Intersection of B-spline Surfaces By Elimination Method

by

Chee Kiang Wong

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

APPROVED:

Dr. Arvid Myklebust, Chairman

Dr. Sankar Jayaram

Dr. Robert L. West

September 24, 1990

Blacksburg, Virginia

Intersection of B-spline Surfaces By Elimination Method

by

Chee Kiang Wong

Dr. Arvid Myklebust, Chairman

Mechanical Engineering

(ABSTRACT)

Parametric surface representations such as the B-spline and Bezier geometries are widely used among the aerospace, automobile, and shipbuilding industries. These surfaces have proven to be very advantageous for defining and combining primitive geometries to form complex models. However, the task of finding the intersection curve between two surfaces has remained a difficult one. Presently, most of the research done in this area has resulted in various subdivision techniques. These subdivision techniques are based on approximations of the surface using planar polygons. This thesis presents an analytical approach to the intersection problem. The approach taken is to approximate the B-spline surface using subsets such as the ruled surface. Once the B-spline surface has been simplified, elimination techniques which solve for the surface variables can be used to analytically determine the intersection curve between two B-spline surfaces.

Acknowledgements

I would like to express my sincere gratitude to my major advisor, Dr. Arvid Myklebust, for his support, guidance, encouragement, and patience throughout my graduate study. I also want to thank Dr. Robert L. West and Dr. Sankar Jarayam for serving on my graduate committee.

Special thanks goes to J. R. Gloudemans and Krishnan Kolady for their assistance in this research. Thanks also to Robert Jones for reviewing this manuscript. I am grateful to the rest of the ACSYNT team for their friendship and the fun times during the meetings.

Last and foremost, I want to thank my family who makes my dream come true.

Table of Contents

List of Illustrations

Chapter 1

Introduction

Parametric free-form surfaces are widely used in the aerospace, automobile, and shipbuilding industries. However, the problem of finding the curves of intersection remains a critical item in the modeling of these free-form surfaces when they are combined to form complex geometry. Also it is important to find the intersection curves between parametric surfaces for many meaningful CAD/CAM applications such as shape design, design of fillets, and generation of numerical controlled tool paths.

From the analytical point of view, the problem of intersection may be to find the roots of a high-order equation with a single degree of freedom. However, due to the complexity of the problem, some analytical methods such as algebraic and projective geometries or numerical methods such as Newton's iterative scheme, may fail mysteriously or be unmanageable in solving the intersection equations. Some ***divide-and-conquer*** algorithms and techniques have evolved recently for finding the intersection curve, in which the surfaces are approximated with planar polygons. The subdivision scheme used in these methods rely heavily on the type of surface. They face the problems of enormous numbers of subdivisions and reduced accuracy of the approximations due to the nature of the planar polygons.

1.1 Objectives

The objective of this thesis is to create a new algorithm which solves for the intersection curves between two bicubic B-spline surfaces. An analytical approach is taken to ensure the accuracy of the intersections, where an elimination method is used to solve systems of nonlinear equations in the process. B-spline ruled surface approximation is used to reduce the complexity of the problem, as well as to improve the surface approximations with fewer subdivisions.

1.2 Literature Review

There is a substantial amount of literature focusing on the intersection of surfaces. As noted by Faux and Pratt (1979), "the calculation of the intersection curve between two surfaces may be regarded as a problem of solving simultaneous (usually nonlinear) equations, or as a minimization problem. In dealing with the problem of solving simultaneous equations, an additional step constraint function is introduced, whereas with the minimization problem, a least-squares function is imposed. In both approaches, the system of equations is solved iteratively by the Newton-Raphson method."

Timmer (1977) utilizes a hunting grid on a primary surface to locate at least one starting point on each isolated loop of the total intersection curve. He then traces the loop across the region of the hunting grid stepwise using Newton's method.

Phillips and Odell (1984) displayed the intersection points of two implicitly defined surfaces by iteratively computing a system of ordinary differential equations to generate a solution trajectory. The trajectory starts at an arbitrary point. The successive points are displayed only after they are within a specified distance from the intersection, thus displaying the solution trajectory. Sabin (1976) uses the Cayley function to display the intersection curve of two quadric surfaces. The Cayley function is set up by implementing an eye position and a picture plane, this allows the repetitive and time consuming calculations to be done in the 2D picture plane in place of the 3D problem space.

Ocken, Schwartz and Sharir (1983) analyzed the problem of computing the intersection curve of two rational, quadric and algebraic surfaces. The surfaces are projectively transformed and normalized to a simple form in which the parameterization of the intersection curve can be easily obtained in homogeneous coordinates. Waggenspack (1987) generates a piecewise parametric approximation of a general degree algebraic intersection curve by performing coordinate transformation when the curve is of degree three or less. Curves of higher degree are first subdivided and approximated with a curve of degree three or less. Comba (1968) combines the algebraic surfaces into a single pseudocharacteristic function, and hence uses this function to detect any intersection region.

The following approaches center around a fundamental numerical approach for the intersection problem; namely subdivision, or the ***divide and conquer*** method. Lane and

Riesenfeld (1980) developed new algorithms for the evaluation and subdivision of B-spline and Bernstein curves and surfaces, allowing the intersection problem to be solved by planar approximation. Peng (1984) uses the same subdivision algorithm and planar approximation for his ***divide-intersect-stretch-divide again*** process to trace along the intersection curve of B-spline surfaces. Lasser (1986) as well as Aziz and Bata (1990) apply the subdivision and planar approximation to the intersection of Bernstein-Bezier surfaces. Carlson (1982) uses a modified Catmull recursive subdivision scheme to find the intersection of two bicubic patches, whereas Hanna, Abel and Greenberg (1983) store the points of the subdivided patches in look-up tables to detect the intersection points. Lee and Fredricks (1984) solve the intersection of a parametric surface and a plane utilizing curve/plane intersection algorithm; where the surface is subdivided recursively until exactly two boundaries of the subpatch intersect the plane, and hence provide two points on the whole intersection curve.

Chapter 2

Uniform Bicubic B-spline Surfaces

This chapter presents background information on uniform bicubic B-spline surfaces to which the intersection solution is to be applied. Also presented in this chapter is the description of the method for ruled surface approximation where it will be utilized in the intersection algorithm.

2.1 Modeling of B-spline Surfaces

A periodic, uniform cubic B-spline curve can be represented in matrix notation (Mortenson, 1985) as follows:

$$p_i(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} M \begin{bmatrix} p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \end{bmatrix} \qquad \begin{matrix} u \in [0, 1] \\ i \in [1 : n-2] \end{matrix} \tag{2.1}$$

where

$$M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \tag{2.2}$$

i denotes the curve segment number, n is the number of control points, and M is the universal transformation matrix.

The general matrix form of a periodic, uniform and bicubic B-spline surface that approximates an $(m+1) \times (n+1)$ rectangular array of control points is (Mortenson, 1985):

$$p_{st}(u, w) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} M \; P_{kl} \; M^T \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix} \qquad \begin{matrix} s \in [1{:}m-2] \\ t \in [1{:}n-2] \\ u,w \in [0, 1] \\ k \in [s-1{:}s+2] \\ l \in [t-1 : t+2] \end{matrix} \tag{2.3}$$

s and t identify a particular patch of the surface, k and l define the control points to be evaluated, and M is identical to the transformation matrix of a B-spline curve (Eq. 2.2).

2.2 Surface Inversion for B-spline Surfaces (single patch)

The application of matrix algebra allows the derivation of the periodic and bicubic B-spline surface inversion algorithm as seen below (Gandhi, 1989):

$$B = M^{-1}\ U^{-1}\ P_{st}\ W_{-1}\ (M^T)^{-1} \tag{2.4}$$

where

$$U = \begin{bmatrix} u_1^3 & u_1^2 & u_1 & 1 \\ u_2^3 & u_2^2 & u_2 & 1 \\ u_3^3 & u_3^2 & u_3 & 1 \\ u_4^3 & u_4^2 & u_4 & 1 \end{bmatrix} \qquad W = \begin{bmatrix} w_1^3 & w_2^3 & w_3^3 & w_4^3 \\ w_1^2 & w_2^2 & w_3^2 & w_4^2 \\ w_1 & w_2 & w_3 & w_4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

B is the control hull matrix (4 x 4) that approximates the surface defined by the sixteen points $P_{st}\,(4 \times 4)$. U and W are the matrix of parametric variables. Figure 1 on page 8 shows a B-spline surface with its control hull that approximates the given sixteen points in 3D space.

2.3 Construction of B-spline Ruled Surfaces

The general definition of a ruled surface is that for each point on the surface, there will be at least one straight line that passes through the point, while lying completely on the surface. Given two curves $p(u)$ and $q(u)$, a ruled surface can be constructed by

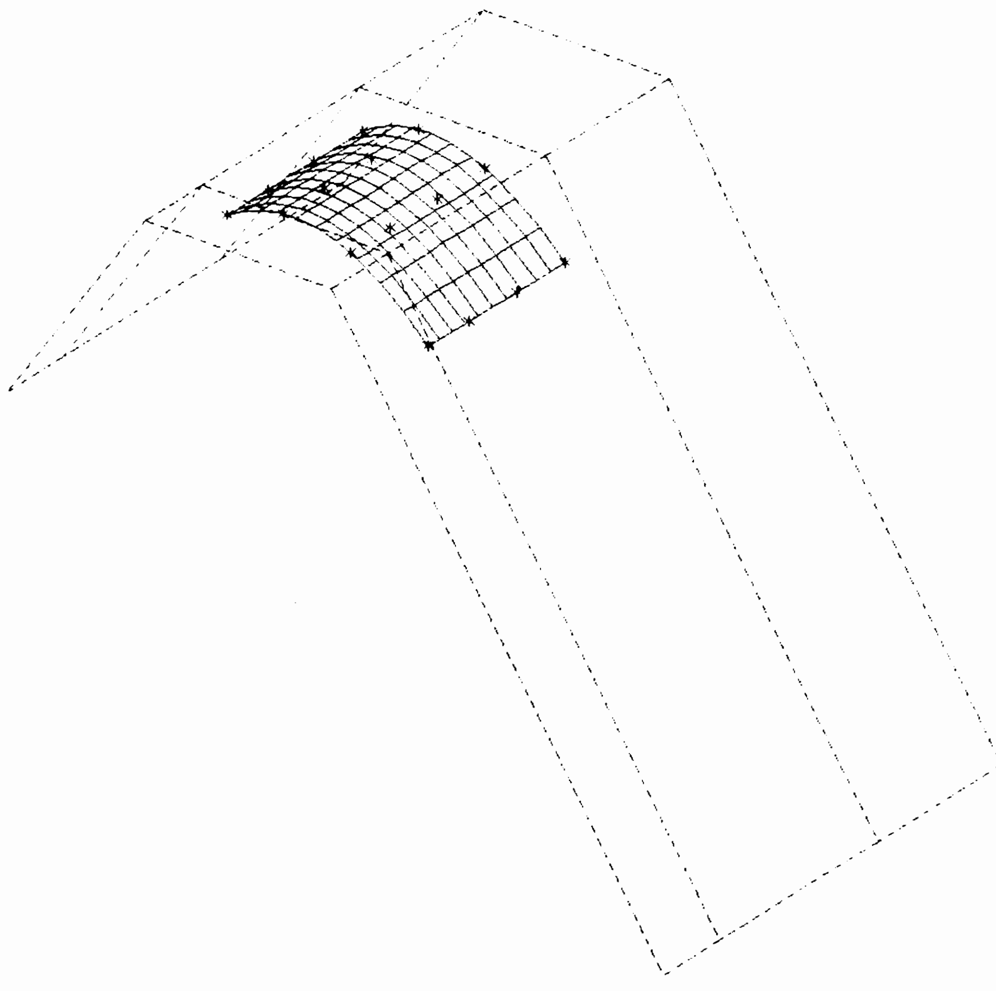

Figure 1. Inverse B-spline surface with the control hull: Dotted lines represent the control hull, whereas solid lines show the B-spline surface that approximates the marked sixteen points.

joining each point along the curves with a straight line. Similarly, a ruled surface can be built upon the boundary curves of a general surface by using the surface patch inversion algorithm discussed earlier. Figure 2 on page 10 gives an illustrative example. The surface matrix composed of sixteen points on the surface for this procedure is as follows:

$$\begin{bmatrix} s_{11} & a1s_{14}+(1-a1)s_{11} & a2s_{14}+(1-a2)s_{11} & s_{14} \\ s_{21} & a1s_{24}+(1-a1)s_{21} & a2s_{24}+(1-a2)s_{21} & s_{24} \\ s_{31} & a1s_{34}+(1-a1)s_{31} & a2s_{34}+(1-a2)s_{31} & s_{34} \\ s_{41} & a1s_{44}+(1-a1)s_{41} & a2s_{44}+(1-a2)s_{41} & s_{44} \end{bmatrix} \text{ or in the opposite direction}$$

$$\begin{bmatrix} s_{11} & s_{12} & s_{13} & s_{14} \\ a1s_{41}+(1-a1)s_{11} & a1s_{42}+(1-a1)s_{12} & a1s_{43}+(1-a1)s_{13} & a1s_{44}+(1-a1)s_{14} \\ a2s_{41}+(1-a2)s_{11} & a2s_{42}+(1-a2)s_{12} & a2s_{43}+(1-a2)s_{13} & a2s_{44}+(1-a2)s_{14} \\ s_{41} & s_{42} & s_{43} & s_{44} \end{bmatrix} \quad (2.5)$$

where $a1 = 1/3$, $a2 = 2/3$, and the parametric variables u and w will be set as $0, 1/3, 2/3$ and 1 respectively in the inversion algorithm for uniform B-spline ruled surface.

Note that the interior two points along a parametric direction are linearly and equally interpolated between the two end points. This will form a uniform B-spline ruled surface; where one of the surface parameters is linear.

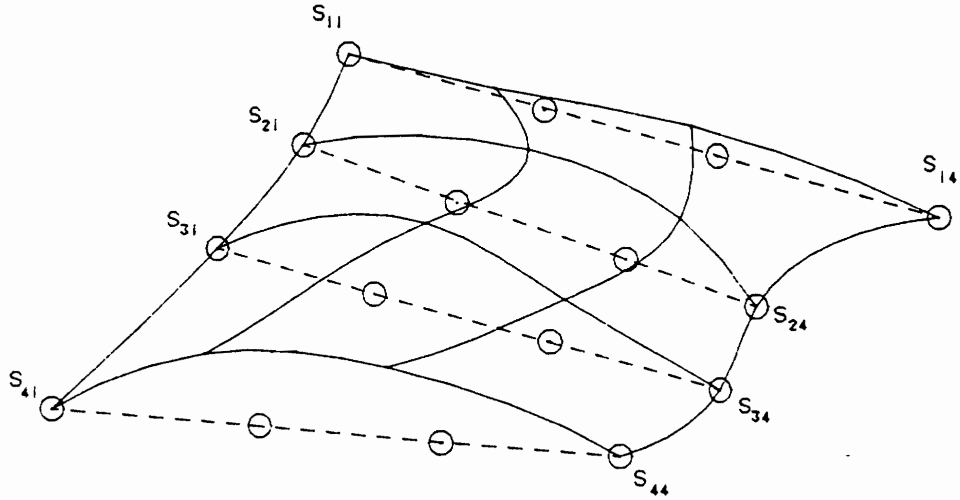

Figure 2. Construction of ruled surface: Solid lines represent the general surface, whereas dotted lines show the ruled surface.

2.4 Subdivision and Ruled Surface Approximation

Using the B-spline ruled surface construction method discussed earlier, two ruled surfaces can be built in both parametric directions from a general surface in order to start the approximation and comparison process. The criterion of comparison is based on the maximum distance between the lines on the ruled surface and the corresponding curves on the general surface (see Figure 3 on page 13). The lines and the curves are obtained by assigning values to one of the surface parameters; u or w, in the surface equations. Only three comparisons are made to reduce the processing time. This should be sufficient, especially for those approximations of fairly smooth or regular surfaces.

The maximum distance between a curve $p(u)$ and a line $l(t)$, where the line $l(t)$ is connected at both ends of the curve $p(u)$, is determined by finding a vector which is normal to the curve as well as to the line. Figure 4 on page 14 shows the vector geometry. Mathematically the normal vector intersects the curve $p(u)$ at q when:

$$\vec{p}^{\,u} \bullet \left[\vec{b1} \times \vec{b2}\right] = 0$$

$$\text{or} \quad \left[3\vec{a1}u^2 + 2\vec{a2}u + \vec{a3}\right] \bullet \left[\vec{b1} \times \vec{b2}\right] = 0 \qquad (2.6)$$

where

$$\vec{p}(u) = \vec{a1}u^3 + \vec{a2}u^2 + \vec{a3}u + \vec{a4}$$

$$\vec{l}(t) = \vec{b1}t + \vec{b2}$$

Similarly the normal vector intersects the line $l(t)$ at g when:

$$[\vec{g} - \vec{q}] \bullet \vec{b1} = 0 \tag{2.7}$$

So the maximum distance between the curve and the line is equal to the magnitude of the vector $[\vec{g} - \vec{q}]$.

The subdivision process utilized the B-spline surface inversion method discussed earlier. For example, in order to subdivide a B-spline surface in the parametric u direction, sixteen points are first obtained from the original B-spline surface by assigning u_1, u_2, u_3, u_4 as 0, 1/6, 1/3, 1/2, and w_1, w_2, w_3, w_4 as 0, 1/3, 2/3, 1, then the subsurface is generated by approximating these sixteen points using the surface inversion method.

The subdivisions and ruled surface approximations start in the parametric direction which has the smaller maximum difference in distance. For each iteration, the ***subsurface*** is subdivided into two parts and approximated using a B-spline ruled surface. The process terminates when the maximum distance between the surfaces is within a specified tolerance. The iteration process starts again with the remaining ***subsurface*** until the whole surface is approximated. Figure 5 on page 15 shows an example of the B-spline ruled surface approximations of a general B-spline surface. Notice that the ruled surfaces share common edges (C^0 continuity between the patches).

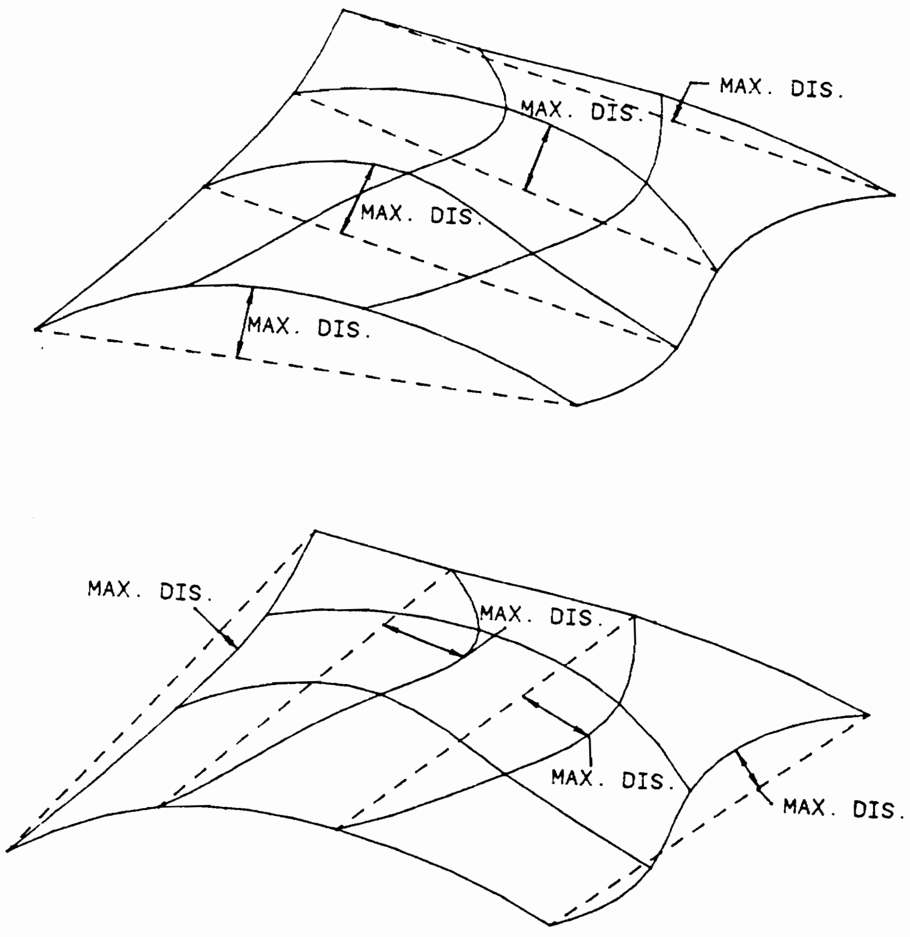

Figure 3. Ruled surface approximations in both parametric directions

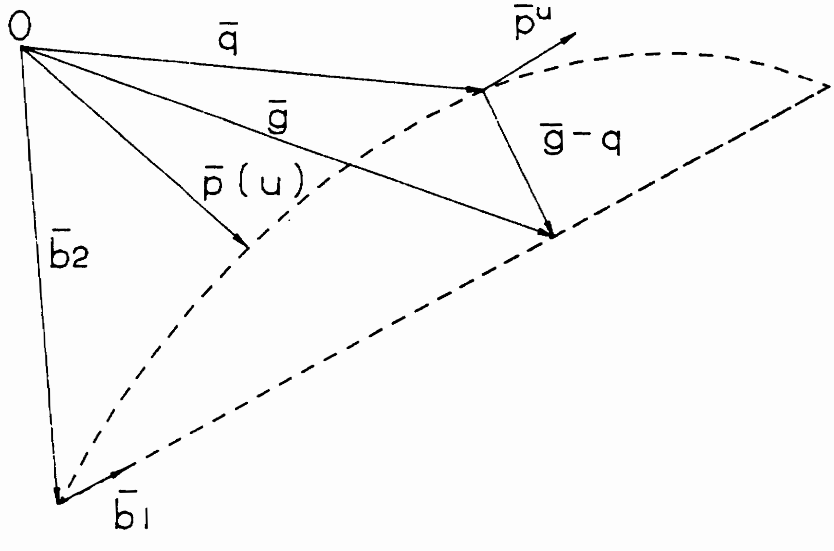

Figure 4. Vector geometry of the approximations

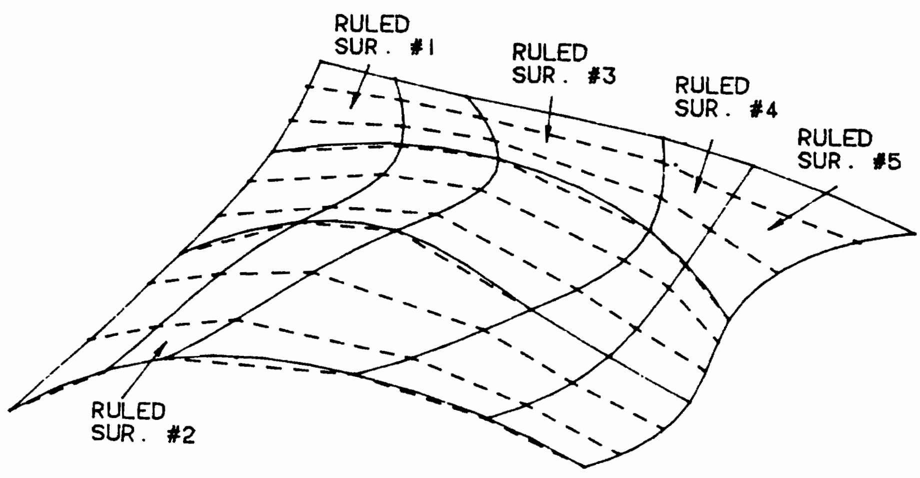

Figure 5. Ruled surface approximations with subdivided patches

Chapter 3

B-spline Surfaces Intersection

In this chapter, the basic idea that leads to the solution strategy of the intersection algorithm will be discussed. This is followed by the description of the subsets of B-spline surfaces and an outline of the intersection algorithm. Different cases of intersection which are solved by the process of elimination will be explained, and lastly the logic for selection of the intersection algorithms will be presented.

3.1 Geometric Interpretation of B-spline Surfaces Intersection

The underlying idea of most of the numerical ***divide and conquer*** algorithms for the intersection of B-spline or Bezier surfaces is to successively subdivide the surfaces until they can be approximated with planar polygons.

This solution method however, arose from the observations of the behavior of B-spline surface control volumes. It was noted that when two control volumes ***having fold lines*** intersected on the fold lines, the resulting intersection points defined a control polygon for an intersection curve between the two B-spline surfaces. ***Having fold lines*** means that in either parametric direction, each set of four control vertices for a bicubic B-spline patch are colinear. After many observations and a search for a general underlying principle, it was decided that general B-spline patches should be approximated by ruled surfaces, since the B-spline patches whose control hulls have fold lines are ruled surfaces. It was rightly assumed that eliminations would be relatively easy to accomplish analytically, since it was easy to observe the surface intersection curve by using intersection points of the control volumes. If the fold lines of the control volumes did not intersect, a linear proportion resulted in control vertices defining a curve which was offset from both surfaces but had some similarity to the intersection curve. The author suspects that additional investigation may lead to more significant results regarding the relationship between surface intersection curves and intersection in the control vertex space.

The resulting observations produced the following elimination cases. These cases are limited to a special class of B-spline surfaces in which the control hull matrix consists of columns of multiples of the first column, or rows of multiples of the first row as described. The control hulls intersect in such a way that the fold lines cross each other (see Figure 6 on page 19). The intersection curve is then constructed by using the intersection points of the fold lines as the control points for a cubic B-spline curve as described above (see Figure 7 on page 20). These discoveries indirectly lead to the solution strategies of this research in which the intersecting B-spline surfaces are approximated using subsets of B-spline surfaces. The approximations simplify the intersection problem to such a degree that they are manageable using the elimination method.

3.2 Subsets of B-spline Surfaces

The bicubic B-spline surface has the generic equation form:

$$\begin{aligned} P_{xyz}(u, w) = \; & C1_{xyz}u^3w^3 + C2_{xyz}u^3w^2 + C3_{xyz}u^3w + C4_{xyz}u^3 + \\ & C5_{xyz}u^2w^3 + C6_{xyz}u^2w^2 + C7_{xyz}u^2w + C8_{xyz}u^2 + \\ & C9_{xyz}uw^3 + C10_{xyz}uw^2 + C11_{xyz}uw + C12_{xyz}u + \\ & C13_{xyz}w^3 + C14_{xyz}w^2 + C15_{xyz}w + C16_{xyz} \end{aligned} \tag{3.1}$$

where $C_1, C_2, \ldots.C_{16}$ are dependent on the defining control hull and the blending functions, and u and w are the parametric variables of the surface.

Depending on the nature of the control hulls, the B-spline ruled surface constructed earlier can be broken into two special cases :

Subset 1 of B-spline Surfaces

This is a special case of surfaces where the control hull matrix consists of columns that are multiples of the first column, or rows that are multiples of the first row. This kind of surface will always lies parallel with one of the coordinate axis. Below are the examples of the control hull matrix of this special surface:

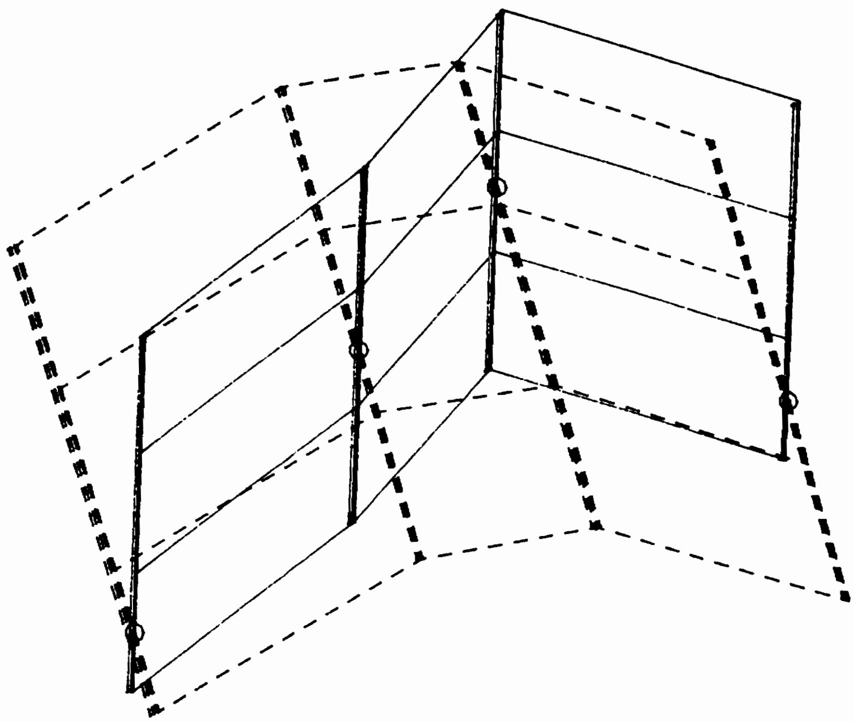

Figure 6. Intersection of two B-spline control hulls: The fold lines (heavier lines) cross each other at the marked points.

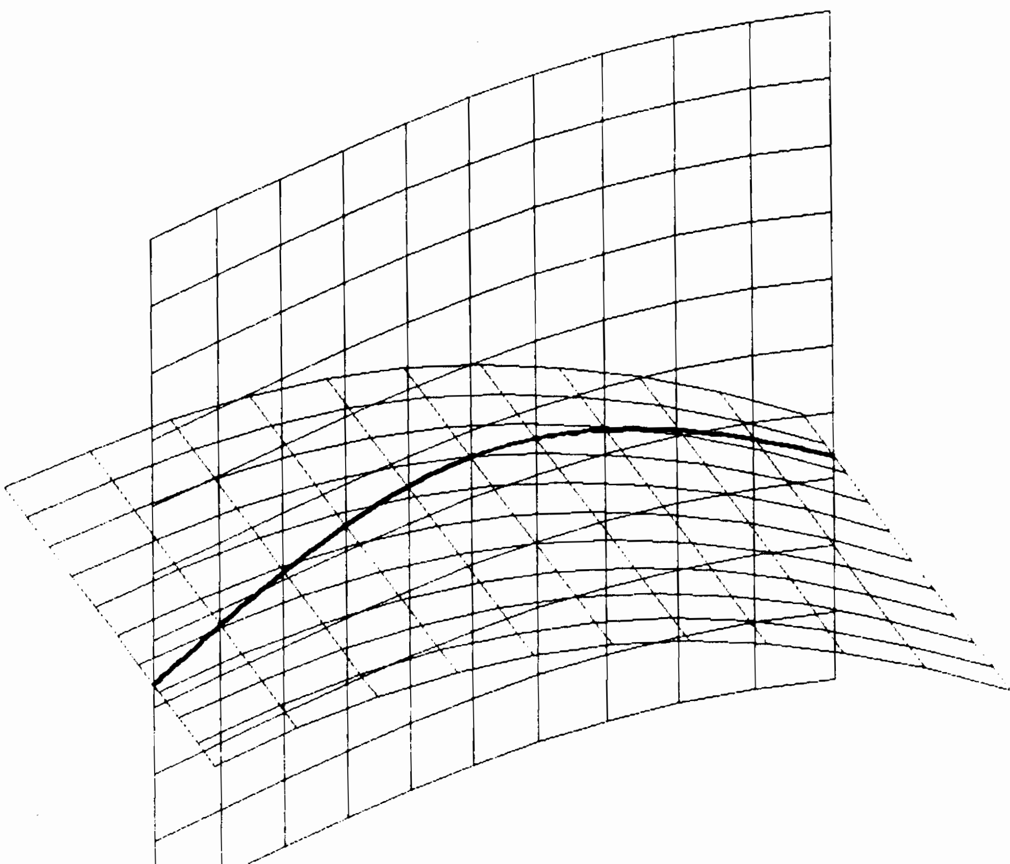

Figure 7. B-spline intersection curve: It is constructed using the intersection points of the fold lines (in Fig. 6) as the control points.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix} \tag{3.2}$$

and the examples of the surface equations:

$$\begin{aligned} P_x(u,w) &= a_x w^3 + b_x w^2 + c_x w + d_x \\ P_y(u,w) &= a_y w^3 + b_y w^2 + c_y w + d_y \\ P_z(u,w) &= a_z u^3 + b_z u^2 + c_z u + d_z \end{aligned} \tag{3.3}$$

where a, b, c and d are the surface coefficients. Notice that each scalar parametric equation is a function of just one parametric variable, and the arrangement of the variables in the equations is governed by the configuration of the control hull matrix.

Subset 2 of B-spline Surfaces

This subset is actually equivalent to the uniform B-spline ruled surface, where one of the parametric variables is linear in the surface equations. The control hull matrix has the same form and characteristics as the surface matrix in the B-spline surface inversion equation (see Eq. 2.5):

$$\begin{bmatrix} p_{11} & a1p_{14}+(1-a1)p_{11} & a2p_{14}+(1-a2)p_{11} & p_{14} \\ p_{21} & a1p_{24}+(1-a1)p_{21} & a2p_{24}+(1-a2)p_{21} & p_{24} \\ p_{31} & a1p_{34}+(1-a1)p_{31} & a2p_{34}+(1-a2)p_{31} & p_{34} \\ p_{41} & a1p_{44}+(1-a1)p_{41} & a2p_{44}+(1-a2)p_{41} & p_{44} \end{bmatrix} \text{ or in the opposite direction}$$

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ a1p_{41}+(1-a1)p_{11} & a1p_{42}+(1-a1)p_{12} & a1p_{43}+(1-a1)p_{13} & a1p_{44}+(1-a1)p_{14} \\ a2p_{41}+(1-a2)p_{11} & a2p_{42}+(1-a2)p_{12} & a2p_{43}+(1-a2)p_{13} & a2p_{44}+(1-a2)p_{14} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix} \quad (3.4)$$

where $a1 = 1/3$ and $a2 = 2/3$. The surface equations for this subset:

$$\begin{aligned} P_{xyz}(u,w) &= w\,(a_{xyz}u^3 + b_{xyz}u^2 + c_{xyz}u + d_{xyz}) + \\ &\quad (e_{xyz}u^3 + f_{xyz}u^2 + g_{xyz}u + h_{xyz}) \qquad \text{or} \\ P_{xyz}(u,w) &= u\,(a_{xyz}w^3 + b_{xyz}w^2 + c_{xyz}w + d_{xyz}) + \\ &\quad (e_{xyz}w^3 + f_{xyz}w^2 + g_{xyz}w + h_{xyz}) \end{aligned} \quad (3.5)$$

3.3 Algorithm for the Intersection of B-spline Surfaces

The intersection algorithm described here consists of the following steps :

1. Use the bounding box method as a preliminary intersection check to eliminate non-intersecting B-spline surface patches (two single patches at a time) as early as possible to avoid unnecessary computations in the algorithm. Bounding boxes are built by connecting the maximum and minimum values of x, y, z coordinates of the

corners of each patch, and the interior four points of the control hull (see Figure 8 on page 24). This will make sure the patch will be totally enclosed in the bounding box due to the convex hull property of B-spline surfaces (Peng, 1985).

2. Approximate and subdivide the intersecting B-spline patches (which have passed the bounding box check) with B-spline ruled surfaces as discussed in chapter 2. This generates a number of B-spline ruled surface patches for each intersecting B-spline patch.

3. Use bounding boxes again to check the intersections of the approximated ruled surfaces patches.

4. Solve ruled surface intersections using the elimination method (discussed later). The elimination begins along the edges of the ruled surface patches for two end points of the intersection curve. This is achieved by defining 0 or 1 to one of the surface parameters in the elimination algorithm, thus a total of eight eliminations will be carried out to find the two end points of the intersection curve (along four edges for each intersecting ruled surface patch).[1]

5. Find the intermediate intersection points by varying the parametric variable which has the largest difference between the end points. This will assure that all portions of the intersection curve are considered as shown in Figure 9 on page 25. This step generates a string of intersection points in global coordinates between two intersecting ruled surface patches.

---

1 The assumption that the intersection curves start along the edges is sufficient enough for general CAD applications, which usually need more than a patch for any geometry modelling. For a closed intersection curve where it does not cross the boundaries of the surfaces, it will still probably pass the edges of the patches.

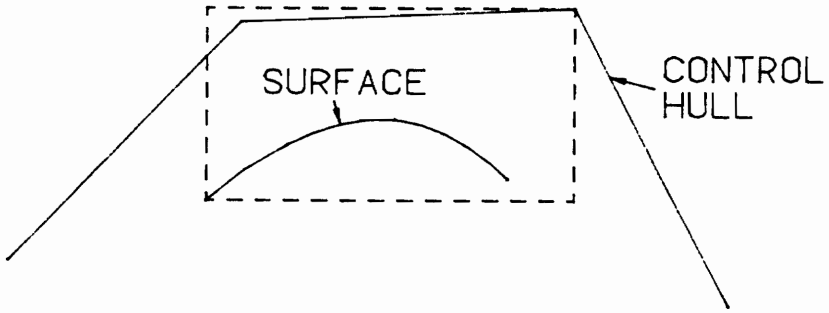

Figure 8. 2D View of the bounding box

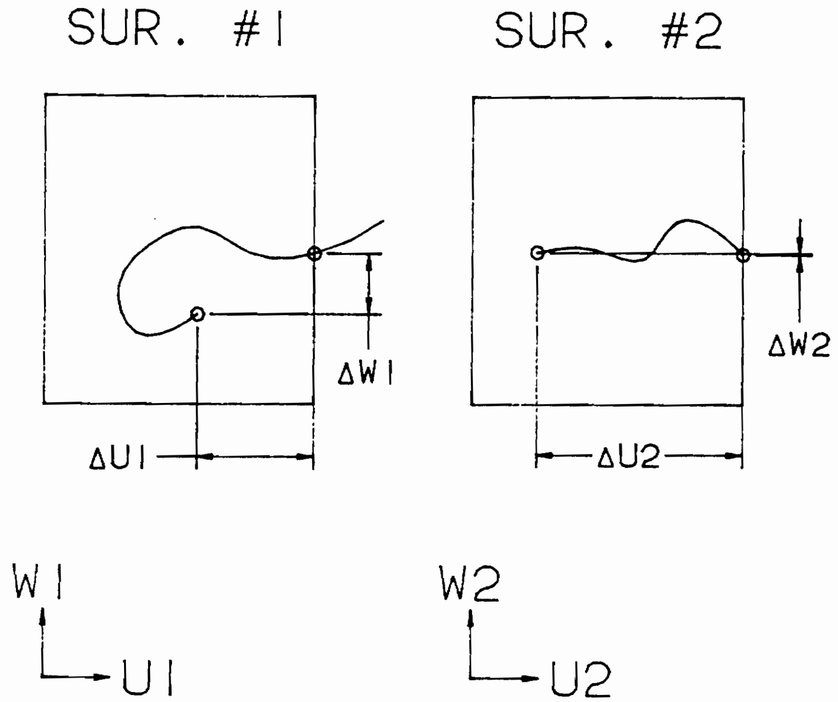

Figure 9. Selection of parametric variables for intermediate points: Only by varying $u2$ in the elimination algorithm will the whole intersection curve be covered.

6. Once all the approximated ruled surface patches of the two intersecting B-spline surfaces have been investigated, project the intersection points to the original B-spline surfaces in order to obtain the parametric values.

7. Sort these intersection points in correct order in parametric space. The algorithm produces a maximum of two curves. However it can be modified easily for a greater number of curves.

8. Interpolate the points to generate cubic B-spline curves in the original parametric space.

3.4 Cases of Intersection Using Elimination Method

The elimination algorithm contains the following steps:

1. Equate the surface parametric equations for the three cartesian coordinates of two approximated ruled surface patches.

2. Define one of the surface variables explicitly in the preceding equations.

3. Eliminate the surface variables to yield a high order polynomial. Depending on the orientations of the ruled surface patches, or the order of defining the surface variables, the polynomial can be of order 3, 6, 12 or 33. For all cases, the intersection problems are reduced to an implicit function of one of the surface variables.

4. Solve for the roots of the resultant polynomial.

5. The number of real roots that are obtained from the preceding procedure will most likely be greater than one, thus a screening scheme is required to select the correct root. Only one root is needed from the polynomial by assuming that only one intersection point is solved for each elimination process.

6. Screening scheme: All the real roots are substituted back into the original intersection equations in Step 1 to obtain the surface variables. Keep those roots which yield surface variables that are within the range of 0 and 1. Invert the resultant surface variables to points in global coordinates and select the pair of points which are closest to each other. This generates one intersection point on each approximated ruled surface patch in global coordinates.

Depending on the subset of B-spline surfaces the algorithm is applied to, the elimination algorithm can be divided into three categories. For each category, there will be different cases due to the orientations of the surfaces with respect to each other, or the order of defining the surface variables in the algorithm.

In the following discussion of subset 1 surface, the surface variable that appears twice in the surface parametric equations (Eq. 3.3) is denoted as the repeated variable, and the other as the singular variable. Likewise for subset 2 surface, the two surface variables are distinguished by referring them as cubic and linear variables (refer to Eq. 3.5). This naming scheme is to make the following discussion clearer and more understandable. Also the arrangements or the occurrences of the surface variables in the following examples will most likely be different from real applications.

Category 1

The following is a description of the intersection of two subset 1 surfaces. Based on the orientations of the surfaces, the problem can be broken into two cases:

a) Case 1 of Category 1

This is the case where the fold lines of control hull of the intersecting surfaces are parallel (see Figure 10 on page 29). Derive the intersection equations as follows:

$$a_x w_1^3 + b_x w_1^2 + c_x w_1 + d_x = a'_x u_2^3 + b'_x u_2^2 + c'_x u_2 + d'_x \tag{3.6}$$

$$a_y w_1^3 + b_y w_1^2 + c_y w_1 + d_y = a'_y u_2^3 + b'_y u_2^2 + c'_y u_2 + d'_y \tag{3.7}$$

$$a_z u_1^3 + b_z u_1^2 + c_z u_1 + d_z = a'_z w_2^3 + b'_z w_2^2 + c'_z w_2 + d'_z \tag{3.8}$$

The singular variables (u_1, w_2) are evaluated by first defining one of the two variables explicitly in Eq. 3.8, then solving for the corresponding variable from the remaining cubic function. Solving for the repeated variables (w_1, u_2) eliminates one of the variables from Eqs. 3.6 and 3.7 to yield a twelfth order polynomial (see Appendix A).

b) Case 2 of Category 1

This is the case where the fold lines of the control hull of the surfaces are not parallel (see Figure 11 on page 31). Compute the intersection equations as follows:

$$a_x w_1^3 + b_x w_1^2 + c_x w_1 + d_x = a'_x u_2^3 + b'_x u_2^2 + c'_x u_2 + d'_x \tag{3.9}$$

$$a_y w_1^3 + b_y w_1^2 + c_y w_1 + d_y = a'_y w_2^3 + b'_y w_2^2 + c'_y w_2 + d'_y \tag{3.10}$$

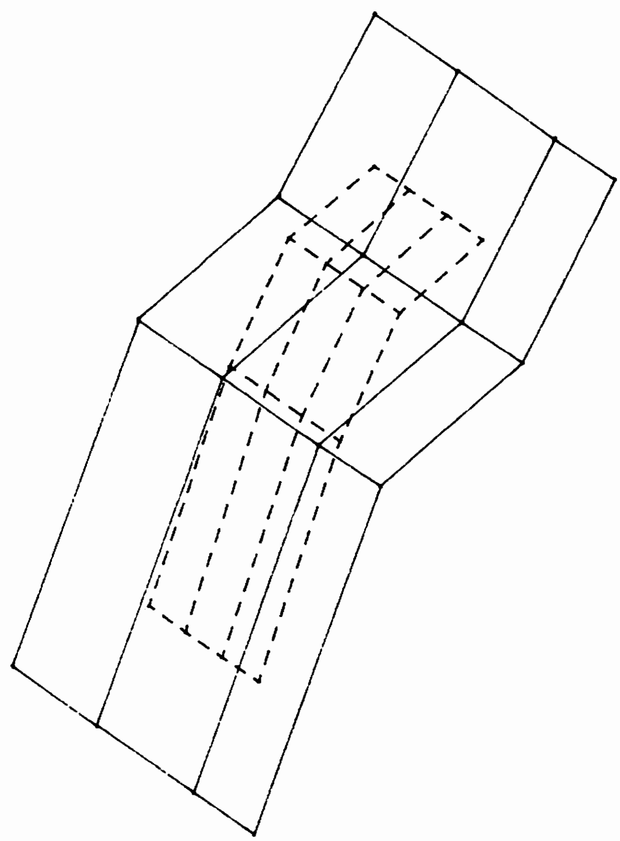

Figure 10. Control hulls of two intersecting subset 1 surfaces - 1: Parallel fold lines

$$a_z u_1^3 + b_z u_1^2 + c_z u_1 + d_z = a'_z w_2^3 + b'_z w_2^2 + c'_z w_2 + d'_z \tag{3.11}$$

The variables are evaluated by first defining any one of the variables explicitly, rearranging the equations, and finally solving the resultant cubic functions. For example, define w_1 in Eqs. 3.9 and 3.10 yield two cubic functions of u_2 and w_2. Solve the resultant cubic functions and eliminate w_2 in Eq. 3.11 result a cubic function of u_1. Since the problem only involves cubic functions, the intersection points are obtained in closed form.

Category 2

The following is a description of the intersection of two subset 2 surfaces. Equate parametric equations of the surfaces for the three cartesian coordinates as follows:

$$X = w_1 X_1(u_1) + X_2(u_1) = u_2 X_a(w_2) + X_b(w_2) \tag{3.12}$$

$$Y = w_1 Y_1(u_1) + Y_2(u_1) = u_2 Y_a(w_2) + Y_b(w_2) \tag{3.13}$$

$$Z = w_1 Z_1(u_1) + Z_2(u_1) = u_2 Z_a(w_2) + Z_b(w_2) \tag{3.14}$$

where $X_1, X_2, X_a, X_b, Y_1, Y_2, Y_a, Y_b, Z_1, Z_2, Z_a$ and Z_b are the cubic functions of the associated variables (refer to Eq. 3.5).

Depending on which variable is defined explicitly, the problems of intersection can be broken into two cases:

a) Case 1 of Category 2

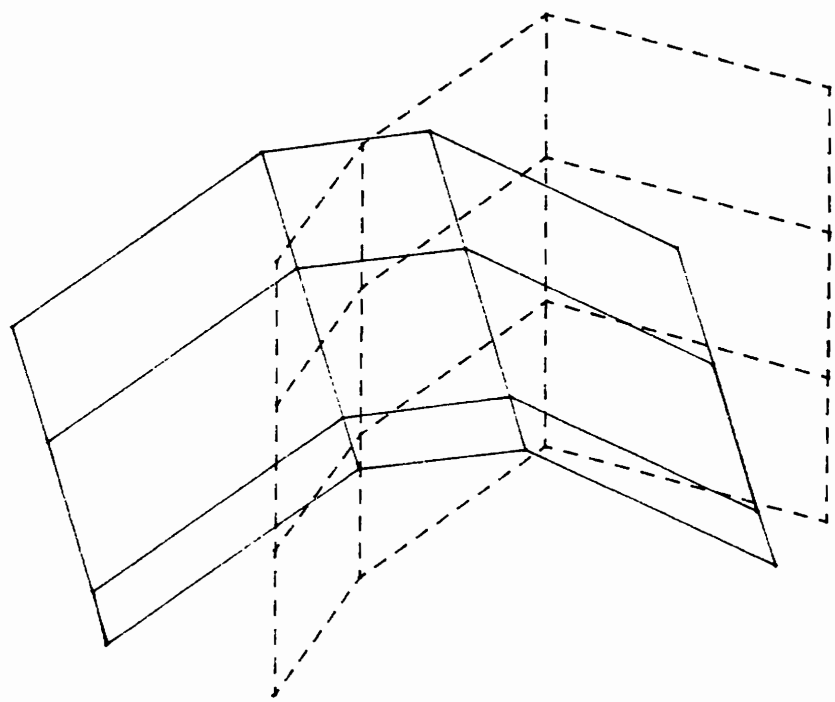

Figure 11. Control hulls of two intersecting subset 1 surfaces - 2: Non-parallel fold lines

This is the case where the cubic variable (u_1) is defined explicitly. First of all, rearrange the linear variable (w_1, u_2) on both intersecting surfaces by using one of the surface parametric expressions (Eq. 3.12 is used for this step):

$$w_1 = \frac{X - X_2(u_1)}{X_1(u_1)} \tag{3.15}$$

$$u_2 = \frac{X - X_b(w_2)}{X_a(w_2)} \tag{3.16}$$

Notice that the criterion from which the parametric expression is used to derive Eq. 3.15 is dependent upon the magnitude of the denominator; where the largest of $X_1(u_1)$, $Y_1(u_1)$ and $Z_1(u_1)$ is selected to avoid any ***singular*** or ***ill-conditioned*** situation in the algorithm. Eliminate w_1 and u_2 from Eqs. 3.13 and 3.14 by using Eqs. 3.15 and 3.16 yield:

$$\frac{X - X_2(u_1)}{X_1(u_1)} Y_1(u_1) + Y_2(u_1) = \frac{X - X_b(w_2)}{X_a(w_2)} Y_a(w_2) + Y_b(w_2) \tag{3.17}$$

$$\frac{X - X_2(u_1)}{X_1(u_1)} Z_1(u_1) + Z_2(u_1) = \frac{X - X_b(w_2)}{X_a(w_2)} Z_a(w_2) + Z_b(w_2) \tag{3.18}$$

Define u_1 in Eqs. 3.17 and 3.18:

$$C_1 X + C_2 = \frac{X - X_b(w_2)}{X_a(w_2)} Y_a(w_2) + Y_b(w_2) \tag{3.19}$$

$$K_1 X + K_2 = \frac{X - X_b(w_2)}{X_a(w_2)} Z_a(w_2) + Z_b(w_2) \tag{3.20}$$

where C_1, C_2, K_1 and K_2 are the values of the defined functions. Finally eliminating X from Eqs. 3.19 and 3.20 will generate a sixth order polynomial (see Appendix B).

b) Case 2 of Category 2

This is the case where the linear variable (w_1) is defined explicitly. Define w_1 in the intersection Eqs. 3.12, 3.13 and 3.14:

$$X_3(u_1) = u_2 X_a(w_2) + X_b(w_2) \tag{3.21}$$

$$Y_3(u_1) = u_2 Y_a(w_2) + Y_b(w_2) \tag{3.22}$$

$$Z_3(u_1) = u_2 Z_a(w_2) + Z_b(w_2) \tag{3.23}$$

where X_3, Y_3 and Z_3 are the evaluated cubic functions of u_1. Select any one of the Eqs. 3.21, 3.22 or 3.23 to rearrange the cubic variable u_2 (Eq. 3.21 is used for this step):

$$u_2 = \frac{X_3(u_1) - X_b(w_2)}{X_a(w_2)} \tag{3.24}$$

Eliminate u_2 from Eqs. 3.22 and 3.23 by using Eq. 3.24, yielding two equations with two unknowns:

$$Y_3(u_1) = \frac{X_3(u_1) - X_b(w_2)}{X_a(w_2)} Y_a(w_2) + Y_b(w_2) \tag{3.25}$$

$$Z_3(u_1) = \frac{X_3(u_1) - X_b(w_2)}{X_a(w_2)} Z_a(w_2) + Z_b(w_2) \tag{3.26}$$

Further elimination of u_1 will generate a thirty third order polynomial of w_2 (see Appendix C).

Category 3

The following is a description of the intersection of subset 1 and subset 2 surfaces. Equate the surface parametric equations for the three cartesian coordinates:

$$a_x w_1^3 + b_x w_1^2 + c_x w_1 + d_x = u_2 X_a(w_2) + X_b(w_2) \tag{3.27}$$

$$a_y w_1^3 + b_y w_1^2 + c_y w_1 + d_y = u_2 Y_a(w_2) + Y_b(w_2) \tag{3.28}$$

$$a_z u_1^3 + b_z u_1^2 + c_z u_1 + d_z = u_2 Z_a(w_2) + Z_b(w_2) \tag{3.29}$$

where the left and right hand sides of the equations are the expressions of subset 1 and subset 2 surfaces respectively. X_a, X_b, Y_a, Y_b, Z_a and Z_b are the cubic functions of the associated variables (refer to Eqs. 3.3 and 3.5).

This category can be broken down into four cases as the variables w_1, u_1, w_2 and u_2 are defined explicitly and separately in the elimination algorithm.

a) Case 1 of Category 3

This is the case where the repeated variable (w_1) is defined explicitly. Define w_1 in the Eqs. 3.27 and 3.28:

$$C_1 = u_2 X_a(w_2) + X_b(w_2) \tag{3.30}$$

$$C_2 = u_2 Y_a(w_2) + Y_b(w_2) \tag{3.31}$$

where C_1 and C_2 are the values of the defined functions. Eliminate u_2 from Eqs. 3.30 and 3.31 to obtain a sixth order polynomial of w_2. The elimination process is similar to the example in case 1 of category 2 (refer to Eqs. 3.19 and 3.20). Solving the polynomial and performing the necessary back substitutions will eventually lead to the solutions of all the variables.

b) Case 2 of Category 3

This is the case where the singular variable (u_1) is defined explicitly. Define u_1 in Eq. 3.29 and rewrite the intersection expressions:

$$a_x w_1^3 + b_x w_1^2 + c_x w_1 + d_x = u_2 X_a(w_2) + X_b(w_2) \tag{3.27}$$

$$a_y w_1^3 + b_y w_1^2 + c_y w_1 + d_y = u_2 Y_a(w_2) + Y_b(w_2) \tag{3.28}$$

$$C_3 = u_2 Z_a(w_2) + Z_b(w_2) \tag{3.32}$$

where C_3 is the value of the defined function. The Eqs. 3.27, 3.28 and 3.32 have the same form as the problems in case 2 of category 2. The procedure of solutions is the same as that particular case (refer to Eqs. 3.21, 3.22 and 3.23).

c) Case 3 of Category 3

This is the case where the cubic variable (w_2) is defined explicitly. Define w_2 in Eqs. 3.27, 3.28 and 3.29:

$$a_x w_1^3 + b_x w_1^2 + c_x w_1 + d_x = E_a u_2 + E_b \tag{3.33}$$

$$a_y w_1^3 + b_y w_1^2 + c_y w_1 + d_y = F_a u_2 + F_b \tag{3.34}$$

$$a_z u_1^3 + b_z u_1^2 + c_z u_1 + d_z = G_a u_2 + G_b \tag{3.35}$$

where E_a, E_b, F_a, F_b, G_a and G_b are the values of the defined functions. Elimination of u_2 from Eqs. 3.33 and 3.34 will create a cubic function of w_1. Solving this cubic function and carrying out the back substitutions will subsequently yield the solutions for all the variables.

d) Case 4 of Category 3

This is the case where the linear variable (u_2) is defined explicitly. Define u_2 in Eqs. 3.27, 3.28 and 3.29:

$$a_x w_1^3 + b_x w_1^2 + c_x w_1 + d_x = X_c(w_2) \tag{3.36}$$

$$a_y w_1^3 + b_y w_1^2 + c_y w_1 + d_y = Y_c(w_2) \tag{3.37}$$

$$a_z u_1^3 + b_z u_1^2 + c_z u_1 + d_z = Z_c(w_2) \tag{3.38}$$

where X_c, Y_c and Z_c are the cubic functions of w_2. Solving the Eqs. 3.36, 3.37 and 3.38 creates the same problem as the example in case 1 of category 1 (refer to Eqs. 3.6, 3.7 and 3.8).

Figure 12 on page 37 shows the logic for the selection of the intersection algorithms for the elimination process.

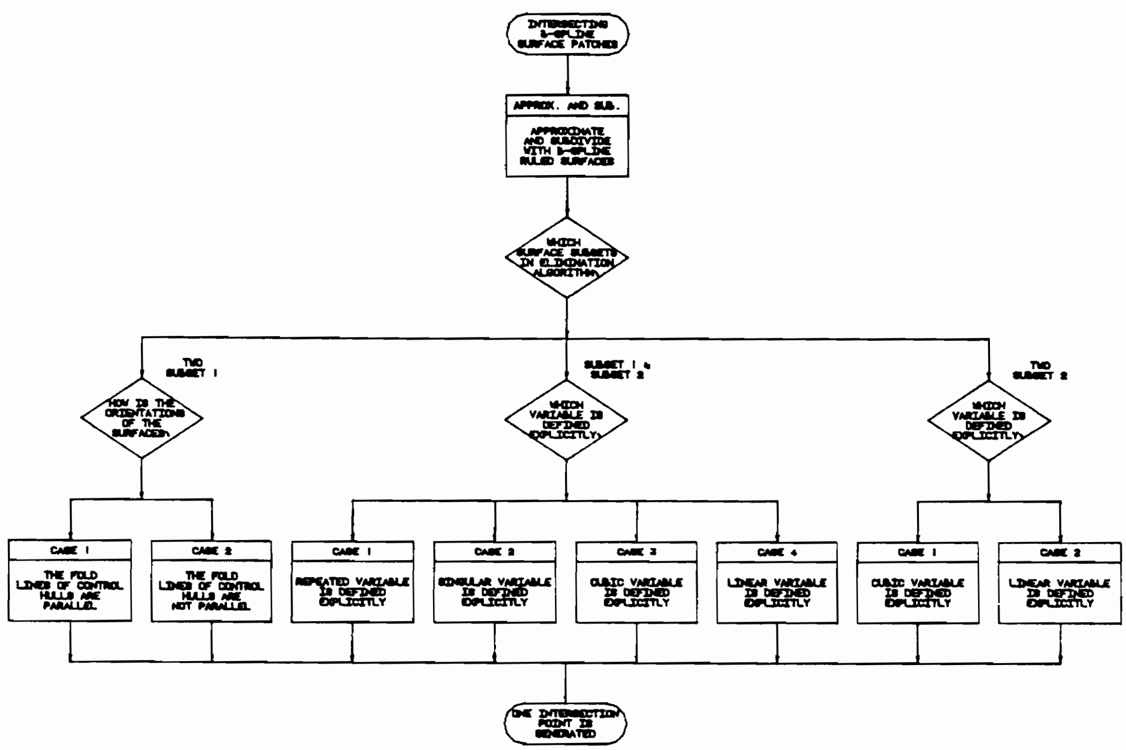

Figure 12. Logic for the selection of the intersection algorithms

Chapter 4

Post Processing of the Intersection Points

The intersection points that are obtained from the previous procedures are in global coordinate space and lie on the approximated ruled surfaces. The intersection points are projected back to the original B-spline surfaces to obtain the parametric values, sorted, then interpolated to generate cubic B-spline curves in the original parametric space.

4.1 Surface Inversion of a Point

The projection of an intersection point from the approximated ruled surface to the original B-spline surface amounts to finding the minimum distance from a point on the original B-spline surface to the given intersection point. The problem can be expressed as a vector equation as follows (Timmer, 1977):

$$\vec{V} = \vec{p}(u,w) - \vec{q} \tag{4.1}$$

where $\vec{V}$ will be minimized. $\vec{p}(u,w)$ is a point on the original B-spline surface, and $\vec{q}$ is the given intersection point. An iteration scheme is required since the problem cannot be easily solved analytically. Computing the variation of Eq. 4.1 :

$$\partial \vec{V} = \frac{\partial \vec{p}}{\partial u} \partial u + \frac{\partial \vec{p}}{\partial w} \partial w \tag{4.2}$$

Making an initial guess (u_i , w_i) such that $\vec{V}(u_i , w_i) = \vec{V}_i$, the variation may be interpreted as:

$$\vec{V}_{i+1} - \vec{V}_i = (\frac{\partial \vec{p}}{\partial u})_i (u_{i+1} - u_i) + (\frac{\partial \vec{p}}{\partial w})_i (w_{i+1} - w_i) \tag{4.3}$$

Set $\vec{V}_{i+1} = 0$ and perform the appropriate vector products to obtain

$$\begin{aligned} u_{i+1} &= u_i + \frac{\vec{A}_i}{\vec{N}_i} \\ w_{i+1} &= w_i + \frac{\vec{B}_i}{\vec{N}_i} \\ \vec{A}_i &= (\frac{\partial \vec{p}}{\partial w})_i \times \vec{V}_i \\ \vec{B}_i &= (\frac{\partial \vec{p}}{\partial u})_i \times \vec{V}_i \\ \vec{N}_i &= (\frac{\partial \vec{p}}{\partial u})_i \times (\frac{\partial \vec{p}}{\partial w})_i \end{aligned} \tag{4.4}$$

The iterations will converge u_{i+1} , w_{i+1} to the point on the B-spline surface that is closest to $\vec{q}$, and the criterion to stop the process may be:

$$|\vec{V}_{i+1} - \vec{V}_i| \leq \varepsilon \tag{4.5}$$

4.2 Sorting of the Intersection Points

The above inversion will produce a string of parameter values of the intersection points. They are stored in a sorting list that is composed of a number of sets. For each set, there are two end points, and a predefined number of intermediate points which make up an intersection curve of two patches. The sorting algorithm will first connect the end points of two sets that share a common edge as shown in Figure 13 on page 41 to form a link, then it will connect the end point of the link to the end point of a set which shares the common edge. Once the sets are put into the link, they will be excluded from the sorting list. The linking process will continue until the sorting list is empty or there is no match between the end points of a set and the link. This algorithm is designed to sort for a maximum of two links from a sorting list. However it can be modified easily for greater number of links.

4.3 Interpolating the Intersection Points

Finally these strings of ordered points (links) are interpolated with cubic B-spline curves. This is done by providing the link and the end condition to a B-spline curve inversion algorithm (Gloudemans, 1990). The end condition can be specified as closed

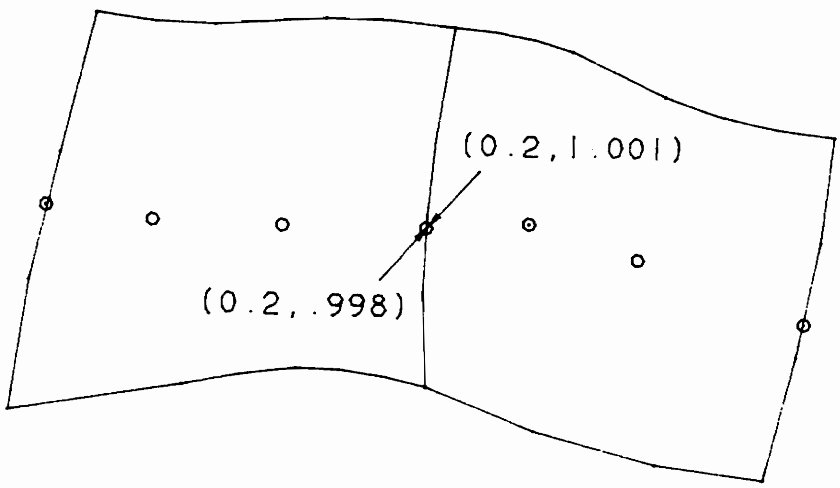

Figure 13. Sorting of intersection points: Connect the point that share the common edge

or open by comparing the two end points of the link. The inverted curves are in the original intersecting B-spline surfaces parametric space. However, they can be expressed in cartesian coordinate space when required.

Chapter 5

Applications and Examples

Several examples will be presented in this chapter demonstrating the applications of the algorithm for the intersection of B-spline surfaces. Figures 14, 15 and 16 show different views of the intersection of two subset 1 surfaces of case 1, where the intersection curve turns out to be a straight line. Figures 17 and 18 show the intersection of two subset 1 surfaces of case 2, where the intersection curve is obtained in closed form as discussed earlier. Figure 19 illustrates the intersection of two B-spline ruled surfaces (subset 2 surfaces). Figure 20 illustrates the intersection of two general B-spline surfaces in which the surfaces are subdivided and approximated with eight and nine subset 2 patches respectively in the intersection algorithm.

The following example is the intersection of the components of a conceptual model of an advanced aircraft in ACSYNT (Wampler et al, 1988). Figure 21 shows the intersections of the wings and the fuselage, and Figure 22 illustrates the intersections of the tail and the afterbody. All the B-spline surface patches in this aircraft model were

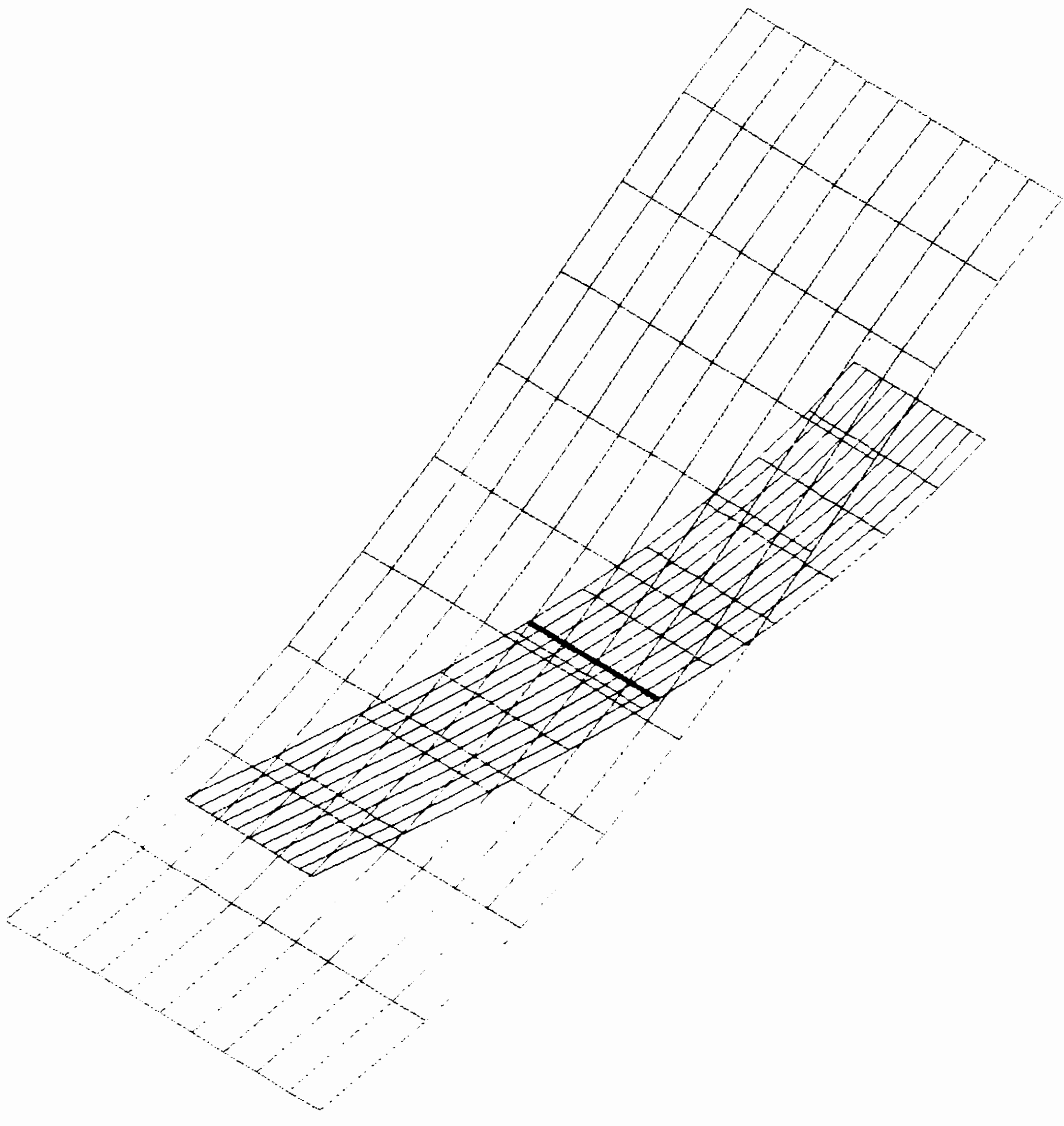

Figure 14. Intersection of subset 1 surfaces of case 1 (view 1): Isometric view.

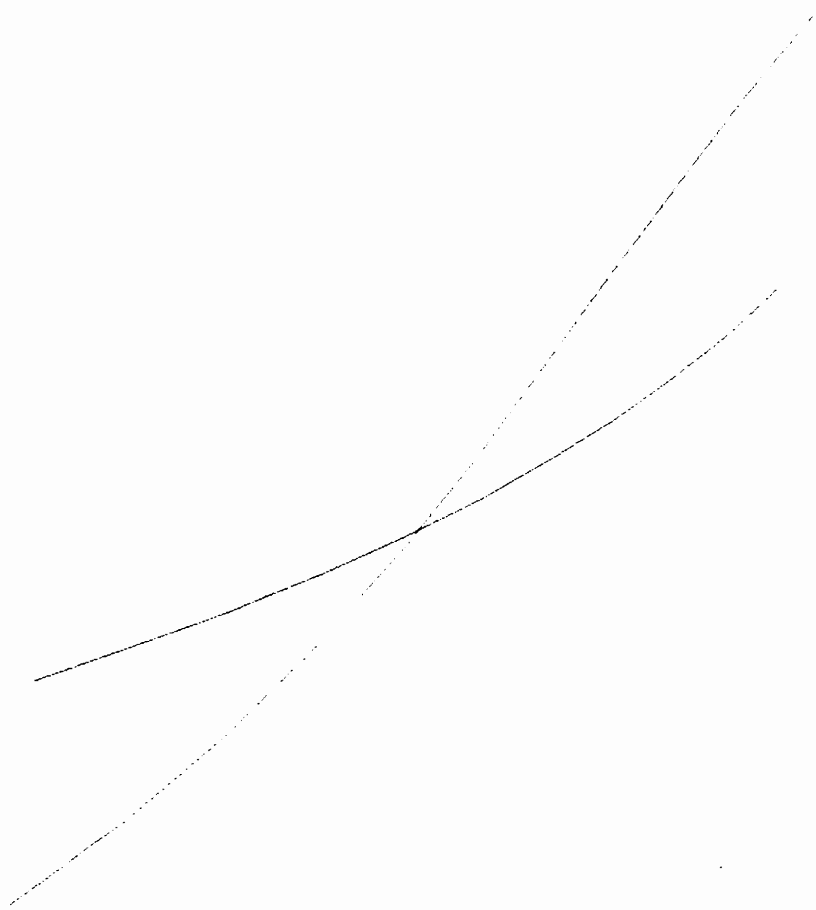

Figure 15. Intersection of subset 1 surfaces of case 1 (view 2): Side view.

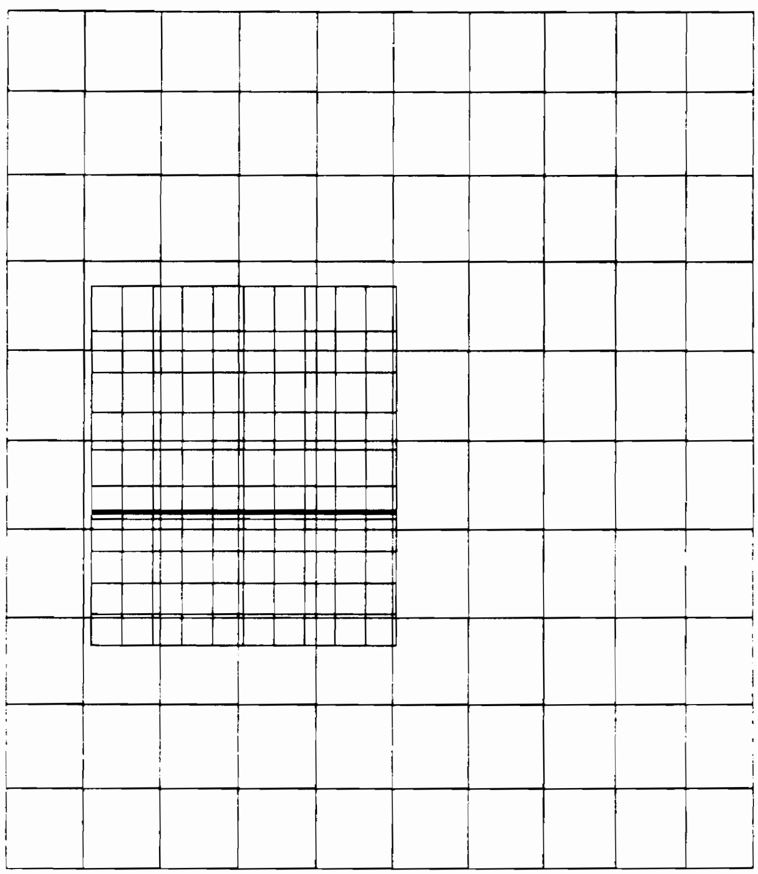

Figure 16. Intersection of subset 1 surfaces of case 1 (view 3): Top view.

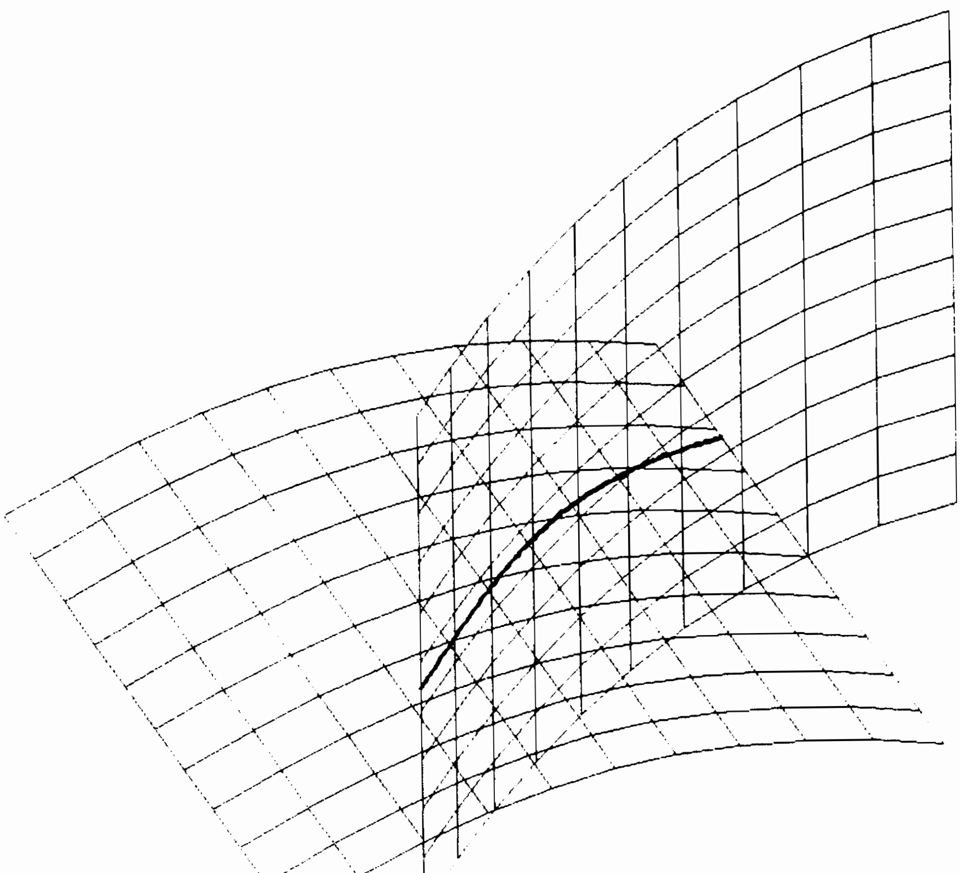

Figure 17. Intersection of subset 1 surfaces of case 2 (view 1): Isometric view.

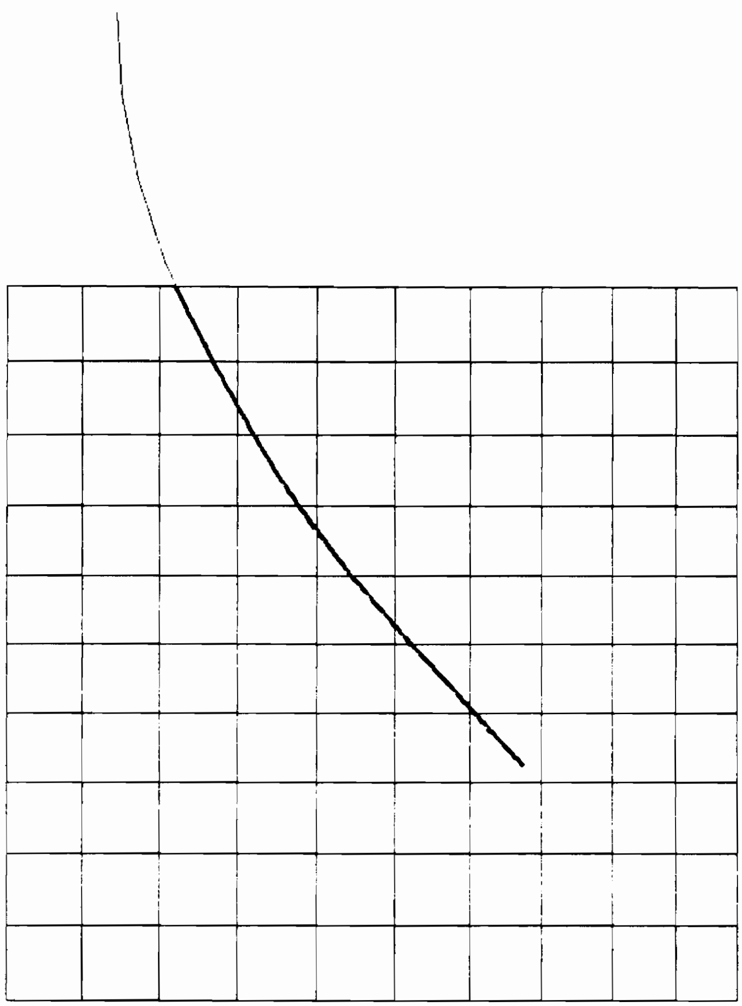

Figure 18. Intersection of subset 1 surfaces of case 2 (view 2): Top view.

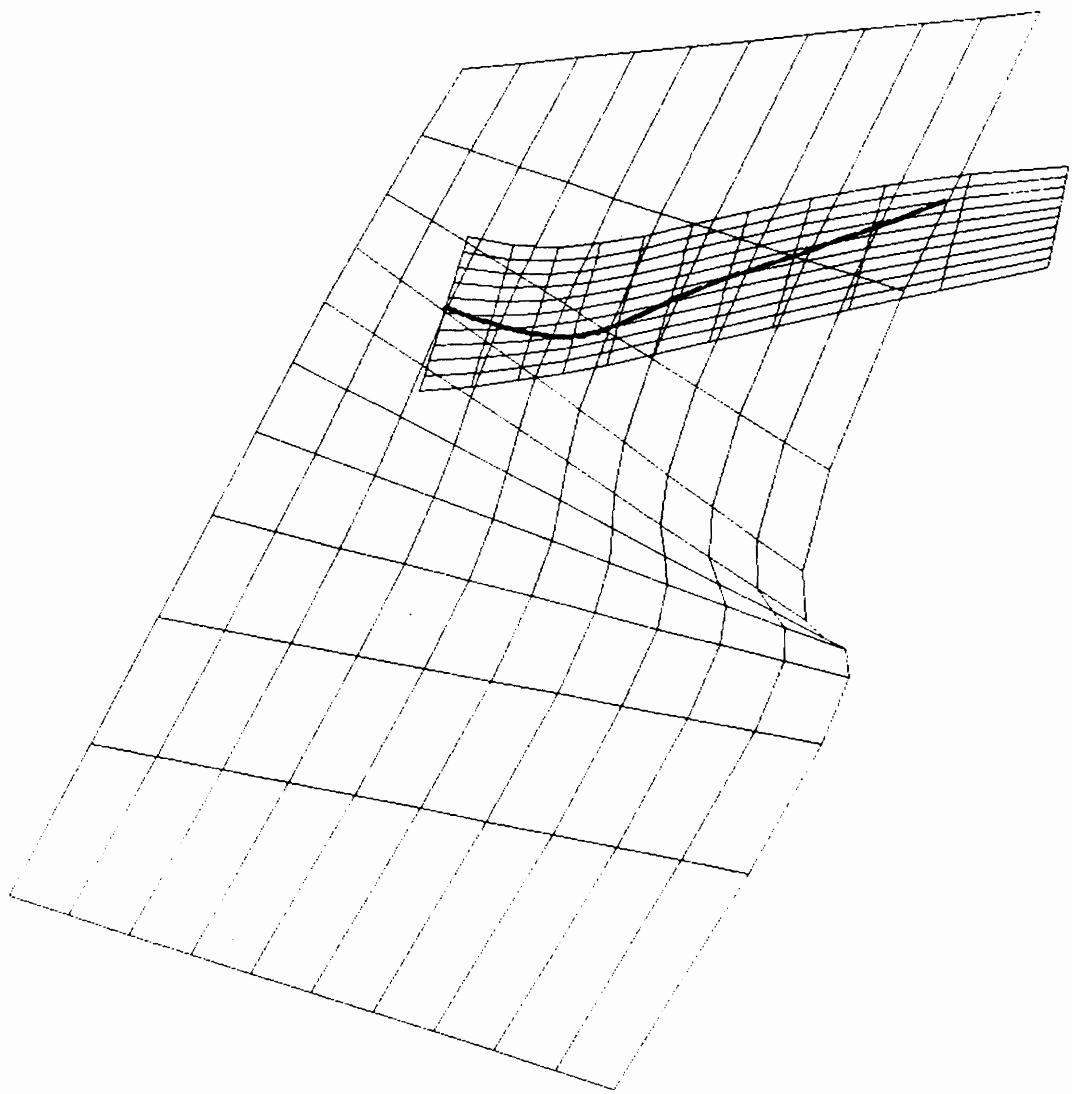

Figure 19. Intersection of two B-spline ruled surfaces

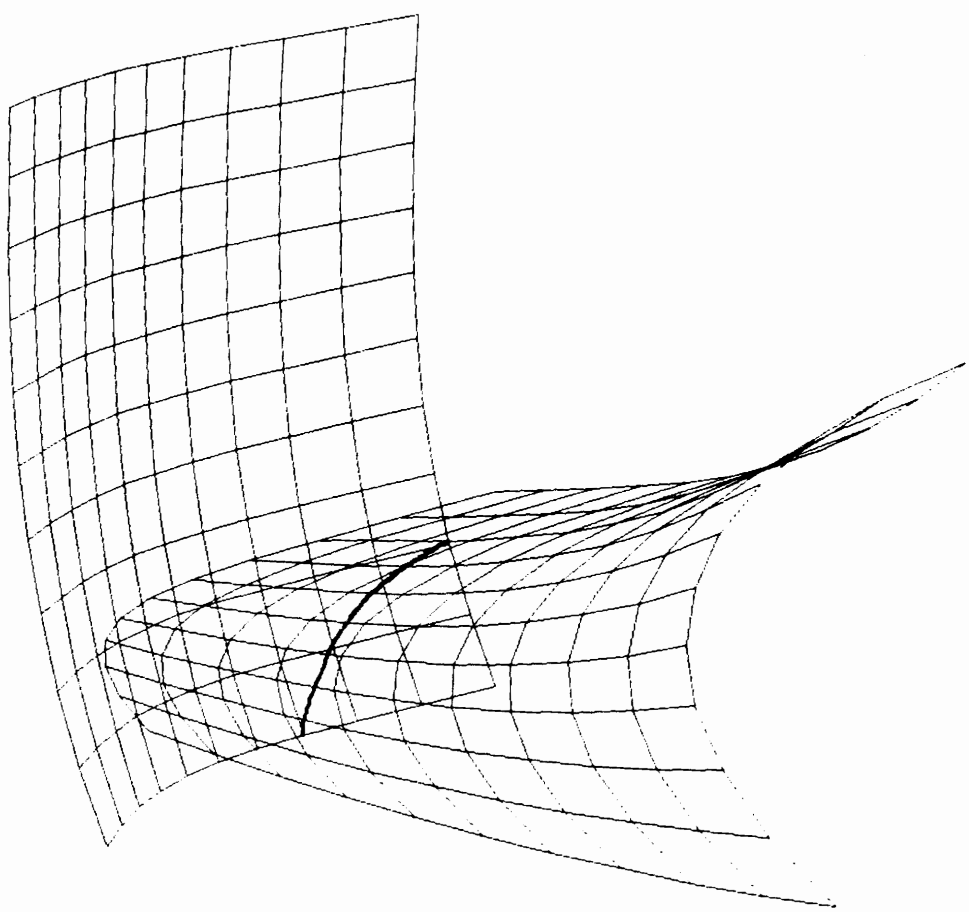

Figure 20. Intersection of two B-spline surfaces

successfully approximated with just one B-spline ruled surface each in the intersection algorithm, with tolerance of 0.012 inch. The fuselage is 27.4286 feet long and it is approximated with subset 1 patches, whereas the wings, tail and afterbody are approximated with subset 2 patches. Finally Figures 23, 24, 25 and 26 show different views of the aircraft with the intersection curves.

Appendices D through J contain the source code listings for all the intersection algorithms described in this thesis. The code is written in FORTRAN 77, and uses IBM's graPHIGS for graphical output. The code is divided into seven sections:

- Appendix D: Main program for intersection algorithms
- Appendix E: Intersection
- Appendix F: Ruled surfaces approximation
- Appendix G: Intersection of two subset 1 surfaces
- Appendix H: Intersection of subset 1 and subset 2 surfaces
- Appendix I: Intersection of two subset 2 surfaces
- Appendix J: Mathematical functions

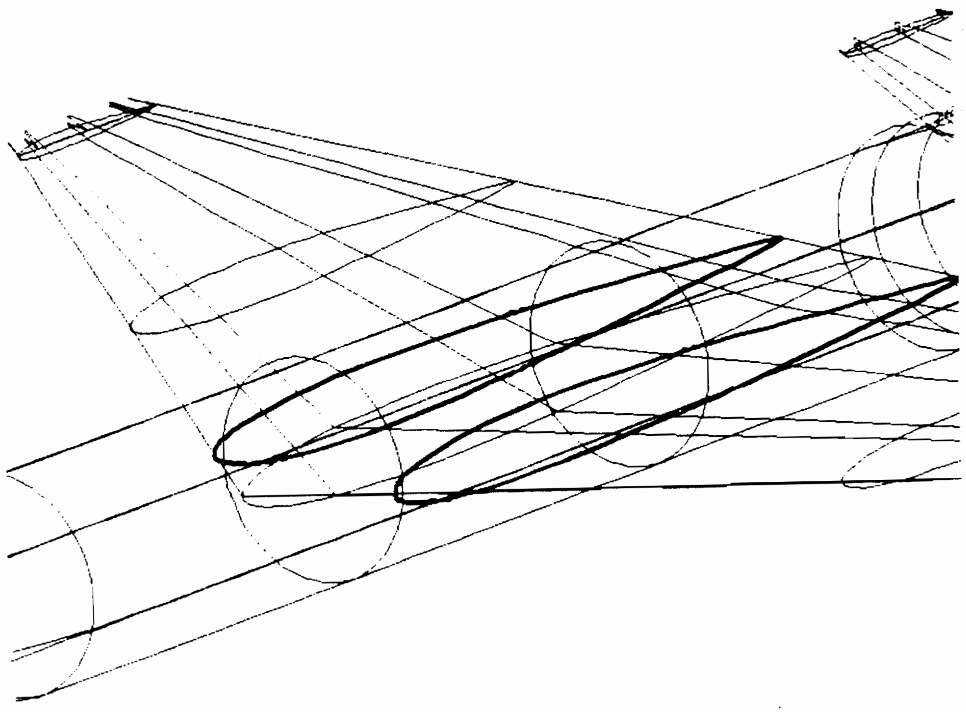

Figure 21. Intersections of the wings and the fuselage

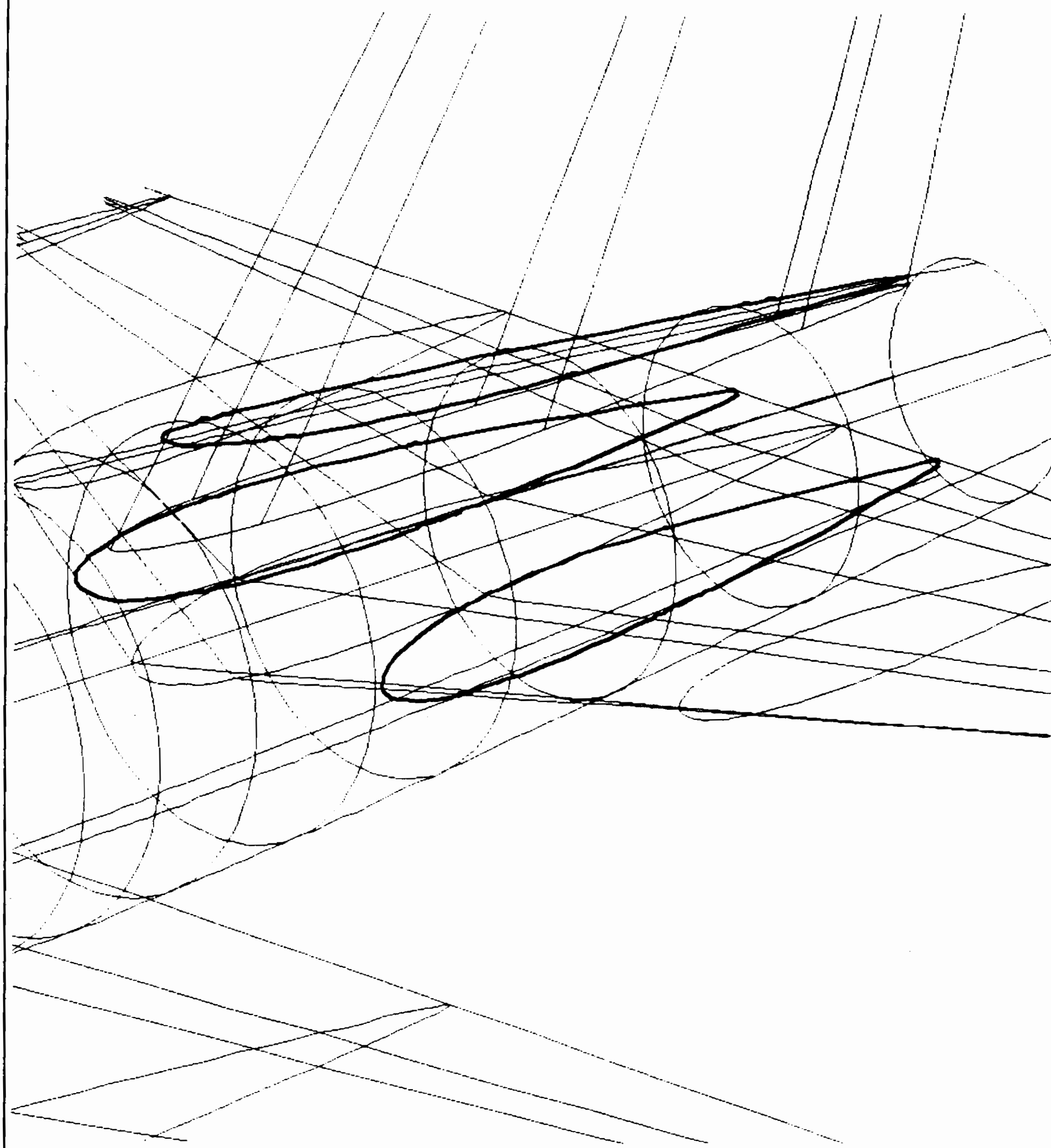

Figure 22. Intersections of the tail and the afterbody

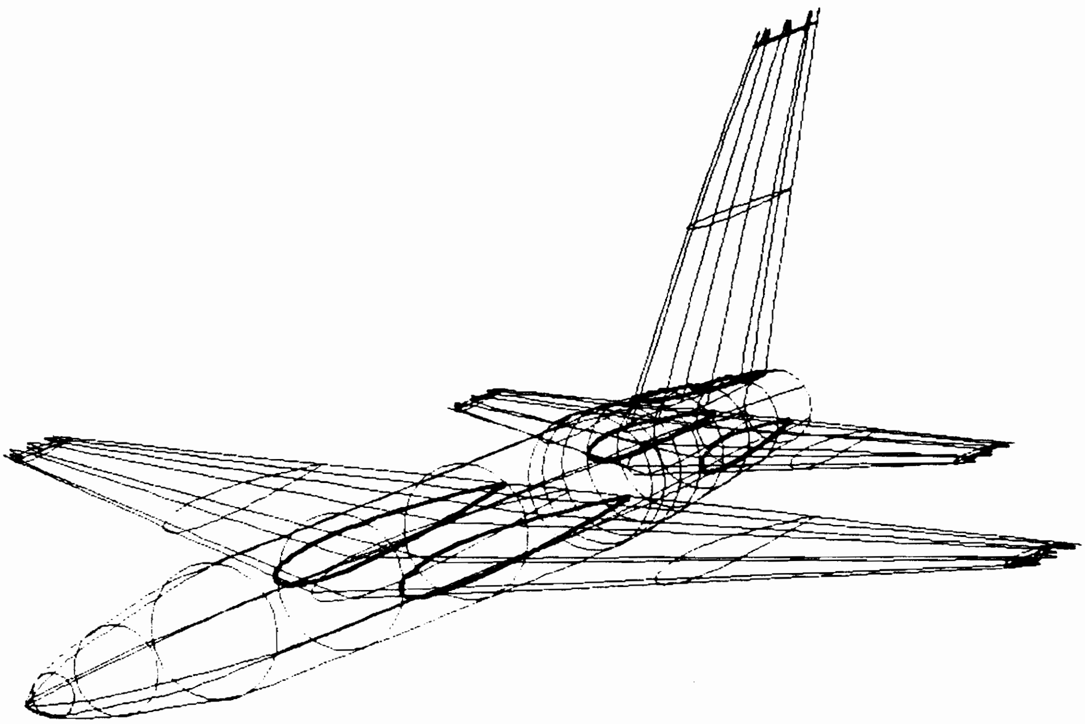

Figure 23. Intersections of the aircraft components (view 1): Isometric view.

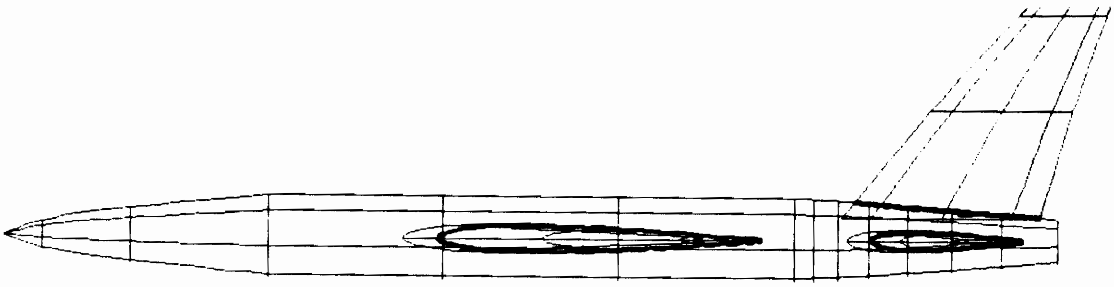

Figure 24. Intersections of the aircraft components (view 2): Side view.

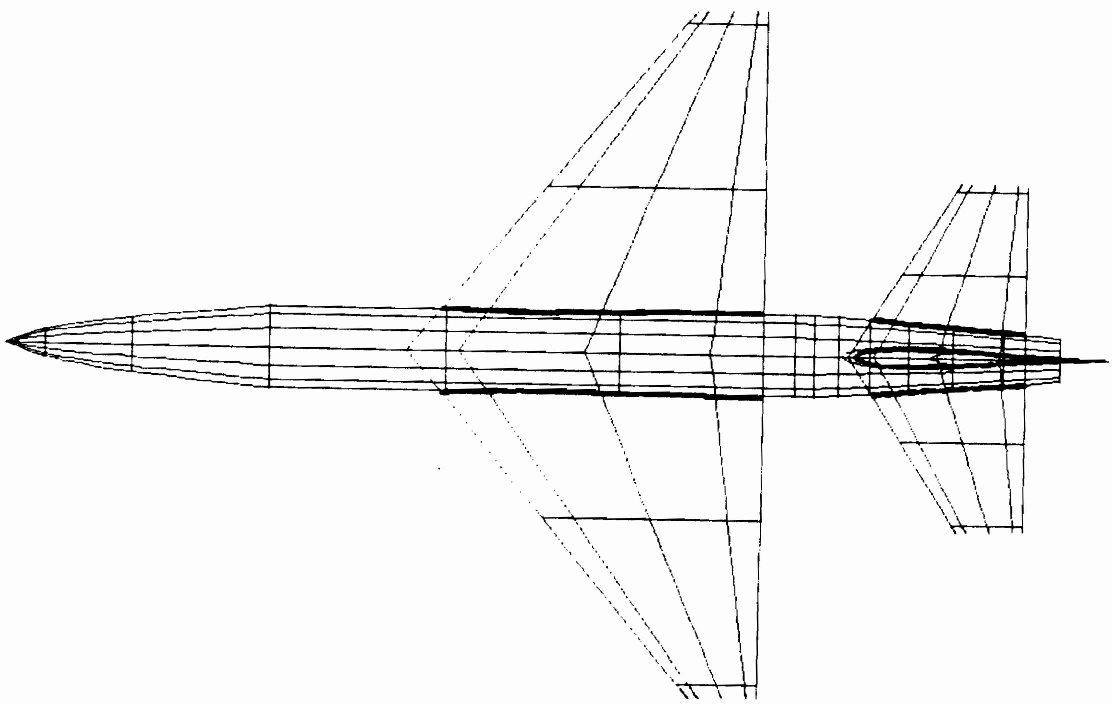

Figure 25. Intersections of the aircraft components (view 3): Top view.

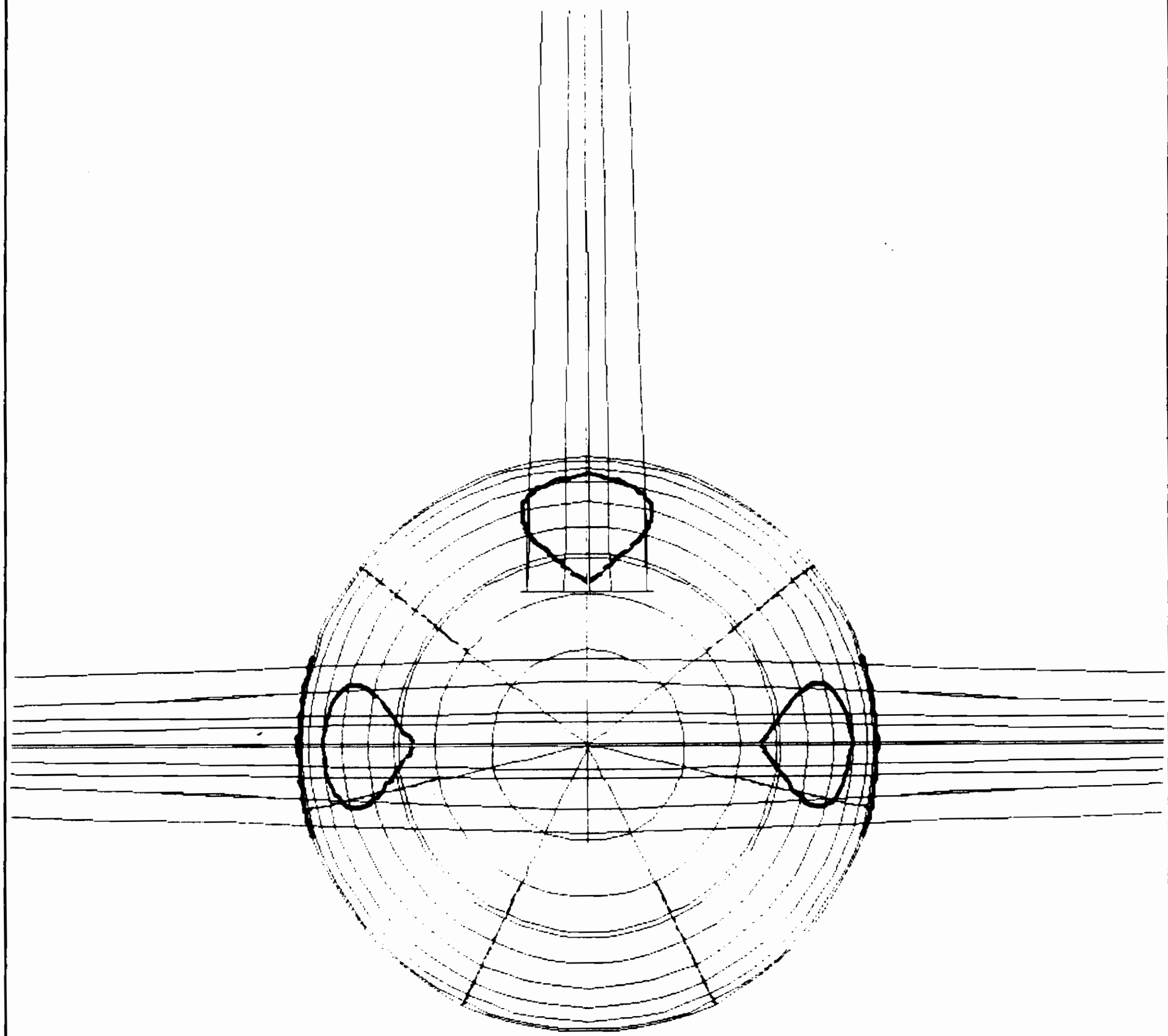

Figure 26. Intersections of the aircraft components (view 4): Front view.

Chapter 6

Conclusions and Recommendations

This research investigated a new approach for solving the intersection of two bicubic B-spline surfaces. It differs from other conventional numerical and analytical methods in that:

1. The surfaces are approximated with ruled surfaces rather than planar polygons to reduce the number of subdivisions, and consequently improve the surface approximations. The ruled surface approximations are especially advantageous in aircraft and ship design where such surfaces can be easily approximated with ruled surface patches.

2. In contrast to algebraic geometry, where the computations quickly become unmanageable, even when dealing with surfaces of relatively low degree (Waggenspack, 1987), the intersection problems are solved analytically because of the ruled surface approximations.

3. The intersection curve is presented in the original parametric space regardless of the approximated nature of the intersection process. This is useful for the purposes of analysis, design, or manufacture, and it can be easily expressed in cartesian coordinates when required.

4. While the solution method is based on uniform B-spline surfaces, it is believed that due to the analytical strategy, the solution is applicable to other mathematical forms; for instance, NURBS or Bezier surfaces.

In the intersection algorithm, a preset number of intersection points are solved for between two single patches, regardless of the spacing between the intersection points. This will create some problems especially when the intersection points are interpolated with a uniform B-spline curve. The curve may be transformed or twisted to an undesired shape due to the uneven spacing of the points. For better control of the shape of the intersection curve, it is suggested that the number of intersection points solved for between two patches be a function of the spacing of the intersection points. IMSL (IMSL User's Manual, 1987) and bisection algorithms were utilized to produce the examples shown earlier. The algorithms are reasonably fast and robust. However, there is definitely room for improvement with a better rootsolving algorithm; for example, Newton's method.

References

Aziz, N. M., Bata, R., and Bhat, S., "Bezier Surface/Surface Intersection" *IEEE Computer Graphics and Applications* , Vol 1, No 1, 1990, pp 50-58.

Barsky, Brian A., "Computer-Aided Geometric Design : A Bibliography with Keywords and Classified Index ", *IEEE Computer Graphics and Applications* , Vol 1, No 3, 1981, pp 67-109.

Calson, W. E., "An Algorithm and Data Structure for 3D Object Synthesis using Surface Patch Intersections ", *Computer Graphics* , Vol 16, No 3, 1982, pp 255-259.

Comba, P., "A Procedure for Detecting Intersections of 3-D Objects ", *Journal of the Association for Computing Machinery* , Vol 15, No 3, 1968, pp 354-366.

Faux, I. D. and Pratt, M. J., *Computational Geometry for Design and Manufacture* , Ellis Horwood, Chichester, UK., 1979.

Gandhi, Ashit R., "Feature-Based Geometric Modeling using B-spline Surfaces and a Natural Language Approach ", PhD Dessertation, Department of Mechanical Engineering, VPI & SU, 1989.

Gloudemans, James R., "Filleting of Aircraft Components Using Non-Uniform B-Spline Surfaces ", Thesis - Master of Science, Department of Mechanical Engineering, VPI & SU, 1990.

Hanna, S. L., Abel, J. F. and Greenberg, D. P., "Intersection of Parametric Surfaces by Means of Look-Up Tables ", *IEEE Computer Graphics and Applications* , Vol 3, No 7, 1984, pp 39-48.

IMSL User's Manual, 1987, Vol 1, International Mathematical and Statistical Libraries, Inc., Houston, Texas.

Lane, J. M. and Riesenfeld, R. F., "A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces ", *IEEE Transactions on Pattern Analysis and Machine Intelligence* , Vol PAMI-2, No 1, 1980, pp 35-46.

Lasser, D., "Intersection of Parametric Surfaces in the Bernstein-Bezier Representation", *Computer-Aided Design* Vol 18, No 4, 1988, pp 162-192.

Lee, R. B. and Fredricks, David A., "Intersection of Parametric Surfaces and a Plane ", *IEEE Computer Graphics and Applications* , Vol 4, No 8, 1984, pp 48-51.

Mortenson, M. E., *Geometric Modeling* , John Wiley & Sons, Inc., USA, 1985.

Ocken, S., Schwartz, J. T. and Sharir, M., "Precise Implementation of CAD Primitives using Rational Parameterizations of Standard Surfaces ", *G.M. Conference on Solid Modeling* , 1983.

Peng, Q. S., "An Algorithm for Finding the Intersection Lines between two B-spline Surfaces ", *Computer-Aided Design* , Vol 16, No 4, 1984, pp 191-196.

Phillips, M. B. and Odell G. M., "An Algorithm for Locating and Displaying the Intersection of Two Arbitrary Surfaces", *IEEE Computer Graphics and Applications*, Vol 4, No 9, 1984, pp 48-58.

Sabin, M. A., "A Method for Displaying the Intersection Curve of two Quadric Surfaces", *Computer Journal* , Vol 19, No 4, 1976, pp 336-338.

Salmon, George D. D., *Lessons Introductory to the Modern Higher Algebra* , Chelsea Publishing Company, New York, 1885.

Timmer, H. G., "A Solution to the Surface Intersection Problem ", McDonnell Douglas Report MDC J7789, 1977.

Waggenspack, Warren N. Jr., "Parametric Curve Approximations for Surface Intersections ", PhD Thesis, Department of Mechanical Engineering, Purdue University, 1987.

Wampler, S., Myklebust, A., Jayaram, S., and Gelhausen, P., "Improving Aircraft Conceptual Design - A PHIGS Interactive Graphics Interface for ACSYNT ", *American Institute of Aeronautics and Astronautics* , AIAA-88-4481, 1988.

Appendix A : Solution for Case 1 of Category 1

In solving the intersection in case 1 of category 1, a system of nonlinear equations (two equations with two unknowns) is solved. This will generate a 12th order polynomial. The system is as follows (refer to Eqs. 3.6 and 3.7):

$$a1u^3 + a2u^2 + a3u + a4 = a5w^3 + a6w^2 + a7w + a8 \qquad (A1)$$

$$a1'u^3 + a2'u^2 + a3'u + a4' = a5'w^3 + a6'w^2 + a7'w + a8' \qquad (A2)$$

Rearranging the equations:

$$a1u^3 + a2u^2 + a3u + a4 - a5w^3 - a6w^2 - a7w - a8 = 0 \qquad (A3)$$

$$a1'u^3 + a2'u^2 + a3'u + a4' - a5'w^3 - a6'w^2 - a7'w - a8' = 0 \qquad (A4)$$

Compute [(Eq. A3) $\times\, a1'$] – [(Eq. A4) $\times\, a1$] to reduce the highest order of u in Eqs. A3 and A4 to quadratic, and use notation such as (Salmon, 1885):

$a2a1' - a2'a1 = (a2a1')$, or $a2'a1 - a2a1' = (a2'a1)$ in the process:

$$(a2a1')u^2 + (a3a1')u + (a4a1') + (a5'a1)w^3 + (a6'a1)w^2 + (a7'a1)w + (a8'a1) = 0 \quad (A5)$$

Again perform $[(a1'u^2 + a2'u + a3') \times (\text{Eq. A3})] - [(a1u^2 + a2u + a3) \times (\text{Eq. A4})]$ to reduce the highest order of u in Eqs. A3 and A4 to quadratic, and use notation such as $a1a5' - a1'a5 = (a1a5')$,or $a5'a1 - a5a1' = (a5'a1)$ to obtain:

$$u^2\left[(a1a5')w^3 + (a1a6')w^2 + (a1a7')w + (a1a8') + (a1'a4)\right] + u\left[(a2a5')w^3 + (a2a6')w^2 + (a2a7')w + (a2a8') + (a2'a4)\right] + \left[(a3a5')w^3 + (a3a6')w^2 + (a3a7')w + (a3a8') + (a3'a4)\right] = 0 \quad (A6)$$

Rewrite Eqs. A5 and A6 as:

$$au^2 + bu + c + dw^3 + ew^2 + fw + g = 0 \quad (A7)$$

$$u^2(t1w^3 + t2w^2 + t3w + t4) + u(t5w^3 + t6w^2 + t7w + t8) + (t9w^3 + t10w^2 + t11w + t12) = 0 \quad (A8)$$

Compute $[(\text{Eq. A7}) \times (t1w^3 + t2w^2 + t3w + t4)] - [(\text{Eq. A8}) \times a]$ to reduce u in Eqs. A7 and A8 to linear:

$$u\left[(bt1 - at5)w^3 + (bt2 - at6)w^2 + (bt3 - at7)w + (bt4 - at8)\right] + w^6(dt1) + w^5(dt2 + et1) + w^4(dt3 + et2 + ft1) + w^3(ct1 + dt4 + et3 + ft2 + gt1 - at9) + w^2(ct2 + et4 + ft3 + gt2 - at10) + w(ct3 + ft4 + gt3 - at11) + (ct4 + gt4 - at12) = 0 \quad (A9)$$

Again perform $\{(\text{Eq. A7}) \times [u(t1uw^3 + t2uw^2 + t3uw + t4u + t5w^3 + t6w^2 + t7w + t8)]\} - [(\text{Eq. A8}) \times (au + b)]$ to reduce u in Eqs. A7 and A8 to linear:

$$
\begin{aligned}
& u[(t1d)w^6 + (t1e + t2d)w^5 + (t1f + t2e + t3d)w^4 + \\
& \quad (t1c + t1g + t2f + t3e + t4d - at9)w^3 + \\
& \quad (t2c + t2g + t3f + t4e - at10)w^2 + (t3c + t3g + t4f - at11)w + \\
& \quad (t4c + t4g - at12)] + \\
& w^6(t5d) + w^5(t5e + t6d) + w^4(t5f + t6e + t7d) + \\
& w^3(t5c + t5g + t6f + t7e + t8d - bt9) + w^2(t6c + t6g + t7f + t8e - bt10) + \\
& w(t7c + t7g + t8f - bt11) + (t8c + t8g - bt12) = 0
\end{aligned}
\qquad (A10)
$$

Rewrite Eqs. A9 and A10 as:

$$
\begin{aligned}
& u(s1w^3 + s2w^2 + s3w + s4) + s5w^6 + s6w^5 + \\
& s7w^4 + s8w^3 + s9w^2 + s10w + s11 = 0
\end{aligned}
\qquad (A11)
$$

$$
\begin{aligned}
& u(v1w^6 + v2w^5 + v3w^4 + v4w^3 + v5w^2 + v6w + v7) + \\
& v8w^6 + v9w^5 + v10w^4 + v11w^3 + v12w^2 + v13w + v14 = 0
\end{aligned}
\qquad (A12)
$$

Finally perform $[(\text{Eq. A11}) \times (v1w^6 + v2w^5 + v3w^4 + v4w^3 + v5w^2 + v6w + v7)] - [(\text{Eq. A12}) \times (s1w^3 + s2w^2 + s3w + s4)]$ to eliminate u in Eqs. A11 and A12, and yield a 12th order polynomial of w:

$$
\begin{aligned}
& w^{12}(s5v1) + w^{11}(s5v2 + s6v1) + w^{10}(s5v3 + s6v2 + s7v1) + \\
& w^{9}(s5v4 + s6v3 + s7v2 + s8v1 - s1v8) + \\
& w^{8}(s5v5 + s6v4 + s7v3 + s8v2 + s9v1 - s1v9 - s2v8) \\
& w^{7}(s5v6 + s6v5 + s7v4 + s8v3 + s9v2 + \\
& \quad s10v1 + s11v1 - s1v11 - s2v10 - s3v9 - s4v8) + \\
& w^{6}(s5v7 + s6v6 + s7v5 + s8v4 + s9v3 + \\
& \quad s10v2 + s11v1 - s1v11 - s2v10 - s3v9 - s4v8) + \\
& w^{5}(s6v7 + s7v6 + s8v5 + s9v4 + s10v3 + s11v2 - s1v12 - s2v11 - s3v10 - s4v9) + \\
& w^{4}(s7v7 + s9v6 + s10v5 + s11v4 - s1v14 - s2v13 - s3v12 - s4v10) + \\
& w^{3}(s8v7 + s9v6 + s10v5 + s11v4 - s1v14 - s2v13 - s3v12 - s4v11) + \\
& w^{2}(s9v7 + s10v6 + s11v5 - s2v14 - s3v13 - s4v12) + \\
& w(s10v7 + s11v6 - s3v14 - s4v13) + (s11v7 - s4v14) \;=\; 0
\end{aligned}
\qquad (A13)
$$

Appendix B : Solution for Case 1 of Category 2

In solving the intersection in case 1 of category 2, a system of nonlinear equations (two equations with two unknowns) is solved. This will generate a sixth order polynomial. The system is as follows (refers to Eqs. 3.19 and 3.20):

$$C_1X + C_2 = \frac{X - X_b(w)}{X_a(w)} Y_a(w) + Y_b(w) \qquad (B1)$$

$$K_1X + K_2 = \frac{X - X_b(w)}{X_a(w)} Z_a(w) + Z_b(w) \qquad (B2)$$

where C_1, C_2, K_1 and K_2 are the values of the defined functions, X_a,X_b, Y_a, Y_b,Z_a and Z_b are the cubic functions of the associated variable, and X is the second unknown (one of the three scalar values of the intersection point). Extend and rearrange the Eqs. B1 and B2 as follows:

$$X[C_1X_a(w) - Y_a(w)] + [C_2X_a(w) + X_b(w)Y_a(w) - X_a(w)Y_b(w)] = 0 \qquad (B3)$$

$$X[K_1X_a(w) - Z_a(w)] + [K_2X_a(w) + X_b(w)Z_a(w) - X_a(w)Z_b(w)] = 0 \qquad (B4)$$

Compute $\{(\text{Eq B4}) \times [C_1X_a(w) - Y_a(w)]\} - \{(\text{Eq B3}) \times [K_1X_a(w) - Z_a(w)]\}$ to eliminate X, and yield a 6th order polynomial of w:

$$\begin{aligned} &C_1K_2X_a(w) + C_1X_b(w)Z_a(w) - C_1X_a(w)Z_b(w) - K_2Y_a(w) + Y_a(w)Z_b(w) \\ &- C_2K_1X_a(w) + K_1X_b(w)Y_a(w) + K_1X_a(w)Y_b(w) - C_2Z_a(w) - Z_a(w)Y_b(w) = 0 \end{aligned} \qquad (B5)$$

Appendix C : Solution for Case 2 of Category 2

In solving of the intersections in case 2 of category 2, one variable must be eliminated from a non-linear system which is composed of two equations with two unknowns. The solution of this non-linear system eventually yields a 33rd order polynomial. The non-linear system is as follows (refer to Eqs. 3.25 and 3.26):

$$Y_3(u) = \frac{X_3(u) - X_b(w)}{X_a(w)} Y_a(w) + Y_b(w) \qquad (C1)$$

$$Z_3(u) = \frac{X_3(u) - X_b(w)}{X_a(w)} Z_a(w) + Z_b(w) \qquad (C2)$$

where X_3, Y_3, Z_3, X_a, X_b, Y_a, Y_b, Z_a and Z_b are the cubic functions of the associated variables. Extend the cubic functions in Eqs. C1 and C2, and rearrange the equations as follows for elimination:

$$\begin{aligned} &au^3w^3 + bu^3w^2 + cu^3w + du^3 + eu^2w^3 + fu^2w^2 + gu^2w + hu^2 + \\ &iuw^3 + iuw^2 + kuw + lu + mw^6 + nw^5 + ow^4 + pw^3 + qw^2 + rw + s = 0 \end{aligned} \qquad (C3)$$

$$a'u^3w^3 + b'u^3w^2 + c'u^3w + d'u^3 + e'u^2w^3 + f'u^2w^2 + g'u^2w + h'u^2 + i'uw^3 + j'uw^2 + k'uw + l'u + m'w^6 + n'w^5 + o'w^4 + p'w^3 + q'w^2 + r'w + s' = 0 \qquad (C4)$$

Compute $[(\text{Eq. C4}) \times (aw^3 + bw^2 + cw + d)]$ −

$[(\text{Eq. C3}) \times (a'w^3 + b'w^2 + c'w + d')]$ to reduce u in Eqs. C3 and C4 to quadratic:

$$\begin{aligned} &aau^2w^6 + bbu^2w^5 + ccu^2w^4 + ddu^2w^3 + eeu^2w^2 + ffu^2w + ggu^2 + \\ &hhuw^6 + iiuw^5 + jjuw^4 + kkuw^3 + lluw^2 + mmuw + nnu + \\ &x1w^9 + x2w^8 + x3w^7 + x4w^6 + x5w^5 + \\ &x6w^4 + x7w^3 + x8w^2 + x9w + x10 = 0 \end{aligned} \qquad (C5)$$

where

$aa = (ae' - ea')$
$bb = (af' - fa') + (be' - eb')$
$cc = (ag' - ga') + (bf' - fb') + (ce' - ec')$
$dd = (ah' - ha') + (bg' - gb') + (cf' - fc') + (de' - ed')$
$ee = (bh' - hb') + (cg' - gc') + (df' - fd')$
$ff = (ch' - hc') + (dg' - gd')$
$gg = (dh' - hd')$
$hh = (ai' - ia')$
$ii = (aj' - ja') + (bi' - ib')$
$jj = (ak' - ka') + (bj' - jb') + (ci' - ic')$
$kk = (al' - la') + (bk' - kb') + (cj' - jc') + (di' - id')$
$ll = (bl' - lb') + (ck' - kc') + (dj' - jd')$
$mm = (cl' - lc') + (dk' - kd')$
$nn = (dl' - ld')$

$$x1 = (am' - ma')$$
$$x2 = (an' - na') + (bm' - mb')$$
$$x3 = (ao' - oa') + (bn' - nb') + (cm' - mc')$$
$$x4 = (ap' - pa') + (bo' - ob') + (cn' - nc') + (dm' - md')$$
$$x5 = (aq' - qa') + (bp' - pb') + (co' - oc') + (dn' - nd')$$
$$x6 = (ar' - ra') + (bq' - qb') + (cp' - pc') + (do' - od')$$
$$x7 = (as' - sa') + (br' - rb') + (cq' - qc') + (dp' - pd')$$
$$x8 = (bs' - sb') + (cr' - rc') + (dq' - qd')$$
$$x9 = (cs' - sc') + (dr' - rd')$$
$$x10 = (ds' - sd')$$

Again compute [(Eq. C4) $\times (auw^3 + buw^2 + cuw + du + ew^3 + fw^2 + gw + h)$] –

[(Eq. C3) $\times (a'uw^3 + b'uw^2 + c'uw + d'u + e'w^3 + f'w^2 + g'w + h')$] to reduce u in Eqs. C3 and C4 to quadratic:

$$\begin{aligned}
&aa'u^2w^6 + bb'u^2w^5 + cc'u^2w^4 + dd'u^2w^3 + ee'u^2w^2 + ff'u^2w + gg'u^2 + \\
&y1'uw^9 + y2'uw^8 + y3'uw^7 + hh'uw^6 + \\
&ii'uw^5 + jj'uw^4 + kk'uw^3 + ll'uw^2 + mm'uw + nn'u + \\
&x1'w^9 + x2'w^8 + x3'w^7 + x4'w^6 + x5'w^5 + \\
&x6'w^4 + x7'w^3 + x8'w^2 + x9'w + x10' = 0
\end{aligned} \tag{C6}$$

where

$$aa' = (ai' - ia')$$
$$bb' = (aj' - ja') + (bi' - ib')$$
$$cc' = (ak' - ka') + (bj' - jb') + (ci' - ic')$$
$$dd' = (al' - la') + (bk' - kb') + (cj' - jc') + (di' - id')$$
$$ee' = (bl' - lb') + (ck' - kc') + (dj' - jd')$$
$$ff' = (cl' - lc') + (dk' - kd')$$
$$gg' = (dl' - ld')$$
$$y1' = (am' - ma')$$
$$y2' = (an' - na') + (bm' - mb')$$
$$y3' = (ao' - oa') + (bn' - nb') + (cm' - mc')$$

$$hh' = (ap' - pa') + (bo' - ob') + (cn' - nc') + (dm' - md') + (ei' - ie')$$
$$ii' = (aq' - qa') + (bp' - pb') + (co' - oc') + (dn' - nd') + (ej' - je') + (fi' + if')$$
$$jj' = (ar' - ra') + (bq' - qb') + (cp' - pc') + (do' - od') + (ek' - ke') + (fj' + jf') + (gi' + ig')$$
$$kk' = (as' - sa') + (br' - rb') + (cq' - qc') + (dp' - pd') + (el' - le') + (fk' + kf') + (gj' + jg') + (hi' + ih')$$
$$ll' = (bs' - sb') + (cr' - rc') + (dq' - qd') + (fl' - lf') + (gk' + kg') + (hj' + jh')$$
$$mm' = (cs' - sc') + (dr' - rd') + (gl' - lg') + (hk' - kh')$$
$$nn' = (ds' - sd') + (hl' - lh')$$
$$x1' = (em' - me')$$
$$x2' = (en' - ne') + (fm' - mf')$$
$$x3' = (eo' - oe') + (fn' - nf') + (gm' - mg')$$
$$x4' = (ep' - pe') + (fo' - of') + (gn' - ng') + (hm' - mh')$$
$$x5' = (eq' - qe') + (fp' - pf') + (go' - og') + (hn' - nh')$$
$$x6' = (er' - re') + (fq' - qf') + (gp' - pg') + (ho' - oh')$$
$$x7' = (es' - se') + (fr' - rf') + (gq' - qg') + (hp' - ph')$$
$$x8' = (fs' - sf') + (gr' - rg') + (hq' - qh')$$
$$x9' = (gs' - sg') + (hr' - rh')$$
$$x10' = (hs' - sh')$$

Compute [(Eq. C6) $\times (aaw^6 + bbw^5 + ccw^4 + ddw^3 + eew^2 + ffw + gg)$] − [(Eq. C5) $\times (aa'w^6 + bb'w^5 + cc'w^4 + dd'w^3 + ee'w^2 + ff'w + gg')$] to reduce u in Eqs. C5 and C6 to linear:

$$\begin{aligned} &auw^{15} + buw^{14} + cuw^{13} + duw^{12} + euw^{11} + fuw^{10} + \\ &guw^9 + huw^8 + iuw^7 + juw^6 + kuw^5 + luw^4 + \\ &muw^3 + nuw^2 + ouw + pu + s1w^{15} + s2w^{14} + s3w^{13} + \\ &s4w^{12} + s5w^{11} + s6w^{10} + s7w^9 + s8w^8 + s9w^7 + \\ &s10w^6 + s11w^5 + s12w^4 + s13w^3 + s14w^2 + s15w + s16 = 0 \end{aligned} \tag{C7}$$

where

$a = (aay1')$

$b = (bby1') + (aay2')$

$c = (ccy1') + (bby2') + (aay3')$

$d = (ddy1') + (ccy2') + (bby3') + (aahh' - hhaa')$

$e = (eey1') + (ddy2') + (ccy3') + (bbhh' - hhbb') + (aaii' - iiaa')$

$f = (ffy1') + (eey2') + (ddy3') + (cchh' - hhcc') + (bbii' - iibb') + (aajj' - jjaa')$

$g = (ggy1') + (ffy2') + (eey3') + (ddhh' - hhdd') + (ccii' - iicc') + (bbjj' - jjbb') + (aakk' - kkaa')$

$h = (ggy2') + (ffy3') + (eehh' - hhee') + (ddii' - iidd') + (ccjj' - jjcc') + (bbkk' - kkbb') + (aall' + llaa')$

$i = (ggy3') + (ffhh' - hhff') + (eeii' - iiee') + (ddjj' - jjdd') + (cckk' - kkcc') + (bbll' + llbb') + (aamm' - mmaa')$

$j = (gghh' - hhgg') + (ffii' - iiff') + (eejj' - jjee') + (ddkk' + kkdd') + (ccll' - llcc') + (bbmm' + mmbb') + (aann' - nnaa')$

$k = (ggii' - iigg') + (ffjj' - jjff') + (eekk' - kkee') + (ddll' + lldd') + (ccmm' - mmcc') + (bbnn' + nnbb')$

$l = (ggjj' - jjgg') + (ffkk' - kkff') + (eell' - llee') + (ddmm' + mmdd') + (ccnn' - nncc')$

$m = (ggkk' - kkgg') + (ffll' - llff') + (eemm' - mmee') + (ddnn' + nndd') + (ddnn' - nndd')$

$n = (ggll' - llgg') + (ffmm' - mmff') + (eenn' - nnee')$

$o = (ggmm' - mmgg') + (ffnn' - nnff')$

$p = (ggnn' - nngg')$

$s1 = (aax1' - x1aa')$

$s2 = (bbx1' - x1bb') + (aax2' - x2aa')$

$s3 = (ccx1' - x1cc') + (bbx2' - x2bb') + (aax3' - x3aa')$

$s4 = (ddx1' - x1dd') + (ccx2' - x2cc') + (bbx3' - x3bb') + (aax4' - x4aa')$

$s5 = (eex1' - x1ee') + (ddx2' - x2dd') + (ccx3' - x3cc') + (bbx4' - x4bb') + (aax5' - x5aa')$

$s6 = (ffx1' - x1ff') + (eex2' - x2ee') + (ddx3' - x3dd') + (ccx4' - x4cc') + (bbx5' - x5bb') + (aax6' - x6aa')$

$s7 = (ggx1' - x1gg') + (ffx2' - x2ff') + (eex3' - x3ee') + (ddx4' - x4dd') + (ccx5' - x5cc') + (bbx6' - x6bb') + (aax7' - x7aa')$

$$s8 = (ggx2' - x2gg') + (ffx3' - x3ff') + (eex4' - x4ee') + (ddx5' - x5dd') +$$
$$(ccx6' - x6cc') + (bbx7' - x7bb') + (aax8' - x8aa')$$
$$s9 = (ggx3' - x3gg') + (ffx4' - x4ff') + (eex5' - x5ee') + (ddx6' - x6dd')$$
$$(ccx7' - x7cc') + (bbx8' - x8bb') + (aax9' - x9aa')$$
$$s10 = (ggx4' - x4gg') + (ffx5' - x5ff') + (eex6' - x6ee') + (ddx7' - x7dd') +$$
$$(ccx8' - x8cc') + (bbx9' - x9bb') + (aax10' - x10aa')$$
$$s11 = (ggx5' - x5gg') + (ffx6' - x6ff') + (eex7' - x7ee') +$$
$$(ddx8' - x8dd') + (ccx9' - x9cc') + (bbx10' - x10bb')$$
$$s12 = (ggx6' - x6gg') + (ffx7' - x7ff') + (eex8' - x8ee') + (ddx9' - x9dd') +$$
$$(ccx10' - x10cc')$$
$$s13 = (ggx7' - x7gg') + (ffx8' - x8ff') + (eex9' - x9ee') + (ddx10' - x10dd')$$
$$s14 = (ggx8' - x8gg') + (ffx9' - x9ff') + (eex10' - x10ee')$$
$$s15 = (ggx9' - x9gg') + (ffx10' - x10ff')$$
$$s16 = (ggx10' - x10gg')$$

Again compute $[(\text{Eq. C6}) \times (aauw^6 + bbuw^5 + ccuw^4 + dduw^3 + eeuw^2 + ffuw + ggu + hhw^6 + iiw^5 + jjw^4 + kkw^3 + llw^2 + mmw + nn)] - [(\text{Eq. C5}) \times (aa'uw^6 + bb'uw^5 + cc'uw^4 + dd'uw^3 + ee'uw^2 + ff'uw + gg'u + hh'w^6 + ii'w^5 + jj'w^4 + kk'w^3 + ll'w^2 + mm'w + nn')]$
to reduce u in Eqs. C5 and C6 to linear:

$$\begin{aligned} &a'uw^{15} + b'uw^{14} + c'uw^{13} + d'uw^{12} + e'uw^{11} + f'uw^{10} + \\ &g'uw^9 + h'uw^8 + i'uw^7 + j'uw^6 + k'uw^5 + l'uw^4 + \\ &m'uw^3 + n'uw^2 + o'uw + p'u + t1w^{18} + t2w^{17} + \\ &t3w^{16} + z1w^{15} + z2w^{14} + z3w^{13} + z4w^{12} + z5w^{11} + \\ &z6w^{10} + z7w^9 + z8w^8 + z9w^7 + z10w^6 + z11w^5 + \\ &z12w^4 + z13w^3 + z14w^2 + z15w + z16 = 0 \end{aligned} \tag{C8}$$

where

$$a' = (x1aa' - aax1')$$
$$b' = (x1bb' - bbx1') + (x2aa' - aax2')$$
$$c' = (x1cc' - ccx1') + (x2bb' - bbx2') + (x3aa' - aax3')$$
$$d' = (x1dd' - ddx1') + (x2cc' - ccx2') + (x3bb' - bbx3') + (x4aa' - aax4')$$
$$e' = (x1ee' - eex1') + (x2dd' - ddx2') + (x3cc' - ccx3') + (x4bb' - bbx4') + (x5aa' - aax5')$$
$$f' = (x1ff' - ffx1') + (x2ee' - eex2') + (x3dd' - ddx3') + (x4cc' - ccx4') + (x5bb' - bbx5') + (x6aa' - aax6')$$
$$g' = (x1gg' - ggx1') + (x2ff' - ffx2') + (x3ee' - eex3') + (x4dd' - ddx4') + (x5cc' - ccx5') + (x6bb' - bbx6') + (x7aa' - aax7')$$
$$h' = (x2gg' - ggx2') + (x3ff' - ffx3') + (x4ee' - eex4') + (x5dd' - ddx5') + (x6cc' - ccx6') + (x7bb' - bbx7') + (x8aa' - aax8')$$
$$i' = (x3gg' - ggx3') + (x4ff' - ffx4') + (x5ee' - eex5') + (x6dd' - ddx6') + (x7cc' - ccx7') + (x8bb' - bbx8') + (x9aa' - aax9')$$
$$j' = (x4gg' - ggx4') + (x5ff' - ffx5') + (x6ee' - eex6') + (x7dd' - ddx7') + (x8cc' - ccx8') + (x9bb' - bbx9') + (x10aa' - aax10')$$
$$k' = (x5gg' - ggx5') + (x6ff' - ffx6') + (x7ee' - eex7') + (x8dd' - ddx8') + (x9cc' - ccx9') + (x10bb' - bbx10')$$
$$l' = (x6gg' - ggx6') + (x7ff' - ffx7') + (x8ee' - eex8') + (x9dd' - ddx9') + (x10cc' - ccx10')$$
$$m' = (x7gg' - ggx7') + (x8ff' - ffx8') + (x9ee' - eex9') + (x10dd' - ddx10')$$
$$n' = (x8gg' - ggx8') + (x9ff' - ffx9') + (x10ee' - eex10')$$
$$o' = (x9gg' - ggx9') + (x10ff' - ffx10')$$
$$p' = (x10gg' - ggx10')$$
$$t1 = (x1y1')$$
$$t2 = (x1y2') + (x2y1')$$
$$t3 = (x1y3') + (x2y2') + (x3y1')$$
$$z1 = (x2y3') + (x3y2') + (x4y1') + (x1hh' - hhx1')$$
$$z2 = (x3y3') + (x4y2') + (x5y1') + (x1ii' - iix1') + (x2hh' - hhx2')$$
$$z3 = (x4y3') + (x5y2') + (x6y1') + (x1jj' - jjx1') + (x2ii' - iix2') + (x3hh' - hhx3')$$
$$z4 = (x5y3') + (x6y2') + (x7y1') + (x1kk' - kkx1') + (x2jj' - jjx2') + (x3ii' - iix3') + (x4hh' - hhx4')$$

$$z5 = (x6y3') + (x7y2') + (x8y1') + (x1ll' - llx1') + (x2kk' - kkx2') + (x3jj' - jjx3') + (x4ii' - iix4') + (x5hh' - hhx5')$$

$$z6 = (x7y3') + (x8y2') + (x9y1') + (x1mm' - mmx1') + (x2ll' - llx2') + (x3kk' - kkx3') + (x4jj' - jjx4') + (x5ii' - iix5') + (x6hh' - hhx6')$$

$$z7 = (x8y3') + (x9y2') + (x10y1') + (x1nn' - nnx1') + (x2mm' - mmx2') + (x3ll' - llx3') + (x4kk' - kkx4') + (x5jj' - jjx5') + (x6ii' - iix6') + (x7hh' - hhx7')$$

$$z8 = (x9y3') + (x10y2') + (x2nn' - nnx2') + (x3mm' - mmx3') + (x4ll' - llx4') + (x5kk' - kkx5') + (x6jj' - jjx6') + (x7ii' - iix7') + (x8hh' - hhx8')$$

$$z9 = (x10y3') + (x3nn' - nnx3') + (x4mm' - mmx4') + (x5ll' - llx5') + (x6kk' - kkx6') + (x7jj' - jjx7') + (x8ii' - iix8') + (x9hh' - hhx9')$$

$$z10 = (x4nn' - nnx4') + (x5mm' - mmx5') + (x6ll' - llx6') + (x7kk' - kkx7') + (x8jj' - jjx8') + (x9ii' - iix9') + (x10hh' - hhx10')$$

$$z11 = (x5nn' - nnx5') + (x6mm' - mmx6') + (x7ll' - llx7') + (x8kk' - kkx8') + (x9jj' - jjx9') + (x10ii' - iix10')$$

$$z12 = (x6nn' - nnx6') + (x7mm' - mmx7') + (x8ll' - llx8') + (x9kk' - kkx9') + (x10jj' - jjx10')$$

$$z13 = (x7nn' - nnx7') + (x8mm' - mmx8') + (x9ll' - llx9') + (x10kk' - kkx10') +$$

$$z14 = (x8nn' - nnx8') + (x9mm' - mmx9') + (x10ll' - llx10')$$

$$z15 = (x9nn' - nnx9') + (x10mm' - mmx10')$$

$$z16 = (x10nn' - nnx10')$$

Finally compute $[(\text{Eq. C8}) \times (aw^{15} + bw^{14} + cw^{13} + dw^{12} + ew^{11} + fw^{10} + gw^9 + hw^8 + iw^7 + jw^6 + kw^5 + lw^4 + mw^3 + nw^2 + ow + p)] - [(\text{Eq. C7}) \times (a'w^{15} + b'w^{14} + c'w^{13} + d'w^{12} + e'w^{11} + f'w^{10} + g'w^9 + h'w^8 + i'w^7 + j'w^6 + k'w^5 + l'w^4 + m'w^3 + n'w^2 + o'w + p')]$ to eliminate u in Eqs. C7 and C8, and yield a 33th order polynomial :

$$q34w^{33} + q33w^{32} + q32w^{31} + q31w^{30} + q30w^{29} + q29w^{28} + q28w^{27} + q27w^{26} + q26w^{25} + q25w^{24} + q24w^{23} + q23w^{22} + q22w^{21} + q21w^{20} + q20w^{19} + q19w^{18} + q18w^{17} + q17w^{16} + q16w^{15} + q15w^{14} + q14w^{13} + q13w^{12} + q12w^{11} + q11w^{10} + q10w^9 + q9w^8 + q8w^7 + q7w^6 + q6w^5 + q5w^4 + q4w^3 + q3w^2 + q2w + q1 = 0 \quad (C9)$$

where

$$q34 = t1a$$
$$q33 = t1b + t2a$$
$$q32 = t1c + t2b + t3a$$
$$q31 = t1d + t2c + t3b + (z1a - s1a')$$
$$q30 = t1e + t2d + t3c + (z1b - s1b') + (z2a - s2a')$$
$$q29 = t1f + t2e + t3d + (z1c - s1c') + (z2b - s2b') + (z3a - s3a')$$
$$\begin{aligned} q28 = {} & t1g + t2f + t3e + (z1d - s1d') + (z2c - s2c') + (z3b - s3b') \\ & + (z4a - s4a') \end{aligned}$$
$$\begin{aligned} q27 = {} & t1h + t2g + t3f + (z1e - s1e') + (z2d - s2d') + (z3c - s3c') \\ & + (z4b - s4b') + (z5a - s5a') \end{aligned}$$
$$\begin{aligned} q26 = {} & t1i + t2h + t3g + (z1f - s1f') + (z2e - s2e') + (z3d - s3d') \\ & + (z4c - s4c') + (z5b - s5b') + (z6a - s6a') \end{aligned}$$
$$\begin{aligned} q25 = {} & t1j + t2i + t3h + (z1g - s1g') + (z2f - s2f') + (z3e - s3e') \\ & + (z4d - s4d') + (z5c - s5c') + (z6b - s6b') + (z7a - s7a') \end{aligned}$$
$$\begin{aligned} q24 = {} & t1k + t2j + t3i + (z1h - s1h') + (z2g - s2g') + (z3f - s3f') \\ & + (z4e - s4e') + (z5d - s5d') + (z6c - s6c') + (z7b - s7b') \\ & + (z8a - s8a') \end{aligned}$$
$$\begin{aligned} q23 = {} & t1l + t2k + t3j + (z1i - s1i') + (z2h - s2h') + (z3g - s3g') \\ & + (z4f - s4f') + (z5e - s5e') + (z6d - s6d') + (z7c - s7c') \\ & + (z8b - s8b') + (z9a - s9a') \end{aligned}$$
$$\begin{aligned} q22 = {} & t1m + t2l + t3k + (z1j - s1j') + (z2i - s2i') + (z3h - s3h') \\ & + (z4g - s4g') + (z5f - s5f') + (z6e - s6e') + (z7d - s7d') \\ & + (z8c - s8c') + (z9b - s9b') + (z10a - s10a') \end{aligned}$$
$$\begin{aligned} q21 = {} & t1n + t2m + t3l + (z1k - s1k') + (z2j - s2j') + (z3i - s3i') \\ & + (z4h - s4h') + (z5g - s5g') + (z6f - s6f') + (z7e - s7e') \\ & + (z8d - s8d') + (z9c - s9c') + (z10b - s10b') + (z11a - s11a') \end{aligned}$$
$$\begin{aligned} q20 = {} & t1o + t2n + t3m + (z1l - s1l') + (z2k - s2k') + (z3j - s3j') \\ & + (z4i - s4i') + (z5h - s5h') + (z6g - s6g') + (z7f - s7f') \\ & + (z8e - s8e') + (z9d - s9d') + (z10c - s10c') + (z11b - s11b') \\ & + (z12a - s12a') \end{aligned}$$

$$
\begin{aligned}
q19 = {} & t1p + t2o + t3n + (z1m - s1m') + (z2l - s2l') + (z3k - s3k') \\
& + (z4j - s4j') + (z5i - s5i') + (z6h - s6h') + (z7g - s7g') \\
& + (z8f - s8f') + (z9e - s9e') + (z10d - s10d') + (z11c - s11c') \\
& + (z12b - s12b') + (z13a - s13a')
\end{aligned}
$$

$$
\begin{aligned}
q18 = {} & t2p + t3o + (z1n - s1n') + (z2m - s2m') + (z3l - s3l') \\
& + (z4k - s4k') + (z5j - s5j') + (z6i - s6i') + (z7h - s7h') \\
& + (z8g - s8g') + (z9f - s9f') + (z10e - s10e') + (z11d - s11d') \\
& + (z12c - s12c') + (z13b - s13b') + (z14a - s14a')
\end{aligned}
$$

$$
\begin{aligned}
q17 = {} & t3p + (z1o - s1o') + (z2n - s2n') + (z3m - s3m') + (z4l - s4l') \\
& + (z5k - s5k') + (z6j - s6j') + (z7i - s7i') + (z8h - s8h') \\
& + (z9g - s9g') + (z10f - s10f') + (z11e - s11e') + (z12d - s12d') \\
& + (z13c - s13c') + (z14b - s14b')(z15a - s15a')
\end{aligned}
$$

$$
\begin{aligned}
q16 = {} & (z1p - s1p') + (z2o - s2o') + (z3n - s3n') + (z4m - s4m') \\
& + (z5l - s5l') + (z6k - s6k') + (z7j - s7j') + (z8i - s8i') \\
& + (z9h - s9h') + (z10g - s10g') + (z11f - s11f') + (z12e - s12e') \\
& + (z13d - s13d') + (z14c - s14c')(z15b - s15b') + (z16a - s16a')
\end{aligned}
$$

$$
\begin{aligned}
q15 = {} & (z2p - s2p') + (z3o - s3o') + (z4n - s4n') + (z5m - s5m') \\
& + (z6l - s6l') + (z7k - s7k') + (z8j - s8j') + (z9i - s9i') \\
& + (z10h - s10h') + (z11g - s11g') + (z12f - s12f') + (z13e - s13e') \\
& + (z14d - s14d') + (z15c - s15c')(z16b - s16b')
\end{aligned}
$$

$$
\begin{aligned}
q14 = {} & (z3p - s3p') + (z4o - s4o') + (z5n - s5n') + (z6m - s6m') \\
& + (z7l - s7l') + (z8k - s8k') + (z9j - s9j') + (z10i - s10i') \\
& + (z11h - s11h') + (z12g - s12g') + (z13f - s13f') + (z14e - s14e') \\
& + (z15d - s15d') + (z16c - s16c')
\end{aligned}
$$

$$
\begin{aligned}
q13 = {} & (z4p - s4p') + (z5o - s5o') + (z6n - s6n') + (z7m - s7m') \\
& + (z8l - s8l') + (z9k - s9k') + (z10j - s10j') + (z11i - s11i') \\
& + (z12h - s12h') + (z13g - s13g') + (z14f - s14f') + (z15e - s15e') \\
& + (z16d - s16d')
\end{aligned}
$$

$$
\begin{aligned}
q12 = {} & (z5p - s5p') + (z6o - s6o') + (z7n - s7n') + (z8m - s8m') \\
& + (z9l - s9l') + (z10k - s10k') + (z11j - s11j') + (z12i - s12i') \\
& + (z13h - s13h') + (z14g - s14g') + (z15f - s15f') + (z16e - s16e')
\end{aligned}
$$

$$
\begin{aligned}
q11 = {} & (z6p - s6p') + (z7o - s7o') + (z8n - s8n') + (z9m - s9m') \\
& + (z10l - s10l') + (z11k - s11k') + (z12j - s12j') + (z13i - s13i') \\
& + (z14h - s14h') + (z15g - s15g') + (z16f - s16f')
\end{aligned}
$$

$$q10 = (z7p - s7p') + (z8o - s8o') + (z9n - s9n') + (z10m - s10m') + (z11l - s11l') + (z12k - s12k') + (z13j - s13j') + (z14i - s14i') + (z15h - s15h') + (z16g - s156')$$

$$q9 = (z8p - s8p') + (z9o - s9o') + (z10n - s10n') + (z11m - s11m') + (z12l - s12l') + (z13k - s13k') + (z14j - s14j') + (z15i - s15i') + (z16h - s16h')$$

$$q8 = (z9p - s9p') + (z10o - s10o') + (z11n - s11n') + (z12m - s12m') + (z13l - s13l') + (z14k - s14k') + (z15j - s15j') + (z16i - s16i')$$

$$q7 = (z10p - s10p') + (z11o - s11o') + (z12n - s12n') + (z13m - s13m') + (z14l - s14l') + (z15k - s15k') + (z16j - s16j') +$$

$$q6 = (z11p - s11p') + (z12o - s12o') + (z13n - s13n') + (z14m - s14m') + (z15l - s15l') + (z16k - s16k')$$

$$q5 = (z12p - s12p') + (z13o - s13o') + (z14n - s14n') + (z15m - s15m') + (z16l - s16l')$$

$$q4 = (z13p - s13p') + (z14o - s14o') + (z15n - s15n') + (z16m - s16m') +$$

$$q3 = (z14p - s14p') + (z15o - s15o') + (z16n - s16n')$$

$$q2 = (z15p - s15p') + (z16o - s16o')$$

$$q1 = (z16p - s16p')$$

Appendix D : Program TTBSUR

```
C-----------------------------------------------
C
C      SUBROUTINE: TTBSUR
C
C      DESCRIPTION: MAIN PROGRAM FOR INTERSECTION
C
C INPUT:
C      ISURF         = DUMMY VARIABLE
C
C OUTPUT:
C
C      C. K. WONG
C      9/12/89
C

      SUBROUTINE TTBSUR (ISURF)

      INTEGER ISURF
      REAL PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER NPATCH,NPOINT,NHULL
     +        ,IEND1(2),IEND2(2),IHUL1(2),IHUL2(2)

      PARAMETER (NPATCH=20,NPOINT=100,NHULL=150)

      REAL TTUWPT(4,4,NPATCH),UWO(4)
     +    ,HULL1(2,3,NHULL),HULL2(2,3,NHULL)


      CHARACTER FNAME*10


C ACSYNT MODEL : 1 , INTERSECTION OF TWO B-SPLINE PATCHES : 2

      WRITE(6,*)'INPUT CHOICE:'
      WRITE(6,*)'MULTIPLE PATCH -->1, SINGLE PATCH -->2 '
      READ(5,*)ICH
```

```
      IF(ICH.EQ.1) THEN

C  ------> MULTIPLE PATCH <---------
         CALL MULPA2

         RETURN
      ENDIF


C  ------> SINGLE PATCH <---------

      WRITE(6,*)'INPUT THE FILE NAME'
      READ(5,222)FNAME
 222  FORMAT(A10)
 333  FORMAT(A2)



C READ IN THE SURFACE DATA

      OPEN(UNIT=40,FILE=FNAME)
      READ(40,*)ITYPE
      READ(40,*)

      DO 22 I=1,3
         DO 33 J=1,4
              READ(40,*)PKL1(J-1,0,I),PKL1(J-1,1,I),
     +                 PKL1(J-1,2,I),PKL1(J-1,3,I)

 33      CONTINUE
 22   CONTINUE

      READ(40,*)

      DO 44 I=1,3
         DO 55 J=1,4
              READ(40,*)PKL2(J-1,0,I),PKL2(J-1,1,I),
     +                 PKL2(J-1,2,I),PKL2(J-1,3,I)
 55      CONTINUE
 44   CONTINUE

C DRAW THE SURFACES
      CALL DSURN2(PKL1,10,10)
      CALL DSURN2(PKL2,10,10)

C DECLARE THE STARTING PARAMETRIC VALUES
      DO 10 I=1,4
         UWO(I)=0.
 10   CONTINUE


C SOLVE THE INTERSECTION
      IPATCH=0
      CALL BRINT(PKL1,PKL2,UWO,IPATCH,TTUWPT)

C SORT THE  INTERSECTION POINTS ON BOTH SURFACES, AND
C DRAW THE CURVES

      IF(IPATCH.EQ.0) RETURN
      CALL SORTUW2(PKL1,PKL2,IPATCH,TTUWPT,IEND1,IEND2
     +                  ,IHUL1,IHUL2,HULL1,HULL2)


      RETURN
      END
C--------------------------------------------------
```

```
C
C       SUBROUTINE: MULPA2
C
C       DESCRIPTION: INTERSECTION ALGORITHM FOR ACSYNT MODEL
C
C INPUT: NONE
C
C OUTPUT: NONE
C
C       C. K. WONG
C       9/12/89
C

        SUBROUTINE MULPA2

        PARAMETER(ICOMP=15)
        REAL COMHULL(50,50,3,ICOMP),PKL(0:3,0:3,3)

        INTEGER NCR(ICOMP),PTCR(ICOMP),L,M,N


C OPEN THE MODEL FILE
        OPEN(UNIT=20,FILE='SRMODEL')

C READ IN NUMBER OF COMPONENTS
C       INUMP=NO. OF COMPONENTS

        READ(20,*)INUMP
        WRITE(6,*)'INUMP-- ',INUMP

C READ IN DATA FOR EACH COMPONENT
C       NCR    = NO. OF CROSS SECTIONS
C       PTCR   = NO. OF POINTS PER CROSS SECTION
C       COMHULL= CONTROL HULL MATRIC

        DO 5 IC=1,INUMP
           CALL READCOM (IC,NCR(IC),PTCR(IC),COMHULL)


C DRAW THE COMPONENT
           II=-1
           DO 10 I=1,NCR(IC)-1
                II=II+1

                JJ=-1
                DO 20 J=1,PTCR(IC)-1
                  JJ=JJ+1
                  DO 30 K=1,4
                     DO 40 L=1,4
                          DO 50 M=1,3
                               PKL(K-1,L-1,M)=COMHULL(II+K,JJ+L,M,IC)
 50                       CONTINUE
 40                  CONTINUE
 30               CONTINUE

                  CALL DSURN(PKL,10,2)

 20             CONTINUE
 10        CONTINUE


 5      CONTINUE

C FIND THE INTERSECTIONS OF THE COMPONENTS

        CALL MULTINT(COMHULL,NCR,PTCR)
```

```
      RETURN
      END
C-------------------------------------------------
C
C     SUBROUTINE: READCOM
C
C     DESCRIPTION: READ IN THE ACSYNT MODEL
C
C INPUT:
C     IC            = COMPONENT NO.
C
C OUTPUT:
C     NCR           = NO. OF CROSS SECTIONS
C     PTCR          = NO. OF POINTS PER CROSS SECTION
C     COMHULL       = CONTROL HULL MATRIC
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE READCOM (IC,NCR,PTCR,COMHULL)

      REAL COMHULL(50,50,3,*)
      INTEGER NCR,PTCR

C COMPONENT NO IN ACSYNT DATA FILE
      READ(20,*)ICONO

C NO. OF CROSS SECTION
      READ(20,*)NCR

C NO. OF PTS PER CROSS SECTION
      READ(20,*)PTCR


C READ IN THE CONTROL HULL
      DO 10 I=1,NCR+2
         DO 20 J=1,PTCR+2
              READ(20,*)COMHULL(I,J,1,IC),COMHULL(I,J,2,IC),
     +                  COMHULL(I,J,3,IC)
 20      CONTINUE
 10   CONTINUE



      RETURN
      END
C-------------------------------------------------
C
C     SUBROUTINE: MULTINT
C
C     DESCRIPTION: FIND THE INTERSECTIONS OF ACSYNT MODEL
C
C INPUT:
C     COMHULL       = CONTROL HULLS OF THE AIRCRAFT COMPONENTS
C     NCR           = NO. OF CROSS SECTIONS IN THE COMPONENTS
C     PTCR          = NO. OF POINTS PER CROSS SECTION
C                     IN THE COMPONENTSC
C
C OUTPUT: NONE
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE MULTINT(COMHULL,NCR,PTCR)
```

```
      REAL COMHULL(50,50,3,*)
     +     ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER NPATCH,NPOINT,NCR(*),PTCR(*)
     +        ,IEND1(2),IEND2(2),IHUL1(2),IHUL2(2)

      PARAMETER(NPATCH=150,NPOINT=300)

      REAL TTUWPT(4,4,NPATCH),UWO(4)
     +     ,HULL1(2,3,NPOINT),HULL2(2,3,NPOINT)


      LOGICAL INTALL

C PROMPT FOR CHOICE OF INTERSECTION
         WRITE(6,*)'MAKE YOUR CHOICE'
         WRITE(6,*)'0= HOZ_TAIL#1 AND AFT_BODY'
         WRITE(6,*)'1= HOZ_TAIL#2 AND AFT_BODY'
         WRITE(6,*)'2= WING#1 AND FUSELAGE'
         WRITE(6,*)'3= WIND#2 AND FUSELAGE'
         WRITE(6,*)'4= VER_TAIL AND AFT_BODY'
         WRITE(6,*)'5= RETURN'
         WRITE(6,*)'6= ALL OF ABOVE'
         READ(5,*)INT

C PROCESS ALL THE INTERSECTION
      IF(INT.EQ.6) THEN
         INTALL=.TRUE.
         INT=0
        ELSE
         INTALL=.FALSE.
      ENDIF


C HOZ_TAIL#1 AND AFT_BODY
 5    IF(INT.EQ.0) THEN
         IC1=6
         IC2=3
         INT=1

C HOZ_TAIL#2 AND AFT_BODY
       ELSEIF(INT.EQ.1) THEN
         IC1=7
         IC2=3
         INT=2

C FUELSLUG AND WING#1
       ELSEIF(INT.EQ.2) THEN
         IC1=4
         IC2=2
         INT=3

C FUELSLUG AND WING#2
       ELSEIF(INT.EQ.3) THEN
         IC1=5
         IC2=2
         INT=4

C VER_TAIL AND AFT_BODY
       ELSEIF(INT.EQ.4) THEN
         IC1=8
         IC2=3
         INT=5

C RETURN
```

```
        ELSEIF(INT.EQ.5) THEN
          RETURN
        ENDIF

C WHICH INTERSECTION ?
          WRITE(6,*)'INT= ',INT-1
          ICOUNT=0

C INITIALIZE NO. OF INTERSECTION SET (1 SET = 4 POINTS)
          IPATCH=0

C GET A SINGLE PATCH FROM THE FIRST COMPONENT
          II=-1
          DO 10 I=1,NCR(IC1)-1
               II=II+1

C STARTING PARAMETRIC VALUE ON THE FIRST COMPONENT (W)
               UWO(2)=REAL(II)

               JJ=-1
               DO 20 J=1,PTCR(IC1)-1
                 JJ=JJ+1

C STARTING PARAMETRIC VALUE ON THE FIRST COMPONENT (U)
                 UWO(1)=REAL(JJ)

                 DO 30 K=1,4
                    DO 40 L=1,4
                         DO 50 M=1,3
                              PKL1(K-1,L-1,M)=COMHULL(II+K,JJ+L,M,IC1)
 50                      CONTINUE
 40                 CONTINUE
 30              CONTINUE

C COUNTER FOR PATCHES THAT BEEN PROCESSED (FOR WING & TAIL ONLY)
C ** TO REDUCE NO. OF PROCESS **
                 ICOUNT=ICOUNT+1
                 IF(ICOUNT.GT.8) GOTO 10

C INTERSECT WITH THE SECOND COMPONENT
                 CALL MULTI2 (IC2,PKL1,UWO,COMHULL
     +                     ,NCR,PTCR,INT,IPATCH,TTUWPT)

 20            CONTINUE
 10       CONTINUE

C SORT THE INTERSECTION POINTS ON BOTH SURFACE, AND
C RETURN WITH THE END CONDITIONS AND THE CONTROL HULLS FOR
C THE INTERSECTION CURVES

       IF (IPATCH.GT.0) THEN
          CALL SORT(IC1,IC2,COMHULL,IPATCH,TTUWPT
     +             ,IEND1,IEND2,IHUL1,IHUL2,HULL1,HULL2)
       ENDIF

C PROCESS ANOTHER INTERSECTION?
       IF(INTALL) GOTO 5

       RETURN
       END
C--------------------------------------------------
C
C      SUBROUTINE: MULTI2
C
C      DESCRIPTION: INTERSECT A SINGLE PATCH FROM THE
C                   FIRST COMPONENT WITH THE SECOND COMPONENT
C
```

```
C INPUT:
C      IC2           = IDENTIFIER FOR THE SECOND COMPONENT
C      PKL1          = SINGLE PATCH ON THE FIRST COMPONENT
C      UWO           = STARTING PARAMETRIC VALUES ON BOTH
C                      INTERSECTING PATCHES
C      COMHULL       = CONTROL HULLS OF THE AIRCRAFT COMPONENTS
C      NCR           = NO. OF CROSS SECTIONS IN THE COMPONENTS
C      PTCR          = NO. OF POINTS PER CROSS SECTION
C                      IN THE COMPONENTSC
C      INT           = INTERSECTION IDENTIFIER
C
C OUTPUT:
C      IPATCH        = NO. OF INTERSECTION SET
C      TTUWPT        = INTERSECTION SETS ( 4 POINTS IN A SET)
C
C      C. K. WONG
C      9/12/89
C

      SUBROUTINE MULTI2 (IC2,PKL1,UWO,COMHULL
     +                    ,NCR,PTCR,INT,IPATCH,TTUWPT)

      REAL COMHULL(50,50,3,*)
     +    ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)
     +    ,TTUWPT(4,4,*),UWO(4)

      INTEGER NCR(*),PTCR(*),L,M,N

C GET A SINGLE PATCH FROM THE SECOND COMPONENT

         II=-1
         DO 10 I=1,NCR(IC2)-1
              II=II+1

C STARTING PARAMETRIC VALUE ON THE SECOND COMPONENT (U)

              UWO(4)=REAL(II)

              JJ=-1
              DO 20 J=1,PTCR(IC2)-1
                JJ=JJ+1
C STARTING PARAMETRIC VALUE ON THE SECOND COMPONENT (W)

                UWO(3)=REAL(JJ)

                DO 30 K=1,4
                   DO 40 L=1,4
                        DO 50 M=1,3
                             PKL2(K-1,L-1,M)=COMHULL(II+K,JJ+L,M,IC2)
 50                     CONTINUE
 40                CONTINUE
 30             CONTINUE


C INTERSECTION OF THE SINGLE PATCHES

                        CALL BRINT(PKL1,PKL2,UWO,IPATCH,TTUWPT)

 20           CONTINUE
 10      CONTINUE

      RETURN
      END
C----------------------------------------------------------
C
C      SUBROUTINE: SORT
C
```

```
C     DESCRIPTION: TO SORT AND INVERT THE INTERSECTION POINTS
C                  FOR B-SPLINE CURVES
C
C INPUT:
C     IC1,IC2        = COMPONENT IDENTIFIER
C     COMHULL        = CONTROL HULLS OF THE COMPONENTS
C     IPATCH         = NO. OF INTERSECTION SET
C     TTUWPT         = INTERSECTION SETS
C
C OUTPUT:
C     IEND1,IEND2    = END CONDITIONS OF THE INTERSECTION CURVES
C     IHUL1,IHUL2    = NO. OF CONTROL POINTS OF THE INTERSECTION
C                      CURVES
C     HULL1,HULL2    = CONTROL POINTS FOR THE INTERSECTION CURVES
C
C
C     C. K. WONG
C     2/20/90
C

      SUBROUTINE SORT(IC1,IC2,COMHULL,IPATCH,TTUWPT,IEND1,IEND2
     +               ,IHUL1,IHUL2,HULL1,HULL2)


      REAL COMHULL(50,50,3,*)

      INTEGER NPOINT,ICUR(2),IEND1(2),IEND2(2),IHUL1(2),IHUL2(2)

      PARAMETER(NPOINT=300)

      REAL TTUWPT(4,4,*),CUR(2,2,NPOINT)
     +    ,PTS(3,NPOINT),HULL(3,NPOINT)
     +    ,HULL1(2,3,*),HULL2(2,3,*)


C INITIALIZE THE NO. OF CONTROL HULL POINTS AND END CONDITIONS
         DO 10 I=1,2
           IHUL1(I)=0
           IHUL2(I)=0
           IEND1(I)=2
           IEND2(I)=2
 10      CONTINUE


C SORT THE INTERSECTION POINTS ON FIRST OR FIRST & SECOND SURFACE
C  INOSUR:    1= FIRST
C             2= FIRST AND SECOND

      INOSUR=1
      DO 1000 ISNO=1,INOSUR


C SORT THE INTERSECTION POINTS; RETURN WITH MAXIMUM TWO
C SORTED SETS OF PARAMETRIC VALUES

      CALL SORTTUW(ISNO,IPATCH,TTUWPT,CUR,ICUR)

C PROCESS THE SORTED SET
      DO 1100 ICNO=1,2

C NO ELEMENT IN THE SET
         IF(ICUR(ICNO).EQ.0) GOTO 1100

C CONVERT THE SET INTO PTS FOR SUBROUTINE INVPTS
         DO 100 I=1,ICUR(ICNO)
            PTS(1,I)=CUR(ICNO,1,I)
            PTS(2,I)=CUR(ICNO,2,I)
```

```
            PTS(3,I)=1.
 100     CONTINUE

C CHECK THE END CONDITION: FIND THE DISTANCE BETWEEN
C THE END POINTS OF THE SET

         TOL=.1E-10
         DIS=0.
         DO 110 I=1,3
            DIS1=ABS(PTS(I,1)-PTS(I,ICUR(ICNO)))
            IF(DIS1.LT.TOL) THEN
               DIS1=0.
              ELSE
               DIS1=DIS1**2
            ENDIF
            DIS=DIS+DIS1
 110     CONTINUE

         TOL2=.1
         DIS=DIS**(.5)

         IF(DIS.LT.TOL2) THEN

C CLOSED SET
            IEND=1

C STORE THE END CONDITION
            IF(ISNO.EQ.1) THEN
               IEND1(ICNO)=1
              ELSE
               IEND2(ICNO)=1
            ENDIF

           ELSE

C OPEN SET
            IEND=2
         ENDIF


C INVERT THE SET FOR A CUBIC B-SPLINE CURVE
C IEND= END CONDITION (1: CLOSE, 2: OPEN)
         CALL INVPTS(IEND,ICUR(ICNO),PTS,HULL)

C NO. OF POINTS  IN CONTROL HULL
         IHUL=ICUR(ICNO)+2

C STORE THE RESULT
         IF(ISNO.EQ.1) THEN
              IHUL1(ICNO)=IHUL
              DO 20 I=1,IHUL
                   DO 30 J=1,3
                      HULL1(ICNO,J,I)=HULL(J,I)
 30                CONTINUE
 20           CONTINUE
            ELSE
              IHUL2(ICNO)=IHUL
              DO 25 I=1,IHUL
                   DO 35 J=1,3
                      HULL2(ICNO,J,I)=HULL(J,I)
 35                CONTINUE
 25           CONTINUE
         ENDIF

C DRAW THE INTERSECTION LINE
         IF(ISNO.EQ.1) THEN
            IC=IC1
```

```
             ICOLOR=3

            ELSE
             IC=IC2
             ICOLOR=3
          ENDIF

          CALL DINTHUL(ICOLOR,IC,COMHULL,IEND
     +                   ,IHUL,HULL)

 1100     CONTINUE
 1000 CONTINUE


      RETURN
      END
C------------------------------------------------
C
C     SUBROUTINE: DINTHUL
C
C     DESCRIPTION: DRAW THE INTERSECTION CURVE
C
C INPUT:
C     ICOLOR        = COLOR NO.
C     IC            = COMPONENT NO.
C     COMHULL       = COMPONENTS
C     IEND          = END CONDITION OF THE CURVE
C     IHUL          = NO. OF CONTROL POINTS OF THE CURVE
C     HULL          = CONTROL POINTS OF THE CURVE
C
C OUTPUT: NONE
C
C     C. K. WONG
C     9/12/89
C
C

      SUBROUTINE DINTHUL(ICOLOR,IC,COMHULL
     +                  ,IEND,IHUL,HULL)


      REAL COMHULL(50,50,3,*),PKL(0:3,0:3,3)

      INTEGER NPOINT,NPT
      PARAMETER(NPOINT=300)

      REAL HULL(3,*),PTXYZ(3,NPOINT),P(3)
     +    ,TPST(1,3)

C FIND NPT POINTS IN X Y Z COOR. FROM THE INTERSECTION CURVE
      NPT=100
      REP=(1./REAL(NPT))
      U=-(REP)

      DO 200 IREPT=1,NPT+1

         U=U+REP

C FIND THE CORRESPONDING PARAMETRIC VALUE ON THE ORIGINAL SURFACE
         CALL PER4(IHUL,HULL,IEND,U,P)

C PARAMETRIC VALUE ON THE SURFACE FOR THE POINT
         JJ=INT(P(1))
         II=INT(P(2))

C THE PATCH WHERE THE POINT LOCATED
                DO 30 K=1,4
```

```
                  DO 40 L=1,4
                       DO 50 M=1,3
                            PKL(K-1,L-1,M)=COMHULL(II+K,JJ+L,M,IC)
 50                    CONTINUE
 40               CONTINUE
 30            CONTINUE

C REPARAMETRIZE THE POINT
         UU=P(2)-REAL(II)
         WW=P(1)-REAL(JJ)

C FIND THE POINT IN GLOBAL COOR.
         CALL FBPT(PKL,UU,WW,TPST,IF)

C STORE THE POINT IN PTXYZ
         DO 210 J=1,3
              PTXYZ(J,IREPT)=TPST(1,J)
 210     CONTINUE

 200  CONTINUE

C DRAW THE LINE
      CALL DLINEN(ICOLOR,NPT+1,PTXYZ)

      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: SORTTUW
C
C     DESCRIPTION: SORT THE INTERSECTION POINTS
C                  (ASSUME THERE ARE MAXIMUM TWO CURVES)
C
C INPUT:
C     ISURFNO      = SURFACE IDENTIFIER
C     IPATCH       = NO. OF INTERSECTION SET
C     TTUWPT       = INTERSECTION SETS
C
C OUTPUT:
C     CUR          = SORTED POINTS
C     ICUR         = NO. OF SORTED POINTS
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE SORTTUW(ISURFNO,IPATCH,TTUWPT,CUR,ICUR)

      INTEGER NPATCH
      PARAMETER(NPATCH=150)

      REAL TTUWPT(4,4,*),TT1(4,2,NPATCH)
     +    ,CUR(2,2,*),STPT(2,2)

      INTEGER ICUR(*)



C COPY THE ORIGINAL DATA INTO TT1 FOR SORTING
C   INOTT1= NO OF DATA SET IN TT1

      INOTT1=IPATCH

      DO 10 I=1,IPATCH
         DO 20 J=1,4

C SORT THE INTERSECTION PTS ON THE FIRST SURFACE
```

```
               IF(ISURFNO.EQ.1) THEN
                    TT1(J,1,I)=TTUWPT(J,1,I)
                    TT1(J,2,I)=TTUWPT(J,2,I)
                   ELSE

C SORT THE INTERSECTION PTS ON THE SECOND SURFACE
                    TT1(J,1,I)=TTUWPT(J,3,I)
                    TT1(J,2,I)=TTUWPT(J,4,I)
               ENDIF

 20       CONTINUE
 10    CONTINUE


C START TO SORT
C    INOC= SORTED SET IDENTIFIER
C    ICUR= NO. OF POINT ON THE SORTED SET


C INITIALIZE
      INOC=1
      ICUR(INOC)=4
      ICUR(2)=0

C USING THE FIRST DATA SET IN TT1 AS A STARTING POINT
C OF THE SORTED SET

 1000 DO 40 I=1,4
         DO 50 J=1,2
            CUR(INOC,J,I)=TT1(I,J,1)
 50      CONTINUE
 40    CONTINUE

C ERASE THE FIRST DATA SET FROM TT1
      CALL ERASETT(1,INOTT1,TT1)
      IF(INOTT1.EQ.0) return

C GET THE TWO END POINTS OF THE SORTED SET
 1500 CALL GETSTPT(INOC,CUR,ICUR,STPT)

C MATCH THE END POINTS WITH TT1
C    IDTT1= IDENTIFIER OF DATA SET THAT DOES MATCH

      CALL MATCHTT(STPT,INOTT1,TT1,IMATCH,IDTT1)

C FIND OUT WHAT KIND OF MATCH AND STORE THE RESULT

C IMATCH=0 : NO MATCH
      IF(IMATCH.EQ.0) THEN

C IF INOC=1 THEN START PROCESS SORTED SET NO 2
         IF(INOC.EQ.1) THEN
            INOC= 2
            ICUR(INOC)=4
            GOTO 1000
           ELSE

C MORE THEN TWO DISJOINTED SORTED SETS

           WRITE(6,*)'MORE THEN TWO DISJOINTED CURVES in SORTTUW'
           return

         ENDIF


C IMATCH=1 : TOP MATCH TOP
       ELSEIF(IMATCH.EQ.1) THEN
```

```
          CALL MATCH1(INOC,CUR,ICUR,TT1,IDTT1)

C IMATCH=2 : TOP MATCH BOTTOM
        ELSEIF(IMATCH.EQ.2) THEN
          CALL MATCH2(INOC,CUR,ICUR,TT1,IDTT1)

C IMATCH=3 : BOTTOM MATCH TOP
        ELSEIF(IMATCH.EQ.3) THEN
          CALL MATCH3(INOC,CUR,ICUR,TT1,IDTT1)

C IMATCH=3 : BOTTOM MATCH BOTTOM
        ELSEIF(IMATCH.EQ.4) THEN
          CALL MATCH4(INOC,CUR,ICUR,TT1,IDTT1)

      ENDIF

C ERASE THE MATCHED DATA SET FROM TT1
      CALL ERASETT(IDTT1,INOTT1,TT1)
      IF(INOTT1.NE.0) GOTO 1500



      RETURN
      END
C---------------------------------------------------
C
C     SUBROUTINE: ERASETT
C
C     DESCRIPTION: ERASE A DATA SET FROM TT1
C
C INPUT:
C     IDTT1         = DATA SET IDENTIFIER
C     INOTT1        = NO. OF DATA SET IN TT1
C
C OUTPUT:
C     TT1           = ERASED DATA SETS
C
C     C. K. WONG
C     9/12/89
C
C

      SUBROUTINE ERASETT(IDTT1,INOTT1,TT1)

      REAL TT1(4,2,*)

      ICO=0
      DO 10 I=1,INOTT1

C SKIP THE STORING PROCESS IF IT GETS TO THE
C DATA SET WITH IDTT1

         IF(I.EQ.IDTT1) GOTO 10
         ICO=ICO+1

         DO 20 J=1,4
              DO 30 K=1,2
                   TT1(J,K,ICO)=TT1(J,K,I)
 30           CONTINUE
 20      CONTINUE
 10   CONTINUE

      INOTT1=INOTT1-1


      RETURN
```

```
      END
C-------------------------------------------------
C
C     SUBROUTINE: GETSTPT
C
C     DESCRIPTION: GET THE STARTING POINTS FROM THE
C                  SORTED SET FOR FURTHER MATCHING
C
C INPUT:
C     INOC         = SORTED SET IDENTIFIER
C     CUR          = SORTED SET
C     ICUR         = NO. OF POINTS IN THE SORTED SET
C
C OUTPUT:
C     STPT         = STARTING POINTS
C
C     C. K. WONG
C     9/12/89
C
C
C

      SUBROUTINE GETSTPT(INOC,CUR,ICUR,STPT)

      REAL CUR(2,2,*), STPT(2,2)
      INTEGER ICUR(*)

C GET THE POINTS

      DO 10 I=1,2
         STPT(1,I)=CUR(INOC,I,1)
         STPT(2,I)=CUR(INOC,I,ICUR(INOC))
 10   CONTINUE

      RETURN
      END
C-------------------------------------------------
C
C     SUBROUTINE: MATCHTT
C
C     DESCRIPTION: MATCH THE STARTING POINTS WITH TT1
C
C INPUT:
C     STPT         = STARTING POINTS
C     INOTT1       = NO. OF DATA SET IN TT1
C     TT1          = DATA SET
C
C OUTPUT:
C     IMATCH       = TYPE OF MATCH
C     IDTT1        = DATA IDENTIFIER IN TT1 THAT MATCH
C                    THE STARTING POINTS
C
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE MATCHTT(STPT,INOTT1,TT1,IMATCH,IDTT1)

      REAL STPT(2,2), TT1(4,2,*)
     +    ,PT1(2),PT2(2)

      LOGICAL IM


C INITIALIZE
```

```
      TOL=.5
      IMATCH=0
      TMIN=100.
      IDTT1=1

      DO 10 I=1,INOTT1

C TOP MATCH TOP
         DO 20 J=1,2
              PT1(J)=STPT(1,J)
              PT2(J)=TT1(1,J,I)
 20      CONTINUE
         CALL PTDIS(I,PT1,PT2,TMIN,IM,IDTT1)
         IF(IM) IMATCH=1

C TOP MATCH BOTTOM
         DO 30 J=1,2
              PT1(J)=STPT(1,J)
              PT2(J)=TT1(4,J,I)
 30      CONTINUE
         CALL PTDIS(I,PT1,PT2,TMIN,IM,IDTT1)
         IF(IM) IMATCH=2

C BOTTOM MATCH TOP
         DO 40 J=1,2
              PT1(J)=STPT(2,J)
              PT2(J)=TT1(1,J,I)
 40      CONTINUE
         CALL PTDIS(I,PT1,PT2,TMIN,IM,IDTT1)
         IF(IM) IMATCH=3

C BOTTOM MATCH BOTTOM
         DO 50 J=1,2
              PT1(J)=STPT(2,J)
              PT2(J)=TT1(4,J,I)
 50      CONTINUE
         CALL PTDIS(I,PT1,PT2,TMIN,IM,IDTT1)
         IF(IM) IMATCH=4

 10   CONTINUE

      IF(TMIN.GT.TOL) IMATCH=0


      RETURN
      END
C---------------------------------------------------
C
C     SUBROUTINE: PTDIS
C
C     DESCRIPTION: FIND THE DISTANCE BETWEEN TWO POINTS
C
C INPUT:
C     I             = DATA IDENTIFIER IN TT1
C     PT1           = STARTING POINT IN THE SORTED SET
C     PT2           = DATA THAT WOULD BE MATCH TO PT1
C
C OUTPUT:
C     TMIN          = DISTANCE BETWEEN PT1 AND PT2
C     IM            = FLAG FOR SUCCESSFUL MATCH
C     IDTT1         = DATA IDENTIFIER THAT MATCHED
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE PTDIS(I,PT1,PT2,TMIN,IM,IDTT1)
```

```
      REAL PT1(2),PT2(2)
      LOGICAL IM


C INITIALIZATION

      IM=.FALSE.
      TOL=.1E-20
      DIS=0.

C FIND THE DISTANCE BETWEEN PT1 AND PT2

      DO 10 J=1,2
         DIS1=PT1(J)-PT2(J)
         IF(ABS(DIS1).LT.TOL) GOTO 10
         DIS=DIS+(DIS1**2)
 10   CONTINUE

C COMPARE THE DISTANCE

      DIS=DIS**(.5)
      IF(DIS.LT.TMIN) THEN
         IM=.TRUE.
         IDTT1=I
         TMIN=DIS
      ENDIF


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: MATCH1
C
C     DESCRIPTION: STORE THE MATCHED SET FROM TT1 INTO CUR
C                  (TOP MATCH TOP)
C
C INPUT:
C     INOC         = SORTED SET IDENTIFIER
C     CUR          = SORTED SET
C     ICUR         = NO. OF POINTS IN THE SORTED SET
C     TT1          = DATA SET
C     IDTT1        = MATCHED SET IDENTIFIER
C
C OUTPUT: NONE
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE MATCH1(INOC,CUR,ICUR,TT1,IDTT1)

      INTEGER NPT
      PARAMETER(NPT=250)

      REAL CUR(2,2,*), TT1(4,2,*)
     +    ,TCUR(2,NPT)

      INTEGER ICUR(*)

C COPY CUR TO TCUR

      DO 10 I=1,ICUR(INOC)
         DO 20 J=1,2
              TCUR(J,I+3)=CUR(INOC,J,I)
 20      CONTINUE
```

```
 10   CONTINUE

C STORE THE MATCHED SET IN TCUR

      DO 30 I=1,3
         DO 40 J=1,2
              TCUR(J,I)=TT1(5-I,J,IDTT1)
 40      CONTINUE
 30   CONTINUE


C COPY TCUR TO CUR

      ICUR(INOC)=ICUR(INOC)+3
      DO 50 I=1,ICUR(INOC)
         DO 60 J=1,2
              CUR(INOC,J,I)=TCUR(J,I)
 60      CONTINUE
 50   CONTINUE


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: MATCH2
C
C     DESCRIPTION: STORE THE MATCHED SET FROM TT1 INTO CUR
C                  (TOP MATCH BOTTOM)
C
C INPUT:
C     INOC          = SORTED SET IDENTIFIER
C     CUR           = SORTED SET
C     ICUR          = NO. OF POINTS IN THE SORTED SET
C     TT1           = DATA SET
C     IDTT1         = MATCHED SET IDENTIFIER
C
C OUTPUT: NONE
C
C     C. K. WONG
C     9/12/89

      SUBROUTINE MATCH2(INOC,CUR,ICUR,TT1,IDTT1)

      INTEGER NPT
      PARAMETER(NPT=250)

      REAL CUR(2,2,*), TT1(4,2,*)
     +    ,TCUR(2,NPT)

      INTEGER ICUR(*)

C COPY CUR TO TCUR

      DO 10 I=1,ICUR(INOC)
         DO 20 J=1,2
              TCUR(J,I+3)=CUR(INOC,J,I)
 20      CONTINUE
 10   CONTINUE

C STORE THE MATCHED SET IN TCUR

      DO 30 I=1,3
         DO 40 J=1,2
              TCUR(J,I)=TT1(I,J,IDTT1)
 40      CONTINUE
 30   CONTINUE
```

```
C COPY TCUR TO CUR

      ICUR(INOC)=ICUR(INOC)+3
      DO 50 I=1,ICUR(INOC)
         DO 60 J=1,2
              CUR(INOC,J,I)=TCUR(J,I)
 60      CONTINUE
 50   CONTINUE

      RETURN
      END
C---------------------------------------------------
C
C     SUBROUTINE: MATCH3
C
C     DESCRIPTION: STORE THE MATCHED SET FROM TT1 INTO CUR
C                  (BOTTOM MATCH TOP)
C
C INPUT:
C     INOC         = SORTED SET IDENTIFIER
C     CUR          = SORTED SET
C     ICUR         = NO. OF POINTS IN THE SORTED SET
C     TT1          = DATA SET
C     IDTT1        = MATCHED SET IDENTIFIER
C
C OUTPUT: NONE
C
C     C. K. WONG
C     9/12/89

      SUBROUTINE MATCH3(INOC,CUR,ICUR,TT1,IDTT1)

      REAL CUR(2,2,*), TT1(4,2,*)

      INTEGER ICUR(*)


      DO 10 I=1,3
         DO 20 J=1,2
              CUR(INOC,J,ICUR(INOC)+I)=TT1(I+1,J,IDTT1)
 20      CONTINUE
 10   CONTINUE

      ICUR(INOC)=ICUR(INOC)+3


      RETURN
      END
C---------------------------------------------------
C
C     SUBROUTINE: MATCH4
C
C     DESCRIPTION: STORE THE MATCHED SET FROM TT1 INTO CUR
C                  (BOTTOM MATCH TOP)
C
C INPUT:
C     INOC         = SORTED SET IDENTIFIER
C     CUR          = SORTED SET
C     ICUR         = NO. OF POINTS IN THE SORTED SET
C     TT1          = DATA SET
C     IDTT1        = MATCHED SET IDENTIFIER
C
C OUTPUT: NONE
C
```

```
C     C. K. WONG
C     9/12/89

      SUBROUTINE MATCH4(INOC,CUR,ICUR,TT1,IDTT1)

      REAL CUR(2,2,*), TT1(4,2,*)

      INTEGER ICUR(*)


      DO 10 I=1,3
         DO 20 J=1,2
              CUR(INOC,J,ICUR(INOC)+I)=TT1(4-I,J,IDTT1)
 20      CONTINUE
 10   CONTINUE

      ICUR(INOC)=ICUR(INOC)+3


      RETURN
      END
c-----------------------------------------------------------
c
c     Subroutine: invpts
c
c     Description: inverts a bunch of n points to return
c             (n+2) control points
c
c      Krishnan V. Kolady
c      08/15/89
c
c-----------------------------------------------------------
c

      subroutine invpts(end, npts, pts, hul)


C     END CONDITIONS:
C         END=1: CLOSE
C         END=2: OPEN


c      declare the variables
      integer end, npts
      real pts(3,*), hul(3,*)

c      initialize the control hull
      call inithl(end, npts, pts, hul)

c      find hul
      call fndhul(end, npts, pts, hul)

c      return
      return
      end
c
c-----------------------------------------------------------
c
c     Subroutine: inithl
c
c     Description: initializes the control hull
c
c
c      Krishnan V. Kolady
c      08/15/89
c
```

```
c-----------------------------------------------------------
c

      subroutine inithl(end, npts, pts, hul)


c      declare the variables
      integer end, npts
      real pts(3,*), hul(3,*)

c
      do 20 i=1,3
        do 10 j=2, npts+1
           hul(i,j) = pts(i,j-1)
 10     continue
c        initialize the end points
        if(end.eq.1) then
           hul(i,1) = hul(i,npts)
           hul(i,npts+2) = hul(i,3)
        else
           hul(i,1) = hul(i,2)
           hul(i,npts+2) = hul(i,npts+1)
        endif
 20   continue

c      return
      return
      end
c
c-----------------------------------------------------------
c
c     Subroutine: fndhul
c
c     Description: iterates to find the control hull
c
c
c      Krishnan V. Kolady
c      08/15/89
c
c-----------------------------------------------------------
c

      subroutine fndhul(end, npts, pts, hul)


c      declare the variables
      integer end, npts, count
      real pts(3,*), hul(3,*)

      real err, delmax
      parameter(err = 0.0001)

c      initialize delmax
      count = 0

c      initialize delmax
 5    delmax = 0.0

c
      do 20 i=1,3
        do 10 j=2, npts+1
c           find delta error
           call fnddel(pts(i,j-1),hul(i,j-1),hul(i,j),hul(i,j+1),del)

c           find delmax
           if(abs(del).ge.delmax) delmax = abs(del)
```

```
 10     continue
c        initialize the end points
        if(end.eq.1) then
           hul(i,1) = hul(i,npts)
           hul(i,npts+2) = hul(i,3)
        else
           hul(i,1) = hul(i,2)
           hul(i,npts+2) = hul(i,npts+1)
        endif

 20   continue

      count = count + 1
c      check for termination of iteration
      if(delmax.gt.err) goto 5

c      return
      return
      end
c
c-----------------------------------------------------------
c
c     Subroutine: fnddel
c
c     Description: finds delta for the three points given
c
c
c      Krishnan V. Kolady
c      08/15/89
c
c-----------------------------------------------------------
c

      subroutine fnddel(pi, qa, qi, qb, delta)


c      declare the variables
      real pi, qa, qi, qb, delta

c      find delta
      delta = pi - qi + 0.5 * ( pi - 0.5 * (qa + qb))
      qi = qi + delta

c      return
      return
      end
c
c-----------------------------------------------------------
c
```

Appendix E : Program BRINT

```
C-------------------------------------------------------
C
C     SUBROUTINE: BRINT
C
C
C     DESCRIPTION: FIND THE INTERSECTION CURVE OF
C                  TWO B-SPLINE RULED SURFACE.
C
C INPUT:
C     OPKL1,OPKL2  = INTERSECTING PATCHES
C     UWO          = STARTING PARAMETRIC VALUES OF THE
C                    INTERSECTING PATCHES
C
C OUTPUT:
C     IPATCH       = NO. OF SEGMENTS OF THE INTERSECTION CURVE
C     TTUWPT       = INTERSECTION POINTS IN THE ORIGINAL
C                    PARAMETRIC SPACE
C
C
C     C. K. WONG
C     4/12/89
C
C

      SUBROUTINE BRINT (OPKL1,OPKL2,UWO,IPATCH,TTUWPT)

      REAL PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)
     +     ,TXYZ1(3,8),TXYZ2(3,8)
     +     ,OPKL1(0:3,0:3,3),OPKL2(0:3,0:3,3)
     +     ,TTUWG(4,3)
     +     ,TTUWPT(4,4,*),UWO(4),UWO2(4)

C ISUBP = NO. OF SUBDIVIDED PATCHES
      PARAMETER(ISUBP=50)

      REAL SUBPKL1(0:3,0:3,3,ISUBP),SUBPKL2(0:3,0:3,3,ISUBP)
      INTEGER IUW(2),IDIR1(ISUBP),IDIR2(ISUBP)
```

```
      LOGICAL BOX, INTF


C USING BOUNDING BOX FOR PRELIMINARY INTERSECTION CHECK.
      CALL BOBOX (OPKL1,OPKL2,BOX)
      IF(.NOT.BOX) THEN
         RETURN
      ENDIF


C SUBDIVIDE THE ORIGIN SURFACES UNTIL THE RULED SURFACE APPROXIMATION
C IS WITHIN A TOLERANCE VALUE (ASSUME THE SUBDIVISIONS <= ISUBP).

      CALL SUBDIV(1,OPKL1,IDIR1,INOSUB1,SUBPKL1)

      CALL SUBDIV(2,OPKL2,IDIR2,INOSUB2,SUBPKL2)

C PROCESS THE SUBDIVIDED PATCHES: INTERSECT THE SUBPATCHES (1)
C WITH SUBPATCHES (2)

      DO 123 IJK=1,INOSUB1

C DIRECTION OF SUBDIVISION FOR SURFACE 1
         IUW(1)=IDIR1(IJK)

C PROCESS THE SUBPATCHES (1) ONE AT A TIME
         DO 111 I=1,4
              DO 222 J=1,4
                   DO 333 K=1,3
                      PKL1(I-1,J-1,K)=SUBPKL1(I-1,J-1,K,IJK)
 333               CONTINUE
 222          CONTINUE
 111     CONTINUE


C CHECK THE SURFACE TYPE
         CALL CHKSR(PKL1,ITYPE1)

C REGULAR RULED SURFACE: FIND THE SURFACE COEFF. OF
C FIRST INTERSECTING PATCH
         IF(ITYPE1.EQ.0) THEN
           CALL BRFOR(PKL1,1,TXYZ1,IUW)
         ENDIF


C PROCESS SUBPATCHES (2)
         DO 456 IIK=1,INOSUB2

C DIRECTION OF SUBDIVISION FOR SUBPATCH 2
            IUW(2)=IDIR2(IIK)

C PROCESS SUBPATCH (2) ONE AT A TIME
            DO 115 I=1,4
               DO 225 J=1,4
                  DO 335 K=1,3
                     PKL2(I-1,J-1,K)=SUBPKL2(I-1,J-1,K,IIK)
 335              CONTINUE
 225           CONTINUE
 115        CONTINUE


C CHECK THE SUBDIVIDED SURFACES INTERSECTION USING BOUNDING BOX
            CALL BOBOX (PKL1,PKL2,BOX)
            IF(.NOT.BOX) THEN
               GOTO 456
```

```
            ENDIF

C CHECK THE SURFACE TYPE
           CALL CHKSR(PKL2,ITYPE2)

C CHECK WHETHER IT IS SPECIAL CASE OF INTERSECTION
C
C      ITYPE=0    ;NORMAL
C      ITYPE=1    ;PATCH 1 IS SPECIAL RULED SURFACE
C      ITYPE=2    ;PATCH 2 IS SPECIAL RULED SURFACE
C      ITYPE=3    ;BOTH PATCH ARE SPECIAL RULED SURFACES


            IF((ITYPE1.EQ.1).AND.(ITYPE2.EQ.1)) THEN
C ITYPE=3
              ITYPE=3

C INTERSECTION OF TWO SPECIAL RULED SURFACE
              CALL BRINTSS(PKL1,PKL2,INTF,TTUWG)

C INTERSECT?
              IF(.NOT.INTF) THEN
                   GOTO 456
                 ELSE
                   GOTO 1000
              ENDIF

             ELSEIF((ITYPE1.EQ.1).OR.(ITYPE2.EQ.1)) THEN

C FIND OUT WHICH SURFACE IS THE SPECIAL CASE
                   IF(ITYPE1.EQ.1) THEN
                        ITYPE=1
                      ELSE
                        ITYPE=2
                   ENDIF

C INTERSECTION OF ONE SPECIAL AND ONE REGULAR RULED SURFACES
              CALL BRINTSR(PKL1,PKL2,ITYPE,IUW,INTF,TTUWG)

C INTERSECT?
              IF(.NOT.INTF) THEN
                   GOTO 456
                 ELSE
                   GOTO 1000
              ENDIF


            ENDIF

C SURFACE COEFF. FOR SECOND INTERSECTING PATCH
           CALL BRFOR(PKL2,2,TXYZ2,IUW)

C FIND THE TWO STARTING POINTS OF INTERSECTION:
C TWO REGULAR RULED SURFACES

           CALL BRINTRR(TXYZ1,TXYZ2,PKL1,PKL2,IUW,INTF,TTUWG)

C INTERSECT?
              IF(.NOT.INTF) THEN
                   GOTO 456
              ENDIF

C INVERT THE POINTS TO THE ORIGINAL SURFACES AND
C FIND THE CORRESPONDING PARAMETRIC VALUES
 1000    CALL INVERT4(OPKL1,OPKL2,TTUWG,UWO,IPATCH,TTUWPT)
```

```
 456    CONTINUE
 123  CONTINUE


      RETURN
      END

C-------------------------------------------------------------
C
C     SUBROUTINE: INVERT4
C
C     DESCRIPTION: INVERT THE INTERSECTION POINTS
c                  (IN GLOBAL COOR.) TO THE ORIGINAL
C                  SURFACE FOR PARAMETRIC VALUES.
C
C INPUT:
C     OPKL1,OPKL2  = INTERSECTING SURFACES
C     TTUWG        = INTERSECTION POINTS IN GLOBAL COOR.
C     UWO          = STARTING PARAMETRIC VALUES FOR
C                    INTERSECTING SURFACES
C     IPATCH       = NO. OF SEGMENTS OF THE INTERSECTION CURVE
C
C OUTPUT:
C     TTUWPT       = INTERSECTION POINTS IN THE ORIGINAL
C                    PARAMETRIC VALUES
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE INVERT4(OPKL1,OPKL2,TTUWG,UWO,IPATCH,TTUWPT)

      REAL OPKL1(0:3,0:3,3),OPKL2(0:3,0:3,3),TTUWG(4,3)
     +    ,UWO(4),TTUWPT(4,4,*)
     +    ,SRF(3,4,4),P(3)

C INCREMENT THE SEGMENT (INTERSECTION POINTS) NUMBER
      IPATCH=IPATCH+1

C INVERT THE POINTS TO BOTH SURFACES
      DO 10 I=1,2

C CHANGE THE SURFACE FORMAT INTO SUBROUTINE INVERT'S FORMAT
         IF(I.EQ.1) THEN

C SURFACE #1
              DO 20 J=1,3
                   DO 30 K=1,4
                        DO 40 L=1,4
                             SRF(J,L,K)=OPKL1(L-1,K-1,J)
 40                     CONTINUE
 30                CONTINUE
 20           CONTINUE

           ELSE

C SURFACE #2
              DO 25 J=1,3
                   DO 35 K=1,4
                        DO 45 L=1,4
                             SRF(J,L,K)=OPKL2(L-1,K-1,J)
 45                     CONTINUE
 35                CONTINUE
 25           CONTINUE

         ENDIF
```

```
C INVERT ONE POINT AT A TIME (THERE ARE 4 POINTS
C IN A SEGMENT)
         DO 100 J=1,4

              DO 110 K=1,3
                   P(K)=TTUWG(J,K)
 110          CONTINUE

C INVERT THE POINT
              CALL INVERT(SRF,P,U,V,FMAG)

C PUT THE RESULT IN TTUWPT
              IF(I.EQ.1) THEN
                   TTUWPT(J,1,IPATCH)=U+UWO(1)
                   TTUWPT(J,2,IPATCH)=V+UWO(2)
                  ELSE
                   TTUWPT(J,3,IPATCH)=U+UWO(3)
                   TTUWPT(J,4,IPATCH)=V+UWO(4)
              ENDIF

 100     CONTINUE

 10   CONTINUE


      RETURN
      END

C--------------------------------------------------
C
C     SUBROUTINE: BOBOX
C
C     DESCRIPTION: PRELIMINARY INTERSECTION CHECK USING
C                  BOUNDING BOX
C
C INPUT:
C     PKL1,PKL2    = INTERSECTING SURFACES
C
C OUTPUT:
C     BOX          = INTERSECTION FLAG
C
C     C. K. WONG
C     1/12/90
C
C

      SUBROUTINE BOBOX(PKL1,PKL2,BOX)

      REAL PKL1(0:3,0:3,3), PKL2(0:3,0:3,3)
     +     ,BOX1(8,3),BOX2(8,3)
     +     ,TPST1(1,3),TPST2(1,3)
     +     ,MMBOX1(3,2),MMBOX2(3,2)

      LOGICAL BOX

C GET THE CORNER POINTS FROM BOTH SURFACES
C AND PUT THEM INTO BOX1 & BOX2

      IB=1
      CALL FBPT(PKL1,0.,0.,TPST1,IFLAG)
      CALL STOBOX(IB,TPST1,BOX1)
      CALL FBPT(PKL2,0.,0.,TPST2,IFLAG)
      CALL STOBOX(IB,TPST2,BOX2)
      IB=2
      CALL FBPT(PKL1,0.,1.,TPST1,IFLAG)
      CALL STOBOX(IB,TPST1,BOX1)
      CALL FBPT(PKL2,0.,1.,TPST2,IFLAG)
```

```
      CALL STOBOX(IB,TPST2,BOX2)
      IB=3
      CALL FBPT(PKL1,1.,1.,TPST1,IFLAG)
      CALL STOBOX(IB,TPST1,BOX1)
      CALL FBPT(PKL2,1.,1.,TPST2,IFLAG)
      CALL STOBOX(IB,TPST2,BOX2)
      IB=4
      CALL FBPT(PKL1,1.,0.,TPST1,IFLAG)
      CALL STOBOX(IB,TPST1,BOX1)
      CALL FBPT(PKL2,1.,0.,TPST2,IFLAG)
      CALL STOBOX(IB,TPST2,BOX2)

C GET THE INTERIOR 4 POINTS FROM BOTH CONTROL
C HULL AND PUT THEM INTO BOX1 & BOX2

      IB=5
      DO 10 I=1,2
         DO 15 K=1,2
            DO 20 J=1,3
              TPST1(1,J)=PKL1(I,K,J)
              TPST2(1,J)=PKL2(I,K,J)
 20         CONTINUE
            CALL STOBOX(IB,TPST1,BOX1)
            CALL STOBOX(IB,TPST2,BOX2)
            IB=IB+1
 15      CONTINUE
 10   CONTINUE


C FIND THE MAX & MIN OF THE BOXES

      CALL FMMBOX(BOX1,MMBOX1)
      CALL FMMBOX(BOX2,MMBOX2)

C USING THE MAX & MIN BOX FOR INTERSECTION CHECK

      BOX=.TRUE.

      IF(((MMBOX1(1,1).LT.MMBOX2(1,1)).AND.
     +    (MMBOX1(1,2).LT.MMBOX2(1,1))).OR.
     +   ((MMBOX1(1,1).GT.MMBOX2(1,2)).AND.
     +    (MMBOX1(1,2).GT.MMBOX2(1,2)))) THEN
         BOX=.FALSE.
         RETURN
      ENDIF

      IF(((MMBOX1(2,1).LT.MMBOX2(2,1)).AND.
     +    (MMBOX1(2,2).LT.MMBOX2(2,1))).OR.
     +   ((MMBOX1(2,1).GT.MMBOX2(2,2)).AND.
     +    (MMBOX1(2,2).GT.MMBOX2(2,2)))) THEN
         BOX=.FALSE.
         RETURN
      ENDIF

      IF(((MMBOX1(3,1).LT.MMBOX2(3,1)).AND.
     +    (MMBOX1(3,2).LT.MMBOX2(3,1))).OR.
     +   ((MMBOX1(3,1).GT.MMBOX2(3,2)).AND.
     +    (MMBOX1(3,2).GT.MMBOX2(3,2)))) THEN
         BOX=.FALSE.
         RETURN
      ENDIF



      RETURN
      END
```

```
C-----------------------------------------------------------
C
C     SUBROUTINE: STOBOX
C
C     DESCRIPTION: PUT A POINT INTO A BOX
C
C INPUT:
C     IB             = POINT IDENTIFIER
C     TPST           = POINT
C
C OUTPUT:
C     BOX            = MAX/MIN BOX FOR A SURFACE
C
C     C. K. WONG
C     9/12/89
C
C

      SUBROUTINE STOBOX(IB,TPST,BOX)

      REAL TPST(1,3),BOX(8,3)

      DO 10 I=1,3
         BOX(IB,I)=TPST(1,3)
 10   CONTINUE

      RETURN
      END

C-----------------------------------------------------------------
C
C     SUBROUTINE: FMMBOX
C
C     DESCRIPTION: FIND THE MAX & MIN OF A BOUNDING BOX
C
C INPUT:
C     BOX            = BOUNDING BOX FOR A SURFACE
C
C OUTPUT:
C     MMBOX          = MAX/MIN VALUES OF THE BOX
C
C     C. K. WONG
C     1/12/89
C
C

      SUBROUTINE FMMBOX(BOX,MMBOX)

      REAL BOX(8,3),MMBOX(3,2)
     +     ,MAX,MIN

      DO 10 I=1,3
         MAX=BOX(1,I)
         MIN=BOX(1,I)
         DO 20 J=2,8
              IF(BOX(J,I).GT.MAX) THEN
                   MAX=BOX(J,I)
                 ELSEIF(BOX(J,I).LT.MIN) THEN
                   MIN=BOX(J,I)
              ENDIF
 20      CONTINUE

         MMBOX(I,1)=MIN
         MMBOX(I,2)=MAX
 10   CONTINUE

      RETURN
```

```
      END
C---------------------------------------------------
C
C     SUBROUTINE: CHKSR
C
C     DESCRIPTION: CHECK WHETHER IT IS A SPECIAL RULED SURFACE
C
C INPUT:
C     PKL          = A SURFACE
C
C OUTPUT:
C     ITYPE        = FLAG FOR THE SURFACE TYPE
C
C     C. K. WONG
C     9/12/89
C
C

      SUBROUTINE CHKSR(PKL,ITYPE)


      REAL PKL(0:3,0:3,3)

C INITIALIZE VARIABLES
      ITYPE=0
      TOL=.01


C CHECK FOR COLUMNS OF MULTIPLES OF FIRST COLUMN OR
C ROWS OF MULTIPLES OF FIRST ROW

      IT=0
      DO 10 I=1,3
         IS=0
         DO 20 J=1,3
          IF((ABS(PKL(0,0,I)-PKL(0,J,I)).LT.TOL).AND.
     +        (ABS(PKL(1,0,I)-PKL(1,J,I)).LT.TOL).AND.
     +        (ABS(PKL(2,0,I)-PKL(2,J,I)).LT.TOL).AND.
     +        (ABS(PKL(3,0,I)-PKL(3,J,I)).LT.TOL)) THEN
               IS=IS+1

      ELSEIF((ABS(PKL(0,0,I)-PKL(J,0,I)).LT.TOL).AND.
     +        (ABS(PKL(0,1,I)-PKL(J,1,I)).LT.TOL).AND.
     +        (ABS(PKL(0,2,I)-PKL(J,2,I)).LT.TOL).AND.
     +        (ABS(PKL(0,3,I)-PKL(J,3,I)).LT.TOL)) THEN
               IS=IS+1
          ENDIF

 20      CONTINUE

         IF(IS.GE.3) THEN
              IT=IT+1
         ENDIF

 10   CONTINUE

C SPECIAL CASE FOR X, Y & Z: SPECIAL RULED SURFACE

      IF(IT.EQ.3) THEN
         ITYPE=1
      ENDIF



      RETURN
      END
C---------------------------------------------------
```

```
C
C     SUBROUTINE: BRFOR
C
C     DESCRIPTION: CALCULATE THE COEFF. OF A
C                  B-SPLINE RULED SURFACE.
C
C INPUT:
C     PKL          = INTERSECTING PATCH
C     IS           = PATCH IDENTIFIER
C     IUW          = RULED DIRECTION OF THE SURFACE
C
C OUTPUT:
C     TXYZ         = SURFACE COEFF.
C
C     C. K. WONG
C     9/12/89
C
C

      SUBROUTINE BRFOR(PKL,IS,TXYZ,IUW)

      REAL PKL(0:3,0:3,3),P(4,4),C(16),TXYZ(3,8)
      INTEGER IUW(2),ICC(3)

C SET THE VARIABLES

      R36=1./36.
      R12=1./12.
      R4=1./4.
      R2=1./2.
      R6=1./6.
      R3=1./3.
      R9=1./9.

C SURFACE COEFF. FOR X, Y, & Z

      DO 10 I=1,3

         DO 20 J=1,4
              DO 30 K=1,4
                   P(J,K)=PKL(J-1,K-1,I)
 30           CONTINUE
 20      CONTINUE

       C(1)=R36*P(1,1)-R12*P(1,2)+R12*P(1,3)-R36*P(1,4)-R12*P(2,1)
     +      +R4*P(2,2)-R4*P(2,3)+R12*P(2,4)+R12*P(3,1)-R4*P(3,2)
     +      +R4*P(3,3)-R12*P(3,4)-R36*P(4,1)+R12*P(4,2)-R12*P(4,3)
     +      +R36*P(4,4)
       C(2)=-R12*P(1,1)+R6*P(1,2)-R12*P(1,3)+R4*P(2,1)-R2*P(2,2)
     +      +R4*P(2,3)-R4*P(3,1)+R2*P(3,2)-R4*P(3,3)+R12*P(4,1)
     +      -R6*P(4,2)+R12*P(4,3)
       C(3)=R12*P(1,1)-R12*P(1,3)-R4*P(2,1)+R4*P(2,3)+R4*P(3,1)
     +      -R4*P(3,3)-R12*P(4,1)+R12*P(4,3)
       C(4)=-R36*P(1,1)-R9*P(1,2)-R36*P(1,3)+R12*P(2,1)+R3*P(2,2)
     +      +R12*P(2,3)-R12*P(3,1)-R3*P(3,2)-R12*P(3,3)+R36*P(4,1)
     +      +R9*P(4,2)+R36*P(4,3)
       C(5)=-R12*P(1,1)+R4*P(1,2)-R4*P(1,3)+R12*P(1,4)+R6*P(2,1)
     +      -R2*P(2,2)+R2*P(2,3)-R6*P(2,4)-R12*P(3,1)+R4*P(3,2)
     +      -R4*P(3,3)+R12*P(3,4)
       C(6)=R4*P(1,1)-R2*P(1,2)+R4*P(1,3)-R2*P(2,1)+P(2,2)-R2*P(2,3)
     +      +R4*P(3,1)-R2*P(3,2)+R4*P(3,3)
       C(7)=-R4*P(1,1)+R4*P(1,3)+R2*P(2,1)-R2*P(2,3)-R4*P(3,1)+R4*P(3,3)
       C(8)=R12*P(1,1)+R3*P(1,2)+R12*P(1,3)-R6*P(2,1)-(2./3.)*P(2,2)
     +      -R6*P(2,3)+R12*P(3,1)+R3*P(3,2)+R12*P(3,3)
       C(9)=R12*P(1,1)-R4*P(1,2)+R4*P(1,3)-R12*P(1,4)-R12*P(3,1)
     +      +R4*P(3,2)-R4*P(3,3)+R12*P(3,4)
```

```
      C(10)=-R4*P(1,1)+R2*P(1,2)-R4*P(1,3)+R4*P(3,1)-R2*P(3,2)
     +       +R4*P(3,3)
      C(11)=R4*P(1,1)-R4*P(1,3)-R4*P(3,1)+R4*P(3,3)
      C(12)=-R12*P(1,1)-R3*P(1,2)-R12*P(1,3)+R12*P(3,1)+R3*P(3,2)
     +       +R12*P(3,3)
      C(13)=-R36*P(1,1)+R12*P(1,2)-R12*P(1,3)+R36*P(1,4)-R9*P(2,1)
     +       +R3*P(2,2)-R3*P(2,3)+R9*P(2,4)-R36*P(3,1)+R12*P(3,2)
     +       -R12*P(3,3)+R36*P(3,4)
      C(14)=R12*P(1,1)-R6*P(1,2)+R12*P(1,3)+R3*P(2,1)-(2./3.)*P(2,2)
     +       +R3*P(2,3)+R12*P(3,1)-R6*P(3,2)+R12*P(3,3)
      C(15)=-R12*P(1,1)+R12*P(1,3)-R3*P(2,1)+R3*P(2,3)-R12*P(3,1)
     +       +R12*P(3,3)
      C(16)=R36*P(1,1)+R9*P(1,2)+R36*P(1,3)+R9*P(2,1)+(4./9.)*P(2,2)
     +       +R9*P(2,3)+R36*P(3,1)+R9*P(3,2)+R36*P(3,3)

C RULED SURFACE COEFF. (CUBIC IN IUW(IS) DIRECTION)

        IF(IUW(IS).EQ.0) THEN
             TXYZ(I,1)=C(3)
             TXYZ(I,2)=C(7)
             TXYZ(I,3)=C(11)
             TXYZ(I,4)=C(15)
             TXYZ(I,5)=C(4)
             TXYZ(I,6)=C(8)
             TXYZ(I,7)=C(12)
             TXYZ(I,8)=C(16)
         ELSEIF(IUW(IS).EQ.1) THEN
             TXYZ(I,1)=C(9)
             TXYZ(I,2)=C(10)
             TXYZ(I,3)=C(11)
             TXYZ(I,4)=C(12)
             TXYZ(I,5)=C(13)
             TXYZ(I,6)=C(14)
             TXYZ(I,7)=C(15)
             TXYZ(I,8)=C(16)
        ENDIF


 10   CONTINUE


      RETURN
      END
c-----------------------------------------------------------
c
c     Subroutine: invert
c
c     Description: Finds the closest point on the surface
c             to the given point
c
c
c      Krishnan V. Kolady
c      09/27/89
c
c-----------------------------------------------------------
c*/

      subroutine invert(srf, p, u, v, fmag)

c     /* declare the variables
      real srf(3,4,4), p(3), u, v, fmag

      real dumax, dvmax, maxitr, tol, itr
      parameter(dumax=0.1, dvmax=0.1, maxitr=5, tol=1e-10)

      real f(3), pu(3), pv(3)
      real ni(3), delui, delvi
```

```
      real calui, calvi

c     /* set initial values for u, v, and itr
      u = 0.5
      v = 0.5
      itr = 1

c     /* calculate the required function f and pu, pv
 10   call cldrvi(srf, u, v, p, f, pu, pv, fmag)

c     /* calculate ni
      call cross(pu, pv, ni)

c     /* calculate del u
      delui = calui(ni, f, pv)
      if( abs(delui) .gt. dumax ) then
        if( delui .lt. 0 ) then
          delui = -dumax
        else
          delui = dumax
        endif
      endif

c     /* calculate the new value of u
      u = u + delui

c     /* calculate del v
      delvi = calvi(ni, f, pu)
      if( abs(delvi) .gt. dvmax ) then
        if( delvi .lt. 0 ) then
          delvi = -dvmax
        else
          delvi = dvmax
        endif
      endif

c     /* calculate the new value of v
      v = v - delvi

c     /* check for termination
      if( itr .ge. maxitr .or.
     -   (abs(delui) .le. tol .and. abs(delvi) .le. tol )) then
          goto 500
      else
c         /* increment itr
          itr = itr + 1
          goto 10
      endif

c     /*
 500  if(u .lt. 0.0) u=0.0
      if(u .gt. 1.0) u=1.0
      if(v .lt. 0.0) v=0.0
      if(v .gt. 1.0) v=1.0

c     /* return
 5000 return
      end
c/*
c-----------------------------------------------------------
c
c     Subroutine: cldrvi
c
c     Description: Calculates the required derivatives for
c             subroutine invert
c
c
```

```
c       Krishnan V. Kolady
c       09/27/89
c
c--------------------------------------------------------
c*/

      subroutine cldrvi(srf, u, v, p, f, pu, pv, magf)

c     /* declare the variables
      real srf(3,4,4), u, v, p(3), f(3), pu(3), pv(3), magf

      real magn, psrf(3)

c     /* find the point on the surface
      call bpsrf(srf, u, v, psrf(1), psrf(2), psrf(3))

c     /* find function f
      call sub(psrf, p, f)

c     /* find pu
      call bpsrfu(srf, u, v, pu(1), pu(2), pu(3))

c     /* find pv
      call bpsrfw(srf, u, v, pv(1), pv(2), pv(3))

c     /* find the magnitude of the function f
      magf = magn(f)

c     /* return
 5000 return
      end
c/*
c--------------------------------------------------------
c
c     Function: calui
c
c     Description: Calculate the del u for the invert
c             subroutine
c
c
c       Krishnan V. Kolady
c       09/27/89
c
c--------------------------------------------------------
c*/

      real function calui(ni, f, pv)

c     /* declare the variables
      real ni(3), f(3), pv(3)

      real dotni, crpvf(3), nipvf

c     /* find dotni
      call dot(ni, ni, dotni)

c     /* find the cross product of pv & f
      call cross(pv, f, crpvf)

c     /* find the dot product ni and crpvf
      call dot(ni, crpvf, nipvf)

c     /* find the value for del u
      calui = nipvf / dotni

c     /* return
 5000 return
```

```
      end
c/*
c-----------------------------------------------------------
c
c     Function: calvi
c
c     Description: Calculate the del v for the invert
c             subroutine
c
c
c      Krishnan V. Kolady
c      09/27/89
c
c-----------------------------------------------------------
c*/

      real function calvi(ni, f, pu)

c     /* declare the variables
      real ni(3), f(3), pu(3)

      real dotni, crpuf(3), nipuf

c     /* find dotni
      call dot(ni, ni, dotni)

c     /* find the cross product of pu & f
      call cross(pu, f, crpuf)

c     /* find the dot product ni and crpuf
      call dot(ni, crpuf, nipuf)

c     /* find the value for del v
      calvi = nipuf / dotni

c     /* return
 5000 return
      end
c/*
c-----------------------------------------------------------
c
c     Subroutine: dot
c
c     Description: finds the dot product of 2 vectors
c             This is available in acsmath
c
c      Krishnan V. Kolady
c      09/14/89
c
c-----------------------------------------------------------
c*/

      subroutine dot(a,b,c)

c     /* declare the variables
      real a(3), b(3), c

c     /* calculate the dot product
      c = a(1)*b(1) + a(2)*b(2) + a(3)*b(3)

c     /* return
 5000 return
      end
c/*
c-----------------------------------------------------------
c
c     Subroutine: cross
```

```
c
c     Description: finds the cross product of 2 vectors
c             This is available in acsmath
c
c      Krishnan V. Kolady
c      09/14/89
c
c-----------------------------------------------------------
c*/

      subroutine cross(a,b,c)

c     /* declare the variables
      real a(3), b(3), c(3)

c     /* calculate the cross product
      c(1) = a(2)*b(3) - a(3)*b(2)
      c(2) = a(3)*b(1) - a(1)*b(3)
      c(3) = a(1)*b(2) - a(2)*b(1)

c     /* return
 5000 return
      end
c/*
c-----------------------------------------------------------
c
c     Subroutine: sub
c
c     Description: finds the subtraction of 2 vectors
c             This is available in acsmath
c
c      Krishnan V. Kolady
c      09/14/89
c
c-----------------------------------------------------------
c*/

      subroutine sub(a,b,c)

c     /* declare the variables
      real a(3), b(3), c(3)

c     /* calculate the cross product
      c(1) = a(1) - b(1)
      c(2) = a(2) - b(2)
      c(3) = a(3) - b(3)

c     /* return
 5000 return
      end
c/*
c-----------------------------------------------------------
c
c     Subroutine: cpyvct
c
c     Description: copies a vector into another vector
c             This is available in utilty
c
c      Krishnan V. Kolady
c      09/14/89
c
c-----------------------------------------------------------
c*/

      subroutine cpyvct(nitems, vect1, vect2)

c     /* declare the variables
```

```
      integer nitems
      real vect1(*), vect2(*)

c     /*
      do 100 i=1,nitems
         vect2(i) = vect1(i)
 100  continue

c     /* return
 5000 return
      end
c/*
c-----------------------------------------------------
c
c     Subroutine: bpsrf
c
c     Description: uses the geometric form to compute point
c             on a B-spline patch for specific u and w
c
c      Krishnan V. Kolady
c      08/14/89
c
c-----------------------------------------------------
c*/

      subroutine bpsrf(b, u, w, px, py, pz)


c     /* declare the variables
      real b(3,4,4), u, w, px, py, pz

      real fu(4), fw(4)

c     /* calculate blending functions for u
      call bbf(u, fu)

c     /* calculate blending functions for w
      call bbf(w, fw)

c     /* calculate x coord. on patch
      call bclcpt(1, b, fu, fw, px)

c     /* calculate y coord. on patch
      call bclcpt(2, b, fu, fw, py)

c     /* calculate z coord. on patch
      call bclcpt(3, b, fu, fw, pz)

c     /* return
      return
      end
c/*
c-----------------------------------------------------
c
c     Subroutine: bbf
c
c     Description: calculates the four components of the
c             blending functions at specific u
c
c      Krishnan V. Kolady
c      08/14/89
c
c-----------------------------------------------------
c*/

      subroutine bbf(u, f)
```

```
c     /* declare the variables
      real u, f(4)

      real umat(4), mmat(4,4)
      data mmat/ -0.166667, 0.5, -0.5, 0.166667,
           0.5, -1.0, 0.0, 0.666667,
           -0.5, 0.5, 0.5, 0.166667,
           0.166667, 0.0, 0.0, 0.0/

c     /* define the u matrix
      umat(4) = 1.0
      umat(3) = u * umat(4)
      umat(2) = u * umat(3)
      umat(1) = u * umat(2)

c     /* multiply the u matrix with M matrix
      call bmatm(1, 4, 4, umat, mmat, f)

c     /* return
      return
      end
c/*
c----------------------------------------------------------
c
c     Subroutine: bpsrfu
c                                                      u
c     Description: uses the geometric form to compute P(u,w)
c             on a B-spline patch for specific u and w
c
c      Krishnan V. Kolady
c      08/14/89
c
c----------------------------------------------------------
c*/

      subroutine bpsrfu(b, u, w, px, py, pz)


c     /* declare the variables
      real b(3,4,4), u, w, px, py, pz

      real fu(4), fw(4)

c     /* calculate blending functions for u
      call bbfu(u, fu)

c     /* calculate blending functions for w
      call bbf(w, fw)

c     /* calculate x coord. on patch
      call bclcpt(1, b, fu, fw, px)

c     /* calculate y coord. on patch
      call bclcpt(2, b, fu, fw, py)

c     /* calculate z coord. on patch
      call bclcpt(3, b, fu, fw, pz)

c     /* return
      return
      end
c/*
c----------------------------------------------------------
c
c     Subroutine: bpsrfw
c                                                      w
```

```
c     Description: uses the geometric form to compute P(u,w)
c             on a B-spline patch for specific u and w
c
c      Krishnan V. Kolady
c      08/14/89
c
c-----------------------------------------------------------
c*/

      subroutine bpsrfw(b, u, w, px, py, pz)


c     /* declare the variables
      real b(3,4,4), u, w, px, py, pz

      real fu(4), fw(4)

c     /* calculate blending functions for u
      call bbf(u, fu)

c     /* calculate blending functions for w
      call bbfu(w, fw)

c     /* calculate x coord. on patch
      call bclcpt(1, b, fu, fw, px)

c     /* calculate y coord. on patch
      call bclcpt(2, b, fu, fw, py)

c     /* calculate z coord. on patch
      call bclcpt(3, b, fu, fw, pz)

c     /* return
      return
      end
c/*
c-----------------------------------------------------------
c
c     Subroutine: bbfu
c
c     Description: calculates the four components of the
c             blending functions at specific u
c
c      Krishnan V. Kolady
c      08/14/89
c
c-----------------------------------------------------------
c*/

      subroutine bbfu(u, f)


c     /* declare the variables
      real u, f(4)

      real umat(4), mmat(4,4)
      data mmat/ -0.166667, 0.5, -0.5, 0.166667,
           0.5, -1.0, 0.0, 0.666667,
           -0.5, 0.5, 0.5, 0.166667,
           0.166667, 0.0, 0.0, 0.0/

c     /* define the u matrix
      umat(4) = 0.0
      umat(3) = 1.0
      umat(2) = 2.0 * u
      umat(1) = 3.0 * u * u
```

```
c     /* multiply the u matrix with M matrix
      call bmatm(1, 4, 4, umat, mmat, f)

c     /* return
      return
      end
c/*
c-------------------------------------------------------
c
c     Subroutine: bclcpt
c
c     Description: calculates a specified coordinate on a
c             surface
c
c      Krishnan V. Kolady
c      08/14/89
c
c-------------------------------------------------------
c*/

      subroutine bclcpt(xyz, b, fu, fw, p)


c     /* declare the variables
      integer xyz
      real b(3,4,4), fu(4), fw(4), p

      integer row

c     /* initialize coordinate
      p = 0.0

c     /*
      do 20 row = 1,4
c       /* perform matrix mult
        call bsfmlt(xyz, row, b, fu, fw, p)
 20   continue

c     /* return
      return
      end
c/*
c-------------------------------------------------------
c
c     Subroutine: bsfmlt
c
c     Description: performs matrix multiplication for
c             calculating a point on a surface patch
c
c      Krishnan V. Kolady
c      08/14/89
c
c-------------------------------------------------------
c*/

      subroutine bsfmlt(xyz, row, b, fu, fw, p)


c     /* declare the variables
      integer xyz, row
      real b(3,4,4), fu(4), fw(4), p

      integer lcv
      real d

c     /* initialize d
      d = 0.0
```

```
c     /*
      do 10 lcv =1,4
c       /* perform matrix mult
        d= d + b(xyz, row, lcv) * fu(lcv)
 10   continue

c     /* calculate coord.
      p = p + d * fw(row)

c     /* return
      return
      end
c/*
c-----------------------------------------------------
c
c     Subroutine: bpcrv
c
c     Description: uses the geometric form to compute point
c             on a B-spline curve for specific u
c
c      Krishnan V. Kolady
c      08/14/89
c
c-----------------------------------------------------
c*/

      subroutine bpcrv(b, u, px, py, pz)


c     /* declare the variables
      real b(3,4), u, w, px, py, pz

      real fu(4)

c     /* calculate blending functions for u
      call bbf(u, fu)

c     /* calculate x coord. on patch
      call bcrdpt(1, b, fu, px)

c     /* calculate y coord. on patch
      call bcrdpt(2, b, fu, py)

c     /* calculate z coord. on patch
      call bcrdpt(3, b, fu, pz)

c     /* return
      return
      end
c/*
c-----------------------------------------------------
c
c     Subroutine: bpcrvu
c                                                    u
c     Description: uses the geometric form to compute Q(u)
c             on a B-spline curve for specific u
c
c      Krishnan V. Kolady
c      08/14/89
c
c-----------------------------------------------------
c*/

      subroutine bpcrvu(b, u, px, py, pz)
```

```
c     /* declare the variables
      real b(3,4), u, w, px, py, pz

      real fu(4)

c     /* calculate blending functions for u
      call bbfu(u, fu)

c     /* calculate x coord. on patch
      call bcrdpt(1, b, fu, px)

c     /* calculate y coord. on patch
      call bcrdpt(2, b, fu, py)

c     /* calculate z coord. on patch
      call bcrdpt(3, b, fu, pz)

c     /* return
      return
      end
c/*
c-------------------------------------------------------
c
c     Subroutine: bcrdpt
c
c     Description: calculates a specified coordinate on a
c             surface
c
c      Krishnan V. Kolady
c      08/14/89
c
c-------------------------------------------------------
c*/

      subroutine bcrdpt(xyz, b, fu, p)


c     /* declare the variables
      integer xyz
      real b(3,4), fu(4), fw(4), p

      integer row

c     /* initialize coordinate
      p = 0.0

c     /*
      do 20 row = 1,4
c       /* perform matrix mult
        p = p + b(xyz, row) * fu(row)
 20   continue

c     /* return
      return
      end
c/*
c-------------------------------------------------------
c
c     Subroutine: bmatm
c
c     Description: general matrix multiplication routine
c
c
c      Krishnan V. Kolady
c      08/15/89
c
```

```
c-----------------------------------------------------------
c*/

      subroutine bmatm( row1, col1row2, col2, matain, matbin,
        matout)

c     /* declare the variables
      integer row1, col1row2, col2
      real matain(row1, col1row2), matbin(col1row2, col2)
      real matout(row1, col2)

      integer i, j, k

c     /*
      do 5 i=1,row1
        do 4 j=1,col2
c         /* initialize output matrix matout element (i,j)
          matout(i,j) = 0.0

c         /* find the (i,j) th element
          do 3 k=1,col1row2
             matout(i,j) = matout(i,j) + matain(i,k) * matbin(k,j)
 3        continue

 4      continue
 5    continue

c     /* return
      return
      end
c/*
c-----------------------------------------------------------
c
c     Function: magn
c
c     Description: finds the magnitude of a vector
c
c
c      Krishnan V. Kolady
c      09/14/89
c
c-----------------------------------------------------------
c*/

      real function magn(a)

c     /* declare the variables
      real a(3)

c     /* calculate the magnitude of the vector
      magn = sqrt(a(1)**2 + a(2)**2 + a(3)**2)

c     /* return
      return
      end
```

Appendix F : Program SUBDIV

```
C--------------------------------------------------------------
C
C      SUBROUTINE: SUBDIV
C
C      DESCRIPTION: APPROXIMATE A CUBIC B-SPLINE SURFACE
C                   WITH RULED SURFACES
C
C INPUT:
C      ISURNO       = SURFACE IDENTIFIER
C      PKL          = SURFACE
C
C OUTPUT:
C      IDIR         = DIRECTION OF SUBDIVISION
C      INOSUB       = NO. OF SUBDIVIDED PATCHES
C      SUBPKL       = SUBDIVIDED PATCHES
C
C      C. K. WONG
C      2/20/90
C

      SUBROUTINE SUBDIV(ISURNO,PKL,IDIR,INOSUB,SUBPKL)

      REAL PKL(0:3,0:3,3),SO(0:3,0:3,3),SPKL(0:3,0:3,3)
     +    ,RPKL1(0:3,0:3,3),RPKL2(0:3,0:3,3),RPKL(0:3,0:3,3)
     +    ,U(4),W(4)
     +    ,MAX1,MAX2,MAX
     +    ,SUBPKL(0:3,0:3,3,*),LIM

      INTEGER RD,RD2,INOSUB,IDIR(*)


C TOLERANCE OF APPROXIMATION
      TOL=.001
      TOL2=.0001
      LIM=.2

C INITIALIZE THE VARIABLES
```

```
      R=-(1./3.)
      DO 10 I=1,4
         R=R+(1./3.)
         U(I)=R
         W(I)=R
 10   CONTINUE

      U1=0.
      U2=1.
      W1=0.
      W2=1.
      INOSUB=0

C PICK ONE PARAMETRIC DIRECTION ON THE ORIGINAL SURFACE
C FOR RULED SURFACE APPROXIMATION
C     RD: FLAG FOR PARAMETRIC DIRECTION

      RD=0

C FIND 16 POINTS OF A RULED SURFACE USING THE BOUNDARIES OF
C ORIGINAL SURFACE

      CALL RULEDUW(PKL,U1,U2,W1,W2,RD,SO)

C INVERSE FOR A RULED SURFACE PROVIDED 16 POINTS ON
C THE SURFACE

      CALL FBPT2(SO,RPKL1,U,W,IFLAG)

C EVALUATE THE APPROXIMATION

      CALL COMLINE(PKL,RPKL1,RD,MAX1)

C PICK ANOTHER PARAMETRIC DIRECTION AND REPEAT THE
C ABOVE PROCEDURES

      RD=1

      CALL RULEDUW(PKL,U1,U2,W1,W2,RD,SO)
      CALL FBPT2(SO,RPKL2,U,W,IFLAG)
      CALL COMLINE(PKL,RPKL2,RD,MAX2)

C************ FIX FOR ACSYNT MODEL *************
C
C FOR ACSYNT MODEL ONLY, FIX THE RULED DIRECTION
C OF WINGS AND TAIL. HAVE TO BE UNFIXED FOR GENERAL
C INTERSECTION
C

      IF((ABS(MAX1).LT.TOL).AND.
     +   (ABS(MAX2).LT.TOL).AND.
     +   (ISURNO.EQ.1)) THEN
         MAX2=0.
      ENDIF


C CHECK THE ACCURACY OF APPROXIMATION

      IF((ABS(MAX1).LE.TOL).AND.
     +   (ABS(MAX1).LE.ABS(MAX2))) THEN

C DIRECTION RD=0 SATISFY THE TOLERANCE
C     INOSUB: NO. OF APPROXIMATED RULED SURFACES

         INOSUB=1
         RD=0
```

```
         IDIR(INOSUB)=RD

C PUT THE RULED SURFACE DATA INTO SUBPKL

         CALL WRPKL(INOSUB,RPKL1,SUBPKL)
         RETURN

       ELSEIF((ABS(MAX2).LE.TOL).AND.
     +   (ABS(MAX2).LE.ABS(MAX1))) THEN

C DIRECTION RD=1 SATISFLY THE TOLERANCE

         INOSUB=1
         RD=1
         IDIR(INOSUB)=RD
         CALL WRPKL(INOSUB,RPKL2,SUBPKL)
         RETURN
      ENDIF

C PICK ONE DIRECTION TO SUBDIVIDE BASED ON THE
C DIFFERENCE OF APPROXIMATIONS

      IF(ABS(MAX1).GT.ABS(MAX2)) THEN
         RD=1
         U2=U2/2.
        ELSE
         RD=0
         W2=W2/2.
      ENDIF


C SUBDIVIDE THE SURFACE INTO HALF FOR EACH ITERATION

C FIND 16 POINTS OF A RULED SURFACE USING THE BOUNDARIES OF
C ORIGINAL SURFACE WHERE PARAMETRIC DIRECTION RD IS DIVIDED
C INTO HALF

 25   CALL RULEDUW(PKL,U1,U2,W1,W2,RD,SO)
      CALL FBPT2(SO,RPKL,U,W,IFLAG)

C FIND THE CORRESPONDING SUBSURFACE FROM THE ORIGINAL SURFACE
C FOR COMPARISON
      CALL SUBSUR(PKL,U1,U2,W1,W2,RD,SO)
      CALL FBPT2(SO,SPKL,U,W,IFLAG)

      CALL COMLINE(SPKL,RPKL,RD,MAX)

C APPROXIMATION IS OUT OF TOLERANCE:
C FURTHER SUBDIVIDE THE SURFACE AND
C REPEAT THE INVERSION AND COMPARISON PROCEDURES

      IF(ABS(MAX).GT.TOL) THEN

         IF(RD.EQ.0) THEN
              W2=(W1+W2)/2.
            ELSE
              U2=(U1+U2)/2.
         ENDIF

C STOP SUBDIVIDING : THE DISTANCE BETWEEN THE BOUNDARY CURVES
C APPROACH THE LIMIT

         IF((RD.EQ.0).AND.((W2-W1).LE.LIM)) THEN
            CALL RULEDUW(PKL,U1,U2,W1,W2,RD,SO)
            CALL FBPT2(SO,RPKL,U,W,IFLAG)
            GOTO 130
          ELSEIF((RD.EQ.1).AND.((U2-U1).LE.LIM)) THEN
```

```
            CALL RULEDUW(PKL,U1,U2,W1,W2,RD,SO)
            CALL FBPT2(SO,RPKL,U,W,IFLAG)
            GOTO 130
         ENDIF

         GOTO 25
      ENDIF

C APPROXIMATION IS WITHIN TOLERANCE:
C INCREMENT THE APPROXIMATED RULED SURFACE NO.

 130  INOSUB=INOSUB+1
      IDIR(INOSUB)=RD

C PUT THE RULED SURFACE DATA INTO SUBPKL
      CALL WRPKL(INOSUB,RPKL,SUBPKL)


C CHECK FOR COMPLETION OF SUBDIVIDING
      IF(RD.EQ.0) THEN
         IF(ABS(W2-1.).LT.TOL2) RETURN
              W1=W2
              W2=1.
          GOTO 25
       ELSE
         IF(ABS(U2-1.).LT.TOL2) RETURN
              U1=U2
              U2=1.
          GOTO 25
      ENDIF

      RETURN
      END

C--------------------------------------------------
C
C     SUBROUTINE: RULEDUW
C
C     DESCRIPTION: FIND 16 POINTS ON A BICUBIC SURFACE TO FORM
C                  A RULED SURFACE
C
C INPUT:
C     PKL          = BICUBIC SURFACE
C     U1,U2,W1,W2  = STARTING AND ENDING PARAMETRIC VALUES
C                    FOR THE POINTS ON THE RULED SURFACE
C     RD           = RULED DIRECTION
C
C OUTPUT:
C     SO           = 16 POINTS ON THE APPROXIMATED RULED SURFACE
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE RULEDUW(PKL,U1,U2,W1,W2,RD,SO)

      REAL PKL(0:3,0:3,3),SO(0:3,0:3,3),
     +     PST1B(1,3),PST1A(1,3)

      INTEGER RD

C FIND 16 POINTS ON THE SURFACE: WITH A INTERVAL OF 1/3
C OF THE CORNER POINTS

      EINTU=(U2-U1)/3.
      EINTW=(W2-W1)/3.
```

```
      IF(RD.EQ.1) THEN

C RD=1: CUBIC IN W DIRECTION; LINEAR IN U DIRECTION

           WW=W1-EINTW
           DO 20 I5=1,4
              WW=WW+(EINTW)

C FIND THE POINTS ON THE W-BOUNDARY

                      CALL FBPT(PKL,U1,WW,PST1A,ISURF)
                      CALL FBPT(PKL,U2,WW,PST1B,ISURF)

C PUT THE POINTS INTO SO; WHERE THE INTERIOR POINTS
C OF U DIRECTION ARE LINEARLY INTERPOLATED BETWEEN
C THE TWO BOUNDARY POINTS

                      DO 65 II1=1,3
                           SO(0,I5-1,II1)=PST1A(1,II1)
                           SO(3,I5-1,II1)=PST1B(1,II1)
                           SO(1,I5-1,II1)=PST1A(1,II1)*(1.-(1./3.))+
     +                             PST1B(1,II1)*(1./3.)
                           SO(2,I5-1,II1)=PST1A(1,II1)*(1.-(2./3.))+
     +                             PST1B(1,II1)*(2./3.)
 65                   CONTINUE

 20        CONTINUE

       ELSE

C RD=0: CUBIC IN U DIRECTION; LINEAR IN W DIRECTION

           UU=U1-EINTU
           DO 200 I5=1,4
              UU=UU+(EINTU)

C FIND THE POINTS ON THE U-BOUNDARY

                      CALL FBPT(PKL,UU,W1,PST1A,ISURF)
                      CALL FBPT(PKL,UU,W2,PST1B,ISURF)

C PUT THE POINTS INTO SO; WHERE THE INTERIOR POINTS
C OF W DIRECTION ARE LINEARLY INTERPOLATED BETWEEN
C THE TWO BOUNDARY POINTS

                      DO 650 II1=1,3
                           SO(I5-1,0,II1)=PST1A(1,II1)
                           SO(I5-1,3,II1)=PST1B(1,II1)
                           SO(I5-1,1,II1)=PST1A(1,II1)*(1.-(1./3.))+
     +                             PST1B(1,II1)*(1./3.)
                           SO(I5-1,2,II1)=PST1A(1,II1)*(1.-(2./3.))+
     +                             PST1B(1,II1)*(2./3.)
 650                  CONTINUE

 200       CONTINUE

      ENDIF

      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: FBPT2
C
C     DESCRIPTION:  INVERT FOR THE CONTROL HULL OF A
C                   SUB-SURFACE PROVIDED THE PARAMETRIC VALUES
C                   (U & W) OF 16 POINTS (SUB-SURFACE) ON THE
```

```
C                   ORIGINAL SURFACE
C
C INPUT:
C     PKLI          = 16 POINTS ON THE SURFACE
C     U,W           = PARAMETRIC VALUES OF THE 16 POINTS
C
C OUTPUT:
C     PKLO          = SURFACE THAT INTERPOLATED THE 16
C                     ON THE SURFACE
C     IFBPTF        = DUMMY VARIABLE
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE FBPT2(PKLI,PKLO,U,W,IFBPTF)

      INTEGER  IFBPTF,IMULF
      REAL U(4),W(4), U4(4,4), W4(4,4), MLT(4,4), MLTW4(4,4),
     +     MK(4,4), U4MK(4,4), TPKL(4,4), PKLI(0:3,0:3,3),
     +     PKLO(0:3,0:3,3), IMLT(4,4), IMK(4,4),IU4(4,4),IW4(4,4),
     +     PMW4(4,4), TPKL2(4,4)


C B-SPLINE BLENDING MATRIX

      DATA MK/-0.166667,0.5,-0.5,0.166667,
     +        0.5,-1.0,0.0,0.666667,
     +        -0.5,0.5,0.5,0.166667,
     +        0.166667,0.0,0.0,0.0/
      DATA MLT/-0.166667,0.5,-0.5,0.166667,
     +         0.5,-1.0,0.5,0.0,
     +         -0.5,0.0,0.5,0.0,
     +         0.166667,0.666667,0.166667,0.0/


C INITIALIZE TOLERANCE

      TOL=.1E-10

      DO 11 I=1,4

C CALCULATE U MATRIX

         IF(ABS(U(I)-0.).LT.TOL) THEN
              U4(I,1)=0.
              U4(I,2)=0.
              U4(I,3)=0.
              U4(I,4)=1.
          ELSE
              U4(I,1)=U(I)**3
              U4(I,2)=U(I)**2
              U4(I,3)=U(I)
              U4(I,4)=1.
         ENDIF

C CALCULATE W MATRIX

         IF(ABS(W(I)-0.).LT.TOL) THEN
              W4(1,I)=0.
              W4(2,I)=0.
              W4(3,I)=0.
              W4(4,I)=1.
           ELSE
              W4(1,I)=W(I)**3
              W4(2,I)=W(I)**2
              W4(3,I)=W(I)
```

```
             W4(4,I)=1.
          ENDIF
 11    CONTINUE

C INVERT B-SPLINE BLENDING MATRIX

          CALL MATINV2(MK,IMK)
          CALL MATINV2(MLT,IMLT)

C INVERT U & W MATRIX

          CALL MATINV2(U4,IU4)
          CALL MATINV2(W4,IW4)

C MULTIPLY W MATRIX TO BLENDING MATRIX

         CALL MATMUL(4,4,IW4,4,4,IMLT,MLTW4,IMULF)

C MULTIPLY U MATRIX TO BLENDING MATRIX

         CALL MATMUL(4,4,IMK,4,4,IU4,U4MK,IMULF)

C CALCULATE THE SUB-SURFACE CONTROL HULL ( X, Y & Z)

       DO 20 I=1,3

C PUT THE SURFACE POINTS INTO TPKL (MATRIX FORM)

          DO 30 J=1,4
               DO 40 II=1,4
                    TPKL(J,II)= PKLI(J-1,II-1,I)
 40            CONTINUE
 30       CONTINUE

C MULTIPLY THE SURFACE POINTS MATRIX TO W BLENDING MATRIX

          CALL MATMUL(4,4,TPKL,4,4,MLTW4,PMW4,IMULF)

C MULTIPLY THE RESULTING MATRIX TO U BLENDING MATRIX

          CALL MATMUL(4,4,U4MK,4,4,PMW4,TPKL2,IMULF)

C SUB-SURFACE CONTROL HULL

          DO 5 I2=1,4
             DO 15 J2=1,4
                 PKLO(I2-1,J2-1,I)=TPKL2(I2,J2)
 15          CONTINUE
 5        CONTINUE

 20    CONTINUE


       RETURN
       END
************************************************************************
************************************************************************
**     SUBROUTINE MATINV(A,AINV)                                      **
**                                                                    **
**     PROGRAM DESCRIPTION                                            **
**                                                                    **
**     THIS ROUTINE CALCULATES THE INVERSE OF A 4X4 MATRIX            **
**                                                                    **
**                                                                    **
**                                                                    **
**     BY:       ASHIT R. GANDHI                                      **
```

```
**    DATE:     11/12/88                                                **
**                                                                      **
**    PARAMETERS USED:                                                  **
**                                                                      **
**    A      = MATRIX WHOSE INVERSE IS TO BE COMPUTED (REAL,I/P)        **
**    AINV   = INVERSE OF THE MATRIX (REAL,O/P)                         **
**                                                                      **
**************************************************************************
**************************************************************************
      SUBROUTINE MATINV2(A,AINV)

      REAL*4 A(4,4), AINV(4,4), B(3,3)

      VALU = 0.0

*COMPUTE THE ADJOINED MATRIX AND COMPUTE THE DETERMINANT VALUE

      DO 200 I = 1,4
         DO 100 J = 1,4
            CALL COFAC(I,J,A,B)
            CALL DET33(B,VAL)
            AINV(J,I) = ((-1.0)**(I+J))*VAL
            IF (I .EQ. 1)THEN
               VALU = ((-1)**(1+J))*A(1,J)*VAL + VALU
            ENDIF
100      CONTINUE
200   CONTINUE

*IF MATRIX IS SINGULAR NO INVERSE EXISTS

      IF (VALU .EQ. 0.0)THEN
         WRITE(6,*)'   MATINV ===>    MATRIX IS SINGULAR.'
         WRITE(6,*)'   MATINV ===>    NO INVERSE EXISTS.'
         WRITE(6,*)'   MATINV ===>    PROGRAM TERMINATED.'
         STOP
      ENDIF

*COMPUTE THE INVERSE ELEMENTS OF THE MATRIX

      DO 400 I = 1,4
         DO 300 J = 1,4
            AINV(I,J) = AINV(I,J)/VALU
300      CONTINUE
400   CONTINUE

      RETURN
      END
**************************************************************************
**************************************************************************
**    SUBROUTINE COFAC(M,N,A,B)                                         **
**                                                                      **
**    PROGRAM DESCRIPTION                                               **
**                                                                      **
**    THIS ROUTINE WILL COMPUTE THE COFACTORS FOR A MATRIX              **
**                                                                      **
**                                                                      **
**                                                                      **
**    BY:       ASHIT R. GANDHI                                         **
**    DATE:     11/12/88                                                **
**                                                                      **
**    PARAMETERS USED:                                                  **
**                                                                      **
**    M = ROW NUMBER FOR WHICH THE COFACTOR IS TO BE CALCULATED         **
**        (INTEGER,I/P)                                                 **
**    N = COLUMN NUMBER FOR WHICH THE COFACTOR IS TO BE CALCULATED      **
**        (INTEGER,I/P)                                                 **
**    A = PARENT MATRIX (REAL,I/P)                                      **
```

```
**    B = COFACTOR MATRIX (REAL,O/P)                                    **
**                                                                      **
**************************************************************************
**************************************************************************
      SUBROUTINE COFAC(M,N,A,B)

      REAL A(4,4), B(3,3)

      L = 1
      DO 200 I = 1,3
         K = 1
         IF (I .EQ. M)L = L + 1
         DO 100 J = 1,3
            IF (J .EQ. N)K = K + 1
            B(I,J) = A(L,K)
            K = K + 1
100      CONTINUE
         L = L + 1
200   CONTINUE

      RETURN
      END
**************************************************************************
**************************************************************************
**    SUBROUTINE DET33(A,VAL)                                           **
**                                                                      **
**    PROGRAM DESCRIPTION                                               **
**                                                                      **
**    THIS ROUTINE WILL COMPUTE THE DETERMINANT VALUE OF A 3 X 3        **
**    DETERMINANT OR MATRIX                                             **
**                                                                      **
**                                                                      **
**    BY:       ASHIT R. GANDHI                                         **
**    DATE:     11/12/88                                                **
**                                                                      **
**    PARAMETERS USED:                                                  **
**                                                                      **
**    A   = MATRIX/DETERMINANT WHOSE VALUE IS TO BE FOUND (REAL,I/P)    **
**    VAL = DETERMINANT VALUE (REAL,O/P)                                **
**                                                                      **
**                                                                      **
**************************************************************************
**************************************************************************
      SUBROUTINE DET33(A,VAL)

      REAL A(3,3)
      VAL = 0.0

      VAL =   A(1,1)*(A(2,2)*A(3,3)-A(3,2)*A(2,3))
      VAL = - A(1,2)*(A(2,1)*A(3,3)-A(3,1)*A(2,3)) + VAL
      VAL =   A(1,3)*(A(2,1)*A(3,2)-A(3,1)*A(2,2)) + VAL


200   RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: COMLINE
C
C
C     DESCRIPTION: COMPARE A GENERAL SURFACE TO A RULED
C                  SURFACE. THREE COMPARISONS ARE MADE:
C                  TWO ALONG THE EDGES, AND ONE AT THE
C                  MIDDLE OF THE SURFACE
C
C INPUT:
C     PKL          = GENERAL SURFACE
```

```
C      RPKL          = APPROXIMATED RULED SURFACE
C      RD            = RULED DIRECTION
C
C OUTPUT:
C      MAX           = MAXIMUM DISTANCE BETWEEN THE SURFACES
C
C      C. K. WONG
C      9/12/89


      SUBROUTINE COMLINE(PKL,RPKL,RD,MAX)

      REAL PKL(0:3,0:3,3), RPKL(0:3,0:3,3), MAX
     +     ,CUR(4,3),LINE(2,3),LOC(2,3),INTPT(1,3)
     +     ,V(3)

      INTEGER RD

C INITIALIZE THE VARIABLES

      TT=0.
      MAX=0.
      DO 10 I=1,3

C FIND CURVE AND LINE COEFF FOR THE GENERAL AND RULED SURFACES

         CALL CURCOF(PKL,RD,TT,CUR)

         CALL LINECOF(RPKL,RD,TT,LINE)

C FIND THE MAX DISTANCE BETWEEN THE CURVE AND THE LINE

         CALL MAXCL(CUR,LINE,MAX)

C INCREMENT LINE VARIABLE

         TT=TT+.5
 10   CONTINUE



      RETURN
      END
C--------------------------------------------------
C
C      SUBROUTINE: CURCOF
C
C      DESCRIPTION: FIND THE COEFF. OF A CURVE THAT
C                   LIES ON A SURFACE
C
C INPUT:
C      PKL           = SURFACE
C      RD            = DIRECTION OF THE CURVE
C      TT            = PARAMETRIC VARIABLE FOR THE CURVE
C
C OUTPUT:
C      CUR           = CURVE COEFF.
C
C      C. K. WONG
C      9/12/89
C
C

      SUBROUTINE CURCOF(PKL,RD,TT,CUR)

      REAL PKL(0:3,0:3,3), TT, CUR(4,3), SUR(16,3)
```

```
      INTEGER RD

C FIND THE COEFF. OF THE SURFACE

      CALL BFOR(PKL,SUR)

C RULED IN W DIRECTION: FIND CURVE IN U DIRECTION
C U : CUBIC

      IF(RD.EQ.0) THEN
         DO 10 I=1,3
         CUR(1,I)=(SUR(1,I)*(TT**3))+(SUR(5,I)*(TT**2))+
     +          (SUR(9,I)*(TT))+SUR(13,I)
         CUR(2,I)=(SUR(2,I)*(TT**3))+(SUR(6,I)*(TT**2))+
     +          (SUR(10,I)*(TT))+SUR(14,I)
         CUR(3,I)=(SUR(3,I)*(TT**3))+(SUR(7,I)*(TT**2))+
     +          (SUR(11,I)*(TT))+SUR(15,I)
         CUR(4,I)=(SUR(4,I)*(TT**3))+(SUR(8,I)*(TT**2))+
     +          (SUR(12,I)*(TT))+SUR(16,I)
 10      CONTINUE
       ELSE

C RULED IN U DIRECTION: FIND CURVE IN W DIRECTION
C W : CUBIC

         DO 20 I=1,3
         CUR(1,I)=(SUR(1,I)*(TT**3))+(SUR(2,I)*(TT**2))+
     +          (SUR(3,I)*(TT))+SUR(4,I)
         CUR(2,I)=(SUR(5,I)*(TT**3))+(SUR(6,I)*(TT**2))+
     +          (SUR(7,I)*(TT))+SUR(8,I)
         CUR(3,I)=(SUR(9,I)*(TT**3))+(SUR(10,I)*(TT**2))+
     +          (SUR(11,I)*(TT))+SUR(12,I)
         CUR(4,I)=(SUR(13,I)*(TT**3))+(SUR(14,I)*(TT**2))+
     +          (SUR(15,I)*(TT))+SUR(16,I)
 20      CONTINUE
      ENDIF


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: BFOR
C
C     DESCRIPTION: CALCULATE THE COEFF. OF A B-SPLINE SURFACE.
C
C INPUT:
C     PKL            = SURFACE
C
C OUTPUT:
C     C              = SURFACE COEFF.
C
C     C. K. WONG
C     10/12/89
C
C


      SUBROUTINE BFOR(PKL,C)

      REAL PKL(0:3,0:3,3),P(4,4),C(16,3)

C INITIALIZE THE VARIABLES

      R36=1./36.
      R12=1./12.
```

```
      R4=1./4.
      R2=1./2.
      R6=1./6.
      R3=1./3.
      R9=1./9.

C SURFACE COEFF. FOR X, Y, & Z

      DO 10 I=1,3

         DO 20 J=1,4
              DO 30 K=1,4
                   P(J,K)=PKL(J-1,K-1,I)
 30           CONTINUE
 20      CONTINUE

C SURFACE COEFF.

       C(1,I)=R36*P(1,1)-R12*P(1,2)+R12*P(1,3)-R36*P(1,4)-R12*P(2,1)
     +        +R4*P(2,2)-R4*P(2,3)+R12*P(2,4)+R12*P(3,1)-R4*P(3,2)
     +        +R4*P(3,3)-R12*P(3,4)-R36*P(4,1)+R12*P(4,2)-R12*P(4,3)
     +        +R36*P(4,4)
       C(2,I)=-R12*P(1,1)+R6*P(1,2)-R12*P(1,3)+R4*P(2,1)-R2*P(2,2)
     +        +R4*P(2,3)-R4*P(3,1)+R2*P(3,2)-R4*P(3,3)+R12*P(4,1)
     +        -R6*P(4,2)+R12*P(4,3)
       C(3,I)=R12*P(1,1)-R12*P(1,3)-R4*P(2,1)+R4*P(2,3)+R4*P(3,1)
     +        -R4*P(3,3)-R12*P(4,1)+R12*P(4,3)
       C(4,I)=-R36*P(1,1)-R9*P(1,2)-R36*P(1,3)+R12*P(2,1)+R3*P(2,2)
     +        +R12*P(2,3)-R12*P(3,1)-R3*P(3,2)-R12*P(3,3)+R36*P(4,1)
     +        +R9*P(4,2)+R36*P(4,3)
       C(5,I)=-R12*P(1,1)+R4*P(1,2)-R4*P(1,3)+R12*P(1,4)+R6*P(2,1)
     +        -R2*P(2,2)+R2*P(2,3)-R6*P(2,4)-R12*P(3,1)+R4*P(3,2)
     +        -R4*P(3,3)+R12*P(3,4)
       C(6,I)=R4*P(1,1)-R2*P(1,2)+R4*P(1,3)-R2*P(2,1)+P(2,2)-R2*P(2,3)
     +        +R4*P(3,1)-R2*P(3,2)+R4*P(3,3)
       C(7,I)=-R4*P(1,1)+R4*P(1,3)+R2*P(2,1)
     +         -R2*P(2,3)-R4*P(3,1)+R4*P(3,3)
       C(8,I)=R12*P(1,1)+R3*P(1,2)+R12*P(1,3)-R6*P(2,1)-(2./3.)*P(2,2)
     +        -R6*P(2,3)+R12*P(3,1)+R3*P(3,2)+R12*P(3,3)
       C(9,I)=R12*P(1,1)-R4*P(1,2)+R4*P(1,3)-R12*P(1,4)-R12*P(3,1)
     +        +R4*P(3,2)-R4*P(3,3)+R12*P(3,4)
       C(10,I)=-R4*P(1,1)+R2*P(1,2)-R4*P(1,3)+R4*P(3,1)-R2*P(3,2)
     +         +R4*P(3,3)
       C(11,I)=R4*P(1,1)-R4*P(1,3)-R4*P(3,1)+R4*P(3,3)
       C(12,I)=-R12*P(1,1)-R3*P(1,2)-R12*P(1,3)+R12*P(3,1)+R3*P(3,2)
     +         +R12*P(3,3)
       C(13,I)=-R36*P(1,1)+R12*P(1,2)-R12*P(1,3)+R36*P(1,4)-R9*P(2,1)
     +         +R3*P(2,2)-R3*P(2,3)+R9*P(2,4)-R36*P(3,1)+R12*P(3,2)
     +         -R12*P(3,3)+R36*P(3,4)
       C(14,I)=R12*P(1,1)-R6*P(1,2)+R12*P(1,3)+R3*P(2,1)-(2./3.)*P(2,2)
     +         +R3*P(2,3)+R12*P(3,1)-R6*P(3,2)+R12*P(3,3)
       C(15,I)=-R12*P(1,1)+R12*P(1,3)-R3*P(2,1)+R3*P(2,3)-R12*P(3,1)
     +         +R12*P(3,3)
       C(16,I)=R36*P(1,1)+R9*P(1,2)+R36*P(1,3)+R9*P(2,1)+(4./9.)*P(2,2)
     +         +R9*P(2,3)+R36*P(3,1)+R9*P(3,2)+R36*P(3,3)


 10   CONTINUE

      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: LINECOF
C
C     DESCRIPTION: FIND THE COEFF. OF A LINE THAT
C                  LIES ON A RULED SURFACE
```

```
C
C INPUT:
C       RPKL          = RULED SURFACE
C       RD            = RULED DIRECTION
C       TT            = PARAMETRIC VALUE FOR THE LINE
C
C OUTPUT:
C       LINE          = LINE COEFF.
C
C       C. K. WONG
C       9/12/89
C

      SUBROUTINE LINECOF (RPKL,RD,TT,LINE)

      REAL RPKL(0:3,0:3,3),TT,LINE(2,3),SUR(16,3)

      INTEGER RD


C SURFACE COEFF.

      CALL BFOR(RPKL,SUR)

C RULED IN W DIRECTION: LINE COEFF. IN W DIRECTION

      IF(RD.EQ.0) THEN
         DO 10 I=1,3
            LINE(1,I)=(SUR(3,I)*(TT**3))+(SUR(7,I)*(TT**2))+
     +                (SUR(11,I)*TT)+SUR(15,I)
            LINE(2,I)=(SUR(4,I)*(TT**3))+(SUR(8,I)*(TT**2))+
     +                (SUR(12,I)*TT)+SUR(16,I)
 10      CONTINUE
        ELSE

C RULED IN U DIRECTION: LINE COEFF. IN U DIRECTION

         DO 20 I=1,3
            LINE(1,I)=(SUR(9,I)*(TT**3))+(SUR(10,I)*(TT**2))+
     +                (SUR(11,I)*TT)+SUR(12,I)
            LINE(2,I)=(SUR(13,I)*(TT**3))+(SUR(14,I)*(TT**2))+
     +                (SUR(15,I)*TT)+SUR(16,I)
 20      CONTINUE
      ENDIF


      RETURN
      END
C--------------------------------------------------
C
C       SUBROUTINE: MAXCL
C
C       DESCRIPTION: MAXIMUM DISTANCE BETWEEN A CURVE AND A LINE
C
C INPUT:
C       CUR           = CUR COEFF.
C       LINE          = LINE COEFF.
C
C OUTPUT:
C       MAX           = MAXIMUM DISTANCE BETWEEN THE CURVE
C                       AND THE LINE
C
C       C. K. WONG
C       9/12/89
C
C
```

```
      SUBROUTINE MAXCL(CUR,LINE,MAX)

      REAL CUR(4,3), LINE(2,3), LOC(2,3), COF(4)
     +     ,NOR(3), RO(3), MAX


C INITIALIZE THE VARIABLE

      TOL=.1E-10

C FIND THE NORMAL TO THE LINE

      CALL NORMAL(LINE,NOR)

C FIND THE POINT (PARAMETRIC VALUE) ON THE CURVE
C AT WHICH IS THE MAX DISTANCE FROM THE LINE

      COF(2)=0.
      COF(3)=0.
      COF(4)=0.
      DO 20 I=1,3
         COF(2)=COF(2)+(NOR(I)*CUR(1,I))
         COF(3)=COF(3)+(NOR(I)*CUR(2,I))
         COF(4)=COF(4)+(NOR(I)*CUR(3,I))
 20   CONTINUE
      COF(2)=3.*COF(2)
      COF(3)=2.*COF(3)

C SOLVE FOR THE PARAMETRIC VALUE
      IF((ABS(COF(2)).LT.TOL).AND.(ABS(COF(3)).LT.TOL)) THEN

C NO SOLUTION
         RETURN
       ELSEIF(ABS(COF(2)).LT.TOL) THEN
         CALL LINEAR(COF,0.,1.,IR,RO)
       ELSE
         CALL QUADRA(COF,0.,1.,IR,RO)
      ENDIF

C NO SOLUTION
      IF(IR.EQ.0) RETURN

C FIND THE POINTS (CARTESIAN COOR.) ON THE CURVE

      DO 100 K=1,IR
         VV=RO(K)
         DO 30 I=1,3
              DO 40 J=1,4
                 COF(J)=CUR(J,I)
 40           CONTINUE
              CALL EVAFUN(COF,VV,FF)
              LOC(K,I)=FF
 30      CONTINUE
 100  CONTINUE


C FIND THE MINIMUM DISTANCE BETWEEN A POINT AND A LINE
      CALL MINPL(LINE,IR,LOC,MAX)


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: MINPL
C
C     DESCRIPTION: FIND THE MINIMUM DISTANCE BETWEEN A POINT
```

```
C                   AND A STRAIGHT LINE
C
C INPUT:
C     LINE          = LINE COEFF.
C     IR            = NO. OF POINTS
C     LOC           = POINTS
C
C OUTPUT:
C     MAX           = DISTANCE BETWEEN THE POINTS AND THE LINE
C
C     C. K. WONG
C     9/12/89
C
C

      SUBROUTINE MINPL(LINE,IR,LOC,MAX)

      REAL LINE(2,3),LOC(2,3),MAX, XYZ(3)

C SET TOLERANCE
      TOL=.1E-10

      DO 55 J=1,IR
         BB=0.
         AA=0.

C THE PARAMETRIC VALUE OF THE LINE AT WHICH IS THE MAX.
C DISTANCE FROM THE POINT

         DO 10 I=1,3
            BB=BB+(LINE(1,I)*(LINE(2,I)-LOC(J,I)))
            AA=AA+(LINE(1,I)**2)
 10      CONTINUE
         IF(ABS(AA).LT.TOL) GOTO 55
         T=-(BB/AA)

C THE POINT AT T > 1 OR T < 0.
         IF((T.GT.1.).OR.(T.LT.0.)) GOTO 55

C THE POINT ON THE LINE

         DO 20 I=1,3
            XYZ(I)=LINE(1,I)*T+LINE(2,I)
 20      CONTINUE

C DISTANCE BETWEEN THE POINT AND THE LINE

         DIS=0.
         DO 30 I=1,3
            DIS=DIS+((XYZ(I)-LOC(J,I))**2)
 30      CONTINUE
         DIS=DIS**.5

C COMPARE WITH MAX

         IF(DIS.GT.MAX) MAX=DIS

 55   CONTINUE


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: WRPKL
C
C     DESCRIPTION: WRITE SINGLE RULED SURFACE TO A
```

```
C                    RULED SURFACE ARRAY
C
C INPUT:
C      RPKL          = SINGLE RULED SURFACE
C      INOSUB        = PATCH IDENTIFIER
C
C OUTPUT:
C      SUBPKL        = ARRAY OF RULED SURFACES
C
C      C. K. WONG
C      9/12/89
C
      SUBROUTINE WRPKL(INOSUB,RPKL,SUBPKL)

      REAL RPKL(0:3,0:3,3),SUBPKL(0:3,0:3,3,15)

C DESIGNATE THE POSITION
      IP=INOSUB

      DO 10 I=1,4
         DO 20 J=1,4
              DO 30 K=1,3
                   SUBPKL(I-1,J-1,K,IP)=RPKL(I-1,J-1,K)
 30           CONTINUE
 20      CONTINUE
 10   CONTINUE

      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: SUBSUR
C
C     DESCRIPTION: FIND 16 POINTS ON A BICUBIC SURFACE TO FORM
C                  A SUB-SURFACE TO THE GIVEN RESPECTIVE PARAMETRIC
C                  VALUES
C
C INPUT:
C     PKL          = SURFACE
C     U1,U2,W1,W2  = STARTING AND ENDING PARAMETRIC VALUES
C                    FOR THE POINTS
C     RD           = RULED DIRECTION
C
C OUTPUT:
C     S16          = 16 POINTS ON THE SURFACE
C
C     C. K. WONG
C     9/12/89
C
C

      SUBROUTINE SUBSUR(PKL,U1,U2,W1,W2,RD,S16)

      REAL PKL(0:3,0:3,3),S16(0:3,0:3,3),
     +     PST1(1,3)

      INTEGER RD


C FIND 16 POINTS ON THE SURFACE: WITH EQUAL INTERVAL ON
C BOTH DIRECTIONS

      EINTU=(U2-U1)/3.
      EINTW=(W2-W1)/3.

           WW=W1-EINTW
```

```
      DO 20 I5=1,4
         WW=WW+(EINTW)
         UU=U1-EINTU
         DO 30 J5=1,4
              UU=UU+EINTU
              CALL FBPT(PKL,UU,WW,PST1,ISURF)
              DO 65 II1=1,3
                  S16(J5-1,I5-1,II1)=PST1(1,II1)
65            CONTINUE
30       CONTINUE
20    CONTINUE

   RETURN
   END
```

Appendix G : Program BRINTSS

```
C-------------------------------------------------
C
C     SUBROUTINE: BRINTSS
C
C     DESCRIPTION: FIND THE INTERSECTION POINTS OF TWO SPECIAL
C                  B-SPLINE RULED SURFACES
C
C INPUT:
C     PKL1,PKL2     = INTERSECTING SURFACES
C
C OUTPUT:
C     INTF          = INTERSECTION FLAG
C     TTUWG         = FOUR INTERSECTION POINTS (ONE SEGMENT
C                     OF THE WHOLE INTERSECTION CURVE)
C
C     C. K. WONG
C     1/12/90
C
C

      SUBROUTINE BRINTSS(PKL1,PKL2,INTF,TTUWG)

      REAL PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)
     +    ,TXYZ1(3,4),TXYZ2(3,4)
     +    ,TUW(4),TUW2(4),TTUW(8,4),TTUWG(4,3)

      INTEGER IUW1(3),IUW2(3)
      LOGICAL INTF


C SET THE INTERSECTION FLAG

      INTF=.TRUE.


C FIND THE SURFACE COEFF. OF BOTH SURFACES
```

```
      CALL SBFOR(PKL1,TXYZ1,IUW1)
      CALL SBFOR(PKL2,TXYZ2,IUW2)

C TEST WHETHER IT IS A PARALLEL CASE

      IF((IUW1(1).EQ.IUW2(1)).AND.(IUW1(2).EQ.IUW2(2)).AND.
     +   (IUW1(3).EQ.IUW2(3))) THEN
              CALL PARALL(TXYZ1,TXYZ2,PKL1,PKL2
     +                   ,IUW1,IUW2,INTF,TTUWG)
              RETURN
      ENDIF

      IF((IUW1(1).NE.IUW2(1)).AND.(IUW1(2).NE.IUW2(2)).AND.
     +   (IUW1(3).NE.IUW2(3))) THEN
              CALL PARALL(TXYZ1,TXYZ2,PKL1,PKL2
     +                   ,IUW1,IUW2,INTF,TTUWG)
              RETURN
      ENDIF

C INITIALIZE THE COUNTER FOR INTERSECTION POINTS AT THE EDGE

      II=0

C START THE ELIMINATION ALONG THE EDGE FOR THE
C INTERSECTION POINTS, AND PUT THEM IN TTUW :

C ELIMINATION OF THE REPEATED VARIABLE
C ON THE FIRST SURFACE

      UWR1=0.
      CALL INSURR(UWR1,TXYZ1,TXYZ2,PKL1,PKL2,IUW1,IUW2,TUW)
      CALL PUTUW1(II,TUW,TTUW)

      UWR1=1.
      CALL INSURR(UWR1,TXYZ1,TXYZ2,PKL1,PKL2,IUW1,IUW2,TUW)
      CALL PUTUW1(II,TUW,TTUW)

C ELIMINATION OF THE REPEATED VARIABLE
C ON THE SECOND SURFACE

      UWR2=0.
      CALL INSURR(UWR2,TXYZ2,TXYZ1,PKL2,PKL1,IUW2,IUW1,TUW)
      TUW2(3)=TUW(1)
      TUW2(4)=TUW(2)
      TUW2(1)=TUW(3)
      TUW2(2)=TUW(4)
      CALL PUTUW1(II,TUW2,TTUW)

      UWR2=1.
      CALL INSURR(UWR2,TXYZ2,TXYZ1,PKL2,PKL1,IUW2,IUW1,TUW)
      TUW2(3)=TUW(1)
      TUW2(4)=TUW(2)
      TUW2(1)=TUW(3)
      TUW2(2)=TUW(4)
      CALL PUTUW1(II,TUW2,TTUW)

C ELIMINATION OF THE SINGULAR VARIABLE
C ON THE FIRST SURFACE

      UWS1=0.
      CALL INSURS(UWS1,TXYZ1,TXYZ2,PKL1,PKL2,IUW1,IUW2,TUW)
      CALL PUTUW1(II,TUW,TTUW)

      UWS1=1.
      CALL INSURS(UWS1,TXYZ1,TXYZ2,PKL1,PKL2,IUW1,IUW2,TUW)
      CALL PUTUW1(II,TUW,TTUW)
```

```
C ELIMINATION OF THE SINGULAR VARIABLE
C ON THE SECOND SURFACE

      UWS2=0.
      CALL INSURS(UWS2,TXYZ2,TXYZ1,PKL2,PKL1,IUW2,IUW1,TUW)
      TUW2(3)=TUW(1)
      TUW2(4)=TUW(2)
      TUW2(1)=TUW(3)
      TUW2(2)=TUW(4)
      CALL PUTUW1(II,TUW2,TTUW)

      UWS2=1.
      CALL INSURS(UWS2,TXYZ2,TXYZ1,PKL2,PKL1,IUW2,IUW1,TUW)
      TUW2(3)=TUW(1)
      TUW2(4)=TUW(2)
      TUW2(1)=TUW(3)
      TUW2(2)=TUW(4)
      CALL PUTUW1(II,TUW2,TTUW)

C GET RID OF THOSE UNWANTED DATA FROM TTUW, AND
C CHECK FOR BAD CASE

      CALL GRID(TTUW,IN)
      IF(IN.EQ.0) THEN
         INTF=.FALSE.
         RETURN
        ELSEIF(IN.EQ.1) THEN
         WRITE(6,*)'BAD CASE (BRINTSS) IN=1'
         INTF=.FALSE.
         RETURN
      ENDIF

C FIND THE MAX DIFFERENCE BETWEEN THE VARIABLES.

      CALL FMAX(TTUW,IN,JR1,JR2,JC)

C FIND FOUR POINTS ON THE INTERSECTION CURVE:
C TWO ON THE EDGE( MAX & MIN VARIABLES), AND
C TWO IN BETWEEN

      CALL PT4SS(TXYZ1,TXYZ2,PKL1,PKL2,IUW1,IUW2
     +          ,JR1,JR2,JC,TTUW,TTUWG,I2)

C BAD CURVE, COULDN'T FIND THE INTERMEDIATE POINTS

      IF(I2.EQ.0) THEN
         INTF=.FALSE.
         WRITE(6,*)'BAD CURVE (PT4SS) I2=0'
         RETURN
      ENDIF



      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: SBFOR
C
C     DESCRIPTION: CALCULATE THE SURFACE COEFF. OF
C                  A SPECIAL B-SPLINE RULED SURFACE
C
C INPUT:
C     PKL          = SURFACE
C
C OUTPUT:
C     TXYZ         = SURFACE COEFF.
```

```
C     IUW          = SINGULAR OR REPEATED FLAG
C
C     C. K. WONG
C     9/12/89
C
C

      SUBROUTINE SBFOR(PKL,TXYZ,IUW)

      REAL PKL(0:3,0:3,3),P(4,4),C(16),TXYZ(3,4)
      INTEGER IUW(3)


C SET THE VARIABLES

      R36=1./36.
      R12=1./12.
      R4=1./4.
      R2=1./2.
      R6=1./6.
      R3=1./3.
      R9=1./9.
      T=.001

C CALCULATE THE SURFACE COEFF. FOR X, Y, & Z

      DO 10 I=1,3

         DO 20 J=1,4
              DO 30 K=1,4
                   P(J,K)=PKL(J-1,K-1,I)
 30           CONTINUE
 20      CONTINUE

       C(1)=R36*P(1,1)-R12*P(1,2)+R12*P(1,3)-R36*P(1,4)-R12*P(2,1)
     +      +R4*P(2,2)-R4*P(2,3)+R12*P(2,4)+R12*P(3,1)-R4*P(3,2)
     +      +R4*P(3,3)-R12*P(3,4)-R36*P(4,1)+R12*P(4,2)-R12*P(4,3)
     +      +R36*P(4,4)
       C(2)=-R12*P(1,1)+R6*P(1,2)-R12*P(1,3)+R4*P(2,1)-R2*P(2,2)
     +      +R4*P(2,3)-R4*P(3,1)+R2*P(3,2)-R4*P(3,3)+R12*P(4,1)
     +      -R6*P(4,2)+R12*P(4,3)
       C(3)=R12*P(1,1)-R12*P(1,3)-R4*P(2,1)+R4*P(2,3)+R4*P(3,1)
     +      -R4*P(3,3)-R12*P(4,1)+R12*P(4,3)
       C(4)=-R36*P(1,1)-R9*P(1,2)-R36*P(1,3)+R12*P(2,1)+R3*P(2,2)
     +      +R12*P(2,3)-R12*P(3,1)-R3*P(3,2)-R12*P(3,3)+R36*P(4,1)
     +      +R9*P(4,2)+R36*P(4,3)
       C(5)=-R12*P(1,1)+R4*P(1,2)-R4*P(1,3)+R12*P(1,4)+R6*P(2,1)
     +      -R2*P(2,2)+R2*P(2,3)-R6*P(2,4)-R12*P(3,1)+R4*P(3,2)
     +      -R4*P(3,3)+R12*P(3,4)
       C(6)=R4*P(1,1)-R2*P(1,2)+R4*P(1,3)-R2*P(2,1)+P(2,2)-R2*P(2,3)
     +      +R4*P(3,1)-R2*P(3,2)+R4*P(3,3)
       C(7)=-R4*P(1,1)+R4*P(1,3)+R2*P(2,1)-R2*P(2,3)-R4*P(3,1)+R4*P(3,3)
       C(8)=R12*P(1,1)+R3*P(1,2)+R12*P(1,3)-R6*P(2,1)-(2./3.)*P(2,2)
     +      -R6*P(2,3)+R12*P(3,1)+R3*P(3,2)+R12*P(3,3)
       C(9)=R12*P(1,1)-R4*P(1,2)+R4*P(1,3)-R12*P(1,4)-R12*P(3,1)
     +      +R4*P(3,2)-R4*P(3,3)+R12*P(3,4)
       C(10)=-R4*P(1,1)+R2*P(1,2)-R4*P(1,3)+R4*P(3,1)-R2*P(3,2)
     +       +R4*P(3,3)
       C(11)=R4*P(1,1)-R4*P(1,3)-R4*P(3,1)+R4*P(3,3)
       C(12)=-R12*P(1,1)-R3*P(1,2)-R12*P(1,3)+R12*P(3,1)+R3*P(3,2)
     +       +R12*P(3,3)
       C(13)=-R36*P(1,1)+R12*P(1,2)-R12*P(1,3)+R36*P(1,4)-R9*P(2,1)
     +       +R3*P(2,2)-R3*P(2,3)+R9*P(2,4)-R36*P(3,1)+R12*P(3,2)
     +       -R12*P(3,3)+R36*P(3,4)
       C(14)=R12*P(1,1)-R6*P(1,2)+R12*P(1,3)+R3*P(2,1)-(2./3.)*P(2,2)
     +       +R3*P(2,3)+R12*P(3,1)-R6*P(3,2)+R12*P(3,3)
```

```
      C(15)=-R12*P(1,1)+R12*P(1,3)-R3*P(2,1)+R3*P(2,3)-R12*P(3,1)
     +       +R12*P(3,3)
      C(16)=R36*P(1,1)+R9*P(1,2)+R36*P(1,3)+R9*P(2,1)+(4./9.)*P(2,2)
     +       +R9*P(2,3)+R36*P(3,1)+R9*P(3,2)+R36*P(3,3)

C CHECK FOR THE REPEATED OR SINGULAR VARIABLE

        IF((ABS(C(13)).LT.T).AND.(ABS(C(14)).LT.T).AND.
     +     (ABS(C(15)).LT.T)) THEN
             IUW(I)=0
             TXYZ(I,1)=C(4)
             TXYZ(I,2)=C(8)
             TXYZ(I,3)=C(12)
             TXYZ(I,4)=C(16)
            ELSE
             IUW(I)=1
             TXYZ(I,1)=C(13)
             TXYZ(I,2)=C(14)
             TXYZ(I,3)=C(15)
             TXYZ(I,4)=C(16)
        ENDIF

 10   CONTINUE

      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: INSURR
C
C     DESCRIPTION: START THE ELIMINATION BY DEFINING REPEATED
C                  VARIABLE ON THE FIRST INPUT SURFACE
C
C INPUT:
C     UWR1          = PARAMETRIC VALUE
C     TXYZ1,TXYZ2   = SURFACES COEFF.
C     PKL1,PKL2     = SURFACES
C     IUW1,IUW2     = SINGULAR OR REPEATED VARIABLE FLAGS
C
C OUTPUT:
C     TUW           = INTERSECTION POINTS ALONG THE EDGES
C
C     C. K. WONG
C     1/20/90
C
C

      SUBROUTINE INSURR(UWR1,TXYZ1,TXYZ2,PKL1,PKL2,IUW1,IUW2,TUW)

      REAL TXYZ1(3,4),TXYZ2(3,4)
     +    ,TUW(4),COF1A(4),COF1B(4),COF(4)
     +    ,UA1(10),WA1(10),UA2(10),WA2(10)
     +    ,RO1(3),RO2(3),RO3(3)
     +    ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER IUW1(3),IUW2(3)


C LOOK FOR THE SINGULAR VARIABLE ON THE FIRST SURFACE
      IF(IUW1(1).EQ.IUW1(2)) THEN
         IS1=3
         IR1=IUW1(1)
        ELSEIF(IUW1(1).EQ.IUW1(3)) THEN
         IS1=2
         IR1=IUW1(1)
        ELSEIF(IUW1(2).EQ.IUW1(3)) THEN
         IS1=1
```

```
         IR1=IUW1(2)
      ENDIF

C LOOK FOR THE SINGULAR VARIABLE ON THE SECOND SURFACE
      IF(IUW2(1).EQ.IUW2(2)) THEN
         IS2=3
         IR2=IUW2(1)
        ELSEIF(IUW2(1).EQ.IUW2(3)) THEN
         IS2=2
         IR2=IUW2(1)
        ELSEIF(IUW2(2).EQ.IUW2(3)) THEN
         IS2=1
         IR2=IUW2(2)
      ENDIF

C DETERMINE THE ORDER OF SOLVING THE EQUATIONS
      IF(IS1.EQ.3) THEN
         IO3=3
         IF(IS2.EQ.2) THEN
              IO1=1
              IO2=2
           ELSE
              IO1=2
              IO2=1
         ENDIF

       ELSEIF(IS1.EQ.2) THEN
         IO3=2
         IF(IS2.EQ.3) THEN
              IO1=1
              IO2=3
           ELSE
              IO1=3
              IO2=1
         ENDIF

       ELSEIF(IS1.EQ.1) THEN
         IO3=1
         IF(IS2.EQ.3) THEN
              IO1=2
              IO2=3
           ELSE
              IO1=3
              IO2=2
         ENDIF

      ENDIF

C DEFINE THE REPEATED VARIABLE ON THE FIRST SURFACE
      DO 10 I=1,4
         COF1A(I)=TXYZ1(IO1,I)
         COF1B(I)=TXYZ1(IO2,I)
 10   CONTINUE

      CALL EVAFUN(COF1A,UWR1,F1)
      CALL EVAFUN(COF1B,UWR1,F2)

C SOLVE FOR THE TWO VARIABLES ON THE SECOND SURFACE
      DO 20 I=1,4
         COF1A(I)=TXYZ2(IO1,I)
         COF1B(I)=TXYZ2(IO2,I)
 20   CONTINUE
         COF1A(4)=COF1A(4)-F1
         COF1B(4)=COF1B(4)-F2

      IFOUND=1
```

```
      CALL CUROOT(COF1A,IC1,RO1)


      DO 30 I=1,IC1
         UWR2=RO1(I)
         CALL CUROOT(COF1B,IC2,RO2)
         DO 40 J=1,IC2
              UWS2=RO2(J)

C SOLVE FOR THE SINGULAR VARIABLE ON THE
C FIRST SURFACE
              DO 50 K=1,4
                   COF(K)=TXYZ2(IO3,K)
 50           CONTINUE
              CALL EVAFUN(COF,UWR2,FF)

              DO 60 K=1,4
                   COF(K)=TXYZ1(IO3,K)
 60           CONTINUE
              COF(4)=COF(4)-FF

              CALL CUROOT(COF,IC3,RO3)

              DO 70 K=1,IC3
                   UWS1=RO3(K)
                   UA1(IFOUND)=UWR1
                   WA1(IFOUND)=UWS1
                   UA2(IFOUND)=UWR2
                   WA2(IFOUND)=UWS2
                   IFOUND=IFOUND+1
 70           CONTINUE
 40      CONTINUE
 30   CONTINUE


      IFOUND=IFOUND-1

C NO INTERSECTION
      IF(IFOUND.EQ.0) THEN

         DO 400 I=1,4
              TUW(I)=2.
 400     CONTINUE

        ELSE

C COMPARE THE RESULTS AND PICK THE PAIR OF POINTS
C THAT HAS THE SMALLEST DISTANCE AS THE RESULT

         IF((IR1.EQ.0).AND.(IR2.EQ.0)) THEN
             CALL COMRELT(PKL1,PKL2,UA1,WA1,UA2,WA2,IFOUND,TUW)
          ELSEIF((IR1.EQ.1).AND.(IR2.EQ.0)) THEN
             CALL COMRELT(PKL1,PKL2,WA1,UA1,UA2,WA2,IFOUND,TUW)
          ELSEIF((IR1.EQ.0).AND.(IR2.EQ.1)) THEN
             CALL COMRELT(PKL1,PKL2,UA1,WA1,WA2,UA2,IFOUND,TUW)
          ELSEIF((IR1.EQ.1).AND.(IR2.EQ.1)) THEN
             CALL COMRELT(PKL1,PKL2,WA1,UA1,WA2,UA2,IFOUND,TUW)
         ENDIF

      ENDIF


      RETURN
      END
C---------------------------------------------------
C
C     SUBROUTINE: INSURS
```

```
C
C     DESCRIPTION: START THE ELIMINATION BY DEFINING THE
C                  SINGULAR VARIABLE ON THE FIRST SURFACE
C
C INPUT:
C     UWS1         = PARAMETRIC VALUE
C     TXYZ1,TXYZ2  = SURFACES COEFF.
C     PKL1,PKL2    = SURFACES
C     IUW1,IUW2    = SINGULAR OR REPEATED VARIABLE FLAGS
C
C OUTPUT:
C     TUW          = INTERSECTION POINTS ALONG THE EDGES
C
C     C. K. WONG
C     2/12/90

      SUBROUTINE INSURS(UWS1,TXYZ1,TXYZ2,PKL1,PKL2,IUW1,IUW2,TUW)

      REAL TXYZ1(3,4),TXYZ2(3,4)
     +    ,TUW(4),COF(4)
     +    ,UA1(10),WA1(10),UA2(10),WA2(10)
     +    ,RO1(3),RO2(3),RO3(3)
     +    ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER IUW1(3),IUW2(3)


C LOOK FOR THE SINGULAR VARIABLE ON THE FIRST SURFACE
      IF(IUW1(1).EQ.IUW1(2)) THEN
         IS1=3
         IR1=IUW1(1)
        ELSEIF(IUW1(1).EQ.IUW1(3)) THEN
         IS1=2
         IR1=IUW1(1)
        ELSEIF(IUW1(2).EQ.IUW1(3)) THEN
         IS1=1
         IR1=IUW1(2)
      ENDIF

C LOOK FOR THE SINGULAR VARIABLE ON THE SECOND SURFACE
      IF(IUW2(1).EQ.IUW2(2)) THEN
         IS2=3
         IR2=IUW2(1)
        ELSEIF(IUW2(1).EQ.IUW2(3)) THEN
         IS2=2
         IR2=IUW2(1)
        ELSEIF(IUW2(2).EQ.IUW2(3)) THEN
         IS2=1
         IR2=IUW2(2)
      ENDIF

C DETERMINE THE ORDER OF SOLVING THE EQUATIONS
      IF(IS1.EQ.3) THEN
         IO1=3
         IF(IS2.EQ.2) THEN
              IO2=1
              IO3=2
           ELSE
              IO2=2
              IO3=1
         ENDIF

       ELSEIF(IS1.EQ.2) THEN
         IO1=2
         IF(IS2.EQ.3) THEN
              IO2=1
              IO3=3
```

```
          ELSE
             IO2=3
             IO3=1
        ENDIF

      ELSEIF(IS1.EQ.1) THEN
        IO1=1
        IF(IS2.EQ.3) THEN
             IO2=2
             IO3=3
          ELSE
             IO2=3
             IO3=2
        ENDIF

      ENDIF

C DEFINE THE SINGULAR VARIABLE ON THE FIRST SURFACE
      DO 10 I=1,4
         COF(I)=TXYZ1(IO1,I)
 10   CONTINUE

      CALL EVAFUN(COF,UWS1,F1)

C SOLVE FOR THE SINGULAR VARIABLE ON THE SECOND SURFACE

      DO 20 I=1,4
         COF(I)=TXYZ2(IO1,I)
 20   CONTINUE
         COF(4)=COF(4)-F1

      IFOUND=1

      CALL CUROOT(COF,IC1,RO1)

      DO 30 I=1,IC1
         UWR2=RO1(I)

C SOLVE FOR THE REPEATED VARIABLE ON THE FIRST SURFACE

         DO 130 J2=1,4
              COF(J2)=TXYZ2(IO2,J2)
 130     CONTINUE
         CALL EVAFUN(COF,UWR2,F2)

         DO 140 J2=1,4
              COF(J2)=TXYZ1(IO2,J2)
 140     CONTINUE
         COF(4)=COF(4)-F2

         CALL CUROOT(COF,IC2,RO2)

         DO 40 J=1,IC2
              UWR1=RO2(J)

C SOLVE FOR THE SINGULAR VARIABLE ON THE
C SECOND SURFACE
              DO 50 K=1,4
                   COF(K)=TXYZ1(IO3,K)
 50           CONTINUE
              CALL EVAFUN(COF,UWR1,FF)

              DO 60 K=1,4
                   COF(K)=TXYZ2(IO3,K)
 60           CONTINUE
              COF(4)=COF(4)-FF
```

```
            CALL CUROOT(COF,IC3,RO3)

            DO 70 K=1,IC3
                 UWS2=RO3(K)
                 UA1(IFOUND)=UWR1
                 WA1(IFOUND)=UWS1
                 UA2(IFOUND)=UWR2
                 WA2(IFOUND)=UWS2
                 IFOUND=IFOUND+1
 70         CONTINUE
 40      CONTINUE
 30   CONTINUE


      IFOUND=IFOUND-1

C NO INTERSECTION
      IF(IFOUND.EQ.0) THEN

         DO 400 I=1,4
              TUW(I)=2.
 400     CONTINUE

        ELSE

C COMPARE THE RESULTS AND PICK THE PAIR OF POINTS
C THAT HAS THE SMALLEST DISTANCE AS THE RESULT

         IF((IR1.EQ.0).AND.(IR2.EQ.0)) THEN
             CALL COMRELT(PKL1,PKL2,UA1,WA1,UA2,WA2,IFOUND,TUW)
          ELSEIF((IR1.EQ.1).AND.(IR2.EQ.0)) THEN
             CALL COMRELT(PKL1,PKL2,WA1,UA1,UA2,WA2,IFOUND,TUW)
          ELSEIF((IR1.EQ.0).AND.(IR2.EQ.1)) THEN
             CALL COMRELT(PKL1,PKL2,UA1,WA1,WA2,UA2,IFOUND,TUW)
          ELSEIF((IR1.EQ.1).AND.(IR2.EQ.1)) THEN
             CALL COMRELT(PKL1,PKL2,WA1,UA1,WA2,UA2,IFOUND,TUW)
         ENDIF

      ENDIF


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: PT4SS
C
C     DESCRIPTION: FIND FOUR POINTS(EQUAL INTERVAL) ON
C                  THE INTERSECTION CURVE: FOR TWO
C                  SPECIAL RULED SURFACES
C
C INPUT:
C     TXYZ1,TXYZ2  = SURFACES COEFF.
C     PKL1,PKL2    = SURFACES
C     IUW1,IUW2    = SINGULAR OR REPEATED VARIABLE FLAGS
C     JR1,JR2      = ROW IDENTIFIER FOR MAX DIFF. IN
C                    PARAMETRIC VALUE
C     JC           = COLUMN IDENTIFIER FOR MAX DIFF. IN
C                    PARAMETRIC VALUE
C     TTUW         = INTERSECTION POINTS ALONG THE EDGE
C                    IN PARAMETRIC SPACE
C
C OUTPUT:
C     TTUWG        = A SEGMENT OF THE INTERSECTION CURVE
C                    (FOUR POINTS)
C     I2           = BAD CASE FLAG: COULDN'T FIND THE
C                    INTERMEDIATE POINTS
```

```
C
C      C. K. WONG
C      2/12/90
C

      SUBROUTINE PT4SS(TXYZ1,TXYZ2,PKL1,PKL2,IUW1,IUW2
     +                 ,JR1,JR2,JC,TTUW,TTUWG,I2)

      REAL TXYZ1(3,4),TXYZ2(3,4)
     +    ,TUW(4),TUW2(4),TTUW(8,4),TTUWG(4,3)
     +    ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)
     +    ,TPST(1,3),TT5(2,4), TT(2)

      INTEGER IUW1(3),IUW2(3)


C INITIALIZE THE FLAG
      I2=1

C LOOK FOR THE REPEATED VARIABLE ON THE FIRST SURFACE
      IF(IUW1(1).EQ.IUW1(2)) THEN
         IR1=IUW1(1)
        ELSEIF(IUW1(1).EQ.IUW1(3)) THEN
         IR1=IUW1(1)
        ELSEIF(IUW1(2).EQ.IUW1(3)) THEN
         IR1=IUW1(2)
      ENDIF

C LOOK FOR THE REPEATED VARIABLE ON THE SECOND SURFACE
      IF(IUW2(1).EQ.IUW2(2)) THEN
         IR2=IUW2(1)
        ELSEIF(IUW2(1).EQ.IUW2(3)) THEN
         IR2=IUW2(1)
        ELSEIF(IUW2(2).EQ.IUW2(3)) THEN
         IR2=IUW2(2)
      ENDIF

C GET THE PARAMETRIC VALUES FOR MIDDLE TWO POINTS
      TT(1)=(1./3.)*TTUW(JR2,JC)+(2./3.)*TTUW(JR1,JC)
      TT(2)=(2./3.)*TTUW(JR2,JC)+(1./3.)*TTUW(JR1,JC)

C FIND TWO INTERSECTION POINTS IN BETWEEN TWO END POINTS
      DO 10 I=1,2
         V=TT(I)

C CHECK WHICH ROUTINE TO CALL
         IF((JC.EQ.1).OR.(JC.EQ.2)) THEN
              IF((JC.EQ.1).AND.(IR1.EQ.0)) THEN
                   CALL INSURR(V,TXYZ1,TXYZ2,PKL1,PKL2
     +                        ,IUW1,IUW2,TUW)
               ELSEIF((JC.EQ.1).AND.(IR1.EQ.1)) THEN
                   CALL INSURS(V,TXYZ1,TXYZ2,PKL1,PKL2
     +                        ,IUW1,IUW2,TUW)
               ELSEIF((JC.EQ.2).AND.(IR1.EQ.1)) THEN
                   CALL INSURR(V,TXYZ1,TXYZ2,PKL1,PKL2
     +                        ,IUW1,IUW2,TUW)
               ELSEIF((JC.EQ.2).AND.(IR1.EQ.0)) THEN
                   CALL INSURS(V,TXYZ1,TXYZ2,PKL1,PKL2
     +                        ,IUW1,IUW2,TUW)
              ENDIF

           ELSE

              IF((JC.EQ.3).AND.(IR2.EQ.0)) THEN
                   CALL INSURR(V,TXYZ2,TXYZ1,PKL2,PKL1
     +                        ,IUW2,IUW1,TUW2)
               ELSEIF((JC.EQ.3).AND.(IR2.EQ.1)) THEN
```

```
                CALL INSURS(V,TXYZ2,TXYZ1,PKL2,PKL1
     +                      ,IUW2,IUW1,TUW2)
              ELSEIF((JC.EQ.4).AND.(IR2.EQ.1)) THEN
                CALL INSURR(V,TXYZ2,TXYZ1,PKL2,PKL1
     +                      ,IUW2,IUW1,TUW2)
              ELSEIF((JC.EQ.4).AND.(IR2.EQ.0)) THEN
                CALL INSURS(V,TXYZ2,TXYZ1,PKL2,PKL1
     +                      ,IUW2,IUW1,TUW2)
             ENDIF

             TUW(1)=TUW2(3)
             TUW(2)=TUW2(4)
             TUW(3)=TUW2(1)
             TUW(4)=TUW2(2)

         ENDIF

C STORE THE RESULT
         DO 20 J=1,4
             TT5(I,J)=TUW(J)

C BAD DATA
             IF(TUW(J).GT.1.5) THEN
                I2=0
             ENDIF
 20      CONTINUE

 10   CONTINUE


C FIND FOUR POINTS ON THE INTERSECTION CURVE

      CALL FBPT(PKL1,TTUW(JR1,1),TTUW(JR1,2),TPST,IFF)
      DO 500 I=1,3
         TTUWG(1,I)=TPST(1,I)
 500  CONTINUE

      CALL FBPT(PKL1,TTUW(JR2,1),TTUW(JR2,2),TPST,IFF)
      DO 510 I=1,3
         TTUWG(4,I)=TPST(1,I)
 510  CONTINUE

      DO 520 I=1,2
         CALL FBPT(PKL1,TT5(I,1),TT5(I,2),TPST,IFF)
         DO 530 J=1,3
            TTUWG(I+1,J)=TPST(1,J)
 530     CONTINUE

 520  CONTINUE


      RETURN
      END
```

Appendix H : Program BRINTSR

```
C-------------------------------------------------
C
C     SUBROUTINE: BRINTSR
C
C     DESCRIPTION: FIND THE INTERSECTION CURVE OF TWO
C                  B-SPLINE RULED SURFACES (ONE IS SPECIAL
C                  RULED SURFACE)
C
C INPUT:
C     PKL1,PKL2    = INTERSECTING SURFACES
C     ITYPE        = SPECIAL RULED SURFACE FLAG
C     IUW          = SINGULAR OR REPEATED VARIABLE FLAGS
C
C OUTPUT:
C     INTF         = INTERSECTION FLAG
C     TTUWG        = A SEGMENT OF THE INTERSECTION CURVE
C                    (FOUR POINTS)
C
C     C. K. WONG
C     2/12/90
C

      SUBROUTINE BRINTSR (PKL1,PKL2,ITYPE,IUW,INTF,TTUWG)


      REAL PKL1(0:3,0:3,3),PKL2(0:3,0:3,3),TXYZ1(3,8),TXYZ2(3,8),
     +     TTUW(8,4),TUW(4),TTUWG(4,3),MIN

      INTEGER IUW(2),IUW2(2),IUWSR(3)

      LOGICAL INTF

C SET THE INTERSECTION FLAG
      INTF=.FALSE.
```

```
C FIND THE COEFF. OF THE SURFACE:
C SPECIAL CASE STORED IN TXYZ1, AND
C ITYPE TELLS WHICH SURFACE IS A SPECIAL CASE


      IF(ITYPE.EQ.1) THEN
         CALL BRFORSR(PKL1,TXYZ1,IUWSR)
         CALL BRFOR(PKL2,2,TXYZ2,IUW)
        ELSE

C SWAP THE DATA IF SURFACE 2 IS A SPECIAL CASE
         IUW(2)=IUW(1)
         CALL EXPKL(PKL1,PKL2)
         CALL BRFORSR(PKL1,TXYZ1,IUWSR)
         CALL BRFOR(PKL2,2,TXYZ2,IUW)
      ENDIF


C INITIALIZE THE INTERSECTION POINTS AND COUNTER
      DO 132 IIH=1,8
         TTUW(IIH,1)=2.
         TTUW(IIH,2)=2.
         TTUW(IIH,3)=2.
         TTUW(IIH,4)=2.
 132  CONTINUE

C INITIALIZE THE REPEAT FLAG
          IREP=0

 111      II=0

C DEFINE VARIABLE ON SURFACE 1
C FIRST CASE: DEFINE THE REPEATED VARIABLE

          V=0.0
          CALL SR1(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
          CALL PUTUW1(II,TUW,TTUW)

          V=1.0
          CALL SR1(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
          CALL PUTUW1(II,TUW,TTUW)

C DEFINE VARIABLE ON SURFACE 1
C SECOND CASE: DEFINE THE SINGULAR VARIABLE

          V=0.0

          IF(IREP.EQ.1) THEN
             CALL SR2D(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
            ELSE
             CALL SR2(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
          ENDIF
          CALL PUTUW1(II,TUW,TTUW)

          V=1.0
          IF(IREP.EQ.1) THEN
             CALL SR2D(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
            ELSE
             CALL SR2(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
          ENDIF
          CALL PUTUW1(II,TUW,TTUW)


C DEFINE VARIABLE ON SURFACE 2
C FIRST CASE: DEFINE THE CUBIC VARIABLE

          V=0.0
```

```
          CALL SR3(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
          CALL PUTUW1(II,TUW,TTUW)

          V=1.0
          CALL SR3(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
          CALL PUTUW1(II,TUW,TTUW)


C DEFINE VARIABLE ON SURFACE 2
C SECOND CASE: DEFINE THE LINEAR VARIABLE

          V=0.0
          CALL SR4(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
          CALL PUTUW1(II,TUW,TTUW)

          V=1.0
          CALL SR4(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
          CALL PUTUW1(II,TUW,TTUW)

C MAKE SURE THE INTERSECTION DATA IS FILLED OUT

          IF(II.GE.8) GOTO 567
          DO 122 IIH=II+1,8
            TTUW(IIH,1)=2.
            TTUW(IIH,2)=2.
            TTUW(IIH,3)=2.
            TTUW(IIH,4)=2.
 122      CONTINUE

C GET RID OF THOSE UNWANTED DATA
 567      CALL GRID(TTUW,IN)

C BAD CASE: JUST ONE END INTERSECTION POINT
          IF(IN.EQ.1) THEN

C CHECK THE POINT: IS THE POINT LOCATED AT THE SAME
C SPOT ON BOTH SURFACES ?
             CALL TESTBAD(PKL1,PKL2,TTUW(1,1),TTUW(1,2)
     +                   ,TTUW(1,3),TTUW(1,4),MIN)


             IF(MIN.GT.(.5)) THEN
                  IN=0
               ELSE
                IF(IREP.EQ.0) THEN
                  IREP=1
                  GOTO 111
                ENDIF
                  WRITE(6,*)'BAD CASE (IN=1) IN BRINTSR'
             ENDIF
             GOTO 555

           ELSEIF(IN.EQ.0) THEN

C NO INTERSECTION
             GOTO 555

          ENDIF

C FIND THE MAX DIFFERENCE BETWEEN THE VARIABLES.
          CALL FMAX(TTUW,IN,JR1,JR2,IC)

C FIND 4 POINTS ON THE INTERSECTION CURVE, AND STORE
C THEM IN TTUWG

          I2=1
          CALL PT4SR(TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,JR1,JR2,IC,
```

```
     +                  TTUW,TTUWG,I2)

C COULDN'T FIND THE INTERMEDIATE POINT
      IF(I2.EQ.0) THEN
           WRITE(6,*)'BAD CASE(BRINTSR): I2=0'
           GOTO 555
      ENDIF

C SET THE INTERSECTION FLAG
      INTF=.TRUE.

C RESTORE THE DATA IF THEY HAD BEEN SWAPPED
555   IF(ITYPE.EQ.2) THEN
         IUW(1)=IUW(2)
         CALL EXPKL(PKL1,PKL2)
      ENDIF


 5555 RETURN
      END
C---------------------------------------------------
C
C     SUBROUTINE: BRFORSR
C
C     DESCRIPTION: TO CALCULATE THE COEFFICIENTS OF A B-SPLINE
C                  SURFACE (SPECIAL RULED SURFACE)
C
C INPUT:
C     PKL          = SURFACE
C
C OUTPUT:
C     TXYZ         = SURFACE COEFF.
C     IUWSR        = SINGULAR OR REPEATED VARIABLE FLAGS
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE BRFORSR(PKL,TXYZ,IUWSR)

      REAL PKL(0:3,0:3,3),P(4,4),C(16),TXYZ(3,8)
      INTEGER IUWSR(3)

      R36=1./36.
      R12=1./12.
      R4=1./4.
      R2=1./2.
      R6=1./6.
      R3=1./3.
      R9=1./9.
      T=.001


C CALCULATE THE COEFF. FOR X, Y, AND Z
      DO 10 I=1,3

         DO 20 J=1,4
              DO 30 K=1,4
                   P(J,K)=PKL(J-1,K-1,I)
 30           CONTINUE
 20      CONTINUE

       C(1)=R36*P(1,1)-R12*P(1,2)+R12*P(1,3)-R36*P(1,4)-R12*P(2,1)
     +      +R4*P(2,2)-R4*P(2,3)+R12*P(2,4)+R12*P(3,1)-R4*P(3,2)
     +      +R4*P(3,3)-R12*P(3,4)-R36*P(4,1)+R12*P(4,2)-R12*P(4,3)
     +      +R36*P(4,4)
       C(2)=-R12*P(1,1)+R6*P(1,2)-R12*P(1,3)+R4*P(2,1)-R2*P(2,2)
```

```
+       +R4*P(2,3)-R4*P(3,1)+R2*P(3,2)-R4*P(3,3)+R12*P(4,1)
+       -R6*P(4,2)+R12*P(4,3)
  C(3)=R12*P(1,1)-R12*P(1,3)-R4*P(2,1)+R4*P(2,3)+R4*P(3,1)
+       -R4*P(3,3)-R12*P(4,1)+R12*P(4,3)
  C(4)=-R36*P(1,1)-R9*P(1,2)-R36*P(1,3)+R12*P(2,1)+R3*P(2,2)
+       +R12*P(2,3)-R12*P(3,1)-R3*P(3,2)-R12*P(3,3)+R36*P(4,1)
+       +R9*P(4,2)+R36*P(4,3)
  C(5)=-R12*P(1,1)+R4*P(1,2)-R4*P(1,3)+R12*P(1,4)+R6*P(2,1)
+       -R2*P(2,2)+R2*P(2,3)-R6*P(2,4)-R12*P(3,1)+R4*P(3,2)
+       -R4*P(3,3)+R12*P(3,4)
  C(6)=R4*P(1,1)-R2*P(1,2)+R4*P(1,3)-R2*P(2,1)+P(2,2)-R2*P(2,3)
+       +R4*P(3,1)-R2*P(3,2)+R4*P(3,3)
  C(7)=-R4*P(1,1)+R4*P(1,3)+R2*P(2,1)-R2*P(2,3)-R4*P(3,1)+R4*P(3,3)
  C(8)=R12*P(1,1)+R3*P(1,2)+R12*P(1,3)-R6*P(2,1)-(2./3.)*P(2,2)
+       -R6*P(2,3)+R12*P(3,1)+R3*P(3,2)+R12*P(3,3)
  C(9)=R12*P(1,1)-R4*P(1,2)+R4*P(1,3)-R12*P(1,4)-R12*P(3,1)
+       +R4*P(3,2)-R4*P(3,3)+R12*P(3,4)
  C(10)=-R4*P(1,1)+R2*P(1,2)-R4*P(1,3)+R4*P(3,1)-R2*P(3,2)
+        +R4*P(3,3)
  C(11)=R4*P(1,1)-R4*P(1,3)-R4*P(3,1)+R4*P(3,3)
  C(12)=-R12*P(1,1)-R3*P(1,2)-R12*P(1,3)+R12*P(3,1)+R3*P(3,2)
+        +R12*P(3,3)
  C(13)=-R36*P(1,1)+R12*P(1,2)-R12*P(1,3)+R36*P(1,4)-R9*P(2,1)
+        +R3*P(2,2)-R3*P(2,3)+R9*P(2,4)-R36*P(3,1)+R12*P(3,2)
+        -R12*P(3,3)+R36*P(3,4)
  C(14)=R12*P(1,1)-R6*P(1,2)+R12*P(1,3)+R3*P(2,1)-(2./3.)*P(2,2)
+        +R3*P(2,3)+R12*P(3,1)-R6*P(3,2)+R12*P(3,3)
  C(15)=-R12*P(1,1)+R12*P(1,3)-R3*P(2,1)+R3*P(2,3)-R12*P(3,1)
+        +R12*P(3,3)
  C(16)=R36*P(1,1)+R9*P(1,2)+R36*P(1,3)+R9*P(2,1)+(4./9.)*P(2,2)
+        +R9*P(2,3)+R36*P(3,1)+R9*P(3,2)+R36*P(3,3)


C STORE THE COEFF INTO TXYZ (SPECIAL CASE: JUST USE THE FIRST
C FOUR ELEMENTS IN THE ARRAY)
        IF((ABS(C(13)).LT.T).AND.(ABS(C(14)).LT.T).AND.
+          (ABS(C(15)).LT.T)) THEN
             IUWSR(I)=0
             TXYZ(I,1)=C(4)
             TXYZ(I,2)=C(8)
             TXYZ(I,3)=C(12)
             TXYZ(I,4)=C(16)
            ELSE
             IUWSR(I)=1
             TXYZ(I,1)=C(13)
             TXYZ(I,2)=C(14)
             TXYZ(I,3)=C(15)
             TXYZ(I,4)=C(16)
        ENDIF


 10   CONTINUE

      RETURN
      END
C-------------------------------------------------
C
C     SUBROUTINE: EXPKL
C
C     DESCRIPTION: SWAP THE CONTROL HULL MATRICES
C
C INPUT:
C     PKL1,PKL2          = SURFACES
C
C     C. K. WONG
C     2/12/90
```

```
C
C

      SUBROUTINE EXPKL(PKL1,PKL2)

      REAL PKL(0:3,0:3,3),PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)


C SWAP THE MATRIX

      DO 10 I=1,3
         DO 20 J=1,4
              DO 30 K=1,4
                   PKL(J-1,K-1,I)=PKL2(J-1,K-1,I)
                   PKL2(J-1,K-1,I)=PKL1(J-1,K-1,I)
                   PKL1(J-1,K-1,I)=PKL(J-1,K-1,I)
 30           CONTINUE
 20      CONTINUE
 10   CONTINUE

      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: SR1
C
C     DESCRIPTION: DEFINE THE REPEATED VARIABLE ON
C                  SPECIAL SURFACE 1 IN THE INTERSECTION
C                  EQS.
C
C INPUT:
C     U1           = PARAMETRIC VALUE
C     BRC1,BRC2    = SURFACES COEFF.
C     PKL1,PKL2    = SURFACES
C     IUW          = LINEAR OR CUBIC VARIABLE FLAG
C     IUWSR        = SINGULAR OR REPEATED VARIABLE FLAGS
C
C OUTPUT:
C     TUW          = INTERSECTION POINT IN PARAMETRIC SPACE
C
C     C. K. WONG
C     2/12/90
C

      SUBROUTINE SR1(U1,BRC1,BRC2,PKL1,PKL2,IUW,IUWSR,TUW)

      REAL BRC1(3,8),BRC2(3,8)
     +      ,TUW(4)
     +      ,R1(4),R2(4),R3(4),RO(3)
     +      ,COF1(7),COF2(7),COEFF(7)
     +      ,EQ1A(4),EQ1B(4),EQ2A(4),EQ2B(4),EQ3A(4),EQ3B(4)
     +      ,UA1(60),WA1(60),UA2(60),WA2(60)
     +      ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER IUW(2),IUWSR(3),NDEG,M

      COMPLEX  ZERO(6)

      EXTERNAL ZPLRC

C SET THE TOLERANCE
      TOL=.001


C FIND OUT WHICH VARIABLE IS REPEATED

      IF(IUWSR(1).EQ.IUWSR(2)) THEN
```

```
          IRO=IUWSR(1)
          IR1=1
          IR2=2
          IR3=3
        ELSEIF(IUWSR(1).EQ.IUWSR(3)) THEN
          IRO=IUWSR(1)
          IR1=1
          IR2=3
          IR3=2
        ELSEIF(IUWSR(2).EQ.IUWSR(3)) THEN
          IRO=IUWSR(2)
          IR1=2
          IR2=3
          IR3=1
       ENDIF

C ARRANGE THE EQUATIONS: START WITH THE TWO REPEATED EQUATIONS

       DO 10 I=1,4
          R1(I)=BRC1(IR1,I)
          R2(I)=BRC1(IR2,I)
          R3(I)=BRC1(IR3,I)
          EQ1A(I)=BRC2(IR1,I)
          EQ1B(I)=BRC2(IR1,I+4)
          EQ2A(I)=BRC2(IR2,I)
          EQ2B(I)=BRC2(IR2,I+4)
          EQ3A(I)=BRC2(IR3,I)
          EQ3B(I)=BRC2(IR3,I+4)
 10    CONTINUE

C EVALUATE THE REPEATED EQUATIONS
       CALL EVAFUN(R1,U1,F1)
       CALL EVAFUN(R2,U1,F2)

C CREATE THE EQUATION FOR THE ROOTS: A 6TH ORDER POLYNOMIAL

       CALL MULCO10(EQ1B,EQ2A,COF1)
       CALL MULCO10(EQ2B,EQ1A,COF2)

       NDEG=6
       DO 100 I=NDEG+1,5,-1
          COEFF(I)=COF1(8-I)-COF2(8-I)
 100   CONTINUE

       DO 110 I=4,1,-1
          COEFF(I)=COF1(8-I)-COF2(8-I)-(F1*EQ2A(5-I))+(F2*EQ1A(5-I))
 110   CONTINUE


C SOLVE THE POLYNOMIAL
       CALL ZPLRC1(NDEG,COEFF,ZERO)

C SET THE COUNTER OF SOLUTION
       IFOUND=1

       DO 650 M=1,NDEG

C DESIGNATE CUBIC VARIABLE  'REAL' OR 'COMPLEX'

          IF(ABS(AIMAG(ZERO(M))).GE.1.E-4) GOTO 650

          W2=REAL(ZERO(M))

C ROOTS WITHIN THE RANGE?
          IF((W2.LT.-(TOL)).OR.(W2.GT.(1.+TOL))) GOTO 650

C BACK SUBSTITUTION FOR THE LINEAR VARIABLE
```

```
            CALL EVAFUN(EQ1A,W2,E1A)
            CALL EVAFUN(EQ2A,W2,E2A)

            IF(ABS(E1A).GT.(ABS(E2A))) THEN
                 CALL EVAFUN(EQ1B,W2,E1B)
                 U2=(F1-E1B)/E1A
               ELSE
                 CALL EVAFUN(EQ2B,W2,E2B)
                 U2=(F2-E2B)/E2A
            ENDIF

C LINEAR VARIABLE WITHIN THE RANGE?
            IF((U2.GT.(1+TOL).OR.(U2.LT.-(TOL))) GOTO 650

C FIND THE SINGULAR VARIABLE

            CALL EVAFUN(EQ3A,W2,E3A)
            CALL EVAFUN(EQ3B,W2,E3B)
            F3=(U2*E3A)+E3B

            R3(4)=R3(4)-F3

            CALL CUROOT(R3,IR,RO)

C STORE THE VARAIBLES
            DO 150 KK=1,IR
            W1=RO(KK)
                 UA1(IFOUND)=U1
                 WA1(IFOUND)=W1
                 UA2(IFOUND)=U2
                 WA2(IFOUND)=W2
                 IFOUND=IFOUND+1
 150        CONTINUE


 650  CONTINUE

      IFOUND=IFOUND-1

C NO INTERSECTION
      IF(IFOUND.EQ.0) THEN

         DO 400 I=1,4
              TUW(I)=2.
 400     CONTINUE

        ELSE

C SCREEN OUT THE BAD RESULT
         IF((IRO.EQ.0).AND.(IUW(2).EQ.1)) THEN
             CALL COMRELT(PKL1,PKL2,UA1,WA1,UA2,WA2,IFOUND,TUW)
          ELSEIF((IRO.EQ.1).AND.(IUW(2).EQ.1)) THEN
             CALL COMRELT(PKL1,PKL2,WA1,UA1,UA2,WA2,IFOUND,TUW)
          ELSEIF((IRO.EQ.0).AND.(IUW(2).EQ.0)) THEN
             CALL COMRELT(PKL1,PKL2,UA1,WA1,WA2,UA2,IFOUND,TUW)
          ELSEIF((IRO.EQ.1).AND.(IUW(2).EQ.0)) THEN
             CALL COMRELT(PKL1,PKL2,WA1,UA1,WA2,UA2,IFOUND,TUW)
         ENDIF

      ENDIF


      RETURN
      END
C-------------------------------------------------
C
C     SUBROUTINE: MULCO10
```

```
C
C      DESCRIPTION: MULTIPLICATION OF TWO FUNCTIONS (3RD ORDER)
C                   TO FORM A 6TH ORDER FUNCTION
C
C INPUT:
C      COFI1,COFI2  = 3RD ORDER FUNCTIONS
C
C OUTPUT:
C      OCOF         = 6TH ORDER FUNCTION
C
C      C. K. WONG
C      9/12/89
C
C

      SUBROUTINE MULCO10(COFI1,COFI2,OCOF)

      REAL COFI1(4),COFI2(4),OCOF(7)

C INITIALIZATION
      Y6=0.
      Y5=0.
      Y4=0.
      Y3=0.
      Y2=0.
      Y1=0.
      Y0=0.

      DO 10 I=4,1,-1
         DO 20 J=4,1,-1
              YY=0.
              YY=COFI1(5-I)*COFI2(5-J)
              IF((I+J-2).EQ.6) THEN
                  Y6=Y6+YY
               ELSEIF((I+J-2).EQ.5) THEN
                  Y5=Y5+YY
               ELSEIF((I+J-2).EQ.4) THEN
                  Y4=Y4+YY
               ELSEIF((I+J-2).EQ.3) THEN
                  Y3=Y3+YY
               ELSEIF((I+J-2).EQ.2) THEN
                  Y2=Y2+YY
               ELSEIF((I+J-2).EQ.1) THEN
                  Y1=Y1+YY
               ELSEIF((I+J-2).EQ.0) THEN
                  Y0=Y0+YY
              ENDIF
 20      CONTINUE
 10   CONTINUE

      OCOF(1)=Y6
      OCOF(2)=Y5
      OCOF(3)=Y4
      OCOF(4)=Y3
      OCOF(5)=Y2
      OCOF(6)=Y1
      OCOF(7)=Y0

      RETURN
      END
C---------------------------------------------------
C
C      SUBROUTINE: SR2
C
C      DESCRIPTION: DEFINE THE SINGULAR VARIABLE ON SPECIAL
C                   SURFACE 1 IN THE INTERSECTION EQS.
C
```

```
C INPUT:
C       V               = PARAMETRIC VALUE
C       BRC1,BRC2       = SURFACES COEFF.
C       PKL1,PKL2       = SURFACES
C       IUW             = LINEAR OR CUBIC VARIABLE FLAG
C       IUWSR           = SINGULAR OR REPEATED VARIABLE FLAG
C
C OUTPUT:
C       TUW             = INTERSECTION POINT IN PARAMETRIC SPACE
C
C       C. K. WONG
C       2/12/90
C

      SUBROUTINE SR2(V,BRC1,BRC2,PKL1,PKL2,IUW,IUWSR,TUW)

      REAL BRC1(3,8),BRC2(3,8),
     +     TUW(4),EQ1(19),EQ2(19),
     +     XACOF(4),XBCOF(4),YACOF(4),YBCOF(4),ZACOF(4),ZBCOF(4),
     +     X3COF(4),Y3COF(4),Z3COF(4)
     +     ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER IUW(2),IUWSR(3)

C FIND OUT WHICH VARIABLE IS REPEATED

      IF(IUWSR(1).EQ.IUWSR(2)) THEN
         IRO=IUWSR(1)
         INR=3
        ELSEIF(IUWSR(1).EQ.IUWSR(3)) THEN
         IRO=IUWSR(1)
         INR=2
        ELSEIF(IUWSR(2).EQ.IUWSR(3)) THEN
         IRO=IUWSR(2)
         INR=1
      ENDIF

C ASSIGN IUW(1) ACCORDINGLY
      IUW(1)=IRO

C REWRITE THE FIRST SURFACE COEFF
      DO 5 JJ=1,4
            X3COF(JJ)=BRC1(1,JJ)
            Y3COF(JJ)=BRC1(2,JJ)
            Z3COF(JJ)=BRC1(3,JJ)
 5    CONTINUE

C SET UP THE EQUATIONS

      IF(INR.EQ.1) THEN
         CALL EVAFUN(X3COF,V,FF)
         X3COF(1)=0.
         X3COF(2)=0.
         X3COF(3)=0.
         X3COF(4)=FF
       ELSEIF(INR.EQ.2) THEN
         CALL EVAFUN(Y3COF,V,FF)
         Y3COF(1)=0.
         Y3COF(2)=0.
         Y3COF(3)=0.
         Y3COF(4)=FF
       ELSEIF(INR.EQ.3) THEN
         CALL EVAFUN(Z3COF,V,FF)
         Z3COF(1)=0.
         Z3COF(2)=0.
         Z3COF(3)=0.
         Z3COF(4)=FF
```

```
      ENDIF

C SET UP THE REST OF THE EQUATIONS FOR BRPCREQ

      DO 35 JJ=1,4
         XACOF(JJ)=BRC2(1,JJ)
         XBCOF(JJ)=BRC2(1,JJ+4)
         YACOF(JJ)=BRC2(2,JJ)
         YBCOF(JJ)=BRC2(2,JJ+4)
         ZACOF(JJ)=BRC2(3,JJ)
         ZBCOF(JJ)=BRC2(3,JJ+4)
 35   CONTINUE


C CREATE THE EQUATIONS EQ1 & EQ2 FOR BRPSLVA

      CALL BRPCREQ(X3COF,Y3COF,Z3COF,XACOF,XBCOF,
     +             YACOF,YBCOF,ZACOF,ZBCOF,EQ1,EQ2)

C SOLVE THE VARIABLES

      CALL BRPSLVA(PKL1,PKL2,EQ1,EQ2,X3COF,Y3COF,Z3COF,XACOF,XBCOF,
     +             YACOF,YBCOF,ZACOF,ZBCOF,V,IUW,TUW)


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: SR2D
C
C     DESCRIPTION: DEFINE THE SINGULAR VARIABLE ON SPECIAL
C                  SURFACE 1 IN THE INTERSECTION EQS.
C
C INPUT:
C     V            = PARAMETRIC VALUE
C     BRC1,BRC2    = SURFACES COEFF.
C     PKL1,PKL2    = SURFACES
C     IUW          = LINEAR OR CUBIC VARIABLE FLAG
C     IUWSR        = SINGULAR OR REPEATED VARIABLE FLAG
C
C OUTPUT:
C     TUW          = INTERSECTION POINT IN PARAMETRIC SPACE
C
C     C. K. WONG
C     2/12/90
C

      SUBROUTINE SR2D(V,BRC1,BRC2,PKL1,PKL2,IUW,IUWSR,TUW)

      REAL BRC1(3,8),BRC2(3,8),
     +     TUW(4),EQ1(19),EQ2(19),
     +     XACOF(4),XBCOF(4),YACOF(4),YBCOF(4),ZACOF(4),ZBCOF(4),
     +     X3COF(4),Y3COF(4),Z3COF(4)
     +     ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER IUW(2),IUWSR(3)

C FIND OUT WHICH VARIABLE IS REPEATED

      IF(IUWSR(1).EQ.IUWSR(2)) THEN
         IRO=IUWSR(1)
         INR=3
        ELSEIF(IUWSR(1).EQ.IUWSR(3)) THEN
         IRO=IUWSR(1)
         INR=2
        ELSEIF(IUWSR(2).EQ.IUWSR(3)) THEN
```

```
         IRO=IUWSR(2)
         INR=1
      ENDIF

C ASSIGN IUW(1) ACCORDINGLY
      IUW(1)=IRO

C REWRITE THE FIRST SURFACE COEFF
      DO 5 JJ=1,4
             X3COF(JJ)=BRC1(1,JJ)
             Y3COF(JJ)=BRC1(2,JJ)
             Z3COF(JJ)=BRC1(3,JJ)
 5    CONTINUE

C SET UP THE EQUATIONS

      IF(INR.EQ.1) THEN
         CALL EVAFUN(X3COF,V,FF)
         X3COF(1)=0.
         X3COF(2)=0.
         X3COF(3)=0.
         X3COF(4)=FF
       ELSEIF(INR.EQ.2) THEN
         CALL EVAFUN(Y3COF,V,FF)
         Y3COF(1)=0.
         Y3COF(2)=0.
         Y3COF(3)=0.
         Y3COF(4)=FF
       ELSEIF(INR.EQ.3) THEN
         CALL EVAFUN(Z3COF,V,FF)
         Z3COF(1)=0.
         Z3COF(2)=0.
         Z3COF(3)=0.
         Z3COF(4)=FF
      ENDIF

C SET UP THE REST OF THE EQUATIONS FOR BRPCREQ

      DO 35 JJ=1,4
         XACOF(JJ)=BRC2(1,JJ)
         XBCOF(JJ)=BRC2(1,JJ+4)
         YACOF(JJ)=BRC2(2,JJ)
         YBCOF(JJ)=BRC2(2,JJ+4)
         ZACOF(JJ)=BRC2(3,JJ)
         ZBCOF(JJ)=BRC2(3,JJ+4)
 35   CONTINUE


C CREATE THE EQUATIONS EQ1 & EQ2 FOR BRPSLVA

      CALL BRPCREQ(X3COF,Y3COF,Z3COF,XACOF,XBCOF,
     +             YACOF,YBCOF,ZACOF,ZBCOF,EQ1,EQ2)

C SOLVE THE VARIABLES

      CALL BRPSLVD(EQ1,EQ2,X3COF,Y3COF,Z3COF,XACOF,XBCOF,
     +             YACOF,YBCOF,ZACOF,ZBCOF,V,IUW,TUW)


      RETURN
      END
C-----------------------------------------------
C
C     SUBROUTINE: SR3
C
C     DESCRIPTION: DEFINE THE CUBIC VARIABLE ON SURFACE 2
C
```

```
C INPUT:
C      W2            = PARAMETRIC VALUE
C      BRC1,BRC2     = SURFACES COEFF.
C      PKL1,PKL2     = SURFACES
C      IUW           = LINEAR OR CUBIC VARIABLE FLAG
C      IUWSR         = SINGULAR OR REPEATED VARIABLE FLAG
C
C OUTPUT:
C      TUW           = INTERSECTION POINT IN PARAMETRIC SPACE
C
C      C. K. WONG
C      2/12/90
C

      SUBROUTINE SR3(W2,BRC1,BRC2,PKL1,PKL2,IUW,IUWSR,TUW)

      REAL BRC1(3,8),BRC2(3,8)
     +     ,TUW(4)
     +     ,R1(4),R2(4),R3(4)
     +     ,COF1(4),COF2(4),COF(4)
     +     ,EQ1A(4),EQ1B(4),EQ2A(4),EQ2B(4),EQ3A(4),EQ3B(4)
     +     ,RO(3),RO1(3)
     +     ,UA1(60),WA1(60),UA2(60),WA2(60)
     +     ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER IUW(2),IUWSR(3),NDEG,M

C INITIALIZE THE TOLERANCE
      TOL2=0.0001

C FIND OUT THE REPEATED VARIABLE ON SURFACE 1

      IF(IUWSR(1).EQ.IUWSR(2)) THEN
         IRO=IUWSR(1)
         IR1=1
         IR2=2
         IR3=3
       ELSEIF(IUWSR(1).EQ.IUWSR(3)) THEN
         IRO=IUWSR(1)
         IR1=1
         IR2=3
         IR3=2
       ELSEIF(IUWSR(2).EQ.IUWSR(3)) THEN
         IRO=IUWSR(2)
         IR1=2
         IR2=3
         IR3=1
      ENDIF

C ARRANGE THE EQUATIONS: START WITH THE TWO REPEATED EQUATIONS

      DO 10 I=1,4
         R1(I)=BRC1(IR1,I)
         R2(I)=BRC1(IR2,I)
         R3(I)=BRC1(IR3,I)
         EQ1A(I)=BRC2(IR1,I)
         EQ1B(I)=BRC2(IR1,I+4)
         EQ2A(I)=BRC2(IR2,I)
         EQ2B(I)=BRC2(IR2,I+4)
         EQ3A(I)=BRC2(IR3,I)
         EQ3B(I)=BRC2(IR3,I+4)
 10   CONTINUE

C EVALUATE THE REPEATED EQUATIONS
      CALL EVAFUN(EQ1A,W2,E1A)
      CALL EVAFUN(EQ1B,W2,E1B)
      CALL EVAFUN(EQ2A,W2,E2A)
```

```
      CALL EVAFUN(EQ2B,W2,E2B)

C BOTH DENOMINATERS =0 : NO INTERSECTION
      TOL=.1E-15
      IF((ABS(E1A).LT.TOL).AND.(ABS(E2A).LT.TOL)) THEN
         IFOUND=0
         WRITE(6,*)'E1A& E2A ---> 0.  IN SR3 '
         GOTO 555
      ENDIF


C FIRST DENOMINATOR = 0

      IF(ABS(E1A).LT.TOL) THEN
         DO 100 I=1,4
              COF(I)=R1(I)
 100     CONTINUE
         COF(4)=COF(4)-E1B

C SECOND DENOMINATER =0
       ELSEIF(ABS(E2A).LT.TOL) THEN
         DO 110 I=1,4
              COF(I)=R2(I)
 110     CONTINUE
         COF(4)=COF(4)-E2B

       ELSE
         DO 150 I=1,4
              COF1(I)=R1(I)/E1A
              COF2(I)=R2(I)/E2A
 150     CONTINUE
         COF1(4)=COF1(4)-(E1B/E1A)
         COF2(4)=COF2(4)-(E2B/E2A)

         DO 160 I=1,4
              COF(I)=COF1(I)-COF2(I)
 160     CONTINUE

      ENDIF

C SOLVE FOR U1

      CALL CUROOT(COF,IR,RO)

      IFOUND=1
      DO 200 J=1,IR

              U1=RO(J)

C SOLVE FOR U2

              IF(ABS(E1A).GT.(ABS(E2A))) THEN
                   CALL EVAFUN(R1,U1,F1)
                   U2=(F1-E1B)/E1A
                 ELSE
                   CALL EVAFUN(R2,U1,F2)
                   U2=(F2-E2B)/E2A
              ENDIF

C IF U2 OUT OF RANGE?
              IF((U2.GT.(1.+TOL2)).OR.(U2.LT.-(TOL2))) GOTO 200

C FIND THE LAST VARIABLE

              CALL EVAFUN(EQ3A,W2,E3A)
              CALL EVAFUN(EQ3B,W2,E3B)
              F3=(U2*E3A)+E3B
```

```
               R3(4)=R3(4)-F3

               CALL CUROOT(R3,IRR,RO1)

          DO 300 K=1,IRR
               W1=RO1(K)
               UA1(IFOUND)=U1
               WA1(IFOUND)=W1
               UA2(IFOUND)=U2
               WA2(IFOUND)=W2
               IFOUND=IFOUND+1
 300      CONTINUE


 200   CONTINUE

               IFOUND=IFOUND-1

C STORE THE RESULT

 555   IF(IFOUND.EQ.0) THEN

          DO 400 I=1,4
               TUW(I)=2.
 400      CONTINUE

         ELSE

C FIND THE PAIR OF POINTS WITH THE SMALLEST
C DISTANCE AS THE RESULT

          IF((IRO.EQ.0).AND.(IUW(2).EQ.1)) THEN
              CALL COMRELT(PKL1,PKL2,UA1,WA1,UA2,WA2,IFOUND,TUW)
           ELSEIF((IRO.EQ.1).AND.(IUW(2).EQ.1)) THEN
              CALL COMRELT(PKL1,PKL2,WA1,UA1,UA2,WA2,IFOUND,TUW)
           ELSEIF((IRO.EQ.0).AND.(IUW(2).EQ.0)) THEN
              CALL COMRELT(PKL1,PKL2,UA1,WA1,WA2,UA2,IFOUND,TUW)
           ELSEIF((IRO.EQ.1).AND.(IUW(2).EQ.0)) THEN
              CALL COMRELT(PKL1,PKL2,WA1,UA1,WA2,UA2,IFOUND,TUW)
          ENDIF

      ENDIF


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: SR4
C
C     DESCRIPTION: DEFINE THE LINEAR VARIABLE IN RULED SURFACE 2
C
C INPUT:
C     U2            = PARAMETRIC VALUE
C     BRC1,BRC2     = SURFACES COEFF.
C     PKL1,PKL2     = SURFACES
C     IUW           = LINEAR OR CUBIC VARIABLE FLAG
C     IUWSR         = SINGULAR OR REPEATED VARIABLE FLAG
C
C OUTPUT:
C     TUW           = INTERSECTION POINT IN PARAMETRIC SPACE
C
C     C. K. WONG
C     2/12/90
C
```

```
      SUBROUTINE SR4(U2,BRC1,BRC2,PKL1,PKL2,IUW,IUWSR,TUW)

      REAL BRC1(3,8),BRC2(3,8)
     +     ,TUW(4)
     +     ,R1(4),R2(4),R3(4)
     +     ,COF1(8),COF2(8),COEFF(13)
     +     ,EQ1A(4),EQ1B(4),EQ2A(4),EQ2B(4),EQ3A(4),EQ3B(4)
     +     ,RO(3),RO1(3)
     +     ,UA1(60),WA1(60),UA2(60),WA2(60)
     +     ,PKL1(0:3,0:3,3), PKL2(0:3,0:3,3)

      INTEGER IUW(2),IUWSR(3),NDEG,M

      COMPLEX  ZERO(12)

      EXTERNAL WRCRN,ZPLRC

C INITIALIZE THE TOLERANCE
      TOL2=.0001

C FIND OUT WHICH VARIABLE IS REPEATED

      IF(IUWSR(1).EQ.IUWSR(2)) THEN
         IRO=IUWSR(1)
         IR1=1
         IR2=2
         IR3=3
       ELSEIF(IUWSR(1).EQ.IUWSR(3)) THEN
         IRO=IUWSR(1)
         IR1=1
         IR2=3
         IR3=2
       ELSEIF(IUWSR(2).EQ.IUWSR(3)) THEN
         IRO=IUWSR(2)
         IR1=2
         IR2=3
         IR3=1
      ENDIF

C ARRANGE THE EQUATIONS: START WITH THE TWO REPEATED EQUATIONS

      DO 10 I=1,4
         R1(I)=BRC1(IR1,I)
         R2(I)=BRC1(IR2,I)
         R3(I)=BRC1(IR3,I)
         EQ1A(I)=BRC2(IR1,I)
         EQ1B(I)=BRC2(IR1,I+4)
         EQ2A(I)=BRC2(IR2,I)
         EQ2B(I)=BRC2(IR2,I+4)
         EQ3A(I)=BRC2(IR3,I)
         EQ3B(I)=BRC2(IR3,I+4)
 10   CONTINUE

C CREATE THE EQUATION FOR THE ROOTS

      DO 20 I=1,4
         COF1(I)=R1(I)
         COF1(I+4)=U2*EQ1A(I)+EQ1B(I)
         COF2(I)=R2(I)
         COF2(I+4)=U2*EQ2A(I)+EQ2B(I)
 20   CONTINUE

C ELIMINATE ONE VARIABLE FROM TWO EQUATIONS

      CALL SRTWO(COF1,COF2,COEFF)
```

```
C SOLVE FOR THE ROOTS : W2
      NDEG=12
      CALL ZPLRC1(NDEG,COEFF,ZERO)

      IFOUND=1
      DO 650 M=1,NDEG

C DESIGNATE SOLUTION 'REAL' OR 'COMPLEX'

         IF(ABS(AIMAG(ZERO(M))).GE.1.E-4) GOTO 650


         W2=REAL(ZERO(M))

C IF W2 OUT OF RANGE ?
         IF((W2.LT.-(TOL2)).OR.(W2.GT.(1.+TOL2))) GOTO 650

C SOLVE FOR U1
              IF(ABS(R1(1)).GT.ABS(R2(1))) THEN
                   CALL EVAFUN(EQ1A,W2,E1A)
                   CALL EVAFUN(EQ1B,W2,E1B)
                   R1(4)=R1(4)-((U2*E1A)+E1B)
                   CALL CUROOT(R1,IRR,RO1)
                 ELSE
                   CALL EVAFUN(EQ2A,W2,E2A)
                   CALL EVAFUN(EQ2B,W2,E2B)
                   R2(4)=R2(4)-((U2*E2A)+E2B)
                   CALL CUROOT(R2,IRR,RO1)
              ENDIF

         DO 200 JJ=1,IRR

              U1=RO1(JJ)

C FIND THE LAST VARIABLE (W1)

              CALL EVAFUN(EQ3A,W2,E3A)
              CALL EVAFUN(EQ3B,W2,E3B)
              F3=(U2*E3A)+E3B

              R3(4)=R3(4)-F3

              CALL CUROOT(R3,IR,RO)

              DO 210 KK=1,IR
                 W1=RO(KK)
                 UA1(IFOUND)=U1
                 WA1(IFOUND)=W1
                 UA2(IFOUND)=U2
                 WA2(IFOUND)=W2
                 IFOUND=IFOUND+1
 210          CONTINUE

 200     CONTINUE


 650  CONTINUE


      IFOUND=IFOUND-1

      IF(IFOUND.EQ.0) THEN

C NO INTERSECTION
         DO 400 I=1,4
              TUW(I)=2.
 400     CONTINUE
```

```
        ELSE

C FIND THE PAIR OF POINTS HAS THE SMALLEST DISTANCE
C AS THE RESULT

          IF((IRO.EQ.0).AND.(IUW(2).EQ.1)) THEN
              CALL COMRELT(PKL1,PKL2,UA1,WA1,UA2,WA2,IFOUND,TUW)
           ELSEIF((IRO.EQ.1).AND.(IUW(2).EQ.1)) THEN
              CALL COMRELT(PKL1,PKL2,WA1,UA1,UA2,WA2,IFOUND,TUW)
           ELSEIF((IRO.EQ.0).AND.(IUW(2).EQ.0)) THEN
              CALL COMRELT(PKL1,PKL2,UA1,WA1,WA2,UA2,IFOUND,TUW)
           ELSEIF((IRO.EQ.1).AND.(IUW(2).EQ.0)) THEN
              CALL COMRELT(PKL1,PKL2,WA1,UA1,WA2,UA2,IFOUND,TUW)
          ENDIF

      ENDIF


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: SRTWO
C
C     DESCRIPTION: DERIVE A 12TH ORDER POLYNOMIAL BY ELIMINATING
C                  ONE VARIABLE FROM TWO EQUATIONS WITH TWO VARIABLES
C
C INPUT:
C     EQ1,EQ2      = TWO EQS.
C
C OUTPUT:
C     EQ3          = 12TH ORDER POLYNOMIAL
C
C     C. K. WONG
C     2/12/90
C
C

      SUBROUTINE SRTWO(EQ1,EQ2,EQ3)


      REAL EQ1(8),EQ2(8), EQ3(13)
     +    ,A,B,C,D,E,F,G


C INITIAL EQUATIONS

      A1=EQ1(1)
      A2=EQ1(2)
      A3=EQ1(3)
      A4=EQ1(4)
      A5=EQ1(5)
      A6=EQ1(6)
      A7=EQ1(7)
      A8=EQ1(8)

      A1P=EQ2(1)
      A2P=EQ2(2)
      A3P=EQ2(3)
      A4P=EQ2(4)
      A5P=EQ2(5)
      A6P=EQ2(6)
      A7P=EQ2(7)
      A8P=EQ2(8)
```

```
C EQUATION 3
      A=A2*A1P-A2P*A1
      B=A3*A1P-A3P*A1
      C=A4*A1P-A4P*A1
      D=A5P*A1-A5*A1P
      E=A6P*A1-A6*A1P
      F=A7P*A1-A7*A1P
      G=A8P*A1-A8*A1P

C EQUATION 4
      T1=A1*A5P-A1P*A5
      T2=A1*A6P-A1P*A6
      T3=A1*A7P-A1P*A7
      T4=(A1*A8P-A1P*A8)+(A1P*A4-A1*A4P)
      T5=A2*A5P-A2P*A5
      T6=A2*A6P-A2P*A6
      T7=A2*A7P-A2P*A7
      T8=(A2*A8P-A2P*A8)+(A2P*A4-A2*A4P)
      T9=A3*A5P-A3P*A5
      T10=A3*A6P-A3P*A6
      T11=A3*A7P-A3P*A7
      T12=(A3*A8P-A3P*A8)+(A3P*A4-A3*A4P)

C EQUATION 5
      S1=B*T1-A*T5
      S2=B*T2-A*T6
      S3=B*T3-A*T7
      S4=B*T4-A*T8
      S5=D*T1
      S6=D*T2+E*T1
      S7=D*T3+E*T2+F*T1
      S8=C*T1+D*T4+E*T3+F*T2+G*T1-A*T9
      S9=C*T2+E*T4+F*T3+G*T2-A*T10
      S10=C*T3+F*T4+G*T3-A*T11
      S11=C*T4+G*T4-A*T12

C EQUATION 6
      V1=T1*D
      V2=T1*E+T2*D
      V3=T1*F+T2*E+T3*D
      V4=T1*C+T1*G+T2*F+T3*E+T4*D-A*T9
      V5=T2*C+T2*G+T3*F+T4*E-A*T10
      V6=T3*C+T3*G+T4*F-A*T11
      V7=T4*C+T4*G-A*T12
      V8=T5*D
      V9=T5*E+T6*D
      V10=T5*F+T6*E+T7*D
      V11=T5*C+T5*G+T6*F+T7*E+T8*D-B*T9
      V12=T6*C+T6*G+T7*F+T8*E-B*T10
      V13=T7*C+T7*G+T8*F-B*T11
      V14=T8*C+T8*G-B*T12

C FINAL EQUATION :12TH ORDER POLYNOMIAL
      EQ3(13)=S5*V1
      EQ3(12)=S5*V2+S6*V1
      EQ3(11)=S5*V3+S6*V2+S7*V1
      EQ3(10)=S5*V4+S6*V3+S7*V2+S8*V1-S1*V8
       EQ3(9)=S5*V5+S6*V4+S7*V3+S8*V2+S9*V1-S1*V9-S2*V8
       EQ3(8)=S5*V6+S6*V5+S7*V4+S8*V3+S9*V2+S10*V1
     +        -S1*V10-S2*V9-S3*V8
       EQ3(7)=S5*V7+S6*V6+S7*V5+S8*V4+S9*V3+S10*V2+S11*V1
     +        -S1*V11-S2*V10-S3*V9-S4*V8
       EQ3(6)=S6*V7+S7*V6+S8*V5+S9*V4+S10*V3+S11*V2
     +        -S1*V12-S2*V11-S3*V10-S4*V9
       EQ3(5)=S7*V7+S8*V6+S9*V5+S10*V4+S11*V3
     +        -S1*V13-S2*V12-S3*V11-S4*V10
       EQ3(4)=S8*V7+S9*V6+S10*V5+S11*V4
```

```
     +        -S1*V14-S2*V13-S3*V12-S4*V11
       EQ3(3)=S9*V7+S10*V6+S11*V5
     +        -S2*V14-S3*V13-S4*V12
       EQ3(2)=S10*V7+S11*V6
     +        -S3*V14-S4*V13
       EQ3(1)=S11*V7-S4*V14

      RETURN
      END
C-----------------------------------------------
C
C     SUBROUTINE: PT4SR
C
C     DESCRIPTION: FIND FOUR POINTS(EQUAL INTERVAL) ON
C                  THE INTERSECTION CURVE: GENERAL RULED
C                  AND SPECIAL RULED SURFACES
C
C INPUT:
C     TXYZ1,TXYZ2  = SURFACES COEFF.
C     PKL1,PKL2    = SURFACES
C     IUWSR        = SINGULAR OR REPEATED VARIABLE FLAG
C     IUW          = LINEAR OR REPEATED VARIABLE FLAG
C     IR1,IR2      = ROW IDENTIFIERS FOR MAX DIFF. IN
C                    PARAMETRIC VALUE
C     IC           = COLUMN IDENTIFIER FOR MAX DIFF. IN
C                    PARAMETRIC VALUE
C     TTUW         = INTERSECTION POINTS ALONG THE EDGE
C                    IN PARAMETRIC SPACE
C
C OUTPUT:
C     TTUWG        = A SEGMENT OF THE INTERSECTION CURVE
C                    (FOUR POINTS)
C     I2           = BAD CASE FLAG: COULDN'T FIND THE
C                    INTERMEDIATE POINTS
C
C     C. K. WONG
C     2/12/90
C

      SUBROUTINE PT4SR(TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,IR1,IR2,IC,
     +                 TTUW,TTUWG,I2)

      REAL TXYZ1(3,8),TXYZ2(3,8),TT5(2,4),TUW(4),TUW2(4)
     +  ,TTUW(8,4),TTUWG(4,3),TT(2),TPST(1,3)
     +  ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER IUW(2),IUW2(2),IUWSR(3)

C INITIALIZE THE FLAG AND THE TOLERANCE
      TOL2=.0001
      I2=1

C FIND OUT WHICH VARIABLE IS REPEATED

      IF(IUWSR(1).EQ.IUWSR(2)) THEN
         IRO=IUWSR(1)
       ELSEIF(IUWSR(1).EQ.IUWSR(3)) THEN
         IRO=IUWSR(1)
       ELSEIF(IUWSR(2).EQ.IUWSR(3)) THEN
         IRO=IUWSR(2)
      ENDIF

C
C GET THE MIDDLE TWO POINTS
C

      TT(1)=(1./3.)*TTUW(IR2,IC)+(2./3.)*TTUW(IR1,IC)
```

```
      TT(2)=(2./3.)*TTUW(IR2,IC)+(1./3.)*TTUW(IR1,IC)

      DO 200 I=1,2
         V=TT(I)

C CHECK WHICH ROUTINE TO CALL
         IF((IC.EQ.1).OR.(IC.EQ.2)) THEN

C INPUT IN SURFACE 1
              IF((IC.EQ.1).AND.(IRO.EQ.0)) THEN
                   CALL SR1(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
                ELSEIF ((IC.EQ.1).AND.(IRO.EQ.1)) THEN
                   CALL SR2(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
                   IF(TUW(1).GT.1.5) THEN
                      CALL SR2D(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
                   ENDIF

                ELSEIF ((IC.EQ.2).AND.(IRO.EQ.1)) THEN
                   CALL SR1(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
                ELSEIF ((IC.EQ.2).AND.(IRO.EQ.0)) THEN
                   CALL SR2(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
                   IF(TUW(1).GT.1.5) THEN
                      CALL SR2D(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
                   ENDIF

              ENDIF

            ELSE


              IF((IC.EQ.3).AND.(IUW(2).EQ.0)) THEN
                   CALL SR3(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
                ELSEIF ((IC.EQ.3).AND.(IUW(2).EQ.1)) THEN
                   CALL SR4(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)

                ELSEIF ((IC.EQ.4).AND.(IUW(2).EQ.1)) THEN
                   CALL SR3(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)
                ELSEIF ((IC.EQ.4).AND.(IUW(2).EQ.0)) THEN
                   CALL SR4(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,IUWSR,TUW)

              ENDIF

         ENDIF

C STORE THE RESULT
              DO 213 J=1,4
                   TT5(I,J)=TUW(J)
                   IF(TUW(J).GT.1.5) THEN
                      I2=0
                   ENDIF
 213          CONTINUE

 200  CONTINUE


C FIND FOUR POINTS ON THE CURVE IN GLOBAL COOR. SPACE

      CALL FBPT(PKL1,TTUW(IR1,1),TTUW(IR1,2),TPST,IFF)
      DO 500 I=1,3
         TTUWG(1,I)=TPST(1,I)
 500  CONTINUE

      CALL FBPT(PKL1,TTUW(IR2,1),TTUW(IR2,2),TPST,IFF)
      DO 510 I=1,3
         TTUWG(4,I)=TPST(1,I)
 510  CONTINUE
```

```
      DO 520 I=1,2
         CALL FBPT(PKL1,TT5(I,1),TT5(I,2),TPST,IFF)
         DO 530 J=1,3
            TTUWG(I+1,J)=TPST(1,J)
 530     CONTINUE

 520  CONTINUE


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: PARALL
C
C     DESCRIPTION: SOLVE THE INTERSECTION WHERE THE SURFACES ARE
C                  SPECIAL CASE AND PARALLEL : THE VARIABLES OF
C                  THE SURFACES ARE NOT RELATED (COUPLED) IN THE
C                  INTERSECTION  EQS.
C
C INPUT:
C     TXYZ1,TXYZ2  = SURFACES COEFF.
C     PKL1,PKL2    = SURFACES
C     IUW1,IUW2    = SINGULAR OR REPEATED VARIABLE FLAGS
C
C OUTPUT:
C     INTP         = INTERSECTION FLAG
C     TTUWG        = INTERSECTION POINTS IN GLOBAL COOR.
C
C     C. K. WONG
C     2/12/90
C
C

      SUBROUTINE PARALL(TXYZ1,TXYZ2,PKL1,PKL2,IUW1,IUW2
     +                  ,INTP,TTUWG)

      REAL TXYZ1(3,4),TXYZ2(3,4),TTUW(8,4)
     +    ,TTUWG(4,3),COF1(8),COF2(8),COEFF(13)
     +    ,COF(4),RO1(3),MIN
     +    ,UA1(20),WA1(20),UA2(20),WA2(20)
     +    ,V(2),TPST(1,3),TT5(2,4)

      INTEGER IUW1(3),IUW2(3),NDEG

      COMPLEX ZERO(12)
      LOGICAL INTP
      EXTERNAL ZPLRC


C INITIALIZE FLAG AND TOLERANCE
      INTP=.TRUE.
      TLIM=.001

C FIND THE REPEATED VARIABLE ON THE FIRST SURFACE
      IF(IUW1(2).EQ.IUW1(3)) THEN
         IR1=IUW1(2)
         IS1=1
       ELSEIF(IUW1(1).EQ.IUW1(3)) THEN
         IR1=IUW1(1)
         IS1=2
       ELSEIF(IUW1(1).EQ.IUW1(2)) THEN
         IR1=IUW1(1)
         IS1=3
      ENDIF

C FIND THE REPEATED VARIABLE ON THE SECOND SURFACE
```

```
      IF(IUW2(2).EQ.IUW2(3)) THEN
         IR2=IUW2(2)
         IS2=1
       ELSEIF(IUW2(1).EQ.IUW2(3)) THEN
         IR2=IUW2(1)
         IS2=2
       ELSEIF(IUW2(1).EQ.IUW2(2)) THEN
         IR2=IUW2(1)
         IS2=3
      ENDIF

C SOLVE FOR THE REPEATED VARIABLE FOR BOTH SURFACES :

C ARRANGE THE ORDER OF SOLVING THE EQUATIONS
      IF(IS1.EQ.1) THEN
         IO1=2
         IO2=3
         IO3=1
       ELSEIF(IS1.EQ.2) THEN
         IO1=1
         IO2=3
         IO3=2
       ELSEIF(IS1.EQ.3) THEN
         IO1=1
         IO2=2
         IO3=3
      ENDIF

C CREATE A 12TH ORDER POLYNOMIAL BY ELIMINATING
C ONE OF THE REPEATED VARIABLES FROM TWO EQS.
      DO 10 I=1,4
         COF1(I)=TXYZ1(IO1,I)
         COF1(I+4)=TXYZ2(IO1,I)
         COF2(I)=TXYZ1(IO2,I)
         COF2(I+4)=TXYZ2(IO2,I)
 10   CONTINUE

      CALL SRTWO(COF1,COF2,COEFF)

C SOLVE FOR THE ROOTS
      NDEG=12
      CALL ZPLRC1(NDEG,COEFF,ZERO)

C INITIALIZE COUNTER
      IFOUND1=1

      DO 100 I=1,NDEG

C SOLUTION: REAL OR COMPLEX
         IF(ABS(AIMAG(ZERO(I))).GE..1E-4) GOTO 100

C REPEATED VARIABLE ON THE SECOND SURFACE
         UWR2=REAL(ZERO(I))

C VARIABLE OUT OF RANGE
         IF((UWR2.LT.-(TLIM)).OR.(UWR2.GT.(1.+TLIM))) GOTO 100

C SOLVE FOR REPEATED VARIABLE ON THE FIRST SURFACE

         IF(ABS(COF1(1)).GT.ABS(COF2(1))) THEN
              IO=IO1
            ELSE
              IO=IO2
         ENDIF

         DO 110 J=1,4
              COF(J)=TXYZ2(IO,J)
```

```
 110      CONTINUE

          CALL EVAFUN(COF,UWR2,FF)
          DO 120 J=1,4
               COF(J)=TXYZ1(IO,J)
 120      CONTINUE
          COF(4)=COF(4)-FF

          CALL CUROOT(COF,IC1,RO1)

          DO 200 J=1,IC1

C REPEATED VARIABLE ON THE FIRST SURFACE
               UWR1=RO1(J)
               UA1(IFOUND1)=UWR1
               UA2(IFOUND1)=UWR2
               IFOUND1=IFOUND1+1
 200      CONTINUE
 100   CONTINUE

       IFOUND1=IFOUND1-1

C NO SOLUTION
       IF (IFOUND1.EQ.0) THEN
          INTP=.FALSE.
          RETURN
       ENDIF

C SOLVE FOR THE SINGULAR VARIABLES

       IFOUND2=1
       V(1)=0.
       V(2)=1.
       DO 210 K=1,2
          UWS=V(K)
          CALL PARALLS(UWS,IO3,TXYZ1,TXYZ2,IFOUND2,WA1,WA2)
          CALL PARALLS(UWS,IO3,TXYZ2,TXYZ1,IFOUND2,WA2,WA1)
 210   CONTINUE

       IFOUND2=IFOUND2-1

C NO SOLUTION
       IF(IFOUND2.EQ.0) THEN
          INTP=.FALSE.
          RETURN
       ENDIF

C PICK ONE SOLUTION FOR REPEATED VARIABLE USING THE
C FIRST SOLUTION OF SINGULAR VARIABLE AS REFERENCE

       CMIN=100.
       IMIN=1
       DO 300 I=1,IFOUND1

          IF((IR1.EQ.0).AND.(IR2.EQ.0)) THEN
              CALL TESTBAD(PKL1,PKL2,UA1(I),WA1(1)
     +                     ,UA2(I),WA2(1),MIN)
           ELSEIF((IR1.EQ.1).AND.(IR2.EQ.0)) THEN
              CALL TESTBAD(PKL1,PKL2,WA1(1),UA1(I)
     +                     ,UA2(I),WA2(1),MIN)
           ELSEIF((IR1.EQ.0).AND.(IR2.EQ.1)) THEN
              CALL TESTBAD(PKL1,PKL2,UA1(I),WA1(1)
     +                     ,WA2(1),UA2(I),MIN)
           ELSEIF((IR1.EQ.1).AND.(IR2.EQ.1)) THEN
              CALL TESTBAD(PKL1,PKL2,WA1(1),UA1(I)
     +                     ,WA2(1),UA2(I),MIN)
          ENDIF
```

```
         IF(MIN.LT.CMIN) THEN
              CMIN=MIN
              IMIN=I
         ENDIF

 300  CONTINUE


C COMBINE THE RESULTS IN TTUW
      DO 400 I=1,IFOUND2
         IF((IR1.EQ.0).AND.(IR2.EQ.0)) THEN
             TTUW(I,1)=UA1(IMIN)
             TTUW(I,2)=WA1(I)
             TTUW(I,3)=UA2(IMIN)
             TTUW(I,4)=WA2(I)
          ELSEIF((IR1.EQ.1).AND.(IR2.EQ.0)) THEN
             TTUW(I,2)=UA1(IMIN)
             TTUW(I,1)=WA1(I)
             TTUW(I,3)=UA2(IMIN)
             TTUW(I,4)=WA2(I)
          ELSEIF((IR1.EQ.0).AND.(IR2.EQ.1)) THEN
             TTUW(I,1)=UA1(IMIN)
             TTUW(I,2)=WA1(I)
             TTUW(I,4)=UA2(IMIN)
             TTUW(I,3)=WA2(I)
          ELSEIF((IR1.EQ.1).AND.(IR2.EQ.1)) THEN
             TTUW(I,2)=UA1(IMIN)
             TTUW(I,1)=WA1(I)
             TTUW(I,4)=UA2(IMIN)
             TTUW(I,3)=WA2(I)
         ENDIF
 400  CONTINUE

C SET THE REST OF TTUW AS UNWANTED DATA
      DO 410 I=IFOUND2+1,8
         DO 420 J=1,4
              TTUW(I,J)=2.
 420     CONTINUE
 410  CONTINUE

C GET RID OF BAD DATA
      CALL GRID(TTUW,IN)

C NO SOLUTION
      IF(IN.EQ.0) THEN
         INTP=.FALSE.
         RETURN

C BAD CASE: ONE END POINT IS MISSING
       ELSEIF(IN.EQ.1) THEN
         WRITE(6,*)'BAD CASE(PARALL) IN=1'
         INTP=.FALSE.
         RETURN
      ENDIF


C FIND THE MAX DIFF. BETWEEN THE VARIABLES
      CALL FMAX(TTUW,IN,IR1,IR2,IC)

C FIND FOUR POINTS ON THE INTERSECTION CURVE AND
C PUT THE DATA IN TTUWG

      DO 450 I=1,4
        TT5(1,I)=(1./3.)*TTUW(IR2,I)+(2./3.)*TTUW(IR1,I)
        TT5(2,I)=(2./3.)*TTUW(IR2,I)+(1./3.)*TTUW(IR1,I)
 450  CONTINUE
```

```
      CALL FBPT(PKL1,TTUW(IR1,1),TTUW(IR1,2),TPST,IFF)
      DO 500 I=1,3
         TTUWG(1,I)=TPST(1,I)
 500  CONTINUE

      CALL FBPT(PKL1,TTUW(IR2,1),TTUW(IR2,2),TPST,IFF)
      DO 510 I=1,3
         TTUWG(4,I)=TPST(1,I)
 510  CONTINUE

      DO 520 I=1,2
         CALL FBPT(PKL1,TT5(I,1),TT5(I,2),TPST,IFF)
         DO 530 J=1,3
            TTUWG(I+1,J)=TPST(1,J)
 530     CONTINUE

 520  CONTINUE



      RETURN
      END



C------------------------------------------------------------
C
C     SUBROUTINE: PARALLS
C
C     DESCRIPTION: SOLVE FOR THE SINGULAR VARIABLES (PARALLEL CASE)
C
C INPUT:
C     UWS1          = PARAMETRIC VALUE
C     IO3           = EQ. IDENTIFIER
C     TXYZ1,TXYZ2   = SURFACES COEFF.
C
C OUTPUT:
C     IFOUND2       = COUNTER FOR INTERSECTION POINTS
C     WA1,WA2       = INTERSECTION POINTS (SINGLE
C                     PARAMETRIC VARIABLE)
C
C
C     C. K. WONG
C     9/12/89

      SUBROUTINE PARALLS(UWS1,IO3,TXYZ1,TXYZ2,IFOUND2,WA1,WA2)

      REAL TXYZ1(3,4),TXYZ2(3,4)
     +    ,WA1(*),WA2(*)
     +    ,RO1(3),COF(4)


C SOLVE A CUBIC FUNCTION FOR THE SINGULAR VARIABLE
      DO 10 I=1,4
          COF(I)=TXYZ1(IO3,I)
 10   CONTINUE
      CALL EVAFUN(COF,UWS1,FF)

      DO 20 J=1,4
          COF(J)=TXYZ2(IO3,J)
 20   CONTINUE
      COF(4)=COF(4)-FF
      CALL CUROOT(COF,IC1,RO1)
```

```
      DO 300 K=1,IC1
          UWS2=RO1(K)
          WA1(IFOUND2)=UWS1
          WA2(IFOUND2)=UWS2
          IFOUND2=IFOUND2+1
300   CONTINUE

      RETURN
      END
```

Appendix I : Program BRINTRR

```
C-----------------------------------------------
C
C     SUBROUTINE: BRINTRR
C
C     DESCRIPTION: FIND THE INTERSECTION CURVE FOR TWO
C                  B-SPLINE RULED SURFACES
C
C INPUT:
C     TXYZ1,TXYZ2  = SURFACES COEFF.
C     PKL1,PKL2    = INTERSECTING SURFACES
C     IUW          = LINEAR OR CUBIC VARIABLE FLAG
C
C OUTPUT:
C     INTF         = INTERSECTION FLAG
C     TTUWG        = A SEGMENT OF THE INTERSECTION CURVE
C                    (FOUR POINTS)
C
C     C. K. WONG
C     2/12/90
C
      SUBROUTINE BRINTRR(TXYZ1,TXYZ2,PKL1,PKL2,IUW,INTF,TTUWG)

      REAL PKL1(0:3,0:3,3),PKL2(0:3,0:3,3),TXYZ1(3,8),TXYZ2(3,8),
     +     TTUW(8,4)
     +     ,TUW(4),TUW2(4),TTUWG(4,3)
     +     ,MIN

      INTEGER IUW(2),IUW2(2)
      LOGICAL INTF

C     INITIALIZE THE FLAG AND THE COUNTER

      II=0
      INTF=.FALSE.

C DEFINE VARIABLE ON SURFACE 1:
```

```
C FIRST CASE: STRAIGHT LINE AND THE RULED SURFACE

          V=0.0
          CALL BRP9(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,TUW)
          CALL PUTUW1(II,TUW,TTUW)

          V=1.0
          CALL BRP9(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,TUW)
          CALL PUTUW1(II,TUW,TTUW)

C DEFINE VARIABLE ON SURFACE 2:

C FIRST CASE: STRAIGHT LINE AND THE RULED SURFACE

          IUW2(1)=IUW(2)
          IUW2(2)=IUW(1)

          V=0.0
          CALL BRP9(V,TXYZ2,TXYZ1,PKL2,PKL1,IUW2,TUW)
          TUW2(1)=TUW(3)
          TUW2(2)=TUW(4)
          TUW2(3)=TUW(1)
          TUW2(4)=TUW(2)
          CALL PUTUW1(II,TUW2,TTUW)


          V=1.0
          CALL BRP9(V,TXYZ2,TXYZ1,PKL2,PKL1,IUW2,TUW)
          TUW2(1)=TUW(3)
          TUW2(2)=TUW(4)
          TUW2(3)=TUW(1)
          TUW2(4)=TUW(2)
          CALL PUTUW1(II,TUW2,TTUW)

C DEFINE VARIABLE ON SURFACE 1:

C SECOND CASE: CURVE AND THE RULED SURFACE


          V=0.0
          CALL BRP33A(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,TUW)
          CALL PUTUW1(II,TUW,TTUW)

          V=1.0
          CALL BRP33A(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,TUW)
          CALL PUTUW1(II,TUW,TTUW)



C DEFINE VARIABLE ON SURFACE 2:

C SECOND CASE: CURVE AND THE RULED SURFACE
          IUW2(1)=IUW(2)
          IUW2(2)=IUW(1)

          V=0.0
          CALL BRP33A(V,TXYZ2,TXYZ1,PKL2,PKL1,IUW2,TUW)

          TUW2(1)=TUW(3)
          TUW2(2)=TUW(4)
          TUW2(3)=TUW(1)
          TUW2(4)=TUW(2)
          CALL PUTUW1(II,TUW2,TTUW)

          V=1.0
          CALL BRP33A(V,TXYZ2,TXYZ1,PKL2,PKL1,IUW2,TUW)
```

```
            TUW2(1)=TUW(3)
            TUW2(2)=TUW(4)
            TUW2(3)=TUW(1)
            TUW2(4)=TUW(2)
            CALL PUTUW1(II,TUW2,TTUW)


C GET RID OF THOSE UNWANTED DATA
            CALL GRID(TTUW,IN)

C JUST ONE STARTING POINT FOR INTERSECTION: NEED
C FURTHER INVESTIGATION
            IF(IN.EQ.1) THEN

C CHECK IF THE ONLY INTERSECTION POINT IS ON THE SAME LOCATION?
               CALL TESTBAD(PKL1,PKL2,TTUW(1,1),TTUW(1,2)
     +                          ,TTUW(1,3),TTUW(1,4),MIN)

C LIMIT OF DISTANCE
               IF(MIN.GT.0.1) THEN

C DISTANCE OUT OF LIMIT: NO INTERSECTION
                    IN=0
                    RETURN
               ENDIF

C FIND THE STARTING POINT OF INTERSECTION (USING SUBROUTINE
C BISECTG FOR ROOTING ROUTINE)

               CALL BRINDRR(TXYZ1,TXYZ2,PKL1,PKL2,IUW,TTUW)
               CALL GRID(TTUW,IN)


C STILL JUST ONE STARTING POINT FOR INTERSECTION : BAD CASE
               IF(IN.EQ.1) THEN
               CALL TESTBAD(PKL1,PKL2,TTUW(1,1),TTUW(1,2)
     +                          ,TTUW(1,3),TTUW(1,4),MIN)

C THE ONLY ONE INTERSECTION POINT ISN'T ON THE SAME LOCATION:
C NO INTERSECTION

               IF(MIN.GT.0.1) THEN
                    IN=0
                    RETURN
               ENDIF

C CHECK IF THE VARIABLES ARE OUT OF RANGE?
                  DO 444 KKI=1,4
                   IF((TTUW(1,KKI).GT.1.001).OR.
     +                (TTUW(1,KKI).LT.-0.001))THEN
                        IN=0
                        RETURN
                   ENDIF
 444              CONTINUE

C ELSE; BAD CASE
                  WRITE(6,*)'BAD CASE -----BRINTRR'
                  WRITE(6,*)

C RECORD THE BAD CASE
                  CALL BADCASE(PKL1,PKL2,TTUW,IN)
                  RETURN
               ENDIF

C NO INTERSECTION
             ELSEIF(IN.EQ.0) THEN
               RETURN
```

```
           ENDIF

C
C FIND THE MAX DIFFERENCES BETWEEN THE VARIABLES.
C

           CALL FMAX(TTUW,IN,JR1,JR2,JC)

C
C FIND ANOTHER TWO INTERSECTION POINTS IN BETWEEN
C THE TWO STARTING POINTS

           CALL PT4RR(TXYZ1,TXYZ2,PKL1,PKL2,IUW,JR1,JR2,JC
     +                ,TTUW,TTUWG,I2)


C BADCASE : COULDN'T FIND THE INTERMEDIATE POINTS
      IF(I2.EQ.0) THEN
         WRITE(6,*)'BAD CURVE ---BRINTRR, I2=0'
         WRITE(6,*)

C RECORD THE BAD CASE
         CALL BADCASE(PKL1,PKL2,TTUW,IN)
         RETURN
      ENDIF

      INTF=.TRUE.


      RETURN
      END
C-----------------------------------------------------
C
C     SUBROUTINE: BRP9
C
C     DESCRIPTION: DEFINE THE CUBIC VARIABLE ON SURFACE 1
C                  IN THE ELIMINATION ROUTINE
C
C INPUT:
C     V             = PARAMETRIC VALUE
C     BRC1,BRC2     = SURFACES COEFF.
C     PKL1,PKL2     = INTERSECTING SURFACES
C     IUW           = LINEAR OR CUBIC VARIABLE FLAG
C
C OUTPUT:
C     TUW           = INTERSECTION POINT IN PARAMETRIC SPACE
C
C     C. K. WONG
C     2/12/90
C
      SUBROUTINE BRP9(V,BRC1,BRC2,PKL1,PKL2,IUW,TUW)

      REAL BRC1(3,8),BRC2(3,8),COF(4),TUW(4)
     +     ,XACOF(4),XBCOF(4),YACOF(4),YBCOF(4),ZACOF(4),ZBCOF(4)
     +     ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER IUW(2)


C ARRANGE THE EQUATIONS FOR ELIMINATION
      DO 35 JJ=1,4
         XACOF(JJ)=BRC2(1,JJ)
         XBCOF(JJ)=BRC2(1,JJ+4)
         YACOF(JJ)=BRC2(2,JJ)
         YBCOF(JJ)=BRC2(2,JJ+4)
         ZACOF(JJ)=BRC2(3,JJ)
```

```
        ZBCOF(JJ)=BRC2(3,JJ+4)
 35   CONTINUE

C DEFINE THE CUBIC VARIABLES IN THE SURFACE EQS.

      COF(1)=BRC1(1,1)
      COF(2)=BRC1(1,2)
      COF(3)=BRC1(1,3)
      COF(4)=BRC1(1,4)
      CALL EVAFUN(COF,V,X1)

      COF(1)=BRC1(1,5)
      COF(2)=BRC1(1,6)
      COF(3)=BRC1(1,7)
      COF(4)=BRC1(1,8)
      CALL EVAFUN(COF,V,X2)

      COF(1)=BRC1(2,1)
      COF(2)=BRC1(2,2)
      COF(3)=BRC1(2,3)
      COF(4)=BRC1(2,4)
      CALL EVAFUN(COF,V,Y1)


      COF(1)=BRC1(2,5)
      COF(2)=BRC1(2,6)
      COF(3)=BRC1(2,7)
      COF(4)=BRC1(2,8)
      CALL EVAFUN(COF,V,Y2)


      COF(1)=BRC1(3,1)
      COF(2)=BRC1(3,2)
      COF(3)=BRC1(3,3)
      COF(4)=BRC1(3,4)
      CALL EVAFUN(COF,V,Z1)


      COF(1)=BRC1(3,5)
      COF(2)=BRC1(3,6)
      COF(3)=BRC1(3,7)
      COF(4)=BRC1(3,8)
      CALL EVAFUN(COF,V,Z2)


C ARRANGE THE ELIMINATION EQS. SUCH THAT THE DENOMINATOR
C ALWAYS HAS THE LARGEST VALUE AMONG VALUES

      IF((ABS(X1).GT.ABS(Y1)).AND.(ABS(X1).GT.ABS(Z1))) THEN

         CALL BRP9XYZ(V,BRC1,BRC2,PKL1,PKL2,IUW,TUW
     +                ,X1,X2,Y1,Y2,Z1,Z2
     +                ,XACOF,XBCOF,YACOF,YBCOF,ZACOF,ZBCOF)

       ELSEIF((ABS(Y1).GT.ABS(X1)).AND.(ABS(Y1).GT.ABS(Z1))) THEN
         CALL BRP9XYZ(V,BRC1,BRC2,PKL1,PKL2,IUW,TUW
     +                ,Y1,Y2,X1,X2,Z1,Z2
     +                ,YACOF,YBCOF,XACOF,XBCOF,ZACOF,ZBCOF)

       ELSEIF((ABS(Z1).GT.ABS(Y1)).AND.(ABS(Z1).GT.ABS(X1))) THEN
         CALL BRP9XYZ(V,BRC1,BRC2,PKL1,PKL2,IUW,TUW
     +                ,Z1,Z2,Y1,Y2,X1,X2
     +                ,ZACOF,ZBCOF,YACOF,YBCOF,XACOF,XBCOF)

      ENDIF
```

```
      RETURN
      END
C----------------------------------------------
C
C     SUBROUTINE: BRP9XYZ
C
C     DESCRIPTION: ELIMINATION ROUTINE WHERE THE CUBIC VARIABLE
C                  IS DEFINED EXPLICITLY ON SURFACE 1
C
C INPUT:
C     U1           = PARAMETRIC VALUE
C     BRC1,BRC2    = SURFACES COEFF.
C     PKL1,PKL2    = INTERSECTING SURFACES
C     IUW          = LINEAR OR CUBIC VARIABLE FLAG
C     A1,A2,B1,
C     B2,C1,C2     = VALUES OF THE CUBIC FUNCTIONS IN THE
C                    SURFACES EQS.
C     IACOF,IBCOF,JACOF,
C     JBCOF,KACOF,KBCOF
C                  = SURFACES EQS. FOR SURFACE 2
C
C OUTPUT:
C     TUW          = INTERSECTION POINT IN PARAMETRIC SPACE
C
C     C. K. WONG
C     2/12/90
C

      SUBROUTINE BRP9XYZ(U1,BRC1,BRC2,PKL1,PKL2,IUW,TUW
     +                  ,A1,A2,B1,B2,C1,C2
     +                  ,IACOF,IBCOF,JACOF,JBCOF,KACOF,KBCOF)

      REAL BRC1(3,8),BRC2(3,8),COF(4)
     +     ,TUW(4),KST1,KST2,CST1,CST2
     +     ,IACOF(4),IBCOF(4),JACOF(4),JBCOF(4),KACOF(4),KBCOF(4)
     +     ,COF1(7),COF2(7),COF3(7),COF4(7)
     +     ,COF5(7),COF6(7),COEFF(7)
     +     ,IA,IB,JA,JB,KA,KB
     +     ,O1,O2,P1,P2,MIN
     +     ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)
     +     ,UA1(10),WA1(10),UA2(10),WA2(10)

      INTEGER IUW(2),NDEG

      COMPLEX  ZERO(6)

      EXTERNAL ZPLRC

C INITIALIZATION
      TLIM=.001

C
C SOLVE FOR THE CUBIC VARIABLE ON SURFACE 2:
C


      CST1=B1/A1
      CST2=B2-((A2*B1)/A1)
      KST1=C1/A1
      KST2=C2-((A2*C1)/A1)


 12   CALL MULCO10(IBCOF,KACOF,COF1)
      CALL MULCO10(IACOF,KBCOF,COF2)
      CALL MULCO10(JACOF,KBCOF,COF3)
      CALL MULCO10(IBCOF,JACOF,COF4)
      CALL MULCO10(IACOF,JBCOF,COF5)
```

```
      CALL MULCO10(KACOF,JBCOF,COF6)

      NDEG=6
      DO 100 I=NDEG+1,5,-1
         COEFF(I)=(CST1*COF1(8-I))-(CST1*COF2(8-I))+COF3(8-I)
     +            -(KST1*COF4(8-I))+(KST1*COF5(8-I))-COF6(8-I)
 100  CONTINUE

      DO 110 I=4,1,-1
         COEFF(I)=(CST1*COF1(8-I))-(CST1*COF2(8-I))+COF3(8-I)
     +            -(KST1*COF4(8-I))+(KST1*COF5(8-I))-COF6(8-I)
     +            +(CST1*KST2*IACOF(5-I))-(KST2*JACOF(5-I))
     +            -(KST1*CST2*IACOF(5-I))+(CST2*KACOF(5-I))
 110  CONTINUE


      CALL ZPLRC1(NDEG,COEFF,ZERO)

      IFOUND=1
      DO 700 M=1,NDEG

C DESIGNATE SOLUTION 'REAL' OR 'COMPLEX'
         IF(ABS(AIMAG(ZERO(M))).GE.1.E-4) GOTO 700


         W2=REAL(ZERO(M))

C IF THE VARIABLE OUT OF RANGE?
         IF((W2.LT.-(TLIM)).OR.(W2.GT.(1.+TLIM)))GOTO 700

C SOLVE FOR THE LINEAR VARIABLE ON SURFACE 1

                CALL EVAFUN(IACOF,W2,IA)
                CALL EVAFUN(IBCOF,W2,IB)
                CALL EVAFUN(JACOF,W2,JA)
                CALL EVAFUN(JBCOF,W2,JB)
                CALL EVAFUN(KACOF,W2,KA)
                CALL EVAFUN(KBCOF,W2,KB)

C ARRANGE THE ELIMINATION EQS. SUCH THAT THE
C DENOMINATOR WILL HAVE THE LARGEST POSSIBLE VALUE

                MIN=ABS(IA)
                S1=B1
                S2=B2
                SA=JA
                SB=JB
                T1=C1
                T2=C2
                TA=KA
                TB=KB
                IF(ABS(JA).LT.MIN) THEN
                      S1=A1
                      S2=A2
                      SA=IA
                      SB=IB
                      MIN=ABS(JA)

                      IF(ABS(KA).LT.MIN) THEN
                         T1=B1
                         T2=B2
                         TA=JA
                         TB=JB
                      ENDIF

                   ELSEIF(ABS(KA).LT.MIN) THEN
                      T1=A1
```

```
                  T2=A2
                  TA=IA
                  TB=IB
              ENDIF

              O2=S2-SB
              O2=O2/SA
              O1=S1/SA

              P2=T2-TB
              P2=P2/TA
              P1=T1/TA

              Z1=O1-P1
              Z2=O2-P2
              W1=-(Z2)/Z1

C IS THE VARIABLE OUT OF RANGE ?
              IF((W1.GT.(1.+TLIM)).OR.(W1.LT.-(TLIM))) GOTO 700

C SOLVE FOR THE LINEAR VARIABLE ON SURFACE 2:
C ARRANGE THE ELIMINATION EQS. SUCH THAT THE
C DENOMINATOR WILL HAVE THE LARGEST POSSIBLE VALUE

              MIN=ABS(A1)
              S1=B1
              S2=B2
              SA=JA
              SB=JB
              T1=C1
              T2=C2
              TA=KA
              TB=KB
              IF(ABS(B1).LT.MIN) THEN
                  S1=A1
                  S2=A2
                  SA=IA
                  SB=IB
                  MIN=ABS(JA)

                  IF(ABS(C1).LT.MIN) THEN
                     T1=B1
                     T2=B2
                     TA=JA
                     TB=JB
                  ENDIF

                ELSEIF(ABS(C1).LT.MIN) THEN
                     T1=A1
                     T2=A2
                     TA=IA
                     TB=IB
              ENDIF

              O2=SB-S2
              O2=O2/S1
              O1=SA/S1

              P2=TB-T2
              P2=P2/T1
              P1=TA/T1

              Z1=O1-P1
              Z2=O2-P2
              U2=-(Z2)/Z1

C IS THE VARIABLE OUT OF RANGE?
```

```
            IF((U2.GT.(1.+TLIM)).OR.(U2.LT.-(TLIM))) GOTO 700

            UA1(IFOUND)=U1
            WA1(IFOUND)=W1
            UA2(IFOUND)=U2
            WA2(IFOUND)=W2

            IFOUND=IFOUND+1
 700  CONTINUE

      IFOUND=IFOUND-1

C NO INTERSECTION
         IF(IFOUND.EQ.0) THEN
            DO 240 I=1,4
               TUW(I)=2.
 240        CONTINUE

           ELSE

C SELECT THE RESULT BY COMPARING ALL THE
C RESULTS, AND PICK THE PAIR OF INTERSECTION POINTS
C THAT HAS THE SMALLEST DISTANCE


         IF((IUW(1).EQ.0).AND.(IUW(2).EQ.1)) THEN
            CALL COMRELT(PKL1,PKL2,UA1,WA1,UA2,WA2,IFOUND,TUW)
          ELSEIF((IUW(1).EQ.1).AND.(IUW(2).EQ.1)) THEN
            CALL COMRELT(PKL1,PKL2,WA1,UA1,UA2,WA2,IFOUND,TUW)
          ELSEIF((IUW(1).EQ.0).AND.(IUW(2).EQ.0)) THEN
            CALL COMRELT(PKL1,PKL2,UA1,WA1,WA2,UA2,IFOUND,TUW)
          ELSEIF((IUW(1).EQ.1).AND.(IUW(2).EQ.0)) THEN
            CALL COMRELT(PKL1,PKL2,WA1,UA1,WA2,UA2,IFOUND,TUW)
         ENDIF

         ENDIF



      RETURN
      END
C-------------------------------------------------
C
C     SUBROUTINE: BRP33A
C
C     DESCRIPTION: DEFINE THE LINEAR VARIABLE ON SURFACE 1
C                  IN THE ELIMINATION ROUTINE
C
C INPUT:
C     V            = PARAMETRIC VALUE
C     BRC1,BRC2    = SURFACES COEFF.
C     PKL1,PKL2    = INTERSECTING SURFACES
C     IUW          = LINEAR OR CUBIC VARIABLE FLAG
C
C OUTPUT:
C     TUW          = INTERSECTION POINT IN PARAMETRIC SPACE
C
C     C. K. WONG
C     2/12/90
C

      SUBROUTINE BRP33A(V,BRC1,BRC2,PKL1,PKL2,IUW,TUW)

      REAL BRC1(3,8),BRC2(3,8),
     +     TUW(4),EQ1(19),EQ2(19),
     +     XACOF(4),XBCOF(4),YACOF(4),YBCOF(4),ZACOF(4),ZBCOF(4),
     +     X3COF(4),Y3COF(4),Z3COF(4)
```

```
     +      ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER IUW(2)

C DEFINE THE LINEAR VARIABLE ON SURFACE 1,
C AND ARRANGE THE ELIMINATION EQS.

      DO 35 JJ=1,4
         X3COF(JJ)=(V*BRC1(1,JJ))+BRC1(1,JJ+4)
         Y3COF(JJ)=(V*BRC1(2,JJ))+BRC1(2,JJ+4)
         Z3COF(JJ)=(V*BRC1(3,JJ))+BRC1(3,JJ+4)
         XACOF(JJ)=BRC2(1,JJ)
         XBCOF(JJ)=BRC2(1,JJ+4)
         YACOF(JJ)=BRC2(2,JJ)
         YBCOF(JJ)=BRC2(2,JJ+4)
         ZACOF(JJ)=BRC2(3,JJ)
         ZBCOF(JJ)=BRC2(3,JJ+4)
 35   CONTINUE


C CREATE THE EQUATIONS EQ1 & EQ2 BY ELIMINATING
C ONE VARIABLE FROM THE THREE PARAMETRIC SURFACE
C EQS.

      CALL BRPCREQ(X3COF,Y3COF,Z3COF,XACOF,XBCOF,
     +             YACOF,YBCOF,ZACOF,ZBCOF,EQ1,EQ2)


C SOLVE THE VARIABLES USING ELIMINATION METHOD

      CALL BRPSLVA(PKL1,PKL2,EQ1,EQ2,X3COF,Y3COF,Z3COF,XACOF,XBCOF,
     +             YACOF,YBCOF,ZACOF,ZBCOF,V,IUW,TUW)


      RETURN
      END
C-------------------------------------------------------
C
C
C     SUBROUTINE: BRPCREQ
C
C     DESCRIPTION: CREATE THE EQUATIONS EQ1 & EQ2:
C                  ELIMINATE ONE VARIABLE FROM THREE
C                  EQS. WITH THREE UNKNOWNS
C
C INPUT:
C     X3,Y3,Z3     = SURFACE 1 EQS.
C     XA,XB,YA,
C     YB,ZA,ZB     = SURFACE 2 EQS.
C
C OUTPUT:
C     EQ1,EQ2      = INTERSECTION EQS. WITH TWO UNKNOWNS
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE BRPCREQ(X3,Y3,Z3,XA,XB,
     +             YA,YB,ZA,ZB,EQ1,EQ2)


      REAL EQ1(19),EQ2(19),EQ3(19),
     +     XA(4),XB(4),YA(4),YB(4),ZA(4),ZB(4),
     +     X3(4),Y3(4),Z3(4)
     +     ,T3(4),TA(4),TB(4)
     +     ,TEQ1A(19),TEQ1B(19)
     +     ,TEQ2A(19),TEQ2B(19)
```

```
     +     ,MIN,MIN1,MIN2,MIN3

C*****************************************************
C The following procedures need special attention where
C the trouble always occur. This is due to the different setup
C of eqns. for elimination process.
C*****************************************************

C TRY ALL THE POSSIBLE COMBINATION OF ELIMINATION TO
C SELECT THE ONE THAT YIELD THE REASONABLE COEFF.
C FOR THE EQS.

      ITRY=1

C PREPARE FOR THE SECOND COMBINATION OF ELIMINATION
 5    IF(ITRY.EQ.2) THEN
         DO 10 I=1,4
              T3(I)=X3(I)
              TA(I)=XA(I)
              TB(I)=XB(I)
              X3(I)=Y3(I)
              XA(I)=YA(I)
              XB(I)=YB(I)
              Y3(I)=T3(I)
              YA(I)=TA(I)
              YB(I)=TB(I)
 10      CONTINUE
      ENDIF


C THE EQUATIONS : EQ1 & EQ2

      EQ1(1)=Y3(1)*XA(1)-X3(1)*YA(1)
      EQ1(2)=Y3(1)*XA(2)-X3(1)*YA(2)
      EQ1(3)=Y3(1)*XA(3)-X3(1)*YA(3)
      EQ1(4)=Y3(1)*XA(4)-X3(1)*YA(4)
      EQ1(5)=Y3(2)*XA(1)-X3(2)*YA(1)
      EQ1(6)=Y3(2)*XA(2)-X3(2)*YA(2)
      EQ1(7)=Y3(2)*XA(3)-X3(2)*YA(3)
      EQ1(8)=Y3(2)*XA(4)-X3(2)*YA(4)
      EQ1(9)=Y3(3)*XA(1)-X3(3)*YA(1)
      EQ1(10)=Y3(3)*XA(2)-X3(3)*YA(2)
      EQ1(11)=Y3(3)*XA(3)-X3(3)*YA(3)
      EQ1(12)=Y3(3)*XA(4)-X3(3)*YA(4)
      EQ1(13)=XB(1)*YA(1)-YB(1)*XA(1)
      EQ1(14)=XB(1)*YA(2)-YB(1)*XA(2)+
     +         XB(2)*YA(1)-YB(2)*XA(1)
      EQ1(15)=XB(1)*YA(3)-YB(1)*XA(3)+
     +         XB(2)*YA(2)-YB(2)*XA(2)+
     +         XB(3)*YA(1)-YB(3)*XA(1)
      EQ1(16)=XB(1)*YA(4)-YB(1)*XA(4)+
     +         XB(2)*YA(3)-YB(2)*XA(3)+
     +         XB(3)*YA(2)-YB(3)*XA(2)+
     +         XB(4)*YA(1)-YB(4)*XA(1)+
     +         Y3(4)*XA(1)-X3(4)*YA(1)
      EQ1(17)=XB(2)*YA(4)-YB(2)*XA(4)+
     +         XB(3)*YA(3)-YB(3)*XA(3)+
     +         XB(4)*YA(2)-YB(4)*XA(2)+
     +         Y3(4)*XA(2)-X3(4)*YA(2)
      EQ1(18)=XB(3)*YA(4)-YB(3)*XA(4)+
     +         XB(4)*YA(3)-YB(4)*XA(3)+
     +         Y3(4)*XA(3)-X3(4)*YA(3)
      EQ1(19)=XB(4)*YA(4)-YB(4)*XA(4)+
     +         Y3(4)*XA(4)-X3(4)*YA(4)

      EQ2(1)=Z3(1)*XA(1)-X3(1)*ZA(1)
```

```
      EQ2(2)=Z3(1)*XA(2)-X3(1)*ZA(2)
      EQ2(3)=Z3(1)*XA(3)-X3(1)*ZA(3)
      EQ2(4)=Z3(1)*XA(4)-X3(1)*ZA(4)
      EQ2(5)=Z3(2)*XA(1)-X3(2)*ZA(1)
      EQ2(6)=Z3(2)*XA(2)-X3(2)*ZA(2)
      EQ2(7)=Z3(2)*XA(3)-X3(2)*ZA(3)
      EQ2(8)=Z3(2)*XA(4)-X3(2)*ZA(4)
      EQ2(9)=Z3(3)*XA(1)-X3(3)*ZA(1)
      EQ2(10)=Z3(3)*XA(2)-X3(3)*ZA(2)
      EQ2(11)=Z3(3)*XA(3)-X3(3)*ZA(3)
      EQ2(12)=Z3(3)*XA(4)-X3(3)*ZA(4)
      EQ2(13)=XB(1)*ZA(1)-ZB(1)*XA(1)
      EQ2(14)=XB(1)*ZA(2)-ZB(1)*XA(2)+
     +         XB(2)*ZA(1)-ZB(2)*XA(1)
      EQ2(15)=XB(1)*ZA(3)-ZB(1)*XA(3)+
     +         XB(2)*ZA(2)-ZB(2)*XA(2)+
     +         XB(3)*ZA(1)-ZB(3)*XA(1)
      EQ2(16)=XB(1)*ZA(4)-ZB(1)*XA(4)+
     +         XB(2)*ZA(3)-ZB(2)*XA(3)+
     +         XB(3)*ZA(2)-ZB(3)*XA(2)+
     +         XB(4)*ZA(1)-ZB(4)*XA(1)+
     +         Z3(4)*XA(1)-X3(4)*ZA(1)
      EQ2(17)=XB(2)*ZA(4)-ZB(2)*XA(4)+
     +         XB(3)*ZA(3)-ZB(3)*XA(3)+
     +         XB(4)*ZA(2)-ZB(4)*XA(2)+
     +         Z3(4)*XA(2)-X3(4)*ZA(2)
      EQ2(18)=XB(3)*ZA(4)-ZB(3)*XA(4)+
     +         XB(4)*ZA(3)-ZB(4)*XA(3)+
     +         Z3(4)*XA(3)-X3(4)*ZA(3)
      EQ2(19)=XB(4)*ZA(4)-ZB(4)*XA(4)+
     +         Z3(4)*XA(4)-X3(4)*ZA(4)

      IF(ITRY.EQ.1) THEN

C STORE THE RESULT FOR COMPARISON
        DO 200 K=1,19
              TEQ1A(K)=EQ1(K)
              TEQ2A(K)=EQ2(K)
 200    CONTINUE
      ENDIF


C EXCHANGE THE VARIABLES BACK TO ORIGINAL
C VALUE AFTER THE SECOND COMBINATION
      IF(ITRY.EQ.2) THEN
         DO 20 I=1,4
              T3(I)=X3(I)
              TA(I)=XA(I)
              TB(I)=XB(I)
              X3(I)=Y3(I)
              XA(I)=YA(I)
              XB(I)=YB(I)
              Y3(I)=T3(I)
              YA(I)=TA(I)
              YB(I)=TB(I)
 20      CONTINUE

C STORE THE RESULT FOR COMPARISON

         DO 100 K=1,19
              TEQ1B(K)=EQ1(K)
              TEQ2B(K)=EQ2(K)
 100     CONTINUE
      ENDIF

C TRY SECOND COMBINATION
```

```
      IF(ITRY.EQ.1) THEN
         ITRY=2
         GOTO 5
       ENDIF

C PICK THE SMALLEST COEFF. AS THE RESULT
         MIN1=ABS(TEQ1A(13))
         MIN2=ABS(TEQ2A(13))
         MIN3=ABS(TEQ2B(13))

      MIN=MIN1
      IEQ1=1
      IF(MIN2.LT.MIN) THEN
         IEQ1=2
         MIN=MIN2
      ENDIF
      IF(MIN3.LT.MIN) THEN
         IEQ1=3
      ENDIF


      DO 300 K=1,19

         IF(IEQ1.EQ.1) THEN
              EQ1(K)=TEQ2A(K)
              EQ2(K)=TEQ2B(K)
           ELSEIF(IEQ1.EQ.2) THEN
              EQ1(K)=TEQ1A(K)
              EQ2(K)=TEQ2B(K)
           ELSEIF(IEQ1.EQ.3) THEN
              EQ1(K)=TEQ1A(K)
              EQ2(K)=TEQ2A(K)
         ENDIF


 300  CONTINUE


      RETURN
      END
C---------------------------------------------------
C
C     SUBROUTINE: BRPSLVA
C
C     DESCRIPTION: SOLVE FOR THE VARIABLES USING
C                  ELIMINATION METHOD (GENERATE A
C                  33RD ORDER POLYNOMIAL)
C
C INPUT:
C     PKL1,PKL2     = INTERSECTING SURFACES
C     EQ1,EQ2       = TWO INTERSECTION EQS. WITH TWO UNKNOWNS
C     X3,Y3,Z3      = SURFACE 1 EQS.
C     XA,XB,YA,
C     YB,ZA,ZB      = SURFACE 2 EQS.
C     W1            = PARAMETRIC VALUE
C     IUW           = LINEAR OR CUBIC VARIABLE FLAG
C
C OUTPUT:
C     TUW           = INTERSECTION POINT IN PARAMETRIC SPACE
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE BRPSLVA(PKL1,PKL2,EQ1,EQ2,X3,Y3,Z3,XA,XB,
     +             YA,YB,ZA,ZB,W1,IUW,TUW)
```

```
      REAL TUW(4),EQ1(19),EQ2(19),E33(34)
     +     XA(4),XB(4),YA(4),YB(4),ZA(4),ZB(4),
     +     X3(4),Y3(4),Z3(4)
     +     ,MAX,MIN
     +     ,WA1(60),WA2(60),UA1(60),UA2(60)
     +     ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER IUW(2),M
      INTEGER NDEG
      PARAMETER(NDEG=33)
      COMPLEX ZERO(33)
      EXTERNAL ZPLRC


C SET TOLERANCE
      TLIM=.001

C ELIMINATE ONE UNKNOWN FROM EQ1 & EQ2 TO YIELD
C A 33RD ORDER POLYNOMIAL
      CALL BRP33(EQ1,EQ2,E33)

C SOLVE THE 33RD ORDER POLYNOMIAL, AND OBTAIN THE
C CUBIC VARIABLE ON SURFACE 2

      CALL ZPLRC(NDEG,E33,ZERO)

      IFOUND=1
      DO 650 M=1,NDEG

C DESIGNATE SOLUTION 'REAL' OR 'COMPLEX'

         IF(ABS(AIMAG(ZERO(M))).GE.1.E-5) GOTO 650
         W2=REAL(ZERO(M))


C IS THE VARIABLE OUT OF RANGE ?
         IF((W2.GT.(1.+TLIM)).OR.
     +      (W2.LT.-(TLIM))) GOTO 650

C BACK SUBSTITUTING TO SOLVE FOR THE REST OF
C THE VARIABLES

         CALL SBRP33(EQ1,EQ2,X3,Y3,Z3,
     +               XA,XB,YA,YB,ZA,ZB
     +              ,W1,W2,UA1,WA1,UA2,WA2,IFOUND)


 650  CONTINUE

      IFOUND=IFOUND-1

C NO INTERSECTION
         IF(IFOUND.EQ.0) THEN
              DO 240 I=1,4
                  TUW(I)=2.
 240          CONTINUE

            ELSE

C SELECT THE RESULT BY COMPARING ALL THE RESULTS,
C AND PICK THE PAIR OF POINTS THAT HAS THE
C SMALLEST DISTANCE BETWEEN THEM

         IF((IUW(1).EQ.0).AND.(IUW(2).EQ.1)) THEN
              CALL COMRELT(PKL1,PKL2,UA1,WA1,UA2,WA2,IFOUND,TUW)
           ELSEIF((IUW(1).EQ.1).AND.(IUW(2).EQ.1)) THEN
              CALL COMRELT(PKL1,PKL2,WA1,UA1,UA2,WA2,IFOUND,TUW)
```

```
          ELSEIF((IUW(1).EQ.0).AND.(IUW(2).EQ.0)) THEN
             CALL COMRELT(PKL1,PKL2,UA1,WA1,WA2,UA2,IFOUND,TUW)
          ELSEIF((IUW(1).EQ.1).AND.(IUW(2).EQ.0)) THEN
             CALL COMRELT(PKL1,PKL2,WA1,UA1,WA2,UA2,IFOUND,TUW)
        ENDIF

        ENDIF


      RETURN
      END
C-----------------------------------------------------
C
C     SUBROUTINE: BRP33
C
C     DESCRIPTION: ELIMINATE ONE UNKNOWN FROM TWO
C                  ORDERED FUNCTIONS TO YIELD A 33RD
C                  ORDER POLYNOMIAL
C
C INPUT:
C     EQ1,EQ2      = TWO FUNCTIONS WITH TWO UNKNOWNS
C
C OUTPUT:
C     W            = 33RD ORDER POLYNOMIAL
C
C     C. K. WONG
C     9/12/89
C
      SUBROUTINE BRP33(EQ1,EQ2,W)

      REAL EQ1(19),EQ2(19)
      REAL W(34)

      DOUBLE PRECISION A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,
     +     AP,BP,CP,DP,EP,FP,GP,HP,IP,JP,KP,LP,MP,NP,OP,PP,
     +     QP,RP,SP,
     +     AA,BB,CC,DD,EE,FF,GG,HH,II,JJ,KK,LL,MM,NN,
     +     X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,
     +     AAP,BBP,CCP,DDP,EEP,FFP,GGP,HHP,IIP,JJP,KKP,LLP,
     +     MMP,NNP,
     +     X1P,X2P,X3P,X4P,X5P,X6P,X7P,X8P,X9P,X10P,
     +     Y1P,Y2P,Y3P,
     +     S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,
     +     S11,S12,S13,S14,S15,S16,
     +     T1,T2,T3,
     +     Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8,Z9,Z10,
     +     Z11,Z12,Z13,Z14,Z15,Z16


C SET UP THE EQS.


      A=EQ1(1)
      B=EQ1(2)
      C=EQ1(3)
      D=EQ1(4)
      E=EQ1(5)
      F=EQ1(6)
      G=EQ1(7)
      H=EQ1(8)
      I=EQ1(9)
      J=EQ1(10)
      K=EQ1(11)
      L=EQ1(12)
      M=EQ1(13)
      N=EQ1(14)
```

```
      O=EQ1(15)
      P=EQ1(16)
      Q=EQ1(17)
      R=EQ1(18)
      S=EQ1(19)


      AP=EQ2(1)
      BP=EQ2(2)
      CP=EQ2(3)
      DP=EQ2(4)
      EP=EQ2(5)
      FP=EQ2(6)
      GP=EQ2(7)
      HP=EQ2(8)
      IP=EQ2(9)
      JP=EQ2(10)
      KP=EQ2(11)
      LP=EQ2(12)
      MP=EQ2(13)
      NP=EQ2(14)
      OP=EQ2(15)
      PP=EQ2(16)
      QP=EQ2(17)
      RP=EQ2(18)
      SP=EQ2(19)

C STEP 3

      AA=(A*EP-E*AP)
      BB=(A*FP-F*AP)+(B*EP-E*BP)
      CC=(A*GP-G*AP)+(B*FP-F*BP)+(C*EP-E*CP)
      DD=(A*HP-H*AP)+(B*GP-G*BP)+(C*FP-F*CP)+(D*EP-E*DP)
      EE=(B*HP-H*BP)+(C*GP-G*CP)+(D*FP-F*DP)
      FF=(C*HP-H*CP)+(D*GP-G*DP)
      GG=(D*HP-H*DP)
      HH=(A*IP-I*AP)
      II=(A*JP-J*AP)+(B*IP-I*BP)
      JJ=(A*KP-K*AP)+(B*JP-J*BP)+(C*IP-I*CP)
      KK=(A*LP-L*AP)+(B*KP-K*BP)+(C*JP-J*CP)+(D*IP-I*DP)
      LL=(B*LP-L*BP)+(C*KP-K*CP)+(D*JP-J*DP)
      MM=(C*LP-L*CP)+(D*KP-K*DP)
      NN=(D*LP-L*DP)
      X1=(A*MP-M*AP)
      X2=(A*NP-N*AP)+(B*MP-M*BP)
      X3=(A*OP-O*AP)+(B*NP-N*BP)+(C*MP-M*CP)
      X4=(A*PP-P*AP)+(B*OP-O*BP)+(C*NP-N*CP)+(D*MP-M*DP)
      X5=(A*QP-Q*AP)+(B*PP-P*BP)+(C*OP-O*CP)+(D*NP-N*DP)
      X6=(A*RP-R*AP)+(B*QP-Q*BP)+(C*PP-P*CP)+(D*OP-O*DP)
      X7=(A*SP-S*AP)+(B*RP-R*BP)+(C*QP-Q*CP)+(D*PP-P*DP)
      X8=(B*SP-S*BP)+(C*RP-R*CP)+(D*QP-Q*DP)
      X9=(C*SP-S*CP)+(D*RP-R*DP)
      X10=(D*SP-S*DP)

C STEP 4

      AAP=(A*IP-I*AP)
      BBP=(A*JP-J*AP)+(B*IP-I*BP)
      CCP=(A*KP-K*AP)+(B*JP-J*BP)+(C*IP-I*CP)
      DDP=(A*LP-L*AP)+(B*KP-K*BP)+(C*JP-J*CP)+(D*IP-I*DP)
      EEP=(B*LP-L*BP)+(C*KP-K*CP)+(D*JP-J*DP)
      FFP=(C*LP-L*CP)+(D*KP-K*DP)
      GGP=(D*LP-L*DP)
      Y1P=(A*MP-M*AP)
      Y2P=(A*NP-N*AP)+(B*MP-M*BP)
      Y3P=(A*OP-O*AP)+(B*NP-N*BP)+(C*MP-M*CP)
```

```
      HHP=(A*PP-P*AP)+(B*OP-O*BP)+(C*NP-N*CP)+(D*MP-M*DP)+(E*IP-I*EP)
      IIP=(A*QP-Q*AP)+(B*PP-P*BP)+(C*OP-O*CP)+(D*NP-N*DP)+(E*JP-J*EP)+
     +     (F*IP-I*FP)
      JJP=(A*RP-R*AP)+(B*QP-Q*BP)+(C*PP-P*CP)+(D*OP-O*DP)+(E*KP-K*EP)+
     +     (F*JP-J*FP)+(G*IP-I*GP)
      KKP=(A*SP-S*AP)+(B*RP-R*BP)+(C*QP-Q*CP)+(D*PP-P*DP)+(E*LP-L*EP)+
     +     (F*KP-K*FP)+(G*JP-J*GP)+(H*IP-I*HP)
      LLP=(B*SP-S*BP)+(C*RP-R*CP)+(D*QP-Q*DP)+(F*LP-L*FP)+(G*KP-K*GP)+
     +     (H*JP-J*HP)
      MMP=(C*SP-S*CP)+(D*RP-R*DP)+(G*LP-L*GP)+(H*KP-K*HP)
      NNP=(D*SP-S*DP)+(H*LP-L*HP)
      X1P=(E*MP-M*EP)
      X2P=(E*NP-N*EP)+(F*MP-M*FP)
      X3P=(E*OP-O*EP)+(F*NP-N*FP)+(G*MP-M*GP)
      X4P=(E*PP-P*EP)+(F*OP-O*FP)+(G*NP-N*GP)+(H*MP-M*HP)
      X5P=(E*QP-Q*EP)+(F*PP-P*FP)+(G*OP-O*GP)+(H*NP-N*HP)
      X6P=(E*RP-R*EP)+(F*QP-Q*FP)+(G*PP-P*GP)+(H*OP-O*HP)
      X7P=(E*SP-S*EP)+(F*RP-R*FP)+(G*QP-Q*GP)+(H*PP-P*HP)
      X8P=(F*SP-S*FP)+(G*RP-R*GP)+(H*QP-Q*HP)
      X9P=(G*SP-S*GP)+(H*RP-R*HP)
      X10P=(H*SP-S*HP)

C STEP 5

      A=(AA*Y1P)
      B=(BB*Y1P)+(AA*Y2P)
      C=(CC*Y1P)+(BB*Y2P)+(AA*Y3P)
      D=(DD*Y1P)+(CC*Y2P)+(BB*Y3P)+(AA*HHP-AAP*HH)
      E=(EE*Y1P)+(DD*Y2P)+(CC*Y3P)+(BB*HHP-BBP*HH)+(AA*IIP-AAP*II)
      F=(FF*Y1P)+(EE*Y2P)+(DD*Y3P)+(CC*HHP-CCP*HH)+(BB*IIP-BBP*II)+
     +  (AA*JJP-AAP*JJ)
      G=(GG*Y1P)+(FF*Y2P)+(EE*Y3P)+(DD*HHP-DDP*HH)+(CC*IIP-CCP*II)+
     +  (BB*JJP-BBP*JJ)+(AA*KKP-AAP*KK)
      H=(GG*Y2P)+(FF*Y3P)+(EE*HHP-EEP*HH)+(DD*IIP-DDP*II)+
     +  (CC*JJP-CCP*JJ)+(BB*KKP-BBP*KK)+(AA*LLP-AAP*LL)
      I=(GG*Y3P)+(FF*HHP-FFP*HH)+(EE*IIP-EEP*II)+(DD*JJP-DDP*JJ)+
     +  (CC*KKP-CCP*KK)+(BB*LLP-BBP*LL)+(AA*MMP-AAP*MM)
      J=(GG*HHP-GGP*HH)+(FF*IIP-FFP*II)+(EE*JJP-EEP*JJ)+(DD*KKP-DDP*KK)+
     +  (CC*LLP-CCP*LL)+(BB*MMP-BBP*MM)+(AA*NNP-AAP*NN)
      K=(GG*IIP-GGP*II)+(FF*JJP-FFP*JJ)+(EE*KKP-EEP*KK)+(DD*LLP-DDP*LL)+
     +  (CC*MMP-CCP*MM)+(BB*NNP-BBP*NN)
      L=(GG*JJP-GGP*JJ)+(FF*KKP-FFP*KK)+(EE*LLP-EEP*LL)+(DD*MMP-DDP*MM)+
     +  (CC*NNP-CCP*NN)
      M=(GG*KKP-GGP*KK)+(FF*LLP-FFP*LL)+(EE*MMP-EEP*MM)+(DD*NNP-DDP*NN)
      N=(GG*LLP-GGP*LL)+(FF*MMP-FFP*MM)+(EE*NNP-EEP*NN)
      O=(GG*MMP-GGP*MM)+(FF*NNP-FFP*NN)
      P=(GG*NNP-GGP*NN)
      S1=(AA*X1P-AAP*X1)
      S2=(BB*X1P-BBP*X1)+(AA*X2P-AAP*X2)
      S3=(CC*X1P-CCP*X1)+(BB*X2P-BBP*X2)+(AA*X3P-AAP*X3)
      S4=(DD*X1P-DDP*X1)+(CC*X2P-CCP*X2)+(BB*X3P-BBP*X3)+(AA*X4P-AAP*X4)
      S5=(EE*X1P-EEP*X1)+(DD*X2P-DDP*X2)+(CC*X3P-CCP*X3)+(BB*X4P-BBP*X4)
     +  +(AA*X5P-AAP*X5)
      S6=(FF*X1P-FFP*X1)+(EE*X2P-EEP*X2)+(DD*X3P-DDP*X3)+(CC*X4P-CCP*X4)
     +  +(BB*X5P-BBP*X5)+(AA*X6P-AAP*X6)
      S7=(GG*X1P-GGP*X1)+(FF*X2P-FFP*X2)+(EE*X3P-EEP*X3)+(DD*X4P-DDP*X4)
     +  +(CC*X5P-CCP*X5)+(BB*X6P-BBP*X6)+(AA*X7P-AAP*X7)
      S8=(GG*X2P-GGP*X2)+(FF*X3P-FFP*X3)+(EE*X4P-EEP*X4)+(DD*X5P-DDP*X5)
     +  +(CC*X6P-CCP*X6)+(BB*X7P-BBP*X7)+(AA*X8P-AAP*X8)
      S9=(GG*X3P-GGP*X3)+(FF*X4P-FFP*X4)+(EE*X5P-EEP*X5)+(DD*X6P-DDP*X6)
     +  +(CC*X7P-CCP*X7)+(BB*X8P-BBP*X8)+(AA*X9P-AAP*X9)
      S10=(GG*X4P-GGP*X4)+(FF*X5P-FFP*X5)+(EE*X6P-EEP*X6)+
     + (DD*X7P-DDP*X7)+(CC*X8P-CCP*X8)+(BB*X9P-BBP*X9)+(AA*X10P-AAP*X10)
      S11=(GG*X5P-GGP*X5)+(FF*X6P-FFP*X6)+(EE*X7P-EEP*X7)+
     +     (DD*X8P-DDP*X8)+(CC*X9P-CCP*X9)+(BB*X10P-BBP*X10)
      S12=(GG*X6P-GGP*X6)+(FF*X7P-FFP*X7)+(EE*X8P-EEP*X8)+
     +     (DD*X9P-DDP*X9)+(CC*X10P-CCP*X10)
```

```
      S13=(GG*X7P-GGP*X7)+(FF*X8P-FFP*X8)+(EE*X9P-EEP*X9)+
     +    (DD*X10P-DDP*X10)
      S14=(GG*X8P-GGP*X8)+(FF*X9P-FFP*X9)+(EE*X10P-EEP*X10)
      S15=(GG*X9P-GGP*X9)+(FF*X10P-FFP*X10)
      S16=(GG*X10P-GGP*X10)


C STEP 6

      AP=(AAP*X1-AA*X1P)
      BP=(BBP*X1-BB*X1P)+(AAP*X2-AA*X2P)
      CP=(CCP*X1-CC*X1P)+(BBP*X2-BB*X2P)+(AAP*X3-AA*X3P)
      DP=(DDP*X1-DD*X1P)+(CCP*X2-CC*X2P)+(BBP*X3-BB*X3P)+(AAP*X4-AA*X4P)
      EP=(EEP*X1-EE*X1P)+(DDP*X2-DD*X2P)+(CCP*X3-CC*X3P)+(BBP*X4-BB*X4P)
     +  +(AAP*X5-AA*X5P)
      FP=(FFP*X1-FF*X1P)+(EEP*X2-EE*X2P)+(DDP*X3-DD*X3P)+(CCP*X4-CC*X4P)
     +  +(BBP*X5-BB*X5P)+(AAP*X6-AA*X6P)
      GP=(GGP*X1-GG*X1P)+(FFP*X2-FF*X2P)+(EEP*X3-EE*X3P)+(DDP*X4-DD*X4P)
     +  +(CCP*X5-CC*X5P)+(BBP*X6-BB*X6P)+(AAP*X7-AA*X7P)
      HP=(GGP*X2-GG*X2P)+(FFP*X3-FF*X3P)+(EEP*X4-EE*X4P)+(DDP*X5-DD*X5P)
     +  +(CCP*X6-CC*X6P)+(BBP*X7-BB*X7P)+(AAP*X8-AA*X8P)
      IP=(GGP*X3-GG*X3P)+(FFP*X4-FF*X4P)+(EEP*X5-EE*X5P)+(DDP*X6-DD*X6P)
     +  +(CCP*X7-CC*X7P)+(BBP*X8-BB*X8P)+(AAP*X9-AA*X9P)
      JP=(GGP*X4-GG*X4P)+(FFP*X5-FF*X5P)+(EEP*X6-EE*X6P)+(DDP*X7-DD*X7P)
     +  +(CCP*X8-CC*X8P)+(BBP*X9-BB*X9P)+(AAP*X10-AA*X10P)
      KP=(GGP*X5-GG*X5P)+(FFP*X6-FF*X6P)+(EEP*X7-EE*X7P)+(DDP*X8-DD*X8P)
     +  +(CCP*X9-CC*X9P)+(BBP*X10-BB*X10P)
      LP=(GGP*X6-GG*X6P)+(FFP*X7-FF*X7P)+(EEP*X8-EE*X8P)+(DDP*X9-DD*X9P)
     +  +(CCP*X10-CC*X10P)
      MP=(GGP*X7-GG*X7P)+(FFP*X8-FF*X8P)+(EEP*X9-EE*X9P)+
     +   (DDP*X10-DD*X10P)
      NP=(GGP*X8-GG*X8P)+(FFP*X9-FF*X9P)+(EEP*X10-EE*X10P)
      OP=(GGP*X9-GG*X9P)+(FFP*X10-FF*X10P)
      PP=(GGP*X10-GG*X10P)
      T1=(Y1P*X1)
      T2=(Y2P*X1)+(Y1P*X2)
      T3=(Y3P*X1)+(Y2P*X2)+(Y1P*X3)
      Z1=(HHP*X1-HH*X1P)+(Y3P*X2)+(Y2P*X3)+(Y1P*X4)
      Z2=(IIP*X1-II*X1P)+(HHP*X2-HH*X2P)+(Y3P*X3)+(Y2P*X4)+(Y1P*X5)
      Z3=(JJP*X1-JJ*X1P)+(IIP*X2-II*X2P)+(HHP*X3-HH*X3P)+
     +   (Y3P*X4)+(Y2P*X5)+(Y1P*X6)
      Z4=(KKP*X1-KK*X1P)+(JJP*X2-JJ*X2P)+(IIP*X3-II*X3P)+(HHP*X4-HH*X4P)
     +  +(Y3P*X5)+(Y2P*X6)+(Y1P*X7)
      Z5=(LLP*X1-LL*X1P)+(KKP*X2-KK*X2P)+(JJP*X3-JJ*X3P)+(IIP*X4-II*X4P)
     +  +(HHP*X5-HH*X5P)+(Y3P*X6)+(Y2P*X7)+(Y1P*X8)
      Z6=(MMP*X1-MM*X1P)+(LLP*X2-LL*X2P)+(KKP*X3-KK*X3P)+(JJP*X4-JJ*X4P)
     +  +(IIP*X5-II*X5P)+(HHP*X6-HH*X6P)+(Y3P*X7)+(Y2P*X8)+(Y1P*X9)
      Z7=(NNP*X1-NN*X1P)+(MMP*X2-MM*X2P)+(LLP*X3-LL*X3P)+
     +    (KKP*X4-KK*X4P)+(JJP*X5-JJ*X5P)+(IIP*X6-II*X6P)+
     +    (HHP*X7-HH*X7P)+(Y3P*X8)+(Y2P*X9)+(Y1P*X10)
      Z8=(NNP*X2-NN*X2P)+(MMP*X3-MM*X3P)+(LLP*X4-LL*X4P)+
     +    (KKP*X5-KK*X5P)+(JJP*X6-JJ*X6P)+(IIP*X7-II*X7P)+
     +    (HHP*X8-HH*X8P)+(Y3P*X9)+(Y2P*X10)
      Z9=(NNP*X3-NN*X3P)+(MMP*X4-MM*X4P)+(LLP*X5-LL*X5P)+
     +    (KKP*X6-KK*X6P)+(JJP*X7-JJ*X7P)+(IIP*X8-II*X8P)+
     +    (HHP*X9-HH*X9P)+(Y3P*X10)
      Z10=(NNP*X4-NN*X4P)+(MMP*X5-MM*X5P)+(LLP*X6-LL*X6P)+
     +    (KKP*X7-KK*X7P)+(JJP*X8-JJ*X8P)+(IIP*X9-II*X9P)+
     +    (HHP*X10-HH*X10P)
      Z11=(NNP*X5-NN*X5P)+(MMP*X6-MM*X6P)+(LLP*X7-LL*X7P)+
     +    (KKP*X8-KK*X8P)+(JJP*X9-JJ*X9P)+(IIP*X10-II*X10P)
      Z12=(NNP*X6-NN*X6P)+(MMP*X7-MM*X7P)+(LLP*X8-LL*X8P)+
     +    (KKP*X9-KK*X9P)+(JJP*X10-JJ*X10P)
      Z13=(NNP*X7-NN*X7P)+(MMP*X8-MM*X8P)+(LLP*X9-LL*X9P)+
     +    (KKP*X10-KK*X10P)
      Z14=(NNP*X8-NN*X8P)+(MMP*X9-MM*X9P)+(LLP*X10-LL*X10P)
      Z15=(NNP*X9-NN*X9P)+(MMP*X10-MM*X10P)
```

```
      Z16=(NNP*X10-NN*X10P)

C FINAL STEP

      W(34)=T1*A
      W(33)=T1*B+T2*A
      W(32)=T1*C+T2*B+T3*A
      W(31)=T1*D+T2*C+T3*B+(Z1*A-S1*AP)
      W(30)=T1*E+T2*D+T3*C+(Z1*B-S1*BP)+(Z2*A-S2*AP)
      W(29)=T1*F+T2*E+T3*D+(Z1*C-S1*CP)+(Z2*B-S2*BP)+(Z3*A-S3*AP)
      W(28)=T1*G+T2*F+T3*E+(Z1*D-S1*DP)+(Z2*C-S2*CP)+(Z3*B-S3*BP)+
     +      (Z4*A-S4*AP)
      W(27)=T1*H+T2*G+T3*F+(Z1*E-S1*EP)+(Z2*D-S2*DP)+(Z3*C-S3*CP)+
     +      (Z4*B-S4*BP)+(Z5*A-S5*AP)
      W(26)=T1*I+T2*H+T3*G+(Z1*F-S1*FP)+(Z2*E-S2*EP)+(Z3*D-S3*DP)+
     +      (Z4*C-S4*CP)+(Z5*B-S5*BP)+(Z6*A-S6*AP)
      W(25)=T1*J+T2*I+T3*H+(Z1*G-S1*GP)+(Z2*F-S2*FP)+(Z3*E-S3*EP)+
     +      (Z4*D-S4*DP)+(Z5*C-S5*CP)+(Z6*B-S6*BP)+(Z7*A-S7*AP)
      W(24)=T1*K+T2*J+T3*I+(Z1*H-S1*HP)+(Z2*G-S2*GP)+(Z3*F-S3*FP)+
     +      (Z4*E-S4*EP)+(Z5*D-S5*DP)+(Z6*C-S6*CP)+(Z7*B-S7*BP)+
     +      (Z8*A-S8*AP)
      W(23)=T1*L+T2*K+T3*J+(Z1*I-S1*IP)+(Z2*H-S2*HP)+(Z3*G-S3*GP)+
     +      (Z4*F-S4*FP)+(Z5*E-S5*EP)+(Z6*D-S6*DP)+(Z7*C-S7*CP)+
     +      (Z8*B-S8*BP)+(Z9*A-S9*AP)
      W(22)=T1*M+T2*L+T3*K+(Z1*J-S1*JP)+(Z2*I-S2*IP)+(Z3*H-S3*HP)+
     +      (Z4*G-S4*GP)+(Z5*F-S5*FP)+(Z6*E-S6*EP)+(Z7*D-S7*DP)+
     +      (Z8*C-S8*CP)+(Z9*B-S9*BP)+(Z10*A-S10*AP)
      W(21)=T1*N+T2*M+T3*L+(Z1*K-S1*KP)+(Z2*J-S2*JP)+(Z3*I-S3*IP)+
     +      (Z4*H-S4*HP)+(Z5*G-S5*GP)+(Z6*F-S6*FP)+(Z7*E-S7*EP)+
     +      (Z8*D-S8*DP)+(Z9*C-S9*CP)+(Z10*B-S10*BP)+(Z11*A-S11*AP)
      W(20)=T1*O+T2*N+T3*M+(Z1*L-S1*LP)+(Z2*K-S2*KP)+(Z3*J-S3*JP)+
     +      (Z4*I-S4*IP)+(Z5*H-S5*HP)+(Z6*G-S6*GP)+(Z7*F-S7*FP)+
     +      (Z8*E-S8*EP)+(Z9*D-S9*DP)+(Z10*C-S10*CP)+(Z11*B-S11*BP)+
     +      (Z12*A-S12*AP)
      W(19)=T1*P+T2*O+T3*N+(Z1*M-S1*MP)+(Z2*L-S2*LP)+(Z3*K-S3*KP)+
     +      (Z4*J-S4*JP)+(Z5*I-S5*IP)+(Z6*H-S6*HP)+(Z7*G-S7*GP)+
     +      (Z8*F-S8*FP)+(Z9*E-S9*EP)+(Z10*D-S10*DP)+(Z11*C-S11*CP)+
     +      (Z12*B-S12*BP)+(Z13*A-S13*AP)
      W(18)=T2*P+T3*O+(Z1*N-S1*NP)+(Z2*M-S2*MP)+(Z3*L-S3*LP)+
     +      (Z4*K-S4*KP)+(Z5*J-S5*JP)+(Z6*I-S6*IP)+(Z7*H-S7*HP)+
     +      (Z8*G-S8*GP)+(Z9*F-S9*FP)+(Z10*E-S10*EP)+(Z11*D-S11*DP)+
     +      (Z12*C-S12*CP)+(Z13*B-S13*BP)+(Z14*A-S14*AP)
      W(17)=T3*P+(Z1*O-S1*OP)+(Z2*N-S2*NP)+(Z3*M-S3*MP)+
     +      (Z4*L-S4*LP)+(Z5*K-S5*KP)+(Z6*J-S6*JP)+(Z7*I-S7*IP)+
     +      (Z8*H-S8*HP)+(Z9*G-S9*GP)+(Z10*F-S10*FP)+(Z11*E-S11*EP)+
     +      (Z12*D-S12*DP)+(Z13*C-S13*CP)+(Z14*B-S14*BP)+(Z15*A-S15*AP)
      W(16)=(Z1*P-S1*PP)+(Z2*O-S2*OP)+(Z3*N-S3*NP)+(Z4*M-S4*MP)+
     +      (Z5*L-S5*LP)+(Z6*K-S6*KP)+(Z7*J-S7*JP)+(Z8*I-S8*IP)+
     +      (Z9*H-S9*HP)+(Z10*G-S10*GP)+(Z11*F-S11*FP)+(Z12*E-S12*EP)+
     +      (Z13*D-S13*DP)+(Z14*C-S14*CP)+(Z15*B-S15*BP)+(Z16*A-S16*AP)
      W(15)=(Z2*P-S2*PP)+(Z3*O-S3*OP)+(Z4*N-S4*NP)+(Z5*M-S5*MP)+
     +      (Z6*L-S6*LP)+(Z7*K-S7*KP)+(Z8*J-S8*JP)+(Z9*I-S9*IP)+
     +      (Z10*H-S10*HP)+(Z11*G-S11*GP)+(Z12*F-S12*FP)+(Z13*E-S13*EP)+
     +      (Z14*D-S14*DP)+(Z15*C-S15*CP)+(Z16*B-S16*BP)
      W(14)=(Z3*P-S3*PP)+(Z4*O-S4*OP)+(Z5*N-S5*NP)+(Z6*M-S6*MP)+
     +      (Z7*L-S7*LP)+(Z8*K-S8*KP)+(Z9*J-S9*JP)+(Z10*I-S10*IP)+
     +      (Z11*H-S11*HP)+(Z12*G-S12*GP)+(Z13*F-S13*FP)+(Z14*E-S14*EP)+
     +      (Z15*D-S15*DP)+(Z16*C-S16*CP)
      W(13)=(Z4*P-S4*PP)+(Z5*O-S5*OP)+(Z6*N-S6*NP)+(Z7*M-S7*MP)+
     +      (Z8*L-S8*LP)+(Z9*K-S9*KP)+(Z10*J-S10*JP)+(Z11*I-S11*IP)+
     +      (Z12*H-S12*HP)+(Z13*G-S13*GP)+(Z14*F-S14*FP)+(Z15*E-S15*EP)+
     +      (Z16*D-S16*DP)
      W(12)=(Z5*P-S5*PP)+(Z6*O-S6*OP)+(Z7*N-S7*NP)+(Z8*M-S8*MP)+
     +      (Z9*L-S9*LP)+(Z10*K-S10*KP)+(Z11*J-S11*JP)+(Z12*I-S12*IP)+
     +      (Z13*H-S13*HP)+(Z14*G-S14*GP)+(Z15*F-S15*FP)+(Z16*E-S16*EP)
      W(11)=(Z6*P-S6*PP)+(Z7*O-S7*OP)+(Z8*N-S8*NP)+(Z9*M-S9*MP)+
```

```
     +      (Z10*L-S10*LP)+(Z11*K-S11*KP)+(Z12*J-S12*JP)+(Z13*I-S13*IP)+
     +      (Z14*H-S14*HP)+(Z15*G-S15*GP)+(Z16*F-S16*FP)
      W(10)=(Z7*P-S7*PP)+(Z8*O-S8*OP)+(Z9*N-S9*NP)+(Z10*M-S10*MP)+
     +      (Z11*L-S11*LP)+(Z12*K-S12*KP)+(Z13*J-S13*JP)+(Z14*I-S14*IP)+
     +      (Z15*H-S15*HP)+(Z16*G-S16*GP)
      W(9)=(Z8*P-S8*PP)+(Z9*O-S9*OP)+(Z10*N-S10*NP)+(Z11*M-S11*MP)+
     +      (Z12*L-S12*LP)+(Z13*K-S13*KP)+(Z14*J-S14*JP)+(Z15*I-S15*IP)+
     +      (Z16*H-S16*HP)
      W(8)=(Z9*P-S9*PP)+(Z10*O-S10*OP)+(Z11*N-S11*NP)+(Z12*M-S12*MP)+
     +      (Z13*L-S13*LP)+(Z14*K-S14*KP)+(Z15*J-S15*JP)+(Z16*I-S16*IP)
      W(7)=(Z10*P-S10*PP)+(Z11*O-S11*OP)+(Z12*N-S12*NP)+(Z13*M-S13*MP)+
     +      (Z14*L-S14*LP)+(Z15*K-S15*KP)+(Z16*J-S16*JP)
      W(6)=(Z11*P-S11*PP)+(Z12*O-S12*OP)+(Z13*N-S13*NP)+(Z14*M-S14*MP)+
     +      (Z15*L-S15*LP)+(Z16*K-S16*KP)
      W(5)=(Z12*P-S12*PP)+(Z13*O-S13*OP)+(Z14*N-S14*NP)+(Z15*M-S15*MP)+
     +      (Z16*L-S16*LP)
      W(4)=(Z13*P-S13*PP)+(Z14*O-S14*OP)+(Z15*N-S15*NP)+(Z16*M-S16*MP)
      W(3)=(Z14*P-S14*PP)+(Z15*O-S15*OP)+(Z16*N-S16*NP)
      W(2)=(Z15*P-S15*PP)+(Z16*O-S16*OP)
      W(1)=(Z16*P-S16*PP)


      RETURN
      END
C-----------------------------------------------------
C
C     SUBROUTINE: SBRP33
C
C     DESCRIPTION: SOLVE FOR THE VARIABLES BY
C                  BACK SUBSTITUTION
C
C INPUT:
C     EQ1,EQ2       = INTERSECTION EQS. WITH TWO UNKNOWNS
C     X3,Y3,Z3      = SURFACE 1 EQS.
C     XA,XB,YA,
C     YB,ZA,ZB      = SURFACE 2 EQS.
C     W1,W2         = PARAMETRIC VALUES
C
C OUTPUT:
C     UA1,WA1,
C     UA2,WA2       = INTERSECTION POINTS IN PARAMETRIC SPACE
C     IFOUND        = NO. OF INTERSECTION POINTS
C
C     C. K. WONG
C     5/16/90
C


      SUBROUTINE SBRP33(EQ1,EQ2,X3,Y3,Z3,XA,XB,YA,YB,ZA,ZB,
     +                  W1,W2,UA1,WA1,UA2,WA2,IFOUND)

      REAL EQ1(19),EQ2(19),COF(4),C(4),Y(4),Q,RO(3)
     +     ,X3(4),Y3(4),Z3(4)
     +     ,XA(4),XB(4),YA(4),YB(4),ZA(4),ZB(4)
     +     ,UA1(*),UA2(*),WA2(*),WA1(*)

      INTEGER M

      EXTERNAL ZPLRC

C TOLERANCE
      TLIM=.001

C SOLVE FOR THE CUBIC VARIABLE ON SURFACE 1 (U1):

      A1=W2**3
```

```
      A2=W2**2

      C(1)=(EQ1(1)*A1)+(EQ1(2)*A2)+(EQ1(3)*W2)+EQ1(4)
      C(2)=(EQ1(5)*A1)+(EQ1(6)*A2)+(EQ1(7)*W2)+EQ1(8)
      C(3)=(EQ1(9)*A1)+(EQ1(10)*A2)+(EQ1(11)*W2)+EQ1(12)
      C(4)=(EQ1(13)*(W2**6))+(EQ1(14)*(W2**5))+(EQ1(15)*(W2**4))+
     +     (EQ1(16)*A1)+(EQ1(17)*A2)+(EQ1(18)*W2)+EQ1(19)

      Y(1)=(EQ2(1)*A1)+(EQ2(2)*A2)+(EQ2(3)*W2)+EQ2(4)
      Y(2)=(EQ2(5)*A1)+(EQ2(6)*A2)+(EQ2(7)*W2)+EQ2(8)
      Y(3)=(EQ2(9)*A1)+(EQ2(10)*A2)+(EQ2(11)*W2)+EQ2(12)
      Y(4)=(EQ2(13)*(W2**6))+(EQ2(14)*(W2**5))+(EQ2(15)*(W2**4))+
     +     (EQ2(16)*A1)+(EQ2(17)*A2)+(EQ2(18)*W2)+EQ2(19)

      DO 105 I=1,4
       COF(I)=C(I)+Y(I)
 105  CONTINUE

      CALL CUROOT(COF,IR,RO)


      DO 650 M=1,IR

         U1=RO(M)


C SOLVE FOR LINEAR VARIABLE ON SURFACE 2 (U2):

              CALL EVAFUN(XA,W2,DIV1)
              CALL EVAFUN(YA,W2,DIV2)
              CALL EVAFUN(ZA,W2,DIV3)

C PICK THE LARGEST POSSIBLE DENOMINATOR FOR
C BACK SUBSTITUTION

              MAX=ABS(DIV1)
              ITRY=1
              IF (ABS(DIV2).GT.MAX) THEN
                   MAX=ABS(DIV2)
                   ITRY=2
              ENDIF
              IF (ABS(DIV3).GT.MAX) THEN
                   ITRY=3
              ENDIF


          IF(ITRY.EQ.1) THEN
              CALL EVAFUN(X3,U1,F1)
              CALL EVAFUN(XB,W2,F2)
              U2=(F1-F2)/DIV1
           ELSEIF(ITRY.EQ.2) THEN
              CALL EVAFUN(Y3,U1,F1)
              CALL EVAFUN(YB,W2,F2)
              U2=(F1-F2)/DIV2
           ELSEIF(ITRY.EQ.3) THEN
              CALL EVAFUN(Z3,U1,F1)
              CALL EVAFUN(ZB,W2,F2)
              U2=(F1-F2)/DIV3
          ENDIF

C CHECK IF OUT OF RANGE

         IF((U2.GT.(1.+TLIM)).OR.
     +      (U2.LT.-(TLIM))) GOTO 650

              UA2(IFOUND)=U2
              UA1(IFOUND)=U1
```

```
            WA2(IFOUND)=W2
            WA1(IFOUND)=W1

         IFOUND=IFOUND+1

 650  CONTINUE



      RETURN
      END
C---------------------------------------------
C
C     SUBROUTINE: BRINDRR
C
C     DESCRIPTION: FIND THE INTERSECTION CURVE OF TWO
C                  B-SPLINE RULED SURFACE (DOUBLE PRECISION
C                  VERSION OF BRINTRR)
C
C INPUT:
C     TXYZ1,TXYZ2  = SURFACES COEFF.
C     PKL1,PKL2    = INTERSECTING SURFACES
C     IUW          = LINEAR OR CUBIC VARIABLE FLAG
C
C OUTPUT:
C     TTUW         = INTERSECTION POINT IN PARAMETRIC SPACE
C
C     C. K. WONG
C     2/12/90
C

      SUBROUTINE BRINDRR(TXYZ1,TXYZ2,PKL1,PKL2,IUW,TTUW)

      REAL PKL1(0:3,0:3,3),PKL2(0:3,0:3,3),TXYZ1(3,8),TXYZ2(3,8),
     +     TTUW(8,4)
     +     ,TUW(4),TUW2(4)

      INTEGER IUW(2),IUW2(2)

C SET THE COUNTER
          II=0

C DEFINE CUBIC VARIABLE ON SURFACE 1

          V=0.0
          CALL BRP9(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,TUW)
          CALL PUTUW1(II,TUW,TTUW)

          V=1.0
          CALL BRP9(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,TUW)
          CALL PUTUW1(II,TUW,TTUW)

C DEFINE CUBIC VARIABLE ON SURFACE 2


          IUW2(1)=IUW(2)
          IUW2(2)=IUW(1)

          V=0.0
          CALL BRP9(V,TXYZ2,TXYZ1,PKL2,PKL1,IUW2,TUW)
          TUW2(1)=TUW(3)
          TUW2(2)=TUW(4)
          TUW2(3)=TUW(1)
          TUW2(4)=TUW(2)
          CALL PUTUW1(II,TUW2,TTUW)
```

```
          V=1.0
          CALL BRP9(V,TXYZ2,TXYZ1,PKL2,PKL1,IUW2,TUW)
          TUW2(1)=TUW(3)
          TUW2(2)=TUW(4)
          TUW2(3)=TUW(1)
          TUW2(4)=TUW(2)
          CALL PUTUW1(II,TUW2,TTUW)

C DEFINE LINEAR VARIABLE ON SURFACE 1

          V=0.0
          CALL BRP33D(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,TUW)
          CALL PUTUW1(II,TUW,TTUW)

          V=1.0
          CALL BRP33D(V,TXYZ1,TXYZ2,PKL1,PKL2,IUW,TUW)
          CALL PUTUW1(II,TUW,TTUW)


C DEFINE LINEAR VARIABLE ON SURFACE 2

          IUW2(1)=IUW(2)
          IUW2(2)=IUW(1)

          V=0.0
          CALL BRP33D(V,TXYZ2,TXYZ1,PKL2,PKL1,IUW2,TUW)

          TUW2(1)=TUW(3)
          TUW2(2)=TUW(4)
          TUW2(3)=TUW(1)
          TUW2(4)=TUW(2)
          CALL PUTUW1(II,TUW2,TTUW)

          V=1.0
          CALL BRP33D(V,TXYZ2,TXYZ1,PKL2,PKL1,IUW2,TUW)

          TUW2(1)=TUW(3)
          TUW2(2)=TUW(4)
          TUW2(3)=TUW(1)
          TUW2(4)=TUW(2)
          CALL PUTUW1(II,TUW2,TTUW)

C FILL UP TTUW

          DO 122 IIH=II+1,8
            TTUW(IIH,1)=2.
            TTUW(IIH,2)=2.
            TTUW(IIH,3)=2.
            TTUW(IIH,4)=2.
 122      CONTINUE


      RETURN
      END
C-----------------------------------------------------
C
C     SUBROUTINE: BRP33D
C
C     DESCRIPTION: DEFINE THE LINEAR VARIABLE ON SURFACE 1
C                  IN THE ELIMINATION ROUTINE (DOUBLE PRECISION
C                  VERSION OF BRP33A)
C
C INPUT:
C     V            = PARAMETRIC VALUE
C     BRC1,BRC2    = SURFACES COEFF.
```

```
C       PKL1,PKL2       = INTERSECTING SURFACES
C       IUW             = LINEAR OR CUBIC VARIABLE FLAG
C
C OUTPUT:
C       TUW             = INTERSECTION POINT IN PARAMETRIC SPACE
C
C       C. K. WONG
C       2/12/90
C

      SUBROUTINE BRP33D(V,BRC1,BRC2,PKL1,PKL2,IUW,TUW)

      REAL BRC1(3,8),BRC2(3,8),
     +     TUW(4),EQ1(19),EQ2(19),
     +     XACOF(4),XBCOF(4),YACOF(4),YBCOF(4),ZACOF(4),ZBCOF(4),
     +     X3COF(4),Y3COF(4),Z3COF(4)
     +     ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER IUW(2)

C DEFINE THE LINEAR VARIABLE ON SURFACE 1, AND REWRITE
C THE SURFACE EQS.

      DO 35 JJ=1,4
         X3COF(JJ)=(V*BRC1(1,JJ))+BRC1(1,JJ+4)
         Y3COF(JJ)=(V*BRC1(2,JJ))+BRC1(2,JJ+4)
         Z3COF(JJ)=(V*BRC1(3,JJ))+BRC1(3,JJ+4)
         XACOF(JJ)=BRC2(1,JJ)
         XBCOF(JJ)=BRC2(1,JJ+4)
         YACOF(JJ)=BRC2(2,JJ)
         YBCOF(JJ)=BRC2(2,JJ+4)
         ZACOF(JJ)=BRC2(3,JJ)
         ZBCOF(JJ)=BRC2(3,JJ+4)
 35   CONTINUE


C CREATE THE EQUATIONS EQ1 & EQ2 BY ELIMINATING
C ONE VARIABLE FROM THE THREE PARAMETRIC SURFACE
C EQS.

      CALL BRPCREQ(X3COF,Y3COF,Z3COF,XACOF,XBCOF,
     +             YACOF,YBCOF,ZACOF,ZBCOF,EQ1,EQ2)


C SOLVE FOR THE REST OF THE VARIABLES

      CALL BRPSLVD(PKL1,PKL2,EQ1,EQ2,X3COF,Y3COF,Z3COF,XACOF,XBCOF,
     +             YACOF,YBCOF,ZACOF,ZBCOF,V,IUW,TUW)


      RETURN
      END
C--------------------------------------------------
C
C       SUBROUTINE: BRPSLVD
C
C       DESCRIPTION: SOLVE FOR THE VARIABLES USING
C                    ELIMINATION METHOD TO GENERATE A
C                    33RD ORDER POLYNOMIAL (DOUBLE
C                    PRECISION VERSION OF BRPSLVA)
C
C INPUT:
C       PKL1,PKL2       = INTERSECTING SURFACES
C       EQ1,EQ2         = INTERSECTION EQS. WITH TWO UNKNOWNS
C       X3,Y3,Z3        = SURFACE 1 EQS.
C       XA,XB,YA,
C       YB,ZA,ZB        = SURFACE 2 EQS.
```

```
C      W1             = PARAMETRIC VALUE
C      IUW            = LINEAR OR CUBIC VARIABLE FLAG
C
C OUTPUT:
C      TUW            = INTERSECTION POINT IN PARAMETRIC SPACE
C
C      C. K. WONG
C      9/12/89
C

      SUBROUTINE BRPSLVD(PKL1,PKL2,EQ1,EQ2,X3,Y3,Z3,XA,XB,
     +           YA,YB,ZA,ZB,W1,IUW,TUW)

      DOUBLE PRECISION E33(34)
      REAL TUW(4),EQ1(19),EQ2(19),
     +     XA(4),XB(4),YA(4),YB(4),ZA(4),ZB(4),
     +     X3(4),Y3(4),Z3(4)
     +     ,WW(100)
     +     ,MAX,MIN
     +     ,WA1(60),WA2(60),UA1(60),UA2(60)
     +     ,PKL1(0:3,0:3,3),PKL2(0:3,0:3,3)

      INTEGER IUW(2),M
      INTEGER NDEG
      PARAMETER(NDEG=33)
      COMPLEX ZERO(33)
      EXTERNAL ZPLRC,ZPORC


C     SET TOLERANCE
      TLIM=.001

C ELIMINATE ONE UNKNOWN FROM EQ1 & EQ2 TO YIELD
C A 33RD ORDER POLYNOMIAL

      CALL BRP33ED(EQ1,EQ2,E33)

C SOLVE THE 33RD POLYNOMIAL, AND OBTAIN THE
C CUBIC VARIABLE ON SURFACE 2

      IG=34
      CALL BISECTG(IG,E33,WW,INROOT)

      IFOUND=1
      DO 650 M=1,INROOT

         W2=WW(M)
C IS THE VARIABLE OUT OF RANGE?

         IF((W2.GT.(1.+TLIM)).OR.
     +      (W2.LT.-(TLIM))) GOTO 650

C BACK SUBSTITUTING TO SOLVE FOR THE REST OF
C THE VARIABLES

         CALL SBRP33(EQ1,EQ2,X3,Y3,Z3,
     +               XA,XB,YA,YB,ZA,ZB
     +              ,W1,W2,UA1,WA1,UA2,WA2,IFOUND)



 650  CONTINUE

      IFOUND=IFOUND-1

C NO INTERSECTION
```

```
         IF(IFOUND.EQ.0) THEN
              DO 240 I=1,4
                  TUW(I)=2.
 240          CONTINUE

            ELSE

C SELECT THE RESULT BY COMPARING ALL THE RESULTS,
C AND PICK THE PAIR OF POINTS THAT HAS THE
C SMALLEST DISTANCE BETWEEN THEM

         IF((IUW(1).EQ.0).AND.(IUW(2).EQ.1)) THEN
              CALL COMRELT(PKL1,PKL2,UA1,WA1,UA2,WA2,IFOUND,TUW)
           ELSEIF((IUW(1).EQ.1).AND.(IUW(2).EQ.1)) THEN
              CALL COMRELT(PKL1,PKL2,WA1,UA1,UA2,WA2,IFOUND,TUW)
           ELSEIF((IUW(1).EQ.0).AND.(IUW(2).EQ.0)) THEN
              CALL COMRELT(PKL1,PKL2,UA1,WA1,WA2,UA2,IFOUND,TUW)
           ELSEIF((IUW(1).EQ.1).AND.(IUW(2).EQ.0)) THEN
              CALL COMRELT(PKL1,PKL2,WA1,UA1,WA2,UA2,IFOUND,TUW)
         ENDIF

         ENDIF


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: BRP33ED
C
C     DESCRIPTION: ELIMINATE ONE UNKNOWN FROM TWO
C                  ORDER FUNCTIONS TO YIELD A 33RD
C                  ORDER POLYNOMIAL (DOUBLE PRECISION
C                  VERSION OF BRP33)
C
C INPUT:
C     EQ1,EQ2      = TWO FUNCTIONS WITH TWO UNKNOWNS
C
C OUTPUT:
C     W            = 33RD ORDER POLYNOMIAL
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE BRP33ED(EQ1,EQ2,W)

      REAL EQ1(19),EQ2(19)
      DOUBLE PRECISION W(34)

      DOUBLE PRECISION A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,
     +     AP,BP,CP,DP,EP,FP,GP,HP,IP,JP,KP,LP,MP,NP,OP,PP,
     +     QP,RP,SP,
     +     AA,BB,CC,DD,EE,FF,GG,HH,II,JJ,KK,LL,MM,NN,
     +     X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,
     +     AAP,BBP,CCP,DDP,EEP,FFP,GGP,HHP,IIP,JJP,KKP,LLP,
     +     MMP,NNP,
     +     X1P,X2P,X3P,X4P,X5P,X6P,X7P,X8P,X9P,X10P,
     +     Y1P,Y2P,Y3P,
     +     S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,
     +     S11,S12,S13,S14,S15,S16,
     +     T1,T2,T3,
     +     Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8,Z9,Z10,
     +     Z11,Z12,Z13,Z14,Z15,Z16
```

```
C SET UP THE EQS.

      A=EQ1(1)
      B=EQ1(2)
      C=EQ1(3)
      D=EQ1(4)
      E=EQ1(5)
      F=EQ1(6)
      G=EQ1(7)
      H=EQ1(8)
      I=EQ1(9)
      J=EQ1(10)
      K=EQ1(11)
      L=EQ1(12)
      M=EQ1(13)
      N=EQ1(14)
      O=EQ1(15)
      P=EQ1(16)
      Q=EQ1(17)
      R=EQ1(18)
      S=EQ1(19)


      AP=EQ2(1)
      BP=EQ2(2)
      CP=EQ2(3)
      DP=EQ2(4)
      EP=EQ2(5)
      FP=EQ2(6)
      GP=EQ2(7)
      HP=EQ2(8)
      IP=EQ2(9)
      JP=EQ2(10)
      KP=EQ2(11)
      LP=EQ2(12)
      MP=EQ2(13)
      NP=EQ2(14)
      OP=EQ2(15)
      PP=EQ2(16)
      QP=EQ2(17)
      RP=EQ2(18)
      SP=EQ2(19)

C STEP 3

      AA=(A*EP-E*AP)
      BB=(A*FP-F*AP)+(B*EP-E*BP)
      CC=(A*GP-G*AP)+(B*FP-F*BP)+(C*EP-E*CP)
      DD=(A*HP-H*AP)+(B*GP-G*BP)+(C*FP-F*CP)+(D*EP-E*DP)
      EE=(B*HP-H*BP)+(C*GP-G*CP)+(D*FP-F*DP)
      FF=(C*HP-H*CP)+(D*GP-G*DP)
      GG=(D*HP-H*DP)
      HH=(A*IP-I*AP)
      II=(A*JP-J*AP)+(B*IP-I*BP)
      JJ=(A*KP-K*AP)+(B*JP-J*BP)+(C*IP-I*CP)
      KK=(A*LP-L*AP)+(B*KP-K*BP)+(C*JP-J*CP)+(D*IP-I*DP)
      LL=(B*LP-L*BP)+(C*KP-K*CP)+(D*JP-J*DP)
      MM=(C*LP-L*CP)+(D*KP-K*DP)
      NN=(D*LP-L*DP)
      X1=(A*MP-M*AP)
      X2=(A*NP-N*AP)+(B*MP-M*BP)
      X3=(A*OP-O*AP)+(B*NP-N*BP)+(C*MP-M*CP)
      X4=(A*PP-P*AP)+(B*OP-O*BP)+(C*NP-N*CP)+(D*MP-M*DP)
      X5=(A*QP-Q*AP)+(B*PP-P*BP)+(C*OP-O*CP)+(D*NP-N*DP)
      X6=(A*RP-R*AP)+(B*QP-Q*BP)+(C*PP-P*CP)+(D*OP-O*DP)
      X7=(A*SP-S*AP)+(B*RP-R*BP)+(C*QP-Q*CP)+(D*PP-P*DP)
      X8=(B*SP-S*BP)+(C*RP-R*CP)+(D*QP-Q*DP)
```

```
      X9=(C*SP-S*CP)+(D*RP-R*DP)
      X10=(D*SP-S*DP)

C STEP 4

      AAP=(A*IP-I*AP)
      BBP=(A*JP-J*AP)+(B*IP-I*BP)
      CCP=(A*KP-K*AP)+(B*JP-J*BP)+(C*IP-I*CP)
      DDP=(A*LP-L*AP)+(B*KP-K*BP)+(C*JP-J*CP)+(D*IP-I*DP)
      EEP=(B*LP-L*BP)+(C*KP-K*CP)+(D*JP-J*DP)
      FFP=(C*LP-L*CP)+(D*KP-K*DP)
      GGP=(D*LP-L*DP)
      Y1P=(A*MP-M*AP)
      Y2P=(A*NP-N*AP)+(B*MP-M*BP)
      Y3P=(A*OP-O*AP)+(B*NP-N*BP)+(C*MP-M*CP)
      HHP=(A*PP-P*AP)+(B*OP-O*BP)+(C*NP-N*CP)+(D*MP-M*DP)+(E*IP-I*EP)
      IIP=(A*QP-Q*AP)+(B*PP-P*BP)+(C*OP-O*CP)+(D*NP-N*DP)+(E*JP-J*EP)+
     +    (F*IP-I*FP)
      JJP=(A*RP-R*AP)+(B*QP-Q*BP)+(C*PP-P*CP)+(D*OP-O*DP)+(E*KP-K*EP)+
     +    (F*JP-J*FP)+(G*IP-I*GP)
      KKP=(A*SP-S*AP)+(B*RP-R*BP)+(C*QP-Q*CP)+(D*PP-P*DP)+(E*LP-L*EP)+
     +    (F*KP-K*FP)+(G*JP-J*GP)+(H*IP-I*HP)
      LLP=(B*SP-S*BP)+(C*RP-R*CP)+(D*QP-Q*DP)+(F*LP-L*FP)+(G*KP-K*GP)+
     +    (H*JP-J*HP)
      MMP=(C*SP-S*CP)+(D*RP-R*DP)+(G*LP-L*GP)+(H*KP-K*HP)
      NNP=(D*SP-S*DP)+(H*LP-L*HP)
      X1P=(E*MP-M*EP)
      X2P=(E*NP-N*EP)+(F*MP-M*FP)
      X3P=(E*OP-O*EP)+(F*NP-N*FP)+(G*MP-M*GP)
      X4P=(E*PP-P*EP)+(F*OP-O*FP)+(G*NP-N*GP)+(H*MP-M*HP)
      X5P=(E*QP-Q*EP)+(F*PP-P*FP)+(G*OP-O*GP)+(H*NP-N*HP)
      X6P=(E*RP-R*EP)+(F*QP-Q*FP)+(G*PP-P*GP)+(H*OP-O*HP)
      X7P=(E*SP-S*EP)+(F*RP-R*FP)+(G*QP-Q*GP)+(H*PP-P*HP)
      X8P=(F*SP-S*FP)+(G*RP-R*GP)+(H*QP-Q*HP)
      X9P=(G*SP-S*GP)+(H*RP-R*HP)
      X10P=(H*SP-S*HP)

C STEP 5

      A=(AA*Y1P)
      B=(BB*Y1P)+(AA*Y2P)
      C=(CC*Y1P)+(BB*Y2P)+(AA*Y3P)
      D=(DD*Y1P)+(CC*Y2P)+(BB*Y3P)+(AA*HHP-AAP*HH)
      E=(EE*Y1P)+(DD*Y2P)+(CC*Y3P)+(BB*HHP-BBP*HH)+(AA*IIP-AAP*II)
      F=(FF*Y1P)+(EE*Y2P)+(DD*Y3P)+(CC*HHP-CCP*HH)+(BB*IIP-BBP*II)+
     + (AA*JJP-AAP*JJ)
      G=(GG*Y1P)+(FF*Y2P)+(EE*Y3P)+(DD*HHP-DDP*HH)+(CC*IIP-CCP*II)+
     + (BB*JJP-BBP*JJ)+(AA*KKP-AAP*KK)
      H=(GG*Y2P)+(FF*Y3P)+(EE*HHP-EEP*HH)+(DD*IIP-DDP*II)+
     + (CC*JJP-CCP*JJ)+(BB*KKP-BBP*KK)+(AA*LLP-AAP*LL)
      I=(GG*Y3P)+(FF*HHP-FFP*HH)+(EE*IIP-EEP*II)+(DD*JJP-DDP*JJ)+
     + (CC*KKP-CCP*KK)+(BB*LLP-BBP*LL)+(AA*MMP-AAP*MM)
      J=(GG*HHP-GGP*HH)+(FF*IIP-FFP*II)+(EE*JJP-EEP*JJ)+(DD*KKP-DDP*KK)+
     + (CC*LLP-CCP*LL)+(BB*MMP-BBP*MM)+(AA*NNP-AAP*NN)
      K=(GG*IIP-GGP*II)+(FF*JJP-FFP*JJ)+(EE*KKP-EEP*KK)+(DD*LLP-DDP*LL)+
     + (CC*MMP-CCP*MM)+(BB*NNP-BBP*NN)
      L=(GG*JJP-GGP*JJ)+(FF*KKP-FFP*KK)+(EE*LLP-EEP*LL)+(DD*MMP-DDP*MM)+
     + (CC*NNP-CCP*NN)
      M=(GG*KKP-GGP*KK)+(FF*LLP-FFP*LL)+(EE*MMP-EEP*MM)+(DD*NNP-DDP*NN)
      N=(GG*LLP-GGP*LL)+(FF*MMP-FFP*MM)+(EE*NNP-EEP*NN)
      O=(GG*MMP-GGP*MM)+(FF*NNP-FFP*NN)
      P=(GG*NNP-GGP*NN)
      S1=(AA*X1P-AAP*X1)
      S2=(BB*X1P-BBP*X1)+(AA*X2P-AAP*X2)
      S3=(CC*X1P-CCP*X1)+(BB*X2P-BBP*X2)+(AA*X3P-AAP*X3)
      S4=(DD*X1P-DDP*X1)+(CC*X2P-CCP*X2)+(BB*X3P-BBP*X3)+(AA*X4P-AAP*X4)
      S5=(EE*X1P-EEP*X1)+(DD*X2P-DDP*X2)+(CC*X3P-CCP*X3)+(BB*X4P-BBP*X4)
```

```
+  +(AA*X5P-AAP*X5)
 S6=(FF*X1P-FFP*X1)+(EE*X2P-EEP*X2)+(DD*X3P-DDP*X3)+(CC*X4P-CCP*X4)
+  +(BB*X5P-BBP*X5)+(AA*X6P-AAP*X6)
 S7=(GG*X1P-GGP*X1)+(FF*X2P-FFP*X2)+(EE*X3P-EEP*X3)+(DD*X4P-DDP*X4)
+  +(CC*X5P-CCP*X5)+(BB*X6P-BBP*X6)+(AA*X7P-AAP*X7)
 S8=(GG*X2P-GGP*X2)+(FF*X3P-FFP*X3)+(EE*X4P-EEP*X4)+(DD*X5P-DDP*X5)
+  +(CC*X6P-CCP*X6)+(BB*X7P-BBP*X7)+(AA*X8P-AAP*X8)
 S9=(GG*X3P-GGP*X3)+(FF*X4P-FFP*X4)+(EE*X5P-EEP*X5)+(DD*X6P-DDP*X6)
+  +(CC*X7P-CCP*X7)+(BB*X8P-BBP*X8)+(AA*X9P-AAP*X9)
 S10=(GG*X4P-GGP*X4)+(FF*X5P-FFP*X5)+(EE*X6P-EEP*X6)+
+ (DD*X7P-DDP*X7)+(CC*X8P-CCP*X8)+(BB*X9P-BBP*X9)+(AA*X10P-AAP*X10)
 S11=(GG*X5P-GGP*X5)+(FF*X6P-FFP*X6)+(EE*X7P-EEP*X7)+
+    (DD*X8P-DDP*X8)+(CC*X9P-CCP*X9)+(BB*X10P-BBP*X10)
 S12=(GG*X6P-GGP*X6)+(FF*X7P-FFP*X7)+(EE*X8P-EEP*X8)+
+    (DD*X9P-DDP*X9)+(CC*X10P-CCP*X10)
 S13=(GG*X7P-GGP*X7)+(FF*X8P-FFP*X8)+(EE*X9P-EEP*X9)+
+    (DD*X10P-DDP*X10)
 S14=(GG*X8P-GGP*X8)+(FF*X9P-FFP*X9)+(EE*X10P-EEP*X10)
 S15=(GG*X9P-GGP*X9)+(FF*X10P-FFP*X10)
 S16=(GG*X10P-GGP*X10)


C STEP 6

 AP=(AAP*X1-AA*X1P)
 BP=(BBP*X1-BB*X1P)+(AAP*X2-AA*X2P)
 CP=(CCP*X1-CC*X1P)+(BBP*X2-BB*X2P)+(AAP*X3-AA*X3P)
 DP=(DDP*X1-DD*X1P)+(CCP*X2-CC*X2P)+(BBP*X3-BB*X3P)+(AAP*X4-AA*X4P)
 EP=(EEP*X1-EE*X1P)+(DDP*X2-DD*X2P)+(CCP*X3-CC*X3P)+(BBP*X4-BB*X4P)
+  +(AAP*X5-AA*X5P)
 FP=(FFP*X1-FF*X1P)+(EEP*X2-EE*X2P)+(DDP*X3-DD*X3P)+(CCP*X4-CC*X4P)
+  +(BBP*X5-BB*X5P)+(AAP*X6-AA*X6P)
 GP=(GGP*X1-GG*X1P)+(FFP*X2-FF*X2P)+(EEP*X3-EE*X3P)+(DDP*X4-DD*X4P)
+  +(CCP*X5-CC*X5P)+(BBP*X6-BB*X6P)+(AAP*X7-AA*X7P)
 HP=(GGP*X2-GG*X2P)+(FFP*X3-FF*X3P)+(EEP*X4-EE*X4P)+(DDP*X5-DD*X5P)
+  +(CCP*X6-CC*X6P)+(BBP*X7-BB*X7P)+(AAP*X8-AA*X8P)
 IP=(GGP*X3-GG*X3P)+(FFP*X4-FF*X4P)+(EEP*X5-EE*X5P)+(DDP*X6-DD*X6P)
+  +(CCP*X7-CC*X7P)+(BBP*X8-BB*X8P)+(AAP*X9-AA*X9P)
 JP=(GGP*X4-GG*X4P)+(FFP*X5-FF*X5P)+(EEP*X6-EE*X6P)+(DDP*X7-DD*X7P)
+  +(CCP*X8-CC*X8P)+(BBP*X9-BB*X9P)+(AAP*X10-AA*X10P)
 KP=(GGP*X5-GG*X5P)+(FFP*X6-FF*X6P)+(EEP*X7-EE*X7P)+(DDP*X8-DD*X8P)
+  +(CCP*X9-CC*X9P)+(BBP*X10-BB*X10P)
 LP=(GGP*X6-GG*X6P)+(FFP*X7-FF*X7P)+(EEP*X8-EE*X8P)+(DDP*X9-DD*X9P)
+  +(CCP*X10-CC*X10P)
 MP=(GGP*X7-GG*X7P)+(FFP*X8-FF*X8P)+(EEP*X9-EE*X9P)+
+    (DDP*X10-DD*X10P)
 NP=(GGP*X8-GG*X8P)+(FFP*X9-FF*X9P)+(EEP*X10-EE*X10P)
 OP=(GGP*X9-GG*X9P)+(FFP*X10-FF*X10P)
 PP=(GGP*X10-GG*X10P)
 T1=(Y1P*X1)
 T2=(Y2P*X1)+(Y1P*X2)
 T3=(Y3P*X1)+(Y2P*X2)+(Y1P*X3)
 Z1=(HHP*X1-HH*X1P)+(Y3P*X2)+(Y2P*X3)+(Y1P*X4)
 Z2=(IIP*X1-II*X1P)+(HHP*X2-HH*X2P)+(Y3P*X3)+(Y2P*X4)+(Y1P*X5)
 Z3=(JJP*X1-JJ*X1P)+(IIP*X2-II*X2P)+(HHP*X3-HH*X3P)+
+    (Y3P*X4)+(Y2P*X5)+(Y1P*X6)
 Z4=(KKP*X1-KK*X1P)+(JJP*X2-JJ*X2P)+(IIP*X3-II*X3P)+(HHP*X4-HH*X4P)
+  +(Y3P*X5)+(Y2P*X6)+(Y1P*X7)
 Z5=(LLP*X1-LL*X1P)+(KKP*X2-KK*X2P)+(JJP*X3-JJ*X3P)+(IIP*X4-II*X4P)
+  +(HHP*X5-HH*X5P)+(Y3P*X6)+(Y2P*X7)+(Y1P*X8)
 Z6=(MMP*X1-MM*X1P)+(LLP*X2-LL*X2P)+(KKP*X3-KK*X3P)+(JJP*X4-JJ*X4P)
+  +(IIP*X5-II*X5P)+(HHP*X6-HH*X6P)+(Y3P*X7)+(Y2P*X8)+(Y1P*X9)
 Z7=(NNP*X1-NN*X1P)+(MMP*X2-MM*X2P)+(LLP*X3-LL*X3P)+
+    (KKP*X4-KK*X4P)+(JJP*X5-JJ*X5P)+(IIP*X6-II*X6P)+
+    (HHP*X7-HH*X7P)+(Y3P*X8)+(Y2P*X9)+(Y1P*X10)
 Z8=(NNP*X2-NN*X2P)+(MMP*X3-MM*X3P)+(LLP*X4-LL*X4P)+
+    (KKP*X5-KK*X5P)+(JJP*X6-JJ*X6P)+(IIP*X7-II*X7P)+
```

```
+     (HHP*X8-HH*X8P)+(Y3P*X9)+(Y2P*X10)
      Z9=(NNP*X3-NN*X3P)+(MMP*X4-MM*X4P)+(LLP*X5-LL*X5P)+
+     (KKP*X6-KK*X6P)+(JJP*X7-JJ*X7P)+(IIP*X8-II*X8P)+
+     (HHP*X9-HH*X9P)+(Y3P*X10)
      Z10=(NNP*X4-NN*X4P)+(MMP*X5-MM*X5P)+(LLP*X6-LL*X6P)+
+     (KKP*X7-KK*X7P)+(JJP*X8-JJ*X8P)+(IIP*X9-II*X9P)+
+     (HHP*X10-HH*X10P)
      Z11=(NNP*X5-NN*X5P)+(MMP*X6-MM*X6P)+(LLP*X7-LL*X7P)+
+     (KKP*X8-KK*X8P)+(JJP*X9-JJ*X9P)+(IIP*X10-II*X10P)
      Z12=(NNP*X6-NN*X6P)+(MMP*X7-MM*X7P)+(LLP*X8-LL*X8P)+
+     (KKP*X9-KK*X9P)+(JJP*X10-JJ*X10P)
      Z13=(NNP*X7-NN*X7P)+(MMP*X8-MM*X8P)+(LLP*X9-LL*X9P)+
+     (KKP*X10-KK*X10P)
      Z14=(NNP*X8-NN*X8P)+(MMP*X9-MM*X9P)+(LLP*X10-LL*X10P)
      Z15=(NNP*X9-NN*X9P)+(MMP*X10-MM*X10P)
      Z16=(NNP*X10-NN*X10P)

C FINAL STEP

      W(34)=T1*A
      W(33)=T1*B+T2*A
      W(32)=T1*C+T2*B+T3*A
      W(31)=T1*D+T2*C+T3*B+(Z1*A-S1*AP)
      W(30)=T1*E+T2*D+T3*C+(Z1*B-S1*BP)+(Z2*A-S2*AP)
      W(29)=T1*F+T2*E+T3*D+(Z1*C-S1*CP)+(Z2*B-S2*BP)+(Z3*A-S3*AP)
      W(28)=T1*G+T2*F+T3*E+(Z1*D-S1*DP)+(Z2*C-S2*CP)+(Z3*B-S3*BP)+
+       (Z4*A-S4*AP)
      W(27)=T1*H+T2*G+T3*F+(Z1*E-S1*EP)+(Z2*D-S2*DP)+(Z3*C-S3*CP)+
+       (Z4*B-S4*BP)+(Z5*A-S5*AP)
      W(26)=T1*I+T2*H+T3*G+(Z1*F-S1*FP)+(Z2*E-S2*EP)+(Z3*D-S3*DP)+
+       (Z4*C-S4*CP)+(Z5*B-S5*BP)+(Z6*A-S6*AP)
      W(25)=T1*J+T2*I+T3*H+(Z1*G-S1*GP)+(Z2*F-S2*FP)+(Z3*E-S3*EP)+
+       (Z4*D-S4*DP)+(Z5*C-S5*CP)+(Z6*B-S6*BP)+(Z7*A-S7*AP)
      W(24)=T1*K+T2*J+T3*I+(Z1*H-S1*HP)+(Z2*G-S2*GP)+(Z3*F-S3*FP)+
+       (Z4*E-S4*EP)+(Z5*D-S5*DP)+(Z6*C-S6*CP)+(Z7*B-S7*BP)+
+       (Z8*A-S8*AP)
      W(23)=T1*L+T2*K+T3*J+(Z1*I-S1*IP)+(Z2*H-S2*HP)+(Z3*G-S3*GP)+
+       (Z4*F-S4*FP)+(Z5*E-S5*EP)+(Z6*D-S6*DP)+(Z7*C-S7*CP)+
+       (Z8*B-S8*BP)+(Z9*A-S9*AP)
      W(22)=T1*M+T2*L+T3*K+(Z1*J-S1*JP)+(Z2*I-S2*IP)+(Z3*H-S3*HP)+
+       (Z4*G-S4*GP)+(Z5*F-S5*FP)+(Z6*E-S6*EP)+(Z7*D-S7*DP)+
+       (Z8*C-S8*CP)+(Z9*B-S9*BP)+(Z10*A-S10*AP)
      W(21)=T1*N+T2*M+T3*L+(Z1*K-S1*KP)+(Z2*J-S2*JP)+(Z3*I-S3*IP)+
+       (Z4*H-S4*HP)+(Z5*G-S5*GP)+(Z6*F-S6*FP)+(Z7*E-S7*EP)+
+       (Z8*D-S8*DP)+(Z9*C-S9*CP)+(Z10*B-S10*BP)+(Z11*A-S11*AP)
      W(20)=T1*O+T2*N+T3*M+(Z1*L-S1*LP)+(Z2*K-S2*KP)+(Z3*J-S3*JP)+
+       (Z4*I-S4*IP)+(Z5*H-S5*HP)+(Z6*G-S6*GP)+(Z7*F-S7*FP)+
+       (Z8*E-S8*EP)+(Z9*D-S9*DP)+(Z10*C-S10*CP)+(Z11*B-S11*BP)+
+       (Z12*A-S12*AP)
      W(19)=T1*P+T2*O+T3*N+(Z1*M-S1*MP)+(Z2*L-S2*LP)+(Z3*K-S3*KP)+
+       (Z4*J-S4*JP)+(Z5*I-S5*IP)+(Z6*H-S6*HP)+(Z7*G-S7*GP)+
+       (Z8*F-S8*FP)+(Z9*E-S9*EP)+(Z10*D-S10*DP)+(Z11*C-S11*CP)+
+       (Z12*B-S12*BP)+(Z13*A-S13*AP)
      W(18)=T2*P+T3*O+(Z1*N-S1*NP)+(Z2*M-S2*MP)+(Z3*L-S3*LP)+
+       (Z4*K-S4*KP)+(Z5*J-S5*JP)+(Z6*I-S6*IP)+(Z7*H-S7*HP)+
+       (Z8*G-S8*GP)+(Z9*F-S9*FP)+(Z10*E-S10*EP)+(Z11*D-S11*DP)+
+       (Z12*C-S12*CP)+(Z13*B-S13*BP)+(Z14*A-S14*AP)
      W(17)=T3*P+(Z1*O-S1*OP)+(Z2*N-S2*NP)+(Z3*M-S3*MP)+
+       (Z4*L-S4*LP)+(Z5*K-S5*KP)+(Z6*J-S6*JP)+(Z7*I-S7*IP)+
+       (Z8*H-S8*HP)+(Z9*G-S9*GP)+(Z10*F-S10*FP)+(Z11*E-S11*EP)+
+       (Z12*D-S12*DP)+(Z13*C-S13*CP)+(Z14*B-S14*BP)+(Z15*A-S15*AP)
      W(16)=(Z1*P-S1*PP)+(Z2*O-S2*OP)+(Z3*N-S3*NP)+(Z4*M-S4*MP)+
+       (Z5*L-S5*LP)+(Z6*K-S6*KP)+(Z7*J-S7*JP)+(Z8*I-S8*IP)+
+       (Z9*H-S9*HP)+(Z10*G-S10*GP)+(Z11*F-S11*FP)+(Z12*E-S12*EP)+
+       (Z13*D-S13*DP)+(Z14*C-S14*CP)+(Z15*B-S15*BP)+(Z16*A-S16*AP)
      W(15)=(Z2*P-S2*PP)+(Z3*O-S3*OP)+(Z4*N-S4*NP)+(Z5*M-S5*MP)+
+       (Z6*L-S6*LP)+(Z7*K-S7*KP)+(Z8*J-S8*JP)+(Z9*I-S9*IP)+
```

```
     +       (Z10*H-S10*HP)+(Z11*G-S11*GP)+(Z12*F-S12*FP)+(Z13*E-S13*EP)+
     +       (Z14*D-S14*DP)+(Z15*C-S15*CP)+(Z16*B-S16*BP)
      W(14)=(Z3*P-S3*PP)+(Z4*O-S4*OP)+(Z5*N-S5*NP)+(Z6*M-S6*MP)+
     +       (Z7*L-S7*LP)+(Z8*K-S8*KP)+(Z9*J-S9*JP)+(Z10*I-S10*IP)+
     +       (Z11*H-S11*HP)+(Z12*G-S12*GP)+(Z13*F-S13*FP)+(Z14*E-S14*EP)+
     +       (Z15*D-S15*DP)+(Z16*C-S16*CP)
      W(13)=(Z4*P-S4*PP)+(Z5*O-S5*OP)+(Z6*N-S6*NP)+(Z7*M-S7*MP)+
     +       (Z8*L-S8*LP)+(Z9*K-S9*KP)+(Z10*J-S10*JP)+(Z11*I-S11*IP)+
     +       (Z12*H-S12*HP)+(Z13*G-S13*GP)+(Z14*F-S14*FP)+(Z15*E-S15*EP)+
     +       (Z16*D-S16*DP)
      W(12)=(Z5*P-S5*PP)+(Z6*O-S6*OP)+(Z7*N-S7*NP)+(Z8*M-S8*MP)+
     +       (Z9*L-S9*LP)+(Z10*K-S10*KP)+(Z11*J-S11*JP)+(Z12*I-S12*IP)+
     +       (Z13*H-S13*HP)+(Z14*G-S14*GP)+(Z15*F-S15*FP)+(Z16*E-S16*EP)
      W(11)=(Z6*P-S6*PP)+(Z7*O-S7*OP)+(Z8*N-S8*NP)+(Z9*M-S9*MP)+
     +       (Z10*L-S10*LP)+(Z11*K-S11*KP)+(Z12*J-S12*JP)+(Z13*I-S13*IP)+
     +       (Z14*H-S14*HP)+(Z15*G-S15*GP)+(Z16*F-S16*FP)
      W(10)=(Z7*P-S7*PP)+(Z8*O-S8*OP)+(Z9*N-S9*NP)+(Z10*M-S10*MP)+
     +       (Z11*L-S11*LP)+(Z12*K-S12*KP)+(Z13*J-S13*JP)+(Z14*I-S14*IP)+
     +       (Z15*H-S15*HP)+(Z16*G-S16*GP)
      W(9)=(Z8*P-S8*PP)+(Z9*O-S9*OP)+(Z10*N-S10*NP)+(Z11*M-S11*MP)+
     +       (Z12*L-S12*LP)+(Z13*K-S13*KP)+(Z14*J-S14*JP)+(Z15*I-S15*IP)+
     +       (Z16*H-S16*HP)
      W(8)=(Z9*P-S9*PP)+(Z10*O-S10*OP)+(Z11*N-S11*NP)+(Z12*M-S12*MP)+
     +       (Z13*L-S13*LP)+(Z14*K-S14*KP)+(Z15*J-S15*JP)+(Z16*I-S16*IP)
      W(7)=(Z10*P-S10*PP)+(Z11*O-S11*OP)+(Z12*N-S12*NP)+(Z13*M-S13*MP)+
     +       (Z14*L-S14*LP)+(Z15*K-S15*KP)+(Z16*J-S16*JP)
      W(6)=(Z11*P-S11*PP)+(Z12*O-S12*OP)+(Z13*N-S13*NP)+(Z14*M-S14*MP)+
     +       (Z15*L-S15*LP)+(Z16*K-S16*KP)
      W(5)=(Z12*P-S12*PP)+(Z13*O-S13*OP)+(Z14*N-S14*NP)+(Z15*M-S15*MP)+
     +       (Z16*L-S16*LP)
      W(4)=(Z13*P-S13*PP)+(Z14*O-S14*OP)+(Z15*N-S15*NP)+(Z16*M-S16*MP)
      W(3)=(Z14*P-S14*PP)+(Z15*O-S15*OP)+(Z16*N-S16*NP)
      W(2)=(Z15*P-S15*PP)+(Z16*O-S16*OP)
      W(1)=(Z16*P-S16*PP)


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: BISECTG
C
C     DESCRIPTION: FIND THE ROOTS OF A POLYNOMIAL USING
C                  BISECTION METHOD
C
C INPUT:
C     IG           = DEGREE OF THE FUNCTION
C     E33          = COEFF. OF THE FUNCTION
C
C OUTPUT:
C     WW           = ROOTS OF THE FUNCTION
C     INROOT       = NO. OF ROOTS
C
C     C. K. WONG
C     9/12/89
C
      SUBROUTINE BISECTG(IG,E33,WW,INROOT)

      DOUBLE PRECISION E33(*),FA,FB,FC,A,B,C,TOL,WIDTH
     +                 ,AS,BS,WS(100)
     +                 ,DIVV,NITER

      REAL WW(100)
```

```
c************************************************
c Change the no. of bisection if the result does not
c convert in this routine. Sometime it might go up
c to 10000.D0 for a good solution. This slows down
c the process tremendously.
c************************************************


C ASSIGN NO. OF BISECTION
      NITER=1000.D0

C INCREMENT
      DIVV=(1.0D0/NITER)

C TOLERANCE
      TOL=.1D-25

C START THE BISECTION PROCESS
      INROOT=0
      AS=-(DIVV)
      BS=0.0D0

      DO 200 I=1,11000
         IROOT=0
         AS=AS+(DIVV)
         BS=BS+(DIVV)

C STOP THE PROCESS: THE LOWER BOUND HIT 1.
         IF(BS.GT.1.0D0) GOTO 555


         A=AS
         B=BS

C EVALUATE THE FUNCTION WITH THE UPPER AND
C LOWER BOUND VALUES WITHIN A BISECTION SEGMENT

         CALL EVAFUNG(IG,E33,A,FA)
         CALL EVAFUNG(IG,E33,B,FB)

C BOTH BOUNDS > 0 : NO ROOT
         IF((FA.GT.0.0D0).AND.(FB.GT.0.0D0)) THEN
            IROOT=0

C BOTH BOUNDS < 0 : NO ROOT
           ELSEIF((FA.LT.0.0D0).AND.(FB.LT.0.0D0)) THEN
            IROOT=0

C LOWER BOUND IS THE ROOT

           ELSEIF(DABS(FB).LE.TOL) THEN
            IROOT=1
            INROOT=INROOT+1
            WS(INROOT)=B

C UPPER BOUND IS THE ROOT

           ELSEIF(DABS(FA).LE.TOL) THEN
            IROOT=1
            INROOT=INROOT+1
            WS(INROOT)=A

           ELSE

C ELSE, THERE IS A ROOT WITHIN THE BOUNDS
C FIND THE ROOT BY BISECTION METHOD
```

```
          WIDTH=B-A
 10       C=(A+B)*.5D0
          CALL EVAFUNG(IG,E33,C,FC)
          WIDTH=WIDTH*.5D0


          IF(DSIGN(1.0D0,FA).EQ.DSIGN(1.0D0,FC)) THEN
              A=C
              FA=FC
           ELSE
              B=C
              FB=FC
          ENDIF

          IF(WIDTH.LE.TOL) THEN
              IROOT=1
              INROOT=INROOT+1
              WS(INROOT)=C

            ELSEIF(DABS(FA).LE.TOL) THEN
              IROOT=1
              INROOT=INROOT+1
              WS(INROOT)=A

            ELSEIF(DABS(FB).LE.TOL) THEN
              IROOT=1
              INROOT=INROOT+1
              WS(INROOT)=B

          ENDIF

          IF(IROOT.EQ.0) GOTO 10

        ENDIF

 200  CONTINUE


 555  IF(INROOT.EQ.0) THEN
         IROOT=0

C CONVERT THE ROOT TO SINGLE PRECISION

        ELSEIF(INROOT.GE.1) THEN
         IROOT=1
         DO 400 I=1,INROOT
             WW(I)=WS(I)
 400     CONTINUE
      ENDIF


      RETURN
      END


C--------------------------------------------------
C
C     SUBROUTINE: EVAFUNG
C
C     DESCRIPTION: TO EVALUATE THE FUNCTION E33
C
C INPUT:
C     IG             = DEGREE OF THE POLYNOMIAL
C     E33            = COEFF. OF THE POLYNOMIAL
C     F              = VARIABLE
C
C OUTPUT:
```

```
C       FC              = VALUE OF THE POLYNOMIAL
C
C       C. K. WONG
C       9/12/89
C

        SUBROUTINE EVAFUNG(IG,E33,F,FC)

        DOUBLE PRECISION E33(*),F,FC,P,TOL,TF
     +                  ,PP

C INITIALIZE
        TOL=.1D-50
        FC=0.0D0

C VARIABLE TOO SMALL, ASSUME ALL THE COEFF --> 0
C EXCEPT THE LAST ONE

        IF(F.LT.TOL) GOTO 20

C EVALUATE THE COEFF. ONE AT A TIME

        DO 10 I=IG,2,-1

C MAKE SURE THE CALCULATION IS MANAGEBLE

               TF=(.1D-50)**(1.0D0/DBLE(I-1))
               IF(F.LE.TF) GOTO 10

               P=F**(I-1)
               IF(DABS(P).LT.TOL) GOTO 10

               IF(DABS(E33(I)).GT.0.1D10) GOTO 5
               PP=TOL/E33(I)

               IF(DABS(P).LT.DABS(PP)) GOTO 10

 5              FC=FC+(E33(I)*P)

 10     CONTINUE


 20     FC=FC+E33(1)

        RETURN
        END
```

Appendix J : Program UTILITY

```
C-----------------------------------------------
C
C     SUBROUTINE: FBPT
C
C     DESCRIPTION: FIND A POINT ON THE B-SPLINE SURFACE
C
C INPUT:
C     PKL          = B-SPLINE SURFACE CONTROL HULL
C     U,W          = PARAMETRIC VALUES
C
C OUTPUT:
C     TPST         = A POINT ON THE SURFACE
C     IFBPTF       = DUMMY VARIABLE
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE FBPT(PKL,U,W,TPST,IFBPTF)

      INTEGER  IFBPTF,IMULF
      REAL U,W,TPST(1,3), WLT(4,1), UK(1,4), MK(4,4), MLT(4,4),
     +     MLTWLT(4,1), UKMK(1,4), PKL(0:3,0:3,3), TPKL(4,4),
     +     PMW(4,1), UMP(1,1)

      DATA MK/-0.166667,0.5,-0.5,0.166667,
     +        0.5,-1.0,0.0,0.666667,
     +        -0.5,0.5,0.5,0.166667,
     +        0.166667,0.0,0.0,0.0/
      DATA MLT/-0.166667,0.5,-0.5,0.166667,
     +         0.5,-1.0,0.5,0.0,
     +         -0.5,0.0,0.5,0.0,
     +         0.166667,0.666667,0.166667,0.0/
      DATA III/1/

C SET UP THE PARAMETRIC MATRIX
```

```
      DO 10 I=1,3
         WLT(I,1)=W**(4-I)
         UK(1,I)=U**(4-I)
 10   CONTINUE
      WLT(4,1)=1.0
      UK(1,4)=1.0

C CALCULATE THE POINT

      CALL MATMUL(4,4,MLT,4,1,WLT,MLTWLT,IMULF)

      CALL MATMUL(1,4,UK,4,4,MK,UKMK,IMULF)

      DO 20 I=1,3
         DO 30 J=1,4
              DO 40 II=1,4
                   TPKL(J,II)= PKL(J-1,II-1,I)
 40           CONTINUE
 30      CONTINUE

         CALL MATMUL(4,4,TPKL,4,1,MLTWLT,PMW,IMULF)

         CALL MATMUL(1,4,UKMK,4,1,PMW,UMP,IMULF)

         TPST(1,I)=UMP(1,1)

 20   CONTINUE


      RETURN
      END
C---------------------------------------------------
C
C     SUBROUTINE: MATMUL
C
C     DESCRIPTION: MATRIX MULTIPLICATION
C
C INPUT:
C     IM,IN          = NO. OF ROWS AND COLUMNS FOR MATRIX 1
C     MAT1,MAT2      = MATRIX 1 AND MATRIX 2
C     IMM,INN        = NO. OF ROWS AND COLUMNS FOR MATRIX 2
C
C OUTPUT:
C     MAT3           = RESULTANT MATRIX
C     IMULF          = DUMMY VARIABLE
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE MATMUL(IM,IN,MAT1,IMM,INN,MAT2,MAT3,IMULF)

      INTEGER IM,IN,IMM,INN,IMULF
      REAL MAT1(IM,IN), MAT2(IMM,INN), MAT3(IM,INN)


C CHECK THE INPUT MATRIX: NO. OF COLUMNS OF MAT1 HAS
C TO BE THE SAME AS NO. OF ROWS OF MAT2

      IF(IN.NE.IMM) THEN
         WRITE(6,*)'BAD MATRIX INPUT IN MATMUL'
         RETURN
      ENDIF

C MULTIPLICATION
```

```
      DO 30 II=1,INN
         DO 20 J=1,IM
              MAT3(J,II)=0.
              DO 10 I=1,IN
                   MAT3(J,II)=MAT3(J,II)+(MAT1(J,I)*MAT2(I,II))
 10           CONTINUE
 20      CONTINUE
 30   CONTINUE

      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: PER4
C
C     DESCRIPTION: COMPUTE THE COORDINATES OF A POINT ON
C                  A  PERIODIC CUBIC B-SPLINE CURVE.
C
C INPUT:
C     N            = NUMBER OF CONTROL POINTS
C     CI           = THE CONTROL POINTS OF THE CURVE
C     I2           = FLAG SPECIFYING IF THE CURVE IS OPEN(I2=2)
C                    OR CLOSED(I2=1)
C     U            = PARAMETRIC VALUE
C
C OUTPUT:
C     P            = THE POINT ON THE CURVE
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE PER4(N,CI,I2,U,P)

      INTEGER NPOINT
      PARAMETER(NPOINT=300)
      REAL CI(3,*), U, P(3),  M4(4,4), TCI(4,3),
     +     TMM(4,3), TU(4), CI0(0:NPOINT,3)

      INTEGER N, I1, I2

      DATA M4/-1.,3.,-3.,1.,3.,-6.,0.,4.,-3.,3.,3.,1.,1.,0.,0.,0./
      DATA TOL/.1E-8/

      UU=U
      NN=N-1

C COPY CI INTO CI0 (FOR COMPUTATIONAL SAKE)

      DO 500 I=1,N
         DO 510 J=1,3
              CI0(I-1,J)=CI(J,I)
 510     CONTINUE
 500  CONTINUE


C OPENED CURVE
      II=NN-2

C CLOSED CURVE
      IF(I2.EQ.1) II=NN+1

C UU=0.
      IF(ABS(UU-0.).LT.TOL) THEN
         IRANGE=0
         GOTO 100
C UU=1.
```

```
        ELSEIF (ABS(UU-1.).LT.TOL) THEN
          IRANGE=II-1
          GOTO 100
      ENDIF

C FIND THE RANGE THAT U FALL IN (FIND I)
      IRANGE=0
      RII=REAL(II)
      DO 10 I=II,1,-1
         RI=REAL(I)
         RANGE=RI/RII

C THE RANGE THAT U FALL IN
         IF(UU.GE.RANGE) THEN
              IRANGE=I
              GOTO 100
         ENDIF
 10   CONTINUE

C REPARAMETRIZE U WITHIN THE RANGE
 100  IF(IRANGE.EQ.0) THEN
         RANGE1=0.
        ELSE
         RANGE1=REAL(IRANGE)/REAL(II)
      ENDIF

      RANGE2=(1./REAL(II))+RANGE1
      UU=(UU-RANGE1)/(RANGE2-RANGE1)

C FIND THE COORDINATE OF THE POINT WITHIN THE RANGE OF U
      IT=IRANGE

C OPEND CURVE
      IF(I2.EQ.2) THEN
         DO 50 I= 1,4
              TCI(I,1)=CIO(IT,1)
              TCI(I,2)=CIO(IT,2)
              TCI(I,3)=CIO(IT,3)
              IT=IT+1
 50      CONTINUE
      ENDIF

C CLOSE CURVE
      IF(I2.EQ.1) THEN
          DO 60 I= 1,4
              IF(IT.GT.II) THEN
                   RMOD=REAL(IT)/REAL(II)
                   RMOD=AINT(RMOD)
                   RMOD=REAL(II)*RMOD
                   IF=IT-(INT(RMOD))
                ELSEIF(IT.EQ.II) THEN
                   IF=0
                ELSE
                   IF=IT
              ENDIF
                   TCI(I,1)=CIO(IF,1)
                   TCI(I,2)=CIO(IF,2)
                   TCI(I,3)=CIO(IF,3)
                   IT=IT+1
 60       CONTINUE
      ENDIF


C MULTIPLICATION OF M AND CI

      DO 20 I=1,3
         DO 30 J=1,4
```

```
               AA=0.
               DO 40 K=1,4
                    AA=AA+(M4(J,K)*TCI(K,I))
 40            CONTINUE
               TMM(J,I)=AA
 30        CONTINUE
 20     CONTINUE

C FIND U MATRIX

        DO 65 I=1,3
           IF(ABS(UU-0.).LT.TOL) THEN
               TU(I)=0.
               GOTO 65
             ELSEIF(ABS(UU-1.).LT.TOL) THEN
               TU(I)=1.
               GOTO 65
           ENDIF
           TU(I)=UU**(4-I)
 65     CONTINUE
        TU(4)=1.

C MULT. 1./6.
        DO 67 I=1,4
           TU(I)=TU(I)*(1./6.)
 67     CONTINUE

C FIND THE POINT

        DO 70 I=1,3
           P(I)=0.
           DO 80 J=1,4
               P(I)=P(I)+(TU(J)*TMM(J,I))
 80        CONTINUE
 70     CONTINUE

        RETURN
        END
C--------------------------------------------------
C
C       SUBROUTINE: DSURN2
C
C       DESCRIPTION: DRAW THE B-SPLINE SURFACE,
C                    PROVIDED BY THE CONTROL HULL AND THE NUMBER
C                    OF CURVES ON THE SURFACE
C
C INPUT:
C       PKL          = CONTROL HULL OF THE SURFACE
C       ICOLOR       = COLOR NUMBER
C       N            = NO. OF CURVES ON THE SURFACE
C
C OUTPUT: NONE
C
C       C. K. WONG
C       9/12/89
C
C
C

        SUBROUTINE DSURN2(PKL,ICOLOR,N)

        INTEGER ISURF, ICOLOR
        REAL UU,WW,PST1(1,3), PST2(1,3),PKL(0:3,0:3,3),PLIST(6)

        INTEGER N

C SET THE COLOR
```

```
      CALL GPPLCI(ICOLOR)

C DRAW THE SURFACE IN ONE PARAMETRIC DIRECTION

      TT=1./REAL(N)
      UU=-TT

      DO 10 I5=1,N+1
         UU=UU+TT
         WW=-TT
         DO 20 J5=1,N
              WW=WW+TT
              IF(J5.EQ.1) THEN

                   CALL FBPT(PKL,UU,WW,PST1,ISURF)
                   WW=WW+TT
              ENDIF

              CALL FBPT(PKL,UU,WW,PST2,ISURF)


C FIND THE POINTS ON THE SURFACE

              DO 50 K1=1,3
                   PLIST(K1)=PST1(1,K1)
                   PLIST(K1+3)=PST2(1,K1)
                   PST1(1,K1)=PST2(1,K1)
 50           CONTINUE

C DRAW THE CURVE

              CALL GPPL3(2,3,PLIST)

 20      CONTINUE
 10   CONTINUE

C DRAW THE SURFACE IN THE OTHER PARAMETRIC DIRECTION

      WW=-TT
      DO 100 I=1,N+1
         WW=WW+TT
         UU=-TT
         DO 200 J=1,N
              UU=UU+TT
              IF(J.EQ.1) THEN
                   CALL FBPT(PKL,UU,WW,PST1,ISURF)

                   UU=UU+TT
              ENDIF

              CALL FBPT(PKL,UU,WW,PST2,ISURF)

C FIND THE POINTS ON THE SURFACE

              DO 500 K=1,3
                   PLIST(K)=PST1(1,K)
                   PLIST(K+3)=PST2(1,K)
                   PST1(1,K)=PST2(1,K)
 500          CONTINUE

C DRAW THE CURVE

              CALL GPPL3(2,3,PLIST)

 200     CONTINUE
 100  CONTINUE
```

```
      RETURN
      END
C-----------------------------------------------------
C
C     SUBROUTINE: DSURN
C
C     DESCRIPTION: DRAW THE B-SPLINE SURFACE (THE BOUNDARIES),
C                  PROVIDED BY THE CONTROL HULL
C
C INPUT:
C     PKL          = CONTROL HULL OF THE SURFACE
C     ICOLOR       = COLOR NUMBER
C     N            = DUMMY VARIABLE
C
C OUTPUT: NONE
C
C     C. K. WONG
C     9/12/89

      SUBROUTINE DSURN(PKL,ICOLOR,N)

      INTEGER ISURF, ICOLOR
      REAL UU,WW,PST1(1,3), PST2(1,3),PKL(0:3,0:3,3),PLIST(6)

      INTEGER N

C SET THE COLOR

      CALL GPPLCI(ICOLOR)

C DRAW THE SURFACE IN ONE PARAMETRIC DIRECTION

      UU=-1.
      DO 10 I5=1,2
         UU=UU+1.
         WW=-.1
         DO 20 J5=1,10
              WW=WW+.1
              IF(J5.EQ.1) THEN

                   CALL FBPT(PKL,UU,WW,PST1,ISURF)
                   WW=WW+.1

              ENDIF
              CALL FBPT(PKL,UU,WW,PST2,ISURF)


C FIND THE POINTS ON THE SURFACE

              DO 50 K1=1,3
                   PLIST(K1)=PST1(1,K1)
                   PLIST(K1+3)=PST2(1,K1)
                   PST1(1,K1)=PST2(1,K1)
 50           CONTINUE

C DRAW THE SURFACE

              CALL GPPL3(2,3,PLIST)

 20      CONTINUE
 10   CONTINUE

C DRAW THE SURFACE IN THE OTHER PARAMETRIC DIRECTION

      WW=-1.
      DO 100 I=1,2
```

```
        WW=WW+1.
        UU=-.1
        DO 200 J=1,10
             UU=UU+.1
             IF(J.EQ.1) THEN

                  CALL FBPT(PKL,UU,WW,PST1,ISURF)
                  UU=UU+.1

             ENDIF

             CALL FBPT(PKL,UU,WW,PST2,ISURF)

C FIND THE POINTS ON THE SURFACE

             DO 500 K=1,3
                  PLIST(K)=PST1(1,K)
                  PLIST(K+3)=PST2(1,K)
                  PST1(1,K)=PST2(1,K)
 500         CONTINUE

C DRAW THE SURFACE

             CALL GPPL3(2,3,PLIST)

 200    CONTINUE
 100  CONTINUE

      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: LINEAR
C
C     DESCRIPTION: SOLVE A LINEAR EQUATION
C
C INPUT:
C     COF         = COEFF. OF THE EQ.
C     UMIN,UMAX   = UPPER AND LOWER BOUNDS FOR THE SOLUTION
C
C OUTPUT:
C     IR          = NO. OF SOLUTION
C     RO          = SOLUTION
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE LINEAR(COF,UMIN,UMAX,IR,RO)

      REAL COF(4),RO(3)

      IR=1
      RO(1)=-COF(4)/COF(3)

C SOLUTION NOT WITHIN THE RANGE

      IF((RO(1).GT.UMAX).OR.(RO(1).LT.UMIN)) IR=0

      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: QUADRA
C
C     DESCRIPTION: SOLVE A QUADRATIC EQUATION
C
```

```
C INPUT:
C     COF            = COEFF. OF THE EQ.
C     UMIN,UMAX      = UPPER AND LOWER BOUNDS FOR THE SOLUTION
C
C OUTPUT:
C     IR             = NO. OF SOLUTION
C     RO             = SOLUTION
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE QUADRA(COF,UMIN,UMAX,IR,RO)

      REAL COF(4),RO(3)

C SOLVE FOR THE ROOTS

      CC=COF(3)**2-(4.*COF(2)*COF(4))

C NO SOLUTION

      IF(CC.LT.0) THEN
         IR=0
         RETURN
      ENDIF

      SC=SQRT(CC)
      RO(1)=(-COF(3)+SC)/(2.*COF(2))
      RO(2)=(-COF(3)-SC)/(2.*COF(2))

      IR=2

C CHECK THE RANGE OF THE SOLUTION

      IF((RO(1).GT.UMAX).OR.(RO(1).LT.UMIN)) THEN
         IR=IR-1
         RO(1)=RO(2)
      ENDIF

      IF((RO(2).GT.UMAX).OR.(RO(2).LT.UMIN)) THEN
         IR=IR-1
         RETURN
      ENDIF

C ARRANGE THE RESULTS WITH INCREASING ORDER

      IF(IR.EQ.2) THEN
         AA=RO(1)
         IF((RO(1)-RO(2)).GT.0.) THEN
              RO(1)=RO(2)
              RO(2)=AA
         ENDIF
      ENDIF

      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: CUROOT
C
C     DESCRIPTION: FIND THE ROOTS OF A CUBIC FUNCTION
C                  WITHIN 0 AND 1
C
C INPUT:
C     R3             = COEFF. OF THE EQ.
C
```

```
C OUTPUT:
C     IR            = NO. OF SOLUTION
C     RO            = SOLUTION
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE CUROOT(R3,IR,RO)

      REAL R3(4),RO(3)

C SET THE TOLERANCE

      TLIM=.001
      TOL=.1E-10
      IR=1

C LINEAR FUNCTION
      IF((ABS(R3(1)).LT.TOL).AND.(ABS(R3(2)).LT.TOL))THEN
           RO(1)=-R3(4)/R3(3)

           IF((RO(1).GT.(1.+TLIM)).OR.(RO(1).LT.-(TLIM))) THEN
               IR=0
           ENDIF

C QUADRADIC FUNCTION
         ELSEIF(ABS(R3(1)).LT.TOL) THEN
           CALL QUADRA(R3,-(TLIM),(1.+TLIM),IR,RO)
         ELSE

C CUBIC FUNCTION
           CALL CUBIC(R3,-(TLIM),(1.+TLIM),IR,RO)
      ENDIF


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: CUBIC
C
C     DESCRIPTION: FIND THE ROOTS OF A CUBIC EQUATION WITHIN
C                  A GIVEN INTERVAL. SORT THE ROOTS
C                  IN A ASCENDING ORDER
C
C INPUT:
C     COF           = COEFF. OF THE CUBIC EQUATION
C     UMIN,UMAN     = LOWER AND UPPER BOUNDS FOR THE SOLUTION
C
C OUTPUT:
C     NU            = NO. OF SOLUTION
C     U             = SOLUTION
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE CUBIC(COF, UMIN, UMAX, NU,U)

      REAL COF(4), UMAX,UMIN,U(3),UU(3),
     +     D,Q,R,S,T,A1,A2,A3,SD,SQ,SV,PHI

      INTEGER NU

C CALCULATION OF THE ROOTS
```

```
      A1=COF(2)/COF(1)
      A2=COF(3)/COF(1)
      A3=COF(4)/COF(1)

      Q=(3.*A2-A1*A1)/9.
      R=(-9.*A1*A2+27.*A3+2.*A1**3)/54.
      D=Q**3+R*R
      IF(D)10,10,11
 11   SD=SQRT(D)
      IF(-R+SD)12,13,13
 12   S=-(ABS(-R+SD))**(1./3.)
      GOTO 14
 13   S=(-R+SD)**(1./3.)
 14   IF(-R-SD)15,16,16
 15   T=-(ABS(-R-SD))**(1./3.)
      GOTO 17
 16   T=(-R-SD)**(1./3.)
 17   U(1)=S+T-A1/3.

C NO ROOT

      IF(U(1).GT.UMAX.OR.U(1).LT.UMIN) THEN
         NU=0
         GOTO 100
      ENDIF

C ONE ROOT

      NU=1
      GOTO 100
C  D<0, Q**3+R**2<0, -Q**3>R**2>0, -Q>0

 10   PHI=ACOS(R/SQRT(-Q**3))
      SQ=-2.*SQRT(-Q)
      U(1)= SQ*COS(PHI/3.)-A1/3.
      U(2)= SQ*COS(PHI/3.+120./57.295780)-A1/3.
      U(3)= SQ*COS(PHI/3.+240./57.295780)-A1/3.
      NU=3

C THREE ROOTS : ARRANGE THEM IN ASCENDING ORDER

      AA=U(1)
      IF((U(1)-U(2)).GT.0) THEN
         U(1)=U(2)
         U(2)=AA
      ENDIF

      AA=U(2)
      IF((U(2)-U(3)).GT.0) THEN
         U(2)=U(3)
         U(3)=AA
      ENDIF

      AA=U(1)
      IF((U(1)-U(2)).GT.0) THEN
         U(1)=U(2)
         U(2)=AA
      ENDIF

C CHECK THE RANGE OF THE ROOTS

      II=1
      DO 200 I=1,3

         IF(U(I).GT.UMAX.OR.U(I).LT.UMIN) THEN
               NU=NU-1
               GOTO 200
```

```
         ENDIF

         UU(II)=U(I)
         II=II+1

 200  CONTINUE

      IF(NU.EQ.0) GOTO 100
      DO 50 I=1,NU
         U(I)=UU(I)
 50   CONTINUE

 100  RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: EVAFUN
C
C     DESCRIPTION: EVALUATE A CUBIC FUNCTION
C
C INPUT:
C     CCOF          = COEFF. OF THE CUBIC FUNCTION
C     VV            = PARAMETRIC VALUE
C
C OUTPUT:
C     FUNV          = FUNCTION VALUE
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE EVAFUN(CCOF,VV,FUNV)

      REAL CCOF(4),VV,FUNV

C SET TOLERANCE

      TOL=.1E-10

      IF(ABS(VV).LT.TOL) THEN
         FUNV=CCOF(4)
         RETURN
      ENDIF

      FUNV=((VV**3)*CCOF(1))+((VV**2)*CCOF(2))+
     +     (VV*CCOF(3))+CCOF(4)


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: COMRELT
C
C     DESCRIPTION: COMPARE THE INTERSECTION POINTS AND
C                  PICK THE ONE WITH THE MINIMUM DISTANCE BETWEEN
C                  TWO POINTS ON THE SURFACES
C
C INPUT:
C     PKL1,PKL2     = SURFACES CONTROL HULLS
C     U1,W1,U2,W2   = PARAMETRIC VALUES
C     IF            = NO. OF INTERSECTION POINTS
C
C OUTPUT:
C     TUW           = THE FINAL INTERSECTION POINT
C
C     C. K. WONG
```

```
C      2/12/90
C

       SUBROUTINE COMRELT(PKL1,PKL2,U1,W1,U2,W2,IF,TUW)

       REAL PKL1(0:3,0:3,3), PKL2(0:3,0:3,3)
     +      ,U1(*),W1(*),U2(*),W2(*),TUW(4)
     +      ,PT1(1,3), PT2(1,3)
     +      ,DIS(3),MIN,PTDIS


C SET THE TOLERANCE
      MIN=100.

      TMIN=.5

      DO 10 I=1,IF

C FIND THE POINTS ON BOTH SURFACE IN X, Y, Z SPACE
         CALL FBPT(PKL1,U1(I),W1(I),PT1,JJ)
         CALL FBPT(PKL2,U2(I),W2(I),PT2,JJ)

C FIND THE DISTANCE BETWEEN THEM
         DO 20 J=1,3
            DIS(J)=PT1(1,J)-PT2(1,J)
 20      CONTINUE

         PTDIS=0.
         DO 30 K=1,3
              IF(ABS(DIS(K)).LT.0.1E-20) GOTO 30
              PTDIS=PTDIS+(ABS(DIS(K))**2)
 30      CONTINUE

         PTDIS=PTDIS**.5

C COMPARE THE DISTANCE
         IF(PTDIS.LT.MIN) THEN
              IIR=I
              MIN=PTDIS
         ENDIF

 10   CONTINUE

C DISCARD THE POINTS IF THE DISTANCE IS OUT OF LIMIT
      IF(MIN.GT.TMIN) THEN
         TUW(1)=2.
         TUW(2)=2.
         TUW(3)=2.
         TUW(4)=2.
        ELSE

C OR ELSE KEEP THEM
         TUW(1)=U1(IIR)
         TUW(2)=W1(IIR)
         TUW(3)=U2(IIR)
         TUW(4)=W2(IIR)
      ENDIF

      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: PUTUW1
C
C     DESCRIPTION: WRITE THE THE INTERSECTION POINT INTO TTUW
C
C INPUT:
```

```
C      II             = NO. OF DATA IN TTUW
C      TUW            = INTERSECTION POINT
C
C OUTPUT:
C      TTUW           = DATA SET CONTAINS INTERSECTION POINTS
C
C      C. K. WONG
C      9/12/89
C

      SUBROUTINE PUTUW1(II,TUW,TTUW)

      REAL TUW(4),TTUW(8,4)

      II=II+1
      DO 10 I=1,4
         TTUW(II,I)=TUW(I)
 10   CONTINUE

      RETURN
      END
C--------------------------------------------------
C
C      SUBROUTINE: GRID
C
C      DESCRIPTION: GET RID OF THE UNWANTED DATA.
C
C INPUT:
C      TTUW           = INTERSECTION POINT IN PARAMETRIC SPACE
C
C OUTPUT:
C      IN             = NO. OF VALID DATA
C
C      C. K. WONG
C      9/12/89
C
C

      SUBROUTINE GRID(TTUW,IN)

      REAL TTUW(8,4)
      INTEGER ID(8)

C SET THE FLAG AND TOLERANCE
      IREP=0
      TLIM=.0001

      TLIM2=.01

C CHECK FOR THE SAME POINT
      DO 100 I=1,7
        IF((TTUW(I,1).GT.1.5).OR.
     +     (TTUW(I,2).GT.1.5).OR.
     +     (TTUW(I,3).GT.1.5).OR.
     +     (TTUW(I,4).GT.1.5)) THEN
           GOTO 100
        ENDIF

         DO 110 J=I+1,8
           IF((TTUW(J,1).GT.1.5).OR.
     +        (TTUW(J,2).GT.1.5).OR.
     +        (TTUW(J,3).GT.1.5).OR.
     +        (TTUW(J,4).GT.1.5)) THEN
              GOTO 110
           ENDIF
```

```
C SAME POINT
          IF((ABS(TTUW(I,1)-TTUW(J,1)).LT.TLIM2).AND.
     +       (ABS(TTUW(I,2)-TTUW(J,2)).LT.TLIM2).AND.
     +       (ABS(TTUW(I,3)-TTUW(J,3)).LT.TLIM2).AND.
     +       (ABS(TTUW(I,4)-TTUW(J,4)).LT.TLIM2)) THEN

C PICK THE ONE THAT WITHIN 0 AND 1
              IB1=0
              IB2=0
              DO 120 K=1,4
                   IF((TTUW(J,K).GT.(1.+TLIM)).OR.
     +                (TTUW(J,K).LT.-(TLIM))) THEN
                        IB2=1
                   ENDIF
                   IF((TTUW(I,K).GT.(1.+TLIM)).OR.
     +                (TTUW(I,K).LT.-(TLIM))) THEN
                        IB1=1
                   ENDIF

 120          CONTINUE

C DISCARD THE POINT THAT OFF THE RANGE
                   IF(IB1.EQ.1) THEN
                        DO 130 KK=1,4
                             TTUW(I,KK)=2.
 130                    CONTINUE
                     ELSE
                        DO 140 KK=1,4
                             TTUW(J,KK)=2.
 140                    CONTINUE
                   ENDIF

C SET THE REPEATANCE FLAG FOR FURTHER INVESTIGATION
              IREP=1
          ENDIF

 110     CONTINUE
 100  CONTINUE


C INITIALIZE COUNTERS
      II=0
      IN=8

C IDENTIFY THE DATA THAT OUT OF RANGE
      DO 10 I=1,8
        IF((TTUW(I,1).GT.1.5).OR.
     +     (TTUW(I,2).GT.1.5).OR.
     +     (TTUW(I,3).GT.1.5).OR.
     +     (TTUW(I,4).GT.1.5)) THEN
              II=II+1
              ID(II)=I
         ENDIF
 10   CONTINUE

      IN=8-II

C ALL OUT OF RANGE
      IF(IN.EQ.0) RETURN

C ONE REPEATED INTERSECTION POINT: DISCARD
      IF((IN.EQ.1).AND.(IREP.EQ.1)) THEN
         IN=0
         RETURN
      ENDIF
```

```
C GET RID OF THE OUT OF RANGE DATA AND
C REORDER TTUW
      JJ=0
      DO 20 I=1,8
         DO 30 J=1,II
           IF(I.EQ.ID(J)) GOTO 20
 30      CONTINUE
         JJ=JJ+1
         DO 40 K=1,4
              TTUW(JJ,K)=TTUW(I,K)
 40      CONTINUE
 20   CONTINUE

      DO 50 I=IN+1,8
         DO 60 J=1,4
              TTUW(I,J)=2.
 60      CONTINUE
 50   CONTINUE


      RETURN
      END
C-------------------------------------------------
C
C     SUBROUTINE: FMAX
C
C     DESCRIPTION: FIND THE MAX DIFFERENCE AMONG THE VARIABLES
C
C INPUT:
C     TTUW          = INTERSECTION POINTS IN PARAMETRIC SPACE
C     IN            = NO. OF VALID DATA
C
C OUTPUT:
C     IR1,IR2       = ROW IDENTIFIERS FOR MAX/MIN VALUES
C     IC            = COLUMN IDENTIFIER FOR MAX/MIN VALUES
C
C     C. K. WONG
C     9/12/89
C
C

      SUBROUTINE FMAX(TTUW,IN,IR1,IR2,IC)

      REAL TTUW(8,4)

C SET THE TOLERANCE
      V=0.

C COMPARE THE DIFF.
      DO 10 I=1,IN-1
         DO 20 J=I+1,IN
              DO 30 K=1,4
                   VV=ABS(TTUW(I,K)-TTUW(J,K))
                   IF(VV.GT.V) THEN
                        V=VV
                        IR1=I
                        IR2=J
                        IC=K
                   ENDIF
 30           CONTINUE
 20      CONTINUE
 10   CONTINUE

      RETURN
      END
C-------------------------------------------------
C
```

```
C      SUBROUTINE: TESTBAD
C
C      DESCRIPTION: FIND THE DISTANCE BETWEEN TWO POINTS
C
C INPUT:
C      PKL1,PKL2     = INTERSECTING SURFACES
C      U1,W1,U2,W2   = SURFACE PARAMETRIC VALUES
C
C OUTPUT:
C      MIN           = THE DISTANCE BETWEEN TWO POINTS
C
C      C. K. WONG
C      2/12/90
C
C

      SUBROUTINE TESTBAD(PKL1,PKL2,U1,W1,U2,W2,MIN)

      REAL PKL1(0:3,0:3,3), PKL2(0:3,0:3,3)
     +     ,PT1(1,3), PT2(1,3)
     +     ,DIS(3),MIN


C FIND THE POINTS IN CARTESIAN SPACE
         CALL FBPT(PKL1,U1,W1,PT1,JJ)
         CALL FBPT(PKL2,U2,W2,PT2,JJ)

C CALCULATE THE DISTANCE
         DO 20 J=1,3
            DIS(J)=PT1(1,J)-PT2(1,J)
 20      CONTINUE

         PTDIS=0.
         DO 30 K=1,3
              IF(ABS(DIS(K)).LT.0.1E-20) GOTO 30
              PTDIS=PTDIS+(ABS(DIS(K))**2)
 30      CONTINUE

         PTDIS=PTDIS**.5
         MIN=PTDIS


      RETURN
      END
C--------------------------------------------------
C
C      SUBROUTINE: BADCASE
C
C      DESCRIPTION: RECORD THE BAD INTERSECTION CASE, AND
C                   DRAW THE SURFACES AND THE ONLY POINT
C                   IN THE SEGMENT
C
C INPUT:
C      PKL1,PKL2     = INTERSECTING SURFACES
C      TTUW          = INTERSECTION POINTS IN PARAMETRIC SPACE
C      IN            = NO. OF VALID DATA
C
C      C. K. WONG
C      9/12/89
C
      SUBROUTINE BADCASE(PKL1,PKL2,TTUW,IN)

      REAL PKL1(0:3,0:3,3),PKL2(0:3,0:3,3),
     +     TTUW(8,4),TPST(1,3)


C WRITE SURFACE 1 TO FILE
```

```
      WRITE(10,*)'PKL1'
      DO 119 IIII=1,3
         DO 1009 JJJJ=1,4
              WRITE(10,*)PKL1(JJJJ-1,0,IIII),PKL1(JJJJ-1,1,IIII)
     +                   ,PKL1(JJJJ-1,2,IIII),PKL1(JJJJ-1,3,IIII)
 1009    CONTINUE
 119  CONTINUE

      WRITE(10,*)

C WRITE SURFACE 2 TO FILE
      DO 129 IIII=1,3
         DO 1229 JJJJ=1,4
              WRITE(10,*)PKL2(JJJJ-1,0,IIII),PKL2(JJJJ-1,1,IIII)
     +                   ,PKL2(JJJJ-1,2,IIII),PKL2(JJJJ-1,3,IIII)
 1229    CONTINUE
 129  CONTINUE

C DRAW SURFACE 1 & 2
         CALL DSURN(PKL1,2,2)
         CALL DSURN(PKL2,3,2)

C DRAW THE ONLY INTERSECTION POINT IN A SEGMENT
C ON BOTH SURFACES

      DO 1123 JJH=1,IN
      CALL FBPT(PKL1,TTUW(JJH,1),TTUW(JJH,2),TPST,IFF)
      CALL D3PT(TPST,3)

      CALL FBPT(PKL2,TTUW(JJH,3),TTUW(JJH,4),TPST,IFF)
      CALL D3PT(TPST,4)
 1123 CONTINUE


      RETURN
      END
C--------------------------------------------------
C
C     SUBROUTINE: ZPLRC1
C
C     DESCRIPTION: FIND THE ROOTS OF A NDEG POLYNOMIAL
C
C INPUT:
C     NDEG          = DEGREE OF THE POLYNOMIAL
C     COEFF         = COEFF. OF THE POLYNOMIAL
C
C OUTPUT:
C     ZERO          = ROOTS OF THE POLYNOMIAL
C
C     C. K. WONG
C     9/12/89
C


      SUBROUTINE ZPLRC1(NDEG,COEFF,ZERO)

      REAL COEFF(*)
      COMPLEX ZERO(*)
      INTEGER NDEG,N1,N2

C INITIALIZE

      N1=NDEG+1


C FIND THE ZERO COEFF
      DO 10 I=N1,1,-1
```

```
         IF(ABS(COEFF(I)).GT.(0.1E-10)) THEN
              N2=I
              GOTO 20
         ENDIF
         IF(I.EQ.1) THEN

              WRITE(6,*)'ALL COEFF = 0 IN ZPLRC1'
              NDEG=0
              RETURN

         ENDIF
 10   CONTINUE


 20   CALL ZPLRC2(NDEG,COEFF,N2,ZERO)

      RETURN
      END

C--------------------------------------------------
C
C     SUBROUTINE: ZPLRC2
C
C     DESCRIPTION: GET RID OF THE ZERO COEFF AND
C                  FIND THE ROOTS OF A POLYNOMIAL
C
C INPUT:
C     NDEG         = DEGREE OF THE POLYNOMIAL
C     COEFF        = COEFF. OF THE POLYNOMIAL
C     N2           = HIGHEST DEGREE OF NON-ZERO COEFF.
C
C OUTPUT:
C     ZERO         = ROOTS OF THE POLYNOMIAL
C
C     C. K. WONG
C     9/12/89
C


      SUBROUTINE ZPLRC2 (NDEG,COEFF,N2,ZERO)


      REAL COEFF(*)
      INTEGER NDEG,N2
      COMPLEX ZERO(*)
      REAL CO34(34),CO33(33),CO32(32),CO31(31),CO30(30),CO29(29),
     +     CO28(28),CO27(27),CO26(26),CO25(25),CO24(24),CO23(23),
     +     CO22(22),CO21(21),CO20(20),CO19(19),CO18(18),CO17(17),
     +     CO16(16),CO15(15),CO14(14),CO13(13),CO12(12),CO11(11),
     +     CO10(10),CO9(9),CO8(8),CO7(7),CO6(6),CO5(5),CO4(4),
     +     CO3(3),CO2(2)

      COMPLEX ZO34(33),ZO33(32),ZO32(31),ZO31(30),ZO30(29),ZO29(28),
     +     ZO28(27),ZO27(26),ZO26(25),ZO25(24),ZO24(23),ZO23(22),
     +     ZO22(21),ZO21(20),ZO20(19),ZO19(18),ZO18(17),ZO17(16),
     +     ZO16(15),ZO15(14),ZO14(13),ZO13(12),ZO12(11),ZO11(10),
     +     ZO10(9),ZO9(8),ZO8(7),ZO7(6),ZO6(5),ZO5(4),ZO4(3),
     +     ZO3(2),ZO2(1)

C REARRANGE THE POLYNOMIAL: GET RID OF ZERO COEFF.

      IF (N2.EQ.34) THEN
         CALL ZPLRC3(NDEG,COEFF,N2,CO34,ZO34,ZERO)
        ELSEIF(N2.EQ.33) THEN
         CALL ZPLRC3(NDEG,COEFF,N2,CO33,ZO33,ZERO)
        ELSEIF(N2.EQ.32) THEN
         CALL ZPLRC3(NDEG,COEFF,N2,CO32,ZO32,ZERO)
```

```
      ELSEIF(N2.EQ.31) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO31,ZO31,ZERO)
      ELSEIF(N2.EQ.30) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO30,ZO30,ZERO)
      ELSEIF(N2.EQ.29) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO29,ZO29,ZERO)
      ELSEIF(N2.EQ.28) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO28,ZO28,ZERO)
      ELSEIF(N2.EQ.27) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO27,ZO27,ZERO)
      ELSEIF(N2.EQ.26) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO26,ZO26,ZERO)
      ELSEIF(N2.EQ.25) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO25,ZO25,ZERO)
      ELSEIF(N2.EQ.24) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO24,ZO24,ZERO)
      ELSEIF(N2.EQ.23) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO23,ZO23,ZERO)
      ELSEIF(N2.EQ.22) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO22,ZO22,ZERO)
      ELSEIF(N2.EQ.21) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO21,ZO21,ZERO)
      ELSEIF(N2.EQ.20) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO20,ZO20,ZERO)
      ELSEIF(N2.EQ.19) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO19,ZO19,ZERO)
      ELSEIF(N2.EQ.18) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO18,ZO18,ZERO)
      ELSEIF(N2.EQ.17) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO17,ZO17,ZERO)
      ELSEIF(N2.EQ.16) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO16,ZO16,ZERO)
      ELSEIF(N2.EQ.15) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO15,ZO15,ZERO)
      ELSEIF(N2.EQ.14) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO14,ZO14,ZERO)
      ELSEIF(N2.EQ.13) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO13,ZO13,ZERO)
      ELSEIF(N2.EQ.12) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO12,ZO12,ZERO)
      ELSEIF(N2.EQ.11) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO11,ZO11,ZERO)
      ELSEIF(N2.EQ.10) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO10,ZO10,ZERO)
      ELSEIF(N2.EQ.9) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO9,ZO9,ZERO)
      ELSEIF(N2.EQ.8) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO8,ZO8,ZERO)
      ELSEIF(N2.EQ.7) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO7,ZO7,ZERO)
      ELSEIF(N2.EQ.6) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO6,ZO6,ZERO)
      ELSEIF(N2.EQ.5) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO5,ZO5,ZERO)
      ELSEIF(N2.EQ.4) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO4,ZO4,ZERO)
      ELSEIF(N2.EQ.3) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO3,ZO3,ZERO)
      ELSEIF(N2.EQ.2) THEN
       CALL ZPLRC3(NDEG,COEFF,N2,CO2,ZO2,ZERO)

    ENDIF


    RETURN
    END
```

```
C-------------------------------------------------
C
C     SUBROUTINE: ZPLRC3
C
C     DESCRIPTION: GET RID OF THE ZERO COEFF AND
C                  FIND THE ROOTS OF A POLYNOMIAL
C
C INPUT:
C     NDEG         = DEGREE OF THE POLYNOMIAL
C     COEFF        = COEFF. OF THE POLYNOMIAL
C     N2           = HIGHEST DEGREE OF NON-ZERO COEFF.
C     CO           = REARRANGED COEFF. OF THE POLYNOMIAL
C
C
C OUTPUT:
C     ZERO         = ROOTS OF THE POLYNOMIAL
C     ZO           = REARRANGED ROOTS OF THE POLYNOMIAL
C
C     C. K. WONG
C     9/12/89
C

      SUBROUTINE ZPLRC3(NDEG,COEFF,N2,CO,ZO,ZERO)

      REAL COEFF(*), CO(*)
      INTEGER NDEG, N2, NDEG2
      COMPLEX ZO(*), ZERO(*)

      EXTERNAL ZPLRC

C REWRITE COEFF INTO CO

      NDEG2=N2-1
      DO 10 I=N2,1,-1
         CO(I)=COEFF(I)
 10   CONTINUE

      CALL ZPLRC(NDEG2,CO,ZO)

      IDEG=NDEG-NDEG2

C REWRITE ZO INTO ZERO

      DO 20 I=1,NDEG2
         ZERO(I)=ZO(I)
 20   CONTINUE

      NDEG=NDEG2

      RETURN
      END
```

Vita

Chee Kiang Wong was born on July 14, 1965 in Seremban, Malaysia along with a twin brother. After receiving his Bachelor of Science Degree in Mechanical Engineering from University of Southwestern Louisiana, he discovered that the CAD/CAM field is fascinating. Thus he came to Virginia Tech for graduate study in the Fall of 1988 . He will receive a Master of Science degree in Mechanical Engineering in the Fall of 1990.