GEOMETRIC MODELLING WITH

MULTIVARIATE B-SPLINES

by

Timothy Irwin Mueller

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

June 1986

ABSTRACT

The tensor product B-spline surface, while regarded as a powerful boundary representation for computer aided geometric design (CAGD), still suffers restrictions because of its inherent rectangular nature. One manifestation of this problem is the difficulty of modelling nonrectangular regions. The need for three, five, and six-sided regions often arises naturally in CAGD. In recent years, the theory of univariate B-splines has been extended to multidimensional spaces. These multivariate splines provide generalizations of the univariate spline that preserve desirable features while providing more general domains.

This thesis investigates the use of a parametric box spline surface representation over a three direction grid to define spline surfaces over three, four, five, and six-sided regions. Results are reported showing adaptive refinement of box spline surfaces and applications to display of shaded raster images and isoparametric curves. Modelling experiments which exploit the new capabilities provided by the box spline surface including nontensor product rectangular surfaces, nonrectangular surfaces, and hybrid nonrectangular regions composed of both box spline and tensor product surfaces are presented as well. Conditions for maintaining tangent plane continuity between nonuniform tensor product B-spline surfaces and box splines are developed.

To my parents, Gordon and Arvilla

CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

The author is indebted to many people for assistance in this research. First of all the author wishes to thank the committee and especially the chairperson, Elaine Cohen, for providing the topic and subsequently providing a great deal of positive reinforcement, encouragement, and consultation. The author is also grateful to the Alpha_1 group for providing software and social support. The author also wishes to acknowledge the University of Utah and IBM for providing graduate research fellowships. Photographic services were expertly provided by Mike Milochik.

This work was supported in part by the National Science Foundation (DCR-8341796 and MCS-8121750) and the Defense Advanced Research Projects Agency (DAAK11-84-K-0017). All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

CHAPTER 1

INTRODUCTION

The field of computer aided geometric design (CAGD) seeks to derive mathematical representations for the modelling of physical objects so that these objects may be designed and manipulated with the aid of computer systems; thus reducing design costs and increasing productivity. Since the first conception of this idea, researchers have strived to develop mathematical representations that were powerful and useful for such applications. Sources for these representations have come from a wide range of mathematical fields, from the simplest Euclidean geometry with points, lines, and planes; to the classical theory of conic sections with circles, arcs, ellipses, etc.; to models of solids bounded by planar and quadric surfaces; to curve and sculptured surface models based on interpolation and approximation techniques. Although very powerful representations have been developed, there are still engineering problems that cannot easily be solved using available techniques.

1.1 The B-Spline Representation

The B-spline representation is a curve and surface representation that has its roots in approximation theory. In this section a very brief overview of B-spline approximation and its properties is presented, along with some of the developments which have resulted in the use of B-spline based models in CAGD systems.

1.1.1 B-Spline Approximation

The theory of B-spline approximation was first introduced by Schoenberg and Curry [24], and later developed by de Boor, Cox, Mansfield, and others (see de Boor [7] for a survey). A B-spline approximation to a function f is a piecewise polynomial curve given by the following equation:

$$A_{f,T,k}(x) = \sum_{i=0}^{n} f(\xi_i)\, N_{i,k,T}(x)$$

where T is a nondecreasing sequence of real numbers $t_0 \leq t_1 \leq \cdots \leq t_{n+k}$ called the *knots* and the ξ_i's are called the *nodes* and are found in the interval $[t_0,t_{n+k}]$. The B-spline basis functions, $N_{i,k,T}(x)$, are smooth piecewise polynomials (with joints at the knots) of degree k-1 that are normalized so that $\sum_{i=0}^{n} N_{i,k} = 1$.*

Riesenfeld, in his thesis [51], applied the theory of B-spline approximation to CAGD. The thesis notes that curves and tensor product surfaces constructed with B-spline bases have many attractive properties for the representation and computation of three-dimensional objects in CAGD systems. First of all, it is noted that the B-spline approximation is a local approximation scheme. At a given value, x, there are at most k nonzero basis functions. This means that a local change in the approximated function, f, produces only a local change in the approximation $A_{f,T,k}$. A second important property is that the B-spline basis functions are nonnegative and sum to one. This guarantees that the approximation lies within the convex hull of the coefficients, $f(\xi_i)$. Perhaps the most interesting property, in view of the design process, is what is called the *variation diminishing property*. This property describes how the approximation "undulates." Roughly speaking, the variation diminishing property guarantees that the approximation will not intersect a straight line more than the function f itself does, and it thus "smoothes" the function f. This also guarantees that linear functions are reproduced exactly.

1.1.2 Using B-Spline Approximation for Design

1.1.2.1 B-Spline curves. To realize the desirable properties of B-spline approximation, a parametric B-spline curve in $\Re^d$ can be defined in terms of the B-spline basis functions described above.

$$\gamma(t) = \sum_{i=0}^{n} P_i\, N_{i,k}(t) \tag{1.1}$$

*To simplify the notation, the B-spline basis functions will subsequently be referred to as $N_{i,k}$, with the dependence on T implied.

Here the coefficients, P_i, are vectors in $\Re^d$ so $\gamma(t)$ maps an interval of the real line (which depends on the choice of T) to a piecewise polynomial curve in $\Re^d$. The points P_i can be thought of as a "control polygon" for the curve. This type of representation is desirable because the method inherits the properties of the approximation scheme from which it is derived, in this case, the convex hull property and the variation diminishing property. The result of equation (1.1) is that the curve $\gamma(t)$ is a smooth piecewise polynomial curve that approximates the *shape* of the control polygon. This provides the designer with an intuitive method for controlling the shape of the curve being designed. The fact that $\gamma(t)$ is defined parametrically increases the range of shapes that can be defined and also frees the designer from being concerned with coordinate systems.

1.1.2.2 B-Spline surfaces. The B-spline approximation method, as originally derived, is inherently univariate, so to generalize the B-spline curve design method to surfaces requires an extension of the B-spline basis to higher dimensions. One way to do this is to simply take products of the univariate B-splines. If we take two nondecreasing sequences of scalars (as above) $S = \{s_i\}$ $i = 0, \ldots, m+k_1$, and $T = \{t_j\}$ $j = 0, \ldots, n+k_2$ a bivariate tensor product B-spline basis can be defined by taking products of the univariate B-splines. These are called the tensor product B-spline basis functions.

$$B_{i,j}(u,v) = N_{i,k_1}(u)\, N_{j,k_2}(v)\ .$$

A tensor product B-spline surface can now be defined as a map from $\Re^2$ to a surface in $\Re^d$ as follows:

$$S(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} P_{i,j}\, B_{i,j}(u,v), \qquad \text{for } P_{i,j} \in \Re^d.$$

Here the points $P_{i,j}$ can be thought of as a "control mesh." The tensor product surface inherits most of the properties of the univariate B-spline curve except the variation diminishing property, the concept of which has not readily extended to higher dimensions. The tensor product basis functions are nonnegative and sum to one, meaning that the surface is contained by the convex hull of the control mesh. The basis functions also retain the local control property, allowing changes to the control mesh to affect only a local area of the surface. Also maintained is the piecewise polynomial nature of the B-spline approximation method. The tensor product basis functions are piecewise polynomial surface generalizations of the univariate B-spline basis functions. Whereas the univariate B-splines are polynomial curves of degree k-1 along each interval $[t_j, t_{j+1}]$, products of such functions are polynomial surfaces of degree $(k_1-1)+(k_2-1)$ over each rectangular domain $[s_i, s_{i+1}] \times [t_j, t_{j+1}]$ in $\Re^2$.

1.1.2.3 Discrete B-splines. The theory of discrete B-spline approximation has also proven useful in the application of continuous B-spline curve and surface methods to CAGD. Cohen, Lyche, and Riesenfeld [20] applied the theory of discrete B-splines to knot refinement to provide efficient algorithms for arbitrary refinement of nonuniform B-splines. This theory has provided the grounds for many applications including efficient computational algorithms for raster graphic display of B-spline curves and surfaces, algorithms for providing flexible modelling tools using B-spline bases, as well as algorithms for the computation of the intersection of two B-spline surfaces and, consequently, the use of B-spline boundary representations for volume modelling techniques [52, 54, 18, 53, 33].

1.1.2.4 Representation power. In view of the above results, it can be concluded that the B-spline representation is a powerful modelling tool for CAGD, capable of representing complex sculptured surfaces. With the results regarding surface intersection techniques [53], it has been shown that the B-spline representation can also be used to represent the primitive objects and boolean operations typically used in constructive solid geometry (CSG) systems.*

However, tensor product B-spline surfaces still suffer restrictions because of their inherent rectangular nature. Tensor products of univariate B-splines form bases for bivariate spaces of piecewise polynomial functions, and, as previously noted, there are lines along which individual polynomial surface pieces meet. Since the basis functions are products of univariate functions, these lines are necessarily parallel to the parametric axes; dividing the plane into rectangular regions. This is the major limitation of the tensor product surface that has motivated the study of alternative representations.

---

*See [50] for a survey of these modelling techniques.

1.2 Anomalous Regions

Although many geometrically complex objects can be modelled using tensor product B-spline surfaces, Forrest points out some of the problem areas that can arise in a model that is based on a purely rectangular representation [38]. In certain modelling situations, a three or five sided region may be desirable in an otherwise rectangular assembly. Forrest calls these "anomalous regions." An often used example of a five-sided region is the blend that occurs where an airplane wing meets the fuselage. In Figure 1 it is shown that this region is defined by five boundary curves or feature lines. The corners of a rounded box have been used as an example of the triangular region. If a cube is "rounded" with edges of equal radii, a triangular region is needed at the corners. Figure 2 illustrates this type of region.

1.2.1 Blend Surfaces

Both of the above examples can be seen as special cases of the "blend surface" problem. This catch-all term includes (conceptually) simple problems such as filleting and chamfering simple solids (e.g., cubes, cones, cylinders, etc.), to complex problems of fillets or more arbitrary blends between two sculptured surfaces, such as the airplane example. Rossignac and Requicha [49] categorize blends into four major groups:

- *Functional* blends. These are surfaces which are governed by strong functional constraints; where an analytic description of the geometry is important. The airplane example fits into this category.

- *Aesthetic* blends. These are blends that are important for appearance rather than function. It this case a precise specification of the surface is not important, only that it "does something nice."

- *Fairings*. These are transition surfaces which are relatively large compared to the surfaces being blended, and whose shapes are not strongly constrained by function or aesthetics.

- *Fillets and Rounds*. These are relatively small blends, usually defined to have circular cross sections. These are weakly constrained by function, and are often defined by the manufacturing process. The main purpose for the fillet or round is to eliminate sharp edges to reduce stress and facilitate fabrication.

Blend surfaces are not necessarily synonymous with anomalous regions, but there is a close association, as blend surfaces are difficult to model in both CSG and sculptured surface

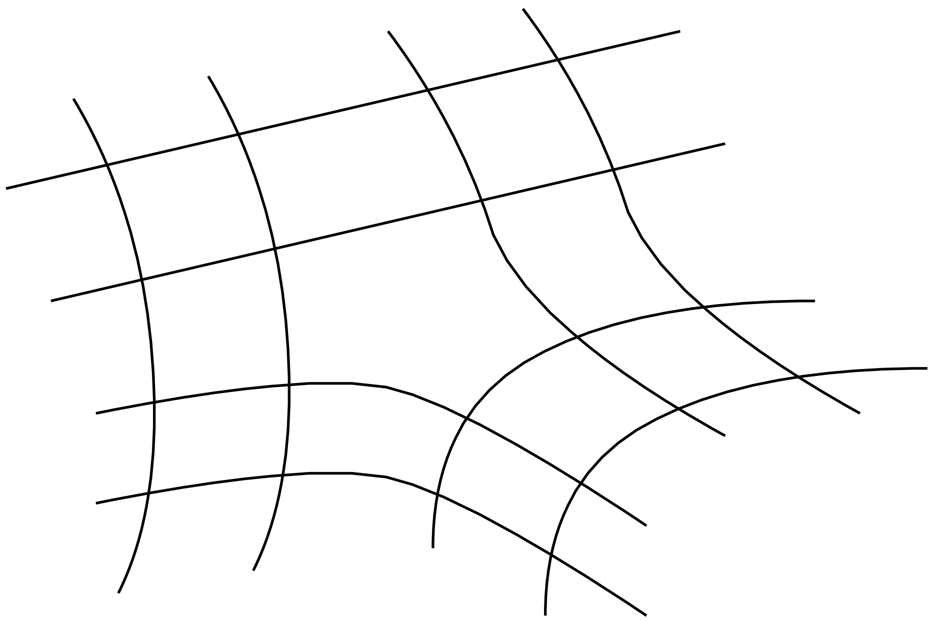

Figure 1: Five-sided region at wing/fuselage blend

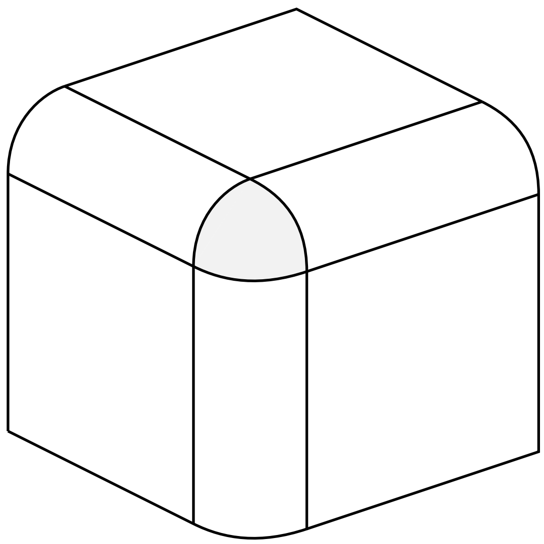

Figure 2: Corner region of a rounded cube

representation modelling paradigms. In research addressing the problem of blend surfaces in CSG modellers, Middleditch and Sears [46] note the association between the blend surface and the nonrectangular region by pointing out that sculptured surfaces are of limited use because "... it is difficult to represent blend surfaces with the rectangular patches usually employed." Indeed, the tensor product surface, being inherently rectangular in nature, presents certain problems in modelling nonrectangular regions.

1.3 Overview

The aim of this thesis is to look at the possible application of more general forms of approximation theory to the geometric modelling process. In recent years, the theory of univariate spline approximation has been extended to multidimensional spaces [45, 25, 26, 42]. This new theory gives rise to the possibility of piecewise polynomial surfaces being defined over more general domains (e.g., nonrectangular ones). It has been hoped that these multivariate splines might be applicable to the above mentioned problems in CAGD; however, bridging the gap between these approximation schemes and a geometric modelling system has been difficult because of the complex nature of multivariate spline theory.

In view of this, there are two major motivations for this thesis. First of all, more general forms of mathematical representation for CAGD need to be derived. This need is clear from some of the preceding examples. These examples illustrate real engineering problems that should eventually be within the capabilities of CAGD systems. The second motivation is to gain a better understanding of multivariate splines in general, and what is required to apply such theories to geometric modelling problems.

The following is a brief outline of the thesis. Chapter 2 discusses previous techniques that have been used for modelling anomalous regions and Chapter 3 gives an introduction to the various forms of multivariate spline approximation. Chapters 4, 5, and 6 present the results of the design and implementation of an experimental modelling system based on a form of multivariate spline approximation, and Chapter 7 discusses the conclusions and hopes for future research.

CHAPTER 2

MODELLING ANOMALOUS REGIONS

This chapter will explore existing techniques that could be or have previously been used for the modelling of anomalous regions. This discussion will cover methods based on the use of triangular regions, degenerate rectangular regions, arbitrary convex regions, algorithms for generating surfaces on arbitrary mesh topologies, and some techniques and algorithms for generating blend surfaces.

2.1 Surfaces Defined over Triangular Regions

2.1.1 Local Interpolation Schemes

There are many schemes based on triangles for interpolating and approximating surfaces [2, 3, 39]; however most of these techniques fall into the interpolation category. Research in triangular regions first stemmed from the scattered data problem (the problem of fitting a surface to arbitrarily spaced data in the plane). Since interpolating or approximating scattered data does not fit into rectangular or tensor product schemes, a natural solution is to produce a set of triangles (called a triangulation) from the data, develop a triangular interpolant (with sufficient smoothness) and produce a surface through the given data. Some of these techniques have developed from the rectangular interpolants of Coons and Gregory and involve the use of "projectors" and Boolean sums of such projectors.*

A simple example of projectors and Boolean sums is the bilinearly blended Coons patch. Let P_1 and P_2 be linear projectors defined to operate on a primitive bivariate function F(u,v) as follows:

---

*See Barnhill [2, 3] for surveys of these methods.

$$P_1F = (1\text{-}u)\ F(0,v) + u\ F(1,v)$$

$$P_2F = (1\text{-}v)\ F(u,0) + v\ F(u,1).$$

The Boolean sum of these projectors is denoted

$$P_1 \oplus P_2 = P_1 + P_2 - P_1P_2,$$

where P_1P_2 is the composition, or tensor product, of the projectors. The bilinearly blended Coons patch is then defined as $(P_1 \oplus P_2)F$, which interpolates positional data all around the edge of the patch. This rectangular interpolation scheme produces polynomial surfaces that can be joined to make a C^0 surface. More continuity can be provided by defining higher order projectors that in turn require more data from the primitive function F.

Triangular interpolants can be derived by defining three such projectors each interpolating in a direction parallel to an edge of the standard triangle as seen in Figure 3. One can then define a symmetric "trilinear" interpolant, Q^*, where

$$Q^* = P_1 + P_2 + P_3 - P_1P_2P_3.$$

There are many schemes based on this approach (for a survey see [3]). The various projectors in these cases are polynomial or rational polynomial equations in two variables involving various data from the primitive function F. For example, the projectors may require as little as three function values of F, in the linear case, to as many as twenty-one parameters, including function values and first and second directional derivatives, for a C^1 quintic interpolant.

These methods are useful in the scattered data interpolation problem, but have certain limitations in CAGD. First of all, an arbitrary polygonal region can be modelled using a triangular interpolant, but the result is dependent on a triangulation algorithm which may be coordinate system dependent. This means that the same data, in different coordinate systems, might not produce the same surface. A second problem is that these schemes are inherently interpolatory. In a modelling environment there is no primitive function to provide data, and the modeller is left to make up or estimate data such as second directional derivatives of

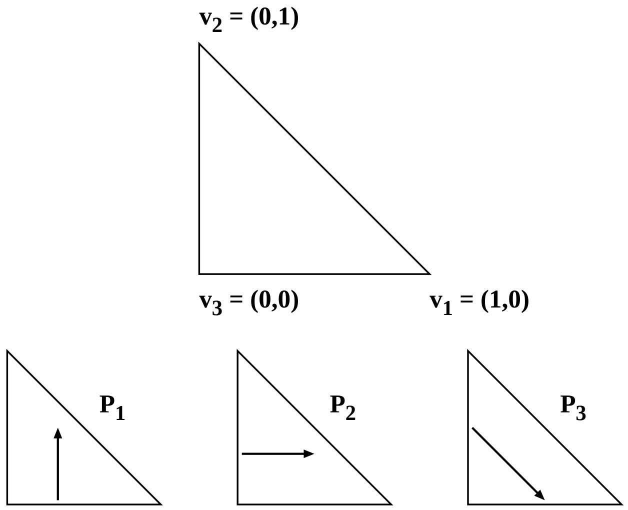

Figure 3: The standard triangle and three projectors

bivariate functions. This is a nonintuitive process, to be sure. Thirdly, these methods have limited continuity. The surfaces are polynomials or rational polynomials of fixed degree and continuity class. If the application requires higher continuity, new schemes must be invented and, as noted above, the required data for such schemes can become complex and difficult to specify.

2.1.2 Local Approximation Schemes

The triangular methods discussed so far have been interpolatory in nature, seeking to match data from a primitive function. In contrast, there are forms of triangular approximation which seek to approximate or smooth the given data. One such method is the triangular Bézier method.

The Bézier triangle is a generalization of the univariate Bernstein-Bézier method of approximation. This approximation method is based on the binomial distribution and produces polynomial curves and bipolynomial tensor product surfaces over rectangular domains.* The

---

*See Riesenfeld [51], Gordon and Riesenfeld [40], or Bézier [4] for more detailed discussions of Bernstein-Bézier methods.

generalization to triangular domains is based on bivariate Bernstein polynomials defined in terms of barycentric coordinates of the given triangle. Surfaces constructed with such bases are bivariate polynomial surfaces defined over triangular regions that interpolate their corner points and have univariate Bézier curves as their boundaries.

This local approximation method is attractive for CAGD since it is coordinate system independent and does not rely on estimating derivatives of nonexistent primitive functions. Also, many subdivision algorithms exist for efficient numerical computation [39].

However, the Bézier triangle used to model an arbitrary region gives only a C^0 surface, analogous to piecewise Bézier curves. To produce higher degrees of continuity, constraints on the individual meshes must be introduced which add complexity of the local method [35, 36]. On the other hand, when used globally over a large triangular region the degree of the polynomial surface might become larger than is desirable.

2.2 Surfaces Defined over Arbitrary Regions

2.2.1 Interpolation over Arbitrary Regions

One example of a surface defined over a arbitrary convex polyhedral domain is a further generalization of the projector technique defined in Section 2.1. Gregory defined a type of projector that interpolates a primitive function and its derivatives along two edges of a polyhedral domain [41]. To produce an arbitrary patch, certain convex combinations of these "two sided" interpolants are formed to produce the surface. This method eliminates the triangulation preprocess associated with strictly triangular methods, but is still dependent on positional and derivative data from a primitive function F.

2.2.2 An Algorithmic Approach

Another method for generating arbitrary regions is the recursive subdivision techniques developed by Catmull and Clark [13] and Doo and Sabin [34]. These algorithms extend a uniform floating B-spline subdivision algorithm to arbitrary regions in the bicubic and biquadratic cases. In using a geometric interpretation of the subdivision algorithm, various types of subdivision points are identified. For example, a "face point" is a point in the refined mesh that is computed by averaging four points from a "face" of the old mesh.* Similarly, edge and vertex points can be defined. The subdivision algorithm can then be described as a set of rules relating new mesh points to old mesh points. The algorithm can then be extended to arbitrary meshes by generalizing these rules. For example, the new rule for computing a face point would allow a face to have n points. The rectangular subdivision rule is then a special case when n = 4.

Repeated applications of the algorithm tend to "rectangularize" a nonrectangular mesh, except near certain points called *extraordinary* points. These are points that do not have four edges incident on them. After one iteration of the algorithm, all faces are four-sided and the number of extraordinary points remains constant. The faces surrounding the extraordinary points continue to shrink in area as the subdivisions proceed. Catmull and Clark implemented the algorithms and from the generated images were able to speculate that the surfaces were at least C^1 near the extraordinary points; although no proof was given. Indeed, it can be shown that these algorithms produce tensor product B-spline surfaces everywhere except near these extraordinary points where the behavior was not known.

Doo and Sabin developed a method to describe the behavior near these extraordinary points. This analysis is based on eigenvalues of certain matrices used in the subdivision algorithm. Using this they found ways to "fine tune" the algorithms to produce more continuity near the extraordinary points. Doo and Sabin developed a C^1 biquadratic method using their analysis, but a C^2 bicubic has not been developed.

This technique is an interesting one in that it incorporates the troublesome regions into an existing representation in a systematic way: The algorithms used by one representation are generalized to handle the anomalous region. This way the nonrectangular region is integrated with the rectangular one without special treatment. However, the behavior near the extraordinary points can still cause problems, and there is no analytic means of describing the surface near these points.

---

*These faces are not necessarily planar.

2.3 Blend Surfaces

The blend surface was mentioned in Chapter 1 as characterizing many of the types of anomalous regions where problems occur. Four types of blend surfaces were identified by their design constraints. It is perhaps more applicable to recategorize these into two main groups:

1. *Exact* blends. These are cases where precise control of the shape and an analytical description of the blend surface is required. Surfaces with strong functional constraints, such as the airplane example, certainly fall into this category. Fillet surfaces may or may not need to be precisely defined depending on the application. Certain aesthetic applications may also fit here because of the precise control that may be needed to achieve to desired design.

2. *Approximate* blends. These are cases where the shape is only required to be reasonable. Fairings or fillets are in this category when they are only required to provide "some" transition from one surface to another that is reasonable.

2.3.1 Blending Between CSG Primitives

CSG modelling paradigms have recently been extended to include blend surfaces [46, 49]. This research is not concerned with defining surface representations for nonrectangular regions, as such, but is an interesting alternative to modelling regions that are otherwise difficult to represent with boundary representations or traditional CSG approaches.

2.3.1.1 Piecewise toroidal blends. Rossignac and Requicha [49] present an extension to CSG based on the "rolling sphere" definition of a fillet blend. This definition simply states that a fillet surface between two surfaces to be blended is swept out by rolling a sphere of radius r between the two surfaces. The basic steps of this technique are to define a "blend primitive" which is solid constructed from offset primitives, and then trim the blend primitive to have the correct boundaries.

Since CSG modelling techniques rely on the ability to classify points in relation to a solid primitive and thus perform boolean operations, one must ensure that the extended CSG representations are closed under boolean operations. Since the blending operations introduce what Rossignac and Requicha call "canal surfaces," which are not represented in CSG modellers, they introduce an extended form of CSG called CSGO (CSG with offsetting) and approximate the blend surfaces by a "spine" curve and a radius. The spine curves are defined as piecewise circular curves to simplify the intersection and offset computations. The blend primitives are piecewise toroidal and can be represented by CSGO primitives. This allows the typical algorithms for intersection, boolean combinations, and display to be easily extended.

This technique certainly solves the problems of modelling rounded primitives and approximates fillet blends with circular cross sections between primitives. However, if more arbitrary blends are required (e.g., with cross sections that are not circular) or blends between sculptured surfaces are required the CSGO representation will not be sufficient.

2.3.1.2 Blended conics. Middleditch and Sears [46] generalize the theory of conic sections to provide a blended conic representation for blending between CSG solids. A pencil of conics can be defined in the plane by three line equations and one extra parameter as follows:

$$(1 - \lambda) l_1 l_2 + \lambda l_3^2 = 0 \,. \tag{2.1}$$

Given three points P_1, T, and P_2; three lines l_1, l_2, and l_3; and the parameter λ, the implicit quadratic curve that is defined by equation (2.1) passes through the two points P_1 and P_2, and is tangent to the lines l_1 and l_2 as seen in Figure 4. This one parameter family of conics is called a pencil of conics. The last degree of freedom, defined as λ in equation (2.1), can be specified by picking a point, P_c (called a "shoulder point"), in the interior of the triangle P_1TP_2 to be interpolated by the curve. This conic can be seen as a blend between the degenerate hyperbola (the pair of lines l_1 and l_2) and the line l_3 (also a degenerate conic). Figure 4 shows P_c appropriately chosen such that the resulting conic is a parabola.

To generalize the blend conic to surfaces, a blend surface can be defined which is tangent to the two base surfaces along certain curves, which is the generalization of the curve tangent to two lines at two points above. The generalization of the shoulder point is to define a "spine" curve, which will be interpolated by the blend surface. Middleditch and Sears describe a method for constructing blend surfaces that have the following properties:

- The blend surfaces interpolates a spine curve defined to be a fixed distance βd, for some constant β, from both the base surfaces.
- The blend surface is tangent to the two base surfaces along curves that are offset by a distance d from the alternate base surface.

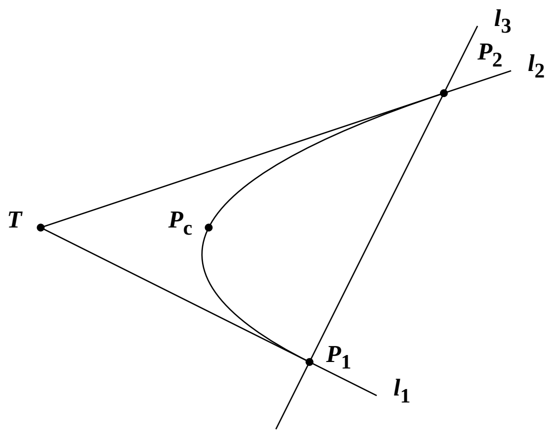

Figure 4: Conic section through three points and tangent to two lines

The surface can be controlled by specifying the distance d and the parameter β. The resulting blend surface does not necessarily have conic cross sections, however, and grows in order as the base surfaces grow in order. For example, when the base surfaces are normalized planes, the blend surface is indeed a conic cylinder; however a blend between a cylinder and a plane produces a fourth order function that does not have conic cross sections.

This method solves the problem of modelling rounded and chamfered solids and does it with exact implicit equations; although these blends only have conic cross sections when the base surfaces are planar. The method is more general that the piecewise toroidal method above for blending between more complex objects such as spheres, cylinders, and tori, but the surfaces are no longer quadrics and can reach rather large degree. Although there are exact implicit equations for these blends, a designer has limited control over their shape and they may or may not be useful for the *exact* blends as identified above.

2.3.2 Blending with Sculptured Surfaces

Since CAGD systems based on rectangular surface patches are well known and used by research and industry, various techniques have been devised for modelling blend surfaces using rectangular sculptured surfaces. Although some effort has been made to incorporate nonrectangular surfaces into modelling systems, most methods involve the creative use of rectangular surfaces to model blend regions, sometimes requiring degeneracies of one form or another.

Bézier in his paper about the use of the system UNISURF at Renault describes various methods for modelling the triangular regions that result in certain areas of car body design [4]. Three possible techniques are:

1. Use a triangular patch of some sort. This leads to problems matching derivatives with the surrounding rectangular patches because the triangular method is usually based on different representation.

2. Divide the triangular area into three rectangular ones. This method also has difficulties with continuity at the point where the three patches meet.

3. Use a degenerate patch. This is a very common technique; however, collapsing an edge to a point gives a certain unnecessary bias to the point parametrically. A degenerate edge also means that the normal vector to the surface is undefined along the edge and the normal vector is important in many graphics and modelling applications.

Most of these techniques succeed in simplifying the problem by transforming the problem region into a new one, but do not eliminate it. In the first case, using different representations shifts the problem to matching derivatives from two different representations and then only provides fixed degrees of continuity. In the second case, the problem region becomes a problem point and similar difficulties occurs with degenerate edges. Forrest sums up the difficulties in modelling these regions as follows:

> Attempts to define complex objects solely in terms of regular arrays of rectangular or even triangular patches lead to difficulties which manifest themselves, typically, as regions which are unintentionally flattened, lacking fairness [37].

Although there can be problems with using rectangular patches, successful schemes for modelling certain blend surfaces have been designed and implemented. The next section covers some of these techniques that have been based on tensor product B-spline surfaces.

2.3.3 B-Spline Blending Surfaces

2.3.3.1 Rounded primitives. Rounded corner regions on cubes (as in Figure 2) have traditionally been used as examples of the need for triangular regions [38, 2]; however, Stay [54] notes that the triangle is the special case in this problem, and in general the region is four sided. Stay presents a method for modelling these regions using tensor product B-splines.

Stay's thesis presents methods for modelling CSG-type primitive objects, as well as rounded edge extensions of CSG primitives, as volumes bounded by tensor product B-spline surfaces. In all cases the blend surfaces have circular cross sections and in most cases degenerate edges are not required on the blend surfaces. The reason for this is that the problem areas are usually transferred to planar regions from the blend regions. For example, the chamfer at the end of a rounded cylinder is easily modelled as a circular arc swept around a circular path; a well-defined rectangular patch. The difficulty is transferred to the caps of the chamfered cylinder, which are disks, not inherently rectangular in nature. But since these are planar regions a degenerate normal (if necessary) is not as detrimental.

The difficult case occurs when a cube is rounded with equal radii on all the edges. In this case a triangular region at the corner cannot be avoided and a degenerate edge must be constructed at one of the corners of the triangular region.

2.3.3.2 B-Spline fillets. A more general facility for filleting between arbitrary B-spline surfaces has also been developed using tensor products. Fish [23], using the ability to intersect B-spline surfaces developed by Thomas [53], derives a fillet surface by offsetting the base surfaces and intersecting the offset surfaces with the alternate base surfaces. This provides two intersection curves, one on each base surface. The cross boundary tangents are then computed along these curves and a fillet surface in constructed that matches these tangents. The fillet surface has circular cross sections and approximates the "rolling sphere" definition of a fillet.

This technique has been used successfully to model complex fillets such as the region where the airfoil of a turbine blade meets its platform. These fillets are very important to avoid the high stress that can occur in these areas. This method avoids nonrectangular regions by constructing the fillet as a sweep operator and using B-spline intersection techniques to find the path along which to sweep. This models a very specific type of blend surface (C^1 with circular cross sections) but applies to a very wide range of base surfaces. A more powerful tool is still required to model highly complex blends, like that of the airplane wing example.

2.3.3.3 B-Spline joins. Donahue, in his thesis [33], looks at the construction of more general C^1 blends between tensor product B-spline surfaces, which he calls "joins." The aim of this work is to provide modelling tools to construct what we have identified as *approximate* blends. These are for cases where functional constraints are not as important. The idea is to provide an intuitive means of designing surfaces in these regions and to give a designer a powerful enough tool to accomplish the design. The methods presented construct blend surfaces without computing intersection curves between B-spline surfaces as used in the previously discussed B-spline filleting technique.

The blend surface, called the "join" surface, is constructed to blend between the end of the "joining" surface (which is some generalized sweep) and the "joinee" surface. A rectangular section of the joinee surface is appropriately "cut out" and the join surface is then constructed to provide the transition from the end of the joining surface to the hole in the joinee surface.

It is of course desirable to provide a join surface that is C^1 continuous with both the joinee and the joining surfaces, but Donahue points out there are tangent discontinuities which are due to the rectangular nature of the surfaces involved. The thesis presents methods to construct join surfaces which minimize the effect of these discontinuities.

2.4 Multivariate Splines: A Possible Solution

The techniques discussed so far for dealing with anomalous regions and blend surfaces have relied on interpolation techniques, special representations imbedded in rectangular mesh schemes, offsetting CSG primitives and B-spline surfaces, or representing anomalous regions with degenerate rectangular regions. A more elegant solution might be one in which the representation is not inherently rectangular, but includes rectangular regions as a special case. This would yield a uniform representation of all desired regions without special cases for anomalous regions. An object could be modelled as a whole, rather than piecing together different representations with complicated blending procedures. And when the design process is naturally divided into parts, as is often the case, a uniform representation greatly aids the process of integrating the individual pieces into the final model.

The theory of multivariate spline approximation may be applicable to the above problems in CAGD. The multivariate spline is a true generalization of the univariate B-spline defined in any dimension. A basis for a space of bivariate piecewise polynomial surfaces could, in theory, be defined on arbitrary polygonal regions of the parametric plane (including rectangular ones). However, research in multivariate splines is still in its infancy and there are many issues to be solved. One of those issues is how to use the enormous flexibility inherent in the multivariate spline to provide a larger capability for modelling three dimensional objects.

CHAPTER 3

MULTIVARIATE SPLINES

In this chapter a brief introduction to multivariate spline approximation is presented including definitions of the various forms. Some notation is presented, and examples are given of the univariate spline as a special case of the multivariate spline.

3.1 Notation

Before beginning, some notation is presented that will aid in the discussion of multivariate splines. Standard multi-index notation will be used to deal with n-dimensional spaces. For a set $A \subset \Re^s$, $\langle A \rangle$ and $[A]$ are the linear span and the convex hull of A, respectively. $A \setminus B$ is the set of elements in A but not in B (set difference). For vectors in $\Re^d$, where $d > 1$, superscripts represent enumeration, and subscripts represent components of vectors. Therefore x_i^j is the i^{th} component of the vector x^j. Scalars will be enumerated by subscripts, e.g., $t_i \in \Re$. The s-dimensional space of integer vectors, $\mathbf{Z}^s$, will be denoted by $\mathbf{J}$. The n-dimensional unit cube $[0, 1]^n$ is denoted by $\mathbf{I}^n$.

3.2 The Multivariate B-Spline

One of the goals of multivariate spline research is to provide an alternate generalization of the univariate spline that admits arbitrary knot configurations while preserving desirable features. With this as a motivation, some background material on the definition and properties of the univariate spline is in order.

3.2.1 The Univariate B-Spline

There are three notable formulations for defining the univariate B-spline, each lending different insights to its various properties. From [8], the k^{th} order B-spline is defined as a k^{th} divided difference of the truncated power function.

Definition 1: Let **t** := (t_i) be a nondecreasing sequence (which may be finite, infinite, or biinfinite). The i^{th} (normalized) B-spline of order k for the knot sequence **t** is denoted $N_{i,k}$ and is defined by the rule

$$N_{i,k}(x) := (t_{i+k} - t_i)[t_i, \ldots, t_{i+k}](\cdot - x)_+^{k-1}, \text{ all } x \in \Re,$$

where $[t_i, \ldots, t_{i+k}]f$ is the k^{th} divided difference of the univariate function f defined recursively as follows:

$$[t_i, \ldots, t_{i+k}]f = \frac{[t_{i+1}, \ldots, t_{i+k}]f - [t_i, \ldots, t_{i+k-1}]f}{t_{i+k} - t_i}, \quad t_i \neq t_{i+k}, \; k > 0$$

and

$$[t_i]f = f(t_i).$$

The "dot" notation used in the definition ($(\cdot - x)_+^{k-1}$) indicates that the divided difference is taken with respect to some variable "$\cdot$" and x is constant with respect to the divided difference.

From this definition, some properties of the B-spline are evident. First of all, $N_i = N_{i,k}$ has small support, i.e.,

$$N_i(x) = 0 \quad \text{for } x \notin [t_i, t_{i+k}].$$

Since $f(t) = (t - x)_+^{k-1}$ is a polynomial of degree < k on $[t_i, t_{i+k}]$ when $x \notin [t_i, t_{i+k}]$, the divided difference $[t_i, \ldots, t_{i+k}]f = 0$ (by properties of divided differences). From this it follows that at most k B-splines are nonzero on any particular interval $[t_j, t_{j+1}]$.

An equivalent definition of the B-spline is a recursive one which is due to de Boor, Cox, and Mansfield [7], commonly called the de Boor-Cox algorithm.

Definition 2: (de Boor-Cox)

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

where

$$N_{j,1}(x) = \begin{cases} 1, & t_j \le x < t_{j+1} \\ 0, & \text{otherwise.} \end{cases}$$

This formulation demonstrates that a B-spline can be defined in terms of lower order B-splines, and therefore computed recursively. It also emphasizes the piecewise polynomial nature of the B-spline and provides an efficient stable means of evaluation.

The third and perhaps most interesting definition (for visualization purposes) is the geometric one. This definition as well as the divided difference formulation were both described in an early paper by Curry and Schoenberg [24]. The geometric definition relates the k^{th} order B-spline to a k-dimensional simplex, in which the vertices of the simplex project orthogonally onto the knots of the B-spline in $\Re^1$. The value of the B-spline at any point is the (k-1)-dimensional volume of the cross section of the simplex at that point.

Let P be the orthogonal projection of $\Re^n$ onto $\Re^1$ and $y^0, \ldots, y^n$ be vectors in $\Re^n$ in general position such that $P(y^i) = t_i$ for a sequence of scalars t_i (knots). Also let σ be the n-dimensional simplex defined by $y^0, \ldots, y^n$. The geometric definition of the B-spline of order n in terms of σ is as follows:

Definition 3: Geometric Definition of the B-spline.*

$$M(t \mid t_0, \ldots, t_n) = \frac{\mathrm{vol}_{n-1}\{y \in \sigma \mid P(y) = t\}}{\mathrm{vol}_n \sigma} .$$

Some examples will illustrate this definition. In Figure 5 an example of a linear B-spline

---

*This definition is normalized so that the integral of the B-spline function is 1, i.e., $\int M(t|t_0, \ldots, t_n)\, dt = 1$. The B-spline normalized this way is denoted as M(t) and the B-spline normalized to sum to 1 is denoted as N(t) as defined previously.

is given. The value of the spline is the length of the vertical line intersecting the triangle divided by the area of the triangle. The spline curve mimics the shape of the triangle, but may differ by a constant. This definition shows the B-spline to be a piecewise polynomial (linear in this case) with joints at the knots. Figure 6 shows the geometric definition of a quadratic B-spline. Here the simplex is a tetrahedron in $\Re^3$. In the case shown there are only single knots and there are some cross sections that are triangular and some that are quadrilateral. This ensures a smooth (C^1) transition between the quadratic cross-sectional area functions.

The geometric definition, while not useful for computation, provides many interesting insights into the B-spline. Perhaps the most interesting is the ease of extension to higher dimensions. Let P be the orthogonal projection of $\Re^n$ onto $\Re^s$ and let y^i be vectors in $\Re^n$, in general position, and $x^0, \ldots, x^n$ be vectors in $\Re^s$ such that $P(y^i) = x^i$. For $n > s$ and $x \in \Re^s$ a multivariate version of definition 3 is as follows:

Definition 4: Multivariate Geometric Definition.

$$M(x \mid x^0, \ldots, x^n) = \frac{\mathrm{vol}_{n-s}\{y \in \sigma \mid P(y) = x\}}{\mathrm{vol}_n \sigma} .$$

De Boor took this generalization to be the definition of the multivariate spline in his survey paper of 1976 [7]. De Boor admitted that, although this was an interesting development, it was unclear whether this generalization would lead to any useful applications.

Since the early developments, much has been accomplished in the field of multivariate spline approximation theory. Algorithms for the computation of individual basis functions exist [44], many of the properties of the univariate B-spline have also been proven for the multivariate B-spline [25], and spaces for approximating with linear combinations of multivariate B-splines have been developed [26, 27, 28, 42].

3.2.2 Simplex Splines

The key to the unification of the geometric definition with an analytical one is the Hermite-Gennochi formulation for the divided difference of a function.

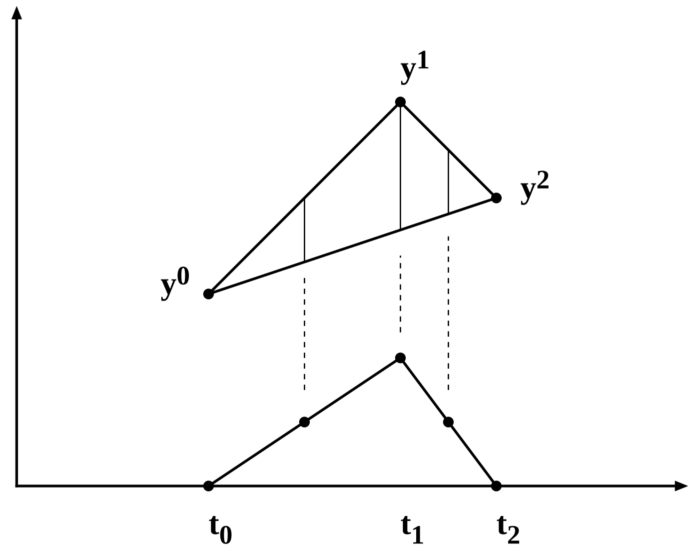

Figure 5: Geometric definition of linear B-spline

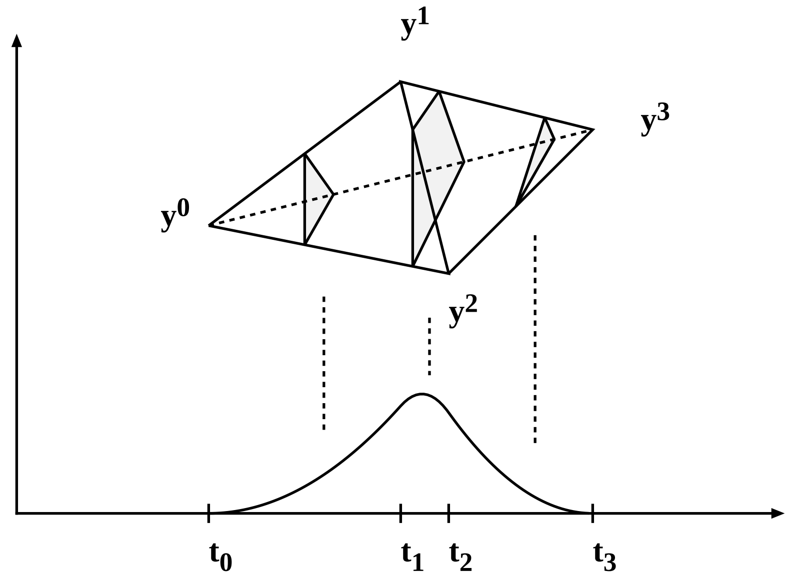

Figure 6: Geometric definition of quadratic B-spline

Theorem 5: (Hermite-Gennochi) For $t_i \in \Re^1$,

$$[t_0, \ldots, t_n]g = \int_{S^n} g^{(n)}(\nu_0 t_0 + \cdots + \nu_n t_n)\, d\nu_1 \ldots d\nu_n$$

where the integration is over the standard n-simplex,

$$S^n = \{(\nu_0, \cdots, \nu_n) \mid \nu_0 + \ldots + \nu_n = 1, \nu_j \geq 0, j = 0,1, \ldots, n\}.$$

Another form of the divided difference of a function is in terms of B-splines. This can be obtained by a Taylor expansion of the function g or by the application of a theorem of Peano (see Davis [32, p. 69]).

$$[t_0, \ldots, t_n]g = \int_{\Re^1} M(t \mid t_0, \ldots, t_n) g^{(n)}(t)\, dt . \qquad (3.1)$$

Combining theorem 5 and equation (3.1) gives the important relation

$$\int_{\Re^1} M(t \mid t_0, \ldots, t_n) h(t)\, dt = \int_{S^n} h(\nu_0 t_0 + \cdots + \nu_n t_n)\, d\nu_1 \ldots d\nu_n, \qquad (3.2)$$

that holds for any locally integrable univariate function h. It is this relation that provides the derivation of the geometric definition of the B-spline given in definition 3.

A multivariate version of equation (3.2) can be defined which gives rise to the geometric definition of the multivariate spline, definition 4. For any locally integrable function f in $\Re^s$,

$$\int_{\Re^s} M(x \mid x^0, \ldots, x^n) f(x)\, dx = \int_{S^n} f(\nu_0 x^0 + \cdots + \nu_n x^n)\, d\nu_1 \ldots d\nu_n.$$

The multivariate splines M which satisfy this relation are called simplex splines or simplicial splines, since they are defined by integrals over the n-dimensional simplex. Since this multivariate spline is a generalization of the Curry-Schoenberg geometric definition of the univariate spline, it is obvious that the univariate spline is a special case where s = 1, and the knots t_i are the projections of the vertices of the simplex σ onto $\Re^1$.

3.2.3 Cone Splines

The univariate spline in definition 1 is defined in terms of the truncated power function $(\cdot - x)_+^m$ where,

$$y_+^m = \begin{cases} y^m & \text{if } y > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Here x is fixed and $(t - x)_+^m$ is a function of t. Dahmen introduced the multivariate version of the truncated power function in [25] (see also [29, 27, 28, 44]). Briefly, from [27], let $x^1, \ldots, x^n \in \Re^s$ such that $vol_s([0, x^1, \ldots, x^n]) > 0$ and $0 \notin [x^1, \ldots, x^n]$. The function $T(x \mid x^1, \ldots, x^n)$ is defined by requiring that

$$\int_{\Re^s} f(x)\, T(x|x^1, \ldots, x^n)\, dx = \int_0^\infty \ldots \int_0^\infty f(\tau_1 x^1 + \cdots + \tau_n x^n)\, d\tau_1 \ldots d\tau_n \tag{3.3}$$

holds for all continuous functions f in $\Re^s$ with compact support.

This distributional definition leads to a geometric interpretation similar to definition 4. Let B be the convex polyhedral cone (i.e., a cone with a convex polyhedral cross section) defined by $u^1, \ldots, u^n \in \Re^n$ (where the u^i's are some "lifting" of the x^i's from $\Re^s$ to $\Re^n$) such that

$$vol_{n-s}(\{u \in B \mid Pu = x\}) < \infty \text{ for } x \in \Re^s, u \in \Re^n.$$

The multivariate truncated power $T_B(x)$, $x \in \Re^s$ is defined by

$$T_B(x) = vol_{n-s}(\{u \in B \mid Pu = x\}),$$

the cross-sectional (n-s)-dimensional volume of the polyhedral cone. Intuitively this is equivalent to equation (3.3) since the integration on the right side of equation (3.3) is performed over the cone defined by the positive vectors $e^1, \ldots, e^n$ in $\Re^n$.

Figure 7 shows the linear truncated power $T_B(x) = cx_+$, where c is some constant, as the cross-sectional length (one-dimensional volume) of a two-dimensional cone. Here u^1 and u^2 are points in $\Re^2$ such that $Pu^i = x_i$, and B is the resulting cone. Figure 8 shows $T_B(x) = cx_+^2$ as the cross-sectional volume of a three-dimensional cone. Here the cone B is defined by u^1, u^2, and u^3, points in $\Re^3$. The cross-sectional triangles of B provide a quadratic area function.

The multivariate truncated power function has many of the properties of the multivariate B-spline (or simplex spline). They are smooth piecewise polynomial surfaces in s dimensions with recurrence relations and derivative relations for computation [27, 44]. The multivariate simplex spline can also be formulated in terms of these truncated powers, reminiscent of the univariate truncated power formulation obtained by expanding the divided difference formulation for distinct knots. Again from [27],

$$M(x \mid x^0, \ldots, x^n) = n! \sum_{j=0}^{n} (-1)^j \, T(x-x^j \mid x^j-x^0, \ldots, x^j-x^{j-1}, x^{j+1}-x^j, \ldots, x^n-x^j). \tag{3.4}$$

Geometrically this means that any s-simplex can be represented as (boolean) combinations of infinite cones. For example, a linear univariate spline can be represented as the cross-sectional height of a triangle in the plane (as in Figure 5). Equation (3.4) states that this spline can be defined in terms of truncated powers, or equivalently that the triangle can be described in terms of two-dimensional cones.

Let $M(t \mid t_0, t_1, t_2)$ be the linear B-spline defined by the triangle T consisting of x^0, x^1, and x^2 in the top figure in Figure 9 where the x^i's are vectors in $\Re^2$ and $P(x^i) = t_i$. From equation (3.4), we have

$$\begin{aligned} M(t \mid t_0, t_1, t_2) = n! \, (\; & T(t-t_0 \mid t_1-t_0, t_2-t_0) \\ & - T(t-t_1 \mid t_1-t_0, t_2-t_1) \\ & + T(t-t_2 \mid t_2-t_0, t_2-t_1)). \end{aligned} \tag{3.5}$$

The middle figure in Figure 9 shows the three translated cones defined on the right hand side of equation (3.5). Here they are named T_0, T_1, and T_2 respectively. At the bottom the functions $T_0(t)$, $-T_1(t)$, and $T_2(t)$ are shown and $M(t)$ is shown as the dashed line.

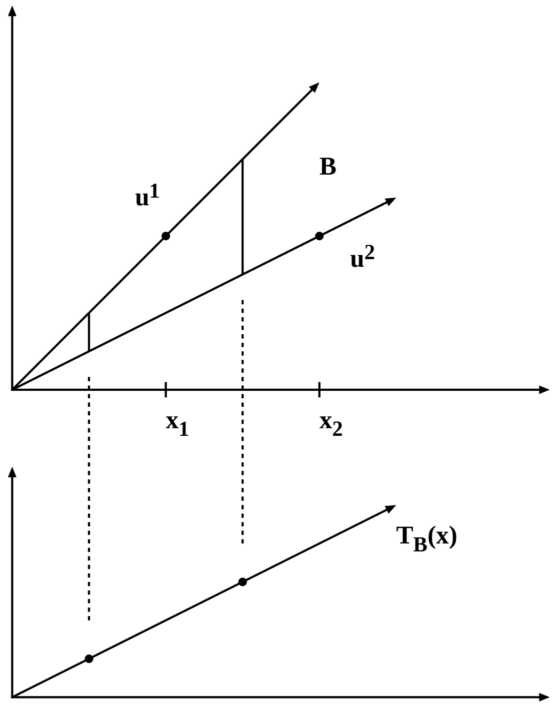

Figure 7: Multivariate truncated power for s = 1, n = 2

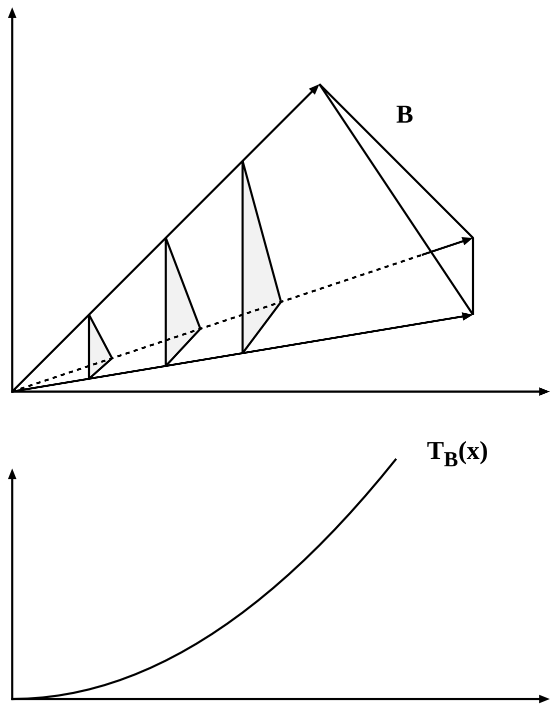

Figure 8: Multivariate truncated power for s = 1, n = 3

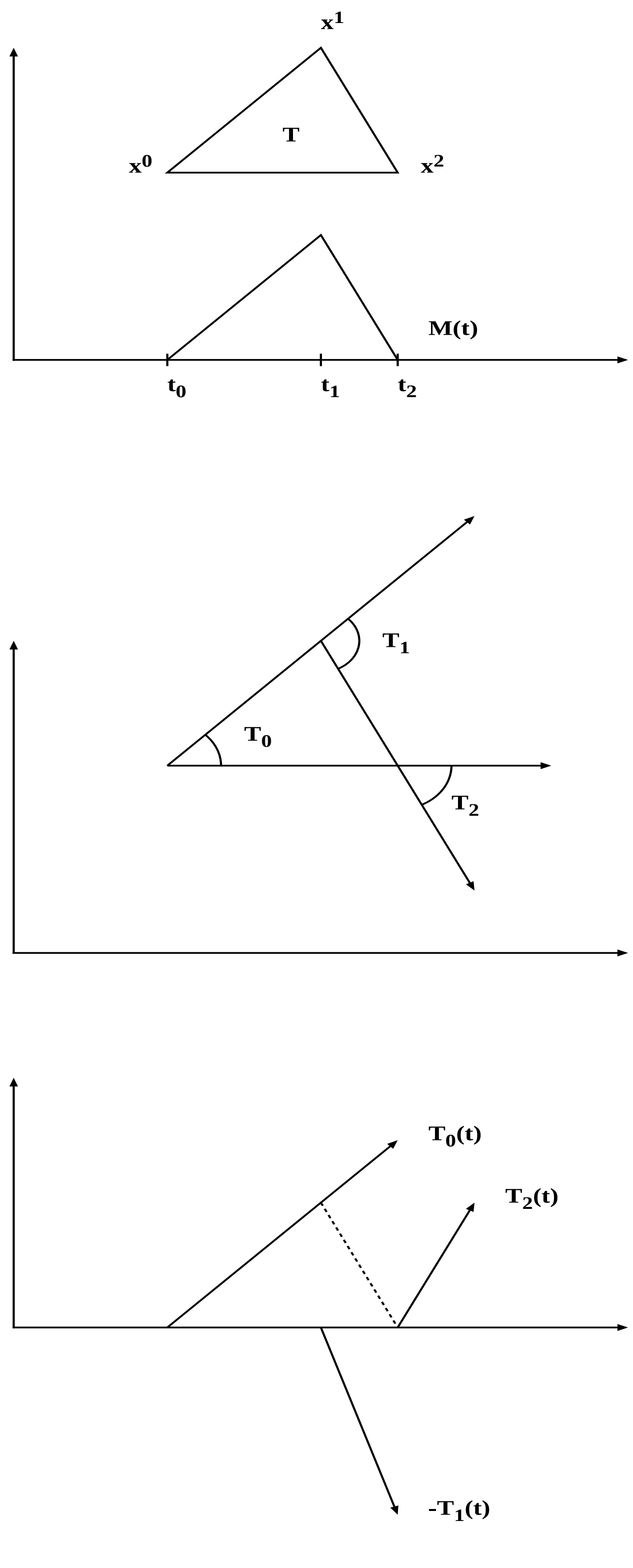

Figure 9: Linear univariate simplex spline in terms of cones

3.2.4 Box Splines

The previous sections have discussed multivariate splines derived from simplices and multivariate truncated powers defined by polyhedral cones. One might speculate that piecewise polynomial surfaces could be generated from arbitrary n-dimensional convex polyhedral bodies, and such is the case.

De Boor and Hoellig took the recurrence relations developed for simplex splines and cone splines and generalized them for any convex polyhedral body as well as showing equivalence with a distributional definition [12]. For any such body B in $\Re^n$ they note that the boundary of B consists of (n-1)-dimensional convex bodies, whose boundaries are (n-2)-dimensional convex bodies and so on. The proof of the recurrence relation for such splines is then carried out using Stoke's theorem.

The projection of an n-dimensional simplex onto a s-dimensional space results in the complete graph of the projected points. This can result in complex patterns even for simplices of low dimension. A motivation for de Boor and Hoellig's work is based on the fact that some standard finite elements can be obtained as projections of parallelepipeds, and that the support of piecewise polynomials defined using these bodies (rather than simplices) can be less complex. Thus we have a third type of multivariate spline, the box spline, which is based on the n-dimensional parallelepiped.

The box spline was introduced in [10] and further studied in [11, 9, 19]. For integers $n \geq s \geq 1$, and a sequence of vectors $X = (x^1, \ldots, x^n)$ with $x^i \in \Re^s$, the box spline $M_X(x) = M(x \mid X)$ is defined by the now familiar distributional relation

$$\int_{\Re^s} M_X(x) f(x)\, dx = \int_{\mathbf{I}^n} f(\nu_1 x^1 + \cdots + \nu_n x^n)\, d\nu_1 \ldots d\nu_n, \tag{3.6}$$

for all $f \in C_0^{\infty}(\Re^s)$. The distributional definition leads to a volume definition as in previous cases. Here the box spline is defined as the (n-s)-dimensional cross-sectional volume of an n-dimensional parallelepiped.

De Boor and Hoellig prove recurrence relations for box splines and their derivatives in [11]. Furthermore, it is shown that these box splines are piecewise polynomials of degree n-s with local support when $\langle X \rangle = \Re^s$. The support of the box spline is the orthogonal projection, or the "shadow," of the parallelepiped onto $\Re^s$. In symbols,

$$\text{supp}\, M_X = X(\mathbf{I}^n) = \{X\nu : \nu \in \mathbf{I}^n\} \tag{3.7}$$

where

$$X\nu = \nu_1 x^1 + \cdots + \nu_n x^n.$$

Box splines can be seen to have uniform univariate B-splines as a special case when s = 1, and tensor products of such univariate B-splines when s = 2 and the defining directions x^i are chosen from the set {(1,0), (0,1)}. For example, let n = 3, s = 1, and X = (1, 1, 1). Let U = $\{u^i\}$ = {(1,0,0), (1,0,1), (1,1,0)} be a "lifting" of the x^i's from $\Re^1$ to $\Re^3$. Figure 10 shows the resulting curve defined by cross-sectional areas of the parallelepiped defined by U . M_X is a uniform quadratic B-spline with knots **t** = (0, 1, 2, 3).

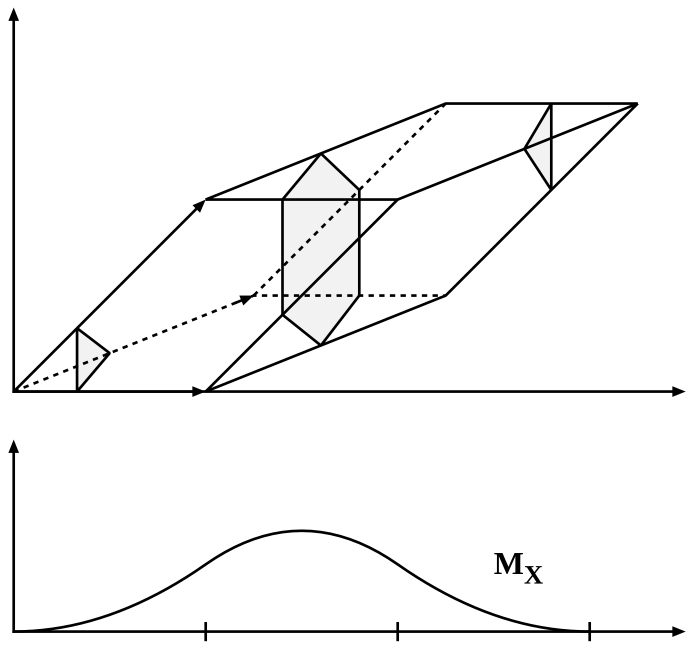

Figure 10: Uniform quadratic B-spline as a box spline

CHAPTER 4

BOX SPLINE SURFACES

4.1 Introduction

The previous chapter introduced various formulations of the multivariate spline. All of these offer intriguing possibilities for modelling more general regions; however, it is not presently known how to incorporate any of these into a general modeller. Multivariate splines provide many new flexibilities that must be understood. Under certain constraints, many of these generalizations reduce to tensor product splines or univariate splines, and any or all might be useful. It has not been determined what types of basis patterns and associated continuity conditions would be feasible for use in CAGD schemes. The generalizations also mean that the definition of "basis," functions contained in the span, continuity conditions, and domain must all be studied for appropriateness and applicability to this area.

To apply an approximation scheme for use in CAGD requires a certain computational capacity as well as theoretical properties well-suited for design and modelling purposes: A theoretically elegant representation that cannot be efficiently computed is of limited use. Assuming an approximation method has desirable properties, certain problems must be solved to use it as a representation for geometric modelling. The problems can be divided into two main categories:

1. Computational - Assuming a surface can be defined over a finite area of the parametric plane (which is not a trivial assumption), efficient algorithms for displaying surfaces on vector and raster graphics devices are needed. An algorithm for calculating basis functions is usually not sufficient because of the computational complexity of many multivariate functions and graphics algorithms.

2. Modelling - Assuming a capability for computation, certain problems must be solved to begin to use the new representation. These include the representation of nonrectangular regions and using the added flexibility of continuity conditions.

4.2 A Candidate Representation

Much of the current research effort in multivariate splines is involved with determining their theoretical properties; for example, constructing approximation spaces, determining their dimension and span, and computing approximation orders. Generalizing the univariate B-spline to multivariate spaces brings enormous new flexibilities when compared to tensor product methods. With many formulations of the multivariate spline it is a difficult problem to define a set of basis functions that form a partition of unity over a domain in the parametric plane. To develop computational algorithms or compute the raster image of a surface defined with such functions, if possible, would be premature in these cases. Frankly, the maturity of the field does not permit the application of many of these forms to geometric modelling at this point. However, recent computational results using the box spline representation have shown much promise toward the above mentioned problems [19, 11, 9, 31, 6, 5].

The box spline, as mentioned in Chapter 3, is a generalization of the uniform B-spline. It will be shown that using translations of certain uniform box spline basis functions it is relatively simple to define surfaces over various regions of the parametric plane. The uniformity of the basis functions simplifies many of the computational problems. Recently Cohen, Lyche and Riesenfeld [19] have developed a general refinement algorithm for box splines using a discrete spline method reminiscent of the Oslo Algorithm [20]. Boehm [6] has also developed algorithms for special cases. The algorithm presented in [19] is a global refinement algorithm that specializes to the Lane-Riesenfeld algorithm [43] for uniform floating univariate B-spline curves and tensor product surfaces. It will be shown that these results provide a sufficient computational base for designing algorithms to enable the box spline to be used as a geometric modelling tool.

The rest of this thesis will concentrate on this form of multivariate spline as a possible candidate for a geometric modelling tool. The trade-off in choosing this representation is between representation power and computation power. By taking a simple step into the multivariate realm the relationship between the multivariate and the tensor product is clearer, and the two are more easily compared. It will become evident throughout the thesis that even with the simplest generalization to higher dimensions, many issues and algorithms that are simple for the tensor product become much more complicated. The remainder of this chapter will present a detailed look at box spline basis functions, box spline surfaces as linear combinations of box splines, and algorithms for refining such surfaces.

4.3 Box Spline Basis Functions

In this section we continue the introduction to box spline approximation began in Chapter 3. To restate the definition, for integers $n \geq s \geq 1$ and a sequence of vectors $X = (x^1, \ldots, x^n)$ with $x^i \in \Re^s$, the box spline $M_X(x) = M(x \mid X)$ is defined as the distribution on $\Re^s$ given by the rule

$$\int_{\Re^s} M_X(x) f(x)\, dx = \int_{\mathbf{I}^n} f(\nu_1 x^1 + \cdots + \nu_n x^n) d\nu_1 \ldots d\nu_n,$$

for all $f \in C_0^\infty(\Re^s)$. Given that $\langle X \rangle = \Re^s$, a volume definition can be defined similar to that for the simplex spline. Let $P : \Re^n \rightarrow \Re^s$ be the orthogonal projection and suppose $\{y^i\}_1^n$ is a sequence of n linearly independent vectors in $\Re^n$ such that $P(y^i) = x^i$. From [11] a parallelepiped B is defined as

$$B = \{ \sum_{i=1}^{n} \nu_i y^i : \nu \in \mathbf{I}^n \}. \tag{4.1}$$

The geometric definition of the box spline can now be stated as

$$M(x \mid x^1, \ldots, x^n) = \frac{\mathrm{vol}_{n-s}\{y \in B \mid P(y) = x\}}{\mathrm{vol}_n B}.$$

The geometric definition makes some properties of the box spline evident. First of all the box spline in nonnegative everywhere and zero outside a "small" region. This region, called the support, is the "shadow" of the parallelepiped B in $\Re^n$ onto the s-dimensional space, $\Re^s$. The degree of the function is the dimension of the cross-sectional volumes, $n - s$. Also shown by de Boor and Hoellig [11] is that the function M_X has $d - 1$ continuous derivatives on $\Re^s$ (i.e., $M_X \in C^{d-1}(\Re^s)$) where

$$d = \max\{ r : \langle X \setminus Z \rangle = \Re^s \ \forall\, Z \subset X \text{ with } |Z| = r\}.$$

This is equivalent to stating that d is the largest number such that every set of $n - d$ "knot directions" spans $\Re^s$.*

4.3.1 Support of Box Spline Functions

The support of a box spline basis function is given by de Boor and Hoellig [11] as

$$\operatorname{supp} M_X = X(\mathbf{I}^n) = \{X\nu : \nu \in \mathbf{I}^n\} .$$

As an example for $n = s = 2$, let $X = \{e^1, e^2\} = \{(1,0), (0,1)\}$. In this simple example the support of M_X is just $\mathbf{I}^2$ the unit square in $\Re^2$. For $n = 3$ and $s = 2$, let $X = \{e^1, e^2, e^1 + e^2\}$. Here the support of M_X is a six-sided region as seen in Figure 11. The region is the "shadow" of a three-dimensional parallelepiped (i.e., a cube skewed by a linear transformation). The extremes of this region are the origin and the sum of the knot directions $\sum_{i=1}^{3} x^i = (2,2)$.

The piecewise polynomial nature of the box spline function is also related to the parallelepiped B. If B is considered as a wireframe representation of a parallelepiped, its projection onto $\Re^s$ gives a set of line segments, some of which form the boundaries of the support of M_X as described above. It is shown in [11] that the box spline M_X agrees with a polynomial of degree $\leq n - s$ on every connected subregion of the described projection.

Consider again the above example where $n = 3$, $s = 2$, and $X = \{e^1, e^2, e^1 + e^2\}$. The projection of the wireframe parallelepiped B based on these vectors is seen in Figure 12. It is noted that while there are an infinite number of parallelepipeds satisfying equation (4.1) where $P(y^i) = x^i$, the projection onto $\Re^s$ is unique. From the properties discussed so far we know that M_X in this case is a piecewise linear ($n - s = 1$) function comprised of six linear pieces, one over each of the triangles in Figure 12. Furthermore, these pieces are joined together with C^0

*We will call the x^i's "knot directions" since some of the intuition associated with knots in the univariate does not carry into multivariate spaces. The x^i's are knots in that they are a set of points in $\Re^s$ that completely define the basis function up to a constant. They are not knots in the sense that they describe where the polynomial pieces that make up the function are "tied" together. In that sense they can be thought of as knot *directions* describing grids of lines along which polynomial surfaces (for s = 2) are joined.

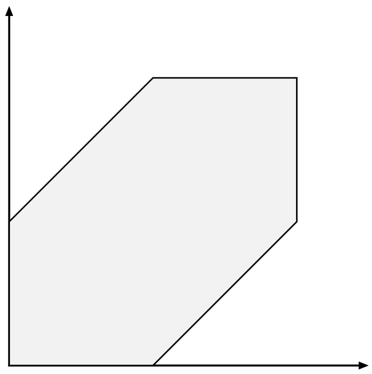

Figure 11: Support of M_X, where $X = \{e^1, e^2, e^1 + e^2\}$

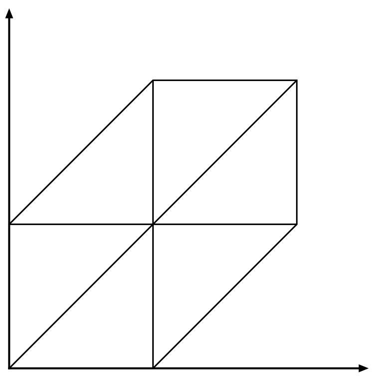

Figure 12: Projection of 3-dimensional parallelepiped

continuity (d = 1), the function is nonnegative over the six-sided region in Figure 11 and zero elsewhere. This particular box spline function is a standard linear finite element.

4.3.1.1 Algorithm. While the parallelepiped B on which the box spline function is based is a solid convex polyhedron, there is more to be gained by considering the above described wireframe parallelepiped consisting of a set of edges (line segments) in $\mathfrak{R}^n$. In subsequent references to the support of the box spline we will mean the projection of the wireframe parallelepiped and assume that it is clear that the support of the function is the convex hull of such a projection and the interior line segments represent the boundaries of the domains of individual polynomial pieces.

Given that the projection of the parallelepiped onto $\mathfrak{R}^s$ is unique, an algorithm can be defined to calculate such a projection based only on the knot directions $(x^i)_1^n$. Since the vertices of the parallelepiped in $\mathfrak{R}^n$ are sums of the defining vectors y^i, as in equation (4.1), the projections of these points are sums of the knot directions x^i. To describe a method for generating the projection of B, a "path" of length m is defined in terms of a sequence of m knot directions as the line segment from the origin to the first knot direction, the line segment from the first knot direction to the point which is the sum of the first two knot directions, and so on up to the segment from the sum of the first m – 1 knot directions to the point which is the sum of all m knot directions. To generate all the edges of the projection of the parallelepiped B, one simply needs to draw all "paths" of length n. This, however, requires drawing all n! such paths, and generates many of the edges redundantly.

A more efficient method can be defined using a recursive algorithm that is guaranteed to draw each edge of the projection once. The basic idea of the algorithm is to start with each knot direction and construct a path one segment at a time, with recursion, keeping track of the path that leads to the current point. If at any time the path leads to a redundancy, or the last point (the sum of the knot directions) the recursion terminates. A pseudocode description of the algorithm is given in the following three procedures. First, a top level procedure simply starts off the recursion by calling the "DrawSegs" procedure for each knot direction. The "DrawSegs" procedure is the recursive procedure which draws the segments. Finally, the "BeenThere" procedure determines whether the recursion needs to continue. This algorithm relies on the fact that the knot directions are considered in order. Every edge incident on a point in the projection is drawn at the first time possible in the recursion and is not repeated.

```
procedure DrawProjection( X );
begin
  % For each knot direction.
  for i := 1 to n do
    DrawSegs( X, Origin, list(i), 0 );
end

procedure DrawSegs( X, Start, ISet, Level);
begin
  if Level = 0 then             % If first Level.
  {
    % Draw a segment from the origin to xj.
    j := first( ISet );
    MoveTo( Start );    LineTo( xj );

    DrawSegs( X, Start + xj, ISet, Level + 1 );
  }
  else if Level < n - 1 then % Not first or last level.
  {
    % For each knot direction not already used.
    foreach xj where j is not in ISet do
    {
      MoveTo( Start );    LineTo( Start + xj );

      % Check to see that we have not already
      % been to the point we just drew.
      if not BeenThere(j, ISet) then
        DrawSegs( X, Start + xj, append( j, ISet),
                  Level + 1);
    }
  }
  else  % The last level.
  {
    % Draw from Start to the sum of the knot directions.
    MoveTo( Start );    LineTo( KnotSum );
  }
end;
```

```
procedure BeenThere( I, ISet );
begin
  foreach index in ISet do
  {
    % If any index in the set is greater than I,
    % we have been here before (Deja Vu).
    if I < index then
      return TRUE;
  }
  return FALSE;
end
```

Using an implementation of the above algorithm, the following figures where produced for s = 2 and various sets of knot directions. The support of the linear box spline M_X, where X = $\{e^1, e^2, e^1 + e^2\}$, was previously shown in Figure 12. The support of a slightly more general linear box spline is shown in Figure 13, where X = {(1,0), (0,1.5), (2,1)}. In this figure it is more obvious that the projection is that of a skewed three-dimensional cube.

Increasing the number of knot directions to four, the box spline becomes a piecewise quadratic function, and the support can become very complex. In Figure 14 the support of M_X is shown where X = $\{e^1, e^1, e^2, e^2\}$. This is a very special case where the parallelepiped B is a four-dimensional hypercube that is projected straight onto the xy plane producing a simple pattern. M_X in this example is the uniform tensor product bilinear function. A quadratic finite element function can be produced with X = $\{e^1, e^2, e^1 + e^2, e^1 - e^2\}$ as seen in Figure 15. Here the choice of the knot directions again produces a regular pattern, but it is more complex than the bilinear case. In the general case, the projection of the four-dimensional parallelepiped consists of 32 edges. In Figure 16, for X = {(1,0), (0,1.5), (2,1), (1.5,-1)} the support of M_X is shown. In this projection, all 32 edges are distinct.

4.3.2 Computing Box Splines

As with the univariate B-spline, the multivariate spline can be calculated with a recurrence relation. De Boor and Hoellig [11] present the following recurrence relation for the box spline.

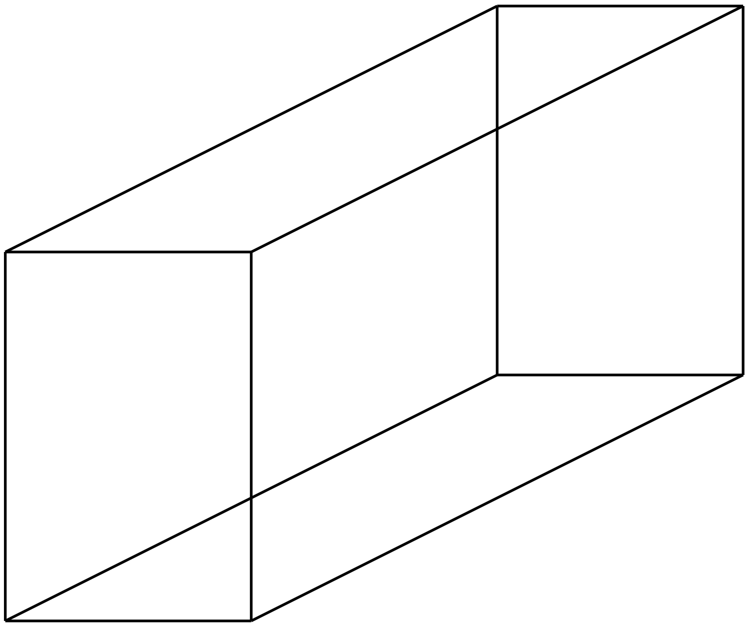

Figure 13: Support of M_X, where X = {(1,0), (0,1.5), (2,1)}

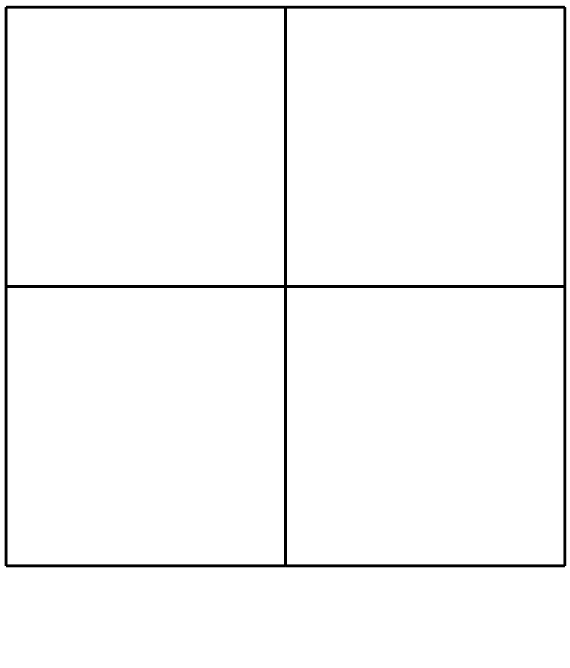

Figure 14: Support of M_X, where X = {e^1, e^1, e^2, e^2}

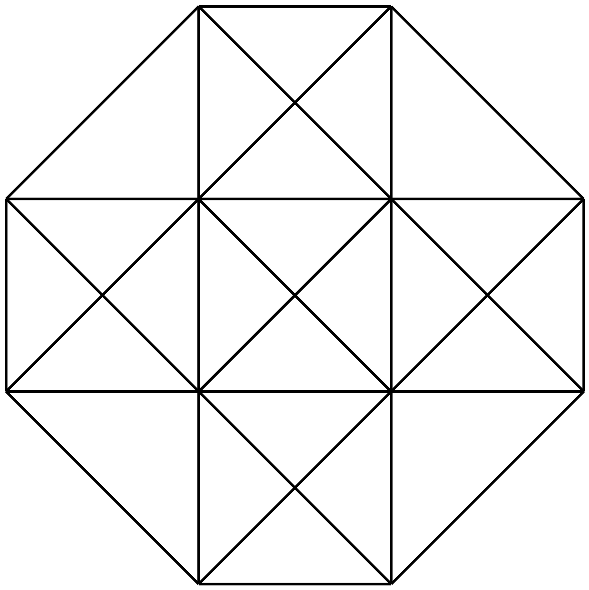

Figure 15: Support of M_X, where $X = \{e^1, e^2, e^1 + e^2, e^1 - e^2\}$

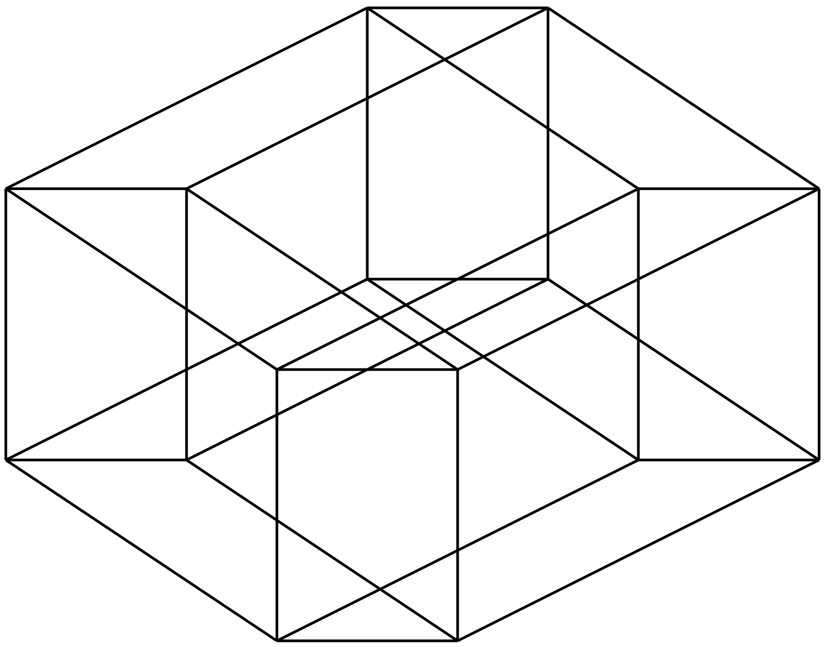

Figure 16: Support of M_X, where $X = \{(1,0), (0,1.5), (2,1), (1.5,-1)\}$

$$(n - s)\, M_X(x) = \sum_{q=1}^{n} [\lambda_q M_{X_q}(x) + (1 - \lambda_q) M_{X_q}(x - x^q)], \quad x = \sum_{q=1}^{n} \lambda_q x^q$$

where

$$X_q = X \setminus \{x^q\} = \{x^1, \ldots, x^{q-1}, x^{q+1}, \ldots, x^n\}\ .$$

There are two aspects of this multivariate formulation that are computationally complex compared to the univariate recursive de Boor-Cox algorithm. First, the point of evaluation, x, must be written as a linear combination of the knot directions. This is true in the univariate case, but trivial. In the multivariate case, if n > s the solution to the set of equations $x = \Sigma_{q=1}^{n} \lambda_q x^q$ is nonunique. One must find an appropriate way to choose the λ_q's or a method of appropriately choosing s linearly independent knot directions and then finding the corresponding λ_q's. This must be done at each level of the recursion also.

The second difficulty is found in the terminal level of the recursion. In the univariate case the lowest level of the recursion (the constant function) is computationally equivalent to testing whether a scalar is in an interval. For bivariate box splines (s = 2), the constant functions are equivalent to deciding whether a point in inside a polygon; a computationally expensive procedure. The number of constant evaluations grows exponentially and in the worst case the number of constant evaluations is $n!2^n$. The boundary conditions for the constant functions are also difficult since these functions have discontinuities along the borders of their support and a consistent convention for evaluating along the borders must be adopted.

4.3.3 Examples

To finish this section, some example box spline basis functions will be examined to provide some intuition into the various properties that have been discussed.

4.3.3.1 Linear finite element. The linear finite element as a box spline has previously been mentioned. Figure 18 again shows the support of this function and in Figure 17 the surface itself is shown with a set of coordinate axes for orientation. From the figures it is easy to see that the function is piecewise linear, that it is C^0, and that it consists of six nonzero pieces, one over each triangle in its support.

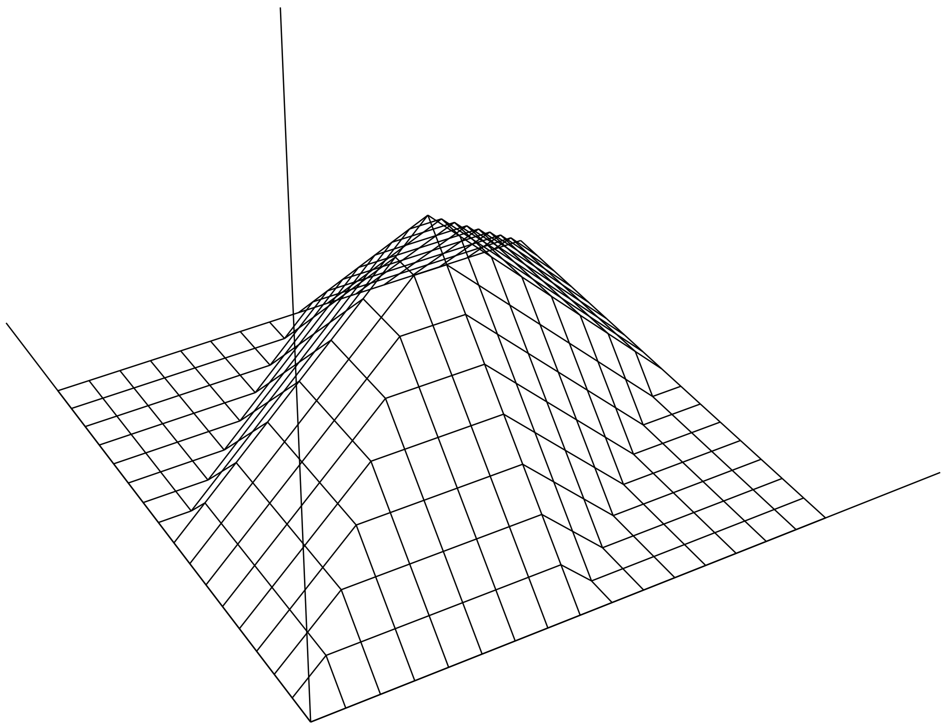

Figure 17: Linear finite element

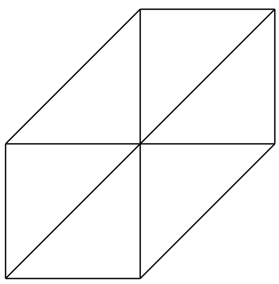

Figure 18: Support of linear box spline

4.3.3.2 A C^0 quadratic. By making the knot direction $e^1 + e^2$ multiple, an interesting quadratic basis function is defined. M_X where $X = \{e^1, e^2, e^1 + e^2, e^1 + e^2\}$ is shown in Figure 19 and its support in Figure 20. This function is a piecewise quadratic surface, consisting of ten triangular polynomial pieces. In Figure 19 a tangent discontinuity can be seen in the diagonal direction which is caused by the multiple knot direction. The narrow six-sided region of the support is also evident.

4.3.3.3 A C^1 cubic. The knot direction sequence $X = \{e^1, e^2, e^1, e^2, e^1 + e^2\}$ defines a piecewise cubic function that is tangent continuous everywhere. This function is shown in Figure 21 and its support in Figure 22. More careful observation, in this case, is necessary to see that the support of the function is six-sided, and the function is only symmetric about the line $x = y$.

4.4 Box Spline Surfaces

The discussion of box spline functions so far has only been concerned with individual basis functions. In this section we will explore spaces of piecewise polynomial functions defined by linear combinations of box splines.

4.4.1 Simplifying Assumptions

The traditional parametric representation of a tensor product B-spline surface is the linear combination of a rectangular mesh of three-dimensional points as seen in Chapter 1.

$$S(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} P_{i,j}\, N_i(u)\, N_j(v), \qquad \text{for } P_{i,j} \in \Re^3.$$

Surfaces formed from such tensor product bases provide convex combinations of the mesh over finite rectangular regions of the parametric plane and the sizes of the mesh, n and m, are directly related to the pair of knot vectors that define the basis functions.

To define a box spline surface, sets of box spline basis functions must first be defined using translations (since there is only *one* basis function defined by a set of knot directions), and further, these sets must define a basis for a set of piecewise polynomials over a finite, possibly nonrectangular, region of the plane. This is much more complicated than the tensor

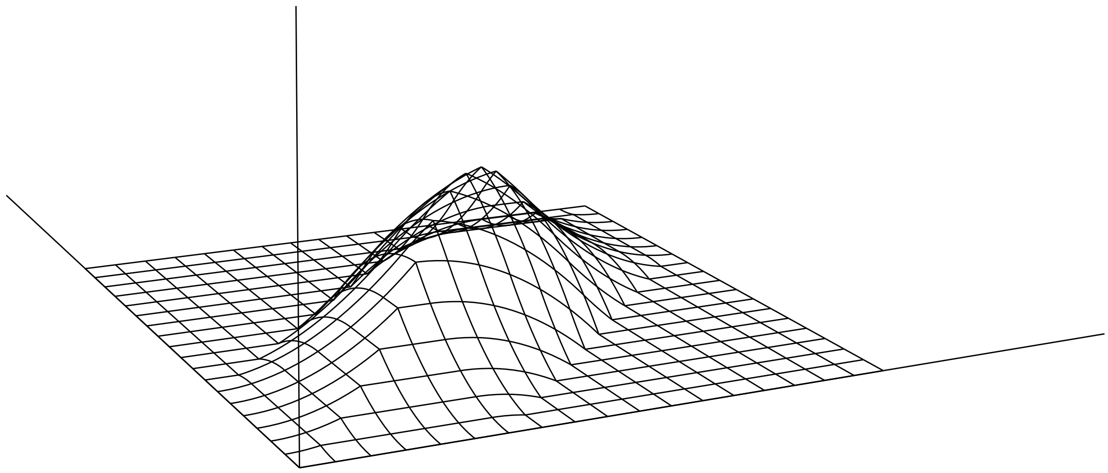

Figure 19: C^0 quadratic box spline

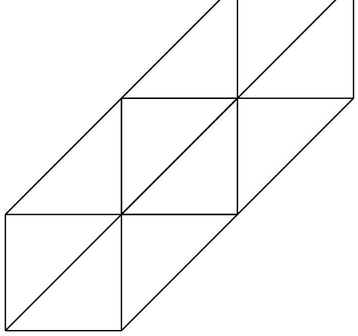

Figure 20: Support of quadratic box spline

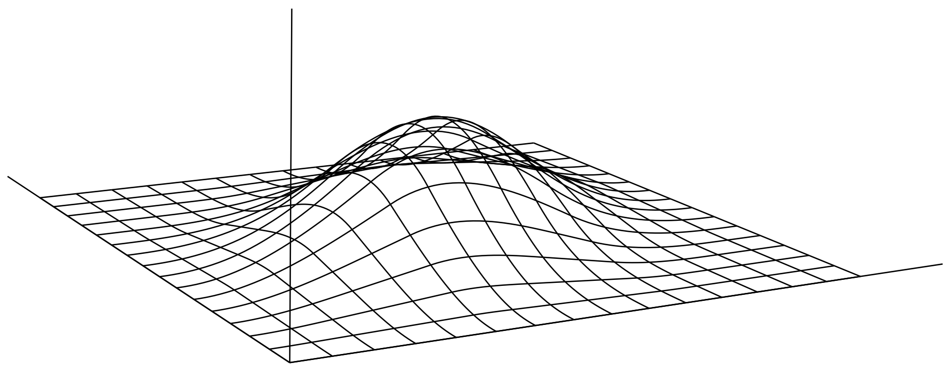

Figure 21: C^1 cubic box spline

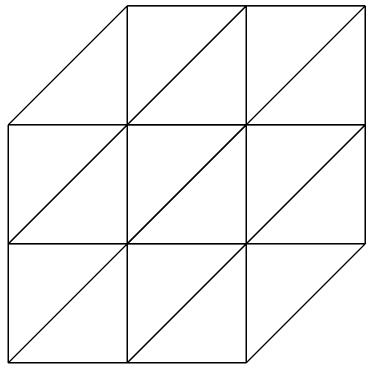

Figure 22: Support of cubic box spline

product case since the support of each basis function is not necessarily rectangular. Although this problem is simplified by the uniform nature of the box spline, when considering box splines in their full generality it is still difficult to define box spline surfaces with the desired properties. In view of this, it is a choice of this thesis to restrict the generality of the box spline to certain types of spaces which have previously been studied by researchers in approximation theory, and for which certain properties are known.

First of all, since the main goal of the thesis is modelling three-dimensional objects it will subsequently be assumed that s = 2 unless otherwise indicated. The defining sequence or set of knot directions will first be restricted to be the multi-integers $\mathbf{J} = \mathbf{Z}^2$ and the box spline M_Z is defined by $Z = (z^1, \ldots, z^n)$ for $z^i \in \mathbf{J}$. Translates of M_Z are defined as follows:

$$B_j(x) = M_Z(x - j), \qquad \text{for} \quad x, j \in \mathbf{J}. \tag{4.2}$$

By further restricting the domain of the knot directions z^i, the support of the functions B_j form certain regular grid patterns in the plane. Specifically, if the z^i's are chosen from the set $\{d^1,d^2,d^3\}$ where $d^1 = e^1$, $d^2 = e^2$, and $d^3 = e^1 + e^2$, what de Boor and Hoellig [9] call a three-direction mesh (or grid) Δ is formed. This grid is a unit square mesh with one diagonal. The special case of uniform tensor product B-splines are obtained when the z^i's are chosen from the set $\{d^1,d^2\}$. This is the two-direction unit square grid.

To further simplify the notation, integers r, s, and t are introduced with the following definitions.

$$\begin{aligned} r &= |\{i : z^i = d^1\}|, && \text{the number of } d^1\text{'s in } Z \\ s &= |\{i : z^i = d^2\}|, && \text{the number of } d^2\text{'s in } Z \\ t &= |\{i : z^i = d^3\}|, && \text{the number of } d^3\text{'s in } Z\,. \end{aligned}$$

Since the box spline M_Z on the three-direction grid is completely characterized by these three integers, $M_{r,s,t}$ will be considered synonymous with M_Z.

The box spline $M_{r,s,t}$ is a piecewise polynomial of degree $r + s + t - 2$ since $n = r + s + t$. The continuity of $M_{r,s,t}$ also depends on the multiplicities r, s, and t. It is shown in [9] that

$$d = (n - \max\{r,s,t\}) - 1 \ .$$

Another very important property, shown by de Boor and Hoellig in [11], is that the sum of all the translates of $M_{r,s,t}$ is one for every point in $\Re^2$.*

$$\sum_{j \in \mathbf{J}} B_j(x) = 1, \quad x \in \Re^2$$

A space of explicit piecewise polynomials S can be defined on the three direction grid Δ as linear combinations of the box splines B_j.

$$S = \{\sum_j P_j B_j : j \in \mathbf{J}, P_j \in \Re\}$$

S is a space of piecewise polynomial functions of degree $\leq n - 2$ on Δ and is a subspace of the set of all piecewise polynomial functions of degree $k = n - 2$, with continuity $\rho = d - 1$, i.e.,

$$S \subset \pi^{\rho}_{k,\Delta} \ .$$

It is further shown in [11] that for $Z \subset \mathbf{J}$ the set of box splines $\{B_j\}$ are linearly independent if for every subset Z_b of Z which spans $\Re^2$, $|\det Z_b| = 1$. For the three-direction grid this is always true. This means the B_j's form a unique basis for S.

Although it may seem a severe restriction to limit the box splines to the three-direction grid, these are not simple spaces of functions. Box splines, and piecewise polynomial functions in general, over regular grid patterns have received much attention [1, 17, 14, 15, 16, 9, 6, 5]. In describing the above space of piecewise polynomials S, de Boor and Hoellig [9] comment that "even in this simple setting, we find much challenge."

---

*This is true for any M_Z, where $(z^i)_1^n \in \mathbf{J}$ and $\langle Z \rangle = \Re^2$

4.4.2 The Support of Collections of Box Splines

To summarize, a number of facts about the box splines $M_{r,s,t}$ and their translates B_j have been presented. These properties are:

- $M_{r,s,t}$ is a piecewise polynomial of degree n – 2 with d – 1 continuous derivatives.
- $M_{r,s,t}$ is nonnegative everywhere, and zero outside its support.
- The support of $M_{r,s,t}$ is a four or six sided region composed of triangles from the grid Δ.
- The set of all B_j for $j \in \mathbf{J}$ form a partition of unity for every point in $\Re^2$.
- A space of piecewise polynomials *S* can be defined in terms of the box splines B_j.
- The box splines B_j form a basis for *S*.

We will now consider the definition of box spline surfaces as linear combinations of finite numbers of box splines to provide a method of surface representation that captures these desirable properties. The problem is to find a finite set $I \subset \mathbf{J}$ such that $\sum_{j \in I} B_j = 1$ over some finite region Ω, where Ω is a subpartition of the grid Δ. This is equivalent to finding the set of B_j's that are nonzero on some portion of the interior of Ω. First of all, a more precise definition of Δ is necessary.

> **Definition 1:** The three-direction grid Δ is defined as a set of triangular regions formed by the union,
>
> $$\Delta = U \cup L$$
>
> where
>
> $$L = \{ \tau_j^1 : j \in \mathbf{J} \} \text{ and } U = \{ \tau_j^2 : j \in \mathbf{J} \}$$
>
> where
>
> $$\tau_j^1 = [\{j, j + d^1, j + d^3\}] \text{ and } \tau_j^2 = [\{j, j + d^2, j + d^3\}].$$

The grid Δ divides $\Re^2$ into triangular regions of two types. The τ_j^1's will be called "type-1" triangles and the τ_j^2's will be called "type-2" triangles.

It is useful to look at the support of $M_{r,s,t}$ in terms of the τ's. From [9] the support of M can be written in terms of the three knot directions as follows:

$$\text{supp } M_{r,s,t} = \{\sum_{i=1}^{3} \lambda_i d^i : \lambda \in [0,r] \times [0,s] \times [0,t]\} . \tag{4.3}$$

From this it can be determined that the support of $M_{r,s,t}$ is, in general, a hexagon consisting of N type-1 triangles and N type-2 triangles, where N = rs + rt + st. This means that a given B_j is nonzero over at most N type-1 and N type-2 triangles in Δ and therefore given a triangle $\tau \in \Delta$ there are at most N B_j's that are nonzero over τ and these B_j's must sum to one over τ.

We will now consider the problem of finding the set I_τ defined as follows:

$$I_\tau = \{ j : \text{supp } B_j \cap \tau^\circ \neq \emptyset\},$$

where τ° is the interior of the triangle τ. To do this some sets of multi-integers are first defined.

Definition 2: Given r, s, and t, let $z^* \in \mathbf{J}$ be defined as $z^* = \sum_{i=1}^{n} z^i = rd^1 + sd^2 + td^3$ = (r+t,s+t). The set $R_{r,s,t}$ is defined as the set of nonnegative (coordinatewise) multi-integers j less than z^*,

$$R_{r,s,t} = \{ j : \mathbf{0} \le j < z^* \}.$$

Another useful set of points in **J** is the set of multi-integers contained by the support of the box spline $M_{r,s,t}$.

Definition 3: For r,s,t ≥ 1, the set $S_{r,s,t}$ is defined as

$$S_{r,s,t} = \{ j \in \mathbf{J} : j \in \text{supp } M_{r,s,t} \}.$$

This set can be enumerated as the union of four mutually disjoint sets as follows:

$$\begin{aligned} S_{r,s,t} = \; & \{ j \in \mathbf{J} : \mathbf{0} \le j \le (r,s) \} \\ & \cup \{ j \in \mathbf{J} : j = sd^2 + \alpha d^1 + \beta d_3,\ \alpha = 0, \ldots, r\text{-}1,\ \beta = 1, \ldots, t \} \\ & \cup \{ j \in \mathbf{J} : j = rd^1 + \alpha d^2 + \beta d_3,\ \alpha = 0, \ldots, s\text{-}1,\ \beta = 1, \ldots, t \} \\ & \cup \{ j \in \mathbf{J} : j = rd^1 + sd^2 + \beta d_3,\ \beta = 1, \ldots, t \}. \end{aligned}$$

The cardinality of $S_{r,s,t}$ is easily calculated from the above enumeration as

$$\begin{aligned}|S_{r,s,t}| &= (r + 1)(s + 1) + rt + st + t \\ &= (r + 1)(s + 1) + t(r + s + 1).\end{aligned}$$

Using these sets, the following proof is offered for determining the set of basis functions which are nonzero on τ.

Theorem 4: Given the infinite set of box splines $B_j = M_{r,s,t}(\cdot - j)$ for $r,s,t \geq 1$, define $T_1 = R_{r,s,t} \cap S_{r,s-1,t}$. If τ^* is the type-1 triangle, $\tau^* = [\{z^* - d^3, z^* - d^2, z^*\}]$, then the set $I_{\tau^*} = T_1$.

Proof: The outline of the proof is to first examine the set T_1 and show that it has the correct number of elements. Then it is shown that for each element j in the set, the support of the box spline B_j contains the triangle τ^*.

The set T_1 can be enumerated by simply applying the constraint $\mathbf{0} \leq j < z^*$ to the set $S_{r,s-1,t}$. Therefore,

$$\begin{aligned}T_1 = \quad & \{ j : \mathbf{0} \leq j \leq (r, s-1) \} \\ & \cup \{ j : j = (s-1)d^2 + \alpha d^1 + \beta d_3, \;\; \alpha = 0, \ldots, r\text{-}1, \beta = 1, \ldots, t \} \\ & \cup \{ j : j = rd^1 + \alpha d^2 + \beta d_3, \;\; \alpha = 0, \ldots, s\text{-}2, \beta = 1, \ldots, t\text{-}1 \} \\ & \cup \{ j : j = rd^1 + (s-1)d^2 + \beta d_3, \;\; \beta = 1, \ldots, t\text{-}1 \} .\end{aligned}$$

Again, these are mutually disjoint sets and the number of elements in this set is easily computed.

$$\begin{aligned}|T_1| &= (r + 1)s + rt + (s - 1)(t - 1) + (t - 1) \\ &= rs + s + tr + st - t - s + 1 + t - 1 \\ &= rs + rt + st = N.\end{aligned}$$

This shows that T_1 has the correct number of elements as previously derived from equation (4.3). From equations (4.2) and (4.3) the support of B_j is

$$\text{supp } B_j = \{\sum_{i=1}^{3} [\lambda_i d^i] + j : \lambda \in [0,r] \times [0,s] \times [0,t]\} . \tag{4.4}$$

We now show that for every $j \in T_1$, supp B_j contains the points z^*, $z^* - d^2$, and $z^* - d^3$, and therefore contains the triangle τ^*. Every $j \in T_1$ is also in $S_{r,s\text{-}1,t}$ and, therefore, also a member of supp $M_{r,s\text{-}1,t}$. Therefore,

$$\exists\, \lambda \in \{0,1, \ldots ,r\} \times \{0,1, \ldots ,s\text{-}1\} \times \{0,1, \ldots ,t\} \ni$$
$$j = \sum_{i=1}^{3} \lambda_i d^i.$$

The λ_i's can be restricted to be integers because of the choice of the d^i's. Now,

$$z^* - j = (r - \lambda_1)d^1 + (s - \lambda_2)d^2 + (t - \lambda_3)d^3.$$

Since $\lambda \in [0,r] \times [0,s\text{-}1] \times [0,t]$, if we let $\Gamma = (r - \lambda_1,\ s - \lambda_2,\ t - \lambda_3)$, Γ must be in $[0,r] \times [1,s] \times [0,t]$ and

$$z^* = j + \sum_{i=1}^{3} \Gamma_i d^i$$
$$\in \text{supp } B_j.$$

Writing $z^* - d^2 - j$ in terms of r, s, t, and λ, we have

$$z^* - d^2 - j = (r - \lambda_1)d^1 + (s - 1 - \lambda_2)d^2 + (t - \lambda_3)d^3.$$

Now if we let $\Gamma = (r - \lambda_1,\ s - 1 - \lambda_2,\ t - \lambda_3)$, then $\Gamma \in [0,r] \times [0,s\text{-}1] \times [0,t]$ and

$$z^* - d^2 = j + \sum_{i=1}^{3} \Gamma_i d^i$$
$$\in \text{supp } B_j.$$

Finally, consider the point $z^* - d^3$. Writing $z^* - d^3 - j$ in terms of r, s, t, and λ we have

$$z^* - d^3 - j = (r - 1 - \lambda_1)d^1 + (s - 1 - \lambda_2)d^2 + (t - \lambda_3)d^3 .$$

So far we have only used the fact that $j \in M_{r,s\text{-}1,t}$. In this case we must introduce the fact that j is also in $R_{r,s,t}$. Since $j = \sum_{i=1}^{3} \lambda_i d^i = (\lambda_1 + \lambda_3, \lambda_2 + \lambda_3)$, j can be restricted to be $\mathbf{0} \le j < z^*$ by restricting $\lambda \in \{0,1, \ldots, r\text{-}1\} \times \{0,1, \ldots, s\text{-}1\} \times \{0,1, \ldots, t\}$. Now if $\Gamma = (r - 1 - \lambda_1, s - 1 - \lambda_2, t - \lambda_3)$, then $\Gamma \in [0,r\text{-}1] \times [0,s\text{-}1] \times [0,t]$ and

$$\begin{aligned} z^* - d^3 &= j + \sum_{i=1}^{3} \Gamma_i d^i \\ &\in \text{supp } B_j. \end{aligned}$$

Since for every $j \in T_1$, supp B_j contains each of the three points that define τ^*, supp B_j also contains τ^* itself. Since there are exactly N such j's, $I_{\tau^*} = T_1$. □

Corollary 5: By symmetry it is argued that if $T_2 = R_{r,s,t} \cap S_{r\text{-}1,s,t}$ and τ^* is the type-2 triangle $\tau^* = [\{z^* - d^3, z^* - d_1, z^*\}]$, then $I_{\tau^*} = T_2$. □

Corollary 6: By translation, if τ is the generalized type-1 triangle with upper right corner $z^* + j$, for any $j \in \mathbf{J}$, then $I_\tau = \{ k + j : k \in T_1 \}$. Similarly, if τ is the generalized type-2 triangle with upper right corner $z^* + j$, for any $j \in \mathbf{J}$, then $I_\tau = \{ k + j : k \in T_2 \}$. □

To generalize the above arguments to more arbitrary regions, let Ω be a convex region of the plane composed of a finite union of τ's in Δ. The set I_Ω, defined as

$$I_\Omega = \{ j : \text{supp } B_j \cap \Omega^\circ \ne \emptyset \},$$

where Ω° is the interior of Ω, is simply the union of all the sets I_τ for each τ in Ω:

$$I_\Omega = \cup_{\tau \in \Omega} I_\tau.$$

Some examples will illustrate the above results. For the linear finite element $M_{1,1,1}$, the sum of the knot directions $z^* = (2,2)$. For $\tau = [(2,2),(2,1),(1,1)]$ and $S_{1,0,1} = \{(0,0), (1,0), (1,1), (2,1)\}$ the set $I_\tau = \{(0,0), (1,0), (1,1)\}$. Figure 23 shows τ as the shaded region and the set I_τ as the dots. For the C^1 cubic box spline $M_{2,2,1}$ and the triangle $\tau = [(3,3), (3,2), (2,2)]$ the set I_τ consists of eight multi-integers in the pattern shown in Figure 24. An interesting situation occurs when the C^0 quadratic $M_{1,1,2}$ is used with a rectangular domain. A square domain, Ω, consisting of just two τ's is shown in Figure 25 along with the set I_Ω. Although the domain is rectangular the set I_Ω is not is this case. This means that a rectangular surface could be constructed over this domain although the mesh would have certain "holes" in it.

We have now arrived at the point where the box spline surface can be defined over a finite region of the parametric plane.

Definition 7: Given the domain Ω, and the knot direction sequence Z, characterized by r, s, and t, the box spline surface S is a bivariate parametric function from $\Re^2 \to \Re^3$ defined as

$$S(u,v) = \sum_{j \in I_\Omega} P_j B_j(u,v), \quad P_j \in \Re^3.$$

The set of points $\{P_j\}$ is called the control mesh for the surface, although it is not necessarily a rectangular array of points. For the three-direction grid, the domain Ω permits surfaces defined over regions with three, four, five, or six sides, as well as providing a third direction of possible derivative discontinuity that tensor product surfaces do not provide. The uniform floating tensor product surfaces are a special case where t = 0. The surface S is a piecewise polynomial of degree $n - s$ with $d - 1$ continuous derivatives which exhibits local control and lies within the convex hull of the mesh.

4.5 Refinement

Recurrence relations have been developed for evaluation of the various forms of the multivariate spline [44, 12, 11, 9, 29] and a recurrence relation for box splines has been previously discussed, but the computational cost of such algorithms make their use for graphical display algorithms prohibitive. Refinement algorithms such as the Oslo algorithm [20], a general refinement algorithm for nonuniform B-splines, have proven to be useful in these applications for univariate and tensor product B-splines. Recently, the first such general refinement algorithms have been developed for a multivariate spline representation. The discrete spline method presented by Cohen, Lyche and Riesenfeld in [19] is reminiscent of the Oslo Algorithm. Other algorithms defined in terms of point averaging have been developed by Boehm [6, 5], Prautzsch [47, 48], and Dahmen and Micchelli [30, 31].

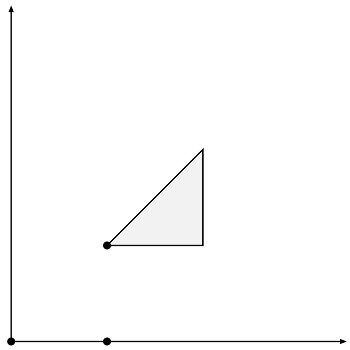

Figure 23: Linear box splines over a triangle

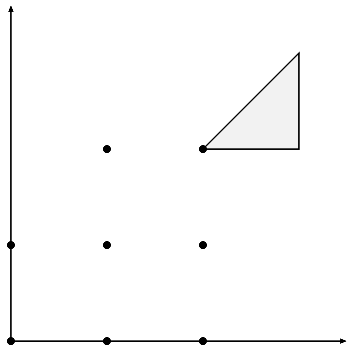

Figure 24: Cubic box splines over a triangle

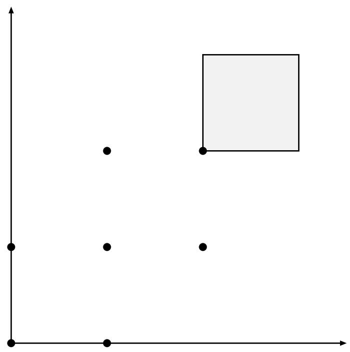

Figure 25: Quadratic box splines over a square

4.5.1 Knot Line Refinement of Box Splines

The refinement problem can be stated as follows: Given a surface Γ,

$$\Gamma = \sum_i P_i B_i, \quad i \in \mathbf{J} \tag{4.5}$$

find a new control mesh $\{d_j\}$ such that

$$\Gamma = \sum_j d_j N_j, \quad \text{where} \quad N_j(x) = M_Z(mx - j), \quad j \in \mathbf{J}, x \in \Re^2, \tag{4.6}$$

for some positive integer m. The new functions N_j are scaled versions of the box splines B_i on a new finer grid.

This problem can be solved with the introduction of the discrete box spline. The discrete box spline was introduced in [19] as a discretization of the distributional definition of the continuous box spline, equation (3.6). Given a set of multi-integers $Z = (z^1, \ldots, z^n)$, $z^q \in \mathbf{J}$ and a positive integer m, the function $\beta(j) = \beta(j \mid Z, m) : \mathbf{J} \rightarrow \Re$, must satisfy the equation

$$m^n \sum_j f(j)\beta(j) = \sum_{\nu \in N_m^n} f(Z\nu), \quad \text{where} \quad Z\nu = \sum_{q=1}^{n} \nu_q z^q,$$

for all discrete functions $f : \mathbf{J} \rightarrow \Re$ with a finite number of nonzero values, where the set of n-dimensional multi-integers N_m^n is defined as follows:

$$N_m^n = \{ (\nu_1, \ldots, \nu_n) : \nu_i \in \{0, 1, \ldots, m-1\}, i = 1, \ldots, n\}.$$

By fixing j and letting $f(i) = \delta_{ij}$, β is explicitly defined in [19] as:

$$\beta(j \mid Z,m) = \frac{1}{m^n} |\{ \nu \in N_m^n : Z\nu = j \}| . \tag{4.7}$$

The discrete box spline can also be computed with the recurrence relation presented in [19]. For integers $n \geq 2$, $s \geq 1$, and $q = 1, 2, \ldots, n$,

$$\beta(j \mid Z,m) = \frac{1}{m}\sum_{k=0}^{m-1} \beta(j - kz^q \mid Z_q,m) \quad \text{where } Z_q = Z \setminus \{z^q\}. \tag{4.8}$$

It is shown in [19] that if $|\det(z^1, \ldots ,z^s)| = 1$, the continuous box spline B_i can be written in terms of the discrete box splines β, and the continuous box splines N_j on the finer partition. Specifically,

$$B_i = \sum_j \alpha_{i,n}(j)N_j,$$

where, for $r = s, s + 1, \ldots ,n$

$$\alpha_{i,r}(j) = m^s\beta(j - mi \mid z^1, \ldots ,z^r,m),\ i, j \in \mathbf{J}.$$

It is further shown in [19] that

$$\Gamma = \sum_j d_jN_j = \sum_i P_iB_i, \qquad \text{where } d_j = d_j^n \text{ and}$$

$$d_j^r = \sum_i \alpha_{i,r}(j)P_i, \quad r = s, s + 1, \ldots ,n, \quad j \in \mathbf{J}. \tag{4.9}$$

4.5.2 Refinement Algorithms

Cohen et al. [19] present the following recursive definition for d_j^r which is the basis for a refinement algorithm.

$$d_j^r = \frac{1}{m}\sum_{k=0}^{m-1} d_{j-kz^r}^{r-1}, \quad r = s + 1, \ldots ,n$$

where

$$d_j^s = \begin{cases} P_i & \text{if } \exists\,(\nu_1, \ldots ,\nu_s) \in N_m^s, \quad j = mi + \nu_1z^1 + \cdots + \nu_sz^s \\ 0 & \text{otherwise.} \end{cases}$$

Using this definition a simple algorithm for refining a mesh with a factor m is stated which uses point averaging.

1. Create an array d of points with m^s copies of each P_i such that $d_j = P_i$ for $j = mi + \nu_1z^1 + \cdots + \nu_sz^s$, for all $\nu \in N_m^s$.

2. For each direction z^r, $r = s + 1, \ldots, n$ take an average of m d_j^{r-1}'s to form d_j^r.

This algorithm specializes to the Lane-Riesenfeld algorithm [43] for uniform tensor product surfaces when $m = 2$ and $t = 0$, and, for any m, is equivalent to inserting $m - 1$ equally spaced knots in each interval using the Oslo algorithm [20]. It is noted that the above results are stated for all $j \in \mathbf{J}$. The algorithm can be specialized to a finite domain Ω by summing over i,j $\in I_\Omega$ in equations (4.5) and (4.6).

The above algorithm can be made computationally more efficient in the number of arithmetic operations by combining all the averaging operations into a one pass algorithm. Consider a point $d_j = d_j^n$ that is the result of the point averaging algorithm. The resulting point in the refined mesh is simply a linear combination of certain original mesh points, i.e., $d_j^n = \sum \alpha_{i,n}(j)P_i$ from equation (4.9). Since for a fixed j there are only a finite number of nonzero $\alpha_{i,n}(j)$'s the above equation is a discrete linear filtering of the mesh $\{P_i\}$ by the $\alpha_{i,n}(j)$'s. The $\alpha_{i,n}(j)$'s are, however, dependent on j and the algorithm could be made more efficient if a constant windowing function were used. This can be done using the first step in the original refinement algorithm. Consider the enlarged mesh $\{Q_j\}$ where there are m^s copies of each P_i, i.e.,

$$Q_j = P_i, \quad \text{for } j = mi + \nu_1 z^1 + \cdots + \nu_s z^s, \quad \text{for all } \nu \in N_m^s.$$

Using this mesh we wish to find a matrix of coefficients ζ_i so that d_j can be computed with the discrete convolution, $d_j = \sum_i \zeta_i Q_{j-i}$. First of all, looking at the point averaging algorithm we can see that the range of indices that influence d_j is limited to a set that depends on the set of knot directions $Z_R = (z^{s+1}, \ldots, z^n)$. Since the averaging algorithm involves $n - s$ averages of m points each, there must also be a factor of $m^{-(n-s)}$ in each ζ_i. These two facts hint that ζ might be a discrete box spline, which is shown in the following proof.

Theorem 8: For $n \geq 2$, $s \geq 1$, and $|\det(z^1, \ldots, z^s)| = 1$,

$$d_j = \sum_i \beta(i \mid Z_R, m)\, Q_{j-i}.$$

Proof: To simplify the notation, the set R_i is defined as follows:

$$R_i = \{\, p = mi + \nu_1 z^1 + \cdots + \nu_s z^s,\ \forall\, \nu \in N_m^s \}.$$

Since $z^1, \ldots, z^s$ span **J** the sets R_i and R_j are disjoint for $i \neq j$ and the union over all i, $\cup R_i = \mathbf{J}$. For a fixed value of j, the sum over all $i \in \mathbf{J}$,

$$\sum_i \beta(i \mid Z_R, m)\, Q_{j-i} = \sum_p \beta(j - p \mid Z_R, m)\, Q_p$$

for $p = j - i$. Since R_i and R_j are disjoint for $i \neq j$, this is equivalent to splitting the sum into a double sum as follows:

$$\sum_p \beta(j - p \mid Z_R, m)\, Q_p = \sum_i \sum_{p \in R_i} \beta(j - p \mid Z_R, m)\, Q_p.$$

Since for all $p \in R_i$, $Q_p = P_i$ we have

$$\sum_i \sum_{p \in R_i} \beta(j - p \mid Z_R, m)\, Q_p = \sum_i P_i \sum_{p \in R_i} \beta(j - p \mid Z_R, m). \tag{4.10}$$

Now consider the discrete box spline $\beta(j - mi \mid Z, m)$. From the recurrence relation for discrete box splines, equation (4.8), we have for a fixed i and m,

$$\beta(j - mi \mid Z, m) = \frac{1}{m} \sum_{k=0}^{m-1} \beta(j - mi - kz^q \mid Z_q, m).$$

Applying this relation s times for $q = 1, \ldots, s$ we have

$$\beta(j - mi \mid Z, m) = \frac{1}{m^s} \sum_{k_1=0}^{m-1} \cdots \sum_{k_s=0}^{m-1} \beta(j - mi - k_1 z^1 - k_2 z^2 - \cdots - k_s z^s \mid Z_R, m).$$

Multiplying by m^s, letting $p = mi + k_1 z^1 + k_2 z^2 + \cdots + k_s z^s$ and using equation (4.9) we have

$$\alpha_{i,n}(j) = m^s\, \beta(j - mi \mid Z, m) = \sum_{p \in R_i} \beta(j - p \mid Z_R, m).$$

Substituting this into equation (4.10) we have

$$\sum_i \beta(i \mid Z_R,m)\, Q_{j-i} = \sum_i P_i \sum_{p \in R_i} \beta(j-p \mid Z_R,m) = \sum_i P_i\, \alpha_{i,n}(j) = d_j. \;\square$$

A one-pass refinement algorithm can now be stated in terms of the discrete box spline.

Algorithm 9: Given a knot direction sequence Z, coefficients $\{P_i\}$, and a positive integer m:

1. Calculate the matrix $\{Q_j\}$ consisting of m^2 copies of each P_i.
2. Calculate the discrete box spline $\beta(j \mid Z_R, m)$
3. Each new mesh point, d_j, is the discrete convolution of the mesh $\{Q_j\}$ with the discrete box spline $\beta(j \mid Z_R, m)$ at j.

4.6 Examples

In this section examples of box spline surfaces and refinement will be shown to illustrate the results of this chapter. The simplest box spline is the linear finite element $M_{1,1,1}$. A surface S defined with this box spline is simply a piecewise linear (triangular) interpolation of the control mesh with one diagonal. For the square region, Ω, consisting of four unit squares and eight triangles appropriately translated, the set I_Ω is the set $(0,0) \le j \le (2,2)$. This means that a 3×3 rectangular control mesh defines a box spline surface over this region. Figure 26 shows such a mesh. The mesh is drawn by showing all the rows and columns as well as showing the diagonal mesh lines from P_j to $P_{j+(1,1)}$ for the appropriate j's. Figure 27 shows the result of applying the refinement algorithm with m = 5. The refined mesh is a 11×11 control mesh.

The next example is a C^0 quadratic surface over a single unit square domain Ω. This is a surface with a tangent discontinuity in the diagonal direction. This can be used to produce a four-sided surface with a crease from one corner to the opposite corner, something that cannot be done with a tensor product. Figure 28 shows the control mesh for the surface. As previously noted, the control mesh in this case is a rectangular set of points with two corners missing. Figure 29 shows the refined mesh when the refinement algorithm is applied with m = 5. The refined mesh shows the tangent discontinuity as well as the fact that the mesh is converging to the "missing" corner points.

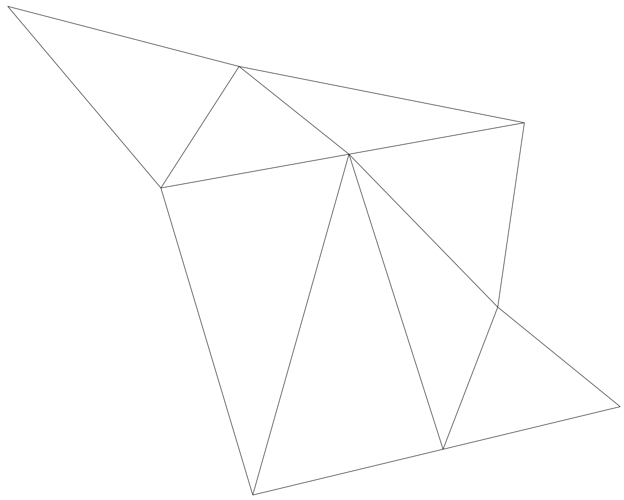

Figure 26: Control mesh for piecewise linear surface

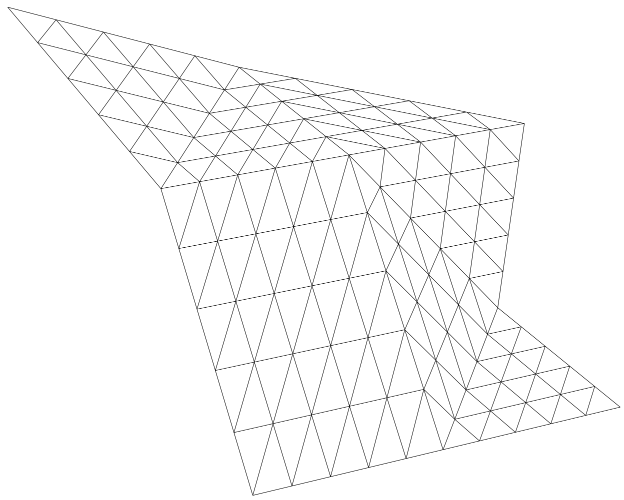

Figure 27: Refined control mesh for linear surface

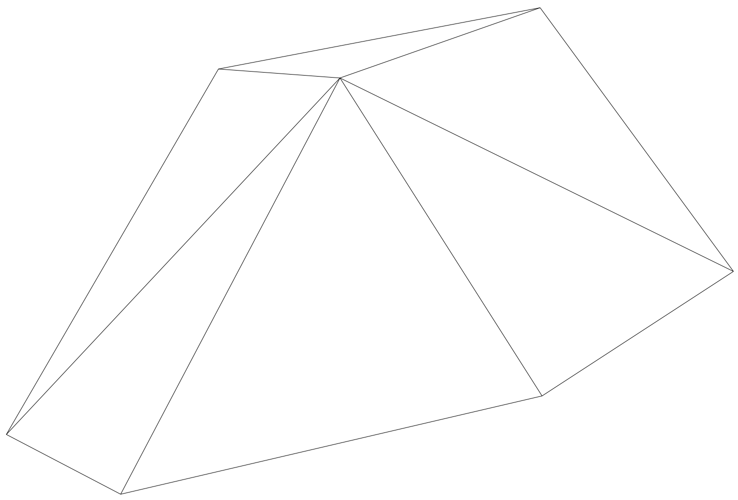

Figure 28: Control mesh for C^0 quadratic surface

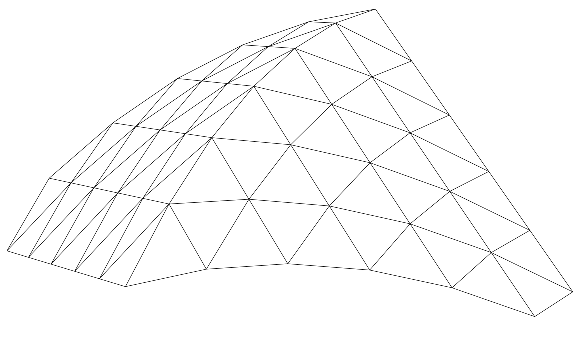

Figure 29: Refined control mesh for quadratic surface

The last example is a C^1 cubic box spline defined over a six-sided region Ω. In this case the mesh is a 4×4 mesh with two corner points missing. Figure 30 shows both the control mesh and a refined mesh for m = 5. The uniform nature of the box spline is exhibited by the fact that the refined mesh "floats" inside the original control mesh and does not interpolate any of the original control points.

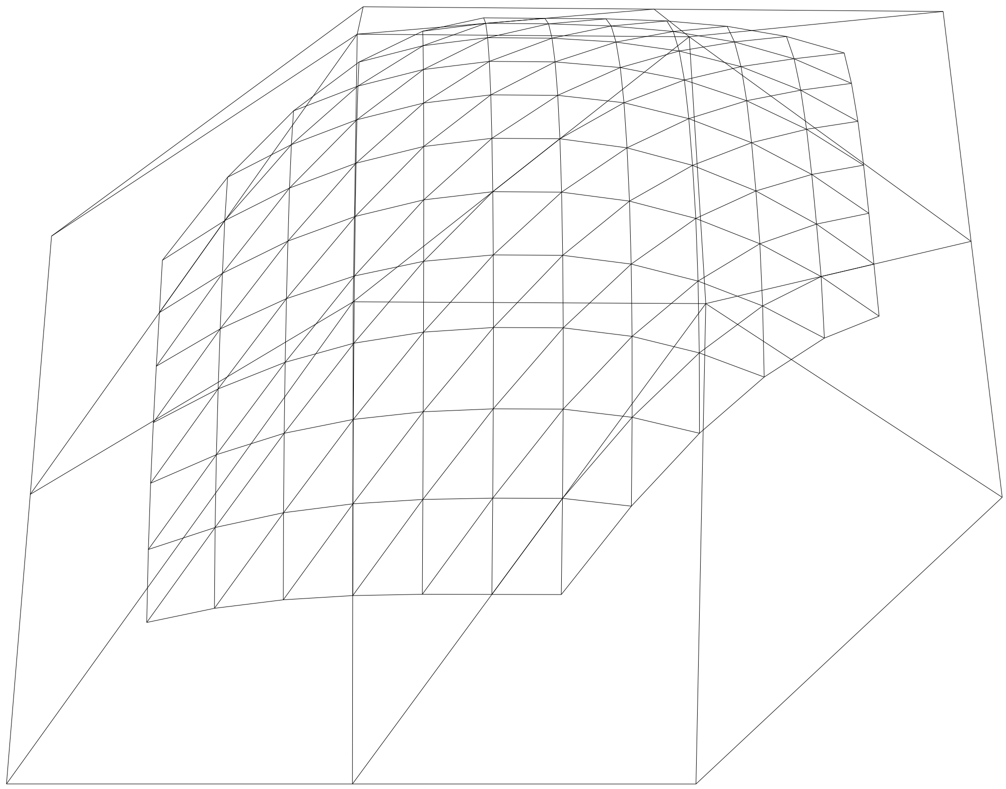

Figure 30: Control mesh and refined mesh for C^1 cubic surface

CHAPTER 5

RENDERING BOX SPLINE SURFACES

As raster graphics plays an increasing roll in CAGD systems, computationally efficient algorithms for rendering are a necessary tool. It is desirable that these algorithms be adaptive so that they selectively refine the surface mesh representation where the surface is most curved, thus generating the minimum necessary data to approximate the surface to a given tolerance. The previously discussed refinement algorithms for box splines are of a global nature, which refine the surface in a uniform manner without regard for the surface geometry. Also, a refinement algorithm, per se, does not address the issue of converting the surface to a polygonal representation for rendering. It is noted that previously discussed algorithms for univariate and tensor product B-splines (viz., the de Boor-Cox and Oslo algorithms) do not address theses issues either. In this chapter the previously discussed discrete box spline refinement algorithm is applied to develop algorithms for adaptively refining and rendering box spline surfaces.

5.1 Subdivision

An adaptive subdivision algorithm usually consists of two subalgorithms. The first is a method of adding degrees of freedom (refinement) and then "splitting" a surface (or curve) into two or more pieces. Usually the surface mesh is split into disjoint pieces, hence the name "subdivision." The second part of the algorithm is a criterion for termination. Usually this is a test of whether the surface in question is "flat" enough to be approximated by polygons within a tolerance. Included in this part is a method for constructing the polygonal approximation. A typical adaptive subdivision algorithm might look like the following:

```
%
% Adaptive subdivision for graphical display.
%
procedure display-surface( Surface );
begin
    % If the surface is "flat," then display it.
    if flat-enough( Surface ) then
        display( make-polygons-from-surface( Surface ))
    else
    {
        % Split the surface into independent pieces.
        New-Surfaces := split-surface( Surface );

        % Adaptively subdivide each new piece.
        foreach surf in New-Surfaces do
            display-surface( surf )
    };
end;
```

The recursive nature of the algorithm adapts to the curvature of the surface, subdividing curved sections into smaller and smaller pieces until they are considered flat.

The Oslo algorithm has been used in adaptive subdivision algorithms for nonuniform tensor product B-splines with open end conditions [52]. By inserting a k-fold knot in a B-spline curve of order k, the curve is subdivided into two separate curves since the k-fold knot introduces a discontinuity. This discontinuity enforces the open end conditions at the split and is only in the parametric sense since the two new curves are pieces of the original curve. Using this technique with a tensor product B-spline surface, the surface is refined by inserting a k-fold knot in one of the two knot vectors. The surface can then be split into two surfaces along a curve at the point of the multiple knot. With the assumption that the starting surface is defined with open end conditions, this process maintains these conditions at each step.

The adaptive subdivision algorithm presented above relies heavily on the properties of the open B-spline, namely that an open curve interpolates the first and last polygon points and is tangent to the polygon at these points. The interpolation property allows a surface mesh to be subdivided into disjoint pieces. It also provides that the corner points of a mesh are points on the surface which can be used when the surface is converted to a polygonal representation. The tangency property is important for generating surface normal vectors at the corners which are necessary for the rendering process. At each corner of a mesh a tangent vector can easily be computed for each of the two boundary curves from their polygons and the cross product of these vectors is normal to the surface. Finally, open end conditions also simplify the process of flatness testing. Since a boundary curve of a surface is only dependent upon one row or column of the mesh, it is easy to test whether these curves are linear to within some ε. Thus if all the boundary curves of a surface are linear within ε, and the mesh is planar within δ, then the surface can be approximated by piecewise linear surfaces (i.e., polygons).

5.2 Adaptive Refinement of Box Spline Surfaces

The box spline surface, based on uniform basis functions, exhibits floating end conditions: No point in the mesh is necessarily interpolated by the surface and tangent vectors cannot simply be extracted from rows or columns of mesh points. This provides a number of challenges in designing an adaptive subdivision algorithm. These problems exist with tensor product floating surfaces as well. The following problems must be solved:

1. Splitting - Methods for splitting a refined surface into separate pieces must be defined. These pieces will not be made up of disjoint sections of the mesh, but must define surfaces that are disjoint with the exception of their boundaries (i.e., there must be some redundancy in the separated meshes).

2. Flatness testing - A method for testing boundary curves for linearity must be developed. Testing the mesh for planarity and linearity in the rows and columns is not sufficient. For example, a tensor product floating surface can have a mesh that lies in a plane, with all the rows and columns of the mesh being linear, but the surface still having curved boundaries.

3. Surface evaluation - Points on the surface must be evaluated when flat surface pieces are converted into polygons. The floating mesh converges slowly along the boundary curves, while it converges more quickly elsewhere, making a special method for surface evaluations along the boundaries necessary for speed and accuracy.

4. Normal vectors - A method for calculating normal vectors at the corners of a surface is also necessary for the rendering process.

Since a floating surface mesh cannot be split into disjoint pieces, we cannot call such an algorithm "subdivision," per se, but since the algorithm will adapt to the curvature of the surface we will call the process "adaptive refinement."

5.2.1 Adaptive Floating Curve Refinement

As a starting point, an algorithm for adaptively refining a floating uniform B-spline curve (which is a special case of the box spline for s = 1) is presented. This will illustrate possible solutions to the first and third problems identified above. The top level algorithm is the typical recursive algorithm as above.

```
procedure display-curve( Curve );
begin
    % If the curve is "straight," then display it.
    if straight-enough( Curve ) then
        display( make-segments-from-curve( Curve ))
    else
    {
        % Split the curve into independent pieces.
        New-Curves := split-curve( Curve );

        % Adaptively refine each new piece.
        foreach crv in New-Curves do
            display-curve( crv )
    };
end;
```

To develop the curve splitting algorithm, consider the refinement process for uniform floating B-spline curves. If a control polygon consists of n points, the curve has $n - (k - 1)$ polynomial pieces each of degree $k - 1$. The support of the curve is the interval $[k - 1, n]$. Over each interval $[i, i+1]$ there are k nonzero basis functions, and thus only k control points influence any one polynomial piece of the curve. Refining such a curve with the discrete box spline refinement algorithm with $m = 2$ results in a new refined curve consisting of $2n - (k - 1)$ points with $2(n - (k - 1))$ polynomial pieces. This is equivalent to the Lane-Riesenfeld algorithm [43] and equivalent to inserting one new knot at the midpoint of each interval $[i, i+1]$ for $i = k - 1 - \lfloor \frac{k}{2} \rfloor, \ldots, n + k - \lfloor \frac{k}{2} \rfloor - 1$ with the Oslo algorithm. The first n points of the refined polygon control the first $n - (k - 1)$ polynomial pieces of the curve and the last n points define the second $n - (k - 1)$ pieces. The curve can now be split into two new pieces that have the same properties of the original, i.e., curves of order k with n control points. These new control polygons have points in common, but they describe separate curves.

The splitting algorithm for adaptive curve refinement can now be stated as follows:

```
%
%       Refine a curve and split into two new curves.
%
procedure split-curve( curve );
begin
  % Refine the curve (order k) with m = 2.
  new-curve := curve-refine( curve, 2 );

  % Make a control polygon from the first n points.
  first-poly := first-n-points( new-curve );

  % Make a control polygon from the last n points.
  second-poly := last-n-points( new-curve );

  % Return two new curves of order k.
  return( make-curve( first-poly, k ),
          make-curve( second-poly, k ));
end;
```

An example will help illustrate the curve splitting algorithm as well as the problem of converting the refined floating curves into a piecewise linear representation. In Figure 31 a cubic B-spline (order 4) control polygon is shown as $P_0, \ldots, P_4$. The curve defined by this control polygon consists of two cubic pieces, as described above. After one level of refinement the new control polygon is shown as $Q_0, \ldots, Q_6$ (shown as the hollow dots). At this point two new curves can be defined, one with control polygon $Q_0, \ldots, Q_4$ and one with control polygon $Q_2, \ldots, Q_6$. Now suppose each of these is tested for straightness and the first curve fails the test but the second one passes. If so, the the curve with control polygon $Q_0, \ldots, Q_4$ must be refined further. After this curve is refined, a new curve is defined with control polygon $R_0, \ldots, R_6$. This is again split into two new curves with control polygons $R_0, \ldots, R_4$, and $R_2, \ldots, R_6$ respectively. Suppose that both these curves pass the straightness test. Now the refinement is complete and the final piecewise linear approximation must be constructed.

The problem of floating end conditions now becomes evident. Since the control polygons which describe neighboring curves overlap, it is not clear how to construct the linear approximation. If the approximation is to be constructed out of the control polygons, which parts of which polygon should be used where they overlap? This is particularly complicated

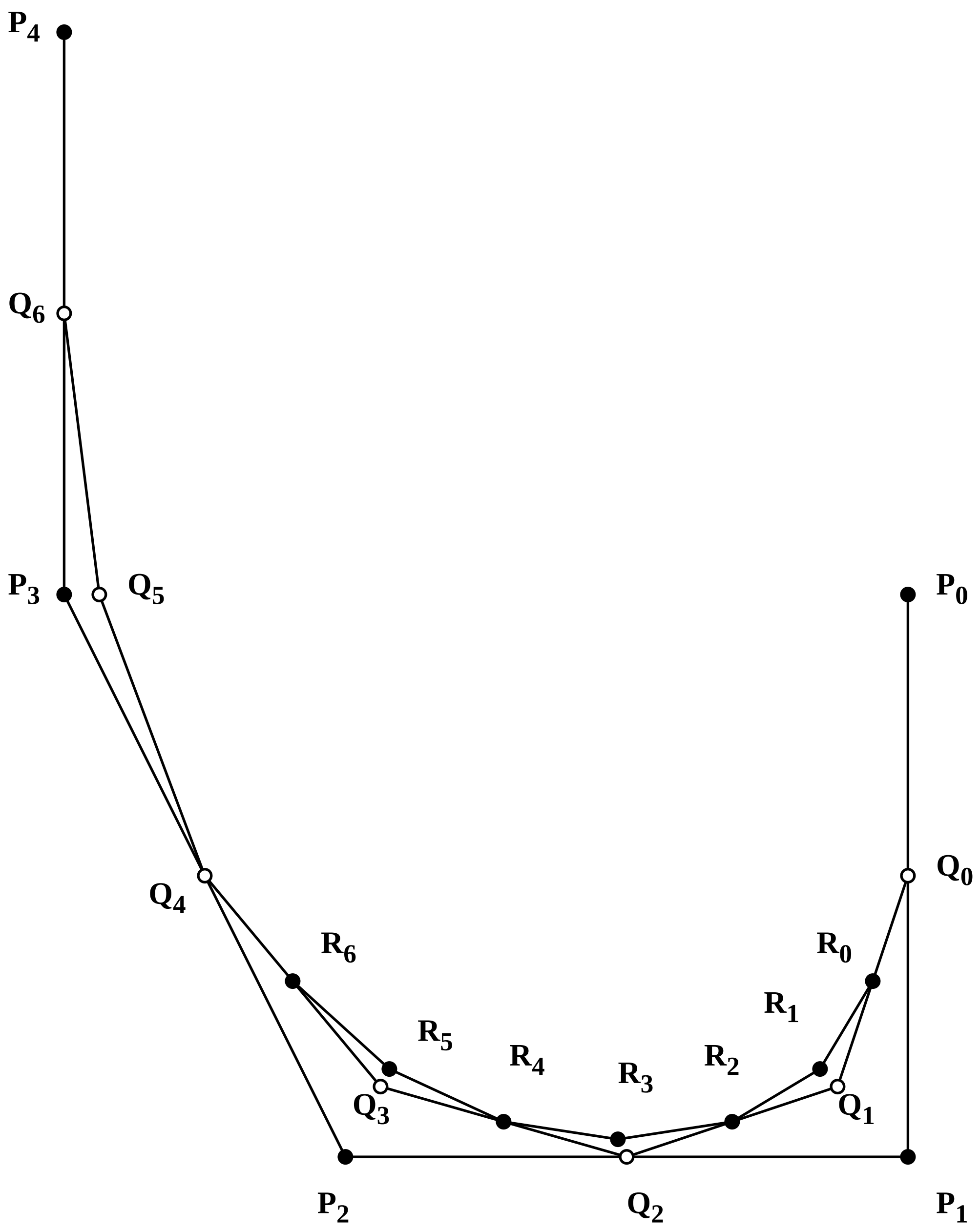

Figure 31: Floating curve refinement

where the neighboring polygons are the result of different levels of refinement (e.g., the area where $R_2, \ldots, R_6$ overlaps $Q_2, \ldots, Q_6$). It is noted that although in this case the control polygons of different levels of refinement touch each other, this is not true in general (for higher degrees) and makes the problem even more complicated. A second problem is that the convergence of the control polygons to the curve in the interior is much better than at the end points. Figure 32 shows the Q, and R polygons and the floating curve. It can be seen in the figure that the polygon points $R_1, \ldots, R_6$ and $Q_2, \ldots, Q_5$ are fairly close to the actual curve after just a few refinements, but the end points R_0 and Q_6 tend to "overshoot." In general, if the tolerance is adjusted so that the end points are within the acceptable range, the interior points will be within a much closer tolerance.

The solution to both these problems, as suggested earlier, is to actually evaluate the curves at the end points. This, of course, guarantees that the end points of the original curve are reproduced. If the end points are evaluated for each final curve in the adaptive refinement, it also solves the problem of constructing the final piecewise linear approximation by eliminating the overlapping polygons. Now each curve in the final approximation is represented with a single line segment and the result is guaranteed to be a C^0 piecewise linear curve. While evaluating points on B-spline curves (or surfaces) is usually regarded as a computationally expensive procedure, the uniform nature of the basis functions is now turned into an advantage since the basis functions only need to be evaluated once and tabled at the necessary boundary points. Evaluating a point on the curve can then be done with k multiplies and $k - 1$ additions. The algorithm for constructing line segments from the curves can now be stated as follows:

```
procedure make-segments-from-curve( Curve );
begin
  % Evaluate the end points of the curve.
  p0 := curve-eval( Curve, left-end );
  p1 := curve-eval( Curve, right-end );

  % Display the segment.
  display-segment( p0, p1 );
  return;
end;
```

Figure 33 shows the results of using the adaptive refinement algorithm with a "coarse" resolution. The curve is rendered with more segments over the first section where it is more curved, and only a few segments where it is relatively flat.

5.2.2 Surface Splitting

To generalize the above algorithm from curves to surfaces, the refinement process for box spline surfaces must be examined. In Chapter 4 the set of box splines defined over a region Ω was defined as I_Ω. The set of control points $\{P_j : j \in I_\Omega\}$ is not necessarily a rectangular mesh of points, even for a surface defined over a rectangular region Ω. It is convenient, however, for discussion (and implementation issues) to consider the control mesh of a surface as a (rectangular) matrix of points. This can be done assuming a marker can be defined to denote what might be called "null points," which are points that do not affect the given surface. We will make this assumption and consider the control mesh of any box spline surface to be the smallest rectangular matrix (or array or mesh) of points that define the surface, with the knowledge that some of the points in the matrix may be null points. Therefore, when a matrix is referred to as "square" or "rectangular" this refers to the sizes of the mesh, and *not* the region Ω that the surface is defined over. Furthermore, as a matter of convention, the rows of the matrix will be associated with the v (or y) direction in the parametric plane and the columns, the u (or x) direction so that a $p \times q$ matrix is a matrix $\{P_j\}$ such that $(0,0) \le j \le (p,q)$.

Using these conventions, consider the refinement of a box spline surface S with control mesh $\{P_j\}$ using the generic box spline $M_{r,s,t}$. If the minimum j is the origin (0,0) then the maximum j in the rectangular matrix of points $\{P_j\}$, will be $z^* - d^3 = (r + t - 1, s + t - 1)$ for a surface S defined over either the type-1 triangle $\tau^1 = [\{z^*, z^* - d^3, z^* - d^2\}]$ or the type-2 triangle $\tau^2 = [\{z^*, z^* - d^3, z^* - d^1\}]$, or the square region $\Omega = \tau^1 \cup \tau^2$.* Therefore, if the control mesh of S is a $p \times q$ matrix of points (with possible null points included), the surface S is defined over a region of the plane Ω that is bounded by the rectangular region $[r + t - 1, p] \times [s + t - 1, q]$. Refining this surface with a factor m results in a new control mesh $\{Q_j\}$, which

*The maximum or minimum of a set of multi-integers is j such that |j| is maximized or minimized respectively, where |j| is the L_0 norm $|j_1 + j_2|$.

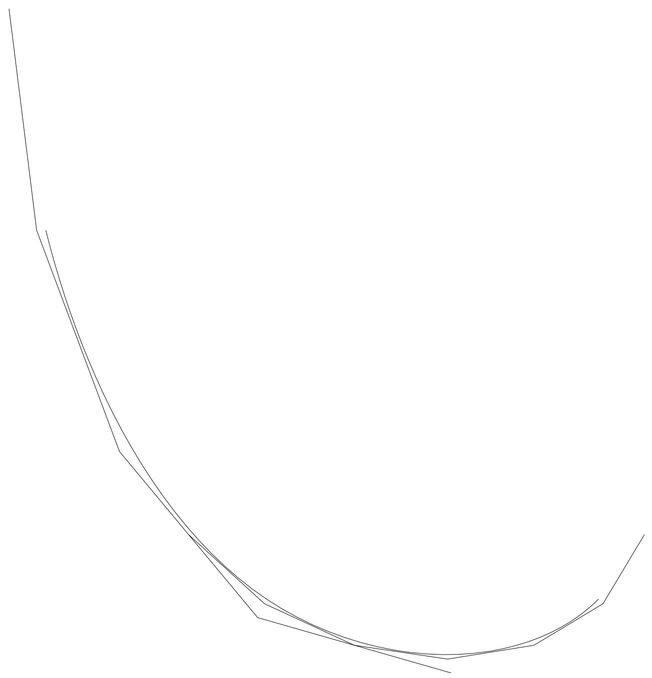

Figure 32: Final control polygons and curve

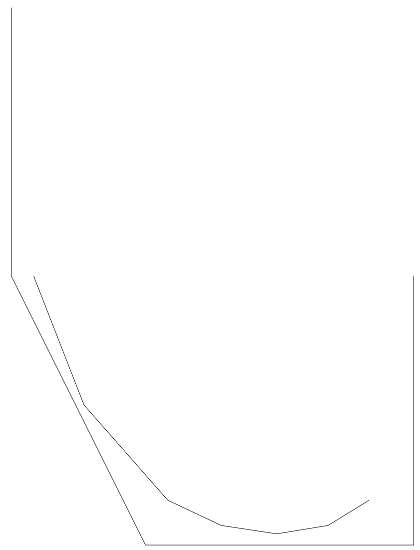

Figure 33: Adaptively refined curve with original control polygon

is a $(p + (m - 1)(p - (r + t - 1))) \times (q + (m - 1)(q - (s + t - 1)))$ matrix of points. The new surface is defined over m^2 times as many polynomial pieces as the original surface and is defined on the region Ω' which is a subpartion of a new finer three-direction grid Δ' which is Δ with $m - 1$ additional grid lines between each original grid line in each direction.*

To illustrate the problem of surface splitting, consider the example of the box spline surface S defined over the region consisting of one unit square of Δ (two triangular polynomial pieces). The control mesh for this surface is the $r + t \times s + t$ matrix of points $\{P_j\}$ for $(0,0) \le j \le (r + t - 1, s + t - 1)$. In general, this mesh may have null points and still define a surface over the square region. Refining the surface with $m = 2$ results in a refined mesh $\{Q_j\}$ which defines a surface over a square domain now consisting of four square subregions (eight triangular polynomial pieces). The size of the new mesh is $r + t + 1 \times s + t + 1$. The matrix of points $\{Q_j\}$ where $(0,0) \le j \le (r + t - 1, s + t - 1)$ defines a surface over the lower left square of the new domain, $\{Q_j\}$ where $(1,0) \le j \le (r + t, s + t - 1)$ defines the lower right portion, $\{Q_j\}$ where $(0,1) \le j \le (r + t - 1, s + t)$ the upper left, and $\{Q_j\}$ where $(1,1) \le j \le (r + t, s + t)$ the upper right region. For example, with the cubic box spline $M_{2,2,1}$, $(r + t, s + t) = (3,3)$. The surface over one square region is defined by a 3×3 mesh, and the refined surface over four square regions is defined by a 4×4 mesh. The domain Ω is shown on the left in Figure 34 along with set I_Ω as the dots. The refined domain Ω' and the set $I_{\Omega'}$ is shown on the right. ** Four new surfaces can now be defined, one over each square in Ω' as described above.

If only one of the triangles in the original domain Ω above is considered, the sizes of the original and refined meshes remain the same as stated above. However, certain control points will be null points. The original triangular domain is refined into a new domain consisting of four triangles for $m = 2$. The new domain consists of three triangles of the same type as the original and one of the opposite type. For the type-1 case, the matrix of points $\{Q_j\}$ where $(0,0) \le j \le (r + t - 1, s + t - 1)$ defines the lower left type-1 triangular surface, $\{Q_j\}$ where

---

*Ω and Ω' are distinguished because they are defined on different grids Δ and Δ', i.e., they define the same region, but are composed of different sets of triangles.

**After refinement, the uv coordinate system is scaled and translated such that the minimum d_j is $j = (0,0)$ and $B'_j = N_j$ where B'_j is in the new coordinate system and N_j is in the old. Thus, although it appears that the region in the figure on the right has been enlarged as well as refined, this is the result of the change of variables, and the surfaces are defined over the same original parametric range.

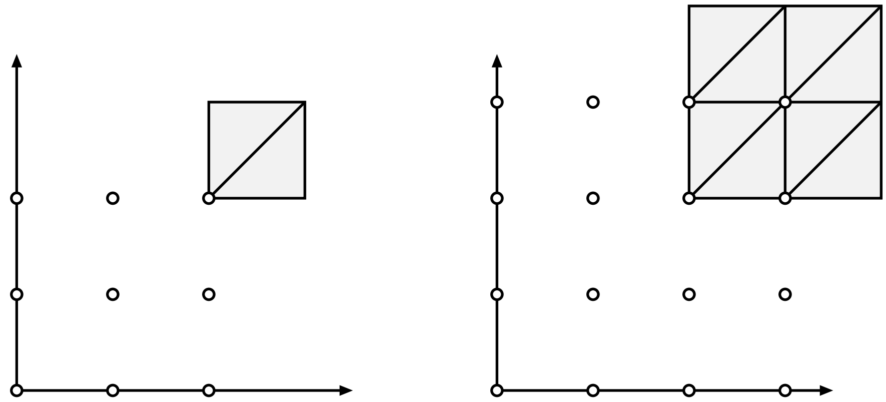

Figure 34: Cubic box splines over a square

$(1,1) \le j \le (r + t, s + t)$ defines the upper right type-1 triangular surface and $\{Q_j\}$ where $(1,0) \le j \le (r + t, s + t - 1)$ defines both the lower right type-1 triangular surface and the type-2 triangular surface adjacent to it. Two triangular surfaces are described by the same rectangular portion of the refined mesh because these surfaces are really described by *subsets* of these rectangular portions of mesh. The subsets are determined by the sets I_{τ^1} and I_{τ^2} for the type-1 and type-2 triangular regions that make up the lower right square of the new refined region Ω'. Consider the cubic box spline $M_{2,2,1}$. The type-1 region Ω and the set I_Ω are shown on the left in Figure 35. On the right, the refined region Ω' is shown along with $I_{\Omega'}$. Figure 36 shows how each of the four submeshes is related to the refined mesh $\{Q_j\}$ for each of the four triangular regions in Ω'.

In the above two examples, the refining and splitting of the surfaces resulted in new surfaces that were of the same form as the original, that is the new surfaces have the same sized meshes as the originals, and are defined over the same type of region. The square surface produces four new square surfaces; the triangular surface produces four new triangular surfaces (three of the same type, one of the opposite type). This can also be done with any rectangular or triangular region; however, when a surface defined on a more arbitrary region (such as a five-sided or six-sided region) is refined it cannot readily be split into new surfaces of the same type. It is more appropriately split into various triangular and rectangular surfaces.

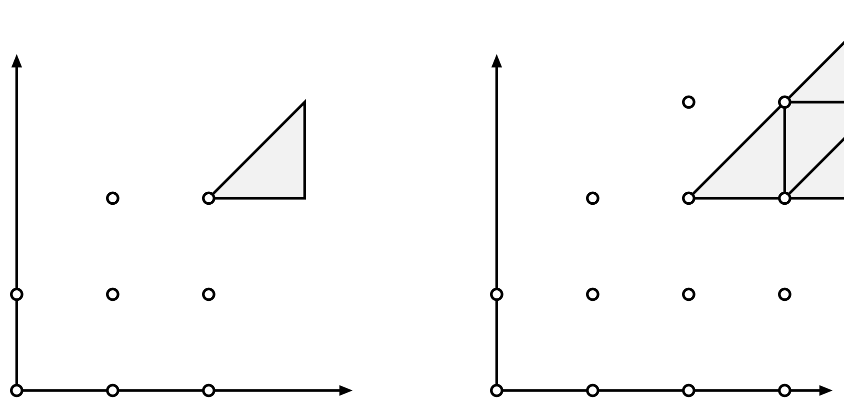

Figure 35: Cubic box splines over a triangle

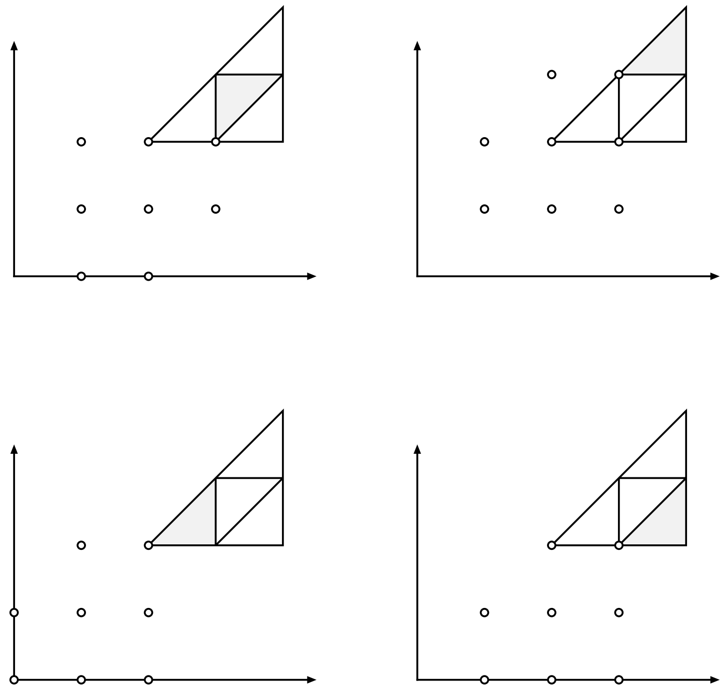

Figure 36: Cubic box splines over each new triangle

Since this is the case, the splitting algorithm will be designed for rectangular and triangular surfaces only, and a preprocess algorithm will be designed to provide an initial separation of an arbitrary surface (on the three-directional grid) into the largest possible three and four sided regions.

5.2.2.1 Windowing. Since meshes may contain null points, surface splitting in general cannot simply consist of copying rectangular sections of refined meshes into new surface meshes. Such a copy would contain every point necessary to define the new surface, but may also contain extraneous points where there should be null points. To solve this problem a two step copying process can be used. First, a rectangular portion of a refined mesh is copied into a new mesh. Second, a window is applied to mask out any extraneous points and turn them into null points. The shape of the window used will depend on which type of surface the new mesh is supposed to represent. As seen in the above example, the same rectangular portion of the refined mesh could be used to define a square region, a type-1 triangular region, or a type-2 triangular region, depending on the window used.

Windowing is only necessary is certain cases, however. The first case is when a triangular subsurface is extracted from a square surface. In this case a rectangular region *of the mesh* defines two triangular surfaces and some points must obviously be masked out if the same mesh is to define only one of these triangular surfaces. The second case is where a rectangular surface is described by a nonrectangular mesh. In this case, when a new rectangular subsurface is extracted from another rectangular surface, a windowing function must be applied to maintain the correct mesh. This phenomenon is exhibited by all surfaces with tangent discontinuities in the diagonal direction. These are surfaces where $d = n - \max\{r, s, t\} - 1 = 1$ and $\max\{r, s, t\} = t$, and is only true if $r = s = 1$ and $t \geq 2$. It is interesting to note that if a relative ε is used in the flatness testing, a square surface with a diagonal tangent discontinuity would cause an adaptive algorithm to infinitely recurse, since the surface would never be flat along the diagonal curve. This is an anomaly that the preprocess algorithm must check for. Surfaces with C^1 discontinuities in the u or v direction should initially be split along these discontinuities as well by the preprocessor. If the preprocessor is designed correctly, and the refinement process then limited to triangular and rectangular surfaces, windowing will be minimized.

5.2.2.2 Splitting arbitrary rectangular surfaces. Consider the surface S over a rectangular domain Ω with control mesh $\{P_j\}$, with possible null points, where $(0,0) \le j \le (p,q)$. If S is at least C^1 on $\Omega°$, we will call such a surface a *type-3* surface. It is again assumed that Ω is translated such that the minimum $j \in I_\Omega$ is the origin. If $M_{r,s,t}$ is the generic box spline, (p,q) must be at least (r + t, s + t) to define the smallest rectangular region, the unit square.

Refining S with m = 2, the new refined mesh $\{Q_j\}$ is a $2p - (r + t - 1) \times 2q - (s + t - 1)$ matrix of points. The new refined region Ω' can be split up many ways, three of which seem useful. The first way would be to divide Ω' into four new regions of the same proportions as the original region, thus the splitting algorithm would produce four new surfaces of the same description as the original. Another useful option would be to split Ω' in only one direction, vertically or horizontally. This could be used if a surface was linear in one direction, for example, since it would not need to be refined in this direction. These types of splitting are always possible for a surface that has been refined at least once with m = 2, since the number of polynomial pieces will be a multiple of four (m^2).

5.2.2.3 Splitting arbitrary triangular surfaces. Now consider the surface S with control mesh $\{P_j\}$ defined on an arbitrary triangular domain Ω such that S is at least C^1 on $\Omega°$. Since any region Ω on the three-direction grid is composed of τ_i's in Δ, any triangular region must be either a scaled type-1 triangle or a scaled type-2 triangle. A *type-1 surface* is then defined as a surface that meets the above criterion defined over a region Ω whose boundary is a scaled version of the type-1 triangle. A *type-2 surface* is similarly defined over a region that is a scaled type-2 triangle. To clarify, a type-1 or type-2 *triangle* is half of a unit square in the grid Δ over which a single polynomial surface can be defined; a type-1 or type-2 *region* is a union of type-1 and type-2 triangles that has the same *shape* as a type-1 or type-2 triangle respectively; and a type-1 or type-2 *surface* is a surface defined over a type-1 or type-2 region respectively.

The control mesh of S is the set of points $\{P_j\}$, with null points, where $(0,0) \le j \le (p,q)$. It is again assumed that Ω is translated such that the minimum $j \in I_\Omega$ is the origin. For the box spline $M_{r,s,t}$, (p, q) must be at least (r + t, s + t) to define a surface over at least one type-1 or type-2 triangle.

Refining S with m = 2, the new refined mesh $\{Q_j\}$ is again a $2p - (r + t - 1) \times 2q - (s + t - 1)$ matrix of points. In previous examples it was shown that a type-1 or type-2 surface defined over a single type-1 or type-2 triangle could be refined and split into four new surfaces. Since two of the new triangular pieces are really part of one square region, there is really no reason to split these into two individual pieces. The same is true for the arbitrary type-1 or type-2 surface. If S is a type-1 surface it can be refined and split into two new type-1 surfaces and one new type-3 surface. Similarly, a type-2 surface can be refined and split into two new type-2 surface and one type-3 surface. Extracting the square region also eliminates the need for any windowing, which simplifies the algorithm.

5.2.2.4 Preprocess algorithm. The algorithms described above can be used to split any refined type-1, type-2 or type-3 surface into new surfaces that are also of these types. To use these in an adaptive refinement algorithm, a preprocess must be used to convert an arbitrary surface to a set of type-1, type-2, and type-3 surfaces. The simplest preprocess would be to separate the surface into its polynomial triangular pieces (which are type-1 and type-2 surfaces). This can be used with C^0 surfaces (to ensure that the resulting pieces are free of tangent discontinuities), and in other anomalous cases. For C^1 surfaces, however, it is desirable to split the surface into the minimum number of pieces.

The general region Ω based on the three-direction grid, always has a lower corner (minimum point) and an upper corner (maximum point) and *may* have a lower right corner, an upper left corner, both, or neither. Of course, if all the corners exist the region is rectangular. If three of the corners exists, the region might be three-sided or five-sided (a rectangle with one corner "cut off"), and if only two of the four corners exist, the region is six-sided. Figure 37 shows a possible scheme for dividing the arbitrary region into triangular and rectangle regions. First, the rectangular region that bounds Ω is found. This region is shown in the figure from the origin to the point (max-u, max-v). There are two possible triangular regions that can be "missing" from this rectangular region. One is the type-1 region from the lower right corner, and the other the type-2 region from the upper left corner. Depending on the size of these regions, it may be necessary to divide Ω into as many as five pieces as shown in Figure 37.

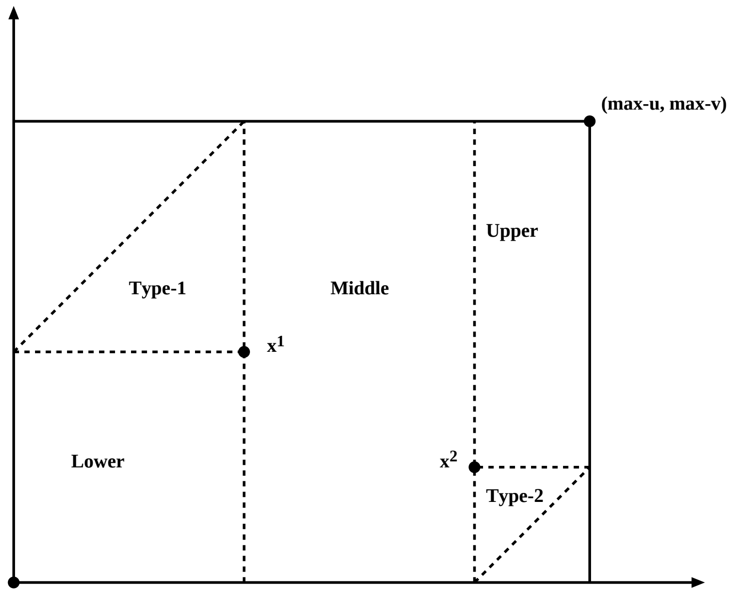

Figure 37: Dividing a region into type-1, type-2, and type-3 regions

To find which pieces are necessary, the points x^1 and x^2 are found in parametric space which define the type-1 and type-2 portions of Ω adjacent to the "missing" triangular regions of the opposite type. These two points can be used to determine the shape of the region Ω. For example, if x^1 and x^2 are coincident, the middle rectangular piece is not necessary and the six-sided region is divided into two triangular pieces and two rectangular pieces. These two points can be used to identify the presence of each of the five regions, or lack thereof, except when x^1 and x^2 "overlap" (i.e., $x_1^2 < x_1^1$ and $x_2^2 < x_2^2$). In this case a general algorithm for dividing Ω into triangular and rectangular pieces becomes very complicated and the region must be divided into many regions. Instead of making the preprocess overly complex, these cases can be treated as anomalous and the surface split into type-1 and type-2 polynomial pieces.

5.2.2.5 Splitting algorithm. Some details of an adaptive refinement algorithm for the display of box spline surfaces can now be established. First, the top level algorithm is defined by incorporating the two preprocess algorithms discussed in the previous section into the generic recursive "display-surface" algorithm stated at the beginning of the chapter. The "split-surface" sub-algorithm can be defined in terms of the previously discussed methods for splitting triangular (type-1 and type-2) and rectangular (type-3) surfaces. The top level algorithm and the splitting algorithm are now stated as follows:

```
%  Adaptively refine a surface and display.
procedure adaptively-refine-and-display( surface );
begin
  % Divide surface along discontinuities,
  % or split into type-1, type-2 and type-3 surfaces.
  if surface is not C^1
    surfs := polynomial-pieces( surface );
  else
    surfs := surface-pieces( surface );

  for each srf in surfs
    display-surface( srf );
end;

%  Refine a surface and split into new surfaces.
procedure split-surface( surface );
begin
  % First refine the surface with m = 2.
  refined-srf := surface-refine( surface, 2 );

  if surface is type-1 or type-2 then
  {
    return( split-triangular-srf( refined-srf ));
  }
  else   % type-3
  {
    if flat-in-v-direction( refined-srf ) then
      return( split-vertical-type-3( refined-mesh ))
    else if flat-in-u-direction( refined-srf ) then
      return( split-horizontal-type-3( refined-srf ))
    else
      return( split-into-four-type-3( refined-srf ));
  };
end;
```

5.2.3 Edge Curves

In section 5.1 it was shown that subdivision algorithms benefited greatly from open end conditions. Many of the advantages of open end conditions involve the information present in the edge curve control polygons, viz., tangent vectors and surface corner points. If the boundary curves of a box spline surface could be represented by open B-spline curves many of the problems of adaptively refining floating surfaces could be solved. The edge curves would provide corner points on the surface, eliminating the problem of overlapping surface meshes when constructing the final piecewise linear approximation, and the tangent vectors could be used to calculate surface normals. The edge curves would also simplify flatness testing since they can easily be tested for linearity. Since the edge curves (including curves in the diagonal direction) of a box spline surface are piecewise polynomial curves of degree $n - 2$ and have $d - 1$ continuous derivatives, such a conversion of representation is possible, but a computationally efficient means of doing so must be defined.

Consider the box spline surface S defined on the square $\Omega = [z_1^* - 1, z_1^*] \times [z_2^* - 1, z_2^*]$ consisting of one type-1 triangle and one type-2 triangle. There are five curves of interest in this region, the four boundary curves plus the diagonal curve. These are defined as follows:

$$
\begin{aligned}
E(t) &= S(t, z_2^* - 1) = \sum_{j \in I_\Omega} P_j B_j(t, z_2^* - 1), && \text{for } t \in [z_1^* - 1, z_1^*] \\
F(t) &= S(t, z_2^*) = \sum_{j \in I_\Omega} P_j B_j(t, z_2^*), && \text{for } t \in [z_1^* - 1, z_1^*] \\
G(t) &= S(z_1^* - 1, t) = \sum_{j \in I_\Omega} P_j B_j(z_1^* - 1, t), && \text{for } t \in [z_2^* - 1, z_2^*] \\
H(t) &= S(z_1^*, t) = \sum_{j \in I_\Omega} P_j B_j(z_1^*, t), && \text{for } t \in [z_2^* - 1, z_2^*] \\
K(t) &= S(t, t - z_1^* + z_2^*) = \sum_{j \in I_\Omega} P_j B_j(t, t - z_1^* + z_2^*), && \text{for } t \in [z_1^* - 1, z_1^*].
\end{aligned}
\tag{5.1}
$$

To simplify the notation, the piecewise polynomial curve $\theta_{j,E}(t)$ is defined as

$$\theta_{j,E}(t) = B_j(t, z_2^* - 1), \quad \text{for } t \in [z_1^* - 1, z_1^*] \quad \text{and } j \in I_\Omega.$$

The functions $\theta_{j,F}(t)$, $\theta_{j,G}(t)$, $\theta_{j,H}(t)$, and $\theta_{j,K}(t)$ are similarly defined from the above equations in (5.1). Since each of these five univariate functions is an isoparametric curve on a box spline (basis function) surface (i.e., the intersection of a plane with the surface), they are piecewise polynomial curves of degree n – 2; moreover, in this case, they are single polynomials because of the choice of Ω. Since this is the case, and since B-splines of order k = n – 1 form a basis for piecewise polynomials of degree k – 1 = n – 2, each of these functions has a open B-spline representation defined by the following equations:

$$\theta_{j,E}(t) = \sum_{i=0}^{n-2} E_j^i N_i(t), \qquad \theta_{j,F}(t) = \sum_{i=0}^{n-2} F_j^i N_i(t),$$

$$\theta_{j,G}(t) = \sum_{i=0}^{n-2} G_j^i N_i(t), \qquad \theta_{j,H}(t) = \sum_{i=0}^{n-2} H_j^i N_i(t), \tag{5.2}$$

$$\theta_{j,K}(t) = \sum_{i=0}^{n-2} K_j^i N_i(t).$$

The functions $N_i(t)$ are the B-spline basis of order k = n – 1, based on the Bézier knot vector $t_0 = t_1 = \cdots = t_{k-1} = 0$, $t_k = t_{k+1} = \cdots = t_{2k-1} = 1$.

For a fixed j ($j \in I_\Omega$), a linear system of n – 1 equations with n – 1 unknowns can be produced from each of the five above equations in (5.2) by evaluating each equation at n – 1 different values. The unknowns are the coefficients E_j^i, F_j^i, G_j^i, H_j^i, and K_j^i for i = 0, . . . ,n – 2 respectively. Solving these linear systems for each $j \in I_\Omega$, the boundary curves of the box spline surface can now be written in terms of the B-spline functions N_i. For example, the bottom curve,

$$E(t) = \sum_{j \in I_\Omega} P_j\, \theta_{j,E}(t) = \sum_{j \in I_\Omega} P_j \sum_{i=0}^{n-2} E_j^i N_i(t) = \sum_{i=0}^{n-2} \Big(\sum_{j \in I_\Omega} P_j E_j^i\Big) N_i(t),$$

can be written as the open B-spline curve

$$E(t) = \sum_{i=0}^{n-2} Q_i N_i(t) \text{ where } \qquad Q_i = \sum_{j \in I_\Omega} P_j E^i_j.$$

The control polygons of the other curves are similarly defined in terms of F^i_j, G^i_j, H^i_j, and K^i_j.

To extend this to more arbitrary regions Ω, there are at least two possible strategies:

1. Allow j to range over the new set I_Ω, calculate a new knot vector t_i with equally spaced interior knots of the correct multiplicity to match the continuity of B_j in the given direction, and compute the coefficients E^i_j, F^i_j, G^i_j, H^i_j, and K^i_j as before.
2. Use the coefficients E^i_j, F^i_j, G^i_j, H^i_j, and K^i_j defined for the Bézier case repeatedly for each polynomial span of the edge curve of the arbitrary surface and construct the edge curve by concatenating the individual curves together to form a piecewise Bézier curve.

If the second method is used, the uniform nature of the box spline allows the coefficients to be computed once, for the given box spline basis, and stored in tables as constants. The edge curves of any surface S defined with the box spline $M_{r,s,t}$ can then be computed with the same tables, no matter the size and shape of S. The piecewise Bézier formulation of the edge curve also provides a "pre-subdivided" representation of the curve which means the control polygon will be a better approximation to the curve than if the first method were used.

5.2.4 Flatness Testing

With the presence of open edge curves, the problem of flatness testing is much less formidable than first supposed. The term "flatness" testing is a bit of a misnomer since what is required is more than a surface that lies in a plane; the edge curves must be straight (linear) also. A typical algorithm for the tensor product surface (with open end conditions) might be to test each row and column of the mesh for linearity. If the surface passes this test it must be linear in each direction to within ε, including the edge curves. The last test (which is then really the "flatness" test) is to test how close the four corners are to lying in a plane.

For the box spline with open edge curves, a very similar algorithm can be used. Rows and columns of type-1 or type-2 surfaces are of varying lengths, but still can be used to determine the linearity of the surface. If the edge curves are straight to within ε and the rows and columns are all straight also, the surface is considered flat. For the type-3 surface the algorithm is the same as the tensor product case, only the edge curves are tested for straightness using their open B-spline representations.

5.2.5 Examples

Some examples of box spline surfaces rendered with the adaptive refinement technique are now presented. To show the refinement pattern achieved by the algorithm, individual polygons are colored differently resulting in a "quilted" image. Type-3 surfaces deemed to be "flat" are converted into four triangles each. Type-1 and type-2 surfaces deemed to be "flat" are converted to single triangles.

5.2.5.1 Adaptive subdivision vs. adaptive refinement. Since a uniform floating tensor product surface is a special case of the box spline, the adaptive refinement algorithm can be used on such a tensor product surface. This can then be compared to the adaptive subdivision of an equivalent open representation of the surface. The tensor product floating surface can be easily converted using the Oslo algorithm to insert multiple knots at the ends of the two knot vectors.

In this example the box spline $M_{3,3,0}$ is used, which is the uniform floating tensor product biquadratic B-spline. The surface is defined over a four by one rectangular domain by a 6×3 mesh of control points. In Figure 38 the left image is that of the adaptively refined box spline surface, and the right image is an adaptively subdivided *open* tensor product B-spline representation of the same surface.

The surface is designed so that the leftmost square polynomial piece is planar. The second section of surface is linear in v but curved in u. The surface then begins to curve in both directions until it is equally curved in u and v over the last polynomial piece. The subdivision patterns are very similar with both producing a minimum number of polygons where the surface is flat, and many polygons where it is curved. The box spline surface, however, appears to be refined more than the open B-spline in the v direction. This is due to the fact that type-3 surfaces can only be refined in both directions simultaneously, whereas the open B-spline tensor product is subdivided in each direction independently. Thus, at the very first level of refinement, the box spline is divided into four pieces since the surface is not flat, and it is not *completely* linear in either of the two directions. The open tensor product B-spline,

Figure 38: Adaptive refinement vs. subdivision

however, in this case is first subdivided in the u direction into a left and a right half. The left half in this case is the first two polynomial pieces in the original surface, both of which are linear in the v direction. Since this the case, this piece is subsequently only subdivided in the u direction. With the box spline surface the second level of refinement is reached before any surfaces are produced that are completely linear in the v direction. This example shows that although adaptive refinement is not quite as "adaptive" as subdivision, the extra refinement produced is not unreasonable.

5.2.5.2 Five-sided C^1 cubic surface. The next example is a five-sided C^1 cubic surface defined by the box spline $M_{2,2,1}$. The five-sided region Ω is shown in Figure 39 along with the set of control mesh indices I_Ω. In this example the preprocessor divides the surface into three pieces. The first is the rectangular type-3 piece consisting of the four left-most triangular pieces of Ω. A second type-3 surface is defined over the upper right square of Ω and the last piece is a type-2 surface defined by the type-2 triangle in the lower right.

The result of rendering the adaptively refined surface is shown in Figure 40. The surface is designed to be linear in u over the lower left square in Ω, linear in v over the upper right square, planar over the upper left square, and curved in u *and* v over the type-2 triangle in the lower right of Ω. This surface shows various capabilities of the adaptive refinement process. For instance, the planar section of surface is not refined at all. The two sections which are linear in one direction and curved in the other are refined only in the direction in which they are curved.

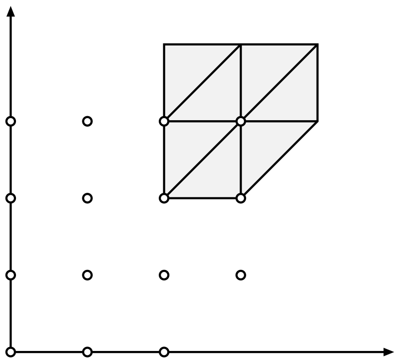

Figure 39: Five-sided C^1 cubic

Figure 40: Five-sided C^1 cubic surface

5.2.5.3 C^0 quadratic surface. The last example is a C^0 cubic surface defined by the box spline $M_{1,1,3}$ over the domain Ω consisting of one unit square. Since this is an example of a surface with a tangent discontinuity in the diagonal direction, the preprocess splits the surface into its two polynomial pieces before the adaptive refinement begins. The results of the adaptive refinement are shown in Figure 41.

This surface is designed such that the type-1 surface is linear in v and the type-2 surface is linear in u. The resulting surface is like the mitered corner of a picture frame, with the diagonal curve being the joint. With this surface, the interesting refinement occurs along the diagonal curve. In the first view the type-1 (red) and type-2 (blue) pieces can be seen along the diagonal curve. These are produced because the preprocess split the original surface along this curve, and many refinements are necessary to produce a smooth approximation to this curve. The type-3 regions that are products of refining triangular pieces can also be seen. At the first level, a large square portion of each original triangular surface is factored out. As further triangular pieces are refined similar but smaller rectangular pieces are produced.

5.2.6 Isoparametric Line Drawing

While standard scan-line or ray tracing methods for rendering surfaces are important, line drawing techniques for real-time vector graphics devices are certainly equally important tools in CAGD systems. Rendering box spline surfaces as refined meshes is not a very efficient tool for this purpose since the convergence rate is slow because of, once again, the floating uniform nature of the box spline. However, as a direct consequence of the ease of open edge curve calculations, isoparametric curve renderings of box spline surfaces can be efficiently produced. It is noted that using the recurrence relation to compute isoparametric curves provides an unacceptable user interface because of the complexity of evaluating box spline basis functions.

There are two possible uses for the edge curve calculation to produce isoparametric renderings. First, an adaptive method can be defined by simply altering the last step of the adaptive refinement algorithm to render the edge curves of refined surfaces instead of converting the surface to polygons for rendering. Secondly, a uniform set of isoparametric curves can be produced by first applying the refinement algorithm with an arbitrary value of m and then

Figure 41: A C^0 cubic surface

using the conversion tables to produce an isoparametric curve along each grid line in the refined domain.

5.2.6.1 Examples. The following two figures show examples of isoparametric line generation. In both examples a uniform set of isoparametric lines are produced in both the u and v directions and only boundary curves are drawn in the diagonal direction. Both surfaces are C^1 cubic box splines. The first surface in Figure 42 is defined over a five-sided region and was previously shown in Figure 40 as an example of adaptive refinement. The second surface, shown in Figure 43, is a six-sided surface.

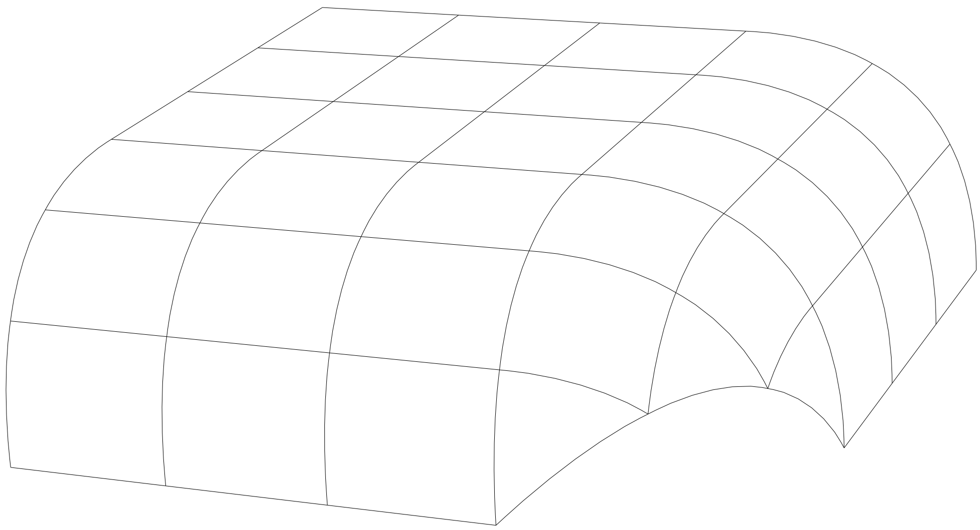

Figure 42: Isoparametric rendering of a five-sided box spline surface

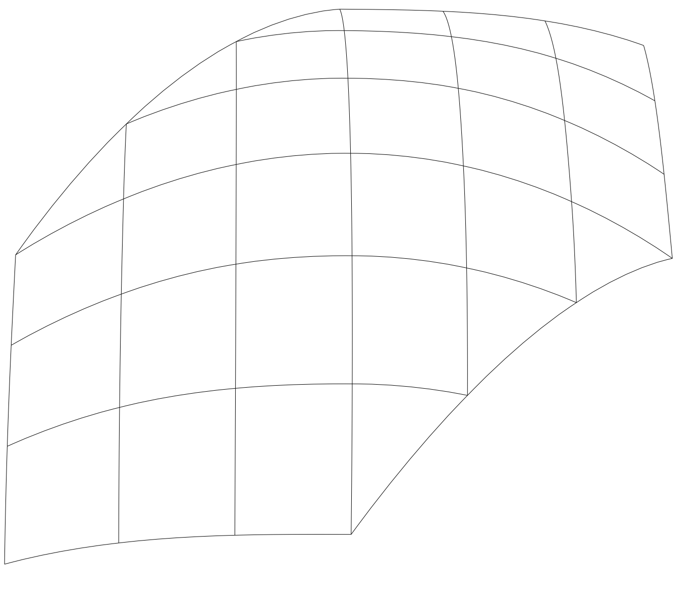

Figure 43: Isoparametric rendering of a six-sided box spline surface

CHAPTER 6

MODELLING WITH BOX SPLINE SURFACES

In this chapter modelling techniques and examples are presented which provide insight into the new capabilities provided by the box spline surface representation. The aim of the thesis is not to replace the tensor product, but to explore the possibility of modelling with this multivariate representation; consequently, all the examples presented will not necessarily be impossible to achieve with the tensor product. Using the tools provided in the previous chapters, we hope to learn more about the nature of the box spline surface in general, and examine new capabilities that might be useful for geometric modelling. With this in mind, it might be interesting to look at models that *can* be produced with tensor products, but produced differently with the box spline, as well as models that cannot be produced with tensor products or can only be produced using degeneracies.

6.1 New Modelling Intuitions

Much of the modelling intuition associated with tensor product formulations is based on open end conditions and the separable nature of tensor product basis functions. There are many new capabilities that must be explored and new intuitions that must be developed in using a multivariate representation. Some of these issues are complicated by the floating nature of the box spline as well as their inherent multivariate nature. Some questions that arise are:

- When is an edge curve linear? When does an edge curve lie in a plane? The shape of the edge curves of a box spline surface is not intuitively obvious from the surface mesh as is the case with open tensor product surfaces.

- What makes a surface linear in one direction? The rows or (columns) of the mesh do not necessarily completely control the shape of the surface in a given direction because of the inherent multivariate nature of the basis functions.

- What makes a surface symmetric in one direction? Symmetric meshes do not necessarily produce symmetric surfaces if the directions of symmetry do not match the basis functions, which in many cases are not symmetric in u or v.

- What are the cross-boundary derivatives along the boundary curves? This, again, is not intuitively obvious from the mesh because of the floating end conditions.
- How can continuity conditions in the diagonal direction be used? This is a new capability introduced by the multivariate nature of the box spline.
- How can nonrectangular surfaces be used? What can and cannot be done with the three, five, or six-sided box spline surface? Can tensor product surfaces be easily attached with continuity or tangent continuity?

These questions raise an important issue: To which problems should the box spline be applied and what approaches should be used? To answer this question we will look at some of the original motivations for using multivariate splines identified in Chapter 1.

6.2 A Hybrid Representation

The major motivation for more powerful representations is the need for nonrectangular regions when modelling so-called anomalous regions or blend surfaces. The ideal would be a representation that is powerful enough to represent any object desired in an intuitive and computationally efficient manner. The box spline, or multivariate spline in general, is currently not in a position to provide this ideal. This being the case, we must ask what type of compromise between the ideal and currently used techniques we can hope to achieve.

Since a single ideal representation has not been determined, it seems a necessity to imbed the box spline surface into a tensor product framework. The are a number of reasons for this. First of all, it is not desirable to replace the tensor product representation with a more complicated and computationally expensive representation when it is not necessary. Secondly, by attaching tensor product surfaces to a box spline surface a hybrid nonrectangular region can be constructed. The tensor product surfaces can be used to provide the open end conditions that the box spline surface lacks and the hybrid surface is then easier to use and can be used to model a wider variety of objects. Lastly, objects are naturally divided into parts which are often modelled or designed separately in practice. A logical place to use the box spline would be the difficult regions where various pieces of a model are put together, which are regions traditionally identified as anomalous regions or blend surfaces in many cases.

6.2.1 Matching Box Spline and Tensor Product B-spline Surfaces

To imbed a box spline within a tensor product framework, the continuity between the box spline surface and the tensor product surfaces must be constructed. The simplest constructions are ones that ensure continuity and tangent continuity between adjoining surfaces. Constructing a tensor product surface, with open end conditions, that is C^0 with a given box spline surface is trivial using the ability to extract edge curves of a box spline surface as open B-spline curves. The converse is not true, however. Constructing a box spline surface that has a given B-spline edge curve is an underconstrained problem because of the floating end conditions. This problem is compounded if a box spline surface is also required to be tangent continuous with a given tensor product surface. This being the case, we will consider the problem of constructing tensor product surfaces, with open end conditions, that match boundary curves and cross-boundary derivatives of a given box spline surface.

6.2.1.1 Derivatives of box spline surfaces. From [11], the directional derivative of the box spline basis function $B_i(x \mid Z) = M_Z(x - i)$ is given as

$$D_z B(x \mid Z) = \sum_{i=1}^{n} \lambda_i \left(B(x \mid Z \setminus \{z^i\}) - B(x - z^i \mid Z \setminus \{z^i\})\right) \tag{6.1}$$

where

$$z = \sum_{i=1}^{n} \lambda_i z^i.$$

From this the directional derivative in one of the knot directions z^i is

$$D_{z^i} B(x \mid Z) = B(x \mid Z \setminus \{z^i\}) - B(x - z^i \mid Z \setminus \{z^i\}).$$

This relationship can now be applied to the box spline surface to compute the partial derivative in one of the knot directions. For the box spline surface $S = \sum_j P_j B_j(x \mid Z)$ (for $Z \subset \mathbf{J}$) the derivative in the direction z^i is

$$D_{z^i} S = \sum (P_j - P_{j - z^i}) B_j(x \mid Z \setminus \{z^i\}).$$

The derivative of the box spline surface is another box spline surface. The derivative (or tangent) surface is of degree n – 3, one less than the degree of the original surface. Now consider the continuity of the tangent surface. For the box spline on the three-direction grid and for r, s, t ≥ 1, without loss of generality let $z^1 = d^1$, $z^2 = d^2$, and $z^3 = d^3$. If z^i is such that max(r',s',t') = max(r,s,t), where r', s', and t' are associated with the derivative surface and r, s, and t are associated with the original surface, the number of degrees of continuity of the derivative surface is also one less than the original. For example, the directional derivative of a C^1 cubic box spline surface based on $M_{2,2,1}$ in the direction z^1 or z^2 is a C^0 quadratic box spline based on $M_{1,2,1}$ or $M_{2,1,1}$ respectively. The derivative with respect to z^3 is a bilinear tangent surface, also a degree two C^0 surface. In this example all three directional derivatives of the C^1 cubic surface result in C^0 degree two tangent surfaces.

Since the tangent surface is a box spline surface, the edge curves can be extracted as open B-spline curves. These curves, considered as vector functions, represent tangent vectors on the original surface. Of these, the ones which are *not* in the direction z^i are cross-boundary derivatives of the original surface. The boundary curve of the original surface (in its open B-spline representation) and the B-spline cross-boundary derivative function across that boundary provide the necessary information for constructing a matching tensor product surface.

Recall that the cross-boundary partial derivative of a tensor product B-spline surface is a B-spline function of the *same* degree and continuity class as the boundary curve itself.* Thus to produce a tangent continuous match between a tensor product surface and box spline surface, the cross-boundary derivative function and the boundary curve of the box spline will have to be defined over the same knot vector. This can be done by first refining the boundary curve, using the Oslo algorithm, so that it has one less degree of continuity (assuming the tangent surface does) and degree raising the cross-boundary derivative function.** Since the matching tensor product surface should be at least C^1 along the boundary curve in question, the box spline surface must be at least the C^2 quartic, $M_{2,2,2}$, to provide such a match. In this

*See de Boor [7, 8] for background on B-spline curves and tensor product surfaces.

**See Cohen, Lyche, and Schumaker [21, 22] for algorithms for degree raising B-splines.

case the boundary curve is a C^2 quartic and the cross-boundary derivatives are C^1 cubic functions. The boundary curve can be refined by inserting knots to be a C^1 quartic and the cross-boundary derivative can be degree raised to also be a C^1 quartic. This means that a tensor product B-spline surface can be constructed that is tangent continuous with a C^2 quartic box spline surface if the tensor product surface is a C^1 quartic in the direction of the matching boundary curve.

6.2.1.2 Generalized cross-boundary derivatives. The above methods construct a tensor product surface that exactly matches a partial derivative of the box spline surface. Such a match is called "strict" C^1 continuity. Since the goal is to surround a box spline surface with tensor product surfaces, such a construction is of limited use. Consider the triangular box spline surface. Suppose we wish to attach a tensor product along the diagonal curve. If the tensor product was constructed so that it matched the derivative with respect to z^1 ($\partial S/\partial u$) or z^2 ($\partial S/\partial v$) this does little more than turn the triangular region back into a rectangular region. Matching partials along the other boundary curves of a triangular surface produce similar results.

A more general derivative of the box spline surface is needed so that the nonrectangular nature is extended when matched by a tensor product. Suppose a box spline surface is defined over a type-2 triangular region. Since the desired cross-boundary derivatives need to be degree raised, a linear blend of partial derivatives might be considered. This would accomplish both raising the degree and providing the generality desired. For example, let $\gamma_1(t)$ be the open B-spline boundary curve of the tangent surface $D_{z^1}S$ along the diagonal boundary. Similarly, let $\gamma_2(t)$ be the diagonal boundary curve of the tangent surface $D_{z^2}S$. Furthermore, let both curves be parametrized for $t \in [0,1]$ where $t = 0$ is the lower left corner of the type-2 region and $t = 1$ is the upper right. Now consider the vector function $R(t) = (t - 1)\,\gamma_2(t) + t\,\gamma_1(t)$. $R(t)$, for any $t \in [0,1]$, is a tangent vector of the surface S since it is a blend of the two partial derivatives. At the endpoints, $R(0)$ is $-D_{z^2}S$ evaluated at the lower left corner of the type-2 region, and $R(1)$ is $D_{z^1}S$ evaluated at the upper right corner of the region. This radial derivative, if matched by an adjoining tensor product surface, would make the tensor product surface an extension of the original triangular box spline surface since one boundary curve of the tensor product would be tangent to and extend the left boundary curve of the box spline and another boundary curve of the tensor product would be tangent to and extend the top boundary curve of the box spline.

If γ_1 and γ_2 are degree n − 3 B-spline curves with r continuous derivatives, R(t) is a degree n − 2 B-spline curve also with r continuous derivatives. Thus, using the linearly blended cross-boundary derivative accomplishes the desired degree raising as well as providing a useful cross-boundary derivative. Since R(t) is no longer strictly a partial derivative, we have sacrificed "strict" C^1 continuity. R(t), however, lies in the tangent plane of the surface S, and therefore we have what is called "visual" C^1 continuity or "tangent plane" continuity, i.e., the tensor product and the box spline have the same tangent plane at every point along the given boundary curve.

To further generalize, let R(t) be the linearly blended cross-boundary derivative defined by

$$R(t) = \lambda_1(t)\, \gamma_1(t) + \lambda_2(t)\, \gamma_2(t),$$

where λ_1 and λ_2 are linear functions and γ_1 and γ_2 are B-spline boundary curves from one of the three tangent surfaces $D_{z^1}S$, $D_{z^2}S$, or $D_{z^3}S$. Given this generalized cross-boundary derivative, an algorithm for constructing a tensor product surface that is tangent plane continuous with a given box spline surface along a boundary is:

Algorithm 1:

1. Extract the desired boundary curve of the box spline surface as an open B-spline curve.
2. Compute the desired generalized cross-boundary derivative along the chosen boundary by computing the tangent surfaces, extracting their boundary curves and computing the linearly blended cross-boundary derivative.
3. Refine the boundary curve of the box spline surface to be on the same knot vector as the generalized cross-boundary derivative.
4. Use these two B-spline curves to construct the first two rows (or columns) of the surface mesh of a matching tensor product B-spline surface.

The above generalized cross-boundary derivative can be used in various ways to attach tensor product surfaces to a box spline surface and examples will be shown later in this chapter. However, this cross-boundary derivative does not allow a box spline surface to be completely surrounded by tensor product surfaces. The reason for this is the mixed partials (the *derivatives* of the cross-boundary derivatives), sometimes called the "twist" vectors.

Consider the corner of a rectangular surface. Suppose two surfaces are constructed which are C^1 with the given surface; one for each of the two boundary curves which meet at the given corner. Now there are three surfaces which meet at the original corner point. To define a fourth surface which is C^1 with both attaching surfaces, the attaching surfaces must be $C^{1,1}$ at the corner, i.e., the derivatives of the cross-boundary derivatives must be equal at the corner.

To consider the corner of a box spline surface is very similar. To completely surround a box spline surface, say a triangular surface, with tensor products, four surfaces will meet at each corner of the original box spline surface. Two of these surfaces are first constructed to match the box spline with visual C^1 continuity along the two boundary curves meeting at the corner. The fourth surface can only be defined if the twists of these two surfaces are equal at the corner. These twist values are the values of the derivatives of the linearly blended cross-boundary derivatives used to construct the first two matching surfaces. It is noted that these values are only required to be equal, they do not necessarily have to be the value of the mixed partial of the box spline surface at the corner.

The linearly blended cross-boundary derivatives can only be used to provide consistent twists in certain cases. For instance, consider the type-2 triangular box spline surface. Figure 44 shows a diagram of how the generalized cross-boundary derivatives might be used to surround the box spline surface with tensor products.

Let R_L and R_T be the cross-boundary derivatives along the left and top boundaries given by

$$R_L(v) = \frac{1}{2}(1 - v)\, D_{z^3}S|_L + v\, D_{z^1}S|_L,$$

and

$$R_T(u) = (1 - u)\, D_{z^2}S|_T + \frac{u}{2}\, D_{z^3}S|_T.$$

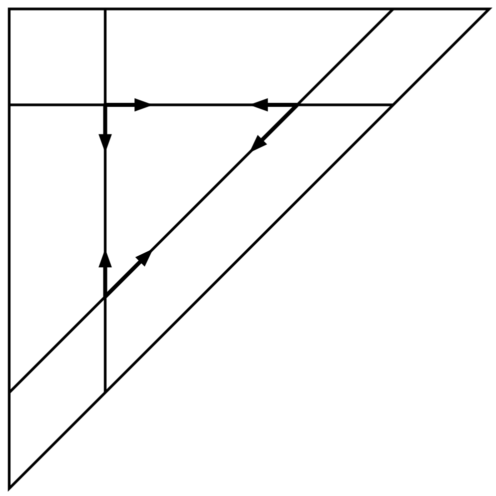

Figure 44: Type-2 region surrounded by tensor products

Here the functions $D_{z^i}S|_L$ and $D_{z^i}S|_T$ denote the left and top boundary curves of the tangent surface $D_{z^i}S$, respectively. Also let R_L and R_T be parametrized such that $R_L(0)$ is the value at the lower left corner, $R_L(1)$ is the value and the upper left corner, $R_T(0)$ is the value at the upper left corner and $R_T(1)$ is the value at the upper right corner of the type-2 region. Now consider the derivatives of R_L and R_T,

$$\frac{d\,R_L}{dv} = \frac{1}{2}(1-v)\,D^2_{z^3z^2}S|_L - \frac{1}{2}D_{z^3}S|_L + v\,D^2_{z^1z^2}S|_L + D_{z^1}|_L,$$

and

$$\frac{d\,R_T}{du} = (1-u)\,D^2_{z^2z^1}S|_T - D_{z^2}S|_T + \frac{u}{2}D^2_{z^3z^1}S|_T + \frac{1}{2}D_{z^3}|_T.$$

These are the twists that a tensor product surface would have if it matched the given cross-boundary derivative. Evaluating these at the corner we have

$$\frac{d\,R_L}{dv}(1) = -\frac{1}{2}D_{z^3}|_c + D^2_{z^1z^2}S|_c + D_{z^1}|_c,$$

and

$$\frac{d\,R_T}{du}(0) = D^2_{z^2z^1}S|_c - D_{z^2}S|_c + \frac{1}{2}D_{z^3}|_c.$$

Here the subscript "c" denotes the evaluation of the given function at the upper left corner in question. Since $z^3 = z^1 + z^2$, from equation (6.1) we have that $D_{z^3}S = D_{z^1}S + D_{z^2}S$. Therefore we have

$$\frac{d\,R_L}{dv}(1) = D^2_{z^1 z^2}S|_c + \frac{1}{2}\,D_{z^1}|_c - \frac{1}{2}\,D_{z^2}|_c = \frac{d\,R_T}{du}(0) = D^2_{z^2 z^1}S|_c + \frac{1}{2}\,D_{z^1}|_c - \frac{1}{2}\,D_{z^2}S|_c.$$

This shows that a third tensor product could be attached at this corner of the triangular box spline surface with C^1 continuity. However, a useful cross-boundary derivative cannot be constructed along the diagonal curve that has consistent twists with R_L and R_T. For example, the desired cross-boundary derivative would be

$$R_D(t) = -\frac{1}{2}\,(1 - t)\,D_{z^2}S|_D + \frac{t}{2}\,D_{z^1}S|_D.$$

The cross-boundary derivative that *is* consistent with R_L and R_T is

$$R_D(t) = \frac{1}{2}\,(1 - t)\,D_{z^2}S|_D + \frac{t}{2}\,D_{z^1}S|_D.$$

Since these two are not scalar multiples the box spline cannot be completely surrounded by tensor product using the linearly blended cross-boundary derivatives. Similar problems occur at the end points of the diagonal curve if a five or six-sided box spline is used.

A solution to this problem would be to use cubicly blended cross-boundary derivatives, where the blending functions λ_1 and λ_2 have zero derivatives at each end. This would make the derivatives of the blended cross-boundary derivatives equal to the twists of the original box spline surface at each corner and allow tensor product surfaces to be consistently defined at any corner. Using this technique would require the ability to multiply a B-spline by a cubic polynomial and the boundary curves of the box spline surface would have to be degree raised to match the degree of the cross-boundary derivatives which would be at least degree six.

6.3 Examples

Examples are now presented of objects modelled with box spline surfaces and combinations of box splines and tensor products using some of the above techniques. In all cases the box spline surfaces were rendered using an implementation of the adaptive refinement algorithm presented in Chapter 5. The first section examines the possibility of using box spline surfaces for the construction of "cap" surfaces without degeneracies. The second section presents examples exploring capabilities of the box spline surface with diagonal derivative discontinuities. Lastly some hybrid modelling examples are presented which use the linearly blended cross-boundary derivatives described in the previous section.

6.3.1 Flat Surfaces without Degeneracies

The simplest box spline surface to be examined might be the flat surface, or "cap." In using a boundary representation to model three-dimensional volumes, cap surfaces (flat surfaces with curved boundaries) are often needed to complete a model that encloses a volume. Various research has been directed at automating this process using tensor product surfaces [18, 54, 33]. Problems occur, however, when the boundary curve of the cap is either a continuous curve (e.g., a disk) or the region is not four-sided. If these regions are modelled with a single tensor product surface, degenerate normals are often required. Donahue [33] presents a technique for automatically constructing caps using multiple surfaces based on the idea of skeleton curves. A simple use for the box spline might be to include them in such cap surfaces. Donahue's technique eliminates degeneracies by using multiple tensor product surfaces that are constructed to be C^0 (these surfaces are visually C^1 because they are all planar). Box spline surfaces could be used to provide alternate parameterizations for such cap surfaces because of their nonrectangular nature.

The following example shows the use of C^1 cubic box spline surfaces (based on $M_{2,2,1}$) to construct a six-sided cap surface without any degeneracies. Figure 45 shows a six-toothed gear-like object. The body of the object is constructed by extruding a profile curve which is a C^0 curve consisting of six pieces. Each piece of the C^0 curve is a C^1 cubic B-spline. To complete the volume model, two cap surfaces are required. Since the body consists of six pieces, it would be natural to use a six-sided cap, or six triangular pieces which is done is this

Figure 45: Cap surface constructed from three-sided surfaces

case. Using tensor products would be somewhat unintuitive because of the symmetry. The isoparametric lines delimit the individual polynomial pieces of the cap surfaces and show there are no degeneracies. The parametrization of the cap is symmetric without any bias towards any particular corner.

6.3.2 Surfaces with Diagonal Discontinuities

In addition to nonrectangular regions, the box spline on the three-direction grid also introduces a third direction of continuity. This means that nontensor product rectangular surfaces can be defined. One possible use of this capability is to have tangent discontinuities in the diagonal direction of rectangular surfaces.

The first example of this type of surface has already been seen in Figure 41 as an example of the adaptive refinement algorithm in Chapter 5. This surface uses the C^0 cubic box spline basis $M_{1,1,3}$. In this example a square surface is constructed out of just two triangular polynomial pieces such that the partial derivative with respect to u of one piece is constant and the partial derivative with respect to v of the second piece is constant and these two vectors are perpendicular. The result is a surface that might be used to represent the miter at the corner of a picture frame. Figure 46 shows a piece of a picture frame constructed in this fashion. Each of the three surfaces shown is a C^0 cubic box spline consisting of two polynomial pieces. The red and blue surfaces extending from the corner are designed so that the tangent discontinuity disappears. The green surface is the same surface previously shown in Chapter 5. The green and the red or the green and the blue surfaces could be constructed as a single rectangular box spline surface, but were defined separately in this example. This is an example of an object that certainly can be easily modelled with tensor product surfaces, but not modelled in this manner.

The next example shows more capabilities of the C^0 box spline surface. In this instance, the C^0 quadratic box spline $M_{1,1,2}$ is used. This surface consists of 72 polynomial pieces on a 6×6 square region and the shaded image is shown in Figure 47. As in the previous example, this surface shows that the discontinuities can be controlled by the geometry of the mesh making the actual discontinuity appear and disappear. In general there could be 11 "ridges" in a surface defined on this parametric range and these tangent discontinuities could run

Figure 46: Picture frame corner

Figure 47: C^0 quadratic box spline

diagonally from a given boundary curve to another. In this case only two of these discontinuities really exist in the geometry and they only exist on the interior of the surface.

The next example combines various capabilities of the box spline surface and is constructed from both box spline and tensor product surfaces. The model is a threaded bolt and is shown in Figure 48. A bolt without threads is easily modelled out of primitive objects (the union of a cylinder and a box), but does not provide a very accurate model. A thread can be modelled with a tensor product surface by sweeping a "V"-shaped cross section along a helical curve. This, however, leaves a piece at each end that needs to be "trimmed."

Since the topology of a threaded bolt is really the topology of a cylinder with diagonal ridges it is candidate to be modelled by a C^0 box spline surface. In this case the C^0 quadratic box spline $M_{1,1,2}$ is used. This model also demonstrates an advantage of floating end conditions. If a surface mesh is "wrapped" around so that the top boundary curve is equal to the bottom boundary curve, the surface is periodic in the v direction. Thus the surface floats within the mesh with no distinguished boundary curves in the u direction. Using the periodic properity plus the fact the the tangent discontinuities can be controlled by the geometry, the shaft of the bolt is constructed as a single periodic box spline surface such that the boundary curve at each end is C^1 and only part of the shaft is threaded. This provides an accurate model of a threaded bolt that is easily capped at the ends.

The bolt head and cap surfaces are constructed from the shaft surface. The edge curve at the bolt head end of the shaft is extracted and a C^0 biquadratic tensor product surface is used to model all of the bolt head except a six-sided linear cap that is required. This is provided by a six-sided piecewise linear box spline. The other end of the shaft is similarly capped with a C^0 biquadratic and a linear box spline. Figure 49 shows an exploded view of the bolt model using isoparametric lines. These surfaces form a valid volume model of the bolt upon which mass property calculations, for instance, could be performed.

6.3.3 C^1 Surfaces

This section presents an example using the techniques presented earlier in this chapter to construct tangent plane continuity between box spline and tensor product surfaces. As previously mentioned, this technique cannot be used to completely surround a box spline

Figure 48: C^0 quadratic box spline bolt

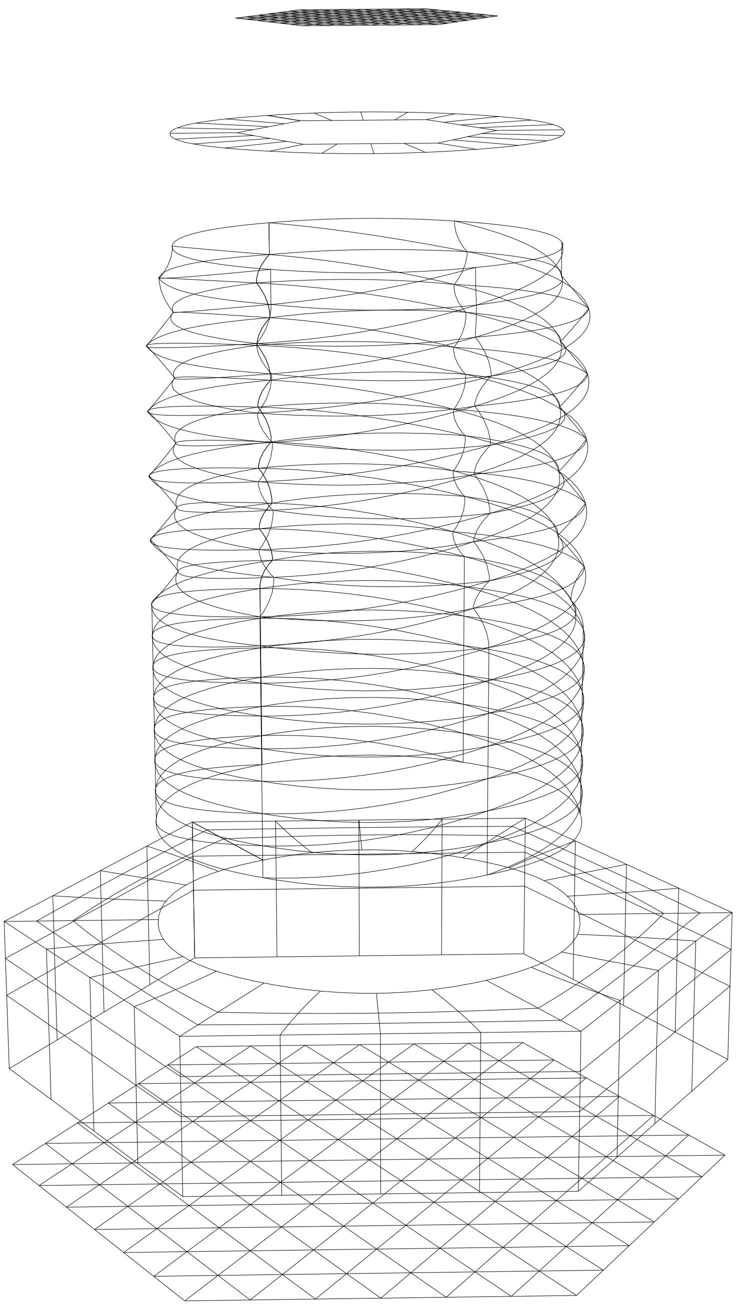

Figure 49: Exploded view of bolt

surface with tensor products, but since it can be used to attach tensor products to arbitrarily refined box splines along certain edge curves some interesting examples can be generated.

6.3.3.1 Five-sided surfaces. The traditional example of the airplane wing and fuselage is an example of the blend surface. This type of region occurs whenever two surfaces which have the topology of a cylinder are required to join smoothly. A five-sided region is typically needed to accomplish a smooth transition from the inherently rectangular parametrization of one surface to the different rectangular parametrization of the other.

This example uses a five-sided box spline surface with attaching tensor product surfaces to produce a hybrid five-sided surface to be used for such blends. Figure 50 shows a diagram of how tensor products can be attached to a five-sided region of the three-direction grid using the generalized cross-boundary derivatives discussed earlier.

The C^2 quartic box spline $M_{2,2,2}$ is used to model a region like the blend between an airplane wing and fuselage. Four five-sided box splines and four attaching tensor products are used. The edge curves where the matches occur are C^1 quartic B-spline curves. Using the algorithm previously discussed, the edge curves are first refined to be C^1 quartics and the linearly blended cross-boundary derivatives are computed. In this case the tensor product surfaces are C^2 quartics in the v direction. The degree and knot vector in the v direction can be chosen arbitrarily since they only contribute a constant factor to the matching algorithm. Figure 51 shows the resulting object and Figure 52 shows the box splines as the orange surfaces and the tensor products as the purple surfaces.

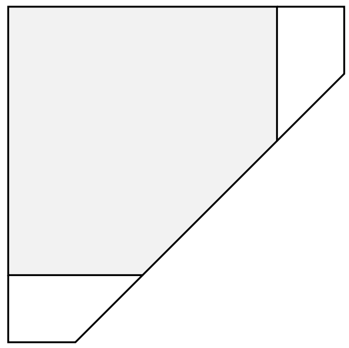

Figure 50: Parametric view of five-sided region with two attaching tensor products

Figure 51: Blend region using five-sided box splines

Figure 52: Composition of blend region

CHAPTER 7

CONCLUSIONS

The aim of the thesis was to consider the application of a multivariate spline representation to geometric modelling. A candidate multivariate spline was chosen and investigations into an experimental modelling system were performed. In doing so, a number of goals have been accomplished.

The first result is that computational algorithms have been defined making the use of a multivariate representation for geometric modelling possible. Although a relatively simple form of the multivariate spline was chosen, the problems associated with providing the basic computational algorithms necessary were still significant. The multivariate nature of the representation complicates many issues that are simple in a tensor product setting. Although it is not clear if other multivariate representations will become computationally feasible for geometric modelling, the experience gained by this application may be valuable in future research.

Secondly, a new inherently multivariate representation has been used for geometric modelling. The generality gained by the use of the box spline surface should be compared to the uniform floating tensor product B-spline, which is a special case. From the examples it seems clear that the box spline provides a significant increase in generality over the uniform tensor product. It is noted that historically the uniform B-spline was studied for many years before further theoretical results and experience permitted the use of nonuniform B-splines, which still are not widely used in industry. Even after computation of nonuniform tensor products became feasible it was a number of years before practice and experience permitted the modelling of geometrically complex objects such as turbine blades. It seems reasonable for the development of multivariate representations to parallel that of tensor product ones.

The last result is that a testbed for further research has been developed. Multivariate splines by nature are complicated functions. The ability to interactively construct surfaces and produce color shaded images provides an important foundation for conducting research into further modelling techniques and can also be used to provide insight and ideas for further theoretical research.

7.1 Future Research

To extend these results, one might consider box spline surfaces on a more general parametric grid. Although all the computational results presented in the thesis are based on box splines on the three-direction grid, many of the results can be extended to more general regular grids. As the computational results are extended to box splines on more general grids, more modelling capabilities will be introduced and will also need to be studied.

Another area which has only begun to be studied is the use of periodic box spline surfaces. This property might be used in combination with other properties of the box spline, such as their nonrectangular nature, to provide interesting modelling capabilities. More work must also be done on integrating box spline surfaces with tensor products.

Certainly much work remains to be done in providing tools and experience for modelling complex blend surfaces and other anomalous regions with multivariate splines. This thesis only begins to examine these issues, but with a computational base to build on, more research in this area can be carried out.

REFERENCES

1. Bamberger, L., "Zweidimensionale Splines Auf Regularen Triangulationen," Ph.D. dissertation, University of Munich (Munchen), May 1985.

2. Barnhill, R. E., "Smooth Interpolation over Triangles," in *Computer Aided Geometric Design*, Robert E. Barnhill and Richard F. Riesenfeld, eds., Academic Press, New York, 1974, pp. 45-70.

3. Barnhill, R. E., "Representation and Approximation of Surfaces," in *Mathematical Software III*, John R. Rice, ed., Academic Press, New York, 1977, pp. 68-119.

4. Bézier, P. E., "Mathematical and Practical Possibilities of UNISURF," in *Computer Aided Geometric Design*, Robert E. Barnhill and Richard F. Riesenfeld, eds., Academic Press, New York, 1974.

5. Boehm, W., "Triangular Spline Algorithms," *Computer Aided Geometric Design*, Vol. 2, No. 1-3, September 1985, pp. 61-67.

6. Boehm, W., "Subdividing Multivariate Splines," *Computer-Aided Design*, Vol. 15, No. 6, November 1983, pp. 345-352.

7. de Boor, C., "Splines as Linear Combinations of B-Splines. A Survey," in *Approximation Theory II*, G. G. Lorentz, C. K. Chui, and L. L. Schumaker, eds., Academic Press, New York, 1976, pp. 1-47.

8. de Boor, C., *A Practical Guide to Splines*, Springer-Verlag, Applied Mathematical Sciences, Vol. 27, 1978.

9. de Boor, C. and Hoellig, K., "Bivariate Box Splines and Smooth pp Functions on a Three Directional Mesh," *Journal of Computational and Applied Mathematics*, Vol. 9, 1983, pp. 13-28.

10. de Boor, C. and Devore, R., *Approximation by Smooth Multivariate Splines*, to appear - MRC TSR 2319, 1981, Trans. Amer. Math. Soc.

11. de Boor, C. and Hoellig, K., "B-Splines from Parallelepipeds," *Journal d'Analyse Mathematique*, Vol. 42, 1982/83, pp. 99-115.

12. de Boor, C. and Hoellig, K., "Recurrence Relations for Multivariate B-Splines," *Proceedings of the American Mathematical Society*, American Mathematical Society, July 1982, pp. 397-400.

13. Catmull, E. E. and Clark, J. H., "Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes," *Computer-Aided Design*, Vol. 10, No. 6, November 1978, pp. 350-355.

14. Chui, C. K. and Schumaker, L. L., "On Spaces of Piecewise Polynomials with Boundary Conditions I. Rectangles," in *Multivariate Approximation Theory II*, W. Schempp and K. Zeller, eds., Birkhauser Verlag, Basel, 1982, pp. 69-80.

15. Chui, C. K., Schumaker, L. L. and Wang, R. H., "On Spaces of Piecewise Polynomials

with Boundary Conditions II. Type-1 Triangulations," *Conference Proceedings*, Canadian Mathematical Society, 1983, pp. 51-66.

16. Chui, C. K., Schumaker, L. L. and Wang, R. H., "On Spaces of Piecewise Polynomials with Boundary Conditions III. Type-2 Triangulations," *Conference Proceedings*, Canadian Mathematical Society, 1983, pp. 67-80.

17. Chui, C. K. and Wang, R. H., "On A Bivariate B-Spline Basis," Tech. report CAT 7, Center For Applied Theory, Texas A&M Univ., 1981.

18. Cobb, E. S., "Design of Sculptured Surfaces Using the B-Spline Representation," Ph.D. dissertation, University of Utah, June 1984.

19. Cohen, E., Lyche, T. and Riesenfeld, R., "Discrete Box Splines and Refinement Algorithms," *Computer Aided Geometric Design*, Vol. 1, No. 2, 1984, pp. 131-148.

20. Cohen, E., Lyche, T. and Riesenfeld, R. F., "Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics," *Computer Graphics and Image Processing*, Vol. 14, No. 2, October 1980, pp. 87-111, also Tech. Report No. UUCS-79-117, Department of Computer Science, University of Utah, October 1979.

21. Cohen, E., Lyche, T. and Schumaker L. L., "Degree Raising of Splines," *Journal of Approximation Theory*, to appear.

22. Cohen, E., Lyche, T. and Schumaker L. L., "Algorithms for Degree Raising of Splines," *ACM Transactions of Graphics*, to appear.

23. E. Cohen and R. F. Riesenfeld, eds., *Alpha_1: Selected Research*, unpublished manuscript, Department of Computer Science, University of Utah.

24. Curry, H. B. and Schoenberg, I. J., "On Polya Frequency Functions IV: The Fundamental Spline Functions and their Limits," *Journal d'Analyse Mathematique*, Vol. 17, 1966, pp. 71-107.

25. Dahmen, W., "On Multivariate B-Splines," *SIAM Journal on Numerical Analysis*, Vol. 17, No. 2, April 1980, pp. 179-191.

26. Dahmen, W., "Approximation by Linear Combinations of Multivariate B-Splines," *Journal of Approximation Theory*, Vol. 31, 1981, pp. 299-324.

27. Dahmen, W. and Micchelli, C. A., "Multivariate Splines - A New Constructive Approach," in *Surfaces in Computer Aided Geometric Design*, Robert E. Barnhill and Wolfgang Boehm, eds., North-Holland, Amsterdam, 1983, pp. 191-215.

28. Dahmen, W. and Micchelli, C.A., "On Linear Independence of Multivariate B-Splines II. Complete Configurations," *Mathematics of Computation*, Vol. 41, No. 163, July 1983, pp. 143-163.

29. Dahmen, W., "Multivariate B-Splines-Recurrence Relations and Linear Combinations of Truncated Powers," in *Multivariate Approximation Theory*, W. Schempp and K. Zeller, eds., Birkhauser Verlag, Basel, 1979, pp. 64-82, proceedings of the conference held at Mathematical Research Ints. at Oberwolfach, Feb. 1979.

30. Dahmen, W. and Micchelli, C.A., "Subdivision Algorithms for the Generation of Box Spline Surfaces.," *Computer Aided Geometric Design*, Vol. 1, 1984, pp. 115-129.

31. Dahmen, W. and Micchelli, C.A., "Line Average Algorithm: A Method for the Computer Generation of Smooth Surfaces.," *Comuter Aided Geometric Design*, Vol. 2, No.

1-3, September 1985, pp. 77-85.

32. Davis, P. J, *Interpolation and Approximation*, Dover, New York, 1975, originally published by Ginn-Blaisdell, 1963.

33. Donahue, B., "Modelling Complex Objects with Generalized Sweeps," Master's thesis, University of Utah, December 1985.

34. Doo, D. W. H. and Sabin, M. A., "Behaviour of Recursive Division Surfaces Near Extraordinary Points," *Computer-Aided Design*, Vol. 10, No. 6, November 1978, pp. 356-360.

35. Farin, G., "A Construction for Visual C^1 Continuity of Polynomial Surface Patches," *Computer Graphics and Image Processing*, Vol. 20, 1982, pp. 272-282.

36. Farin, G., "Bézier Polynomials over Triangles and the Construction of Piecewise C^r Polynomials," Tech. report 91, Department of Mathematics, Brunel University, 1980.

37. Forrest, A. R., "Computational Geometry -- Achievements and Problems," in *Computer Aided Geometric Design*, Robert E. Barnhill and Richard F. Riesenfeld, eds., Academic Press, New York, 1974.

38. Forrest, A. R., "A Unified Approach to Geometric Modelling," *Proceedings of SIGGRAPH '78*, ACM, August 1978, pp. 264-269.

39. Goldman, R., "Subdivision Algorithms for Bézier Triangles," *Computer-Aided Design*, Vol. 15, No. 3, May 1983, pp. 159-166.

40. Gordon, W. J. and Riesenfeld, R. F., "Bernstein-Bézier Methods for Computer-Aided Design or Free Form Curves and Surfaces," *Journal of the ACM*, Vol. 21, No. 2, April 1974, pp. 293-310.

41. Gregory, J. A., "C^1 Rectangular and Nonrectangular Surface Patches," in *Surfaces in Computer Aided Geometric Design*, Robert E. Barnhill and Wolfgang Boehm, eds., North-Holland, Amsterdam, 1983, pp. 25-33.

42. Kochevar, P. D., "A Multidimensional Analogue of Shoenberg's Spline Approximation," Master's thesis, University of Utah, August 1982.

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

44. Micchelli, C. A., "On a Numerically Efficient Method for Computing Multivariate B-Splines," in *Multivariate Aproximation Theory, ISNM 30*, Walter Schempp and Karl Zeller, eds., Birkhaeuser Verlag, Basel, 1979, pp. 211-248, also Research Report No. RC 7716 (#33432), IBM T. J. Watson Research Center, Yorktown Heights, N.Y.

45. Micchelli, C. A., "A Constructive Approach to Kergin Interpolation in $\mathbf{R}^k$: Multivariate B-Splines and Lagrange Interpolation," *Rocky Mountain Journal of Mathematics*, Vol. 10, 1980, pp. 485-497, also Math. Res. Center Report No. 1895, University of Wisconsin, Madison.

46. Middleditch, A. E. and Sears, K. H., "Blend Surfaces for Set Theoretic Volume Modelling Systems," *SIGGRAPH '85 Conference Proceedings*, Brian A. Barsky, ed., SIGGRAPH, July 1985, pp. 161-170.

47. Prautzsch, H., "Unterteilungsalgorithmen fur Multivariate Splines - Ein Geometrischer Zugang," Ph.D. dissertation, Technische Universitat Braunschweig, 1984.

48. Prautzsch, H., "Generalized Subdivision and Convergence," *Computer Aided Geometric Design*, Vol. 2, No. 1-3, September 1985, pp. 69-75.

49. Requicha, A. A. G. and Rossignac, J. R., "Constant-Radius Blending in Solid Modeling," *Computers in Mechanical Engineering*, July 1984, pp. 65-73.

50. Requicha, A. A. G. and Voelker, H. B., "Solid Modeling: A Historical Summary and Contemporary Assessment," *IEEE Computer Graphics and Applications*, Vol. 2, No. 2, March 1982, pp. 9-14.

51. Riesenfeld, R. F., "Applications of B-Spline Approximation to Geometric Problems of Computer-Aided Design," Ph.D. dissertation, Syracuse University, May 1973, available as Tech. Report No. UTEC-CSc-73-126, Department of Computer Science, University of Utah.

52. Riesenfeld, R. F., Cohen, E., Fish, R. D., Thomas, S. W., Cobb, E. S., Barsky, B. A., Schweitzer, D. L. and Lane, J. M., "Using the Oslo Algorithm as a Basis for CAD/CAM Geometric Modelling," *Proceedings of the Second Annual Conference of the NCGA*, R. T. Aangeenbrug, ed., National Computer Graphics Association, Inc., Baltimore, 14-18 June 1981.

53. Thomas, S. W., "Modelling Volumes Bounded by B-Spline Surfaces," Ph.D. dissertation, University of Utah, June 1984.

54. Stay, P. R., "Rounded Edge Primitives and Their Use in Computer Aided Geometric Design," Master’s thesis, University of Utah, August 1984.