Smooth Subdivision Surfaces Based on Triangles

by

Charles Teorell Loop

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

Master of Science

Department of Mathematics

The University of Utah

August 1987

Abstract

An algorithm for generating a smooth surface from an irregular mesh of triangles is presented. The method is based on a recursive subdivision process that refines the mesh into a piecewise linear approximation of a smooth surface. The rules which govern the mesh refinement are based on well known properties of B-spline curves as well as more recent results from multivariate spline theory. A careful analysis of the smoothness of the resulting surface is made. This analysis reveals that a surface generated by this method is curvature continuous except at a fixed number of extraordinary points corresponding to mesh vertices. At these points it is found that the surface has a well defined tangent plane. An explicit formulation of this tangent plane is given and the issue of curvature continuity at these points is also discussed. Finally, a pseudo code implementation of the algorithm is given which allows the surface to be treated as a collection of individual patches.

Contents

List of Figures

Acknowledgment

I have received inspiration, encouragement, and countless hours of consultation from many generous and talented people. My deepest thanks go to individuals associated with the Mathematics department at the University of Utah: my advisors Robert Barnhill and Gerald Farin; Paul Arner, Bruce Piper, and Mark Watkins. Others whose help has been invaluable include: Tom Jensen, Allyn Rockwood, and Tony De Rose.

Chapter 1

Introduction

The purpose of this thesis is to present an algorithm that allows a designer to create a sculptured smooth surface. This surface is defined and manipulated by a structured set of *control points*. The shape of the surface is determined by the positions of the control points in space. A designer need only understand the relationship between the control points and the surface, and not the mathematics of the underlying implementation. This is one of the fundamental paradigms of Computer Aided Geometric Design (CAGD) [Boehm, Farin, and Kahmann 1984].

Many techniques have been developed that exhibit this behavior. At issue in this thesis is underlying structure of the control points. Most existing surface design schemes require that the control points take on a *regular* structure. This shall be described shortly. A regular structure can be quite restrictive from a design point of view. This thesis will present a method for designing a smooth surface that is not encumbered by this restriction, thereby giving a designer more freedom to create complex shapes.

In most design schemes the underlying shape is represented in a piecewise fashion by a polynomial basis. Piecewise polynomial means that a curve or surface is represented by a collection of individual polynomial *segments* or *patches*. Control point schemes are generally either interpolating or approximating. The choice of which scheme to use depends upon the application. For all applications, the scheme

should be *coordinate free*. This means that the relationship between the control points and the shape is independent of any coordinate system. The shape defined by an interpolating scheme will pass through all of the control points. This type of scheme is well suited for *representation*, i.e., if the control points are known to belong to an existing shape. Interpolating methods may not be the natural choice for design because they generally lack *local control*. This means that modifying the shape by moving a control point will affect the entire shape. Local control is important from a design point of view. A designer wants to be sure that moving a control point is not going to affect some part of the shape that is already deemed correct. In order to obtain local control, interpolation is sacrificed. This leaves schemes which approximate the control points.

Several such approximating schemes exist; the simplest and best known are B-splines (B for basic). The control points of a B-spline scheme are known as *de Boor points*. For B-spline curves, the de Boor points are an ordered set that forms the *de Boor polygon*. These points can be specified by a designer. The resulting curve is a smooth approximation to the de Boor polygon. Figure 1.1 illustrates the relationship between the de Boor points and a B-spline curve. B-spline schemes exhibit the important local control property. The arrow in Figure 1.1 represents the displacement of a single de Boor point, the local effect on the curve has been shaded.

The classic approach of extending B-spline curves to surfaces involves taking a *tensor product* of B-spline curves. This can loosely be described as taking a series of parallel curves in one direction, and *sweeping* these with another curve in the perpendicular direction to form a surface. Similarly, a series of parallel de Boor polygons may be connected in the perpendicular direction to form a *de Boor net*. From this construction, the de Boor net takes on a *rectangular* structure. Figure 1.2 illustrates the de Boor net and resulting tensor product B-spline surface. The

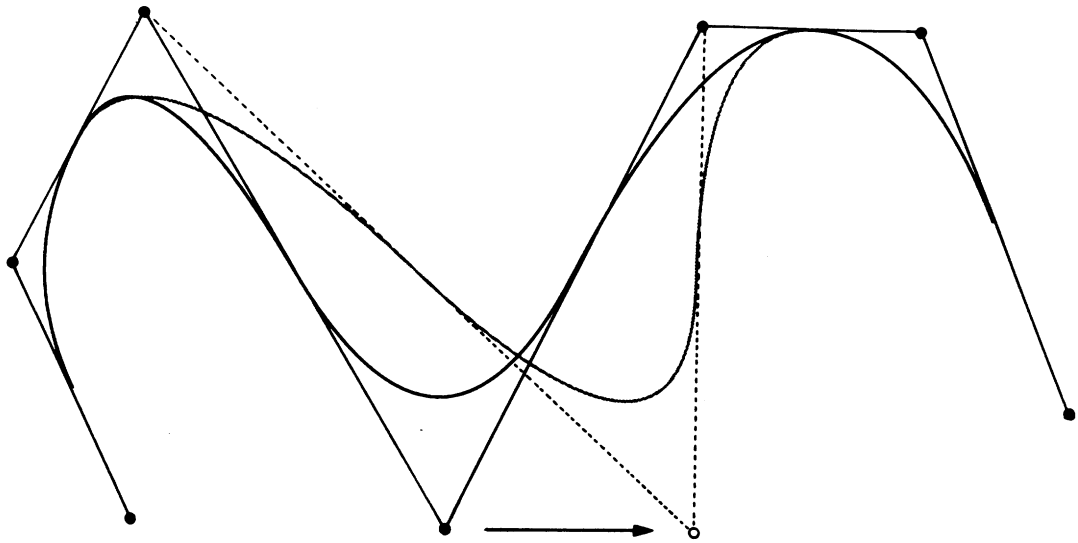


Figure 1.1: A B-spline curve.

tensor product B-spline scheme has local control and produces a smooth surface that approximates the de Boor net.

A more recent addition to the class of smooth, locally controlled, approximating surface schemes are *triangular splines*. Triangular spline surfaces behave much like tensor product B-spline surfaces. A triangular spline surface is defined by a de Boor net that has a regular *triangular* structure. Such a surface is illustrated in Figure 1.3. The properties of triangular splines are of special interest in this thesis. The surface design scheme to be presented is a generalization of triangular splines, where the de Boor net need not be regular.

A variety of techniques exist for computing a representation of the underlying shape of a B-spline scheme. Generally one explicitly evaluates the underlying piecewise polynomials a sufficient number of times to obtain a piecewise linear approximation to the curve or surface. B-spline schemes offer an interesting alternative. Each curve segment may be reparameterized as two or more sub-segments.

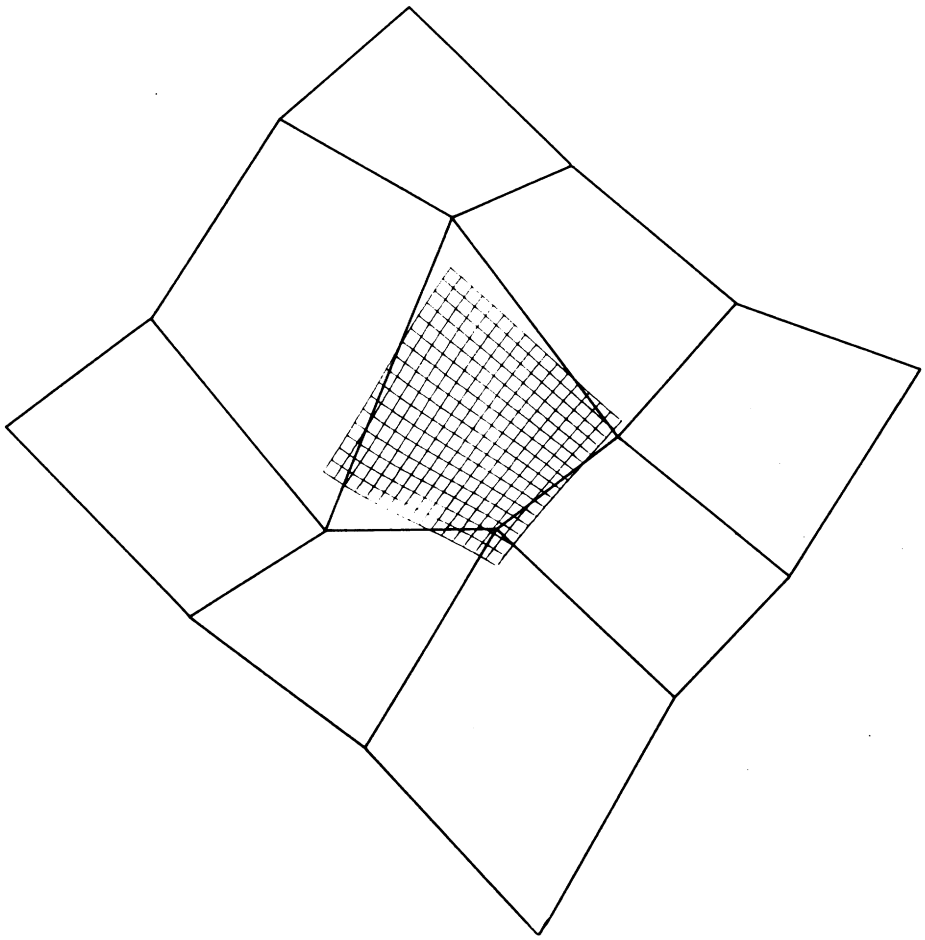


Figure 1.2: A tensor product B-spline surface and associated de Boor net.

This reparameterization has the effect of *refining* the de Boor polygon. The refined de Boor polygon is a denser collection of points that is in some sense "closer" to the underlying curve than the original de Boor polygon. Such a reparameterization or ***subdivision*** of the polynomial segments may be done repeatedly, i.e., each sub-segment may be subdivided. After repeated subdivision, the refined de Boor polygon is indistinguishable from the B-spline curve. This same principle may be applied to a surface, resulting in a refined de Boor net that is indistinguishable from the surface. In this thesis, only *binary* subdivision is considered. For curves, this means each polynomial segment is reparameterized as two sub-segments. For

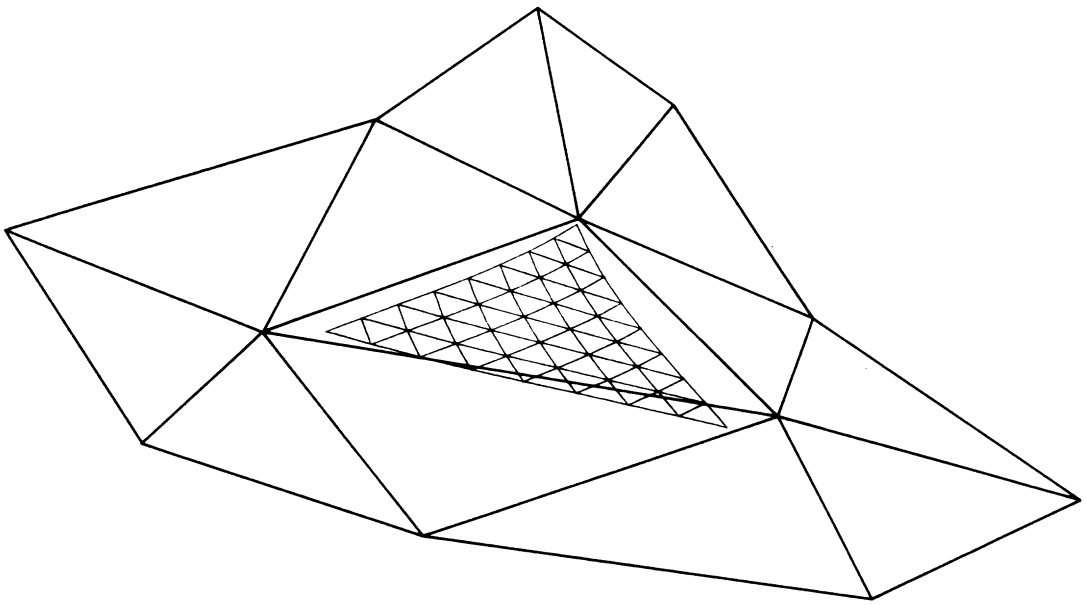


Figure 1.3: A triangular spline surface and associated de Boor net.

surfaces, each polynomial patch is subdivided into four sub-patches.

In principle, refinement of the de Boor net is a *geometric* operation. A set of geometric rules based on the regular structure of the de Boor net governs the refinement process. This geometric approach to subdivision will be adapted to work for irregular or arbitrary control point sets. Such extensions of B-splines to arbitrary topologies are not new. Extending the geometric properties of tensor product B-splines has been considered [Doo and Sabin 1978][Catmull and Clark 1978]. Extending the subdivision rules for triangular splines will be considered in this thesis.

Chapter 2

Binary Subdivision of B-splines

In this chapter, B-spline curves, Tensor product B-spline surfaces, and Triangular spline surfaces will be discussed. Fundamental concepts, properties, and formulas are presented for each. Special attention is paid to deriving *Binary* subdivision formulas.

2.1 Univariate B-splines

A degree r, piecewise polynomial B-spline curve $S^r(u)$ is defined:

$$S^r(u) = \sum_i d_i^r N_i^r(u). \tag{2.1}$$

The vector valued coefficients d_i^r form the *de Boor polygon*. The $N_i^r(u)$ are *normalized* B-splines of degree r defined over a sequence of knots i . This knot sequence forms a partition of the real u-axis. Only the case of equidistant knot spacing is discussed. For simplicity, knot sequences will be derived from the sequence

$$\mathcal{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}.$$

Under these restrictions, (2.1) becomes

$$S^r(u) = \sum_{i \in \mathcal{Z}} d_i^r N^r(u - i). \tag{2.2}$$

The $N^r(u - i)$ can be viewed as translates of a single B-spline $N^r(u)$. $N^r(u)$ has the following properties:

- Partition of unity : $\sum_{i \in \mathcal{Z}} N^r(u-i) = 1$

- Positivity : $N^r(u) \geq 0$

- Local support : $N^r(u-i) = 0 \quad \text{if } u \notin [i, i+r+1]$

- Continuity : $N^r(u)$ is $(r-1)$ times continuously differentiable

- Recursion : $N^r(u-i) = \frac{u-i}{r} N^{r-1}(u-i) + \frac{i+r+1-u}{r} N^{r-1}(u-i-1)$

  where $\quad N^0(u) = \begin{cases} 0, & \text{if } u \in [i, i+1] \\ 1, & \text{otherwise.} \end{cases}$

Figure 2.1 illustrates some examples.

The first three properties imply the convex hull property, i.e., for any u , $S^r(u)$ is a convex combination of a local subset of the de Boor points d_i^r . This local nature means that a change in a de Boor point will affect only a small portion of the curve. The local control property defines $[i, i+r+1]$ as the *support* of $N^r(u-i)$. The $N^r(u)$ form a basis for degree r polynomials, i.e., any univariate degree r polynomial is a linear combination of B-splines. The important recursion formula, independently discovered by Cox, de Boor, and Mansfield [de Boor 1978] shows that a B-spline of degree $r > 0$ is a linear blend of lower degree B-splines. It also provides a stable and efficient means of evaluating $S^r(u)$.

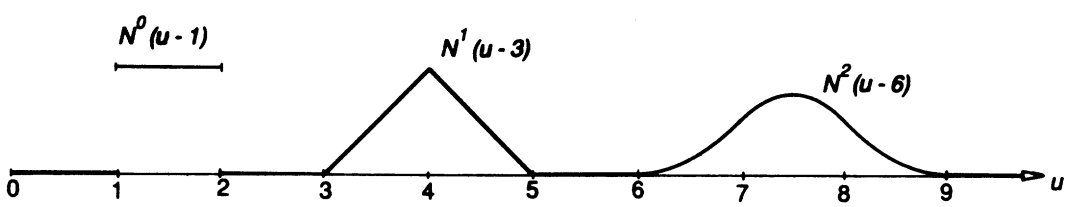


Figure 2.1: Examples of univariate B-splines

2.2 Subdivision of B-spline curves

Subdivision of the de Boor polygon results in a refined polygon that more closely approximates the shape of the underlying curve. It is well known [Riesenfeld 1975] that under repeated subdivision, the de Boor polygon will converge to the underlying curve.

The idea behind a subdivision scheme is to rewrite the curve (2.2), as a curve over a refined knot sequence. Binary refinement of the sequence $\mathcal{Z}$ results in the sequence

$$\mathcal{Z}/2 = \{\ldots, -1, -\tfrac{1}{2}, 0, \tfrac{1}{2}, 1, \ldots\}.$$

From this, (2.2) becomes

$$S^r(u) = \sum_{j \in \mathcal{Z}/2} \hat{d}_j^r N^r(2(u-j)) \tag{2.3}$$

where the $\hat{d}_j^r$ make up the refined de Boor polygon. Note that the support of $N^r(2u)$ is half that of $N^r(u)$.

It is possible to determine the $\hat{d}_j^r$ by considering the subdivision of a single B-spline. A single B-spline may be *decomposed* into similar B-splines of half the support, as in Figure 2.2. This is accomplished algebraically by rewriting $N^r(u)$ as a linear combination of B-splines of degree r over the refined knot vector. This results in

$$N^r(u) = \sum_{j \in \mathcal{Z}/2} c_j^r N^r(2(u-j)). \tag{2.4}$$

The c_j^r are the special case $\hat{d}_j^r$ for a single B-spline. Each c_j^r is a function of r and j, which may be written

$$c_j^r = 2^{-r}\binom{r+1}{2j}. \tag{2.5}$$

Proof that this formula is correct may be found by induction on r, using the B-spline recursion formula to relate B-splines of different degrees.

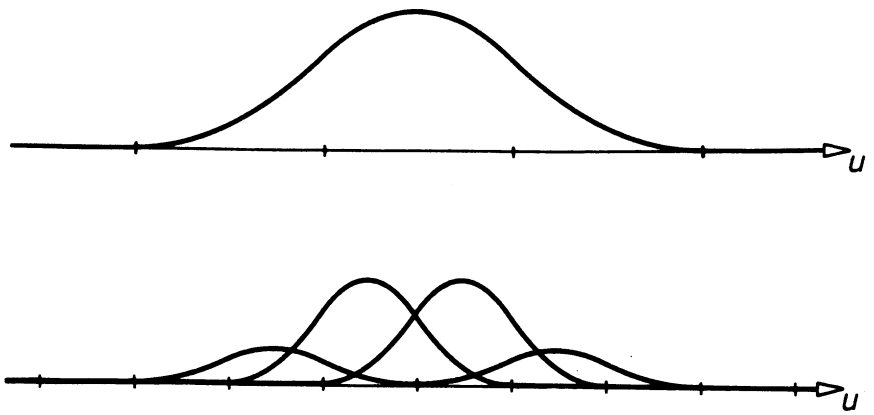


Figure 2.2: A single B-spline decomposed into similar B-splines of half the support.

The process of subdividing an entire curve follows. The translated B-splines $N^r(u-i)$ are subdivided

$$N^r(u-i) = \sum_{j \in \mathcal{Z}/2} c^r_{j-i} N^r(2(u-j)). \tag{2.6}$$

This is substituted into (2.1) to give

$$S^r(u) = \sum_{i \in \mathcal{Z}} d^r_i \sum_{j \in \mathcal{Z}/2} c^r_{j-i} N^r(2(u-j)). \tag{2.7}$$

Rearranging the order of summation gives

$$S^r(u) = \sum_{j \in \mathcal{Z}/2} \sum_{i \in \mathcal{Z}} c^r_{j-i} d^r_i N^r(2(u-j)). \tag{2.8}$$

It then follows that

$$\hat{d}^r_j = \sum_{i \in \mathcal{Z}} c^r_{j-i} d^r_i. \tag{2.9}$$

The importance of (2.9) is best understood by example. It is used to compute points of the refined de Boor polygon. For example, if $r = 2$ then

$$\begin{aligned} \hat{d}^2_0 &= \sum_{i \in \mathcal{Z}} c^2_{-i} d^2_i \\ &= \ldots + (0)d^2_{-2} + (\tfrac{3}{4})d^2_{-1} + (\tfrac{1}{4})d^2_0 + (0)d^2_1 + \ldots \end{aligned}$$

and similarly

$$\hat{d}^2_{\frac{1}{2}} = \ldots + (0)d^2_{-2} + (\tfrac{1}{4})d^2_{-1} + (\tfrac{3}{4})d^2_0 + (0)d^2_1 + \ldots \ .$$

Note that the $\hat{d}^r_j$ are a convex combination of only a few of the d^r_i. For $r = 2$ the refined de Boor points are created on edge segments of the de Boor polygon at a ratio of $1 : 3$, and $3 : 1$ from each end point. These ratios of *weights* are referred to as subdivision *masks*. Masks are important because they imply a *geometric construction* algorithm. The masks can be thought of as being *applied* to the de Boor polygon to create new points. The set of all such new points forms the refined de Boor polygon. This procedure may be repeated until the refined de Boor polygon is sufficiently close to the underlying curve. Figure 2.3 illustrates two iterations of a geometric construction algorithm.

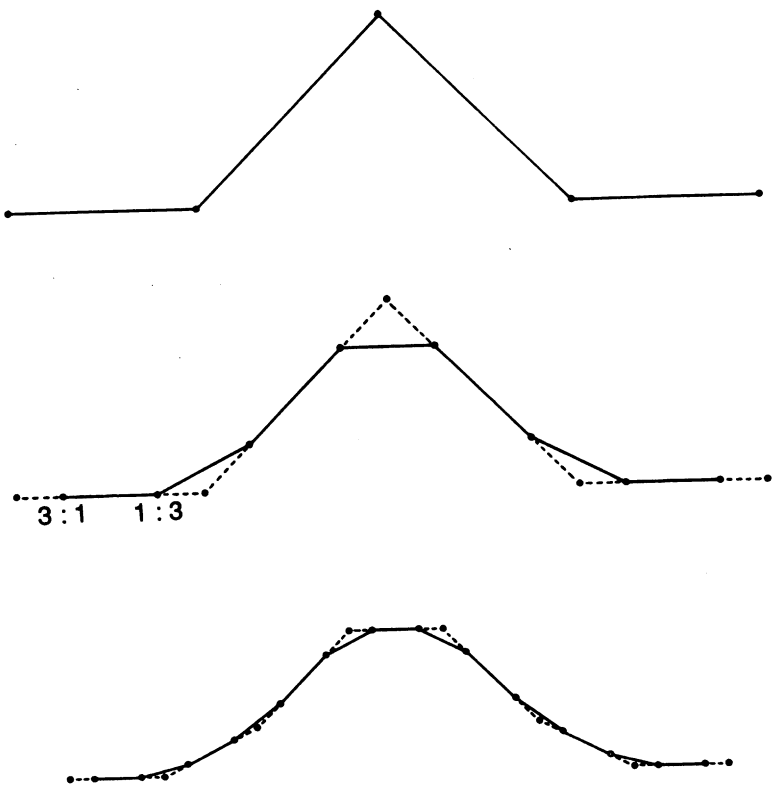


Figure 2.3: Construction of the new de Boor points from the old.

2.3 Tensor Product B-spline Surfaces

A piecewise polynomial tensor product B-spline surface $S^{r,s}(\mathbf{u})$ is defined

$$S^{r,s}(\mathbf{u}) = \sum_{\mathbf{i}\in\mathcal{Z}^2} d_{\mathbf{i}}^{r,s} N^{r,s}(\mathbf{u}-\mathbf{i}) \tag{2.10}$$

where $\mathbf{u} \in \mathcal{R}^2$. The coefficients $d_{\mathbf{i}}^{r,s}$ are a rectangular structure of points that form the *de Boor net*. $N^{r,s}(\mathbf{u}-\mathbf{i})$ are normalized tensor product B-splines of degree rs defined over the knots $\mathbf{i}$. The knot set $\mathbf{i}$ forms a rectangular grid of points which is naturally ordered by $\mathcal{Z}^2$. Figure 2.4 illustrates some examples.

A tensor product B-spline is the product of two independently parameterized univariate B-splines , i.e.,

$$N^{r,s}(\mathbf{u}-\mathbf{i}) \equiv N^r(u-i)N^s(v-j)$$

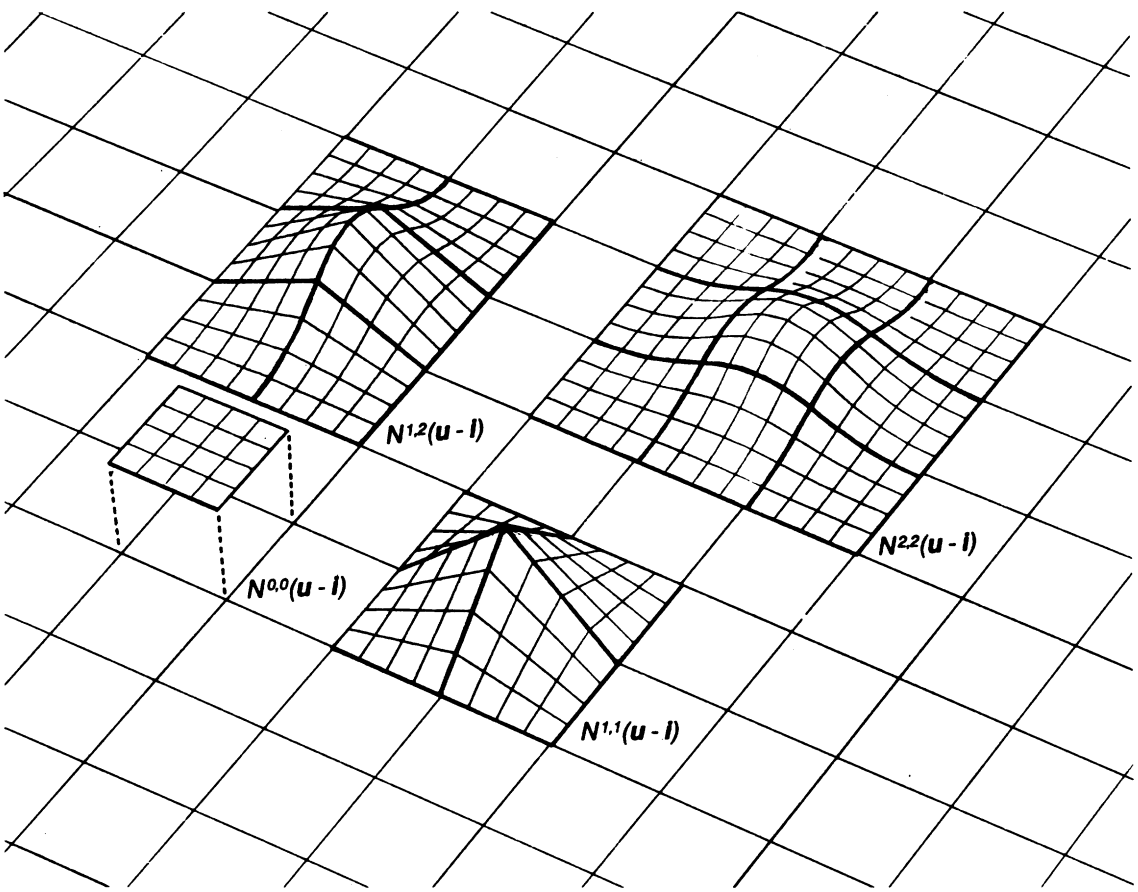


Figure 2.4: Examples of tensor product B-splines .

where $N^r(u-i)$ and $N^s(v-j)$ are univariate B-splines of degree r and s, such that $\mathbf{u} = (u, v)$ and $\mathbf{i} = (i, j)$. It is not surprising that the properties and formulas for tensor product B-splines are found directly from univariate B-splines.

The $N^{r,s}(\mathbf{u} - \mathbf{i})$ form a partition of unity and are non-negative. This implies that a tensor product B-spline surface lies in the convex hull of its de Boor points. $N^{r,s}(\mathbf{u})$ has local support, that is, $N^{r,s}(\mathbf{u} - \mathbf{i})$ is nonzero only over a finite support, i.e., when $u \in [i, i+r+1]$ and $v \in [j, j+s+1]$, for $\mathbf{u} = (u, v)$ and $\mathbf{i} = (i, j)$. Thus a change in a single de Boor point will only affect the surface locally. $N^{r,s}(u, v)$ is $r-1$ and $s-1$ times continuously differentiable in the u and v directions respectively.

2.4 Subdivision of Tensor Product B-spline Surfaces

Subdivision of tensor product B-spline surfaces is analogous to subdivision of B-spline curves. Similarly, the de Boor net is refined and becomes a better approximation to the underlying surface. As with curves, this process will converge to the underlying surface.

The knot set $\mathbf{i}$ is refined to $\mathbf{i}/2 = \mathbf{j}$, so $\mathbf{j} \in \mathcal{Z}^2/2$, i.e. $\mathbf{j} \in \{(\frac{i}{2}, \frac{j}{2}) | i, j \in \mathcal{Z}\}$. A single translated tensor product B-spline $N^{r,s}(\mathbf{u} - \mathbf{i})$ is rewitten over the refined grid as

$$N^{r,s}(\mathbf{u} - \mathbf{i}) = \sum_{\mathbf{j} \in \mathcal{Z}^2/2} c^{r,s}_{\mathbf{j}-\mathbf{i}} N^{r,s}(2(\mathbf{u} - \mathbf{j})). \tag{2.11}$$

This is substituted into (2.10) to give

$$S^{r,s}(\mathbf{u}) = \sum_{\mathbf{i} \in \mathcal{Z}^2} d^{r,s}_{\mathbf{i}} \sum_{\mathbf{j} \in \mathcal{Z}^2/2} c^{r,s}_{\mathbf{j}-\mathbf{i}} N^{r,s}(2(\mathbf{u} - \mathbf{j})). \tag{2.12}$$

It follows that

$$\hat{d}^{r,s}_{\mathbf{j}} = \sum_{\mathbf{i} \in \mathcal{Z}^2} c^{r,s}_{\mathbf{j}-\mathbf{i}} d^{r,s}_{\mathbf{i}} \qquad \mathbf{j} \in \mathcal{Z}^2/2 \tag{2.13}$$

where the $\hat{d}^{r,s}_{\mathbf{j}}$ form the refined de Boor net. Also

$$c^{r,s}_{\mathbf{j}} \equiv c^r_i c^s_j$$

where $\mathbf{j} = (i, j)$. So from (2.5)

$$c_{\mathbf{j}}^{r,s} = 2^{-(r+s)} \binom{r+1}{2i} \binom{s+1}{2j} \tag{2.14}$$

where $\mathbf{j} = (i, j)$. Equations (2.13) and (2.14) describe the construction of new de Boor points of the refined net. For example, if $r = s = 2$ then

$$\begin{aligned} \hat{d}_{0,0}^{2,2} &= \ldots + (\tfrac{9}{16}) d_{-1,-1}^{2,2} + (\tfrac{3}{16}) d_{0,-1}^{2,2} + \ldots \\ &\quad \ldots + (\tfrac{3}{16}) d_{-1,0}^{2,2} + (\tfrac{1}{16}) d_{0,0}^{2,2} + \ldots \end{aligned}$$

Again, the $\hat{d}_{\mathbf{j}}^{r,s}$ are a convex combination of a local subset of the $d_{\mathbf{i}}^{r,s}$. The ratio of weights in this convex combination forms the subdivision mask. From the above example, the mask set :

$$\begin{array}{ccc} 9 & — & 3 \\ | & & | \\ 3 & — & 1 \end{array} \qquad \begin{array}{ccc} 3 & — & 9 \\ | & & | \\ 1 & — & 3 \end{array} \qquad \begin{array}{ccc} 3 & — & 1 \\ | & & | \\ 9 & — & 3 \end{array} \qquad \begin{array}{ccc} 1 & — & 3 \\ | & & | \\ 3 & — & 9 \end{array}$$

will completely determine all new de Boor points. Each mask is applied to each rectangular *face* of the de Boor net to create a new de Boor point. This is implied by the structure of the mask. Figure 2.5 illustrates the construction of the refined de Boor net.

Such symmetric simplicity is not always the case. The masks for bicubic ($r = s = 3$) tensor product B-splines are :

$$\begin{array}{ccccc} 1 & — & 6 & — & 1 \\ | & & | & & | \\ 6 & — & 36 & — & 6 \\ | & & | & & | \\ 1 & — & 6 & — & 1 \end{array} \qquad \begin{array}{ccc} 4 & — & 4 \\ | & & | \\ 24 & — & 24 \\ | & & | \\ 4 & — & 4 \end{array} \qquad \begin{array}{ccccc} 4 & — & 24 & — & 4 \\ | & & | & & | \\ 4 & — & 24 & — & 4 \end{array} \qquad \begin{array}{ccc} 16 & — & 16 \\ | & & | \\ 16 & — & 16 \end{array}$$

The application of these masks is implied from their structure. The weights are arranged to form a template, and applied to every configuration of de Boor points with the same structure. Three iterations of the geometric construction algorithm

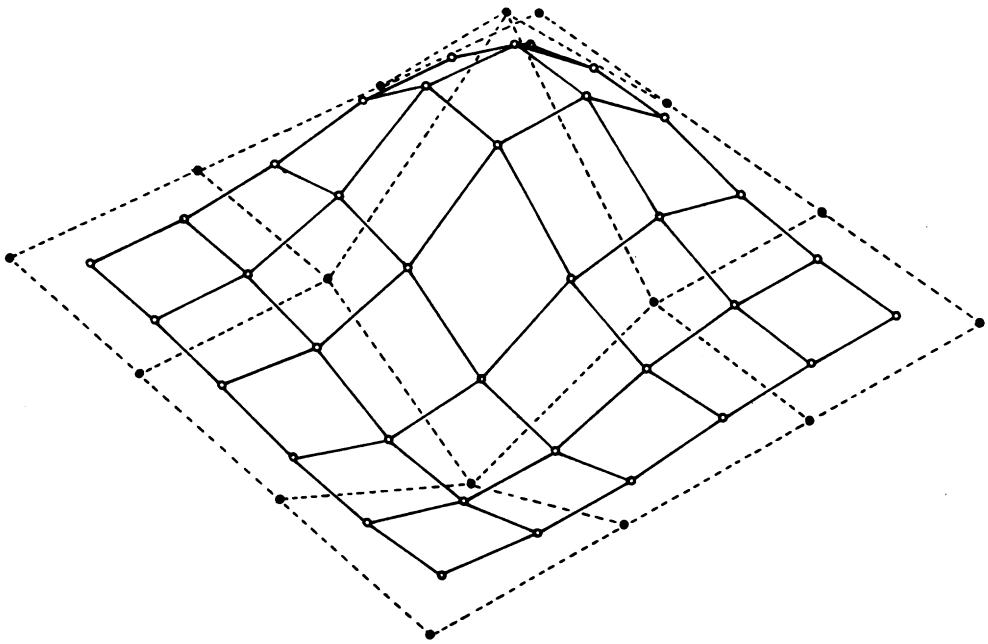


Figure 2.5: Construction of the refined de Boor net for $S^{2,2}(\mathbf{u})$.

implied by the bicubic binary subdivision masks are shown in Figure 2.6.

2.5 Triangular Splines

Another class of splines sharing the properties of univariate and tensor product B-splines are Triangular splines. Triangular splines are a type of *Box Spline* surface [Boehm 1985] [Boehm 1984] [Dahmen 1984]. A triangular spline surface $S^{r,s,t}(\mathbf{u})$ may be written

$$S^{r,s,t}(\mathbf{u}) = \sum_{\mathbf{i}} d_{\mathbf{i}}^{r,s,t} N^{r,s,t}(\mathbf{u} - \mathbf{i}) \tag{2.15}$$

where $\mathbf{u} \in \mathcal{R}^2$ and $\mathbf{i} \in \mathcal{Z}^2$. The $d_{\mathbf{i}}$ form a triangular de Boor net. The set $\mathbf{i}$ is a triangular grid with three primary directions, u, v, and w, labeled as in Figure 2.7.

$N^{r,s,t}(\mathbf{u})$ is a normalized triangular spline of degree $r+s+t-2$ over the grid $\mathbf{i}$. The support of $N^{r,s,t}(\mathbf{u})$ is characterized by r, s, and t. This support is the convex set of r, s, and t grid units in the u, v, and w directions respectively. Figure 2.8

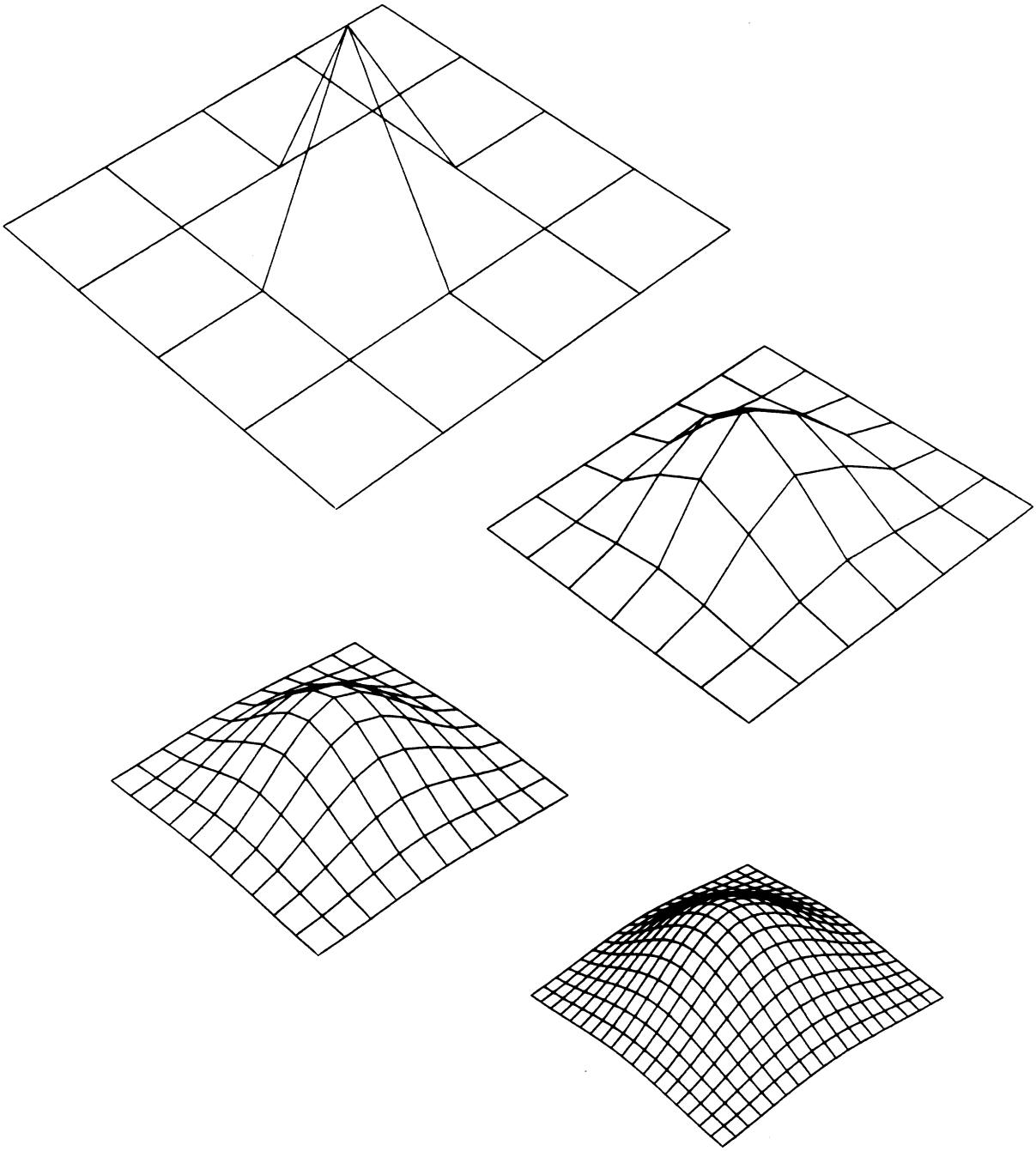


Figure 2.6: Binary refinement of a bicubic de Boor net.

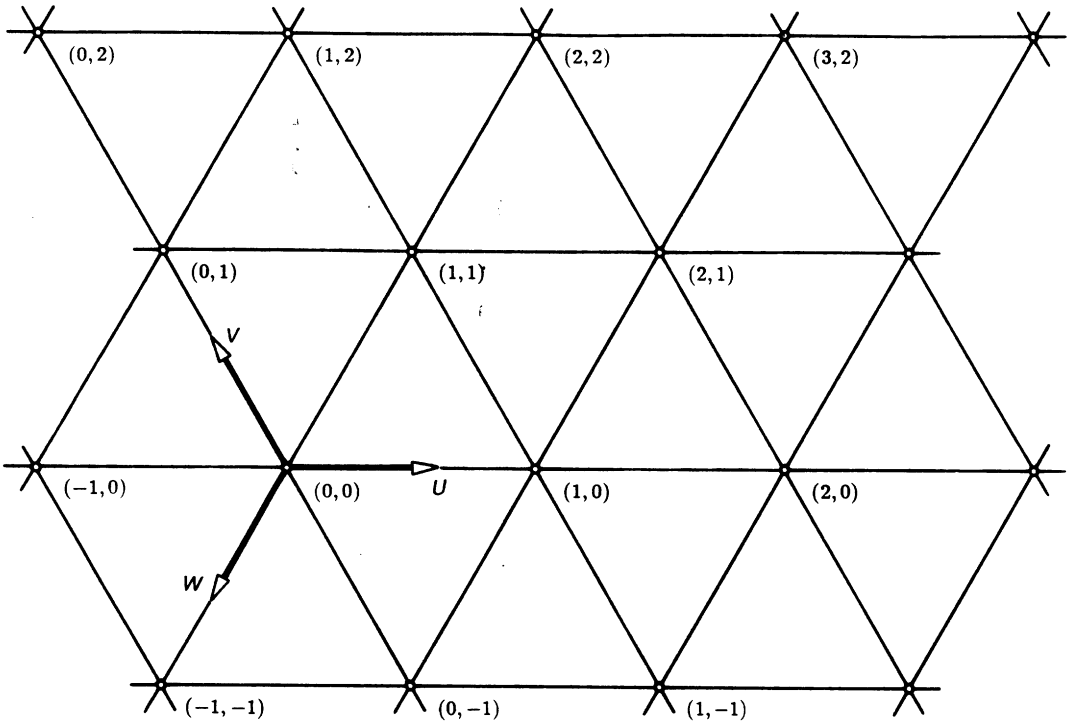


Figure 2.7: Triangular grid.

shows some examples. The $N^{r,s,t}(\mathbf{u}-\mathbf{i})$ form a partition of unity, are non-zero, and have local support. Unlike univariate B-splines , triangular splines do not, in general, span the space of polynomials of their total degree.

A triangular spline can be defined as the *shadow* of a cube in an n-dimensional space, projected onto a 2-dimensional space [Boehm 1986]. This shadow for a 3-cube is shown in Figure 2.9. Note that the vertices of the cube project onto the grid. $N^{r,s,t}(\mathbf{u})$ is the intersection of the n-cube with the projector that has $\mathbf{u}$ as its image in the plane. A recursion formula for triangular splines was found by de Boor and Höllig [Boehm 1985] [Boehm 1983b].

2.6 Subdivsion of Triangular Splines

The procedure for subdividing triangular splines exactly parallels the subdivision schemes presented so far. The grid $\mathbf{i}$ is refined to a grid $\mathbf{j} = \mathbf{i}/2$. The surface

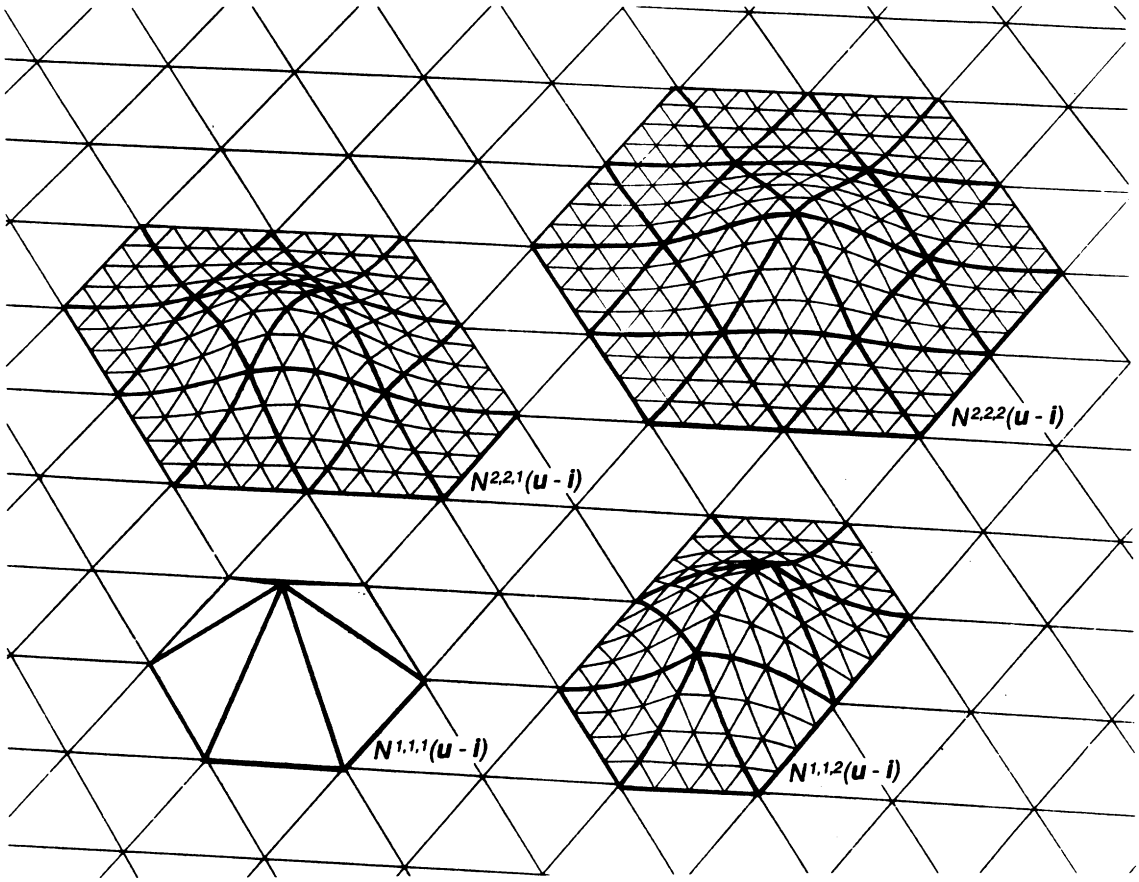


Figure 2.8: Examples of triangular splines.

corresponding to equation (2.15) is rewritten over the refined grid to become

$$S^{r,s,t}(\mathbf{u}) = \sum_{\mathbf{j}} d_{\mathbf{j}}^{r,s,t} N^{r,s,t}(2(\mathbf{u} - \mathbf{j})). \tag{2.16}$$

A single triangular spline is decomposed into splines of identical degree over the refined grid, i.e.,

$$N^{r,s,t}(\mathbf{u}) = \sum_{\mathbf{j}} c_{\mathbf{j}}^{r,s,t} N^{r,s,t}(2(\mathbf{u} - \mathbf{j})). \tag{2.17}$$

By substituting (2.17) into (2.16) and rearranging the order of summation it is found that

$$\hat{d}_{\mathbf{j}}^{r,s,t} = \sum_{\mathbf{i}} c_{\mathbf{j}-\mathbf{i}}^{r,s,t} d_{\mathbf{i}}^{r,s,t}. \tag{2.18}$$

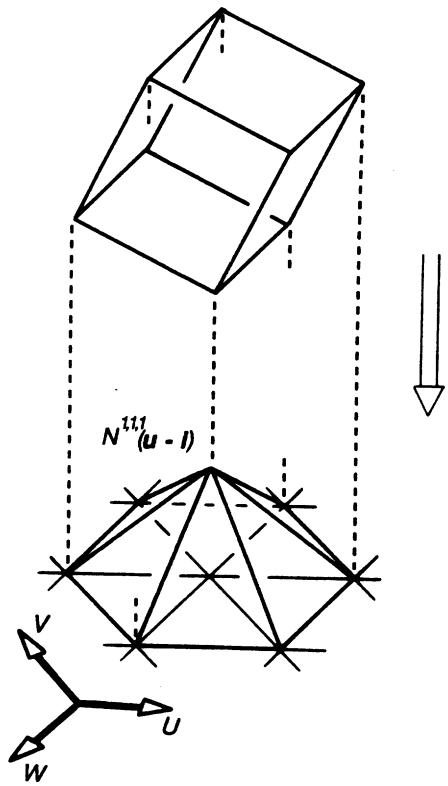


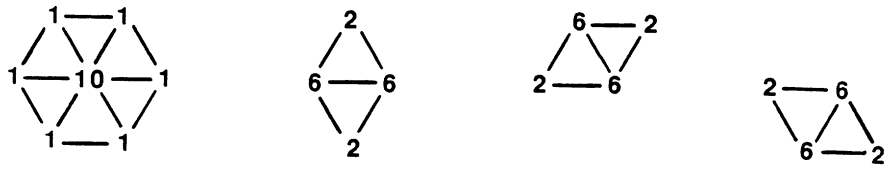
Figure 2.9: A triangular spline as the shadow of a cube.

For the special case of binary subdivision it is possible to show that

$$c_{\mathbf{j}} = 2^{-(r+s+t)} \sum_{k=0}^{t} \binom{r}{2i-k} \binom{s}{2j-k} \binom{t}{k}. \tag{2.19}$$

It is cautioned that this formula depends on the labeling of the grid. Equation (2.19) represents the weights associated with the binary decompositon of a triangular spline. Equations (2.18) and (2.19) give a complete description of how to refine the de Boor net for triangular splines. As in the case of univariate and tensor product B-splines, the $\hat{d}_{\mathbf{j}}^{r,s,t}$ are dependent only on a local subset of the $d_{\mathbf{i}}^{r,s,t}$. The ratio of weights used to compute the new de Boor points from these local subsets form the binary subdivision masks. Of particular interest in this thesis are the binary

subdivision masks for the triangular spline $N^{2,2,2}(\mathbf{u})$, which are :



These masks describe the geometric constructions needed to refine the de Boor net for a surface $S^{2,2,2}(\mathbf{u})$. Three interations of this algorithm are shown in Figure 2.10.

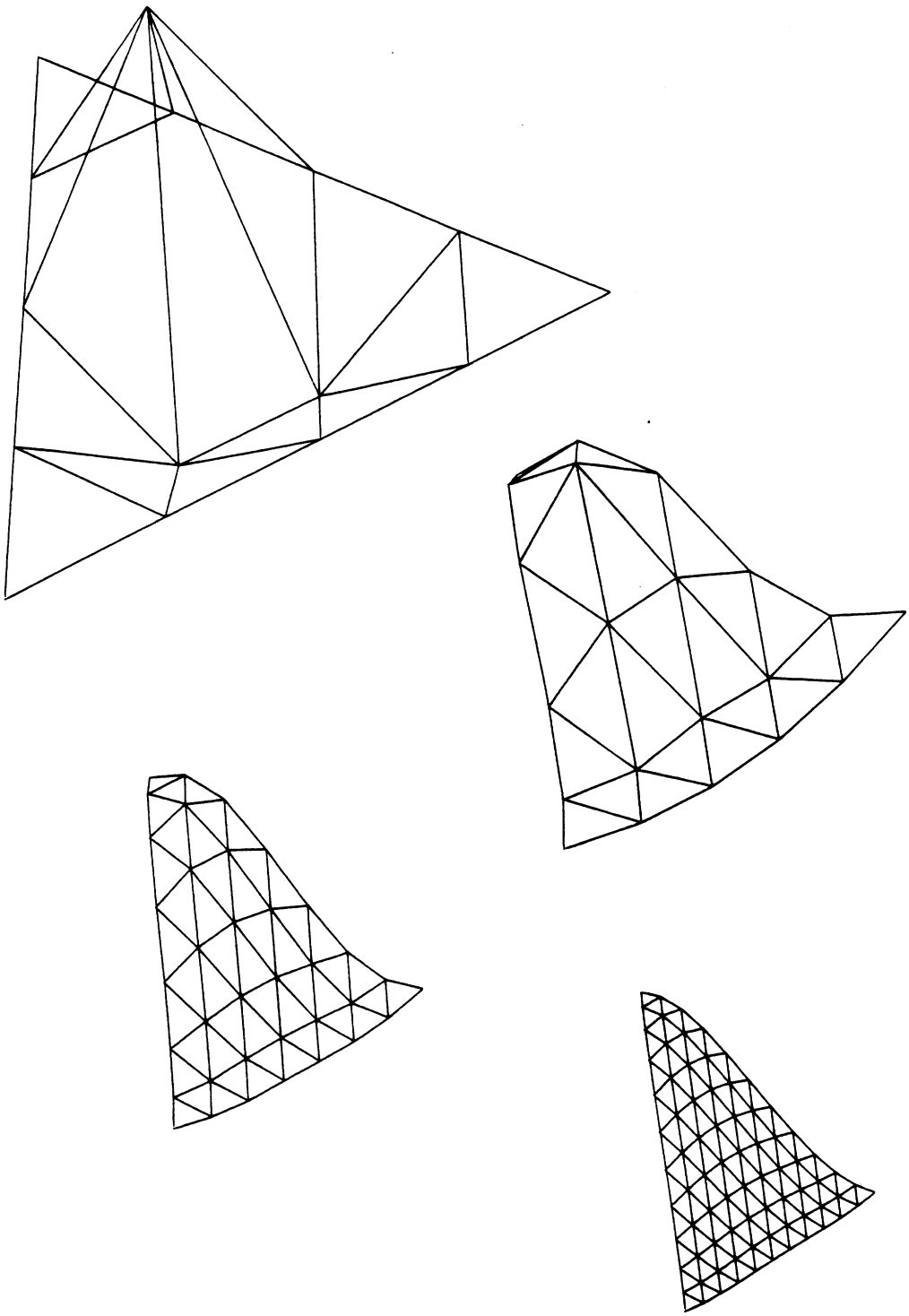


Figure 2.10: Refinement of the de Boor net for triangular splines.

Chapter 3

Generalized Subdivision

Binary subdivision formulas for B-spline curves and surfaces that generate a refined set of de Boor points have been presented. Repeated refinement will converge to the underlying curve or surface defined by the original de Boor points. The refinement approach to generating surfaces is generalized in this chapter. The result is a wider range of surfaces that share many of the positive design attributes of conventional B-spline surfaces.

3.1 Chaikin's algorithm

In 1974, Chaikin presented 'An algorithm for high speed curve generation' [Chaikin 1974]. This algorithm generates a smooth curve that is approximately the shape of a user specified control polygon. The algorithm works by 'clipping' the corners of the control polygon to form a new smoother control polygon. For each point of the control polygon clipped, two new points are found. Repeated application of this clipping process produces finer control polygons that converge to a smooth curve. Chaikin's algorithm stops clipping when the distance between adjacent control points is below the resolution of a display device.

Chaikin devised simple rules for clipping the corners of the control polygon. On each segment of the control polygon, two new points are found at a distance $\frac{1}{4}$ and $\frac{3}{4}$ between the end points. The ratio $\frac{1}{4} : \frac{3}{4}$ was chosen for the speed and ease of performing the clipping operation on a digital computer. The new control

points can be computed as add and shift operations on the old control points. This justified Chaikin's claim of 'high speed'.

By choosing the ratio $\frac{1}{4} : \frac{3}{4}$ the curve generated by this algorithm is piecewise quadratic. In fact, Chaikin's algorithm is the refinement algorithm for quadratic B-splines presented in Chapter 2. It is presented here mainly for its historical interest, and due to the geometric approach taken. Refinement of the univariate B-spline basis is generally associated with Chaikin. His original formulation is recursive in nature, subdividing segments locally until their lengths fall below a certain tolerance. Chaikin's geometric approach to refinement was soon extended to surfaces.

3.2 The Doo/Sabin algorithm

Subdivision formulas for tensor product B-spline surfaces were presented in Chapter 2. When $r = s = 2$, the result is the tensor product extension of Chaikin's algorithm. However, some very rigid restrictions are placed on the topology (i.e., connectivity relationships) of the de Boor net. Specifically, each vertex of the net must have *order* 4, i.e., a vertex is the endpoint of 4 edges. This may be relaxed to define a surface with boundary, but in general, the net must have a rectangular structure. This restriction makes the design of many surfaces difficult. For example, closed surfaces seldom exhibit a rectangular topology.

In 1978, Doo and Sabin [Doo 1978] presented an algorithm that eliminated this restriction by generalizing the biquadratic B-spline subdivision rules to include arbitrary topologies. They created an algorithm with the corner clipping notion of Chaikin, that generates smooth surfaces from an arbitrary control point net or *mesh*. In the special case of a rectangular mesh, the algorithm generates a biquadratic B-spline surface.

To derive the Doo/Sabin algorithm, recall the subdivision masks for biquadratic

B-splines :

Application of these masks generates four new de Boor points on each face of the original de Boor net. These new de Boor points are found by taking a convex combination of the four vertices of each face of the de Boor net. However, the new de Boor points can also be found indirectly by observing that each lies at the midpoint of the line segment connecting each vertex to the centroid of a face.

This geometric view of biquadratic B-spline subdivision may be generalized to include control point meshes of arbitrary topology. In a mesh of arbitrary topology, the faces may be n-sided ($n \geq 3$). The centroid of a face is easily found as the average of its vertices. A new control point may be created at the midpoint of the line segment connecting a face's centroid to each of its vertices, as in Figure 3.1. This is the essence of the Doo/Sabin algorithm.

These steps are repeated until a control point mesh of the desired smoothness has been obtained. Three iterations of the Doo/Sabin algorithm are illustrated in Figure 3.2. At each step, the faces, edges, and vertices of the old mesh are clipped to form the new mesh. As subdivision proceeds, the refined control point mesh becomes locally rectangular everywhere except at a fixed number of points. These points, referred to as *extraordinary points*, correspond to the vertices and faces of the original mesh. It is interesting to note that over every group of four rectangles with an order 4 vertex in common, a biquadratic B-spline surface is locally an exact representation. As the refined mesh becomes increasingly regular, more of the surface is exactly representable. Only in neighborhoods of the extraordinary points does the surface defy this explicit representation. Doo and Sabin considered these regions as *holes* in an exact representation using biquadratic B-splines.

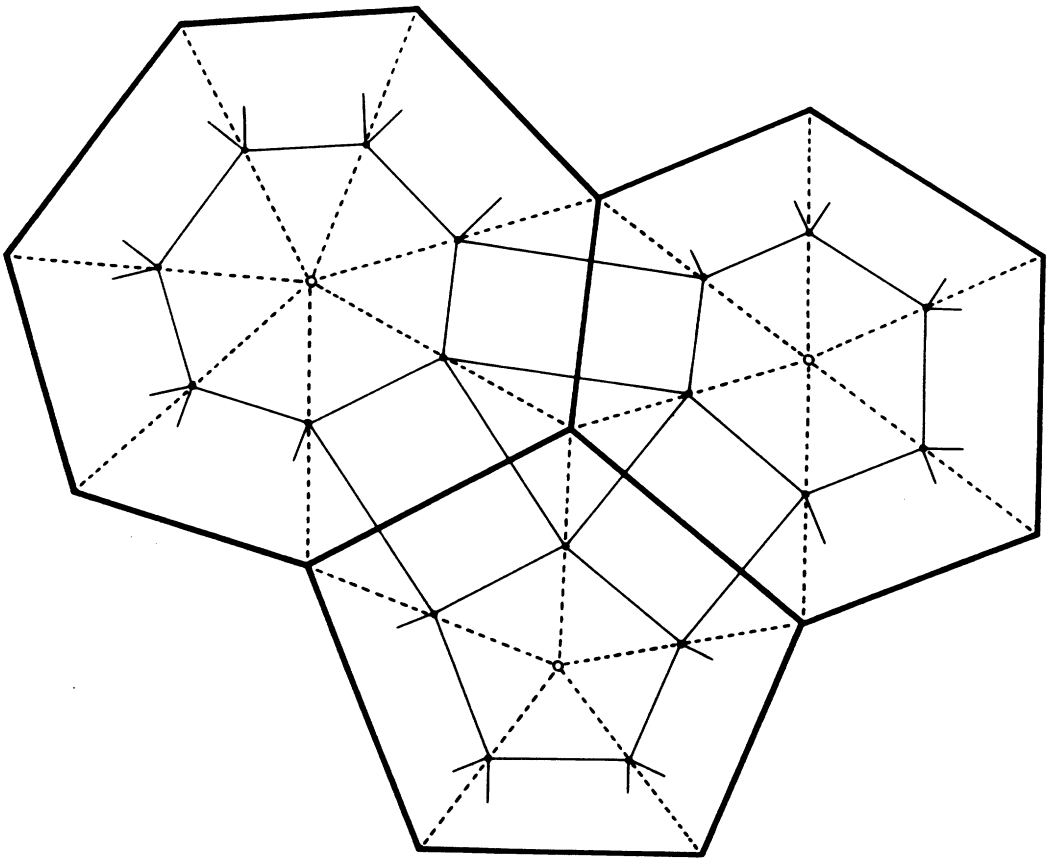


Figure 3.1: Construction of new control points using the Doo/Sabin algorithm.

This representation maintains all the properties of piecewise biquadratic B-splines. Since biquadratic B-splines are C^1, the surfaces generated by the Doo/Sabin algorithm are considered locally C^1 everywhere except at the extraordinary points. Since all new control points are found by taking convex combinations of the old control points, the refined mesh (and ultimately the surface) is guaranteed to lie in the convex hull of the original control point mesh. Also, each new control point is only dependent on a single face of the control point mesh. Thus, changing a control point will affect only a few faces, giving the Doo/Sabin surfaces a local control property.

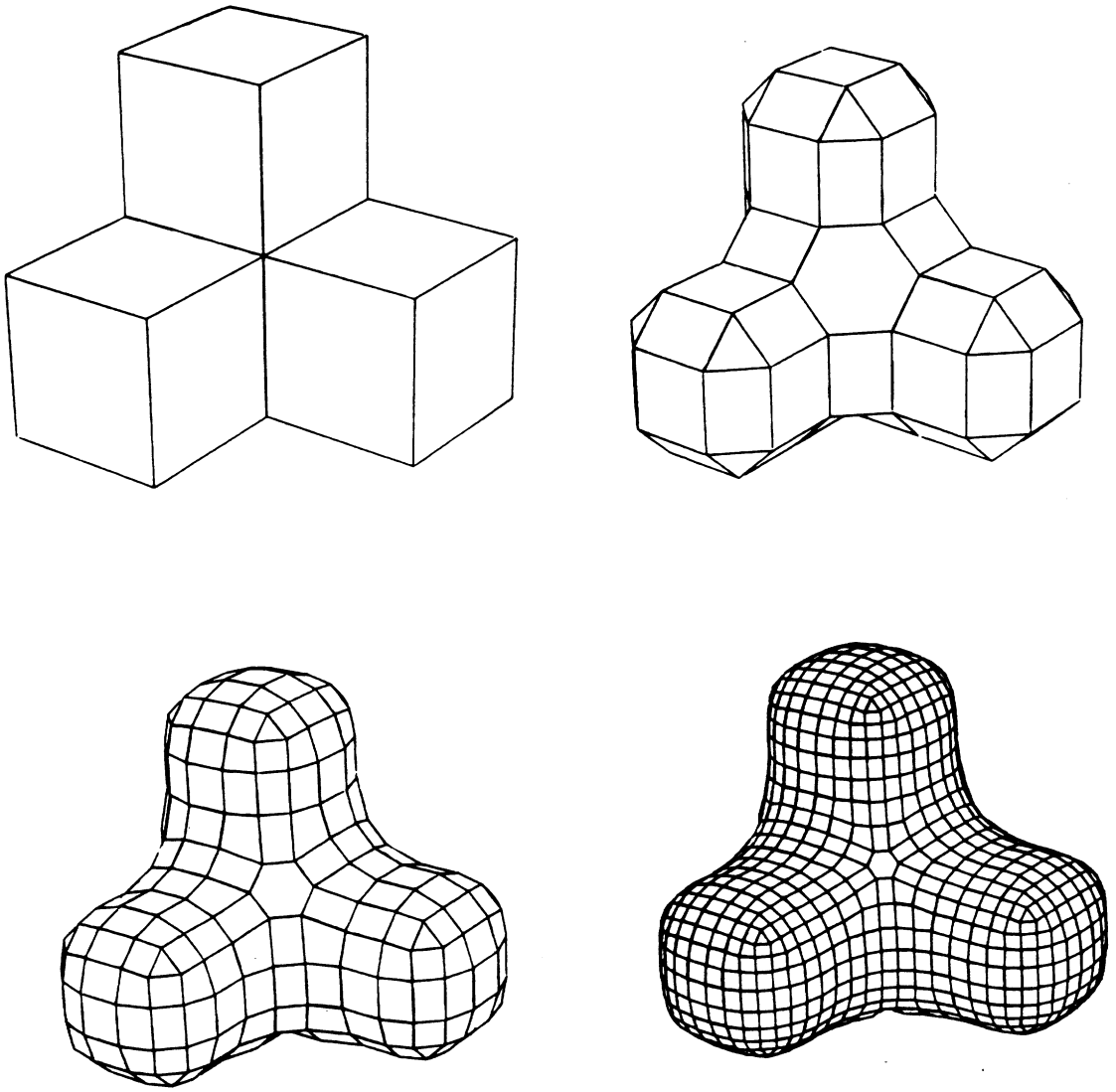
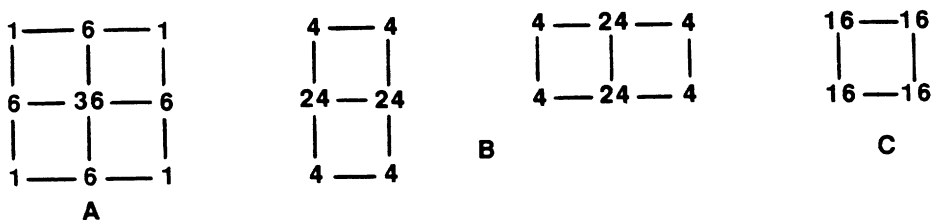


Figure 3.2: Three iterations of the Doo/Sabin algorithm.

3.3 The Catmull/Clark algorithm

At roughly the same time as Doo and Sabin, Catmull and Clark presented a similar algorithm for generating a smooth surface from an arbitrary control point mesh [Catmull and Clark 1978]. Their approach is a generalization of bicubic, as opposed to biquadratic, B-spline subdivision. Since bicubic surfaces are higher order than biquadratic, the resulting algorithm is more complex.

The Catmull/Clark algorithm is derived by abstracting the geometric properties of the bicubic B-spline subdivision masks :



The application of mask A generates a new control point corresponding to each vertex of the old control point mesh. The masks B generate new points corresponding to each edge, and mask C generates a new point corresponding to each face of the control point mesh. These shall be referred to as *new vertex*, *edge*, and *face points* respectively. The geometric properties of the subdivision masks used to generate these points may be abstracted to control point meshes of arbitrary topology.

To determine the rule for computing new face points, consider mask C. This mask gives equal weight to all the vertices belonging to a face, i.e., it computes the centroid of the face. An obvious generalization is to create a new face point at the centroid of each face of the arbitrary mesh. Consider masks B to determine the rule for computing new edge points. These masks generate new edge points as convex combinations of the vertices of the two faces adjacent at an edge. Geometric insight reveals that the point is found by taking the average of the centroids of the two adjacent faces along with the midpoint of the shared edge. This idea is easily

applied to an arbitrary mesh. Generalizing the rules to generate a new vertex point is not as simple.

Catmull and Clark determined that the vertex point $\hat{\mathbf{S}}$ generated by mask A is found by taking a convex combination of three points. These points are : $\mathbf{Q}$, the average of the new face points of all faces sharing an old vertex point; $\mathbf{R}$, the average of the midpoints of all old edges incident on the old vertex point; $\mathbf{S}$ the old vertex point. These three points may be similarly found for each vertex of an arbitrary mesh. The initial convex combination Catmull and Clark proposed was

$$\hat{\mathbf{S}} = \tfrac{1}{4}\mathbf{Q} + \tfrac{1}{2}\mathbf{R} + \tfrac{1}{4}\mathbf{S}. \tag{3.1}$$

The generalization of the new vertex point subdivision rule, like the new face and edge point rules, is equivalent to bicubic B-spline subdivision in the special case of a rectangular mesh. The rules just described were initially used by Catmull and Clark to generate smooth surfaces from arbitrary meshes. These surfaces, like those of the Doo/Sabin algorithm, are locally regular except at a constant number of extraordinary points. These points correspond to the faces and vertices of the original control point mesh. It was observed that in some arbitrary meshes, such as tetrahedra, tangent plane continuity was not maintained at extraordinary points. This was remedied by modifying the generalization of the new vertex point rule to take into account the order of the vertex. The modified rule is

$$\hat{\mathbf{S}} = \tfrac{1}{N}\mathbf{Q} + \tfrac{1}{N}\mathbf{R} + \tfrac{N-3}{N}\mathbf{S} \tag{3.2}$$

where N is the order of the vertex. This rule generates surfaces that exhibit tangent plane continuity at all extraordinary points. Figure 3.3 shows three iterations of the Catmull/Clark algorithm.

Note that as the algorithm proceeds, the mesh becomes increasingly regular. Over these regions, the surface is exactly representable with bicubic B-splines. For

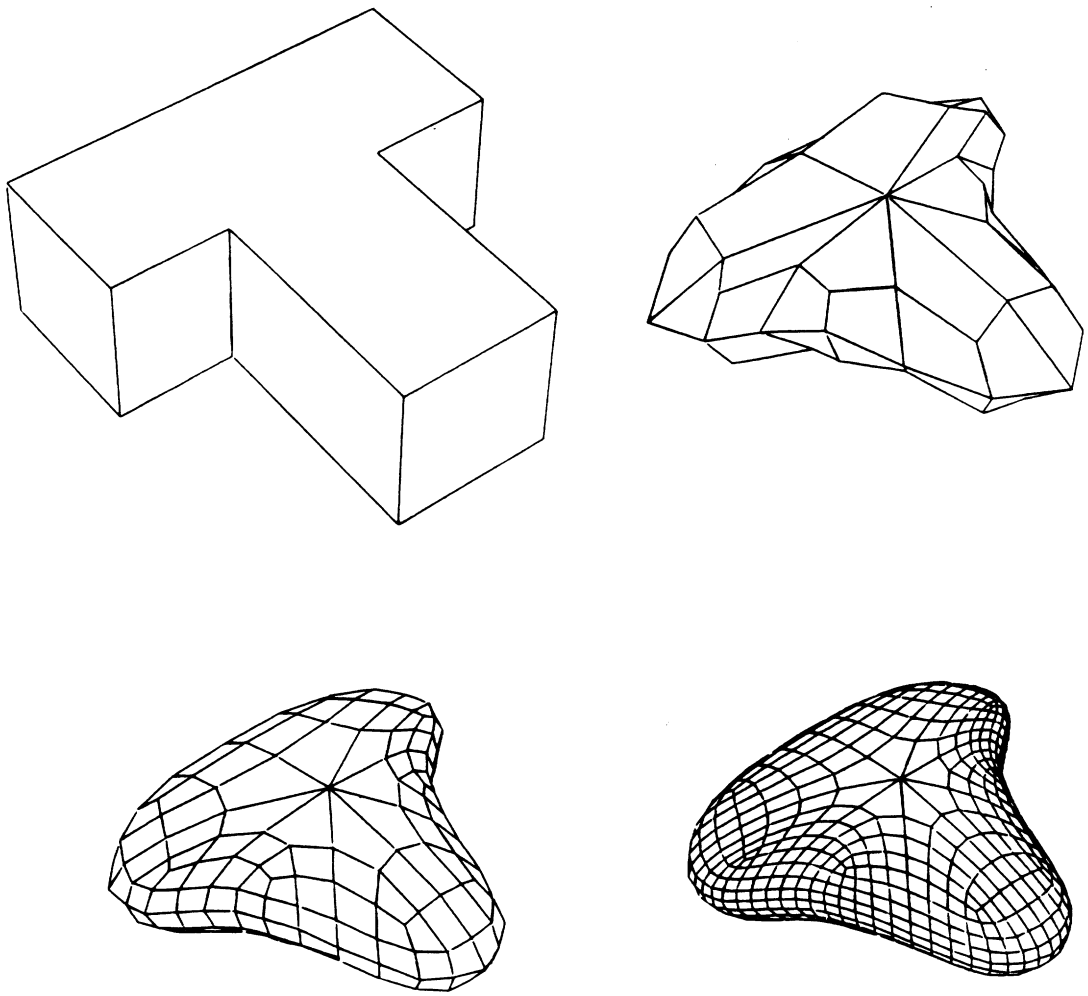


Figure 3.3: Three iterations of the Catmull/Clark algorithm.

this reason, Catmull/Clark surfaces inherit many of the important properties from bicubic B-splines. Catmull/Clark surfaces have the convex hull property, local control, and are locally C^2 everywhere except at the extraordinary points. A proof that Catmull/Clark surfaces have a continuous tangent plane at the extraordinary points was given by Doo and Sabin [Doo and Sabin 1978].

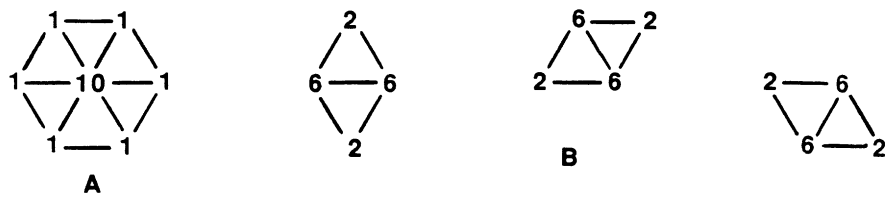
The Doo/Sabin and Catmull/Clark algorithms were derived from geometric properties of tensor product B-spline subdivision. Since then, other non-tensor product B-spline surfaces have appeared. The triangular splines presented in Chapter 2 are an example. From properties of these surfaces, an algorithm analogous to the Doo/Sabin and Catmull/Clark algorithms will be derived.

3.4 A generalized triangular subdivision surface

In Chapter 2, triangular spline surfaces were presented as a more recent surface extension of B-splines. Special attention was given to the triangular spline $\mathbf{N}^{2,2,2}(\mathbf{u})$. This triangular spline was chosen for study because it is the lowest order (in this case degree 4) triangular spline that has trilateral symmetry and C^2 smoothness. The surface generated by $\mathbf{N}^{2,2,2}(\mathbf{u})$ is a smooth approximation to a de Boor net that is topologically a regular triangular grid. In generalizing the subdivision of this type of surface, it is natural to consider only the special case of arbitrary *triangular* meshes. A triangular mesh is a control point mesh whose faces are all triangles. This may at first seem restrictive. However, when one considers the frequency with which triangulations and triangle based algorithms appear in computer graphics and CAGD, it becomes more reasonable.

Like the Doo/Sabin and Catmull/Clark algorithms, derivation of the generalized subdivision rules for this new algorithm begins with an abstraction of the geometric

properties of the subdivision masks for $N^{2,2,2}(\mathbf{u})$. These masks are :



Mask A generates new control points for each vertex, and masks B generate new control points for each edge of the original regular triangular mesh.

The masks B compute the new edge points as convex combinations of the vertices of the two triangles that share the edge. In an arbitrary triangular mesh, each edge will be shared by two triangles. Therefore, an obvious generalization is to leave this subdivision rule intact. Like the Catmull/Clark algorithm, generalization of a vertex point rule is more difficult.

To derive the new vertex point rule, consider mask A. The new vertex point $\hat{\mathbf{V}}$, can be computed as a convex combination of the old vertex, and all old vertices that share an edge with it. Alternatively, this same point may be found indirectly as a convex combination of two points. These points are : $\mathbf{V}$, the old vertex point, and $\mathbf{Q}$, the average of the old points that share an edge with $\mathbf{V}$. The new vertex point is computed as

$$\hat{\mathbf{V}} = \tfrac{5}{8}\mathbf{V} + \tfrac{3}{8}\mathbf{Q} \tag{3.3}$$

This same idea may be applied to an arbitrary triangular mesh.

In the special case of a regular triangulation, the algorithm is equivalent to binary subdivision of a surface based on $N^{2,2,2}(\mathbf{u})$. Three iterations of the algorithm based on the rules just described are shown in Figure 3.4. As subdivision proceeds, the triangular control point mesh becomes locally regular, except at a fixed number of extraordinary points. For this new algorithm, the extraordinary points correspond only to vertices of the original mesh, and not to its faces. Because of

the properties inherited from $N^{2,2,2}(\mathbf{u})$ it will be shown that the underlying surface of this algorithm is locally C^2 everywhere, except at the extraordinary points.

Note that in Figure 3.4 tangent plane continuity is apparently lost at one of the extraordinary points. This situation is similar to the one encountered by Catmull and Clark in the initial formulation of their algorithm. This may be remedied by considering the order of the vertex when taking the convex combination of $\mathbf{V}$ and $\mathbf{Q}$. This results in a new vertex point rule of the form

$$\hat{\mathbf{V}} = \alpha_N \mathbf{V} + (1 - \alpha_N)\mathbf{Q} \tag{3.4}$$

where α_N is a function of the vertex order N. As long as $\alpha_6 = \frac{5}{8}$, the subdivision algorithm will be a superset of the subdivision algorithm for $N^{2,2,2}(\mathbf{u})$. If $0 < \alpha_N < 1$, the resulting surface will lie in the convex hull of the control mesh. In the next chapter, a value of α_N will be derived to satisfy these conditions, and will be shown to generate a highly smooth surface.

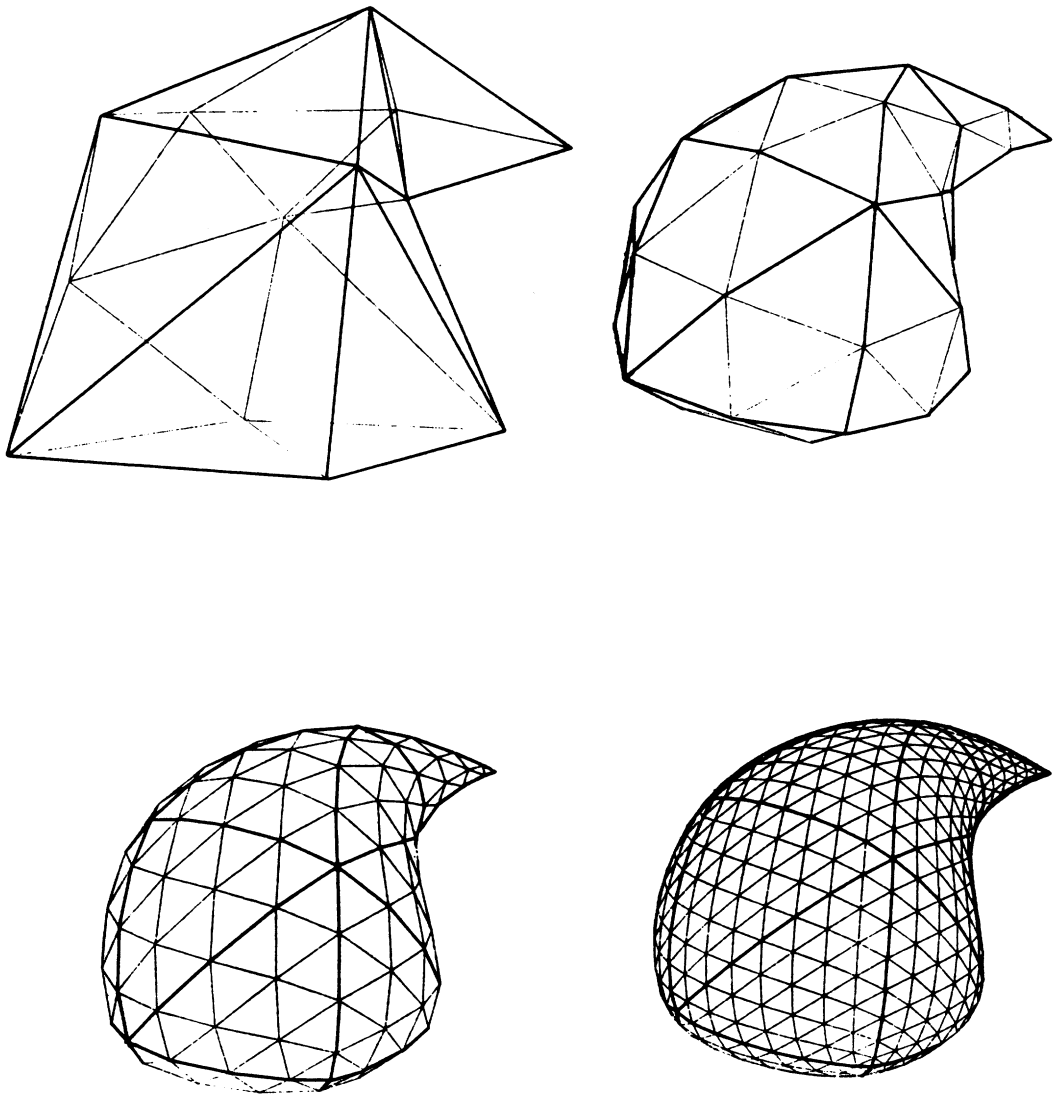


Figure 3.4: Three iterations of the initial triangular subdivision algorithm.

Chapter 4

Subdivision Analyzed

In the last chapter, an algorithm for refining an arbitrary triangular mesh was presented. This algorithm is a generalization of binary subdivision of surfaces based on the triangular spline $N^{2,2,2}(\mathbf{u})$. In its initial formulation, the algorithm generated surfaces that did not appear to have a well defined tangent plane at all of the extraordinary points. The algorithm was modified with the intent of remedying this situation. A parameter α_N was introduced, but not defined for general N (N = vertex order). In this chapter, α_N is defined and the effect it has on the surface is carefully analyzed.

4.1 Definitions and Notation

In order to study the geometric characteristics of the triangular subdivision algorithm presented in Chapter 3, precise notation must be introduced that allows for appropriate algebraic manipulation. Notation is simplified by the observation that each extraordinary point is isolated after one step of the algorithm. In other words, after one step, all edge sharing neighbors of an extraordinary point are *ordinary points* (N=6). This can be seen in Figure 3.4. For this reason, analysis of the surface may be done locally.

The local region to be studied consists of an extraordinary point $\mathbf{V}$, and all the vertices of the triangular mesh sharing an edge with $\mathbf{V}$. These points shall be

contained in the set

$$\mathbf{P} = \{\mathbf{P}_0, \ldots, \mathbf{P}_{N-1}\}$$

where N is the vertex order of $\mathbf{V}$, and $\mathbf{P}_{i \bmod N}$ shares an edge with $\mathbf{P}_{(i+1) \bmod N}$. In this way, $\mathbf{P}$ is cyclicly ordered. It is always assumed that all subscripts are to be taken $\bmod N$.

As subdivision proceeds, the geometry of $\mathbf{V}$ and $\mathbf{P}$ changes, but their structure does not. It is useful to superscript these points with the corresponding level of subdivision. Thus, $\mathbf{V}^\ell$ refers to the image of point $\mathbf{V}$ after ℓ iterations of the subdivision algorithm. Figure 4.1 illustrates an initial local configuration of points.

The subdivision rules derived in Chapter 3 may be applied to this configuration. On the ℓth iteration, the new vertex point $\mathbf{V}^\ell$ is found by taking a convex combination of the points $\mathbf{V}^{\ell-1}$, the extraordinary point ; and $\mathbf{Q}^{\ell-1}$, the average of

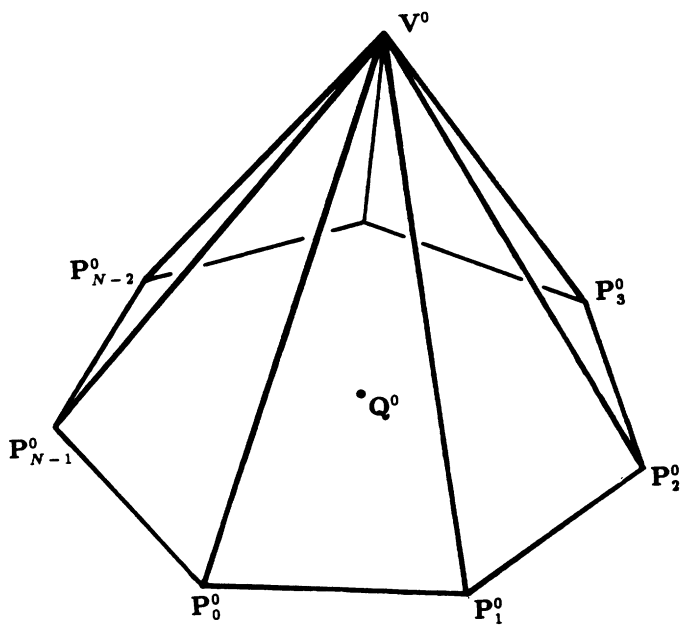


Figure 4.1: The initial configuration of points.

$\mathbf{V}^{\ell-1}$'s edge sharing neighbors. Formally let

$$\mathbf{Q}^{\ell} = \tfrac{1}{N}\sum_{i=0}^{N-1}\mathbf{P}_i^{\ell}. \tag{4.1}$$

The new vertex point rule states

$$\mathbf{V}^{\ell} = \alpha_N\mathbf{V}^{\ell-1} + (1-\alpha_N)\mathbf{Q}^{\ell-1}. \tag{4.2}$$

The masks for determining new edge points indicate that

$$\mathbf{P}_i^{\ell} = \tfrac{1}{8}\mathbf{P}_{i-1}^{\ell-1} + \tfrac{3}{8}\mathbf{P}_i^{\ell-1} + \tfrac{1}{8}\mathbf{P}_{i+1}^{\ell-1} + \tfrac{3}{8}\mathbf{V}^{\ell-1}. \tag{4.3}$$

With the above definitions, the configuration of points about an extraordinary point may be found for any level of subdivision. In particular, it is possible to study this configuration as $\ell \to \infty$.

In the following discussion, it will be useful to use an alternate definition of $\mathbf{Q}$. By substituting (4.3) into (4.1) it is found that

$$\begin{aligned}
\mathbf{Q}^{\ell} &= \tfrac{1}{N}\sum_{i=0}^{N-1}(\tfrac{1}{8}\mathbf{P}_{i-1}^{\ell-1} + \tfrac{3}{8}\mathbf{P}_i^{\ell-1} + \tfrac{1}{8}\mathbf{P}_{i+1}^{\ell-1} + \tfrac{3}{8}\mathbf{V}^{\ell-1}) \\
&= \tfrac{3}{8}\mathbf{V}^{\ell-1} + \tfrac{5}{8}(\tfrac{1}{N}\sum_{i=0}^{N-1}\mathbf{P}_i^{\ell-1}) \\
&= \tfrac{3}{8}\mathbf{V}^{\ell-1} + \tfrac{5}{8}\mathbf{Q}^{\ell-1}.
\end{aligned} \tag{4.4}$$

Next, a few points should be made to clarify possible confusion with subsequent notation. These clarifications regard ambiguities that might be caused by *type clashes*. The set $\mathbf{P}$ is considered a periodic vector valued function with argument i, whereas $\mathbf{V}$ and $\mathbf{Q}$ are points. It will be the case that on occasion a function such as $\mathbf{P}$ occurs in an equation with a point. In such cases, the point should be considered as a constant function. This is an instance of *function notation*. It will also be the case that the set elements (i.e., function values) need individual treatment. In these cases, the function is subscripted and should be considered as a point. This is an instance of function *value* notation.

4.2 Convergence

Figure 3.4 suggests that the triangular subdivision surface converges at extraordinary points. An empirical justification of convergence is not very insightful, nor does it guarantee success in the realm of strange and unforeseen configurations of triangles. Perhaps a triangular mesh exists that causes a loss of convergence in much the same way that tangent plane continuity is lost in Figure 3.4. With the introduction of the parameter α_N to the subdivision algorithm, this possibility becomes very real. In this section, bounds on α_N are derived which generate surfaces that are guarenteed to converge. Once it has been established that the surfaces converge at extraordinary points for α_N's in a specific range, an explicit formula for this point of convergence is given.

The proof of convergence is given in two parts. In the first part, it is shown that as $\ell \to \infty$ (i.e., as subdivision proceeds) $\mathbf{V}^\ell \to \mathbf{Q}^\ell$. If $\mathbf{V}^\ell \not\to \mathbf{Q}^\ell$ as $\ell \to \infty$, then $\mathbf{V}^\ell$ must be disjoint from the surface, but the fact that $\mathbf{V}^\ell \to \mathbf{Q}^\ell$ as $\ell \to \infty$ does not guarantee convergence. $\mathbf{Q}^\ell$ is an average of the $\mathbf{P}_i^\ell$'s. While this average may be approaching $\mathbf{V}^\ell$ as $\ell \to \infty$, the individual $\mathbf{P}_i^\ell$'s might be diverging. It is therefore also necessary to show that the components of $\mathbf{P}^\ell \to \mathbf{Q}^\ell$ as $\ell \to \infty$. This constitutes the second part of the proof.

In order to show that $\mathbf{V}^\ell \to \mathbf{Q}^\ell$ as $\ell \to \infty$, consider the vector

$$\mathbf{D}^\ell = \mathbf{V}^\ell - \mathbf{Q}^\ell. \tag{4.5}$$

If $\mathbf{V}^\ell \to \mathbf{Q}^\ell$ as $\ell \to \infty$, then $\mathbf{D}^\ell \to \mathbf{0}$ (the zero vector). Substituting equations (4.2) and (4.4) into (4.5) yields

$$\begin{aligned}
\mathbf{D}^\ell &= (\alpha_N \mathbf{V}^{\ell-1} + (1-\alpha_N)\mathbf{Q}^{\ell-1}) - (\tfrac{3}{8}\mathbf{V}^{\ell-1} + \tfrac{5}{8}\mathbf{Q}^{\ell-1}) \\
&= (\alpha_N - \tfrac{3}{8})(\mathbf{V}^{\ell-1} - \mathbf{Q}^{\ell-1}) \\
&= (\alpha_N - \tfrac{3}{8})\mathbf{D}^{\ell-1} \\
&= (\alpha_N - \tfrac{3}{8})(\alpha_N - \tfrac{3}{8})\mathbf{D}^{\ell-2}
\end{aligned}$$

$$\vdots$$

$$= \quad (\alpha_N - \tfrac{3}{8})^\ell \mathbf{D}^0. \tag{4.6}$$

$\mathbf{D}^0$ is not, in general, zero. If $\mathbf{D}^\ell = \mathbf{0}$ as $\ell \to \infty$, it must be that $(\alpha_N - \frac{3}{8})^\ell \to 0$. This is always the case for $-1 < (\alpha_N - \frac{3}{8}) < 1$, that is for $-\frac{5}{8} < \alpha_N < \frac{11}{8}$. Therefore $\mathbf{V}^\ell \to \mathbf{Q}^\ell$ as $\ell \to \infty$ for values of α_N in this range. The rate at which $\mathbf{V}^\ell \to \mathbf{Q}^\ell$ is $(\alpha_N - \frac{3}{8})$.

The second part of the proof is more complicated. The goal is to show that each $\mathbf{P}_i^\ell \to \mathbf{Q}^\ell$, for $i = 0, \ldots, N-1$ as $\ell \to \infty$. It is convenient to introduce some new notation. Let

$$\mathbf{M} = \{\tfrac{3}{8}, \tfrac{1}{8}, 0, \ldots, 0, \tfrac{1}{8}\}$$

be considered as a discrete periodic function (period N) such that $\mathbf{M}_0 = \frac{3}{8}$, $\mathbf{M}_1 = \frac{1}{8}$, $\mathbf{M}_{N-1} = \frac{1}{8}$, and $\mathbf{M}_i = 0$, for all $i \neq (N-1, 0, 1) \bmod N$. Using $\mathbf{M}$, equation (4.3) may be rewritten

$$\mathbf{P}_i^\ell = \sum_{j=0}^{N-1} \mathbf{M}_j \mathbf{P}_{i-j}^{\ell-1} + \tfrac{3}{8}\mathbf{V}^\ell. \tag{4.7}$$

Equation (4.7) is a discrete convolution [Brigham 1974] of the periodic function $\mathbf{P}^{\ell-1}$ with the even function $\mathbf{M}$. Thus equation (4.7) may be written

$$\mathbf{P}^\ell = \mathbf{M} * \mathbf{P}^{\ell-1} + \tfrac{3}{8}\mathbf{V}^\ell. \tag{4.8}$$

where $*$ is the convolution operator.

$\mathbf{Q}^\ell$ may also, when treated as a constant function, be expressed as a discrete convolution, with the aid of the following, let

$$\mathbf{A} = \{\tfrac{1}{N}, \tfrac{1}{N}, \ldots, \tfrac{1}{N}\}$$

be the discrete function of N values identically equal to $\frac{1}{N}$. Equation (4.1) may now be written as

$$\begin{aligned} \mathbf{Q}^\ell &= \sum_{j=0}^{N-1} \mathbf{A}_j \mathbf{P}_{i-j}^\ell \\ &= \mathbf{A} * \mathbf{P}^\ell. \end{aligned} \tag{4.9}$$

Combining (4.8) and (4.9) gives

$$\begin{aligned}
\mathbf{P}^{\ell} &= \mathbf{M} * \mathbf{P}^{\ell-1} + \tfrac{3}{8}\mathbf{V}^{\ell-1} \\
&= \mathbf{M} * \mathbf{P}^{\ell-1} - \tfrac{5}{8}\mathbf{A} * \mathbf{P}^{\ell-1} + \tfrac{3}{8}\mathbf{V}^{\ell-1} + \tfrac{5}{8}\mathbf{A} * \mathbf{P}^{\ell-1} \\
&= (\mathbf{M} - \tfrac{5}{8}\mathbf{A}) * \mathbf{P}^{\ell-1} + \mathbf{Q}^{\ell}. \qquad (4.10)
\end{aligned}$$

Let $\bar{\mathbf{M}} = \mathbf{M} - \frac{5}{8}\mathbf{A}$, (4.10) may be written

$$\mathbf{P}^{\ell} = \bar{\mathbf{M}} * \mathbf{P}^{\ell-1} + \mathbf{Q}^{\ell}. \qquad (4.11)$$

Equation (4.11) may be further simplified by observing that

$$\sum_{i=0}^{N-1} \mathbf{M}_i = \tfrac{5}{8} \qquad \text{and} \qquad \sum_{i=0}^{N-1} \tfrac{5}{8}\mathbf{A}_i = \tfrac{5}{8}$$

therefore

$$\sum_{i=0}^{N-1} \bar{\mathbf{M}}_i = 0.$$

From (4.11),

$$\begin{aligned}
\mathbf{P}^{\ell} &= \bar{\mathbf{M}} * \mathbf{P}^{\ell-1} + \mathbf{Q}^{\ell} \\
&= \bar{\mathbf{M}} * (\bar{\mathbf{M}} * \mathbf{P}^{\ell-2} + \mathbf{Q}^{\ell-1}) + \mathbf{Q}^{\ell}.
\end{aligned}$$

Since $\bar{\mathbf{M}}$ sums to zero and $\mathbf{Q}^{\ell-1}$ is a constant function, it follows that $\bar{\mathbf{M}} * \mathbf{Q}^{\ell-1} = \mathbf{0}$, therefore (4.11) may be written

$$\mathbf{P}^{\ell} = \bar{\mathbf{M}}^{\ell} * \mathbf{P}^{0} + \mathbf{Q}^{\ell} \qquad (4.12)$$

where $\bar{\mathbf{M}}^{\ell}$ denotes the function $\bar{\mathbf{M}}$ convolved with itself $\ell - 1$ times.

Recall that the goal is to show that $\mathbf{P}^{\ell} \to \mathbf{Q}^{\ell}$ as $\ell \to \infty$. From (4.12) it is clear that this is equivalent to showing that $\bar{\mathbf{M}}^{\ell} * \mathbf{P}^{0} \to \mathbf{0}$ as $\ell \to \infty$. For this to occur for any set $\mathbf{P}^{0}$, it must be that $\bar{\mathbf{M}}^{\ell} \to 0$ as $\ell \to \infty$. This can be shown quite easily with the aid of the Convolution Theorem [Brigham 1974]. This theorem states that convolution of two functions is equivalent to multiplication of their Fourier

transforms in the frequency domain. In order to avoid digression, the properties of discrete Fourier transforms relevant to the current discussion are described in the Appendix.

From the Convolution Theorem

$$\mathcal{F}(\bar{\mathbf{M}}^{\ell}) = [\mathcal{F}(\bar{\mathbf{M}})]^{\ell} \tag{4.13}$$

where $\mathcal{F}(\bar{\mathbf{M}}^{\ell})$ denotes the discrete Fourier transform of $\bar{\mathbf{M}}^{\ell}$. Therefore, to show that $\bar{\mathbf{M}}^{\ell} \to 0$ is equivalent to showing that $[\mathcal{F}(\bar{\mathbf{M}})]^{\ell} \to 0$, since $\mathcal{F}(\mathbf{G}) = 0$ implies that $\mathbf{G} = 0$. Since the function $\bar{\mathbf{M}}$ is different for every vertex order N, the discrete Fourier transform of $\bar{\mathbf{M}}$ should be found for general N. This is done by using the linearity property of discrete Fourier transforms (Appendix). The discrete periodic even function $\bar{\mathbf{M}}$ is illustrated in Figure 4.2. $\bar{\mathbf{M}}$ may be decomposed into the sum of three simpler functions whose Fourier transforms are well known. Recall that $\bar{\mathbf{M}} = \mathbf{M} - \frac{5}{8}\mathbf{A}$, let $-\frac{5}{8}\mathbf{A}$ represent one of these functions. $\mathbf{M}$ is the sum of two discrete periodic even functions $\{\frac{3}{8}, 0, \ldots, 0\}$ and $\{0, \frac{1}{8}, 0, \ldots, 0, \frac{1}{8}\}$. These three functions, and their discrete Fourier transforms are illustrated graphically in Figure 4.2. These discrete Fourier transforms are combined to form the discrete Fourier transform of $\bar{\mathbf{M}}$, which has the form

$$\mathcal{F}(\bar{\mathbf{M}})_j = \begin{cases} 0 & \text{if } j = 0 \\ \frac{3}{8} + \frac{1}{4}\cos\frac{2\pi j}{N} & \text{otherwise.} \end{cases} \tag{4.14}$$

Since values of the cosine are between -1 and 1, values of $\mathcal{F}(\bar{\mathbf{M}})$ are between $\frac{1}{8}$ and $\frac{5}{8}$, for all N. Upon repeated multiplication, these terms will vanish. So, as $\ell \to \infty$, $[\mathcal{F}(\bar{\mathbf{M}})]^{\ell} \to 0$, and consequently $\bar{\mathbf{M}}^{\ell} \to 0$. Therefore from (4.12), $\mathbf{P}^{\ell} \to \mathbf{Q}^{\ell}$ as $\ell \to \infty$, concluding the proof.

The second part of this proof did not involve α_N. It showed that as subdivision proceeds, the edge sharing neighbors of the extraordinary point will converge to a constant function , i.e., a single point. The first part of the proof showed that

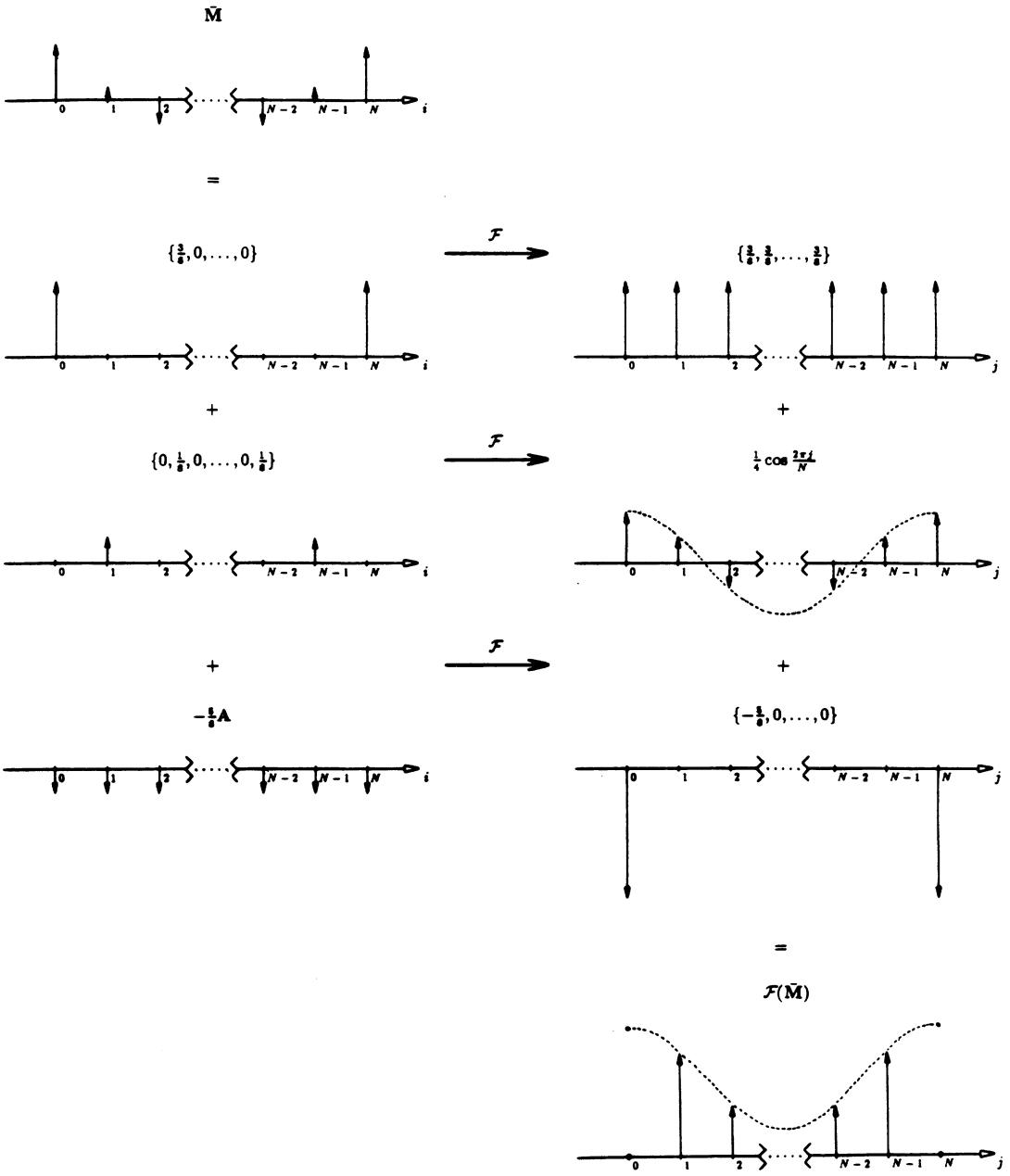


Figure 4.2: Computing $\mathcal{F}(\bar{\mathbf{M}})$ by a decomposition into simpler functions.

the extraordinary point will converge to this point only if α_N lies within a specific range. It can be concluded that surfaces generated by the triangular subdivision algorithm converge when $-\frac{5}{8} < \alpha_N < \frac{11}{8}$.

4.3 Explicit Point of Convergence

The surface generating algorithms presented in this paper are all 'approximating' as opposed to 'interpolating'. This means that the resulting surface in general will *not* contain any of the original triangular mesh points. It might be argued that with a refinement algorithm, one has only an approximation to a surface and not the surface itself. This is not the case. Since all ordinary points belong to a well defined surface element (a local patch based on $N^{2,2,2}(\mathbf{u})$), they are explicitly represented. The entire surface has an explicit representation if an explicit representation can be found for the limiting position of each extraordinary point. This amounts to finding an explicit representation of $\mathbf{Q}^\ell$ as $\ell \to \infty$. This is accomplished with some algebraic manipulation using equations (4.4), (4.5), and (4.6)

$$
\begin{aligned}
\mathbf{Q}^\ell &= \tfrac{3}{8}\mathbf{V}^{\ell-1} + \tfrac{5}{8}\mathbf{Q}^{\ell-1} \\
&= \tfrac{3}{8}(\mathbf{V}^{\ell-1} - \mathbf{Q}^{\ell-1}) + \mathbf{Q}^{\ell-1} \\
&= \tfrac{3}{8}\mathbf{D}^{\ell-1} + \mathbf{Q}^{\ell-1} \\
&= \tfrac{3}{8}\mathbf{D}^{\ell-1} + (\tfrac{3}{8}\mathbf{D}^{\ell-2} + \mathbf{Q}^{\ell-2}) \\
&= \tfrac{3}{8}\mathbf{D}^{\ell-1} + \tfrac{3}{8}\mathbf{D}^{\ell-2} + \tfrac{3}{8}\mathbf{D}^{\ell-3} + \cdots + \tfrac{3}{8}\mathbf{D}^{0} + \mathbf{Q}^{0} \\
&= \tfrac{3}{8}\sum_{k=0}^{\ell-1}\mathbf{D}^{k} + \mathbf{Q}^{0} \\
&= \tfrac{3}{8}\sum_{k=0}^{\ell-1}\mathbf{D}^{0}(\alpha_N - \tfrac{3}{8})^{k} + \mathbf{Q}^{0} \\
&= \mathbf{D}^{0}(\tfrac{3}{8}\sum_{k=0}^{\ell-1}(\alpha_N - \tfrac{3}{8})^{k}) + \mathbf{Q}^{0} \\
&= (\mathbf{V}^{0} - \mathbf{Q}^{0})\beta_N + \mathbf{Q}^{0} \\
&= \beta_N\mathbf{V}^{0} + (1-\beta_N)\mathbf{Q}^{0}.
\end{aligned}
\tag{4.15}
$$

Therefore, the point of convergence is a convex combination of $\mathbf{V}^0$ and $\mathbf{Q}^0$ depending on

$$\beta_N = \tfrac{3}{8}\sum_{k=0}^{\ell-1}(\alpha_N - \tfrac{3}{8})^k \qquad \text{as} \qquad \ell \to \infty.$$

This is a geometric series with a limit of the form

$$\sum_{k=0}^{\infty} a^k = \frac{1}{1-a} \qquad \text{if} \qquad -1 < a < 1.$$

The bounds on α_N to satisfy this expression are the same bounds found previously, so as $\ell \to \infty$

$$\beta_N = \frac{3}{11 - 8\alpha_N}. \tag{4.16}$$

Thus the point of convergence corresponding to each extraordinary point can be found explicitly. For example, if $\alpha_N = \frac{5}{8}$, $\beta_N = \frac{1}{2}$. So the point of convergence for the extraordinary point is at the midpoint of $\mathbf{V}^0$ and $\mathbf{Q}^0$.

4.4 Tangent Plane Continuity

It has been established that with an appropriate choice of α_N, the triangular subdivision surface converges at extraordinary points. The next logical step is to consider the behavior of the tangent plane at extraordinary points. Figure 3.4 gives empirical evidence that for $\alpha_N = \frac{5}{8}$, tangent plane continuity is not always guaranteed; this observation shall be confirmed algebraically. The parameter α_N was introduced in the hope that some amount of control might be had over the geometric characteristics of the surface. The bounds on α_N that guaranteed convergence are narrowed to guarantee tangent plane continuity. Once it has been established that the surface has a well defined tangent plane at extraordinary points , an explicit formulation of the plane is given. From this plane, a well defined surface normal may be found.

Let $\mathbf{T}^\ell$ represent a periodic function whose i^{th} component is the vector from $\mathbf{V}^\ell$ to $\mathbf{P}_i^\ell$. Therefore

$$\mathbf{T}^\ell = (\tfrac{1}{r})^\ell(\mathbf{P}^\ell - \mathbf{V}^\ell). \tag{4.17}$$

The term $(\frac{1}{r})^\ell$ represents a series of scalars that exactly cancels the contraction of $(\mathbf{P}^\ell - \mathbf{V}^\ell)$ between iterations of the algorithm. An exact value for r will be found shortly. By substituting (4.12) into (4.17) and using (4.5) and (4.6), (4.17) may be rewritten

$$\begin{aligned} \mathbf{T}^\ell &= (\tfrac{1}{r})^\ell(\bar{\mathbf{M}}^\ell * \mathbf{P}^0 + \mathbf{Q}^\ell - \mathbf{V}^\ell) \\ &= (\tfrac{1}{r})^\ell(\bar{\mathbf{M}}^\ell * \mathbf{P}^0 - \mathbf{D}^\ell) \\ &= (\tfrac{1}{r}\bar{\mathbf{M}})^\ell * \mathbf{P}^0 - (\tfrac{\alpha_N - \frac{3}{8}}{r})^\ell \mathbf{D}^0. \end{aligned} \tag{4.18}$$

From (4.18) it is clear that the limiting behavior of $\mathbf{T}^\ell$ depends on $(\frac{1}{r}\bar{\mathbf{M}})^\ell * \mathbf{P}^0$ and $(\frac{\alpha_N - \frac{3}{8}}{r})^\ell \mathbf{D}^0$ as $\ell \to \infty$.

The behavior of $(\frac{1}{r}\bar{\mathbf{M}})^\ell * \mathbf{P}^0$ as $\ell \to \infty$ can be determined by examining the behavior of $[\mathcal{F}(\frac{1}{r}\bar{\mathbf{M}})]^\ell$. By the linearity property of discrete Fourier transforms, $\mathcal{F}(\frac{1}{r}\bar{\mathbf{M}}) = \frac{1}{r}\mathcal{F}(\bar{\mathbf{M}})$. Let $r = \frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{N}$; using (4.14) it is found that the non-zero components of $\frac{1}{r}\mathcal{F}(\bar{\mathbf{M}})$ are of the form

$$\tfrac{1}{r}\mathcal{F}(\bar{\mathbf{M}})_j = \frac{\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi j}{N}}{\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{N}} \qquad j = 1, \ldots, N-1. \tag{4.19}$$

This is illustrated graphically in Figure 4.3. Note that with this choice of r, $\frac{1}{r}\mathcal{F}(\bar{\mathbf{M}})_j = 1$ for $j = 1, N-1$ and, $\frac{1}{r}\mathcal{F}(\bar{\mathbf{M}})_j < 1$ for $j \neq 1, N-1$. As $\ell \to \infty$, all terms of $[\frac{1}{r}\mathcal{F}(\bar{\mathbf{M}})]^\ell$ will vanish except for the pair of 1's at $j = 1, N-1$. Therefore, as $\ell \to \infty$,

$$[\tfrac{1}{r}\mathcal{F}(\bar{\mathbf{M}})_j]^\ell = \begin{cases} 1 & \text{if } j = 1, N-1 \\ 0 & \text{otherwise.} \end{cases} \tag{4.20}$$

The discrete inverse Fourier transform of this result is $\frac{2}{N}\cos\frac{2\pi i}{N}$ (Appendix). Or, in function notation, $\frac{2}{N}\mathbf{C}$ where $\mathbf{C}_i = \cos\frac{2\pi i}{N}$. Therefore, as $\ell \to \infty$, $(\frac{1}{r}\bar{\mathbf{M}})^\ell \to \frac{2}{N}\mathbf{C}$.

With the above argument equation (4.18) may be written

$$\mathbf{T}^\ell \to \tfrac{2}{N}\mathbf{C} * \mathbf{P}^0 - (\tfrac{\alpha_N - \frac{3}{8}}{r})^\ell \mathbf{D}^0 \qquad \text{as} \qquad \ell \to \infty. \tag{4.21}$$

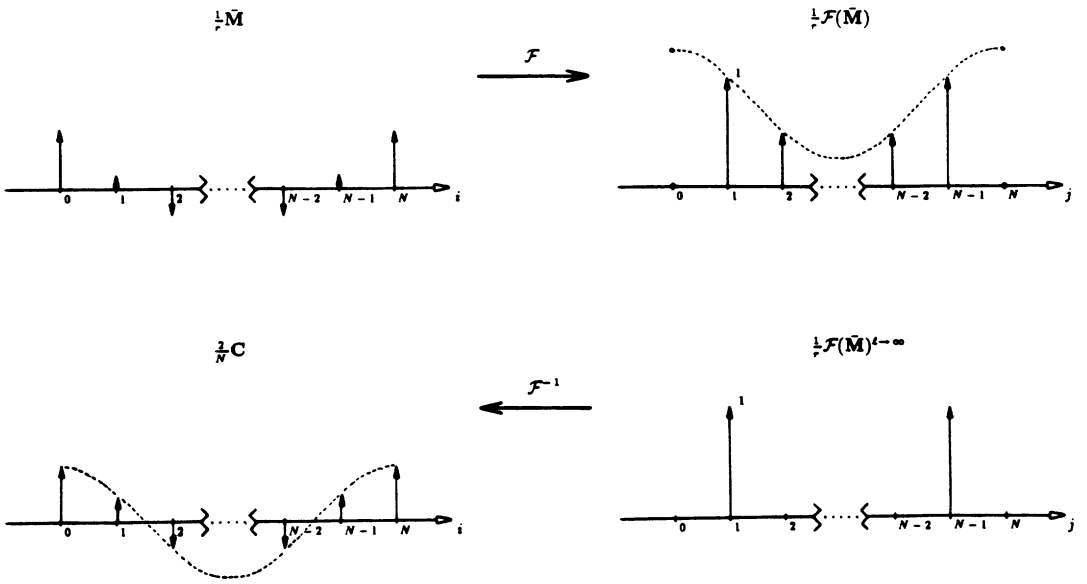


Figure 4.3: Determination of $(\frac{1}{r}\bar{\mathbf{M}})^\ell$ as $\ell \to \infty$.

In the limit, $(\frac{\alpha_N - \frac{3}{8}}{r})^\ell$ will be 0 , ± 1 , or ∞ depending on the value of $\frac{\alpha_N - \frac{3}{8}}{r}$. The parameter α_N may be chosen to give any of these effects. The appropriate choice will be made shortly. For the time being, set $\mathbf{k}^\ell = (\frac{\alpha_N - \frac{3}{8}}{r})^\ell \mathbf{D}^0$, so that equation (4.21) becomes

$$\mathbf{T}^\ell = \tfrac{2}{N}\mathbf{C} * \mathbf{P}^0 - \mathbf{k}^\ell. \tag{4.22}$$

Converting (4.22) to function value notation gives

$$\begin{aligned}
\mathbf{T}_i^\ell &= \tfrac{2}{N}\sum_{j=0}^{N-1} \cos\tfrac{2\pi(i-j)}{N}\mathbf{P}_j^0 - \mathbf{k}^\ell \\
&= \tfrac{2}{N}\sum_{j=0}^{N-1} (\cos\tfrac{2\pi i}{N}\cos\tfrac{2\pi j}{N} + \sin\tfrac{2\pi i}{N}\sin\tfrac{2\pi j}{N})\mathbf{P}_j^0 - \mathbf{k}^\ell \\
&= (\tfrac{2}{N}\sum_{j=0}^{N-1} \cos\tfrac{2\pi j}{N}\mathbf{P}_j^0)\cos\tfrac{2\pi i}{N} + (\tfrac{2}{N}\sum_{j=0}^{N-1} \sin\tfrac{2\pi j}{N}\mathbf{P}_j^0)\sin\tfrac{2\pi i}{N} - \mathbf{k}^\ell
\end{aligned} \tag{4.23}$$

Since the coefficients of $\cos\frac{2\pi i}{N}$ and $\sin\frac{2\pi i}{N}$ are constants, the vectors represented by $\mathbf{T}^\ell$ will lie in plane as $\ell \to \infty$, if $\mathbf{k}^\ell \to 0$ as $\ell \to \infty$.

Recall that $\mathbf{k}^\ell = (\frac{\alpha_N - \frac{3}{8}}{r})^\ell \mathbf{D}^0$, this is zero as $\ell \to \infty$, when $-1 < \frac{\alpha_N - \frac{3}{8}}{r} < 1$. This provides bounds on α_N which assure a well defined tangent plane. These bounds are

$$-\tfrac{1}{4}\cos\tfrac{2\pi}{N} < \alpha_N < \tfrac{3}{4} + \tfrac{1}{4}\cos\tfrac{2\pi}{N}. \tag{4.24}$$

Note that this range of α_N lies within the range that assured convergence for the surface.

In Figure 3.4, tangent plane continuity is apparently lost at the vertex of order 3. This surface was generated by an algorithm, which used $\alpha_3 = \frac{5}{8}$. The bounds on α_3 from (4.24) are $\frac{1}{8} < \alpha_3 < \frac{5}{8}$. In this case $\alpha_3 = \frac{5}{8}$ is not strictly within the range necessary for tangent plane continuity. This analysis confirms the empirical evidence that at this point, the surface does not have a well defined tangent plane.

If the α_N's are chosen to satisfy (4.24), then the tangent vector function at each extraordinary point can be written

$$\mathbf{T} = \tfrac{2}{N}\mathbf{C} * \mathbf{P}^0 \tag{4.25}$$

where $\mathbf{T}$ denotes $\mathbf{T}^\ell$ as $\ell \to \infty$. Note that $\mathbf{T}$ is a function of $\mathbf{P}^0$ and does not depend on $\mathbf{V}^0$. A surface normal for any extraordinary point can be found by evaluating (4.25) at any i and $i+1$, and taking the cross product of the resulting vectors. Also note that for $N = 6$, (4.25) is the directional derivative function for the parametric surface based on $N^{2,2,2}(\mathbf{u})$.

4.5 Curvature Continuity

It has been demonstrated in the previous section that the triangular subdivision surface has a well defined tangent plane everywhere with correctly chosen α_N's. This range of α_N's still allows a certain amount of 'control' over the shape of the surface. It is known that when $\alpha_6 = \frac{5}{8}$ the underlying parametric surface is locally C^2 at the corresponding ordinary point. This means that the surface has well

defined Gaussian curvature at this points. In this section, the issue of curvature continuity is discussed for general N. In what follows, the analysis has been guided by intuition and heuristics, the most important being that everything works for the case $N = 6$. Some arguments are made that are not fully justified.

The curvature function about a point on the surface refers to the normal curvature at the point in all directions. Therefore this function is periodic. Rather than trying to explicitly formulate a curvature function about an extraordinary point , a related function is developed instead, the justification being that if this function is well defined, then so is the curvature function. This related function is the rate of change of the tangent function "with respect to" the subdivision process. The rate of change of the tangent function at an extraordinary point may be defined as

$$\dot{\mathbf{T}}^\ell = (\tfrac{1}{r})^\ell(\mathbf{T}^\ell - \mathbf{T}). \tag{4.26}$$

Where again, the term $(\frac{1}{r})^\ell$ is a series of scalars that should cancel the convergent effect of $\mathbf{T}^\ell \to \mathbf{T}$ as $\ell \to \infty$. Using (4.5),(4.12),and (4.17), (4.26) may be written

$$\begin{aligned}
\dot{\mathbf{T}}^\ell &= (\tfrac{1}{r})^\ell((\tfrac{1}{r})^\ell(\mathbf{P}^\ell - \mathbf{V}^\ell) - \tfrac{2}{N}\mathbf{C} * \mathbf{P}^0) \\
&= (\tfrac{1}{r^2})^\ell((\bar{\mathbf{M}}^\ell * \mathbf{P}^0 + \mathbf{Q}^\ell) - \mathbf{V}^\ell - \tfrac{2r^\ell}{N}\mathbf{C} * \mathbf{P}^0) \\
&= (\tfrac{1}{r^2})^\ell(\bar{\mathbf{M}}^\ell * \mathbf{P}^0 - \tfrac{2r^\ell}{N}\mathbf{C} * \mathbf{P}^0 - \mathbf{D}^\ell).
\end{aligned} \tag{4.27}$$

Let $\bar{\bar{\mathbf{M}}} = \bar{\mathbf{M}} - \frac{2r}{N}\mathbf{C}$. With this substition, and using (4.6), (4.27) reduces to

$$\begin{aligned}
\dot{\mathbf{T}}^\ell &= (\tfrac{1}{r^2})^\ell(\bar{\bar{\mathbf{M}}}^\ell * \mathbf{P}^0 - (\alpha_N - \tfrac{3}{8})^\ell\mathbf{D}^0) \\
&= (\tfrac{1}{r^2}\bar{\bar{\mathbf{M}}})^\ell * \mathbf{P}^0 - (\tfrac{\alpha_N - \frac{3}{8}}{r^2})^\ell\mathbf{D}^0.
\end{aligned} \tag{4.28}$$

This equation represents the rate of change of $\mathbf{T}^\ell$ about an extraordinary point. The limiting behavior of (4.28) may be analyzed in terms of the discrete Fourier transform of the term $(\frac{1}{r^2}\bar{\bar{\mathbf{M}}})^\ell * \mathbf{P}^0$. The remaining term $(\frac{\alpha_N - \frac{3}{8}}{r^2})^\ell\mathbf{D}^0$ is analyzed as a geometric sequence.

To obtain the discrete Fourier transform of $\frac{1}{r^2}\bar{\bar{\mathbf{M}}}$, recall that $\bar{\bar{\mathbf{M}}} = \bar{\mathbf{M}} - \frac{2r}{N}\mathbf{C}$, thus $\mathcal{F}(\bar{\bar{\mathbf{M}}}) = \mathcal{F}(\bar{\mathbf{M}}) - \frac{2r}{N}\mathcal{F}(\mathbf{C})$. $\mathcal{F}(\bar{\mathbf{M}})$ is known from (4.14). $\frac{2r}{N}\mathcal{F}(\mathbf{C})$ is the δ function illustrated in Figure 4.4. Subtracting $\frac{2r}{N}\mathcal{F}(\mathbf{C})$ from $\mathcal{F}(\bar{\mathbf{M}})$ gives $\mathcal{F}(\bar{\bar{\mathbf{M}}})$, which is of the form

$$\mathcal{F}(\bar{\bar{\mathbf{M}}})_j = \begin{cases} 0 & \text{if } j = 0, 1, N-1 \\ \frac{3}{8} + \frac{1}{4}\cos\frac{2\pi j}{N} & \text{otherwise} \end{cases} \tag{4.29}$$

and is illustrated in Figure 4.4.

Recall that $r = \frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{N}$, thus the non-zero components of $\frac{1}{r^2}\mathcal{F}(\bar{\bar{\mathbf{M}}})$ are of the form

$$\frac{1}{r^2}\mathcal{F}(\bar{\bar{\mathbf{M}}})_j = \frac{\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi j}{N}}{(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{N})^2} \qquad j \neq 0, 1, N-1 \tag{4.30}$$

which is at a maximum when $j = 2$. Therefore, the limiting behavior of $(\frac{1}{r^2}\bar{\bar{\mathbf{M}}})^\ell$ is determined by $\frac{1}{r^2}\mathcal{F}(\bar{\bar{\mathbf{M}}})_2^\ell$. Values of $\frac{1}{r^2}\mathcal{F}(\bar{\bar{\mathbf{M}}})_2$ for various N are given in Table 4.1.

Note that $\frac{1}{r^2}\mathcal{F}(\bar{\bar{\mathbf{M}}})$ does not depend on α_N, therefore α_N cannot affect the limiting behavior of $\frac{1}{r^2}\mathcal{F}(\bar{\bar{\mathbf{M}}})^\ell$. From the above table, it is clear that $\dot{\mathbf{T}}^\ell$, as $\ell \to \infty$, is stable only when $N = 6$, i.e., at ordinary points. It is concluded that no choice of α_N can, in general, assure well defined curvature function at extraordinary points. This result has been confirmed by numerous attempts to estimate second order effects at extraordinary points.

In practice, the rate at which $\dot{\mathbf{T}}^\ell$ vanishes or diverges can be influenced by the remaining $(\frac{\alpha_N - \frac{3}{8}}{r^2})^\ell \mathbf{D}^0$ in (4.28). This term has some remaining freedom in that α_N may take on a range of values. A natural choice of α_N is one where $(\frac{\alpha_N - \frac{3}{8}}{r^2})^\ell \to 1$ as $\ell \to \infty$. This assured if $(\frac{\alpha_N - \frac{3}{8}}{r^2}) = 1$. Under this restriction, α_N takes on a unique value. Namely

$$\alpha_N = (\tfrac{3}{8} + \tfrac{1}{4}\cos\tfrac{2\pi}{N})^2 + \tfrac{3}{8}. \tag{4.31}$$

Values of α_N chosen in this way satisfy the previous bounds for a well defined tangent plane. Note that $\alpha_6 = \frac{5}{8}$, the correct value for ordinary points. A surface using this definition of α_N is illustrated in Figure 4.5.

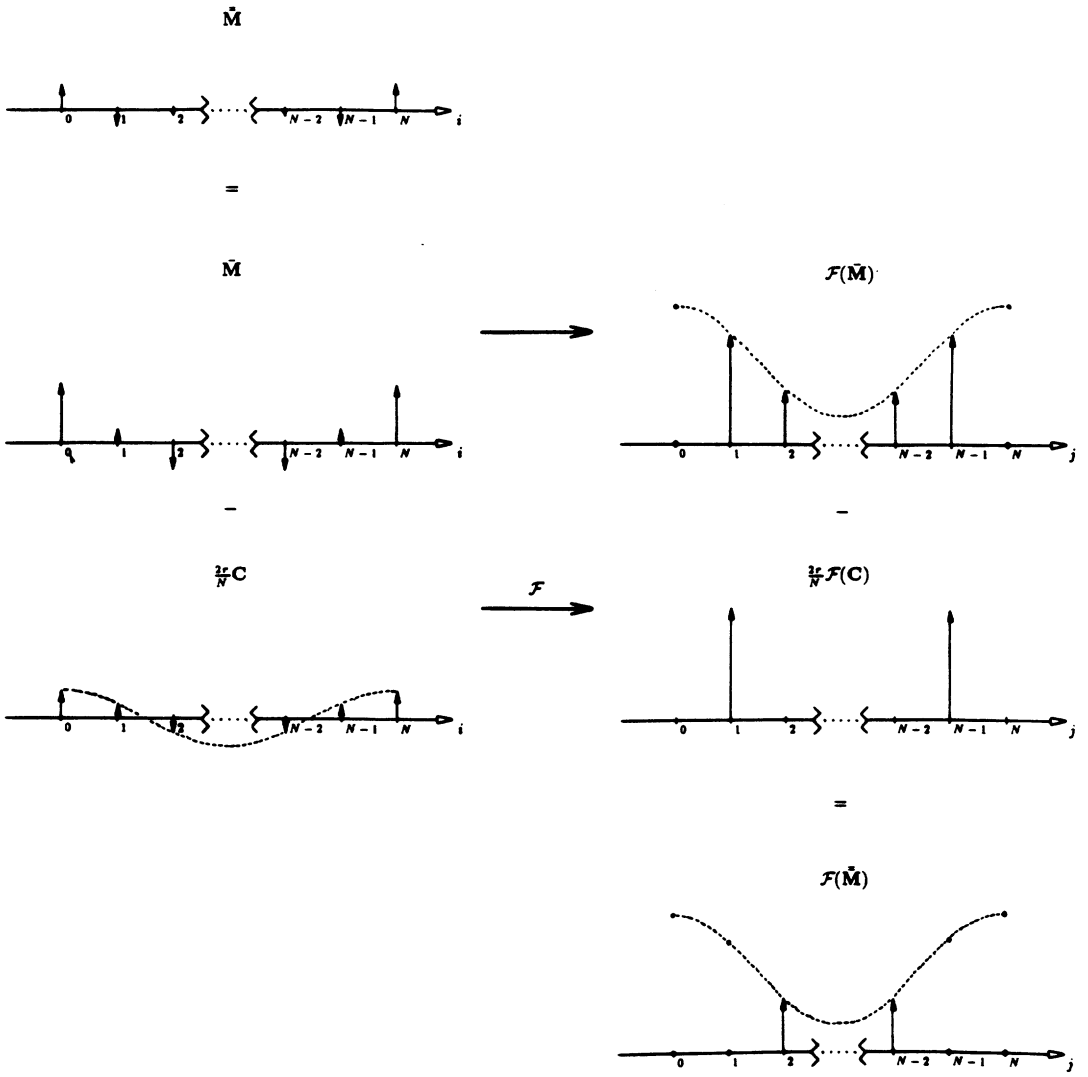


Figure 4.4: Derivation of $\mathcal{F}(\bar{\bar{\mathbf{M}}})$.

Table 4.1: Values of $\frac{1}{r^2}\mathcal{F}(\bar{\bar{\mathbf{M}}})_2$ for various N

N	$\frac{1}{r^2}\mathcal{F}(\bar{\mathbf{M}})_2$
3	0.000000
4	0.888889
5	0.844582
6	1.000000
7	1.133218
8	1.231699
9	1.303729
10	1.357214
11	1.397739
12	1.429062
∞	1.600000

4.6 Conclusion of analysis

An analysis of the triangular subdivision surface has been presented which confirms empirical evidence and predicts a value of α_N which leads to the *best* looking surface. It has been shown that this value of α_N results in surfaces which converge to a well defined tangent plane. Analysis of second order effects suggest that well defined Gaussian curvature does not exist at all points of the surface. However, there are only a finite number of such points and with appropriate α_N, the surface appears quite smooth.

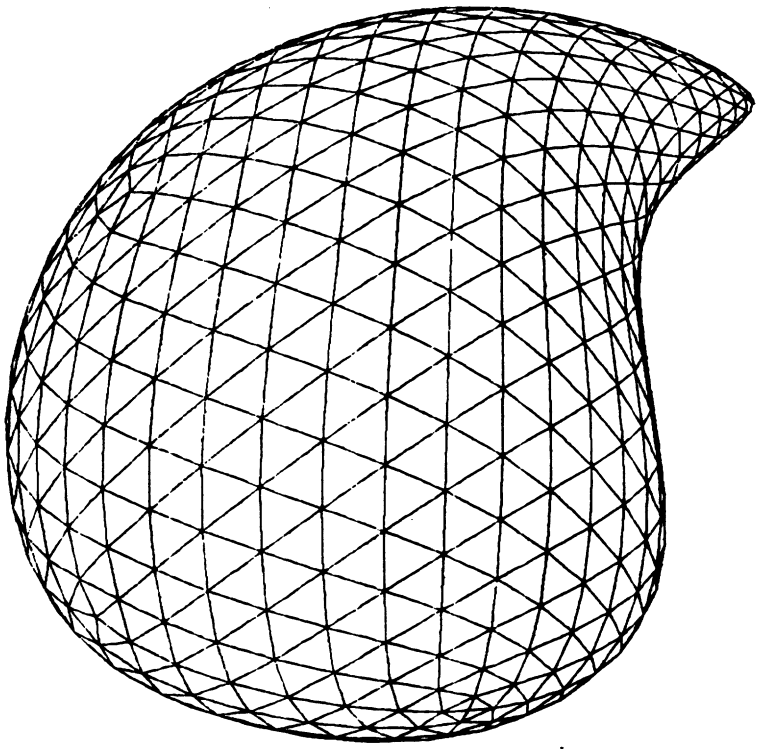


Figure 4.5: A surface using the modified new vertex point rule.

Chapter 5

Subdivision Implemented

An algorithm for refining an arbitrary triangular mesh was roughly outlined in Chapter 3. Chapter 4 presented further details of the algorithm, and some structures used for its analysis. In this chapter, these ideas are incorporated into a pseudo code outline of a working implementation of the subdivision algorithm.

An important aspect of any control point scheme for generating curves or surfaces is the ability to establish relationships among the control points. Two points are considered to be *related* if they are end points of a common edge in a structure. For curves, relationships are trivially established by the ordering of the control points. For example, given a control point $\mathbf{c}_i$, it is clear that $(\mathbf{c}_{i-1}, \mathbf{c}_i)$ and $(\mathbf{c}_i, \mathbf{c}_{i+1})$ are segments of the control polygon. This is also true for surfaces whose control points exhibit a regular grid structure. With appropriate addition and subtraction of the double indices, it is easy to determine the edge sharing neighbors of a control point. In the case of an arbitrary control point mesh, no such simple indexing scheme exists.

In order to establish relationships among control points in an arbitrary mesh, a special data structure must be built. This data structure amounts to a *graph* corresponding to the topology of the mesh. The construction of such a data structure is non-trivial (compared to implementing a regular grid), and it will not be described. The most important feature of this structure is its ability to provide an

ordered *adjacency list* for each control point of the mesh. An ordered adjacency list is a list of points with a common edge sharing neighbor, such that any pair of consecutive points in the list are edge sharing neighbors. In Chapter 4, the set **P** is an ordered adjacency list for the point **V**. The set of all points and associated ordered adjacency list is the required description of the control point mesh.

Each triangular face of the control point mesh is composed of three points denoted **Vr**,**Vs**, and **Vt** with corresponding ordered adjacency lists **Pr**,**Ps**, and **Pt** of lengths Nr,Ns, and Nt respectively. Note that Nr,Ns,and Nt are the vertex orders of **Vr**,**Vs**, and **Vt** respectively. Since the points of a triangle all share edges with one another, each point's ordered adjacency list will contain the other two points. By convention, the first two elements of each point's ordered adjacency list are the other two points of the triangle such that the list has a counterclockwise orientation when viewed from above the triangle. An illustration of the information associated with a single triangular face is given in Figure 5.1. Note that the first element of a point's ordered adjacency list would be different for different triangles. The three ordered adjacency lists associated with the points of each triangle completely determine a *patch* of the subdivision surface. The algorithm SUBDIVIDE is presented in Figure 5.2 which computes a piecewise linear approximation to this patch. The input ℓ indicates the depth of subdivision for the patch.

Note that $*$ is the discrete convolution operator. In practice, this operation computes a sum (see Appendix), which is implemented with a *for* loop. The convolution operands **A** and **M** are defined in Chapter 4. Values of α_{N_i} and β_{N_i} ($i = r, s, t$) may be stored in a table indexed by N_i. The set of points **qr**,**qs**, and **qt** correspond to the adjacency lists of new edge points. These sets are of length 8 rather than 6 (new edge points are ordinary points ($N = 6$)), and $\mathbf{q}i_6 = \mathbf{q}i_0$, $\mathbf{q}i_7 = \mathbf{q}i_1$. This is done so that the starting element may be either $\mathbf{q}i_0$, $\mathbf{q}i_1$, or $\mathbf{q}i_2$, depending on which *sub* patch is to be computed. This is indicated by the set

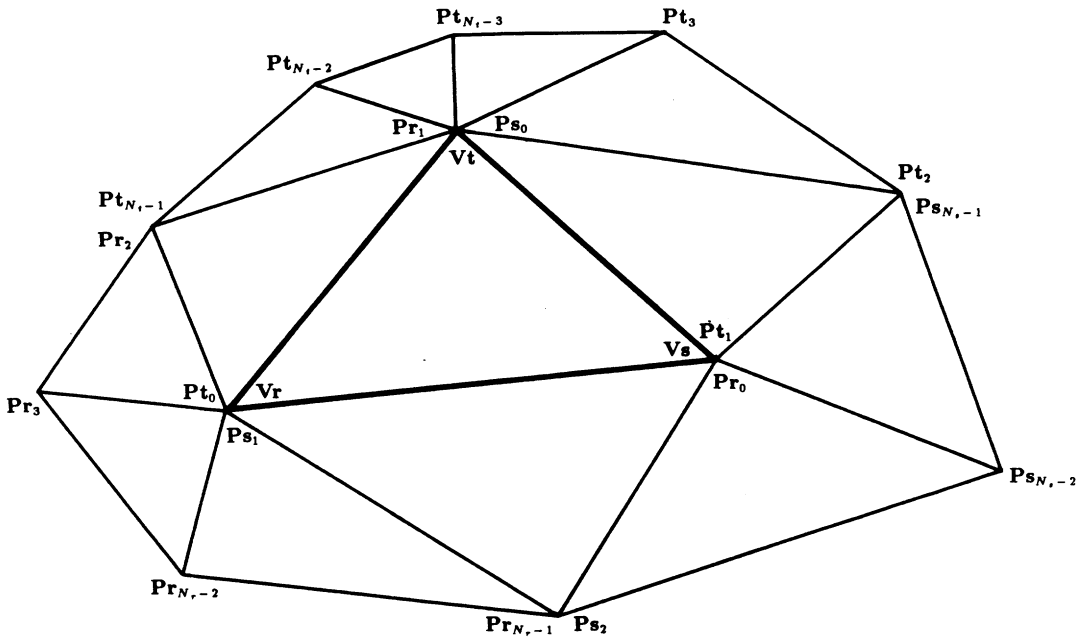


Figure 5.1: The configuration of control points needed to compute one surface patch.

offset in the recursive invocation of SUBDIVIDE.

This algorithm may easily be modified to handle *adaptive* subdivision. Rather than subdividing to a fixed depth, each patch is tested for *flatness*. Patches that are not deemed flat are further subdivided. The surface in Figure 5.3 was computed in this manner.

Using the algorithm just outlined, smooth surfaces may be created with very simple input. The input is a set of control points that roughly approximate a desired shape. The behavior of the resulting surface is completely analogous to much less general splines, for which the technique is a superset.

```
SUBDIVIDE(Pr,Ps,Pt ,Nr,Ns,Nt ,ℓ)
let Vr,Vs,Vt ,vr,vs,vt be points;
let pr,ps,pt be a set {0,...,maxN} of points;
let qr,qs,qt be a set {0,...,7} of points;
Begin
    Vr = Pt_0;      Vs = Pr_0;      Vt = Ps_0;
    if (ℓ > 0) then
        begin
            vr = α_{N_r} Vr + (1 − α_{N_r})A * Pr;
            vs = α_{N_s} Vs + (1 − α_{N_s})A * Ps;
            vt = α_{N_t} Vt + (1 − α_{N_t})A * Pt;

            pr = M * Pr + 3/8 Vr;
            ps = M * Ps + 3/8 Vs;
            pt = M * Pt + 3/8 Vt;

            qr_0 = vt;        qs_0 = vr;        qt_0 = vs;
            qr_1 = pt_0;      qs_1 = pr_0;      qt_1 = ps_0;
            qr_2 = pr_0;      qs_2 = ps_0;      qt_2 = pt_0;
            qr_3 = vs;        qs_3 = vt;        qt_3 = vr;
            qr_4 = ps_{N_s-1}; qs_4 = pt_{N_t-1}; qt_4 = pr_{N_r-1};
            qr_5 = pt_2;      qs_5 = pr_2;      qt_5 = ps_2;
            qr_6 = qr_0;      qs_6 = qs_0;      qt_6 = qt_0;
            qr_7 = qr_1;      qs_7 = qs_1;      qt_7 = qt_1;

            SUBDIVIDE(qr+1, qs+1, qt+1, 6,  6,  6,  ℓ − 1);
            SUBDIVIDE(pr,   qt+2, qs,   Nr, 6,  6,  ℓ − 1);
            SUBDIVIDE(qt,   ps,   qr+2, 6,  Ns, 6,  ℓ − 1);
            SUBDIVIDE(qs+2, qr,   pt ,  6,  6,  Nt ,ℓ − 1);

        end;
    else
        begin
            vr = β_{N_r} Vr + (1 − β_{N_r})A * Pr;
            vs = β_{N_s} Vs + (1 − β_{N_s})A * Ps;
            vt = β_{N_t} Vt + (1 − β_{N_t})A * Pt;

            OUTPUT(vr,vs,vt);

        end;
    end if;
end;
```

Figure 5.2: Pseudo code for the algorithm SUBDIVIDE.

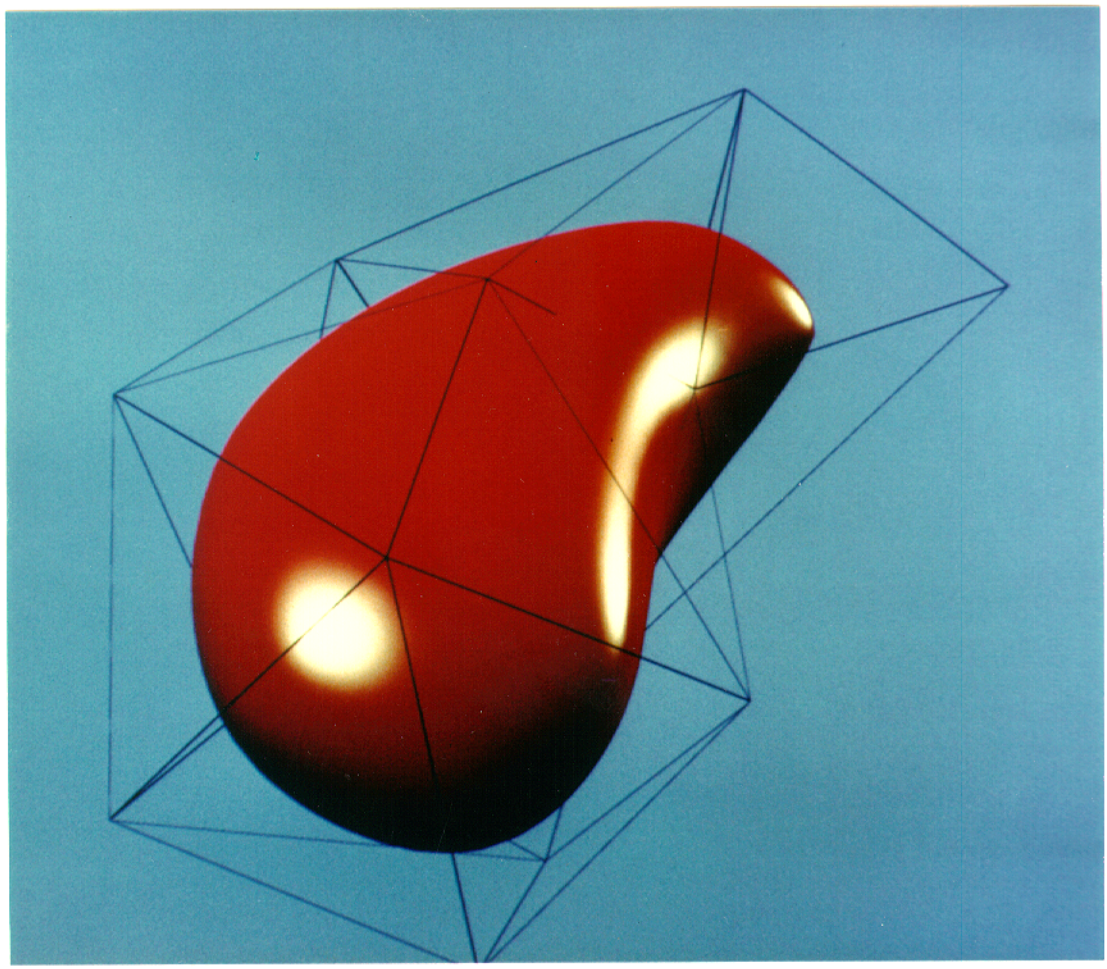


Figure 5.3: A surface and its defining control point set.

Appendix

Discrete Fourier Transforms

This appendix describes a few of the important properties of discrete Fourier transforms that are relevant to this work [Brigham 1974].

Let $\mathbf{G}$ be a discrete function in i with period N. Let $\iota = \sqrt{-1}$, then

$$\mathcal{F}(\mathbf{G})_j = \sum_{i=0}^{N-1} \mathbf{G}_i e^{\frac{2\pi\iota ij}{N}} \qquad j = 0, \ldots, N-1 \tag{A.1}$$

is the discrete Fourier transform of $\mathbf{G}$. If $\mathbf{H} = \mathcal{F}(\mathbf{G})$ then

$$\mathcal{F}^{-1}(\mathbf{H})_i = \frac{1}{N} \sum_{j=0}^{N-1} \mathbf{H}_j e^{\frac{-2\pi\iota ij}{N}} \qquad i = 0, \ldots, N-1 \tag{A.2}$$

is the discrete inverse Fourier transform of $\mathbf{H}$, and $\mathcal{F}^{-1}(\mathbf{H}) = \mathbf{G}$.

If $\mathbf{G}_i = \mathbf{G}_{-i}$, then $\mathbf{G}$ is an *even* function. In this case, the discrete Fourier transform of $\mathbf{G}$ simplifies to

$$\mathcal{F}(\mathbf{G})_j = \sum_{i=0}^{N-1} \mathbf{G}_i \cos \frac{2\pi ij}{N} \qquad j = 0, \ldots, N-1 \tag{A.3}$$

If $\mathbf{H} = \mathcal{F}(\mathbf{G})$ the discrete inverse Fourier transform simplifies to

$$\mathcal{F}^{-1}(\mathbf{H})_i = \frac{1}{N} \sum_{j=0}^{N-1} \mathbf{H}_j \cos \frac{2\pi ij}{N} \qquad i = 0, \ldots, N-1 \tag{A.4}$$

The discrete Fourier transform and its inverse are *linear* transformations. If $\mathbf{G}$ and $\mathbf{H}$ are discrete periodic functions then

$$\mathcal{F}(\mathbf{G} + \mathbf{H}) = \mathcal{F}(\mathbf{G}) + \mathcal{F}(\mathbf{H}). \tag{A.5}$$

The *convolution* of the discrete periodic functions **G** and **H** is defined

$$\mathbf{G} * \mathbf{H} = \sum_{i=0}^{N-1} \mathbf{G}_i \mathbf{H}_{j-i} \qquad j = 0, \ldots, N-1 \tag{A.6}$$

where $*$ is the convolution operator. The *convolution theorem* for discrete Fourier transforms states :

$$\mathcal{F}(\mathbf{G} * \mathbf{H}) = \mathcal{F}(\mathbf{G})\mathcal{F}(\mathbf{H}). \tag{A.7}$$

The discrete Fourier transform of a few simple and useful periodic function are illustrated in Figure A.1.

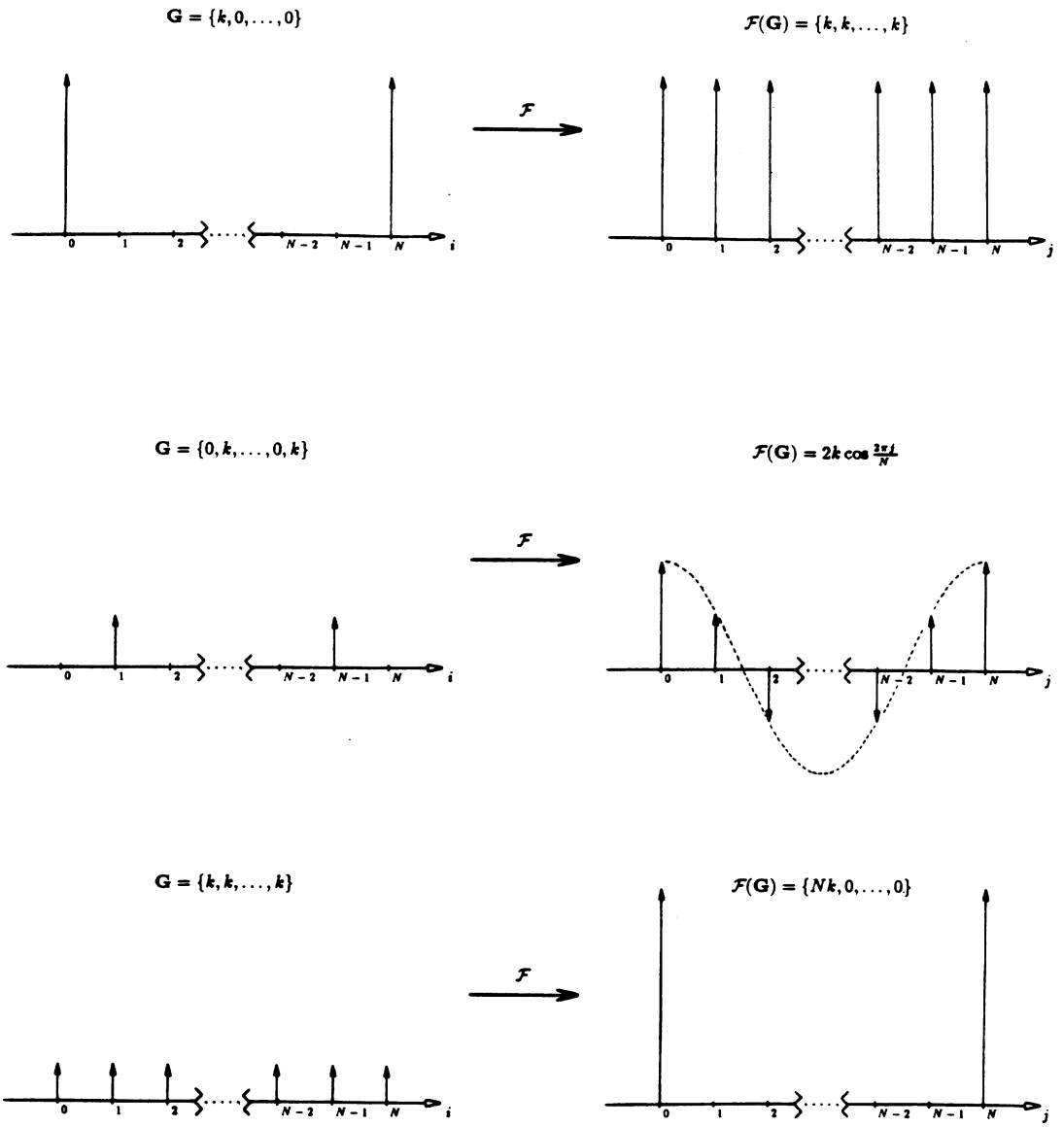


Figure A.1: Some discrete periodic functions and their Fourier transforms

References

Boehm, W. (1985), *Triangular spline algorithms*, Computer Aided Geometric Design, **2**, 61-67.

Boehm, W. (1984), *Calculating with box splines*, Computer Aided Geometric Design, **1**, 149-162.

Boehm, W., Farin, G., and Kahmann, J. (1984), *A survey of curve and surface methods in CAGD*, Computer Aided Geometric Design **1**, 1-60.

Boehm, W. (1983), *Generating the Bezier points of triangular splines*, in **Surfaces in Computer Aided Geometric Design**, Barnhill, R. E. and Boehm, W., eds., North-Holland, Amsterdam.

Boehm, W. (1983), *The de Boor algorithm for triangular splines*, in **Surfaces in Computer Aided Geometric Design**, Barnhill, R. E. and Boehm, W., eds., North-Holland, Amsterdam.

Boehm, W. (1983), *Subdividing multivariate splines*, Computer Aided Design **15**, 354-352.

Boehm, W. (1980), *Inserting new knots into B-spline curves*, Computer Aided Design **12**, 99-110.

Brigham, E. O. (1974), **The Fast Fourier Transform**, Prentice-Hall, New Jersey.

Catmull, E. and Clark, J. (1978), *Recursively generated B-spline surfaces on arbitrary topological meshes*, Computer Aided Design **10**, 350-355.

Chaikin, G. M. (1974), *An Algorithm for High-Speed Curve Generation*, Computer Graphics and Image Processing **3**, 346-349.

de Boor, C. (1986), *B(asic)-Spline Basics*, Extension of B-spline curve algorithms to surfaces (Siggraph course notes 5) Dallas.

de Boor, C. (1978), **A practical guide to splines**, Springer-Verlag, New York.

Dahmen, W. and Micchelli, C. (1984), *Subdivision algorithms for the generation of box spline surfaces*, Computer Aided Geometric Design **1**, 115-129.

Doo, D. and Sabin, M. (1978), *Behavior of recursive division surfaces near extraordinary points*, Computer Aided Design **10**, 356-360.

Doo, D. (1978), *A subdivision algorithm for smoothing down irregularly shaped polyhedrons*, Proceedings on Interactive Techniques in Computer Aided Design Bologna.

Farin, G. (1986), *Triangular Bernstein-Bezier patches*, Computer Aided Geometric Design **3**, 83-127.

Goldman, R. (1983), *Subdivision algorithms for Bezier triangles*, Computer-aided Design **15**, 159-166.

Micchelli, C. (1986), *Subdivision algorithms for curves and surfaces*, Extension of B-spline curve algorithms to surfaces (Siggraph course notes 5) Dallas.

Riesenfeld, R. F. (1975), *On Chaikin's Algorithm*, Computer Graphics and Image Processing **4**, 304-310.