Geometric Trimming and Curvature Continuous Surface Blending for Aircraft Fuselage and Wing Shapes

By

Xijun Wang

Thesis Submitted to the Faculty of

Virginia Polytechnic Institute and State University

In partial fulfillment of the requirements for the degree of

Master of Science

In

Mechanical Engineering

Dr. A. Myklebust, Chairman

Dr. J. H. Bøhn

Dr. R. Mitchiner

February 26, 2001
Blacksburg, Virginia Tech

Keywords: Geometric, Trimming, B-spline, Filleting

Geometric Trimming and Curvature Continuous Surface Blending for Aircraft Fuselage and Wing Shapes

By

Xijun Wang

Dr. Arvid Myklebust, chairman

Mechanical Engineering

(ABSTRACT)

Most of the work accomplished on surface blending is based on visual trimming. In the process of visual trimming, the unwanted portion of a surface is only hidden but not removed. Geometric trimming provides a complete mathematical description of the wanted portion of the trimming surface, and generates a new mathematical surface or sets of surface patches. The new surface is intended to resemble closely the corresponding portion of the original surface. A robust procedure is developed to geometrically trim the intersecting surfaces and blend the trimmed surface patches into one new surface. This research generates a filleting algorithm for surface blending of an aircraft fuselage shape and a wing shape at a closed trimming intersection curve, and verifies the properties of the newly created surface. In order to distinguish how well the new surface approximates the original, an error comparison tool developed in MATLAB has been employed.

Acknowledgements

I would like to thank my advisor, Dr. Arvid Myklebust for his advice and help on my thesis. I also appreciate Dr. Jan Helge Bøhn and Dr. Reginald Mitchiner for their providing advice and taking time as my committee members.

Thanks to Karthik Bindiganavle for taking the time to clarify my confusion.

Special thanks to Ben Poe and Jamie Archual for their prompt help, whenever I got trouble in using MECA Lab computers.

Finally, I want to thank my wife for her constant love and support in my entire graduate study. I want to make the thesis a gift to our son she will soon bear.

Table of Contents

Table of Figures

1.0 Introduction

Computer Aided Geometric Design (CAGD) is described as an approach to approximate and represent curves and surfaces by using computers [Bohm84], while geometric modeling is defined as a collection of methods used to describe geometric characteristics of an object [Mort85]. Geometric modeling is a fundamental component in many CAD applications, such as animations, simulations, and robotics [Roja94]. The advance in these fields is limited by the effectiveness of the geometric models. A number of research studies has been done to describe surface intersection, geometric trimming, error approximation, and surface filleting [Jone91][Roja94][Bind00][Jain99]. Particularly, geometric trimming is an important component in the fields of aerospace, automotive, and shipbuilding industries that require free-form surfaces to be modeled with a certain degree of precision. Figure 1 shows an application of geometric trimming in aircraft design in which the aircraft fuselage intersects the wing at a closed trimming curve. In the process of geometric trimming, the unwanted portion of the fuselage and the wing should be cut off at the closed trimming curve. Since the trimmed curve resulting from the intersection of two cubic surfaces has a very high degree, which will introduce significant numerical error [Jain99], the trimming curve can be approximated by a certain number of cubic curves, in order to keep it within a certain geometric precision.

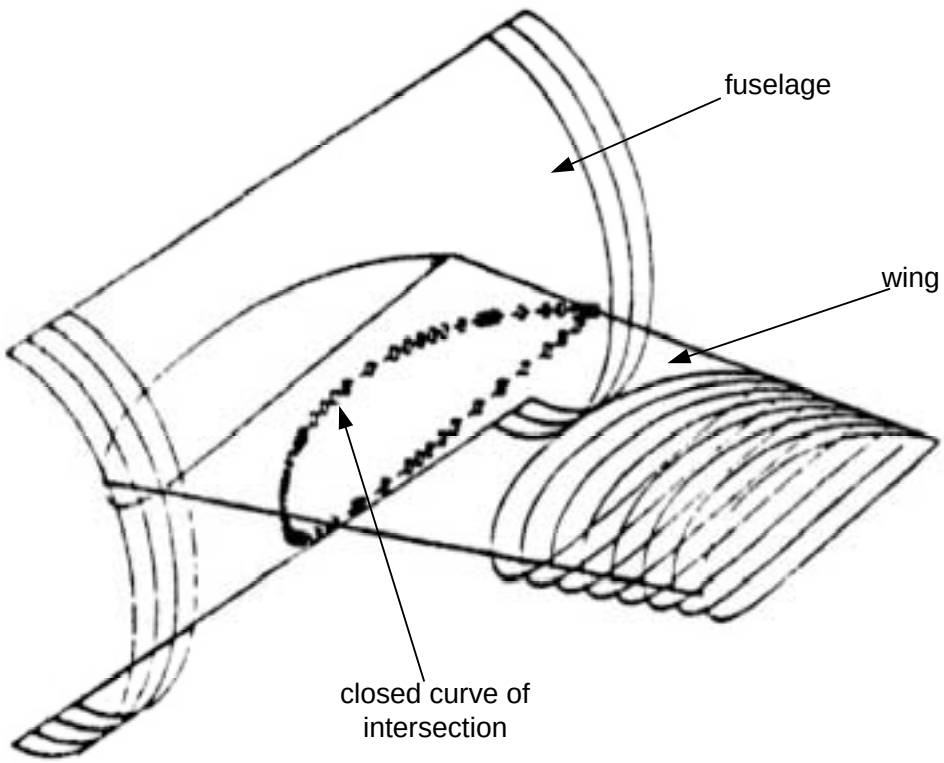


Figure 1 Aircraft wing intersects the fuselage at a closed intersection curve [Appl89]

In geometric trimming, each trimmed surface patch is defined in a parametric domain. It is given a specific mathematical identity rather than being a subset of the original surface as in visual

trimming. Mathematical descriptions of surfaces are very applicable in the calculation of surface areas and volumes of aircraft and automobiles. They are also primary components in manufacturing purposes such as NC programming. Traditionally, surface blending is based on visual trimming. A trimming curve is a two-dimensional curve in a u-v space, in which when a trimming curve is drawn, only the portion on one side of the trimming curve is displayed. After trimming, the unwanted portion of the original surface still exists mathematically. Thus, an analytical result cannot be achieved accurately from the surface modeled. In contrast, surface blending based on geometric trimming has the advantage of CFD computation in the preliminary design phase of aircraft design, in which only wanted data is manipulated. The procedure of geometric trimming and surface blending requires the following steps:

- Define a surface model including two cubic B-spline surfaces resembling the shape of aircraft fuselage and wing.
- Determine the curve of intersection.
- Determine parametric trimming data for each surface respectively.
- Convert the closed trimming curve into two open trimming curves.
- Calculate the offset of each trimming curve in parametric space.
- Trim the surfaces geometrically.
- Relimit unwanted surface patches.
- Subdivide the trimmed surfaces.
- Recreate a new blending surface.

2.0 Research Objectives and Thesis Organization

Surface intersecting, trimming and filleting are performed in almost all solid modeling. Particularly visual trimming has been widely used in surface trimming. As explained in the previous chapter, visual trimming only displays the untrimmed (wanted) portion of the original surface to the viewers. The disadvantage of visual trimming is over-defined in surface definitions. Mathematically, the unwanted portion still exists but it is not shown.

Geometric trimming has always been a challenging topic in the area of CAGD [Roja94]. Generally, the trimming curve resulting from the intersection of two free-form surfaces has a degree that is either unknown or very high in mathematical definition. Rojas used Fleming's constraint-based inversion algorithm to recreate a new B-spline surface, which interpolates the set of data points resulting from reparameterization and mapping [Roja94]. His method assures that the intersection curve has a certain degree after trimming. It has been proved acceptably accurate that the newly created surface with an open trimming curve resembles the original surface for the corresponding portion. Furthermore, Bindiganavle provided an improved approach to prove that this geometric trimming method is very accurate when used in the case of an open trimming curve [Bind00]. However, the algorithm of geometric trimming can only be applied to an open trimming curve. When the surface is trimmed at a closed trimming curve, Fleming's continuity constraint method can not be applied to the trimming boundary. The reason will be addressed in detail later.

The objective of this thesis is to develop a method to trim the model surfaces geometrically in the shape of an aircraft fuselage and a wing and to create a non-uniform cubic B-spline surface that blends the model surfaces. This procedure is outlined below:

- Develop a user interface as a platform to get the visual feedback of the surface trimming and filleting. C++ programming language, 3D graphics standard OpenGL, and xmotif are employed to build the visualization tool kit.
- Create a B-spline surface model including two uniform bicubic B-spline surfaces, one of which represents the fuselage of an aircraft, the other which represents the wing. The data for the model surfaces are generated using SDRC's I-DEAS Master Series 8.
- Determine the intersecting curve of the two surfaces and find the parametric values of the curve in its own parameter space.
- Convert the closed trimming curve into open trimming curves in parameter space for the wing and fuselage, respectively.
- Compute the offset values parametrically for each open trimming curve.
- Trim the surface geometrically along the offset trimming curve.
- Create new surface patches that resemble the parts of the original surfaces without the portions clipped by the trimming curves.
- Subdivide the newly created fuselage based on the fixed control vertices inversion process.
- Implement an error measurement to check the accuracy of the procedure of geometric trimming.
- Blend the trimmed surface patches into one cubic B-spline surface.
- Display the blending surface visually.
- Implement curvature and second derivative analysis to verify the quality of the new surface.

3.0 Literature Review

In the preliminary design phase of aircraft design, a method that automatically produces a fillet is an advanced requirement. Surface intersecting, trimming and filleting are essential procedures for surface operations. In addition, error estimation is needed to check the results from trimming and filleting.

3.1 Surface Intersection Techniques

Computing the intersection curve between two free-form surfaces has been a challenging task. Most of the literature addressing this topic is based on subdivision techniques [Cohe79][Lane80][Dokk85]. Peng [Peng84] and Lasser [Lass86] used "Divide and Conquer" algorithms recursively to subdivide two intersecting surfaces and compute the intersection curve between B-spline or Bezier surfaces. Wong [Wong90] developed a robust surface intersection algorithm by employing algebraic geometry methods to find the intersection curve. One year later, Jones [Jone91] combined Peng's and Lasser's algorithms to generate another robust intersection approach for non-uniform bicubic B-spline surfaces. His method has been successfully tested on several arbitrary B-spline surfaces and incorporated as a part of the ACSYNT B-spline Module. This research does not address means of computing the intersection curve of two surfaces; The intersection data is obtained using SDRC's I-DEAS.

3.2 Surface Trimming

Surface trimming is another key process in surface operations. As stated earlier, surface trimming can be classified in two categories: visual trimming (or graphical trimming) and geometric trimming (or mathematical trimming).

Visual trimming results in hiding the unwanted portion of the original surface from the viewers. It is very useful when no geometrical analysis is required. Visual trimming has been widely used in most engineering surface manipulations. Figure 31 and Figure 32 illustrate the difference between geometric trimming and visual trimming in the surface filleting operation. The topic of visual trimming was discussed by Casale and Bobrow [Casa89a] [Fari87].

Geometric trimming approximates the wanted portion of the original surface with a new mathematically described patch. The issue of geometric trimming was firstly addressed by Hoscheck et al. who divided the parametric space of a given B-spline surface into rectangle subsets, each of which belongs to the parametric domains of individual cubic Bezier patches [Hosc87] [Hosc88] [Hosc89] [Hosc90]. This conversion is accomplished by knot insertion and changing of the basis functions. The cubic Bezeir patches are used to ensure the continuity of second derivative between new adjacent patches. Applegarth [Appl89] developed a new approach to generate the untrimmed patch by clipping each iso-parametric curve lying on a bicubic B-spline surface and recreating the clipped curves to reconstruct the untrimmed patch.

Rojas [Roja94] provided a reasonable solution to the geometric trimming of surface. He used Fleming's constraint-based inversion algorithm to define a new set of B-spline patches which make up the corresponding region of the original surface [Flem92a] [Flem92b]. The performance of his method is affected by the number of sampled data points, the direction and the shape of the trimming curve in parametric space. Better results are achieved when data points defining the trimmed portion of the surface are closely mapped or when the trimming curve is more perpendicular to a specific direction in parametric space. However, his approach could give a good result only in the open trimming curve case. The approximation error is much higher

when a closed trimming curve is used, since continuity constraints could not be specified for the boundary of the closed trimming curve [Roja94].

3.3 Surface Filleting

Typically, surface filleting follows surface trimming in a surface processing procedure. Warren [Warr89] provided a method to blend two algebraic surfaces into one C^2 continuous surface. His approach is only applied to a limited set of algebraic surfaces including cylinders, cones, and spheres. Gloudemans [Glou89] developed an algorithm to blend two arbitrary free-form surfaces into one smooth C^2 continuous surface. Firstly, he used fixed control points to specify the end conditions of the intermediate surface, then found more immediate points to specify the shape of fillet. Jones [Jone91] used a conic shape as the intermediate surface to blend two intersecting surfaces into one C^2 continuous surface.

However, all of the filleting methods mentioned above are based on visual trimming. This research provides a filleting approach based on geometric trimming.

3.4 Error Estimation

Jain [Jain99] developed a comprehensive error estimation tool to compare the difference between two matching surfaces. The plot of positional difference of corresponding points versus parametric value gives the users a good visual feedback about how close the matching surfaces are. Bindiganavle [Bind00] provided an improved approach to compare the closeness of two matching surfaces resulting from geometric trimming. His approach further proved Rojas's method of geometric trimming very accurate when the trimming method is applied in the case of open trimming curve.

4.0 Parametric B-Splines

4.1 Introduction

The term “spline” comes from the drafting tool. The spline is a long strip of flexible material that can be pulled in any direction by using weighted control handles called ducks. The thin strip could pass through any number of points resulting in a gradual change in curvature. Unless severely stressed, it maintains C^2 continuity. The spline shape could physically minimize the strain energy, so that it has widely been used in the fields of computer aided design. Schoenberg first used B-splines for statistical data smoothing in 1946. Gordon and Riesenfeld formally introduced parametric B-splines into computer aided design [Fari97].

4.2 Cubic Parametric Equations

The mathematical description of a curve is essential in CAD applications. A parametric curve is a point-bound collection of points whose coordinates are defined by one parameter function of the form:

$$x = x(t)$$
$$y = y(t)$$
$$z = z(t)$$

The most common parametric polynomial curve is of the third degree, called the cubic curve.
The algebraic form of a parametric cubic curve segment in vector form is defined:

$$\boldsymbol{P}(u) = \sum_{i=0}^{3} C_i u^i \qquad (4.1)$$

where u is a parametric variable, $0 \le u \le 1$.

In the expended Cartesian form:

$$x(u) = C_{3x}u^3 + C_{2x}u^2 + C_{1x}u + C_{0x}$$
$$y(u) = C_{3y}u^3 + C_{2y}u^2 + C_{1y}u + C_{0y}$$
$$z(u) = C_{3z}u^3 + C_{2z}u^2 + C_{1z}u + C_{0z}$$

The 12 constant coefficients are called the algebraic coefficients, which define a unique parametric cubic curve in parametric space.

4.3 The Representation of B-spline Curve

B-spline is the spline built up from several curve segments. The "B" represents the basis, because the spline is a weighted sum of polynomial basis functions. A cubic B-spline curve, $\boldsymbol{P}(u)$, can be defined as the sum of weighted basis functions. It is a piecewise polynomial of degree three with C^2 continuity at the common points between adjacent segments. The cubic B-spline curve results from mapping a knot sequence in parametric space onto Cartesian space. Equation (4.1) gives the mathematical description of an arbitrary order of the B-spline function. The basis functions (N_i) are the functions of the parameter u and they are weighted by control vertices $\boldsymbol{B_i}$.

$$\boldsymbol{P}(u) = \sum_{i=0}^{n} \boldsymbol{B_i} N_i^k(u) \qquad (4.2)$$

where

- u are the parametric variables: $0 \le u \le 1$;
- $\boldsymbol{P}(u)$ are the points along the curve;

- ***B****i* *{ i = 0, 1, 2,…….,n}* are the control points;

- N_i^k (u) are the basis functions;

- *k* is the order of the polynomial segments of the B-spline curve. Order *k* means that the curve is made up of piecewise polynomial segments of degree *k-1*;

Thus, B-splines are completely defined by the control points, the order of curve, and the basis functions.

A recursive mathematical description of the basis function has been provided by Mansfield, deBoor and Cox [Fari97]. The basis functions $N_i^k(u)$ are defined by the formulas:

when $k = 1$:

$$N_i^1(u) = 1 \qquad \text{if } u_i \le u \le u_{i+1}$$

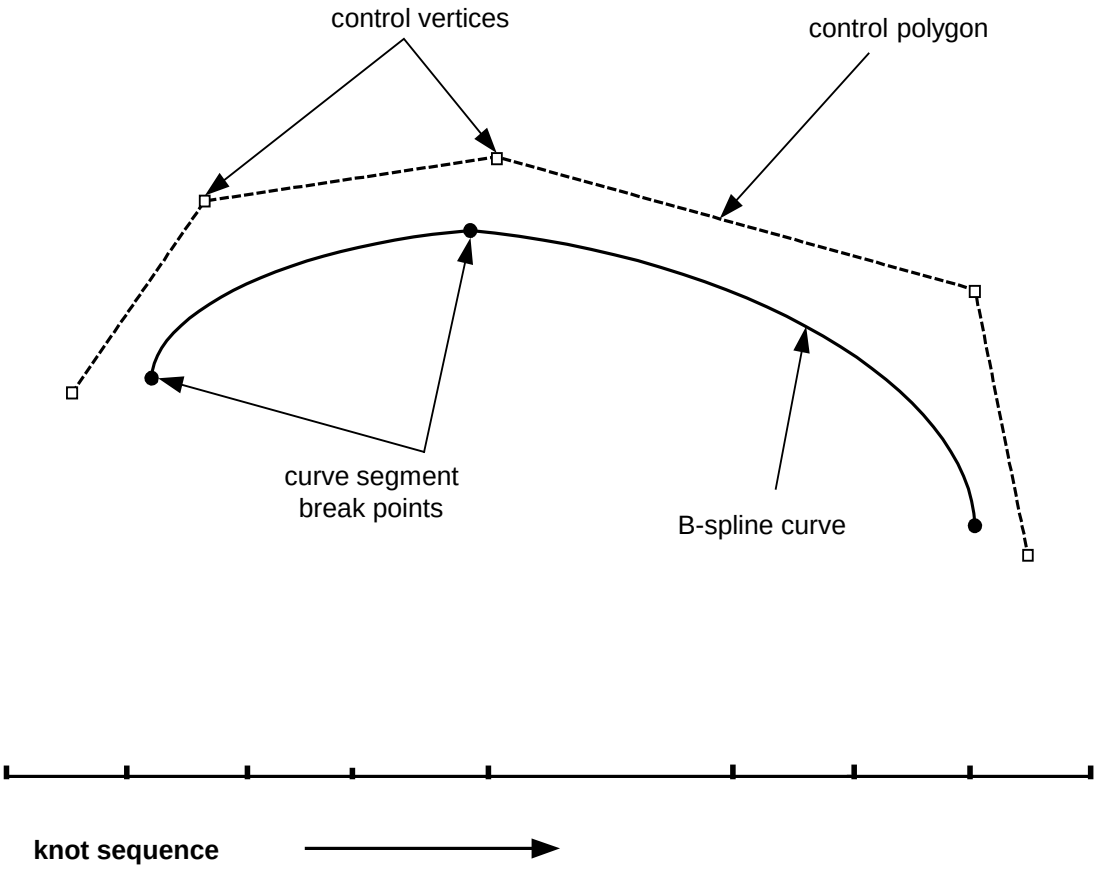
$$N_i^1(u) = 0 \qquad \text{otherwise} \qquad (4.3)$$

when $k \ge 1$:

recursively $$N_i^k(\mathrm{u}) = \frac{u - u_i}{u_{i+k-1} - u_i} N_i^{k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1}^{k-1}(u) \qquad (4.4)$$

where *u* is a parametric value between u_i and u_{i+1} in the knot sequence. The distinct feature of the basis function is the knot sequence u_i, which is a non-decreasing sequence of real numbers{ u_i : *i = 0, 1,……, n+k* }. As shown in Figure 2 below, a B-spline is defined by a set of control points. The B-spline curves approximate the control vertices, but do not pass through them. The control

vertices are connected each other to make up the control polygon. The general shape of the B-spline curve is taken from the shape of control polygon.



number of curve segments: 2

● number of break points: n = 3

□ number of control vertices: n + 2 = 5

▮ number of knots: n + 6 = 9

Figure 2 Cubic B-spline curve with its control polygon

For a cubic B-spline segment, the basis functions are made up of four continuous polynomial segments over the parameter range $[u_i, u_{i+4}]$ as shown in Figure 3. If the knots are evenly spaced in parametric space, a uniform cubic B-spline basis function is produced. If the knot sequence is spaced with unequal intervals, the basis functions of non-uniform B-splines are generated. Therefore, a uniform B-spline curve has the distinct feature of having the same blending function coefficients throughout the entire curve. The formula for a uniform B-spline can be simplified to the following form:

$$P_i(u) = U\,M\,B \tag{4.5}$$

where

- $P_i(u)$ are the points on the B-spline curve
- U is the parametric variable matrix: $U = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$
- M is the matrix of basis function: $M = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$
- B is the matrix of control vertices: $B = \begin{bmatrix} B_{i-1} \\ B_i \\ B_{i+1} \\ B_{i+2} \end{bmatrix}$

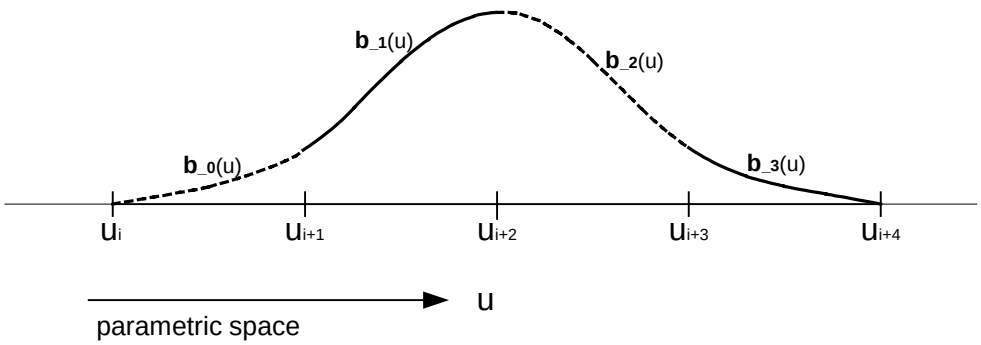


Figure 3 A single uniform cubic B-spline basis function

Figure 4 plots the blending functions for one curve segment of a uniform cubic B-spline curve. Each basis function is defined over four successive knot intervals. The section of the basis function defined over each interval is defined as a basis segment. Each basis function has four basis segments, $\mathbf{b}_0$, $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$. Each B-spline basis function is a shifted copy of the B-spline basis function. The basis functions are always positive and sum to 1 at any parametric value between 0.0 and 1.0.

The B-spline curve is a piecewise combination of curve segments. Each of which is a spline curve defined by a certain number of corresponding control vertices and basis functions. Each cubic B-spline segment is defined by only four adjacent control vertices. Thus, N control vertices define a cubic B-spline curve with N-3 curve segments.

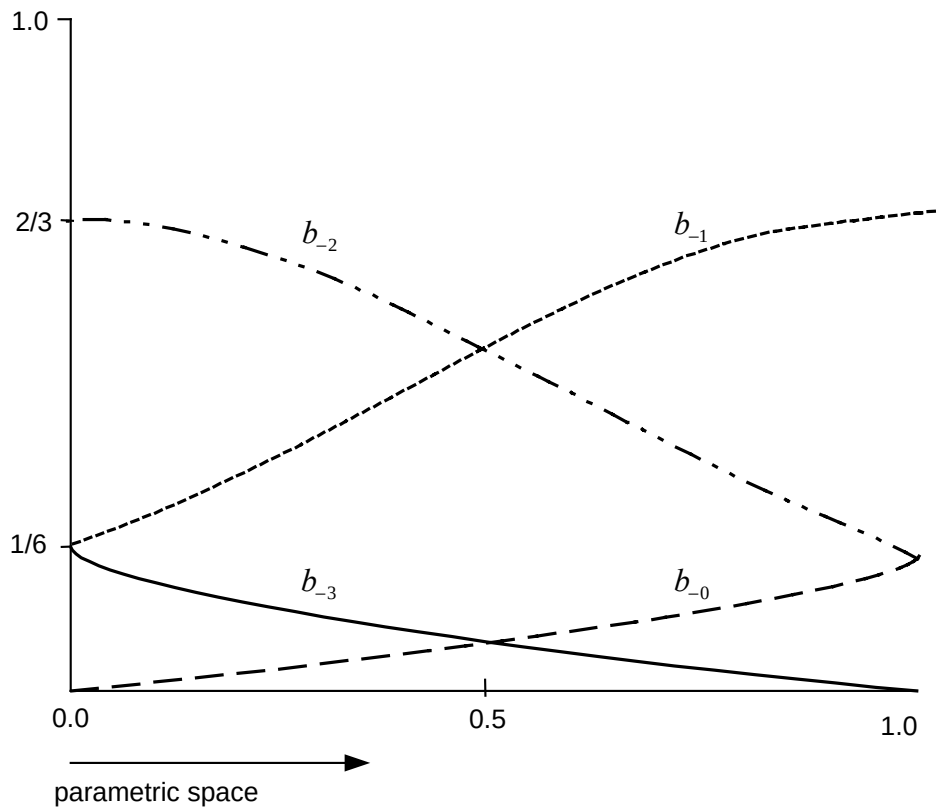


Figure 4 Uniform cubic B-spline blending functions

As addressed earlier, a B-spline can be classified to two categories: uniform and non-uniform, depending on the spacing of the knot sequence. If the intervals between all knots are equal, the curve is uniform, and its blending functions for each curve segment are the same, but the parametric value u is different in the range from 0.0 to 1.0. If the knots are unevenly spaced in the knot sequence, the curve is called a non-uniform B-spline. Its blending functions are no longer identical. For this case, Equations 4.3 and 4.4 are to compute the basis functions.

The parameterization technique of B-spline curves is very important for modeling B-splines. Farin discussed the different parameterization methods for non-uniform B-splines in detail [Fari97]. The methods include Centripetal, Chord length, and Foley. Parametric cubic B-splines are used in this thesis, since C^2 continuity is a key surface property required for the surface operation in the research.

4.4 Knot Insertion

Knot insertion is a simple way of mapping more break points to a B-spline curve without changing the shape of the curve. De Boor [DeBo72] provided the first algorithm on knot insertion for B-Splines. Yamaguchi summarized the algorithm of knot insertion, which involves inserting new knots in the knot sequence, calculating a new set of control points, and computing the corresponding break points [Yama88]. It results in more curve segments for the description of the original curve. This property is very useful in subdividing a curve or surface into multiple components.

The equations of knot insertion for a cubic B-spline curve:

$$Q_j = (1-\alpha_j)\, q_{j-1} + \alpha_j\, q_j$$

$$\text{where:} \quad \alpha_j = \begin{cases} 1 & (j \le i-3) \\ \dfrac{t-t_j}{t_{j+3}-t_j} & (i-2 \le j \le i) \\ 0 & (j \ge i+1) \end{cases} \qquad (4.6)$$

i is the interval where the knot is inserted

j is the index of array: { 0, 1, 2, 3 }

q_j is the original control vector

Q_j is the new control vector

The following example explains the simplest case of knot insertion for a non-uniform cubic B-spline curve:

The knot sequence given: T = [t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7] = [0 1 2 3 4 5 7 9]

The original control points: { q_0 q_1 q_3 q_4 }

A knot is inserted at $t = 3.5$

j = { 0, 1, 2, 3}

The interval where the knot is inserted: $i = 3$ since $t_3 \leq t \leq t_4$

Using equation 4.5 to calculate $\alpha_j = \begin{cases} 1 & (j \leq 0) \\ \dfrac{t - t_j}{t_{j+3} - t_j} & (1 \leq j \leq 3) \\ 0 & (j \geq 4) \end{cases}$

$$\alpha_0 = 1.0$$

$$\alpha_1 = \frac{t - t_1}{t_4 - t_1} = \frac{3.5 - 1}{4 - 1} = 0.8333$$

$$\alpha_2 = \frac{t - t_2}{t_5 - t_2} = \frac{3.5 - 2}{5 - 2} = 0.5$$

$$\alpha_3 = \frac{t - t_3}{t_6 - t_3} = \frac{3.5 - 3}{7 - 3} = 0.125$$

$$\alpha_4 = 0.0$$

The new control points are calculated:

$$Q_0 = (1 - \alpha_0)\ q_{-1} + \alpha_0\ q_0 = q_0$$

$$Q_1 = (1 - 0.8333)\ q_0 + 1.0\ q_1$$

$$Q_2 = (1 - 0.5)\ q_1 + 0.8333\ q_2$$

$$Q_3 = (1 - 0.125)\ q_2 + 0.5\ q_3$$

$$Q_4 = (1 - 0.0)\ q_3 = q_3$$

Therefore, if the control vertices and knot sequence are known, when inserting knots in the knot sequence, the new control polygon can be figured out without changing the shape of the curve. Knot insertion results in more curve segment break points which correspond to the knots inserted. This algorithm can be extended to surface refinement. Figure 5 illustrates the knot insertion for a non-uniform cubic B-spline curve.

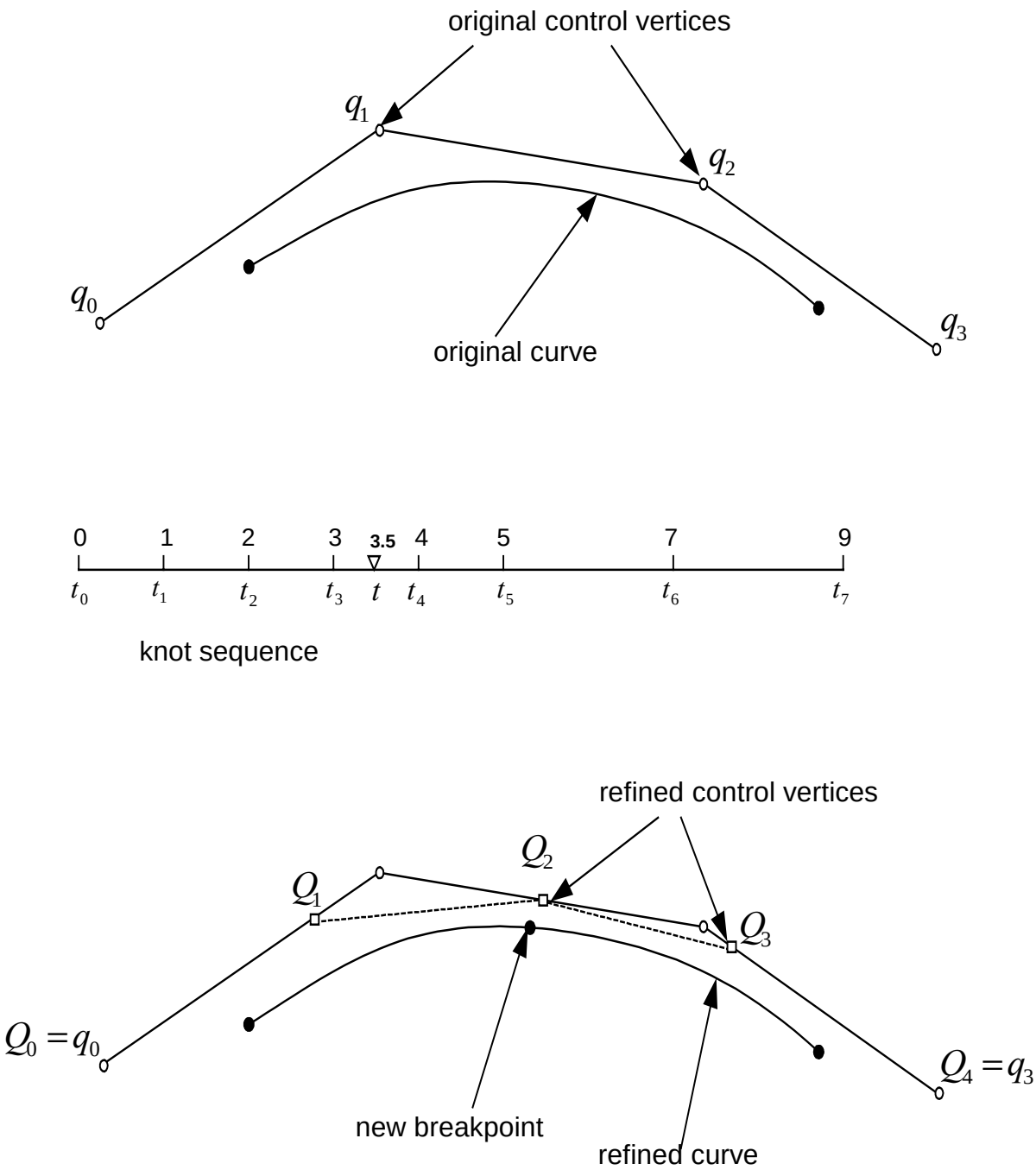


Figure 5 Knot insertion for a non-uniform cubic B-spline curve

4.5 B-Spline Surfaces

Basically, a B-spline surface is the two-dimensional formulation of a B-spline curve. A tensor product surface is the most common type of B-spline surfaces, and its basis functions are the products of two-dimensional B-spline blending functions. As shown in Figure 6, a set of control points of a B-spline surface form a control net that approximates and controls the shape of a B-spline surface. Mathematically a tensor product B-spline surface is defined:

$$\boldsymbol{P}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \boldsymbol{B}_{ij} N_i^k(u) N_j^l(v) \qquad (4.7)$$

Where

- $\boldsymbol{P}(u, v)$: Points on the surface
- $\boldsymbol{B}_{ij}$: Control points
- $N_i^k(u)$: ith basis function of order k
- $N_j^l(v)$: jth basis function of order l

The basis functions are the same as the equation (4.3) and (4.4):

when $k = 1$:

$N_i^1(u) = 1$ if $u_i \le u \le u_{i+1}$

$N_i^1(u) = 0$ otherwise

when $k \ge 1$:

$$N_i^k(u) = \frac{u - u_i}{u_{i+k-1} - u_i} N_i^{k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1}^{k-1}(u) \qquad (4.8)$$

when $l = 1$:

$N_j^1(v) = 1$ if $v_j \le v \le v_{j+1}$

$N_j^1(v) = 0$ otherwise

when $l \ge 1$:

$$N_j^l(v) = \frac{v - v_j}{v_{j+l-1} - v_j} N_j^{l-1}(v) + \frac{v_{j+1} - v}{v_{j+l} - v_{j+1}} N_{j+1}^{l-1}(v) \qquad (4.9)$$

As shown in Figure 6, the bicubic B-spline surface is known as a tensor product surface. Each individual iso-parametric curve on the surface is a B-spline curve on its own. For example, each iso-parametric curve of constant v is a B-spline curve defined by equation 4.10:

$$\boldsymbol{P}_i(u) = \sum_{i=0}^{n} \boldsymbol{B}_i N_i^k(u) \qquad (4.10)$$

where $\boldsymbol{B}_i$ are the control points of B-spline curve at the constant v parametric value.

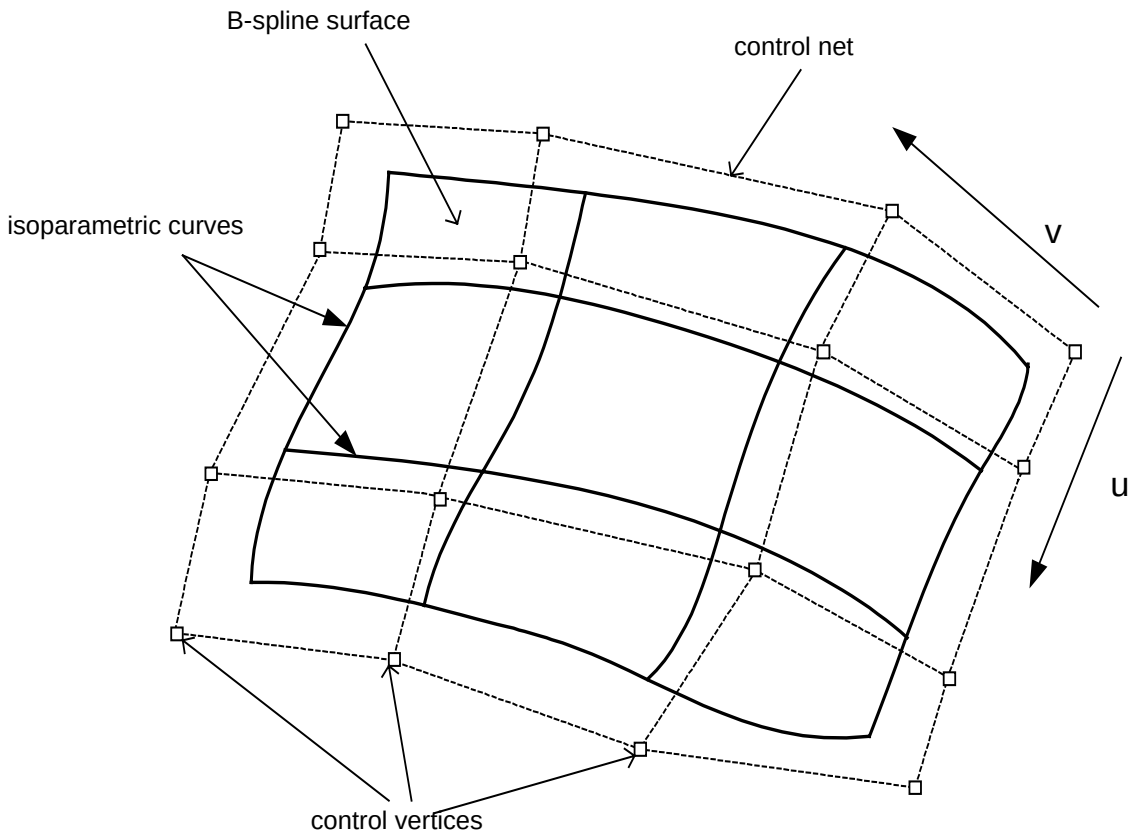


Figure 6 Bicubic B-spline surface with its control net

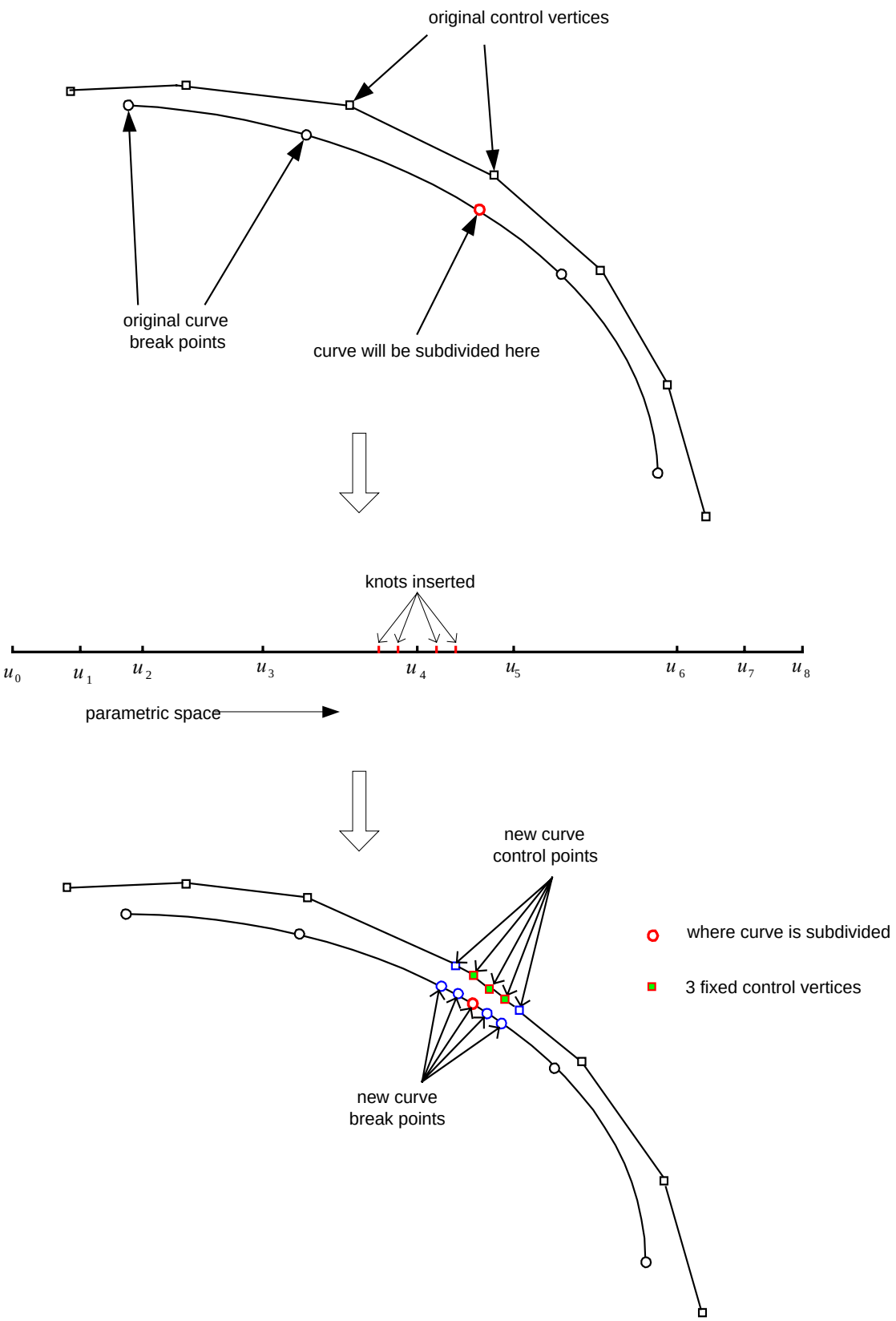
Similar to a B-spline curve that is generally made up of piecewise curve segments, a B-spline surface is composed of polynomial patches. In the case of a non-uniform bicubic B-spline surface, the basis functions are evaluated recursively as in the description of Equation 4.4. The following matrix formula defines a non-uniform bicubic B-spline surface:

$$P_{ij}(u, v) = \begin{bmatrix} N_0^4(u) & N_1^4(u) & N_2^4(u) & N_3^4(u) \end{bmatrix} \begin{bmatrix} B_{i-1,j-1} & B_{i-1,j} & B_{i-1,j+1} & B_{i-1,j+2} \\ B_{i,j-1} & B_{i,j} & B_{i,j+1} & B_{i,j+2} \\ B_{i+1,j-1} & B_{i+1,j} & B_{i+1,j+1} & B_{i+1,j+2} \\ B_{i+2,j-1} & B_{i+2,j} & B_{i+2,j+1} & B_{i+2,j+2} \end{bmatrix} \begin{bmatrix} N_0^4(v) \\ N_1^4(v) \\ N_2^4(v) \\ N_3^4(v) \end{bmatrix} \tag{4.11}$$

Gloudeman's non-uniform B-spline inversion algorithm is based on this equation [Glou89]. Fleming provided a constraint-based inversion method to assist the inversion algorithm [Flem92].

4.6 Fixed Control Vertex Inversion

Fixed control vertices inversion [Glou89], combined with knot insertion, provides a method to subdivide one surface into two surfaces. Figure 7 illustrates the subdivision algorithm for a cubic B-spline curve. Two knots are evenly inserted on the both side of the knot that corresponds to the curve subdividing point. Thus three control vertices, which are defined as fixed control vertices, are obtained from the five evenly spaced knots. The three fixed control vertices require the curve to keep the original shape after the curve is divided into two pieces. In this research, as shown in Figure 40, the trimmed fuselage can be subdivided by the algorithm described above.


original control vertices
original curve
break points
curve will be subdivided here
knots inserted
u_0
u_1
u_2
u_3
u_4
u_5
u_6
u_7
u_8
parametric space
new curve
control points
where curve is subdivided
3 fixed control vertices
new curve
break points

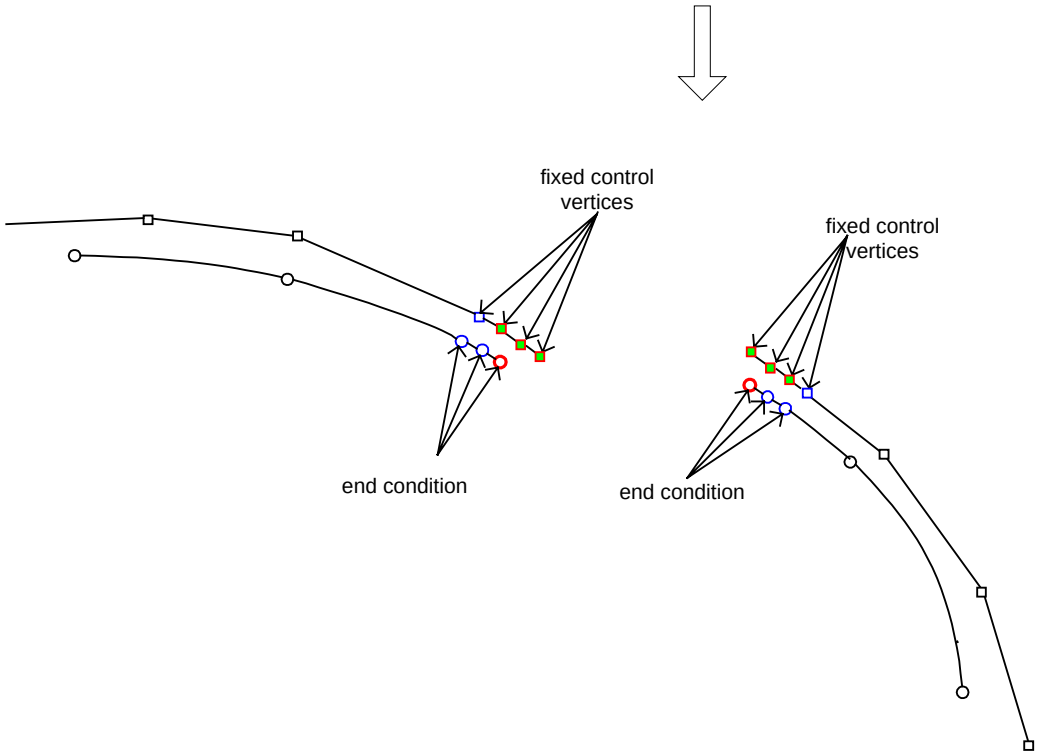


Figure 7 Fixed control vertex inversion for curve subdivision

4.7 Constraint-Based B-Spline Inversion

The B-spline inversion process involves determining the control vertices for a B-spline curve that passes through a given set of points. Yamaguchi developed an inversion method to find the control polygon for a uniform B-spline curve [Yama88]. Gloudemans extended the process to the case of a non-uniform B-spline curve and surface [Glou89]. Typically, the B-spline inversion process consists of setting up and solving a system of linear equations, which are composed of the basis functions and the continuity constraints of end conditions [Flem92a] [Flem92b].

In the case of cubic B-spline curve, as explained earlier, the basis functions are locally non-zero. Each curve segment is affected by at most four adjacent control vertices and four basis functions. However, only three basis functions are non-zero at each knot value u_I. Thus, each value of the points on the cubic B-spline curve can be solved by:

$$P_i(u_{i+2}) = N^4_{i-1}(u_{i+2})\,B_{i-1} + N^4_i(u_{i+2})\,B_i + N^4_{i+1}(u_{i+2})\,B_{i+1} \qquad (4.12)$$

$$\{\ i = 1, 2, \ldots\ldots, n\ \}$$

P_i is the point value on the curve. There is n number of data points known. Thus, only n number of equations can be set up. In order to solve $n+2$ control points, two more equations need to be specified. Generally, curve end conditions can provide the extra constraints to make up to $n+2$ equations. With the addition of the two end conditions, the number of equations is equal to the number of unknown control points. Therefore, the $n+2$ control points can be solved. The end conditions are typically specified as tangency condition.

The inversion process is generalized to a matrix form shown below:

$$\mathbf{P} = \mathrm{N}\,\mathbf{B}$$

$$\begin{bmatrix} end \\ P_1 \\ P_2 \\ \cdot \\ \cdot \\ \cdot \\ P_n \\ end \end{bmatrix} = \begin{bmatrix} end & condition & & & & & 0 \\ N^4_0(u_3) & N^4_1(u_3) & N^4_2(u_3) & & & & \\ & N^4_1(u_4) & N^4_2(u_4) & N^4_3(u_4) & & & \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & & \cdot & & \\ & & & N^4_{n-1}(u_{n+2}) & N^4_n(u_{n+2}) & N^4_{n+1}(u_{n+2}) & \\ 0 & & & & end & condition & \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ \cdot \\ \cdot \\ \cdot \\ B_n \\ B_{n+1} \end{bmatrix} \qquad (4.13)$$

Tensor product B-spline surface inversion is an extension of B-spline curve inversion. It is to find the rectangular array of control points which approximate a given set of data points with respect to two different blending functions. The inversion process can be expressed as following matrix form:

$$\mathbf{P} = N(v)\ \mathbf{B}\ N^T(u) \tag{4.14}$$

where **P** is the matrix of data points and end conditions. $N(v)$ and $N^T(u)$ are the matrix of blending functions as the functions of v and u respectively. B is the array of control points to be solved. As stated in the previous section, a B-spline blending function of order k is $k-2$ times differentiable at any points on the B-spline curve if a knot does not coincide with any other knot. Therefore, a cubic B-spline basis function can be mathematically differentialed into a lower order polynomial which can be used to represent C^1, C^0 or no continuity in the geometry of a B-spline curve. However, the knot multiplicity algorithm can implement a similar continuity constraint change [Fari97]. Fleming provided a method to incorporate these continuity constraints into the cubic B-spline geometry model [Flem92]. In his approach, knot multiplicity is employed to add multiple knots at the points where the continuity will be changed. For a B-spline curve of order k, if a knot has multiplicity of m, the B-spline exhibits C^{k-m-1} continuity at the particular knot. In the case of cubic B-spline, adding one extra knot will change the continuity from C^2 to C^1 at the specified point. If two additional knots are added, the continuity will be changed from C^2 to C^0. Three knots are added to make the curve split. Figure 8 shows the continuity change with respect to knot multiplicity. Fleming's inversion process firstly specifies the parameterization of the knot sequence with one of the following techniques: Uniform, Chord length or Foley. Secondly, one or more knots are added into the specific knot sequence where the continuity constraints should be specified. Thirdly, the new blending functions are evaluated from the new set of knots. Finally a new linear system including continuity constraints and basis functions can be assembled and solved.

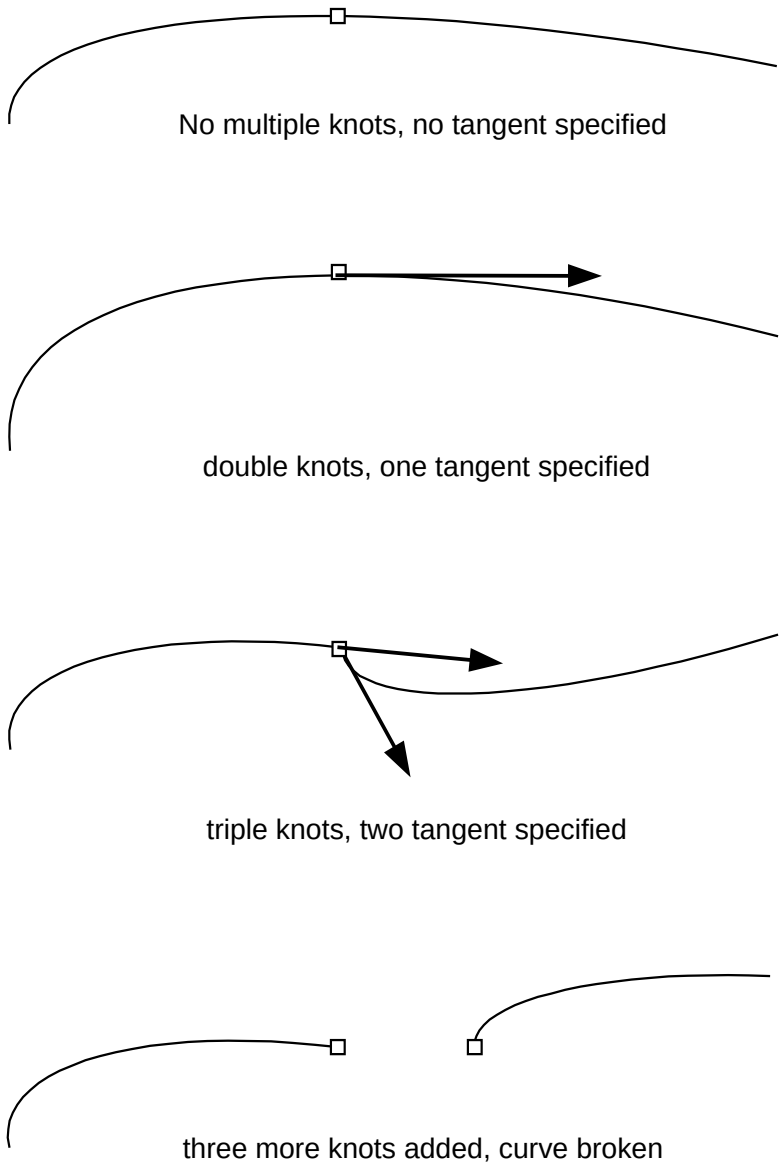


Figure 8 Effect of adding multiple knots to a cubic B-spline curve

5.0 Geometric Trimming

5.1 Theory of Geometric Trimming

Geometric modeling is defined as a collection of methods which are used to describe the geometric characteristics of an object [Roja94]. As stated in the previous chapter, geometric trimming is a key component in geometric modeling. It provides a suitable approach which mathematically approximates the untrimmed portion of the trimming surface. The beauty of the process is that the trimming curve keeps a certain degree and the newly approximated patch is mathematically fully defined. It caters to the requirements of surface analysis for exact mathematical representation of a surface and it is useful for the calculation of surface areas and object volume. In this research, the surface models are bicubic B-spline surfaces. Similar to the cubic B-spline curve as described earlier, one surface segment on a bicubic B-spline surface is affected by the surrounding sixteen control vertices. In order to map a point onto the surface it is necessary to know the control vertices which affect the point on the single surface patch. As shown in Figure 9, if 16 defining vertices are given, it is easy to calculate any points on the single surface patch. Rojas [Roja94] described the topic of geometric surface trimming in detail. In his approach the control points defining the trimming surface need to be found first, then the set of data points on the wanted portion of the trimming surface can be calculated and interpolated into a new surface patch by using Fleming's constraint-based inversion algorithm [Flem92].

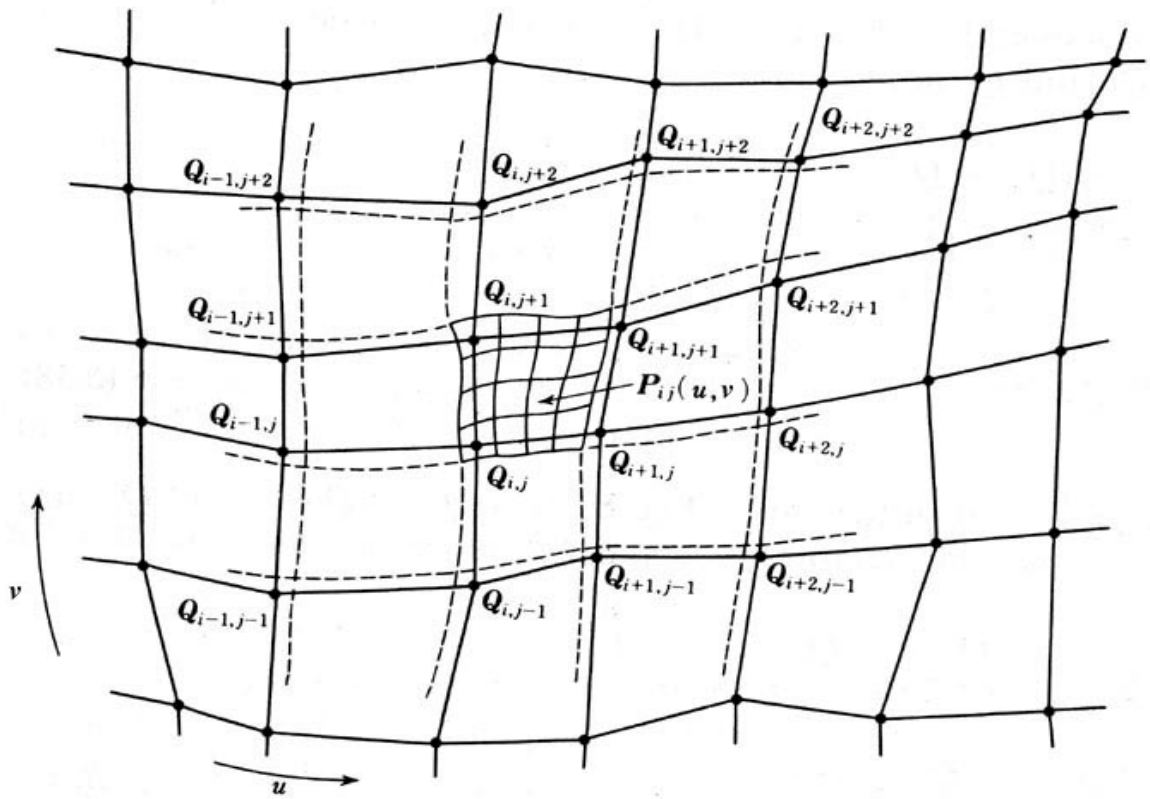


Figure 9 Bicubic B-spline surface patch and its control net [Yama88]

From the example shown in Figure 10, a trimming curve can be simplified as a set of points in both model surface and parametric space. The trimming curve intersects the equidistant *u* iso-parametric curves. Then each of the *u* iso-parametric curves is clipped into two pieces one of which is preserved as the iso-parametric curve in the newly created patch, and the other one is chopped off. Once all these curves are truncated at the trimming curve, the retained portion of each curve can be rebuilt by a set of data points, which are found by subdividing the parametric domain of the retained portion and mapped from the parametric values that come from the subdivision.

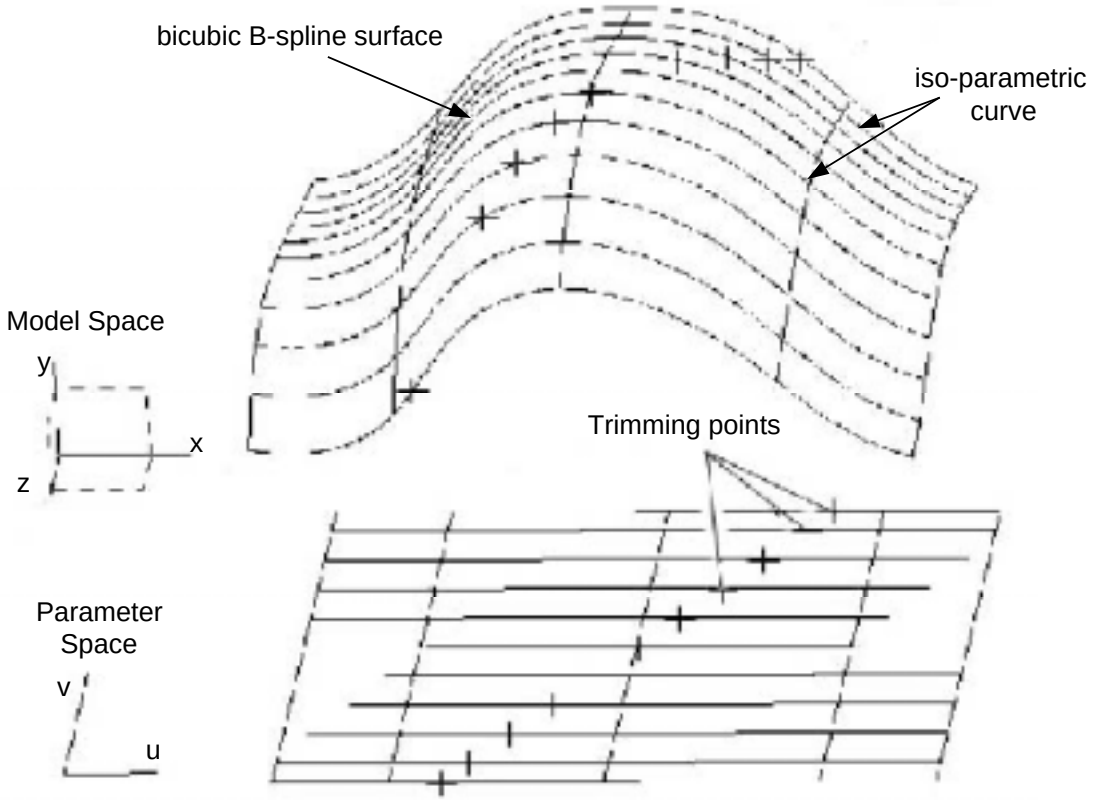


Figure 10 *u* iso-parametric curves are trimmed in both model and parametric space [Bind00]

Figure 11 shows how the parametric domain of the original surface is subdivided along the trimming curve in parametric space. In the *u* parametric direction, every truncated parametric line is evenly subdivided by the number of points defining the original surface in the same parametric direction. The number of subdivisions for the *v* parametric direction is equal to the number of points used to describe the trimming curve. Once the parametric values from the subdivision in both parametric directions are obtained, the corresponding data points on the model surface can be calculated and they are used to describe the retained patch. Apparently the more surface points are used to define the original surface and to describe the trimming curve, the more accurate the newly created patch would match the original surface. But this approach may sacrifice computational speed. It depends on how many points are used to define the trimming

surface. Generally, if the trimming curve has a complex shape, it is necessary to use more points to represent the trimming curve. If the surface shape is more irregular, then more points need to be specified to describe the original surface.

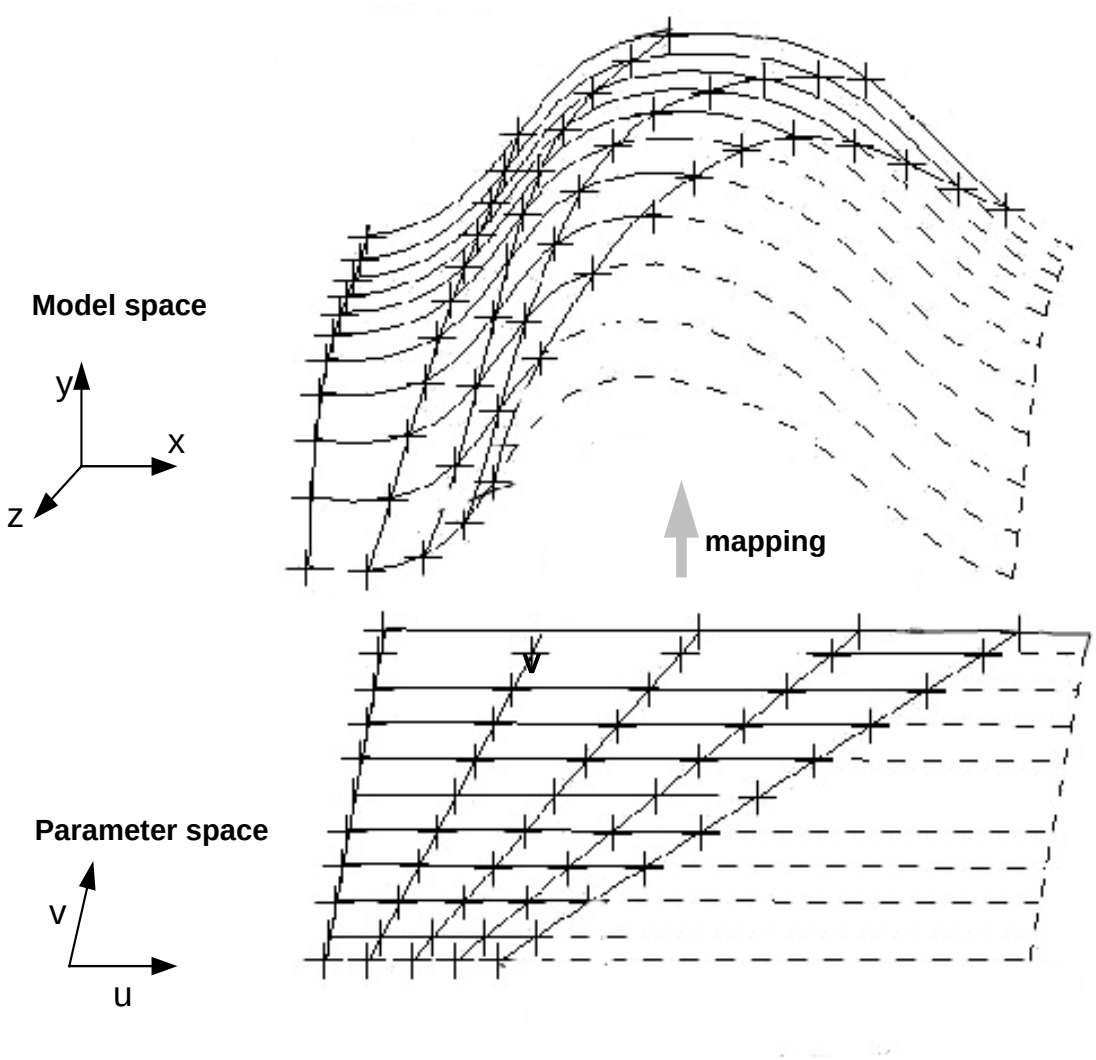


Figure 11 Subdividing parametric domain and mapping the trimming surface [Bind00]

Figure 12 shows the data points obtained to describe the untrimmed surface patch.

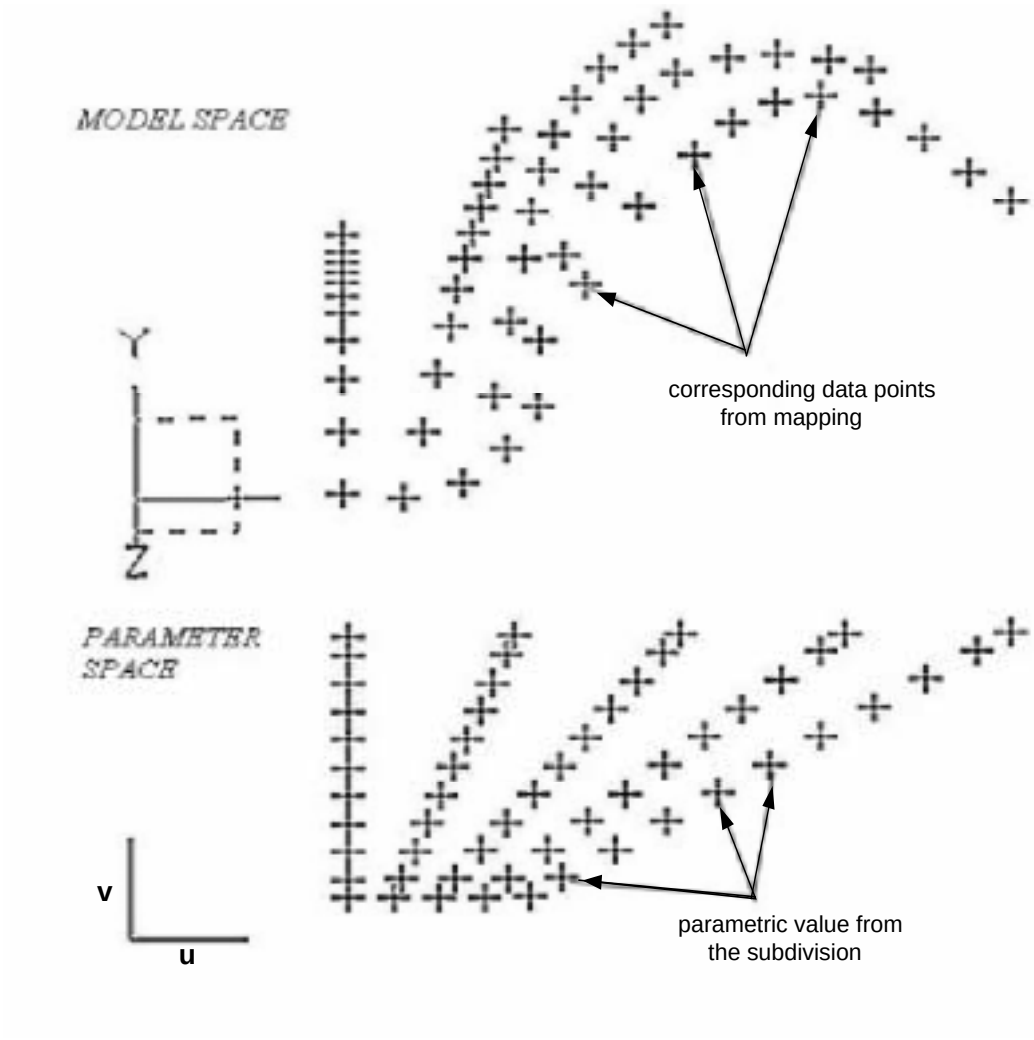


Figure 12 Data points of the trimmed surface are obtained from the mapping [Bind00]

After the set of data points has been found, in order to use Fleming's constraint-based inversion algorithm to generate the trimmed surface, two end conditions need to be specified. They are typically equal to the first derivatives of the original surface at the trimming curve. A linear system can be set up and solved to get the control vertices for describing the new surface patch. The evaluation of a B-spline basis function at a knot of multiplicity of two results in the first

derivative at the corresponding points. The first derivatives used as end conditions are calculated by the following equations:

$$\frac{d}{du}P(u) = \frac{d}{du}\sum_{i=0}^{n} d_i N_i^k(u) = \sum_{i=0}^{n} d_i \frac{d}{du}(N_i^k(u)) \tag{5.1}$$

$$\frac{d}{du}P(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} d_{ij}(\frac{d}{du}N_i^k(u))N_j^l(v) \tag{5.2}$$

$$\frac{d}{dv}P(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} d_{ij}N_i^k(u)(\frac{d}{dv}N_j^l(v)) \tag{5.3}$$

Equation (5.1) shows a mathematical description of the first derivative for B-spline curves. Equation (5.2) and (5.3) extend the evaluation to B-spline surfaces. All the functions are numerically computed by using the non-recursive method described in section 4.6.

5.2 Trimming Curve Conversion

Typically, a trimming curve has two categories: "open trimming curve" and "closed trimming curve". The term of "open trimming curve" is defined as a curve that intersects exactly two different boundaries of the parametric space [Roja94]. While a "closed trimming curve" does not intersect any boundaries of the parametric domain and it is totally located inside the original surface as shown in Figure 13.

When defining the end conditions by using first derivative at the trimming curve, it should be noted that in the constraints based inversion procedure, if a continuity constraint is specified at a certain location of the trimming surface, the entire row or column of trimming points will be set to the same continuity constraint. For instance, if we set the C^1 continuity constraint on a point on a trimming curve in v direction, the continuities of the entire v direction curves will be reduced from C^2 to C^1 and the original C^2 continuity in u direction is disrupted. Therefore, specifying continuity constraints at a trim point in both parametric directions is impossible for the inversion algorithm.

In this research, only one-directional (u direction) iso-parametric curves were trimmed when surface trimming. Therefore, setting a continuity constraint only in the u-direction will provide a reasonable accuracy.

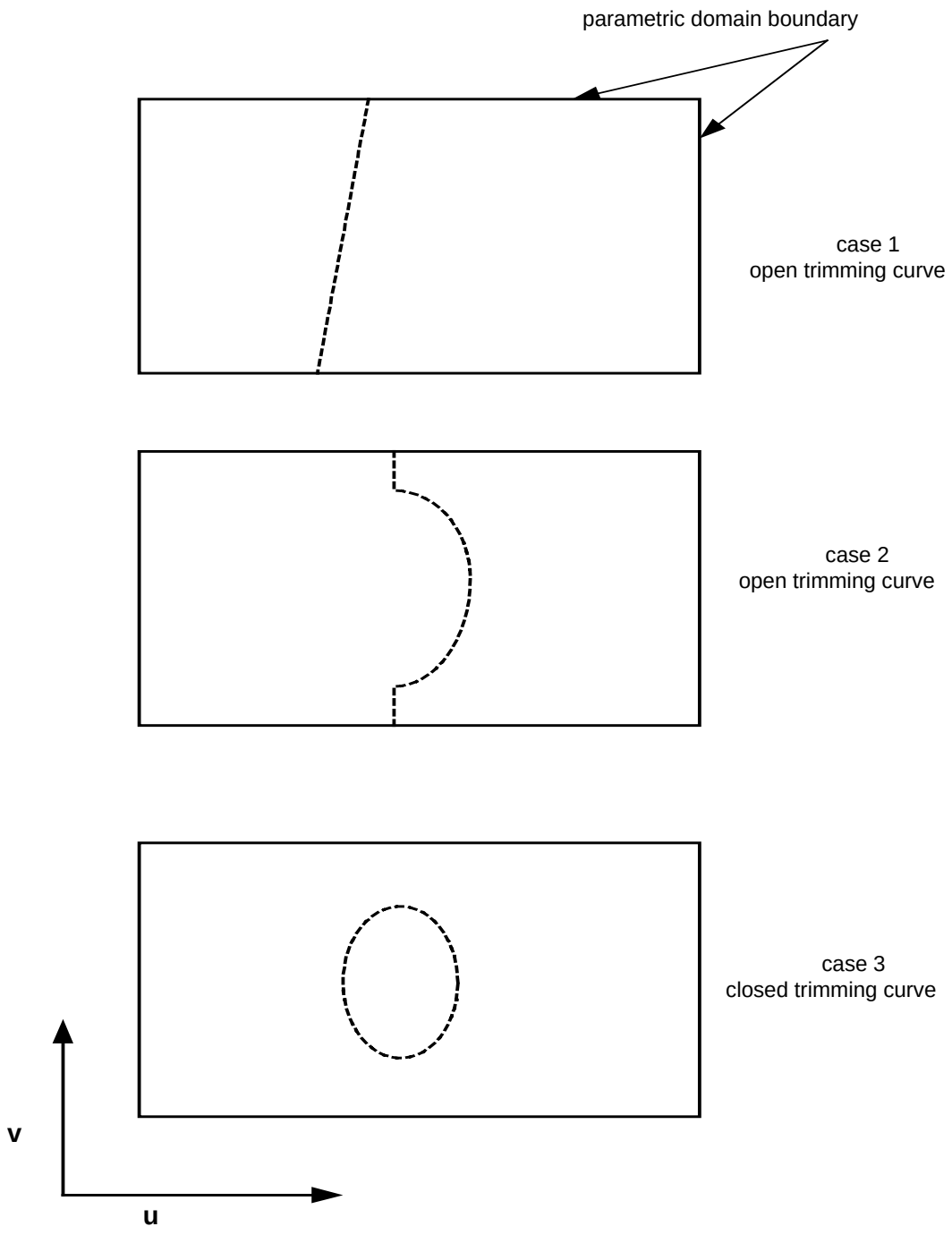


Figure 13 Cases of geometric trimming

In the case of closed trimming curves, Rojas [Roja94] changed the parameterization of the surface by reordering the parametric data points. In order that the trimming method described above can be applied, the parameterization procedure is described in Figure 14 and Figure 15. The first row of parametric values (u = 0, v = 0) starts at one corner of the original parametric space, then goes counterclockwise until the "artifact curve" is reached. The second row and the following rows proceed the same way as the first row. The procedure goes on until the closed trimming curve is reached. When specifying continuity constraints in this procedure, the entire set of iso-parametric curves is affected. For example, if a continuity constraint is added to the third point in the parametric *u* direction, all curves in *u* direction will be affected. Therefore, Fleming's constraint based inversion algorithm can not be applied to this case for the preservation of C^2 continuity.

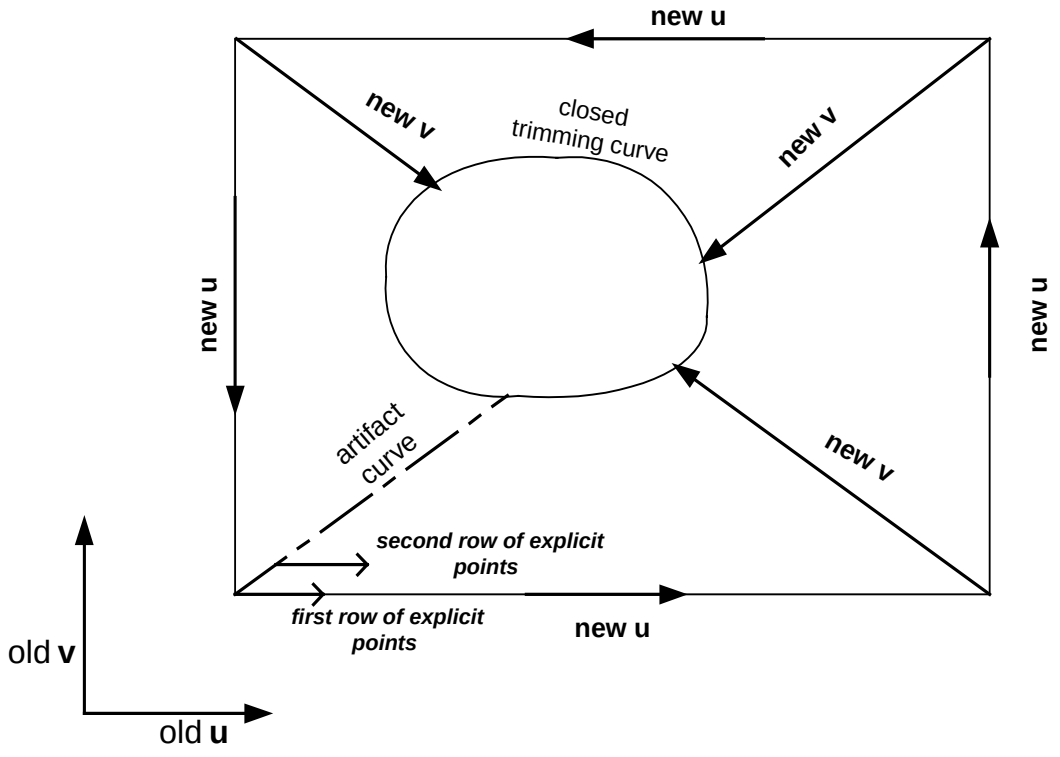


Figure 14 Parameterization for a closed trimming curve

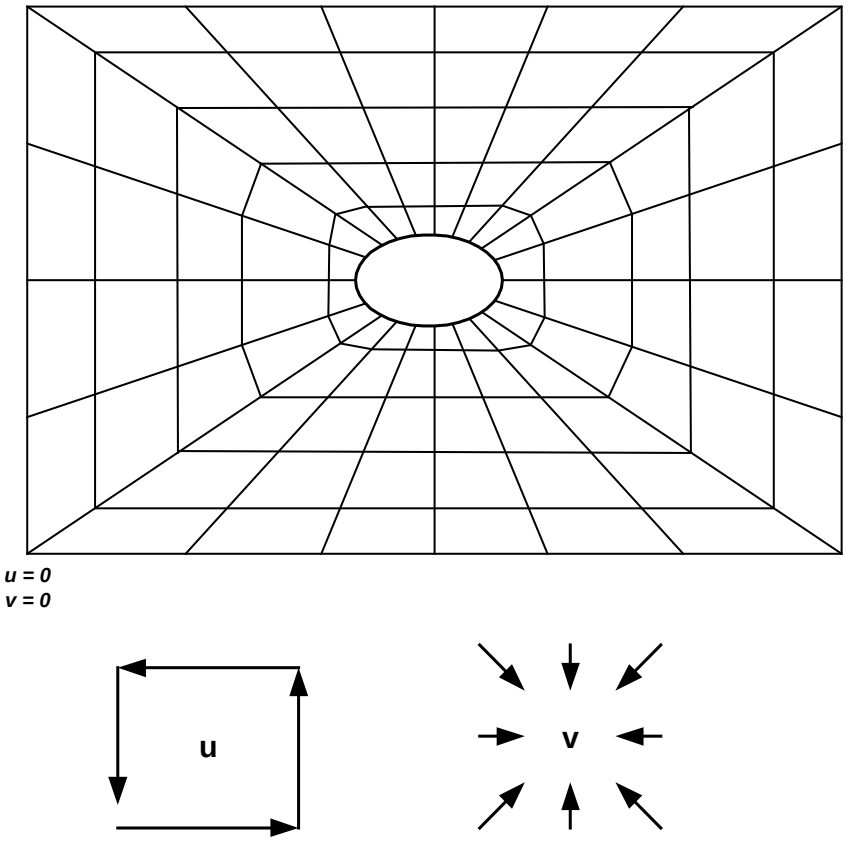


Figure 15 Mesh generation for a closed trimming curve

This parameterization scheme may cause a sharp discontinuity since the information about the original surface boundary is lost when setting the constraints on the trimming boundary. However, the closed trimming curve, as the resultant of surface intersection, usually occurs when surfaces intersect. The purpose of this thesis is to geometrically trim surfaces at a closed trimming curve, and blend the trimmed surface into one surface. As an example, the aircraft fuselage intersects the wing at a closed trimming curve. Rojas' trimming approach has been proved very accurate for surface trimming at an open trimming curve. In order to use this trimming method for the case of a closed trimming curve, a procedure is developed to convert a closed trimming curve case into an open trimming case. Figure 16 and Figure 17 illustrate the conversion method. Obviously, a closed trimming curve in model space also has a closed shape in its parameter space as shown in Figure 16. The closed curve in parameter space can be separated into two pieces at two points (point A and point B in Figure 17), which have the maximum separation apart. Thus, one closed trimming curve is changed into two open trimming curves, each of which is an individual open trimming curve in the same parameter space.

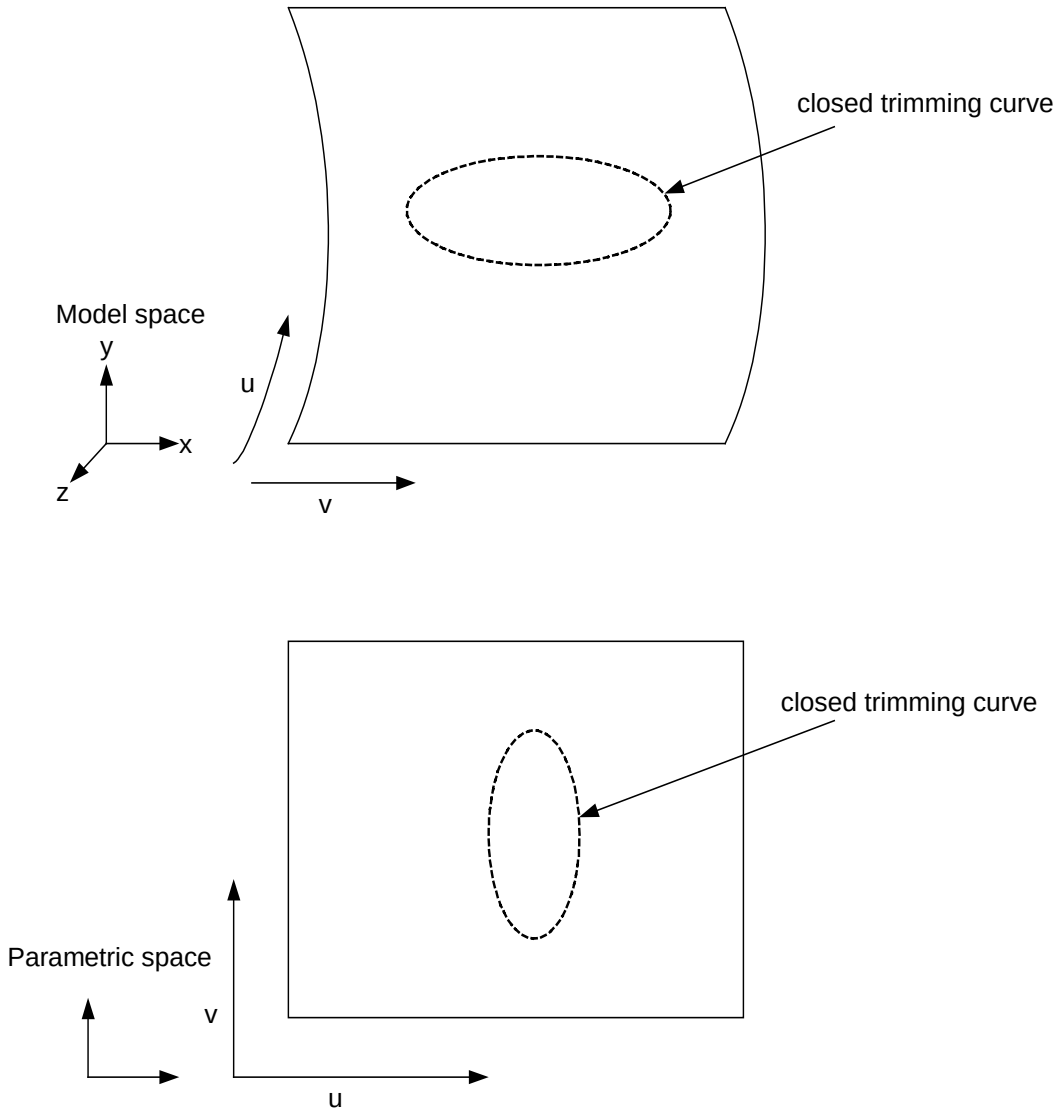


Figure 16 Closed trimming curve in model space and parametric space

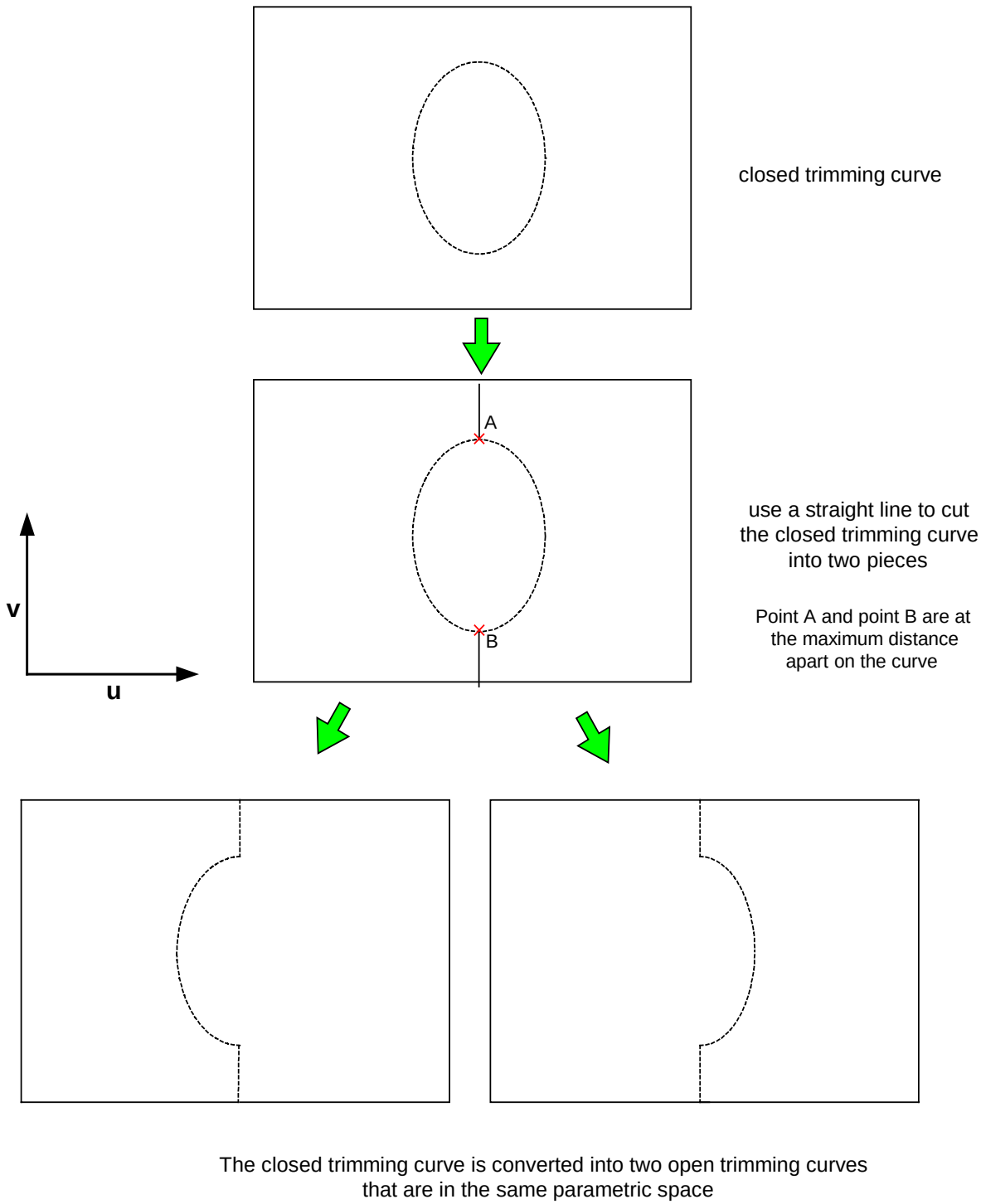


Figure 17 Converting a closed trimming curve into two open trimming curves

5.3 Geometric Trimming

The following steps are performed to trim the model surfaces (aircraft fuselage and wing):

5.3.1 Setting up Model Surfaces

Data points describing the fuselage are read from a data file "*fuselage*". Data points describing the wing are read from a data file "*wing_upper*" and "*wing_lower*". The aircraft fuselage and wing are modeled using SDRC's I-DEAS as shown in Figure 18. The fuselage is a bicubic uniform B-spline surface composed of 10×10 patches and the wing consists of $2 \times 9 \times 9$ patches. All the data files used in this research are compiled in Appendix A.

5.3.2 Getting Trimming Data

The closed trimming curve resulting from the intersection of the fuselage and wing is approximated from I-DEAS as well. It is defined as a set of points in its own parametric space. Since the topic about how to find the intersection data has been discussed by Jones [Jone91], this research assumes the trimming data is a given condition. The sample rate of the trimming data (or how many data points are selected to represent the trimming curve) depends on the cases. Generally, it is a similar amount of data points used to represent one row data points of the original surface. Apparently, the more points are used to describe the trimming curve, the more accurate the newly created surface resembles the original surface. In this research, the trimming data is interrogated from the model surfaces using I-DEAS. The trimming data file for fuselage is read from "*f_upper_trimdata*" and "*f_lower_trimdata* ", and the files for wing is read from "*w_upper_trimdata*" and "*w_lower_trimdata*".

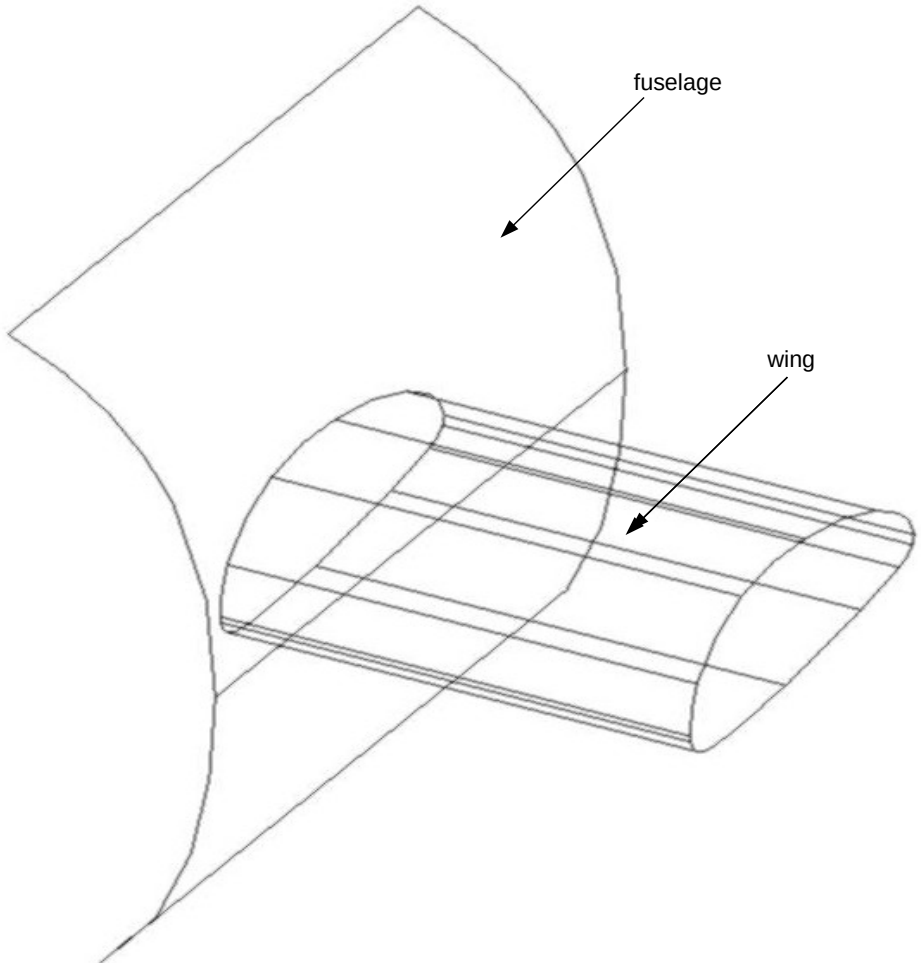


Figure 18 Aircraft fuselage and wing modeled from I-DEAS

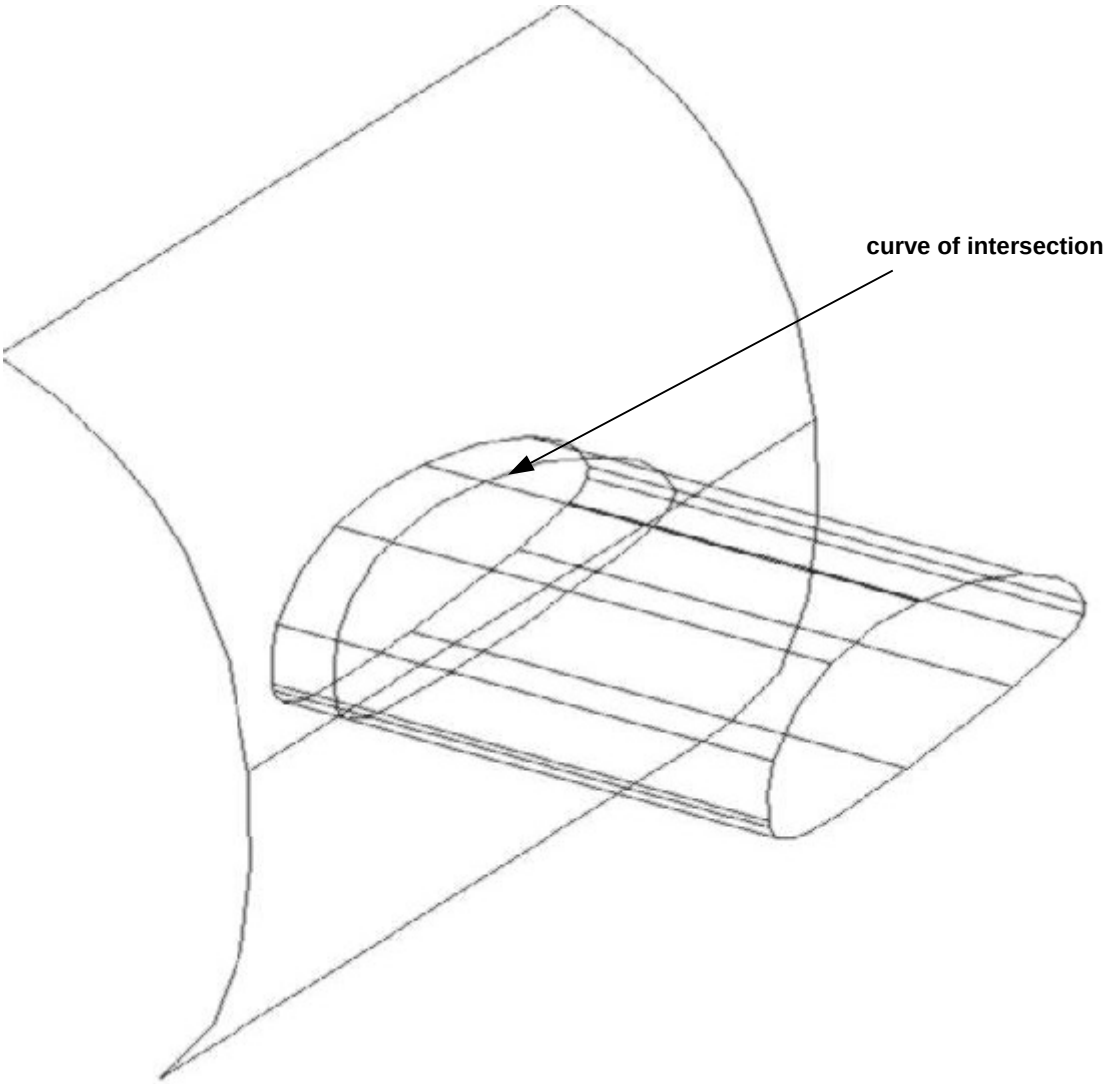


Figure 19 Fuselage intersects the wing at a closed trimming curve

In parameter space, the closed trimming curve of the fuselage is converted into a set of points that represent an open trimming curve as shown in Figure 20. The conversion process is described as shown in Figure 17.

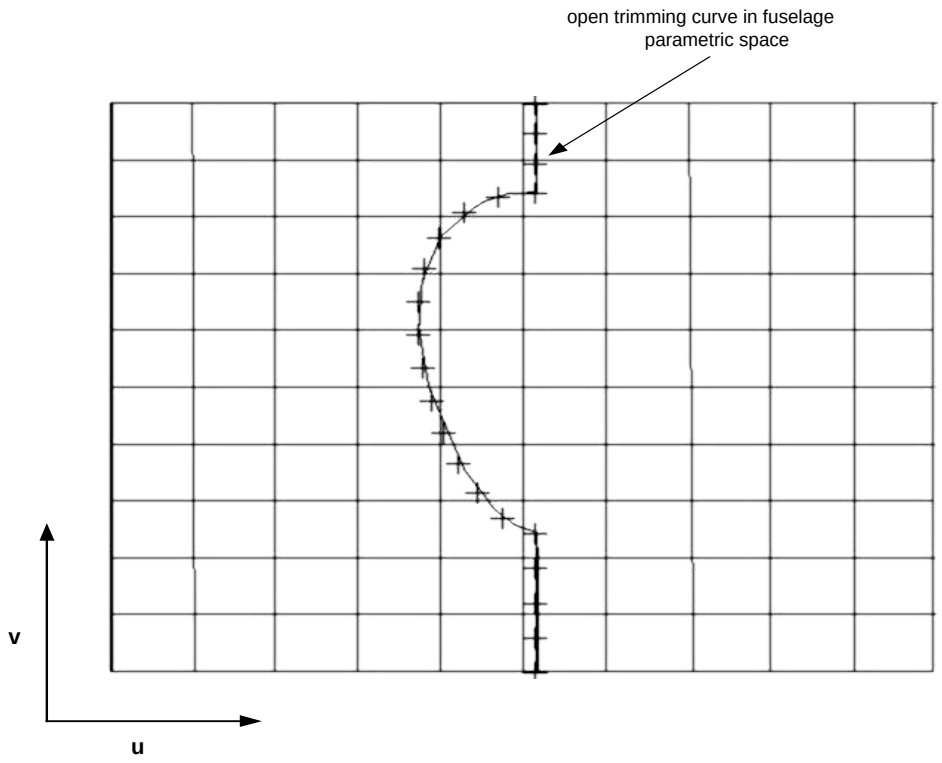


Figure 20 Closed trimming curve is converted into open trimming curve

5.3.3 Parametric Subdivision

Respectively the retained portion of the parametric domain is subdivided and the new parametric values are mapped onto the original surface to obtain the trimmed patch. Figure 21 and Figure 22 show the subdivision of the parametric domain for the fuselage. Thus, the original fuselage is trimmed and divided into two parts: upper fuselage and lower fuselage. As well, the wing is trimmed and divided into two patches, upper wing and lower wing as shown in Figure 23, Figure 24 and Figure 25.

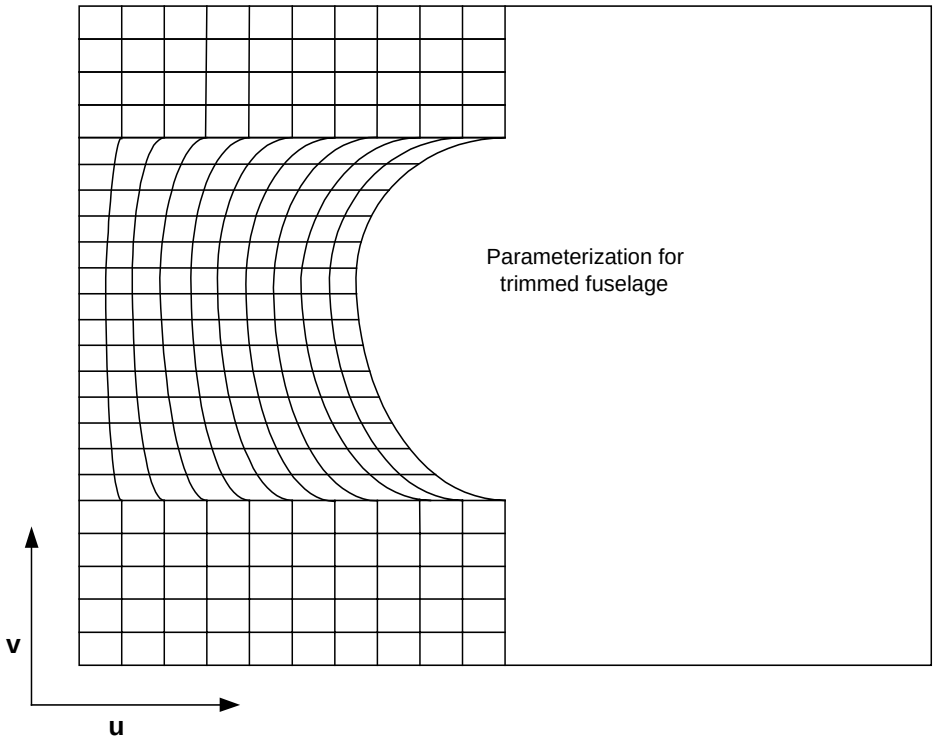


Figure 21 Subdividing parametric domain for the fuselage

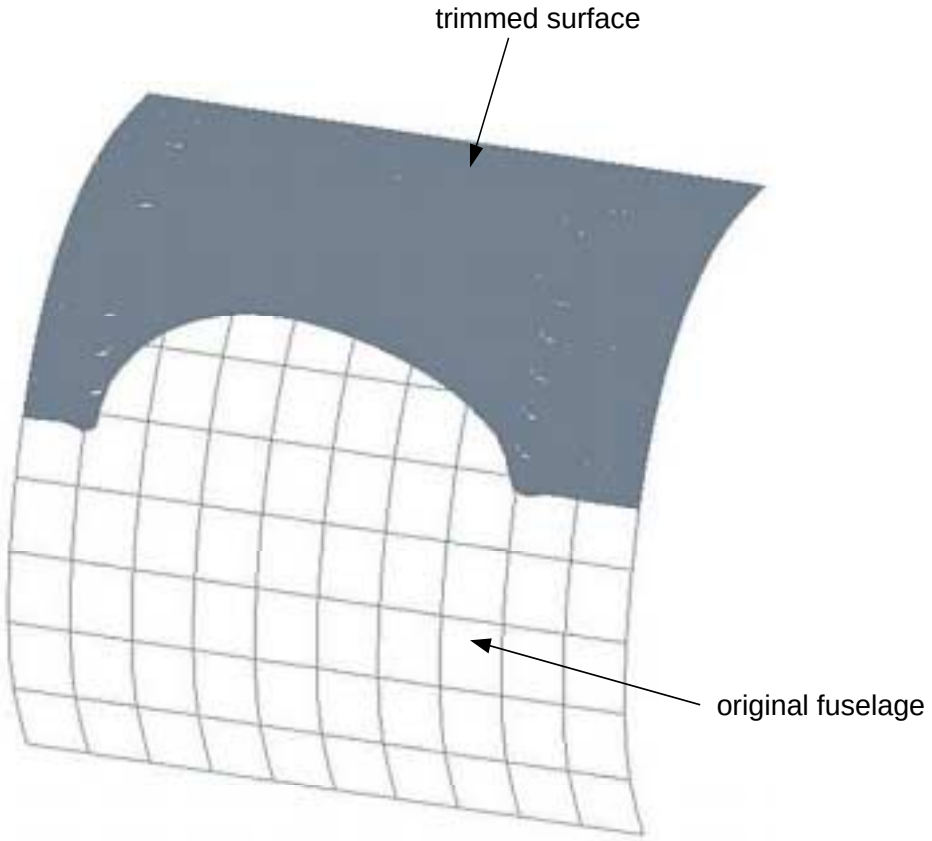


Figure 22 Original fuselage and the trimmed fuselage

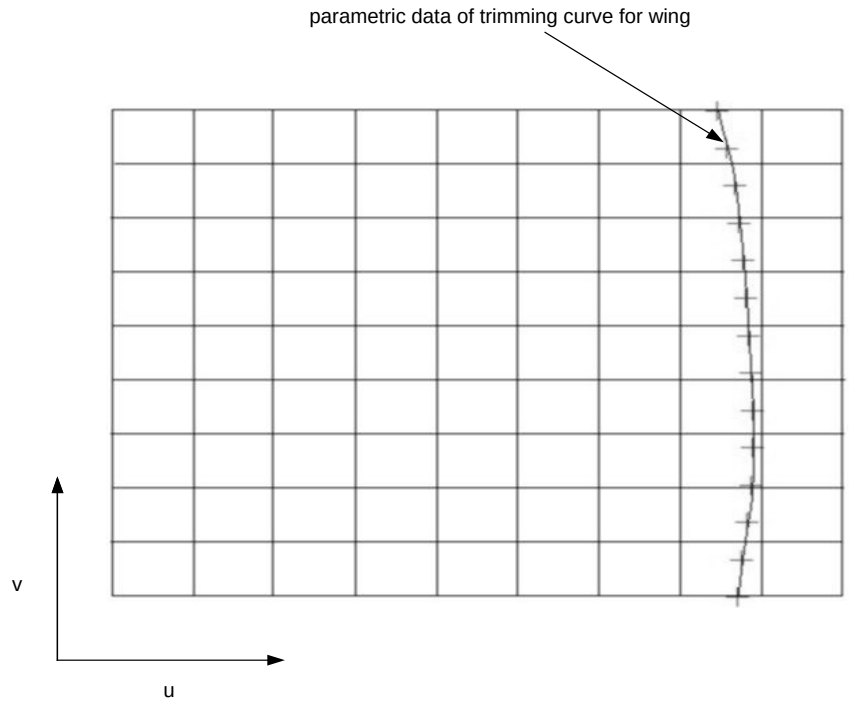


Figure 23 Trimming curve of the wing in parametric space

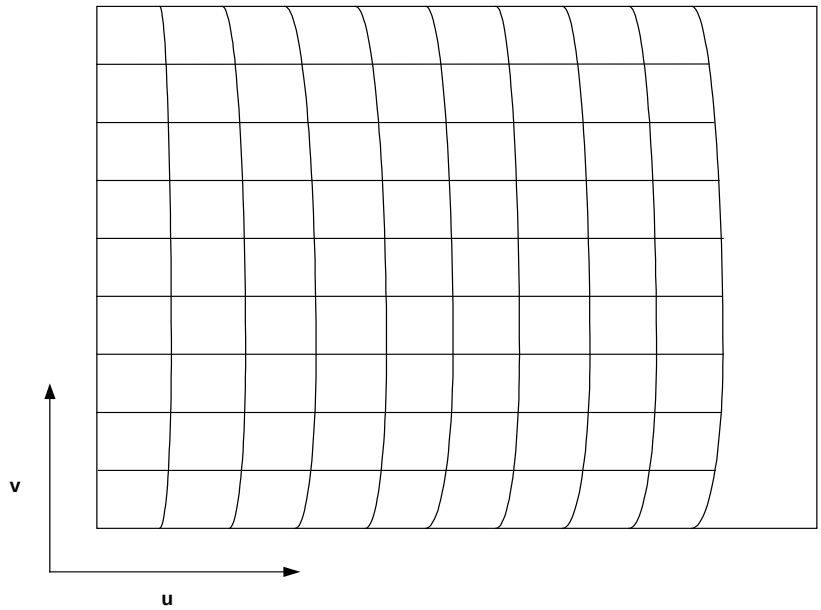


Figure 24 Subdividing parametric domain for the wing

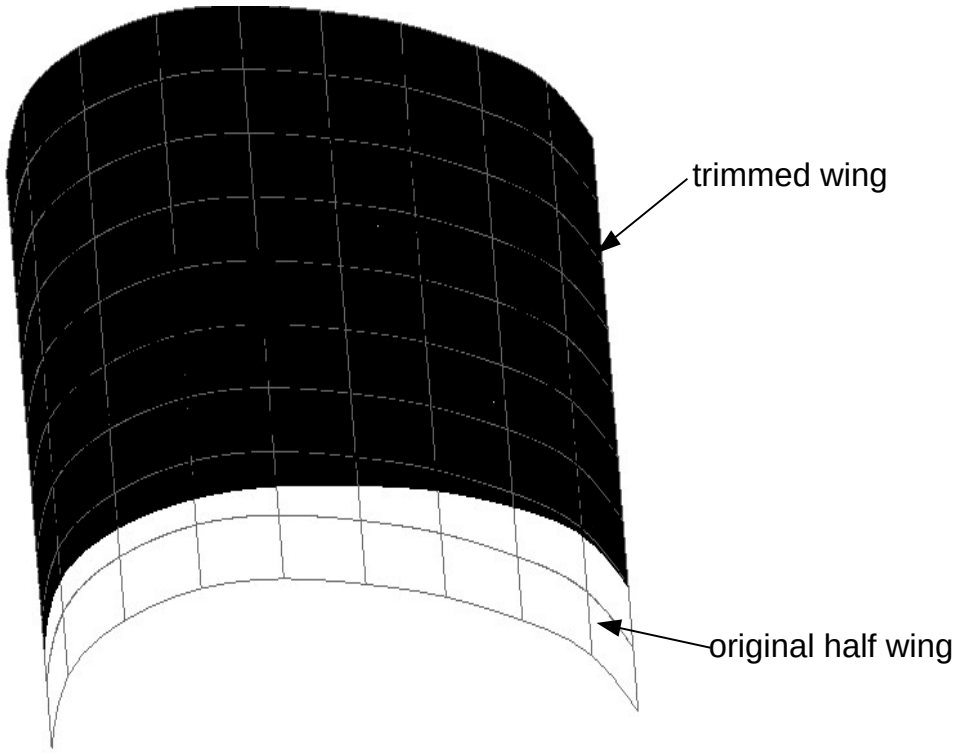


Figure 25 Original wing and the trimmed wing

The following figure shows the upper portion of the original fuselage and wing.

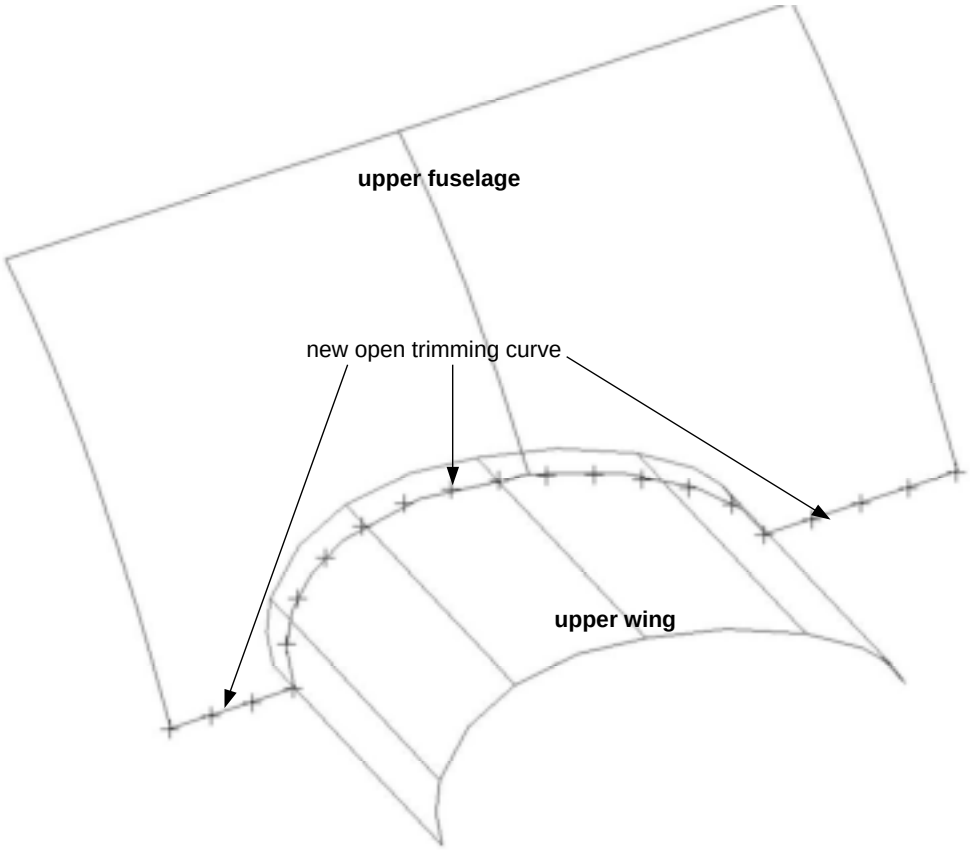


Figure 26 Fuselage and wing are trimmed and divided into two pieces respectively

5.3.4 Offset Computation

When surfaces are blended at the intersection curve, typically the actual blending location is a curve away from the intersection curve with a certain distance. The offset curve is calculated from the trimming curve in its own parameter space (u-v space) by offsetting the trimming curve. The following procedures are followed to compute the offset curves.

In the example as shown in Figure 27, the fuselage has ($l-1 + m + r-1$) parametric data on the trimming curve, where $l = 4, m = 14, r = 5$. The middle portion of the trimming curve is an arbitrary two-dimensional curve and two end sections of the trimming curve are straight lines. The offset curve can be described as a set of data points in the parametric space. For the middle portion of the trimming curve, each middle point between any two adjacent points is easily found by averaging the two points. A unit vector is calculated through each middle point. The unit vector is the vector of magnitude one and it is perpendicular to the straight line connecting two neighboring points. Each offset point is computed by multiplying an offset distance on the unit vector. The new end points of the middle curve are calculated by shifting original end points the offset distance in v direction. For two straight-line sections, new points are evenly mapped on the new line section with the original data number. In the example, originally $l = 4$, $m = 14$, $r = 5$. After offset, $l = 4, m = 15, r = 5$.

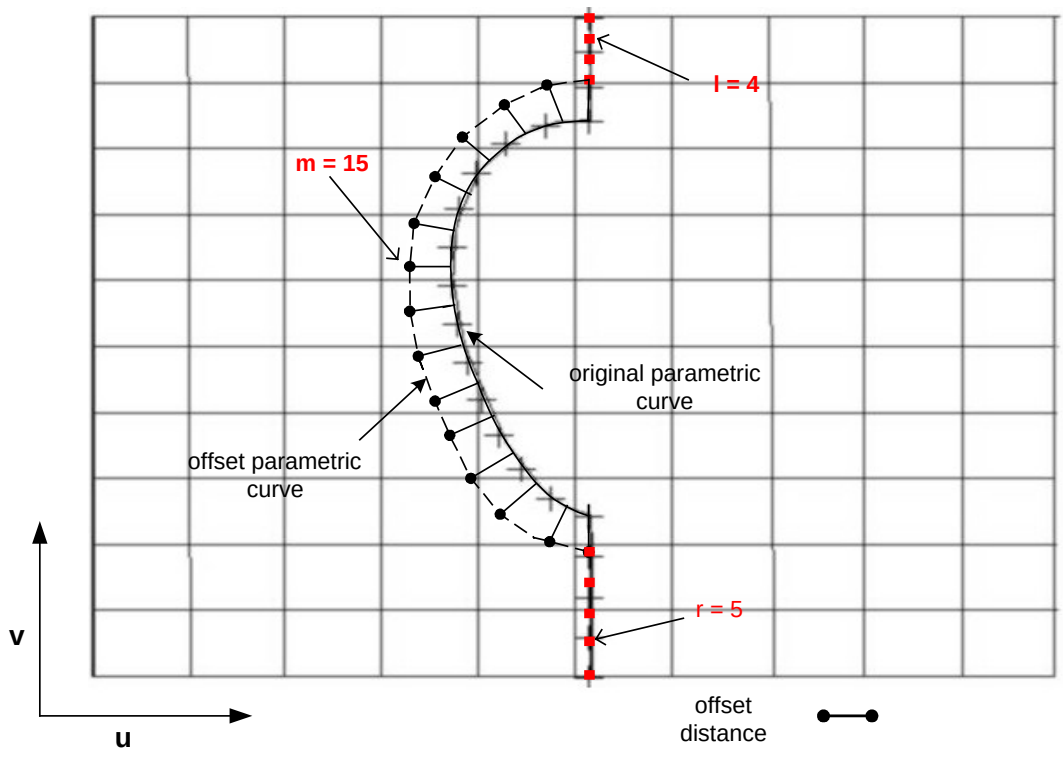


Figure 27 Computing the offset of trimming curve for the fuselage

Offset computation for the wing is similar to the procedure described above. However, the end points are determined by shifting the original end the offset distance in *u* direction.

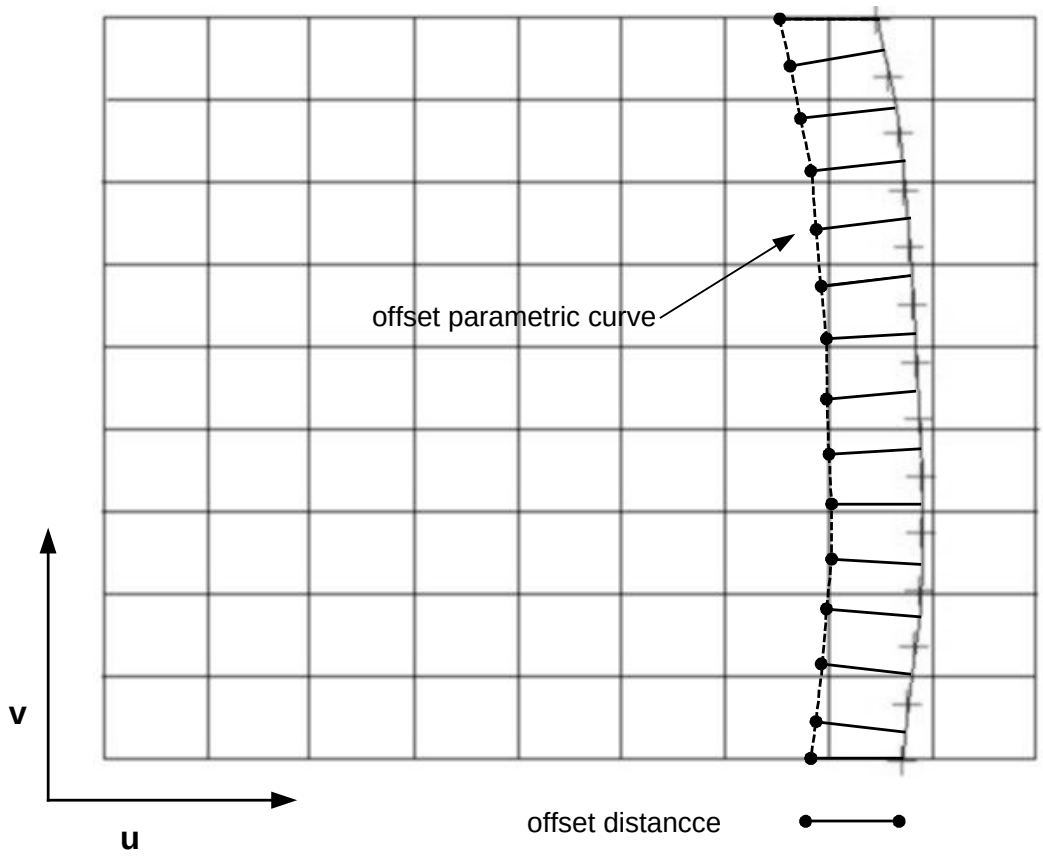


Figure 28 Computing the offset of trimming curve for the wing

The following figures show the trimmed fuselage and wing after making the offset:

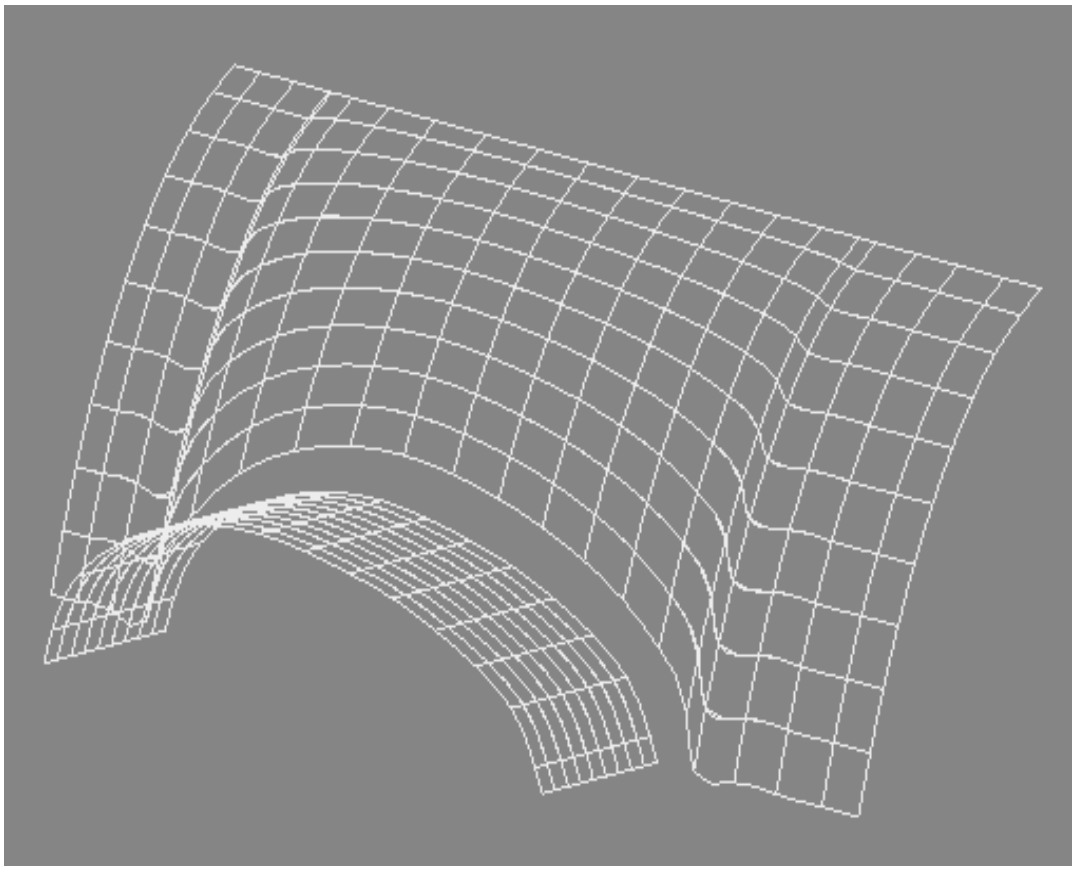


Figure 29 Grid image of the trimmed fuselage and wing after making offset

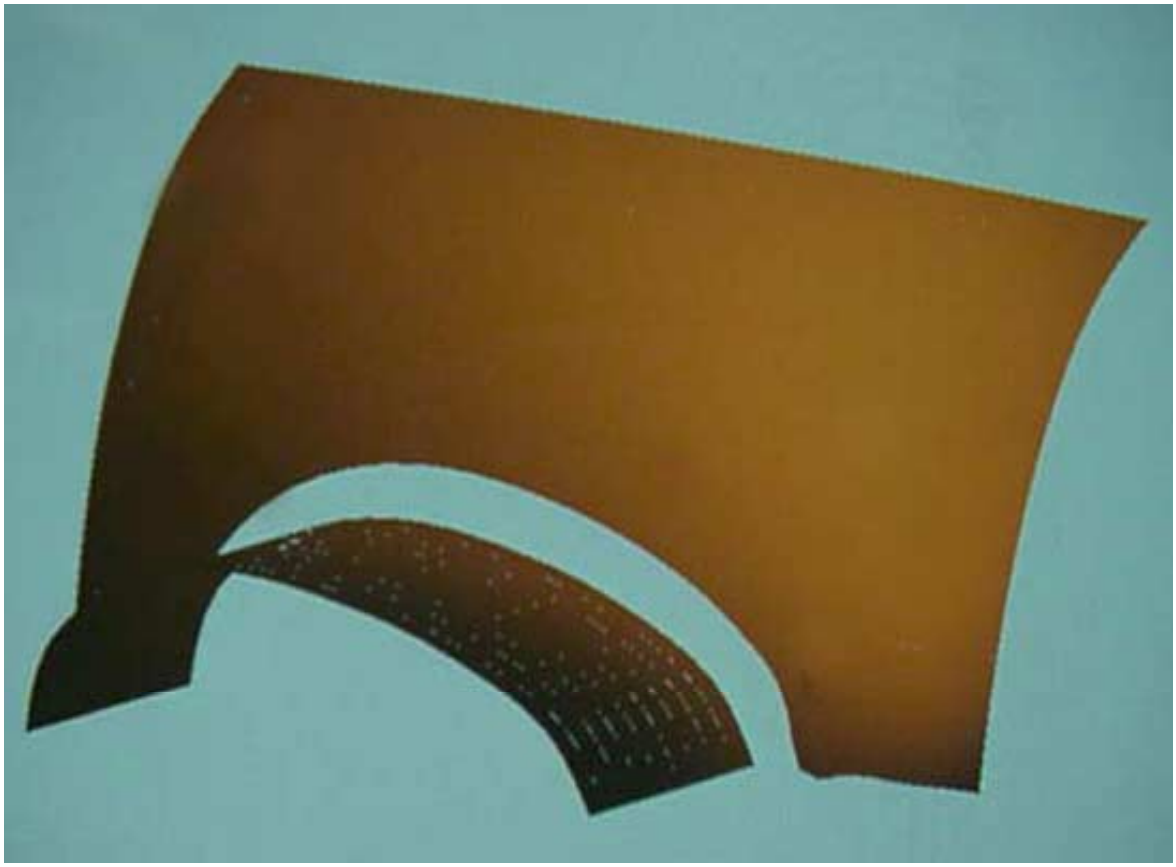


Figure 30 Shaded image of the trimmed fuselage and wing

6.0 Filleting

6.1 Introduction

A fillet is defined as an intermediate surface that blends two surfaces into one smooth surface. Preliminary aircraft design requires C^2 continuity for the filleting of surfaces. Gloudeman provided a method to blend two B-spline surfaces into a C^2 continuous surface [Glou90]. His approach also guarantees C^2 continuity at the blending location where the fillet joins the original surfaces. However, this algorithm is based on visual trimming. After filleting, the blended surface is actually made up of three surfaces: the fillet and the two original surfaces. As described earlier, the unwanted portion of the trimming surface still exists but is hidden from the designers. For example, the final blending surface shown in Figure 31 consists of three patches: surface A, fillet and surface B (each of which has its own parametric defining domain). The disadvantage of visual trimming is that it over-defines the trimming surface. Even the trimmed portion is useless; it still mathematically exists and needs management. Thus, analytical results cannot be achieved accurately on the surface. As opposed to visual trimming, geometric trimming provides a full mathematical representation of the trimming surface. In other words, the mathematical equations for both the original surface and the blended surface are the same as shown in Figure 32.

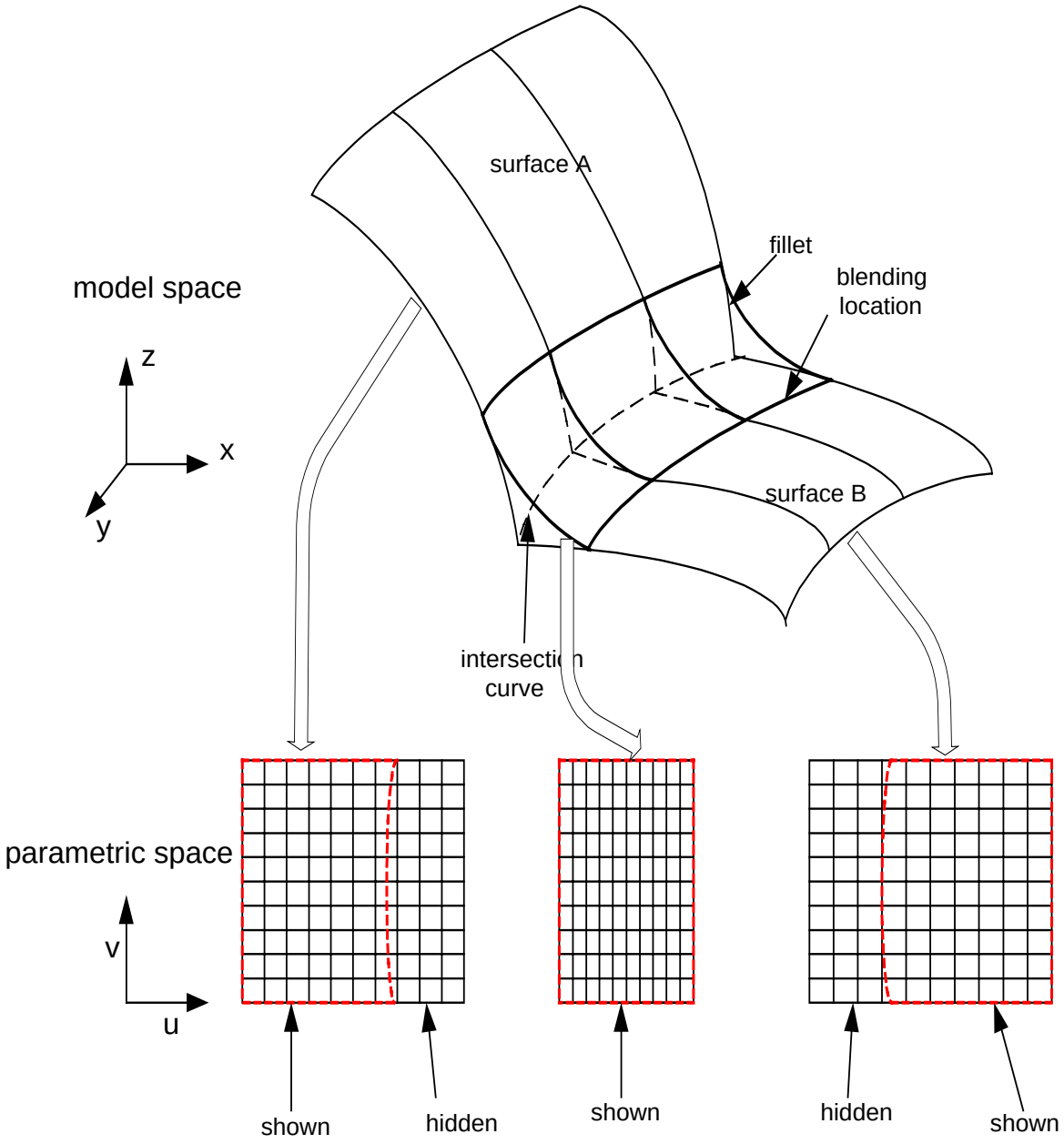


Figure 31 Surface blending based on visual trimming

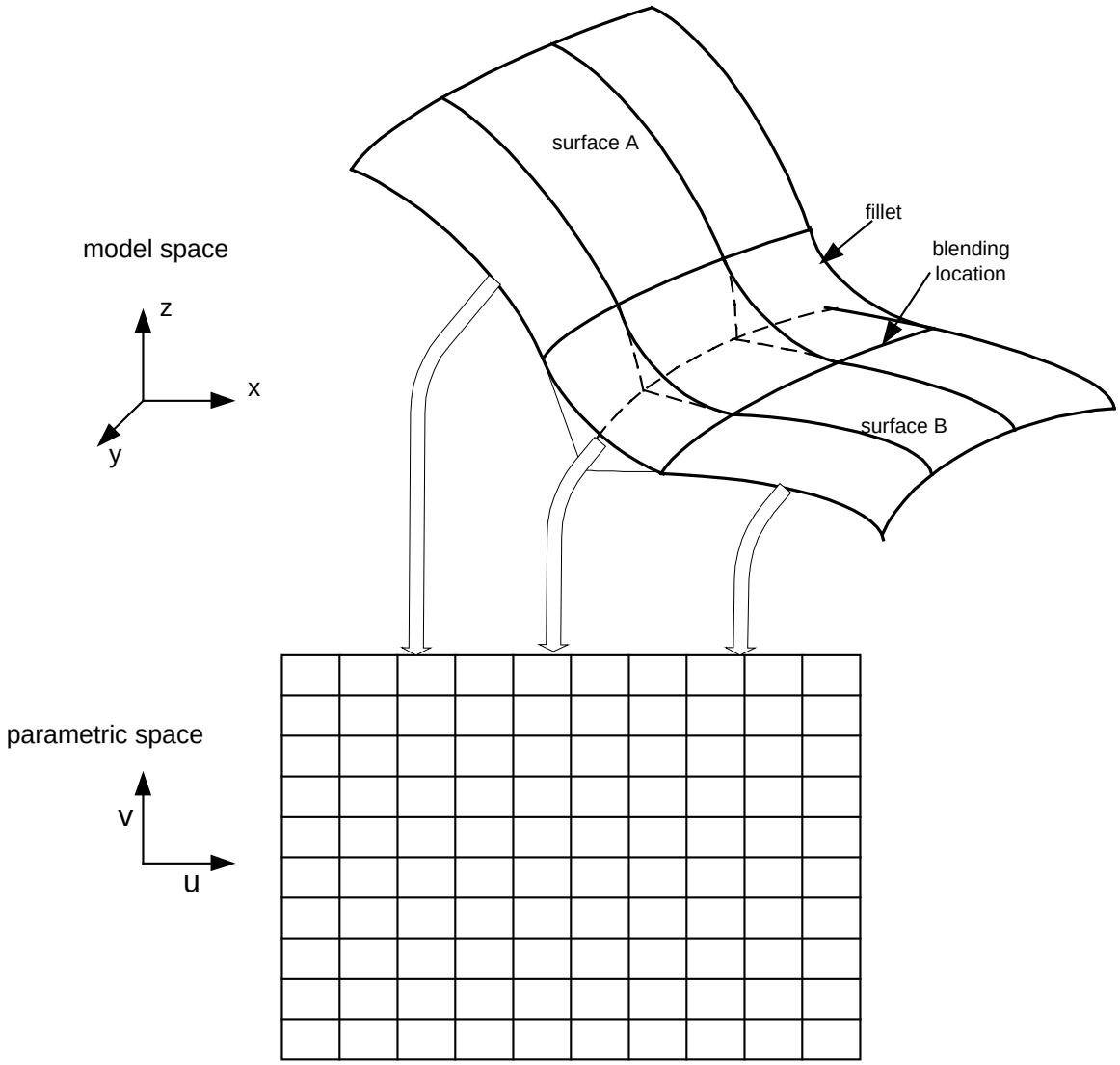


Figure 32 Surface blending based on geometric trimming

A new approach for surface filleting based on geometric trimming is addressed in the following section. First, the algorithm will be developed for blending two B-spline curves. Then an extension of this algorithm for blending two surfaces will be discussed.

6.2 Curve Filleting

When two curves intersect, the intersection point is found first. Then these curves are trimmed at the offset break points where the fillet joins two curves as shown in Figure 33. In the phase of curves blending, a new curve is created by interpolating the set of data points coming from the break points of two trimmed curves and the fillet. This process requires the number of data points, the data point coordinates, the type of parameterization and continuity constraints. Generally, the parameterization type is chosen as chord length for non-uniform B-splines. The array of data points results from the trimmed curves. The continuity constraints are set as two for the C^2 continuous requirement. Therefore, if a set of data points is given, a C^2 continuous curve can be generated by interpolation. However, the new blending curve starts to oscillate with smaller amplitudes as it gets farther from said trimming point. As shown in Figure 34, the oscillation impacts the new blending curve to approximate the original curve in the trimmed portion.

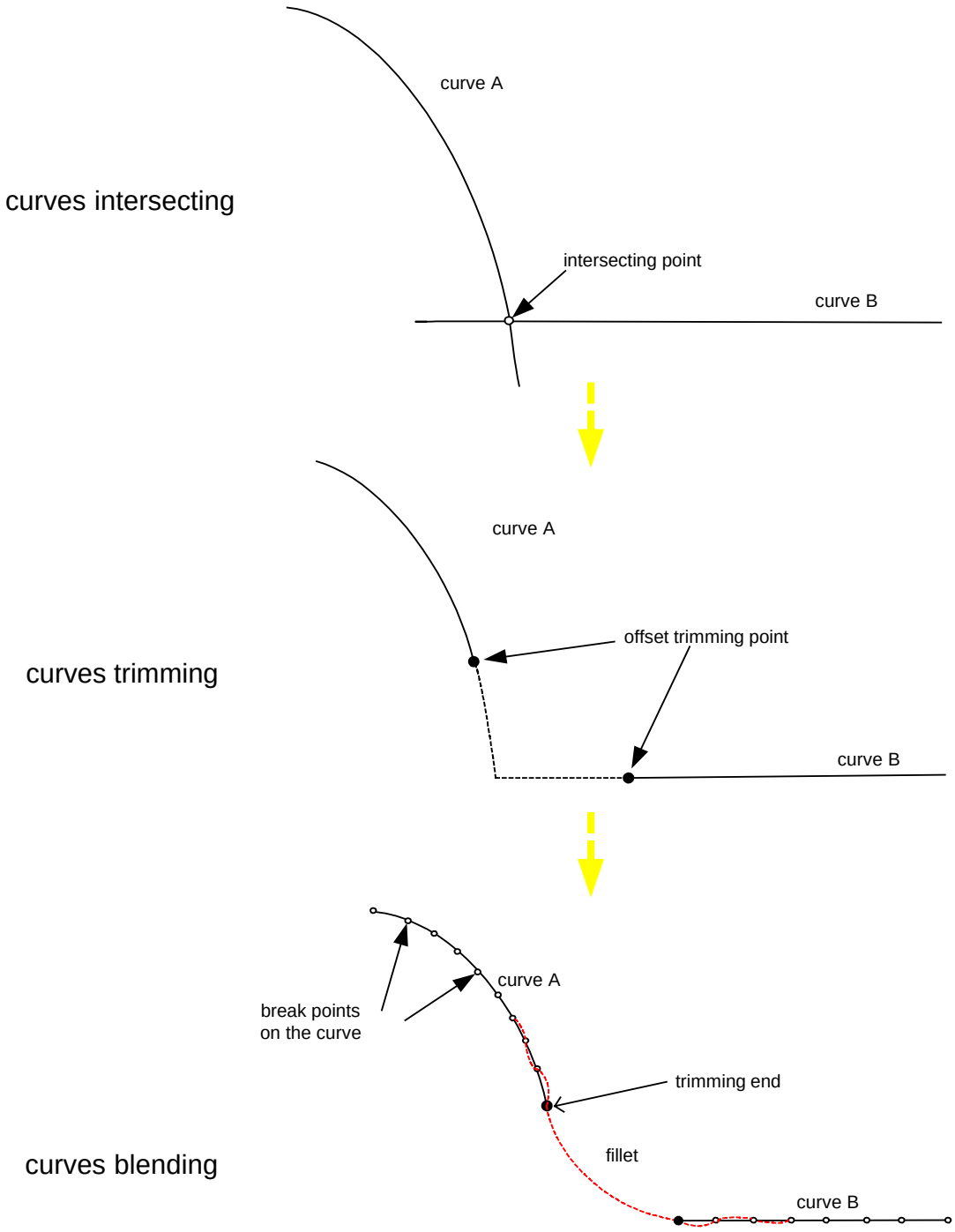


Figure 33 Blending two B-spline curves

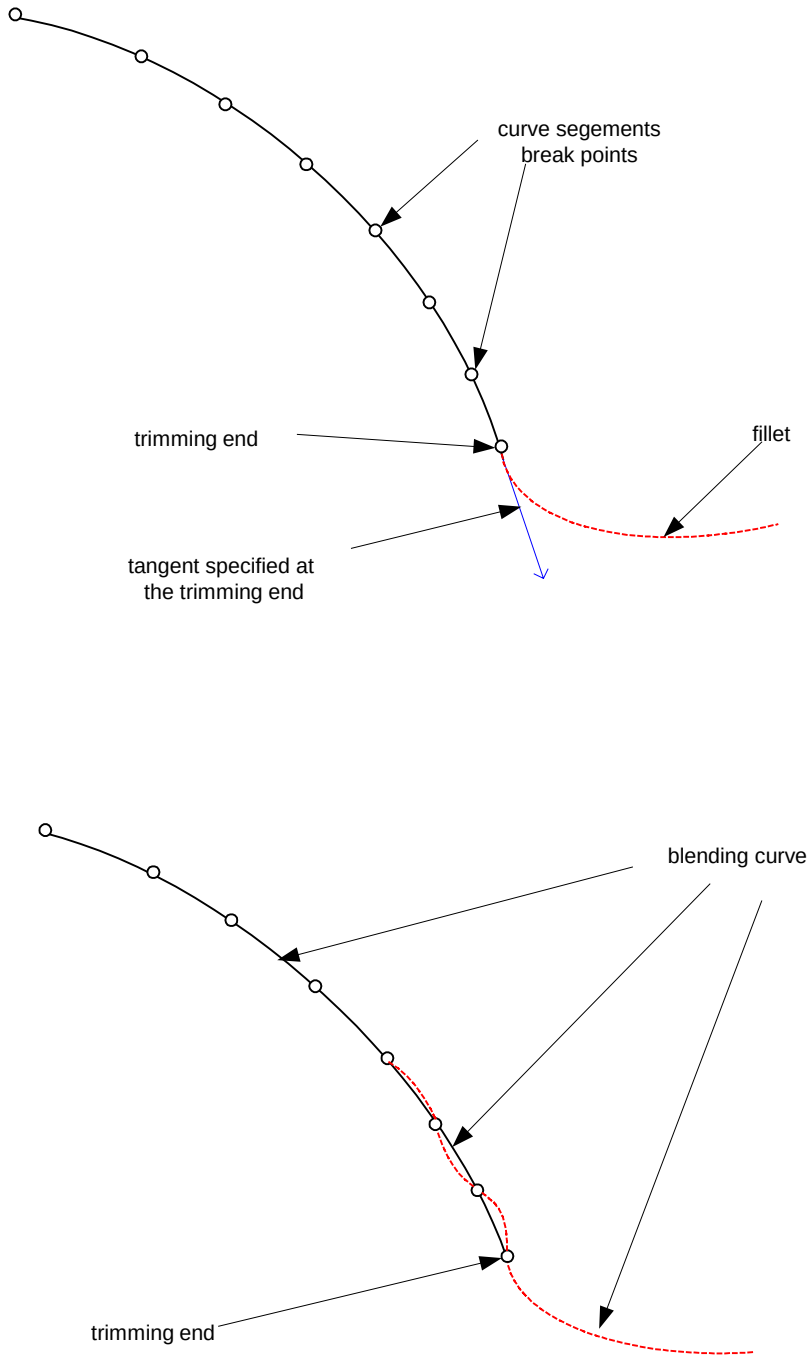


Figure 34 Blending curve oscillates about the original curve

In the case of uniform cubic B-spline curves, if a C^2 continuous fillet is required, any break point along the curve is a weighted average of only three control vertices. The following equations shows that only three control vertices, q_{i-1}, q_i and q_{i+1}, affect the position, slope and curvature of a cubic B-spline curve at the curve segment break points [Glou89].

$$P(0)_i = \frac{1}{6} q_{i-1} + \frac{2}{3} q_i + \frac{1}{6} q_{i+1} \tag{6.1}$$

$$P_u(0)_i = -\frac{1}{2} q_{i-1} + \frac{1}{6} q_{i+1} \tag{6.2}$$

$$P_{uu}(0)_i = q_{i-1} - 2q_i + q_{i+1} \tag{6.3}$$

For a non-uniform cubic B-spline curve, similar to uniform cubic B-splines, the position, slope and curvature of non-uniform cubic B-splines are the functions of three adjacent control vertices and five knot values at the break points. At the same time, the position, slope and curvature at the points other than the break points are the functions of four adjacent control vertices and six knot values. As presented in Figure 33 and Figure 34, when the new data points are interpolated, the new blending curve is intended to match the original curve on the trimmed section. Actually, it oscillates about the trimming end of the original curve, because the points near the trimming points of the new curve are affected by three control points which are from the intermediate points on the fillet or from the other trimmed curve. In order to force the blending curve to lie on the original trimmed curve around the trimming end, three points are necessary to be added near the trimming ends before the new curve is interpolated. They result from knot insertion on the trimmed curve. Knot insertion is a simple method of adding control vertices to the B-spline curve description without changing the shape of the curve. The detailed procedure of knot insertion has been discussed in section 4.4. The trimmed curve has a certain number of break points and control vertices, and its shape is the same as the original. If new data points need to be found on the trimmed curve, the corresponding knot values should be determined first, then the new control points can be calculated. With the new knot sequence, the new set of data points can be figured out without changing the shape of the curve as shown in Figure 35. The three new knots are added in the interval between the trimming end (knot u_8) and the next interior knot (

knot u_7). The first knot *1* is inserted at the middle point of u_7 and u_8. The second knot *2* is added at the middle point of *1* and u_8 and the third knot *3* is added at the middle point of *2* and u_8. The new knots are added by such a ratio in order to make sure that the newly mapped break points can approximate the original curve shape no matter what the distance of the interval (u_7 -u_8) is. Of course, other addition ratios will work in this case only if the knots are inserted in the interval near the trimming end. After adding three points on each trimming end of the curves, the new blending curve is forced to match the original curve on the overlay portion as shown in Figure 36. However the fillet shape is naturally made and is controlled by the four surrounding control vertices which may be the control points of two trimmed curve.

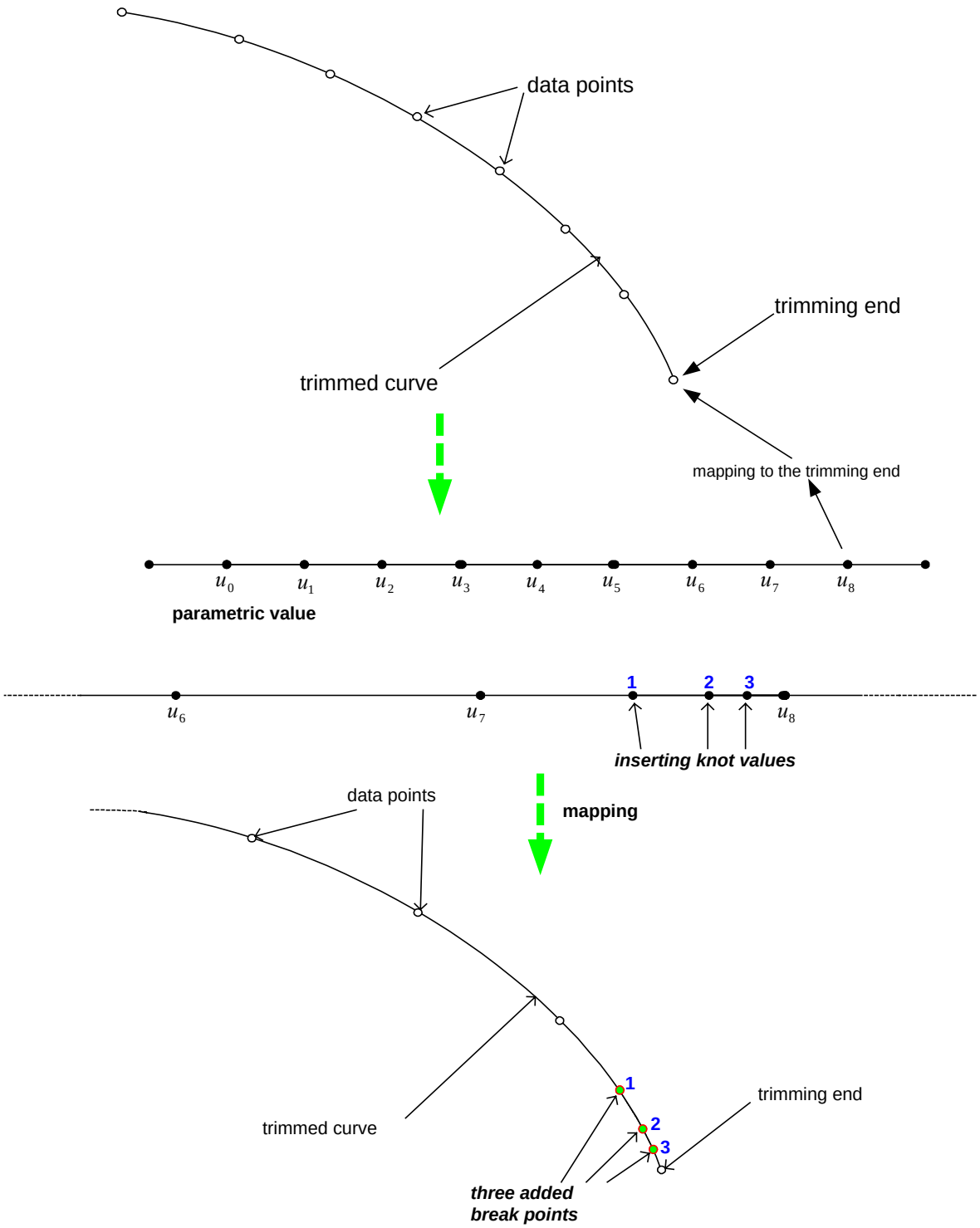


Figure 35 Adding three break points to the trimmed curve

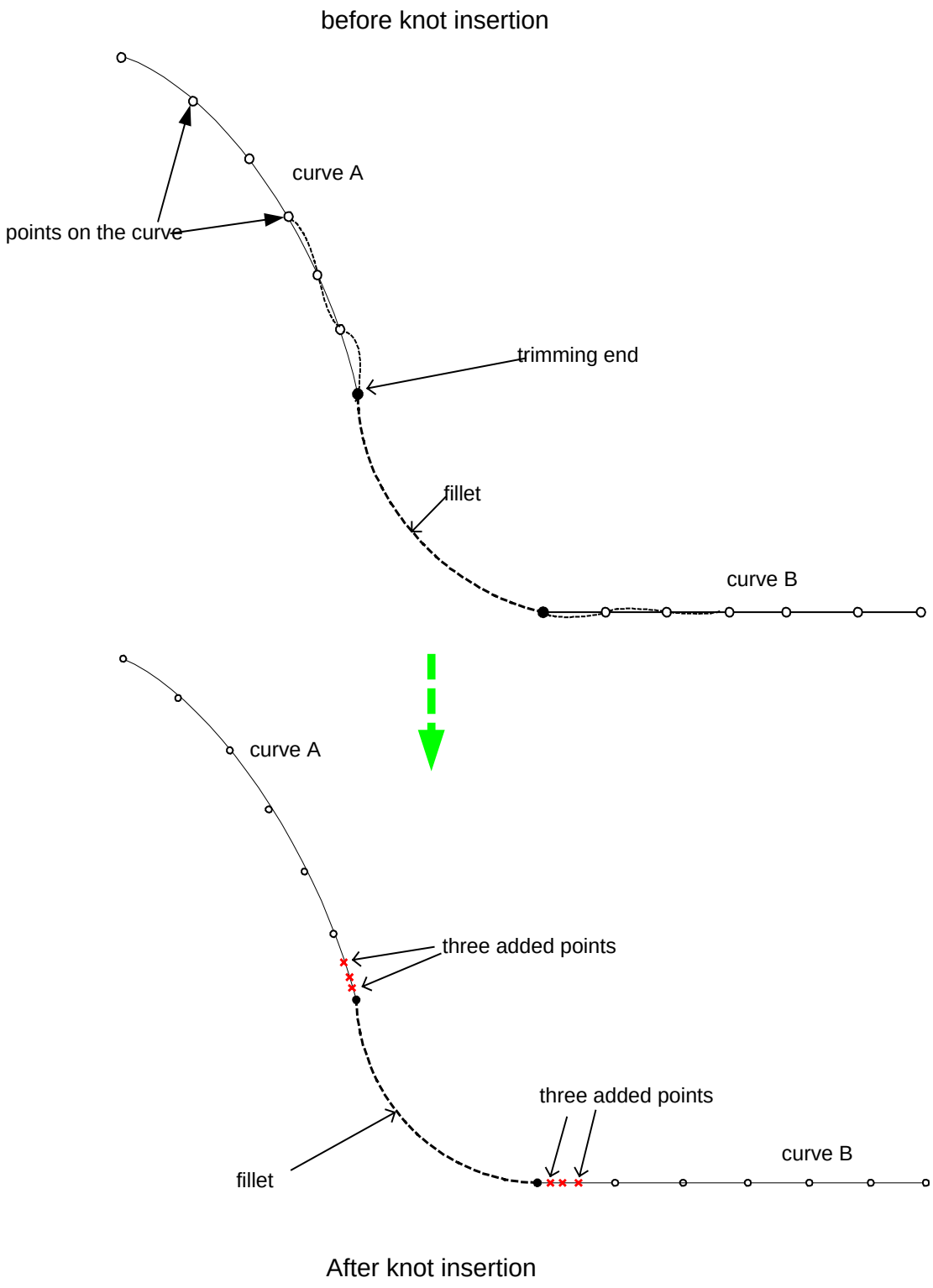


Figure 36 Knot insertion refines the shape of the blending curve

In order to control the shape of the fillet, more intermediate points need to be added between two trimming ends of the original curves. The following steps are for adding more intermediate points to be interpolated to make the fillet as shown in Figure 37. A conic definition can be used to define these intermediate points [Yama88]. The conic is defined by three points (***1, 2, 3***), parameter value ***p*** and ***t***. Points ***1*** and ***2*** are set to be the break points where the fillet blends with two curves. The parameter of ***p*** is used to control the radius of the conic used to represent the fillet. The parameter of ***t*** is to define different points on a specific conic. The parameter value ***p = 0.0*** corresponds to a straight line between points ***1*** and ***2***. While the value ***p = 1.0*** degenerates the conic curve to two straight lines (***1-3*** and ***2-3***). The parameter value ***t = 0.0*** corresponds to a straight line collinear to the line ***1-2*** and the value ***t = 1.0*** corresponds to the straight line ***2-3***. The following equations are applied to compute the coordinates of a point on the conic curve.

$$x(t) = \frac{\{(1-p)Q_{1x} - 2pQ_{3x} + (1-p)Q_{2x}\}t^2 - 2\{(1-p)Q_{1x} - pQ_{3x}\}t + (1-p)Q_{1x}}{2(1-2p)t^2 - 2(1-2p)t + 1 - p} \qquad (6.4)$$

$$y(t) = \frac{\{(1-p)Q_{1y} - 2pQ_{3y} + (1-p)Q_{2y}\}t^2 - 2\{(1-p)Q_{1y} - pQ_{3y}\}t + (1-p)Q_{1y}}{2(1-2p)t^2 - 2(1-2p)t + 1 - p} \qquad (6.5)$$

$$\{\ 0.0 \le t \le 1.0\ \}$$

$$\{\ 0.0 \le p \le 1.0\ \}$$

where Q_{1x}, Q_{1y}, Q_{2x}, Q_{2y} and Q_{3x}, Q_{3y} are the coordinates of control points ***1, 2*** and ***3***.

If necessary, more intermediate points can be calculated on a specific conic by varying the parameter value ***t***. The example shown in Figure 37 illustrates three points calculated at ***t = 0.25, t = 0.5*** and ***t = 0.75***. Although there is no fixed requirement for the number of intermediate points, consistently more evenly spaced points facilitate parameterization and produce a smoother fillet.

In summary, the curve filleting process can be described as the following steps:

- Find the intersection point of two intersecting curves using the subdivision algorithm discussed by Jones [Jone91].
- Calculate the offset trimming points where the fillet joins both intersecting curves. In the case of surface filleting in this research, a computation procedure is discussed earlier to find the offset trimming curve in parametric space, then geometrically trim the original surface to get the offset of the trimming curve.
- Insert knots at the trimming end of each trimmed curve and find the corresponding curve segment break points.
- Define a conic curve as the fillet by using the point of intersection and two trimming end points as three defining vertices of the conic curve.
- Change the parameter value $\boldsymbol{p}$ until the desired fillet shape is achieved. In other words, changing the value of $\boldsymbol{p}$ can vary the radius of a fillet.
- Vary the parameter value $\boldsymbol{t}$ to find different points on the conic as the intermediate points to be used as the interpolating points of the fillet.
- Interpolate all the points defined earlier to generate a non-uniform cubic B-spline curve.

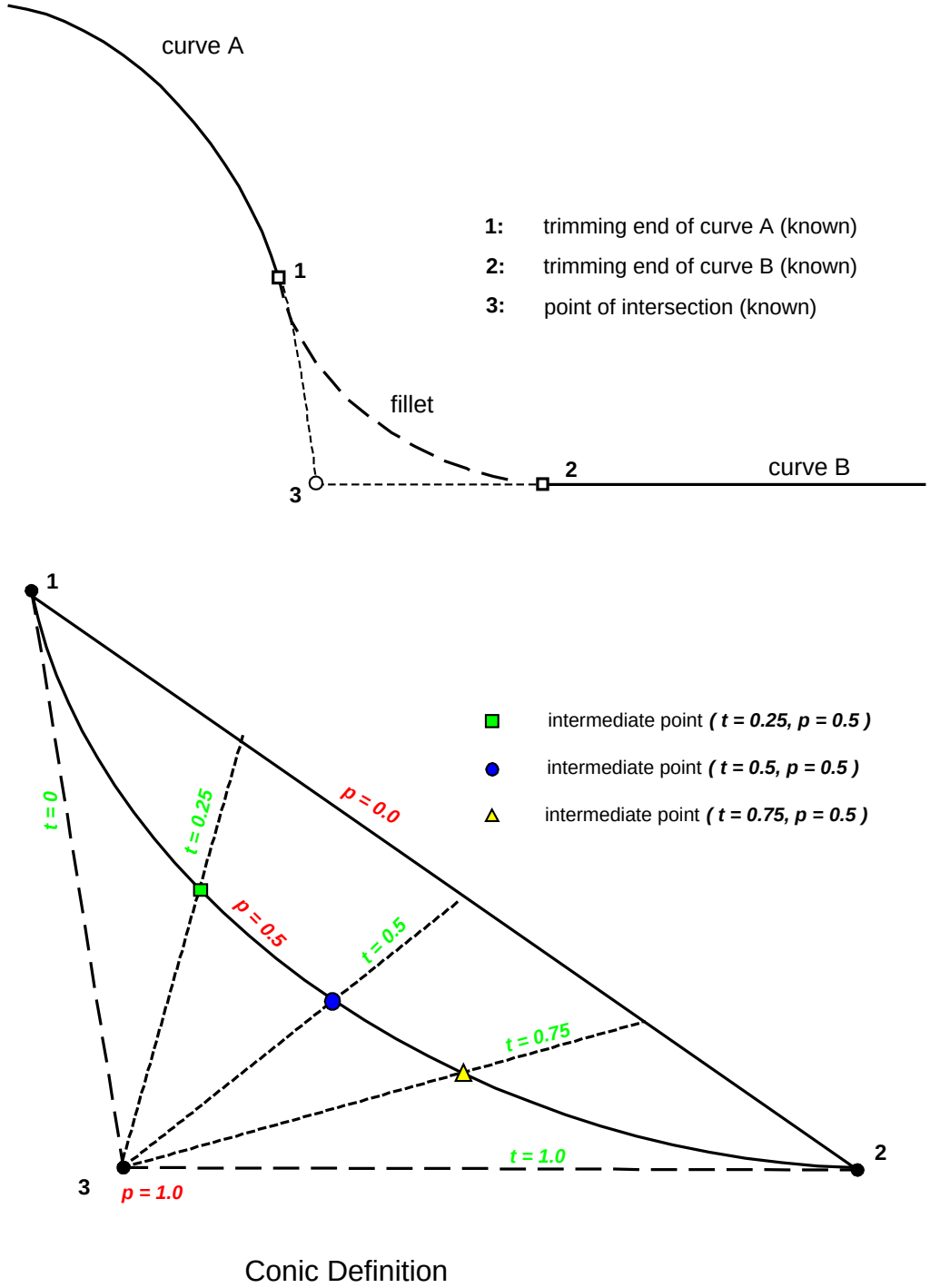


Figure 37 A conic definition used to find intermediate points for the fillet

6.3 Surface Filleting

In this research, one-dimensional filleting is used to blend two intersecting surfaces (fuselage and wing) into one C^2 continuous non-uniform bicubic B-spline surface. The term of one-dimensional filleting is defined as blending two intersecting surfaces in one parametric direction along iso-parametric curves [Jone91]. Figure 38 demonstrates the concept of iso-parametric curves. An iso-parametric curve is the curve generated at a constant parametric value u or v. A blending surface can be interpolated along iso-parametric curves in the same manner as the curve filleting process discussed in the previous section. However, the model surfaces might not have been one-dimensional before geometric trimming. In this case, some conversion needs to be performed to change the model surface into one-demensional.

Blending two trimmed surfaces along iso-parametric curves requires these surfaces to have the same number of iso-parametric curves in the parametric direction. The previous chapter has discussed the process of geometric trimming. There is the same amount of trimming data describing the common portion of the trimming curves where the fuselage intersects the wing. In this research, the number of trimming data points is defined as 14 before making the offset. After offset, the number is changed to 15 as shown in Figure 39.

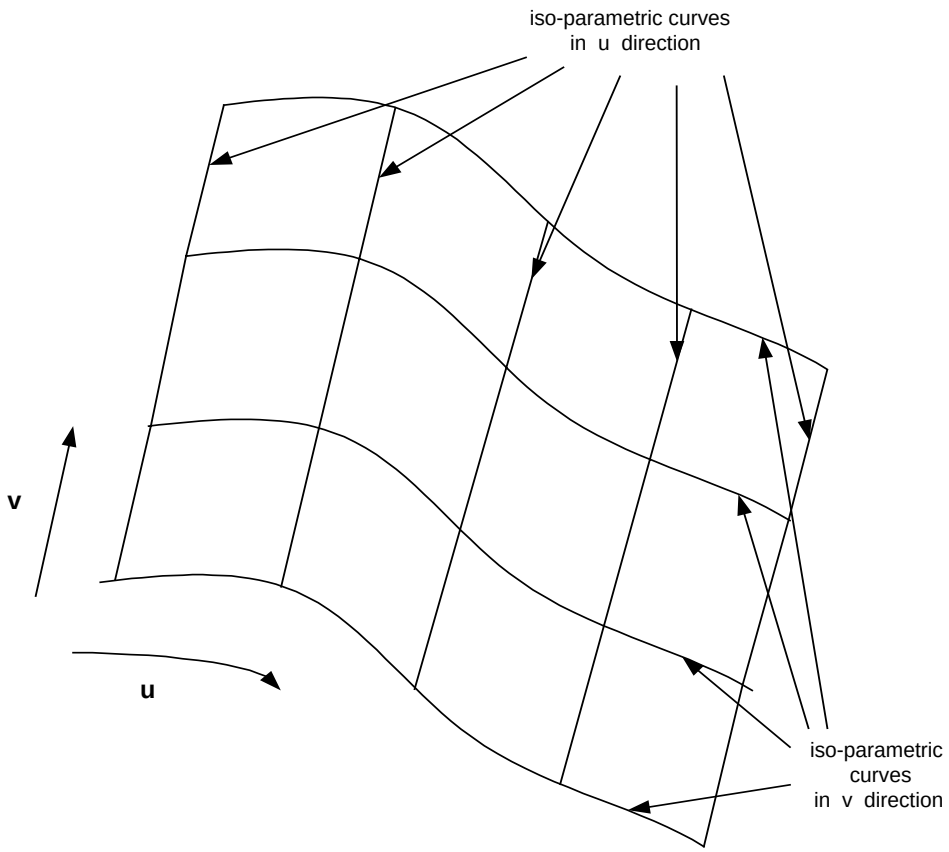


Figure 38 Definition of iso-parametric curves

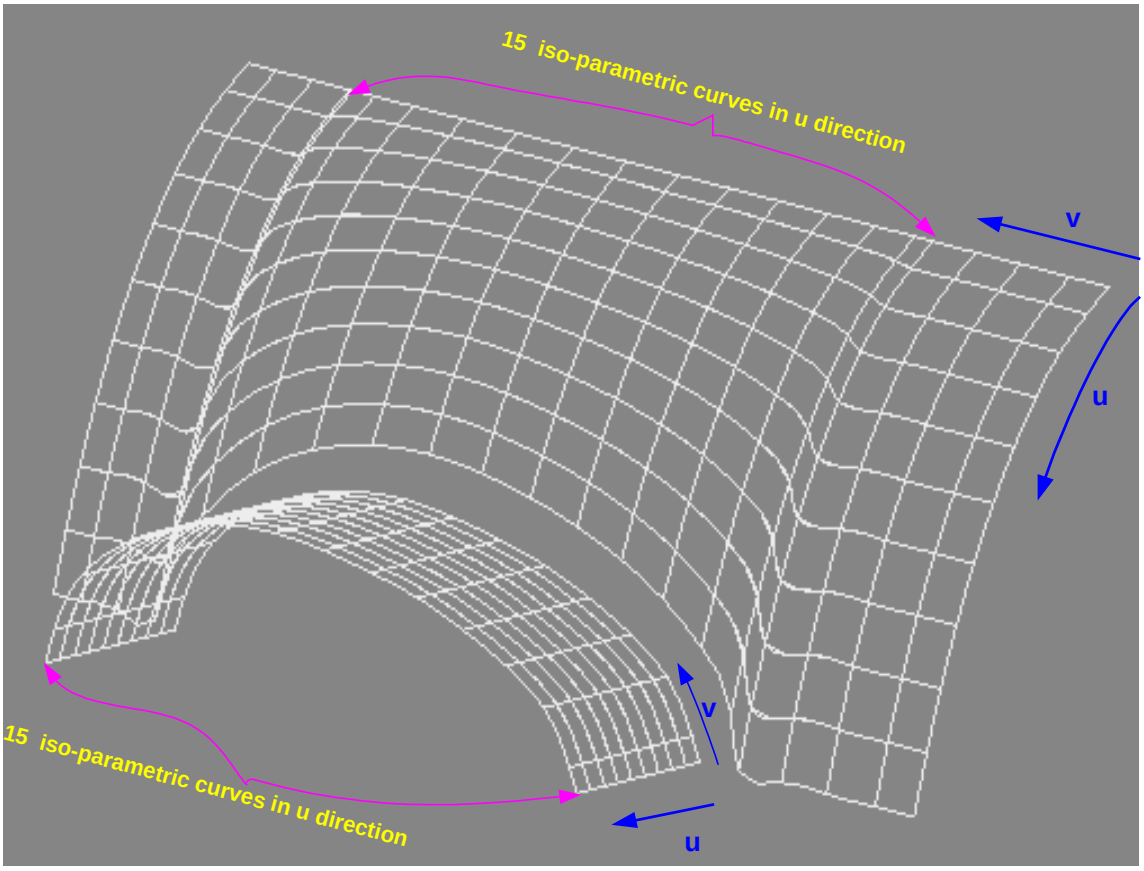


Figure 39 Same number of *u* iso-parametric curves for surface interpolation

The blending surface will be interpolated along these ***u*** iso-parametric curves. Firstly the trimmed fuselage is subdivided into three patches: left patch, middle patch and right patch. The middle patch will be used to generate the new blending surface with the trimmed wing patch as shown in Figure 40.

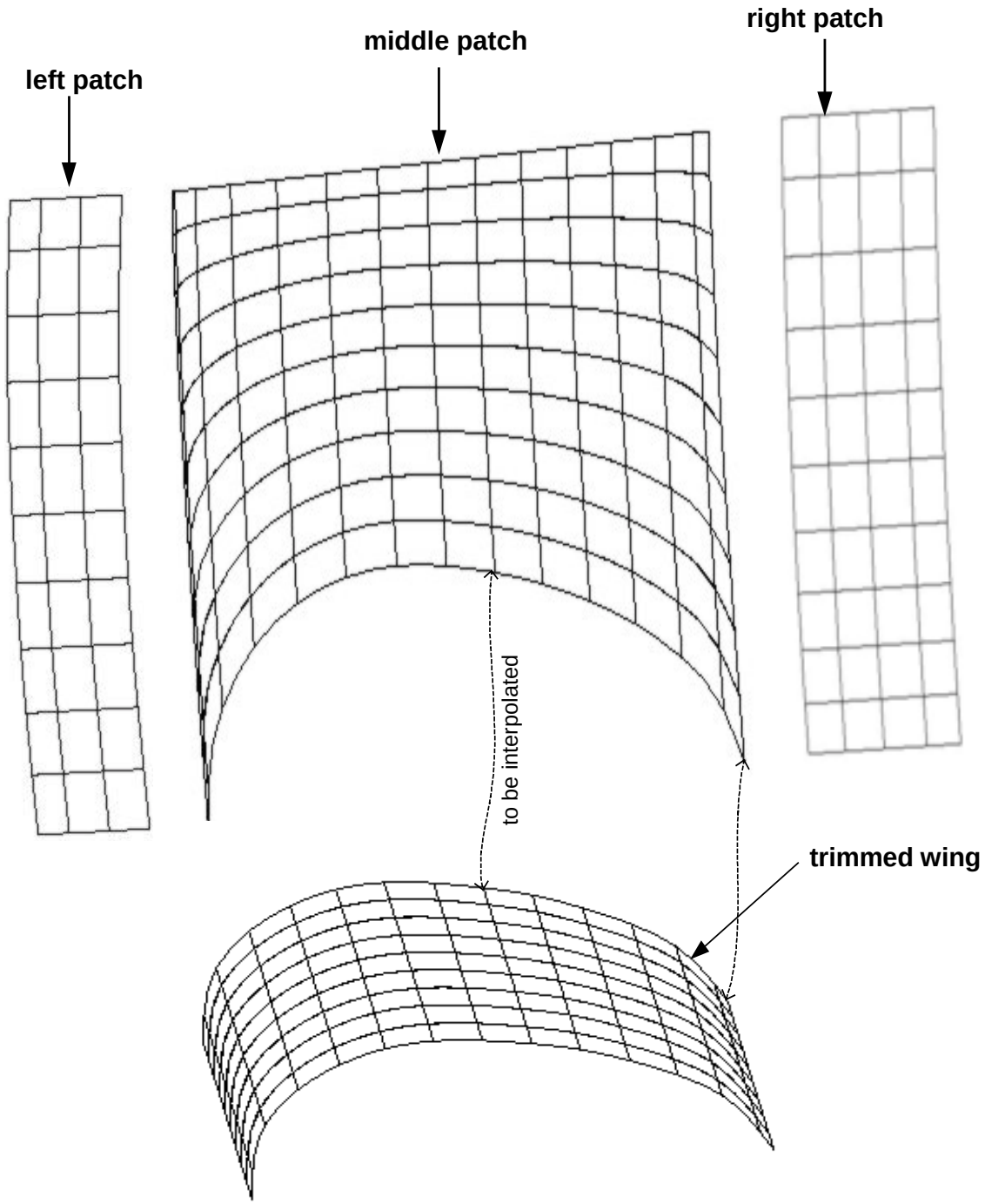


Figure 40 Subdividing the trimmed fuselage into three patches

The middle patch and trimmed wing make up a rectangular grid of data points, which are interpolated to create a new non-uniform bicubic B-Spline surface. Since the knot spacing in each parametric direction is not identical, as is mentioned earlier the chord length parameterization is used for generating the basis functions. In programming code, the continuity constraints for both parametric directions are equal to two in order to make the new surface C^2 continuous.

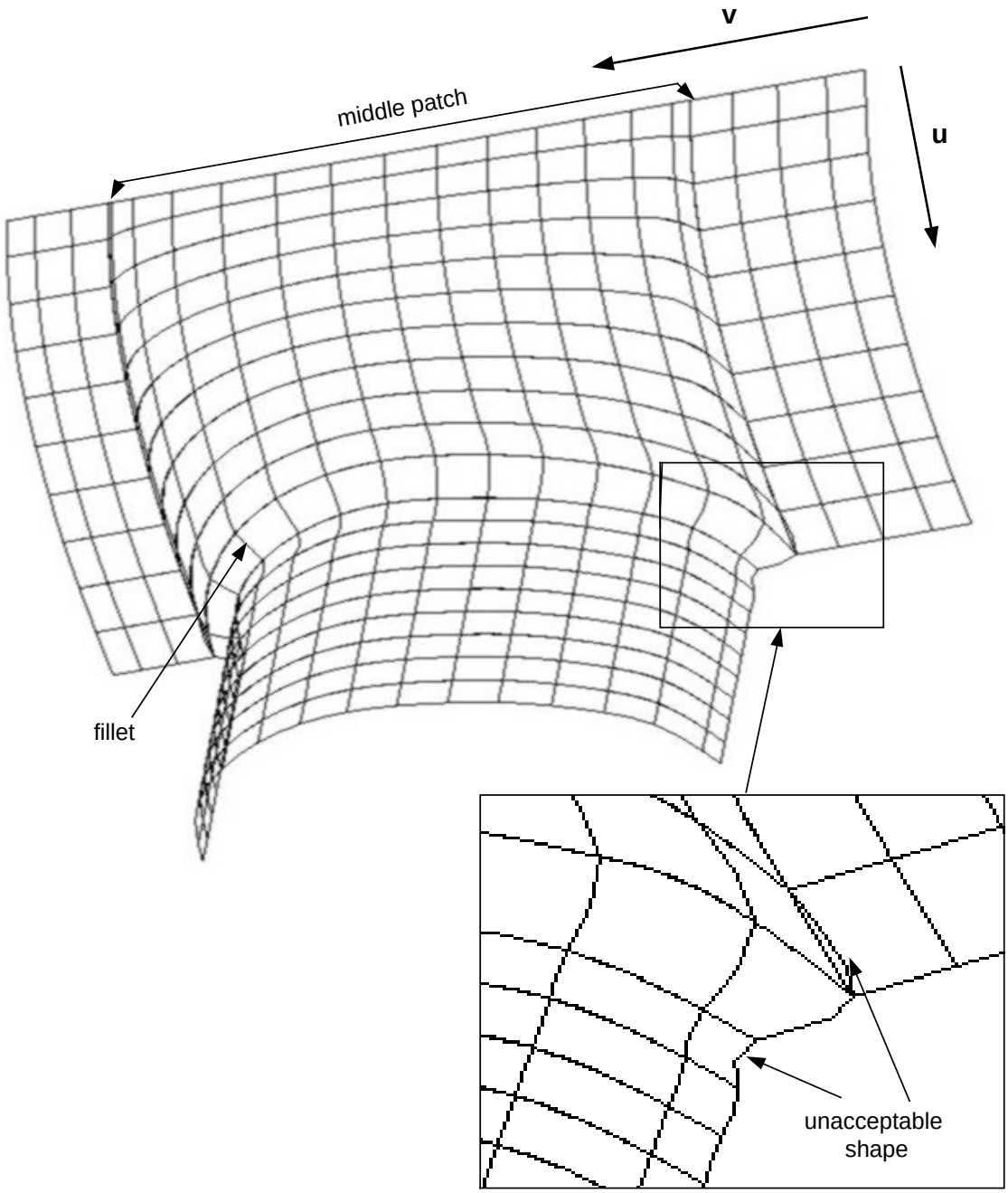


Figure 41 Interpolating the middle patch and the trimmed wing

The enlarged picture shows the ***u*** iso-parametric curves do not match the original curves well near the trimming boundaries. Three more ***v*** iso-parametric curves need to be added near the trimming boundaries. The addition procedure is performed in the same way as curve filleting

addressed in the previous section. The procedure involves firstly inserting three knots in the trimming end interval of each ***u*** iso-parametric curve in parametric space as shown in Figure 42, secondly computing the new control net for the trimmed surface, thirdly calculating a new set of data points for interpolation purpose. This process generates three new ***v*** iso-parametric curves near the trimming boundary. Figure 43 shows a trimmed wing with three ***v*** iso-parametric curves added around the trimming end.

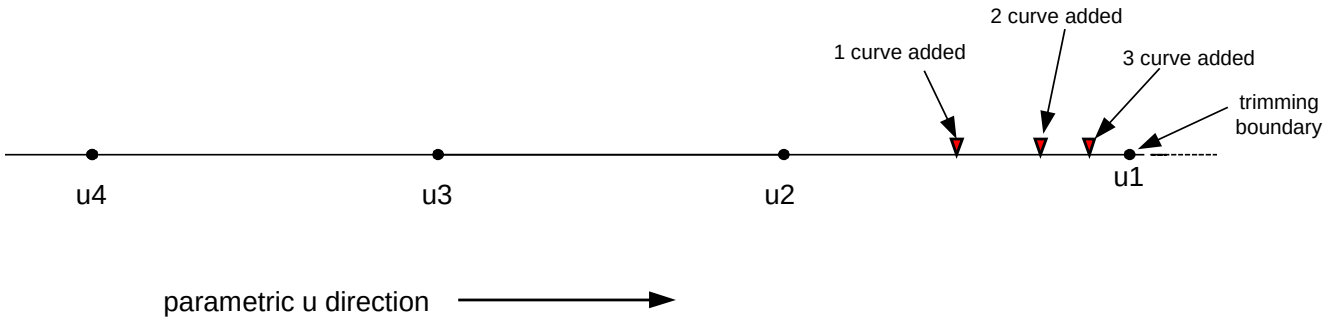


Figure 42 Inserting three knots near the trimming boundary

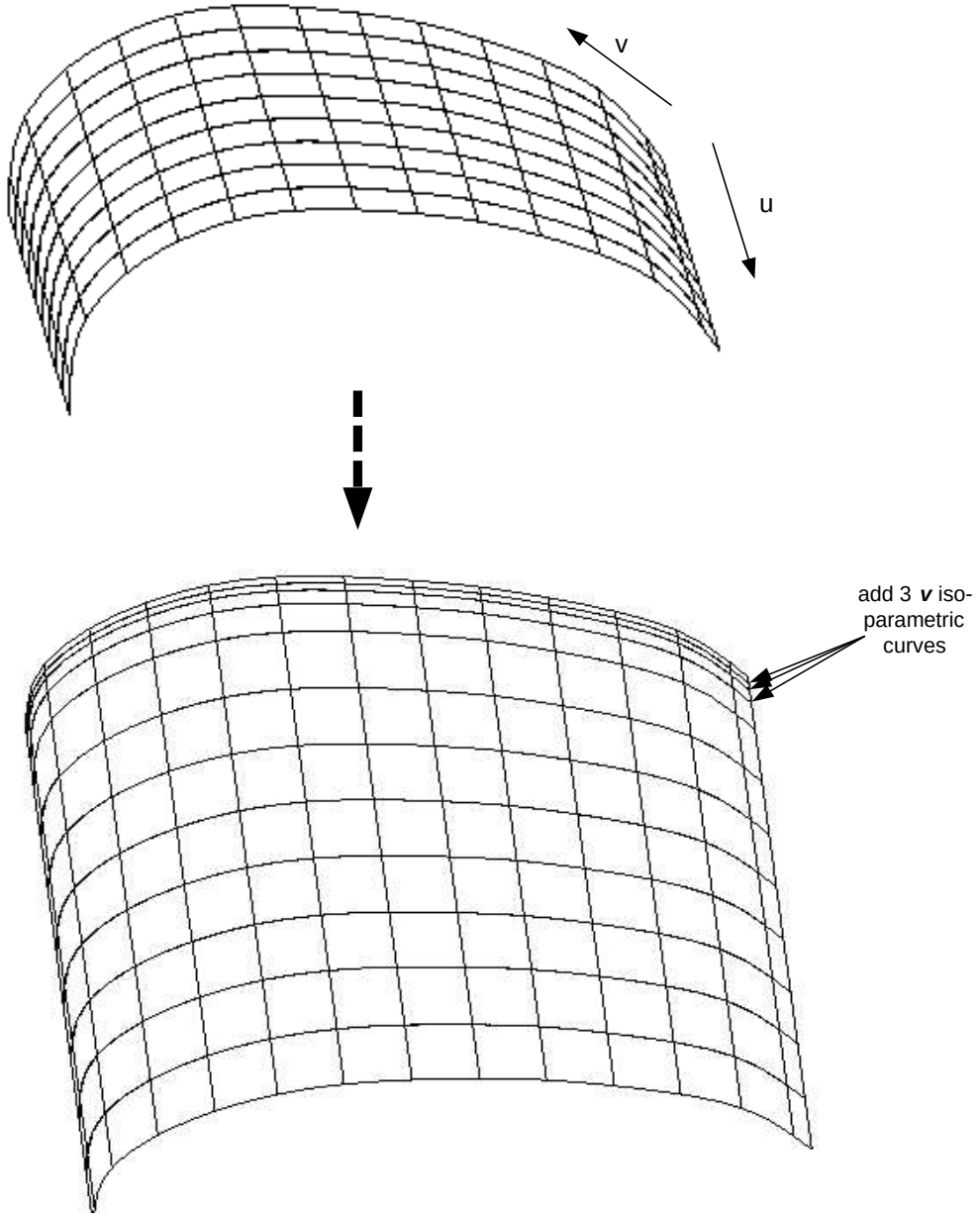


Figure 43 Trimmed wing with 3 *v* iso-parametric curves

These three ***v*** iso-parametric curves are used to force the ***u*** iso-parametric curves to fit the corresponding original curves as displayed in Figure 44.

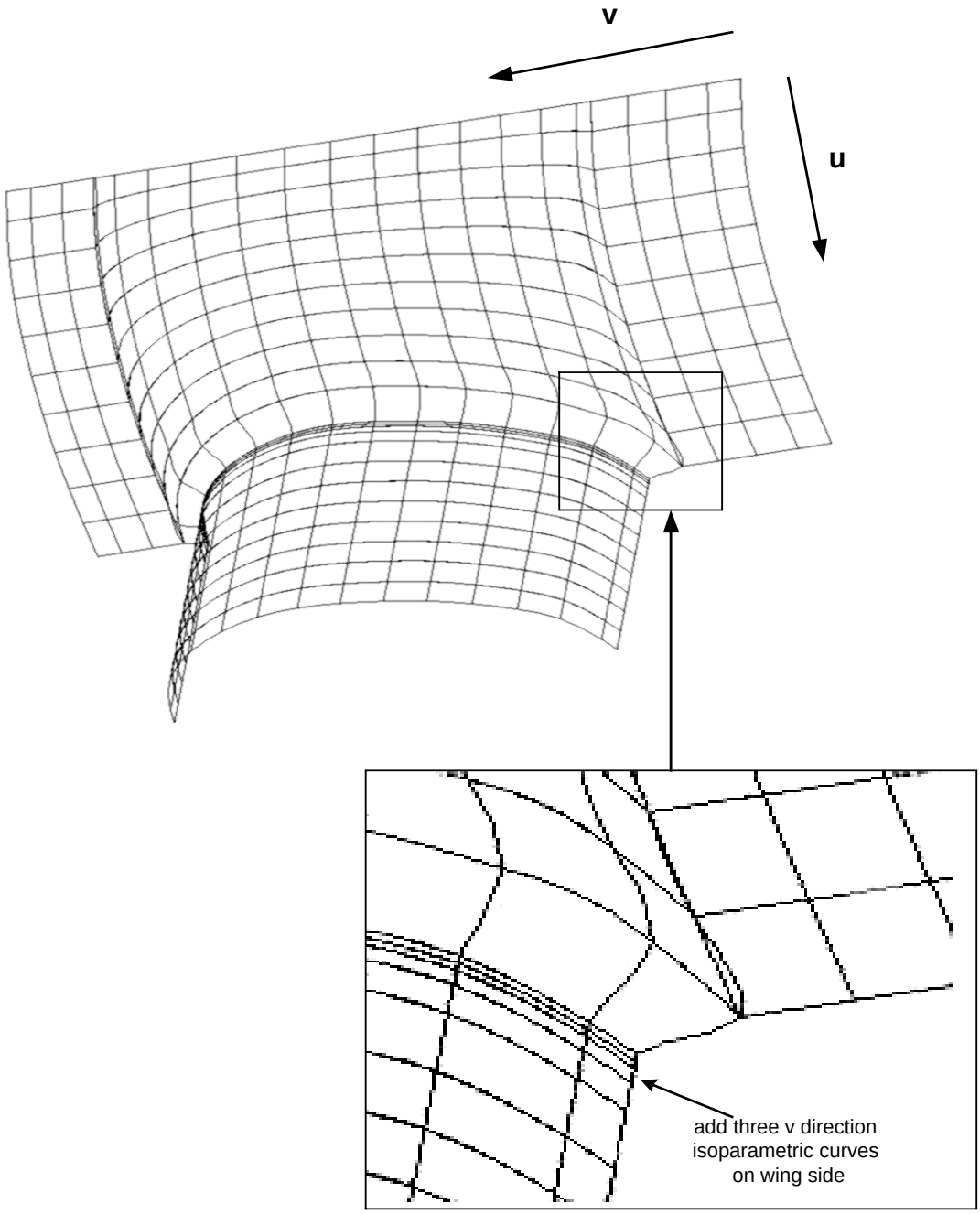


Figure 44 Inserting three *v* iso-parametric curves on the trimmed wing

Another three ***v*** iso-parametric curves are added on fuselage side as shown in Figure 45.

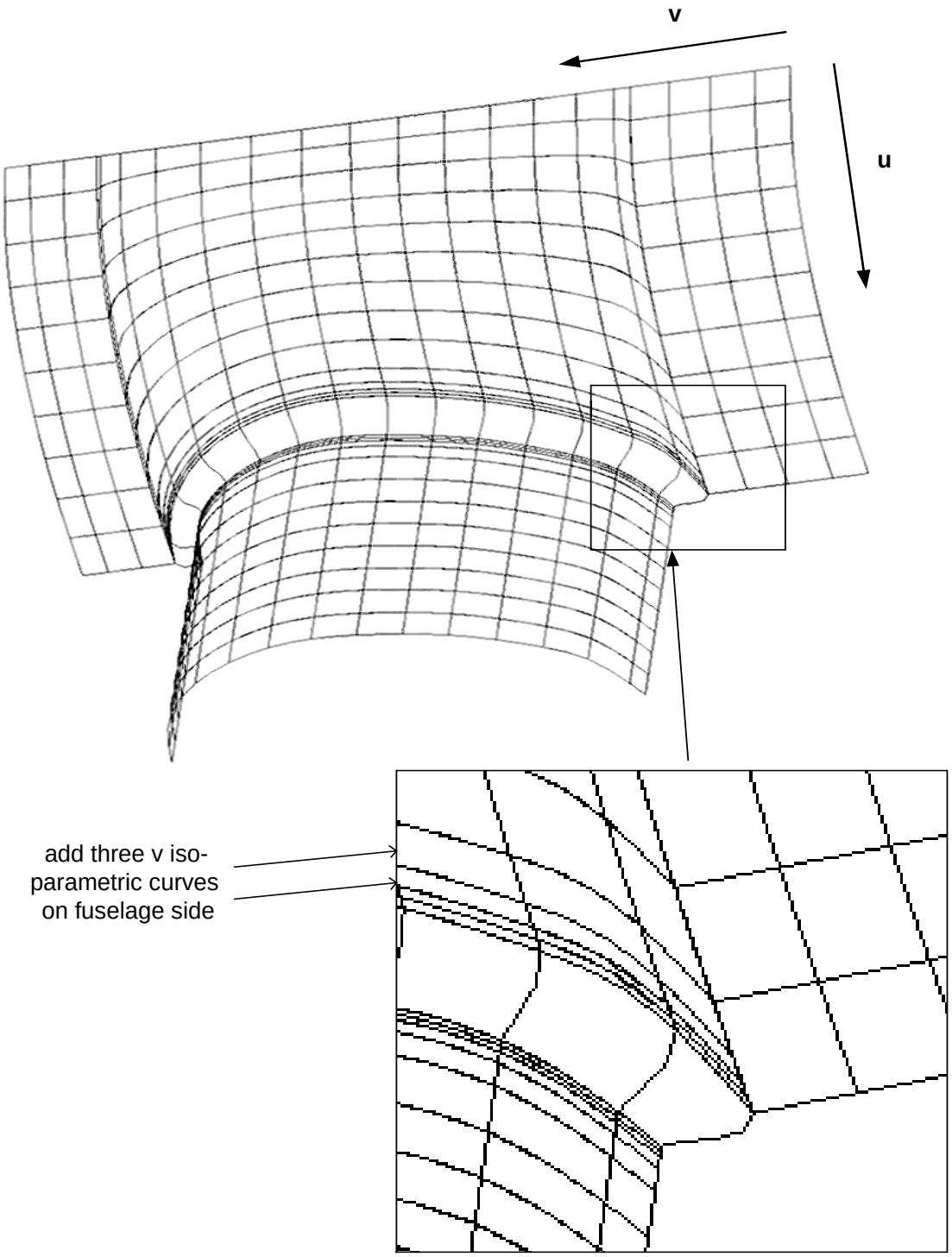


Figure 45 Inserting three *v* iso-parametric curves on the trimmed fuselage

One or more extra ***v*** iso-parametric curves are added between two trimming boundaries in order to control the shape of fillet. Figure 46 shows that one curve is added. The curve is generated from a set of intermediate points, which are computed from a set of conic curves. The process has been described in the previous section. The following figure exhibits one ***u*** iso-parametric curve as an example.

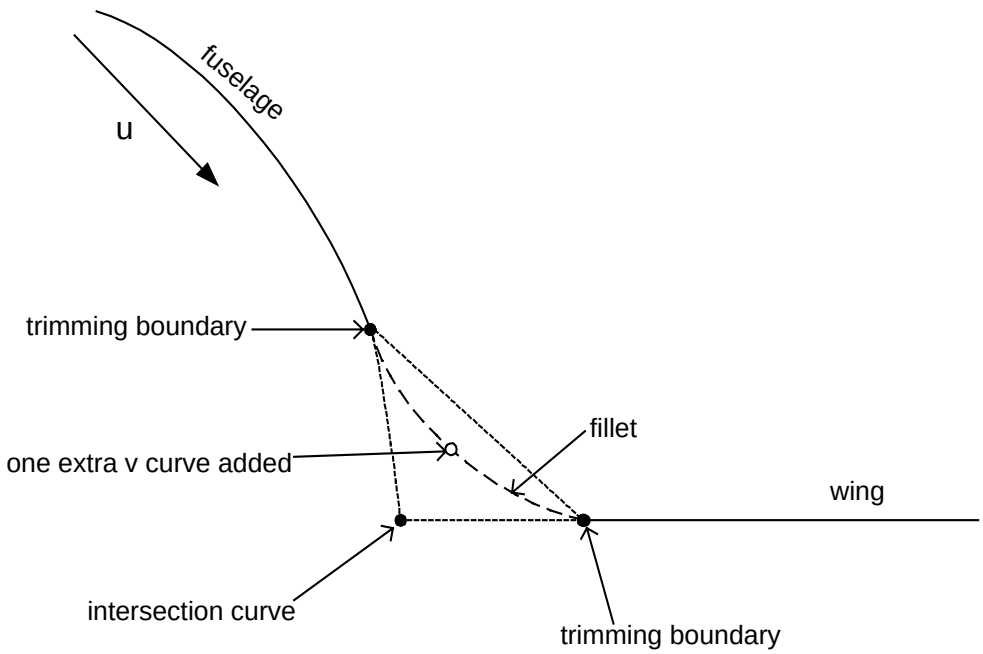


Figure 46 Adding one *v* iso-parametric curve to control the shape of fillet

If necessary, more curves can be calculated to make the fillet fit a conic curve. Figure 47 explains how to compute more intermediate points from a conic. In the same way, more ***v*** iso-parametric curves can be generated. Figure 47 displays that one ***v*** iso-parametric curve is added to make the fillet in a desired shape.

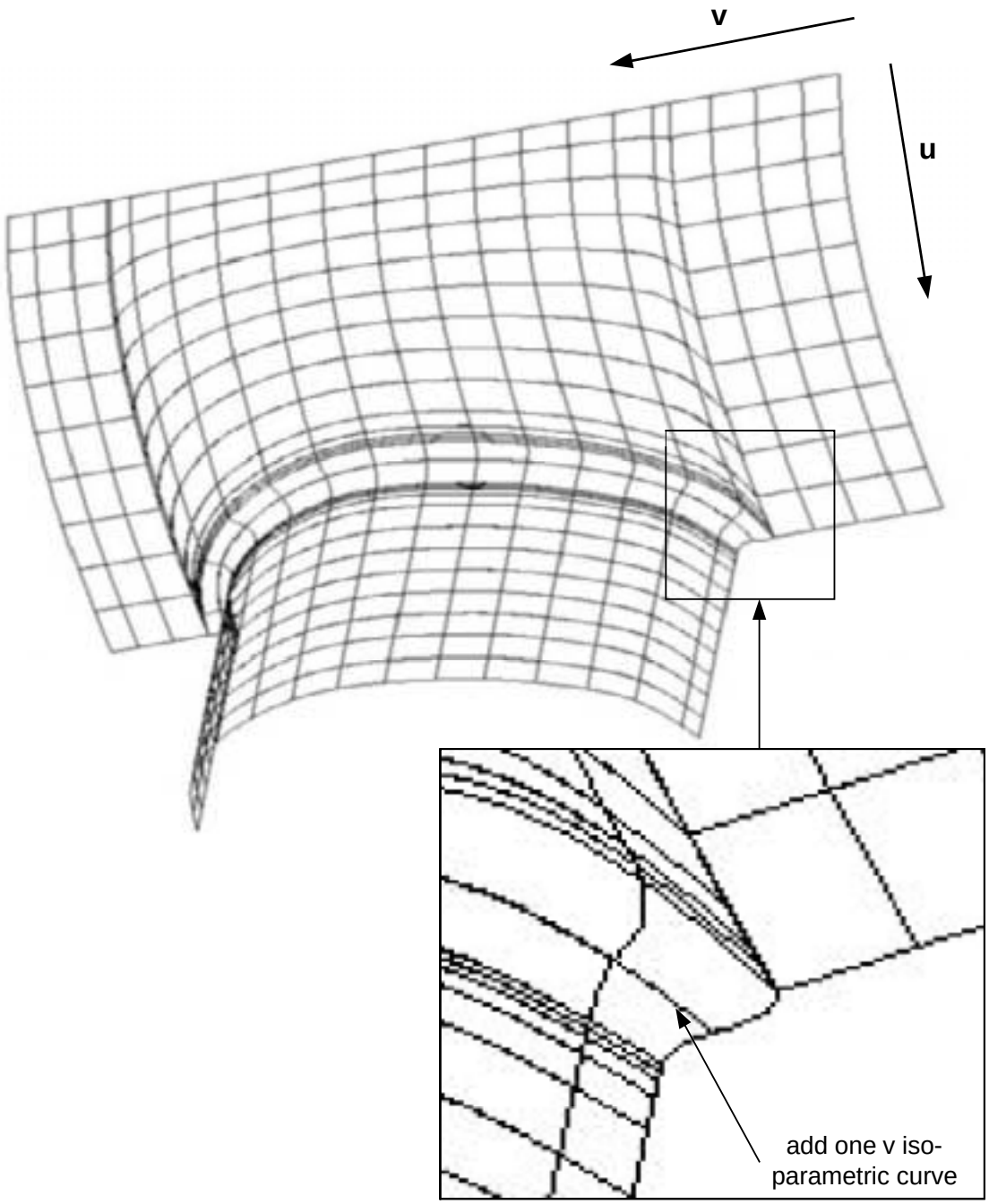


Figure 47 Adding one *v* iso-parametric curve to control the shape of fillet

From the zoom-in figure above, it is found that ***u*** iso-parametric curves present a minute convex bugle around the trimming boundary of the fuselage side. This is because the ***u*** iso-parametric curves of fuselage and wing are not aligned with each other very well when interpolating. The alignment is especially bad on the two sides of the blending surface. The problem can be minimized by adding more intermediate ***v*** iso-parametric curves between the two trimming boundaries. Typically, the more intermediate curves that are added, the closer to a conic the fillet is. Figure 48 exhibits the new blending surface for the upper half portion of the original fuselage and wing. Visually the new surface matches the original very well in the common portion. However, there is a concern that the new surface is unacceptably different from the original one. In order to check whether it is a major problem or not, a mathematical comparison will be used to determine the approximation error between the new surface and the original surface. An error approximation tool by Jain is imported to facilitate this comparison [Jain99]. This process will be explained later.

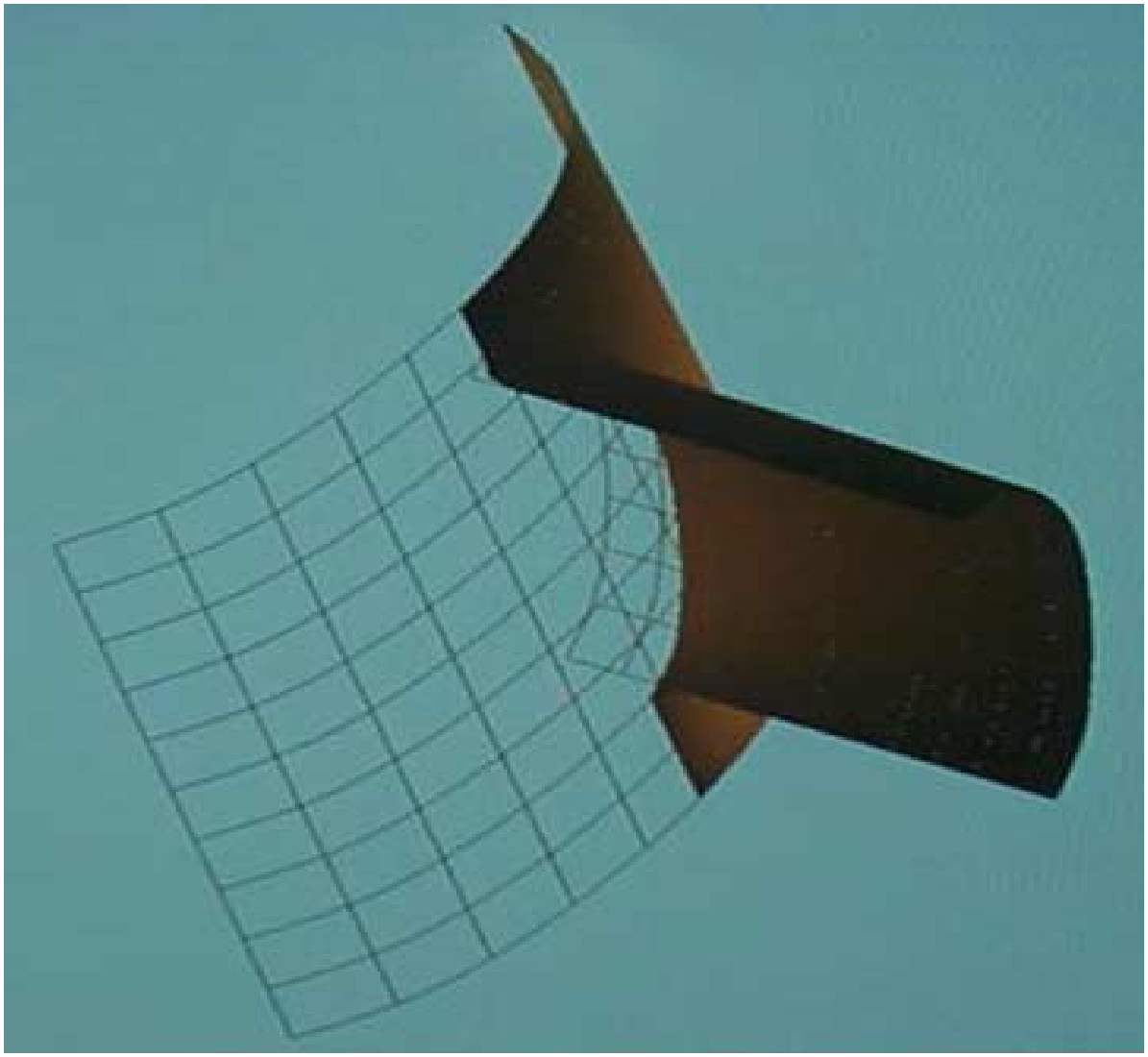


Figure 48 Blending surface (shaded) and original surface (grid)

Following the same steps above, the lower portion of the original fuselage and wing is generated. As shown in Figure 49, the left patch can be combined with the upper left fuselage patch into one patch called left. The same is done for the right patch. Finally, four surface patches are achieved: left, right, upper and lower. On the connecting boundaries they share the same surface data points. These four patches are mutually connected and they make up the whole blending

surface, and each of them is mathematically defined in its own parametric space. The unwanted portion of original fuselage and wing are completely discarded with the use of geometric trimming. The new blending surface is made up of the wanted portion and is fully mathematically defined. Therefore, the whole blending approach is called "geometric trimming and surface blending."

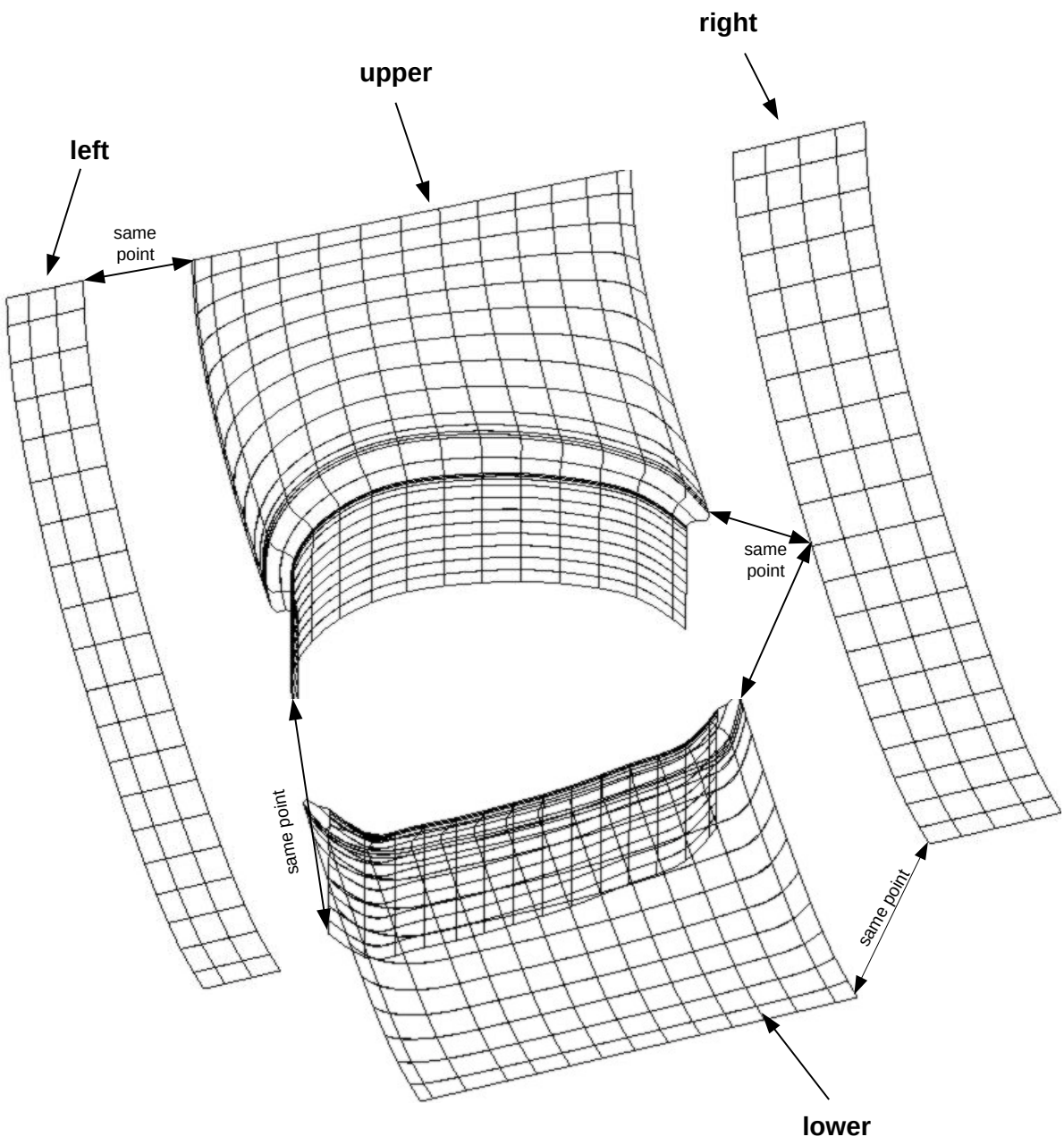


Figure 49 The whole blending surface is made up of four patches

Figure 50 displays the shaded image of the whole blending surface.

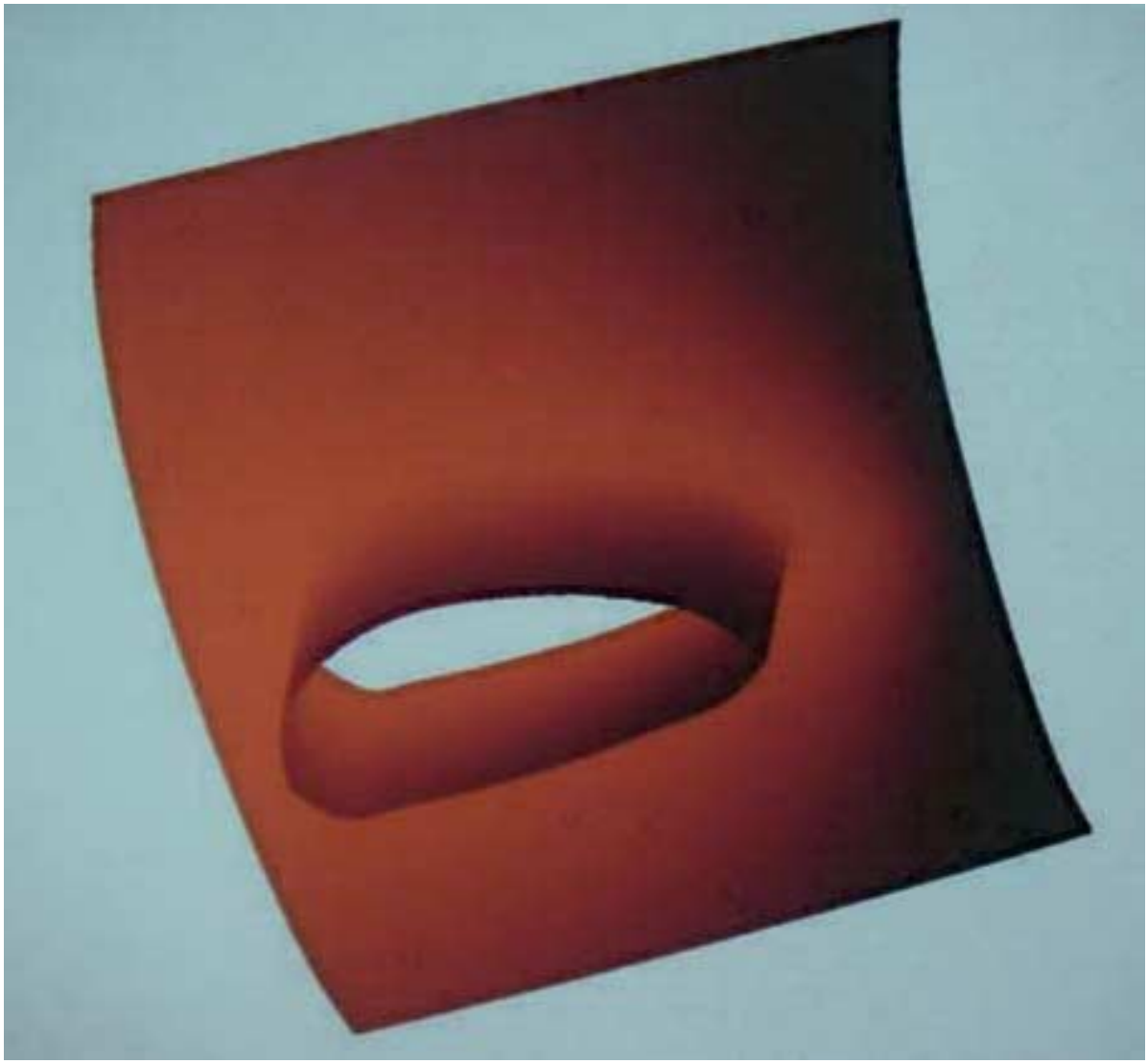


Figure 50 Shaded image of blending surface for aircraft fuselage and wing

7.0 Results

7.1 Visualization Tool Kits

In order to get visual feedback about the results of geometric trimming and surface blending, a visualization tool kit was employed and modified from the prototype developed Bindiganavle [Bind00]. As shown in Figure 51, the platform supports the capabilities such as rotating and zooming the models, rendering multiple surfaces, and applying smooth shading for better visualization to the shape of free-form surfaces. The users interface is developed in the C++ programming language with the 3D graphic standard OpenGL and X/motif. OpenGL graphics system is a software interface to graphics hardware [Woo97], where the GL stands for Graphics Library. OpenGL routines simplify the development of graphics software, which can render a simple geometric point, line, or filled polygon and create the most complex lighted and texture-mapped NURBS (Non-Uniform Rational B-Spline) curved surface. OpenGL gives the designer access to geometric and image primitives, modeling transformations, lighting and texturing, blending and many other features. The NURBS routines are used to create the curves and surfaces in this research, they require the knowledge of control points, knot sequences, the order of NURBS objects and the type of the evaluators. X/motif is a library of ready-to-use user-interface elements developed in the C programming language. It facilitates the creation of user-interface components of the X windowing system such as menus, windows, command buttons, etc.

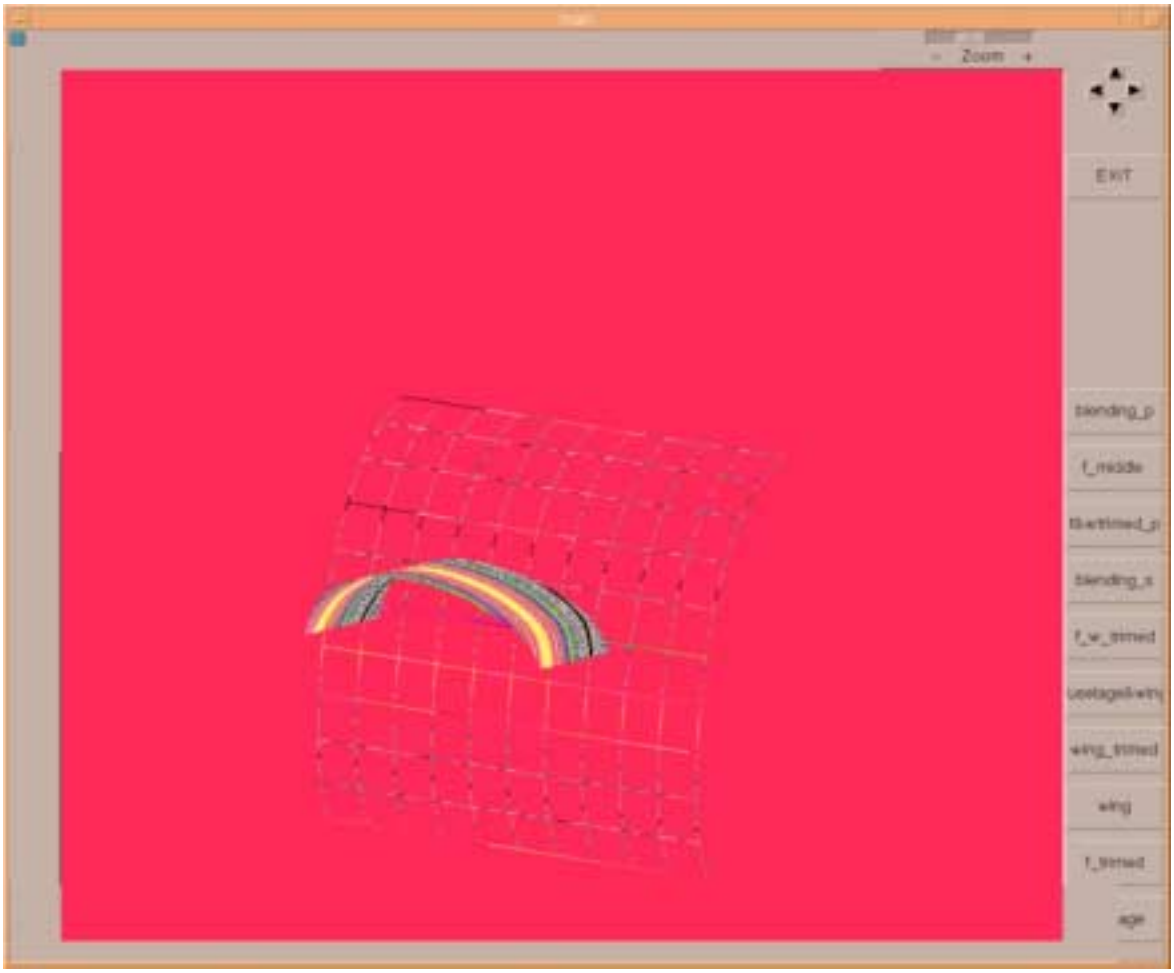


Figure 51 Visualization tool kit

7.2 Geometric Trimming of Fuselage and Wing

This research provides an approach to manipulate the surface operations for a typical aircraft fuselage and wing. The approach includes surface intersecting, geometric surface trimming and surface filleting. The data for surface intersection comes from a set up file. Rojas' geometric trimming method is employed to make the surfaces trimmed. This method utilizes Fleming's continuity constraint based interpolating algorithm to generate a new set of B-spline surface patches approximating the untrimmed portion of the original surface. It has been proved very accurate when used in an open trimming curve. In the case of closed trimming curve, a conversion procedure is developed to change the closed trimming curve into open trimming curve for geometric trimming as addressed in Chapter 5.

An intuitive way to determine how well the newly created patch resembles the original surface is simply displaying both surfaces on the computer screen. However that only gives the designer a visual feedback of how closely the new surface approximates the original surface. It is hard to tell how accurate the newly created surface matches original one. In our case, shaded and grid images are given to the newly created surface and the original surface (see Figure 48). However, the visual image can not give us any numerical information about the closeness of the new surface and the original.

A comprehensive error estimation tool was developed by Jain [Jain99], who uses positional error comparison to characterize the difference of two matching surfaces. By plotting the ordinate intercept differences against the corresponding parametric values, positional error is proposed to be a good way of telling the numerical difference of two surfaces. The following section demonstrates the position error analysis for fuselage and wing.

- The error visualization tool developed by Jain is used to compare the positional difference between the original fuselage surface and newly created surface against the corresponding parametric values as shown in Figure 22. The positional difference is the absolute distance of two three-dimensional points, one of which is from the original surface and the other is from the trimmed surface, both of them are mapped by using the knot insertion algorithm stated earlier. Figure 52 displays the positional error plot for the trimmed fuselage.

- A comparison of the positional difference between the original wing surface and newly created surface versus the corresponding parametric values is shown in Figure 25. Figure 53 displays the positional error plot for the trimmed wing.

From the error analysis, it can be concluded that the positional difference between the new patch and the original tends to oscillate about the original one with larger amplitudes close to the trimming curve. Especially, the oscillation is bigger on the area where the trimming curve has large curvature changes. Figure 54 shows the trimming curve has a big curvature change on the corner where curve and straight line meet. Rojas [Roja94] described the geometric trimming of B-spline surfaces for various trimming curve cases. His method gives the best result when the trimming curve is a straight line and perpendicular to ***u*** direction in parametric space. The magnitude of error progressively increases as the trimming curve deviates from perpendicular. This results from the constraint-based inversion algorithm, in which only the parametric ***u*** direction continuity constraints can be specified at the trimming curve in order to not disrupt the overall continuity of the remaining surface. In this research the wing has a more straight and perpendicular trimming curve in parametric space than fuselage does, so it presumed to have less positional difference. In fact, the wing has the maximum percentage error 0.072%, while the fuselage has the maximum percentage error 0.996%. Thus, the maximum radius deviation from the original fuselage is less than one percent. The trimming method employed in this research is deemed acceptable for surface approximation. If the trimming curve is more straight and more perpendicular to the ***u*** parametric direction, the trimming and blending procedure addressed in this research should be more accurate as illustrated in Figure 54.

-

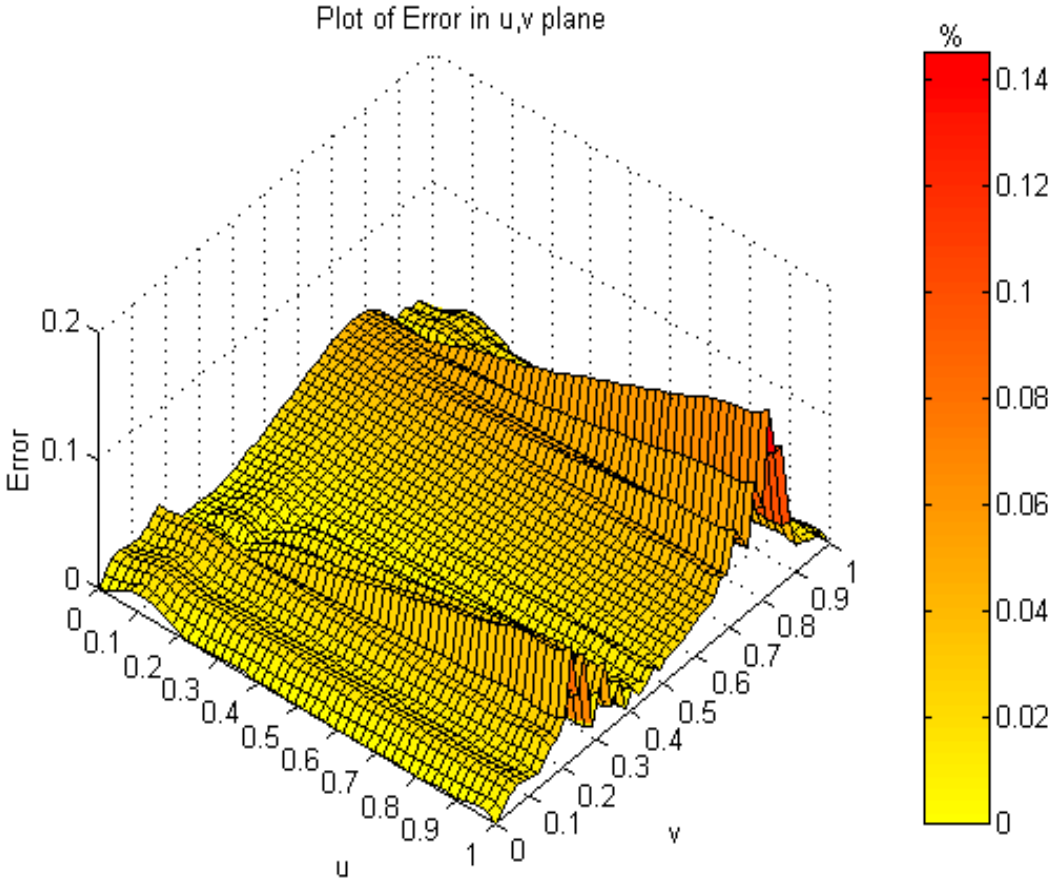


Figure 52 Positional error plot for the trimmed fuselage

```
The maximum percentage error is 0.996%

The actual size of the model surface:

The length of the bounding box along x axis = 0.154 inches
The length of the bounding box along y axis = 0.294 inches
The length of the bounding box along z axis = 0.270 inches
```

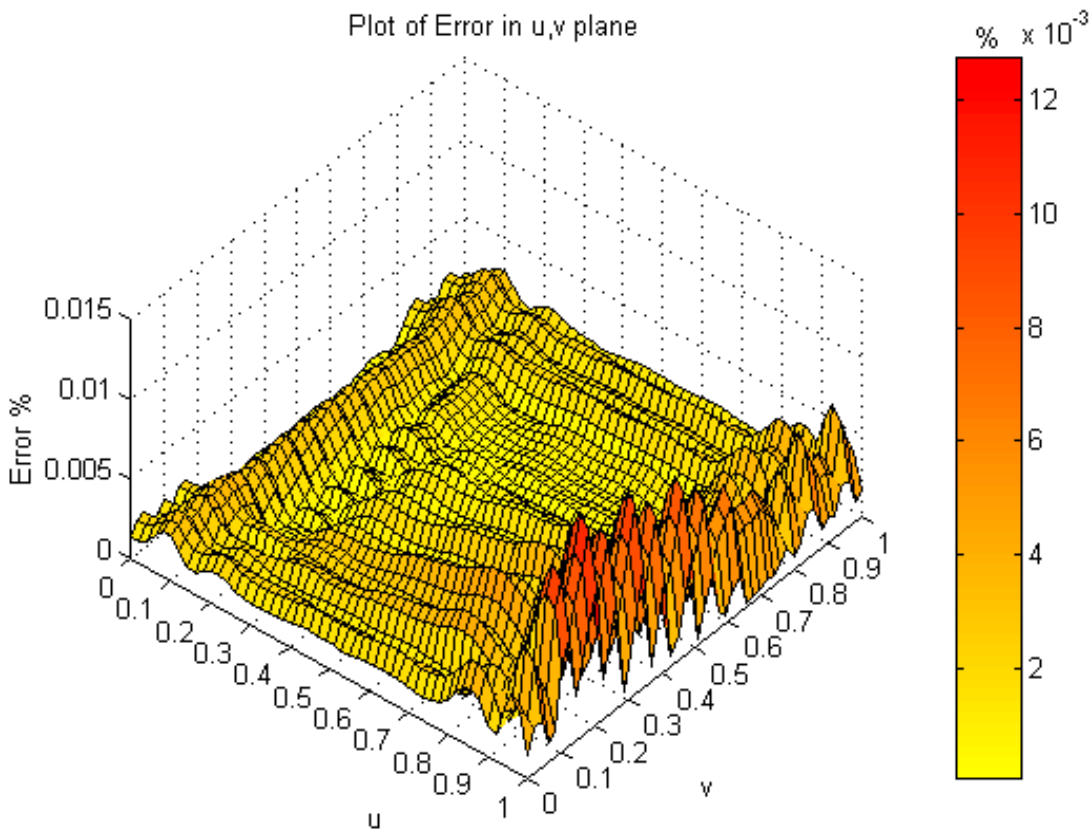


Figure 53 Positional error plot for the trimmed wing

```
The maximum percentage error is 0.072%

The actual size of the model surface:

The length of the bounding box along x axis = 0.156 inches
The length of the bounding box along y axis = 0.016 inches
The length of the bounding box along z axis = 0.188 inches
```

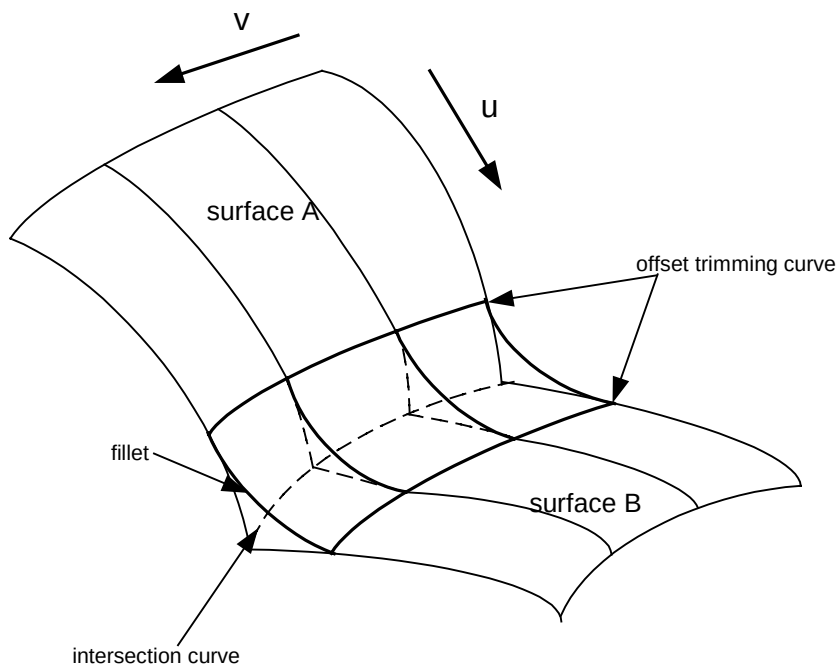


Figure 54 Trimming curve is near straight and perpendicular to u parametric direction

The offset trimming curves lying on surface A and surface B are more perpendicular to the ***u*** direction in parametric space. In addition, one-dimensional filleting works well in this case, since all *u* iso-parametric curves of two surfaces are aligned with each other.

7.3 The Property Analysis of The Blending Surface

7.3.1 Introduction

Intuitively, it is hard to tell whether the shape of the curve is acceptable or not from the display on the screen, because curves and surfaces may look very smooth on the screen while they are not so numerically. Actually, the bicubic B-Spline surface is guaranteed to be C^2 continuity when it is generated using constraint-based B-Spline inversion process. To double-check the geometric quality of curves and surfaces, the plot of curvature and second-erivative are generated to how smooth curves and surfaces are mathematically. Curvature is the rate of turning of a unit tangent vector with respect to the length of a curve [Yama88]. It is a tool of measuring how rapidly or slowly a curve is turning. If each iso-parametric curve of a surface has a continuously changing curvature, it can be concluded that the surface is curvature continuous. Figure 56 shows the middle portion of the blending surface. The iso-parametric curves from this surface are selected to make curvature and second-derivative analysis, since the middle portion is the part that is actually blended.

7.3.2 Curvature and Second-derivative

A curve can be described as a vector using the parameter t:

$$P(t) = [\ x(t)\ \ y(t)\ \ z(t)\ ]$$

The derivative of the vector is evaluated at $t = t_0$:

$$P'(t_0) = [\ x'(t_0)\ \ y'(t_0)\ \ z'(t_0)\ ]$$

If the condition $P'(t_0) \neq 0$ is satisfied, the curve is defined as regular at $t = t_0$. If the derivatives of $x(t)$, $y(t)$ and $z(t)$ of a curve $P(t)$ to order r exist and are continuous, and if the curve is regular, then the curve is said to be of class C^r.

The second-derivative of a point at t_0 on the curve is described as:

$$\ddot{p}(t_0) = \lim_{\Delta t \to 0} \frac{\dot{p}(t_0 + \Delta t) - \dot{p}(t_0)}{\Delta t}$$

The unit tangent vector is described as:

$$P'(s) = \frac{\dot{p}(t)}{\sqrt{\dot{p}(t)^2}}$$

The curvature is the derivative of the the unit tangent vector:

$$P''(s) = \frac{d}{dt}\left(\frac{\dot{p}(t)}{\sqrt{\dot{p}(t)^2}}\right)\frac{dt}{ds} = \frac{\ddot{p}(t) - \frac{\dot{p}(t)}{\sqrt{\dot{p}(t)^2}}\left(\frac{\dot{p}(t)}{\sqrt{\dot{p}(t)^2}}\ddot{p}(t)\right)}{\dot{p}(t)^2}$$

The second-derivative of the curve is defined as $P''(s) \equiv d^2P/ds^2$. At a specific location of the curve:

$$P''(s_0) = \lim_{\Delta s \to 0} \frac{P'(s_0 + \Delta s) - P'(s_0)}{\Delta s}$$

As shown in Figure 55, when the limit $\Delta s \to 0$, the vector $P'(s_0 + \Delta s) - P'(s_0)$ is perpendicular to the tangent vector $P'(s_0)$ and it points towards the center of the curvature of the curve.

The magnitude of $P''(s_0)$ is:

$$\left|P''(s_0)\right| = \lim_{\Delta s \to 0} \frac{\Delta\theta}{\Delta s} = \lim_{\Delta s \to 0} \frac{\frac{1}{\rho}\Delta s}{\Delta s} = \frac{1}{\rho} \equiv \kappa$$

where ρ is the radius of the curvature and κ is the curvature.

The form of the curvature vector can be expressed as:

$$\kappa = \frac{\left|\dot{p} \times \ddot{p}\right|}{\left|\dot{p}\right|^3} = \frac{\left|\dot{p} \times \ddot{p}\right|}{(\dot{p})^3} \tag{4.1}$$

where $\dot{p}$ and $\ddot{p}$ are the functions of the parameter *t*.

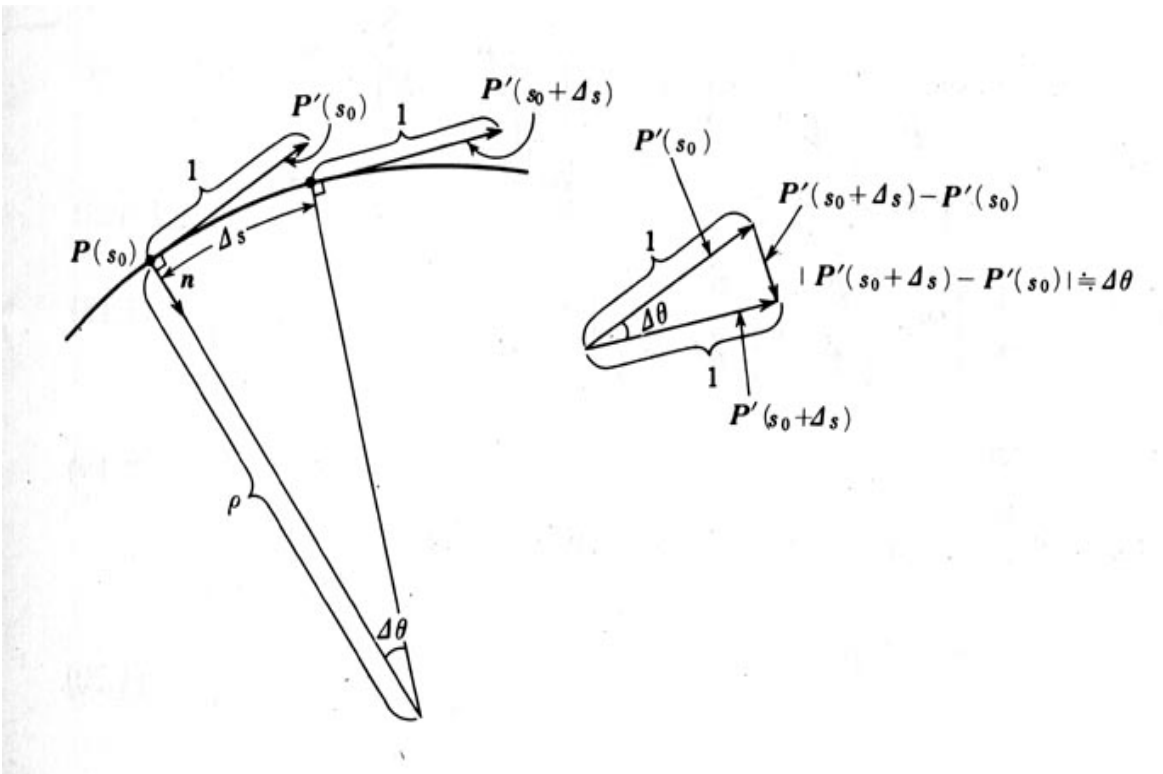


Figure 55 Geometric description of the Curvature of a curve [Yama88]

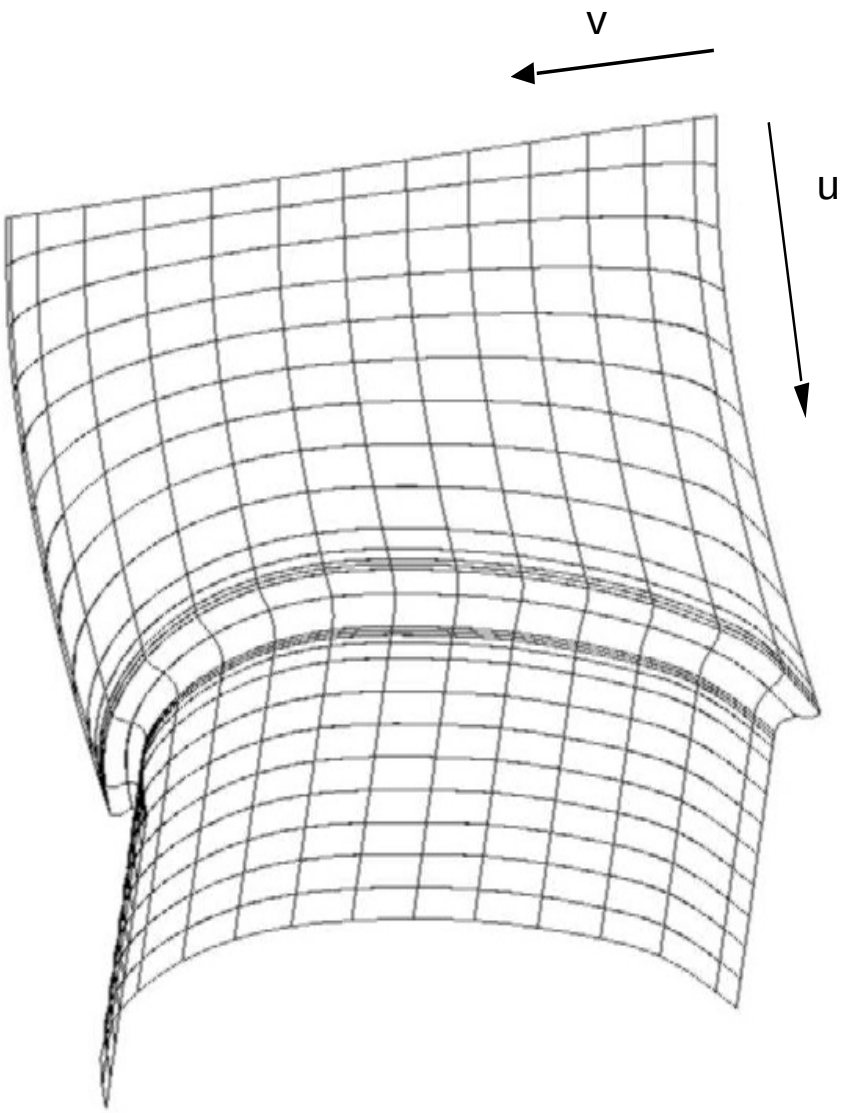


Figure 56 Middle portion of the blending surface

Curvature and second derivative analysis for ***v*** iso-parametric curves:

Select ***v*** iso-parametric curves $u = 7$, $u = 9$ and plot curvature change versus parameter value ***v***.

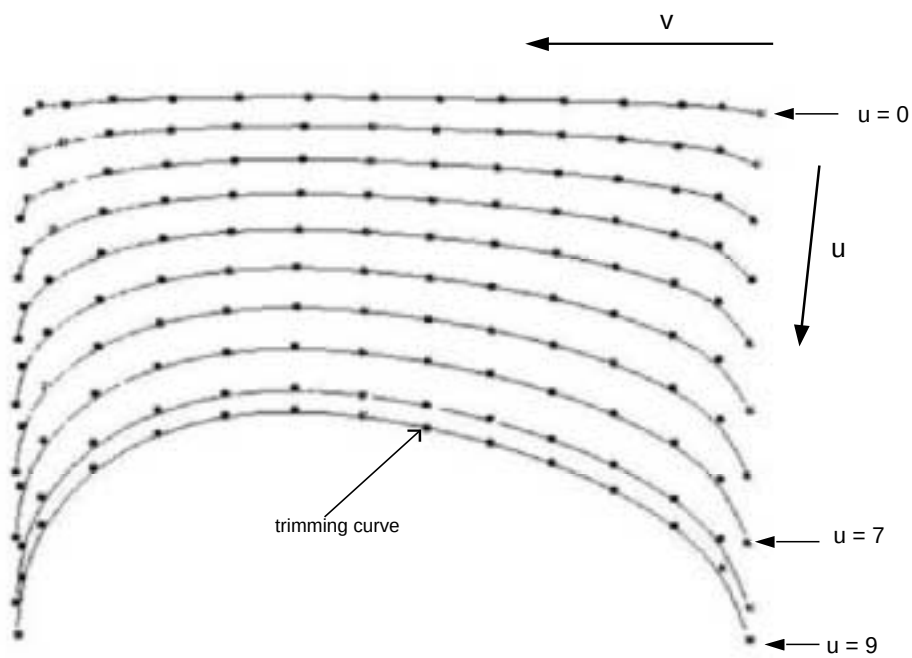
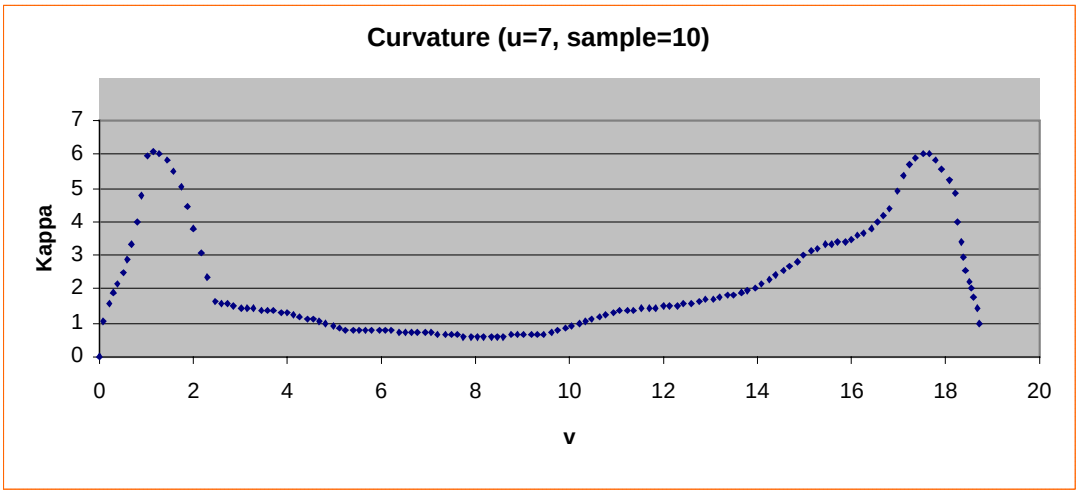


Figure 57 *v* iso-parametric curves on the fuselage side

The following figures present the curvature change of the ***v*** iso-parametric curves with respect to ***v*** values. The "sample" value is for mapping more points in any two neighboring parametric values. For example, if sample = 10, it means that any two adjacent parameter values are evenly divided by 10, then the corresponding points on curve are mapped and the curvatures of these points is calculated.



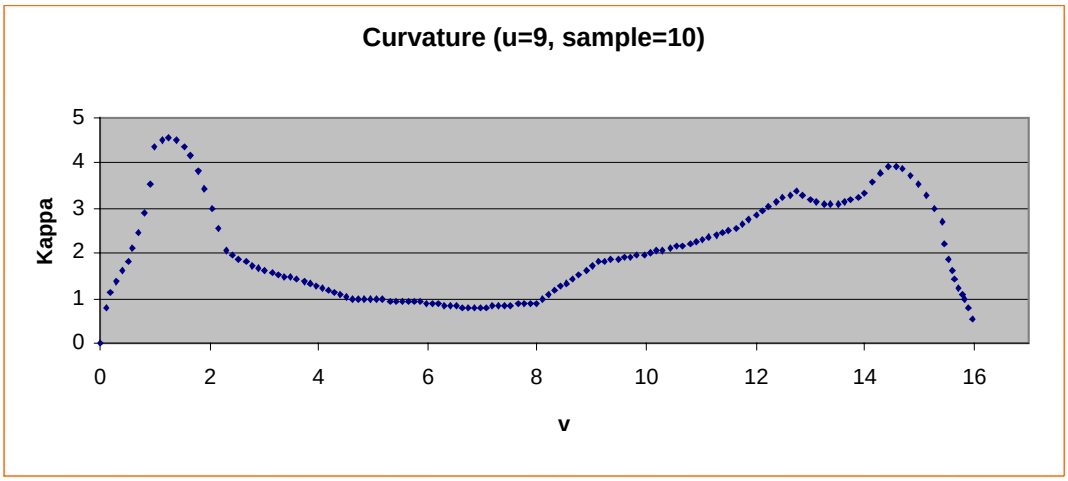


Figure 58 Curvature plot for *v* iso-parametric curves

From the figure above, it has been observed that the ***v*** iso-parametric curves are curvature continuous.

The following is a second-derivative analysis for the ***v*** iso-parametric curves:

When designing a smooth curve or a smooth surface, it is necessary to make sure that the curve segment or surface patches are smoothly connected. Mathematically continuity is another way to inspect the smoothness of curves and surfaces. The following figure displays the second derivative change against the ***v*** parameter value.

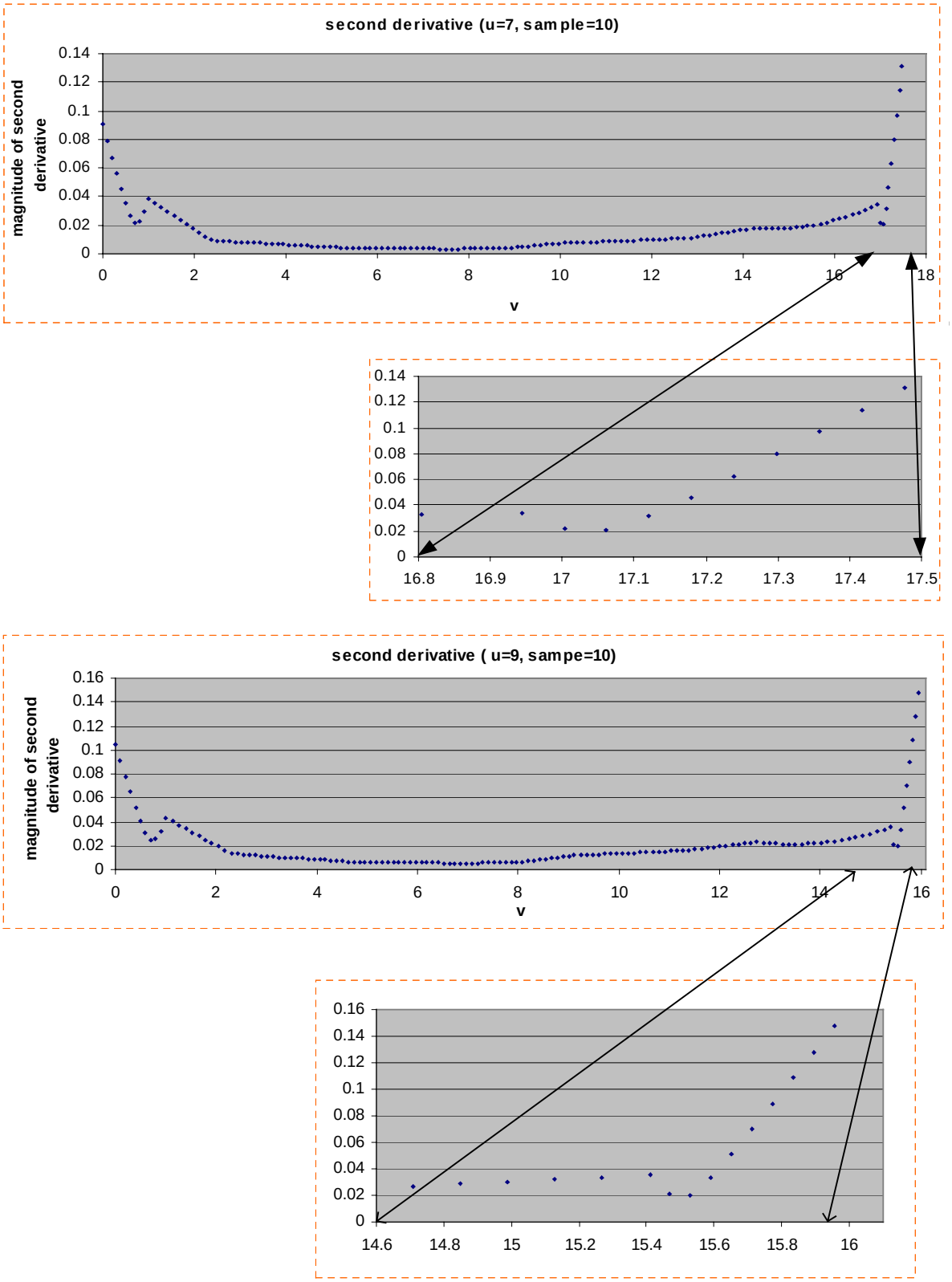


Figure 59 Second derivative plot for *v* iso-parametric curves

From the figure above, it can be concluded that the ***v*** iso-parametric curves are second derivative continuous.

Curvature and second derivative analysis for ***u*** iso-parametric curves:

The next section presents curvature and second derivative analysis of ***u*** direction curves those include the fillet.

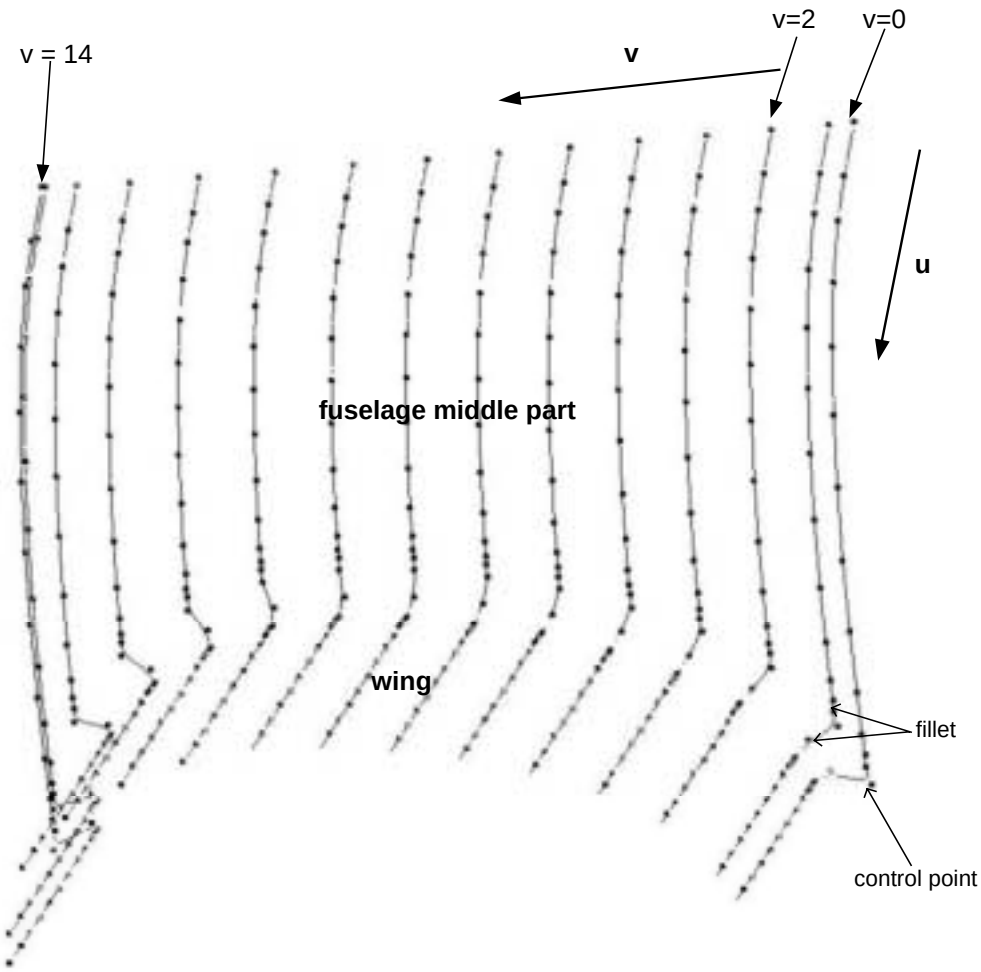


Figure 60 *u* iso-parametric curves on the blending surface

Curvature change versus ***u*** parameter value:

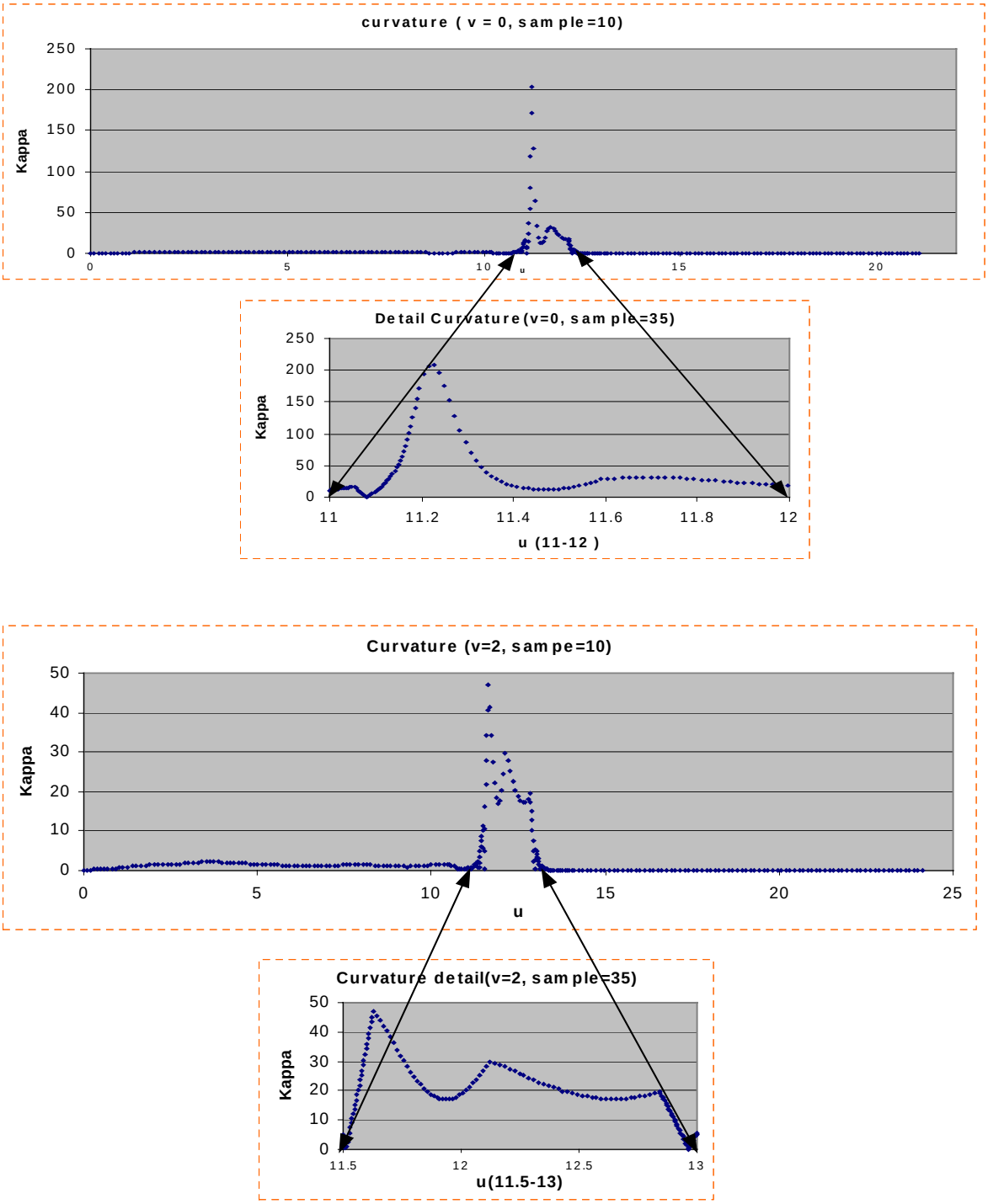


Figure 61 Curvature plot for *u* iso-parametric curves

Second derivative against ***u*** parametric value:

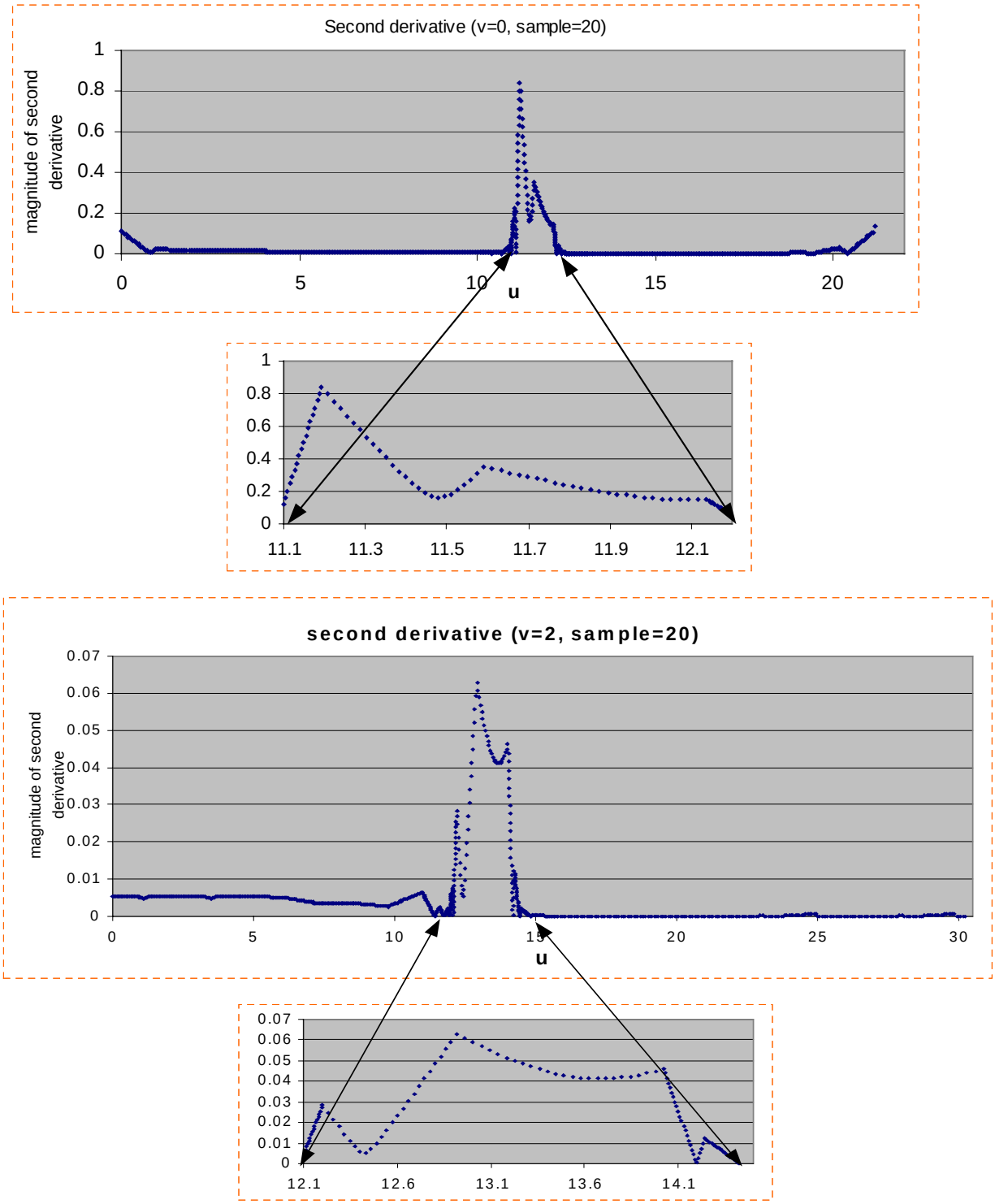


Figure 62 Second derivative plot for *u* iso-parametric curves

From the figures above, it can be concluded that ***u*** iso-parametric curves are curvature and second derivative continuous.

7.4 Conclusions and Recommendations

The research provides a robust procedure to make surface filleting for typical aircraft fuselage and wing shapes. It satisfies the requirements of preliminary design systems for aircraft design. The objective of this thesis is summarized as:

- Geometric trimming for the case of a closed trimming curve. A conversion process was developed to change closed trimming curves to open trimming curves in parametric space.
- Surface filleting is implemented by blending two geometrically trimmed surfaces into one C^2 continuous surface. It involves refining the control net of the trimmed surface to ensure the match of the new surface and the original, adding intermediate points to control the shape of fillet.
- Develop a platform to display the results visually.
- Analyze the continuity and curvature properties of the blending surface.

From the results discussed in the earlier sections, it can be concluded that the procedure of trimming and filleting can accomplish a complete mathematical description of a blending surface which blends two B-spline surfaces in the general shape of an aircraft fuselage and wing. This research provides an approach to manipulate the surface operations for a typical aircraft fuselage and wing. The approach includes surface intersecting, geometric surface trimming and surface filleting. The data for surface intersection comes from a set up file. Rojas' geometric trimming method is employed to make the surfaces trimmed. This method utilizes Fleming's constraint-based interpolating algorithm to generate a new set of B-spline surface patches approximating the untrimmed portion of the original surface. It has been proved very accurate when using an open trimming curve. In the case of closed trimming curve, a conversion procedure is developed to change the closed trimming curve into open trimming curve for geometric trimming as addressed in Chapter 5.

Some future work can be implemented to improve the trimming and filleting algorithm:

- In the phase of geometric trimming, accurate results can be achieved by selecting more data to describe the model surface and the trimming curve.
- The shape of the fillet is determined by the number of intermediate points which come from the conic definition. More intermediate points can force the shape of the fillet closer to the conic's.
- The whole blending surface consists of four patches, which are simply connected to each other by the common points. On the joining boundaries, the surface is not guaranteed to be C^2 continuous. The problem can be solved by surface subdivision to make the surfaces separated but keep them C^2 continuous on the boundaries. Fixed control vertices inversion [Glou89], combined with knot insertion, provides a method to subdivide one surface into two surfaces, which keeps the same position, slope and curvature at the dividing boundaries as the original surface. In this research, as shown in Figure 40, the trimmed fuselage can be subdivided by the algorithm described above.

8.0 References

[Appl89] Applegarth, I., Catley, D., Bradley, I., "Clipping of B-spline patches at surface Curves," *The Mathematics of Surfaces III*, Clarendon Press, Oxford, 1989, pp. 229-242.

[Bind00] Bindiganavle, K., "An Optimal Approach to Geometric Trimming of B-Spline Surfaces," *Master's thesis,* VPI&SU, March 2000.

[Bohm84] Bohm, W., Farin, G., Kahmann, J., "A Survey of Curve and Surface Methods in CAGD," *Computer-Aided Design,* vol. 1, no.1, July 1984, pp. 1-60.

[Casa87] Casale, M.S., "Free-form Solid Modeling with Trimmed Surface Patches," *IEEE Computer Graphics & Applications,* vol. 7, January 1987, pp. 33-43.

[Casa89a] Casale, M. S., Bobrow, J. E., "Solid Modeling Using Parametric Surfaces Defined Over Non-Square Domains," *Presented at the Eighth International Conference on Off-shore Mechanics and Arctic Engineering,* March 1989, pp. 231-236.

[Cohe79] Cohen, E., Lyche, T., Reisenfeld, R., "Discrete B-spline and Subdivision Techniques in Computer Aided Geometric Design and Computer Graphics," *Computer Graphics and Image Processing*, vol. 14, 1987, PP.87-111.

[Casa89b] Casale, M. S., Bobrow, J. E., "The Analysis of Solids Without Mesh Generation Using Trimmed Patch Boundary Elements," *Engineering with Computers,* vol. 5, 1989, pp. 249-257.

[DeBo72] De Boor, C., "On calculating with B-Splines," *Journal of Approximation Theory*, vol. 6, 1972, pp.50-62.

[Dokk85] Dokken, T., "Finding intersections of B-spline represented geometries using recursive subdivision techniques," *Computer Aided Geometric Design*, vol. 2, 1985, pp. 189-195.

[Fari97] Farin, G., Curves and Surfaces for Computer Aided Geometric Design, *Fourth edition*, Academic Press, San Diego, California, 1997.

[Flem92a] Fleming, S., "The Enhancement of PHIGS Plus B-spline Functionality for Geometric Modeling in CAD," *Master's thesis,* VPI&SU, April 1992.

[Flem92b] Fleming, S., Myklebust., A., "The Enhancement of PHIGS Plus B-spline Functionality for Geometric Modeling in CAD," *Fourth IFIP WG5.2 on Geometric Modeling in Computer Aided Design,* Rensselaerville, New York, September 27 – October 1, 1992.

[Glou89] Gloudemans, J. R., "Filleting of Aircraft Components Using Non-Uniform B-spline Surfaces," *Master's thesis,* VPI&SU, May 1989.

[Hosc90] Hoscheck, J., Schneider, F., "Spline Conversion for Trimmed Rational Bezier and B-spline surfaces," *Computer-Aided Design,* vol. 22, no. 9, November 1990, pp. 580-590.

[Hosc89] Hoscheck, J., Schneider, F., Wassum, P., "Optimal Approximate Conversion of Spline Surfaces," C*omputer-Aided Geometric Design,* vol. 6, 1989, pp. 293-306.

[Hosc88] Hoscheck, J., "Intrinsic Parameterization for Approximation," *Computer-Aided Geometric Design,* vol. 5, 1988, pp. 27-31.

[Hosc87] Hoscheck, J., "Approximate Conversion of Spline Curves," *Computer-Aided Geometric Design,* vol. 4, 1987, pp. 59-66.

[Jain99] Jain, A., “Error Visualization in Comparison of B-spline Surfaces,” *Master’s thesis,* VPI&SU, April 1999.

[Jone91] Jones, R. W., “Intersection and Filleting of Non-Uniform B-spline Surfaces,” *Master’s thesis,* VPI&SU, January 1991.

[Kilg96] Kilgard, M.J., OpenGL, Programming for X Window System. *Addison-Wesley Developers Press, 1997.*

[Lane80] Lane, J., Riesenfeld, F., “A Theoretical Development for the Computer Generation and Display of Piecewise Polynomial Surfaces,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, 1980, pp. 35-46.

[Lass86] Lasser, D., “Intersection of parametric surfaces in the Bernstein-Bezier representation,” *Computer Aided Design*, vol. 18, no. 4, 1986, pp.186-192.

[Mort85] Mortenson, M. E., Geometric Modeling, *John Wiley & Sons, 1985.*

[Peng84] Peng, Q. S., “An algorithm for finding the intersection lines between two B-spline surfaces,” *Computer Aided Design*, vol. 16, no. 4, 1984, pp.191-196.

[Roja94] Rojas, R., "Geometric Trimming of B-spline Surfaces," *Master’s thesis, VPI&SU, July 1994.*

[Shan88] Shantz, M., Chang, S., “Rendering Trimmed NURBS with Adaptive Forward Differencing,” *Computer Graphics,* vol. 22, no. 4, August 1988.

[Tjun93] Tjung, Jie Wen, “Projection, Design and Representation of Curves on B-Spline Surfaces,” *M.S. Thesis,* VPI&SU, January 1993.

[Warr89] Warren, J., “Blending Algebraic Surfaces,” *ACM Transactions on Graphics*, vol. 8, no. 4, 1989, pp.263-278.

[Wong90] Wong, C.K., “Intersection of B-spline Surfaces by Elimination method,” *M.S. Thesis, VPI&SU, September 1990.*

[Woo97] Woo, M., Neider, J., Davis, T., OpenGL Programming Guide, *Second Edition*, Addison-Wesley Developers Press, 1997.

[Yama88] Yamaguchi, F., Curves And Surfaces in Computer Aided Geometric Design, Springer-Verlag, 1988.

Vita

Xijun Wang

The author was born on March 15, 1969, in China. In 1988, he began his undergraduate studies in Mechanical Engineering at Shanghai Jiao Tong University. After graduation, the author worked for Tsingtao Brewery Company for six years, then went to America to continue his graduate studies in Mechanical Engineering in the fall of 1998. After two terms in the University of Arkansas, he transferred to Virginia Tech, and finished his Master's degree in the spring of 2001.

Xijun plans to go to Richmond to join his family and start a career there.