FREE FORM SURFACE ANALYSIS USING A HYBRID OF SYMBOLIC AND NUMERIC COMPUTATION

by

Gershon Elber

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

The University of Utah

December 1992

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a dissertation submitted by

Gershon Elber

This dissertation has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

Chair: Elaine Cohen

Rich Riesenfeld

Sam Drake

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of The University of Utah:

I have read the dissertation of Gershon Elber in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to the Graduate School.

Date

Elaine Cohen
Chair, Supervisory Committee

Approved for the Major Department

Tom Henderson
Chair/Dean

Approved for the Graduate Council

B. Gale Dick
Dean of The Graduate School

ABSTRACT

Detailed analysis of many mathematical properties of sculptured models has been hindered by the fact that the properties do not have the same representation as the surface. For example, unit tangents, surface normals, and principal curvatures are typically computed at predefined discrete sets of points on the surface. As such, aliasing can occur and features between samples can be missed. Synthesizing information about the shape of an object and operating on the model, whether by physical machining tools, graphics display programs, or mathematical analysis, has been treated as either a discrete or local problem in general. The research being reported on here has focused on another approach, that of creating algorithms that construct the mathematical properties in closed form, or construct approximations to those mathematical properties through symbolic computation. Global analysis can then be applied while an accurate error bound is obtained.

Basic tools required for such symbolic computation are presented and their usage in a broad range of applications from offset approximations through curvature analysis to generation of machining toolpaths are demonstrated. The combination is not only shown to be powerful but it also provides a novel approach to problem solving in an elegant and robust way.

This thesis is dedicated to my wife

Irit

and my two daughters

Lotem and Sivan.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank all the people who explicitly or implicitly made the completion of this work a possibility.

Elaine Cohen, with her unwillingness to accept the unclear, pushed me into better and well-defined algorithms. Her mathematicals skill, combined with the practical view of the problems the CAGD field is facing, were a tremendous help for this work.

Rich Riesenfeld and Elaine Cohen provided generous funding that allowed me to devote my time to interesting research. IBM generously supported me for two years on a fellowship.

Sam Drake, the third member of my committee, helped in testing all the NC machining related algorithms.

Mike Milochik provided the picture developing services and made all the color enlargements of this document. Hank Driskill volunteered his scanner and scanned some of the noncomputer generated images back into electronic form.

It was impossible to perform this research without the excellent environment that the Alpha_1 group has created. I could not have hoped for a better environment in which to work. Special thanks go to Beth Cobb and Robert Mecklenburg for their willingness to try and help with problems that arose.

Finally, this work would never have started, had not my wife Irit been willing to make this huge sacrifice. Her support and control on our growing family allowed me to devote more time to this research than I could ever hoped. My gratitude extends to my parents, who encouraged me to apply and study at the best school I could find, not necessarily the closest.

CHAPTER 1

INTRODUCTION

"Where shall I begin, please your Majesty?" he asked. "Begin at the beginning," the King said, gravely, "and go on till you come to the end: then stop."

Alice's Adventures in Wonderland, Lewis Carroll

The field of CAGD has evolved significantly in the last decade. The B-spline representation, introduced to the field of CAGD during the 70s, has become a dominant representation in mechanical design. Techniques to manipulate, evaluate, mill, render, and analyze freeform surface based models have undergone extensive research. It is common for surfaces/curves to be approximated by a set of polygons/polylines for milling, rendering, and analysis purposes. Other properties are computed at discrete locations and interpolation is used to provide the information over the entire domain.

This thesis applies symbolic computation to some of these problems. The power of symbolic computation for freeform curves and surfaces will be demonstrated throughout this document. Symbolic computation provides the ability to compute properties with exact precision. It virtually eliminates the fundamental problems that rise from discrete sampling. If the domain being sampled contains information in higher frequencies than that of the samples, the original data cannot be reconstructed precisely (Nyquist theorem). Unfortunately, many important problems have that characteristic.

The use of symbolic computation opens the door for solving problems using a global approach. It may be useful to demonstrate the differences between global and local methods using an example from computer graphics. Two common methods are used to render realistic scenes: ray tracing and radiosity. The first "fires" a single ray at a time and samples the world along that particular line. This is clearly a local approach. The second looks over the problem globally and finds all the energy distribution simultaneously in the equilibrium state. Given a scene, this second technique is ideally global because it takes into account *all* data. We say "ideally" because several approximations are commonly performed in this method to obtain faster results. It is not surprising then, that ray tracing methods face severe aliasing problems. Major research efforts in the computer graphics community are devoted to overcoming the ray sampling problems. The ideal radiosity method does not introduce aliasing problems, although the scene subdivision and polygonization stage does, simply because most of the subdivision techniques are local.

Another way to distinguish global techniques from local ones is that global techniques can arrive at all their results at the same time while local methods provide the viewer with sequential information. A global method may be used to analyze a whole surface at once. The ray tracing technique from the above example processes one sampled ray at a time, whereas the radiosity method computes and returns the light distribution information for all elements in the scene simultaneously.

Greedy algorithms are local, as shown by the coin based example in [1], pp. 321 emphasizing their global inefficiency. Taylor approximations are another example of exploiting local information. The Newton Raphson curve root finding is clearly a local technique in that it converges to roots in a neighborhood of the initial guess. Recall that it does not ensure finding all the roots. On the other hand, if an algorithm uses properties of the *entire* surface and makes decisions based on both local and global information, it is referred to as a *global* algorithm. An ideal method should find all the solutions quickly and so must be global.

We use derived surfaces, called *property surfaces*, whose definitions are derived from different attributes of the original surface as auxiliary surfaces to help analyze the original surface. For example, $\frac{\partial F}{\partial u}(u, v)$ is a property surface of F, and so is $n(u, v)$, the surface of unit normals. The two surfaces of principal curvatures, $\kappa_n^1(u, v)$ and $\kappa_n^2(u, v)$ are also property surfaces.

Definition 1.1 *Suppose $\mathcal{S}_1$ and $\mathcal{S}_2$ are vector spaces of surfaces. An operator $\mathcal{P} : F \rightarrow p \in \mathcal{S}_2$, for all $F \in \mathcal{S}_1$, is called a* property operator *if the image surface, p, is associated with a property of F, the domain surface. In that case, p is called a* property surface.

Some property surfaces are of the same "type" as the original surface whereas others are not. If F is a tensor product NURBs surface, then $\frac{\partial F}{\partial u}(u, v)$ is a property surface which is also a tensor product NURBs surface with the same knot vectors, but with different (lower) order and continuity properties. $\frac{\partial F}{\partial u}(u, v) \times \frac{\partial F}{\partial v}(u, v)$ is also a property surface, that is, a tensor product NURBs surface, but with different knot vectors, different (higher) order, and different (lower) continuity. These two property surfaces share the trait with F that they are NURBs surfaces, but $n(u, v)$, $\kappa_n^1(u, v)$ and $\kappa_n^2(u, v)$ are not NURBs surfaces, in general. They cannot be represented as a piecewise parametric rational functions, as we shall later see, and hence, cannot be represented as NURBs surfaces.

We restrict ourselves (definition 1.1) to using property surfaces that are either representable as NURBs or to property surfaces for which we can derive approximations that are representable as NURBs. This restriction enables us to apply any algorithmic approach developed for the NURBs representation to the property surfaces. Not every property is representable as a NURBs surface. A unit normal surface has a square root in the denominator of its normalization which is not representable as a NURBs.

Contouring techniques [4, 45, 59] developed for freeform surfaces can be applied immediately to a NURBs representation of a property surface. Because both the original and the property surfaces share the same parametric domain, one can easily create a trimmed surface consisting of those regions in the original surface having the desired property values. In other cases the zero set of certain property surfaces may be required. For example, let $\hat{n}_z(u,v)$ be the z component of $\hat{n}(u,v)$, where $\hat{n}(u,v) = \frac{\partial F(u,v)}{\partial u} \times \frac{\partial F(u,v)}{\partial v}$ is orthogonal to the parametric surface $F(u,v)$ at (u,v). Then the set of zeros of $\hat{n}_z(u,v)$ is simply the parameter values along the silhouettes of the original surface when F is being viewed from $(0,0,\infty)$. Hence, the silhouette extraction problem is equivalent to a root finding problem (contouring), which is usually simpler. Trimmed surfaces [9, 47] are the natural way to represent the regions defined by the contouring operator. In fact, the parameter values of the contours of certain property surfaces can serve as the parameter values of a trimming curve for the original surface.

In this thesis, we apply global techniques based on symbolic computation to a broad range of problems. In Chapter 2, we develop the tools, some of which are described above, that will be used throughout this dissertation. It may be a surprise how small the number of required tools is.

As is discussed in Chapter 3, offsets of freeform piecewise polynomial/rational curves and surfaces are not, in general, representable in the same domain. Approximation techniques are used instead. However, we use a symbolic method to compute the error function of these approximations. We use this error function in two ways. First its extrema serve as a bound on the error. In addition, isolation of the regions of the error function with large error allow us to automatically improve the approximation until a prescribed tolerance is achieved.

In Chapter 4, surface curvature analysis is performed globally using symbolic computation of curvature property surfaces. Curvature analysis has applications in modeling as well as in manufacturing. Symbolic computation provides the ability to trichotomize a surface into three regions, convex, concave, and saddlelike regions.

This trichotomy, which can dramatically improve milling algorithms efficiency, is almost impossible without a global approach.

In Chapter 5, the questions of toolpath generation for NC machining is dealt with. Optimal toolpaths for freeform surfaces is known as a difficult problem and current approaches fails in extreme cases. We will present a symbolic algorithm that generates a toolpath for machining freeform surfaces that performs much better than current local schemes. This algorithm is then enhanced so it can automatically generate machining toolpaths for real models consisting of several trimmed surfaces while avoiding any gouging. Because the algorithm provides a bound on the redundancy in the generated toolpath, it was successfully modified and used as a rendering tool. The toolpath curves are rendered using curve rendering techniques to form the image.

Finally in Chapter 6, several other applications are addressed. Approximation of higher order curves using lower order ones is the first. We also add a composition operator to the set of operators we defined in Chapter 2 and discuss its potential. Other surface properties such as speed, and slope are defined and considered, and twist is considered as part of a global symbolic approach.

CHAPTER 2

SYMBOLIC AND NUMERIC COMPUTATION

Equations are more important to me, because politics is for the present, but an equation is something for eternity.

Albert Einstein

This Chapter develops the symbolic representational and numeric computational tools required to carry out the analysis performed throughout this document. The derivations of the tools for the Bézier representation are presented while the appropriate references to derivations for the equivalent tools for the NURBs representation are made.

2.1 Symbolic Representation

The following basic symbolic representations will be required for Bézier and NURBs curves and surfaces:

- derivative representation.
- sum/difference representation.
- product representation.

This quite minimal set of representations is extremely powerful tool as is demonstrated by the following Chapters. Because a division by a scalar entity can always be defined as a rational expression, we never have to compute such an operation explicitly: $x/y \equiv \frac{x}{y}$. It should be carefully noted that these representations represent the result of symbolic operators. That is, they represent the result by means of symbols instead of evaluating it numerically at a single point. More practically, the result is represented in the same Bézier or NURBs domain as a curve or a surface, so a closure is formed. This closure enables arbitrary composition of these operators.

2.1.1 Derivatives Representation

Representing the derivatives of Bézier and NURBs curves and surface is straightforward [27]. Differentiation of a single Bézier basis function may be expressed as a linear combination of two lower order Bézier basis functions:

$$\begin{aligned}
\mathbf{B}_i^{n(1)}(t) &= \binom{i}{n} i t^{i-1}(1-t)^{n-i} - \binom{i}{n}(n-i)t^i(1-t)^{n-i-1} \\
&= n\left(\frac{(n-1)!}{(i-1)!(n-i)!}t^{i-1}(1-t)^{n-i} - \frac{(n-1)!}{i!(n-i-1)!}t^i(1-t)^{n-i-1}\right) \\
&= n\left(\binom{i-1}{n-1}t^{i-1}(1-t)^{n-i} - \binom{i}{n-1}t^i(1-t)^{n-i-1}\right) \\
&= n(\mathbf{B}_{i-1}^{n-1}(t) - \mathbf{B}_i^{n-1}(t)). \qquad (2.1)
\end{aligned}$$

For curves it immediately follows that:

$$\begin{aligned}
\frac{dC(t)}{dt} &= \sum_{i=0}^{n} P_i \mathbf{B}_i^{n(1)}(t) \\
&= \sum_{i=0}^{n} n P_i (\mathbf{B}_{i-1}^{n-1} - \mathbf{B}_i^{n-1}) \\
&= n(\sum_{i=0}^{n} P_i \mathbf{B}_{i-1}^{n-1} - \sum_{i=0}^{n} P_i \mathbf{B}_i^{n-1}) \\
&= n(\sum_{i=1}^{n} P_i \mathbf{B}_{i-1}^{n-1} - \sum_{i=0}^{n-1} P_i \mathbf{B}_i^{n-1}) \\
&= n\sum_{i=0}^{n-1}(P_{i+1} - P_i)\mathbf{B}_i^{n-1}. \qquad (2.2)
\end{aligned}$$

Extension to tensor product surfaces is straightforward because there is no dependency between the two surface parameters (m is the degree).

$$\begin{aligned}\frac{\partial S(u,v)}{\partial u} &= \sum_{i=0}^{m}\sum_{j=0}^{n} P_{ij}\mathbf{B}_i^m(u)^{(1)}\mathbf{B}_j^n(v) \\ &= m\sum_{i=0}^{m-1}\sum_{j=0}^{n}(P_{(i+1)j}-P_{ij})\mathbf{B}_i^{m-1}(u)\mathbf{B}_j^n(v), \end{aligned} \tag{2.3}$$

and similarly for $\frac{\partial S(u,v)}{\partial v}$.

Differentiation in the NURBs domain follows the same procedure [27] (k is the degree):

$$\begin{aligned}\frac{dC(t)}{dt} &= \sum_{i=0}^{n-1} P_i\mathbf{B}_{i,\tau}^k(t)^{(1)}(t) \\ &= k\sum_{i=0}^{n-2}\frac{(P_{i+1}-P_i)}{t_{i+k}-t_i}\mathbf{B}_{i,\tau}^{k-1}(t), \end{aligned} \tag{2.4}$$

and for surfaces:

$$\begin{aligned}\frac{\partial S(u,v)}{\partial u} &= \sum_{i=0}^{m}\sum_{j=0}^{n} P_{ij}\mathbf{B}_{i,\tau}^k(u)^{(1)}\mathbf{B}_{j,\xi}^l(v) \\ &= k\sum_{i=0}^{m-1}\sum_{j=0}^{n}\frac{P_{(i+1)j}-P_{ij}}{t_{i+k}^u-t_i^u}\mathbf{B}_{i,\tau}^{k-1}(u)\mathbf{B}_{j,\xi}^l(v). \end{aligned} \tag{2.5}$$

2.1.2 Representation of Sum/Difference

Finding a representation for the sum or difference of two Bézier or NURBs curves or surfaces can be achieved by bringing them to a common representation. If the two curves do not share the same parametric domain, their knot vectors can always be affinely transformed without modifying the curves, so their parametric domains will match. If the two curves or surfaces are not of the same polynomial order, the lower one should be degree raised [16, 17] to the higher order. If internal knots (of a B-spline curve) have different multiplicities in the two curves, at each knot, the curve with the lower multiplicity should be refined [7, 15] to match the higher multiplicity. Once both curves are transformed to have a common order and knot vector, their control polygons can simply be summed or differenced because:

$$\begin{aligned} C_1(t) \pm C_2(t) &= \sum_{i=0}^{m} P_i \mathbf{B}_{i,\tau}^{m}(t) \pm \sum_{i=0}^{m} Q_i \mathbf{B}_{i,\tau}^{m}(t) \\ &= \sum_{i=0}^{m} (P_i \pm Q_i)\, \mathbf{B}_{i,\tau}^{k}(u) \end{aligned} \tag{2.6}$$

This condition also holds for surfaces. Once the two surfaces share the same orders and knot vectors, their meshes can be simply subtracted or added:

$$\begin{aligned} S_1(u,v) \pm S_2(u,v) &= \sum_{i=0}^{k}\sum_{j=0}^{l} P_{ij} \mathbf{B}_{i,\tau^u}^{m}(u) \mathbf{B}_{j,\tau^v}^{n}(v) \pm \sum_{i=0}^{k}\sum_{j=0}^{l} Q_{kl} \mathbf{B}_{i,\tau^u}^{m}(u) \mathbf{B}_{j,\tau^v}^{n}(v) \\ &= \sum_{i=0}^{k}\sum_{j=0}^{l} (P_{ij} \pm Q_{kl}) \mathbf{B}_{i,\tau^u}^{m}(u) \mathbf{B}_{j,\tau^v}^{n}(v) \end{aligned} \tag{2.7}$$

Bringing two Bézier curves to a common domain only requires elevating the degree of the lower one.

$$\begin{aligned} (1-t)\mathbf{B}_i^n(t) &= \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i+1} \\ &= \frac{n-i+1}{n+1} \binom{i}{n+1} t^i (1-t)^{n-i+1} \\ &= \frac{n-i+1}{n+1} \mathbf{B}_i^{n+1}(t) \end{aligned}$$

and

$$\begin{aligned} t\mathbf{B}_i^n(t) &= \frac{n!}{i!(n-i)!} t^{i+1} (1-t)^{n-i} \\ &= \frac{i+1}{n+1} \binom{i+1}{n+1} t^{i+1} (1-t)^{n-i} \\ &= \frac{i+1}{n+1} \mathbf{B}_{i+1}^{n+1}(t). \end{aligned}$$

Therefore a Bézier basis function of degree n may be represented as a convex combination of two Bézier basis functions of degree $n+1$:

$$\mathbf{B}_i^n(t) = (1-t)\mathbf{B}_i^n(t) + t\mathbf{B}_i^n(t)$$

$$= \frac{n-i+1}{n+1}\mathbf{B}_i^{n+1}(t) + \frac{i+1}{n+1}\mathbf{B}_{i+1}^{n+1}(t). \qquad (2.8)$$

Given a Bézier curve of degree n, raising it to degree $n+1$ involves:

$$\begin{aligned} & C(t) \\ = & \sum_{i=0}^{n} P_i \mathbf{B}_i^n(t) \\ = & \sum_{i=0}^{n} P_i \left(\frac{n+1-i}{n+1}\mathbf{B}_i^{n+1}(t) + \frac{i+1}{n+1}\mathbf{B}_{i+1}^{n+1}(t) \right) \\ = & P_0 \mathbf{B}_0^{n+1}(t) + \sum_{i=1}^{n} \left(\frac{n+1-i}{n+1} P_i + \frac{i}{n+1} P_{i-1} \right) \mathbf{B}_i^{n+1}(t) + P_n \mathbf{B}_{n+1}^{n+1}(t). \end{aligned} \qquad (2.9)$$

Once again, similar procedures can be followed for Bézier surfaces.

It is desired to be able to subtract or add a constant to a freeform surface or curve. Because $\sum_{i=0}^{m} \mathbf{B}_i^m \equiv 1$, a K constant curve may be represented as the following Bézier curve:

$$K = K \sum_{i=0}^{m} \mathbf{B}_i^m(t) = \sum_{i=0}^{m} K \mathbf{B}_i^m(t) \qquad (2.10)$$

and hence, using equations (2.6) and (2.10) constant subtraction or addition is equivalent to subtracting or adding this constant from all curve coefficients. An equivalent formulation holds for surfaces, and the NURBs representation.

A simple formulation can be defined for adding and subtracting rational curves $C_i(t) = \left(\frac{c_i^x(t)}{w_i(t)}, \frac{c_i^y(t)}{w_i(t)}, \frac{c_i^z(t)}{w_i(t)} \right) = \frac{(c_i^x(t), c_i^y(t), c_i^z(t))}{w_i(t)}$:

$$\begin{aligned} C_1(t) \pm C_2(t) &= \frac{(c_1^x(t), c_1^y(t), c_1^z(t))}{w_1(t)} \pm \frac{(c_2^x(t), c_2^y(t), c_2^z(t))}{w_2(t)} \\ &= \frac{(c_1^x(t), c_1^y(t), c_1^z(t)) w_2(t) \pm (c_2^x(t), c_2^y(t), c_2^z(t)) w_1(t)}{w_1(t) w_2(t)}. \end{aligned} \qquad (2.11)$$

Addition of rational curves requires the capability to find products. Rational surface addition and/or subtraction may be represented in a similar way. See [27, 30] for additional details.

2.1.3 Product Representation

Although finding the symbolic derivative, sum and difference of Bézier or NURBs curves and surfaces is straightforward, finding products of curves and surfaces is more difficult. We start by considering the product of two Bézier curves and then derive the analogous formulation for surfaces.

Given two Bézier basis functions $\mathbf{B}_i^m$ and $\mathbf{B}_j^n$, their product [30, 62] is equal to:

$$\begin{aligned} \mathbf{B}_i^m(t)\mathbf{B}_j^n(t) &= \binom{m}{i} t^i (1-t)^{m-i} \binom{n}{j} t^j (1-t)^{n-j} \\ &= \binom{m}{i}\binom{n}{j} t^{i+j}(1-t)^{m+n-i-j}. \end{aligned}$$

On the other hand:

$$\mathbf{B}_{i+j}^{m+n} = \binom{m+n}{i+j} t^{i+j}(1-t)^{m+n-i-j},$$

therefore:

$$\mathbf{B}_i^m(t)\mathbf{B}_j^n(t) = \frac{\binom{m}{i}\binom{n}{j}}{\binom{m+n}{i+j}} \mathbf{B}_{i+j}^{m+n}(t). \tag{2.12}$$

Using equation (2.12), one can easily derive product formulas for curves and surfaces:

$$\begin{aligned} C_1(t)C_2(t) &= \sum_{i=0}^{m} P_i \mathbf{B}_i^m(t) \sum_{j=0}^{n} Q_j \mathbf{B}_j^n(t) \\ &= \sum_{i=0}^{m}\sum_{j=0}^{n} P_i Q_j \mathbf{B}_i^m(t)\mathbf{B}_j^n(t) \\ &= \sum_{i=0}^{m}\sum_{j=0}^{n} P_i Q_j \frac{\binom{m}{i}\binom{n}{j}}{\binom{m+n}{i+j}} \mathbf{B}_{i+j}^{m+n}(t) \\ &= \sum_{k=0}^{m+n} R_k \mathbf{B}_k^{m+n}(t) \end{aligned} \tag{2.13}$$

where:

$$R_k = \sum_{i=\max(0,k-n)}^{\min(k,m)} P_i Q_{k-i} \frac{\binom{m}{i}\binom{n}{k-i}}{\binom{m+n}{k}}.$$

The formulation for product surface follows much the same derivation:

$$S_1(u,v)S_2(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} P_{ij}\mathbf{B}_i^m(u)\mathbf{B}_j^n(v) \sum_{k=0}^{p}\sum_{l=0}^{q} Q_{kl}\mathbf{B}_k^p(u)\mathbf{B}_l^q(v)$$

$$\begin{aligned}
&= \sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{k=0}^{p}\sum_{l=0}^{q} P_{ij}Q_{kl}\mathbf{B}_i^m(u)\mathbf{B}_k^p(u)\mathbf{B}_j^n(v)\mathbf{B}_l^q(v) \\
&= \sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{k=0}^{p}\sum_{l=0}^{q} P_{ij}Q_{kl}\frac{\binom{m}{i}\binom{p}{k}}{\binom{m+p}{i+k}}\mathbf{B}_{i+k}^{m+p}(u)\frac{\binom{n}{j}\binom{q}{l}}{\binom{n+q}{j+l}}\mathbf{B}_{j+l}^{n+q}(v) \\
&= \sum_{r=0}^{m+p}\sum_{s=0}^{n+q} R_{rs}\mathbf{B}_{i+k}^{m+p}(u)\mathbf{B}_{j+l}^{n+q}(v) \qquad (2.14)
\end{aligned}$$

where:

$$R_{rs} = \sum_{i=\max(0,r-p)}^{\min(r,m)} \sum_{j=\max(0,s-q)}^{\min(s,n)} P_{i,j}Q_{r-i,s-j}\frac{\binom{m}{i}\binom{p}{r-i}}{\binom{m+p}{r}}\frac{\binom{n}{j}\binom{q}{s-j}}{\binom{n+q}{s}}.$$

Finding the products of polynomial B-spline and NURBs is far more difficult. A direct approach has recently been developed in [49] which supports symbolic computation of the coefficients of the product after finding the knot vector. However, because it is computationally expensive and complex to implement, one might choose to exploit the uniqueness property of the process and compute the coefficients of the product by solving an interpolation problem. First, one would form the knot vector of the product, which can be derived from the knot vectors and orders of the factors. The order of the product curve, $C(t) = C^1(t)C^2(t)$, is equal to $O = O^1 + O^2 - 1$, where O^1 and O^2 are the orders of $C^1(t)$ and $C^2(t)$, respectively. The knot values and the continuity of the product curve at its knots are determined by the factor curve with the lower degree of continuity at that knot. Let ν^i be a vector holding all distinct values in τ^i, the knot vector of C^i, arranged in ascending order. At each knot value λ_j of ν^i, let $\mu^i(\lambda_j)$ be multiplicity of λ_j in C^i. Then the continuity of C^i at this knot is equal to $\mathcal{C}_j^i = O^i - \mu^i(\lambda_j) - 1$. τ^i can be decomposed into two vectors: ν^i holding all distinct values and $\mathcal{C}^i$ holding their continuities, i.e., $\tau^1 \Leftrightarrow \{\nu^1, \mathcal{C}^1\}$ and similarly $\tau^2 \Leftrightarrow \{\nu^2, \mathcal{C}^2\}$. Let ν be the merged ordered set of the distinct values from both ν^1 and ν^2, and let $\mathcal{C}$ be defined so that the mth knot $\mathcal{C}_m = min(\mathcal{C}_j^1, \mathcal{C}_k^2)$ if $\nu_j^1 = \nu_k^2$. The resulting knot vector τ will contain all the distinct values in ν with multiplicity μ equal to $\mu(\lambda_m) = O - \mathcal{C}_m - 1, \forall \lambda_m \in \nu$. The resulting knot vector, τ, is minimal in the sense that any curve $C(t)$ representing the product $C^1(t)C^2(t)$ will have a knot vector which contains τ. One can find the *unique* B-spline curve defined over τ that interpolates $C^1(\xi_i)C^2(\xi_i)$ for all ξ_i, the node values of τ. From uniqueness, such a curve interpolates $C^1(t)C^2(t)$. This process transforms the problem into an interpolation problem, producing a set of linear equations that must be solved for the control polygon points of $C(t)$. The matrix formed is banded.

The resulting curve is unique in the sense that it minimizes the loss of continuity. Each interval between two adjacent distinct knots of τ may be represented as a polynomial, and can be represented as a Bézier segment. However, such an approach guarantees only C^0 continuity at the knots.

2.2 Numeric Computation

Knowing the zero set of a property surface and/or knowing all regions in which the property surface values are larger than some threshold or even equal to some specified value is frequently useful in extracting shape information from given curve(s) and surface(s). The ability to slice the given curve/surface with a plane is called contouring and is equivalent to finding the intersection of a curve and a line or a surface and a plane.

Contouring is used extensively to extract data from property surfaces (see Chapter 1). Once the contours are computed, their domain values in the parametric space can serve as bounding trimming curves for the original surface, $S(u, v)$, so that the trimmed surface will hold all regions in $S(u, v)$ known to have property values larger (or smaller) than the contouring level, or will contain regions bounded between two property values.

2.2.1 Contouring in E^3

This contouring process is closely related to the surface-surface intersection and ray-surface intersection [41] problems, with their inherent numerical complexities and instabilities.

Let $F(u,v) = (\frac{x(u,v)}{w(u,v)}, \frac{y(u,v)}{w(u,v)}, \frac{z(u,v)}{w(u,v)})$ and $P = Ax + By + Cz + D = 0$ be a property surface and a contouring plane, respectively. By substituting the coordinate functions of $F(u,v)$ into P one can solve for all the values of u and v in the domain for which $F(u,v) \cap P \neq \emptyset$.

$$\begin{aligned} S(u,v) &= A\frac{x(u,v)}{w(u,v)} + B\frac{y(u,v)}{w(u,v)} + C\frac{z(u,v)}{w(u,v)} + D \\ &= \frac{Ax(u,v) + By(u,v) + Cz(u,v) + Dw(u,v)}{w(u,v)}. \end{aligned} \tag{2.15}$$

A single NURBs surface representation for equation (2.15) can be found using the operators defined in 2.1.2, 2.1.3, namely surface addition and surface multiplication. The zero set of the surface $S(u,v)$ is the set of parametric values for the required intersection. Because both $F(u,v)$ and $S(u,v)$ share the same parametric domain, mapping the parametric domain information to $F(u,v)$ is trivial. $S(u,v)$ is a *scalar* surface, which leads to a simpler and faster computation. Assuming $w(u,v) \neq 0$, the zero set of $S(u,v)$ is computed using only the numerator of $S(u,v)$. Thus, even when $F(u,v)$ is a rational surface, the contouring computation can be performed on a scalar polynomial surface.

To find the contours, the scalar surface resulting from equation (2.15) is recursively subdivided so subsurfaces intersecting the contouring plane are isolated. At each stage, the scalar surface coefficients are classified into three categories:

1. all coefficients are positive.

2. all coefficients are negative.

3. coefficients with different signs exists.

Using the convex hull property of Bézier and NURBs surfaces, it is clear the first two cases have no intersection with the contouring plane. The third case

Algorithm 2.1

Input:
$S(u,v)$, input surface.
P, the contouring plane
τ, tolerance of subdivision used in termination criteria.

Output:
Subsurfaces of $S(u,v)$, intersecting the contouring plane.

Algorithm:
```
IntersectingSubSrf( S )
begin
  M ⇐ S control mesh.
  If termination criteria hold with τ
    return { S }.
  else if M control points are all positive or all negative
    return φ.
  else
  begin
    Subdivide S into two subsurfaces S^1 and S^2.
    return IntersectingSubSrf( S^1 ) ∪ IntersectingSubSrf( S^2 ).
  end
end
```

suggests the surfaces *may* intersect and further investigation is in order, so only the third type needs further subdivision. These steps are similar to the one presented in [23, 59].

Figure 2.1 shows this first stage of isolating the subsurfaces crossing the contouring plane.

Termination criteria obviously relate to flatness testing. However, it is also required that the cross section of the patch with the contouring plane be *simple*, where

Definition 2.1 *A* simple *patch during the contouring process is a patch which intersects the contouring plane along one and only one connected*

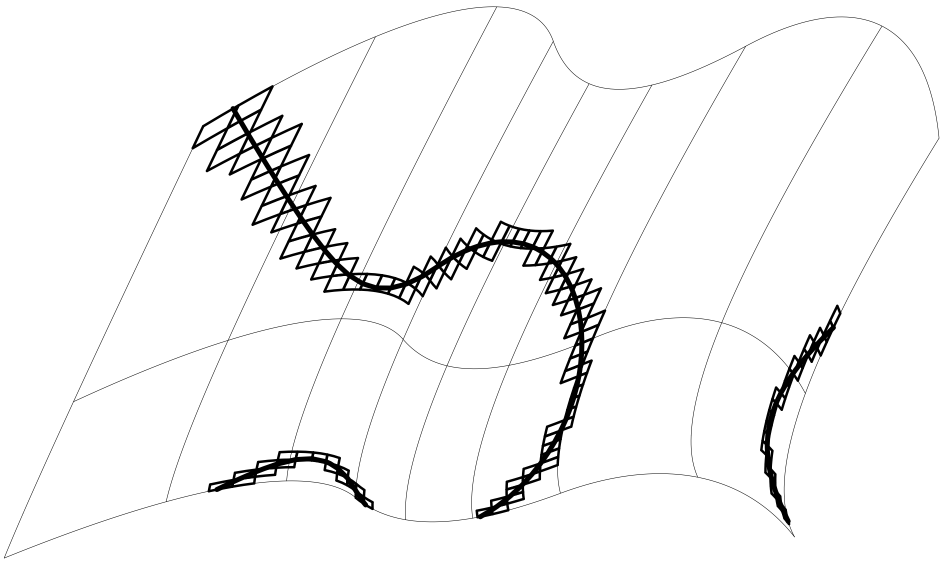

Figure 2.1. Subsurfaces intersecting the XY parallel contouring plane.

curve. Furthermore, this curve must start and end on two different boundaries of the patch.

It is clear from Figure 2.2 that tracing the patches to form a piecewise linear approximation of the contour may be ambiguous, because a single patch may have more than two (one in and one out) neighbors.

Coercing the termination condition to allow only simple patches at the lowest level simplifies the task of disambiguating and connecting the patches into a piecewise linear approximation. This termination condition also simplifies the problem of correctly identifying the contours at degenerate points such as saddles. These points can never satisfy the simplicity condition (definition 2.1), because at a saddle point four contour curves meet. The flatness criteria will terminate the subdivision, and will mark such points so they can be treated in a special manner, depending on the application.

Once traced into a a list of patches, one can pick the middle of each patch to form the piecewise linear approximation of the contouring curve. However, by using a higher order approximation on the intersection of the patch boundary and the

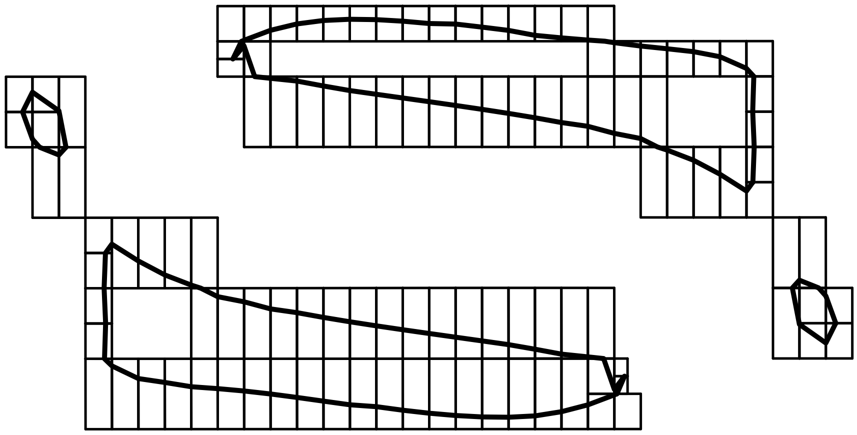

Figure 2.2. Subsurfaces chaining into piecewise linear approx. may be ambiguous.

contouring plane one can obtain a much better result (Figures 2.1 and 2.2). Because the patches are simple, a boundary crossing the contouring plane must have one of its ends above the contouring plane and the other below it. The use of a first order approximation (a linear segment connecting the two end points) to find the intersection with the contouring plane was found to be inadequate because it provided no information about the interior of the boundary curve. The subdivision approach described in 2.2.2 can be used. Numeric methods may be used as well considering that the single solution is bounded by the boundary curve end points. In practice, as in Figure 2.1, the secant method was used, which is derived in any introductory numerical analysis book, and which guarantees convergence because the patch (and boundary) are simple.

One can, at this stage, attempt to numerically improve the curve by “marching” along the surface as suggested in [23]. Furthermore, middle points may be introduced and improved as well. We found that for most purposes, as used in later Chapters, this stage was unnecessary and acceptable accuracy could be extracted in reasonable time using only subdivision. The robustness of this “marching” process has not been proven to be reliable enough.

Algorithm 2.2

Input:
```
C(u), input curve.
τ, tolerance of output.
```

Output:
```
Zero set of C(u).
```

Algorithm:
```
zeroSet( C )
begin
  P ⇐ C control polygon.

  If length of P, ||P||, is smaller than τ
    return { middle point of P }.
  else if P control points are all positive or all negative
    return φ.
  else
  begin
    Subdivide C into two subcurves C^1 and C^2.
    return zeroSet( C^1 ) ∪ zeroSet( C^2 )
  end
end
```

2.2.2 Contouring in E^2

Computation of zero sets of curves is, in general, a much simpler task because the result, for nondegenerate curves, is a finite set of points. A simple subdivision based algorithm exploiting the Convex Hull property of Bézier and NURBs curves can be easily formulated in a similar fashion [44]:

As pointed out in 2.2.1, numeric improvement is possible, but for our purposes subdivision what found more robust and sufficiently fast.

One can easily extend algorithm 2.2 to find the solution(s) of $C(u) = K$ for some constant K by defining a new curve $\hat{C}(u)$ as follows:

$$\begin{aligned}
\hat{C}(u) &= C(u) - K \\
&= \sum_{i=0}^{m} P_i B_i^k(u) - \sum_{i=0}^{m} K B_i^k(u) \\
&= \sum_{i=0}^{m} (P_i - K) B_i^k(u) \qquad (2.16)
\end{aligned}$$

using equation (2.6) and equation (2.10).

CHAPTER 3

OFFSETS

When it is dark enough you can see the stars.

Ralph Waldo Emerson

Offset curves and surfaces are very important in manufacturing, Therefore, computation and approximation of offset curves and surfaces have undergone extensive research. For curves, the offset is an intuitive operation and has been mathematically known for more than a hundred years [6, 58, 65]. The offset operation is closed for arcs and lines, i.e., an offset of an arc and a line are an arc and a line, respectively. This is not so, in general, for Bézier and NURBs curves, so approximations are usually derived.

Two methods for finding approximations to offset curves are commonly used. The first approximates the curve using piecewise lines and arcs and then finds the representation of the exact offset to the arc and line approximation. That approach was introduced [52] and used successfully in [12]. The second method attempts to approximate the offset by directly transforming the curve representation, in particular the control points [13, 18, 28, 36, 37, 50]. To improve the accuracy of the approximation in the second method, the original curve is subdivided [36, 37, 50] or manually refined [13] when the error is above a prespecified tolerance level. The same offset technique is then applied to each of the subdivided pieces. The original curve is usually subdivided in the middle of its parametric domain [36, 37, 50], although in general, that is not the optimal location. Curve inflection points have also been considered as splitting points for offsets [37].