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.
I would like to thank Kolady, Dr. J, the Butcher, and the rest of the ACSYNT team 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

**Introduction and Literature Review** ........................................... 1  
1.1 Introduction ......................................................... 1  
1.2 Objectives and Overview ............................................. 2  
1.3 Literature Review ..................................................... 4  

**B-Splines** .............................................................. 7  
2.1 Splines ............................................................. 7  
2.2 Uniform B-spline Curves ............................................... 8  
2.3 Non-Uniform B-splines ............................................... 11  
2.4 Properties .......................................................... 13  
2.5 Uniform Cubic B-spline Inversion ................................... 16  
2.6 Non-Uniform Cubic B-spline Inversion ................................ 20  
2.7 Parameterization Techniques ....................................... 23  
2.8 Bi-cubic B-spline Surfaces ......................................... 24  
2.9 Surface Inversion ..................................................... 27  

**Curve Filleting** ........................................................... 29  

*Table of Contents*
Data for Example Wing-Fuselage Fillet ......................................................... 155

Vita ........................................................................................................ 163

Table of Contents vi
# List of Illustrations

Figure 1. Aircraft Component Filleting and B-spline Conversion Process .............. 3
Figure 2. B-spline Curve with Control Vertices ........................................... 9
Figure 3. Uniform Cubic B-spline Blending Functions .................................. 10
Figure 4. Cubic B-spline Curve ..................................................................... 14
Figure 5. Effect of Knot Spacing on a Cubic B-spline Curve ......................... 15
Figure 6. Local Shape Control of a B-spline Curve ..................................... 17
Figure 7. Open and Closed B-spline Curves ................................................. 19
Figure 8. B-spline Surface with Control Polyhedron ..................................... 25
Figure 9. Simple Control Vertex Curve Filleting ............................................ 31
Figure 10. Curve Matching Problem Associated with Re-inversion ................. 33
Figure 11. Effect of Knot Insertion on the Control Hull ............................... 35
Figure 12. Knot Insertion Curve Filleting - Step 1 ....................................... 37
Figure 13. Knot Insertion Curve Filleting - Step 2 ....................................... 38
Figure 14. Knot Insertion Curve Filleting - Step 3 ....................................... 39
Figure 15. Filleting Using Fixed Control Vertex Inversion ........................... 41
Figure 16. One-dimensional Surface Fillet .................................................... 46
Figure 17. General One-dimensional Filleting - Step 1 .................................. 49
Figure 18. General One-dimensional Filleting - Step 2 .................................. 50
Figure 19. General One-dimensional Filleting - Step 3 .................................. 51
Figure 20. Parametric Corner Filleting - Step 1 .................................. 56
Figure 21. Parametric Corner Filleting - Step 2 .................................. 57
Figure 22. Parametric Corner Filleting - Step 3 .................................. 58
Figure 23. Example Wing-Fuselage Fillet - Shaded Image (View 1) .......... 64
Figure 24. Example Wing-Fuselage Fillet - Shaded Image (View 2) .......... 65
Figure 25. Filleted Aircraft - Shaded Image (View 1) .......................... 66
Figure 26. Filleted Aircraft - Shaded Image (View 2) .......................... 67
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. Schoenberg 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.

$$p_i(u) = UMQ_i$$

where:
Figure 2. B-spline Curve with Control Vertices
Figure 3. Uniform Cubic B-spline Blending Functions
p(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 \(i = 1, 2, \ldots, n\)
U is the parametric variable matrix
M is the blending function matrix
Q_i is the control vertex matrix

\[
U = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}, \quad M = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}, \quad Q_i = \begin{bmatrix} q_{i-1} \\ q_i \\ q_{i+1} \\ q_{i+2} \end{bmatrix}
\]

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.

\[ p(u) = \sum_{i=0}^{n} q_i N_{i,M}(u) \]

where:

- \( p(u) \) is a point on the B-spline curve
- \( n+1 \) is the number of control vertices
- \( q_i \) are the control vertices
- \( N_{i,M} \) are the blending functions

The blending functions are defined recursively as shown below.

\[ N_{i,M}(u) = \begin{cases} 1 & (t_i \leq u < t_{i+1}) \\ 0 & (outside \ above \ range) \end{cases} \]

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

where:

- \( t_i \) is the knot value
- \( u \) is the parametric variable \( (t_i \leq u < t_{n-1}) \)
- \( M \) 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)

- **C² 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
  \[ p_i \quad (i = 1, 2, ..., n) \]

- **Find:** Control vertices
  \[ q_i \quad (i = 0, 1, ..., n, n+1) \]

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
\[
\frac{1}{6} q_{-1} + \frac{2}{3} q_i + \frac{1}{6} q_{i+1} = p_i \quad (i = 1, 2, ..., n)
\]

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:

\[ q_0 = q_1 \quad and \quad q_{n+1} = q_n \]

End conditions for closed curves:

\[ q_0 = q_n \quad and \quad q_{n+1} = q_1 \]

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
\( q_i = p_i \quad (i = 1, 2, \ldots, n) \)

Step 2: Find the difference between the \( k \)-th and \((k-1)\)-st iterations of \( q_i \) and calculate a new value for \( q_i \).

\[
\delta_i^k = p_i + \frac{1}{2} \left\{ p_i - q_i^{k-1} + \frac{1}{2} \left( q_i^k + q_i^{k-1} \right) \right\} \\
q_i^k = \delta_i^k + q_i^{k-1}
\]

Step 3: If the maximum difference \((\max \delta_i)\) 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

\[ p_i \quad (i = 1, 2, ..., n) \]
\[ k_i \quad (i = -1, 0, ..., n, n+1, n+2) \]

Find: Control vertices

\[ q_i \quad (i = 0, 1, ..., n, n+1) \]

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

\[
p(u) = \sum_{i=0}^{n} q_i N_{i,4}(u)
\]

\[
p(u) = N_{0,4}(t_i)q_{i-1} + N_{1,4}(t_i)q_i + N_{2,4}(t_i)q_{i+1} + N_{3,4}(t_i)q_{i+2}
\]

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.

\[ N_{0,4}(t)q_{l-1} + N_{1,4}(t)q_l + N_{2,4}(t)q_{l+1} = p_l \quad (i = 1, 2, ..., n) \]

where:

- \( N_{0,4}(t) \), \( N_{1,4}(t) \), \( N_{2,4}(t) \) are the blending functions
- \( q_i \) 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.

\[ \delta_i^k = q_i^k - q_i^{k-1} \]

\[ p_i = N_{0,4}(t_i)q_{i-1}^k + N_{1,4}(t_i)q_i^k - N_{2,4}(t_i)q_{i+1}^{k-1} \]

Solving the second equation for \( q_i^k \) and back-substituting into the first equation yields:

\[ \delta_i^k = -q_i^{k-1} + \frac{1}{N_{1,4}(t_i)} \left[ p_i - N_{0,4}(t_i)q_{i-1}^k - N_{2,4}(t_i)q_{i+1}^{k-1} \right] \]

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.

\[ p_{i,j}(u, w) = U M Q M^T W^T \]

where:
- \( p_{i,j} \) - point on the B-spline surface
- \( U \) - parametric variable matrix
- \( M \) - blending function matrix
- \( Q \) - control vertex matrix
- \( W \) - parametric variable matrix

\[
U = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \quad W = \begin{bmatrix} w^3 & w^2 & w & 1 \end{bmatrix}
\]

\[
M = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \quad Q = \begin{bmatrix} q_{i-1,j-1} & q_{i-1,j} & q_{i-1,j+1} & q_{i-1,j+2} \\ q_{i,j-1} & q_{i,j} & q_{i,j+1} & q_{i,j+2} \\ q_{i+1,j-1} & q_{i+1,j} & q_{i+1,j+1} & q_{i+1,j+2} \\ q_{i+2,j-1} & q_{i+2,j} & q_{i+2,j+1} & q_{i+2,j+2} \end{bmatrix}
\]
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.

\[ p_{i,j}(u,w) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,M}(u)N_{j,L}(w)Q_{i,j} \]

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

\[ p_{i,j}(u,w) = \begin{bmatrix} N_{0,4}(u) & N_{1,4}(u) & N_{2,4}(u) & N_{3,4}(u) \end{bmatrix} \begin{bmatrix} q_{i-1,j-1} & q_{i-1,j} & q_{i-1,j+1} & q_{i-1,j+2} \\ q_{i,j-1} & q_{i,j} & q_{i,j+1} & q_{i,j+2} \\ q_{i+1,j-1} & q_{i+1,j} & q_{i+1,j+1} & q_{i+1,j+2} \\ q_{i+2,j-1} & q_{i+2,j} & q_{i+2,j+1} & q_{i+2,j+2} \end{bmatrix} \begin{bmatrix} N_{0,4}(w) \\ N_{1,4}(w) \\ N_{2,4}(w) \\ N_{3,4}(w) \end{bmatrix} \]

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

\[
\begin{bmatrix}
  p(1,1) & \ldots & p(1,5) \\
  \vdots & \ddots & \vdots \\
  p(5,1) & \ldots & p(5,5)
\end{bmatrix}
\]

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.
Step 2: Invert the intermediate control points in the w parametric direction to find the control hull for the surface.
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 ($q_{m-1}, q_m, q_{m+1}$) are the only contributors to position, slope, and curvature at any arbitrary curve segment breakpoint.

$$
\begin{align*}
    p_{\nu}(0) &= -\frac{1}{2} q_{m-1} + \frac{1}{2} q_{m+1} \\
    p_{\nu}(0) &= -\frac{1}{2} q_{m-1} + \frac{1}{2} q_{m+1} \\
    p_{\nu\nu}(0) &= q_{m-1} - 2q_m + q_{m+1}
\end{align*}
$$

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:

\[ q_{\text{new}} = (1 - \alpha_j)q_{j-1} + \alpha_j q_j \]

where:

- \( q_i \) are the old control vertices
- \( q_{\text{new}} \) are the new control vertices
- \( \alpha_j = 1 \) if \( j \leq i - 2 \)
- \( \alpha_j = \frac{t_{\text{new}} - t_{j-1}}{t_{j+2} - t_{j-1}} \) if \( i - 1 \leq j \leq i + 1 \)
- \( \alpha_j = 0 \) 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 - \( p_i \) (\( i = 1, 2, ..., n \))

**Fixed Control Vertex Inversion Input:**

- End condition control vertices - \( q_i \) (\( i = 0, 1, 2 \)) and (\( i = n-1, n, n+1 \))
- Points to be interpolated - \( p_i \) (\( i = 3, 4, ..., n-3, n-2 \))

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:

\[
\begin{bmatrix}
q(1,1) & \ldots & q(1,5) \\
\vdots & \ddots & \vdots \\
q(5,1) & \ldots & q(5,5)
\end{bmatrix}
\]

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 \( \alpha \) values for the new knot. (See Section 3.3 which explains curve knot insertion)

\[
\alpha_j = 1 \quad \text{if } j \leq 1
\]

\[
\alpha_j = \frac{t_{\text{new}} - t_{j-1}}{t_{j+2} - t_{j-1}} \quad \text{if } 2 \leq j \leq 4
\]

\[
\alpha_j = 0 \quad \text{if } j \geq 5
\]

\[
\alpha_1 = 1
\]

\[
\alpha_2 = \frac{3.5 - 1}{4 - 1} = 0.833
\]

\[
\alpha_3 = \frac{3.5 - 2}{5 - 2} = 0.500
\]

\[
\alpha_4 = \frac{3.5 - 3}{6 - 3} = 0.167
\]

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

\[
q(1,1)_{\text{new}} = (1 - 1)q(0,1) + (1)q(1,1) = q(1,1)
\]

\[
q(2,1)_{\text{new}} = (1 - 0.833)q(1,1) + (0.833)q(2,1)
\]

\[
q(3,1)_{\text{new}} = (1 - 0.500)q(2,1) + (0.500)q(3,1)
\]

\[
q(4,1)_{\text{new}} = (1 - 0.167)q(3,1) + (0.833)q(4,1)
\]

\[
q(5,1)_{\text{new}} = (1 - 0)q(4,1) + (0)q(5,1) = q(4,1)
\]

\[
q(6,1)_{\text{new}} = (1 - 0)q(5,1) + (0)q(6,1) = q(5,1)
\]

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:

\[
\begin{bmatrix}
q(1,1)_{\text{new}} & \cdots & q(1,5)_{\text{new}} \\
\vdots & \ddots & \vdots \\
q(6,1)_{\text{new}} & \cdots & q(6,5)_{\text{new}}
\end{bmatrix}
\]

### 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 portion 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, 5, 6, 7
Control Vertices - q₁, q₂, q₃, q₄, q₅, q₆, q₇

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

\[ p(4.000) = 0.067q₁ + 0.489q₂ + 0.444q₃ \]
\[ p(4.250) = 0.008q₁ + 0.186q₂ + 0.681q₃ + 0.125q₄ \]
\[ p(4.499) = q₅ \]
\[ p(4.500) = q₆ \]
\[ p(4.750) = 0.125q₆ + 0.681q₇ + 0.186q₈ + 0.008q₉ \]
\[ p(5.000) = 0.444q₉ + 0.489q₁₀ + 0.067q₉ \]

where:
\[ p(u) \] is a point on the B-spline curve

Notice how the shape of the curve is dictated by the control vertices \( q_1, q_2, q_3, \) and \( q_5 \) when the parameter value \( u \) is less than 4.50. When \( u \) is greater or equal to 4.50 then the next set of control vertices \( q_6, q_7, q_8, \) and \( q_9 \) 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 cannot 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 to 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 delimited.
Chapter 6

References


SUBROUTINE FILLET(COMP)

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

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

C CHECK IF DATA FOR COMPONENT IS AVAILABLE
IF (IAVAIL .EQ. 1) THEN
COMPUTE MID-POINTS, INVERT IN U DIRECTION
CALL MIDPTS(COMP)

ADD KNOTS TO FUSELAGE AND PUT CONTROL VERTICES INTO COMP
CALL AFUSKICOMP

ADD KNOTS TO MING AND PUT CONTROL VERTICES INTO COMP
CALL ANINGKICOMP

INVERT IN M DIRECTION
CALL WINVTCOMP

ENDIF
RETURN
END

C MODULE GETUMS
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:
C NOUNS - INTEGER - NUMBER OF U AND M PAIRS
C IFUS - INTEGER - FUSELAGE COMPONENT NUMBER
C IFUS - INTEGER - NUMBER OF POINTS ON FUS
C MFUSFLG - INTEGER - FLAGS CONTROLLING FUS FILLET
C MING - INTEGER - MING COMPONENT NUMBER
C MFUSFLG - INTEGER - NUMBER OF POINTS ON MING
C MFUSFLG - INTEGER - FLAGS CONTROLLING MING FILLET
C UORN - INTEGER - FLAGS INDICATING U OR M DIRECTION
C DUMCMP - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C FUNS - REAL - U AND M VALUES FOR FUS BLEND
C NUWS - REAL - U AND M VALUES FOR MING BLEND
C FINT - REAL - U AND M INTERMEDIATE VALUES FOR FUS BLEND
C NINT - REAL - U AND M INTERMEDIATE VALUES FOR MING 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 GETUMS(COMP, IAVAIL)

INTEGER COMP, IAVAIL

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

INTEGER NOUNS, IFUS, IFUS, NFUSIN, MINGIN, UORN(2, 5), DUMCMP(2)
INTEGER MFUSFLG(MAXPTS), MFUSFLG(MAXPTS)
REAL FUNS(2, MAXPTS), NUWS(2, MAXPTS)
REAL FINT(2, MAXPTS), NINT(2, MAXPTS)

COMMON /FILVAL/NOUNS, IFUS, IFUS, NFUSIN, MINGIN, MFUSFLG, MFUSFLG,
                   UORN, DUMCMP, FUNS, NUWS, FINT, NINT
C INITIALIZE AVAIL FLAG
IAVAIL = 0
C OPEN AND RENIND FILE
OPENUNIT = NOUT, FILE = 'FILVAL')
RENNIND(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 UM POINTS
READ(NOUT,*)FILCMP, NODUM
C CHECK IF CURRENT COMP IS A FILLET
IF (FILCMP .EQ. COMP) THEN
C SET FLAG AND READ UM INFORMATION
IAVAIL = 1
NOUNS = NODUM
READ(NOUT,*)IFUS, IMING, NFUSIN, NMING IN
DO 50 J=1,2
READ(NOUT,*)UORM(J,1), UORM(J,2), UORM(J,3), UORM(J,4), UORM(J,5)
50 CONTINUE
READ(NOUT,*)DUMCMP(1), DUMCMP(2)
DO 100 J=1, NOUNS
READ(NOUT,*)FUSFLG(J), MNGFLG(J), FUNS(1,J), FUNS(2,J), > MUNS(1,J), MUNS(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 MODULE MIDPTS
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 NOUNS - INTEGER - NUMBER OF U AND W PAIRS
C IFUS - INTEGER - FUSELAGE COMPONENT NUMBER
C IMING - INTEGER - WING COMPONENT NUMBER
C NFUSIN - INTEGER - NUMBER OF POINTS ON FUS
C NMING IN - INTEGER - NUMBER OF POINTS ON WING
C FUSFLG - INTEGER - FLAGS CONTROLLING FUS FILLET
C MNGFLG - INTEGER - FLAGS CONTROLLING WING FILLET

Program FILLET
C CORN — INTEGER — FLAGS INDICATING U OR H DIRECTION
C DUMCMP — INTEGER — COMPONENT NUMBERS FOR DUMMY COMPS
C FUWS — REAL — U AND N VALUES FOR FUS BLEND
C NUNS — REAL — U AND H VALUES FOR NING BLEND
C FINT — REAL — U AND N INTERMEDIATE VALUES FOR FUS BLEND
C NINT — REAL — U AND H INTERMEDIATE VALUES FOR NING 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 NOUNS, IFUS, IMING, NFUSIN, NNNGIN, UORN(2, 5), DUMCMP(2)
INTEGER FUSFLG(MAXPTS), NNGFLG(MAXPTS)
REAL FUNS(2, MAXPTS), HUNS(2, MAXPTS)
REAL FINT(2, MAXPTS), NINT(2, MAXPTS)

COMMON /FILVAL/NOUNS, IFUS, IMING, NFUSIN, NNNGIN, FUSFLG, NNGFLG,
> UORN, DUMCMP, FUNS, HUNS, FINT, NINT

C LOAD FUSELAGE INTERMEDIATE POINTS INTO COMP DATABASE
DO 300 I=1, NOUNS
DO 100 J=1, NFUSIN
C FIND INTERMEDIATE U AND W VALUES ON FUS
C FIND POINTS ON FUS
CALL FNDPNTIFUS: U: W: P)
C ADD POINTS TO COMPONENT DATABASE
CALL PTBTSF(I: J: 1: COMP: P: IERR)
100 CONTINUE
DO 200 J=NNNGIN: 1: -1
C FIND INTERMEDIATE U AND H VALUES ON NING
C FIND POINTS ON HINGS
CALL FNDPNT(INING: U: H: P)
C GUESS AN INTERSECTION POINT?
IF (UORN(+2:5) .EQ. 2) THEN
CALL PTBTSF(I: NNNGIN-J+NFUSIN+2: 1: COMP: P: IERR)
ELSE
CALL PTBTSF(I: NNNGIN-J+NFUSIN+1: 1: COMP: P: IERR)
ENDIF
200 CONTINUE
C PROJECT LINES AND GUESS AN INTERSECTION POINT
IF (UORN(+2:5) .EQ. 2) THEN
CALL CALFIL(I: NFUSIN, NNNGIN, COMP, P)
ENDIF

Program FILLET
CALL PTBTSF(i, NFUSIN+1, 1, COMP, P, IERR)
ENDIF

300 CONTINUE
C
ADD NOUNS TO DATABASE
CALL PTICMP(5, COMP, NOUNS, IERR)

C
ADD NFUSIN+NNGIN TO DATABASE
IF (UORN(2,5) .EQ. Z) THEN
  CALL PTICMP(6, COMP, NFUSIN+NNGIN+1, IERR)
ELSE
  CALL PTICMP(6, COMP, NFUSIN+NNGIN, 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 MODULE CALFIL
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 NNGIN - INTEGER - NUMBER OF HING 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 METHODS/ALGORITHMS:
C N/A
C=======================================================================
C CODED BY: J. R. GLOUDEMANS DATE: 3/15/90
C=======================================================================

SUBROUTINE CALFIL(INT, NFUSIN, NNGIN, COMP, P)

INTEGER INT, NFUSIN, NNGIN, COMP
REAL P(S), FUSP(S), NNGP(S)
REAL BFUSP(S), BNGP(S), MIDP(S)
REAL FVEC(S), NVEC(S), TVEC(S), DIST, SCALE
REAL LONBD, 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)

Program FILLET
CALL GTBTSF(INTE, NFUSIN, 1, COMP, FUSP, IERR)

C GET MING POINTS FROM DATABASE
CALL GTBTSF(INTE, NFUSIN+2, 1, COMP, NNGP, IERR)
CALL GTBTSF(INTE, NFUSIN+3, 1, COMP, BNNGP, IERR)

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

C CALCULATE DIRECTION VECTORS
CALL SUB(FUSP, BFUSP, FVEC)
CALL VNORM(FVEC, VAL)
CALL SUB(NNGP, BNNGP, NVEC)
CALL VNORM(NVEC, VAL)
CALL ADD(FVEC, NVEC, TVEC)

C CALCULATE ANGLE BETWEEN DIRECTION VECTOR
CALL CALAN(FVEC, MVEC, THETA)
CALL PIDIS(FUSP, NNGP, DIST)
SCALE = 3.0
HIBD = 6.0
LOMBD = 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, NNGP, MVEC)
CALL CALAN(FVEC, MVEC, FHE)

IRFLAG = 0
C CHECK TOLERANCES
IF (FHE .LE. THETA-TOL) THEN
   HIBD = SCALE
   SCALE = (LOMBD+SCALE)/2.0
   IRFLAG = 1
ELSE IF (FHE .GE. THETA+TOL) THEN
   LOMBD = SCALE
   SCALE = (HIBD+SCALR)/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 MODULE CALANG
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

Program FILLET
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

SUBROUTINE FNDPNT(COMP, U, N, P)
INTEGER COMP
REAL U, N, P(3)

Program FILLET
INTEGER UINT, MINT
REAL UREM, WREM, B(3,4,4), PX, PY, PZ

C FIND THE INTERVAL OF U AND M
UINT = INT(U)
MINT = INT(M)

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

C LOAD CONTROL POINTS
CALL LOADB(COMP, UINT, MINT, 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 MODU LE LOAD 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 MINT - INTEGER - M 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. GLOUDEMAN S DATE: 3/15/90
C==========================================

SUBROUTINE LOADB(COMP, UINT, MINT, B)
INTEGER COMP, UINT, MINT
REAL B(3,4,4)

C LOOP THROUGH FOUR TIMES IN U AND M 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, MINT+J-3, 2, COMP, B(I,J,1), IERR)
   50 CONTINUE
100 CONTINUE
RETURN
END

C==========================================
Program FILLET
SUBROUTINE NINTUN1NOPTS(INOPTS, ICNT, NUM, U, N)
INTEGER NOPTS, ICNT
REAL NUM(2), U(2), N
REAL UDIST, NDIST; SCALE

C CALCULATE U AND N DISTANCES IN PARAMETER SPACE
UDIST = U(1) - NUM(1)
NDIST = U(2) - NUM(2)

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

C CALCULATE U AND N VALUES
U = NUM(1) + SCALE*UDIST
N = NUM(2) + SCALE*NDIST

RETURN
END

SUBROUTINE UKNTSP(COMP, DCMP)
INTEGER COMP, DCMP
INTEGER ICLOSE, NSECT, MAXPTS
PARAMETER(ICLOSE = 0, MAXPTS = 50)
REAL PTS(2, MAXPTS); KNTS(MAXPTS)
C GET NUMBER OF POINTS IN THE U DIRECTION
CALL GTICMP5, COMP, NXSECT, IERR)

C LOOP THROUGH ALL POINTS
DO 100 I=1, NXSECT
C RETRIEVE POINTS ON SURFACE FROM THE DATA STRUCTURE
CALL GTBTSFI, 1, 1, COMP, PTS(1, I), IERR)
100 CONTINUE
CALL CENTRPI, NXSECT, PTS, 1.0, ICLOSE, UKNTS)

DO 200 I=1, NSECT+4
C PUT THE KNOT VALUES INTO THE DATA STRUCTURE
CALL PTKNOT(I, 1, COMP, UKNTS(I), IERR)
CALL PTKNOT(I, 1, DCM, UKNTS(I), IERR)
200 CONTINUE
RETURN
END

C MODULE UINVRT

C PROJECT: NASA/AMES VERSION: V1.0.0 B-SPLINE TEST

C INPUT PARAMETERS:
C COMP - INTEGER - COMPONENT IDENTIFIER
C OUTPUT PARAMETERS:
C NONE
C COMMON INPUTS:
C NONE
C COMMON OUTPUTS:
C NONE

C FUNCTIONAL DESCRIPTION:
C INVERT THE FILLET SURFACE IN THE U DIRECTION
C METHODS/ALGORITHMS:
C N/A

C CODED BY:  J. R. GLOUDEMANS DATE: 3/15/90

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 GTICMP5, COMP, NXSECT, IERR)
CALL GTICMP6, COMP, NPPXS, IERR)

C SET FLAGS FOR CLOSED INVERSION
DO 100 I = 2, NXSECT+1
    UFLAG(I) = 4
100 CONTINUE
CONTINUE
UFLAG(1) = 2
UFLAG(NXSECT+2) = 2

LOAD U KNOTS
DO 200 J=1, NXSECT+4
   CALL GTKNOT(J, 1, COMP, UKNTS(J), IERR)
200 CONTINUE

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

LOOP THROUGH ALL NPPXS
DO 500 I=1, NPPXS

LOAD POINTS IN U DIRECTION
DO 300 J=1, NXSECT
   CALL GTBTSF(J, I, 1, COMP, PTS(I, J+1), IERR)
300 CONTINUE

NON—UNIFORM INVERSION
CALL NNVEPT(NKSECT+2, PTS, UFLAG, UBLN, OUTPTS)

LOAD CONTROL VERTICES INTO DATABASE
DO 400 J=1, NXSECT+2
   CALL PTBTSF(J, I+1, 2, COMP, OUTPTS(J, I), IERR)
400 CONTINUE
500 CONTINUE
RETURN
END

C=============:===========2=====:==========:2=======================:2
C M O D U L E A D D C O R
C22::::*;::::2::==================:====:==::2::===:====================
C PROJECT: NASA/AMES VERSION: V1.0.0 B—$PLINE TEST
C=====::2::::::2::2:::::2:::=:2::::=======•'====2=:===:=================
C INPUT PARAMETERS:
C COMP — INTEGER - COMPONENT IDENTIFIER
C OUTPUT PARAMETERS:
C NONE
C COMMON INPUTS:
C NONE
C COMMON OUTPUTS: FILVAL
C NOUNS - INTEGER - NUMBER OF U AND M PAIRS
C IFUM - INTEGER - FUSELAGE COMPONENT NUMBER
C IMING - INTEGER - WING COMPONENT NUMBER
C NFUSIN - INTEGER - NUMBER OF POINTS ON FUS
C NINGIN - INTEGER - NUMBER OF POINTS ON WING
C FUSFLG - INTEGER - FLAGS CONTROLLING FUS FILLET
C MNGFLG - INTEGER - FLAGS CONTROLLING WING FILLET
C UORN - INTEGER - FLAGS INDICATING U OR M DIRECTION
C DUMCMP - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C FUNS · REAL · U AND M VALUES FOR FUS BLEND
C NUMS - REAL - U AND M VALUES FOR WING BLEND
C FINT - REAL - U AND M INTERMEDIATE VALUES FOR FUS BLEND
C MINT - REAL - U AND M INTERMEDIATE VALUES FOR WING BLEND
C==================================================:2::==:2::2:::2::
Program FILLET

83
SUBROUTINE ADDCOR(COMP)

INTEGER COMP

INTEGER MAXPTS
PARAMETER (MAXPTS = 50)
REAL KNOT

INTEGER NOUNS, IFUS, IMING, NFUSIN, NNGIN, UORM(2, 5), DUMCMP(2)
INTEGER FUSFLG(MAXPTS), NNGFLG(MAXPTS)
REAL FUNS(2, MAXPTS), UUNS(2, MAXPTS)
REAL FINTZ, NINT(2, MAXPTS)

COMMON /FILVAL/NOUNS, IFUS, IMING, NFUSIN, NNGIN, FUSFLG, NNGFLG, > UORM, DUMCMP, FUNS, UUNS, FINT, NINT

C LOOP THROUGH THE POINTS ON THE FILLET SURFACE
DO 200 I = 3, NOUNS + 2
C CHECK IF THE POINT IS A CORNER POINT
IF (NNGFLG(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, NNGECSCT, IERR)
C CHECK IF A INTERSECTION POINT WAS GUESSED
IF (UORM(2, 5) .EQ. 2) THEN
C SHIFT CONTROL VERTICES TO ALLOW FUS KNOTS
CALL SHIFTV(COMP, NFUSIN NNGIN+1, DUMCMP(1))
ELSE
C SHIFT CONTROL VERTICES TO ALLOW FUS KNOTS
CALL SHIFTV(COMP, NFUSIN NNGIN, DUMCMP(1))
ENDIF

RETURN
END

C==============================================
C MODULE SHIFT V
C==============================================
C PROJECT: NASA/AMES VERSION: V1.0.0 B-SPLINE TEST
C==============================================
C INPUT PARAMETERS:
C COMP - INTEGER - COMPONENT IDENTIFIER
C NPGTS - INTEGER - NUMBER OF POINTS
C DCMP - INTEGER - DUMMY COMPONENT IDENTIFIER
C==============================================
C OUTPUT PARAMETERS:
C NONE
C==============================================
C COMMON INPUTS:
C NONE
COMMON OUTPUTS:
C NONE

FUNCTIONAL DESCRIPTION:
C SHIFT THE CONTROL VERTICES IN THE DATA STRUCTURE
C TO ALLOW FOR THE FUSELAGE BLENDING CONTROL POINTS

METHODS/ALGORITHMS:
C N/A

CODED BY: J. R. GLOUDEMANS DATE: 3/15/90

SUBROUTINE SHIFTV(COMP, NOPTS, DCMP)

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

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

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

MODULE AFUSK

PROJECT: NASA/AMES VERSION: V1.0.0 B-SPLINE TEST

INPUT PARAMETERS:
C COMP - INTEGER - COMPONENT IDENTIFIER

OUTPUT PARAMETERS:
C NONE

COMMON OUTPUTS:
C NONE

COMMON INPUTS: FILVAL
C NUNMS - INTEGER - NUMBER OF U AND W PAIRS
C IFUM - INTEGER - FUSELAGE COMPONENT NUMBER
C IHING - INTEGER - WING COMPONENT NUMBER
C NFUSIN - INTEGER - NUMBER OF POINTS ON FUS
C NHNSIN - INTEGER - NUMBER OF POINTS ON WING
C FUSFLG - INTEGER - FLAGS CONTROLLING FUS FILLET
C WNGFLG - INTEGER - FLAGS CONTROLLING WING FILLET
C UORN - INTEGER - FLAGS INDICATING U OR W DIRECTION
C DUMCMP - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C FUMS - REAL - U AND W VALUES FOR FUS BLEND
C WUMS - 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

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 AFUSKICOMP

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

INTEGER NOUNS, IFUS, IMING, NFUSIN, NMNGIN, UORM(2, 5), DUMCMP(2)
INTEGER FUSFLG(MAXPTS), NNGFLG(MAXPTS)
REAL FUNS(2, MAXPTS), NUNS(2, MAXPTS)
REAL FINT(2, MAXPTS), NINT(2, MAXPTS)

COMMON /FILVAL/NOUNS, IFUS, IMING, NFUSIN, NMNGIN, FUSFLG, NNGFLG,
> UORM, DUMCMP, FUNS, NUNS, FINT, NINT

C INITIALIZE THE COUNTERS AND FLAGS
CALL PTICMPIS, DUMCMPIZ), 1, IERR)
ISIDE = 1
IBEG = 1
ICNT = 1
ISTART = IBEG

C CHECK IF BEGIN COUNTER TOO SMALL
100 IF (IBEG .LE. 0) IBEG = NOUNS-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. NOUNS) THEN
IFIN = 1
IEND = NOUNS
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. NOUNS) IEND = IFIN
ICURR = IBEG - 1
DO 300 I = 1, ICNT
ICURR = ICURR+1
IF (ICURR .GE. NOUNS) ICURR = 1
KLINE(I) = ICURR
300 CONTINUE

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

Program FILLET
IBEG = IFIN
ISIDE = ISIDE + 1

C CHECK FOR END
IF (IEND .NE. NOUNS .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:::2:::2:::::=======:==================2::===========================

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(NXSECT, 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:::2:::2:::::=======:==================2::===========================

C PROJECT: NASA/AMES                     VERSION: V1.0.0 B-SPLINE TEST
C=======================================================================

Program FILLET
Subroutine CONADD(COMP, ISIDE, INDUM, ISTART, IEND, ICNT, KLINE)

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

INTEGER MAXUMS
PARAMETER (MAXUMS = 30)
REAL UM(MAXUMS)

C FIND ADJUSTED U OR N VALUES TO ADD
CALL ADJUM(COMP, INDUM, ICNT, KLINE, UM)

C ADD KNOTS TO DUMMY COMPONENT
CALL ADDKNT(ISIDE, INDUM, ICNT, KLINE, UM)

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

RETURN
END

Module ADJUM

PROJECT: NASA/AMES
VERSION: V1.0.0 B-SPLINE TEST

C INPUT PARAMETERS:
C COMP - INTEGER - COMPONENT IDENTIFIER
C INDUM - INTEGER - U OR N DIRECTION INDICATOR
C KLINE - INT ARRAY - INDEXES TO CONTROL HULL LOCATIONS

C OUTPUT PARAMETERS:
C UM - REAL ARRAY - ADJUSTED U OR N VALUES

C COMMON OUTPUTS:
C NONE

C COMMON INPUTS: FILVAL
C NOUNS - INTEGER - NUMBER OF U AND N PAIRS
C IFUM - INTEGER - FUSELAGE COMPONENT NUMBER
C INING - INTEGER - MING COMPONENT NUMBER
C NFUSIN - INTEGER - NUMBER OF POINTS ON FUS

Program FILLET
SUBROUTINE ADJUN(COMP, INDUN, ICNT, KLINE, UN]
    INTEGER COMP, INDUN, ICNT, KLINE(*)
    REAL UN(*), CENT(3), P(3)

    INTEGER MAXPTS
    PARAMETER (MAXPTS = 50)

    INTEGER NOUNS, IFUS, INING, NFUSIN, NNNGIN, UORN(2, 5), DUMCMP(2)
    INTEGER FUSFLG(MAXPTS), NNGFLG(MAXPTS)
    REAL FUNS(2, MAXPTS), NMHS(2, MAXPTS)
    REAL FINT(2, MAXPTS), HINT(2, MAXPTS)

    COMMON /FILVAL/NOUNS, IFUS, INING, NFUSIN, NNNGIN, FUSFLG, NNGFLG, >
       UORN, DUMCMP, FUNS, NMHS, FINT, HINT

    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), INDUN, KLINE(I)+2, CENT)
          LOWER = FUNS(INDUN, KLINE(I-1))
          UPPER = FUNS(INDUN, KLINE(I+1))
          BOT = (CENT(3) - CENT(1))/(UPPER - LOWER)
          C CALCULATE NEW U OR W VALUE
          UN(I) = LOWER + (CENT(2) - CENT(1))/BOT
       ENDIF
100 CONTINUE

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

    RETURN
END
SUBROUTINE GCENT(DCMP, INDUN, POINT, CENT)

INTEGER DCMP, POINT, INDUN
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

SUBROUTINE ADDKNT(ISIDE, INDUN, ICNT, KLINE, UM)

INTEGER INDUN, ICNT, KLINE(*)
REAL UM(*)

SUBROUTINE ADDKNT(ISIDE, INDUN, ICNT, KLINE, UM)
INTEGER MAXPTS
PARAMETER (MAXPTS = 50)

INTEGER NOUNS, IFUS, IWHING, NFUSIN, NHNGIN, UOIRW(2, 5), DUMCMP(2)
INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
REAL FUNS(2, MAXPTS), NUMS(2, MAXPTS)
REAL FINT(2, MAXPTS), MINT(2, MAXPTS)

COMMON /FILVAL/NOUNS, IFUS, IWHING, NFUSIN, NHNGIN, FUSFLG, WNGFLG,
> UOIRW, DUMCMP, FUNS, NUMS, FINT, MINT

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 SORTKTIINDUW; DUMCMP(1); FUNS(INDUW; KLINE(I))
    100 CONTINUE
C CHECK FOR MODIFICATION FLAG
ELSE IF (FUSFLG(KLINE(I)) .EQ. 2) THEN
    CALL SORTKTIINDUW; DUMCMP(1); FUNS(KLINE(I))
ENDIF
200 CONTINUE
C GET THE DELTA VALUE FROM FILE
CALL GTDELT(IWHING, ISIDE, DELT)
C CHECK FOR U OR N KNOT ADD
IF (INDU .EQ. 1) THEN
    DO 300 J=1,2
        C ADD N KNOTS
        CALL NKTON(DUMCMP(1); 1; FUNS(2; KLINE(1))—(FLOAT(J)*DELT))
        CALL NKTON(DUMCMP(1); 1; FUNS(2; KLINE(1))+(FLOAT(J)*DELT))
    300 CONTINUE
ELSE
    DO 400 J=1,2
        C ADD U KNOTS
        CALL UKTON(DUMCMP(1); 1; FUNS(1; KLINE(1))—(FLOAT(J)*DELT))
        CALL UKTON(DUMCMP(1); 1; FUNS(1; KLINE(1))+(FLOAT(J)*DELT))
    400 CONTINUE
ENDIF
RETURN
END

C============================================================================
C MODULE LDUMMY
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:

Program FILLET
SUBROUTINE LDUMNY(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 PTICMP5(5, OUTCMP, NXSECT, IERR)

C GET NUMBER OF POINTS PER X-SECTION
CALL GTICMP(6, IC, NPPXS, IERR)
CALL PTICMP6(6, OUTCMP, NPPXS, IERR)

C COPY CONTROL POINTS
DO 200 I=1, NXSECT+2
   DO 100 J=1, NPPXS+2
      CALL GTBTST(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 FUNCTIONAL DESCRIPTION:
C ADD AN U OR N KNOT DEPENDING ON VALUE OF INDUN
C=================================================================================================
C METHODS/ALGORITHMS:
C N/A
C=================================================================================================
C CODED BY: J. R. GLOUDEMANS DATE: 3/15/90
C=================================================================================================

SUBROUTINE SORTKT(INDUN, DUMCMP, KVAL)
INTEGER INDUN, DUMCMP
REAL KVAL
C CHECK FOR U KNOT
IF (INDUN .EQ. 1) THEN
C ADD U KNOT
CALL UKTRON(DUMCMP, 1, KVAL)
ELSE
C ADD N KNOT
CALL NKTRON(DUMCMP, 1, KVAL)
ENDIF
RETURN
END

C=================================================================================================
C MODULE GTDELT
C=================================================================================================
C PROJECT: NASA/AMES VERSION: V1.0.0 B-SPLINE TEST
C=================================================================================================
C INPUT PARAMETERS:
C INING — INTEGER — INING 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(INING, ISIDE, DELT)
INTEGER INING, ISIDE, NOUT
REAL DELT, TDELT(10)
PARAMETER (OUT=66)
C SET DEFAULT DELTA DISTANCE
DELT = 0.01
C OPEN THE FILE
OPEN(UNIT = NOUT, FILE = 'KNOTDELT')
RELOAD(NOUT)

Program FILLET
C READ THE NUMBER OF COMPONENTS
READ (INOUT, *, END = 900) NOCOMP

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

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

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

200 CONTINUE

900 CLOSE (INOUT)
RETURN
END

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  INDUN - INTEGER - U OR N INDICATOR
C  ISTART - INTEGER - STARTING INDEX
C  IEND - INTEGER - ENDING INDEX
C
C OUTPUT PARAMETERS:
C  NONE
C
C COMMON INPUTS: FILVAL
C  NOUNS - INTEGER - NUMBER OF U AND N PAIRS
C  IFUN - INTEGER - FUSELAGE COMPONENT NUMBER
C  INING - INTEGER - NING COMPONENT NUMBER
C  NFUSIN - INTEGER - NUMBER OF POINTS ON FUS
C  NNINGIN - INTEGER - NUMBER OF POINTS ON NING
C  FUSFLG - INTEGER - FLAGS CONTROLLING FUS FILLET
C  NNGFLG - INTEGER - FLAGS CONTROLLING NING FILLET
C  UORN - INTEGER - FLAGS INDICATING U OR N DIRECTION
C  DUMCMP - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C  FUNS - REAL - U AND N VALUES FOR FUS BLEND
C  NUNS - REAL - U AND N VALUES FOR NING BLEND
C  FINT - REAL - U AND N INTERMEDIATE VALUES FOR FUS BLEND
C  NINT - REAL - U AND N INTERMEDIATE VALUES FOR NING 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. GLOUDEHANS  DATE: 3/15/90
C
SUBROUTINE LOADVT (COMP, INDUN, ISTART, IEND)
   INTEGER COMP, INDUN, ISTART, IEND, OPPIM

Program FILLET
INTEGER MAXPTS
PARAMETER (MAXPTS = 50)

INTEGER NOUNS, IFUS, IMING, NFUSIN, NNGIN, UORN(2,5), DUMCMP(2)
INTEGER Fusflg(MAXPTS), NNGflg(MAXPTS)
REAL FUMS(2, MAXPTS), MUNS(2, MAXPTS)
REAL FIN(S(2, MAXPTS), WINT(2, MAXPTS)

COMMON /FILVAL/NOUNS, IFUS, IMING, NFUSIN, NNGIN, FUSFLG, NNGFLG,
> UORN, DUMCMP, FUMS, MUNS, FIN, WINT

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

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

C SET A FLAG OPPOSITE TO INDUM
IF (INDUM .EQ. 1) THEN
OPPUM = 2
ELSE
OPPUM = 1
ENDIF

C SET DIRECTION FLAG
IF (FUMS(OPPUM, ISTART) .GE. FIN(T(OPPUM, ISTART)) THEN
IDIR = 1
ELSE
IDIR = 0
ENDIF

CALL FINDBG(OPPUM, DUMCMP(1), FUMS(OPPUM, ISTART), ILINE)
CALL GTICMP(S, 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(INDUM, 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(INDUM, IDIR, ICNT, I, ILINE, DUMCMP(1), COMP)
200 CONTINUE
ENDIF
C ADJUST NUMBER OF POINTS IN U DIRECTION
CALL PTICMP(S, DUMCMP(2), ICNT, IERR)

Program FILLET
SUBROUTINE FINDBG(INDUN, DUMCMP, UN, IBEG)

   INTEGER INDUN, DUMCMP, IBEG
   REAL UN, ERROR, KNOT
   PARAMETER (ERROR = 0.0001)

   C SET ONCE THROUGH FLAG
   IFLAG = 0

   C GET NUMBER OF KNOT VALUES
   CALL GTICMP(INDUN+4, DUMCMP, NPTS, IERR)
   DO 100 I=1, NPTS+4

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

   C CALCULATE DELTA VALUE
   DELTA = ABS(UN - 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

   C ENDIF

   IBEG = I-1

   ENDIF

  100 CONTINUE

   RETURN

END
SUBROUTINE FINDFN(INDUW, DUMCMP, UN, IFIN)
INTEGER INDUW, DUMCMP, IFIN
REAL UN, ERROR, KNOT
PARAMETER (ERROR = 0.0001)

C SET ONCE THROUGH FLAG
IFLAG = 0
C GET NUMBER OF KNOTS
CALL GTICMP(INDUN+4, DUMCMP, NPTS, IERR)
DO 100 I=NPTS+4, 1, -1
C GET KNOT VALUE
CALL GTKNOT(I, INDUN, DUMCMP, KNOT, IERR)
C CALCULATE DELTA VALUE
DELTA = ABS(UN — 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

Program FILLET
SUBROUTINE GETPUT(INDUN; IDIR; ICNT; ICURR; ILINE; DUMCMP; COMP)

INTEGER INDUN, 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 N FLAG
    IF (INDUN .EQ. 1) THEN
      C COPY CONTROL VERTEX INTO FILLET DATA STRUCTURE
      CALL GTBTSF(ICURR—1; ILINE—5+J; 2; DUMCMP; P; IERR)
      CALL PTBTSFIICNT; 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 PTBTSFlICNT; J; 2; COMP; P; IERR)
    ENDIF
  100 CONTINUE
ELSE
  C OTHER DIRECTION
  DO 200 J = 3,1,-1
    IF (INDUN .EQ. 1) THEN
      CALL GTBTSF(ICURR-1; ILINE-3+J; 2; DUMCMP; P; IERR)
      CALL PTBTSFlICNT; 4-J; 2; COMP; P; IERR)
    ELSE
      CALL GTBTSF(ILINE—3+J; ICURR-1; 2; DUMCMP; P; IERR)
      CALL PTBTSFlICNT; 4*J; 2; COMP; P; IERR)
    ENDIF
  200 CONTINUE
ENDIF
RETURN
END

PROGRAM FILLET
SUBROUTINE ANINGK(COMP)

INTEGER COMP, MAXNUN
PARAMETER (MAXNUN = 50)
REAL UN(MAXNUN)

ADJUST KNOT SPACING TO MATCH CENTRIPETAL SPACING
CALL NADJUNICOMP; UN)

LOAD DUMMY COMPONENT AND ADD KNOTS
CALL NADDKTM(UN)

LOAD CONTROL VERTICES INTO COMPONENT
CALL NLODVTICOMPI

RETURN
END

MODULE NADJU

PROJECT: NASA/AMES VERSION: V1.0.0 B-SPLINE TEST

INPUT PARAMETERS:
C COMP - INTEGER - COMPONENT IDENTIFIER
C UN - REAL ARRAY - ADJUSTED U OR N VALUES
C COMMON INPUTS: FILVAL
C NOUNS - INTEGER - NUMBER OF U AND M PAIRS
C IFUM - INTEGER - FUSELAGE COMPONENT NUMBER
C IMING - INTEGER - WING COMPONENT NUMBER
C NFUSIN - INTEGER - NUMBER OF POINTS ON FUS
C NNINGIN - INTEGER - NUMBER OF POINTS ON WING
C FUSFLG - INTEGER - FLAGS CONTROLLING FUS FILLET
C MINGFLG - INTEGER - FLAGS CONTROLLING WING FILLET
C UORH - INTEGER - FLAGS INDICATING U OR M DIRECTION
C DUMCMP - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C FUN - REAL - U AND M VALUES FOR FUS BLEND
C NMUNS - REAL - U AND M VALUES FOR MING BLEND
C FINT - REAL - U AND M INTERMEDIATE VALUES FOR FUS BLEND
C MINT - REAL - U AND M INTERMEDIATE VALUES FOR WING BLEND

COMMON OUTPUTS:
C NONE

FUNCTIONAL DESCRIPTION:
C FIND ADJUSTED U OR M VALUES FOR THE WING

Program FILLET
SUBROUTINE NADJUN(COMP, UN)

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

INTEGER NOUNS, IFUS, IHING, NFUSIN, NHNGIN, UORM(2, 5), DUMCMP(2)
INTEGER FUSFLG(MAXPTS), NHNGFLG(MAXPTS)
REAL FUNS(2, MAXPTS), NUNS(2, MAXPTS)
REAL FINT(2, MAXPTS), LINT(2, MAXPTS)

COMMON /FILVAL/NOUNS, IFUS, IHING, NFUSIN, NHNGIN, FUSFLG, NHNGFLG,
UORM, DUMCMP, FUNS, NUNS, FINT, LINT

C SET INDUN FLAG
INDUN = UORM(2,1)

C LOOP THROUGH ALL U/N VALUES
DO 100 I=2, NOUNS-1
C CHECK FOR ADJUSTMENT FLAG
IF (NHNGFLG(I) .EQ. 2) THEN
C GET ADJACENT U/N VALUES
CALL GCENT(DUMCMP(2), INDUN, I+2, CENT)
LOWER = NUNS(INDUN, I-1)
UPPER = NUNS(INDUN, I+1)
BOT = (CENT(3) - CENT(1))/(UPPER - LOWER)
C CALCULATE ADJUSTED U OR N VALUE
UN(I) = LOWER + (CENT(2) - CENT(1))/BOT
ENDIF
100 CONTINUE
C SET LAST U OR N VALUE
UN(NOUNS+1) = 2*NUNS(INDUN, NOUNS) - UN(NOUNS-1)
RETURN
END

MODULE NADJUN

PROJECT: NASA/AMES
VERSION: V1.0.0 B-SPLINE TEST

INPUT PARAMETERS:
C UN — REAL ARRAY - ADJUSTED U OR N VALUES
C
OUTPUT PARAMETERS:
C NONE

COMMON INPUTS: FILVAL
C NOUNS — INTEGER — NUMBER OF U AND N PAIRS
C IFUS — INTEGER — FUSELAGE COMPONENT NUMBER
C IHING — INTEGER — WING COMPONENT NUMBER
C NFUSIN — INTEGER — NUMBER OF POINTS ON FUS
C NHNGIN — INTEGER — NUMBER OF POINTS ON WING
C FUSFLG — INTEGER — FLAGS CONTROLLING FUS FILLET
C NHNGFLG — INTEGER — FLAGS CONTROLLING WING FILLET
C UORM — INTEGER — FLAGS INDICATING U OR N DIRECTION
C DUMCMP — INTEGER — COMPONENT NUMBERS FOR DUMMY COMPS
C FUNS — REAL — U AND N VALUES FOR FUS BLEND
C NUNS — REAL — U AND N VALUES FOR WING BLEND
C FINT — REAL — U AND N INTERMEDIATE VALUES FOR FUS BLEND

Program FILLET 100
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 NADDKTKUN)
REAL UN(*); DELT
INTEGER MAXPTS
PARAMETER (MAXPTS = 50, DELT = 0.1)

INTEGER NOUNS, IFUS, IHING, NFUSIN, NNNGIN, UORH(2, 5), DUMCMP(2)
INTEGER AFUSFLG(MAXPTS), NNNGFLG(MAXPTS)
REAL FUMS(2, MAXPTS), MUNS(2, MAXPTS)
REAL FINT(2, MAXPTS), NINT(2, MAXPTS)

COMMON /FILVAL/NOUN, IFUS, IHING, NFUSIN, NNNGIN, FUSFLG, WNGFLG, >
> UORH, DUMCMP, FUMS, MUNS, FINT, WINT

C SET U OR W INDICATOR
INDUN = UORH(2, 1)

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

C SET FIRST AND LAST CONTROL VERTICES
CALL LOOPNG(DUMCMPl1); INDUN)

C LOOP THROUGH ALL POINTS
DO 200 I=1, NOUNS
C CHECK FOR CORNER POINT
IF (NNNGFLGI) .EQ. 1) THEN
C ADD THREE KNOT TO CORNER
DO 100 J=1,3
CALL SORrKt(INDUN, DUMCMPl1, MUNS(INDUN, I))
100 CONTINUE
ELSE IF (NNNGFLG(I) .EQ. 2) THEN
C ADD ONE KNOT TO ADJUSTED U OR W VALUE
CALL SORrKt(INDUN, DUMCMPl1, UN(INDUN, I))
ENDIF
200 CONTINUE

CALL SORrKt(INDUN, DUMCMPl1, UN(NOUNS+1))

C CHECK U OR W INDICATOR
IF (INDUN .EQ. 1) THEN
DO 300 J=1,2
C ADD W KNOTS
CALL MKTRON(DUMCMP(1), 1, MUNS(2, 1)-(FLOAT(J)*DELT))
CALL MKTRON(DUMCMP(1), 1, MUNS(2, 1)+(FLOAT(J)*DELT))
300 CONTINUE
ELSE
DO 400 J=1,2
C ADD U KNOTS
CALL UKTRON(DUMCMP(1), 1, MUNS(1, 1)-(FLOAT(J)*DELT))
400 CONTINUE
ENDIF

Program FILLET
CALL UKTRON(DUMCMP(1), 1, MUMS(1, 1)+FLOAT(J)*DELT))

CONTINUE

ENDIF

RETURN

END

C=====================================================================================================================
C MODULE LOOPWG
C=====================================================================================================================
C PROJECT: NASA/AMES VERSION: V1.0.0 B-SPLINE TEST
C=====================================================================================================================
C INPUT PARAMETERS:
C DCMP — INTEGER — DUMMY COMPONENT IDENTIFIER
C INDUN — INTEGER — U OR M 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 M VALUE
C=====================================================================================================================
C METHODS/ALGORITHMS:
C N/A
C=====================================================================================================================
C CODED BY: J. R. GLOUDEMANS DATE: 3/15/90
C=====================================================================================================================

SUBROUTINE LOOPWG(DCMP, INDUN)

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

C GET NUMBER OF POINTS AND X-SECTIONS
CALL GTICMP(5, DCMP, NXSECT, IERR)
CALL GTICMPT6, DCMP, NPPXS, IERR)

C CHECK U OR M INDICATOR
IF (INDUN .EQ. Z) THEN
C LOOP THROUGH ALL CONTROL POINTS
DO 100 I=1, NXSECT+2
C GET AND PUT NEXT TO LAST CONTROL POINT
CALL GBTSF(I, 3, 2, DCMP, P, IERR)
CALL PTBTSF(I, NPPXS+3, 2, DCMP, P, IERR)
CALL GBTSF(I, 4, 2, DCMP, P, IERR)
CALL PTBTSF(I, NPPXS+4, 2, DCMP, P, IERR)
CALL PTICMP(6, DCMP, NPPXS+2, IERR)
100 CONTINUE
C ELSE
DO 200 I=1, NPPXS+2
C CALL GBTSF(I, 3, 2, DCMP, P, IERR)
CALL PTBTSF(I, NXSECT+3, 2, DCMP, P, IERR)
C CALL GTBTSF(I, 4, 2, DCMP, P, IERR)
CALL PTBTSF(I, NXSECT+4, 2, DCMP, P, IERR)
200 CONTINUE

Program FILLET
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 MODULE WLODVT
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 NOUNS - INTEGER - NUMBER OF U AND M PAIRS
C IFUM - INTEGER - FUSELAGE COMPONENT NUMBER
C IMING - INTEGER - WING COMPONENT NUMBER
C NFUSIN - INTEGER - NUMBER OF POINTS ON FUS
C NINGIN - INTEGER - NUMBER OF POINTS ON WING
C FUSFLG - INTEGER - FLAGS CONTROLLING FUS FILLET
C NINGFLG - INTEGER - FLAGS CONTROLLING WING FILLET
C UORM - INTEGER - FLAGS INDICATING U OR M DIRECTION
C DUMCMP - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C FUNS - REAL - U AND M VALUES FOR FUS BLEND
C NUNS - REAL - U AND M VALUES FOR NING BLEND
C FINT - REAL - U AND M INTERMEDIATE VALUES FOR FUS BLEND
C NINT - REAL - U AND M INTERMEDIATE VALUES FOR NING BLEND
C==============================================================================

C COMMON OUTPUTS:
C NONE
C==============================================================================

C FUNCTIONAL DESCRIPTION:
C LOAD CONTROL VERTICES FOR NING 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, OPPUM
PARAMETER (MAXPTS = 50)

INTEGER NOUNS, IFUS, IMING, NFUSIN, NINGIN, UORM(2, 5), DUMCMP(2)
INTEGER FUSFLG(MAXPTS), NINGFLG(MAXPTS)
REAL FUNS(2, MAXPTS), MUNS(2, MAXPTS)
REAL FINT(2, MAXPTS), NINT(2, MAXPTS), P(13)

COMMON /FILVAL/NOUNS, IFUS, IMING, NFUSIN, NINGIN, FUNS, MUNS, FINT, NINT

C SET U OR M INDICATOR
INDUM = UORM(2,1)
C GET NUMBER OF POINTS
CALL GTCMP(5, COMP, NOPTS, IERR)

IF (INDUM .EQ. 1) THEN
  OPPUM = 2
ELSE
  OPPUM = 1
ENDIF

C FIND BEGINNING INDEX
CALL FINDBG(INDUN, DUMCMPl), NUNSIOPPUW, 1), ILINE)

C SET OFFSET
  IOFF = 4+NFUSIN+NNNGIN
ELSE
  IOFF = 3+NFUSIN+NNNGIN
ENDIF

DO 100 I=1, NOPTS+2
C GET AND PUT CONTROL VERTICES
CALL MGTPUT(INDUM, I, ILINE, IOFF, DUMCMPl), 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(I+IOFF, 2, COMP, P, IERR)

300 CONTINUE

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

RETURN
END

C M O D U L E M G T P U T
C
C PROJECT: NASA/AMES  VERSION: V1.0.0  B-SPLINE TEST
C
C INPUT PARAMETERS:
C  INDUM - 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
SUBROUTINE NGTPUT(INDUW, ICURR, ILINE, IOFF, DCMP, COMP)

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

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

SUBROUTINE NINVT(COMP)

INTEGER COMP
INTEGER NPPXS, NXSECT, IERR
INTEGER MAXPTS
PARAMETER IMAXPTS = 50)
INTEGER NFLAGS(MAXPTS)
REAL NKNTS(MAXPTS), NBLN(3, MAXPTS), PTS(3,MAXPTS)
REAL OUTPTS(3,MAXPTS), UKNT, OLDKNT, TOL
REAL P(3)
PARAMETER (TOL = 0.0001)

CALL NLODKT(COMP)

Program FILLET
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, NKNTS(I), IERR)
MFLAGS(I) = 4
100 CONTINUE

C SET close/open flags
DO 200 I=1,3
     MFLAGS(I) = 3
     MFLAGS(I+NPPXS-1) = 3
200 CONTINUE

C calculate Non-uniform blending functions
CALL NONBLN(NPPXS+2, NKNTS, MBLN)

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(I, 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(I, 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(I, J), IERR)
350 CONTINUE
ENDIF

C PERFORM NON-UNIFORM INVERSION
CALL NNVEPT(NPPXS+2, PTS, MFLAGS, MBLN, OUTPTS)
DO 400 J=4, NPPXS+2
C LOAD POINTS INTO DATA STRUCTURE
CALL PTBTSF(I, J, 2, COMP, OUTPTS(I, J), IERR)

Program FILLET
C CHECK POINT SPACING
CALL CHECKP(COMP)
RETURN
END

C M O D U L E  W L O D K T

C PROJECT: NASA/AMES   VERSION: V1.0.0 B-SPLINE TEST

C INPUT PARAMETERS:
C COMP - INTEGER - COMPONENT IDENTIFIER
C OUTPUT PARAMETERS:
C NONE

C COMMON INPUTS: FILVAL
C NOUNS - INTEGER - NUMBER OF U AND W PAIRS
C IFUS - INTEGER - FUSELAGE COMPONENT NUMBER
C I MING - INTEGER - WING COMPONENT NUMBER
C NFUSIN - INTEGER - NUMBER OF POINTS ON FUS
C NHNGIN - INTEGER - NUMBER OF POINTS ON WING
C FUSFLG - INTEGER - FLAGS CONTROLLING FUS FILLET
C WNGFLG - INTEGER - FLAGS CONTROLLING WING FILLET
C UORN - INTEGER - FLAGS INDICATING U OR W DIRECTION
C DUMCMP - INTEGER - COMPONENT NUMBERS FOR DUMMY COMPS
C FUNS - REAL - U AND W VALUES FOR FUS BLEND
C MUNS - REAL - U AND W VALUES FOR WING BLEND
C FINT - REAL - U AND W INTERMEDIATE VALUES FOR FUS BLEND
C MINT - REAL - U AND W INTERMEDIATE VALUES FOR WING BLEND

C COMMON OUTPUTS:
C NONE

C FUNCTIONAL DESCRIPTION:
C CALCULATE AND LOAD N KNOTS INTO DATA STRUCTURE

C METHODS/ALGORITHMS:
C N/A

C CODED BY: J. R. GLOUDEMANS   DATE: 3/15/90

C SUBROUTINE  W L O D K T(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 NOUNS, IFUS, I MING, NFUSIN, NHNGIN, UORN(2, 5), DUMCMP(2)
INTEGER FUSFLG(MAXPTS), WNGFLG(MAXPTS)
REAL FUNS(2, MAXPTS), MUNS(2, MAXPTS)
REAL FINT(2, MAXPTS), MINT(2, MAXPTS)

COMMON /FILVAL/NOUNS, IFUS, I MING, NFUSIN, NHNGIN, FUNS, MUNS, FINT, MINT

C GET NUMBER OF POINTS
CALL GTICMP(6, COMP, NPTS, IERR)
C CALCULATE FIRST AND LAST POINTS FROM U AND N VALUES
CALL FNDPNT(IFUS, FUMS(1,1), FUMS(2,1), PF)
CALL FNDPNT(IMING, HMNS(1,1), HMNS(2,1), PL)

DO 100 I=1, NPTS-4
C GET REST OF POINTS FROM DATA STRUCTURE
CALL GTBTSF(I, I, COMP, PTS(I, 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)) * NDELTA
50 CONTINUE
C USE CENTRIPEAL 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) - IKNTSIZ) - 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 M O D U L E C H E C K P
C PROJECT: NASA/AMES VERSION: V1.0.0 B-SPLINE TEST
C INPUT PARAMETERS:
C COMP — INTEGER - COMPONENT IDENTIFIER
C OUTPUT PARAMETERS:
C NONE
C COMMON INPUTS:
C NONE
C COMMON OUTPUTS:
C NONE
C FUNCTIONAL DESCRIPTION:
C CHECK POINT SPACING TO COMPARE WITH GUESSED SPACING
C FOR DEBUG PURPOSES ONLY
C METHODS/ALGORITHMS:
C N/A
C CODED BY: J. R. GLOUDEMANS DATE: 3/15/90
SUBROUTINE CHECKP(COMPI

INTEGER COMP
INTEGER NXSECT, NPPXS, IERR, MAXPNT
PARAMETER(MAXPNT = 50)
REAL UKNTS(MAXPNT), MKNTS(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(1, 1, COMP, U, IERR)
CALL GTKNOT(1, 2, COMP, W, IERR)

C CALCULATE BLENDING FUNCTIONS
CALL BLENDU(UKNTS, 3, UBLEND)
CALL BLENDW(WKNTS, 3, WBLEND)

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

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

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

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

C FIND DISTANCE BETWEEN SECOND AND THIRD POINTS
CALL SUBIPS(P2, P3, 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
C 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=============================================================================
SUBROUTINE STRKNT(COMP, NNSECT, NPPXS)

INTEGER COMP, NNSECT, NPPXS
REAL T

C LOOP THROUGH ALL U KNOTS
DO 100 I = 1, NNSECT + 6
   T = FLOAT(I)
   CALL PTKNOT(I, 1, COMP, T, IERR)
100 CONTINUE

C LOOP THROUGH ALL N KNOTS
DO 200 J = 1, NPPXS + 6
   T = FLOAT(J)
   CALL PTKNOT(J, 2, COMP, T, IERR)
200 CONTINUE

RETURN
END
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 MODULE CHORDL
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

 Program FILLET  111
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 MODULE VARPAR  
C======================================================================  
C PROJECT: NASA/AMES  
C 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 COMMON OUTPUTS:  
C NONE  
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+2)-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 MODULE PDIST
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

Program FILLET
MODULE UKTRON

INPUT PARAMETERS:
  COMP - INTEGER - COMPONENT IDENTIFIER
  PATCH - INTEGER - SUB-SURFACE ON COMPONENT
  UKNOT - REAL - KNOT VALUE TO BE ADDED

OUTPUT PARAMETERS:
  NONE

COMMON INPUTS:
  NONE

COMMON OUTPUTS:
  NONE

FUNCTIONAL DESCRIPTION:
  ADD ONE ROW OF U KNOTS AND VERTICES TO A B-SPINE SURFACE

METHODS/ALGORITHMS:
  ADAPTED FROM M. BOHM'S ALGORITHM FOUND IN "INSERTING NEW KNOTS INTO B-SPINE CURVES", J. APPROX TH. 6, 50-62

CODED BY: J. R. GLOUDEMANS DATE: 3/15/90

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(S, 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 NUVERTICOMP(PATCH, NU, UKVALS, UKNOT, INTER)

C ADJUST KNOT SPACING AND ADD TO DATABASE
CALL ADJUKTICOMP(PATCH, NU, UKVALS, UKNOT, INTER)
RETURN
END

C M O D U L E F I N D I T
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. GLOUEMANS 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
         SET INTERVAL AND BREAK LOOP
         INTER = I
         GOTO 200
      ENDIF
   100 CONTINUE
module AFACT

C PROJECT: NASA/AMES
VERSION: V1.0.0 B-SPLINE TEST

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 OUTPUT PARAMETERS:
C FACT - REAL ARRAY - FACTORS USED TO MODIFY CONTROL POINTS

C COMMON INPUTS:
C NONE

C COMMON OUTPUTS:
C NONE

C FUNCTIONAL DESCRIPTION:
C FIND THE FACTORS USED TO ADJUST THE CONTROL VERTICES

C METHODS/ALGORITHMS:
C N/A

C CODED BY: J. R. GLOUDEMANS
DATE: 3/15/90

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

module NUVERT

C PROJECT: NASA/AMES
VERSION: V1.0.0 B-SPLINE TEST

Program NEWKNOT
SUBROUTINE NUVERT(COMP, PATCH, NU, FACT)
    INTEGER COMP, PATCH, NU
    REAL FACT(*)

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

    C GET NUMBER OF VERTICES IN THE W DIRECTION
    CALL GTICMPI6; COMP; NPPXS; IERR
    NN = NPPXS+2

    C GET THE EXISTING CONTROL GRID
    CALL GTGRIDICOMP; NU; W; GRID

    DO 300 J = 1, NW
       DO 100 I = 1, NU
          DO 50 K=1,3
             ULINE(K, I) = GRID(K, I, J)
          50 CONTINUE
       100 CONTINUE
    300 CONTINUE

    C ADJUST U VERTICES FOR THE NEW KNOT VALUE
    CALL ADJVRTINU; ULINE; FACT; NEWLIN

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

RETURN
END

SUBROUTINE GTGRID

MODULE GTGRID

Program NEWKNOT
SUBROUTINE GTGRIDICOMP, NU, NN, 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,NN
      C GET POINT FROM DATA STRUCTURE
      CALL GTBTSF(I, J, Z, 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 MODUL E A DJVRT
C PROJECT: NASA/AMES  VERSION: V1.0.0 B-SPLINE TEST
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 OUTPUT PARAMETERS:
C NEWLIN - REAL ARRAY - NEW OR ADJUSTED ROW OF CONTROL VERTICES
C COMMON INPUTS:
C NONE
C COMMON OUTPUTS:
C NONE
C FUNCTIONAL DESCRIPTION:
C ADJUST ONE ROW OF CONTROL VERTICES FOR NEW KNOT

Program NEWKNOT
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 NEH 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

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

RETURN
END

Program NEWKNOT
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 PTICMPIS(COMP, NU-1, IERR)
RETURN
END

C============================================================================:::::::2:

C MODULE NKTRON
C============================================================================:::::::2:

C PROJECT: NASA/AMES VERSION: V1.0.0 B-SPLINE TEST
C============================================================================:::::::2:

C INPUT PARAMETERS:
C COMP - INTEGER - COMPONENT IDENTIFIER
C PATCH - INTEGER - SUB-COMPONENT IDENTIFIER
C NKNOT - REAL - M KNOT VALUE
C============================================================================:::::::2:

C OUTPUT PARAMETERS:
C NONE
C============================================================================:::::::2:

C COMMON INPUTS:
C NONE
C============================================================================:::::::2:

C COMMON OUTPUTS:
C NONE
C============================================================================:::::::2:

C FUNCTIONAL DESCRIPTION:
C ADD ONE ROW OF M KNOTS AND VERTICES TO THE B-SPLINE SURFACE
C============================================================================:::::::2:

C METHODS/ALGORITHMS:
C N/A
C============================================================================:::::::2:

C CODED BY: J. R. GLOUDEMANS DATE: 3/15/90
C============================================================================:::::::2:

SUBROUTINE NKTRON(COMP, PATCH, NKNOT)

INTEGER COMP, PATCH
REAL NKNOT

INTEGER N, INTER, MAXPTS, NPPXS
PARAMETER(MAXPTS=150)
REAL UKVALS(MAXPTS), FACT(MAXPTS)

C GET NUMBER OF M CONTROL VERTICES FROM DATABASE
CALL GTICMPIS(COMP, NPPXS, IERR)
N = NPPXS*2

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

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

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

C FIND THE NEW CONTROL VERTICES AND ADD TO DATABASE
CALL NNVERTICOMP; PATCH; NN; FACT)

C ADJUST KNOT SPACING AND ADD TO DATABASE
CALL ADJNKT(COMP; PATCH; NN; MKVALS; MKNOT; INTER)

RETURN
END

C M O D U L E N N V E R T

C PROJECT: NASA/AMES  VERSION: V1.0.0  B-SPLINE TEST

C INPUT PARAMETERS:
C COMP - INTEGER - COMPONENT IDENTIFIER
C PATCH - INTEGER - SUB-COMPONENT IDENTIFIER
C NN - INTEGER - NUMBER OF N KNOTS
C FACT - REAL ARRAY - FACTORS USED TO ADJUST CONTROL VERTICES

C OUTPUT PARAMETERS:
C NONE

C COMMON INPUTS:
C NONE

C COMMON OUTPUTS:
C NONE

C FUNCTIONAL DESCRIPTION:
C FIND THE NEN CONTROL VERTICES AND ADD THEM TO THE DATA STRUCTURE

C METHODS/ALGORITHMS:
C N/A

C CODED BY:  J. R. GLOUDEMANS  DATE:  3/15/90

SUBROUTINE NNVERTICOMP; PATCH; NN; FACT)
INTEGER COMP, PATCH, NU
REAL FACT(*)

INTEGER MAXU, MAXN, NXSECT
PARAMETER (IMAXU = 150; MAXN = 150)
REAL GRID(3; MAXU; MAXN); NLINE(3; MAXH)
REAL NENLIN(3; MAXN)

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 GTGRIDICOMP; NU; NN; GRID)

DO 300 J = 1; NU
DO 100 I = 1; NH
DO 50 K=1,3
C LOAD ARRAY FOR EACH N VERTEX
NLINElK; I) = GRIDlK; J; I)
50 CONTINUE
100 CONTINUE

C ADJUST N VERTICES FOR THE NEW KNOT VALUE
CALL ADJVRTINN; NLINE; FACT; NENLIN

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

Program NEWKNOT
CALL PTBTSF(J, I, 2, COMP, NEWLIN(I,I), IERR)

200 CONTINUE
300 CONTINUE

RETURN
END

MODULE ADJNKT

PROJECT: NASA/AMES  VERSION: V1.0.0  B-SPLINE TEST

INPUT PARAMETERS:
COMP — INTEGER - COMPONENT IDENTIFIER
PATCH — INTEGER - SUB-COMPONENT IDENTIFIER
NN — INTEGER — NUMBER OF M CONTROL VERTICES
NKVALS — REAL ARRAY - M KNOT VALUES
NKNOT — REAL - M KNOT VALUE TO BE ADDED
INTER — INTEGER - INTERVAL OF ADDED KNOT

OUTPUT PARAMETERS:
NONE

COMMON INPUTS:
NONE

COMMON OUTPUTS:
NONE

FUNCTIONAL DESCRIPTION:
ADJUST N KNOT SPACING AND ADD KNOTS TO THE DATA STRUCTURE

METHODS/ALGORITHMS:
N/A

CODED BY: J. R. GLOUDEMANS  DATE: 3/15/90

SUBROUTINE ADJNKT(COMP, PATCH, NN, NKVALS, NKNOT, INTER)

INTEGER COMP, PATCH, NN, INTER
REAL NKVALS(*), NKNOT

C INCREMENT ALL KNOT VALUES BY ONE AFTER INSERTED KNOT
DO 100 I=NN+4, INTER+2, -1
   NKVALS(I) = NKVALS(I-1)
100 CONTINUE

C ADJUST FIRST KNOT AFTER INTERVAL
   NKVALS(INTER+1) = NKNOT

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

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

RETURN
END

Program NEWKNOT
Appendix C

Program NONINVT

C=========================================================================
C MODULE 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
SUBROUTINE NNVEHL(NXSECT, NPPXS, PTS, FLAGS, UKNTS, NKNTS, 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(*), NKNTS(*), Ni, SECT+2)

C COMPUTE NON—UNIFORM BLENDING FUNCTIONS
CALL NONBLN(NXSECT+2, UKNTS, HUL)
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 MATSNC(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, NKNTS, N)
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 MATSNC(NXSECT+2, NPPXS+2, TMP, HUL)
C RETURN
RETURN
END
C
Cathom
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
Program NONINVT
SUBROUTINE INIENPT(NPTS, PTS, FLAGS, N, HUL)

C DECLARE THE VARIABLES
INTEGER END, NPTS, FLAGS(*)
REAL PTS(5,*)
REAL HUL(5,*)
REAL N(5,*)

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

C FIND HULL
CALL NNDEHLINPTS(PTS, FLAGS, N, HUL)

C RETURN
RETURN
END

SUBROUTINE INITEL(NPTS, PTS, FLGS, HUL)

C DECLARE THE VARIABLES
INTEGER END, NPTS, FLGS(*)
REAL PTS(5,*)
REAL HUL(5,*)
REAL N(5,*)

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

C FIND HULL
CALL NNDEHLINPTS(PTS, FLGS, N, HUL)

C RETURN
RETURN
END

SUBROUTINE NNDEHLINPTS(PTS, FLGS, N, HUL)

C DECLARE THE VARIABLES
INTEGER END, NPTS, FLGS(*)
REAL PTS(5,*)
REAL HUL(5,*)
REAL N(5,*)

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

C FIND HULL
CALL NNDEHLINPTS(PTS, FLGS, N, HUL)

C RETURN
RETURN
END

SUBROUTINE NINENPT(NPTS, PTS, FLGS, N, HUL)

C DECLARE THE VARIABLES
INTEGER END, NPTS, FLGS(*)
REAL PTS(5,*)
REAL HUL(5,*)
REAL N(5,*)

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

C FIND HULL
CALL NNDEHLINPTS(PTS, FLGS, N, HUL)

C RETURN
RETURN
END

SUBROUTINE NNDEHLINPTS(PTS, FLGS, N, HUL)

C DECLARE THE VARIABLES
INTEGER END, NPTS, FLGS(*)
REAL PTS(5,*)
REAL HUL(5,*)
REAL N(5,*)

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

C FIND HULL
CALL NNDEHLINPTS(PTS, FLGS, N, HUL)

C RETURN
RETURN
END
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

C RETURN
RETURN
END
DO 10 J = 2, NPTS - 1
C CHECK FOR HOLD FLAG
IF (FLAGS(J) .EQ. 4) THEN
    CALL NONDEL(PTS(I,J), HUL(I,J-1), HUL(I,J), HUL(I,J+1), N(1,J+1), DEL)
    FIND DELMA
    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 MODULE NONDEL
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 MODULE NONBLN
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. GLOUDEMAN DATE: 3/15/90
C==================================:

SUBROUTINE NONBLN(NPTS, KNOTS, N)

INTEGER NPTS
REAL KNOTS(*), N(3,*), 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)

Program NONINVT 128
\[ N(2, NPTS+2) = \text{BLEND}(3) \]
\[ N(3, NPTS+2) = \text{BLEND}(4) \]

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

Program NONINVT 129
100 CONTINUE
200 CONTINUE

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

RETURN
END

C====::2:2::3======2:2:=========================2::====:22=======:====
C M O D U L E M T R A N S
C===:::2:::2::::::::::2::22::===========:2::::::.1=====2===============
C PROJECT: NASA/AMES VERSION: V1.0.0 B-SPLINE TEST
C======2::::2:::;*:::::*;:::::::::2:::::::;*::=:=======:=================
C INPUT PARAMETERS:
C NRONS - INTEGER; NUMBER OF RONS
C NCOLS - INTEGER; NUMBER OF COLUMNS
C MATIN - REAL, (R, C) INPUT MATRIX
C==============:========:============:===3=================:==========
C OUTPUT PARAMETERS:
C MATOUT - REAL, (C, R) OUTPUT MATRIX
C==============:========:============:===3=================:==========
C COMMON INPUTS:
C NONE
C==============:========:============:===3=================:==========
C COMMON OUTPUTS:
C NONE
C==============:========:============:===3=================:==========
C FUNCTIONAL DESCRIPTION:
C SWITCHES THE RONS AND COLUMNS OF (R,C) MATRIX
C==============:========:============:===3=================:==========
C METHODS/ALGORITHMS:
C N/A
C==============:========:============:===3=================:==========
C CODED BY: K. V. KOLADY DATE: 3/15/90
C==============:========:============:===3=================:==========
C SUBROUTINE MTRANS(NRONS, NCOLS, MATIN, MATOUT)

C DECLARE THE VARIABLES
INTEGER NRONS, NCOLS
INTEGER SECT, NODE
PARAMETER(SECT=50, NODE=50)
INTEGER MATN(SECT, NODE), MATOT(SECT, NODE)

C DO 30 I=1,NRONS
   DO 20 J=1,NCOLS
      MATOT(I,J) = MATN(J,I)
20 CONTINUE
30 CONTINUE
C RETURN
END
Appendix D

Program RENDER

```
C=================================================================================================
C          MODULE NONSRF
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  NN - INTEGER - NUMBER M 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, NN)
  INTEGER COMP, NXSECT, NPPXS, NU, NN
  INTEGER TOTU, TOTN
  REAL P(3)
  
  C COMPUTE TOTAL NUMBER OF U AND M LINES
  TOTU = INT((NPPXS-1)*NU+1)
```
TOTN = INT((NXSEC-1)*NH+1)

C GENERATE U LINES
CALL UNON(COMP, NXSEC, NPPXS, TOTU, TOTN)

C GENERATE U LINES
CALL WNUN(COMP, NXSEC, NPPXS, TOTU, TOTN)

RETURN
END

C========================================================================
C MODULE UNON
C========================================================================
C PROJECT: NASA/AMES  VERSION: V1.0.0 B-SPLINE TEST
C========================================================================
C INPUT PARAMETERS:
C COMP - INTEGER - COMPONENT IDENTIFIER
C NXSEC - INTEGER - NUMBER OF X-SECTIONS
C NPPXS - INTEGER - NUMBER OF POINTS PER X-SECTION
C TOTU - INTEGER - TOTAL NUMBER U LINES
C TOTN - INTEGER - TOTAL NUMBER U 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, NXSEC, NPPXS, TOTU, TOTN)
INTEGER COMP, NKSECT, NPPX$, TOTU, TOTN

INTEGER MAXKNT, NPTS
PARAMETER (MAXKNT = 50, NPTS=10)

INTEGER IU, IN
REAL UKNTS(MAXKNT), NKNTS(MAXKNT), UDIFF, WDIFF
REAL BLNU(4), BLNM(4), XX(NPTS), YY(NPTS), ZZ(NPTS)
REAL TOL, HSI(300)
PARAMETER (TOL = 0.000000001)

C LOAD KNOT ARRAYS
CALL LOADKT(COMP, NXSEC, NPPXS, UKNTS, NKNTS)

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

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

C COMPUTE M BLENDING FUNCTIONS AND INTERVAL
CALL BLNINT(I, TOTU, NPPXS+4, WDIFF, NKNTS, IM, BLNM)

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

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

200 CONTINUE

100 CONTINUE

END
IF (UDIFF .GT. 0.0) THEN
   DO 50 K=0,NPTS-1
   C COMPUTE U BLENDING FUNCTIONS AND INTERVAL
      CALL SMLBLNK(K, NPTS, UDIFF, UKNTS, J, BLNU)
   C COMPUTE POINT ON SURFACE
      CALL NONPT(COMJ,J,IM,BLNJ,BLNM,XX(K+1),YY(K+1),ZZ(K+1))
   50 CONTINUE
ENDIF
C DRAN POLYLINE
   CALL VPL3(NPTS, XX, YY, ZZ)
100 CONTINUE
200 CONTINUE
RETURN
END

C==============================================================================
C MODULE NNON
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 - TOTAL NUMBER U LINES
C TOTU - INTEGER - TOTAL NUMBER U LINES
C TOTN - 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 N 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 NNON(COMP, NXSECT, NPPXS, TOTU, TOTN)
   INTEGER COMP, NXSECT, NPPXS, TOTU, TOTN
   INTEGER MAXKNT, NPTS
   PARAMETER (MAXKNT = 50, NPTS=10)

   INTEGER IU, IM
   REAL UKNTS(MAXKNT), WKNTS(MAXKNT), UDIFF, MDIFF
   REAL BLNU(4), BLNM(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, NKNTS)

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

C LOOP THROUGH ALL M LINES
DO 200 I=1, TOTM

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

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

MDIFF = UKNTS(J+1)-UKNTS(J)-TOL
IF (MDIFF .NE. 0.0) THEN
  DO 50 K=0,NPTS-1
   C COMPUTE H BLENDING FUNCTIONS AND INTERVAL
   CALL SMLBLN(IK, NPTS, MDIFF, NKNTS, J, BLNH)
   C COMPUTE POINT ON SURFACE
   CALL NONPT(COMP, IU, J, BLN, BLU, 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 MODULE LOADKT
C=========================================================================
C PROJECT: NASA/AMES
C 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 NKNTS - REAL ARRAY - H 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 CODED BY: J. R. GLOUDEMANS
C DATE: 3/15/90
C=========================================================================

SUBROUTINE LOADKT(COMP, NXSECT, NPPXS, UKNTS, NKNTS)
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 N KNOTS
DO 200 I=1, NPPXS+5
C GET N 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::::::::2:::2::=======:2:::::::2:::2:::::2::2:2:=:============:======
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 UKNTS - REAL ARRAY - U KNOT ARRAY
C NKNTS - REAL ARRAY - N KNOT ARRAY
C:::::::::::::::::::::::::::::::=:::::::::::::::::=:::=:::::::::::::::
C OUTPUT PARAMETERS:
C UDIFF - REAL — DIFFERENCE BETWEEN LON AND HIGH U KNOT VALUES
C NDIFF — REAL — DIFFERENCE BETWEEN LON AND HIGH N KNOT VALUES
C:::::::::::::::::::::::::::::::::::::::::::::::==:::::=::::::::::::::
C COMMON INPUTS:
C NONE
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C COMON OUTPUTS:
C NONE
C::=:::::::::::::::::::=::==::=::=::::::::::=:==:::::::::::=:::=::=:::
C FUNCTIONAL DESCRIPTION:
C COMPUTE DIFFERENCE BETNEEN HIGH AND LON 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, MDIFF)
INTEGER NXSECT, NPPXS
REAL UKNTS(*), WKNTS(*), UDIFF, MDIFF, TOL
PARAMETER (TOL = 0.000000000001)

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

RETURN
END

C================='=====::::2=============:==========2=:====$::::::::2:
C M O D U L E  B L N I N T
C:::=:=::::::=:::::::::::==:::::::::::=::::::::::=:::::::::::=:::::==:
Program RENDER 135
MODULE NONPT

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 BLNH   - REAL ARRAY - H 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==========================================

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

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

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

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

RETURN
END
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 NONPTICOMP, UINT, WINT, BLNU, BLNW, XPT, YPT, ZPT
INTEGER COMP, UINT, WINT
REAL BLNU(4), BLNW(4), XPT, YPT, ZPT
REAL G(3), TEMP(1:4), ROW1(4,4), 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
   ROW1(I,1) = BLNU(I)
   COL(I,1) = BLNW(I)
100 CONTINUE
C LOAD CONTROL VERTICES
DO 300 I = 1,4
   DO 200 J = 1,4
      CALL GTBTF(UINT+I-3, WINT+J-3, 2, COMP, G, IERR)
      XHUL(I,J) = G(1)
      YHUL(I,J) = G(2)
      ZHUL(I,J) = G(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 MODULE SMLBLN
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

Program RENDER
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 SMLBLN1 IPOIN1-7 N1-OT; DIFF• KNTS» INTER, BLEND)
INTEGER IPOINT, N1, 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 NMXSECT - INTEGER - NUMBER OF X-SECTIONS
C UKNTS - REAL ARRAY - U KNOT VALUES
C TOTN - INTEGER — TOTAL NUMBER OF N 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, NKSECT, UKNTS, TOTN, US)
INTEGER COMP, NKSECT, TOTN
REAL UKNTS(*), US(*), TOL
PARAMETER (TOL = 0.00000000001)
PARAMETER (MAXPTS = 50)
REAL TDIST(MAXPTS), UDISTF(MAXPTS), DIST, STEP
REAL P1(3), P2(3), P3(3)

C INITIALIZE DISTANCE ARRAY

Program RENDER 138
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 SUBP(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(TOTN-1)+TOL)

C COMPUTE ADJUSTED PARAMETER SPACING
US(1) = UKNTS(3)
ITOP = 2
DO 200 I=2, TOTN

C COMPUTE REQUIRED DISTANCE
REQD = STEP*FLOAT(I-1)
DO 150 J=ITOP, NKSECT

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 MODULE M DIST
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  HKNTS  - REAL ARRAY - N KNOT VALUES
C  TOTU   - INTEGER - TOTAL NUMBER OF U LINES
C===============================================
C OUTPUT PARAMETERS:
C  NS     - REAL ARRAY - CORRECTLY SPACED N VALUES
C  N E C  - REAL ARRAY - CORRECTLY SPACED N VALUES
C  C O M M  - INTEGER - COMPONENT IDENTIFIER
C  C O M M  - INTEGER - NUMBER OF POINTS PER X-SECTIONS
C  C O M M  - REAL ARRAY - N KNOT VALUES
C  C O M M  - INTEGER - TOTAL NUMBER OF U LINES
C===============================================
C COMMON INPUTS:
C  N E C
C===============================================
C COMMON OUTPUTS:
C  N E C
SUBROUTINE NDIST1COMP, NPPXS, NKNTS, TOTU, NS)

INTEGER COMP, NPPXS, TOTU
REAL NKNTS(*), NS(*), TOL
PARAMETER (TOL = 0.00000000001)

INTEGER MAXPTS
PARAMETER (MAXPTS = 50)
REAL TDIST(MAXPTS), NDISTF(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 H DIST
NKNTD = NKNTS(I+2) - NKNTS(I+1)
IF (NKNTD .LT. TOL) THEN
  DIST = 0.0
  NDISTF(I) = 1.0
ELSE
  C GET POINTS FROM DATA STRUCTURE
  CALL GTBTSF(2, I, 2, COMP, P1, IERR)
  CALL GTBTSF(I+1, 2, COMP, P2, IERR)
  CALL SUB(P2, P1, P3)
  CALL VECVAL(P3, DIST)
  NDISTF(I-1) = DIST/NKNTD
ENDIF
TDIST(I) = DIST+TDIST(I-1)

100 CONTINUE

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

C COMPUTE ADJUSTED PARAMETER SPACING
MS(1) = NKNTS(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 M PARAMETER ARRAY
175 MS(I) = NKNTS(ITOP+2) + (REQD - TDIST(ITOP))/NDISTF(ITOP)

200 CONTINUE
RETURN
END

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

  CALL FINDIT(NKNTS, KNTS, T, INTER)
  CALL BLENDT(T, KNTS, INTER, BLEND)

RETURN
END

SUBROUTINE TILEIT(NKID, NCOMP, COMPID, VIENID, BLEND)
  INTEGER NKID, NCOMP, COMPID(NCOMP), VIENID, BLEND(4)

  CALL BLENDT(NKID, NCOMP, COMPID, VIENID, BLEND)

RETURN
END
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. GLOUBEMANS DATE: 3/15/90
C==========================================================================

SUBROUTINE BSTILE (NKID, NCOMP, COMPID, VIEMID)

INTEGER COMPID(100), VIEMID, NKID
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 HDFIL(COMPS(ICOMP))
ELSE

C LOOP THROUGH ALL RELIMITED PARTS
DO 200 IP = 1, NPRTS

Program RENDER
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 M O D U L E P L S S H D

C PROJECT: NASA/AMES
VERSION: V1.0.0 B-SPLINE TEST

C INPUT PARAMETERS:
C ITCOMP - INTEGER - COMPONENT IDENTIFIER

C OUTPUT PARAMETERS:
C ISFLAG - INTEGER - SHADE INDICATOR FLAG

C COMMON INPUTS:
C NONE

C COMMON OUTPUTS:
C NONE

C FUNCTIONAL DESCRIPTION:
C CHECK SHDIND FILE TO DETERMINE IF COMPONENT SHOULD BE SHADED

C METHODS/ALGORITHMS:
C N/A

C Coded BY: J. R. GLOUDEMANS
            DATE: 3/15/90

SUBROUTINE PLSSHD(ITCOMP, ISFLAG)

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

C INITIALIZE SHADE FLAG
ISFLAG = 0

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

C READ IN NUMBER OF COMPONENTS
READ (NOUT, *) 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 M O D U L E C H K P R T

Program RENDER
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 REMIND FILE
OPEN(UNIT = NOUT, FILE = 'RELIMIT')
REMIN(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)
200 CONTINUE
PFLAG = 1
ELSE

C SKIP RECORDS
DO 150 J=1, NPARTS
    READ(NOUT, *)
150 CONTINUE
ENDIF

100 CONTINUE
SUBROUTINE MKTIBS ( COMP, PART )
INTEGER COMP, PART
INTEGER NXSITM, NUITEM, NSECT, NPPITM, NPPXS, ERRIND
INTEGER NUPOL, NNPOL
PARAMETER(NXSITM = 5, NPPITM = 6, NUITEM = 8, NSECT = 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, NSECT, 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
GET NUMBER OF SURFACE LINES IN THE M DIRECTION
CALL GTICMP(NITEM, ICOMP, NNPOL, ERRIND)

CALCULATE NUMBER OF POLYGONS IN THE M DIRECTION
NNPOL = NNPOL - 1

GENERATE SURFACE POLYGONS
CALL GTIGPS(PART, NXSECT, NPPXS, NUPOL, NNPOL)
RETURN
END

MODULE GTIGPS

PROJECT: NASA/AMES VERSION: V1.0.0 B-SPLINE TEST

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 DIRECTION FOR EACH SURFACE OF THE COMPONENT
C NNPOL - INTEGER; NUMBER OF POLYGONAL ELEMENTS IN THE M DIRECTION FOR EACH SURFACE OF THE COMPONENT

OUTPUT PARAMETERS:
NONE

COMMON INPUTS:
NONE

COMMON OUTPUTS:
NONE

FUNCTIONAL DESCRIPTION:
GENERATES SURFACE POLYGONS FOR A SPECIFIED COMPONENT

METHODS/ALGORITHMS:
N/A

Coded by: J. R. GLODEHAMS DATE: 3/15/90

SUBROUTINE GTIGPS

INTEGER COMP, NXSECT, NPPXS, NUPOL, NNPOL
INTEGER I, J
REAL XYZSUB(3,100000,5)

INTEGER NPSUB(100000), NPARTS, PRTLMT(2,100)
COMMON / POLDAT / NPARTS, PRTLMT, NSUBT, NPSUB, XYZSUB

INCREMENT THE NO. OF PARTS
NPARTS = NPARTS + 1

SET THE INDEX NUMBER FOR THE FIRST POLYGON OF THE NEW PART
PRTLMT(1, NPARTS) = NSUBT + 1

GENERATE POLYGONAL ELEMENTS FOR (I,J)TH SURFACE
CALL GNSPS(PART, NXSECT, NPPXS, NUPOL, NNPOL)

SET THE INDEX NUMBER FOR THE LAST POLYGON OF THE NEW PART
PRTLMT(2, NPARTS) = NSUBT
WRITE(*,*)'COMP, TOTAL POLYGONS =', COMP, NSUBT
RETURN

Program RENDER
SUBROUTINE GNSFBS(COMP, NXSECT, NPPXS, TOTU, TOTH)

INTEGER COMP, NXSECT, NPPXS, TOTU, TOTH
INTEGER MAXKNT
PARAMETER (MAXKNT = 50)
REAL UKNTS(MAXKNT), HKNTS(MAXKNT), TOL
PARAMETER (TOL = 0)
INTEGER NDIVU, NDIVM
REAL BLNU(4), DLMU(4), BLNM(4), DLMN(4)

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

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
  NDIVM = TOTH
  NDIVU = TOTU
  C COMPUTE DIFFERENCE BETWEEN MAX AND MIN U VALUES
  MDIFF = HKNTS(J+1) - HKNTS(J) - TOL
  IF (MDIFF .GT. 0.0) THEN
    DO 100 K=1, NDIVU
    C COMPUTE NEXT U BLENDING FUNCTION
    CALL SMLBLN(K, NDIVU+1, UDIFF, UKNTS, I, DLNM)
    C COMPUTE BLENDING U BLENDING FUNCTIONS
    CALL BLENDT(HKNTS(J), HKNTS, J, BLNM)
  END IF
  END IF
  END DO
END DO
C END SUBROUTINE GNSFBS

Program RENDER
DO 50 L=1, NDIVN
   COMPUTE NEXT U BLENDING FUNCTION
   CALL SMLBLN(L, NDIVN+1, NDIFF, MKNTS, J, DLNM)
C
   DRAW POLYGONS
   CALL DRPOBS(COMP,I,I,J,J,1,BLNJ,DLNJ,BLNM,DLNM)
      DO 25 M=1,4
         BLNM(M) = DLNN(M)
      CONTINUE
25 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================================:=======::2:===========:=:===========
C PROJECT: NASA/AMES VERSION: V1.0.0 B-SPLINE TEST
C====::::::2:::========:=========::::2:::::=====================::2:::
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 DM — REAL; INCREMENTAL VALUE OF M FOR THE SURFACE POLYGONS
C NNMPOL — INTEGER, NUMBER OF POLYGONAL ELEMENTS IN THE M
C DIRECTION FOR EACH SURFACE OF THE COMPONENT
C BMAT(3,4,4) — REAL, SURFACE MATRIX
C==================================::2::2::================2==========================
C OUTPUT PARAMETERS:
C NONE
C3:===============================2==================='-:::::*;:::2:=====
C COMMON INPUTS:
C NONE
C========2::::::7-':=====================:==============================
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+DM,W) ON THE SURFACE. THE THIRD VERTEX IS
C AT THE LOCATION (U+DM,W+DM) ON THE SURFACE. THE FOURTH VERTEX IS
C AT THE LOCATION (U+DM+W) ON THE SURFACE.
C==================================::2::2::================2==========================
C METHODS/ALGORITHMS:
C N/A
C==================================::2::2::================2==========================
C CODED BY: J. R. GLOUDEMANS DATE: 3/15/90
C==================================::2::2::================2==========================
C SUBROUTINE DRPOBS(COMP, IU, DU, IM, DM, NNMPOL, BLNJ, DLNJ, BLNM, DLNM)
C INTEGER COMP, IU, DU, IM, DM, NNMPOL
C REAL BLNJ(4), DLNJ(4), BLNM(4), DLNM(4)
C REAL XYZSUB(3,100000,5)
C 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
C SET THE NUMBER OF VERTICES OF THE POLYGON TO 4
NPSUB(NSUBT) = 4
C
C GENERATE THE FIRST VERTEX OF THE POLYGON AND STORE IT IN COMMON
C BLOCK
CALL POLPT (COMP, IU, IM, NMPOL, BLNJ, BLNM, 1)
C
C GENERATE THE SECOND VERTEX OF THE POLYGON AND STORE IT IN COMMON
C BLOCK
CALL POLPT (COMP, DU, IM, NMPOL, DLNJ, BLNM, 2)
C
C GENERATE THE THIRD VERTEX OF THE POLYGON AND STORE IT IN COMMON
C BLOCK
CALL POLPT (COMP, DU, DW, NMPOL, DLNJ, DLNM, 3)
C
C GENERATE THE FOURTH VERTEX OF THE POLYGON AND STORE IT IN COMMON
C BLOCK
CALL POLPT (COMP, IU, DW, NMPOL, BLNJ, DLNM, 4)
RETURN
END

C=====================================================================
C MODULE POLPT
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 N - REAL; VALUE OF THE PARAMETER 'N'
C NNPOL - INTEGER; NUMBER OF POLYGONAL ELEMENTS IN THE N
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, IM, NMPOL, BLNJ, BLNM, POLYNO)
INTEGER COMP; IU; IM; NMPOL; BLNJ, BLNM, POLYNO
REAL BLNJ(4), BLNM(4), X, Y, Z
REAL XYZSUB(3,100000,5)
INTEGER NPSUB(100000), NPARTS, PRTLMT(2,100)
COMMON / POLDAT / NPARTS, PRTLMT, NSUBT, NPSUB, XYZSUB

Program RENDER 149
C CALCULATE A POINT ON NON-UNIFORM B-SPLINE SURFACE
CALL NONPT(COMP, IU, IW, BLNW, BLMH, X, Y, Z)

C FILL COMMON BLOCK
XYZSUB(1,NSUBT,POLYND) = X
XYZSUB(2,NSUBT,POLYND) = Y
XYZSUB(3,NSUBT,POLYNO) = Z

RETURN
END

C MODULE HRDFIL
C PROJECT: NASA/AMES VERSION: V1.0.0 B-SPLINE TEST
C INPUT PARAMETERS:
C COMP — INTEGER — COMPONENT IDENTIFIER
C OUTPUT PARAMETERS:
C NONE
C COMMON INPUTS:
C NONE
C COMMON OUTPUTS:
C NONE
C FUNCTIONAL DESCRIPTION:
C WRITES THE B-SPLINE CONTROL POINTS AND KNOT VALUES INTO A FILE
C METHODS/ALGORITHMS:
C N/A
C CODED BY: J. R. GLODEMANS DATE: 3/15/90

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, NKSECT, IERR)
CALL GTICMP(6, ICOMP, NPPXS, IERR)

C WRITE COLOR FLAG
WRITE(NOUT,'(A)')'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, NKSECT, NPPXS)
ELSE IF (COMP .GE. 100) THEN
  C  REVERSE CONTROL HULL OUTPUT
  CALL SMHLOD(NOUT, COMP, NXSECT, NPPXS)
ELSE
  C  NORMAL CONTROL HULL OUTPUT
  CALL NRMLOD(NOUT, COMP, NXSECT, NPPXS)
ENDIF
C  CLOSE FILE
CLOSE(NOUT)
RETURN
END

==================================:::2:::==================================:
C  MODULE NRMLOD
C  PROJECT: NASA/AMES  VERSION: V1.0.0  B-SPLINE TEST
C  INPUT PARAMETERS:
C  NOUT — INTEGER — FILE IDENTIFIER
C  COMP — INTEGER — COMPONENT IDENTIFIER
C  NMSECT — INTEGER — NUMBER OF X-SECTIONS
C  NPPXS — INTEGER — NUMBER OF POINTS X-SECTIONS
C  OUTPUT PARAMETERS:
C  NONE
C  COMMON INPUTS:
C  NONE
C  COMMON OUTPUTS:
C  NONE
C  FUNCTIONAL DESCRIPTION:
C  WRITE OUT CONTROL POINTS IN NORMAL ORDER
C  METHODS/ALGORITHMS:
C  N/A
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

Program RENDER
WRITE OUT NUMBER OF H KNOTS
WRITE(NOUT,10)NPPXS+6

COMPUTE AND WRITE FIRST KNOT VALUE
TEMP = MKNTS(1) - (MKNTS(2)-MKNTS(1))
WRITE(NOUT,20)TEMP

WRITE OUT H KNOT VALUES
DO 200 I=1,NPPXS+4
   WRITE(NOUT,20)MKNTS(I)
200 CONTINUE

COMPUTE AND WRITE LAST KNOT VALUE
TEMP = MKNTS(NPPXS+4) + (MKNTS(NPPXS+4)-MKNTS(NPPXS+3))
WRITE(NOUT,20)TEMP

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

RETURN

C=================================================================================================
C MODULE SMHLOD
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 POINTS AND KNOT VALUES
C=================================================================================================
C METHODS/ALGORITHMS:
C N/A
C=================================================================================================
C CODED BY: J. R. GLOUDEMSANS DATE: 3/15/90
C=================================================================================================
SUBROUTINE SMHLOD(NOUT, COMP, NXSECT, NPPXS)
  INTEGER NOUT, COMP, NXSECT, NPPXS, MAXPTS
  PARAMETER (MAXPTS = 50)
  REAL P(3), UKNTS(MAXPTS), MKNTS(MAXPTS)
END
C LOAD KNOT VALUES
CALL LOADKT(COMP; RXSECT; NPPXS; UKNTS; NKNTS)
C WRITE OUT NUMBER OF U KNOTS
WRITE(NOUT;10)NMSECT+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,NMSECT+4
   WRITE(NOUT;20)UKNTS(I)
100 CONTINUE
C COMPUTE AND WRITE LAST U KNOT
TEMP = UKNTS(NMSECT+4) + (UKNTS(NMSECT+4) - UKNTS(NMSECT+3))
WRITE(NOUT;20)TEMP
C WRITE OUT NUMBER OF N KNOTS
WRITE(NOUT;10)NPPXS+6
C COMPUTE AND WRITE OUT FIRST N KNOT
TEMP = 100.0 - NKNTS(NPPXS+4) - (NKNTS(NPPXS+4) - NKNTS(NPPXS+3))
WRITE(NOUT;20)TEMP
DO 200 I=1,NPPXS+4
   TEMP = 100.0 - NKNTS(NPPXS+5-I)
   WRITE(NOUT;20)TEMP
200 CONTINUE
C COMPUTE AND WRITE LAST N KNOT
TEMP = 100.0 - NKNTS(1) + (NKNTS(2) - NKNTS(1))
WRITE(NOUT;20)TEMP
C WRITE CONTROL POINTS
DO 400 I=1,NMSECT+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 MODULE RELIMIT
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:

Program RENDER
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 IULON; IUHIGH; INLON; INHIGH
PARAMETER (NOUT = 66)
REAL P(3)

C OPEN AND RENIND FILE
OPEN(UNIT = NOUT, FILE = 'RELIMIT')
RENIND(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, IULON, IUHIGH, IMLOW, IMHIGH

C PUT IN INFORMATION INTO SUB-COMPONENT
CALL PTICMP(5, ITEMP, IUHIGH-IULON-1, IERR)
CALL PTICMP(6, ITEMP, IMHIGH-IMLOW-1, IERR)

C LOAD IN UNIFORM KNOT SPACING
CALL STRKNT(ITEMP, IUHIGH-IULON-1, IMHIGH-IMLOW-1)

C INITIALIZE COUNTER
KCNT = 0
DO 300 K=IULON, IUHIGH
LCNT = 0
DO 400 L=IMLOW, IMHIGH

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
CLOSE(NOUT)

RETURN
END

Program RENDER
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*ft
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 degrees

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:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.77475119</td>
<td>6.38239956</td>
<td>2.71827698</td>
</tr>
<tr>
<td>3.43466422</td>
<td>5.96624136</td>
<td>2.72647905</td>
</tr>
<tr>
<td>3.08028746</td>
<td>5.53309631</td>
<td>2.73501611</td>
</tr>
<tr>
<td>3.56776049</td>
<td>5.43121243</td>
<td>2.85185339</td>
</tr>
<tr>
<td>2.76596475</td>
<td>4.76079941</td>
<td>2.80285287</td>
</tr>
<tr>
<td>2.57624006</td>
<td>4.08400011</td>
<td>2.88859153</td>
</tr>
<tr>
<td>0.34014575</td>
<td>3.22140980</td>
<td>2.54482913</td>
</tr>
<tr>
<td>-1.85391414</td>
<td>2.88048506</td>
<td>2.76382223</td>
</tr>
<tr>
<td>-2.83218888</td>
<td>2.24523377</td>
<td>3.42578220</td>
</tr>
<tr>
<td>-3.41480541</td>
<td>1.16464567</td>
<td>3.92081761</td>
</tr>
<tr>
<td>-3.93959022</td>
<td>1.63693950</td>
<td>3.78619480</td>
</tr>
<tr>
<td>-4.05959034</td>
<td>1.63693950</td>
<td>3.78619480</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>3.26309490</td>
<td>6.38240004</td>
<td>2.50005341</td>
</tr>
<tr>
<td>2.90318632</td>
<td>5.96624184</td>
<td>2.50005531</td>
</tr>
<tr>
<td>2.52850734</td>
<td>5.53309727</td>
<td>2.50005722</td>
</tr>
<tr>
<td>2.10317228</td>
<td>5.43280326</td>
<td>2.50006352</td>
</tr>
<tr>
<td>1.61049356</td>
<td>4.70567223</td>
<td>2.49652910</td>
</tr>
<tr>
<td>1.08781358</td>
<td>4.13192634</td>
<td>2.51685500</td>
</tr>
<tr>
<td>-0.41059712</td>
<td>2.90164379</td>
<td>2.58822961</td>
</tr>
<tr>
<td>-2.97862935</td>
<td>3.16108108</td>
<td>2.51170383</td>
</tr>
<tr>
<td>-3.34965967</td>
<td>3.17644334</td>
<td>2.53491282</td>
</tr>
<tr>
<td>-3.87988400</td>
<td>3.44416881</td>
<td>2.74643826</td>
</tr>
<tr>
<td>-3.93959022</td>
<td>3.30405859</td>
<td>2.63490554</td>
</tr>
<tr>
<td>-4.05959034</td>
<td>3.30405853</td>
<td>2.63490554</td>
</tr>
</tbody>
</table>

Data for Example Wing-Fuselage Fillet
<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4.00019026</td>
<td>3.99851465</td>
<td>-0.45724016</td>
</tr>
<tr>
<td>-4.05959034</td>
<td>3.99851465</td>
<td>-0.45724016</td>
</tr>
<tr>
<td>4.8164302</td>
<td>6.38239861</td>
<td>2.17016459</td>
</tr>
<tr>
<td>5.1614475</td>
<td>5.9623993</td>
<td>2.15777040</td>
</tr>
<tr>
<td>4.2035478</td>
<td>5.53309536</td>
<td>2.14487028</td>
</tr>
<tr>
<td>4.13682842</td>
<td>5.43234301</td>
<td>2.14102463</td>
</tr>
<tr>
<td>3.60478830</td>
<td>4.75393248</td>
<td>2.12685821</td>
</tr>
<tr>
<td>3.35357475</td>
<td>4.11648130</td>
<td>2.07823515</td>
</tr>
<tr>
<td>1.2294745</td>
<td>3.00479102</td>
<td>2.23265001</td>
</tr>
<tr>
<td>0.18800285</td>
<td>3.58008409</td>
<td>2.01359105</td>
</tr>
<tr>
<td>-2.26461029</td>
<td>3.98575401</td>
<td>0.61864787</td>
</tr>
<tr>
<td>-3.30267055</td>
<td>4.02410555</td>
<td>-0.20868762</td>
</tr>
<tr>
<td>-3.9399022</td>
<td>3.89971590</td>
<td>-0.89095955</td>
</tr>
<tr>
<td>-4.00019026</td>
<td>3.89971590</td>
<td>-0.89095955</td>
</tr>
<tr>
<td>-4.05959034</td>
<td>3.89971590</td>
<td>-0.89095955</td>
</tr>
<tr>
<td>4.8164302</td>
<td>6.38239861</td>
<td>2.17016459</td>
</tr>
<tr>
<td>5.1614475</td>
<td>5.9623993</td>
<td>2.15777040</td>
</tr>
<tr>
<td>4.2035478</td>
<td>5.53309536</td>
<td>2.14487028</td>
</tr>
<tr>
<td>4.13682842</td>
<td>5.43234301</td>
<td>2.14102463</td>
</tr>
<tr>
<td>3.60478830</td>
<td>4.75393248</td>
<td>2.12685821</td>
</tr>
<tr>
<td>3.35357475</td>
<td>4.11648130</td>
<td>2.07823515</td>
</tr>
<tr>
<td>1.2294745</td>
<td>3.00479102</td>
<td>2.23265001</td>
</tr>
<tr>
<td>0.18800285</td>
<td>3.58008409</td>
<td>2.01359105</td>
</tr>
<tr>
<td>-2.26461029</td>
<td>3.98575401</td>
<td>0.61864787</td>
</tr>
<tr>
<td>-3.30267055</td>
<td>4.02410555</td>
<td>-0.20868762</td>
</tr>
<tr>
<td>-3.9399022</td>
<td>3.89971590</td>
<td>-0.89095955</td>
</tr>
<tr>
<td>-4.00019026</td>
<td>3.89971590</td>
<td>-0.89095955</td>
</tr>
<tr>
<td>4.8164302</td>
<td>6.38239861</td>
<td>2.17016459</td>
</tr>
<tr>
<td>5.1614475</td>
<td>5.9623993</td>
<td>2.15777040</td>
</tr>
<tr>
<td>4.2035478</td>
<td>5.53309536</td>
<td>2.14487028</td>
</tr>
<tr>
<td>4.13682842</td>
<td>5.43234301</td>
<td>2.14102463</td>
</tr>
<tr>
<td>3.60478830</td>
<td>4.75393248</td>
<td>2.12685821</td>
</tr>
<tr>
<td>3.35357475</td>
<td>4.11648130</td>
<td>2.07823515</td>
</tr>
<tr>
<td>1.2294745</td>
<td>3.00479102</td>
<td>2.23265001</td>
</tr>
<tr>
<td>0.18800285</td>
<td>3.58008409</td>
<td>2.01359105</td>
</tr>
<tr>
<td>-2.26461029</td>
<td>3.98575401</td>
<td>0.61864787</td>
</tr>
<tr>
<td>-3.30267055</td>
<td>4.02410555</td>
<td>-0.20868762</td>
</tr>
<tr>
<td>-3.9399022</td>
<td>3.89971590</td>
<td>-0.89095955</td>
</tr>
<tr>
<td>-4.00019026</td>
<td>3.89971590</td>
<td>-0.89095955</td>
</tr>
<tr>
<td>4.8164302</td>
<td>6.38239861</td>
<td>2.17016459</td>
</tr>
<tr>
<td>5.1614475</td>
<td>5.9623993</td>
<td>2.15777040</td>
</tr>
<tr>
<td>4.2035478</td>
<td>5.53309536</td>
<td>2.14487028</td>
</tr>
<tr>
<td>4.13682842</td>
<td>5.43234301</td>
<td>2.14102463</td>
</tr>
<tr>
<td>3.60478830</td>
<td>4.75393248</td>
<td>2.12685821</td>
</tr>
<tr>
<td>3.35357475</td>
<td>4.11648130</td>
<td>2.07823515</td>
</tr>
<tr>
<td>1.2294745</td>
<td>3.00479102</td>
<td>2.23265001</td>
</tr>
<tr>
<td>0.18800285</td>
<td>3.58008409</td>
<td>2.01359105</td>
</tr>
<tr>
<td>-2.26461029</td>
<td>3.98575401</td>
<td>0.61864787</td>
</tr>
<tr>
<td>-3.30267055</td>
<td>4.02410555</td>
<td>-0.20868762</td>
</tr>
<tr>
<td>-3.9399022</td>
<td>3.89971590</td>
<td>-0.89095955</td>
</tr>
<tr>
<td>-4.00019026</td>
<td>3.89971590</td>
<td>-0.89095955</td>
</tr>
</tbody>
</table>

Data for Example Wing-Fuselage Fillet
<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.009707373</td>
<td>4.04379988</td>
<td>0.07801699</td>
</tr>
<tr>
<td>5.999999925</td>
<td>3.96080852</td>
<td>-0.68231356</td>
</tr>
<tr>
<td>5.99999857</td>
<td>3.90756226</td>
<td>-0.85520530</td>
</tr>
<tr>
<td>5.99999807</td>
<td>3.89977217</td>
<td>-0.89010870</td>
</tr>
<tr>
<td>5.99999809</td>
<td>3.89164369</td>
<td>-0.92493540</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>10.11766529</td>
<td>6.38239098</td>
<td>2.28239393</td>
</tr>
<tr>
<td>10.02062035</td>
<td>5.96623278</td>
<td>2.27461869</td>
</tr>
<tr>
<td>9.91961384</td>
<td>5.53308773</td>
<td>2.26570606</td>
</tr>
<tr>
<td>10.5672750</td>
<td>5.43065453</td>
<td>2.31560408</td>
</tr>
<tr>
<td>10.2069296</td>
<td>4.76413779</td>
<td>2.27640163</td>
</tr>
<tr>
<td>10.0647587</td>
<td>4.06814683</td>
<td>2.31762981</td>
</tr>
<tr>
<td>10.4826896</td>
<td>3.32693434</td>
<td>2.00901651</td>
</tr>
<tr>
<td>11.85914135</td>
<td>3.83185554</td>
<td>1.40832806</td>
</tr>
<tr>
<td>12.29638481</td>
<td>3.97080278</td>
<td>0.37953548</td>
</tr>
<tr>
<td>15.68132210</td>
<td>4.03245258</td>
<td>-0.04917977</td>
</tr>
<tr>
<td>12.70450783</td>
<td>3.90756226</td>
<td>-0.85520530</td>
</tr>
<tr>
<td>12.70450783</td>
<td>3.89977217</td>
<td>-0.89010870</td>
</tr>
<tr>
<td>12.70450783</td>
<td>3.89164369</td>
<td>-0.92493540</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>11.08759212</td>
<td>6.38238907</td>
<td>2.35446239</td>
</tr>
<tr>
<td>11.02774239</td>
<td>5.96623087</td>
<td>2.34993554</td>
</tr>
<tr>
<td>10.96545029</td>
<td>5.53308582</td>
<td>2.34330153</td>
</tr>
<tr>
<td>10.96403659</td>
<td>5.43027401</td>
<td>2.34711409</td>
</tr>
<tr>
<td>10.90704060</td>
<td>4.76445646</td>
<td>2.32605243</td>
</tr>
<tr>
<td>10.70423944</td>
<td>4.05723953</td>
<td>2.37225530</td>
</tr>
<tr>
<td>10.29899216</td>
<td>3.39795339</td>
<td>2.0158716</td>
</tr>
<tr>
<td>11.60358429</td>
<td>3.60434556</td>
<td>1.67882662</td>
</tr>
<tr>
<td>14.02186012</td>
<td>4.05826141</td>
<td>0.48194847</td>
</tr>
<tr>
<td>14.93281460</td>
<td>3.98700190</td>
<td>-0.55197507</td>
</tr>
<tr>
<td>15.35225582</td>
<td>3.90756226</td>
<td>-0.85520530</td>
</tr>
<tr>
<td>15.35225582</td>
<td>3.89977217</td>
<td>-0.89010870</td>
</tr>
<tr>
<td>15.35225582</td>
<td>3.89164369</td>
<td>-0.92493540</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>11.36456871</td>
<td>6.38238907</td>
<td>2.37506056</td>
</tr>
<tr>
<td>11.31534100</td>
<td>5.96623039</td>
<td>2.37036562</td>
</tr>
<tr>
<td>11.26410484</td>
<td>5.53308582</td>
<td>2.36547923</td>
</tr>
<tr>
<td>11.24883080</td>
<td>5.43079805</td>
<td>2.36540985</td>
</tr>
<tr>
<td>11.19252682</td>
<td>4.76325703</td>
<td>2.35024548</td>
</tr>
<tr>
<td>10.9992532</td>
<td>4.07231970</td>
<td>2.35000020</td>
</tr>
<tr>
<td>11.61539452</td>
<td>3.29921794</td>
<td>2.13556829</td>
</tr>
<tr>
<td>12.07660103</td>
<td>3.54117299</td>
<td>2.02668262</td>
</tr>
<tr>
<td>14.28509900</td>
<td>3.99289823</td>
<td>0.61624444</td>
</tr>
<tr>
<td>15.30061817</td>
<td>4.02337790</td>
<td>-0.20844282</td>
</tr>
<tr>
<td>16.00000381</td>
<td>3.90756226</td>
<td>-0.85520530</td>
</tr>
<tr>
<td>16.00000381</td>
<td>3.89977217</td>
<td>-0.89010870</td>
</tr>
<tr>
<td>16.00000381</td>
<td>3.89164639</td>
<td>-0.92493540</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>11.36456871</td>
<td>6.38238907</td>
<td>2.37506056</td>
</tr>
<tr>
<td>11.31534100</td>
<td>5.96623039</td>
<td>2.37036562</td>
</tr>
<tr>
<td>11.26410484</td>
<td>5.53308582</td>
<td>2.36547923</td>
</tr>
<tr>
<td>11.24883080</td>
<td>5.43079805</td>
<td>2.36540985</td>
</tr>
<tr>
<td>11.19252682</td>
<td>4.76325703</td>
<td>2.35024548</td>
</tr>
<tr>
<td>10.9992532</td>
<td>4.07231970</td>
<td>2.35000020</td>
</tr>
<tr>
<td>11.61539452</td>
<td>3.29921794</td>
<td>2.13556829</td>
</tr>
<tr>
<td>12.07660103</td>
<td>3.54117299</td>
<td>2.02668262</td>
</tr>
<tr>
<td>14.28509900</td>
<td>3.99289823</td>
<td>0.61624444</td>
</tr>
<tr>
<td>15.30061817</td>
<td>4.02337790</td>
<td>-0.20844282</td>
</tr>
<tr>
<td>15.93960190</td>
<td>3.89971590</td>
<td>-0.89009607</td>
</tr>
<tr>
<td>16.00000218</td>
<td>3.89971590</td>
<td>-0.89009607</td>
</tr>
<tr>
<td>16.05960274</td>
<td>3.89971590</td>
<td>-0.89009601</td>
</tr>
</tbody>
</table>

Data for Example Wing-Fuselage Fillet
### Data for Example Wing-Fuselage Fillet

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.8946075</td>
<td>4.08245564</td>
<td>2.38462305</td>
</tr>
<tr>
<td>11.8283582</td>
<td>3.25161250</td>
<td>2.21729955</td>
</tr>
<tr>
<td>12.39414787</td>
<td>3.49871516</td>
<td>2.12602850</td>
</tr>
<tr>
<td>14.46208468</td>
<td>3.49649730</td>
<td>0.70650893</td>
</tr>
<tr>
<td>15.54780865</td>
<td>4.04782581</td>
<td>0.02244188</td>
</tr>
<tr>
<td>15.93960190</td>
<td>3.99851465</td>
<td>-0.45724028</td>
</tr>
<tr>
<td>16.00020218</td>
<td>3.99851465</td>
<td>-0.45724028</td>
</tr>
<tr>
<td>16.05960274</td>
<td>3.99851441</td>
<td>-0.45724025</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.40635681</td>
<td>6.38238811</td>
<td>2.45261025</td>
</tr>
<tr>
<td>12.39707947</td>
<td>5.96622944</td>
<td>2.45082951</td>
</tr>
<tr>
<td>12.38746447</td>
<td>5.53308392</td>
<td>2.44897604</td>
</tr>
<tr>
<td>11.85893967</td>
<td>5.43284798</td>
<td>2.40739487</td>
</tr>
<tr>
<td>12.02540874</td>
<td>4.75078297</td>
<td>2.42599940</td>
</tr>
<tr>
<td>11.80501747</td>
<td>4.13131571</td>
<td>2.37544084</td>
</tr>
<tr>
<td>12.51390839</td>
<td>2.90575528</td>
<td>2.63161778</td>
</tr>
<tr>
<td>13.76750278</td>
<td>3.37849879</td>
<td>2.18467379</td>
</tr>
<tr>
<td>14.84814011</td>
<td>3.83241940</td>
<td>1.62801237</td>
</tr>
<tr>
<td>15.41323853</td>
<td>4.08239625</td>
<td>0.26283246</td>
</tr>
<tr>
<td>15.93960190</td>
<td>4.05532169</td>
<td>0.75249875</td>
</tr>
<tr>
<td>16.00020218</td>
<td>4.05532169</td>
<td>0.75249875</td>
</tr>
<tr>
<td>16.05960274</td>
<td>4.05532169</td>
<td>0.75249875</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.91802120</td>
<td>6.38238621</td>
<td>2.51008892</td>
</tr>
<tr>
<td>12.92835671</td>
<td>5.96622753</td>
<td>2.51046801</td>
</tr>
<tr>
<td>12.93912369</td>
<td>5.53308249</td>
<td>2.51366259</td>
</tr>
<tr>
<td>13.28142262</td>
<td>5.43185663</td>
<td>2.51363707</td>
</tr>
<tr>
<td>13.19340134</td>
<td>4.75679779</td>
<td>2.51062155</td>
</tr>
<tr>
<td>13.23216534</td>
<td>4.10284758</td>
<td>2.52577800</td>
</tr>
<tr>
<td>13.47341633</td>
<td>3.09560728</td>
<td>2.43688917</td>
</tr>
<tr>
<td>14.83822250</td>
<td>3.13547587</td>
<td>2.50464320</td>
</tr>
<tr>
<td>15.75946730</td>
<td>3.18114257</td>
<td>2.55609547</td>
</tr>
<tr>
<td>15.87730885</td>
<td>3.44569149</td>
<td>2.74631763</td>
</tr>
<tr>
<td>15.93601900</td>
<td>3.50408549</td>
<td>2.63490534</td>
</tr>
<tr>
<td>16.00020218</td>
<td>3.50408549</td>
<td>2.63490534</td>
</tr>
<tr>
<td>16.05960274</td>
<td>3.50408549</td>
<td>2.63490510</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.40637875</td>
<td>6.38238764</td>
<td>2.55858561</td>
</tr>
<tr>
<td>12.39710331</td>
<td>5.96622896</td>
<td>2.55795240</td>
</tr>
<tr>
<td>12.38744831</td>
<td>5.53308344</td>
<td>2.50136300</td>
</tr>
<tr>
<td>11.85829212</td>
<td>5.43160534</td>
<td>2.59351778</td>
</tr>
<tr>
<td>12.02961540</td>
<td>4.75833464</td>
<td>2.58538866</td>
</tr>
<tr>
<td>11.78533173</td>
<td>4.09560204</td>
<td>2.59786797</td>
</tr>
<tr>
<td>12.64546863</td>
<td>3.14593210</td>
<td>2.57912087</td>
</tr>
<tr>
<td>13.75011635</td>
<td>2.89072490</td>
<td>2.76379037</td>
</tr>
<tr>
<td>14.85200882</td>
<td>2.24355456</td>
<td>3.42661476</td>
</tr>
<tr>
<td>15.41290855</td>
<td>1.16483748</td>
<td>3.92073274</td>
</tr>
<tr>
<td>15.93601900</td>
<td>1.63603926</td>
<td>3.78619054</td>
</tr>
<tr>
<td>16.00020218</td>
<td>1.63603926</td>
<td>3.78619049</td>
</tr>
<tr>
<td>16.05960274</td>
<td>1.63603926</td>
<td>3.78619004</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.68312931</td>
<td>6.38238811</td>
<td>2.60280943</td>
</tr>
<tr>
<td>11.66411816</td>
<td>5.96622944</td>
<td>2.60667276</td>
</tr>
<tr>
<td>11.60759544</td>
<td>5.53308439</td>
<td>2.61095969</td>
</tr>
<tr>
<td>11.36157937</td>
<td>5.43299055</td>
<td>2.62694168</td>
</tr>
<tr>
<td>11.40201855</td>
<td>4.74992180</td>
<td>2.63504518</td>
</tr>
<tr>
<td>11.18746662</td>
<td>4.13539124</td>
<td>2.61247230</td>
</tr>
<tr>
<td>11.84188080</td>
<td>2.87850701</td>
<td>2.81309539</td>
</tr>
<tr>
<td>12.39258787</td>
<td>2.85099030</td>
<td>2.91732740</td>
</tr>
<tr>
<td>14.46241856</td>
<td>1.57057881</td>
<td>3.70962334</td>
</tr>
<tr>
<td>15.54774313</td>
<td>0.92261153</td>
<td>3.94076112</td>
</tr>
<tr>
<td>15.93601900</td>
<td>0.44588995</td>
<td>4.00001385</td>
</tr>
<tr>
<td>16.00020218</td>
<td>0.44588995</td>
<td>4.00001385</td>
</tr>
<tr>
<td>16.05960274</td>
<td>0.44588995</td>
<td>4.00001385</td>
</tr>
</tbody>
</table>

Data for Example Wing-Fuselage Fillet
<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.26414680</td>
<td>5.53308582</td>
<td>2.63451576</td>
</tr>
<tr>
<td>11.24895287</td>
<td>5.4352137</td>
<td>2.63460255</td>
</tr>
<tr>
<td>11.1920862</td>
<td>4.74839163</td>
<td>2.6497165</td>
</tr>
<tr>
<td>11.00154209</td>
<td>4.14262962</td>
<td>2.62008476</td>
</tr>
<tr>
<td>11.59902573</td>
<td>2.83031750</td>
<td>2.86589932</td>
</tr>
<tr>
<td>12.07856750</td>
<td>2.76174164</td>
<td>2.98498368</td>
</tr>
<tr>
<td>14.28474140</td>
<td>1.48970032</td>
<td>3.75567029</td>
</tr>
<tr>
<td>15.30065441</td>
<td>0.69204175</td>
<td>3.96588287</td>
</tr>
<tr>
<td>15.9360190</td>
<td>0.00000134</td>
<td>4.00001478</td>
</tr>
<tr>
<td>16.00020218</td>
<td>0.00000134</td>
<td>4.00001478</td>
</tr>
<tr>
<td>16.0596274</td>
<td>0.00000134</td>
<td>4.00001478</td>
</tr>
<tr>
<td>11.36460876</td>
<td>6.38238907</td>
<td>2.62493491</td>
</tr>
<tr>
<td>11.31553820</td>
<td>5.9662039</td>
<td>2.6292961</td>
</tr>
<tr>
<td>11.26414680</td>
<td>5.53308582</td>
<td>2.63451576</td>
</tr>
<tr>
<td>11.24895287</td>
<td>5.4352137</td>
<td>2.63460255</td>
</tr>
<tr>
<td>11.1920862</td>
<td>4.74839163</td>
<td>2.6497165</td>
</tr>
<tr>
<td>11.00154209</td>
<td>4.14262962</td>
<td>2.62008476</td>
</tr>
<tr>
<td>11.59902573</td>
<td>2.83031750</td>
<td>2.86589932</td>
</tr>
<tr>
<td>12.07856750</td>
<td>2.76174164</td>
<td>2.98498368</td>
</tr>
<tr>
<td>14.28474140</td>
<td>1.48970032</td>
<td>3.75567029</td>
</tr>
<tr>
<td>15.30065441</td>
<td>0.69204175</td>
<td>3.96588287</td>
</tr>
<tr>
<td>16.0000038</td>
<td>0.03576284</td>
<td>3.99990010</td>
</tr>
<tr>
<td>16.0000038</td>
<td>0.00000143</td>
<td>4.00007248</td>
</tr>
<tr>
<td>16.0000038</td>
<td>-0.03576000</td>
<td>3.99990034</td>
</tr>
<tr>
<td>10.08763589</td>
<td>6.38238907</td>
<td>2.62493491</td>
</tr>
<tr>
<td>10.02778816</td>
<td>5.9662039</td>
<td>2.6292961</td>
</tr>
<tr>
<td>10.96549797</td>
<td>5.53308582</td>
<td>2.63451576</td>
</tr>
<tr>
<td>10.96746892</td>
<td>5.4352137</td>
<td>2.63460255</td>
</tr>
<tr>
<td>10.90527248</td>
<td>4.74839163</td>
<td>2.6497165</td>
</tr>
<tr>
<td>10.71292400</td>
<td>4.14262962</td>
<td>2.62008476</td>
</tr>
<tr>
<td>10.24181080</td>
<td>2.83031750</td>
<td>2.86589932</td>
</tr>
<tr>
<td>14.2067539</td>
<td>2.76174164</td>
<td>2.98498368</td>
</tr>
<tr>
<td>16.0000038</td>
<td>0.69204175</td>
<td>3.96588287</td>
</tr>
<tr>
<td>16.0000038</td>
<td>0.03576284</td>
<td>3.99990010</td>
</tr>
<tr>
<td>16.0000038</td>
<td>0.00000143</td>
<td>4.00007248</td>
</tr>
<tr>
<td>16.0000038</td>
<td>-0.03576000</td>
<td>3.99990034</td>
</tr>
<tr>
<td>10.11771965</td>
<td>6.38238907</td>
<td>2.62493491</td>
</tr>
<tr>
<td>10.02067566</td>
<td>5.9662039</td>
<td>2.6292961</td>
</tr>
<tr>
<td>9.9167201</td>
<td>5.53308582</td>
<td>2.63451576</td>
</tr>
<tr>
<td>10.5678106</td>
<td>5.4352137</td>
<td>2.63460255</td>
</tr>
<tr>
<td>10.20717239</td>
<td>4.74839163</td>
<td>2.6497165</td>
</tr>
<tr>
<td>10.0784203</td>
<td>4.14262962</td>
<td>2.62008476</td>
</tr>
<tr>
<td>10.4013540</td>
<td>2.76174164</td>
<td>2.98498368</td>
</tr>
<tr>
<td>11.8698864</td>
<td>2.22369432</td>
<td>3.90085367</td>
</tr>
<tr>
<td>12.2941071</td>
<td>1.25403202</td>
<td>3.79079080</td>
</tr>
<tr>
<td>13.6815176</td>
<td>0.03576284</td>
<td>3.99990010</td>
</tr>
<tr>
<td>12.70450783</td>
<td>0.00000143</td>
<td>4.00007248</td>
</tr>
<tr>
<td>12.70450783</td>
<td>-0.03576000</td>
<td>3.99990034</td>
</tr>
<tr>
<td>8.09056664</td>
<td>6.38238907</td>
<td>2.62493491</td>
</tr>
<tr>
<td>7.91578579</td>
<td>5.9662039</td>
<td>2.6292961</td>
</tr>
<tr>
<td>7.7387003</td>
<td>5.53308582</td>
<td>2.63451576</td>
</tr>
<tr>
<td>7.6932106</td>
<td>5.4352137</td>
<td>2.63460255</td>
</tr>
<tr>
<td>7.39666605</td>
<td>4.74839163</td>
<td>2.6497165</td>
</tr>
<tr>
<td>7.18607175</td>
<td>4.14262962</td>
<td>2.62008476</td>
</tr>
<tr>
<td>6.35738230</td>
<td>3.02383733</td>
<td>3.16793203</td>
</tr>
<tr>
<td>5.95276692</td>
<td>1.75045073</td>
<td>3.58201432</td>
</tr>
<tr>
<td>6.00868320</td>
<td>0.9868633</td>
<td>3.92085958</td>
</tr>
<tr>
<td>5.99999979</td>
<td>0.2155305</td>
<td>4.01576867</td>
</tr>
<tr>
<td>5.9999995</td>
<td>0.03576204</td>
<td>3.99990010</td>
</tr>
<tr>
<td>5.99999905</td>
<td>-0.03576000</td>
<td>3.99990034</td>
</tr>
</tbody>
</table>

Data for Example Wing-Fuselage Fillet
Data for Example Wing-Fuselage Fillet

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.06341743</td>
<td>6.38259622</td>
<td>2.88800049</td>
</tr>
<tr>
<td>5.81089755</td>
<td>5.96623002</td>
<td>2.90258002</td>
</tr>
<tr>
<td>5.54805995</td>
<td>5.5309545</td>
<td>2.91775513</td>
</tr>
<tr>
<td>4.81865882</td>
<td>5.43555560</td>
<td>2.90693355</td>
</tr>
<tr>
<td>4.58594989</td>
<td>4.7453769</td>
<td>2.93536162</td>
</tr>
<tr>
<td>4.29450464</td>
<td>4.15134671</td>
<td>2.95561294</td>
</tr>
<tr>
<td>2.30650401</td>
<td>2.77165296</td>
<td>2.99269319</td>
</tr>
<tr>
<td>0.03660091</td>
<td>2.21608806</td>
<td>3.39645910</td>
</tr>
<tr>
<td>-0.27720580</td>
<td>1.25427468</td>
<td>3.79154038</td>
</tr>
<tr>
<td>-1.68322921</td>
<td>0.84918702</td>
<td>3.94180989</td>
</tr>
<tr>
<td>-0.70450234</td>
<td>0.0000143</td>
<td>4.00007248</td>
</tr>
<tr>
<td>-0.70450234</td>
<td>-0.03576000</td>
<td>3.99990034</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>5.09349918</td>
<td>6.38239765</td>
<td>2.84980369</td>
</tr>
<tr>
<td>4.80378437</td>
<td>5.96623898</td>
<td>2.86294794</td>
</tr>
<tr>
<td>4.50224400</td>
<td>5.5309441</td>
<td>2.87662888</td>
</tr>
<tr>
<td>4.38921928</td>
<td>5.4355274</td>
<td>2.87496734</td>
</tr>
<tr>
<td>3.89082408</td>
<td>4.7453767</td>
<td>2.89988875</td>
</tr>
<tr>
<td>3.64621544</td>
<td>4.15134671</td>
<td>2.90946564</td>
</tr>
<tr>
<td>1.56006372</td>
<td>2.77274990</td>
<td>3.00067691</td>
</tr>
<tr>
<td>0.28291884</td>
<td>2.62706865</td>
<td>3.07728529</td>
</tr>
<tr>
<td>-2.00098729</td>
<td>1.36970818</td>
<td>3.85318233</td>
</tr>
<tr>
<td>-2.93490791</td>
<td>0.34894413</td>
<td>4.00952387</td>
</tr>
<tr>
<td>-3.35224509</td>
<td>0.03576284</td>
<td>3.99990010</td>
</tr>
<tr>
<td>-3.35224533</td>
<td>0.0000143</td>
<td>4.00007248</td>
</tr>
<tr>
<td>-3.35224533</td>
<td>-0.03576000</td>
<td>3.99990034</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>4.81652689</td>
<td>6.38239813</td>
<td>2.8293947</td>
</tr>
<tr>
<td>4.51619053</td>
<td>5.96623393</td>
<td>2.84223366</td>
</tr>
<tr>
<td>4.20359516</td>
<td>5.5309536</td>
<td>2.85513377</td>
</tr>
<tr>
<td>4.13699150</td>
<td>5.43512740</td>
<td>2.85824132</td>
</tr>
<tr>
<td>3.60413933</td>
<td>4.74516553</td>
<td>2.87774346</td>
</tr>
<tr>
<td>3.35693383</td>
<td>4.13902712</td>
<td>2.90017724</td>
</tr>
<tr>
<td>1.20725596</td>
<td>2.85442567</td>
<td>2.91145253</td>
</tr>
<tr>
<td>-0.18509635</td>
<td>2.75855613</td>
<td>2.97896361</td>
</tr>
<tr>
<td>-2.26514411</td>
<td>1.49028528</td>
<td>3.75977540</td>
</tr>
<tr>
<td>-3.30261564</td>
<td>0.69182089</td>
<td>3.96847057</td>
</tr>
<tr>
<td>-3.99999046</td>
<td>0.03576284</td>
<td>3.99990010</td>
</tr>
<tr>
<td>-3.99999046</td>
<td>0.0000143</td>
<td>4.00007248</td>
</tr>
<tr>
<td>-3.99999046</td>
<td>-0.03576000</td>
<td>3.99990034</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>4.81652689</td>
<td>6.38239813</td>
<td>2.8293947</td>
</tr>
<tr>
<td>4.51619053</td>
<td>5.96623393</td>
<td>2.84223366</td>
</tr>
<tr>
<td>4.20359516</td>
<td>5.5309536</td>
<td>2.85513377</td>
</tr>
<tr>
<td>4.13699150</td>
<td>5.43512740</td>
<td>2.85824132</td>
</tr>
<tr>
<td>3.60413933</td>
<td>4.74516553</td>
<td>2.87774346</td>
</tr>
<tr>
<td>3.35693383</td>
<td>4.13902712</td>
<td>2.90017724</td>
</tr>
<tr>
<td>1.20725596</td>
<td>2.85442567</td>
<td>2.91145253</td>
</tr>
<tr>
<td>-0.18509635</td>
<td>2.75855613</td>
<td>2.97896361</td>
</tr>
<tr>
<td>-2.26514411</td>
<td>1.49028528</td>
<td>3.75977540</td>
</tr>
<tr>
<td>-3.30261564</td>
<td>0.69182089</td>
<td>3.96847057</td>
</tr>
<tr>
<td>-3.99999022</td>
<td>0.0000149</td>
<td>4.00001526</td>
</tr>
<tr>
<td>-4.00019026</td>
<td>0.0000149</td>
<td>4.00001526</td>
</tr>
<tr>
<td>-4.05959034</td>
<td>0.0000149</td>
<td>4.00001526</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>4.49800491</td>
<td>6.38239661</td>
<td>2.80668071</td>
</tr>
<tr>
<td>4.18545341</td>
<td>5.96624041</td>
<td>2.81041230</td>
</tr>
<tr>
<td>3.86014462</td>
<td>5.5309545</td>
<td>2.83041453</td>
</tr>
<tr>
<td>4.02439117</td>
<td>5.43264035</td>
<td>2.85109711</td>
</tr>
<tr>
<td>3.39431262</td>
<td>4.7590747</td>
<td>2.86162567</td>
</tr>
<tr>
<td>3.17056251</td>
<td>1.43079548</td>
<td>2.89451957</td>
</tr>
<tr>
<td>0.96736568</td>
<td>2.90351940</td>
<td>2.84725428</td>
</tr>
<tr>
<td>-0.49927797</td>
<td>2.84692979</td>
<td>2.91290532</td>
</tr>
<tr>
<td>-2.44274971</td>
<td>1.87132399</td>
<td>3.71043644</td>
</tr>
<tr>
<td>-3.54974222</td>
<td>0.92253542</td>
<td>3.94087863</td>
</tr>
</tbody>
</table>
Data for Example Wing-Fuselage Fillet
The vita has been removed from the scanned document