The art of machining complex, doubly curved surfaces has been advanced by experimental extensions to the Automatic Programmed Tool Language (APT) and its numerical control program.

Described are mathematical programming procedures and language extensions for directing the cutting path of a machine tool over a ship's propeller, which serves as an example.

Illustrative of an area for future development is the possible extension of interactively designed surfaces—using graphic displays—to interface with an APT processor.

Numerical control for machining complex surfaces

by D. B. Almond

The machining of three dimensional objects that embody complex surfaces such as aircraft contours or ship propellers challenges the machine tool numerical control programmer. In industries where automated manufacturing is playing an increasingly important role, product designs often call for physical surfaces that cannot be represented analytically by simple mathematical expressions. An aircraft fuselage or an automobile body, for example, may be conceived in clay, but eventually it is produced in aluminum or steel by methods that demand a numerical representation of the surfaces. With the growth of numerical control, the need has increased to devise analytical techniques for the description of these surfaces, particularly for the application of the techniques to the Automatic Programmed Tool (APT) numerical control processor.[1] In fact, the application of APT, the most powerful and widely used numerical control parts programming language to more sophisticated and complex machining requirements, has emphasized the need for a more advanced numerically controlled sculptured surface technology. By *sculptured surface* is meant the mathematical description of a conceptualized physical surface that requires a process similar to sculpturing to produce.

The APT language is a high-level application language that permits a programmer to describe a part shown on an engineering drawing and to describe the steps required to machine that part. Briefly stated, the IBM System/360 APT processor program con-

verts numerical control statements to a form that ultimately causes a machine tool to produce the part in metal.[2] Generally, the geometric surfaces used to define the part are of the classic types, such as planes, cylinders, and spheres.

The objective of our programming research and development is to generate APT program statements for the machining of complex, doubly curved surfaces that are furnished as tables of surface points. The project undertaken by the author has resulted in extensions to APT that permit the writing of statements for machining ship propellers whose surfaces are to be produced by sculpturing methods such as milling and grinding. Further, the required part-sculpturing program has the generality to accept input parameters of a variety of propeller designs, and to machine each propeller by choosing from among a multiplicity of possible cutter paths. The example application to propeller machining was performed under contract with the Danly Machine Corporation.

After reviewing concepts of the APT language and the processor to translate APT-language instructions into tool commands, ways in which the language can be extended are introduced. Discussed are surface notations, emphasizing the elemental surface—surface patch—that are to be used. This experimental investigation involves the use of vector notation to calculate cutter-to-surface geometry and cutter path. It is then shown how these calculations can be embodied into the core of the APT Arithmetic Element (ARELEM). Practical experience is discussed in terms of computer time for calculating sculpturing cuts for a specified ship propeller blade. Illustrated is a capability for visually verifying the completed propeller through interactive graphics.

The result is an experimentally augmented APT having the following two novel features: (1) an APT sculptured surface command, called SCSURF; and (2) an improved Arithmetic Element ARELEM for generating tool-motion commands for sculptured surfaces. The addition of SCSURF to APT permits the reduction of sculptured surface inputs to a canonical form as discussed in this paper. The capabilities of the ARELEM have been increased to calculate tool end points and surface normals for machining a sculptured surface.

APT language

basic principles

For background and a better understanding of APT it may be profitable to discuss briefly the APT language principles and their extensions before considering the surface design logic in the framework of the APT system.

The APT language allows a numerical control programmer to define the geometry of a part to be machined, to describe the path the cutter should take, and to issue machine tool commands. In addition, it permits various arithmetic operations and calculations using such functions as the sine, tangent, and square root. The language specifies a variety of geometric forms, including points, planes, circles, and tabulated cylinders for describing a part. Typical APT geometric definitions might be the following:

```
P1 = POINT/10, −3.375, 18.75
P2 = POINT/INTOF, LN1, LN2
```

Here P1 is defined by three coordinates, whereas P2 is the result of the intersection of two lines LN1 and LN2.

For each geometric type, there may be several methods of definition that are reduced by the APT processor to a single canonical form. For so-called "small" surfaces, the canonical form is fixed in length, as, for example, a point definition reduces to the x, y, and z coordinates. For "large" surfaces the length is variable as in the case of our sculptured surface, which requires a canonical form length somewhat proportional to the amount of input data. Part surface geometry may be named for later reference in a program or unnamed for temporary application only. Further, a surface may be defined in one coordinate system and used in another through the use of matrix translation and rotation. Such surfaces are termed "APT surfaces."

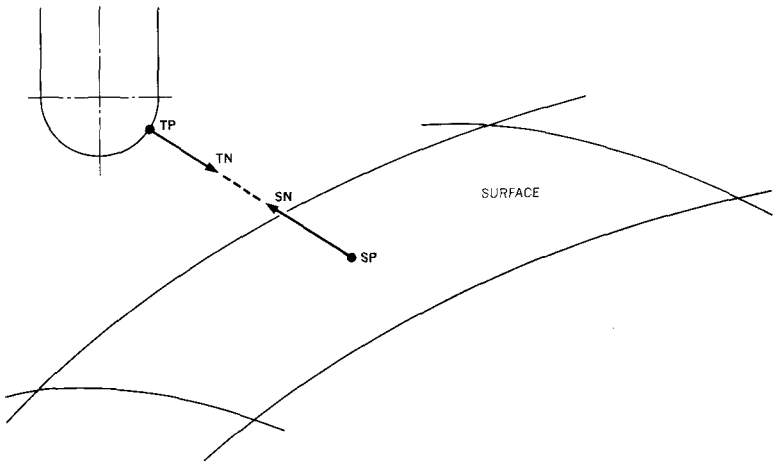
Motion commands are used to position the cutter and to describe the orientation of the cutter to the surfaces to be machined. These commands also give the direction of motion. The following are typical motion commands:

```
FROM/P1
GOTO/P2
GO/ON, CIR, TO, PL1, ON, PL2
GOFWD/CIR, PAST, TAB1
```

The APT words to the left of the slash describe the motion to be performed, and the words to the right denote surfaces and positional modifiers (TO, ON, PAST) to specify cutter relationships with the surfaces.

Statements prepared by the numerical control programmer become inputs to the APT processor for interpreting the data and for performing the computations to produce an output file. This file may be deciphered by a user postprocessor program to create machine-readable commands for a specific machine tool. It is the input-translator section of APT that reads programmer input (usually cards) and converts the data into an intermediate program file. This file becomes input to the arithmetic element

Figure 1 Normal distance



where cutter path coordinates are computed while maintaining the tolerances specified by the programmer. The process relies on an iterative technique to generate the cutter path data that is written into an output file together with other pertinent machining data.

Moving the cutter to the surfaces as prescribed by the program is an iterative procedure using what is termed the normal distance calculation. The *normal distance* is the distance from a point **TP** on the cutter envelope to a point **SP** on the surface as illustrated in Figure 1. Such points are oriented so that a line joining them is normal to the surface and to the cutter envelope. To compute the normal distance, it is necessary to determine the directed distance and surface unit normal **SN** shown in Figure 1. If one is given a point **TP** on the cutter and a unit vector **TN** normal to the cutter envelope at **TP**, the surface point **SP** is the intersection of a line through **TP** parallel to **TN**. The surface normal **SN** is a unit vector normal to the surface at the pierce point **SP**.

For a surface type to be used as an APT surface it is sufficient to provide the directed-distance and the unit-normal routines as well as provisions for dispatching the routines as required by the APT processor. We now discuss the essential problem of adding a new APT surface type of the Coons class.

extensions

Required are language extensions for defining sculptured surfaces that are to be incorporated into the IBM APT system. The form of the surface representation is such that a surface may be named and used as any APT surface. Complex surfaces are constructed by building sets of surface "patches," each of which is

expressed in terms of its boundaries and "blending" functions that assure continuity across boundaries between adjacent surface patches. For a surface form to be fully integrated with the APT system it is not sufficient simply to calculate surface points, but also requires solving for the pierce point of a line with the APT surface. This problem is complicated by the fact that a sculptured surface is composed of a large set of the aforementioned surface patches. A patch search scheme has been devised to select candidate patches. Then a determination is made of the line and patch intersection within the candidate patch.

The analytical model employed to define a surface is based on the work of Coons.[3] It should be noted that Coons and others who have defined this class of surfaces were concerned primarily with the surface design process itself when integrated into computer graphics packages. Nevertheless, because of its simplicity, the Coons surface modeling technique is quite a satisfactory way to describe a surface whether it is related to designing through visual control at an interactive display terminal, or expressing analytically a surface composed of a given set of input points used in a batch processing mode.

The present numerical control state-of-the-art includes a number of specialized programs that have been developed to interface with APT processors to satisfy specific machining requirements and raw data inputs. Several of these programs are available to members of the APT Long Range Program (ALRP) administered by the Illinois Institute of Technology Research Institute.[4] On the one hand, a specialized program may contain the particular characteristics that allow a numerical control programmer to fulfill a specific assignment. It may, however, be disconcerting to one with a complex programming task that involves sculptured surfaces to discover he lacks the generalized programming tools to easily complete the task.

The approach taken in the development of the APT sculptured surface programming discussed in this paper is to achieve a fairly high degree of generality while maintaining simplicity of use. For example, input is an ordered set of points or points and surface tangents that may be specified by the programmer or points that are defined in APT. Although inputs must form a mesh-type surface, it is not necessary that the pattern of points be as a lattice. Furthermore, there is no restriction imposed as to the single-valuedness of the surface with respect to some reference plane. However, a surface that is intersected by a line at more than one point could encounter difficulties in the APT Arithmetic Element (ARELEM).[2] The machining of ship propellers (which is the APT application discussed in this paper) demonstrates the attractiveness of the APT language extensions as a parts programming instrument.

Surface description

Coons surfaces

The surfaces to be described and machined are Coons surfaces composed of sets of piecewise continuous surface patches each having four corner points and boundary curves. Two intersecting sets of curves are constructed through the points to divide the surface into the quadrilateral surface segments or *patches* with which we are concerned. As previously mentioned, the Coons modeling technique is intended for the designing of a surface, beginning with few constraints and a single patch and subdividing only as design requirements dictate. On the other hand, our concern here is to describe a surface that contains exactly the given surface points, although not without paying a small penalty for having to deal with a large number of patches.

The particular advantages of Coons surfaces are that they are first of all parametric, and, secondly, that they may be expressed in a matrix form that permits simple arithmetic calculations. A parametrically designed surface allows for the definition of slopes that in a nonparametric description may be infinite and, therefore, troublesome to handle. The boundaries and blending functions are of the same form, thereby permitting the surface equation to be expressed in an easily calculable tensor form. *Blending functions* are scalar functions of a single variable that serve to influence the shape of the surface by a blending of the boundary curves over the surface.

bicubic surfaces

Although the mathematical structure of the Coons surface may be implemented in a variety of ways, the usual method is that of parametric bicubics. Other surface modeling forms that may be of interest to the reader include rational polynomials,[5] quartics,[6] and Bézier functions.[7] The author, however, seeks the simplest APT canonical form for a sculptured surface without recourse to the preprocessing of input data or interfacing with a computer-graphics-designed surface.

A cubic is the lowest order polynomial that can symbolize a curve that twists through space and, therefore, form patch boundaries that are nonplanar. In addition, bicubic surfaces offer computational economy because the cubic is easily evaluated. Illustrative of recourse to bicubic surfaces is the engineering design program of Eshleman and Meriwether.[8]

surface-patch notation

Before considering the surface-patch equation in detail, we should first discuss some matrix notation that it uses. If we let the x, y, and z coordinates of a point on a surface be expressed in terms of two independent variables u and w, we have

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

For any pair of u and w values, the surface point is specified assuming that the functions f, g, and h are known. A curve can be generated in space by holding either of the independent variables u or w constant.

By symbolizing a surface point **P** in vector notation we have

$$\mathbf{P} = [x \quad y \quad z] = [f(u,w) \qquad g(u,w) \qquad h(u,w)]$$

or, in terms of the independent variables, the surface point is

$$\mathbf{P} = [u \quad w]$$

Further simplication of notation may be made by removing the brackets so that the bilateral symbol uw represents the vector.

One constraint placed on the variables u and w is that their range over a patch is from 0 to 1. A unit square in uw space is topographically equivalent to a surface patch.

Figure 2 A surface patch

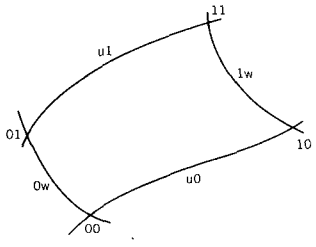


Figure 2 illustrates a patch having four boundary curves $u0$, $u1$, $0w$, and $1w$ and four corner points 00, 01, 10, and 11. Typically $u0$ stands for the boundary curve with $w = 0$ and u varying. Similarly, 00 represents the x, y, and z coordinates of the vector $[f(0,0) \quad g(0,0) \quad h(0,0)]$.

The patch equation is expressed in terms of the blending functions and a boundary-condition matrix. In matrix notation, the equation has the following compact form where the superscript t indicates the transposed matrix:

$$uW = UMBM^tW^t$$

UM and MW denote the blending functions, which are functions of u and w respectively. When the boundary curves and blending functions are related to cubic basis vectors, the surface patch becomes a bicubic surface, and the cubic basis vectors are

$$U = [u^3 \quad u^2 \quad u \quad 1]$$
$$W = [w^3 \quad w^2 \quad w \quad 1]$$

The blending functions are related to the basis vectors by the following constant matrix, which can be evaluated by the method given by Coons[3]:

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

B is the boundary-condition matrix, which contains all the information concerning the patch boundaries. In compact notation, the boundary-condition matrix is expressed as follows:

$$\left[\begin{array}{cc|cc} 00 & 01 & 00_w & 01_w \\ 10 & 11 & 10_w & 11_w \\ \hline 00_u & 01_u & 00_{uw} & 01_{uw} \\ 10_u & 11_u & 10_{uw} & 11_{uw} \end{array}\right]$$

(The subscripts refer to partial derivatives with respect to the subscripted elements. For example, $00_u = \partial(uw)/\partial u$ where $u = w = 0$.)

The upper left four values are the corner coordinates, and the upper right and lower left submatrices contain the corner slopes. The lower right submatrix consists of the corner cross derivatives (also called the twist vectors), which are considered to be zero by the program. The B matrix is a tensor that contains the x, y, and z components of coordinates, slopes, and twists.

For this experiment, we deliberately chose to let the twist vectors be zero, realizing that first-order slope continuity at patch boundaries alone could leave a residue of second-order quasi-flattening at patch corners. In our experiment, the decision was justified in that propellers machined by the present SCSURF program exhibit no measureable flattening. We do, however, intend to improve our program by incorporating second-order continuity control, which will, of course, involve additional storage and execution time.

For a particular patch, the matrix product MBM^t is constant and is computed before evaluating the patch equation for given u and w values. If we let S_k represent MBM^t, the equation may now be expressed as follows:

$$P_k = US_kW^t$$

where

$$k = 1,2,3 \cdots$$

and

$$x = P_1$$

$$y = P_2$$

$$z = P_3$$

The slopes or tangents along w-curves and u-curves may be readily obtained by computing the derivative of the patch equation with respect to u and w as follows:

$$\frac{\partial P_k}{\partial u} = [3u^2 \quad 2u \quad 1 \quad 0] \quad S_k \quad [w^3 \quad w^2 \quad w \quad 1]^t$$

$$\frac{\partial P_k}{\partial w} = [u^3 \quad u^2 \quad u \quad 1] \quad S_k \quad [3w^2 \quad 2w \quad 1 \quad 0]^t$$

for $k = 1,2$, and 3

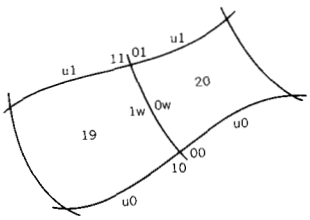
and $0 \leq u, w \leq 1$

It follows that the surface normal may be obtained by computing the cross product of the tangent vectors evaluated at the given u and w values.

Tangent vectors and boundary continuity

Thus far the discussion has mostly concerned a single surface patch. The total surface that this paper addresses, however, consists of a piecewise fitting of many surface patches so as to provide continuity across patch boundaries. To attain the desired continuity, a method of computing tangent vectors at the patch corners has been developed.

Figure 3 Two surface patches with a common boundary



Consider two patches with a common boundary such as those in Figure 3. Patch 19 has boundary $1w$, and patch 20 has boundary $0w$. Assume that the corner point coordinates and the slopes along the boundary at the corner points are respectively equal. Then the boundary is common, and the patches are said to have zero order or C^0 continuity across the common boundary curve. This condition may be expressed as follows:

$$19\ (1w) = 20(0w)$$

The two patches are not necessarily continuous in slope across their common boundary; that is, the partial derivative with respect to u may differ when considering each patch equation. In the shorthand symbolism

$$uw_u = \frac{\partial(uw)}{\partial u}$$

is the partial derivative with respect to u. Coons has shown that by imposing certain contraints on the blending functions, the derivatives across the boundary in the u direction depend only upon the derivatives of the end points of the boundary. That is, if

$19(10)u = 20(00)u$ and

$19(11)u = 20(01)u$, then

$19(1w)u = 20(0w)u$

That is to say, if the boundary curves are continuous in slope in the u direction at the end points, the slopes in the u direction are continuous everywhere along the common boundary of the end points.

tangent vector

Our method for obtaining the required tangent vectors on a surface is based on the method of South and Kelly,[9] which consists

of generating cubic space curves through the given input points such that the required continuity is achieved between adjoining curves. (A similar strategy is employed by Ahuja to develop spline-like curves using rational polynomials.[10]) Referring to Figure 4, the technique involves equating the second derivative of the $(j\text{-}1)$st curve at $\mathbf{P}_j$ with the jth curve second derivative at $\mathbf{P}_j$.

A space curve may be written in terms of parametric equations, where v is the single parameter, as follows:

$$x = x(v)$$

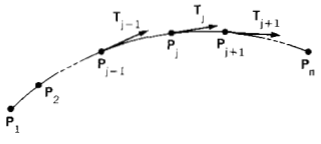
$$y = y(v)$$

$$z = z(v)$$

The notation may be simplified by denoting the functions as the following vector:

$$\mathbf{S}(v) = [x(v)\ y(v)\ z(v)]$$

Figure 4 Tangent vectors on a surface



A cubic space curve would have the following vector form:

$$\mathbf{S}(v) = \mathbf{A} + \mathbf{B}v + \mathbf{C}v^2 + \mathbf{D}v^3$$

where the vector coefficients are evaluated as in the following example:

$$\mathbf{A} = a_1 i + a_2 j + a_3 k$$

Referring again to Figure 4, the jth space curve and its derivative become

$$\mathbf{S}_j(v) = \mathbf{A}_j + \mathbf{B}_j v + \mathbf{C}_j v^2 + \mathbf{D}_j v^3$$

$$\mathbf{S}'_j(v) = \mathbf{B}_j + 2\mathbf{C}_j v + 3\mathbf{D}_j v^2$$

for v in the interval $(0,R_j)$. R_j represents the distance $\mathbf{P}_j$ to $\mathbf{P}_{j+1}$.

Evaluating these equations at the end points $\mathbf{P}_j$ and $\mathbf{P}_{j+1}$ yields

$$\mathbf{S}_j(0) = \mathbf{P}_j = \mathbf{A}_j$$

$$\mathbf{S}_j(R_j) = \mathbf{P}_{j+1} = \mathbf{A}_j + \mathbf{B}_j R_j + \mathbf{C}_j R_j^2 + \mathbf{D}_j R_j^3$$

$$\mathbf{S}'_j(0) = \mathbf{T}_j = \mathbf{B}_j$$

$$\mathbf{S}'_j(R_j) = \mathbf{T}_{j+1} = \mathbf{B}_j + 2\mathbf{C}_j R_j + 3\mathbf{D}_j R_j^2$$

$\mathbf{T}_j$ and $\mathbf{T}_{j+1}$ are tangents to the curve at $\mathbf{P}_j$ and $\mathbf{P}_{j+1}$ respectively.

Solving for the coefficients we have

$$\mathbf{A}_j = \mathbf{P}_j$$

$$\mathbf{B}_j = \mathbf{T}_j$$

$$\mathbf{C}_j = \frac{3}{R_j^2}(\mathbf{P}_{j+1} - \mathbf{P}_j) - \frac{1}{R_j}(\mathbf{T}_{j+1} + 2\mathbf{T}_j)$$

$$\mathbf{D}_j = -\frac{2}{R_j^3}(\mathbf{P}_{j+1} - \mathbf{P}_j) + \frac{1}{R_j^2}(\mathbf{T}_{j+1} + \mathbf{T}_j)$$

Now consider the pair of curves $\mathbf{S}_{j-1}$ and $\mathbf{S}_j$ with the common end point $\mathbf{P}_j$. Here the slope of curve $\mathbf{S}_{j-1}$ (evaluated at $v = R_{j-1}$) is T_j as is the slope of $\mathbf{S}_j$ for $v = 0$. The constraint to be imposed on the curves is that the first and second derivatives across the point $\mathbf{P}_j$ be continuous. The second derivatives are expressed as follows:

$$\mathbf{S}''_{j-1}(v) = 2\mathbf{C}_{j-1} + 6\mathbf{D}_{j-1}v \text{ for } v \text{ in interval } [0,R_{j-1}]$$

$$\mathbf{S}''_j(v) = 2\mathbf{C}_j + 6\mathbf{D}_j v \text{ for } v \text{ in interval } [0,R_j]$$

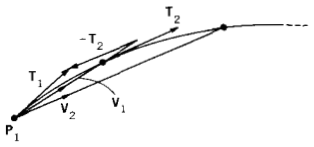
Evaluating these equations at $\mathbf{P}_j$ and setting the results equal, we obtain the following difference equation:

$$\mathbf{C}_j = \mathbf{C}_{j-1} + 3\mathbf{D}_{j-1}R_{j-1}$$

when the values previously determined for the coefficients of $\mathbf{C}_j$, $\mathbf{C}_{j-1}$, and $\mathbf{D}_{j-1}$ are substituted in the equation, simplification yields the following equation:

$$R_j\mathbf{T}_{j-1} + 2(R_j + R_{j-1})\,\mathbf{T}_j + R_{j-1}\mathbf{T}_{j+1} = 3[R_{j-1}^2(\mathbf{P}_{j+1} - \mathbf{P}_j) + R_j^2\,(\mathbf{P}_j - \mathbf{P}_{j-1})]/R_jR_{j-1}$$

Figure 5 Tangent vector at a point



There are $n-2$ of these difference equations for $j = 2, 3, \cdots, n-1$. Thus, two more equations are required, namely, those that impose end conditions on the curves.

An arbitrary but useful scheme is used to obtain the tangent vector $\mathbf{T}_1$ at end point $\mathbf{P}_1$, which represents the slope at that point as shown in Figure 5.

A scalar value k is solved for such that $k = 2\,\mathbf{V1}\cdot\mathbf{V2}$, where **V1** is a unit vector along $\mathbf{P}_1\mathbf{P}_2$, and **V2** is a unit vector along $\mathbf{P}_1\mathbf{P}_3$. By vector subtraction we have

$$\mathbf{T}_1 = k\mathbf{V1} - \mathbf{T}_2$$

and

$$\mathbf{T}_1 + \mathbf{T}_2 = 2\mathbf{V1}\cdot\mathbf{V2V1}$$

$\mathbf{T}_2$ is approximately equal to **V2**. When $\mathbf{T}_2 = \mathbf{V2}$, the tangent vector $\mathbf{T}_1$ is a unit vector.

Similarly, an end condition equation may be developed for point $\mathbf{P}_n$.

There now exist n equations that may be solved for the n tangent vectors. Let

$$H_{1,1} = 1$$

$$H_{1,2} = 1$$

$$\mathbf{B}_1 = 2\,\mathbf{V1} \cdot \mathbf{V2V1}$$

$$H_{n-1,n} = 1$$

$$H_{n,n} = 1$$

$$\mathbf{B}_n = 2\,\mathbf{V3} \cdot \mathbf{V4V3}$$

and

$$H_{j,j-1} = R_j$$

$$H_{j,j} = 2(R_j + R_{j-1})$$

$$H_{j,j-1} = R_{j-1}$$

$$B_j = \frac{3[R_{j-1}{}^2(\mathbf{P}_{j+1} - \mathbf{P}_j) + R_j{}^2(\mathbf{P}_j - \mathbf{P}_{j-1})]}{R_j R_{j-1}}$$

for $j = 2, 3, 4, \cdots, n-1$

The equations may be expressed in matrix notation, so that, for $n = 5$, we have the following matrix equation.

$$\begin{bmatrix} H_{1,1} & H_{1,2} & 0 & 0 & 0 \\ H_{2,1} & H_{2,2} & H_{2,3} & 0 & 0 \\ 0 & H_{3,2} & H_{3,3} & H_{3,4} & 0 \\ 0 & 0 & H_{4,3} & H_{4,4} & H_{4,5} \\ 0 & 0 & 0 & H_{5,4} & H_{5,5} \end{bmatrix} \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \\ \mathbf{T}_3 \\ \mathbf{T}_4 \\ \mathbf{T}_5 \end{bmatrix} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \mathbf{B}_3 \\ \mathbf{B}_4 \\ \mathbf{B}_5 \end{bmatrix}$$

The tridiagonal matrix on the left may be converted to an upper triangular matrix and the tangent vectors solved by backward substitution.

The tangent vectors are approximately unit vectors that must be modified by a magnitude that is the minimum of the distances to the two adjacent points. The magnitude of an end-point tangent vector is chosen as the distance from the end point to its adjacent neighbor. The slopes of the tangent vectors provide the directions of the curve at the points, and the magnitudes influence the shapes of the curves. By choosing the lesser of the distances to adjacent points, undesired loops or bulges are eliminated—although a flattening of the curve may result. For best results, the point inputs should be approximately equally spaced or changing moderately with changes in curvature. For surface continuity, the slope entering a curve at a point must equal the slope leaving the point on the next curve, but the magnitudes may differ. The purpose of choosing a single vector is to minimize the amount of data that define the surface canonical form.

surface definition

The canonical form of the surface is given by the following quantities:

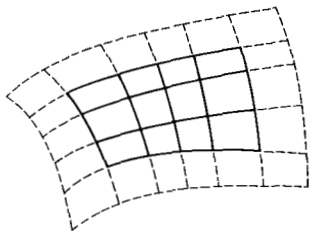
- Two integer words that provide values for the number of curves (m) and the number of points per curve (n)
- The surface points
- The w-curve tangent vectors
- The u-curve tangent vectors

When a particular patch equation is to be evaluated, the quantities m and n are used to extract the boundary-condition matrix elements from the canonical-form data array. The MBM^t matrix multiplication results in forty-eight equation coefficients, sixteen for each of the x, y, and z coordinates to be computed.

extended surface

Generally, a sculptured surface is usable as an APT surface only within the topographical area defined by its input. There are sometimes requirements for an APT numerical control programmer to machine to the edge of a surface or even beyond. Therefore, a special case has been devised to extend the surface on all sides. Figure 6 illustrates such a surface, where the solid lines represent the defined surface and the dashed lines denote the extended portion. The extension is accomplished by generating new point coordinates, hence, four new boundary curves. The new points are a result of the extrapolation of existing curves.

Figure 6 An extended surface



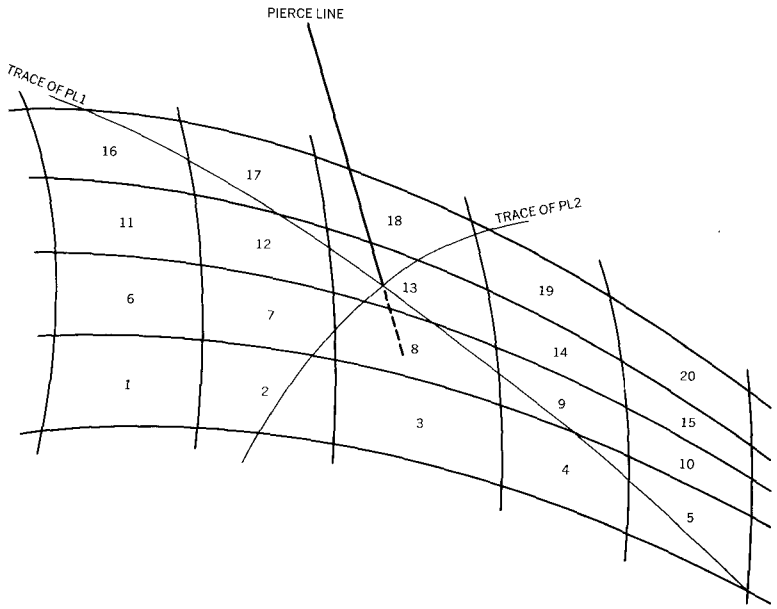
APT ARELEM considerations

To add a new surface type to the APT processor Arithmetic Element (ARELEM), it is necessary to provide directed distance S and surface normal **SN** calculation routines for the surface. The routines are independent of basic APT ARELEM except for the input and output conditions imposed by the ARELEM design. Of course, provisions must be made in ARELEM for dispatching to the new routines.

The primary problem to be solved is that of locating the pierce point **SP** of the surface by a line that contains the tool point **TP** and parallels the tool normal **TN** vector, all in the relationship shown in Figure 1. When the solution is found, the directed distance S and surface normal **SN** at **SP** are calculated.

Before the surface and line intersection can be computed, it is necessary to determine which surface patch contains the point. If ARELEM is in a start-up mode, that is, an APT GO/ command is being processed, a search of the surface is conducted to select candidate patches. This is done by listing in two sets the patches intersected by two planes that contain the pierce line, and isolating the patches common to the two sets. Figure 7 illustrates the two planes intersecting a surface with patches 13 and 18 as candidate patches. The selected patches are solved in sequence for the point of intersection, and patch 13 in Figure 7 contains the

Figure 7 Locating the surface-line pierce point



correct solution. Also, the solution in terms of the parameters u and w must be within the patch boundaries, that is, $0 \leq u, w \leq 1$.

After a solution is obtained for a start-up condition, the patch number is saved for subsequent calls to the directed distance routines of the sculptured surface with the assumption that the next pierce line will intersect this patch or an adjacent neighbor.

Assume that a patch and two planes containing the pierce line have been selected. The intersection of the patch and planes is transformed into parametric space, which yields two simultaneous, nonlinear, bicubic equations in u and w. Thus if $\mathbf{P}(u,w) = [x(u,w)\ y(u,w)\ z(u,w)]$ is a representative surface point, then for a particular pair of u and w we obtain the following equations:

$$RX_1 = PL1_1 x(u,w) + PL1_2 y(u,w) + PL1_3 z(u,w) + PL1_4$$

$$RX_2 = PL2_1 x(u,w) + PL2_2 y(u,w) + PL2_3 z(u,w) + PL2_4$$

The values RX_1 and RX_2 signify a measure of the distances from the uw point to each of the planes $PL1$ and $PL2$. The RX_i (where $i = 1,2$) may be said to be a measure of error. If $RX_i \leq \varepsilon$, where ε is a preassigned small number, the solution has been obtained; otherwise correction values are computed.

The RX_i are differentials of the function PLi of the following form:

$$RX_1 = du\,\frac{\partial RX_1}{\partial u} + dw\,\frac{\partial RX_1}{\partial w}$$

$$RX_2 = du\,\frac{\partial RX_2}{\partial u} + dw\,\frac{\partial RX_2}{\partial w}$$

The differential equations may also be expressed in the following m-matrix form:

$$\begin{bmatrix} \dfrac{\partial RX_1}{\partial u} & \dfrac{\partial RX_1}{\partial w} \\ \dfrac{\partial RX_2}{\partial u} & \dfrac{\partial RX_2}{\partial w} \end{bmatrix} \begin{bmatrix} du \\ dw \end{bmatrix} = \begin{bmatrix} RX_1 \\ RX_2 \end{bmatrix}$$

If u_n and w_n are typically the nth trial values of u and w then

$u_{n+1} = u_n - du$

$w_{n+1} = w_n - dw$

become the new trial values.

The iteration usually succeeds in three and four trials if the point of convergence is within the patch boundaries, that is, $0 \leq u$, $w \leq 1$. Because of uncertainties of the patch configuration exterior to patch boundaries, there may be a divergence. In any case, a solution exterior to the patch is considered to be a failure, and another patch must be selected.

APT SCSURF definition formats

In the surface sculpturing program additions, referred to as SCSURF, there are five formats available to a numerical control programmer to define a SCSURF sculptured surface. As an example, let SRF1 = SCSURF/m, n, k (=1). In this case, the point input formats for sculpturing one surface are expressed in records as follows:

$x_1,\ y_1,\ z_1$

$x_2,\ y_2,\ z_2$

$\vdots\quad\ \vdots\quad\ \vdots$

$x_n,\ y_n,\ z_n$

$\vdots\quad\ \vdots\quad\ \vdots$

$x_{mn},\ y_{mn},\ z_{mn}$

This is the first format ($k = 1$), with m equal to the number of curves and n being the number of points per curve. The coordinates of input points numbering one through mn immediately follow the definition statement.

The second format ($k = 2$) is similar except that the tangent vectors are included. For example, the jth record would be as follows:

$x_j, y_j, z_j, ux_j, uy_j, uz_j, wx_j, wy_j, wz_j$

Formats three and four have been most frequently applied in our propeller sculpturing program because they use the inclusive subscript format of IBM APT to list previously defined and symbolically named points as surface points. This is particularly important because input parameters to the propeller sculpturing program had to be transformed in APT to coordinate values that represent surface points. Thus it is possible to go from the programmer-coded input parameters to the post-processed output for the machine tool controller without intermediate handling of data. An example definition format is the following statement:

SRF3 = SCSURF/$m,n,k, P(a,$ THRU, $b)$

Here $P(a,$ THRU, $b)$ represent an array of points $P(a)$ through $P(b)$, the number of points being equal to mn. When $k = 3$ the surface consists only of the area spanned by the input points. When $k = 4$, we have the extended surface previously discussed.

The fifth case permits the numerical control programmer to define a sculptured surface parallel to a previously defined surface. This has been used to machine an area that is parallel to the basic propeller blade surface. The format of an example statement is the following:

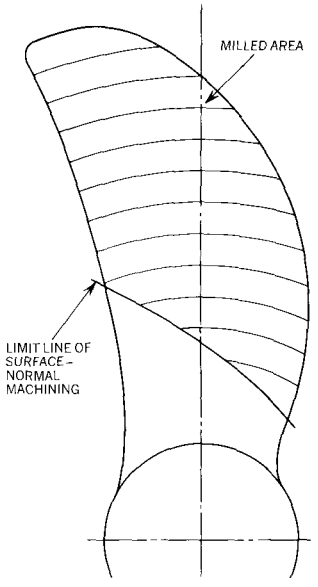
SRF5 = SCSURF/PARLEL, t, SRF3

This statement defines a surface parallel to a surface symbolized by SRF3 at a distance of t. The direction from SRF3 is determined by the direction of surface normal vectors of SRF3 and the sign given t.

One notable problem that complicates the writing of the propeller sculpturing program led to the defining of two additional APT geometric constructions using SCSURF. Propeller surfaces are specified to be machined by maintaining the milling cutter normal to the surface. This is not possible, however, where the cutter housing intersects an adjacent propeller blade. Further, propellers differ in number of blades and in individual surface configurations.

To generalize the calculations for the limiting line of surface-normal machining, a *point definition* is used. The surface normal at the defined point on the surface contains a point exterior to the surface. By using a series of points on an adjacent blade, the numerical control programmer can find surface points to define a space curve that becomes the machining limit, as shown in Figure 8. The point format is as follows:

Figure 8 Example surface-normal limiting curve



$PT1$ = POINT/$P1$, PERPTO, SRF1

where $P1$ is a point exterior to the named surface SRF1.

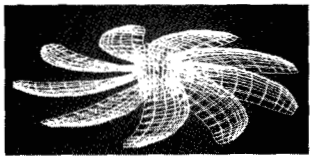
To determine an optimum tool-axis vector to machine the surface eclipsed by an adjacent blade, a *vector definition* is included in the expanded APT program to define surface normal vectors at given surface points. The format for the vector is as follows:

V1 = VECTOR/m, n, PERPTO, SRF3

where SRF3 is the symbolically named SCSURF, m is the curve number and n the point on curve m at which to compute the vector. One can see in Figure 9 eclipsed areas associated with adjacent propeller blades where limit lines for surface-normal machining must be established.

Experience and concluding remarks

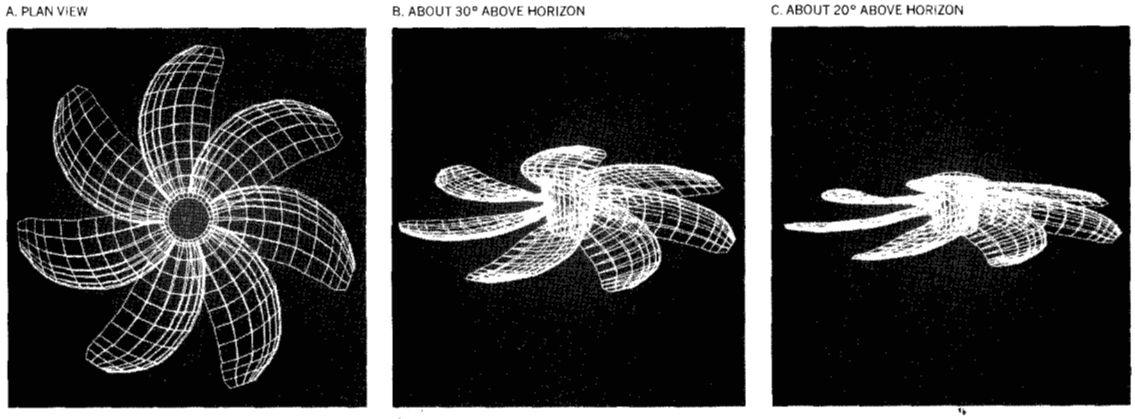
Figure 9 Graphic display showing areas of blade overlap



One of the limitations of the sculptured surface routines that are embodied in the IBM APT processor is the size of the input array and the canonical form of the surface, both of which must be compared with the expansion of other APT tables to satisfy the generalized sculpturing program requirements and a 360K storage region. For example, every input point is accepted as exact and only one pair of tangent vectors can be accommodated at each point. Thus far there has been no effort to allow defining data other than an ordered array of points. There is neither preprocessing to eliminate "wild points," nor averaging of point data. For the user whose data might be large in quantity and inexact in nature—having been extracted from a rough model, for example—it is presently necessary to preprocess that data through a user-developed program before its application to surface sculpturing. A similar confrontation exists for the user whose conceptual surface exists as a family of curves, whether planar or three-dimensional. Preprocessing is again necessary.

Thus far there has been little opportunity to compare processing time with other similar programs because none are available that possess similar programming attributes. Emperical evidence gathered in generating cutter path coordinates for machining propeller blades indicates a relatively slow processing. To the experienced numerical control programmer it may be of interest that, using a System/360 Model 65, three minutes of CPU time are required to process the program that mills the surface shown in Figure 8. Here approximately five hundred cutter offset positions are calculated. The surface under the adjacent blade is cut using a ball end mill and a fixed tool axis. ARELEM requires five minutes to generate approximately twelve hundred cutter positions.

Figure 10 Interactive displays using a single data base



Future considerations may include additional provisions for a variety of functions to suit the design needs of a skilled numerical control programmer. Other problems might include allowing for a *break line*, that is, a discontinuity of slope along a curve in a named surface. Another facility might be the blending of a sculptured surface with another surface. It might be useful to have a regional programming capability that utilizes a sculptured surface for permitting one APT language statement to describe a bounded surface area to be machined. Such a capability would have the advantage that the number of program statements and opportunities for human error would be reduced.

In pondering future application of sculptured surface technology in the framework of APT, we should perhaps consider the role of interactive graphics in numerical control. Although there have been a number of programs developed to design surfaces at a graphics terminal, the utilization of these programs is in the embryonic stage as they relate to APT. It appears reasonable, however, to project the development of data bases that interface between the computer graphics design of complex surfaces and an APT processor. Figure 10 shows three skeleton views of a seven-bladed propeller, all developed from the same data base. Although any other angle could have been produced at the IBM 2250 display terminal, the three shown here are: (A) plan view; (B) about thirty degrees above horizon; and, (C) about twenty degrees above the horizon.

The display programs and visual outputs were developed to confirm the designs and machinability of marine propellers. The obvious next step is to design propellers and other complex surfaces directly at an interactive graphics terminal and to apply the potency of APT to machining the surfaces.

The program extensions to the IBM System/360 APT processor described here may be applied to a variety of sculptured-surface machining problems. Maximum benefit is derived where a portion of a defined surface is to be machined, thus necessitating the sort of bounding capability found in using APT drive and check surfaces. We feel that the prime contribution of the program extensions to System/360 APT is the cutter path calculation, that is, the directed-distance and unit-normal computations. Further improvements might include expanding the surface-defining techniques, especially with regard to interactive-graphics-developed surfaces and the APT canonical form.

ACKNOWLEDGMENT

The author wishes to acknowledge the implementation of the APT extensions by T. J. Thometz and R. L. Stormberg, and the program testing by R. J. Roth, F. M. Balay, G. J. Vesely, and R. L. Steele. Programming and producing the propeller displays are the work of A. Appel of the IBM Research Division.

CITED REFERENCES AND FOOTNOTE

1. *IBM System/360 APT Numerical Control Processor, 360A-CN-10X, Version 4, Part Programming Manual*, Form GH20-0309-4, Fifth Edition (November 1970). International Business Machines Corporation, Data Processing Division, White Plains, New York 10604.
2. *IBM System/360 APT Numerical Control Processor, 360A-CN-10X, Version 4, Systems Manual*, Form GY20-0080-2, Third Edition (February 1969). International Business Machines Corporation, Data Processing Division, White Plains, New York 10604.
3. S. A. Coons, *Surfaces for Computer-Aided Design of Space Forms*, MAC-TR-41, Clearinghouse for Federal Scientific and Technical Information, Springfield, Virginia (June 1967).
4. Examples are FMILL/APTLFT and GEMESH available from ALRP APT Library, IIT Research Institute, 10 West 35 Street, Chicago, Illinois 60616.
5. D. V. Ahuja and S. A. Coons, "Geometry for construction and display," *IBM Systems Journal* **7**, Nos. 3 and 4, 188–205 (1968).
6. K. J. MacCallum, "Surfaces of interactive graphical design," *The Computer Journal* **13**, No. 4 (November 1970).
7. P. Bézier, "Procédé de définition numérique des courbes et surfaces non mathématiques," *Automatisme* **XIII**, No. 5 (May 1968).
8. A. L. Eshleman and H. D. Meriwether, "Graphic applications to aerospace structural design problems," McDonnell Douglas Corporation. Presented at the SHARE 4th Annual Design Automation Workshop, Los Angeles, California (June 21, 1967).
9. J. P. Kelly and N. E. South, *Analytic Surface Methods Numerical Control Development Unit*, Ford Motor Company, Dearborn, Michigan (December 1965).
10. D. V. Ahuja, "An algorithm for generating spline-like curves," *IBM Systems Journal* **7**, Nos. 3 and 4, 206–217 (1968).