Filleting of Aircraft Components Using Non-Uniform B-Spline Surfaces

by

James R. Gloudemans

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:

A. Myklebust, Chairman

S. G. Dhande

M. P. Deisenroth

May 29, 1989

Blacksburg, Virginia

Filleting of Aircraft Components Using Non-Uniform B-Spline Surfaces

by

James R. Gloudemans

A. Myklebust, Chairman

Mechanical Engineering

(ABSTRACT)

Conceptual and preliminary aircraft design codes have different geometry requirements. Conceptual design codes use component based models, while preliminary design codes require a more complete and integrated geometry. An automatic procedure to convert between the two types of models would prevent geometry inconsistencies and speed up the design process. This thesis describes some of the concepts and algorithms necessary to develop such a procedure. Specifically, the theory and development of C^2 continuous fillets between aircraft components is discussed. B-spline surface generation from conceptual geometry data and the relimiting of the surfaces is also presented.

Acknowledgements

I would like to thank Kolady, Dr. J, the Butcher, and the rest of the ACSYNT team for for their advice and constant support. Special thanks to Dr. Myklebust for his guidance, encouragement, and patience. Thanks are also extended to Dr. Dhande and Dr. Deisenroth.

Table of Contents

List of Illustrations

Chapter 1

Introduction and Literature Review

1.1 Introduction

Aircraft geometry requirements differ for conceptual and preliminary design codes. Conceptual design codes, such as NASA's AirCraft SYNThesis (ACSYNT), need surface models for individual components. These models are used to generate wireframe and shaded images so the designer can easily visualize the new aircraft. In order to keep the design process as interactive as possible, the aircraft models must be quickly generated and easily modified.

Preliminary design codes require a more complete and integrated geometry description. The geometry is used for detailed aerodynamic and radar cross-section analysis, thus requiring the surface patches to meet with positional, gradient, and curvature continuity

(C^2 continuity). In addition, C^2 fillets may be required between individual components such as the wing and fuselage.

A procedure to convert between conceptual and preliminary geometry formats would speed up the design process and prevent any inconsistencies between the two models. This procedure requires four steps: definition of the C^2 continuous surfaces for each component; computation of the intersection between components; filleting between components when required; and relimiting of any unwanted surface patches. Figure 1 on page 3 illustrates the conversion procedure for aircraft models in ACSYNT. Wong (1990) is investigating methods to compute the intersection of aircraft components.

1.2 Objectives and Overview

The objective of this thesis is to develop and test procedures which will generate C^2 continuous fillets between aircraft components. The fillets are created using non-uniform cubic B-spline surfaces. Several innovative algorithms will be presented to produce the fillets. These algorithms include non-uniform B-spline inversion, fixed control vertex inversion, and control vertex filleting. The algorithms are tested by integrating coded filleting procedures into a developmental version of ACSYNT.

This thesis emphasizes the concepts and algorithms required to produce C^2 continuous fillets. Surface definition, with C^2 continuity, and relimiting are also discussed in detail. The first chapter covers the basics of B-spline curves and surfaces. The following chapter describes the algorithms required to generate curve fillets. Chapter four extends

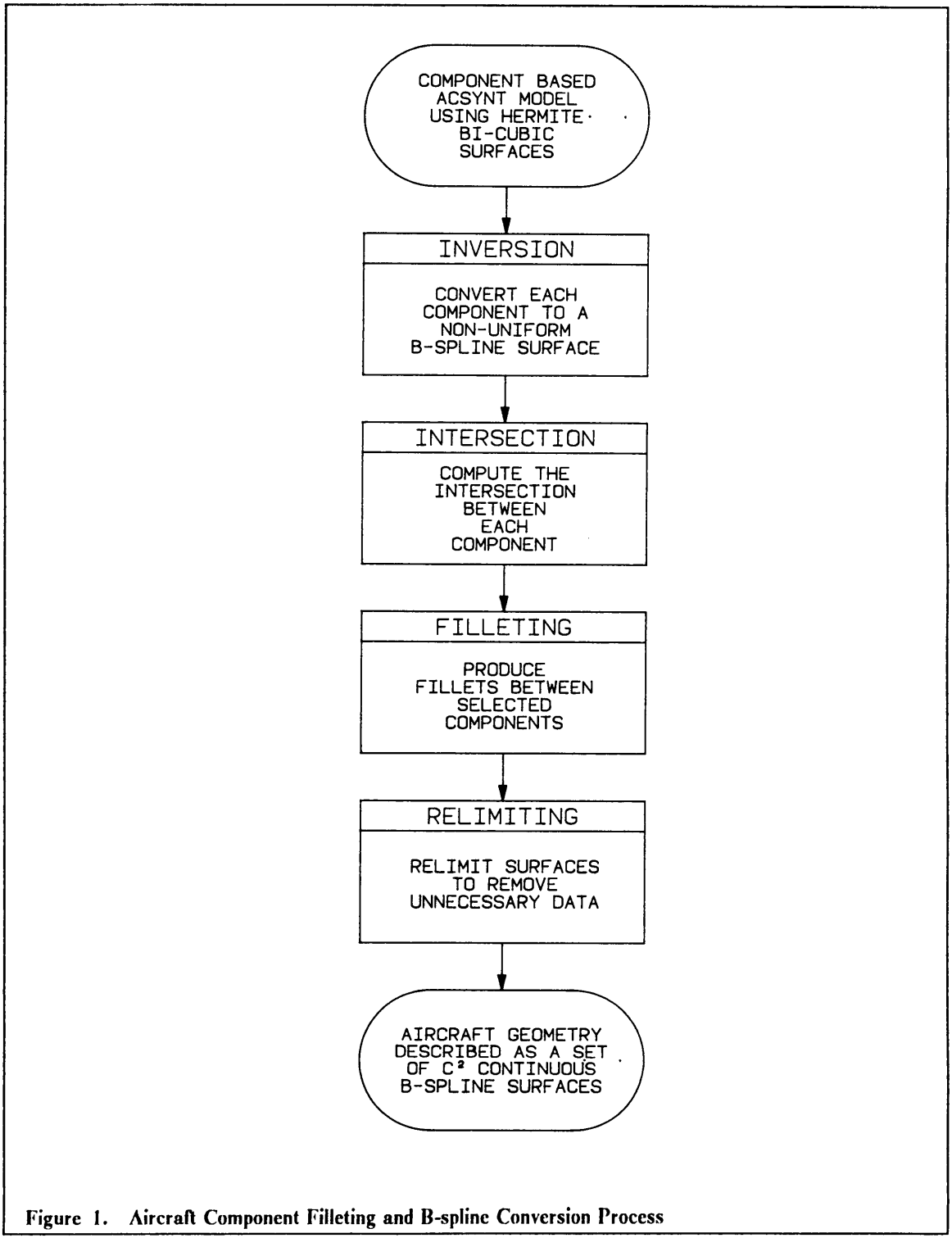


Figure 1. Aircraft Component Filleting and B-spline Conversion Process

curve filleting theory to surfaces and describes two types of surface fillets. The last chapter discusses the current level of B-spline surface and filleting integration within ACSYNT. Recommendations for future work and related research are also listed in the last chapter.

1.3 Literature Review

Two examples of conceptual design codes are ACSYNT and HESCOMP. ACSYNT uses Hermite bi-cubic surface patches to define the aircraft geometry (Wampler, et al, 1988). Each component is defined individually from a series of cross-sections. There is no method of guaranteeing that components will meet with curvature, slope, or even positional continuity. The surface patches within each component are only insured of C^1 continuity if the tangent vectors are properly matched.

HESCOMP is a helicopter design code used for sizing and performance analysis. HESCAD is the interactive CADAM interface designed to produce 3-D helicopter wire-frame images from the output of the design code (Lu, et al, 1987). The resulting geometry model is a collection of cross-sections and curve segments linking each cross-section. Because the surfaces are not completely defined, this geometry may not be suitable for many preliminary design codes.

Computational Fluid Dynamics (CFD) is an example of a class of preliminary design codes with specialized geometry requirements. The geometry must contain surface information which matches the actual geometry as closely as possible. CFD surface

geometry definition and preparation is currently not a routine process and there is a demand for easily applied geometry generation procedures. (Kutler, 1986; Rubbert, 1988)

One method of generating the required CFD geometry data is outlined by Edwards (1988). A commercial computer-aided design/computer-aided manufacturing (CAD/CAM) system is used to define the various aircraft components. The surfaces are then "manually" trimmed and filleted. The resulting surfaces are cut with a plane and points are determined along the curve of intersection. These points are interpolated with a B-spline to form a series of C^2 continuous cross-sections. The complete surface grid is constructed by parametric curve sampling each B-spline cross-section.

The effect of a fillet on the flow past a wing/fuselage juncture is presented by Kubendran et al (1988). This paper describes experiments which indicate that in certain cases, fillets provide "a sizable reduction in the juncture drag". The geometry of the fillet is not described in detail.

There are a few references to fillets and blends in the geometric modeling literature. Pegna and Wolter (1989) present a criteria used to guarantee that a blended surface is second order smooth. They prove that C^2 -smooth blends can be generated when the linkage or intersection curve is first order smooth. Practical algorithms detailing how to produce C^2 -smooth blends are not included in this paper.

Several techniques used to produce fillets and blends are reviewed by Rockwood (1983). These algorithms include the "rolling ball" technique and Chaiken chamfering. These methods do not guarantee C^2 continuous blends and may require specialized input geometry. In another paper by Rockwood (1989), a method of filleting called

displacement blending is presented. The blending surfaces are implicitly defined and are C^1 continuous.

Warren (1989) discusses the problem of blending algebraic surfaces. His paper proposes a measure of surface smoothness and describes the theory behind the new blending method. This method is limited to blending a limited set of common algebraic surfaces including cylinders, cones, and spheres.

Chapter 2

B-Splines

2.1 Splines

One of the most important curves in the aircraft industry is the spline (Mortenson, 1985). Splines are important because they solve the "connection" problems inherent with other types of piecewise curves. Spline curves were first drawn using long, thin strips of plastic or steel flexed to pass through a series of points. The resulting curve has a gradual rate of curvature change and no kinks.

A spline curve can be represented using mathematical functions. There are a number of different representations including parametric splines, Bezier splines and B-splines. For the reasons discussed in the following sections, B-splines have several advantages over the other representations.

2.2 *Uniform B-spline Curves*

Basis splines or B-splines were first introduced by I. J. Schoenburg in 1946 (Farin, 1988). Gordon and Riesenfeld then used a recursive blending function to define a general parametric B-spline. C. de Boor, W. Boehm, and many other researchers have contributed to the development of B-splines.

As shown in Figure 2 on page 9, B-splines are defined by a set of points called control vertices. The B-spline curve approximates the vertices, but does not pass through them. If the control vertices are connected together, they form a control polygon or control hull. The general shape of the curve can be approximated from the shape of the control polygon.

Any point on the B-spline is a weighted average of a finite number of neighboring control vertices. Blending functions determine how much each vertex affects the curve at any given parametric value. These functions blend the effects or contributions of each control vertex. Figure 3 on page 10 plots the blending functions for a uniform cubic B-spline curve. The general notation for a B-spline blending function is $N_{i,M}$, where i corresponds to a control vertex and the degree of the curve is equal to $M-1$. If the parameter value (u) varies uniformly along the entire curve, the B-spline is called uniform. The formula for a uniform cubic B-spline is presented below in matrix form.

$$\mathbf{p}_i(u) = UMQ_i$$

where:

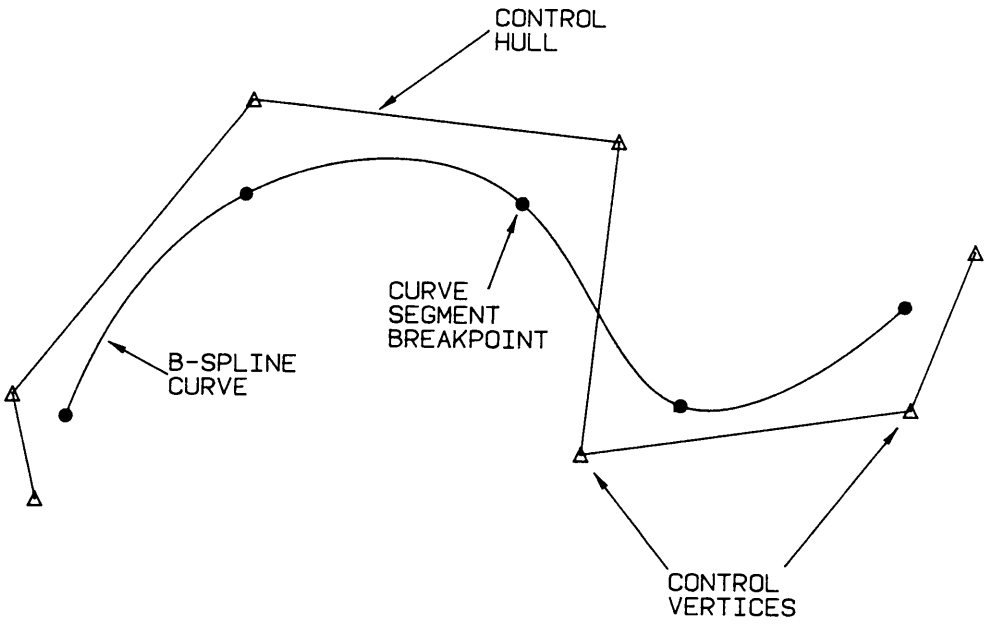


Figure 2. B-spline Curve with Control Vertices

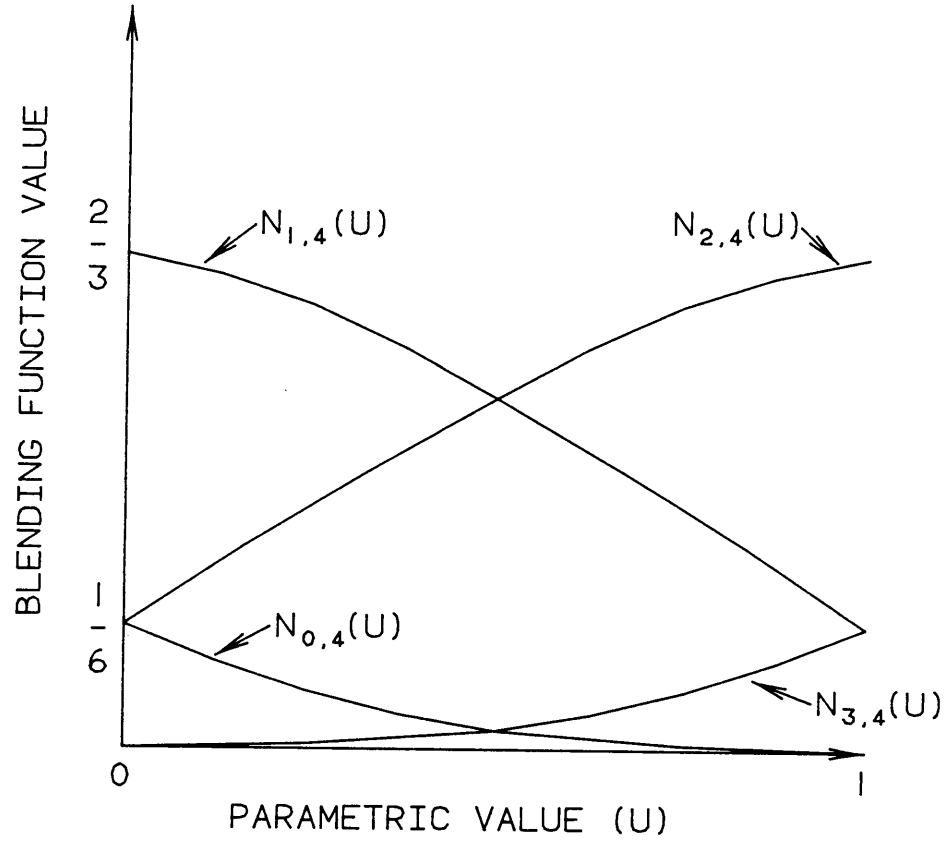


Figure 3. Uniform Cubic B-spline Blending Functions

$\mathbf{p}_i(u)$ is a point on the B-spline curve

u is the parametric variable $(0 \leq u \leq 1)$

i is the curve segment index $(i = 1, 2, ..., n)$

U is the parametric variable matrix

M is the blending function matrix

Q_i is the control vertex matrix

$$U = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \quad M = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \quad Q_i = \begin{bmatrix} \mathbf{q}_{i-1} \\ \mathbf{q}_i \\ \mathbf{q}_{i+1} \\ \mathbf{q}_{i+2} \end{bmatrix}$$

B-splines are defined in a piecewise manner. Each piece is a spline curve segment defined by a number of surrounding control vertices. A cubic B-spline segment is defined by four control vertices and if there are N control vertices, then N-3 curve segments define the entire cubic B-spline.

2.3 Non-Uniform B-splines

Shape problems can occur when a uniform B-spline curve is used to interpolate unevenly spaced points. Therefore, it is desirable to formulate a B-spline curve with non-uniform knot spacing. Knots relate the parameter value (u) to the control points. The following

non-uniform B-spline formulation utilizes recursion to define the curve blending functions.

$$\mathbf{p}(u) = \sum_{i=0}^{n} \mathbf{q}_i N_{i,M}(u)$$

where:

$\mathbf{p}(u)$ is a point on the B-spline curve

(n+1) is the number of control vertices

$\mathbf{q}_i$ are the control vertices

$N_{i,M}$ are the blending functions

The blending functions are defined recursively as shown below.

$$N_{1,M}(u) = \left\{ \begin{array}{l} 1\ (t_i \le u < t_{i+1}) \\ 0\ (\textit{outside above range}) \end{array} \right\}$$

$$N_{i,M}(u) = \frac{u - t_i}{t_{i+M-1} - t_i} N_{i,M-1} + \frac{t_{i+M} - u}{t_{i+M} - t_{i+1}} N_{i+1,M-1}$$

where:

t_i is the knot value

u is the parametric variable ($t_1 \le u < t_{n-1}$)

M controls the degree (M - 1) of the curve

Knot values are required at every control vertex. For a cubic B-spline, one additional knot value is needed for the beginning and end of the curve. The relationship between the number of knots, control vertices, and breakpoints for a cubic B-spline is shown in Figure 4 on page 14. The knot sequence must be non-decreasing and since the

denominators can be zero, the convention of 0/0 = 0 is used. Figure 5 on page 15 shows the effect of changing the knot spacing for a non-uniform B-spline curve.

C. de Boor (1972) presents a non-recursive method for calculating non-uniform B-spline blending functions. This algorithm is stable and handles coincident knots without difficulty. Because recursion is not a feature of Fortran 77, the non-recursive de Boor algorithm is used for all blending function calculations. The subroutine "BLENDT" in Appendix C contains the Fortran 77 code listing of this algorithm.

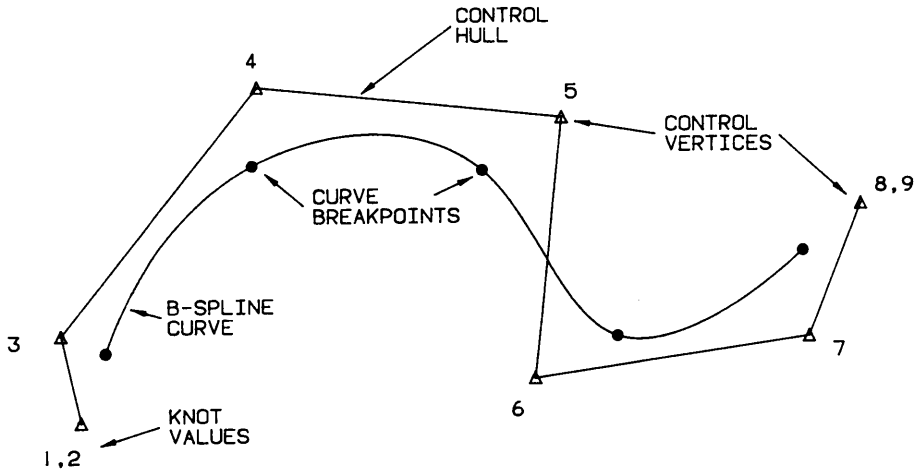
2.4 Properties

B-splines have several important properties which make them useful for practical CAD applications (Yamaguchi, 1988)

- C^2 Continuity: At the points where cubic B-splines segments are connected, the slope and curvature vectors are continuous for any given set of non-coincident control points and knot values.

- Variation Diminishing Property: A B-spline curve never intersects an arbitrary straight line more times than its control polygon. Consequently, the B-spline curve does not contain any variations that are not in the general shape of the control polygon.

- Local Shape Control: If one control vertex is moved, the shape change of the B-spline curve is locally confined. For example, a cubic B-spline curve segment is



NUMBER OF CURVE SEGMENTS = (# OF CONTROL VERTICES - DEGREE) = 4

NUMBER OF CURVE BREAKPOINTS = (# OF CURVE SEGMENTS +1) = 5

NUMBER OF CONTROL VERTICES = (# OF INTERPOLATED POINTS + DEGREE - 1) = 7

NUMBER OF KNOTS = (# OF CONTROL VERTICES + DEGREE - 1) = 9

Figure 4. Cubic B-spline Curve

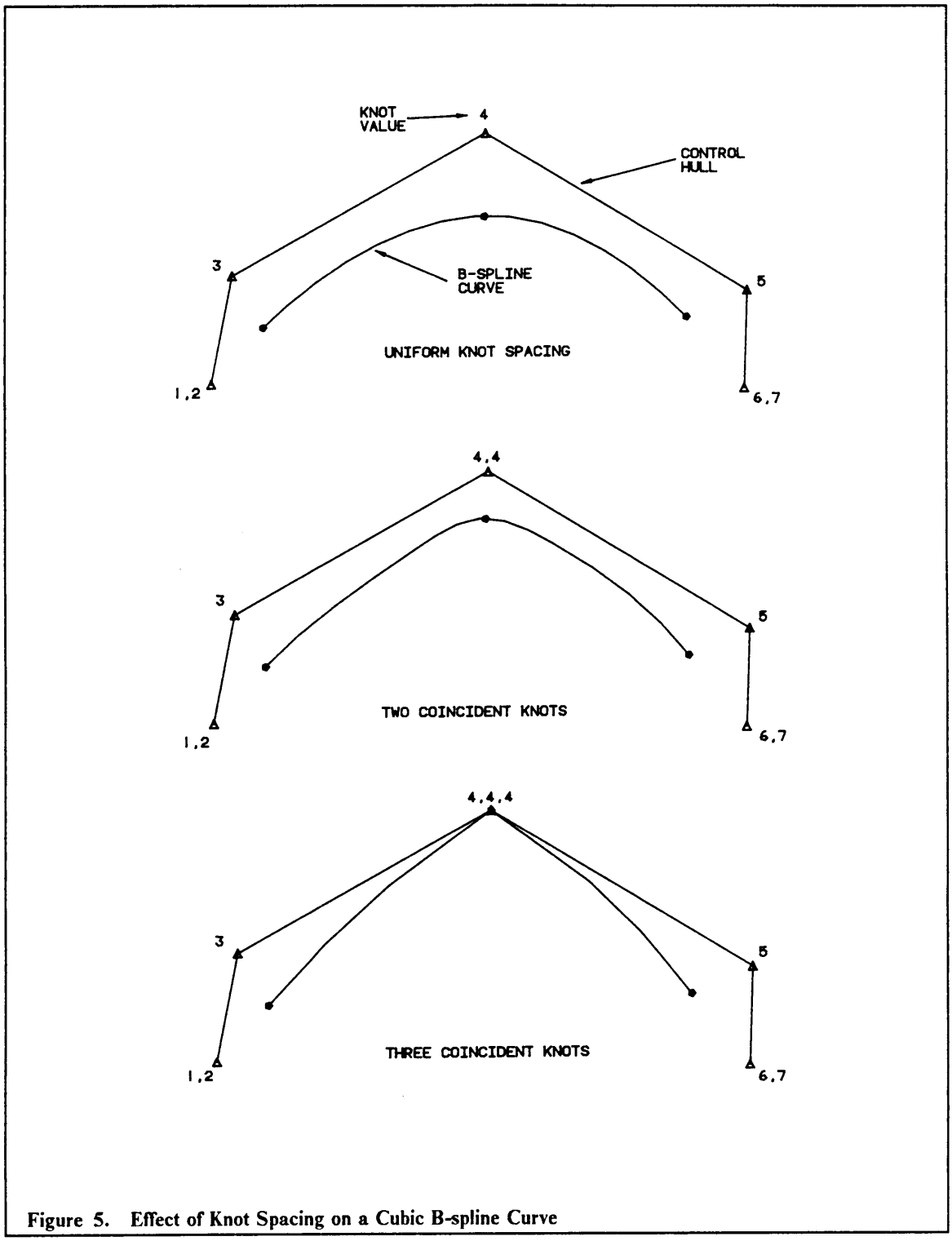


Figure 5. Effect of Knot Spacing on a Cubic B-spline Curve

defined by only four adjacent control vertices. Figure 6 on page 17 illustrates the property of local shape control.

- Curve Degree Control: Unlike Bezier curves, the degree of a B-spline curve is independent of the number of control vertices.

2.5 Uniform Cubic B-spline Inversion

It is often necessary to find the control hull of a uniform B-spline curve which passes through a given set of points. This is called the inverse transformation or B-spline inversion process (Yamaguchi, 1988).

Uniform B-spline Inversion Process

Given: Points to be interpolated

$$\mathbf{p}_i \quad (i = 1, 2, ..., n)$$

Find: Control vertices

$$\mathbf{q}_i \quad (i = 0, 1, ..., n, n+1)$$

The inverse problem can be solved by substituting $u=0$ into the uniform cubic B-spline function. This yields the following set of simultaneous equations:

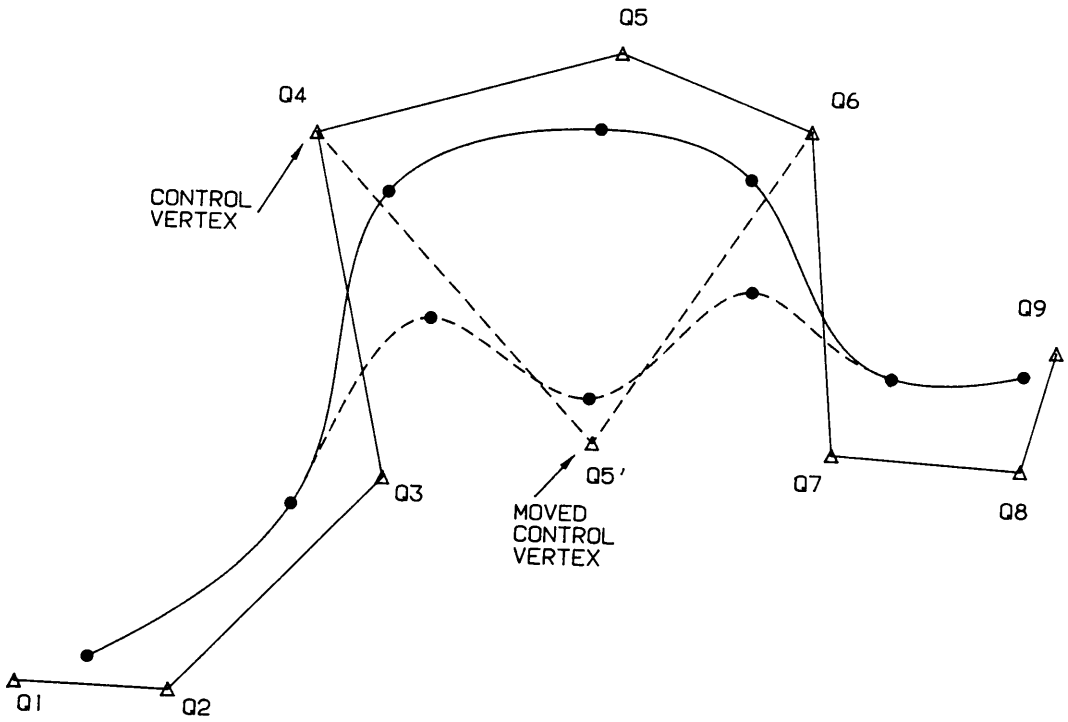


Figure 6. Local Shape Control of a B-spline Curve

$$\frac{1}{6}\mathbf{q}_{i-1} + \frac{2}{3}\mathbf{q}_i + \frac{1}{6}\mathbf{q}_{i+1} = \mathbf{p}_i \qquad (i = 1, 2, ..., n)$$

Since there are two fewer equations than unknowns, the end conditions need to be specified. As shown in Figure 7 on page 19, the interpolated curves can be open or closed.

End conditions for open curves:

$$\mathbf{q}_0 = \mathbf{q}_1 \quad \textit{and} \quad \mathbf{q}_{n+1} = \mathbf{q}_n$$

End conditions for closed curves:

$$\mathbf{q}_0 = \mathbf{q}_n \quad \textit{and} \quad \mathbf{q}_{n+1} = \mathbf{q}_1$$

These equations can be solved by either matrix methods or iteratively. Yamaguchi (1988) details an iterative algorithm which converges quickly and is easy to program. There are three main steps to this algorithm:

Step 1: Make an initial guess for the control hull by setting the control vertices equal to the points to be interpolated. Also set the boundary conditions by specifying either an open or closed curve.

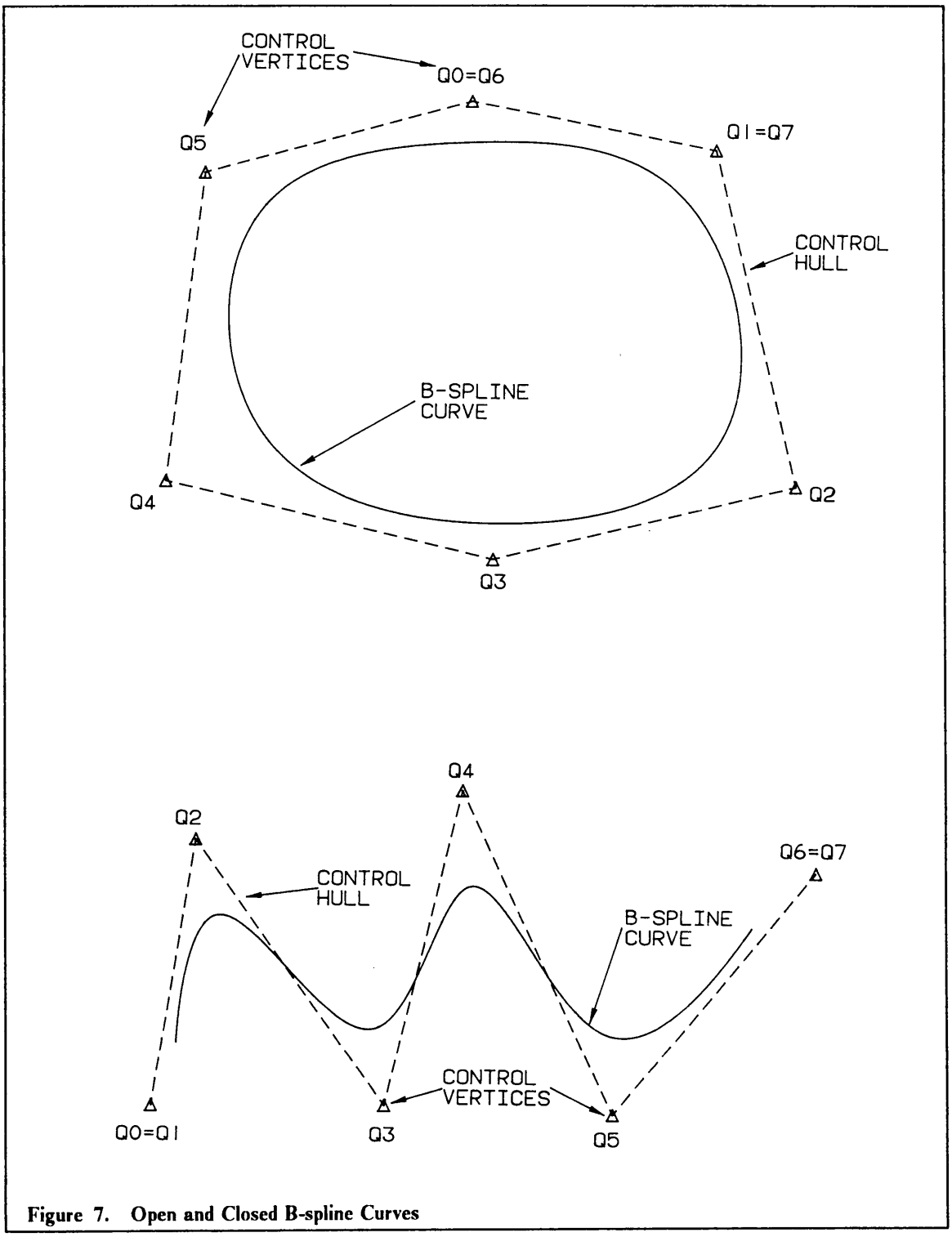


Figure 7. Open and Closed B-spline Curves

$$\mathbf{q}_i = \mathbf{p}_i \qquad (i = 1, 2, ..., n)$$

Step 2: Find the difference between the k-th and (k-1)-st iterations of $\mathbf{q}_i$ and calculate a new value for $\mathbf{q}_i$.

$$\delta_i^k = \mathbf{p}_i + \frac{1}{2}\left\{\mathbf{p}_i - \mathbf{q}_i^{k-1} + \frac{1}{2}\left(\mathbf{q}_{i-1}^k + \mathbf{q}_{i+1}^{k-1}\right)\right\}$$

$$\mathbf{q}_i^k = \delta^{ik} + \mathbf{q}_i^{k-1}$$

Step 3: If the maximum difference (max δ_i) is greater than the allowable error then repeat Step 2.

2.6 Non-Uniform Cubic B-spline Inversion

The problem of finding the inverse transformation for a non-uniform cubic B-spline is slightly more complex than the uniform case. The first step is to choose a suitable parameter or knot spacing. There are several techniques for selecting knot spacing including uniform, chord length, centripetal, and Foley parameterization. Section 2.7 discusses parameterization in greater detail. A non-uniform cubic B-spline inversion method is outlined below.

Non-uniform B-spline Inversion Process

Given: Points to be interpolated and knot values

$$\mathbf{p}_i \qquad (i = 1, 2, ..., n)$$

$$k_i \qquad (i = -1, 0, ..., n, n+1, n+2)$$

Find: Control vertices

$$\mathbf{q}_i \qquad (i = 0, 1, ..., n, n+1)$$

Start with the equation for a cubic non-uniform B-spline curve.

$$\mathbf{p}(u) = \sum_{i=0}^{n} \mathbf{q}_i N_{i,4}(u)$$

$$\mathbf{p}(u) = N_{0,4}(t_i)\mathbf{q}_{i-1} + N_{1,4}(t_i)\mathbf{q}_i + N_{2,4}(t_i)\mathbf{q}_{i+1} + N_{3,4}(t_i)\mathbf{q}_{i+2}$$

Substitute in u = t_i and the fourth blending function component drops out.

$$N_{3,4}(t_i) = \frac{t_i - t_i}{t_{i+3} - t_i} N_{3,3} + \frac{t_{i+4} - t_i}{t_{i+4} - t_{i+1}} N_{4,3}$$

$$N_{4,3}(t_i) = \frac{t_{i+4} - t_i}{t_{i+4} - t_{i+1}} N_{5,2}$$

$$N_{5,2}(t_i) = \frac{t_{i+4} - t_i}{t_{i+4} - t_{i+1}} N_{6,1}$$

$$N_{6,1} = 0$$

Use the knot values to solve for the blending functions at each control vertex. To find the control hull, the following set of simultaneous equations is solved.

$$N_{0,4}(t_i)\mathbf{q}_{i-1} + N_{1,4}(t_i)\mathbf{q}_i + N_{2,4}(t_i)\mathbf{q}_{i+1} = \mathbf{p}_i \quad (i = 1, 2, ..., n)$$

where:

$N_{0,4}(t_i)$, $N_{1,4}(t_i)$, $N_{2,4}(t_i)$ are the blending functions

$\mathbf{q}_i$ is the i-th control vertex

The iterative technique explained in Section 2.5 is used to solve these equations. The difference between the k-th and (k-1)-st iterations is derived below.

$$\delta_i^k = \mathbf{q}_i^k - \mathbf{q}_i^{k-1}$$

$$\mathbf{p}_i = N_{0,4}(t_i)\mathbf{q}_{i-1}^k + N_{1,4}(t_i)\mathbf{q}_i^k - N_{2,4}(t_i)\mathbf{q}_{i+1}^{k-1}$$

Solving the second equation for $\mathbf{q}_i^k$ and back-substituting into the first equation yields:

$$\delta_i^k = -\mathbf{q}_i^{k-1} + \frac{1}{N_{1,4}(t_i)}\left[\mathbf{p}_i - N_{0,4}(t_i)\mathbf{q}_{i-1}^k - N_{2,4}(t_i)\mathbf{q}_{i+1}^{k-1}\right]$$

This algorithm is listed in code form in Appendix C.

2.7 Parameterization Techniques

Parameterization is the process of selecting the knot values for a given set of data points. There are four common methods of parameterization; uniform, chord length, centripetal and Foley (Farin, 1988). Each of these methods are described below.

- Uniform: The spacing between knot values is constant. Uniform knot spacing works well if the interpolated points are evenly spaced. Interpolating unevenly spaced data points with uniform parameterization can produce B-spline curves with undesirable shapes and large curvature fluctuations.

- Chord length: The spacing between knot values is proportional to the distance between data points. In most cases, chord length parameterization produces better results than uniform knot spacing.

- Centripetal: The spacing between knot values is proportional to the square root of the distance between data points. This method attempts to minimize the centripetal force that would act on a point moving along the curve. Centripetal parameterization produces smooth, consistent curves and is used in the filleting algorithms in this thesis.

- Foley: Two factors affect the knot spacing for Foley parameterization; the distance between the data points and the angle formed by the line segments connecting the data points. The curves produced by this parameterization method are similar to the curves calculated using the centripetal method.

2.8 Bi-cubic B-spline Surfaces

The formulation of a B-spline surface is an extension of the B-spline curve description. As shown in Figure 8 on page 25, the control polyhedron or control hull determines the shape of the surface. The uniform bi-cubic B-spline surface formulation is listed below.

$$\mathbf{p}_{i,j}(u, w) = UMQM^TW^T$$

where:

$\mathbf{p}_{i,j}$ - point on the B-spline surface

U - parametric variable matrix

M - blending function matrix

Q - control vertex matrix

W - parametric variable matrix

$$U = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \qquad W = \begin{bmatrix} w^3 & w^2 & w & 1 \end{bmatrix}$$

$$M = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \qquad Q = \begin{bmatrix} \mathbf{q}_{i-1,j-1} & \mathbf{q}_{i-1,j} & \mathbf{q}_{i-1,j+1} & \mathbf{q}_{i-1,j+2} \\ \mathbf{q}_{i,j-1} & \mathbf{q}_{i,j} & \mathbf{q}_{i,j+1} & \mathbf{q}_{i,j+2} \\ \mathbf{q}_{i+1,j-1} & \mathbf{q}_{i+1,j} & \mathbf{q}_{i+1,j+1} & \mathbf{q}_{i+1,j+2} \\ \mathbf{q}_{i+2,j-1} & \mathbf{q}_{i+2,j} & \mathbf{q}_{i+2,j+1} & \mathbf{q}_{i+2,j+2} \end{bmatrix}$$

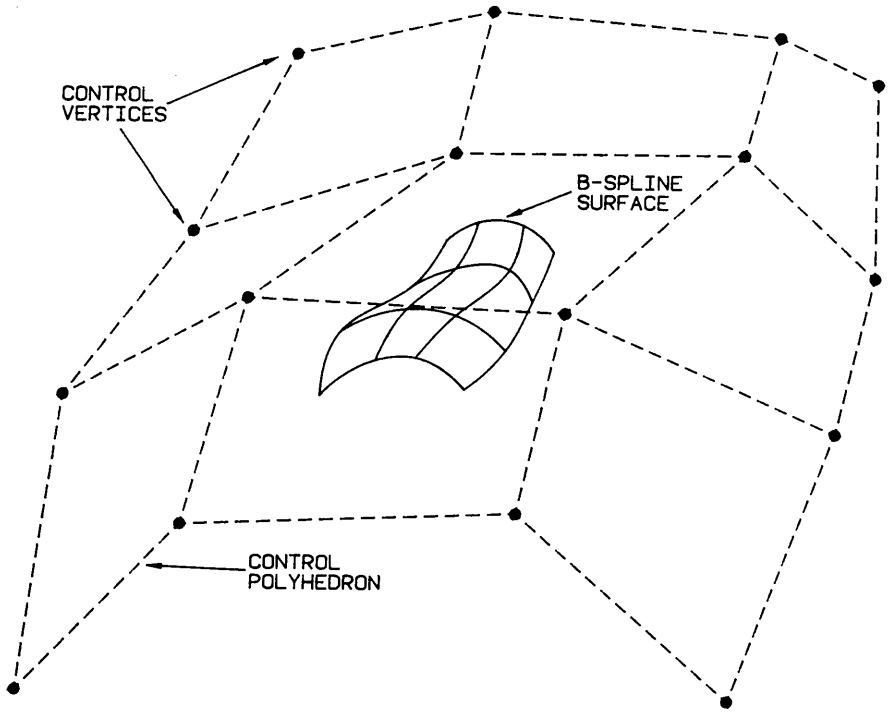


Figure 8. B-spline Surface with Control Polyhedron

The equation for a non-uniform B-spline surface is listed below. The surface blending functions are calculated with the same methods used to compute curve blending functions.

$$\mathbf{p}_{i,j}(u,w) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,M}(u) N_{j,L}(w) Q_{i,j}$$

A cubic non-uniform B-spline surface can be expressed in matrix form.

$$\mathbf{p}_{i,j}(u,w) = [N_{0,4}(u)\ ,\ N_{1,4}(u)\ ,\ N_{2,4}(u)\ ,\ N_{3,4}(u)] \begin{bmatrix} \mathbf{q}_{i-1,j-1} & \mathbf{q}_{i-1,j} & \mathbf{q}_{i-1,j+1} & \mathbf{q}_{i-1,j+2} \\ \mathbf{q}_{i,j-1} & \mathbf{q}_{i,j} & \mathbf{q}_{i,j+1} & \mathbf{q}_{i,j+2} \\ \mathbf{q}_{i+1,j-1} & \mathbf{q}_{i+1,j} & \mathbf{q}_{i+1,j+1} & \mathbf{q}_{i+1,j+2} \\ \mathbf{q}_{i+2,j-1} & \mathbf{q}_{i+2,j} & \mathbf{q}_{i+2,j+1} & \mathbf{q}_{i+2,j+2} \end{bmatrix} \begin{bmatrix} N_{0,4}(w) \\ N_{1,4}(w) \\ N_{2,4}(w) \\ N_{3,4}(w) \end{bmatrix}$$

B-spline surfaces have all of the properties and benefits associated with B-spline curves. The only additional requirement is that the control hull vertices form a rectangular matrix.

2.9 *Surface Inversion*

The method of finding a B-spline surface which passes through a lattice of points is an extension of the curve inversion technique (Yamaguchi, 1988). First, each row of points is inverted using the inverse transformation algorithm listed in Section 2.5 or in Section 2.6. The resulting control vertices from the first inversion are inverted again in the other "parametric" direction. An example of B-spline surface inversion is presented below.

Uniform B-spline Surface Inversion Example

Given: Grid of points to interpolate

$$w \rightarrow$$

$$u \downarrow \begin{bmatrix} \mathbf{p}(1,1) & \cdots & \mathbf{p}(1,5) \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \mathbf{p}(5,1) & \cdots & \mathbf{p}(5,5) \end{bmatrix}$$

Find: The control hull for a uniform B-spline surface

Step 1: Invert the points in the u parametric direction to find intermediate control points. Use the curve inversion algorithms explained in Section 2.5 and Section 2.6.

$$
\begin{array}{c}
 & w \rightarrow \\
u \downarrow & \begin{bmatrix}
\mathbf{q}_{int}(1,1) & \cdots & \mathbf{q}_{int}(1,5) \\
\vdots & \cdots & \vdots \\
\vdots & \cdots & \vdots \\
\vdots & \cdots & \vdots \\
\vdots & \cdots & \vdots \\
\vdots & \cdots & \vdots \\
\mathbf{q}_{int}(7,1) & \cdots & \mathbf{q}_{int}(7,5)
\end{bmatrix}
\end{array}
$$

Step 2: Invert the intermediate control points in the w parametric direction to find the control hull for the surface.

$$
\begin{array}{c}
 & w \rightarrow \\
u \downarrow & \begin{bmatrix}
\mathbf{q}(1,1) & \cdots & \mathbf{q}(1,7) \\
\vdots & \cdots & \vdots \\
\vdots & \cdots & \vdots \\
\vdots & \cdots & \vdots \\
\vdots & \cdots & \vdots \\
\vdots & \cdots & \vdots \\
\mathbf{q}(7,1) & \cdots & \mathbf{q}(7,7)
\end{bmatrix}
\end{array}
$$

Chapter 3

Curve Filleting

3.1 Definition and Overview

A fillet is defined as a surface which joins two surfaces smoothly, but without stringent shape requirements (Rockwood, 1983). In many cases, the fillets used in aircraft and ship applications have exact requirements for smoothness. For such applications, the fillets must blend into the adjoining surfaces with C^2 continuity. The first step in understanding a surface fillet is to study curve fillets.

The objective of this chapter is to present a fast and efficient method of calculating curve fillets. This algorithm will join any two B-spline curves with a C^2 continuous B-spline fillet curve. The first section describes how the slope and curvature of the fillet are blended to the curves at the points of connection. The next section reviews knot

insertion. Knot insertion allows the fillet to join the curves at arbitrary points. The last section introduces a method of specifying the shape of the fillet.

3.2 Control Vertex Filleting

Control vertex filleting utilizes the control vertices of two B-spline curves to join them with a smooth fillet. Because a C^2 continuous fillet is desired, three control vertices are used from each uniform cubic B-spline curve. The equations below show that three control vertices ($\mathbf{q}_{i-1}$, $\mathbf{q}_i$, and $\mathbf{q}_{i+1}$) are the only contributors to position, slope, and curvature at any arbitrary curve segment breakpoint.

$$\mathbf{p}(0)_i = \frac{1}{6}\mathbf{q}_{i-1} + \frac{2}{3}\mathbf{q}_i + \frac{1}{6}\mathbf{q}_{i+1}$$
$$\mathbf{p}_u(0)_i = -\frac{1}{2}\mathbf{q}_{i-1} + \frac{1}{2}\mathbf{q}_{i+1}$$
$$\mathbf{p}_{uu}(0)_i = \mathbf{q}_{i-1} - 2\mathbf{q}_i + \mathbf{q}_{i+1}$$

Figure 9 on page 31 shows the steps required to construct a fillet by using the control vertex filleting method. First select the locations where the fillet should blend with the curves; these locations are labeled "Fillet Join" in Figure 9. The control vertices used to generate the fillet are the three control vertices which describe the curve at the fillet join location. There are two problems associated with this type of simplified filleting.

- The fillet can only join the two B-splines at curve segment breakpoints.
- There is no way to control the shape of the fillet.

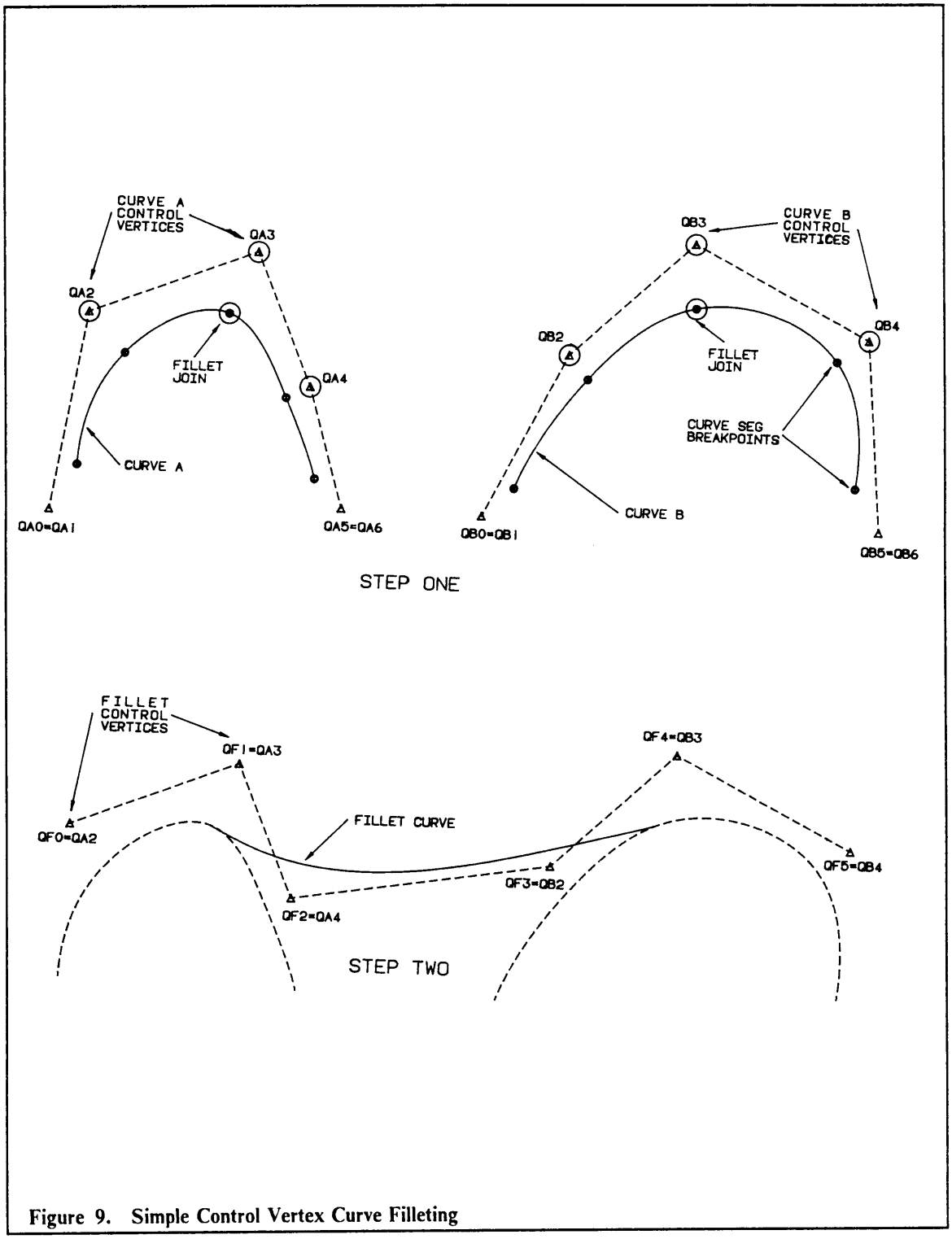


Figure 9. Simple Control Vertex Curve Filleting

The next two sections present solutions to these problems.

3.3 Knot Insertion

Control vertex filleting is restricted to joining the two curves at breakpoints. Therefore, new breakpoints must be added to the B-spline description to allow general filleting. There are two potential methods of adding breakpoints and control vertices to a B-spline curve.

- Add a point to the original curve and perform an inverse transformation to find the new control hull.

- Insert a knot into the B-spline curve description

Figure 10 on page 33 shows the problem associated with re-inverting a B-spline curve. There is no guarantee that the new uniform curve will match the original and the curve shape variations are not locally confined. Non-uniform parameter spacing may help in matching the curves, but determining the correct knot spacing is a difficult problem.

Knot insertion is a simple method of adding control vertices to the B-spline curve description without changing the shape of the curve. A new curve segment breakpoint can be added anywhere along the curve by adding a knot at the desired parameter value. W. Boehm (1980) has developed a general knot insertion algorithm which is presented below. This algorithm is adjusted to account for the array subscripts in the computer

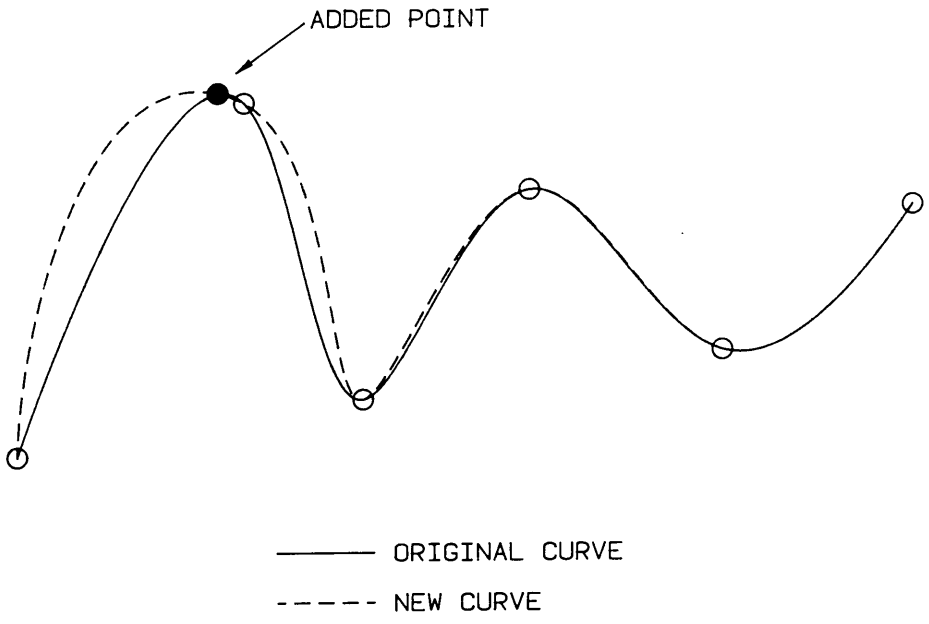


Figure 10. Curve Matching Problem Associated with Re-inversion

code. For an illustration of how knot insertion affects the control hull see Figure 11 on page 35.

Knot Insertion:

$$\mathbf{q}_{newj} = (1 - \alpha_j)\mathbf{q}_{j-1} + \alpha_j\mathbf{q}_j$$

where:

$\mathbf{q}_j$ are the old control vertices

$\mathbf{q}_{newj}$ are the new control vertices

$\alpha_j = 1 \quad \text{if } j \leq i - 2$

$\alpha_j = \dfrac{t_{new} - t_{j-1}}{t_{j+2} - t_{j-1}} \quad \text{if } i - 1 \leq j \leq i + 1$

$\alpha_j = 0 \quad \text{if } j \geq i + 2$

i is the interval where the knot is added

Knot insertion allows a fillet to connect two curves at an arbitrary point or parameter value along the curve. Listed below are the steps required to produce a C^2 continuous fillet utilizing knot insertion.

1. Select the parameter value (u) for each curve where the fillet will join each B-spline curve. In Figure 12 on page 37, the fillet join locations are u = 3.4 on curve A and u = 4.6 on curve B.

2. Insert a knot to each curve at the selected parameter values.

3. Add four additional knots to each curve at a constant parameter spacing away from the knot inserted in Step 2; two on each side of the selected parameter value. In the

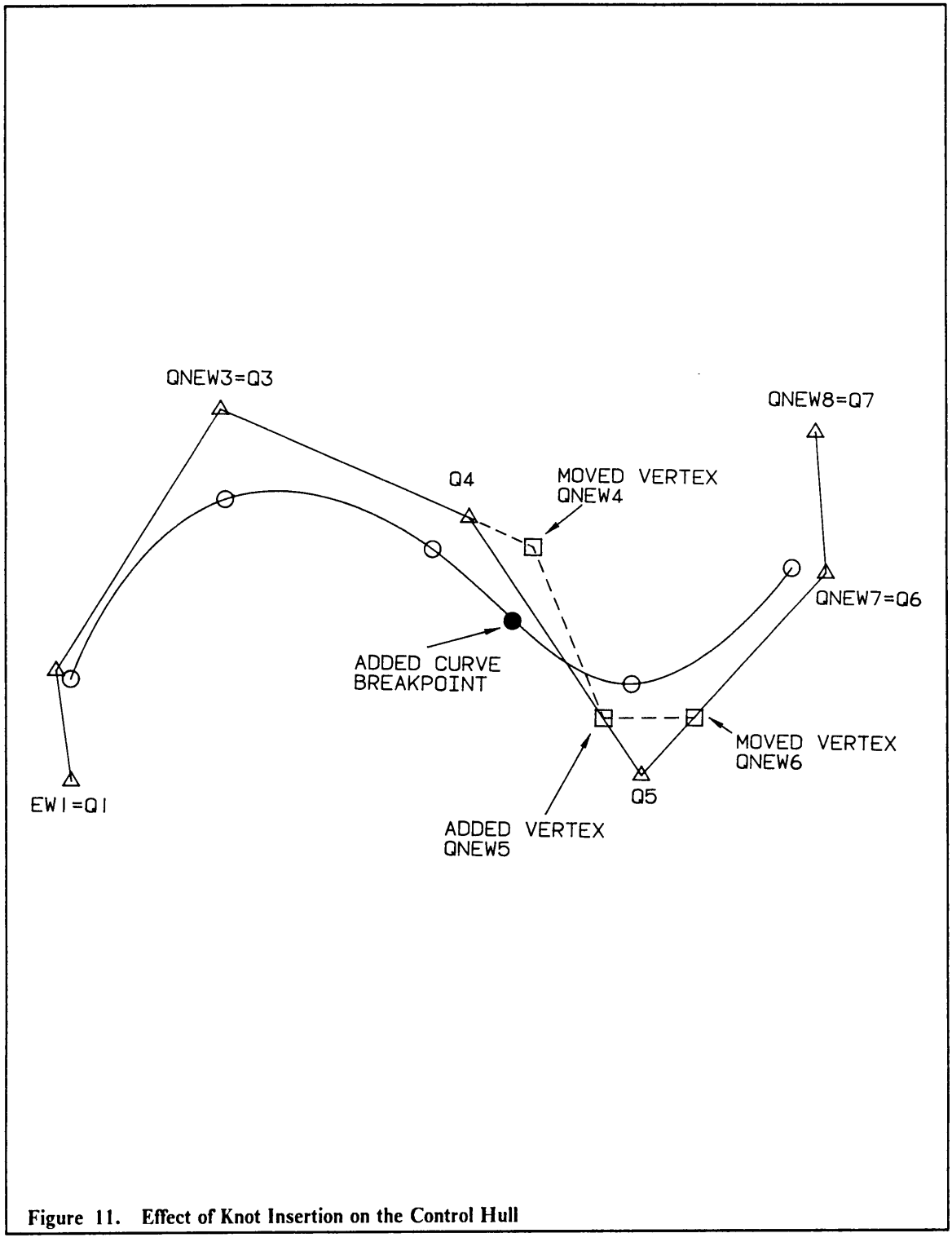


Figure 11. Effect of Knot Insertion on the Control Hull

example shown in Figure 13 on page 38, knots are inserted at u = 3.2, 3.3, 3.4, 3.5, 3.6 for curve A and u = 4.4, 4.5, 4.6, 4.7, 4.8 for curve B.

4. Use the three control vertices corresponding to the three interior inserted knots on each curve to produce the fillet. These control vertices are circled in Figure 13.

5. Use uniform parameter or knot spacing to describe the fillet.

Figure 14 on page 39 shows the completed fillet generated using knot insertion.

When a knot is inserted into a uniform B-spline, the parameter spacing and B-spline curve becomes non-uniform. Position, slope and curvature at the breakpoint of a non-uniform cubic curve segment depends on two components. These components are the three control vertices and five knot values surrounding the breakpoint. The five uniformly spaced knots are inserted to force the curve to be uniform at the desired parameter value.

3.4 Fixed Control Vertex Inversion

A method is presented here for computing fixed control vertex inversion. This provides a means of specifying the shape of a control vertex fillet while maintaining the correct position, slope, and curvature at the end-points of the fillet. Fixed control vertex inversion is similar to the inverse transformation algorithms presented in Sections 2.5 and 2.6 with the exception that several input points are fixed and not modified in the inversion process. These fixed points are actually control vertices which allow the shape

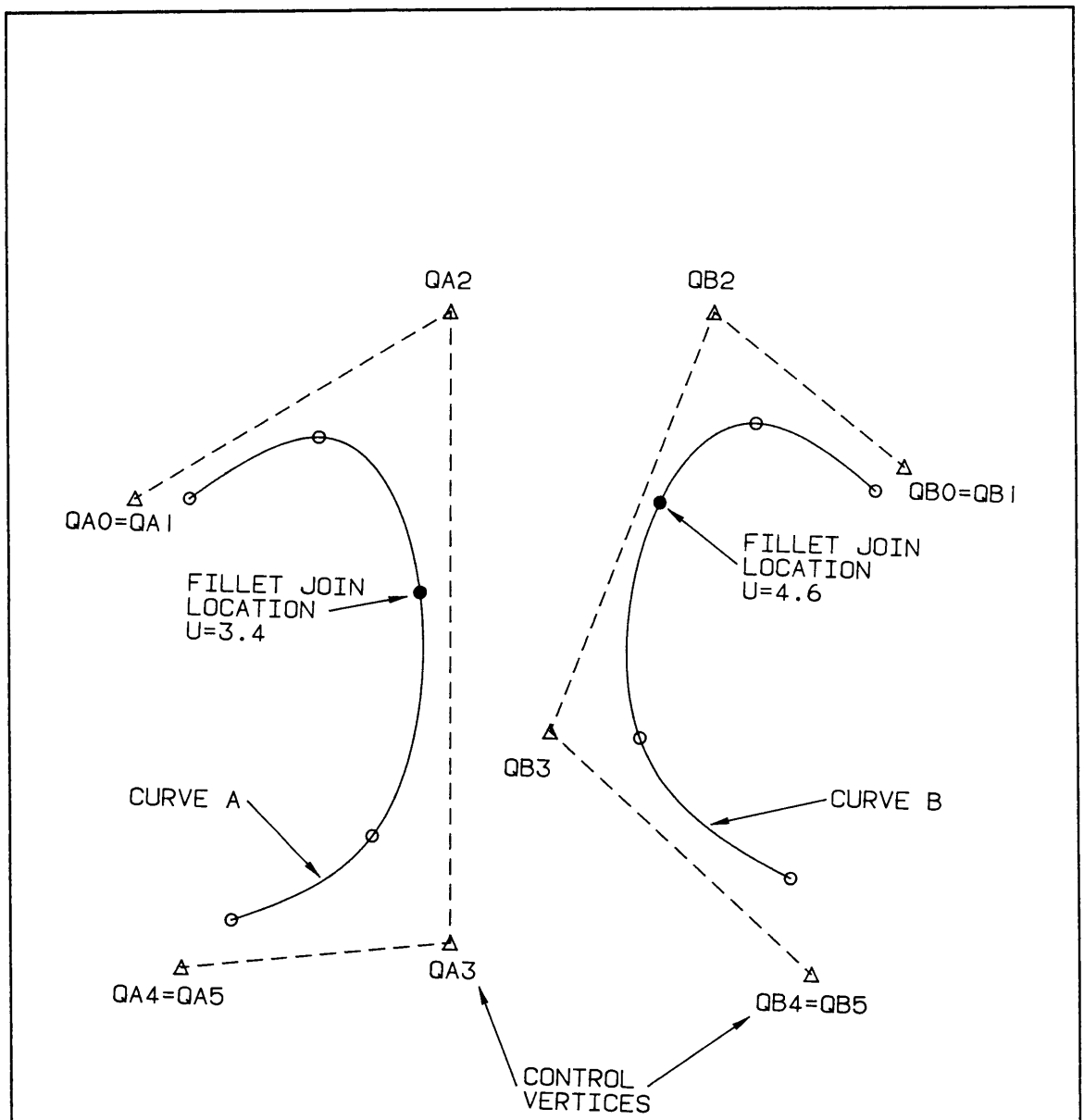


Figure 12. Knot Insertion Curve Filleting - Step 1

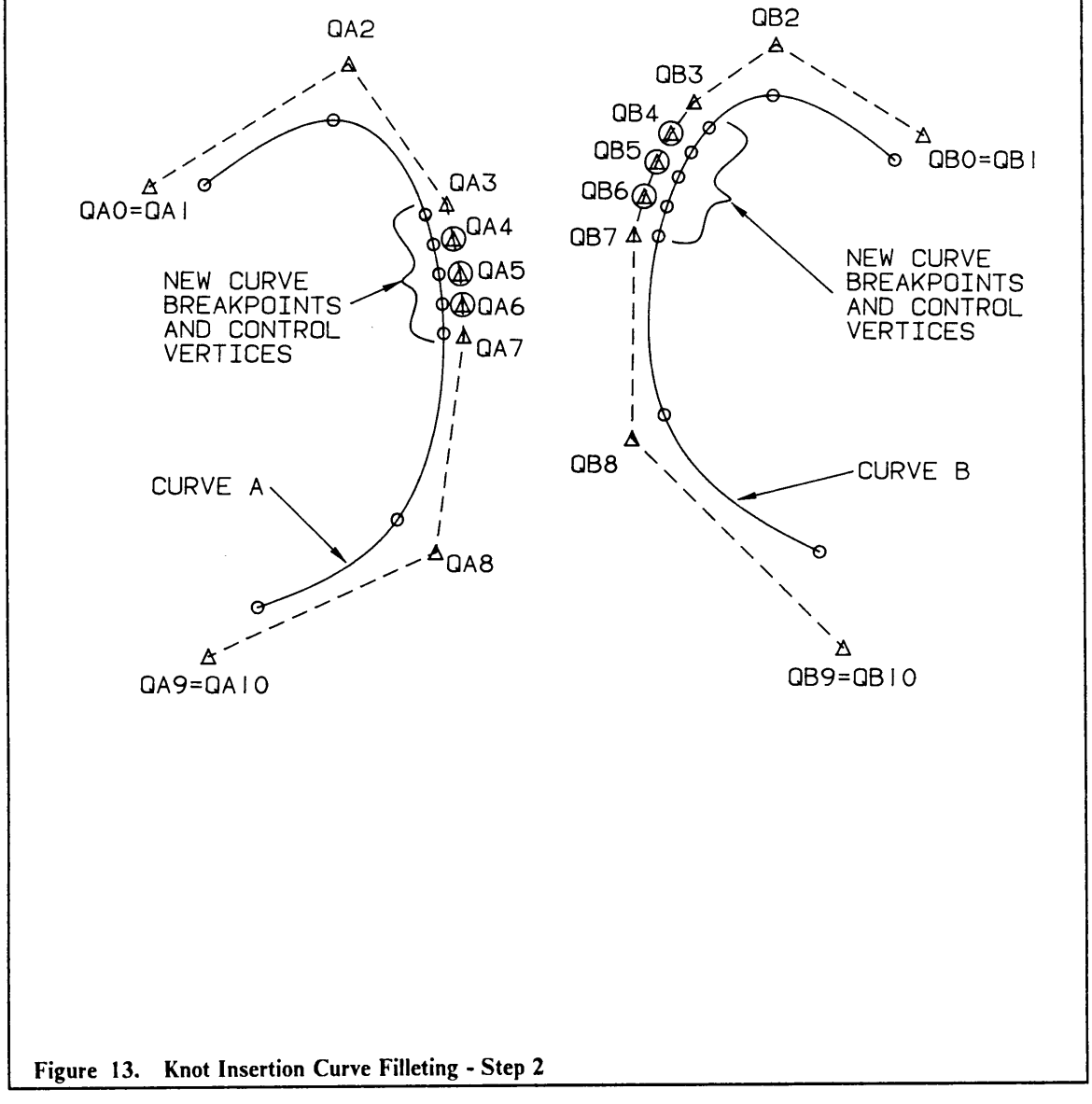


Figure 13. Knot Insertion Curve Filleting - Step 2

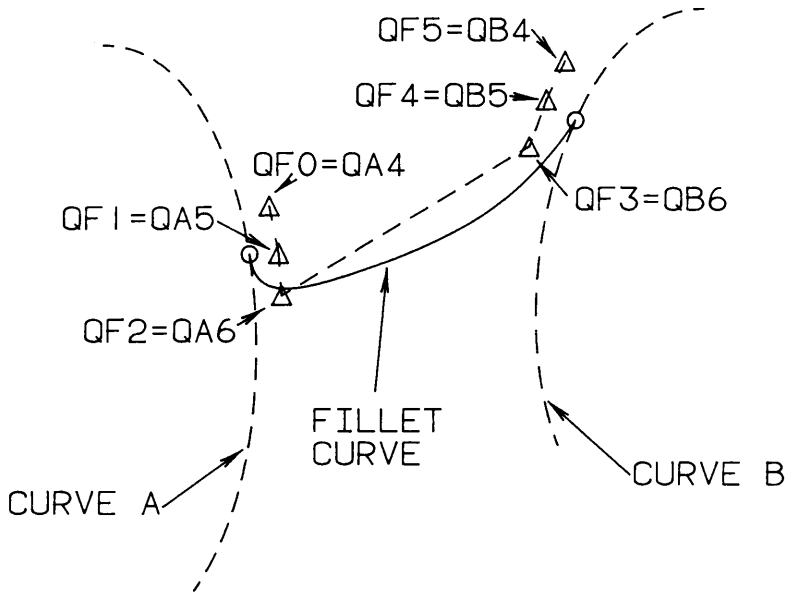


Figure 14. Knot Insertion Curve Filleting - Step 3

of a portion of the curve to be fixed. The difference in the input requirements is shown below.

Normal B-spline Inversion Input:

- Points to be interpolated - $\mathbf{p}_i$ (i = 1, 2, ..., n)

Fixed Control Vertex Inversion Input:

- End condition control vertices - $\mathbf{q}_i$ (i = 0, 1, 2) and (i = n-1, n, n+1)

- Points to be interpolated - $\mathbf{p}_i$ (i = 3, 4, ..., n-3,n-2)

When the fillet is inverted, the six control vertices specifying the end points are not modified. As shown in Figure 15 on page 41, a fillet created using fixed control vertex inversion can interpolate any number of intermediate points and still smoothly blend with the original curves. The computer code designed to compute a fixed control vertex non-uniform B-spline inversion is listed in Appendix C.

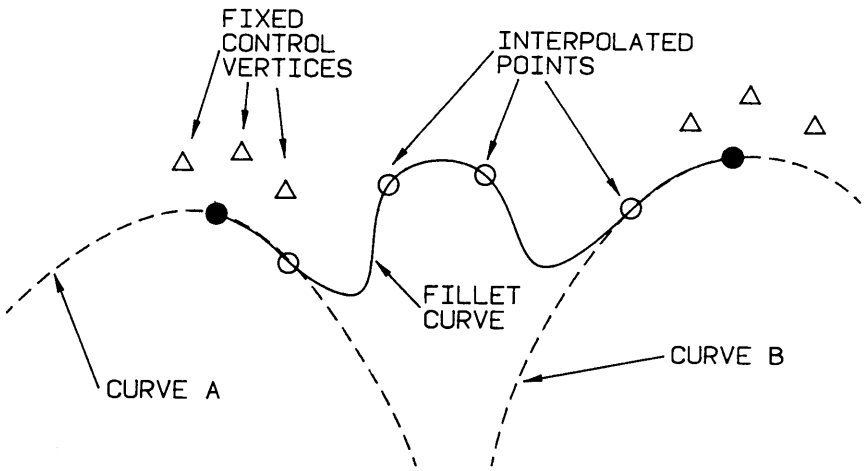


Figure 15. Filleting Using Fixed Control Vertex Inversion

Chapter 4

Surface Filleting

4.1 Overview

Generating a C^2 continuous fillet between two surfaces is slightly more complex than generating curve fillets. In Section 4.2, the knot insertion algorithm for curves is extended to include surfaces. Sections 4.3 and 4.4 discuss two main obstacles to directly adapting curve filleting techniques to surface filleting. These problems are variable knot insertion filleting and parametric "corner" filleting. The solution to the first concern of variable knot spacing also provides an elegant method of relimiting the filleted surfaces.

4.2 Surface Knot Insertion

To calculate surface fillets, the curve knot insertion algorithm must be modified to accommodate surfaces. When a knot is inserted into a B-spline surface, each row (or column) of the matrix containing the control vertices is modified to include the added control vertex. Computer code designed to insert knots into B-spline surfaces is listed in Appendix B. An example of surface knot insertion is presented below.

B-spline Surface Knot Insertion Example

Given: Uniform Bi-cubic B-spline Surface

Knot values:

u = 1, 2, 3, 4, 5, 6, 7

w = 1, 2, 3, 4, 5, 6, 7

Control vertices:

$$w \rightarrow$$

$$u \downarrow \begin{bmatrix} \mathbf{q}(1,1) & \cdots & \mathbf{q}(1,5) \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \mathbf{q}(5,1) & \cdots & \mathbf{q}(5,5) \end{bmatrix}$$

Find: The new control hull matrix if a knot is inserted at u = 3.5

Step 1: Determine the index or interval of the new knot.

$$i = 3$$

Step 2: Calculate the α values for the new knot. (See Section 3.3 which explains curve knot insertion)

$$\alpha_j = 1 \quad \text{if } j \leq 1$$
$$\alpha_j = \frac{t_{new} - t_{j-1}}{t_{j+2} - t_{j-1}} \quad \text{if } 2 \leq j \leq 4$$
$$\alpha_j = 0 \quad \text{if } j \geq 5$$

$$\alpha_1 = 1$$
$$\alpha_2 = \frac{3.5 - 1}{4 - 1} = 0.833$$
$$\alpha_3 = \frac{3.5 - 2}{5 - 2} = 0.500$$
$$\alpha_4 = \frac{3.5 - 3}{6 - 3} = 0.167$$

Step 3: Adjust each column of the control hull matrix to include the new control vertices.

$$\mathbf{q}(1,1)_{new} = (1 - 1)\mathbf{q}(0,1) + (1)\mathbf{q}(1,1) = \mathbf{q}(1,1)$$
$$\mathbf{q}(2,1)_{new} = (1 - 0.833)\mathbf{q}(1,1) + (0.833)\mathbf{q}(2,1)$$
$$\mathbf{q}(3,1)_{new} = (1 - 0.500)\mathbf{q}(2,1) + (0.500)\mathbf{q}(3,1)$$
$$\mathbf{q}(4,1)_{new} = (1 - 0.167)\mathbf{q}(3,1) + (0.833)\mathbf{q}(4,1)$$
$$\mathbf{q}(5,1)_{new} = (1 - 0)\mathbf{q}(4,1) + (0)\mathbf{q}(5,1) = \mathbf{q}(4,1)$$
$$\mathbf{q}(6,1)_{new} = (1 - 0)\mathbf{q}(5,1) + (0)\mathbf{q}(6,1) = \mathbf{q}(5,1)$$

Step 4: Repeat Step 3 for each column of the control hull matrix.

The resulting knot sequence and control hull is listed below.

Knots values:

u = 1, 2, 3, 3.5, 4, 5, 6, 7

w = 1, 2, 3, 4, 5, 6, 7

Control vertices:

$$w \rightarrow$$

$$u \downarrow \begin{bmatrix} \mathbf{q}(1,1)_{new} & \cdots & \mathbf{q}(1,5)_{new} \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \mathbf{q}(6,1)_{new} & \cdots & \mathbf{q}(6,5)_{new} \end{bmatrix}$$

4.3 One-Dimensional Surface Fillets

A fillet will be called one-dimensional if it joins two surfaces along a single parametric direction. Figure 16 on page 46 shows the simplest case of a one-dimensional fillet. This fillet joins the two surfaces along the parameter direction w at a constant u value. Simple one-dimensional surface fillets are constructed by directly extending curve filleting techniques.

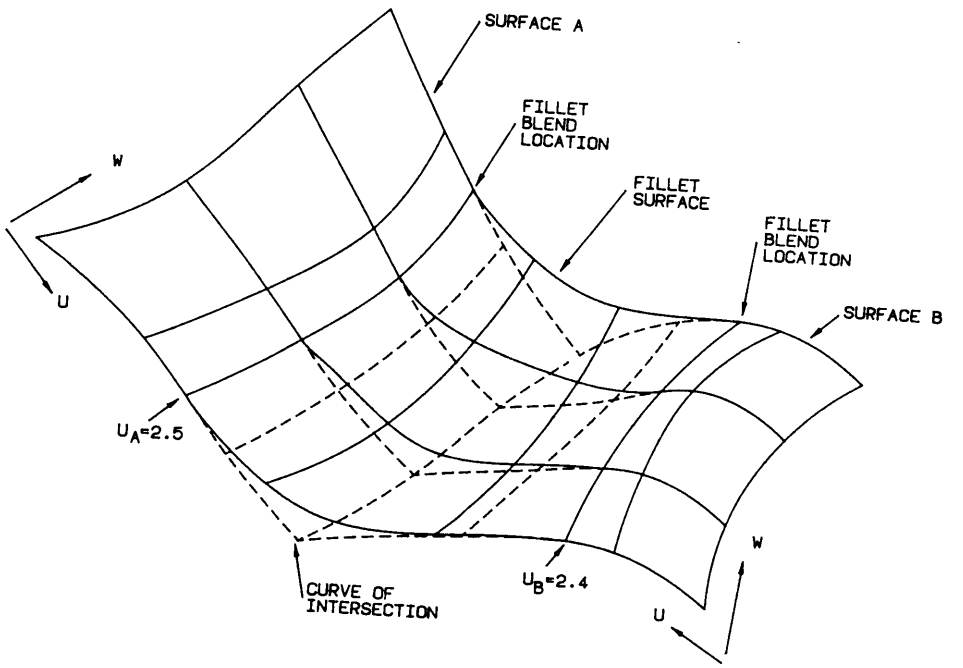


Figure 16. One-dimensional Surface Fillet

Fillets do not necessarily blend with the surfaces at constant parametric values. Although a set of knots can be added at every different parameter value, there is no guarantee that the fillet will smoothly blend with the surface. This is because the position, slope, and curvature of a cubic B-spline surface is dependent on nine control vertices at a breakpoint.

Fixed end-condition inversion, combined with knot insertion, provides a method of creating C^2 continuous surface fillets which blend along variable parameter values. The basic idea is to add knots along a constant parametric value and re-interpolate the original surface until reaching the fillet blend location. The "interior" three control vertices resulting from knot insertion are used as input for the end-condition inversion. The steps required to compute a general one-dimensional fillet are listed below.

1. Determine the parameter values where the fillet should blend with the two surfaces. These values are usually computed from a curve in parametric (u-w) space. These curves are generated by offsetting the surface intersection curves by a certain distance. Figure 17 on page 49 shows two surfaces which require filleting along variable parameter curves. The dotted lines indicate where the fillet should blend with the surfaces.

2. Find a constant u (or w) value which is larger (or smaller) than any u value on the blending curve. The constant parameter value for surface A is u = 3.0 and u = 4.0 for surface B.

3. Add two knots on either side of the constant parameter value on each surface. As shown in Figure 18 on page 50, knots are added at u = 2.8, 2.9, 3.1, 3.2 for surface

A and at u = 3.8, 3.9, 4.1, 4.2 for surface B. Only four knots are inserted because there is already a control vertex present at the constant parameter values.

4. Compute intermediate points on each surface up to the fillet blend location. Figure 18 depicts three points calculated at every w line, although there is no fixed requirement for number of points. Consistently spaced points facilitates parameterization and produces smoother fillets.

5. Perform a fixed control vertex inversion to compute the control hull for the fillet surface. The circled control vertices and the intermediate points in Figure 18 are the inputs for the inversion. The knot spacing in the u direction must be uniform for the first and last five knots, but is not restricted in the center. The w knot spacing must match the knot spacing of the two surfaces. Figure 19 shows the completed fillet surface.

There is a concern that re-interpolation will produce a new surface that is unacceptably different from the original. To check if this was a major problem, a re-interpolated aircraft fuselage was compared to the original. The maximum radius deviation from the original was less than one percent. End-condition inversion and centripetal parameterization techniques are the reasons for this low level of re-interpolation error. The actual amount of re-interpolation error may increase depending on the shape of the surface and the spacing variation of the points to be interpolated.

Fixed end-condition inversion also solves the problem associated with B-spline surface relimiting. When two surfaces are filleted, the unwanted portion of the surface has to be discarded or relimited. Some relimiting techniques utilize a trimming curve to dictate

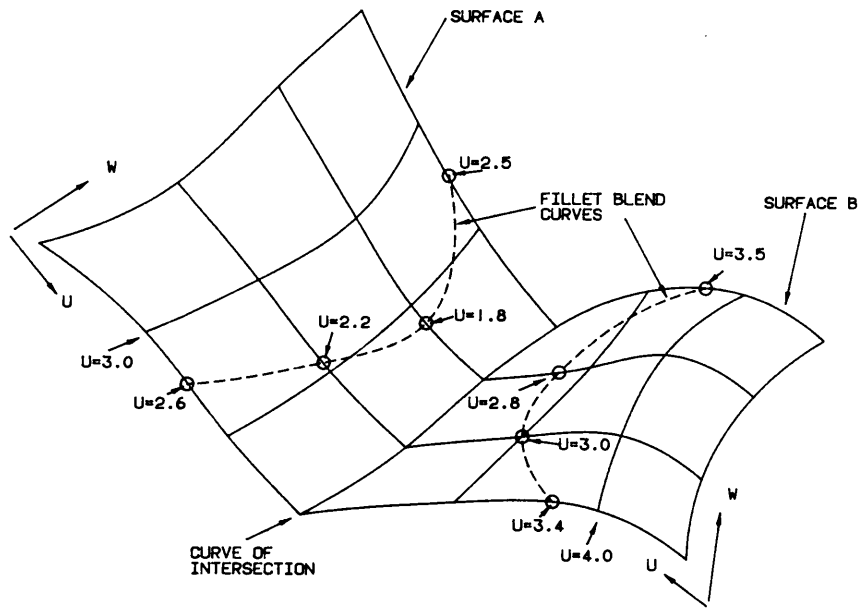


Figure 17. General One-dimensional Filleting - Step 1

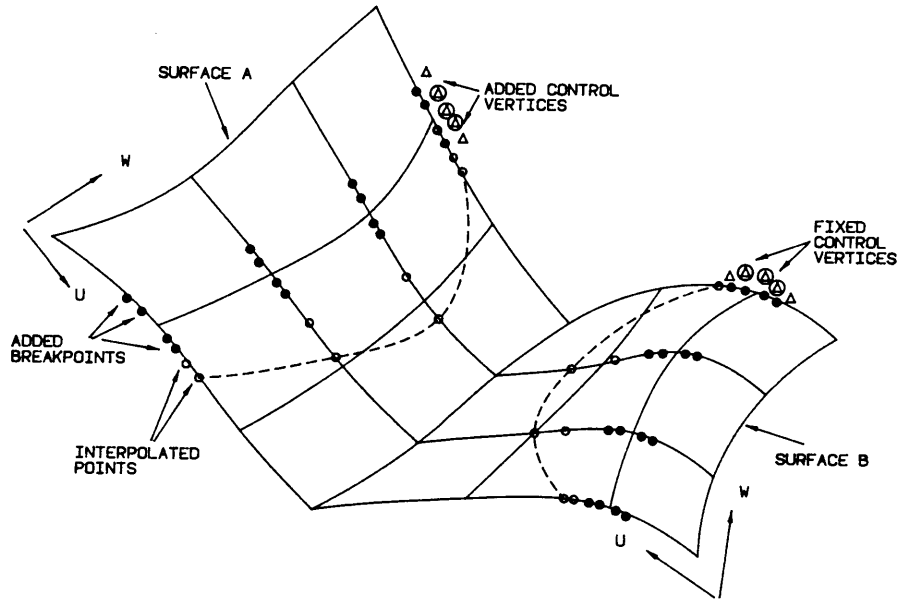


Figure 18. General One-dimensional Filleting - Step 2

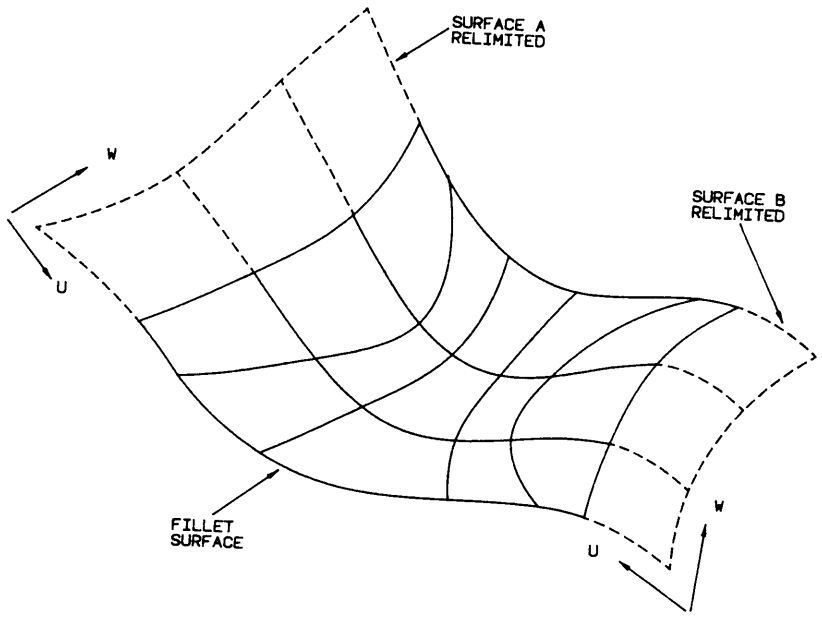


Figure 19. General One-dimensional Filleting - Step 3

where relimited surface ends. A trimming curve is essentially a two-dimensional curve in parametric (u,w) space. When the relimited curve is drawn, only the portion on one "side" of the trimming curve is displayed. Trimming curves require additional computer memory and may limit surface data transfer between CAD/CAM systems.

Relimiting is easily accomplished on surfaces which have been filleted with knot insertion and end-condition inversion techniques. Because the fillet joins the surface along a constant parametric value, the unwanted potion of the surface can be discarded by simply deleting the control vertices corresponding to that portion.

4.4 Parametric Corner Fillets

When a fillet blends with a surface along more than one parametric (u or w) direction, the fillet must smoothly wrap around the corner. This is a difficult problem because the corner patch is five-sided. Non-quadrilateral B-spline surface patches have been investigated by several researchers. Sabin (1986) proves that combinations of triangular and quadrilateral cubic B-spline patches will not produce C^2 continuous surfaces. Therefore, fillets must be composed entirely of one type of B-spline patch. There are several reasons non-quadrilateral patches were not implemented.

- A new inverse transformation algorithm is required
- Knot insertion techniques need to be modified

- Non-quadrilateral patches may not be compatible with other CAD or graphics systems

One solution to the corner filleting problem is to isolate the surface along a single parametric direction and utilize degenerate quadrilateral patches to smoothly blend the corner. The surface is isolated by adding four knots at the same u or w value. The four coincident knots produce a breakpoint at the parametric value where the knots are added. On either side of this breakpoint, the surface is controlled by two independent sets of control vertices. An example of coincident knot insertion for a B-spline curve is listed below.

Coincident Knot Insertion Example

Given: Cubic B-spline curve with four coincident knots

Knot Values - 1, 2, 3, 4, 4.5, 4.5, 4.5, 4.5, 5, 6, 7

Control Vertices - $\mathbf{q}_1$, $\mathbf{q}_2$, $\mathbf{q}_3$, $\mathbf{q}_4$, $\mathbf{q}_5$, $\mathbf{q}_6$, $\mathbf{q}_7$, $\mathbf{q}_8$, $\mathbf{q}_9$

Find: The curve governing control vertices on either side of the breakpoint.

$$\mathbf{p}(4.000) = 0.067\mathbf{q}_2 + 0.489\mathbf{q}_3 + 0.444\mathbf{q}_4$$

$$\mathbf{p}(4.250) = 0.008\mathbf{q}_2 + 0.186\mathbf{q}_3 + 0.681\mathbf{q}_4 + 0.125\mathbf{q}_5$$

$$\mathbf{p}(4.499) = \mathbf{q}_5$$

$$\mathbf{p}(4.500) = \mathbf{q}_6$$

$$\mathbf{p}(4.750) = 0.125\mathbf{q}_6 + 0.681\mathbf{q}_7 + 0.186\mathbf{q}_8 + 0.008\mathbf{q}_9$$

$$\mathbf{p}(5.000) = 0.444\mathbf{q}_7 + 0.489\mathbf{q}_8 + 0.067\mathbf{q}_9$$

where:

$\mathbf{p}(u)$ is a point on the B-spline curve

Notice how the shape of the curve is dictated by the control vertices $\mathbf{q}_2$, $\mathbf{q}_3$, $\mathbf{q}_4$, and $\mathbf{q}_5$ when the parameter value u is less then 4.50. When u is greater or equal to 4.50 then the next set of control vertices ($\mathbf{q}_6$, $\mathbf{q}_7$, $\mathbf{q}_8$, and $\mathbf{q}_9$) govern the curve's shape.

Parametric corner fillets are generated by adding coincident knots and using the filleting techniques developed for one-dimensional fillets. Listed below are the steps required to generate a parametric corner fillet.

1. Find the curves where the fillet should join and blend with both surfaces. These curves are shown in Figure 20 on page 56 and are obtained by offsetting the curve of intersection by some distance. This distance is roughly equivalent to the radius of the fillet.

2. Compute the intermediate points on both surfaces up to the fillet join curves. Also calculate some points on the fillet to provide shape control. Figure 21 on page 57 shows all of the intermediate and fillet shape control points.

3. Invert the intermediate points in the parametric u direction on the fillet. The inversion in the w direction will be performed in step 13.

4. Add three knots to the control hull of the fillet at the parametric value which will line up with the corner on surface A.

5. Copy the control points and knot values for both surfaces into a temporary data structure. This allows control hull modification without altering the shape of the original surfaces.

6. Insert three knots at w = 2.0 into surface A. This will isolate the parametric corner of surface A.

7. Insert two knots on both sides of the parametric line u = 2.0 on Surface A. As shown in Figure 21, the knots are added at u = 1.9, 1.95, 2.05, and 2.10.

8. Store the control vertices from surface A which will guarantee the fillet will smoothly blend with the surface along the u = 2.0 parametric line.

9. Repeat steps 3, 4, 5, and 6 to obtain the control vertices required to blend the fillet along the w = 2.0 parametric line on Surface A.

10. Insert three knots into surface B at u = 5.0. This step is necessary to match the number of control vertices along both sides of the fillet.

11. Add two knots to surface B along both sides of the parameter value w = 3.0.

12. Store the fillet blend controlling vertices from surface B.

13. Invert the fillet in the w parametric direction using the fixed control vertex inversion technique. The interior points required for the inversion are obtained from step 3. The end-condition control vertices are calculated in steps 8 and 12. Figure 22 on page 58 shows the final fillet surface with the relimited portion of Surface B. Surface A is completely replaced by the fillet surface.

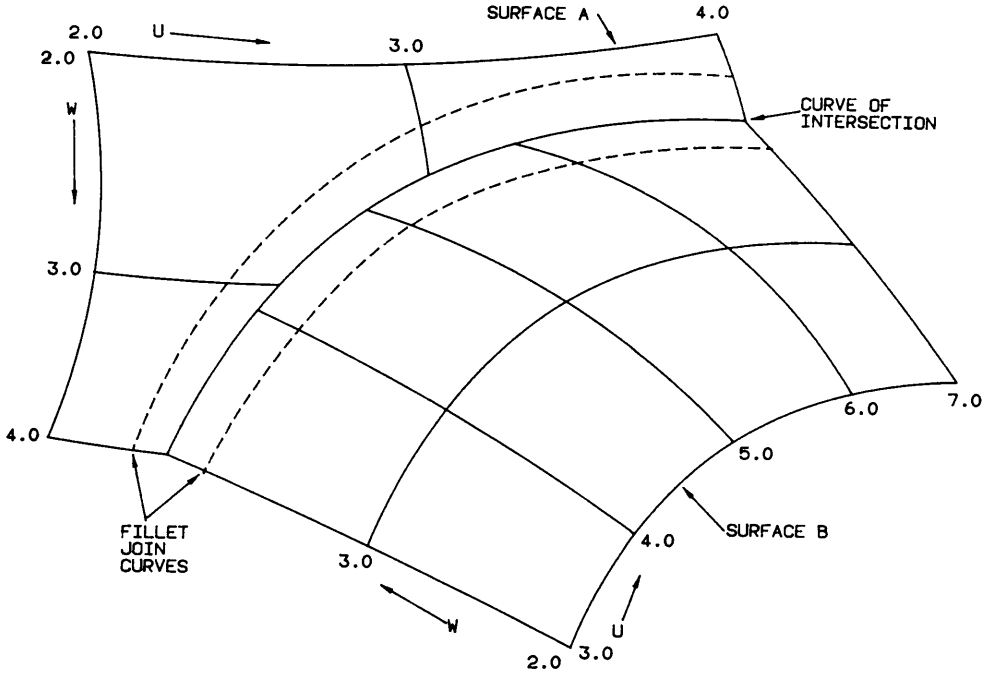


Figure 20. Parametric Corner Filleting - Step 1

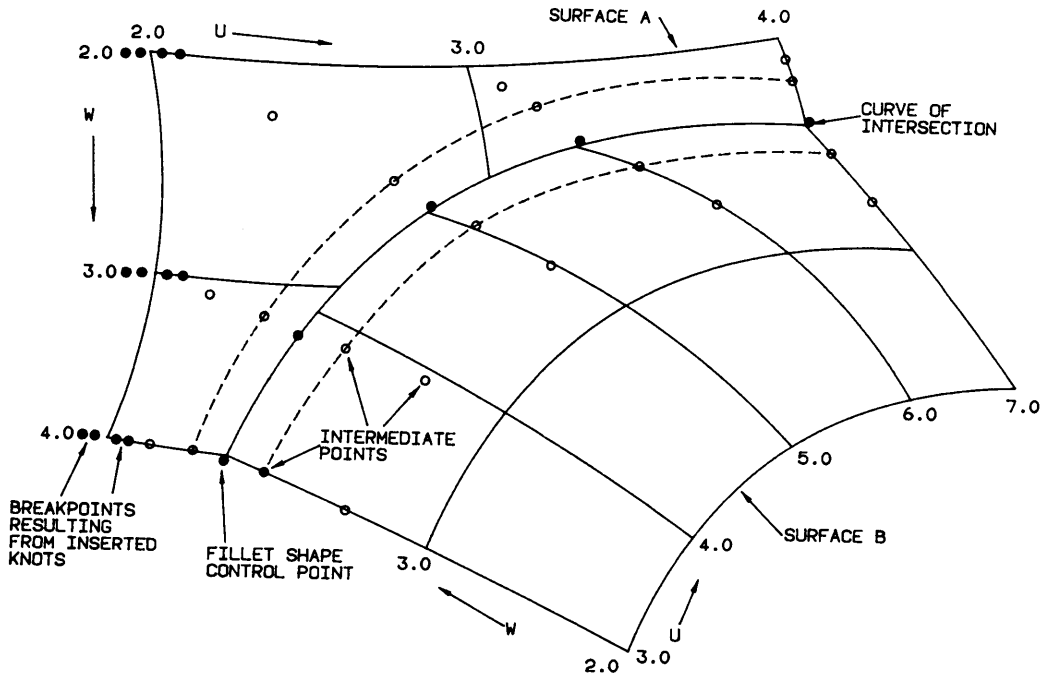


Figure 21. Parametric Corner Filleting - Step 2

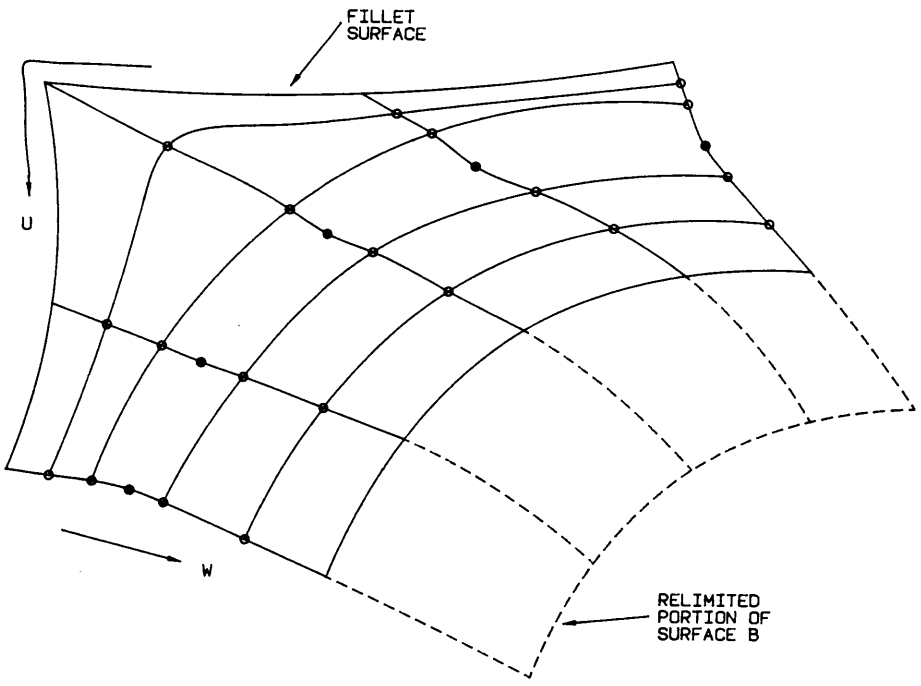


Figure 22. Parametric Corner Filleting - Step 3

Chapter 5

Results and Recommendations

5.1 Overview

This chapter describes the B-spline inversion, rendering and filleting algorithms that are presently integrated with a developmental version of ACSYNT geometry. The first section also discusses the limitations of the current B-spline implementation. The next section presents several examples of filleted and relimited aircraft geometry using shaded images. The last section outlines some recommendations for improvement of the filleting algorithm and describes some of the steps required to complete a production B-spline version of ACSYNT.

5.2 B-spline Surfaces and Filleting in ACSYNT

The following filleting and B-spline surface functions are currently implemented in a developmental B-spline version of ACSYNT.

- Conversion of aircraft components to uniform bi-cubic B-spline surfaces using an inversion algorithm.

- Filleting between the wing and fuselage using non-uniform B-spline surfaces. Set-up files are used to dictate where the fillet should blend with the wing and fuselage.

- Relimiting of filleted components using a set-up file.

- Wire-frame rendering of both uniform and non-uniform B-spline surfaces.

- Shaded images of non-uniform B-spline surfaces.

Several other B-spline utility functions have been coded. These functions are presented below.

- Non-uniform bi-cubic B-spline surface inversion. This algorithm allows the end conditions of the surface to be specified using fixed control points.

- Chord length and centripetal parameterization techniques.

- Generation of a file containing the control hull and knot information for selected aircraft components. This file is in a format suitable for hardware B-spline surface rendering on certain workstations.

The limitations of the current filleting and B-spline implementation are listed below. These concerns must be addressed before a "production" B-spline version of ACSYNT is released.

- The current geometry storage data structure is not designed to handle B-spline control points and knot values.

- Coincident knot values are presently stored as real values. Multiple knots should be flagged in a separate integer array to prevent floating point errors.

- Components can not be grouped together to form C^2 continuous "super" components. For example, the nose, mid-section, and after-body should be inverted together to form a continuous B-spline fuselage.

- Currently, only wing and fuselage fillets can be generated.

- The filleting procedure uses set-up files, instead of intersection data, to dictate where the fillet should blend with the components.

- User control over the shape of the fillet is not available.

- The rendering schemes have not been optimized for speed.

Appendix A through D contains the complete source code listings for all of the filleting and B-spline algorithms described in this thesis. The code is written in Fortran 77 and uses the ANSI graphics standard, PHIGS, to handle all graphics. The code is divided into the following four sections.

- Appendix A: Filleting
- Appendix B: Knot insertion
- Appendix C: Non-uniform B-spline surface inversion
- Appendix D: B-spline surface rendering (wire-frame and shaded image)

5.3 Example Wing/Fuselage Fillets

This section contains two examples of fillets created using non-uniform B-spline surfaces. Both examples were generated using the B-spline version of ACSYNT.

The first example is a small wing blended with a cylinder representing the fuselage. The parametric corner methods for filleting are required to compute the fillet surface. The fillet is composed of one bi-cubic non-uniform B-spline surface with 28 patches in the u direction and 10 patches in the w direction. Two views of the example fillet are shown in Figure 23 on page 64 and Figure 24 on page 65. These shaded images were generated using the hardware B-spline rendering available on the IBM 6090 display device. Appendix E contains the data used the generate the wing, fuselage, and fillet surfaces.

The wing and fuselage parameters are listed in a form compatible with ACSYNT. The control vertices and knot values are listed for the fillet.

The second example is a completely filleted aircraft consisting of a fuselage, two wings, two horizontal tails and a vertical tail. All surfaces are relimited to remove portions of components that are in the interior of the aircraft. Figure 25 on page 66 and Figure 26 on page 67 show two different views of the shaded image.

5.4 Recommendations

This section discusses some potential methods to improve the filleting algorithm. Some recommendations concerning effective B-spline surface integration into ACSYNT are also presented.

- Utilize the results from an intersection procedure to automatically determine where the fillet should blend with the two surfaces.

- Allow the user to interactively specify the shape of fillet cross-sections at several locations. The fixed control vertex inversion technique (Section 3.4) can be used to control the shape of the fillet.

- Generate fillets for all intersecting aircraft components.

- Investigate rational non-uniform B-spline surfaces (NURBS). NURBS provide more shape control and may produce smoother fillets.

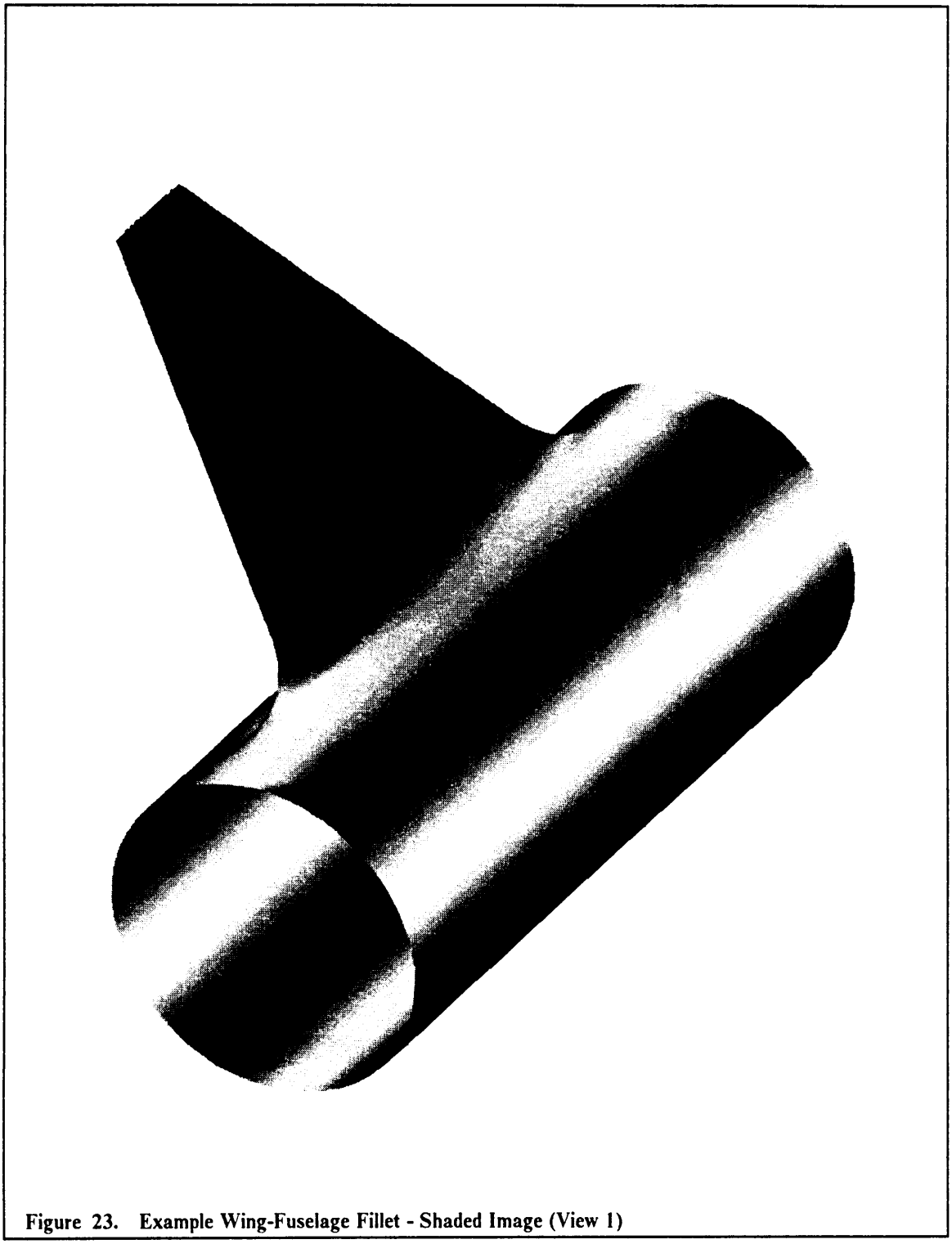


Figure 23. Example Wing-Fuselage Fillet - Shaded Image (View 1)

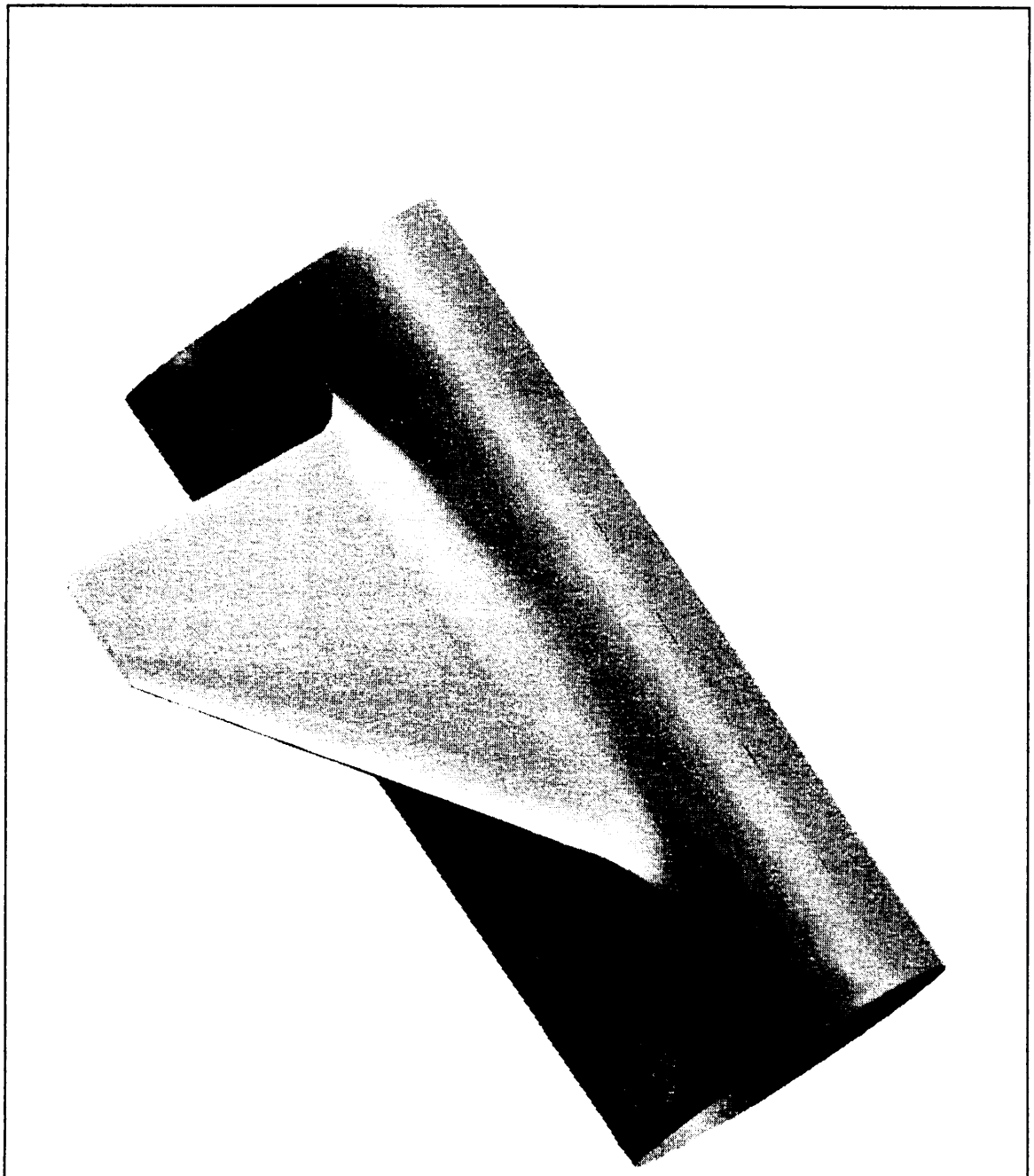


Figure 24. Example Wing-Fuselage Fillet - Shaded Image (View 2)

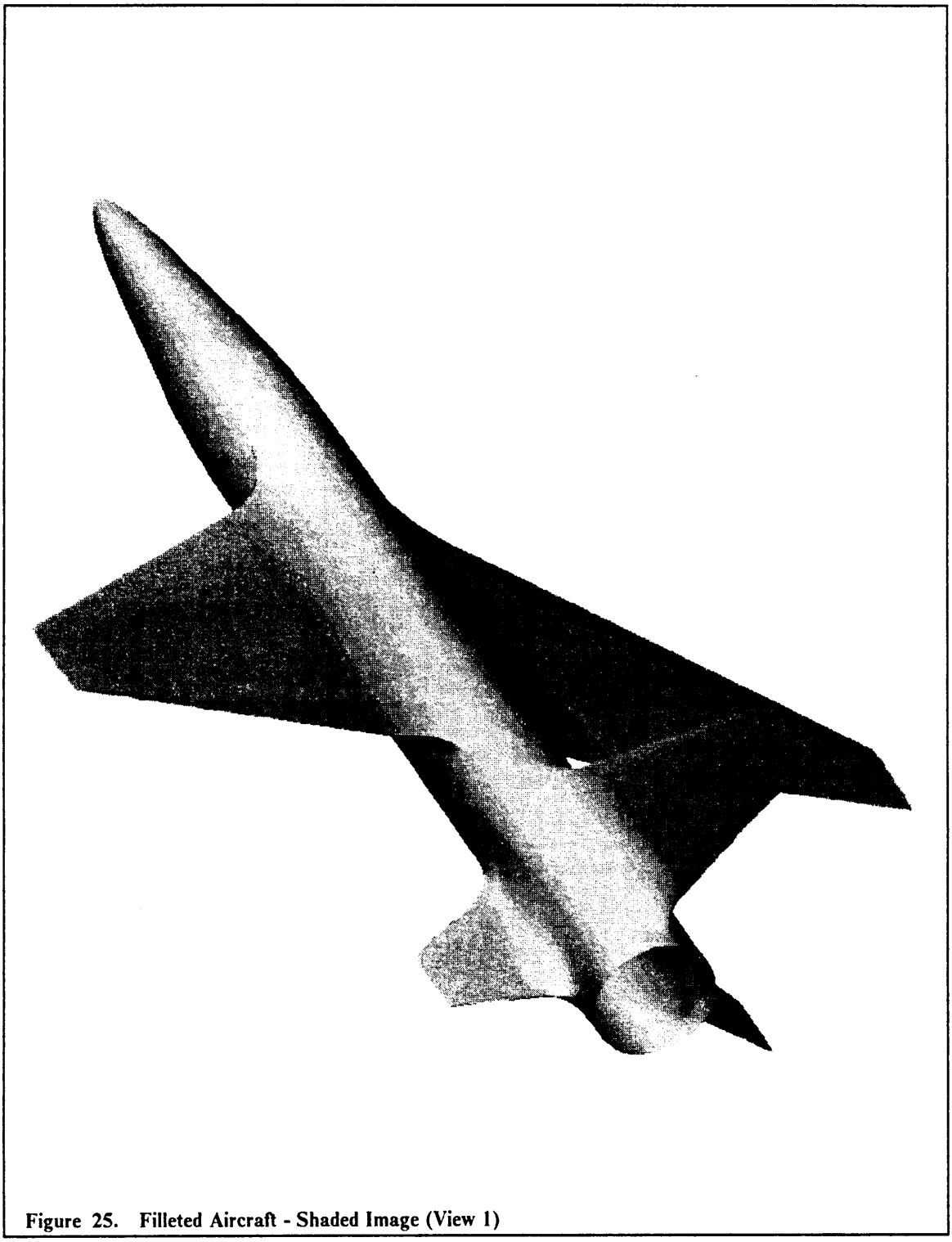


Figure 25. Filleted Aircraft - Shaded Image (View 1)

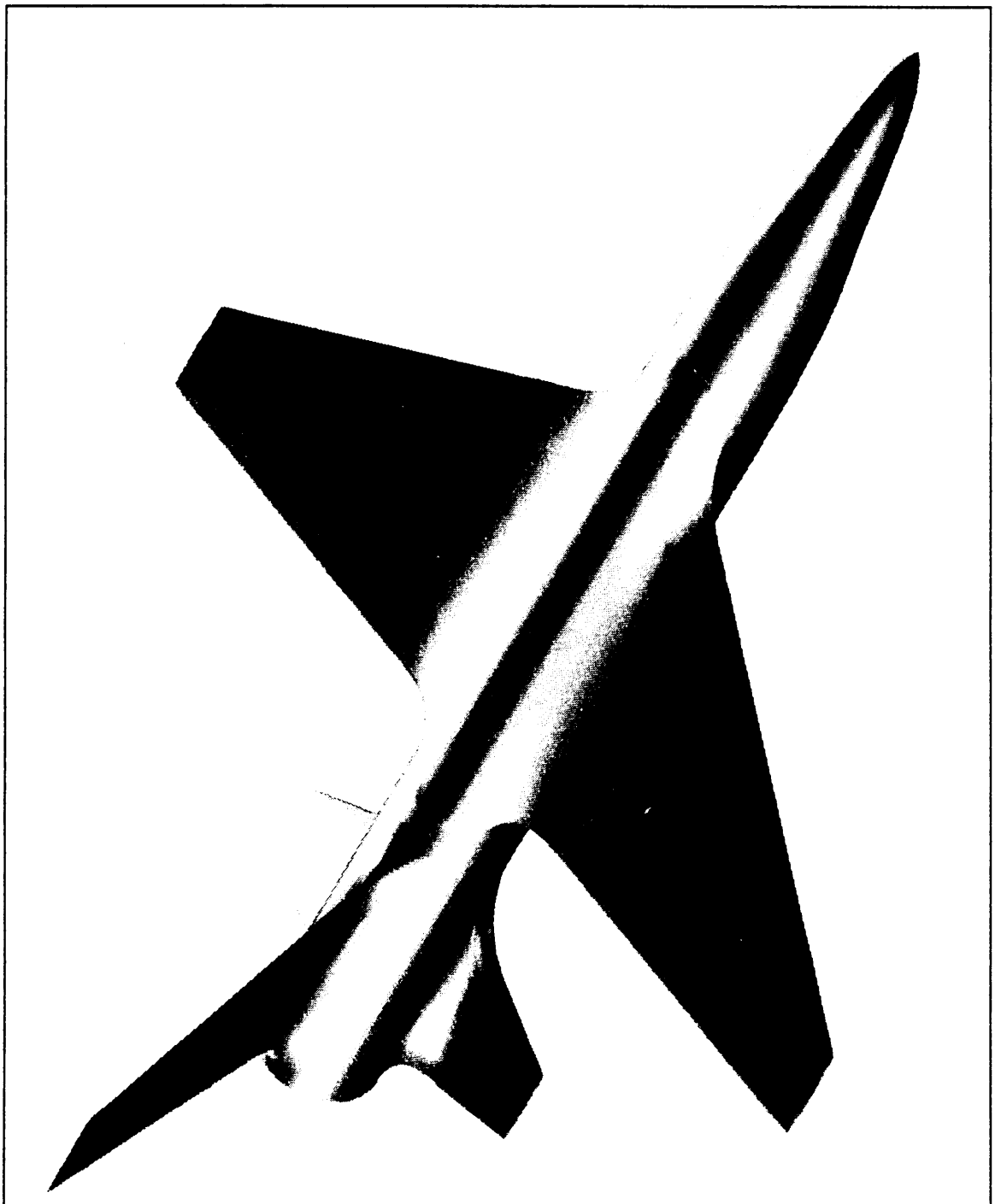


Figure 26. Filleted Aircraft - Shaded Image (View 2)

- Research methods of producing fillets between filleted surfaces.

- Design a new ACSYNT geometry data structure. This data structure should handle B-spline control hulls and knot values without increasing the size of the structure.

- Develop methods to efficiently handle the geometry data for grouped components and partial components. Partial components are created when surfaces are relimited.

Chapter 6

References

1. Boehm, W., "Inserting New Knots into B-spline Curves", *Computer Aided Design*, Vol. 12, 1980.

2. de Boor, C., "On calculating with B-splines", *Journal of Approximation Theory*, Vol. 6, 1972.

3. Farin, G., "Curves and Surfaces for Computer Aided Geometric Design", *Academic Press*, San Diego, 1988.

4. Kubendran, L., Bar-Sever, A., and Harvey, W., "Flow Control in a Wing/Fuselage-type Juncture" *American Institute of Aeronautics and Astronautics*, AIAA-88-0614, 1988.

5. Kutler, P., "A Perspective of Computational Fluid Dynamics", *Tenth International Conference on Numerical Methods in Fluid Dynamics*, 1988.

6. Lu, L., Myklebust, A., and War, S., "Integration of a Helicopter Sizing Code with a Computer-Aided Design System", *Journal of the American Helicopter Society*, Volume 32, 1987.

7. Mortenson, M., "Geometric Modeling", *Wiley*, New York, 1985.

8. Pegna J., and Wolter F., "A Simple Practical Criterion to Guarantee Second Order Smoothness of Blend Surfaces", *Advances in Design Automation - 1989* , ASME, 1989.

9. Rockwood, A., "Introducing Sculptured Surfaces into a Geometric Modeler", *G.M. Conference on Solid Modeling*, 1983.

10. Rockwood, A., "The Displacement Method for Implicit Blending Surfaces in Solid Models", *ACM Transactions on Graphics*, Association for Computing Machinery, 1989.

11. Rubbert, P., "On the Continued Growth of CFD in Airplane Design", *Computational Fluid Dynamics*, Elsevier Science Publishers, 1988.

12. Sabin, M., "Some Negative Results in N-sided Patches", *Computer Aided Design*, Vol. 18, 1986.

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

14. Warren, J., "Blending Algebraic Surfaces", *ACM Transactions on Graphics*, Association for Computing Machinery, 1989.

15. Wong, C., "Intersection of B-spline Surfaces", Thesis - Master of Science in Mechanical Engineering, Virginia Polytechnic Institute and State University, To be submitted, 1990.

16. Yamaguchi, F., "Curves and Surfaces in Computer Aided Geometric Design", *Springer-Verlag*, 1988.

Appendix A

Program FILLET

```
C=====================================================================
C                      M O D U L E   F I L L E T
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C    CONTROLS FILLETING PROCEDURE
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE FILLET(COMP)

      INTEGER COMP
      INTEGER MAXPTS, IAVAIL
      PARAMETER (MAXPTS = 50)

C     LOAD U AND W VALUES AND FLAGS INTO COMMON BLOCK
      CALL GETUWS(COMP, IAVAIL)

C     CHECK IF DATA FOR COMPONENT IS AVAILABLE
      IF (IAVAIL .EQ. 1) THEN
```

```
C       COMPUTE MID-POINTS, INVERT IN U DIRECTION
        CALL MIDPTS(COMP)

C       ADD KNOTS TO FUSELAGE AND PUT CONTROL VERTICES INTO COMP
        CALL AFUSK(COMP)

C       ADD KNOTS TO WING AND PUT CONTROL VERTICES INTO COMP
        CALL AWINGK(COMP)

C       INVERT IN W DIRECTION
        CALL WINVT(COMP)

      ENDIF

      RETURN
      END
C=======================================================================
C                      M O D U L E  G E T U W S
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     IAVAIL   - INTEGER - COMPONENT AVAILABLE FLAG
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS: FILVAL
C     NOUWS    - INTEGER - NUMBER OF U AND W PAIRS
C     IFUW     - INTEGER - FUSELAGE COMPONENT NUMBER
C     IWING    - INTEGER - WING COMPONENT NUMBER
C     NFUSIN   - INTEGER - NUMBER OF POINTS ON FUS
C     NWNGIN   - INTEGER - NUMBER OF POINTS ON WING
C     FUSFLG   - INTEGER - FLAGS CONTROLLING FUS FILLET
C     WNGFLG   - INTEGER - FLAGS CONTROLLING WING FILLET
C     UORW     - INTEGER - FLAGS INDICATING U OR W DIRECTION
C     DUMCMP   - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C     FUWS     - REAL    - U AND W VALUES FOR FUS BLEND
C     WUWS     - REAL    - U AND W VALUES FOR WING BLEND
C     FINT     - REAL    - U AND W INTERMEDIATE VALUES FOR FUS BLEND
C     WINT     - REAL    - U AND W INTERMEDIATE VALUES FOR WING BLEND
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     LOAD FILLETING DATA INTO COMMON BLOCKS
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE GETUWS(COMP, IAVAIL)

      INTEGER COMP, IAVAIL

      INTEGER NOUT, MAXPTS, NOFILL, NODUM, FILLCMP
      PARAMETER (NOUT = 66, MAXPTS = 50)

      INTEGER NOUWS, IFUS, IWING, NFUSIN, NWNGIN, UORW(2, 5), DUMCMP(2)
      INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
      REAL FUWS(2, MAXPTS), WUWS(2, MAXPTS)
      REAL FINT(2, MAXPTS), WINT(2, MAXPTS)

      COMMON /FILVAL/NOUWS, IFUS, IWING, NFUSIN, NWNGIN, FUSFLG, WNGFLG,
     >               UORW, DUMCMP, FUWS, WUWS, FINT, WINT
```

```
C     INITIALIZE AVAIL FLAG
      IAVAIL = 0

C     OPEN AND REWIND FILE
      OPEN(UNIT = NOUT, FILE = 'FILVAL')
      REWIND(NOUT)

C     READ NUMBER OF FILLETS IN FILE
      READ (NOUT, *, END=900) NOFILL

      DO 200 I=1, NOFILL

C       READ FILLET COMP NUMBER AND NUMBER OF UW POINTS
        READ(NOUT,*)FILCMP, NODUM

C       CHECK IF CURRENT COMP IS A FILLET
        IF (FILCMP .EQ. COMP) THEN

C         SET FLAG AND READ UW INFORMATION
          IAVAIL = 1
          NOUWS = NODUM
          READ(NOUT,*)IFUS, IWING, NFUSIN, NWNGIN
          DO 50 J=1,2
           READ(NOUT,*)UORW(J,1),UORW(J,2),UORW(J,3),UORW(J,4),UORW(J,5)
50        CONTINUE
          READ(NOUT,*)DUMCMP(1), DUMCMP(2)

          DO 100 J=1, NOUWS

               READ(NOUT,*)FUSFLG(J), WNGFLG(J), FUWS(1,J), FUWS(2,J),
     >                     WUWS(1,J), WUWS(2,J), FINT(1,J), FINT(2,J),
     >                     WINT(1,J), WINT(2,J)
100       CONTINUE
        ELSE

C         SKIP THIS COMPONENTS FILLET INFORMATION
          DO 150 J=1, NODUM+4
               READ(NOUT,*)
150       CONTINUE
          ENDIF
200   CONTINUE

C     CLOSE FILE
900   CLOSE(NOUT)

      RETURN
      END
C=======================================================================
C                      M O D U L E  M I D P T S
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS: FILVAL
C     NOUWS    - INTEGER - NUMBER OF U AND W PAIRS
C     IFUW     - INTEGER - FUSELAGE COMPONENT NUMBER
C     IWING    - INTEGER - WING COMPONENT NUMBER
C     NFUSIN   - INTEGER - NUMBER OF POINTS ON FUS
C     NWNGIN   - INTEGER - NUMBER OF POINTS ON WING
C     FUSFLG   - INTEGER - FLAGS CONTROLLING FUS FILLET
C     WNGFLG   - INTEGER - FLAGS CONTROLLING WING FILLET
```

```
C     UORW    - INTEGER - FLAGS INDICATING U OR W DIRECTION
C     DUMCMP  - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C     FUWS    - REAL    - U AND W VALUES FOR FUS BLEND
C     WUWS    - REAL    - U AND W VALUES FOR WING BLEND
C     FINT    - REAL    - U AND W INTERMEDIATE VALUES FOR FUS BLEND
C     WINT    - REAL    - U AND W INTERMEDIATE VALUES FOR WING BLEND
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     COMPUTE MID-POINTS ON THE TWO SURFACES TO BE FILLETED
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE MIDPTS(COMP)

      INTEGER COMP

      REAL U, W, P(3)

      INTEGER MAXPTS
      PARAMETER (MAXPTS = 50)

      INTEGER NOUWS, IFUS, IWING, NFUSIN, NWNGIN, UORW(2, 5), DUMCMP(2)
      INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
      REAL FUWS(2, MAXPTS), WUWS(2, MAXPTS)
      REAL FINT(2, MAXPTS), WINT(2, MAXPTS)

      COMMON /FILVAL/NOUWS, IFUS, IWING, NFUSIN, NWNGIN, FUSFLG, WNGFLG,
     >               UORW, DUMCMP, FUWS, WUWS, FINT, WINT


C     LOAD FUSELAGE INTERMEDIATE POINTS INTO COMP DATABASE
      DO 300 I=1, NOUWS

         DO 100 J=1, NFUSIN
C             FIND INTERMEDIATE U AND W VALUES ON FUS
              CALL FINTUW(NFUSIN, J, FUWS(1,I), FINT(1,I), U, W)
C             FIND POINTS ON FUS
              CALL FNDPNT(IFUS, U, W, P)
C             ADD POINTS TO COMPONENT DATABASE
              CALL PTBTSF(I, J, 1, COMP, P, IERR)

100      CONTINUE

         DO 200 J=NWNGIN, 1, -1

C             FIND INTERMEDIATE U AND W VALUES ON WING
              CALL WINTUW(NWNGIN, J, WUWS(1,I), WINT(1,I), U, W)
C             FIND POINTS ON WINGS
              CALL FNDPNT(IWING, U, W, P)
C             GUESS AN INTERSECTION POINT?
              IF (UORW(2,5) .EQ. 2) THEN
                 CALL PTBTSF(I, NWNGIN-J+NFUSIN+2, 1, COMP, P, IERR)
              ELSE
                 CALL PTBTSF(I, NWNGIN-J+NFUSIN+1, 1, COMP, P, IERR)
              ENDIF

200      CONTINUE

C        PROJECT LINES AND GUESS AN INTERSECTION POINT
         IF (UORW(2,5) .EQ. 2) THEN
         CALL CALFIL(I, NFUSIN, NWNGIN, COMP, P)
```

```
         CALL PTBTSF(I, NFUSIN+1, 1, COMP, P, IERR)
         ENDIF

300   CONTINUE

C     ADD NOUWS TO DATABASE
      CALL PTICMP(5, COMP, NOUWS, IERR)

C     ADD NFUSIN+NWNGIN TO DATABASE
      IF (UORW(2,5) .EQ. 2) THEN
         CALL PTICMP(6, COMP, NFUSIN+NWNGIN+1, IERR)
      ELSE
         CALL PTICMP(6, COMP, NFUSIN+NWNGIN, IERR)
      ENDIF

C     COMPUTE U KNOT SPACING AND ADD TO DATABASE
      CALL UKNTSP(COMP, DUMCMP(2))

C     INVERT IN U DIRECTION
      CALL UINVRT(COMP)

C     ADD KNOTS AT CORNERS
      CALL ADDCOR(COMP)

      RETURN
      END
C=======================================================================
C                      M O D U L E  C A L F I L
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     INT      - INTEGER - INTERVAL
C     NFUSIN   - INTEGER - NUMBER OF FUS POINTS
C     NWNGIN   - INTGER  - NUMBER OF WING POINTS
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     P        - REAL    - INTERSECTION POINT
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE CALFIL(INT, NFUSIN, NWNGIN, COMP, P)

      INTEGER INT, NFUSIN, NWNGIN, COMP
      REAL P(3), FUSP(3), WNGP(3)
      REAL BFUSP(3), BWNGP(3), MIDP(3)
      REAL FVEC(3), WVEC(3), TVEC(3), DIST, SCALE
      REAL LOWBD, HIBD, THETA, FHE, TOL

C     SET ERROR TOLERANCE
      PARAMETER(TOL=0.01)

C     GET FUS POINTS FROM DATABASE
      CALL GTBTSF(INT, NFUSIN-1, 1, COMP, BFUSP, IERR)
```

```
      CALL GTBTSF(INT, NFUSIN, 1, COMP, FUSP, IERR)

C     GET WING POINTS FROM DATABASE
      CALL GTBTSF(INT, NFUSIN+2, 1, COMP, WNGP, IERR)
      CALL GTBTSF(INT, NFUSIN+3, 1, COMP, BWNGP, IERR)

C     CALCULATE MIDPOINT ON LINE BETWEEN POINTS ON WING AND FUS
      DO 100 I=1,3
         MIDP(I) = (FUSP(I)+WNGP(I))/2.0
100   CONTINUE

C     CALCULATE DIRECTION VECTORS
      CALL SUB(FUSP, BFUSP, FVEC)
      CALL VNORM(FVEC, VAL)
      CALL SUB(WNGP, BWNGP, WVEC)
      CALL VNORM(WVEC, VAL)
      CALL ADD(FVEC, WVEC, TVEC)

C     CALCULATE ANGLE BETWEEN DIRECTION VECTOR
      CALL CALANG(FVEC, WVEC, THETA)
      CALL PDIST(FUSP, WNGP, DIST)
      SCALE = 3.0
      HIBD = 6.0
      LOWBD = 0.0
      ICNT = 1

150   CONTINUE
      ICNT = ICNT+1
C     CALCULATE INTERSECTION POINT
      DO 200 I=1,3
         P(I) = MIDP(I) + TVEC(I)*DIST*SCALE
200   CONTINUE

      CALL SUB(P, FUSP, FVEC)
      CALL SUB(P, WNGP, WVEC)
      CALL CALANG(FVEC, WVEC, FHE)

      IRFLAG = 0
C     CHECK TOLERANCES
      IF (FHE .LE. THETA-TOL) THEN
         HIBD = SCALE
         SCALE = (LOWBD+SCALE)/2.0
         IRFLAG = 1
      ELSE IF (FHE .GE. THETA+TOL) THEN
         LOWBD = SCALE
         SCALE = (HIBD+SCALE)/2.0
         IRFLAG = 1
      ENDIF
      IF (IRFLAG .EQ. 1 .OR. ICNT .LE. 50) GOTO 150

      DO 300 I=1,3
         P(I) = (MIDP(I)+P(I))/2.0
300   CONTINUE

      RETURN
      END
C=======================================================================
C                      M O D U L E  C A L A N G
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     VEC1    - INTEGER - FIRST VECTOR
C     VEC2    - INTEGER - SECOND VECTOR
C=======================================================================
C OUTPUT PARAMETERS:
C     ANGLE   - REAL    - ANGLE BETWEEN THE TWO VECTORS
```

```
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     CALCULATE THE ANGLE BETWEEN THE TWO VECTORS
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE CALANG(VEC1, VEC2, ANGLE)

      REAL VEC1(3), VEC2(3), ANGLE
      REAL DOTPRO, MAG1, MAG2

C     CALCULATE THE DOT PRODUCT
      CALL DOT(VEC1, VEC2, DOTPRO)

C     CALCULATE THE MAGNITUDE
      CALL VECVAL(VEC1, MAG1)
      CALL VECVAL(VEC2, MAG2)

      TEMP = DOTPRO/(MAG1*MAG2)

C     CALCULATE THE ANGLE
      ANGLE = ACOS(TEMP)

      RETURN
      END

C=====================================================================
C                    M O D U L E   F N D P N T
C=====================================================================
C  PROJECT:  NASA/AMES                 VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP    - INTEGER - COMPONENT NUMBER
C     U       - REAL    - U PARAMETER VALUE
C     W       - REAL    - W PARAMETER VALUE
C=====================================================================
C OUTPUT PARAMETERS:
C     P(3)    - REAL    - POINT ON THE COMPONENT SURFACE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     CALCULATE A POINT ON A COMPONENT SURFACE AND THE U AND
C     W PARAMETER VALUE
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE FNDPNT(COMP, U, W, P)

      INTEGER COMP
      REAL U, W, P(3)
```

```
      INTEGER UINT, WINT
      REAL UREM, WREM, B(3,4,4), PX, PY, PZ

C     FIND THE INTERVAL OF U AND W
      UINT = INT(U)
      WINT = INT(W)

C     FIND THE REMAINING PORTION OF U AND W
      UREM = U - FLOAT(UINT)
      WREM = W - FLOAT(WINT)

C     LOAD CONTROL POINTS
      CALL LOADB(COMP, UINT, WINT, B)

C     CALCULATE THE POINT
      CALL BPSRF(B, UREM, WREM, PX, PY, PZ)

      P(1) = PX
      P(2) = PY
      P(3) = PZ
      RETURN
      END
C=====================================================================
C                      M O D U L E  L O A D B
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     UINT     - INTEGER - U INTERVAL
C     WINT     - INTEGER - W INTERVAL
C=====================================================================
C OUTPUT PARAMETERS:
C     B        - REAL    - 16 CONTROL POINTS FOR PATCH
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     LOAD THE 16 CONTROL POINTS FOR THE UNIFORM B-SPLINE PATCH
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================

      SUBROUTINE LOADB(COMP, UINT, WINT, B)

      INTEGER COMP, UINT, WINT
      REAL B(3,4,4)

C     LOOP THROUGH FOUR TIMES IN U AND W DIRECTION
      DO 100 I=1,4
         DO 50 J=1,4
C           GET THE CONTROL POINT FROM THE DATA-STRUCTURE
            CALL GTBTSF(UINT+I-3, WINT+J-3, 2, COMP, B(1,J,I), IERR)

50       CONTINUE
100   CONTINUE

      RETURN
      END
C=====================================================================
```

```
C                       M O D U L E   F I N T U W
C=======================================================================
C  PROJECT:  NASA/AMES                      VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C      NOPTS    - INTEGER - NUMBER OF INTERMEDIATE POINTS
C      ICNT     - INTEGER - CURRENT POINT
C      FUW      - REAL    - FUSELAGE U AND W VALUE
C      IUW      - REAL    - INTERSECTION U AND W VALUE ON FUS
C=======================================================================
C OUTPUT PARAMETERS:
C      U        - REAL    - INTERMEDIATE U VALUE
C      W        - REAL    - INTERMEDIATE W VALUE
C=======================================================================
C COMMON INPUTS:
C      NONE
C=======================================================================
C COMMON OUTPUTS:
C      NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C      FIND INTERMEDIATE U AND W VALUES ON THE FUSELAGE
C=======================================================================
C METHODS/ALGORITHMS:
C      N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE FINTUW(NOPTS, ICNT, FUW, IUW, U, W)

      INTEGER NOPTS, ICNT
      REAL FUW(2), IUW(2), U, W
      REAL UDIST, WDIST, USTART, WSTART, SCALE, OFFSET
      PARAMETER(OFFSET = 0.20)

C     CALCULATE U AND W DISTANCES IN PARAMETER SPACE
      UDIST = IUW(1) - FUW(1)
      WDIST = IUW(2) - FUW(2)

C     FIND STARTING U AND W VALUES
      USTART = FUW(1) + OFFSET*UDIST
      WSTART = FUW(2) + OFFSET*WDIST

C     DETERMINE HOW FAR ALONG U-W LINE
      SCALE = FLOAT(ICNT-1)/FLOAT(NOPTS-1)

C     CALCULATE U AND W VALUES
      U = USTART + SCALE*UDIST*(1.0-OFFSET)
      W = WSTART + SCALE*WDIST*(1.0-OFFSET)

      RETURN
      END
C=======================================================================
C                       M O D U L E   W I N T U W
C=======================================================================
C  PROJECT:  NASA/AMES                      VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C      NOPTS    - INTEGER - NUMBER OF INTERMEDIATE POINTS
C      ICNT     - INTEGER - CURRENT POINT
C      FUW      - REAL    - WING U AND W VALUE
C      IUW      - REAL    - INTERSECTION U AND W VALUE ON WING
C=======================================================================
C OUTPUT PARAMETERS:
C      U        - REAL    - INTERMEDIATE U VALUE
C      W        - REAL    - INTERMEDIATE W VALUE
C=======================================================================
```

```
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     FIND INTERMEDIATE U AND W VALUES ON THE WING
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================
C     FIND INTERMEDIATE U AND W VALUES ON WING
      SUBROUTINE WINTUW(NOPTS, ICNT, WUW, IUW, U, W)

      INTEGER NOPTS, ICNT
      REAL WUW(2), IUW(2), U, W
      REAL UDIST, WDIST, SCALE

C     CALCULATE U AND W DISTANCES IN PARAMETER SPACE
      UDIST = IUW(1) - WUW(1)
      WDIST = IUW(2) - WUW(2)

C     DETERMINE HOW FAR ALONG U-W LINE
      SCALE = FLOAT(ICNT)/FLOAT(NOPTS)

C     CALCULATE U AND W VALUES
      U = WUW(1) + SCALE*UDIST
      W = WUW(2) + SCALE*WDIST

      RETURN
      END
C=======================================================================
C                     M O D U L E  U K N T S P
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     DCMP     - INTEGER - DUMMY COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     FIND KNOT SPACING IN THE U DIRECTION AND LOAD DATA STRUCTURE
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================
      SUBROUTINE UKNTSP(COMP, DCMP)

      INTEGER COMP, DCMP

      INTEGER ICLOSE, NXSECT, MAXPTS
      PARAMETER(ICLOSE = 0, MAXPTS = 50)
      REAL PTS(3, MAXPTS), UKNTS(MAXPTS)
```

```
C     GET NUMBER OF POINTS IN THE U DIRECTION
      CALL GTICMP(5, COMP, NXSECT, IERR)

C     LOOP THROUGH ALL POINTS
      DO 100 I=1, NXSECT
C        RETRIEVE POINTS ON SURFACE FROM THE DATA STRUCTURE
         CALL GTBTSF(I, 1, 1, COMP, PTS(1, I), IERR)

100   CONTINUE

      CALL CENTRP(NXSECT, PTS, 1.0, ICLOSE, UKNTS)

      DO 200 I=1, NXSECT+4

C        PUT THE KNOT VALUES INTO THE DATA STRUCTURE
         CALL PTKNOT(I, 1, COMP, UKNTS(I), IERR)
         CALL PTKNOT(I, 1, DCMP, UKNTS(I), IERR)

200   CONTINUE

      RETURN
      END
C=======================================================================
C                      M O D U L E  U I N V R T
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP    - INTEGER - COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     INVERT THE FILLET SURFACE IN THE U DIRECTION
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE UINVRT(COMP)

      INTEGER COMP

      INTEGER MAXPTS
      PARAMETER (MAXPTS = 50)
      INTEGER UFLAG(MAXPTS)
      REAL UKNTS(MAXPTS), PTS(3,MAXPTS), OUTPTS(3,MAXPTS)
      REAL UBLN(3, MAXPTS)

C     GET NUMBER OF POINTS AND X-SECTIONS
      CALL GTICMP(5, COMP, NXSECT, IERR)
      CALL GTICMP(6, COMP, NPPXS,  IERR)

C     SET FLAGS FOR CLOSED INVERSION
      DO 100 I = 2, NXSECT+1

         UFLAG(I) = 4
```

```
100   CONTINUE

      UFLAG(1) = 2
      UFLAG(NXSECT+2) = 2

C     LOAD U KNOTS
      DO 200 J=1, NXSECT+4

         CALL GTKNOT(J, 1, COMP, UKNTS(J), IERR)

200   CONTINUE

C     COMPUTE NON-UNIFORM BLENDING FUNCTIONS
      CALL NONBLN(NXSECT+2, UKNTS, UBLN)

C     LOOP THROUGH ALL NPPXS
      DO 500 I=1, NPPXS

C        LOAD POINTS IN U DIRECTION
         DO 300 J=1, NXSECT

              CALL GTBTSF(J, I, 1, COMP, PTS(1, J+1), IERR)

300      CONTINUE

C        NON-UNIFORM INVERSION
         CALL NNVEPT(NXSECT+2, PTS, UFLAG, UBLN, OUTPTS)

C        LOAD CONTROL VERTICES INTO DATABASE
         DO 400 J=1, NXSECT+2

              CALL PTBTSF(J, I+1, 2, COMP, OUTPTS(1, J), IERR)

400      CONTINUE
500   CONTINUE

      RETURN
      END
C=====================================================================
C                      M O D U L E  A D D C O R
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS: FILVAL
C     NOUWS    - INTEGER - NUMBER OF U AND W PAIRS
C     IFUW     - INTEGER - FUSELAGE COMPONENT NUMBER
C     IWING    - INTEGER - WING COMPONENT NUMBER
C     NFUSIN   - INTEGER - NUMBER OF POINTS ON FUS
C     NWNGIN   - INTEGER - NUMBER OF POINTS ON WING
C     FUSFLG   - INTEGER - FLAGS CONTROLLING FUS FILLET
C     WNGFLG   - INTEGER - FLAGS CONTROLLING WING FILLET
C     UORW     - INTEGER - FLAGS INDICATING U OR W DIRECTION
C     DUMCMP   - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C     FUWS     - REAL    - U AND W VALUES FOR FUS BLEND
C     WUWS     - REAL    - U AND W VALUES FOR WING BLEND
C     FINT     - REAL    - U AND W INTERMEDIATE VALUES FOR FUS BLEND
C     WINT     - REAL    - U AND W INTERMEDIATE VALUES FOR WING BLEND
C=====================================================================
```

```
C FUNCTIONAL DESCRIPTION:
C     ADD KNOTS TO THE CORNERS OF THE FILLET SURFACE
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE ADDCOR(COMP)

      INTEGER COMP

      INTEGER MAXPTS
      PARAMETER (MAXPTS = 50)
      REAL KNOT

      INTEGER NOUWS, IFUS, IWING, NFUSIN, NWNGIN, UORW(2, 5), DUMCMP(2)
      INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
      REAL FUWS(2, MAXPTS), WUWS(2, MAXPTS)
      REAL FINT(2, MAXPTS), WINT(2, MAXPTS)

      COMMON /FILVAL/NOUWS, IFUS, IWING, NFUSIN, NWNGIN, FUSFLG, WNGFLG,
     >               UORW, DUMCMP, FUWS, WUWS, FINT, WINT

C     LOOP THROUGH THE POINTS ON THE FILLET SURFACE
      DO 200 I=3, NOUWS+2
C        CHECK IF THE POINT IS A CORNER POINT
         IF (WNGFLG(I-2) .EQ. 1) THEN

C             GET THE KNOT VALUE FROM THE DATA STRUCTURE
              CALL GTKNOT(I, 1, DUMCMP(2), KNOT, IERR)

              DO 100 J=1,3
C                ADD A LINE OF U KNOTS THE THE SURFACE
                 CALL UKTRON(COMP, 1, KNOT)
100           CONTINUE
         ENDIF
200   CONTINUE

      CALL GTICMP(5, COMP, NXSECT, IERR)

C     CHECK IF A INTERSECTION POINT WAS GUESSED
      IF (UORW(2,5) .EQ. 2) THEN
C        SHIFT CONTROL VERTICES TO ALLOW FUS KNOTS
         CALL SHIFTV(COMP, NFUSIN+NWNGIN+1, DUMCMP(1))
      ELSE
C        SHIFT CONTROL VERTICES TO ALLOW FUS KNOTS
         CALL SHIFTV(COMP, NFUSIN+NWNGIN, DUMCMP(1))
      ENDIF

      RETURN
      END
C=====================================================================
C                      M O D U L E  S H I F T V
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP    - INTEGER - COMPONENT IDENTIFIER
C     NOPTS   - INTEGER - NUMBER OF POINTS
C     DCMP    - INTEGER - DUMMY COMPONENT IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
```

```
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     SHIFT THE CONTROL VERTICES IN THE DATA STRUCTURE
C     TO ALLOW FOR THE FUSELAGE BLENDING CONTROL POINTS
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE SHIFTV(COMP, NOPTS, DCMP)

      INTEGER COMP, NOPTS, DCMP
      INTEGER NXSECT, IERR
      REAL P(3)

C     GET THE NUMBER OF POINTS IN THE U DIRECTION
      CALL GTICMP(5, COMP, NXSECT, IERR)

C     SHIFT THE CONTROL POINTS
      DO 300 I=1, NXSECT+2
         DO 100 J=1, NOPTS
              CALL GTBTSF(I, J+1, 2, COMP, P, IERR)
              CALL PTBTSF(I, J+3, 2, DCMP, P, IERR)
100      CONTINUE
         DO 200 J=4, NOPTS+3
              CALL GTBTSF(I, J, 2, DCMP, P, IERR)
              CALL PTBTSF(I, J, 2, COMP, P, IERR)
200      CONTINUE
300   CONTINUE

      RETURN
      END
C=====================================================================
C                        M O D U L E  A F U S K
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C COMMON INPUTS: FILVAL
C     NOUWS    - INTEGER - NUMBER OF U AND W PAIRS
C     IFUW     - INTEGER - FUSELAGE COMPONENT NUMBER
C     IWING    - INTEGER - WING COMPONENT NUMBER
C     NFUSIN   - INTEGER - NUMBER OF POINTS ON FUS
C     NWNGIN   - INTEGER - NUMBER OF POINTS ON WING
C     FUSFLG   - INTEGER - FLAGS CONTROLLING FUS FILLET
C     WNGFLG   - INTEGER - FLAGS CONTROLLING WING FILLET
C     UORW     - INTEGER - FLAGS INDICATING U OR W DIRECTION
C     DUMCMP   - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C     FUWS     - REAL    - U AND W VALUES FOR FUS BLEND
C     WUWS     - REAL    - U AND W VALUES FOR WING BLEND
C     FINT     - REAL    - U AND W INTERMEDIATE VALUES FOR FUS BLEND
C     WINT     - REAL    - U AND W INTERMEDIATE VALUES FOR WING BLEND
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     CONTROLS THE KNOT ADDITION TO THE FUSELAGE AND
```

```
C     THE RETRIEVAL OF THE CONTROL HULL POINTS
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE AFUSK(COMP)

      INTEGER COMP
      INTEGER MAXPTS, IDNUW, DCMP
      PARAMETER (MAXPTS = 50)
      INTEGER KLINE(MAXPTS)

      INTEGER NOUWS, IFUS, IWING, NFUSIN, NWNGIN, UORW(2, 5), DUMCMP(2)
      INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
      REAL FUWS(2, MAXPTS), WUWS(2, MAXPTS)
      REAL FINT(2, MAXPTS), WINT(2, MAXPTS)

      COMMON /FILVAL/NOUWS, IFUS, IWING, NFUSIN, NWNGIN, FUSFLG, WNGFLG,
     >               UORW, DUMCMP, FUWS, WUWS, FINT, WINT

C     INITIALIZE THE COUNTERS AND FLAGS
      CALL PTICMP(5, DUMCMP(2), 1, IERR)
      ISIDE = 1
      IBEG = 1
50    ICNT = 1
      ISTART = IBEG

C     CHECK IF BEGIN COUNTER TOO SMALL
100   IF (IBEG .LE. 0) IBEG = NOUWS-1

C     CHECK IF ITS A CORNER POINT
      IF (FUSFLG(IBEG) .NE. 1) THEN
         IBEG = IBEG-1
         ICNT = ICNT+1
         GOTO 100
      ENDIF

      IFIN = ISTART + 1
      ICNT = ICNT+1
      IEND = 1

C     CHECK IF END COUNTER IS TOO LARGE
200   IF (IFIN .GE. NOUWS) THEN
         IFIN = 1
         IEND = NOUWS
      ENDIF

C     INCREMENT COUNTER AND END COUNTER
      IF (FUSFLG(IFIN) .NE. 1) THEN
         IFIN = IFIN+1
         ICNT = ICNT+1
         GOTO 200
      ENDIF

      IF (IEND .NE. NOUWS) IEND = IFIN
      ICURR = IBEG - 1

      DO 300 I = 1, ICNT
         ICURR = ICURR+1
         IF (ICURR .GE. NOUWS) ICURR = 1
         KLINE(I) = ICURR
300   CONTINUE

C     ADD KNOTS AND GET CONTROL POINTS
      CALL CONADD(COMP, ISIDE, UORW(1,ISIDE), ISTART, IEND, ICNT, KLINE)
```

```
      IBEG = IFIN
      ISIDE = ISIDE + 1

C     CHECK FOR END
      IF (IEND .NE. NOUWS .AND. ISIDE .LE. 5) GOTO 50

C     ADD FIRST AND LAST CONTROL VERTS
      CALL FSTLST(COMP, DUMCMP(2))

      RETURN
      END
C=======================================================================
C                      M O D U L E   F S T L S T
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     DCMP     - INTEGER - DUMMY COMPONENT ID
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     DETERMINE FIRST AND LAST FILLET CONTROL POINTS
C     AND ADD TO THE DATA STRUCTURE
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE FSTLST(COMP, DCMP)

      INTEGER COMP, DCMP, NXSECT, IERR
      REAL P(3)

C     GET THE NUMBER OF POINTS IN THE U DIRECTION
      CALL GTICMP(5, COMP, NXSECT, IERR)

      DO 100 I=1, 3

C        GET THE NEXT TO LAST CONTROL POINT
         CALL GTBTSF(NXSECT, I, 2, COMP, P, IERR)
C        LOAD THE FIRST CONTROL POINT
         CALL PTBTSF(1, I, 2, COMP, P, IERR)

C        GET THE SECOND CONTROL POINT
         CALL GTBTSF(3, I, 2, COMP, P, IERR)
C        LOAD THE LAST CONTROL POINT
         CALL PTBTSF(NXSECT+2, I ,2, COMP, P ,IERR)

100   CONTINUE

      RETURN
      END
C=======================================================================
C                      M O D U L E   C O N A D D
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
```

```
C INPUT PARAMETERS:
C      COMP     - INTEGER - COMPONENT IDENTIFIER
C      ISIDE    - INTEGER - CURRENT SIDE NUMBER
C      INDUW    - INTEGER - U OR W FLAG
C      ISTART   - INTEGER - INDEX TO FUS CONTROL HULL START LOCATION
C      IEND     - INTEGER - INDEX TO FUS CONTROL HULL END LOCATION
C      ICNT     - INTEGER - NUMBER OF POINTS IN KLINE
C      KLINE    - INT ARRAY - INDEXES TO CONTROL HULL LOCATION
C=======================================================================
C OUTPUT PARAMETERS:
C      NONE
C=======================================================================
C COMMON INPUTS:
C      NONE
C=======================================================================
C COMMON OUTPUTS:
C      NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C      CONTROLS ADDING KNOTS AND GETTING CONTROL POINTS
C      FOR ONE SECTION OF FUSELAGE
C=======================================================================
C METHODS/ALGORITHMS:
C      N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================
      SUBROUTINE CONADD(COMP, ISIDE, INDUW, ISTART, IEND, ICNT, KLINE)

      INTEGER COMP, INDUW, ISTART, IEND, ICNT, KLINE(*)

      INTEGER MAXUWS
      PARAMETER (MAXUWS = 30)
      REAL UW(MAXUWS)

C     FIND ADJUSTED U OR W VALUES TO ADD
      CALL ADJUW(COMP, INDUW, ICNT, KLINE, UW)

C     ADD KNOTS TO DUMMY COMPONENT
      CALL ADDKNT(ISIDE, INDUW, ICNT, KLINE, UW)

C     LOAD CONTROL VERTICES INTO COMP
      CALL LOADVT(COMP, INDUW, ISTART, IEND)

      RETURN
      END
C=======================================================================
C                       M O D U L E   A D J U W
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C      COMP     - INTEGER - COMPONENT IDENTIFIER
C      INDUW    - INTEGER - U OR W DIRECTION INDICATOR
C      KLINE    - INT ARRAY - INDEXES TO CONTROL HULL LOCATIONS
C=======================================================================
C OUTPUT PARAMETERS:
C      UW       - REAL ARRAY - ADJUSTED U OR W VALUES
C=======================================================================
C COMMON OUTPUTS:
C      NONE
C=======================================================================
C COMMON INPUTS:  FILVAL
C      NOUWS    - INTEGER - NUMBER OF U AND W PAIRS
C      IFUW     - INTEGER - FUSELAGE COMPONENT NUMBER
C      IWING    - INTEGER - WING COMPONENT NUMBER
C      NFUSIN   - INTEGER - NUMBER OF POINTS ON FUS
```

```
C     NWNGIN  - INTEGER - NUMBER OF POINTS ON WING
C     FUSFLG  - INTEGER - FLAGS CONTROLLING FUS FILLET
C     WNGFLG  - INTEGER - FLAGS CONTROLLING WING FILLET
C     UORW    - INTEGER - FLAGS INDICATING U OR W DIRECTION
C     DUMCMP  - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C     FUWS    - REAL    - U AND W VALUES FOR FUS BLEND
C     WUWS    - REAL    - U AND W VALUES FOR WING BLEND
C     FINT    - REAL    - U AND W INTERMEDIATE VALUES FOR FUS BLEND
C     WINT    - REAL    - U AND W INTERMEDIATE VALUES FOR WING BLEND
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     CONTROLS THE KNOT ADDITION TO THE FUSELAGE AND
C     THE RETRIEVAL OF THE CONTROL HULL POINTS
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE ADJUW(COMP, INDUW, ICNT, KLINE, UW)

      INTEGER COMP, INDUW, ICNT, KLINE(*)
      REAL UW(*), CENT(3), P(3)

      INTEGER MAXPTS
      PARAMETER (MAXPTS = 50)

      INTEGER NOUWS, IFUS, IWING, NFUSIN, NWNGIN, UORW(2, 5), DUMCMP(2)
      INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
      REAL FUWS(2, MAXPTS), WUWS(2, MAXPTS)
      REAL FINT(2, MAXPTS), WINT(2, MAXPTS)

      COMMON /FILVAL/NOUWS, IFUS, IWING, NFUSIN, NWNGIN, FUSFLG, WNGFLG,
     >               UORW, DUMCMP, FUWS, WUWS, FINT, WINT

      DO 100 I=2, ICNT-1
C        CHECK FOR MODIFICATION FLAG
         IF (FUSFLG(KLINE(I)) .EQ. 2) THEN

C             FIND ADJACENT U AND W VALUES FROM FILLET SURFACE
              CALL GCENT(DUMCMP(2), INDUW, KLINE(I)+2, CENT)
              LOWER = FUWS(INDUW, KLINE(I-1))
              UPPER = FUWS(INDUW, KLINE(I+1))
              BOT = (CENT(3) - CENT(1))/(UPPER - LOWER)
C             CALCULATE NEW U OR W VALUE
              UW(I) = LOWER + (CENT(2) - CENT(1))/BOT
         ENDIF
100   CONTINUE

C     FIND CORNER POINTS ON FUSELAGE AND STORE
      ICNT = 1
      DO 200 I=1,NOUWS
         IF (FUSFLG(I) .EQ. 1) THEN
            CALL FNDPNT(IFUS, FUWS(1,I), FUWS(2,I), P)
            CALL PTBTSF(1, ICNT, 1, 77, P, IERR)
            ICNT = ICNT+1
         ENDIF
200   CONTINUE

      RETURN
      END
C=======================================================================
C                      M O D U L E   G C E N T
C=======================================================================
C  PROJECT:  NASA/AMES                 VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
```

```
C     DCMP     - INTEGER - DUMMY COMPONENT IDENTIFIER
C     INDUW    - INTEGER - U OR W DIRECTION INDICATOR
C     POINT    - INTEGER - INDEX FOR U OR W VALUE
C     POINT    - INTEGER - INDEX FOR U OR W VALUE
C=====================================================================
C OUTPUT PARAMETERS:
C     CENT     - REAL ARRAY - ADJACENT U OR W VALUES FROM FILLET
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE GCENT(DCMP, INDUW, POINT, CENT)

      INTEGER DCMP, POINT, INDUW
      REAL CENT(*)

C     GET THE THREE ADJACENT KNOT VALUES
      CALL GTKNOT(POINT-1, 1, DCMP, CENT(1), IERR)
      CALL GTKNOT(POINT, 1, DCMP, CENT(2), IERR)
      CALL GTKNOT(POINT+1, 1, DCMP, CENT(3), IERR)

      RETURN
      END
C=====================================================================
C                    M O D U L E  A D D K N T
C=====================================================================
C  PROJECT:  NASA/AMES                VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     ISIDE    - INTEGER - SIDE COUNTER
C     INDUW    - INTEGER - U OR W DIRECTION INDICATOR
C     ICNT     - INTEGER - COUNTER
C     KLINE    - INTEGER ARRAY - POINTERS TO U OR W VALUES IN DATA STRUCT
C     UW       - REAL ARRAY - ADDITIONAL U OR W VALUES
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     ADD KNOTS TO THE DUMMY FUSELAGE COMPONENT DATA STRUCTURE
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE ADDKNT(ISIDE, INDUW, ICNT, KLINE, UW)

      INTEGER INDUW, ICNT, KLINE(*)
      REAL UW(*), DELT
```

```
      INTEGER MAXPTS
      PARAMETER (MAXPTS = 50)

      INTEGER NOUWS, IFUS, IWING, NFUSIN, NWNGIN, UORW(2, 5), DUMCMP(2)
      INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
      REAL FUWS(2, MAXPTS), WUWS(2, MAXPTS)
      REAL FINT(2, MAXPTS), WINT(2, MAXPTS)

      COMMON /FILVAL/NOUWS, IFUS, IWING, NFUSIN, NWNGIN, FUSFLG, WNGFLG,
     >               UORW, DUMCMP, FUWS, WUWS, FINT, WINT

C     LOAD DUMMY COMPONENT
      CALL LDUMMY(IFUS, DUMCMP(1))

      DO 200 I=1, ICNT
C        CHECK FOR CORNER
         IF (FUSFLG(KLINE(I)) .EQ. 1) THEN

              DO 100 J=1,3
                   CALL SORTKT(INDUW, DUMCMP(1), FUWS(INDUW, KLINE(I)))
100           CONTINUE
C        CHECK FOR MODIFICATION FLAG
         ELSE IF (FUSFLG(KLINE(I)) .EQ. 2) THEN

              CALL SORTKT(INDUW, DUMCMP(1), UW(I))
         ENDIF
200   CONTINUE

C     GET THE DELTA VALUE FROM FILE
      CALL GTDELT(IWING, ISIDE, DELT)

C     CHECK FOR U OR W KNOT ADD
      IF (INDUW .EQ. 1) THEN

         DO 300 J=1,2
C          ADD W KNOTS
           CALL WKTRON(DUMCMP(1), 1, FUWS(2, KLINE(1))-(FLOAT(J)*DELT))
           CALL WKTRON(DUMCMP(1), 1, FUWS(2, KLINE(1))+(FLOAT(J)*DELT))
300      CONTINUE

      ELSE
         DO 400 J=1,2
C          ADD U KNOTS
           CALL UKTRON(DUMCMP(1), 1, FUWS(1, KLINE(1))-(FLOAT(J)*DELT))
           CALL UKTRON(DUMCMP(1), 1, FUWS(1, KLINE(1))+(FLOAT(J)*DELT))
400      CONTINUE

      ENDIF
      RETURN
      END
C=======================================================================
C                      M O D U L E   L D U M M Y
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     INCMP    - INTEGER - INPUT COMPONENT IDENTIFIER
C     OUTCMP   - INTEGER - DUMMY COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
```

```
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     COPY ALL CONTROL POINTS AND KNOT VALUES INTO A DUMMY
C     COMPONENT IDENTIFIER
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================

      SUBROUTINE LDUMMY(INCMP, OUTCMP)

      INTEGER INCMP, OUTCMP
      INTEGER NXSECT, NPPXS, IERR
      REAL P(3)

C     CHECK FOR COMPONENT SYMMETRY
      IF (INCMP .GE. 100) THEN
         IC = INCMP-100
      ELSE
         IC = INCMP
      ENDIF

C     GET NUMBER OF X-SECTIONS
      CALL GTICMP(5, IC, NXSECT, IERR)
      CALL PTICMP(5, OUTCMP, NXSECT, IERR)

C     GET NUMBER OF POINTS PER X-SECTION
      CALL GTICMP(6, IC, NPPXS, IERR)
      CALL PTICMP(6, OUTCMP, NPPXS, IERR)

C     COPY CONTROL POINTS
      DO 200 I=1, NXSECT+2
         DO 100 J=1, NPPXS+2

            CALL GTBTSF(I, J, 2, INCMP, P, IERR)
            CALL PTBTSF(I, J, 2, OUTCMP, P, IERR)

100      CONTINUE
200   CONTINUE

C     LOAD UNIFORM KNOTS INTO DUMMY COMPONENT STRUCTURE
      CALL STRKNT(OUTCMP, NXSECT, NPPXS)

      RETURN
      END
C=====================================================================
C                    M O D U L E   S O R T K T
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     INDUW    - INTEGER - U OR W DIRECTION INDICATOR
C     DUMCMP   - INTEGER - DUMMY COMPONENT IDENTIFIER
C     KVAL     - INTEGER - KNOT VALUE
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
```

```
C FUNCTIONAL DESCRIPTION:
C     ADD AN U OR W KNOT DEPENDING ON VALUE OF INDUW
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE SORTKT(INDUW, DUMCMP, KVAL)

      INTEGER INDUW, DUMCMP
      REAL KVAL

C     CHECK FOR U KNOT
      IF (INDUW .EQ. 1) THEN
C        ADD U KNOT
         CALL UKTRON(DUMCMP, 1, KVAL)

      ELSE
C        ADD W KNOT
         CALL WKTRON(DUMCMP, 1, KVAL)

      ENDIF

      RETURN
      END
C=====================================================================
C                    M O D U L E  G T D E L T
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     IWING   - INTEGER - WING COMPONENT IDENTIFIER
C     ISIDE   - INTEGER - SIDE COUNTER
C=====================================================================
C OUTPUT PARAMETERS:
C     DELT    - REAL    - DELTA VALUE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     READ FILE TO DETERMINE THE DELTA DISTANCE AWAY FROM
C     EACH SIDE TO ADD KNOTS ( IN PARAMETRIC SPACE)
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE GTDELT(IWING, ISIDE, DELT)

      INTEGER IWING, ISIDE, NOUT
      REAL DELT, TDELT(10)

      PARAMETER (NOUT=66)

C     SET DEFAULT DELTA DISTANCE
      DELT = 0.01

C     OPEN THE FILE
      OPEN(UNIT = NOUT, FILE = 'KNOTDELT')
      REWIND(NOUT)
```

```
C     READ THE NUMBER OF COMPONENTS
      READ (NOUT, *, END = 900)NOCOMP

      DO 200 I=1, NOCOMP
C        READ COMPONENT NUMBER AND NUMBER OF DELTS
         READ(NOUT,*)ICOMP, NODELT

C        LOOP THROUGH DELTS
         DO 100 J=1, NODELT
            READ(NOUT, *)TDELT(J)
100      CONTINUE

C        CHECK IF CORRECT COMPONENT
         IF (IWING .EQ. ICOMP) THEN
            DELT = TDELT(ISIDE)
         ENDIF
200   CONTINUE

900   CLOSE(NOUT)

      RETURN
      END
C=======================================================================
C                        M O D U L E   L O A D V T
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP    - INTEGER - COMPONENT IDENTIFIER
C     INDUW   - INTEGER - U OR W INDICATOR
C     ISTART  - INTEGER - STARTING INDEX
C     IEND    - INTEGER - ENDING INDEX
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:  FILVAL
C     NOUWS   - INTEGER - NUMBER OF U AND W PAIRS
C     IFUW    - INTEGER - FUSELAGE COMPONENT NUMBER
C     IWING   - INTEGER - WING COMPONENT NUMBER
C     NFUSIN  - INTEGER - NUMBER OF POINTS ON FUS
C     NWNGIN  - INTEGER - NUMBER OF POINTS ON WING
C     FUSFLG  - INTEGER - FLAGS CONTROLLING FUS FILLET
C     WNGFLG  - INTEGER - FLAGS CONTROLLING WING FILLET
C     UORW    - INTEGER - FLAGS INDICATING U OR W DIRECTION
C     DUMCMP  - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C     FUWS    - REAL    - U AND W VALUES FOR FUS BLEND
C     WUWS    - REAL    - U AND W VALUES FOR WING BLEND
C     FINT    - REAL    - U AND W INTERMEDIATE VALUES FOR FUS BLEND
C     WINT    - REAL    - U AND W INTERMEDIATE VALUES FOR WING BLEND
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     LOAD THE CONTROL VERTICES INTO THE FILLET COMPONENT
C     - THESE POINTS FORCE THE FILLET TO BLEND SMOOTHLY WITH
C     THE FUSELAGE
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE LOADVT(COMP, INDUW, ISTART, IEND)

      INTEGER COMP, INDUW, ISTART, IEND, OPPUW
```

```
      INTEGER MAXPTS
      PARAMETER (MAXPTS = 50)

      INTEGER NOUWS, IFUS, IWING, NFUSIN, NWNGIN, UORW(2, 5), DUMCMP(2)
      INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
      REAL FUWS(2, MAXPTS), WUWS(2, MAXPTS)
      REAL FINT(2, MAXPTS), WINT(2, MAXPTS)

      COMMON /FILVAL/NOUWS, IFUS, IWING, NFUSIN, NWNGIN, FUSFLG, WNGFLG,
     >               UORW, DUMCMP, FUWS, WUWS, FINT, WINT

C     CHECK IF INDEXES ARE ASCENDING OR DESCENDING
      IF (FUWS(INDUW, ISTART) .LT. FUWS(INDUW, IEND)) THEN

C        FIND BEGINNING AND ENDING INDEX
         CALL FINDBG(INDUW, DUMCMP(1), FUWS(INDUW, ISTART), IBEG)
         CALL FINDFN(INDUW, DUMCMP(1), FUWS(INDUW, IEND), IFIN)
C        SET UP/DOWN FLAG
         IUPDN = 0

      ELSE
         CALL FINDFN(INDUW, DUMCMP(1), FUWS(INDUW, ISTART), IBEG)
         CALL FINDBG(INDUW, DUMCMP(1), FUWS(INDUW, IEND), IFIN)
         IUPDN = 1
      ENDIF

C     SET A FLAG OPPOSITE TO INDUW
      IF (INDUW .EQ. 1) THEN
         OPPUW = 2

      ELSE
         OPPUW = 1

      ENDIF

C     SET DIRECTION FLAG
      IF (FUWS(OPPUW, ISTART) .GE. FINT(OPPUW, ISTART)) THEN
         IDIR = 1
      ELSE
         IDIR = 0
      ENDIF


      CALL FINDBG(OPPUW, DUMCMP(1), FUWS(OPPUW, ISTART), ILINE)

      CALL GTICMP(5, DUMCMP(2), ITOTAL, IERR)

      ICNT = ITOTAL

C     CHECK UP/DOWN FLAG
      IF (IUPDN .EQ. 0) THEN

         DO 100 I=IBEG, IFIN
C             GET CONTROL VERTICES AND LOAD INTO FILLET COMPONENT
              CALL GETPUT(INDUW, IDIR, ICNT, I, ILINE, DUMCMP(1), COMP)
100      CONTINUE

      ELSE
         DO 200 I=IBEG, IFIN, -1
C             GET CONTROL VERTICES AND LOAD INTO FILLET COMPONENT
              CALL GETPUT(INDUW, IDIR, ICNT, I, ILINE, DUMCMP(1), COMP)
200      CONTINUE
      ENDIF

C     ADJUST NUMBER OF POINTS IN U DIRECTION
      CALL PTICMP(5, DUMCMP(2), ICNT, IERR)
```

```
      RETURN
      END
C=====================================================================
C                        M O D U L E   F I N D B G
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     INDUW   - INTEGER - U OR W INDICATOR
C     DUMCMP  - INTEGER - DUMMY COMPONENT INDICATOR
C     UW      - REAL    - U OR W VALUE TO SEARCH FOR
C=====================================================================
C OUTPUT PARAMETERS:
C     IBEG    - INTEGER - BEGINNING INDEX
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     FINDS THE BEGINNING INDEX OF THE FUSELAGE SURFACE CORRESPONDING
C     TO THE UW VALUE
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE FINDBG(INDUW, DUMCMP, UW, IBEG)

      INTEGER INDUW, DUMCMP, IBEG
      REAL UW, ERROR, KNOT
      PARAMETER (ERROR = 0.0001)

C     SET ONCE THROUGH FLAG
      IFLAG = 0

C     GET NUMBER OF KNOT VALUES
      CALL GTICMP(INDUW+4, DUMCMP, NPTS, IERR)

      DO 100 I=1, NPTS+4

C        GET KNOT VALUES
         CALL GTKNOT(I, INDUW, DUMCMP, KNOT, IERR)

C        CALCULATE DELTA VALUE
         DELTA = ABS(UW - KNOT)
         IF (DELTA .LE. ERROR) THEN

C           CHECK ONCE THROUGH FLAG
            IF (IFLAG .EQ. 0) THEN
C                 SET BEGINNING INDEX VALUE
                  IBEG = I
                  IFLAG = 1
            ELSE
                  IBEG = I-1
            ENDIF
         ENDIF
100   CONTINUE

      RETURN
      END
C=====================================================================
C                      M O D U L E   F I N D F N
```

```
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     INDUW   - INTEGER - U OR W INDICATOR
C     DUMCMP  - INTEGER - DUMMY COMPONENT INDICATOR
C     UW      - REAL    - U OR W VALUE TO SEARCH FOR
C=====================================================================
C OUTPUT PARAMETERS:
C     IFIN    - INTEGER - FINAL INDEX VALUE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     FIND FINAL INDEX VALUE
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE FINDFN(INDUW, DUMCMP, UW, IFIN)

      INTEGER INDUW, DUMCMP, IFIN
      REAL UW, ERROR, KNOT
      PARAMETER (ERROR = 0.0001)

C     SET ONCE THROUGH FLAG
      IFLAG = 0

C     GET NUMBER OF KNOTS
      CALL GTICMP(INDUW+4, DUMCMP, NPTS, IERR)

      DO 100 I=NPTS+4, 1, -1
C        GET KNOT VALUE
         CALL GTKNOT(I, INDUW, DUMCMP, KNOT, IERR)

C        CALCULATE DELTA VALUE
         DELTA = ABS(UW - KNOT)

C        CHECK DELTA VALUE AGAINST ERROR TOL
         IF (DELTA .LE. ERROR) THEN
            IF (IFLAG .EQ. 0) THEN
C                 SET FINAL INDEX VALUE
                  IFIN = I
                  IFLAG = 1
            ELSE
                  IFIN = I+1
            ENDIF
         ENDIF
100   CONTINUE

      RETURN
      END
C=====================================================================
C                      M O D U L E   G E T P U T
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     INDUW   - INTEGER - U OR W INDICATOR
C     IDIR    - INTEGER - DIRECTION INDICATOR
C     ICNT    - INTEGER - TOTAL COUNTER
```

```
C     ICURR   - INTEGER - CURRENT COUNT
C     ILINE   - INTEGER - U OR W INDEX
C     DUMCMP  - INTEGER - DUMMY COMPONENT NUMBER
C     COMP    - INTEGER - COMPONENT NUMBER
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C    GET CONTROL VERTICES FROM DUMMY COMPONENT AND ADD TO
C    FILLET COMPONENT
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE GETPUT(INDUW, IDIR, ICNT, ICURR, ILINE, DUMCMP, COMP)

      INTEGER INDUW, ICNT, ICURR, ILINE, DUMCMP, COMP, IERR
      REAL P(3)

C     INCREMENT TOTAL COUNTER
      ICNT = ICNT+1

C     CHECK DIRECTION FLAG
      IF (IDIR .EQ. 0) THEN
      DO 100 J = 1,3
C        CHECK U OR W FLAG
         IF (INDUW .EQ. 1) THEN
C             COPY CONTROL VERTEX INTO FILLET DATA STRUCTURE
              CALL GTBTSF(ICURR-1, ILINE-3+J, 2, DUMCMP, P, IERR)
              CALL PTBTSF(ICNT, J, 2, COMP, P, IERR)
         ELSE
C             COPY CONTROL VERTEX INTO FILLET DATA STRUCTURE
              CALL GTBTSF(ILINE-3+J, ICURR-1, 2, DUMCMP, P, IERR)
              CALL PTBTSF(ICNT, J, 2, COMP, P, IERR)
         ENDIF
100   CONTINUE
      ELSE

C     OTHER DIRECTION
      DO 200 J = 3,1,-1
         IF (INDUW .EQ. 1) THEN
              CALL GTBTSF(ICURR-1, ILINE-3+J, 2, DUMCMP, P, IERR)
              CALL PTBTSF(ICNT, 4-J, 2, COMP, P, IERR)
         ELSE
              CALL GTBTSF(ILINE-3+J, ICURR-1, 2, DUMCMP, P, IERR)
              CALL PTBTSF(ICNT, 4-J, 2, COMP, P, IERR)
         ENDIF
200   CONTINUE
      ENDIF

      RETURN
      END
C=======================================================================
C                      M O D U L E  A W I N G K
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
```

```
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     CONTROLS WING BLENDING PROCEDURE
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE AWINGK(COMP)

      INTEGER COMP, MAXWUW
      PARAMETER (MAXWUW = 50)
      REAL UW(MAXWUW)

C     ADJUST KNOT SPACING TO MATCH CENTRIPETAL SPACING
      CALL WADJUW(COMP, UW)

C     LOAD DUMMY COMPONENT AND ADD KNOTS
      CALL WADDKT(UW)

C     LOAD CONTROL VERTICES INTO COMPONENT
      CALL WLODVT(COMP)

      RETURN
      END
C=======================================================================
C                      M O D U L E   W A D J U W
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     UW       - REAL ARRAY - ADJUSTED U OR W VALUES
C=======================================================================
C COMMON INPUTS:  FILVAL
C     NOUWS    - INTEGER - NUMBER OF U AND W PAIRS
C     IFUW     - INTEGER - FUSELAGE COMPONENT NUMBER
C     IWING    - INTEGER - WING COMPONENT NUMBER
C     NFUSIN   - INTEGER - NUMBER OF POINTS ON FUS
C     NWNGIN   - INTEGER - NUMBER OF POINTS ON WING
C     FUSFLG   - INTEGER - FLAGS CONTROLLING FUS FILLET
C     WNGFLG   - INTEGER - FLAGS CONTROLLING WING FILLET
C     UORW     - INTEGER - FLAGS INDICATING U OR W DIRECTION
C     DUMCMP   - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C     FUWS     - REAL    - U AND W VALUES FOR FUS BLEND
C     WUWS     - REAL    - U AND W VALUES FOR WING BLEND
C     FINT     - REAL    - U AND W INTERMEDIATE VALUES FOR FUS BLEND
C     WINT     - REAL    - U AND W INTERMEDIATE VALUES FOR WING BLEND
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     FIND ADJUSTED U OR W VALUES FOR THE WING
```

```
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE WADJUW(COMP, UW)

      INTEGER COMP
      REAL UW(*), CENT(3), LOWER, UPPER, BOT
      INTEGER MAXPTS
      PARAMETER (MAXPTS = 50)

      INTEGER NOUWS, IFUS, IWING, NFUSIN, NWNGIN, UORW(2, 5), DUMCMP(2)
      INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
      REAL FUWS(2, MAXPTS), WUWS(2, MAXPTS)
      REAL FINT(2, MAXPTS), WINT(2, MAXPTS)

      COMMON /FILVAL/NOUWS, IFUS, IWING, NFUSIN, NWNGIN, FUSFLG, WNGFLG,
     >               UORW, DUMCMP, FUWS, WUWS, FINT, WINT
C     SET INDUW FLAG
      INDUW = UORW(2,1)

C     LOOP THROUGH ALL U/W VALUES
      DO 100 I=2, NOUWS-1
C        CHECK FOR ADJUSTMENT FLAG
         IF (WNGFLG(I) .EQ. 2) THEN
C             GET ADJACENT U/W VALUES
              CALL GCENT(DUMCMP(2), INDUW, I+2, CENT)
              LOWER = WUWS(INDUW, I-1)
              UPPER = WUWS(INDUW, I+1)
              BOT = (CENT(3) - CENT(1))/(UPPER - LOWER)

C             CALCULATE ADJUSTED U OR W VALUE
              UW(I) = LOWER + (CENT(2) - CENT(1))/BOT
         ENDIF
100   CONTINUE

C     SET LAST U OR W VALUE
      UW(NOUWS+1) = 2*WUWS(INDUW, NOUWS) - UW(NOUWS-1)

      RETURN
      END
C=====================================================================
C                       M O D U L E  W A D D K T
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     UW       - REAL ARRAY - ADJUSTED U OR W VALUES
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:  FILVAL
C     NOUWS    - INTEGER - NUMBER OF U AND W PAIRS
C     IFUW     - INTEGER - FUSELAGE COMPONENT NUMBER
C     IWING    - INTEGER - WING COMPONENT NUMBER
C     NFUSIN   - INTEGER - NUMBER OF POINTS ON FUS
C     NWNGIN   - INTEGER - NUMBER OF POINTS ON WING
C     FUSFLG   - INTEGER - FLAGS CONTROLLING FUS FILLET
C     WNGFLG   - INTEGER - FLAGS CONTROLLING WING FILLET
C     UORW     - INTEGER - FLAGS INDICATING U OR W DIRECTION
C     DUMCMP   - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C     FUWS     - REAL    - U AND W VALUES FOR FUS BLEND
C     WUWS     - REAL    - U AND W VALUES FOR WING BLEND
C     FINT     - REAL    - U AND W INTERMEDIATE VALUES FOR FUS BLEND
```

```
C     WINT    - REAL    - U AND W INTERMEDIATE VALUES FOR WING BLEND
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     ADD KNOTS TO DUMMY WING COMPONENT
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE WADDKT(UW)

      REAL UW(*), DELT
      INTEGER MAXPTS
      PARAMETER (MAXPTS = 50, DELT = 0.1)

      INTEGER NOUWS, IFUS, IWING, NFUSIN, NWNGIN, UORW(2, 5), DUMCMP(2)
      INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
      REAL FUWS(2, MAXPTS), WUWS(2, MAXPTS)
      REAL FINT(2, MAXPTS), WINT(2, MAXPTS)

      COMMON /FILVAL/NOUWS, IFUS, IWING, NFUSIN, NWNGIN, FUSFLG, WNGFLG,
     >               UORW, DUMCMP, FUWS, WUWS, FINT, WINT

C     SET U OR W INDICATOR
      INDUW = UORW(2, 1)

C     LOAD DUMMY COMPONENT
      CALL LDUMMY(IWING, DUMCMP(1))

C     SET FIRST AND LAST CONTROL VERTICES
      CALL LOOPWG(DUMCMP(1), INDUW)

C     LOOP THROUGH ALL POINTS
      DO 200 I=1, NOUWS
C        CHECK FOR CORNER POINT
         IF (WNGFLG(I) .EQ. 1) THEN
C             ADD THREE KNOT TO CORNER
              DO 100 J=1,3
                   CALL SORTKT(INDUW, DUMCMP(1), WUWS(INDUW, I))
100           CONTINUE

         ELSE IF (WNGFLG(I) .EQ. 2) THEN
C             ADD ONE KNOT TO ADJUSTED U OR W VALUE
              CALL SORTKT(INDUW, DUMCMP(1), UW(I))
         ENDIF
200   CONTINUE

      CALL SORTKT(INDUW, DUMCMP(1), UW(NOUWS+1))

C     CHECK U OR W INDICATOR
      IF (INDUW .EQ. 1) THEN

         DO 300 J=1,2
C          ADD W KNOTS
           CALL WKTRON(DUMCMP(1), 1, WUWS(2, 1)-(FLOAT(J)*DELT))
           CALL WKTRON(DUMCMP(1), 1, WUWS(2, 1)+(FLOAT(J)*DELT))
300      CONTINUE

      ELSE
         DO 400 J=1,2
C          ADD U KNOTS
           CALL UKTRON(DUMCMP(1), 1, WUWS(1, 1)-(FLOAT(J)*DELT))
```

```
            CALL UKTRON(DUMCMP(1), 1, WUWS(1, 1)+(FLOAT(J)*DELT))
400       CONTINUE

      ENDIF

      RETURN
      END
C=====================================================================
C                        M O D U L E   L O O P W G
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     DCMP     - INTEGER - DUMMY COMPONENT IDENTIFIER
C     INDUW    - INTEGER - U OR W INDICATOR
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     ADD KNOT FOR FIRST AND LAST U OR W VALUE
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE LOOPWG(DCMP, INDUW)

      INTEGER DCMP, INDUW, NXSECT, NPPXS
      REAL P(3)

C     GET NUMBER OF POINTS AND X-SECTIONS
      CALL GTICMP(5, DCMP, NXSECT, IERR)
      CALL GTICMP(6, DCMP, NPPXS, IERR)

C     CHECK U OR W INDICATOR
      IF (INDUW .EQ. 2) THEN
C        LOOP THROUGH ALL CONTROL POINTS
         DO 100 I=1, NXSECT+2
C           GET AND PUT NEXT TO LAST CONTROL POINT
            CALL GTBTSF(I, 3, 2, DCMP, P, IERR)
            CALL PTBTSF(I, NPPXS+3, 2, DCMP, P, IERR)

            CALL GTBTSF(I, 4, 2, DCMP, P, IERR)
            CALL PTBTSF(I, NPPXS+4, 2, DCMP, P, IERR)

            CALL PTICMP(6, DCMP, NPPXS+2, IERR)

100      CONTINUE

      ELSE
C        LOOP THROUGH ALL CONTROL POINTS
         DO 200 I=1, NPPXS+2

            CALL GTBTSF(I, 3, 2, DCMP, P, IERR)
            CALL PTBTSF(I, NXSECT+3, 2, DCMP, P, IERR)

            CALL GTBTSF(I, 4, 2, DCMP, P, IERR)
            CALL PTBTSF(I, NXSECT+4, 2, DCMP, P, IERR)
```

```
            CALL PTICMP(5, DCMP, NXSECT+2, IERR)

200      CONTINUE
      ENDIF

      CALL GTICMP(5, DCMP, NXSECT, IERR)
      CALL GTICMP(6, DCMP, NPPXS, IERR)

C     ADD UNIFORM KNOT SPACING TO DUMMY COMPONENT
      CALL STRKNT(DCMP, NXSECT, NPPXS, IERR)

      RETURN
      END
C=======================================================================
C                        M O D U L E  W L O D V T
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:  FILVAL
C     NOUWS    - INTEGER - NUMBER OF U AND W PAIRS
C     IFUW     - INTEGER - FUSELAGE COMPONENT NUMBER
C     IWING    - INTEGER - WING COMPONENT NUMBER
C     NFUSIN   - INTEGER - NUMBER OF POINTS ON FUS
C     NWNGIN   - INTEGER - NUMBER OF POINTS ON WING
C     FUSFLG   - INTEGER - FLAGS CONTROLLING FUS FILLET
C     WNGFLG   - INTEGER - FLAGS CONTROLLING WING FILLET
C     UORW     - INTEGER - FLAGS INDICATING U OR W DIRECTION
C     DUMCMP   - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C     FUWS     - REAL    - U AND W VALUES FOR FUS BLEND
C     WUWS     - REAL    - U AND W VALUES FOR WING BLEND
C     FINT     - REAL    - U AND W INTERMEDIATE VALUES FOR FUS BLEND
C     WINT     - REAL    - U AND W INTERMEDIATE VALUES FOR WING BLEND
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     LOAD CONTROL VERTICES FOR WING BLENDING
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE WLODVT(COMP)

      INTEGER COMP
      INTEGER MAXPTS, OPPUW
      PARAMETER (MAXPTS = 50)

      INTEGER NOUWS, IFUS, IWING, NFUSIN, NWNGIN, UORW(2, 5), DUMCMP(2)
      INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
      REAL FUWS(2, MAXPTS), WUWS(2, MAXPTS)
      REAL FINT(2, MAXPTS), WINT(2, MAXPTS), P(3)

      COMMON /FILVAL/NOUWS, IFUS, IWING, NFUSIN, NWNGIN, FUSFLG, WNGFLG,
     >               UORW, DUMCMP, FUWS, WUWS, FINT, WINT

C     SET U OR W INDICATOR
      INDUW = UORW(2,1)
```

```
C     GET NUMBER OF POINTS
      CALL GTICMP(5, COMP, NOPTS, IERR)

      IF (INDUW .EQ. 1) THEN
         OPPUW = 2

      ELSE
         OPPUW = 1

      ENDIF

C     FIND BEGINNING INDEX
      CALL FINDBG(OPPUW, DUMCMP(1), NUWS(OPPUW, 1), ILINE)
      IF (UORW(2,5) .EQ. 2) THEN
C        SET OFFSET
         IOFF = 4+NFUSIN+NWNGIN
      ELSE
         IOFF = 3+NFUSIN+NWNGIN
      ENDIF

      DO 100 I=1, NOPTS+2
C        GET AND PUT CONTROL VERTICES
         CALL WGTPUT(INDUW, I, ILINE, IOFF,  DUMCMP(1), COMP)

100   CONTINUE

      DO 300 I=1,3
C             GET AND PUT FIRST AND LAST CONTROL VERTICES
              CALL GTBTSF(NOPTS+1, I+IOFF, 2, COMP, P, IERR)
              CALL PTBTSF(2, I+IOFF, 2, COMP, P, IERR)
              CALL GTBTSF(NOPTS, I+IOFF, 2, COMP, P, IERR)
              CALL PTBTSF(1, I+IOFF, 2, COMP, P, IERR)
300   CONTINUE

C     CHECK IF AN INTERSECTING POINT WAS GUESSED
      IF (UORW(2,5) .EQ. 2) THEN
         CALL PTICMP(6, COMP, NFUSIN+NWNGIN+5, IERR)
      ELSE
         CALL PTICMP(6, COMP, NFUSIN+NWNGIN+4, IERR)
      ENDIF

      RETURN
      END
C=======================================================================
C                      M O D U L E  W G T P U T
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     INDUW   - INTEGER - U OR W INDICATOR
C     ICURR   - INTEGER - CURRENT COUNTER
C     ILINE   - INTEGER - OPPOSITE COUNTER
C     IOFF    - INTEGER - INDEX OFFSET
C     DCMP    - INTEGER - DUMMY COMPONENT
C     COMP    - INTEGER - FILLET COMPONENT
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     GET CONTROL VERTICES FROM DUMMY COMPONENT AND ADD
```

```
C     TO FILLET COMPONENT
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================

      SUBROUTINE WGTPUT(INDUW, ICURR, ILINE,IOFF, DCMP, COMP)

      INTEGER INDUW, ICURR, ILINE, IOFF, DCMP, COMP
      REAL P(3)

C     LOOP THROUGH THREE CONTROL VERTICES
      DO 100 I=1,3
         IF (INDUW .EQ. 1) THEN
C             GET AND PUT CONTROL VERTEX
              CALL GTBTSF(ICURR, ILINE-3+I, 2, DCMP, P, IERR)
              CALL PTBTSF(ICURR, I+IOFF, 2, COMP, P, IERR)
         ELSE
C             GET AND PUT CONTROL VERTEX
              CALL GTBTSF(ILINE-3+I, ICURR, 2, DCMP, P, IERR)
              CALL PTBTSF(ICURR, I+IOFF, 2, COMP, P, IERR)
         ENDIF
100   CONTINUE

      RETURN
      END
C=======================================================================
C                      M O D U L E  W I N V T
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     INVERT FILLET IN THE W DIRECTION
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================
      SUBROUTINE WINVT(COMP)

      INTEGER COMP
      INTEGER NPPXS, NXSECT, IERR
      INTEGER MAXPTS
      PARAMETER (MAXPTS = 50)
      INTEGER WFLAGS(MAXPTS)
      REAL WKNTS(MAXPTS), WBLN(3, MAXPTS), PTS(3,MAXPTS)
      REAL OUTPTS(3,MAXPTS), UKNT, OLDKNT, TOL
      REAL P(3)
      PARAMETER (TOL = 0.0001)

C     LOAD KNOTS
      CALL WLODKT(COMP)
```

```
C     GET NUMBER OF POINTS AND CROSS-SECTIONS
      CALL GTICMP(6, COMP, NPPXS, IERR)
      CALL GTICMP(5, COMP, NXSECT, IERR)

C     LOAD W KNOTS
      DO 100 I=1, NPPXS+4
C        GET W KNOTS
         CALL GTKNOT(I, 2, COMP, WKNTS(I), IERR)
         WFLAGS(I) = 4

100   CONTINUE

C     SET CLOSE/OPEN FLAGS
      DO 200 I=1,3
         WFLAGS(I) = 3
         WFLAGS(I+NPPXS-1) = 3
200   CONTINUE

C     CALCULATE NON-UNIFORM BLENDING FUNCTIONS
      CALL NONBLN(NPPXS+2, WKNTS, WBLN)

C     SET INITIAL FLAGS
      UKNT = -999.0
      ICNT = 1

C     LOOP THROUGH ALL X-SECTIONS
      DO 500 I=1, NXSECT+2
         OLDKNT = UKNT
         CALL GTKNOT(I+2, 1, COMP, UKNT, IERR)

C        CHECK IF KNOTS ARE COINCIDENT
         IF (ABS(UKNT - OLDKNT) .GT. TOL .AND. MFLAG .GT. 3) THEN
            MFLAG = 0
            ICNT = ICNT+1
            DO 300 J=1, NPPXS+2
C              GET POINTS TO INVERT
               CALL GTBTSF(I, J, 2, COMP, PTS(1, J), IERR)

300         CONTINUE
         ELSE IF (ABS(UKNT - OLDKNT) .GT. TOL .AND. MFLAG .EQ. 0) THEN
            DO 325 J=1, NPPXS+2
C              GET POINTS TO INVERT
               CALL GTBTSF(I, J, 2, COMP, PTS(1, J), IERR)

325         CONTINUE
         ELSE
            IF (MFLAG .EQ. 0) THEN
               DO 250 J=1, 3
C                 GET CORNER POINT TO INVERT
                  CALL GTBTSF(1, ICNT, 1, 77, PTS(1,J), IERR)
250            CONTINUE
            ENDIF
            MFLAG = MFLAG + 1
            DO 350 J=4, NPPXS+2
C              GET POINTS TO INVERT
               CALL GTBTSF(I, J, 2, COMP, PTS(1, J), IERR)

350         CONTINUE
         ENDIF

C        PERFORM NON-UNIFORM INVERSION
         CALL NNVEPT(NPPXS+2, PTS, WFLAGS, WBLN, OUTPTS)

         DO 400 J=4, NPPXS+2
C           LOAD POINTS INTO DATA STRUCTURE
            CALL PTBTSF(I, J, 2, COMP, OUTPTS(1, J), IERR)
```

```
400      CONTINUE
500   CONTINUE

C     CHECK POINT SPACING
      CALL CHECKP(COMP)

      RETURN
      END
C=======================================================================
C                      M O D U L E  W L O D K T
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP    - INTEGER - COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:  FILVAL
C     NOUWS   - INTEGER - NUMBER OF U AND W PAIRS
C     IFUW    - INTEGER - FUSELAGE COMPONENT NUMBER
C     IWING   - INTEGER - WING COMPONENT NUMBER
C     NFUSIN  - INTEGER - NUMBER OF POINTS ON FUS
C     NWNGIN  - INTEGER - NUMBER OF POINTS ON WING
C     FUSFLG  - INTEGER - FLAGS CONTROLLING FUS FILLET
C     WNGFLG  - INTEGER - FLAGS CONTROLLING WING FILLET
C     UORW    - INTEGER - FLAGS INDICATING U OR W DIRECTION
C     DUMCMP  - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C     FUWS    - REAL    - U AND W VALUES FOR FUS BLEND
C     WUWS    - REAL    - U AND W VALUES FOR WING BLEND
C     FINT    - REAL    - U AND W INTERMEDIATE VALUES FOR FUS BLEND
C     WINT    - REAL    - U AND W INTERMEDIATE VALUES FOR WING BLEND
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     CALCULATE AND LOAD W KNOTS INTO DATA STRUCTURE
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE WLODKT(COMP)

      INTEGER COMP
      INTEGER MAXPTS, NPTS, IERR
      PARAMETER (MAXPTS = 50)

      REAL PTS(3, MAXPTS), KNTS(MAXPTS), FSTKNT, LSTKNT
      REAL PF(3), PL(3), FDELTA
      PARAMETER (FDELTA = 0.5, WDELTA = 0.5)

      INTEGER NOUWS, IFUS, IWING, NFUSIN, NWNGIN, UORW(2, 5), DUMCMP(2)
      INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
      REAL FUWS(2, MAXPTS), WUWS(2, MAXPTS)
      REAL FINT(2, MAXPTS), WINT(2, MAXPTS)

      COMMON /FILVAL/NOUWS, IFUS, IWING, NFUSIN, NWNGIN, FUSFLG, WNGFLG,
     >               UORW, DUMCMP, FUWS, WUWS, FINT, WINT

C     GET NUMBER OF POINTS
      CALL GTICMP(6, COMP, NPTS, IERR)
```

```
C     CALCULATE FIRST AND LAST POINTS FROM U AND W VALUES
      CALL FNDPNT(IFUS, FUWS(1,1), FUWS(2,1), PF)
      CALL FNDPNT(IWING, WUWS(1,1), WUWS(2,1), PL)

      DO 100 I=1, NPTS-4
C        GET REST OF POINTS FROM DATA STRUCTURE
         CALL GTBTSF(1, I, 1, COMP, PTS(1, I+1), IERR)

100   CONTINUE

C     GUESS FIRST AND LAST POINTS
      DO 50 I=1,3
         PTS(I,1) = PF(I)-(PF(I)-PTS(I, 2))*FDELTA
         PTS(I, NPTS-2) = PL(I)-(PL(I)-PTS(I, NPTS-3))*WDELTA
50    CONTINUE

C     USE CENTRIPETAL PARAMETERIZATION TO CALCULATE KNOT VALUES
      CALL CENTRP(NPTS-2, PTS, 1.0, 1, KNTS)

      DO 200 I=1, NPTS+2
         CALL PTKNOT(I+1, 2, COMP, KNTS(I), IERR)
200   CONTINUE

C     CALCULATE FIRST AND LAST KNOTS VALUES
      FSTKNT = KNTS(1) - (KNTS(2) - KNTS(1))
      LSTKNT = KNTS(NPTS+2)*2.0 - KNTS(NPTS+1)

C     PUT KNOT VALUES INTO DATA STRUCTURE
      CALL PTKNOT(1, 2, COMP, FSTKNT, IERR)
      CALL PTKNOT(NPTS+4, 2, COMP, LSTKNT, IERR)

      RETURN
      END
C=======================================================================
C                      M O D U L E  C H E C K P
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP    - INTEGER - COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     CHECK POINT SPACING TO COMPARE WITH GUESSED SPACING
C     FOR DEBUG PURPOSES ONLY
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE CHECKP(COMP)

      INTEGER COMP
      INTEGER NXSECT, NPPXS, IERR, MAXPNT
      PARAMETER(MAXPNT = 50)

      REAL UKNTS(MAXPNT), WKNTS(MAXPNT), U, W
      REAL UBLEND(4), WBLEND(4), P(3)
```

```
      REAL P1(3), P2(3), P3(3)

C     GET NUMBER OF POINTS AND X-SECTIONS
      CALL GTICMP(5, COMP, NXSECT, IERR)
      CALL GTICMP(6, COMP, NPPXS,  IERR)

C     GET ALL U AND W KNOT VALUES
      DO 100 I=1,NXSECT+4
         CALL GTKNOT(I, 1, COMP, UKNTS(I), IERR)
100   CONTINUE

      DO 200 I=1,NPPXS+4
         CALL GTKNOT(I, 2, COMP, WKNTS(I), IERR)
200   CONTINUE

C     GET U AND W KNOTS
      CALL GTKNOT(3, 1, COMP, U, IERR)
      CALL GTKNOT(3, 2, COMP, W, IERR)

C     CALCULATE BLENDING FUNCTIONS
      CALL BLENDT(U, UKNTS, 3, UBLEND)
      CALL BLENDT(W, WKNTS, 3, WBLEND)

C     FIND FIRST POINT
      CALL PNTNON(COMP, 3, 3, UBLEND, WBLEND, P1)

C     FIND SECOND POINT
      CALL GTKNOT(4, 2, COMP, W, IERR)
      CALL BLENDT(W, WKNTS, 4, WBLEND)
      CALL PNTNON(COMP, 3, 4, UBLEND, WBLEND, P2)

C     FIND THIRD POINT
      CALL GTKNOT(5, 2, COMP, W, IERR)
      CALL BLENDT(W, WKNTS, 5, WBLEND)
      CALL PNTNON(COMP, 3, 5, UBLEND, WBLEND, P3)

C     FIND DISTANCE BETWEEN FIRST AND SECOND POINTS
      CALL SUB(P2, P1, P)
      CALL VECVAL(P, DIST)
*     WRITE(*,*)'DISTANCE P1-P2', DIST

C     FIND DISTANCE BETWEEN SECOND AND THIRD POINTS
      CALL SUB(P3, P2, P)
      CALL VECVAL(P, DIST)
*     WRITE(*,*)'DISTANCE P2-P3', DIST

      RETURN
      END
C=======================================================================
C                     M O D U L E  S T R K N T
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP    - INTEGER - COMPONENT IDENTIFIER
C     NXSECT  - INTEGER - NUMBER OF X-SECTIONS
C     NPPXS   - INTEGER - NUMBER OF POINTS PER X-SECTION
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
```

```
C FUNCTIONAL DESCRIPTION:
C     LOAD UNIFORM KNOT VALUES INTO THE COMPONENTS DATA STRUCTURE
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================
      SUBROUTINE STRKNT(COMP, NXSECT, NPPXS)

      INTEGER COMP, NXSECT, NPPXS
      REAL T

C     LOOP THROUGH ALL U KNOTS
      DO 100 I=1,NXSECT+6
         T = FLOAT(I)
C        ADD KNOT TO DATA STRUCTURE
         CALL PTKNOT(I, 1, COMP, T, IERR)
100   CONTINUE

C     LOOP THROUGH ALL W KNOTS
      DO 200 J=1,NPPXS+6
         T = FLOAT(J)
C        ADD KNOT TO DATA STRUCTURE
         CALL PTKNOT(J, 2, COMP, T, IERR)
200   CONTINUE

      RETURN
      END
C=======================================================================
C                      M O D U L E   C E N T R P
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     NPTS     - INTEGER - NUMBER OF INPUT POINTS
C     PTS      - REAL ARRAY - POINTS TO BE INTERPOLATED
C     FIRSTK   - REAL - FIRST KNOT VALUE
C     OCFLAG   - INTEGER - OPEN/CLOSED CURVE FLAG (0-CLOSED, 1-OPEN)
C=======================================================================
C OUTPUT PARAMETERS:
C     KNTS     - REAL ARRAY - KNOTS VALUES
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     COMPUTE KNOT VALUES FOR A SET OF POINTS USING THE CENTRIPETAL
C     PARAMETERIZATION METHOD
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================
      SUBROUTINE CENTRP(NPTS, PTS, FIRSTK, OCFLAG, KNTS)

      INTEGER NPTS, OCFLAG
      REAL PTS(3,*), FIRSTK, KNTS(*)
      REAL DISTI, DISTP, DELKNT

C     INITIALIZE FIRST TWO KNOTS
      KNTS(3) = 0.0
      KNTS(4) = FIRSTK
```

```
      DO 100 I=2, NPTS-1
C        CALCULATE THE DISTANCE BETWEEN THE POINTS
         CALL PDIST(PTS(1,I), PTS(1,I-1), DISTI)
         CALL PDIST(PTS(1,I+1), PTS(1,I), DISTP)

C        FIND THE DISTANCE TO THE NEXT KNOT
         DELKNT = SQRT(DISTP/DISTI)
         DELKNT = DELKNT*(KNTS(I+2)-KNTS(I+1))
         KNTS(I+3) = KNTS(I+2)+DELKNT

100   CONTINUE

C     CHECK OPEN/CLOSED FLAG - 1 = OPEN
      IF (OCFLAG .EQ. 1) THEN
C        FIND FIRST AND LAST TWO KNOT VALUES
         KNTS(2) = KNTS(3)-(KNTS(4)-KNTS(3))
         KNTS(1) = KNTS(2)-(KNTS(4)-KNTS(3))
         KNTS(NPTS+3) = KNTS(NPTS+2)*2.0-KNTS(NPTS+1)
         KNTS(NPTS+4) = KNTS(NPTS+3)*2.0-KNTS(NPTS+2)
      ELSE
C        FIND FIRST AND LAST TWO KNOT VALUES
         KNTS(2) = KNTS(3)-(KNTS(NPTS+2)-KNTS(NPTS+1))
         KNTS(1) = KNTS(2)-(KNTS(NPTS+1)-KNTS(NPTS))
         KNTS(NPTS+3) = KNTS(NPTS+2)+(KNTS(4)-KNTS(3))
         KNTS(NPTS+4) = KNTS(NPTS+3)+(KNTS(5)-KNTS(4))
      ENDIF

      RETURN
      END
C=======================================================================
C                      M O D U L E  C H O R D L
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     NPTS     - INTEGER - NUMBER OF INPUT POINTS
C     PTS      - REAL ARRAY - POINTS TO BE INTERPOLATED
C     FIRSTK   - REAL - FIRST KNOT VALUE
C     OCFLAG   - INTEGER - OPEN/CLOSED CURVE FLAG (0-CLOSED, 1-OPEN)
C=======================================================================
C OUTPUT PARAMETERS:
C     KNTS     - REAL ARRAY - KNOTS VALUES
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     COMPUTE KNOT VALUES FOR A SET OF POINTS USING THE CHORD LENGTH
C     PARAMETERIZATION METHOD
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE CHORDL(NPTS, PTS, FIRSTK, OCFLAG, KNTS)

      INTEGER NPTS, OCFLAG
      REAL PTS(3,*), FIRSTK, KNTS(*)
      REAL DISTI, DISTP, DELKNT

C     INITIALIZE FIRST TWO KNOTS
      KNTS(3) = 0.0
      KNTS(4) = FIRSTK
```

```
      DO 100 I=2, NPTS-1

C        CALCULATE THE DISTANCE BETWEEN THE POINTS
         CALL PDIST(PTS(1,I), PTS(1,I-1), DISTI)
         CALL PDIST(PTS(1,I+1), PTS(1,I), DISTP)

C        FIND THE DISTANCE TO THE NEXT KNOT
         DELKNT = DISTP/DISTI
         DELKNT = DELKNT*(KNTS(I+2)-KNTS(I+1))
         KNTS(I+3) = KNTS(I+2)+DELKNT

100   CONTINUE

C     CHECK OPEN/CLOSED FLAG - 1 = OPEN
      IF (OCFLAG .EQ. 1) THEN
C        FIND FIRST AND LAST TWO KNOT VALUES
         KNTS(2) = KNTS(3)-(KNTS(4)-KNTS(3))
         KNTS(1) = KNTS(2)-(KNTS(4)-KNTS(3))
         KNTS(NPTS+3) = KNTS(NPTS+2)*2.0-KNTS(NPTS+1)
         KNTS(NPTS+4) = KNTS(NPTS+3)*2.0-KNTS(NPTS+2)
      ELSE
C        FIND FIRST AND LAST TWO KNOT VALUES
         KNTS(2) = KNTS(3)-(KNTS(NPTS+2)-KNTS(NPTS+1))
         KNTS(1) = KNTS(2)-(KNTS(NPTS+1)-KNTS(NPTS))
         KNTS(NPTS+3) = KNTS(NPTS+2)+(KNTS(4)-KNTS(3))
         KNTS(NPTS+4) = KNTS(NPTS+3)+(KNTS(5)-KNTS(4))
      ENDIF

      RETURN
      END
C=======================================================================
C                      M O D U L E   V A R P A R
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     NPTS     - INTEGER - NUMBER OF INPUT POINTS
C     PTS      - REAL ARRAY - POINTS TO BE INTERPOLATED
C     FIRSTK   - REAL - FIRST KNOT VALUE
C     OCFLAG   - INTEGER - OPEN/CLOSED CURVE FLAG (0-CLOSED, 1-OPEN)
C     POWER    - REAL - POWER USED TO CALCULATE DELTA
C=======================================================================
C OUTPUT PARAMETERS:
C     KNTS     - REAL ARRAY - KNOTS VALUES
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     COMPUTE KNOT VALUES FOR A SET OF POINTS USING THE VARIABLE
C     POWER PARAMETERIZATION METHOD
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE VARPAR(NPTS, PTS, FIRSTK, OCFLAG, POWER, KNTS)

      INTEGER NPTS, OCFLAG
      REAL PTS(3,*), FIRSTK, KNTS(*), POWER
      REAL DISTI, DISTP, DELKNT

C     INITIALIZE FIRST TWO KNOTS
```

```
      KNTS(3) = 0.0
      KNTS(4) = FIRSTK
      DO 100 I=2, NPTS-1

C        CALCULATE THE DISTANCE BETWEEN THE POINTS
         CALL PDIST(PTS(1,I), PTS(1,I-1), DISTI)
         CALL PDIST(PTS(1,I+1), PTS(1,I), DISTP)

C        FIND THE DISTANCE TO THE NEXT KNOT
         DELKNT = (DISTP/DISTI)**POWER
         DELKNT = DELKNT*(KNTS(I+2)-KNTS(I+1))
         KNTS(I+3) = KNTS(I+2)+DELKNT

100   CONTINUE

C     CHECK OPEN/CLOSED FLAG - 1 = OPEN
      IF (OCFLAG .EQ. 1) THEN
C        FIND FIRST AND LAST TWO KNOT VALUES
         KNTS(2) = KNTS(3)-(KNTS(4)-KNTS(3))
         KNTS(1) = KNTS(2)-(KNTS(4)-KNTS(3))
         KNTS(NPTS+3) = KNTS(NPTS+2)*2.0-KNTS(NPTS+1)
         KNTS(NPTS+4) = KNTS(NPTS+3)*2.0-KNTS(NPTS+2)
      ELSE
C        FIND FIRST AND LAST TWO KNOT VALUES
         KNTS(2) = KNTS(3)-(KNTS(NPTS+2)-KNTS(NPTS+1))
         KNTS(1) = KNTS(2)-(KNTS(NPTS+1)-KNTS(NPTS))
         KNTS(NPTS+3) = KNTS(NPTS+2)+(KNTS(4)-KNTS(3))
         KNTS(NPTS+4) = KNTS(NPTS+3)+(KNTS(5)-KNTS(4))
      ENDIF

      RETURN
      END
C=======================================================================
C                      M O D U L E  P D I S T
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     P1      - REAL ARRAY - FIRST POINT
C     P2      - REAL ARRAY - SECOND POINT
C=======================================================================
C OUTPUT PARAMETERS:
C     DIST    - REAL - DISTANCE BETWEEN POINTS
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     FINDS THE DISTANCE BETWEEN TWO POINTS IN SPACE
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE PDIST(P1, P2, DIST)

      REAL P1(3), P2(3), P(3), DIST

      CALL SUB(P2, P1, P3)
      CALL VECVAL(P3, DIST)

      RETURN
      END
```

Appendix B

Program NEWKNOT

```
C=====================================================================
C                      M O D U L E   U K T R O W
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C      COMP     - INTEGER - COMPONENT IDENTIFIER
C      PATCH    - INTEGER - SUB-SURFACE ON COMPONENT
C      UKNOT    - REAL    - KNOT VALUE TO BE ADDED
C=====================================================================
C OUTPUT PARAMETERS:
C      NONE
C=====================================================================
C COMMON INPUTS:
C      NONE
C=====================================================================
C COMMON OUTPUTS:
C      NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C      ADD ONE ROW OF U KNOTS AND VERTICES TO A B-SPINE SURFACE
C=====================================================================
C METHODS/ALGORITHMS:
C      ADAPTED FROM W. BOHM'S ALGORITHM FOUND IN "INSERTING NEW
C      KNOTS INTO B-SPLINE CURVES", J. APPROX TH. 6, 50-62
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE UKTRON(COMP, PATCH, UKNOT)

      INTEGER COMP, PATCH
      REAL UKNOT

      INTEGER NU, INTER, MAXPTS, NXSECT
      PARAMETER(MAXPTS=150)
      REAL UKVALS(MAXPTS), FACT(MAXPTS)
```

```
C     GET NUMBER OF U CONTROL VERTICES FROM DATABASE
      CALL GTICMP(5, COMP, NXSECT, IERR)
      NU = NXSECT+2

C     GET KNOT VALUES IN U DIRECTION FROM DATABASE
      DO 100 I = 1, NU+2
         CALL GTKNOT(I, 1, COMP, UKVALS(I), ERR)
100   CONTINUE

C     FIND THE INTERVAL OF THE KNOT TO BE ADDED
      CALL FINDIT(NU+2, UKVALS, UKNOT, INTER)

C     FIND THE FACTORS USED TO ADJUST THE VERTICES
      CALL AFACT(NU+2, UKVALS, UKNOT, INTER, FACT)

C     FIND THE NEW CONTROL VERTICES AND ADD TO DATABASE
      CALL NUVERT(COMP, PATCH, NU, FACT)

C     ADJUST KNOT SPACING AND ADD TO DATABASE
      CALL ADJUKT(COMP, PATCH, NU, UKVALS, UKNOT, INTER)

      RETURN
      END
C=======================================================================
C                      M O D U L E   F I N D I T
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     NKNT    - INTEGER - NUMBER OF KNOT VALUES
C     KVALS   - REAL ARRAY - U KNOT VALUES
C     KNOT    - REAL - KNOT VALUE TO BE ADDED
C=======================================================================
C OUTPUT PARAMETERS:
C     INTER   - INTEGER - INTERVAL OF KNOT ON KVALS
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     FIND THE INDEX OR INTERVAL WHERE THE KNOT VALUE IS LOCATED
C     ON THE ARRAY KVALS
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE FINDIT(NKNT, KVALS, KNOT, INTER)

      INTEGER NKNT, INTER
      REAL KVALS(*), KNOT

C     LOOP THROUGH ALL KNOT VALUES AND CHECK INTERVAL
      DO 100 I = NKNT-2, 2, -1

         IF (KNOT .GE. KVALS(I)) THEN
C           SET INTERVAL AND BREAK LOOP
            INTER = I
            GOTO 200
         ENDIF

100   CONTINUE
```

```
C     CONSOLE ERROR
      CALL CONERR('FINDIT','KNOT OUT OF RANGE')

200   CONTINUE

      RETURN
      END
C=======================================================================
C                      M O D U L E   A F A C T
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     NKNT    - INTEGER - NUMBER OF KNOT VALUES
C     KVALS   - REAL ARRAY - KNOT VALUES
C     KNOT    - REAL - KNOT TO BE ADDED
C     INTER   - INTEGER - INTERVAL OF KNOT
C=======================================================================
C OUTPUT PARAMETERS:
C     FACT    - REAL ARRAY - FACTORS USED TO MODIFY CONTROL POINTS
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     FIND THE FACTORS USED TO ADJUST THE CONTROL VERTICES
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE AFACT(NKNT, KVALS, KNOT, INTER, FACT)

      INTEGER NKNT, INTER
      REAL KVALS(*), KNOT, FACT(*)

C     LOOP THROUGH KNOT VALUES
      DO 100 I=1, NKNT-1

C        CHECK INTERVAL AND COMPUTE FACT
         IF (I .GE. INTER+2) THEN
            FACT(I) = 0.0

         ELSE IF (I .GE. INTER-1) THEN
            DENM = KVALS(I+2) - KVALS(I-1)
C           CHECK IF DENOMINATOR IS ZERO
               IF (DENM .EQ. 0.0) THEN
                  FACT(I) = 0.0
               ELSE
                  FACT(I) = (KNOT - KVALS(I-1))/DENM
               ENDIF

         ELSE
            FACT(I) = 1.0
         ENDIF
100   CONTINUE

      RETURN
      END
C=======================================================================
C                      M O D U L E   N U V E R T
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
```

```
C=====================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     PATCH    - INTEGER - SUB-COMPONENT IDENTIFIER
C     FACT     - REAL ARRAY - FACTORS USED TO ADJUST CONTROL VERTICES
C=====================================================================
C OUTPUT PARAMETERS:
C     NU       - INTEGER - NUMBER OF U CONTROL VERTICES
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     COMPUTE THE NEW SET OF CONTROL VERTICES RESULTING FROM
C     THE KNOT INSERTION AND ADD THEM TO THE DATA STRUCTURE
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE NUVERT(COMP, PATCH, NU, FACT)

      INTEGER COMP, PATCH, NU
      REAL FACT(*)

      INTEGER MAXU, MAXW
      PARAMETER (MAXU = 150, MAXW = 150)
      REAL GRID(3, MAXU, MAXW), ULINE(3, MAXU)
      REAL NEWLIN(3, MAXU)

C     GET NUMBER OF VERTICES IN THE W DIRECTION
      CALL GTICMP(6, COMP, NPPXS, IERR)
      NW = NPPXS+2

C     GET THE EXISTING CONTROL GRID
      CALL GTGRID(COMP, NU, NW, GRID)

      DO 300 J = 1, NW
         DO 100 I = 1, NU
            DO 50 K=1,3
C              LOAD ARRAY FOR EACH W vertex
               ULINE(K, I) = GRID(K, I, J)
50          CONTINUE
100      CONTINUE

C        ADJUST U VERTICES FOR THE NEW KNOT VALUE
         CALL ADJVRT(NU, ULINE, FACT, NEWLIN)

C        LOAD GRID INTO DATABASE
         DO 200 I=1, NU+1
               CALL PTBTSF(I, J, 2, COMP, NEWLIN(1,I), IERR)
200      CONTINUE
300   CONTINUE

      RETURN
      END

C=====================================================================
C                       M O D U L E   G T G R I D
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
```

```
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     NU       - INTEGER - NUMBER OF U VERTICES
C     NW       - INTEGER - NUMBER OF W VERTICES
C=====================================================================
C OUTPUT PARAMETERS:
C     GRID     - REAL ARRAY - CONTROL HULL
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     GET CONTROL HULL FROM DATA STRUCTURE
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE GTGRID(COMP, NU, NW, GRID)

      INTEGER COMP, NU, NW
      INTEGER MAXU, MAXW
      PARAMETER (MAXU = 150, MAXW = 150)
      REAL GRID(3, MAXU, MAXW), HULLPT(3)

C     LOOP THROUGH ALL CONTROL POINTS
      DO 200 I=1,NU
         DO 100 J=1,NW
C          GET POINT FROM DATA STRUCTURE
           CALL GTBTSF(I, J, 2, COMP, HULLPT, IER)

C          LOAD GRID ARRAY
           DO 50 K=1,3
              GRID(K, I, J) = HULLPT(K)
50         CONTINUE

100      CONTINUE
200   CONTINUE

      RETURN
      END

C=====================================================================
C                        M O D U L E  A D J V R T
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     NVRT     - INTEGER - NUMBER OF VERTICES IN ROW
C     LINE     - REAL ARRAY - ROW OF CONTROL VERTICES
C     FACT     - REAL ARRAY - FACTOR USED TO ADJUST VERTICES
C=====================================================================
C OUTPUT PARAMETERS:
C     NEWLIN   - REAL ARRAY - NEW OR ADJUSTED ROW OF CONTROL VERTICES
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     ADJUST ONE ROW OF CONTROL VERTICES FOR NEW KNOT
C=====================================================================
```

```
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE ADJVRT(NVRT, LINE, FACT, NEWLIN)

      INTEGER NVRT
      REAL LINE(3,*), FACT(*), NEWLIN(3,*)

C     LOOP THROUGH X, Y, AND Z VALUES
      DO 200 I=1, 3
C        INITIALIZE EXTRA VERTEX
         LINE(I, NVRT+1) = 0.0
C        INITIALIZE NEW VERTICES
         NEWLIN(I,1) = LINE(I,1)

C        LOOP THROUGH ALL VERTICES
         DO 100 J=2, NVRT+1
C           COMPUTE NEW CONTROL VERTEX
            TEMP = (1.0-FACT(J))*LINE(I,J-1)
            NEWLIN(I, J) = TEMP + FACT(J)*LINE(I,J)
100      CONTINUE
200   CONTINUE

      RETURN
      END
C=====================================================================
C                      M O D U L E  A D J U K T
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     PATCH    - INTEGER - SUB-COMPONENT ID
C     NU       - INTEGER - NUMBER OF U VERTICES
C     UKVALS   - REAL ARRAY - U KNOT VALUES
C     UKNOT    - REAL - KNOT VALUE TO BE ADDED
C     INTER    - INTEGER - INTERVAL KNOT IS TO BE ADDED
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE ADJUKT(COMP, PATCH, NU, UKVALS, UKNOT, INTER)

      INTEGER COMP, PATCH, NU, INTER
      REAL UKVALS(*), UKNOT

C     INCREMENT ALL KNOT VALUES BY ONE AFTER INSERTED KNOT
      DO 100 I=NU+4, INTER+2, -1

         UKVALS(I) = UKVALS(I-1)
```

```
100   CONTINUE

C     ADJUST FIRST KNOT AFTER INTERVAL
      UKVALS(INTER+1) = UKNOT

C     ADD UKVALS TO DATABASE
      DO 200 I=1,NU+4
         CALL PTKNOT(I, 1, COMP, UKVALS(I), IERR)
200   CONTINUE

C     ADD NU+1 TO DATABASE
      CALL PTICMP(5, COMP, NU-1, IERR)

      RETURN
      END
C=======================================================================
C                      M O D U L E  W K T R O N
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP    - INTEGER - COMPONENT IDENTIFIER
C     PATCH   - INTEGER - SUB-COMPONENT IDENTIFIER
C     WKNOT   - REAL    - W KNOT VALUE
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     ADD ONE ROW OF W KNOTS AND VERTICES TO THE B-SPLINE SURFACE
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE WKTRON(COMP, PATCH, WKNOT)

      INTEGER COMP, PATCH
      REAL WKNOT

      INTEGER NW, INTER, MAXPTS, NPPXS
      PARAMETER(MAXPTS=150)
      REAL WKVALS(MAXPTS), FACT(MAXPTS)

C     GET NUMBER OF W CONTROL VERTICES FROM DATABASE
      CALL GTICMP(6, COMP, NPPXS, IERR)
      NW = NPPXS+2

C     GET KNOT VALUES IN W DIRECTION FROM DATABASE
      DO 100 I = 1, NW+2
         CALL GTKNOT(I, 2, COMP, WKVALS(I), ERR)
100   CONTINUE

C     FIND THE INTERVAL OF THE KNOT TO BE ADDED
      CALL FINDIT(NW+2, WKVALS, WKNOT, INTER)

C     FIND THE FACTORS USED TO ADJUST THE VERTICES
      CALL AFACT(NW+2, WKVALS, WKNOT, INTER, FACT)

C     FIND THE NEW CONTROL VERTICES AND ADD TO DATABASE
```

```
      CALL NWVERT(COMP, PATCH, NW, FACT)

C     ADJUST KNOT SPACING AND ADD TO DATABASE
      CALL ADJWKT(COMP, PATCH, NW, WKVALS, WKNOT, INTER)

      RETURN
      END
C=======================================================================
C                     M O D U L E   N W V E R T
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     PATCH    - INTEGER - SUB-COMPONENT IDENTIFIER
C     NW       - INTEGER - NUMBER OF W KNOTS
C     FACT     - REAL ARRAY - FACTORS USED TO ADJUST CONTROL VERTICES
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     FIND THE NEW CONTROL VERTICES AND ADD THEM TO THE DATA STRUCTURE
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================
      SUBROUTINE NWVERT(COMP, PATCH, NW, FACT)

      INTEGER COMP, PATCH, NU
      REAL FACT(*)

      INTEGER MAXU, MAXW, NXSECT
      PARAMETER (MAXU = 150, MAXW = 150)
      REAL GRID(3, MAXU, MAXW), WLINE(3, MAXW)
      REAL NEWLIN(3, MAXW)

C     GET NUMBER OF VERTICES IN THE U DIRECTION
      CALL GTICMP(5, COMP, NXSECT, IERR)
      NU = NXSECT+2

C     GET THE EXISTING CONTROL GRID
      CALL GTGRID(COMP, NU, NW, GRID)

      DO 300 J = 1, NU
         DO 100 I = 1, NW
            DO 50 K=1,3
C              LOAD ARRAY FOR EACH W VERTEX
               WLINE(K, I) = GRID(K, J, I)
50          CONTINUE
100      CONTINUE

C        ADJUST W VERTICES FOR THE NEW KNOT VALUE
         CALL ADJVRT(NW, WLINE, FACT, NEWLIN)

         DO 200 I=1, NW+1
C              LOAD GRID ARRAY WITH NEW VERTICES
```

```
               CALL PTBTSF(J, I, 2, COMP, NEWLIN(1,I), IERR)
200       CONTINUE
300    CONTINUE

       RETURN
       END
C=======================================================================
C                        M O D U L E   A D J W K T
C=======================================================================
C  PROJECT:  NASA/AMES                      VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C      COMP     - INTEGER - COMPONENT IDENTIFIER
C      PATCH    - INTEGER - SUB-COMPONENT IDENTIFIER
C      NW       - INTEGER - NUMBER OF W CONTROL VERTICES
C      WKVALS   - REAL ARRAY - W KNOT VALUES
C      WKNOT    - REAL - W KNOT VALUE TO BE ADDED
C      INTER    - INTEGER - INTERVAL OF ADDED KNOT
C=======================================================================
C OUTPUT PARAMETERS:
C      NONE
C=======================================================================
C COMMON INPUTS:
C      NONE
C=======================================================================
C COMMON OUTPUTS:
C      NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C      ADJUST W KNOT SPACING AND ADD KNOTS TO THE DATA STRUCTURE
C=======================================================================
C METHODS/ALGORITHMS:
C      N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================
       SUBROUTINE ADJWKT(COMP, PATCH, NW, WKVALS, WKNOT, INTER)

       INTEGER COMP, PATCH, NW, INTER
       REAL WKVALS(*), WKNOT

C      INCREMENT ALL KNOT VALUES BY ONE AFTER INSERTED KNOT
       DO 100 I=NW+4, INTER+2, -1

          WKVALS(I) = WKVALS(I-1)

100    CONTINUE

C      ADJUST FIRST KNOT AFTER INTERVAL
       WKVALS(INTER+1) = WKNOT

C      ADD UKVALS TO DATABASE
       DO 200 I=1,NW+4
          CALL PTKNOT(I, 2, COMP, WKVALS(I), IERR)
200    CONTINUE

C      ADD NW+1 TO DATABASE
       CALL PTICMP(6, COMP, NW-1, IERR)

       RETURN
       END
```

Appendix C

Program NONINVT

```
C=====================================================================
C                      M O D U L E   N N V E H L
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C INPUT PARAMETERS:
C     COMP   - INTEGER, COMPONENT NUMBER
C     NXSECT - INTEGER, NUMBER OF CROSS-SECTIONS
C     NPPXS  - INTEGER, NUMBER OF POINTS PER CROSS-SECTION
C     PTS    - REAL, POINTS TO BE INTERPOLATED
C     FLAGS  - INTEGER, FLAGS FOR EACH POINT:
C                       1 - OPEN POINT
C                       2 - CLOSED POINT
C                       3 - HOLD POINT - DO NOT MODIFY
C                       4 - INTERPOLATE POINT
C     UKNTS  - REAL, U KNOTS
C     WKNTS  - REAL, W KNOTS
C=====================================================================
C OUTPUT PARAMETERS:
C     HUL    - REAL, CONTROL HULL POINTS WHICH ARE CALCULATED
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     ITERATIVELY FINDS THE CONTROL HULL FOR A NON-UNIFORM
C     B-SPLINE SURFACE - USE FLAGS TO MAINTAIN END CONDITIONS
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
```

```
      SUBROUTINE NNVEHL(NXSECT, NPPXS, PTS, FLAGS, UKNTS, WKNTS, HUL)

C        DECLARE THE VARIABLES
      INTEGER NXSECT, NPPXS

      INTEGER SECT, NODE, DERIV, ERRIND, I
      PARAMETER(SECT=50, NODE=50, DERIV=4)
      INTEGER FLAGS(SECT, NODE), TFLAG(SECT, NODE)
      REAL PTS(3, SECT, NODE), HUL(3, SECT, NODE)
      REAL TMP(3, SECT, NODE), UKNTS(*)
      REAL WKNTS(*), N(3, SECT+2)

C        COMPUTE NON-UNIFORM BLENDING FUNCTIONS
         CALL NONBLN(NXSECT+2, UKNTS, N)

      DO 10 I=1, NPPXS+2
C           INVERT THE SINGLE ARRAY OF POINTS
         CALL NNVEPT(NXSECT+2, PTS(1,1,I), FLAGS(1,I), N, HUL(1,1,I))
 10   CONTINUE

C        SWITCH THE ROWS AND COLUMNS OF (3,R,C) MATRIX
      CALL MATSWC(NPPXS+2, NXSECT+2, HUL, PTS)

C        SWITCH THE ROWS AND COLUMNS OF (R,C) MATRIX
      CALL MTRANS(NPPXS+2, NXSECT+2, FLAGS, TFLAG)

C        COMPUTE NON-UNIFORM BLENDING FUNCTIONS
         CALL NONBLN(NPPXS+2, WKNTS, N)
C
      DO 20 I=1, NXSECT+2
C           INVERT ARRAY OF POINTS IN OTHER DIRECTION
         CALL NNVEPT(NPPXS+2, PTS(1,1,I), TFLAG(1,I), N, TMP(1,1,I))
 20   CONTINUE

C        SWITCH THE ROWS AND COLUMNS OF (3,R,C) MATRIX
      CALL MATSWC(NXSECT+2, NPPXS+2, TMP, HUL)

C        RETURN
      RETURN
      END
C=======================================================================
C                      M O D U L E   N N V E P T
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     NPTS    - INTEGER, NUMBER OF POINTS ON CURVE
C     PTS     - REAL, (3, R, C) MATRIX OF POINTS
C     FLAGS   - INTEGER, FLAGS FOR EACH POINT:
C                        1 - OPEN POINT
C                        2 - CLOSED POINT
C                        3 - HOLD POINT - DO NOT MODIFY
C                        4 - INTERPOLATE POINT
C     N       - REAL, NON-UNIFORM BLENDING FUNCTIONS
C=======================================================================
C OUTPUT PARAMETERS:
C     HUL     - REAL, (3, C, R) OUTPUT MATRIX
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     INVERTS A BUNCH OF N POINTS TO RETURN (N+2) CONTROL POINTS
C     OF A UNIFORM B-SPLINE CURVE
```

```
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE NNVEPT(NPTS, PTS, FLAGS, N,  HUL)


C     DECLARE THE VARIABLES
      INTEGER END, NPTS, FLAGS(*)
      REAL PTS(3,*), HUL(3,*), N(3,*)

C     INITIALIZE THE CONTROL HULL
      CALL INITEL(NPTS, PTS, FLAGS, HUL)

C     FIND HUL
      CALL NNDEHL(NPTS, PTS, FLAGS, N, HUL)

C        RETURN
      RETURN
      END
C=====================================================================
C                      M O D U L E   I N I T E L
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     NPTS   - INTEGER, NUMBER OF POINTS ON CURVE
C     PTS    - REAL, (3, R, C) MATRIX OF POINTS
C     FLAGS  - INTEGER, FLAGS FOR EACH POINT:
C                       1 - OPEN POINT
C                       2 - CLOSED POINT
C                       3 - HOLD POINT - DO NOT MODIFY
C                       4 - INTERPOLATE POINT
C=====================================================================
C OUTPUT PARAMETERS:
C     HUL    - REAL, (3, C, R) OUTPUT MATRIX
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     INITIALIZES THE CONTROL HULL TO BEGIN ITERATIVE TECHNIQUE
C=====================================================================
C METHODS/ALGORITHMS:
C     ITERATIVE TECHNIQUE
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE INITEL(NPTS, PTS, FLGS, HUL)


C        DECLARE THE VARIABLES
      INTEGER END, NPTS, FLGS(*)
      REAL PTS(3,*), HUL(3,*)

      INTEGER I,J

C
      DO 20 I=1,3
        DO 10 J=1, NPTS
           HUL(I,J) = PTS(I,J)
```

```
 10     CONTINUE
C          INITIALIZE THE END POINTS
        IF(FLGS(1) .EQ. 1) THEN
           HUL(I,1) = HUL(I,2)
        ELSE IF (FLGS(1) .EQ. 2) THEN
           HUL(I,1) = HUL(I,NPTS-2)
        ENDIF

        IF (FLGS(NPTS) .EQ. 1) THEN
           HUL(I, NPTS) = HUL(I, NPTS-1)
        ELSE IF (FLGS(NPTS) .EQ. 2) THEN
           HUL(I,NPTS) = HUL(I,3)
        ENDIF
 20   CONTINUE

C        RETURN
      RETURN
      END
C=======================================================================
C                       M O D U L E   N N D E H L
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     NPTS    - INTEGER, NUMBER OF POINTS ON CURVE
C     PTS     - REAL, (3, R, C) MATRIX OF POINTS
C     FLAGS   - INTEGER, FLAGS FOR EACH POINT:
C                        1 - OPEN POINT
C                        2 - CLOSED POINT
C                        3 - HOLD POINT - DO NOT MODIFY
C                        4 - INTERPOLATE POINT
C     N       - REAL, NON-UNIFORM BLENDING FUNCTIONS
C=======================================================================
C OUTPUT PARAMETERS:
C     HUL     - REAL, (3, C, R) OUTPUT MATRIX
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     ITERATES TO FIND THE CONTROL HULL FOR THE GIVEN SET
C     OF POINTS TO BE INTERPOLATED BY A NON-UNIFORM B-SPLINE CURVE
C=======================================================================
C METHODS/ALGORITHMS:
C     ITERATIVE TECHNIQUE
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE NNDEHL(NPTS, PTS, FLAGS, N, HUL)

C        DECLARE THE VARIABLES
      INTEGER END, NPTS, FLAGS(*)
      REAL PTS(3,*), HUL(3,*)
      REAL N(3,*)

      REAL ERR, DEL, DELMAX
      PARAMETER(ERR = 0.000001)
      INTEGER I, J

C        INITIALIZE DELMAX
  5   DELMAX = 0.0

      DO 20 I=1,3
```

```
        DO 10 J=2, NPTS-1
C           CHECK FOR HOLD FLAG
         IF (FLAGS(J) .EQ. 4) THEN
C              FIND DELTA ERROR
           CALL NONDEL(PTS(I,J),HUL(I,J-1),HUL(I,J),HUL(I,J+1),N(1,J+1),
     >                 DEL)
C              FIND DELMAX
           IF(ABS(DEL).GE.DELMAX) DELMAX = ABS(DEL)
         ENDIF

 10     CONTINUE

C          INITIALIZE THE END POINTS
        IF(FLAGS(1) .EQ. 1) THEN
           HUL(I,1) = HUL(I,2)
        ELSE IF (FLAGS(1) .EQ. 2) THEN
           HUL(I,1) = HUL(I,NPTS-2)
        ENDIF

        IF (FLAGS(NPTS) .EQ. 1) THEN
           HUL(I,NPTS) = HUL(I,NPTS-1)
        ELSE IF (FLAGS(NPTS) .EQ. 2) THEN
           HUL(I,NPTS) = HUL(I,3)
        ENDIF

 20   CONTINUE

C        CHECK FOR TERMINATION OF ITERATION
      IF(DELMAX.GT.ERR) GOTO 5

C        RETURN
      RETURN
      END
C=======================================================================
C                      M O D U L E   N O N D E L
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     PTS     - REAL, POINT TO INTERPOLATE
C     HUL     - REAL, CONTROL POINTS
C     KNTS    - REAL, KNOT VALUES
C=======================================================================
C OUTPUT PARAMETERS:
C     DEL     - REAL, ERROR DELTA
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     ITERATES TO FIND THE CONTROL HULL FOR THE GIVEN SET
C     OF POINTS TO BE INTERPOLATED BY A NON-UNIFORM B-SPLINE CURVE
C=======================================================================
C METHODS/ALGORITHMS:
C     ITERATIVE TECHNIQUE
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE NONDEL(PI, QA, QI, QB, N, DELTA)

C      DECLARE THE VARIABLES
      REAL PI, QA, QI, QB, DELTA, N(3)
      REAL TOL
      PARAMETER(TOL = .1)
```

```
C     CHECK FOR NON-CONVERGENCE PROBLEMS
      IF (N(2) .LT. .00000001) THEN
         QI = PI
         DELTA = 0.0
      ELSE
C      FIND DELTA
         DELTA = -QI + (PI-(N(1)*QA)-(N(3)*QB))/N(2)
         QI = QI + DELTA
      ENDIF

      RETURN
      END
C=======================================================================
C                      M O D U L E   N O N B L N
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     NPTS     - INTEGER - NUMBER OF POINTS
C     KNOTS    - REAL ARRAY - KNOT VALUES
C=======================================================================
C OUTPUT PARAMETERS:
C     N        - REAL ARRAY - NON-UNIFORM BLENDING FUNCTIONS
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     COMPUTE NON-UNIFORM B-SPLINE BLENDING FUNCTIONS
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================
      SUBROUTINE NONBLN(NPTS, KNOTS, N)

      INTEGER NPTS
      REAL KNOTS(*), N(3,*)
      REAL BLEND(4), TOL
      PARAMETER(TOL = 1E-20)

      DO 100 I=3, NPTS+1

C        FIND INTERVAL FOR KNOT VALUE
         CALL FINDIT(NPTS+4, KNOTS, KNOTS(I), INTER)

C        FIND BLENDING FUNCTION
         CALL BLENDT(KNOTS(I), KNOTS, INTER, BLEND)

C        COPY BLENDING FUNCTION INTO ARRAY
         N(1,I) = BLEND(1)
         N(2,I) = BLEND(2)
         N(3,I) = BLEND(3)

100   CONTINUE

C     FIND BLEND FOR LAST POINT
      CALL FINDIT(NPTS+4, KNOTS, KNOTS(NPTS+2)-TOL, INTER)
      CALL BLENDT(KNOTS(NPTS+2)-TOL, KNOTS, INTER, BLEND)

C     FILL LAST ARRAY VALUE
      N(1,NPTS+2) = BLEND(2)
```

```
      N(2,NPTS+2) = BLEND(3)
      N(3,NPTS+2) = BLEND(4)

      RETURN
      END
C=====================================================================
C                      M O D U L E   B L E N D T
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     T       - REAL    - PARAMETER VALUE
C     KNOTS   - REAL ARRAY - KNOTS VALUES
C     INTER   - INTEGER - INTERVAL OF PARAMETER VALUE
C=====================================================================
C OUTPUT PARAMETERS:
C     BLEND   - REAL ARRAY - FOUR COMPONENTS OF BLENDING FUNCTION
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     COMPUTE NON-UNIFORM B-SPLINE BLENDING FUNCTIONS
C=====================================================================
C METHODS/ALGORITHMS:
C     ADAPTED FROM C. DEBOOR'S 1970 PAPER, "ON CALCULATING WITH
C     B-SPLINES"
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE BLENDT(T, KNOTS, INTER, BLEND)

      INTEGER INTER
      REAL T, KNOTS(*), BLEND(4)

      REAL DP(3), DM(3), N(4,4), TEMP, DENM

C     INITIALIZE FIRST BLEND VALUE
      N(1,1) = 1.0

C     LOOP THROUGH FOR CUBIC B-SPLINE
      DO 200 ISS = 1,3

C        CALCULATE INTERMEDIATE VALUES
         DP(ISS) = KNOTS(INTER+ISS) - T
         DM(ISS) = T - KNOTS(INTER+1-ISS)
         N(1, ISS+1) = 0.0

         DO 100 IR = 1, ISS
C           COMPUTE DENOMINATOR
            DENM = DP(IR)+DM(ISS+1-IR)

C           CHECK FOR DIVISION BY ZERO
            IF (DENM .EQ. 0.0) THEN
               TEMP = 0.0
            ELSE
               TEMP = N(IR, ISS)/DENM
            ENDIF

C           COMPUTE BLEND VALUES
            N(IR, ISS+1) = N(IR, ISS+1) + DP(IR)*TEMP
            N(IR+1, ISS+1) = DM(ISS+1-IR)*TEMP
```

```
100      CONTINUE
200   CONTINUE

      DO 300 I=1,4
C        FILL BLEND ARRAY
         BLEND(I) = N(I,4)

300   CONTINUE

      RETURN
      END
C=======================================================================
C                      M O D U L E   M T R A N S
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     NROWS  - INTEGER, NUMBER OF ROWS
C     NCOLS  - INTEGER, NUMBER OF COLUMNS
C     MATIN  - REAL, (R, C) INPUT MATRIX
C=======================================================================
C OUTPUT PARAMETERS:
C     MATOUT - REAL, (C, R) OUTPUT MATRIX
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     SWITCHES THE ROWS AND COLUMNS OF (R,C,) MATRIX
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  K. V. KOLADY                                 DATE:  3/15/90
C=======================================================================
      SUBROUTINE MTRANS(NROWS, NCOLS, MATN, MATOT)


C        DECLARE THE VARIABLES
      INTEGER NROWS, NCOLS

      INTEGER SECT, NODE
      PARAMETER(SECT=50, NODE=50)
      INTEGER MATN(SECT, NODE), MATOT(SECT, NODE)

C
      DO 30 I=1,NROWS
         DO 20 J=1,NCOLS
C                TRANSPOSE THE ROWS AND COLUMS
               MATOT(I, J) = MATN(J, I)
 20      CONTINUE
 30   CONTINUE

C        RETURN
      RETURN
      END
```

Appendix D

Program RENDER

```
C=====================================================================
C                      M O D U L E   N O N S R F
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C      COMP     - INTEGER - COMPONENT IDENTIFIER
C      NXSECT   - INTEGER - NUMBER OF X-SECTIONS
C      NPPXS    - INTEGER - NUMBER OF POINTS PER X-SECTION
C      NU       - INTEGER - NUMBER U LINES PER PATCH
C      NW       - INTEGER - NUMBER W LINES PER PATCH
C=====================================================================
C OUTPUT PARAMETERS:
C      NONE
C=====================================================================
C COMMON INPUTS:
C      NONE
C=====================================================================
C COMMON OUTPUTS:
C      NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C      DRAW WIRE FRAME IMAGE OF A NON-UNIFORM B-SPLINE SURFACE
C=====================================================================
C METHODS/ALGORITHMS:
C      N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE NONSRF(COMP, NXSECT, NPPXS, NU, NW)

      INTEGER COMP, NXSECT, NPPXS, NU, NW
      INTEGER TOTU, TOTW
      REAL P(3)

C     COMPUTE TOTAL NUMBER OF U AND W LINES
      TOTU = INT((NPPXS-1)*NU+1)
```

```
      TOTW = INT((NXSECT-1)*NW+1)

C     GENERATE U LINES
      CALL UNON(COMP, NXSECT, NPPXS, TOTU, TOTW)

C     GENERATE W LINES
      CALL WNON(COMP, NXSECT, NPPXS, TOTU, TOTW)

      RETURN
      END
C=======================================================================
C                      M O D U L E   U N O N
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     NXSECT   - INTEGER - NUMBER OF X-SECTIONS
C     NPPXS    - INTEGER - NUMBER OF POINTS PER X-SECTION
C     TOTU     - INTEGER - TOTAL NUMBER U LINES
C     TOTW     - INTEGER - TOTAL NUMBER W LINES
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     GENERATE U PARAMETRIC LINES FOR NON-UNIFORM SURFACE
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE UNON(COMP, NXSECT, NPPXS, TOTU, TOTW)

      INTEGER COMP, NXSECT, NPPXS, TOTU, TOTW
      INTEGER MAXKNT, NPTS
      PARAMETER (MAXKNT = 50, NPTS=10)

      INTEGER IU, IW
      REAL UKNTS(MAXKNT), WKNTS(MAXKNT), UDIFF, WDIFF
      REAL BLNU(4), BLNW(4), XX(NPTS), YY(NPTS), ZZ(NPTS)
      REAL TOL, WS(300)
      PARAMETER (TOL = 0.0000000001)

C     LOAD KNOT ARRAYS
      CALL LOADKT(COMP, NXSECT, NPPXS, UKNTS, WKNTS)

C     FIND DIFFERENCE BETWEEN FIRST AND LAST KNOT VALUES
      CALL GETDIF(NXSECT, NPPXS, UKNTS, WKNTS, UDIFF, WDIFF)

C     LOOP THROUGH ALL U LINES
      DO 200 I=0, TOTU-1

C        COMPUTE W BLENDING FUNCTIONS AND INTERVAL
         CALL BLNINT(I, TOTU, NPPXS+4, WDIFF, WKNTS, IW, BLNW)

C        STEP ALONG U LINE
         DO 100 J=3, NXSECT+1

            UDIFF = UKNTS(J+1)-UKNTS(J)-TOL
```

```
            IF (UDIFF .GT. 0.0) THEN

            DO 50 K=0,NPTS-1

C              COMPUTE U BLENDING FUNCTIONS AND INTERVAL
               CALL SMLBLN(K, NPTS, UDIFF, UKNTS, J, BLNU)

C              COMPUTE POINT ON SURFACE
               CALL NONPT(COMP,J,IW,BLNU,BLNW,XX(K+1),YY(K+1),ZZ(K+1))

50          CONTINUE

            ENDIF

C           DRAW POLYLINE
            CALL VPL3(NPTS, XX, YY, ZZ)

100      CONTINUE

200   CONTINUE

      RETURN
      END

C=====================================================================
C                      M O D U L E  W N O N
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     NXSECT   - INTEGER - NUMBER OF X-SECTIONS
C     NPPXS    - INTEGER - NUMBER OF POINTS PER X-SECTION
C     TOTU     - INTEGER - TOTAL NUMBER U LINES
C     TOTW     - INTEGER - TOTAL NUMBER W LINES
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     GENERATE W PARAMETRIC LINES FOR NON-UNIFORM SURFACE
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE WNON(COMP, NXSECT, NPPXS, TOTU, TOTW)

      INTEGER COMP, NXSECT, NPPXS, TOTU, TOTW
      INTEGER MAXKNT, NPTS
      PARAMETER (MAXKNT = 50, NPTS=10)

      INTEGER IU, IW
      REAL UKNTS(MAXKNT), WKNTS(MAXKNT), UDIFF, WDIFF
      REAL BLNU(4), BLNW(4), XX(NPTS), YY(NPTS), ZZ(NPTS)
      REAL TOL, US(200)
      PARAMETER (TOL = 0.00000000001)

C     LOAD KNOT ARRAYS
```

```
      CALL LOADKT(COMP, NXSECT, NPPXS, UKNTS, WKNTS)

C     FIND DIFFERENCE BETWEEN FIRST AND LAST KNOT VALUES
      CALL UDIST(COMP, NXSECT, UKNTS, TOTW, US)

C     LOOP THROUGH ALL W LINES
      DO 200 I=1, TOTW

C        COMPUTE U BLENDING FUNCTIONS AND INTERVAL
         CALL TBLNIN(US(I), NXSECT+4, UKNTS, IU, BLNU)

C        STEP ALONG U LINE
         DO 100 J=3, NPPXS+1

            WDIFF = WKNTS(J+1)-WKNTS(J)-TOL

            IF (WDIFF .NE. 0.0) THEN

            DO 50 K=0,NPTS-1

C              COMPUTE W BLENDING FUNCTIONS AND INTERVAL
               CALL SMLBLN(K, NPTS, WDIFF, WKNTS, J, BLNW)

C              COMPUTE POINT ON SURFACE
               CALL NONPT(COMP,IU,J,BLNU,BLNW,XX(K+1),YY(K+1),ZZ(K+1))

50          CONTINUE

            ENDIF

C           DRAW POLYLINE
            CALL VPL3(NPTS, XX, YY, ZZ)

100      CONTINUE
200   CONTINUE

      RETURN
      END
C=====================================================================
C                        M O D U L E   L O A D K T
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     NXSECT   - INTEGER - NUMBER OF X-SECTIONS
C     NPPXS    - INTEGER - NUMBER OF POINTS PER X-SECTIONS
C=====================================================================
C OUTPUT PARAMETERS:
C     UKNTS    - REAL ARRAY - U KNOT ARRAY
C     WKNTS    - REAL ARRAY - W KNOT ARRAY
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     LOAD KNOT VALUES INTO ARRAYS
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE LOADKT(COMP, NXSECT, NPPXS, UKNTS, WKNTS)
```

```
      INTEGER COMP, NXSECT, NPPXS
      REAL UKNTS(*), WKNTS(*)

C     LOOP THROUGH ALL U KNOTS
      DO 100 I=1, NXSECT+5
C        GET U KNOTS
         CALL GTKNOT(I, 1, COMP, UKNTS(I), IERR)

100   CONTINUE

C     LOOP THROUGH ALL W KNOTS
      DO 200 I=1, NPPXS+5
C        GET W KNOTS
         CALL GTKNOT(I, 2, COMP, WKNTS(I), IERR)

200   CONTINUE

      RETURN
      END

C=======================================================================
C                      M O D U L E  G E T D I F
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     NXSECT   - INTEGER - NUMBER OF X-SECTIONS
C     NPPXS    - INTEGER - NUMBER OF POINTS PER X-SECTIONS
C     UKNTS    - REAL ARRAY - U KNOT ARRAY
C     WKNTS    - REAL ARRAY - W KNOT ARRAY
C=======================================================================
C OUTPUT PARAMETERS:
C     UDIFF    - REAL - DIFFERENCE BETWEEN LOW AND HIGH U KNOT VALUES
C     WDIFF    - REAL - DIFFERENCE BETWEEN LOW AND HIGH W KNOT VALUES
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     COMPUTE DIFFERENCE BETWEEN HIGH AND LOW KNOT VALUES
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================
      SUBROUTINE GETDIF(NXSECT, NPPXS, UKNTS, WKNTS, UDIFF, WDIFF)

      INTEGER NXSECT, NPPXS
      REAL UKNTS(*), WKNTS(*), UDIFF, WDIFF, TOL
      PARAMETER  (TOL = 0.000000000001)

C     COMPUTE DIFFERENCES
      UDIFF = UKNTS(NXSECT+2) - UKNTS(3)-TOL
      WDIFF = WKNTS(NPPXS+2) - WKNTS(3)-TOL

      RETURN
      END

C=======================================================================
C                      M O D U L E  B L N I N T
C=======================================================================
```

```
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C      IPOINT   - INTEGER - CURRENT POINT LOCATION ALONG CURVE
C      TO       - INTEGER - TOTAL NUMBER OF POINTS
C      NKNTS    - INTEGER - NUMBER OF KNOT VALUES
C      DIFF     - REAL    - DIFFERENCE BETWEEN FIRST AND LAST KNOTS
C      KNTS     - REAL ARRAY - ARRAY OF KNOT VALUES
C=======================================================================
C OUTPUT PARAMETERS:
C      INTER    - INTEGER - INDEX CORRESPONDING TO PARAMETER VALUE
C      BLEND    - REAL ARRAY - BLENDING FUNCTION
C=======================================================================
C COMMON INPUTS:
C      NONE
C=======================================================================
C COMMON OUTPUTS:
C      NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C      COMPUTE INTERVAL AND BLENDING FUNCTION FOR A GIVEN PARAMETER
C      VALUE
C=======================================================================
C METHODS/ALGORITHMS:
C      N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
       SUBROUTINE BLNINT(IPOINT, TOT, NKNTS, DIFF, KNTS, INTER, BLEND)

       INTEGER IPOINT, TOT, NKNTS, INTER
       REAL DIFF, KNTS(*), BLEND(4), T

C      COMPUTE KNOT VALUE
       T = KNTS(3) + DIFF*FLOAT(IPOINT)/FLOAT(TOT-1)

C      FIND INTERVAL FOR KNOT VALUE
       CALL FINDIT(NKNTS, KNTS, T, INTER)

C      COMPUTE NON-UNIFORM BLENDING FUNCTION
       CALL BLENDT(T, KNTS, INTER, BLEND)

       RETURN
       END
C=======================================================================
C                        M O D U L E   N O N P T
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C      COMP     - INTEGER - COMPONENT IDENTIFIER
C      UINT     - INTEGER - U INTERVAL
C      WINT     - INTEGER - W INTERVAL
C      BLNU     - REAL ARRAY - U BLENDING FUNCTION
C      BLNW     - REAL ARRAY - W BLENDING FUNCTION
C=======================================================================
C OUTPUT PARAMETERS:
C      XPT      - REAL - X VALUE OF POINT
C      YPT      - REAL - Y VALUE OF POINT
C      ZPT      - REAL - Z VALUE OF POINT
C=======================================================================
C COMMON INPUTS:
C      NONE
C=======================================================================
C COMMON OUTPUTS:
C      NONE
C=======================================================================
```

```
C FUNCTIONAL DESCRIPTION:
C     COMPUTE A POINT ON A NON-UNIFORM B-SPLINE SURFACE
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=====================================================================
      SUBROUTINE NONPT(COMP, UINT, WINT, BLNU, BLNW, XPT, YPT, ZPT)

      INTEGER COMP, UINT, WINT
      REAL BLNU(4), BLNW(4), XPT, YPT, ZPT

      REAL Q(3), TEMP(1,4), ROW(1,4), COL(4,1)
      REAL XHUL(4,4), YHUL(4,4), ZHUL(4,4)

C     LOAD ROWS AND COLUMNS FOR MATRIX MULT
      DO 100 I = 1,4

         ROW(1,I) = BLNU(I)
         COL(I,1) = BLNW(I)

100   CONTINUE

C     LOAD CONTROL VERTICES
      DO 300 I=1,4
         DO 200 J=1,4

            CALL GTBTSF(UINT+I-3, WINT+J-3, 2, COMP, Q, IERR)
            XHUL(I,J) = Q(1)
            YHUL(I,J) = Q(2)
            ZHUL(I,J) = Q(3)

200      CONTINUE
300   CONTINUE

C     PERFORM MATRIX MULT

      CALL BMATM(1, 4, 4, ROW, XHUL, TEMP)
      CALL BMATM(1, 4, 1, TEMP, COL, XPT)

      CALL BMATM(1, 4, 4, ROW, YHUL, TEMP)
      CALL BMATM(1, 4, 1, TEMP, COL, YPT)

      CALL BMATM(1, 4, 4, ROW, ZHUL, TEMP)
      CALL BMATM(1, 4, 1, TEMP, COL, ZPT)

      RETURN
      END

C=====================================================================
C                    M O D U L E   S M L B L N
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     IPOINT   - INTEGER - CURRENT POINT VALUE
C     NTOT     - INTEGER - TOTAL NUMBER OF POINTS
C     DIFF     - REAL    - DIFFERENCE BETWEEN FIRST AND LAST KNOTS
C     KNTS     - REAL ARRAY - KNOT VALUES
C     INTER    - INTEGER - INTERVAL OF PARAMETER
C=====================================================================
C OUTPUT PARAMETERS:
C     BLEND    - REAL ARRAY - BLENDING FUNCTION
C=====================================================================
C COMMON INPUTS:
C     NONE
```

```
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     COMPUTE BLENDING FUNCTION VALUES FOR A POINT ON A NON-UNIFORM
C     B-SPLINE SURFACE
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE SMLBLN(IPOINT, NTOT, DIFF, KNTS, INTER, BLEND)

      INTEGER IPOINT, NTOT, INTER
      REAL DIFF, KNTS(*), BLEND(4), T

C     COMPUTE KNOT VALUE
      T = KNTS(INTER) + DIFF*FLOAT(IPOINT)/FLOAT(NTOT-1)

C     COMPUTE BLENDING FUNCTIONS
      CALL BLENDT(T, KNTS, INTER, BLEND)

      RETURN
      END
C=====================================================================
C                      M O D U L E  U D I S T
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     NXSECT   - INTEGER - NUMBER OF X-SECTIONS
C     UKNTS    - REAL ARRAY - U KNOT VALUES
C     TOTW     - INTEGER - TOTAL NUMBER OF W LINES
C=====================================================================
C OUTPUT PARAMETERS:
C     US       - REAL ARRAY - CORRECTLY SPACED U VALUES
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     COMPUTE PARAMETER SPACING REQUIRED TO MAKE U LINES EQUALLY
C     SPACED
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE UDIST(COMP, NXSECT, UKNTS, TOTW, US)

      INTEGER COMP, NXSECT, TOTW
      REAL UKNTS(*), US(*), TOL
      PARAMETER (TOL = 0.00000000001)

      INTEGER MAXPTS
      PARAMETER (MAXPTS = 50)
      REAL TDIST(MAXPTS), UDISTF(MAXPTS), DIST, STEP
      REAL P1(3), P2(3), P3(3)

C     INITIALIZE DISTANCE ARRAY
```

```
      TDIST(1) = 0.0

C     LOOP THROUGH ALL POINTS
      DO 100 I=2, NXSECT

C        COMPUTE DELTA U DIST
         UKNTD = UKNTS(I+2) - UKNTS(I+1)
         IF (UKNTD .LT. TOL) THEN
            DIST = 0.0
            UDISTF(I) = 1.0
         ELSE
C          GET POINTS FROM DATA STRUCTURE
           CALL GTBTSF(I, 2, 2, COMP, P1, IERR)
           CALL GTBTSF(I+1, 2, 2, COMP, P2, IERR)
           CALL SUB(P2, P1, P3)
           CALL VECVAL(P3, DIST)
           UDISTF(I-1) = DIST/UKNTD
         ENDIF
         TDIST(I) = DIST+TDIST(I-1)

100   CONTINUE

C     CALCULATE STEP
      STEP = TDIST(NXSECT)/(FLOAT(TOTW-1)+TOL)

C     COMPUTE ADJUSTED PARAMETER SPACING
      US(1) = UKNTS(3)
      ITOP = 2
      DO 200 I=2, TOTW
C        COMPUTE REQUIRED DISTANCE
         REQD = STEP*FLOAT(I-1)
         DO 150 J=ITOP, NXSECT
C           CHECK TOLERANCE
            IF (TDIST(J) .GE. REQD) THEN
               ITOP = J-1
               GOTO 175
            ENDIF

150      CONTINUE

C        LOAD ADJUSTED U PARAMETER ARRAY
175      US(I) = UKNTS(ITOP+2) + (REQD - TDIST(ITOP))/UDISTF(ITOP)

200   CONTINUE

      RETURN
      END
C=====================================================================
C                      M O D U L E  W D I S T
C=====================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     NPPXS    - INTEGER - NUMBER OF POINTS PER X-SECTIONS
C     WKNTS    - REAL ARRAY - W KNOT VALUES
C     TOTU     - INTEGER - TOTAL NUMBER OF U LINES
C=====================================================================
C OUTPUT PARAMETERS:
C     WS       - REAL ARRAY - CORRECTLY SPACED W VALUES
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
```

```
C FUNCTIONAL DESCRIPTION:
C     COMPUTE PARAMETER SPACING REQUIRED TO MAKE W LINES EQUALLY
C     SPACED
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE WDIST(COMP, NPPXS, WKNTS, TOTU, WS)

      INTEGER COMP, NPPXS, TOTU
      REAL WKNTS(*), WS(*), TOL
      PARAMETER (TOL = 0.00000000001)

      INTEGER MAXPTS
      PARAMETER (MAXPTS = 50)
      REAL TDIST(MAXPTS), WDISTF(MAXPTS), DIST, STEP
      REAL P1(3), P2(3), P3(3)

C     INITIALIZE DISTANCE ARRAY
      TDIST(1) = 0.0

C     LOOP THROUGH ALL POINTS
      DO 100 I=2, NPPXS

C        COMPUTE DELTA W DIST
         WKNTD = WKNTS(I+2) - WKNTS(I+1)
         IF (WKNTD .LT. TOL) THEN
            DIST = 0.0
            WDISTF(I) = 1.0
         ELSE
C           GET POINTS FROM DATA STRUCTURE
            CALL GTBTSF(2, I, 2, COMP, P1, IERR)
            CALL GTBTSF(2, I+1, 2, COMP, P2, IERR)
            CALL SUB(P2, P1, P3)
            CALL VECVAL(P3, DIST)
            WDISTF(I-1) = DIST/WKNTD
         ENDIF
         TDIST(I) = DIST+TDIST(I-1)

100   CONTINUE

C     CALCULATE STEP
      STEP = TDIST(NPPXS)/(FLOAT(TOTU-1)+TOL)


C     COMPUTE ADJUSTED PARAMETER SPACING
      WS(1) = WKNTS(3)
      ITOP = 2
      DO 200 I=2, TOTU

C        COMPUTE REQUIRED DISTANCE
         REQD = STEP*FLOAT(I-1)
         DO 150 J=ITOP, NPPXS
C           CHECK TOLERANCE
            IF (TDIST(J) .GE. REQD) THEN
               ITOP = J-1
               GOTO 175
            ENDIF

150      CONTINUE

C        LOAD ADJUSTED W PARAMETER ARRAY
175      WS(I) = WKNTS(ITOP+2) + (REQD - TDIST(ITOP))/WDISTF(ITOP)

200   CONTINUE
```

```
      RETURN
      END
C=====================================================================
C                       M O D U L E   T B L N I N
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     T       - REAL    - KNOT VALUE
C     NKNTS   - INTEGER - NUMBER OF KNOTS
C     KNTS    - REAL ARRAY - KNOT VALUE
C=====================================================================
C OUTPUT PARAMETERS:
C     INTER   - INTEGER - INTERVAL FOR T
C     BLEND   - REAL ARRAY - BLENDING FUNCTIONS FOR PARAMETER T
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     COMPUTE BLENDING FUNCTIONS AND KNOT INTERVAL
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================

      SUBROUTINE TBLNIN(T, NKNTS, KNTS, INTER, BLEND)

      INTEGER IPOINT, TOT, NKNTS, INTER
      REAL DIFF, KNTS(*), BLEND(4), T

C     FIND INTERVAL FOR KNOT VALUE
      CALL FINDIT(NKNTS, KNTS, T, INTER)

C     COMPUTE NON-UNIFORM BLENDING FUNCTION
      CALL BLENDT(T, KNTS, INTER, BLEND)

      RETURN
      END
C=====================================================================
C                       M O D U L E   B S T I L E
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     WKID    - INTEGER - WORKSTATION ID
C     NCOMP   - INTEGER - NO. OF COMPONENTS TO BE TILED
C     COMPID  - INTEGER(NCOMP) - ARRAY OF IDS OF COMPONENTS TO BE TILED
C     VIEWID  - INTEGER - VIEW INDEX TO ASSOCIATE THE TILED IMAGE WITH
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C   THE MODULE TILEIT GENERATES A POLYGONAL MODEL.  THE DATA FOR THE
```

```
C POLYGONS FOR EACH COMPONENT IS GENERATED AND LOADED INTO A COMMON
C BLOCK.  THE POLYGON VERTICES ARE THEN TRANSFORMED TO THE CORRECT
C LOCATION TO LOCATE AND ORIENT THE PART.  LOCAL AND GLOBAL SYMMETRIES
C ARE TAKEN CARE OF BY COPYING THE POLYGONS AND MOVING THEM TO THE
C CORRECT GLOBAL LOCATION.
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  JAYARAM SANKAR                              DATE:  1/17/88
C MODIFIED:  J. R. GLOUDEMANS                            DATE:  3/15/90
C=====================================================================

      SUBROUTINE BSTILE (WKID, NCOMP, COMPID, VIEWID)

      INTEGER COMPID(100), VIEWID, WKID
      INTEGER COMPS(200), NNN, PARTS(10)

      REAL XYZSUB(3,100000,5)

      INTEGER NPSUB(100000), NPARTS, PRTLMT(2,100)

      COMMON / POLDAT / NPARTS, PRTLMT, NSUBT, NPSUB, XYZSUB

      INTEGER IEXST

C     SET THE NUMBER OF PARTS TO ZERO
      NPARTS = 0

C     SET THE NUMBER OF POLYGONS TO ZERO
      NSUBT = 0

C     STEP THROUGH EACH COMPONENT TO GENERATE THE POLYGONS
      CALL GTBTCM(NNN, COMPS)

      DO 100 ICOMP = 1, NNN

         IF ( COMPS(ICOMP) .GT. 100) THEN
            ITCOMP = COMPS(ICOMP) - 100
         ELSE
            ITCOMP = COMPS(ICOMP)
         ENDIF

C     GET THE EXISTENCE FLAG FOR THE COMPONENT
      CALL GTICMP (3, ITCOMP, IEXST, IER)

C     TILE THE COMPONENT IF IT EXISTS
      IF (IEXST .NE. 0 ) THEN

C        CHECK FILE FOR SHADE FLAG
         CALL PLSSHD(COMPS(ICOMP), ISFLAG)

C        GENERATE THE DATA FOR THE POLYGONS AND LOAD THEM INTO THE
C        COMMON BLOCK - 'POLDAT'
         IF (ISFLAG .NE. 0) THEN

C           CHECK FOR SURFACE SHOULD BE RELIMITED
            CALL CHKPRT ( COMPS(ICOMP), PFLAG, NPRTS, PARTS)

            IF (PFLAG .EQ. 0) THEN
               CALL MKTIBS (COMPS(ICOMP), COMPS(ICOMP))
C              WRITE OUT B-SPLINE CONTROL HULL
               CALL HRDFIL(COMPS(ICOMP))
            ELSE
C              LOOP THROUGH ALL RELIMITED PARTS
               DO 200 IP = 1, NPRTS
```

```
                  CALL MKTIBS(COMPS(ICOMP), PARTS(IP))
C                 WRITE OUT B-SPLINE CONTROL HULL
                  CALL HRDFIL(PARTS(IP))
200            CONTINUE
            ENDIF
          ENDIF

        ENDIF

100   CONTINUE

      RETURN
      END
C=====================================================================
C                       M O D U L E   P L S S H D
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     ITCOMP    - INTEGER - COMPONENT IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C     ISFLAG    - INTEGER - SHADE INDICATOR FLAG
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     CHECK SHDIND FILE TO DETERMINE IF COMPONENT SHOULD BE SHADED
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE PLSSHD(ITCOMP, ISFLAG)

      INTEGER ITCOMP, ISFLAG, NOUT
      PARAMETER (NOUT = 66)

C     INITIALIZE SHADE FLAG
      ISFLAG = 0

C     OPEN AND REWIND FILE
      OPEN(UNIT = NOUT, FILE = 'SHDIND')
      REWIND(NOUT)

C     READ IN NUMBER OF COMPONENTS
      READ (NOUT, *, END=900) NOCOMP

C     LOOP THROUGH EACH COMPONENT AND CHECK SHADE FLAG
      DO 100 I=1,NOCOMP
         READ (NOUT,*)ISCOMP

         IF (ISCOMP .EQ. ITCOMP) ISFLAG = 1

100   CONTINUE

900   CLOSE(NOUT)

      RETURN
      END
C=====================================================================
C                       M O D U L E   C H K P R T
```

```
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C     PFLAG    - INTEGER - COMPONENT IDENTIFIER
C     NPRTS    - INTEGER - NUMBER OF RELIMITED PARTS
C     PARTS    - INTEGER ARRAY - RELIMITED COMPONENT COMPONENT NUMBERS
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     READ RELIMIT FILE AND CHECK IF COMPONENT SHOULD BE RELIMITED
C     AND READ SUB-COMPONENT NUMBERS
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE CHKPRT(COMP, PFLAG, NPRTS, PARTS)

      INTEGER COMP, PFLAG, NPRTS, PARTS(*), NOUT
      PARAMETER (NOUT = 66)

C     SET RELIMIT FLAG
      PFLAG = 0

C     OPEN AND REWIND FILE
      OPEN(UNIT = NOUT, FILE = 'RELIMIT')
      REWIND(NOUT)

C     READ NUMBER OF COMPONENTS IN FILE
      READ(NOUT, *, END=900)NOCOMP

C     LOOP THROUGH ALL COMPONENTS
      DO 100 I=1,NOCOMP

C        READ COMPONENT NUMBER
         READ(NOUT,*)ICOMP, NPARTS

C        CHECK COMPONENT NUMBER
         IF (COMP .EQ. ICOMP) THEN

C           SET NUMBER OF SUB-COMPONENTS
            NPRTS = NPARTS

C           READ IN SUB-COMPONENT NUMBERS
            DO 200 J=1,NPARTS
               READ(NOUT,*)PARTS(J)
               READ(NOUT,*)
200         CONTINUE
            PFLAG = 1
         ELSE
C           SKIP RECORDS
            DO 150 J=1,NPARTS
               READ(NOUT,*)
               READ(NOUT,*)
150         CONTINUE
         ENDIF
100   CONTINUE
```

```
900   CLOSE(NOUT)

      RETURN
      END
C=====================================================================
C                    M O D U L E   M K T I B S
C=====================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=====================================================================
C INPUT PARAMETERS:
C     COMP    - INTEGER - COMPONENT IDENTIFIER
C     PART    - INTEGER - SUB-COMPONENT IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C     NONE
C=====================================================================
C COMMON INPUTS:
C     NONE
C=====================================================================
C COMMON OUTPUTS:
C     NONE
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C     THE MODULE MKTIBS FILLS THE COMMON BLOCK 'POLDAT' WITH POLYGONAL
C DATA FOR A COMPONENT.  THE ROUTINE DISPTL CAN BE EXECUTED AFTER
C EXECUTING THIS ROUTINE TO DISPLAY THE TILED IMAGE.
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  JARAYRAM SANKAR                          DATE:  1/15/88
C MODIFIED:  J. R. GLOUDEMANS                         DATE:  3/15/90
C=====================================================================
      SUBROUTINE MKTIBS ( COMP, PART )

      INTEGER COMP, PART
      INTEGER NXSITM, NUITEM, NWITEM, NXSECT, NPPITM, NPPXS, ERRIND
      INTEGER NUPOL, NWPOL

      PARAMETER(NXSITM = 5, NPPITM = 6, NUITEM = 8, NWITEM = 9)

C     CHECK FOR COMPONENT GLOBAL SYMMETRY
      IF ( COMP .GT. 100) THEN
          ICOMP = COMP - 100
      ELSE
          ICOMP = COMP
      ENDIF

C     CHECK FOR SUB-COMPONENT GLOBAL SYMMETRY
      IF ( PART .GT. 100) THEN
          IPART = PART -100
      ELSE
          IPART = PART
      ENDIF

C     GET NUMBER OF X-SECTIONS
      CALL GTICMP(NXSITM, IPART, NXSECT, ERRIND)

C     GET NUMBER OF POINTS PER X-SECTION
      CALL GTICMP(NPPITM, IPART, NPPXS, ERRIND)

C     GET NUMBER OF SURFACE LINES IN THE U DIRECTION
      CALL GTICMP(NUITEM, ICOMP, NUPOL, ERRIND)

C     CALCULATE NUMBER OF POLYGONS IN THE U DIRECTION
      NUPOL = NUPOL - 1
```

```
C     GET NUMBER OF SURFACE LINES IN THE W DIRECTION
      CALL GTICMP(NWITEM, ICOMP, NWPOL, ERRIND)

C     CALCULATE NUMBER OF POLYGONS IN THE W DIRECTION
      NWPOL = NWPOL - 1

C     GENERATE SURFACE POLYGONS
      CALL GNTIBS(PART, NXSECT, NPPXS, NUPOL, NWPOL)

      RETURN
      END
C=======================================================================
C                      M O D U L E   G N T I B S
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP - INTEGER, COMPONENT NUMBER
C     NXSECT - INTEGER, NUMBER OF X-SECTIONS FOR A COMPONENT
C     NPPXS  - INTEGER, NUMBER OF POINTS PER X-SECTION FOR A COMPONENT
C     NUPOL  - INTEGER, NUMBER OF POLYGONAL ELEMENTS IN THE U
C              DIRECTION FOR EACH SURFACE OF THE COMPONENT
C     NWPOL  - INTEGER, NUMBER OF POLYGONAL ELEMENTS IN THE W
C              DIRECTION FOR EACH SURFACE OF THE COMPONENT
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     GENERATES SURFACE POLYGONS FOR A SPECIFIED COMPONENT
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE GNTIBS(COMP, NXSECT, NPPXS, NUPOL, NWPOL)

      INTEGER COMP, NXSECT, NPPXS, NUPOL, NWPOL
      INTEGER I, J

      REAL XYZSUB(3,100000,5)

      INTEGER NPSUB(100000), NPARTS, PRTLMT(2,100)
      COMMON / POLDAT / NPARTS, PRTLMT, NSUBT, NPSUB, XYZSUB

C     INCREMENT THE NO. OF PARTS
      NPARTS = NPARTS + 1

C     SET THE INDEX NUMBER FOR THE FIRST POLYGON OF THE NEW PART
      PRTLMT(1,NPARTS) = NSUBT + 1

C     GENERATE POLYGONAL ELEMENTS FOR (I,J)TH SURFACE
      CALL GNSFBS(COMP, NXSECT, NPPXS, NUPOL, NWPOL)

C     SET THE INDEX NUMBER FOR THE LAST POLYGON OF THE NEW PART
      PRTLMT(2,NPARTS) = NSUBT
      WRITE(*,*)'COMP, TOTAL POLYGONS =',COMP ,NSUBT

      RETURN
```

```
      END
C=======================================================================
C                        M O D U L E   G N S F B S
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C     NXSECT   - INTEGER - NUMBER OF X-SECTIONS
C     NPPXS    - INTEGER - NUMBER OF POINTS X-SECTIONS
C     TOTU     - INTEGER - TOTAL NUMBER OF U DIVISIONS
C     TOTW     - INTEGER - TOTAL NUMBER OF W DIVISIONS
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE GNSFBS(COMP, NXSECT, NPPXS, TOTU, TOTW)

      INTEGER COMP, NXSECT, NPPXS, TOTU, TOTW
      INTEGER MAXKNT
      PARAMETER (MAXKNT = 50)
      REAL UKNTS(MAXKNT), WKNTS(MAXKNT), TOL
      PARAMETER (TOL = 0)
      INTEGER NDIVU,  NDIVW
      REAL BLNU(4), DLNU(4), BLNW(4), DLNW(4)

C     LOAD KNOT ARRAYS
      CALL LOADKT(COMP, NXSECT, NPPXS, UKNTS, WKNTS)

C     LOOP THROUGH ALL U LINES
      DO 300 I=3, NXSECT+1

C        COMPUTE DIFFERENCE BETWEEN MAX AND MIN U VALUES
         UDIFF = UKNTS(I+1) - UKNTS(I) - TOL
         IF (UDIFF .GT. 0.0) THEN

         DO 200 J=3, NPPXS+1
C          COMPUTE NUMBER OF DIVISIONS
           NDIVU = TOTU
           NDIVW = TOTW

C           COMPUTE DIFFERENCE BETWEEN MAX AND MIN U VALUES
            WDIFF = WKNTS(J+1) - WKNTS(J) - TOL
            IF (WDIFF .GT. 0.0) THEN

C           COMPUTE BLENDING U BLENDING FUNCTIONS
            CALL BLENDT(UKNTS(I), UKNTS, I, BLNU)
            DO 100 K=1,NDIVU
C              COMPUTE NEXT U BLENDING FUNCTION
               CALL SMLBLN(K, NDIVU+1, UDIFF, UKNTS, I, DLNU)

C              COMPUTE BLENDING U BLENDING FUNCTIONS
               CALL BLENDT(WKNTS(J), WKNTS, J, BLNW)
```

```
               DO 50 L=1, NDIVW
C                 COMPUTE NEXT U BLENDING FUNCTION
                  CALL SMLBLN(L, NDIVW+1, WDIFF, WKNTS, J, DLNW)

C                 DRAW POLYGONS
                  CALL DRPOBS(COMP,I,I,J,J,1,BLNU,DLNU,BLNW,DLNW)

                  DO 25 M=1,4
                     BLNW(M) = DLNW(M)
25                CONTINUE
50             CONTINUE
               DO 75 M=1,4
                  BLNU(M) = DLNU(M)
75             CONTINUE
100         CONTINUE
            ENDIF
200       CONTINUE
          ENDIF
300     CONTINUE

        RETURN
        END
C=======================================================================
C                       M O D U L E   D R P O L Y
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C      U       - REAL, VALUE OF THE PARAMETER 'U'
C      DU      - REAL, INCREMENTAL VALUE OF U FOR THE SURFACE POLYGONS
C      W       - REAL, VALUE OF THE PARAMETER 'W'
C      DW      - REAL, INCREMENTAL VALUE OF W FOR THE SURFACE POLYGONS
C      NWPOL   - INTEGER, NUMBER OF POLYGONAL ELEMENTS IN THE W
C                DIRECTION FOR EACH SURFACE OF THE COMPONENT
C      BMAT(3,4,4) - REAL, SURFACE MATRIX
C=======================================================================
C OUTPUT PARAMETERS:
C      NONE
C=======================================================================
C COMMON INPUTS:
C      NONE
C=======================================================================
C COMMON OUTPUTS:
C      NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     GENERATES THE DATA FOR A POLYGON ON A SURFACE FOR A
C     GIVEN LOCATION ON THE SURFACE DETERMINED BY THE VALUES
C     OF THE PARAMETERS U AND W.  THE FIRST VERTEX OF THE POLYGON IS
C     AT THE LOCATION (U,W) ON THE SURFACE.  THE SECOND VERTEX IS
C     AT THE LOCATION (U,W+DW) ON THE SURFACE.  THE THIRD VERTEX IS
C     AT THE LOCATION (U+DU,W+DW) ON THE SURFACE.  THE FOURTH VERTEX IS
C     AT THE LOCATION (U+DU,W) ON THE SURFACE.
C=======================================================================
C METHODS/ALGORITHMS:
C      N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================
        SUBROUTINE DRPOBS(COMP,IU,DU,IW,DW,NWPOL,BLNU,DLNU,BLNW,DLNW)

        INTEGER COMP, IU, DU, IW, DW, NWPOL
        REAL BLNU(4), DLNU(4), BLNW(4), DLNW(4)

        REAL XYZSUB(3,100000,5)

        INTEGER NPSUB(100000), NPARTS, PRTLMT(2,100)
```

```
      COMMON / POLDAT / NPARTS, PRTLMT, NSUBT, NPSUB, XYZSUB

C     INCREMENT THE NUMBER OF POLYGONS IN THE COMMON BLOCK
C     NEW POLYGON IN ORDER TO ADD A
      NSUBT = NSUBT + 1

C     SET THE NUMBER OF VERTICES OF THE POLYGON TO 4
      NPSUB(NSUBT) = 4

C     GENERATE THE FIRST VERTEX OF THE POLYGON AND STORE IT IN COMMON
C     BLOCK
      CALL POLPT (COMP, IU, IW, NWPOL, BLNU, BLNW ,1)

C     GENERATE THE SECOND VERTEX OF THE POLYGON AND STORE IT IN COMMON
C     BLOCK
      CALL POLPT (COMP, DU, IW, NWPOL, DLNU, BLNW ,2)

C     GENERATE THE THIRD VERTEX OF THE POLYGON AND STORE IT IN COMMON
C     BLOCK
      CALL POLPT (COMP, DU, DW, NWPOL, DLNU, DLNW ,3)

C     GENERATE THE FOURTH VERTEX OF THE POLYGON AND STORE IT IN COMMON
C     BLOCK
      CALL POLPT (COMP, IU, DW, NWPOL, BLNU, DLNW ,4)

      RETURN
      END
C=======================================================================
C                      M O D U L E   P O L P T
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     U      - REAL, VALUE OF THE PARAMETER 'U'
C     W      - REAL, VALUE OF THE PARAMETER 'W'
C     NWPOL  - INTEGER, NUMBER OF POLYGONAL ELEMENTS IN THE W
C              DIRECTION FOR EACH SURFACE OF THE COMPONENT
C     BMAT(3,4,4) - REAL, SURFACE MATRIX
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C    CALCULATES THE COORDINATES THE FIRST VERTEX OF A POLYGON ON A
C    SURFACE
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE POLPT (COMP, IU, IW, NWPOL, BLNU, BLNW, POLYNO)
      INTEGER COMP, IU, IW, NWPOL, POLYNO
      REAL BLNU(4), BLNW(4), X, Y, Z

      REAL XYZSUB(3,100000,5)

      INTEGER NPSUB(100000), NPARTS, PRTLMT(2,100)

      COMMON / POLDAT / NPARTS, PRTLMT, NSUBT, NPSUB, XYZSUB
```

```
C     CALCULATE A POINT ON NON-UNIFORM B-SPLINE SURFACE
      CALL NONPT(COMP, IU, IW, BLNU, BLNW, X, Y, Z)

C     FILL COMMON BLOCK
      XYZSUB(1,NSUBT,POLYNO) = X
      XYZSUB(2,NSUBT,POLYNO) = Y
      XYZSUB(3,NSUBT,POLYNO) = Z

      RETURN
      END
C=======================================================================
C                      M O D U L E   H R D F I L
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     COMP     - INTEGER - COMPONENT IDENTIFIER
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     WRITES THE B-SPLINE CONTROL POINTS AND KNOT VALUES INTO A FILE
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE HRDFIL(COMP)

      INTEGER COMP, NOUT
      PARAMETER (NOUT = 77)
      OPEN(UNIT = NOUT, FILE = 'BSFILE')

C     MOVE TO END OF FILE
      DO 50 I=1,10000
         READ(NOUT,*,END=75)
50    CONTINUE

C     CHECK FOR GLOBAL SYMETRY
75    IF (COMP .GT. 100) THEN
         ICOMP = COMP -100
      ELSE
         ICOMP = COMP
      ENDIF

C     GET NUMBER OF POINTS AND X-SECTIONS
      CALL GTICMP(5, ICOMP, NXSECT, IERR)
      CALL GTICMP(6, ICOMP, NPPXS, IERR)

C     WRITE COLOR FLAG
      WRITE(NOUT,*)'COLOR'

C     CHECK FOR
      IF (COMP .EQ. 9 .OR. COMP .EQ. 11 .OR. COMP .EQ. 13) THEN
C        REVERSE CONTROL HULL OUTPUT
         CALL SWHLOD(NOUT, COMP, NXSECT, NPPXS)
```

```
      ELSE IF (COMP .GE. 100) THEN
C        REVERSE CONTROL HULL OUTPUT
         CALL SWHLOD(NOUT, COMP, NXSECT, NPPXS)

      ELSE
C        NORMAL CONTROL HULL OUTPUT
         CALL NRMLOD(NOUT, COMP, NXSECT, NPPXS)

      ENDIF

C     CLOSE FILE
      CLOSE(NOUT)

      RETURN
      END
C=======================================================================
C                        M O D U L E   N R M L O D
C=======================================================================
C  PROJECT:  NASA/AMES                  VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     NOUT    - INTEGER - FILE IDENTIFIER
C     COMP    - INTEGER - COMPONENT IDENTIFIER
C     NXSECT  - INTEGER - NUMBER OF X-SECTIONS
C     NPPXS   - INTEGER - NUMBER OF POINTS X-SECTIONS
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     WRITE OUT CONTROL POINTS IN NORMAL ORDER
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=======================================================================
      SUBROUTINE NRMLOD(NOUT, COMP, NXSECT, NPPXS)

      INTEGER NOUT, COMP, NXSECT, NPPXS, MAXPTS
      PARAMETER (MAXPTS = 50)
      REAL P(3), UKNTS(MAXPTS), WKNTS(MAXPTS)

C     LOAD KNOT VALUES
      CALL LOADKT(COMP, NXSECT, NPPXS, UKNTS, WKNTS)

C     WRITE OUT NUMBER OF U KNOT VALUES
      WRITE(NOUT,10)NXSECT+6

C     COMPUTE AND WRITE FIRST KNOT VALUE
      TEMP = UKNTS(1) - (UKNTS(2)-UKNTS(1))
      WRITE(NOUT,20)TEMP

C     WRITE OUT U KNOT VALUES
      DO 100 I=1,NXSECT+4
         WRITE(NOUT,20)UKNTS(I)
100   CONTINUE

C     COMPUTE AND WRITE LAST KNOT VALUE
      TEMP = UKNTS(NXSECT+4) + (UKNTS(NXSECT+4)-UKNTS(NXSECT+3))
      WRITE(NOUT,20)TEMP
```

```
C     WRITE OUT NUMBER OF W KNOTS
      WRITE(NOUT,10)NPPXS+6

C     COMPUTE AND WRITE FIRST KNOT VALUE
      TEMP = WKNTS(1) - (WKNTS(2)-WKNTS(1))
      WRITE(NOUT,20)TEMP

C     WRITE OUT W KNOT VALUES
      DO 200 I=1,NPPXS+4
         WRITE(NOUT,20)WKNTS(I)
200   CONTINUE

C     COMPUTE AND WRITE LAST KNOT VALUE
      TEMP = WKNTS(NPPXS+4) + (WKNTS(NPPXS+4)-WKNTS(NPPXS+3))
      WRITE(NOUT,20)TEMP

C     WRITE CONTROL POINTS
      DO 400 I=1,NXSECT+2
         DO 300 J=1,NPPXS+2
            CALL GTBTSF(I, J, 2, COMP, P, IERR)
            WRITE(NOUT,30)P(1), P(2), P(3)
300      CONTINUE
400   CONTINUE

10    FORMAT(I4)
20    FORMAT(F12.8)
30    FORMAT(F12.8,2X,F12.8,2X,F12.8)

      RETURN
      END

C=======================================================================
C                      M O D U L E   S W H L O D
C=======================================================================
C  PROJECT:  NASA/AMES                      VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     NOUT    - INTEGER - FILE IDENTIFIER
C     COMP    - INTEGER - COMPONENT IDENTIFIER
C     NXSECT  - INTEGER - NUMBER OF X-SECTIONS
C     NPPXS   - INTEGER - NUMBER OF POINTS X-SECTIONS
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C     WRITE OUT CONTROL POINTS AND REVERSE THE U ORDER OF CONTROL
C     POINTS AND KNOT VALUES
C=======================================================================
C METHODS/ALGORITHMS:
C     N/A
C=======================================================================
C CODED BY:  J. R. GLOUDEMANS                              DATE:  3/15/90
C=======================================================================
      SUBROUTINE SWHLOD(NOUT, COMP, NXSECT, NPPXS)

      INTEGER NOUT, COMP, NXSECT, NPPXS, MAXPTS
      PARAMETER (MAXPTS = 50)
      REAL P(3), UKNTS(MAXPTS), WKNTS(MAXPTS)
```

```
C     LOAD KNOT VALUES
      CALL LOADKT(COMP, NXSECT, NPPXS, UKNTS, WKNTS)

C     WRITE OUT NUMBER OF U KNOTS
      WRITE(NOUT,10)NXSECT+6

C     COMPUTE AND WRITE FIRST U KNOT
      TEMP = UKNTS(1) - (UKNTS(2)-UKNTS(1))
      WRITE(NOUT,20)TEMP

C     WRITE U KNOT VALUES
      DO 100 I=1,NXSECT+4
         WRITE(NOUT,20)UKNTS(I)
100   CONTINUE

C     COMPUTE AND WRITE LAST U VALUE
      TEMP = UKNTS(NXSECT+4) + (UKNTS(NXSECT+4)-UKNTS(NXSECT+3))
      WRITE(NOUT,20)TEMP

C     WRITE OUT NUMBER OF W KNOTS
      WRITE(NOUT,10)NPPXS+6

C     COMPUTE AND WRITE OUT FIRST W KNOT
      TEMP = 100.0 - WKNTS(NPPXS+4) - (WKNTS(NPPXS+4)-WKNTS(NPPXS+3))
      WRITE(NOUT,20)TEMP

      DO 200 I=1,NPPXS+4
         TEMP = 100.0 - WKNTS(NPPXS+5-I)
         WRITE(NOUT,20)TEMP
200   CONTINUE

C     COMPUTE AND WRITE LAST W KNOT
      TEMP = 100.0 - WKNTS(1) + (WKNTS(2)-WKNTS(1))
      WRITE(NOUT,20)TEMP

C     WRITE CONTROL POINTS
      DO 400 I=1,NXSECT+2
         DO 300 J=1,NPPXS+2
            CALL GTBTSF(I, NPPXS+3-J, 2, COMP, P, IERR)
            WRITE(NOUT,30)P(1), P(2), P(3)
300      CONTINUE
400   CONTINUE

10    FORMAT(I4)
20    FORMAT(F12.8)
30    FORMAT(F12.8,2X,F12.8,2X,F12.8)

      RETURN
      END
C=======================================================================
C                      M O D U L E   R E L M I T
C=======================================================================
C  PROJECT:  NASA/AMES                    VERSION:  V1.0.0 B-SPLINE TEST
C=======================================================================
C INPUT PARAMETERS:
C     NONE
C=======================================================================
C OUTPUT PARAMETERS:
C     NONE
C=======================================================================
C COMMON INPUTS:
C     NONE
C=======================================================================
C COMMON OUTPUTS:
C     NONE
C=======================================================================
C FUNCTIONAL DESCRIPTION:
```

```
C     READ RELIMIT INFORMATION FROM FILE AND LOAD SUB-COMPONENTS
C=====================================================================
C METHODS/ALGORITHMS:
C     N/A
C=====================================================================
C CODED BY:  J. R. GLOUDEMANS                          DATE:  3/15/90
C=====================================================================
      SUBROUTINE RELMIT

      INTEGER NOUT, NOCOMP, ICOMP, NPARTS, ITEMP
      INTEGER IULOW, IUHIGH, IWLOW, IWHIGH
      PARAMETER (NOUT = 66)

      REAL P(3)

C     OPEN AND REWIND FILE
      OPEN(UNIT = NOUT, FILE = 'RELIMIT')
      REWIND(NOUT)

C     READ IN NUMBER OF COMPONENTS
      READ(NOUT, *, END=900) NOCOMP

C     LOOP THROUGH ALL COMPONENTS
      DO 100 I=1,NOCOMP

C        READ IN COMPONENTS NUMBER AND NUMBER OF SUB-COMPONENTS
         READ(NOUT,*)ICOMP, NPARTS
         DO 200 J=1,NPARTS

C           READ RELIMIT INFORMATION
            READ(NOUT,*)ITEMP
            READ(NOUT,*)IULOW, IUHIGH, IWLOW, IWHIGH

C           PUT IN INFORMATION INTO SUB-COMPONENT
            CALL PTICMP(5, ITEMP, IUHIGH-IULOW-1, IERR)
            CALL PTICMP(6, ITEMP, IWHIGH-IWLOW-1, IERR)

C           LOAD IN UNIFORM KNOT SPACING
            CALL STRKNT(ITEMP, IUHIGH-IULOW-1, IWHIGH-IWLOW-1)

C           INITIALIZE COUNTER
            KCNT = 0
            DO 300 K=IULOW, IUHIGH
               KCNT = KCNT+1
               LCNT = 0
               DO 400 L=IWLOW, IWHIGH
                  LCNT = LCNT+1

C                 GET AND PUT CONTROL POINT
                  CALL GTBTSF(K, L, 2, ICOMP, P, IERR)
                  CALL PTBTSF(KCNT, LCNT, 2, ITEMP, P, IERR)
400            CONTINUE
300         CONTINUE
200      CONTINUE
100   CONTINUE

C     CLOSE FILE
900   CLOSE(NOUT)

      RETURN
      END
```

Appendix E

Data for Example Wing-Fuselage Fillet

ACSYNT PARAMETER DATA

Wing Data:

Aspect Ratio	3.0	
Taper Ratio	0.219	
Wing Area	188.5	feet*feet
Q. C. Sweep	32.35	degrees
Wing Dihedral	0.0	degrees
Root T/C Ratio	0.079	
Tip T/C Ratio	0.083	
Wing Span	23.78	feet
Root Chord	13.0	feet
L. E. Sweep	40.25	degrees
X - Translation	19.0	feet
Y - Translation	2.0	feet
Z - Translation	2.5	feet
X - Rotation	0.0	degrees
Y - Rotation	0.0	degrees
Z - Rotation	90.0	

Fuselage (Mid-Section) Data:

Length	20.0	feet
Front Elliptical Radius 1	4.0	feet
Front Elliptical Radius 2	4.0	feet
Rear Elliptical Radius 1	4.0	feet
Rear Elliptical Radius 1	4.0	feet
X - Translation	12.0	feet
Y - Translation	0.0	feet
Z - Translation	0.0	feet
X - Rotation	0.0	degrees

```
  Y - Rotation ..................  0.0  degrees
  Z - Rotation ..................  0.0  degrees


FILLET B-SPLINE CONTROL HULL DATA

  Number of u control points .... 13
  Number of w control points .... 31

  The Control Points:

        X             Y             Z
   3.77475119    6.38239956    2.71827698
   3.43446422    5.96624136    2.72647905
   3.08028746    5.53309631    2.73501611
   3.54776049    5.43121243    2.83185339
   2.76596475    4.76079941    2.80285287
   2.57624006    4.08400011    2.88859153
   0.14014575    3.22140980    2.54482913
  -1.85391414    2.88048506    2.76832223
  -2.83291888    2.24523377    3.42578220
  -3.41480541    1.16464567    3.92081761
  -3.93959022    1.63603950    3.78619480
  -4.00019026    1.63603950    3.78619480
  -4.05959034    1.63603950    3.78619480
        X             Y             Z
   3.26309490    6.38240004    2.50005341
   2.90318632    5.96624184    2.50005531
   2.52858734    5.53309727    2.50005722
   2.10315728    5.43288326    2.50066352
   1.61094356    4.75067234    2.49652910
   1.08781385    4.13192034    2.51685500
  -0.41059712    2.90184379    2.38822961
  -2.97862935    3.16108108    2.51107383
  -3.34965467    3.17644334    2.53491282
  -3.87988400    3.44416881    2.74643826
  -3.93959022    3.30408549    2.63490534
  -4.00019026    3.30408573    2.63490534
  -4.05959034    3.30408573    2.63490534
        X             Y             Z
   3.77472210    6.38239956    2.28154993
   3.43443370    5.96624136    2.27334118
   3.08025575    5.53309631    2.26479745
   3.54753733    5.43441629    2.16615653
   2.76703072    4.74132872    2.20772099
   2.57096028    4.17610979    2.06014156
   0.17501543    2.60713863    2.79527068
  -1.85852146    3.41796327    2.16304564
  -2.83207202    3.82517290    1.43198323
  -3.41489267    4.08313417    0.26242739
  -3.93959022    4.05532169    0.75249898
  -4.00019026    4.05532169    0.75249898
  -4.05959034    4.05532169    0.75249898
        X             Y             Z
   4.49796391    6.38239861    2.19301939
   4.18541098    5.96624041    2.18148398
   3.86010051    5.53309584    2.16947794
   4.02421618    5.43266296    2.14803696
   3.39504552    4.75198889    2.14342737
   3.16681600    4.12567091    2.08125448
   0.99195921    2.94350910    2.31389785
  -0.50253028    3.53679132    2.11266780
  -2.44215250    3.95541883    0.70896173
  -3.54980397    4.04853773    0.02219189
  -3.93959022    3.99851465   -0.45724013
```

-4.00019026	3.99851465	-0.45724016
-4.05959034	3.99851465	-0.45724016
X	Y	Z
4.81648302	6.38239861	2.17016459
4.51614475	5.96623993	2.15777040
4.20354748	5.53309536	2.14487028
4.13682842	5.43234301	2.14102483
3.60478830	4.75393248	2.12683821
3.35357475	4.11648130	2.07823515
1.22924745	3.00479102	2.23263001
-0.18800285	3.58008409	2.01359105
-2.26461029	3.98575401	0.61864787
-3.30267048	4.02410555	-0.20868762
-3.93959022	3.89971590	-0.89009595
-4.00019026	3.89971590	-0.89009595
-4.05959034	3.89971590	-0.89009595
X	Y	Z
4.81648302	6.38239861	2.17016459
4.51614475	5.96623993	2.15777040
4.20354748	5.53309536	2.14487028
4.13682842	5.43234301	2.14102483
3.60478830	4.75393248	2.12683821
3.35357475	4.11648130	2.07823515
1.22924745	3.00479102	2.23263001
-0.18800285	3.58008409	2.01359105
-2.26461029	3.98575401	0.61864787
-3.30267048	4.02410555	-0.20868762
-3.99999046	3.90756226	-0.85520530
-3.99999046	3.89977217	-0.89010870
-3.99999022	3.89164639	-0.92493540
X	Y	Z
5.09345293	6.38239765	2.15029097
4.80373621	5.96623898	2.13715005
4.50219393	5.53309441	2.12347269
4.38907385	5.43186808	2.12452364
3.89134717	4.75682306	2.10398269
3.64343190	4.10280752	2.07288432
1.57818449	3.09597254	2.11200762
0.28052339	3.64449763	1.86613703
-2.00054693	4.03088999	0.48427802
-2.93495369	3.98775315	-0.55221134
-3.35224533	3.90756226	-0.85520530
-3.35224533	3.89977217	-0.89010870
-3.35224533	3.89164639	-0.92493540
X	Y	Z
6.06336212	6.38239574	2.11255336
5.81084013	5.96623707	2.09799433
5.54801083	5.53309250	2.08284116
4.81852722	5.43276167	2.09220767
4.58634901	4.75137711	2.07245708
4.29224586	4.12856245	2.01007462
2.32104707	2.92420912	2.23968935
0.03467992	3.88507819	1.37784183
-0.27686876	3.96103191	0.38517979
-1.68326545	4.03344631	-0.04975038
-0.70450222	3.90756226	-0.85520530
-0.70450222	3.89977217	-0.89010870
-0.70450217	3.89164639	-0.92493540
X	Y	Z
8.09050560	6.38239384	2.15318203
7.91572189	5.96623516	2.14014959
7.73380423	5.53309059	2.12658525
7.69305658	5.42803812	2.10293245
7.39710760	4.78007269	2.07910562
7.18360329	3.99280524	2.10361576
6.37341976	3.82953620	1.76435077
5.95066547	3.87271261	0.97274429

6.00907373	4.04379988	0.07801699
5.99909925	3.96080852	-0.68231356
5.99999857	3.90756226	-0.85520530
5.99999857	3.89977217	-0.89010870
5.99999809	3.89164639	-0.92493540
X	Y	Z
10.11766529	6.38239098	2.28239393
10.02062035	5.96623278	2.27421689
9.91961384	5.53308773	2.26570606
10.56732750	5.43065453	2.31566048
10.20969296	4.76413774	2.27648163
10.06647587	4.06816483	2.31762981
10.48268986	3.32693434	2.00901651
11.85914135	3.83185554	1.40832806
12.29638481	3.97080278	0.37958348
13.68132210	4.03245258	-0.04917977
12.70450783	3.90756226	-0.85520530
12.70450783	3.89977217	-0.89010870
12.70450783	3.89164639	-0.92493540
X	Y	Z
11.08759212	6.38238907	2.35446239
11.02774239	5.96623087	2.34899354
10.96545029	5.53308582	2.34330153
10.99640369	5.43027401	2.34711409
10.90704060	4.76644564	2.32605243
10.70428944	4.05723953	2.37223530
11.29899216	3.39979339	2.01598716
11.60358429	3.60434556	1.87882662
14.02186012	4.03826141	0.48194847
14.93281460	3.98700190	-0.55197507
15.35225582	3.90756226	-0.85520530
15.35225582	3.89977217	-0.89010870
15.35225582	3.89164639	-0.92493540
X	Y	Z
11.36456871	6.38238907	2.37506056
11.31534100	5.96623039	2.37036562
11.26410484	5.53308582	2.36547923
11.24883080	5.43079805	2.36540985
11.19252682	4.76325703	2.35024548
10.99925232	4.07231998	2.38000202
11.61393452	3.29921794	2.13356829
12.07660103	3.54117298	2.02668262
14.28509903	3.99289823	0.61624444
15.30061817	4.02337790	-0.20844282
16.00000381	3.90756226	-0.85520530
16.00000381	3.89977217	-0.89010870
16.00000381	3.89164639	-0.92493540
X	Y	Z
11.36456871	6.38238907	2.37506056
11.31534100	5.96623039	2.37036562
11.26410484	5.53308582	2.36547923
11.24883080	5.43079805	2.36540985
11.19252682	4.76325703	2.35024548
10.99925232	4.07231998	2.38000202
11.61393452	3.29921794	2.13356829
12.07660103	3.54117298	2.02668262
14.28509903	3.99289823	0.61624444
15.30061817	4.02337790	-0.20844282
15.93960190	3.89971590	-0.89009607
16.00020218	3.89971590	-0.89009607
16.05960274	3.89971590	-0.89009601
X	Y	Z
11.68309498	6.38238907	2.39874887
11.64608192	5.96623039	2.39494419
11.60755920	5.53308535	2.39098430
11.36156464	5.43114996	2.37347889
11.40154839	4.76111317	2.36777949

11.18944073	4.08245564	2.38462305
11.82837582	3.23162150	2.21279955
12.39414787	3.49871516	2.12602830
14.46208668	3.96240973	0.70650893
15.54780865	4.04782581	0.02244188
15.93960190	3.99851465	-0.45724028
16.00020218	3.99851465	-0.45724028
16.05960274	3.99851441	-0.45724025
X	Y	Z
12.40635681	6.38238811	2.45261025
12.39707947	5.96622944	2.45082951
12.38742447	5.53308392	2.44897604
11.83893967	5.43284798	2.40738487
12.02540874	4.75078297	2.42599940
11.80501747	4.13131571	2.37544084
12.51390839	2.90575528	2.63161778
13.76750278	3.37849879	2.18467379
14.84881401	3.83241940	1.42801237
15.41323853	4.08239651	0.26283246
15.93960190	4.05532169	0.75249875
16.00020218	4.05532169	0.75249875
16.05960274	4.05532169	0.75249875
X	Y	Z
12.91802120	6.38238621	2.51008892
12.92836571	5.96622753	2.51046801
12.93913269	5.53308249	2.51086259
13.28142262	5.43185663	2.51363707
13.19340134	4.75679779	2.51062155
13.23216534	4.10284758	2.52577806
13.47341633	3.09560728	2.43688917
14.83822250	3.13547587	2.50464320
15.37546730	3.18114257	2.53609347
15.87730885	3.44369149	2.74631763
15.93960190	3.30408549	2.63490534
16.00020218	3.30408549	2.63490534
16.05960274	3.30408549	2.63490510
X	Y	Z
12.40637875	6.38238764	2.55585361
12.39710331	5.96622896	2.55795240
12.38744831	5.53308344	2.56013680
11.83829212	5.43160534	2.59351778
12.02961540	4.75833464	2.58538866
11.78533173	4.09560204	2.59786797
12.64548683	3.14393210	2.57912087
13.75011635	2.89072490	2.76379037
14.85200882	2.24335456	3.42661476
15.41290855	1.16483748	3.92073274
15.93960190	1.63603926	3.78619504
16.00020218	1.63603926	3.78619504
16.05960274	1.63603926	3.78619504
X	Y	Z
11.68312931	6.38238811	2.60280943
11.64611816	5.96622944	2.60667276
11.60759544	5.53308439	2.61069369
11.36153793	5.43299055	2.62694168
11.40201855	4.74992180	2.63405418
11.18746662	4.13539124	2.61247230
11.84188080	2.87858701	2.81380939
12.39235878	2.85099030	2.91732740
14.46241856	1.57057881	3.70962334
15.54777431	0.92261153	3.94096112
15.93960190	0.44398895	4.00001383
16.00020218	0.44398895	4.00001383
16.05960274	0.44398895	4.00001383
X	Y	Z
11.36460876	6.38238907	2.62493491
11.31538200	5.96623039	2.62962961

11.26414680	5.53308582	2.63451576
11.24895287	5.43324137	2.63460255
11.19209862	4.74839163	2.64974165
11.00154209	4.14262962	2.62008476
11.59902573	2.83031750	2.86589932
12.07856750	2.76174164	2.98498368
14.28474140	1.48970032	3.75867009
15.30065441	0.69204175	3.96858287
15.93960190	0.00000134	4.00001478
16.00020218	0.00000134	4.00001478
16.05960274	0.00000134	4.00001478
X	Y	Z
11.36460876	6.38238907	2.62493491
11.31538200	5.96623039	2.62962961
11.26414680	5.53308582	2.63451576
11.24895287	5.43324137	2.63460255
11.19209862	4.74839163	2.64974165
11.00154209	4.14262962	2.62008476
11.59902573	2.83031750	2.86589932
12.07856750	2.76174164	2.98498368
14.28474140	1.48970032	3.75867009
15.30065441	0.69204175	3.96858287
16.00000381	0.03576284	3.99990010
16.00000381	0.00000143	4.00007248
16.00000381	-0.03576000	3.99990034
X	Y	Z
11.08763599	6.38238907	2.64417410
11.02778816	5.96623039	2.64959168
10.96549797	5.53308582	2.65523052
10.99674892	5.43361712	2.65187454
10.90527248	4.74611473	2.67131448
10.71292400	4.15339947	2.63224077
11.24181080	2.75849891	2.94311905
11.61113548	2.62894940	3.08568525
14.02047539	1.36936200	3.83163977
14.93295860	0.34897846	4.00968266
15.35225582	0.03576284	3.99990010
15.35225677	0.00000143	4.00007248
15.35225677	-0.03576000	3.99990034
X	Y	Z
10.11771965	6.38239098	2.71440196
10.02067566	5.96623230	2.72245836
9.91967201	5.53308678	2.73084378
10.56781006	5.43384886	2.68131900
10.20717239	4.74470854	2.71926332
10.07874203	4.16005659	2.68256593
10.40138340	2.71411014	2.96184778
11.86988640	2.22369432	3.40053678
12.29441071	1.25403202	3.79079080
13.68151760	0.84932983	3.94188643
12.70450783	0.03576284	3.99990010
12.70450783	0.00000143	4.00007248
12.70450783	-0.03576000	3.99990034
X	Y	Z
8.09056664	6.38239384	2.84770918
7.91578579	5.96623516	2.86077499
7.73387003	5.53309059	2.87437415
7.69321060	5.43224049	2.89839101
7.39666605	4.75453138	2.91973162
7.18609715	4.11361980	2.90778065
6.35738230	3.02383733	3.16793203
5.95278692	1.75045073	3.58201432
6.00868320	0.98686033	3.92085958
5.99913979	0.21505305	4.01376867
5.99999857	0.03576284	3.99990010
5.99999905	0.00000143	4.00007248
5.99999905	-0.03576000	3.99990034

X	Y	Z
6.06341743	6.38239622	2.88800049
5.81089735	5.96623802	2.90258002
5.54806995	5.53309345	2.91775513
4.81865883	5.43355560	2.90693355
4.58594894	4.74653769	2.93536162
4.29450464	4.15143871	2.95561290
2.30650401	2.77165294	2.99269319
0.03660091	2.21608806	3.39645910
-0.27722058	1.25542748	3.79154038
-1.68322921	0.84918702	3.94180989
-0.70450234	0.03576284	3.99990010
-0.70450234	0.00000143	4.00007248
-0.70450234	-0.03576000	3.99990034
X	Y	Z
5.09349918	6.38239765	2.84980369
4.80378437	5.96623898	2.86294794
4.50224400	5.53309441	2.87662888
4.38921928	5.43355274	2.87496734
3.89082408	4.74657679	2.89988875
3.64621544	4.15127325	2.90945649
1.56006372	2.77274990	3.00667691
0.28291884	2.62706685	3.07728529
-2.00098729	1.36970818	3.83318233
-2.93490791	0.34894413	4.00952387
-3.35224509	0.03576284	3.99990010
-3.35224533	0.00000143	4.00007248
-3.35224533	-0.03576000	3.99990034
X	Y	Z
4.81652689	6.38239813	2.82983947
4.51619053	5.96623993	2.84223366
4.20359516	5.53309536	2.85513377
4.13699150	5.43312740	2.85824132
3.60413933	4.74916553	2.87774348
3.35693383	4.13902712	2.90017724
1.20725596	2.85442567	2.91145253
-0.18509635	2.75855613	2.97896361
-2.26514411	1.49028528	3.75977540
-3.30261564	0.69198209	3.96847057
-3.99999046	0.03576284	3.99990010
-3.99999046	0.00000143	4.00007248
-3.99999046	-0.03576000	3.99990034
X	Y	Z
4.81652689	6.38239813	2.82983947
4.51619053	5.96623993	2.84223366
4.20359516	5.53309536	2.85513377
4.13699150	5.43312740	2.85824132
3.60413933	4.74916553	2.87774348
3.35693383	4.13902712	2.90017724
1.20725596	2.85442567	2.91145253
-0.18509635	2.75855613	2.97896361
-2.26514411	1.49028528	3.75977540
-3.30261564	0.69198209	3.96847057
-3.93959022	0.00000149	4.00001526
-4.00019026	0.00000149	4.00001526
-4.05959034	0.00000149	4.00001526
X	Y	Z
4.49800491	6.38239861	2.80688071
4.18545341	5.96624041	2.81841230
3.86014462	5.53309584	2.83041453
4.02439117	5.43284035	2.85109711
3.39431262	4.75090742	2.86162567
3.17056251	4.13079548	2.89451957
0.96736568	2.90931940	2.84725428
-0.49927971	2.84692979	2.91290832
-2.44274974	1.57132399	3.71043444
-3.54974222	0.92253542	3.94087863

```
-3.93959022     0.44398904     4.00001383
-4.00019026     0.44398904     4.00001383
-4.05959034     0.44398904     4.00001383
      X               Y              Z
 3.77475119     6.38239956     2.71827698
 3.43446422     5.96624136     2.72647905
 3.08028746     5.53309631     2.73501611
 3.54776049     5.43121243     2.83185339
 2.76596475     4.76079941     2.80285287
 2.57624006     4.08400011     2.88859153
 0.14014575     3.22140980     2.54482913
-1.85391414     2.88048506     2.76832223
-2.83291888     2.24523377     3.42578220
-3.41480541     1.16464567     3.92081761
-3.93959022     1.63603950     3.78619480
-4.00019026     1.63603950     3.78619480
-4.05959034     1.63603950     3.78619480
      X               Y              Z
 3.26309490     6.38240004     2.50005341
 2.90318632     5.96624184     2.50005531
 2.52858734     5.53309727     2.50005722
 2.10317087     5.43288326     2.50066161
 1.61095345     4.75067186     2.49652886
 1.08782601     4.13192129     2.51685309
-0.41059071     2.90184164     2.38822722
-2.97863030     3.16108131     2.51107407
-3.34965444     3.17644286     2.53491282
-3.87988400     3.44416881     2.74643826
-3.93959022     3.30408549     2.63490534
-4.00019026     3.30408573     2.63490534
-4.05959034     3.30408573     2.63490534
      X               Y              Z
 3.77472663     6.38239956     2.28154945
 3.43443847     5.96624136     2.27334070
 3.08026052     5.53309631     2.26479697
 3.54753494     5.43441629     2.16615701
 2.76703119     4.74132872     2.20772076
 2.57096004     4.17610979     2.06014156
 0.17501552     2.60713863     2.79527068
-1.85852146     3.41796327     2.16304564
-2.83207202     3.82517290     1.43198323
-3.41489267     4.08313417     0.26242739
-3.93959022     4.05532169     0.75249898
-4.00019026     4.05532169     0.75249898
-4.05959034     4.05532169     0.75249898

U-knots:
 72.52298737 76.00955200 77.75283051 79.49610901 81.23938751
 83.73460388 86.22557068 90.35762024 94.66218567 96.89354706
 99.000000  100.000000  101.000000  102.000000  103.000000
 104.000000  105.00000000

V-knots
 -2.18704033 -1.59352112 -1.00000191  0.00000000  1.00000000
  1.59351921  1.59351921  1.59351921  1.59351921  2.47660971
  4.32017708  6.16374397  7.04683590  7.04683590  7.04683590
  7.04683590  7.64035511  8.64035511  9.64035702 10.23387623
 10.23387623 10.23387623 10.23387623 11.11696911 12.96053696
 14.80410576 15.68719673 15.68719673 15.68719673 15.68719673
 16.28071594 17.28071785 18.28071785 18.87423706 19.46775627
```

The vita has been removed from the scanned document