containing X-coordinates, dimensioned with N.
Y - Array containing Y-coordinates dimensioned with N.
Z - Array containing Z-coordinates dimensioned with N.
L - X-translation factor (real)
M - Y-translation factor (real)
NN - Z-translation factor (real)

LANGUAGE - FORTRAN

SUBROUTINE MENUH (IT, M1, M2)

FUNCTION - This SUBROUTINE delays execution of the program till a light pen hit is detected on the menu area, and then returns the order of the hit subpicture.

PARAMETERS IT - Order of the hit subpicture, e.g. if there are three options on the menu area and the user points at the second option, MENUH will return IT = 2.

M1 - Tag of the first subpicture.

M2 - Tag of the last subpicture.

USAGE - CALL MENUH (IT, M1, M2)

LANGUAGE - FORTRAN

SUBROUTINE DRAWP1 (BX, BY, BZ, X, Y, Z)

FUNCTION - Same as subroutine DRAWP, the only difference is that DRAWP1 puts the points of the patch on a set of subpictures beginning with a tag of 1000 and ending with a tag of 1032, and the lines joining these points on another set of subpictures beginning with a tag of 102 and ending with a tag of 1034. So, the user can OFF any of these subpictures from his calling program.

SUBROUTINE MODPAT (XOX, YOY, ZOZ, X1, Y1, Z1)

FUNCTION - This SUBROUTINE modifies a patch, by means of a tracking object which can be used to drag one of the original points lying on the patch surface to another position.

USAGE - CALL MODPAT (XOX, YOY, ZOZ, X1, Y1, Z1).

PARAMETERS
X1 - Array of the original X coordinates.
Y1 - Array of the original Y coordinates.
Z1 - Array of the original Z coordinates.
XOX - Array of the modified X coordinates.
ZOZ - Array of the modified Z coordinates.

LANGUAGE - FORTRAN

SUBROUTINE INTRSC (BX, BY, BZ)

FUNCTION - This SUBROUTINE draws the resultant continuous curve from the intersection of a plane and a patch defined by BX, BY, BZ vectors, on the CRT.

USAGE - CALL INTRSC (BX, BY, BZ).

PARAMETERS BX - I/P BX vector of the patch.
BY - Input BY vector of the patch.
BZ - Input BZ vector of the patch.

LANGUAGE - FORTRAN

EXAMPLE

If a call is made to subroutine INTRSC, a representation of the patch on the U-W plane will be drawn on the CRT (See Figure I.1), also a tracking object will appear on the center of the CRT.

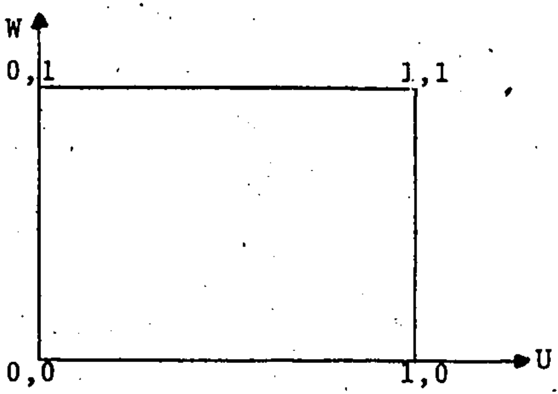


Figure (I.1) Representation of the patch on the U-W plane

The user is then asked to define the plane he wants to intersect the patch, since we are working on the U-W plane, the plane will be represented by a line. To define that line the user is asked to position the tracking object at two points on the borders of the patch (see Figure I.2).

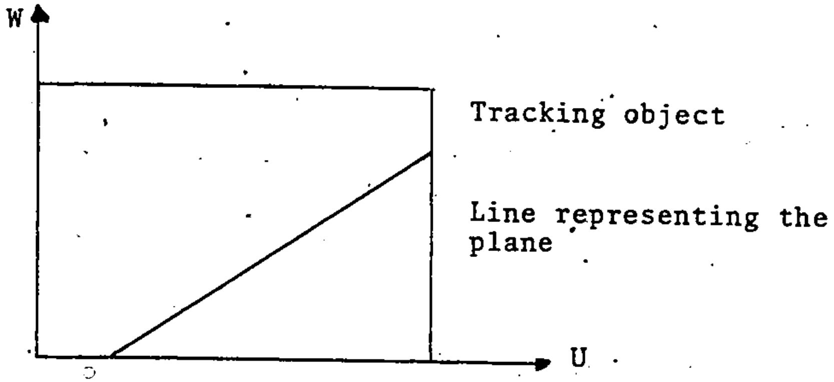


Figure (I.2) Patch and the interacting plane

The subroutine, then, draws automatically the resulting continuous curve.

This way the user is not asked to define his plane mathematically, only graphically.

SUBROUTINE BXBYBZ (BXI, BYI, BZI, BX, BY, BZ)

FUNCTION - This SUBROUTINE calculates the BX, BY, BZ vectors from the B matrices BXI, BYI, BZI of a patch. (refer to equation 13, S would be either one of BX. BY or BZ and B would be either one of BXI, BYI or BZI, e.g.

$$BX = [M][BXI][M]^T$$

definition of [M] matrix is given in equation (5).

USAGE - CALL BXBYBZ (BXI, BYI, BZI, BX, BY, BZ)

PARAMETERS
BXI- Input B matrix of the X coordinates.
BYI- Input B matrix of the Y coordinates.
BZI- Input B matrix of the Z coordinates.
BX - Output BX vector.
BY - Output BY vector.
BZ - Output BZ vector.

LANGUAGE - FORTRAN

SUBROUTINE BETA1 (BX2, BX1, BY2, BY1, BZ2, BZ1, BX14, BY14, BZ14, J, K, L, M, R1, R2, R3, R4)

FUNCTION - This SUBROUTINE blends two patches in the Y direction (refer to the theory of blending patches in this report).

USAGE - CALL BETA1 (BX2, BX1, BY2, BY1, BZ2, BZ1, BX14, BY14, J, K, L, M, R1, R2, R3, R4).

PARAMETERS

BX2 - Output 'B' matrix of the blended patch of the X coordinates.

BX1 - Input 'B' matrix of the original patch, of the X coordinates, to the right of the blended patch.

BX2 - Output B matrix of the blended patch of the Y coordinates.

BY1 - Input 'B' matrix of the original patch, of the Y coordinates, to the right of the blended patch.

BZ2 - Output 'B' matrix of the blended patch of the Z coordinates.

BZ1 - Input 'B' matrix of the original patch, of the Z coordinates, to the right of the blended patch.

BX14 - Input 'B' matrix of the original patch, of the X coordinates, to the left of the blended patch.

BY14 - Input 'B' matrix of the original patch, of the Y coordinates, to the left of the blended patch.

BX14 - Input 'B' matrix of the original patch, of the Z coordinates, to the left of the blended patch.

J - set equal to 1.

K - set equal to 2.

L - set equal to 3.

M - set equal to 4.

R1 - a ratio value; input by the user equal to $\frac{W1}{W2-W1}$ (See Figure) or equal to 1.

R2 - a ratio value, input by the user equal to (1-W2)/(W2-W1) or equal to 1.

R3- - V1/(V2-V1) or 1.

R4- - (1-V2)/(V2-V1) or 1.

SUBROUTINE BETA (BXB, BX1B, BYB, BY1B, BZB, BZ1B, BXXB, BYYB, BZZB, R1, R2, R3, R4)

FUNCTION - same as SUBROUTINE BETA 1, except BETA blends two patches in the X direction.

USAGE - CALL BETA (BXB, BX1B, BYB, BY1B, BZB, BZ1B, BXXB, BYYB, BZZB, R1, R2, R3, R4).

PARAMETERS - same as BETA 1.

LANGUAGE - FORTRAN.

SUBROUTINE TOTA (BBM, BBMT)

FUNCTION		- This subroutine calculates the inverse of the matrix M and M^T.
USAGE		- CALL TOTA (BBM, BBMT).
PARAMETERS	BBM	- output matrix = M^{-1}.
	BBMT	- output matrix = $(M^T)^{-1}$.
LANGUAGE		- FORTRAN

SUBROUTINE MU (A, B, R, N, M, L)

FUNCTION - This SUBROUTINE multiplies two general matrices.

USAGE - CALL MU (A, B, R, N, M, L)

PARAMETERS
A - Name of the first input matrix.
B - Name of the second input matrix.
R - Name of the resultant matrix.
N - Number of rows in A and columns of B.
M - Number of rows in B.
L - Number of rows in R.

LANGUAGE - FORTRAN

SUBROUTINE MINV (A, N, D, L, M)

FUNCTION		- This SUBROUTINE inverts a general non-singular matrix.
USAGE		- CALL MINV (A, N, D, L, M).
PARAMETERS	A	- Name of input matrix (destroyed) when return A is the inverted matrix.
	N	- Number of rows of A.
	D	- = 0 inversion correct = 1 inversion incorrect
	L	- Working array dimensioned with N.
	M	- Working array dimensioned with N.
LANGUAGE		- FORTRAN.

SUBROUTINE GEN (BX, BY, BZ, BGX, BGY, BGZ)

FUNCTION	-	This SUBROUTINE constructs the 'B' matrix of a patch generated by subdivision of another given patch.
USAGE	-	CALL GEN (BX, BY, BZ, BGX, BGY, BGZ).
PARAMETERS	BX -	Input BX vector of the given patch dimensioned with 16.
	BY -	Input BY vector of the given patch dimensioned with 16.
	BZ -	Input BZ vector of the given patch dimensioned with 16.
	BGX -	Output BX vector of the generated patch.
	BGY -	Output BY vector of the generated patch.
	BGZ -	Output BZ vector of the generated patch.
LANGUAGE	-	FORTRAN

SUBROUTINE SDREV (U, W, B, D)

FUNCTION		This SUBROUTINE calculates the second drevative of V(U,W).
USAGE		CALL SDREV (U, W, B, D).
PARAMETERS	U	Input U value to define a certain point on the patch at which SDREV calculates the second drevative.
	W	Input W value to define a certain point on the patch at which SDREV calculates the second drevative.
	B	The 'B' matrix of the patch.
	D	The output value of the drevative.
LANGUAGE		FORTRAN

SUBROUTINE FDREV (U, W, B, D, KOK)

FUNCTION - This SUBROUTINE calculates the first drevatives of V(U, W) (refer to equation 10).

USAGE - CALL FDREV (U, W, B, D, KOK).

PARAMETERS

U - Input U value to define a certain point on the patch.

W - Input W value to define a certain point on the patch.

B - The input 'B' matrix of the patch.

D - The output value of the dreative.

KOK - A flag if = 1 , D = $\frac{\partial V(U,W)}{\partial V}$

if = 2 , D = $\frac{\partial V(U,W)}{\partial W}$

LANGUAGE - FORTRAN

SUBROUTINE POINT (U, W, B, BG)

FUNCTION - This SUBROUTINE calculates the X or Y or Z coordinate of a point lying on a given patch at a corresponding U and W values.

USAGE - CALL POINT (U, W, B, BG).

PARAMETERS

U - Input U value.

W - Input W value.

B - Input 'B' vector defining the patch. (BX or BY or BZ).

BG - Output coordinates,

if B = BX, BG = X
, B = BY, BG = Y
, B = BZ, BG = Z

coordinate.

LANGUAGE - FORTRAN.

```
      SUBROUTINE GENJOL(BX,BY,BZ,X,Y,Z)
      DIMENSION BX(1),BY(1),BZ(1),X(1),Y(1),Z(1)
      DIMENSION U(2),W(2),UU(16),WW(16)
      COMMON/DFILE/IBUF(1)
      COMMON/SUPER/GRD,IP,IPO
C     SUBROUTINE GENJOL
C     FUNCTION:
C     GET X,Y Z COORD. OF THE PATCH GENERATED BY THE
C     SUBDIVI ION OF THE GIVIN PATCH
C     CALL IT X,Y,Z
C
      WRITE(5,3)
    3 FORMAT(1X,32HENTER U(1),U(2),W(1),W(2) VALUES)
      READ(5,*)U(1),U(2),W(1),W(2)
C
      DELU=(U(2)-U(1))/3.
      DELW=(W(2)-W(1))/3.
      UU(1)=U(1)
      UU(2)=U(1)+DELU
      UU(3)=U(1)+2.*DELU
      UU(4)=U(1)+3.*DELU
      DO 1 I=1,4
      WW(I)=W(1)
      WW(I+4)=W(1)+DELW
      WW(I+8)=W(1)+2.*DELW
    1 WW(I+12)=W(1)+3.*DELW
      DO 10 I=1,3
      UU(1+4*I)=UU(1)
      UU(2+4*I)=UU(2)
      UU(3+4*I)=UU(3)
   10 UU(4+4*I)=UU(4)
      CALL POINTS(UU,WW,BX,X,16)
      CALL POINTS(UU,WW,BY,Y,16)
      CALL POINTS(UU,WW,BZ,Z,16)
      RETURN
      END
```

```
      SUBROUTINE INTACT(X,Y,Z,NPATCH,NPL,IFL)
      DIMENSION X(1),Y(1),Z(1),XXX(32),X0(2),Y0(2),Z0(2)
      COMMON/DFILE/IBUF(1)
      COMMON/RECALL/IRECL
      COMMON/SUPER/GRD,IP,IPD
C       SUBROUTINE INTACT
C     FUNCTION:
C     THIS SUBROUTINE RETURNES THE X,Y,Z COORDINATES OF A PATCH
C     TO THE MAIN PROGRAM USING INPUT FROM LIGHT PEN.
C       NPATCH=NUMBER OF PATCHES      (I/P)
C     NPL=NUMBER OF PATCHES PER PLANE    (I/P)
C     IFL=1 DEFINITION OF CURVED SURFACES      (I/P)
C       IFL=2 DEFINITION OF SURFACES OF REVOLUTIONS  (I/P)
C
      WRITE(5,2)
    2 FORMAT(' PLEASE ENTER THE NUMBER OF PATCHES')
      READ(5,3)NPATCH
      WRITE(5,551)
  551 FORMAT(' PLEASE ENTER NUMBER OF PATCHES PER PLANE')
      READ(5,3)NPL
    3 FORMAT(I2)
      K=0
C
C     PLOT FRAME OF WORKING PLANES
C
      XX=512.
      CALL SUBP(IP)
      CALL APNT(XX,XX,,-4)
      CALL VECT(XX,0.0)
      CALL APNT(970.,460.,,-4)
      CALL TEXT(' Y')
      CALL ESUB
      CALL SUBP(IP+1)
      CALL APNT(XX,XX,,-4)
      CALL VECT(0.0,XX)
      CALL APNT(505.,1000.,,-4)
      CALL TEXT(' Z')
      CALL ESUB
      CALL SUBP(IP+2)
      CALL APNT(XX,XX,,-4)
      CALL VECT(-XX,-XX)
      CALL APNT(50.,35.,,-4)
      CALL TEXT(' X')
      CALL ESUB
      II=0
      KK=0
      KKK=0
C
      CALL SUBP(IP+4)
      CALL OFF (IP+4)
      CALL MENU(0.,500.,-100.,2900+IP,'POSITION','DONE')
      CALL ESUB
C
      CALL SUBP(IP+3)
```

```
          CALL OFF(IP+3)
          CALL MENU(2.0,600.,-100.,2000+IP,'DEFINE WORKING PLANE',
         * 'DONE')
          CALL ESUB
    C
          DO 4 J=1,NPATCH/NPL
          WRITE(5,5)J
     5    FORMAT(' DEFINE PATCHES IN PLANE NUMBER ',I2)
          WRITE(5,6)
     6    FORMAT('--------------------------',//)
    C
   700    IF(KKK.EQ.5.OR.KKK.EQ.9.OR.KKK.EQ.13.OR.KKK.EQ.17)KKK=KKK-1
          IF(KKK.EQ.21.OR.KKK.EQ.25.OR.KKK.EQ.29.OR.KKK.EQ.33)KKK=KKK-1
          IF(KKK.EQ.37.OR.KKK.EQ.41.OR.KKK.EQ.45.OR.KKK.EQ.49)KKK=KKK-1
          IF(KK.EQ.4.OR.KK.EQ.8.OR.KK.EQ.12.OR.KK.EQ.16)GO TO 701
          IF(KK.EQ.20.OR.KK.EQ.24)GO TO 701
          GO TO 702
   701    DO 16 MEME=1,KK
    16    CALL OFF(IP+4+MEME)
   702    CALL ON(IP+3)
          CALL MENUH(IT,2000+IP,2001+IP)
          CALL OFF(IP+3)
          GO TO (20,100),IT
    20    CALL TRAK(XX,XX)
          IF(II.EQ.4)II=0
          KK=KK+1
          II=II+1
          WRITE(5,11)II
    11    FORMAT(1X,'POSITION TRAK. OBJ. TO DEFINE SEC. NUMBER  ',I1)
          IF(IFL.EQ.2)WRITE(5,12)
    12    FORMAT(' YOU SHOULD DEFINE 6 POINTS IN THIS SECTION THE FIRST
          IF(IFL.EQ.2)WRITE(5,14)
    14    FORMAT(' TWO WILL DEFINE THE AXIS OF REVOLUTION')
          READ(5,21)M
    21    FORMAT(A2)
    30    CALL LPEN(IH,IT1)
          IF(IH.EQ.0.OR.IT1.LT.101.OR.IT1.GT.103)GO TO 30
          CALL GRID(GRD,GRD)
          CALL TRAKXY(X0,Y0)
          IT1=IT1-100
          GO TO (100,200,300),IT1
   100    CALL APNT(X0,Y0)
          CALL SUBP(IP+4+KK)
          CALL OFF(IP+4+KK)
          CALL VECT(-XX,-XX,,II)
          CALL VECT(0.0,XX,,II)
          CALL VECT(XX,XX,,II)
          CALL VECT(0.0,-XX,,II)
          CALL ESUB
          GO TO 400
   200    CALL APNT(X0,Y0)
          CALL SUBP(IP+4+KK)
          CALL OFF(IP+4+KK)
          CALL VECT(-XX,-XX,,II)
```

```
           CALL VECT(XX,0.0,,II)
           CALL VECT(XX,XX,,II)
           CALL VECT(-XX,0.0,,II)
           CALL ESUB
           GO TO 400
     300   CALL APNT(X0,Y0)
           CALL SUBP(IP+4+KK)
           CALL OFF(IP+4+KK)
           CALL VECT(XX,0.0,,II)
           CALL VECT(0.0,XX,,II)
           CALL VECT(-XX,0.0,,II)
           CALL VECT(0.0,-XX,,II)
           CALL ESUB
     400   CALL ON(IP+4+KK)
           I=1+K*NPL
           IF(II.EQ.2)I=1+4*NPL+K*NPL
           IF(II.EQ.3)I=1+8*NPL+K*NPL
           IF(II.EQ.4)I=1+12*NPL+K*NPL
     C
     1200  KKK=KKK+1
           CALL ON (IP+4)
           CALL ME UH (IT2,2900+IP,2901+IP)
           CALL OF  (IP+4)
           GO TO (600,700),IT2
     600   CALL GRID(GRD,GRD)
           CALL TRAKXY(X(I),Y(I))
           CALL SUBP(IPO+KKK+IRECL*16)
           CALL OFF(IPO+KKK+IRECL*16)
           CALL APNT(X(I),Y(I),1,4)
           CALL ESUB
           CALL ON(IPO+KKK+IRECL*16)
     601   GO TO (800,900,1000),IT1
     800   Z(I)=Y(I)-Y0+X0-X(I)
           X(I)=(X0-X(I))*SQRT(2.)
           Y(I)=X0 512.
           GO TO 1100
     900   XXX(I)=X(I)
           X(I)=(Y0-Y(I))*SQRT(2.)
           Y(I)=XXX(I)-X0+Y0-Y(I)
           Z(I)=Y0-512.
           GO TO 1100
     1000  Z(I)=Y(I)-Y0
           Y(I)=X(I)-X0
           X(I)=(512.-X0)*SQRT(2.)
     1100  I=I+1
           GO TO 1200
     130   DO 1 IOI=1,KK
     1     CALL ERAS(IP+4+IOI)
           CALL CMPRS
           II=0
           K=16*J
     4     CONTINUE
           RETURN
           END
```

```
            SUBROUTINE ROTATE(N,X,Y,Z,TN1,TN2,TN3,T1)
            DIMENSI N U(4,4),V(4,4),X(1),Y(1),Z(1),T(4,4)
C           SUBROUTINE ROTATE
C           FUNCTION:
C           THREE DIMENSIONAL ROTATION @ ANY ARBITRARY AXIS
C           TN1,TN2,TN3 ARE THE DIRECTION COSINES OF THE AXIS
C           OF ROTATION.   (I/P)
C
            DO 1 J= ,4
            DO 1 I=1,N
            U(I,J)=0.0
       1    V(I,J)= .0
            DO 2 I=1,N
            U(I,1)=X(I)            !N=NUMBER OF X,Y,Z TRIPLETS
            U(I,2)=Y(I)
            U(I,3)=Z(I)
       2    U(I,4)=1.0
            DO 3 I= ,4
            DO 3 J=1,4
       3    T(I,J)=0.0             !T IS THE TRANSFORMATION MATRIX
            T2=T1/5 .2957795       !T1 IS ROT. ANGEL IN DEGREES.
            T(4,4)=1.0
            T(1,1)= N1*TN1+(1-TN1*TN1)*COS(T2)
            T(1,2)=TN1*TN2*(1-COS(T2))+TN3*SIN(T2)
            T(1,3)=TN1*TN3*(1-COS(T2))-TN2*SIN(T2)
            T(2,1)= N1*TN2*(1-COS(T2))-TN3*SIN(T2)
            T(2,2)=TN2*TN2+(1-TN2*TN2)*COS(T2)
            T(2,3)=TN2*TN3*(1-COS(T2))+TN1*SIN(T2)
            T(3,1)=TN1*TN3*(1-COS(T2))+TN2*SIN(T2)
            T(3,2)=TN2*TN3*(1-COS(T2))-TN1*SIN(T2)
            T(3,3)=TN3*TN3+(1-TN3*TN3)*COS(T2)
            CALL MU(U,T,V,4,4,4)
            DO 4 I= ,N
            X(I)=V(I,1)
            Y(I)=V(I,2)
            Z(I)=V(I,3)
       4    CONTINUE
            RETURN
            END
```

```
              SUBROUTINE SMOOTH(BXX,BYY,BZZ,NUM,IET,BX2,BY2,BZ2,IFOL)
              DIMENSION BX1(16),BY1(16),BZ1(16),BY2(16),BY2(16),
             *  BZ2(16),BXX(1),BYY(1),BZZ(1),IET(2),FBX1(4,4)
             * ,FBY1(4,4),FBZ1(4,4),FBX2(4,4),FBY2(4,4),FBZ2(4,4)
              COMMON/DFILE/IBUF(1)
              COMMON/ERS/IERAS
        C
        C     SMOOTH.SUB
        C
        C     PURPOSE:
        C     THIS SUBROUTINE SMOOTHES TWO ADJACENT PATCHES ALONG THE
        C     COMMON BORDERS.
        C
        C     ARGUMENTS:
        C     BXX=ARRAY CONTAINING ELEMENTS OF BX VECTOR BEFORE SMOOTHING
        C     BYY= ,,      ,,         ,,       , BY    ,    ,,     ,,
        C     BZZ= ,,      ,,         ,,       , BZ    ,,   ,,     ,,
        C     NUM=NUMBER OF DEFINED PATCHES.
        C     IET=ARR Y CONTAINING THE TAGS OF THE TWO PATCHES TO BE SMOOT
        C     BX2=ARRAY CONTAINING ELEMENTS OF BX VECTOR AFTER SMOOTHING
        C     BY2= ,,      ,,         ,,       , BY    ,,   ,,      ,,
        C     BZ2= ,,      ,,         ,,       , BZ    ,,      ,,    ,,
        C     IFOL=1      SMOOTHING ALONG LONGITUDINAL BORDERS       (O/P)
        C      . =2    ,,          ,,   TRANSVERSE BORDERS           (O/P)
        C
              DO 1 I= ,2
              CALL IDNTFY(NUM,ITO)
        1     IET(I)=ITO
              DO 10 I=1,16
              BX1(I)=BXX(I+(IET(1)-1)*16)
              BY1(I)=BYY(I+(IET(1)-1)*16)
        10    BZ1(I)=BZZ(I+(IET(1)-1)*16)
              DO 20 I=1,16
              BX2(I)= XX(I+(IET(2)-1)*16)
              BY2(I)=BYY(I+(IET(2)-1)*16)
        20    BZ2(I)=BZZ(I+(IET(2)-1)*16)
              CALL GETB(BX1,BY1,BZ1,FBX1,FBY1,FBZ1)
              CALL GE B(BX2,BY2,BZ2,FBX2,FBY2,FBZ2)
              WRITE(5,600)
        600   FORMAT(1X,'IF YOU WANT TO SMOOTH ALONG LONG. BORDER TYPE 1':
              WRITE(5 700)
        700   FORMAT(1X,'ALONG TRANS. BORDER TYPE 2')
              READ(5, 00)IFOL
        800   FORMAT(I2)
              IF(IFOL.EQ.1)GO TO 900
              DO 50 I=1,4
              FBX2(I,1)=FBX1(I,2)
              FBY2(I,1)=FBY1(I,2)
        50    FBZ2(I,1)=FBZ1(I,2)
              DO 60 I=1,4
              FBX2(I,3)=FBX1(I,4)
              FBY2(I,3)=FBY1(I,4)
        60    FBZ2(I,3)=FBZ1(I,4)
              GO TO 1 00
```

```
0035    900  DO 55 I 1,4
0036         FBX2(1,I)=FBX1(2,I)
0037         FBY2(1,I)=FBY1(2,I)
0038    55   FBZ2(1,I)=FBZ1(2,I)
0039         DO 65 I 1,4
0040         FBX2(3,I)=FBX1(4,I)
0041         FBY2(3,I)=FBY1(4,I)
0042    65   FBZ2(3,I)=FBZ1(4,I)
0043    1000 CALL GETBX(FBX2,FBY2,FBZ2,BX2,BY2,BZ2)
0044         RETURN
0045         END
```

```
      SUBROUTINE BLEND(BX,BBX,BY,BBY,BZ,BBZ,FBX,FBY,FBZ)
      DIMENSION BX(4,4),BBX(4,4),BY(4,4),BBY(4,4),BZ(4,4)
      DIMENSI N BBZ(4,4),FBX(4,4),FBY(4,4),FBZ(4,4)
C
C     SUBROUTINE BLEND
C     FUNCTION:
C     THIS SUBROUTINE BLENDS TWO NON ADJACENT BATCHES
C
C     INPUT VARIABLES:
C     BX,BY,BZ ARE ELEMENTS OF THE BOUNDARY MATRICES OF THE
C     FIRST PATCH TO BE BLENDED WITH THE SECOND PATCH.
C     BBX,BBY,BBZ ARE ELEMENTS OF THE BOUNDARY MATRICES OF THE
C     SECOND PATCH.
C     OUTPUT VARIABLES:
C     FBX,FBY,FBZ ARE ELEMENTS OF THE INBETWEEN(GENRATED)PATCH
C
      WRITE(5,30)
   30 FORMAT(1X,'IF YOU WANT BLENDING ALONG LONG DIRECTION ENTER
     *  2 ,TRANS DIRECTION ENTER 1')
      READ(5,40)IB
   40 FORMAT(I2)
      IF(IB.EQ.1)GO TO 10
      DO 1 I=1,4
      FBX(1,I)=BX(2,I)
      FBY(1,I)=BY(2,I)
    1 FBZ(1,I)=BZ(2,I)
C
      DO 2 I=1,4
      FBX(2,I)=BBX(1,I)
      FBY(2,I =BBY(1,I)
    2 FBZ(2,I)=BBZ(1,I)
C
      DO 3 I= ,4
      FBX(3,I =BX(4,I)
      FBY(3,I)=BY(4,I)
    3 FBZ(3,I)=BZ(4,I)
C
      DO 4 I=1,4
      FBX(4,I)=BBX(3,I)
      FBY(4,I)=BBY(3,I)
    4 FBZ(4,I)=BBZ(3,I)
C
      GO TO 60
   10 DO 11 I 1,4
      FBX(I,1)=BX(I,2)
      FBY(I,1)=BY(I,2)
   11 FBZ(I,1)=BZ(I,2)
C
      DO 21 I=1,4
      FBX(I,2)=BBX(I,1)
      FBY(I,2)=BBY(I,1)
   21 FBZ(I,2)=BBZ(I,1)
C
      DO 31 I=1,4
```

```
      FBX(I,3)=BX(I,4)
      FBY(I,3)=BY(I,4)
   31 FBZ(I,3)=BZ(I,4)
C
      DO 41 I=1,4
      FBX(I,4)=BBX(I,3)
      FBY(I,4)=BBY(I,3)
   41 FBZ(I,4)=BBZ(I,3)
C
   60 RETURN
      END
```

```
      SUBROUTINE OBSAVE(X,Y,Z)
      DIMENSION X(1),Y(1),Z(1)
      COMMON/CFILE/IBUF(1)
      COMMON/SUPER/GRD
      LOGICAL 1 FILE1(15),FILE2(15)
C
C     SAVE THE DISPLAY
C
      CALL INFILE(FILE1,FILE2,NPATCH)
      CALL STOP
      CALL ASSIGN(12,FILE2,2)
      DO 1 I=1,NPATCH*16
    1 WRITE(12,*)X(I),Y(I),Z(I)          !WRITE X,Y,Z COORD. IN DATA
      CALL CLOSE(12)                     !FILE (FILE2.DAT)
      CALL SAVE(FILE1)
      CALL CONT
      CALL LPEN(IH,IT)
      RETURN
      END
```

```
              SUBROUTINE GETSHP(X,Y,Z)
              DIMENSION X(1),Y(1),Z(1)
              COMMON/DFILE/IBUF(1)
              COMMON/RECALL/IRECL
              LOGICAL*1 FILE1(15),FILE2(15)
        C
        C     SUBROUTINE GETSHP
        C     FUNCTION:
        C     RECALL A PREVIOUSLY SAVED DISPLAY.
        C
              CALL INFILE(FILE1,FILE2,NPATCH)
              CALL STOP
              CALL ASSIGN(10,FILE2,0)
              DO 1 I=1,NPATCH*16
         1    READ(10,*)X(I),Y(I),Z(I)
              CALL CLOSE(10)
              CALL INIT
              CALL RSTR(FILE1)
              CALL CONT
              CALL LPEN(IH,IT)
              IRECL=IRECL+NPATCH
              DO 10 I=1,10
         10   CALL ERAS(100+I)
              CALL CMPRS
              RETURN
              END
```

```
      SUBROUTINE INTJOL(BX,BY,BZ,X,Y,Z)
      DIMENSION U(16),W(16),X(1),Y(1),Z(1)
     * ,BX(1),BY(1),BZ(1),DELX(15),DELY(15)
      COMMON/DFILE/IBUF(1)
      COMMON/SUPER/GRD,IP,IPC
      COMMON/SEC/ISEC
C
C     SUBROUTINE INTJOL
C     FUNCTION:
C     THIS SUBROUTINE DRAWS THE RESULTANT CONTINUOUS CURVE
C     FROM THE INTERSECTION OF A PLANE AND A PATCH
C
C     INPUT VARIABLES:
C     BX,BY,BZ ARE X,Y,Z BOUNDARY VECTORS DEFINING THE PATCH
C     OUTPUT VARIABLES:
C     X,Y,Z C ORDINATES OF THE CURVE OF INTERSECTION
C
      C=1024.
      CALL APNT(C/8.,5.*C/8.,,-4)
      CALL SUBP(6140+ISEC)
      CALL OFF(6140+ISEC)
      CALL LVECT(.25*C,0.)
      CALL LVECT(0.,.25*C)
      CALL LVECT(-.25*C,0.)
      CALL LVECT(0.,-.25*C)
      CALL APNT(.1*C,4.5*C/8.,,-4)
      CALL TEXT('0,0')
      CALL APNT(.3*C,4.5*C/8.,,-4)
      CALL TEXT('1,0')
      CALL APNT(.3*C,.9*C,,-4)
      CALL TEXT('1,1')
      CALL APNT(.1*C,.9*C,,-4)
      CALL TEXT('0,1')
      CALL ESUB
      CALL ON 6140+ISEC)
C
C     POSITION TRACKING OBJECT ON THE POINTS OF INTRSECTION
C
      CALL TRAK(512.,512.)
      KEMO=0
   10 WRITE(5,11)
   11 FORMAT(1X,'POSITION TRAC. OBJ.,TYPE<CR> WHEN DONE')
      READ(5,20)I
   20 FORMAT(A2)
      IF(KEMO.GT.0)GO TO 30
      CALL TRAKXY(XO,YO)
      KEMO=KEMO+1
      GO TO 10
   30 CALL TRAKXY(XO1,YO1)
C
C     CALCULATE(U1,W1),(U16,W16)
C
      CALL SUBP(6141+ISEC)
      CALL APNT(XO,YO)
```

```
0038            CALL LVECT((X01-X0),(Y01-Y0))
0039            CALL ESUB
0040            U(1)=(X0-(C/8.))/(.25*C)
0041            W(1)=(Y -(5.*C/8.))/(.25*C)
0042            U(16)=(X01-(C/8.))/(.25*C)
0043            W(16)=(Y01-(5.*C/8.))/(.25*C)
0044            DELTAU=(U(16)-U(1))/15.
0045            DELTAW=(W(16)-W(1))/15.
0046            DO 40 I=2,15
0047            U(I)=U(I-1)+DELTAU
0048      40    W(I)=W(I-1)+DELTAW
          C
          C     DRAW THE INTERSECTION CURVE
          C
0049            CALL POINTS(U,W,BX,X,16)
0050            CALL POINTS(U,W,EY,Y,16)
0051            CALL POINTS(U,W,EZ,Z,16)
0052            DO 31 I=1,16
0053            X(I)=X(I)*SQRT(.5)
0054            Y(I)=Y(I)-X(I)+512.
0055            Z(I)=Z(I)-X(I)+512.
0056      31    CONTINUE
0057            DO 33 I=1,15
0058            DELX(I)=Y(I+1)-Y(I)
0059      33    DELY(I)=Z(I+1)-Z(I)
          C
0060            CALL SUBP(6142+ISEC)
0061            DO 34 I=1,15
0062            CALL APNT(Y(I),Z(I),,-4)
0063      34    CALL VECT(DELX(I),DELY(I),,3)
0064            CALL ESUB
0065            RETURN
0066            END
```

```
0001            SUBROUTINE TEST1(X,Y,Z,BX,BY,BZ)
0002            DIMENSION L(24),X(1),Y(1),Z(1),U(16),W(16),C(16,16)
               *    ,BX(1),BY(1),BZ(1),C1(16,16),C2(16,16),C3(16,16)
0003            REAL L
          C     SUBROUTINE TEST1
          C     FUNCTION:
          C     THIS ROUTINE CALCULATES BX,BY,BZ BOUNDARY VECTORS OF
          C     A PATCH DEFINED BY 16 X,Y,Z COORDINATES.
          C     INPUT VARIABLES:
          C     X,Y,Z COORDINATES DEFINING THE PATCH.
          C     OUTPUT VARIABLES:
          C     BX,BY,BZ BOUNDARY VECTORS DEFINING THE PATCH.
          C
          C     GET THE LENGTH BETWEEN EACH SUCCESSIVE POINTS
0004            N=16
          C
0005            CALL LENGTH(L,X,Y,Z)
          C     GET THE RATIO OF THE LENGTHES W.R.T. U VALUES
0006            TL1=L(1)+L(2)+L(3)
0007            TL2=L(4)+L(5)+L(6)
0008            TL3=L(7)+L(8)+L(9)
0009            TL4=L(10)+L(11)+L(12)
          C     CALCULATE THE U VALUES
0010            U(1)=0.0
0011            U(5)=0.0
0012            U(9)=0.0
0013            U(13)=0.0
0014            U(4)=1.
0015            U(8)=1.
0016            U(12)=1.
0017            U(16)=1.
0018            U(2)=L(1)/TL1
0019            U(3)=(L(1)+L(2))/TL1
0020            U(6)=L(4)/TL2
0021            U(7)=(L(4)+L(5))/TL2
0022            U(10)=L(7)/TL3
0023            U(11)=(L(7)+L(8))/TL3
0024            U(14)=L(10)/TL4
0025            U(15)=(L(10)+L(11))/TL4
          C     W.R.T. W VALUES
          C
0026            TL5=L(13)+L(14)+L(15)
0027            TL6=L(16)+L(17)+L(18)
0028            TL7=L(19)+L(20)+L(21)
0029            TL8=L(22)+L(23)+L(24)
          C
          C     CALCULATE THE W VALUES
          C
0030            W(1)=0.0
0031            W(2)=0.
0032            W(3)=0.
0033            W(4)=0.0
0034            W(13)=1.
0035            W(14)=1.
```

```
0036          W(15)=1.
0037          W(16)=1.
0038          W(5)=L(13)/TL5
0039          W(9)=(L(13)+L(14))/TL5
0040          W(6)=L(16)/TL6
0041          W(10)=(L(16)+L(17))/TL6
0042          W(7)=L(19)/TL7
0043          W(11)=(L(19)+L(20))/TL7
0044          W(8)=L(22)/TL8
0045          W(12)=( (22)+L(23))/TL8
        C
        C     CONSTRUCT THE C MATRIX
        C
0046          CALL CMATRX(U,W,C,N)
0047          DO 32 I=1,N
0048          DO 32 J=1,N
0049     32   C1(I,J)=C(I,J)
        C
        C     CONSTRUCT THE B VECTORS
        C
0050          DO 9 I= ,N
0051          BX(I)=X(I)
0052          BY(I)=Y(I)
0053      9   BZ(I)=Z(I)
        C
0054          DO 10 J=1,N
0055          DO 10 I=1,N
0056          C2(I,J)=C1(I,J)
0057          C3(I,J)=C1(I,J)
0058     10   CONTINUE
        C     SOLVE THE SYSTEM OF SIMULTANEOUS LINEAR EQUATIONS
        C
0059          CALL SOLVE(BX,C1,N,1,.000001,KS)
0060          IF(KS.NE.0)STOP
0062          CALL SOLVE(BY,C2,N,1,.000001,KS)
0063          IF(KS.NE.0)STOP
0065          CALL SOLVE(BZ,C3,N,1,.000001,KS)
0066          IF(KS.NE.0)STOP
        C
0068          RETURN
0069          END
```

```
      SUBROUTINE DRWJOL(BX,BY,BZ,ITOTO,MON,NPL,X,Y,Z)
      DIMENSION BX(1),BY(1),BZ(1),X(1),Y(1),Z(1)
      DIMENSION DELTAX(15),DELTAY(15),TOT(6),W(16),U(16)
     * ,UL(16),WW(16),IDELX(15),IDELY(15)
      COMMON/DFILE/IBUF(1)
      COMMON/SUPER/GRD,IP,IPO
C
C     DRWJOL.SUB
C
C     PURPOSE:
C     THIS SUBROUTINE DRAWS THE PATCH
C
C     ARGUMENTS:
C
C     BX    =ARRAY CONTAINING BX VECTOR OF THE PATCH
C     BY    = ,,          ,,        BY    ,,   ,   ,    ,,
C     BZ    =  ,          ,,        BZ    ,,   ,   ,    ,,
C     ITOTO=TAG OF THE SUBPICTURE CONTAINING PATCH      (I/P)
C     MON   =<1  DRWSHP DRAWS LINES OF CONST. U,W PARAMETERS ONLY
C          =>2   ,,     ,,    ,,    ,   ,,      ,       ,,  AND PLOTS
C            NEW CONTROL POINTS                          (I/P)
C       NPL   =NUBER OF PATCHES PER PLANE                    (I/P)
C
      ION=0
      K=0
C
      TOT(1)=0.0
      TOT(2)=.2
      TOT(3)=.4
      TOT(4)=.6
      TOT(5)= 8
      TOT(6)=1.0
C
      MEM=0
      MEM1=0
      DO 99 KOK=1,6
      T=0.0
      DO 40 I=1,16
      W(I)=TOT(KOK)
      U(I)=T
   40 T=T+1./15.
  100 CALL POINTS(U,W,BX,X,16)
      CALL POINTS(U,W,BY,Y,16)
      CALL POINTS(U,W,BZ,Z,16)
      DO 90 I=1,16
      X(I)=X(I)/SQRT(2.)
      Y(I)=Y(I)-X(I)+512.
   90 Z(I)=Z(I)-X(I)+512.
      DO 91 I=[illegible]15
      DELTAX(I)=Y(I+1)-Y(I)
      DELTAY(I)=Z(I+1)-Z(I)
   91 CONTINUE
      CALL APNT(Y(1),Z(1),,-4)
      DO 92 I=1,15
```

```
0035           DELTAX(I)=DELTAX(I)+DIFX
0036           DELTAY(I)=DELTAY(I)+DIFY
0037           IDELX(I)=INT(DELTAX(I))
0038           IDELY(I)=INT(DELTAY(I))
0039           CALL VECT(DELTAX(I),DELTAY(I),1)
0040           DIFX=DELTAX(I)-FLOAT(IDELX(I))
0041     92    DIFY=DELTAY(I)-FLOAT(IDELY(I))
0042           IF(MEM.GT.1)GO TO 999
0044           DO 101 I=1,16
0045           MEM=MEM+1
0046           MEM1=MEM1+1
0047           W(I)=U(I)
0048     101   U(I)=TOT(KOK)
0049           GO TO 100
0050     999   MEM=0
0051     99    CONTINUE
0052           IF(MON.NE.2)GO TO 1000
0054           WW(1)=0.0
0055           WW(2)=.3333
0056           WW(3)=.6666
0057           WW(4)=1.0
0058           DO 10 I=1,4
0059           UU(I)=WW(I)
0060           UU(I+4)=WW(I)
0061           UU(I+8)=WW(I)
0062           UU(I+12)=WW(I)
0063           WW(I)=0.0
0064           WW(I+4)=.3333
0065           WW(I+8)=.6666
0066     10    WW(I+12)=1.0
0067           CALL POINTS(UU,WW,BX,X,16)
0068           CALL POINTS(UU,WW,BY,Y,16)
0069           CALL POINTS(UU,WW,BZ,Z,16)
0070           DO 20 I=1,16
0071           X(I)=X(I)/SQRT(2.)
0072           Y(I)=Y(I)-X(I)+512.
0073     20    Z(I)=Z(I)-X(I)+512.
0074           IPOPO=(ITOTO-1)*16+406
0075           IF(NPL.GT.1)GO TO 1001
0077           DO 50 I=1,16
0078           CALL SUBP(IPOPO+I)
0079           CALL APNT(Y(I),Z(I),1)
0080     50    CALL ESUB
0081           GO TO 1000
0082     1001  DO 25 J=1,4
0083           DO 24 I=1,4
0084           CALL SUBP(406+I+ION+(ITOTO-1)*4)
0085           CALL APNT(Y(I+K),Z(I+K),1)
0086     24    CALL ESUB
0087           K=K+4
0088     25    ION=ION+8
0089     1000  RETURN
0090           END
```

```
      SUBROUTINE GETB(BX2,BY2,BZ2,FBX,FBY,FBZ)
      DIMENSION BX2(1),BY2(1),BZ2(1),FBX(4,4),FBY(4,4),FBZ(4,4)
     *  ,BM(4,4),BMT(4,4),FFX(4,4),FFY(4,4),FFZ(4,4)
C
C     SUBROUTINE GETB
C     FUNCTION:
C     THIS SUBROUTINE CALCULATES THE 'B' MATRIX FROM THE 'S'
C                                     T
C     VECTOR    S=(BM)(B)(BM)    (EQUATION OF THE PARAMETRIC PC
C     INPUT VARIABLES:
C     BX2,BY2,BZ2 ARE ELEMENTS OF THE BOUNDARY VECTORS 'S'
C     OUTPUT VARIABLES:
C     FBX,FBY,FBZ ARE ELEMENTS OF THE BOUNDARY MATRICES B
C
C     BM IS A COSTANT MATRIX, BMT IS THE TRANSPOSE OF BM MATRIX
C
      K=1
      DO 10 I=1,4
      DO 10 J=1,4
      FBX(I,J)=BX2(K)
      FBY(I,J)=BY2(K)
      FBZ(I,J)=BZ2(K)
   10 K=K+1
C
C                             -1
C     DEFINE THE BM AND BM   MATRICIES
C
      BM(1,4)=1.
      BM(2,1)=1.
      BM(2,2)=1.
      BM(2,3)=1.
      BM(2,4)=1.
      BM(3,3)=1.
      BM(4,1)=3.
      BM(4,2)=2.
      BM(4,3)=1.
      BMT(1,2)=1.
      BMT(1,4)=3.
      BMT(2,2)=1.
      BMT(2,4)=2.
      BMT(3,2)=1.
      BMT(3,3)=1.
      BMT(3,4)=1.
      BMT(4,1)=1.
      BMT(4,2)=1.
C
C     GET THE B MATRICES(TENSOR) , CALL IT FBX,FBY,FBZ
C
      CALL MU(BM,FBX,FFX,4,4,4)
      CALL MU(FFX,BMT,FBX,4,4,4)
C
      CALL MU(BM,FBY,FFY,4,4,4)
      CALL MU(FFY,BMT,FBY,4,4,4)
C
```

227

```
0232          CALL MU(BM,FEZ,FFZ,4,4,4)
0233          CALL MU(FFZ,BMT,FBZ,4,4,4)
      C
0234          RETURN
0235          END
```

```
0001          SUBROUTINE POINTS(U,W,B,X,N)
0002          DIMENSION U(N),W(N),B(16),X(N)
        C
        C     SUBROUTINE POINTS
        C     FUNCTION:
        C     CALCULATES THE X,OR Y,OR Z COORDINATES OF N POINTS LYING ON
        C     THE PATCH SURFACE.
        C     U,W ARE INPUT ARRAYS DEFINING THE PARAMETRIC VALUES OF
        C     THE POINTS.
        C     B IS THE INPUT BOUNDARY MATRIX DEFINING THE PATCH.
        C     N=INPUT NUMBER OF POINTS.
        C     X=OUTPUT ARRAY CONTAINING COORDINATES OF POINTS.
        C
0003          DO 1 I=1,N
0004          X(I)=(U(I)**3)*(W(I)**3)*B(1)+(U(I)**3)*(W(I)**2)
             *  *B(2)+(U(I)**3)*(W(I)*B(3))+(U(I)**3)*B(4)+(U(I)
             *  **2)*(W(I)**3)*B(5)+(U(I)**2)*(W(I)**2)*B(6)+(U(I
             *  )**2)*W(I)*B(7)+(U(I)**2)*B(8)+(U(I)*W(I))**3)*B(
             *              9)+(U(I)*W(I)**2)*B(10)+U(I)*W(I)*B(11)
             *  +U(I)*B(12)+(W(I)**3)*B(13)+(W(I)**2)*B(14)
             *  +W(I)*B(15)+B(16)
0005     1    CONTINUE
0006          RETURN
0007          END
```

```
      SUBROUTINE GETBX(FBX,FBY,FBZ,BX,BY,BZ)
C
      DIMENSION BM(4,4),BMT(4,4),FBX(4,4),FBY(4,4),FBZ(4,4)
     * ,BX(16),BY(16),BZ(16),FFX(4,4),FFY(4,4),FFZ(4,4)
C
C     SUBROUTINE GETBX
C     FUNCTION:
C     THIS SUBROUTINE CALCULATES THE 'BX' ,'BY','BZ'
C     BOUNDARY VECTORS FROM THE FBX,FBY,FBZ (B MATRICIES)
C
C     BM IS A CONSTANT MATRIX,BMT IS ITS TRANSPOSE
C
C                                 -1
C     DEFINE THE BM AND BM  MATRICIES
C
      BM(1,1)=2.
      BM(1,2)=-2.
      BM(1,3)=1.
      BM(1,4)=1.
      BM(2,1)=-3.
      BM(2,2)=3.
      BM(2,3)=-2.
      BM(2,4)=-1.
      BM(3,3)=1.
      BM(4,1)=1.
      BMT(1,1)=2.
      BMT(2,2)=3.
      BMT(3,3)=1.
      BMT(1,2)=-3.
      BMT(3,1)=1.
      BMT(1,4)=1.
      BMT(4,1)=1.
      BMT(4,2)=-1.
      BMT(3,2)=-2.
      BMT(2,1)=-2.
C
      CALL MU(BM,FBX,FFX,4,4,4)
      CALL MU(FFX,BMT,FBX,4,4,4)
C
      CALL MU(BM,FBY,FFY,4,4,4)
      CALL MU(FFY,BMT,FBY,4,4,4)
C
      CALL MU(BM,FBZ,FFZ,4,4,4)
      CALL MU(FFZ,BMT,FBZ,4,4,4)
C
      K=1
      DO 10 I=1,4
      DO 10 J=1,4
      BX(K)=FBX(I,J)
      BY(K)=FBY(I,J)
      BZ(K)=FBZ(I,J)
   10 K=K+1
      RETURN
      END
```

```
              SUBROUTINE IDNTFY(NUM,ITO)
              COMMON/DFILE/IBUF(1)
              COMMON/SUPER/GRD
              COMMON/ERS/IERAS
      C
      C       SUBROUTINE IDNTFY
      C       FUNCTION:
      C       THIS SUBROUTINE IDENTIFY THE PATCH
      C       NUM=NUMBER OF DEFINED PATCHES(SUB-PICTURES)  (I/P)
      C       ITO=TAG OF THE IDENTIFIED PATCH               (O/P)
      C
              CALL ERAS(IERAS)
              CALL CMPRS
              CALL SUBP(IERAS)
              CALL APNT(0.0,0.0,,-4)
              CALL ESUB
      C
      C       TURN LP SENSETIVTY ON FOR ALL DEFINED SUBPICTURES
      C
              DO 10 I=1,NUM
              CALL POINTR(9,I)
         10   CALL SENSE(9,1)
              CALL TRAK(512.,512.)
              TYPE *,' IDENTIFY THE  PATCH'
              WRITE(5,1)
         1      FORMAT(' TO DO SO ,POSITION THE TRACKING OBJECT AT ANY PAR
              WRITE(5,40)
         40     FORMAT('OF THE PATCH AND TYPE <CR> WHEN DONE')
              READ(5,2)M
         2    FORMAT(A2)
              CALL MENUH(ITO,1,NUM)
      C
      C       TURN LP SENSETIVTY OFF FOR ALL DEFINED SUBPICTURES
      C
              DO 30 I=1,NUM
              CALL POINTR(9,I)
         30   CALL SENSE(9,-1)
              RETURN
              END
```

```
      SUBROUTINE INFILE(FILE1,FILE2,NPATCH)
      LOGICAL*1 FILE1(15),FILE2(15),DSP(5),DAT(5)
      DATA DSP/'.','D','S','P',0/, DAT/'.','D','A','T',0/
C
C     SUBROUTINE INFILE
C     FUNCTION:
C     INPUT FILE NAME
C     NPATCH=NUMBER OF PATCHES TO BE SAVED OR RECALLED
C     FILE1 IS THE DISPLAY FILE NAME
C     FILE2 IS THE DATA FILE NAME
C
      WRITE(5,30)
 30   FORMAT(' ENTER NUMBER OF PATCHES ;OR SECTIONS;')
      READ(5,3000)NPATCH
 3000 FORMAT(I2)
 1    WRITE(5,11)
 11   FORMAT(' TYPE FILENAME:')
      READ(5,22)N,(FILE1(I),I=1,N)
      IF(N.EQ.0)GO TO 1
 22   FORMAT(Q,6 A1)
      DO 100 I=1,N
 100  FILE2(I)=FILE1(I)
      DO 200 I=1,5
      FILE1(I+N)=DSP(I)
 200  FILE2(I+N)=DAT(I)
      RETURN
      END
```

```
      SUBROUTINE CMATRX(U,W,C,N)
      DIMENSION U(1),W(1),C(N,N)
C
C     SUBROUTINE CMATRX
C     FUNCTION:
C     THIS SUBROUTINE CALCULATES THE C MATRIX(CONTAINES U,W
C     PARAMETRIC PROUDUCTS)USING THE PARAMETRIC PC EXPANDED EQUATI
       C      U,W PARAMETRIC VALUES   (I/P)
C     N=NUMBER OF DATA PINTS (USUALLY 16)  (I/P)
C
      DO 8 I=1,N
      C(I,1)=(U(I)**3)*W(I)**3
      C(I,2)=(U(I)**3)*W(I)**2
      C(I,3)=(U(I)**3)*W(I)
      C(I,4)=U(I)**3
      C(I,5)=(U(I)**2)*W(I)**3
      C(I,6)=(U(I)**2)*W(I)**2
      C(I,7)=(U(I)**2)*W(I)
      C(I,8)=U(I)**2
      C(I,9)=U(I)*W(I)**3
      C(I,10)=U(I)*W(I)**2
      C(I,11)=U(I)*W(I)
      C(I,12)=U(I)
      C(I,13)=W(I)**3
      C(I,14)=W(I)**2
      C(I,15)=W(I)
      C(I,16)=1.
    8 CONTINUE
      RETURN
      END
```

```
0001            SUBROUTINE LENGTH(L,X,Y,Z)
0002            REAL L
0003            DIMENSION L(1),X(1),Y(1),Z(1)
          C
          C     SUBROUTINE LENGTH
          C     FUNCTION:
          C     THIS SUBROUTINE CALCULATES THE LENGTH BETWEEN
          C     EACH SUCCESSIVE POINTS
          C     INPUT VARIABLES:
          C     X,Y,Z COORDINATES OF INPUT POINTS
          C     OUTPUT VARIABLES:
          C     L IS AN ARRAY CONTAINING LENGTH BETWEEN EACH SUCCESSIVE POIN
          C
0004            DO 1 I=1,3
0005            L(I)=SQRT((X(I+1)-X(I))**2+(Y(I+1)-Y(I))**2+
               *  ((Z(I+1)-Z(I))**2))
0006       1    CONTINUE
0007            DO 2 I=1,3
0008            L(I+3)=SQRT((X(I+5)-X(I+4))**2+(Y(I+5)-Y(I+4))**2
               *  +(Z(I+5)-Z(I+4))**2)
0009       2    CONTINUE
0010            DO 3 I=1,3
0011            L(I+6)=SQRT((X(I+9)-X(I+8))**2+(Y(I+9)-Y(I+8))**2
               *  +(Z(I+9)-Z(I+8))**2)
0012       3    CONTINUE
0013            DO 4 I=1,3
0014            L(I+9)=SQRT((X(I+13)-X(I+12))**2+(Y(I+13)-Y(I+12))**2
               *  +(Z(I+13)-Z(I+12))**2)
0015       4    CONTINUE
0016            J=13
0017            DO 5 I=1,4
0018            L(J)=SQRT((X(I+4)-X(I))**2+(Y(I+4)-Y(I))**2
               *  +(Z(I+4)-Z(I))**2)
0019            J=J+3
0020       5    CONTINUE
0021            J=14
0022            DO 6 I=1,4
0023            L(J)=SQRT((X(I+8)-X(I+4))**2+(Y(I+8)-Y(I+4))**2
               *  +(Z(I+8)-Z(I+4))**2)
0024            J=J+3
0025       6    CONTINUE
0026            J=15
0027            DO 7 I=1,4
0028            L(J)=SQRT((X(I+12)-X(I+8))**2+(Y(I+12)-Y(I+8))**2+
               *  (Z(I+12)-Z(I+8))**2)
0029            J=J+3
0030       7    CONTINUE
0031            RETURN
0032            END
```

```
      SUBROUTINE SOLVE(R,A,M,N,EPS,IER)
      DIMENSION A(1),R(1)
C
C     SUBROUTINE SOLVE
C     FUNCTION:
C     THIS SUBROUTINE SOLVES A SYSTEM OF LINEAR EQUATIONS
C     USING GAUSS ELIMINATION.
C     R IS THE M BY N MATRIX OF THE RIGHT HAND SIDE(DESTOYED)
C     ON RETURN R CONTAINS THE SOLUTION OF THE EQUATIONS.
C     A IS THE M BY M COEFFICIENT MATRIX(DESTOYED)
C     M IS THE NUMBER OF EQUATIONS IN THE SYSTEM.
C     N IS THE NUMBER OF RIGHT HAND SIDE VECTORS.
C     EPS IS AN INPUT CONSTANT WHICH IS USED AS RELATIVE TOLERANCE
C     FOR TEST ON LOSS OF SIGNIFICANCE.
C     IER IS THE RESULTING ERROR PARAMETER CODED AS:
C     IER=0          NO ERROR
C     IER=-1 OR N    NO RESULT
C
C     METHOD:
C     SOLUTION IS DONE BY MEANS OF GAUSS ELIMINATION WITHE
C     COMPLETE PIVOTING.
C
      IF(M)23,23,1
    1 IER=0
      PIV=0.0
      MM=M*M
      NM=N*M
      DO 3 L=1,MM
      TB=ABS(A(L))
      IF(TB-PIV)3,3,2
    2 PIV=TB
      I=L
    3 CONTINUE
      TOL=EPS*PIV
      LST=1
      DO 17 K=1,M
      IF(PIV)23,23,4
    4 IF(IER)7,5,7
    5 IF(PIV-TOL)6,6,7
    6 IER=K-1
    7 PIVI=1./A(I)
      J=(I-1)/M
      I=I-J*M-K
      J=J+1-K
      DO 8 L=K,NM,M
      LL=L+I
      TB=PIVI*R(LL)
      R(LL)=R(L)
    8 R(L)=TB
      IF(K-M)9,18,18
    9 LEND=LST+M-K
      IF(J)12,12,10
   10 II=J*M
      DO 11 L=LST,LEND
```

```
       TB=A(L)
       LL=L+II
       A(L)=A(LL)
 11    A(LL)=TB
 12    DO 13 L=LST,MM,M
       LL=L+I
       TB=PIVI*A(LL)
       A(LL)=A(L)
 13    A(L)=TB
       A(LST)=J
       PIV=0.0
       LST=LST+1
       J=0
       DO 16 II=LST,LEND
       PIVI=-A(II)
       IST=II+M
       J=J+1
       DO 15 L=IST,MM,M
       LL=L-J
       A(L)=A(L)+PIVI*A(LL)
       TB=ABS(A(L))
       IF(TB-PIV)15,15,14
 14    PIV=TB
       I=L
 15    CONTINUE
       DO 16 L=K,NM,M
       LL=L+J
 16    R(LL)=R(LL)+PIVI*R(L)
 17    LST=LST+M
 18    IF(M-1)23,22,19
 19    IST=MM+M
       LST=M+1
       DO 21 I=2,M
       II=LST-I
       IST=IST-LST
       L=IST-M
       L=A(L)+.5
       DO 21 J=II,NM,M
       TB=R(J)
       LL=J
       DO 22 K=IST,MM,M
       LL=LL+1
 22    TB=TB-A(K)*R(LL)
       K=J+L
       R(J)=R(K)
 21    R(K)=TB
 22    RETURN
 23    IER=-1
       RETURN
       END
```

```
0001            SUBROUTINE MENUH(IT,M1,M2)
          C
          C     SUBROUTINE MENUH
          C     FUNCTION:
          C     THIS SUBROUTINE RETURNES THE TAG OF A HITTEN SUBPICTURE(IT)
          C     M1=TAG OF THE FIRST DEFINED SUBPICTURE
          C     M2=TAG OF THE LAST DEFINED SUBPICTURE
          C
0002       10   CALL LPEN(IH,IT)
0003            IF(IH.EQ.0.OR.IT.LT.M1.OR.IT.GT.M2)GO TO 10
0005            CALL POINTR(10,IT)
0006            CALL INTENS(10,8)
          C
          C     WAIT FOR A MENU HIT
          C
0007            X=2.
0008            DO 20 I=1,1500
0009       20   X=X/X*2.
          C
0010            CALL LPEN(IH,IX)
0011            CALL INTENS(10,4)
0012            IT=IT+1-M1
0013            RETURN
0014            END
```

ATRICIES.

S OF E.

```
      SUBROUTINE DATFIL(NPATCH,X,Y,Z)
      DIMENSION X(1),Y(1),Z(1)
C
C     SUBROUTINE DATFFIL
C     FUNCTION:
C     THIS SUBROUTINE RETURNES X,Y,Z COORDINATES OF A PATCH
C     OR A GROUP OF PATCHES TO THE MAIN PROGRAM USING DATA FILES
C     NPATCH=NUMBER OF PATCHES    (I/P)
C
C     DEFINE NUMBER OF DATA FILES(=NUMBER OF PATCHES)
C
      BYTE FILE(14)
      WRITE(5,1)
 1    FORMAT('   PLEASE ENTER THE NUMBER OF DATA FILES')
      READ(5,2)NPATCH
 2    FORMAT(I2)
C
      K=0
      DO 10 I=1,NPATCH
      DO 20 J=1,14
 20   FILE(J)=' '
      TYPE 4,I
 4    FORMAT(' ENTER THE NAME OF FILE NUMBER  ',I2)
      ACCEPT 3,FILE
 3    FORMAT(14A1)
      J=I+9
      CALL ASSIGN(J,FILE,0)
      READ(J,*)(X(II+K),Y(II+K),Z(II+K),II=1,16)
      K=16*I
      CALL CLOSE(J)
 10   CONTINUE
      RETURN
      END
```

```
              SUBROUTINE KEYBRD(NPATCH,X,Y,Z)
              DIMENSION X(1),Y(1),Z(1)
      C
      C       SUBROUTINE KEYBRD
      C       FUNCTIO :
      C       THIS SUBROUTINE RETURNS THE X,Y,Z COORDINATES OF A PATCH
      C       OR A GROUP OF PATCHES TO THE MAIN PROGRAM VIA THE KEY BOARD
      C       NPATCH IS THE NUMBER OF PATCHES (I/P)
      C
              WRITE(5,1)
        1     FORMAT('  PLEASE ENTER THE NUMBER OF PATCHES')
              READ(5,2)NPATCH
        2     FORMAT(I2)
      C
      C       DEFINE X,Y,Z COORDINATES OF EACH PATCH
      C
              K=0
              DO 3 J=1,NPATCH
              WRITE(5 4)J
        4     FORMAT('  ENTER 16 X,Y,Z COORDINATES OF PATCH,NUMBER ',I2)
              READ(5,5)(X(I+K),Y(I+K),Z(I+K),I=1,16)
        5     FORMAT(3F10.4)
              K=16*J
        3     CONTINUE
              RETURN
              END
```

```
              SUBROUTINE INTREV(X,Y,Z)
              DIMENSION X(1),Y(1),Z(1),XXX(32),X0(2),Y0(2),Z0(2)
              COMMON/DFILE/IBUF(1)
              COMMON/RECALL/IRECL
              COMMON/SUPER/GRD,IP,IP0
        C
        C     SUBROUTINE INTREV
        C     FUNCTION:
        C     THIS SUBROUTINE RETURNES THE X,Y,Z COORDINATES OF A PATCH
        C     TO THE MAIN PROGRAM USING LIGHT PEN
        C
              K=0
        C
              XX=512.
              CALL SUBP(IP)
              CALL APNT(XX,XX,,-4)
              CALL VECT(XX,0.0)
              CALL APNT(970.,460.,,-4)
              CALL TEXT(' Y')
              CALL ESUB
              CALL SUBP(IP+1)
              CALL APNT(XX,XX,,-4)
              CALL VECT(0.0,XX)
              CALL APNT(505.,1000.,,-4)
              CALL TEXT(' Z')
              CALL ESUB
              CALL SUBP(IP+2)
              CALL APNT(XX,XX,,-4)
              CALL VECT(-XX,-XX)
              CALL APNT(50.,35.,,-4)
              CALL TEXT(' X')
              CALL ESUB
              II=0
              KK=0
              KKK=0
        C
              CALL SUBP(IP+4)
              CALL OFF (IP+4)
              CALL MENU(0.,850.,-100.,2900+IP,'POSITION','DONE')
              CALL ESUB
        C
              CALL SUBP(IP+3)
              CALL OFF(IP+3)
              CALL MENU(0.0,1000.,-100.,2000+IP,'DEFINE WORKING PLANE',
             * 'DONE')
              CALL ESUB
        C
        C
         700  CALL ON(IP+3)
              CALL MENUH(IT,2000+IP,2001+IP)
              CALL OFF(IP+3)
              GO TO (20,130),IT
         20   CALL TRAK(XX,XX)
              IF(II.EQ.4)II=0
```

```
      KK=KK+1
      II=II+1
      WRITE(5,11)II
11    FORMAT(1X,'POSITION TRAK. OBJ. TO DEFINE SEC. NUMBER  ',I1)
      WRITE(5,12)
12    FORMAT(' YOU SHOULD DEFINE 2 POINTS IN THIS SECTION THEY')
      WRITE(5,14)
14    FORMAT(' WILL DEFINE THE AXIS OF REVOLUTION')
      READ(5,21)M
21    FORMAT(A2)
30    CALL LPEN(IH,IT1)
      IF(IH.EQ.0.OR.IT1.LT.IP.OR.IT1.GT.IP+2)GO TO 30
      CALL GRID(GRD,GRD)
      CALL TRAKXY(X0,Y0)
      IT1=IT1-IP+1
      GO TO (100,200,300),IT1
100   CALL APNT(X0,Y0)
      CALL SUBP(IP+4+KK)
      CALL OFF(IP+4+KK)
      CALL VECT(-XX,-XX,,II)
      CALL VECT(0.0,XX,,II)
      CALL VECT(XX,XX,,II)
      CALL VECT(0.0,-XX,,II)
      CALL ESUB
      GO TO 400
200   CALL APNT(X0,Y0)
      CALL SUBP(IP+4+KK)
      CALL OFF(IP+4+KK)
      CALL VECT(-XX,-XX,,II)
      CALL VECT(XX,0.0,,II)
      CALL VECT(XX,XX,,II)
      CALL VECT(-XX,0.0,,II)
      CALL ESUB
      GO TO 400
300   CALL APNT(X0,Y0)
      CALL SUBP(IP+4+KK)
      CALL OFF(IP+4+KK)
      CALL VECT(XX,0.0,,II)
      CALL VECT(0.0,XX,,II)
      CALL VECT(-XX,0.0,,II)
      CALL VECT(0.0,-XX,,II)
      CALL ESUB
400   CALL ON(IP+4+KK)
      I=1+K
      IF(II.EQ.2)I=5+K
      IF(II.EQ.3)I=9+K
      IF(II.EQ.4)I=13+K
C
1000  KKK=KKK+1
      CALL ON (IP+4)
      CALL MENUH (IT2,2920+IP,2931+IP)
      CALL OFF (IP+4)
      GO TO (600,700),IT2
600   CALL GRID(GRD,GRD)
```

```
             CALL TRAKXY(X(I),Y(I))
             CALL SUBP(IPO+KKK+IRECL*16)
             CALL OFF(IPO+KKK+IRECL*16)
             CALL APNT(X(I),Y(I),1,4)
             CALL ESUB
             CALL ON(IPO+KKK+IRECL*16)
      601    GO TO (800,900,1000),IT1
      800    Z(I)=Y(I)-Y0+X0-X(I)
             X(I)=(X0-X(I))*SQRT(2.)
             Y(I)=X0-512.
             GO TO 1100
      900    XXX(I)=X(I)
             X(I)=(Y0-Y(I))*SQRT(2.)
             Y(I)=XXX(I)-X0+Y0-Y(I)
             Z(I)=Y0-512.
             GO TO 1100
      1000   Z(I)=Y(I)-Y0
             Y(I)=X(I)-X0
             X(I)=(512.-X0)*SQRT(2.)
      1100   I=I+1
             GO TO 1200
      130    DO 1 IOI=1,KK
      1      CALL ERAS(IP+4+IOI)
             CALL C PRS
             II=0
             K=16
             RETURN
             END
```

```
R LINK
RK2:SHIP2=RK2:SHIP2,RK2:GLIB,FORLIB//
RK2:SMOSHP/O:1
RK2:CBSAVE/O:1
RK2:DRWSHP/O:1
RK2:TEST1/O:1
RK2:SHIPS1/O:1
RK2:GETSHP/O:1
RK2:INTRSC/O:1
RK2:GEN/O:1
RK2:GETBX/O:2
RK2:IONSHP/O:2
RK2:GETB/O:2
RK2:INFILE/O:2
RK2:LENGTH/O:2
RK2:CMATRX/O:2
RK2:SOLVE/O:2
RK2:POINTS/O:2
RK2:MENUH/O:3
RK2:MU/O:3
//

GT OFF
RU SHIP2
```

```
      DIMENSION X(32),Y(32),Z(32),BX(16),BY(16),BZ(16)
      DIMENSION TEX(16),TEY(16),TEZ(16),BXO(64),BYO(64)
     * ,BZO(64),BX2(16),BY2(16),BZ2(16),IET(4),BXXX(16)
     * ,BZZZ(16),TEX1(32),TEY1(32),TEZ1(32),BYYY(16),XOY(32)
     * ,YOY(32),BXOX(16),BYOY(16),BZOZ(16),NUMBR(4)
      DIMENSION XREAL(128),YREAL(128),ZREAL(128),BXSPLT(16)
     * ,BYSPLT(16),BZSPLT(16)
      REAL L,M,LL1,LL2
      COMMON/DFILE/IBUF(3000)
      COMMON/RECALL/IRECL
      COMMON/SUPER/GRD,IP,IPO
      COMMON/SEC/ISEC
C
C     PROGRAM SHIP2.FOR
C
C       PURPOSE:
C     THIS PROGRAM AIDS IN THE PRELIMINARY DESIGN OF SHIPS HULLS
C
C       UTILIZATION:
C     START UP THE COMPUTER
C     TYPE RU:SHIP2   <CR>
C
C     ALY A. BADAWY
C     UNIV OF MCMASTER
C     MECH. ENGG.
C    11 JUNE 1979
C
      GRD=1.
      NDEF=0
      NUM=0
      IP=101
      IPO=426
      IYOA=1
      KEMO=0
      IRECL=0
      ISEC=0
C
C     GET THE MAJOR DIMENSIONS OF THE SHIP
C    -------------------------------------
C
      WRITE(5,345)
  345 FORMAT(' PROGRAM SHIP2 WILL AID IN THE DESIGN OF SHIP HULLS':
      WRITE(5,33)
   33 FORMAT(1X,'ENTER LENGTH,BREADTH,DRAFT,F,M,A,S')
      READ(5,44)L,B,D,F,M,A,S
   44 FORMAT(7F7.2)
C
C   - BEGIN DESIGNING THE SHIP HULL
C    ------------------------------
      WRITE(5,120)
  120 FORMAT( X,/,'  NOW YOU ARE READY TO DESIGN SHIP HULL')
C
C     SET UP DESIGN MENU
C    -------------------
```

```
0026            CALL INIT(3200)
0027            CALL SCAL(0.0,0.0,L+B,L+B)
0028            CALL SUBP(11000)
0029            CALL OFF(11000)
0030            CALL MENU(0.0,L+B,-L/8.,12010,'DESIGN MIDDLE BODY','DESIGN
               *  FORWARD BODY','DESIGN AFT BODY','DESIGN STERN','SMOOTH','SA
               *  ,'RECALL','REFLECT')
0031            CALL MENU(0.0,B,-B/5.,12013,'MODIFY','ERASE','INTRSC',
               *  'SPLIT','DONE')
0032            CALL ES B
0033      132   CALL ON(11002)
          C
          C     WAIT FOR A MENU HIT AND BRANCH TO SERVE IT
          C    ----------------------------------------
0034            CALL MENUH(IT,12010,12022)
0035            CALL OFF(11000)
          C
          C     MIDDLE BODY
          C    -----------
0036            IF(IT.EQ.1)LL1=M
0038            IF(IT.EQ.1)LL2=F
          C
          C     FORWARD BODY
          C    ------------
0040            IF(IT.EQ.2)LL1=F
0042            IF(IT.EQ.2)LL2=0.0
          C
          C     AFT BODY
          C    --------
0044            IF(IT.EQ.3)LL1=A
0046            IF(IT.EQ.3)LL2=F+M
          C
          C     STERN BODY
          C    ----------
0048            IF(IT.EQ.4)LL1=S
0050            IF(IT.EQ.4)LL2=F+M+A
          C
0052            IF(IT.EQ.5)GO TO 444                      !SMOOTH TWO PATCHES
0054            IF(IT.EQ.6)GO TO 544                      !SAVE DISPLAY
0056            IF(IT.EQ.7)GO TO 554                      !RECALL DISPLAY
0058            IF(IT.EQ.8)GO TO 655                      !REFLCT DISPLAY
0060            IF(IT.EQ.9)GO TO 755                      !MODIFY A PATCH
0062            IF(IT.EQ.10)GO TO 855                     !ERAS A PATCH
0064            IF(IT.EQ.11)GO TO 555                     !INTERSECT
0066            IF(IT.EQ.12)GO TO 955                 !SPLIT
0068            IF(IT.EQ.13)GO TO 999                 !EXIT
          C
          C     BEGIN DESIGNING THE SHIP
          C    ------------------------
          C
0070            CALL SHIPS1(X,Y,Z,NPATCH,NPL,LL1,LL2,L,B,D,XOX,YOY)
0071            IP=IP+40
0072            IPO=IPO-40
0073     5000   NDEF=NDEF+1
```

```
             NUMBR(NDEF)=NPATCH
      C
      C
      C      DRAW THE SURFACE
      C     -----------------
             IF(NPL.LT.2)GO TO 502
      C
             DO 221 NN=1,NPATCH
             DO 222 K=1,4
             K1=(K-1)*4*NPATCH+4*(NN-1)
             KK1=K1+NUM*16
             DO 222 J=1,4
             XREAL(KK1+J)=X(K1+J)
             YREAL(KK1+J)=Y(K1+J)
             ZREAL(KK1+J)=Z(K1+J)
             TEX(J+(K-1)*4)=X(K1+J)
             TEY(J+( -1)*4)=Y(K1+J)
       222   TEZ(J+(K-1)*4)=Z(K1+J)
             CALL TEST1(TEX,TEY,TEZ,BX,BY,BZ)
             DO 77 IENO=1,16
             BXO(IENO+KEMO)=BX(IENO)
             BYO(IENO+KEMO)=BY(IENO)
       77    BZO(IENO+KEMO)=BZ(IENO)
             KEMO=KEMO+16
             CALL SUBP(IYOA+IRECL)
             CALL DRWSHP(BX,BY,BZ,L,ITOTO,2,0)
             CALL ESUB
             IYOA=IYOA+1
       221   CONTINUE
             NUM=NUM+NUMBR(NDEF)
             GO TO 132
       502   J=NUM*16
             DO 11 I=1,16
             XREAL(I+J)=X(I)
             YREAL(I+J)=Y(I)
             ZREAL(I+J)=Z(I)
             TEX(I)=X(I)
             TEY(I)=Y(I)
       11    TEZ(I)=Z(I)
      C
       505   CALL TEST1(TEX,TEY,TEZ,BX,BY,BZ)
             DO 88 IENO=1,16
             BXO(IENO+KEMO)=BX(IENO)
             BYO(IENO+KEMO)=BY(IENO)
       88    BZO(IENO+KEMO)=BZ(IENO)
             KEMO=KEMO+16
             CALL SUBP(IYOA+IRECL)
             CALL DRWSHP(BX,BY,BZ,L,ITOTO,2,0)
             CALL ESUB
             IYOA=IYOA+1
       220   CONTINUE
             NUM=NUM NUMBR(NDEF)
             GO TO 132
      C
```

```
        C      SMOOTH TWO PATCHES AT A COMMON BORDER
        C      -------------------------------------------
        C
0122     444   CALL SMOSHP(BXO,BYO,BZO,NPATCH,IET,BX2,BY2,BZ2,L,IFOL)
0123           IF(IFOL.EQ.1)NPL=2
0125           IF(IFOL.EQ.2)NPL=1
0127           NPOO=IPO-40
0128           NPO=40
0129           NPLA=(IET(2)-1)*16
0130           DO 144 I=1,16
0131           BXO(I+NPLA)=BX2(I)
0132           BYO(I+NPLA)=BY2(I)
0133     144   BZO(I+NPLA)=BZ2(I)
0134           CALL ERAS(IET(2)+IRECL)
0135           CALL CMPRS
0136           IF(NPL.GT.1)GO TO 10
0138           DO 20 I=1,16
0139     20    CALL ERAS(NPOO+I)
0140           CALL CMPRS
0141           GO TO 22
0142     10    DO 25 I=1,4
0143           CALL ERAS(IPO-NPO+I+(IET(2)-1)*4)
0144           CALL ERAS(IPO-NPO+I+8+(IET(2)-1)*4)
0145           CALL ERAS(IPO-NPO+I+16+(IET(2)-1)*4)
0146     25    CALL ERAS(IPO-NPO+I+24+(IET(2)-1)*4)
0147           CALL CMPRS
0148     22    CALL SUBP(IET(2)+IRECL)
0149           CALL DRWSHP(BX2,BY2,BZ2,L,IET(2),2,NPL)
0150           CALL ESUB
0151           GO TO 130
        C
        C      SAVE THE DISPLAY AND CREATE A DATA FILE
        C      ----------------------------------------
        C
0152     544   CALL OBSAVE(XREAL,YREAL,ZREAL)
0153           GO TO 130
        C
        C      RECALL THE DISPLAY
        C      ------------------
        C
0154     554   CALL GETSHP(XREAL,YREAL,ZREAL)
0155           GO TO 130
        C
        C      REFLECT THE DISPLAY ABOUT X-Y OR Y-Z OR Z-X PLANE
        C      ---------------------------------------------------
0156     655   ROL=LL2
0157           ISO=4
0158           IF(NPL.EQ.2)ISO=8
0160           FACO=1.
0161           TEX1(ISO)=XOX(ISO)
0162           DO 1000 I=1,ISO
0163           FA=2.*((B/2.)-(XOX(I)-((L*SQRT(2.)-ROL)*SQRT(.5))))
0164    1000   XOX(I)=XOX(I)+FA
0165           SEKA=-TEX1(ISO)+XOX(ISO)
```

```
              ROL1=ROL+(LL1/3.)*SQRT(2.)
              DO 2000 I=1,ISO
              XOX(I)=XOX(I)-SEKA
              FA=2.*((E/2.)-(XOX(I+ISO)-((L*SQRT(2.)-ROL1)*SQRT(.5))))
        2000  XOX(I+ISO)=XOX(I+ISO)+FA-SEKA
              ROL2=ROL+(2.*LL1/3.)*SQRT(2.)
              DO 3000 I=1,ISO
              FA=2.*((E/2.)-(XOX(I+2*ISO)-((L*SQRT(2.)-ROL2)*SQRT(.5))))
        3000  XOX(I+2*ISO)=XOX(I+2*ISO)+FA-SEKA
              ROL3=ROL+LL1*SQRT(2.)
              DO 4000 I=1,ISO
              FA=2.*((E/2.)-(XOX(I+3*ISO)-((L*SQRT(2.)-ROL3)*SQRT(.5))))
        4000  XOX(I+3*ISO)=XOX(I+3*ISO)+FA-SEKA
              DO 5000      I=1,16*NPL
              CALL APNT(XOX(I),YOY(I))
              TEZ1(I)=YOY(I)-L+LL2*SQRT(.5)
              TEY1(I)=XOX(I)-L+LL2*SQRT(.5)
              TEX1(I)=(LL2*SQRT(.5))*SQRT(2.)
              X(I)=TEX1(I)
              Y(I)=TEY1(I)
        5000  Z(I)=TEZ1(I)
              GO TO 6 00
C
C             MODIFY PATCH
C             ------------
C             FIRST IDENTIFY THE PATCH TO BE MODIFIED
C
         755  WRITE(5,760)
         760  FORMAT(1X,'POSITION TRACK. OBJ. AT ANY PT. OF THE PATCH')
              WRITE(5,765)
         765  FORMAT(' TO BE MODIFIED, TYPE <CR> WHEN DONE')
         770  CALL LPEN(IH,ITOTO)
              IF(IH.EQ.0.OR.ITOTO.LT.1.OR.ITOTO.GT.16)GO TO 770
              READ(5,775)IY
         775  FORMAT(A2)
              WRITE(5 777)ITOTO
         777  FORMAT(1X,' YOU HAVE JUST POINTED AT PATCH NUMBER ',I2)
              CALL TRAK(L,L)
              WRITE(5 780)
         780  FORMAT(1X,'POSITION TRAK. OBJ. AT NEW PT,TYPE <CR> WHEN DONE'
              READ(5,775)IO
              CALL TRAKXY(XOO,YOO)
              IPOD=IPO-40
              IPOPO=IPOD+128
         785  CALL LPEN(IK,ITT)
              IF(IK.EQ.0.OR.ITT.LT.IPOD.OR.ITT.GT.IPOPO)GO TO 785
              MX=ITT-IPOD
              WRITE(5,790)MX
         790  FORMAT(1X,'YOU HAVE POINTED AT PT. NUMBER ',I2)
              XOX(MX)=XOO
              YOY(MX)=YOO
              Y(MX)=XOO+(X(MX)/SQRT(2.))-L
              YREAL(MX+(ITOTO-NUMBR(NDEF))*16)=XOO+(X(MX)/SQRT(2.))-L
              Z(MX)=YOO+(X(MX)/SQRT(2.))-L
```

244

```
              ZREAL(MX+(ITOTO-NUMBR(NDEF))*16)=YOC+(X(MX)/SQRT(2.))-L
              IF(NPL.GT.1)GO TO 795
              MONA=16*(ITOTO-1)
              DO 820 I=1,16
              MONA1=MONA+I
              TEX(I)=XREAL(MONA1)
              TEY(I)=YREAL(MONA1)
        820   TEZ(I)=ZREAL(MONA1)
              GO TO 825
        795   DO 810 K=1,4
              K1=(K-1)*4*NPATCH+4*(ITOTO-NUM+NUMBR(NDEF)-1)
              DO 810 J=1,4
              TEX(J+(K-1)*4)=X(K1+J)
              TEY(J+(K-1)*4)=Y(K1+J)
        810   TEZ(J+(K-1)*4)=Z(K1+J)
        825   CALL TEST1(TEX,TEY,TEZ,BXOX,BYOY,BZOZ)
              IOK=(ITOTO-1)*16
              DO 80 I K=1,16
              BXO(IYK+IOK)=BXOX(IYK)
              BYO(IYK IOK)=BYOY(IYK)
        80    BZO(IYK+IOK)=BZOZ(IYK)
              CALL ERAS(ITOTO)
              IF(NPL.GT.1)GO TO 9600
              DO 900 I=1,16
        900   CALL ERAS(IPOD+I)
              GO TO 9500
        9600  DO 9200 I=1,4
              CALL ERAS(IPOD+I+(ITOTO-1)*4)
              CALL ERAS(IPOD+I+8+(ITOTO-1)*4)
              CALL ERAS(IPOD+I+16+(ITOTO-1)*4)
        9200  CALL ERAS(IPOD+I+24+(ITOTO-1)*4)
        9500  CALL CMPRS
              CALL SUBP(ITOTO)
              CALL DR SHP(BXOX,BYOY,BZOZ,L,ITOTO,2,NPL)
              CALL ESUB
              GO TO 130
      C
      C       ERASE A PREDEFINED PATCH
      C       ----------------------
      C
        855   WRITE(5,760)
              WRITE(5,860)
        860   FORMAT(' TO BE ERASED ,TYPE <CR> WHEN DONE')
        870   CALL LPEN(IH,IERAS)
              IF(IH.EQ.2.OR.IERAS.LT.1.OR.IERAS.GT.16)GO TO 870
              IPOD=IPO-40
              READ(5,775)IE
              WRITE(5,777)IERAS
              CALL ERAS(IERAS)
              IF(NPL.GT.1)GO TO 9610
              DO 880 I=1,16
        880   CALL ERAS(IPOD+I+(IERAS-1)*16)
              GO TO 9510
        9610  DO 9120 I=1,4
```

```
              CALL ERAS(IPO-40+I+(IERAS-1)*4)
              CALL ERAS(IPO-40+I+8+(IERAS-1)*4)
              CALL ERAS(IPO-40+I+16+(IERAS-1)*4)
        9100  CALL ERAS(IPO-40+I+24+(IERAS-1)*4)
        9510  CALL CMPRS
              GO TO 130
C
C             GET INTRSECTION OF A PATCH WITH A PLANE
C               -------------------------------------
C
        555   WRITE(5,55)
        55    FORMAT(' POSITION TRACK. OBJ. AT PATCH,TYPE<CR> WHEN DONE')
        56    CALL LPEN(IHO,IHE)
              IF(IHO.EQ.0.OR.IHE.GT.16.OR.IHE.LT.1)GO TO 56
              READ(5,775)IB
              WRITE(5,777)IHE
              WRITE(5,5550)
        5550  FORMAT(1X,'ENTER NUMBER OF SECTIONS')
              READ(5,5560)NSEC
        5560  FORMAT(I2)
              DO 5660 I=1,NSEC
              CALL INTRSC(BXO,BYO,BZO,L,B,IHE)
              CALL ERAS(2140+ISEC)
              CALL CMPRS
        5660  ISEC=ISEC+7
              GO TO 130
C
C             SPLIT OPTION
C           ------------
C
        955   WRITE(5,131)
        131   FORMAT(' POINT WITH THE L.P. AT THE PATCH TO BE SPLITTED')
        132   CALL LPEN(IH,ITOTO)
              IF(IH.EQ.0.OR.ITOTO.LT.1.OR.ITOTO.GT.16)GO TO 132
              READ(5,775)IS
              WRITE(5,777)ITOTO
              JSPLT=(ITOTO-1)*16
              DO 133 I=1,16
              BXSPLT(I)=BXO(JSPLT+I)
              BYSPLT(I)=BYO(JSPLT+I)
        133   BZSPLT(I)=BZO(JSPLT+I)
C
C            CALL SPLITTING ROPUTINE
C
              CALL GEN(BXSPLT,BYSPLT,BZSPLT,X,Y,Z,L,LL2)
              GO TO 6 00
C
C             DONE OPTION
C           -----------
        999   PAUSE
              CALL FREE
              END
```

```
0001          SUBROUTINE SMOSHP(BXX,BYY,BZZ,NPATCH,IET,BX2,BY2,BZ2,L,IFOL)
0002          DIMENSION BX1(16),BY1(16),BZ1(16),BX2(16),BY2(16),
             *  BZ2(16),BXX(1),BYY(1),BZZ(1),IET(2),FBX1(4,4)
             *  ,FBY1(4,4),FBZ1(4,4),FBX2(4,4),FBY2(4,4),FBZ2(4,4)
0003          REAL L
        C
        C     SMOSHP.SUB
        C
        C     PURPOSE:
        C     THIS SUBROUTINE SMOOTHES TWO ADJACENT PATCHES ALONG THE
        C     COMMON BORDERS.
        C
        C     ARGUMENTS:
        C     BXX=ARRAY CONTAINING ELEMENTS OF BX VECTOR BEFORE SMOOTHING
        C     BYY= ,,          ,,          ,,    , BY    ,,    ,,    ,,
        C     BZZ= ,,          ,,          ,,    , BZ    ,,    ,,    ,,
        C     NPATCH=NUMBER OF PATCHES.
        C     IET=ARRAY CONTAINING THE TAGS OF THE TWO PATCHES TO BE SMOOT
        C     BX2=ARRAY CONTAINING ELEMENTS OF BX VECTOR AFTER SMOOTHING
0004   ?C     BY2= ,,          ,,          ,,    , BY    ,,    ,,    ,,
***** K
        C     BZ2= ,,          ,,          ,,    , BZ    ,,    ,,    ,,
        C     L=LENGTH OF THE SHIP
        C     IFOL=1     SMOOTHING ALONG LONGITUDINAL BORDERS    (O/P)
        C        =2      ,,          ,,  TRANSVERSE BORDERS      (O/P)
        C
***** U
0005          DO 1 I=1,2
0006          CALL IONSHP(NPATCH,ITO,L)
0007    1     IET(I)=ITO
0008          DO 10 I=1,16
0009          BX1(I)=BXX(I+(IET(1)-1)*16)
0010          BY1(I)=BYY(I+(IET(1)-1)*16)
0011   10     BZ1(I)=BZZ(I+(IET(1)-1)*16)
0012          DO 20 I=1,16
0013          BX2(I)=BXX(I+(IET(2)-1)*16)
0014          BY2(I)=BYY(I+(IET(2)-1)*16)
0015   20     BZ2(I)=BZZ(I+(IET(2)-1)*16)
0016          CALL GETB(BX1,BY1,BZ1,FBX1,FBY1,FBZ1)
0017          CALL GETB(BX2,BY2,BZ2,FBX2,FBY2,FBZ2)
0018          WRITE(5,600)
0019  600     FORMAT(1X,'IF YOU WANT TO SMOOTH ALONG LONG. BORDER TYPE 1')
0020          WRITE(5 700)
0021  700     FORMAT(1X,'ALONG TRANS. BORDER TYPE 2')
0022          READ(5,800)IFOL
0023  800     FORMAT(I2)
0024          IF(IFOL.EQ.1)GO TO 900
0026          DO 50 I=1,4
0027          FBX2(I,1)=FBX1(I,2)
0028          FBY2(I,1)=FBY1(I,2)
0029   50     FBZ2(I,1)=FBZ1(I,2)
0030          DO 60 I=1,4
0031          FBX2(I,3)=FBX1(I,4)
0032          FBY2(I,3)=FBY1(I,4)
```

```
62    FBZ2(I,3)=FBZ1(I,4)
      GO TO 1002
920   DO 55 I 1,4
      FBX2(1,I)=FBX1(2,I)
      FBY2(1,I)=FBY1(2,I)
55    FBZ2(1,I)=FBZ1(2,I)
      DO 65 I 1,4
      FBX2(3, )=FBX1(4,I)
      FBY2(3,I)=FBY1(4,I)
65    FBZ2(3,I)=FBZ1(4,I)
1002  CALL GETBX(FBX2,FBY2,FBZ2,BX2,BY2,BZ2)
      RETURN
      END
```

```
      SUBROUTINE SHIPS1(X,Y,Z,NPATCH,NPL,L1,L2,L,B,D,XOX,YOY)
      REAL L,L1,L2
      COMMON/RECALL/IRECL
      DIMENSION X(1),Y(1),Z(1),XXX(32),X0(2),Y0(2),Z0(2),XOX(1)
     *     ,YOY(1)
      COMMON/DFILE/IBUF(1)
      COMMON/ UPER/GRD,IP,IP0
C
C     SHIPS1.SUB
C
C       PURPOSE:
C     THIS SUBROUTINE AIDS IN THE DESIGN OF SHIP HULL
C     IT SETS THE WORKING PLANES WITHIN THE PRESCRIPED LENGTHES
C     OF EACH SECTION OF A SHIP .
C     IT RETURNES THE X,Y,Z COORDINATES OF A PATCH
C     TO THE MAIN PROGRAM USING LIGHT PEN
C
C     ARGUMENTS:
C       X=ARRAY CONTAINING THE X COORD. OF THE 16 POINTS DEFINIG
C         THE PATCH      (O/P)
C       Y=O/P Y COORD. OF PATCH        (O/P)
C     Z=O/P Z COORD. OF PATCH        (O/P)
C     NPATCH=NUMBER OF PATCHES
C     NPL=NUMBER OF PATCHES PER PLANE
C     L=LENGTH OF A SHIP
C     L1=LENGTH OF THE SECTION TO BE DESIGNED
C     L2=LENGTH FROM THE ORIGIN OF THE AXIS TO THE SECTION
C     B=BREADTH OF THE SHIP
C     D=DEPTH OF THE SHIP
C     XOX=ARRAY CONTAINIG THE CORRESPONDING X COORD. OF THE PATCH
C        ON ~~THE~~ CRT X-Y PLANE     (O/P)
C     YOY=ARRAY CONTAINIG THE CORRESPONDING Y COORD. OF THE PATCH
C        ON THE CRT X-Y PLANE       (O/P)
C
      BB=(B/2.)
      DD=D
      FACTOR=2.0
      IRECL=IRECL+IRECL*40
C
      WRITE(5,2)
    2 FORMAT(' PLEASE ENTER THE NUMBER OF PATCHES')
      READ(5,3)NPATCH
      WRITE(5,551)
  551 FORMAT(' PLEASE ENTER NUMBER OF PATCHES PER PLANE')
      READ(5,3)NPL
    3 FORMAT(I2)
      K=2
C
      CALL SCAL(0.0,0.0,L+B,L+B)
      XX=L
      CALL SUBP(IP+IRECL)
      CALL APNT(XX,XX,,-4)
      CALL VECT(XX,0.0)
      CALL ESUB
```

```
0025          CALL SUBP(IP+1+IRECL)
0026          CALL APNT(XX,XX,,-4)
0027          CALL VECT(0.0,XX)
0028          CALL ESUB
0029          CALL SUBP(IP+2+IRECL)
0030          CALL APNT(XX,XX,,-4)
0031          CALL VECT(-XX,-XX)
0032          CALL ESUB
0033          II=0
0034          KK=2
0035          KKK=0
        C
0036          CALL SUBP(IP+4+IRECL)
0037          CALL OFF (IP+4+IRECL)
0038          CALL MENU(0.,L,-100.,2000+IP+IRECL,'POSITION','DONE')
0039          CALL ESUB
        C
0040          CALL SUBP(IP+3+IRECL)
0041          CALL OFF(IP+3+IRECL)
0042          CALL MENU(0.0,L,-100.,2000+IP+IRECL,'DEFINE WORKING PLANE',
             *  'DONE')
0043          CALL ESUB
        C
0044          DO 4 J=1,NPATCH/NPL
0045          WRITE(5 5)J
0046     5    FORMAT(' DEFINE PATCHES IN PLANE NUMBER ',I2)
0047          WRITE(5,6)
0048     6    FORMAT('-------------------------',//)
        C
0049     700  IF(KKK.EQ.5.OR.KKK.EQ.9.OR.KKK.EQ.13.OR.KKK,EQ.17)KKK=KKK-1
0051          IF(KKK.EQ.21.OR.KKK.EQ.25.OR.KKK.EQ.29.OR.KKK.EQ.33)KKK=KKK-
0053          IF(KK.EQ.4.OR.KK.EQ.8.OR.KK.EQ.12.OR.KK.EQ.16)GO TO 701
0055          IF(KK.EQ.20.OR.KK.EQ.24)GO TO 701
0057          GO TO 702
0058     701  DO 16 MEME=1,KK
0059     16   CALL OFF(IP+4+MEME+IRECL)
0060     702  CALL ON(IP+3+IRECL)
0061          CALL MENUH(IT,2000+IP+IRECL,2001+IP+IRECL)
0062          CALL OFF(IP+3+IRECL)
0063          GO TO (20,130),IT
0064     20   CALL TRAK(XX,XX)
0065          IF(II.EQ.4)II=0
0067          KK=KK+1
0068          II=II+1
0069          WRITE(5,11)II
0070     11   FORMAT(1X,'POSITION TRAK. OBJ. TO DEFINE SEC. NUMBER  ',I1)
0071          READ(5,21)M
0072     21   FORMAT(A2)
0073     30   CALL LPEN(IH,IT1)
0074          IF(IH.EQ.0.OR.IT1.LT.101.OR.IT1.GT.103)GO TO 30
0076          IT1=IT1-100
0077          GO TO (100,200,300),IT1
0078     100  GO TO 300
0079     200       GO TO 300
```

```
0080    300  CALL APNT(L-L2-FACTOR,L-L2-FACTOR,,-4)
0081         CALL SUBP(IP+4+KK+IRECL)
0082         CALL OFF(IP+4+KK+IRECL)
0083         CALL VECT(BB,0.0,,II)
0084         CALL VECT(0.0,DD,,II)
0085         CALL VECT(-BB,0.0,,II)
0086         CALL VECT(0.0,-DD,,II)
0087         CALL VECT(BB/10.,0.0,,-4)
0088         CALL VECT(0.0,DD,,II)
0089         CALL VECT(BB/10.,0.0,,-4)
0090         CALL VECT(0.0,-DD,,II)
0091         CALL VECT(BB/10.,0.0,,-4)
0092         CALL VECT(0.0,DD,,II)
0093         CALL VECT(BB/10.,0.0,,-4)
0094         CALL VECT(0.0,-DD,,II)
0095         CALL VECT(BB/10.,0.0,,-4)
0096         CALL VECT(0.0,DD,,II)
0097         CALL VECT(BB/10.,0.0,,-4)
0098         CALL VECT(0.0,-DD,,II)
0099         CALL VECT(BB/10.,0.0,,-4)
0100         CALL VECT(0.0,DD,,II)
0101         CALL VECT(BB/10.,0.0,,-4)
0102         CALL VECT(0.0,-DD,,II)
0103         CALL VECT(BB/10.,0.0,,-4)
0104         CALL VECT(0.0,DD,,II)
0105         CALL VECT(BB/10.,0.0,,-4)
0106         CALL VECT(0.0,-DD/10.,,-4)
0107         CALL VECT(-BB,0.0,,II)
0108         CALL VECT(0.0,-DD/10.,,-4)
0109         CALL VECT(BB,0.0,,II)
0110         CALL VECT(0.0,-DD/10.,,-4)
0111         CALL VECT(-BB,0.0,,II)
0112         CALL VECT(0.0,-DD/10.,,-4)
0113         CALL VECT(BB,0.0,,II)
0114         CALL VECT(0.0,-DD/10.,,-4)
0115         CALL VECT(-BB,0.0,,II)
0116         CALL VECT(0.0,-DD/10.,,-4)
0117         CALL VECT(BB,0.0,,II)
0118         CALL VECT(0.0,-DD/10.,,-4)
0119         CALL VECT(-BB,0.0,,II)
0120         CALL VECT(0.0,-DD/10.,,-4)
0121         CALL VECT(BB,0.0,,II)
0122         CALL VECT(0.0,-DD/10.,,-4)
0123         CALL VECT(-BB,0.0,,II)
0124         CALL ESUB
0125    400  CALL ON(IP+4+KK+IRECL)
0126         FACTOR=FACTOR+L1/3.
0127         I=1+K*NPL
0128         IF(II.EQ.2)I=1+4*NPL+K*NPL
0130         IF(II.EQ.3)I=1+8*NPL+K*NPL
0132         IF(II.EQ.4)I=1+12*NPL+K*NPL
        C
0134   1200  KKK=KKK+1
0135         CALL ON (IP+4+IRECL)
```

256

```
      CALL MENUH (IT2,2930+IP+IRECL,2921+IP+IRECL)
      CALL OFF (IP+4+IRECL)
      GO TO (600,700),IT2
600   CALL TRAKXY(X(I),Y(I))
      XOX(I)=X(I)
      YOY(I)=Y(I)
      CALL SUBP(IPO+KKK+IRECL)
      CALL OFF(IPO+KKK+IRECL)
      CALL APNT(X(I),Y(I),1,4)
      CALL ESUB
      CALL ON(IPO+KKK+IRECL)
601   GO TO (800,900,1000),IT1
800   GO TO 1 00
900     GO TO 1000
1000  Z(I)=Y(I)-L+L2*SQRT(.5)
      Y(I)=X(I)-L+L2*SQRT(.5)
      X(I)=(L2*SQRT(.5))*SQRT(2.)
1100  I=I+1
      GO TO 1200
130   DO 1 IOI=1,KK
1     CALL ERAS(IP+4+IOI+IRECL)
      CALL CMPRS
      II=0
      K=16*J
4     CONTINUE
      RETURN
      END
```

```
0001            SUBROUTINE GETSHP(X,Y,Z)
0002            DIMENSION X(1),Y(1),Z(1)
0003            COMMON/DFILE/IBUF(1)
0004            COMMON/ ECALL/IRECL
0005            LOGICAL*1 FILE1(15),FILE2(15)
          C
          C     SUBROUTINE GETSHP
          C     FUNCTION:
          C     RECALL A PREVIOUSLY SAVED DISPLAY.
          C
0006            CALL INFILE(FILE1,FILE2,NPATCH)
0007            CALL STOP
0008            CALL ASSIGN(10,FILE2,0)
0009            DO 1 I=1,NPATCH*16
0010      1     READ(10,*)X(I),Y(I),Z(I)
0011            CALL CLOSE(10)
0012            CALL INIT
0013            CALL RSTR(FILE1)
0014            CALL CONT
0015            CALL LPEN(IH,IT)
0016            IRECL=IRECL+NPATCH
0017            DO 10 I=1,12
0018      10    CALL ERAS(100+I)
0019            CALL CMPRS
0020            RETURN
0021            END
```

```
0001          SUBROUTINE INTRSC(BX,BY,BZ,L,B,IHE)
0002          DIMENSION U(20),W(20),XINTR(20),YINTR(20),ZINTR(20)
             *  ,BX(1),BY(1),BZ(1),DELX(19),DELY(19),BBX(16),BBY(16)
             *  ,BBZ(16)
0003          COMMON/DFILE/IBUF(1)
0004          COMMON/SUPER/GRD,IP,IPC
0005          COMMON/SEC/ISEC
0006          REAL L
        C
        C     THIS SUBROUTINE DRAWS THE RESULTANT CONTINUOUS CURVE
        C     FROM THE INTERSECTION OF A PLANE AND A PATCH
        C
0007          C=L+B
0008          CALL APNT(C/8.,5.*C/8.,,-4)
0009          CALL SUBP(6140+ISEC)
0010          CALL OFF(6140+ISEC)
0011          CALL LVECT(.25*C,0.)
0012          CALL LVECT(0.,.25*C)
0013          CALL LVECT(-.25*C,0.)
0014          CALL LVECT(0.,-.25*C)
0015          CALL APNT(.4*C,5.*C/8.,,-4)
0016          CALL TEXT(' U')
0017          CALL APNT(C/9.,.95*C,,-4)
0018          CALL TEXT(' W')
0019          CALL APNT(.1*C,4.5*C/8.,,-4)
0020          CALL TEXT('0,0')
0021          CALL APNT(.3*C,4.5*C/8.,,-4)
0022          CALL TEXT('1,0')
0023          CALL APNT(.3*C,.9*C,,-4)
0024          CALL TEXT('1,1')
0025          CALL APNT(.1*C,.9*C,,-4)
0026          CALL TEXT('0,1')
0027          CALL ESUB
0028          CALL ON(6140+ISEC)
        C
        C     POSITION TRACKING OBJECT ON THE POINTS OF INTRSECTION
        C
0029          CALL TRAK(L,L)
0030          KEMO=0
0031     10   WRITE(5,11)
0032     11   FORMAT(1X,'POSITION TRAC. OBJ.,TYPE<CR> WHEN DONE')
0033          READ(5,20)I
0034     20   FORMAT(A2)
0035          IF(KEMO.GT.0)GO TO 30
0037          CALL TRAKXY(X0,Y0)
0038          KEMO=KEMO+1
0039          GO TO 10
0040     30   CALL TRAKXY(X01,Y01)
        C
        C     CALCULATE(U1,W1),(U20,W20)
        C
0041          CALL SUBP(6141+ISEC)
0042          CALL APNT(X0,Y0)
0043          CALL LVECT((X01-X0),(Y01-Y0))
```

```
              CALL ESUB
              U(1)=(X0-(C/8.))/(.25*C)
              W(1)=(Y0-(5.*C/8.))/(.25*C)
              U(20)=(X01-(C/8.))/(.25*C)
              W(20)=(Y01-(5.*C/8.))/(.25*C)
              DELTAU=(U(20)-U(1))/19.
              DELTAW=(W(20)-W(1))/19.
              DO 42 I=2,19
              U(I)=U(I-1)+DELTAU
        42    W(I)=W(I-1)+DELTAW
       C
       C      DRAW THE INTERSECTION CURVE
       C
              DO 55 I=1,16
              BBX(I)=BX(I+(IHE-1)*16)
              BBY(I)=BY(I+(IHE-1)*16)
        55    BBZ(I)=BZ(I+(IHE-1)*16)
       C
              CALL POINTS(U,W,BBX,XINTR,20)
              CALL POINTS(U,W,BBY,YINTR,20)
              CALL POINTS(U,W,BBZ,ZINTR,20)
              DO 31 I=1,20
              XINTR(I)=XINTR(I)*SQRT(.5)
              YINTR(I)=YINTR(I)-XINTR(I)+L
              ZINTR(I)=ZINTR(I)-XINTR(I)+L
        31    CONTINUE
              DO 33 I=1,19
              DELX(I)=YINTR(I+1)-YINTR(I)
        33    DELY(I)=ZINTR(I+1)-ZINTR(I)
       C
              CALL SUBP(6142+ISEC)
              CALL APNT(YINTR(1),ZINTR(1),,-4)
              DO 34 I=1,19
        34    CALL LVECT(DELX(I),DELY(I),,3)
              CALL ESUB
              RETURN
              END
```

```
      SUBROUTINE GEN(BX,BY,BZ,X,Y,Z,L,L2)
      DIMENSION BX(16),BY(16),BZ(16),X(16),Y(16),Z(16)
      DIMENSION U(2),W(2),UU(16),WW(16)
      REAL L,L2
      COMMON/DFILE/IBUF(1)
      COMMON/SUPER/GRD,IP,IPO
C
C     GET X,Y,Z COORD. OF THE PATCH GENERATED BY THE
C     SUBDIVISION OF THE GIVIN PATCH
C     CALL IT X,Y,Z
C
      WRITE(5,3)
    3 FORMAT(1X,32HENTER U(1),U(2),W(1),W(2) VALUES)
      READ(5,*)U(1),U(2),W(1),W(2)
C
      DELU=(U(2)-U(1))/3.
      DELW=(W(2)-W(1))/3.
      UU(1)=U(1)
      UU(2)=U(1)+DELU
      UU(3)=U(1)+2.*DELU
      UU(4)=U(1)+3.*DELU
      DO 1 I=1,4
      WW(I)=W(1)
      WW(I+4)=W(1)+DELW
      WW(I+8)=W(1)+2.*DELW
    1 WW(I+12)=W(1)+3.*DELW
      DO 10 I=1,3
      UU(1+4*I)=UU(1)
      UU(2+4*I)=UU(2)
      UU(3+4*I)=UU(3)
   10 UU(4+4*I)=UU(4)
      CALL POINTS(UU,WW,BX,X,16)
      CALL POINTS(UU,WW,BY,Y,16)
      CALL POINTS(UU,WW,BZ,Z,16)
      DO 20 I=1,16
      X(I)=X(I)/SQRT(2.)
      Y(I)=Y(I)-X(I)+L
   20 Z(I)=Z(I)-X(I)+L
      IPOPO=IPO-40                             !!!!!
      DO 30 I=1,16
      CALL ERAS(IPOPO+I)
      CALL SUBP(IPOPO+I)
      CALL APNT(Y(I),Z(I),1,8)
   30 CALL ESUB
      CALL CMPRS
      DO 40 I=1,16
      Z(I)=Z(I)-L+L2*SQRT(.5)
      Y(I)=Y(I)-L+L2*SQRT(.5)
   40 X(I)=(L *SQRT(.5))*SQRT(2.)
      RETURN
      END
```

```
          SUBROUTINE IDNSHP(NPATCH,ITO,L)
          REAL L
          COMMON/DFILE/IBUF(1)
          COMMON/ UPER/GRD
      C   IDNSHP.SLB
      C
      C   PURPOSE:
      C   THIS SU ROUTINE IDENTIFY THE PATCH
      C
      C   ARGUMENTS
      C   NPATCH=NUMBER OF PATCHES
      C   ITO   =TAG OF THE SUBPICTURE OF THE HITTEN PATCH   (O/P)
      C   L     =LENGTH OF THE SHIP
      C
          DO 10 I 1,2
          CALL POINTR(9,I)
     10   CALL SENSE(9,1)
          TYPE *,' IDENTIFY THE  PATCH'
          CALL TR K(L,L)
          WRITE(5,1)
     1      FORMA (' TO DO SO ,POSITION THE TRACKING OBJECT AT ')
          WRITE(5 40)
     40     FORMAT('ANY PART OF THE PATCH AND TYPE <CR> WHEN DONE')
          READ(5,2)M
     2    FORMAT(A2)
          CALL ME UH(ITO,1,20)
          DO 30 I=1,2
          CALL POINTR(9,I)
     30   CALL SENSE(9,-1)
          RETURN
          END
```

262

```
0001            SUBROUTINE FDREV(U,W,B,D,KOK)
0002            DIMENSION B(16)
          C
          C     THIS SUBROUTINE CALCULATES THE FIRST DREVATIVES OF V(U,W)
          C
0003            IF(KOK.EQ.2)GO TO 1
          C     GET DV/DU
0005            D=(3.*U**2)*(W**3)*B(1)+(3.*U**2)*(W**2)*B(2)+
               *  (3.*U**2)*(W*B(3))+(3.*U**2)*B(4)+
               *     2.*U*(W**3)*B(5)+2.*U*(W**2)*B(6)+
               *     2.*U*W*B(7)+2.*U*B(8)+(W**3)*B(9)+(W**2)*B(10)
               *  +W*B(11)+B(12)
0006            GO TO 3
          C     GET DV/DW
0007       1    D=B(15)+2.*W*B(14)+(3.*W**2)*B(13)+U*B(11)+2.*U*W*B(10)
               *  +U*(3.*W**2)*B(9)+(U**2)*B(7)+(2.*W)*(U**2)*B(6)+(3.*W**2
               *  *(U**2)*B(5)+(U**3)*B(3)+2.*W*(U**3)*B(2)
               *  +(3.*W**2)*(U**3)*B(1)
0008       3    RETURN
0009            END
```

```
0001            SUBROUTINE SDREV(U,W,B,D)
0002            DIMENSION B(16)
          C
          C     THIS SUB. CALCULATES D**2V/DUDW
          C
0003            D=(3.*U**2)*(3.*W**2)*B(1)+(3.*U**2)*(2.*W)*B(2)+
               *  (3.*U**2)*B(3)+2.*U*(3.*W**2)*B(5)+(2.*U)*(2.*W)*B(6)
               *  +2.*U*B(7)+(3.*W**2)*B(9)+2.*W*B(10)+B(11)
0004            RETURN
0005            END
```

```
      SUBROUTINE CRVFIT(X,Y,N,XCRV,YCRV,NCRV)
      DIMENSION X(N),Y(N),XCRV(NCRV),YCRV(NCRV)
      NCRV1=INT(NCRV/N)
      K=0
      DO 10 J=1,N
      DO 20 JJ=0,NCRV1
      K=K+1
      T=FLOAT(JJ)/FLOAT(NCRV)+FLOAT(J)
      FI=FLOAT(J)
      WIM1=(T**2-(2.*FI*T)-2.*T+2.*FI+J**2+1.)/2.
      WI=(-2.*T**2+(4.*FI*T)+(2.*T)-2.*FI**2-2.*FI+1.)/2.
      WIP1=(T**2-(2.*FI*T)+J**2)/2.
      XCRV(K)=X(J)*WIM1+X(J+1)*WI+X(J+2)*WIP1
      YCRV(K)=Y(J)*WIM1+Y(J+1)*WI+Y(J+2)*WIP1
   20 CONTINUE
   10 CONTINUE
      RETURN
      END
```

```
0001            SUBROUTINE CURVPT(U,W,B,JP,X,N)
0002            DIMENSION U(N),W(N),B(16,3),X(N)
          C
          C     SUBROUTINE CURVPT
          C     FUNCTION:
          C     CALCULATES THE X,OR Y,OR Z COORDINATES OF N POINTS LYING ON
          C     THE PATCH SURFACE.
          C     U,W ARE INPUT ARRAYS DEFINING THE PARAMETRIC VALUES OF
          C     THE POINTS.
          C     B IS THE INPUT BOUNDARY MATRIX DEFINING THE COEFFICIENTS OF
          C     N=INPUT NUMBER OF POINTS.
          C     X=OUTPUT ARRAY CONTAINING COORDINATES OF POINTS.
          C..
          C.    ALY BADAWY
          C.    OCT 10-79
          C
0003            DO 1 I=1,N
0004            X(I)=(U(I)**3)*(W(I)**3)*B(1,JP)+(U(I)**3)*(W(I)**2)
               *  *B(2,JP)+(U(I)**3)*(W(I)*B(3,JP))+(U(I)**3)*B(4,JP)+(U(I)
               *  **2)*(W(I)**3)*B(5,JP)+(U(I)**2)*(W(I)**2)*B(6,JP)+(U(I
               *  )**2)*W(I)*B(7,JP)+(U(I)**2)*B(8,JP)+(U(I)*W(I)**3)*
               *  B(9,JP)+(U(I)*W(I)**2)*B(10,JP)+U(I)*W(I)*B(11,JP)
               *  +U(I)*B(12,JP)+(W(I)**3)*B(13,JP)+(W(I)**2)*B(14,JP)
               *  +W(I)*B(15,JP)+B(16,JP)
0005      1     CONTINUE
0006            RETURN
0007            END
```

```
      SUBROUTINE DRAWP(BX,BY,BZ)
      DIMENSION BX(16),BY(16),BZ(16),X(16),Y(16),Z(16)
      DIMENSION DELTAX(15),DELTAY(15),TOT(11),W(16),U(16)
      COMMON/DFILE/IBUF(1)
C
C     THIS SUBROUTINE DRAWS THE PATCH
C
      TOT(1)=0.0
      TOT(2)=.1
      TOT(3)=.2
      TOT(4)=.3
      TOT(5)=.4
      TOT(6)=.5
      TOT(7)=.6
      TOT(8)=.7
      TOT(9)=.8
      TOT(10)=.9
      TOT(11)=1.0
C
      MEM=0
      MEM1=0
      DO 99 KOK=1,11
      T=0.0
      DO 48 I=1,16
      W(I)=TOT(KOK)
      U(I)=T
   48 T=T+1./15.
  100 CALL POINTS(U,W,BX,X,16)
      CALL POINTS(U,W,BY,Y,16)
      CALL POINTS(U,W,BZ,Z,16)
      DO 90 I=1,16
      X(I)=X(I)/SQRT(2.)
      Y(I)=Y(I)-X(I)
   90 Z(I)=Z(I)-X(I)
      DO 91 I=1,15
      DELTAX(I)=Y(I+1)-Y(I)
      DELTAY(I)=Z(I+1)-Z(I)
   91 CONTINUE
      Y(1)=Y(1)+512.
      Z(1)=Z(1)+512.
      CALL APNT(Y(1),Z(1),,-4)
      DO 92 I=1,15
   92 CALL VECT(DELTAX(I),DELTAY(I))
      IF(MEM.GT.1)GO TO 999
      DO 101 I=1,16
      MEM=MEM+1
      MEM1=MEM1+1
      W(I)=U(I)
  101 U(I)=TOT(KOK)
      GO TO 100
  999 MEM=0
   99 CONTINUE
      RETURN
      END
```