RECURSIVE SUBDIVISION ALGORITHMS FOR CURVE AND SURFACE DESIGN

by

RUIBIN QU

Department of Mathematics And Statistics,

Brunel University,

Uxbridge, Middlesex, England.

Thesis submitted for the degree of

Doctor of Philosophy at Brunel University, August 1990

ABSTRACT

In this thesis, the author studies recursive subdivision algorithms for curves and surfaces. Several subdivision algorithms are constructed and investigated. Some graphic examples are also presented.

Inspired by the *Chaikin's* algorithm and the *Catmull-Clark's* algorithm, some non-uniform schemes, the non-uniform corner cutting scheme and the recursive subdivision algorithm for non-uniform B-spline curves, are constructed and analysed. The adapted parametrization is introduced to analyse these non-uniform algorithms. In order to solve the surface interpolation problem, the *Dyn-Gregory-Levin's* 4-point interpolatory scheme is generalized to surfaces and the 10-point interpolatory subdivision scheme for surfaces is formulated. The so-called *Butterfly Scheme*, which was firstly introduced by *Dyn, Gregory Levin* in 1988, is just a special case of the scheme. By studying the *Cross-Differences of Directional Divided Differences*, a matrix approach for analysing uniform subdivision algorithms for surfaces is established and the convergence of the 10-point scheme over both uniform and non-uniform triangular networks is studied. Another algorithm, the subdivision algorithm for uniform bi-quartic B-spline surfaces over arbitrary topology is introduced and investigated. This algorithm is a generalization of *Doo-Sabin's* and *Catmull-Clark's* algorithms. It produces uniform Bi-quartic B-spline patches over uniform data. By studying the local subdivision matrix, which is a circulant, the tangent plane and curvature properties of the limit surfaces at the so-called Extraordinary Points are studied in detail.

ABSTRACT

In this thesis, the author studies recursive subdivision algorithms for curves and surfaces. Several subdivision algorithms are constructed and investigated. Some graphic examples are also presented.

Inspired by the *Chaikin's* algorithm and the *Catmull-Clark's* algorithm, some non-uniform schemes, the non-uniform corner cutting scheme and the recursive subdivision algorithm for non-uniform B-spline curves, are constructed and analysed. The adapted parametrization is introduced to analyse these non-uniform algorithms. In order to solve the surface interpolation problem, the *Dyn-Gregory-Levin's* 4-point interpolatory scheme is generalized to surfaces and the 10-point interpolatory subdivision scheme for surfaces is formulated. The so-called *Butterfly Scheme*, which was firstly introduced by *Dyn, Gregory Levin* in 1988, is just a special case of the scheme. By studying the *Cross-Differences of Directional Divided Differences*, a matrix approach for analysing uniform subdivision algorithms for surfaces is established and the convergence of the 10-point scheme over both uniform and non-uniform triangular networks is studied. Another algorithm, the subdivision algorithm for uniform bi-quartic B-spline surfaces over arbitrary topology is introduced and investigated. This algorithm is a generalization of *Doo-Sabin's* and *Catmull-Clark's* algorithms. It produces uniform Bi-quartic B-spline patches over uniform data. By studying the local subdivision matrix, which is a circulant, the tangent plane and curvature properties of the limit surfaces at the so-called Extraordinary Points are studied in detail.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr ***John A. Gregory*** *for his constant help and supervision during the course of this work.*

I wish to thank ***The Chinese Educational Commission*** *and* ***The British Council*** *for the financial support* ***(SBFSS/1987)*** *of this work; especially, the* ***Educational Section*** *of* ***The Chinese Embassy*** *in London, and my Programme Officer in* ***The British Council****, Mrs.* ***Deborah Clarke****, for their constant help.*

I am deep in debt to my wife, ***Nancy GONG****, for her love and unceasing support and assistance. Without her love and encouragement, I would never have completed this thesis.*

I also thank ***Dr V. K. H. Lau*** *at Brunel University for his reading of the manuscript and many valuable suggestions.*

Finally, I also wish to thank all the members of the ***CAGD*** *group, the staff members of the Department of Mathematics and Statistics who helped me in one way or another during the course of this work.*

CONTENTS

INTRODUCTION

With the development of science and technology, shape design techniques have been forging ahead tremendously, especially in the past few decades, to meet the increasing demands for machine tool design and manufacture. The *CAD/CAM* systems, the *UNISURF* system and various *CAGD* systems are only a few examples of this development. It is obvious that the recursive subdivision technique plays a more and more important role in this development.

Recursive subdivision methods consist of a class of numerically stable, highly efficient, easily manipulated and implemented algorithms for the generation of parametric curves and surfaces. All these methods use the idea that the desired curves and surfaces are to be generated from some finite points, called *control points, control polygon* or *control net,* by some iterative methods consisting predominantly of simple *local weighting processes*. Some of them can be explained as generalized corner cutting algorithms. For example, algorithms with the convex-hull property, such as *de Casteljau's* and *Catmull-Clark's* algorithms, are this type of algorithm. These algorithms are eminently suited for use in interactive computer aided design systems because the produced curves and surfaces are smooth and can be controlled locally by adjusting corresponding parameters. Therefore, they are very popular and widely used in computer systems. Hence, more and more such algorithms are being studied.

In order to study mathematically the properties of the the schemes as well as their generated curves and surfaces, a variety of techniques are introduced: *Dyn-Gregory-Levin's* generator matrix analysis, *Micchelli-Prautzsch's* invariant approach and *Cavaretta-Dahmen-Micclelli's* regular subdivision technique. These methods are only suitable for analysing the so-called *uniform subdivision schemes*. The non-uniform schemes, which are very useful in practice but difficult to analyse, are now being studied.

In order to develop the existing curve and surface design techniques and mathematical analyses, the thesis is concerned with the construction and mathematical analysis of recursive subdivision algorithms, especially for non-uniform subdivision schemes and interpolatory schemes. The major part of our research work is contained in Chapter 3, 5 and 6. Chapter 1 is a brief review of recursive subdivision algorithms. Some currently used examples are briefly described. Chapter 2 is a survey of mathematical methods used to analyse subdivision algorithms for curves and surfaces. In Chapter 3, we study the non-uniform subdivision scheme for smooth curve generation and derive a recursive subdivision algorithm for B-spline curves with simple knots. The *Adapted Parametrization* technique is introduced to analyse non-uniform subdivision schemes. Chapter 4 describes the existing surface generating algorithms and their corresponding mathematical descriptions. In Chapter 5, we derive the subdivision algorithm for uniform bi-quartic B-spline surfaces and generalize it to arbitrary networks. The tangent plane and (normal) curvature properties of the limit surface at *extraordinary points* are studied. Using the *Block Circulant Matrix* technique, the *Ball-Storry's* method for C^1 and C^2 surface analyses at an *extraordinary point* is generalized. In Chapter 6, we study a *10-point interpolatory subdivision scheme* for surfaces over both uniform and non-uniform triangular control nets. The *Cross-Differences of Directional Divided Difference* approach for analysing uniform subdivision algorithms for surfaces is presented. Using this method, the necessary and sufficient conditions for the 10-point interpolatory scheme to produce C^0 and C^1 surfaces are investigated in details. Chapter 7 is a brief summary of the thesis.

The concern of this thesis is the mathematics and the techniques of analyses for recursive subdivision schemes and so the applications of these algorithms will not be discussed further. However, most of the discussed schemes have been implemented in *FORTRAN*. Hence, some computer graphics are included.

CHAPTER ONE

A REVIEW OF RECURSIVE SUBDIVISION ALGORITHMS

In this Chapter, we present a brief introduction to the development of recursive subdivision algorithms. Some mathematical notations are introduced and some of the most widely used examples are also described.

1.1. A Brief History of Recursive Subdivision Techniques

Recursive subdivision algorithms can be viewed as a class of iterative algorithms that are used to calculate, to generate and to approximate curves and surfaces. Their main feature is that they use initially only some finite data points (control points) to finally produce continuous, or even differentiable curves or surfaces. These methods are mainly based on using some form of local line averaging processes. The most important advantage of using these algorithms in interactive computer aided design is that they are numerically stable, very efficient, locally adjustable and easily implemented.

Although *Recursive Subdivision Algorithms*, or ***RSA*** for short, have been used for curve and surface generation for a long time by both mathematicians and other scientists and technicians, they only received a good deal of research attention in recent years. Some of the early examples are the *Carpenter's Technique* which was used to produce a smooth corner from a sharp angular corner with simple tools [46] and the corner cutting technique that was used by some ancient *Chinese* mathematicians to approximate a circle from a regular hexagon by repeatedly chopping off corners in order to find the circumference of the circle [72].

About forty years ago, *de Rahm* studied a subdivision algorithm from a mathematical point of view in order to find some generally singular functions (in the sense of *Lebesgue*). This triggered off the modern era for the investigation of subdivision algorithms. He used a functional equation to introduce the curve subdivision idea and then studied a simple corner cutting algorithm, the *"trisection algorithm"*, in detail in [101,102]. In the early sixties, *de Casteljau* from the French car company *Citroen* developed an iterative line averaging algorithm, the *de Casteljau* algorithm (a subdivision algorithm), for the calculation of *Bernstein-Bezier* curves [9,26]. These techniques play a central role in the rapid development of recursive subdivision algorithms.

With the advent of computers, recursive subdivision algorithms are being used increasingly in approximation theory and computer aided geometric design as a method for the generation and definition of curves and surfaces. Consequently, more and more attentions are being given to the studies of stable and efficient algorithms for the computation of curves and surfaces. Other problems, such as efficiency and user friendly software in interactive computer aided design, also arise. As a result, *de Boor* and *Cox* derived independently a recursive algorithm for the calculation of B-spline curves [24,37]; and *G.M. Chaikin*, in 1974, constructed an algorithm (*Chaikin's* algorithm) for high speed curve generation [30]. In 1975, *Riesenfeld* proved that the curves produced by *Chaikin's* algorithm were uniform quadratic B-spline curves [103]. The success of the studies of these algorithms encouraged more and more studies in curve and surface generation algorithms [1,28,33,69,71,76,81,85,92,...]

In 1978, *Doo* and *Sabin* constructed and analysed a surface generating recursive subdivision algorithm over arbitrary topology (*Doo-Sabin's* algorithm) [44,46]. In the same year, *Catmull* and *Clark*, in order to seek subdivision algorithms producing smoother fitted patches over arbitrary networks, developed a subdivision algorithm for uniform bi-cubic B-spline patches (*Catmull-Clark's* algorithm) [27]. This algorithm was then fully analysed by *Ball* and *Storry* in [1,2,3,116]. Since then, a huge amount of research work on subdivision algorithms and techniques have been developed.

The studies and implementations of recursive subdivision algorithms has been developing very fast in the recent years. As a result, many new and powerful methods and techniques are being developed to construct and analyse more and more useful, flexible and complicated schemes. In the next section, we give a brief list of some of the important results about recursive subdivision algorithms. These include the *Oslo* algorithm [33] and *Boehm's* knot insertion algorithm for the calculation of B-spline curves [14]; the *DGL* (*Dyn-Gregory-Levin*) scheme for curve and surface design [50,51]; the *Micchelli-Prautzsch* invariant curve technique [85,86,88-90]; the regular subdivision approach [28,29] (or generating polynomial analysis) of *Cavaretta-Dahmen-Micchelli* and some analyses of non-uniform subdivision schemes [53,70,71,81].

1.2. Examples of Recursive Subdivision Algorithms

In this section, for the sake of notational convenience, some mathematical notations are introduced. These notations will be used throughout the whole thesis to describe recursive subdivision algorithms. Some examples of currently used subdivision algorithms are also given.

1.2.1. Mathematical Representations of Recursive Subdivision Algorithms

What is a recursive subdivision algorithm? How to describe it mathematically? In order to get some intuitive ideas about subdivision algorithms, we describe some examples of the currently used recursive subdivision algorithms below.

For simplicity, we suppose that the control points for curves are in $\boldsymbol{R}^2$ and the control points for surfaces are in $\boldsymbol{R}^3$. We also assume that the curves and surfaces are functions instead of parametric curves and surfaces. Unless stated, the *Uniform Parametrization* is assumed (this will be discussed in detail in Chapter 2).

In order to describe the algorithm clearly and mathematically, some terminology and mathematical notations are needed. In the curve case, a *control polygon* $\{f_0, f_1, f_2, f_3, \dots, f_n\}$, or just $\{f_i\}$ for short, is a piecewise linear curve which interpolates the ordered data $\{f_0, f_1, f_2, \dots, f_n\}$. Any parametrization of the piecewise linear curve is called the parametrization of the control polygon, which is also called the parametrization of the limit curve if the limit of the control polygon sequence is considered.

The basic idea of recursive subdivision algorithms is quite simple. Given an initial control polygon $\{f_i^0\}$, a ***RSA*** uses certain rules to generate a refined control polygon $\{f_i^1\}$ and this process is repeated successively to obtain $\{f_i^k\}$, the control polygon at level k, $k = 1, 2, \dots$. Therefore, the ***RSA*** can be described by some formulae corresponding to the rules which relates the control polygon $\{f_i^{k+1}\}$ and $\{f_i^k\}$. Hence, in the thesis, we use the refinement formulae to describe its scheme for both curves and surfaces.

In the surface case, the *control polygon* $\{f_{i,j}\}$, or sometimes called *control net* or *control polyhedron*, is the piecewise linear (for triangular networks) or piecewise bi-linear (for tensor-product type schemes) surface which interpolates the ordered data $\{f_{i,j}\}$. These control polygons may have some "*holes*" near the so-called *extraordinary points*, where some special treatments are used to make the control polygons continuous. However, this irregular situation will not be dealt with in this Chapter.

1.2.2. Some Currently Used Recursive Subdivision Algorithms

In this subsection, we list some important algorithms in curve and surface calculation and generation. Although some of them are not recursive subdivision algorithms, in the sense that we define them in this thesis, for example, the *de Boor* algorithm and the *Boehm's* knot insertion algorithm, they provide an important tool for the analyses of certain subdivision algorithms.

Example 1.1. The Carpenter's Technique

No one knows when the *Carpenter's Technique* was firstly used by carpenters though it is one of the oldest and the most popular example of ***RSA***. The technique is to use simple tools to produce smooth corners by a *corner cutting* process, as it is called now.

The technique is like this. Suppose a corner B is to be smoothed off from A to C (Figure 1.1), the carpenter will divide AB and BC into an equal number of portions. By sawing along the straight lines joining the corresponding markings, as shown in the figure, a fairly smooth curve can be obtained.

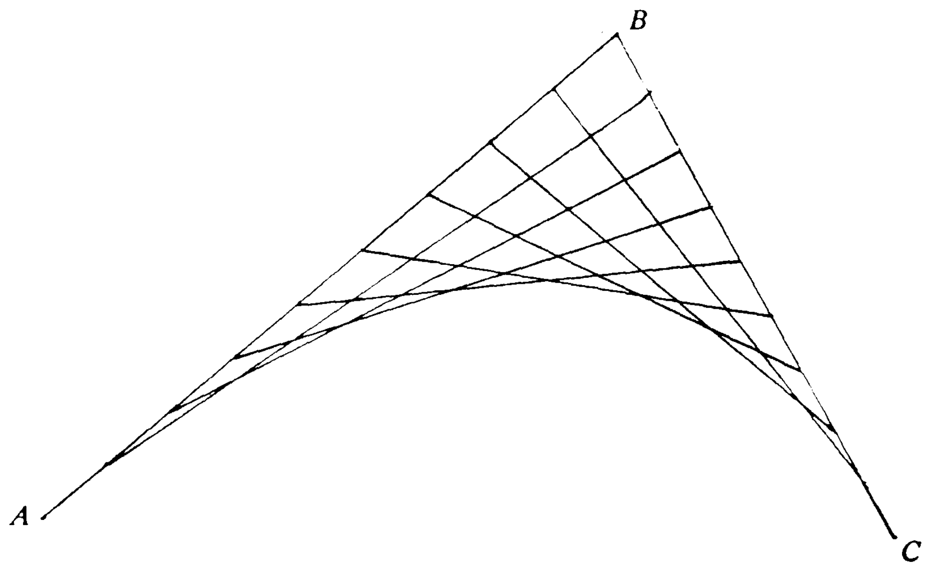


Figure 1.1.

It can be proved that the line segments left form the "*envelope*" of the sawing lines if the number of portions is infinitely large. Furthermore, the envelope is a *parabolic curve* [46,115].

There are many ways to prove the result. A simple proof can be obtained by comparing the algorithm with the *de Casteljau's* algorithm, or *Chaikin's* algorithm, for quadratics, which are described later in this subsection. The technique is exactly the same as the *Chaikin's* algorithm if the number of portions goes to infinity in a doubling way 2, 4, 8, 16, 32, ... So from the known result of *Chaikin's* algorithm we know that the resulting smooth curve is a *parabolic* segment.

Example 1.2. The De Casteljau Algorithm

In 1959, *de Casteljau* formulated an algorithm for the computation of polynomial curves [26]. The algorithm for quadratics is the same as the *Carpenter's* algorithm and the *Chaikin's* algorithm (locally). So the algorithm for cubic polynomial curves is presented here.

Given four control points, say $\{f_i\}$, the cubic *Bernstein-Bezier* polynomial curve is defined by:

$$(1.2) \quad P(t) := \sum_{i=0}^{3} f_i \, B_{i,3}(t), \qquad \textit{for} \quad 0 \leq t \leq 1,$$

where, $B_{i,3}(t)$ is the *Bernstein-Bezier* basis of the cubic polynomials:

$$(1.2) \qquad B_{i,3}(t) := \frac{6}{i!(3-i)!} t^i (1-t)^{3-i}, \text{ for } 0 \leq t \leq 1.$$

The *de Casteljau* algorithm asserts that the curve $P(t)$ can be split into two cubic segments at any point on the curve and that their control polygons can be calculated from the original one by a line averaging algorithm. For example, suppose the curve is split at the ***midpoint*** $P(1/2)$, then the new control polygons are given by the following formulae (Figure 1.2):

$$(1.3) \quad \left| \begin{array}{l} f_0^L := f_0 \\ f_1^L := (f_0 + f_1)/2 \\ f_2^L := (f_0 + 2f_1 + f_2)/4 \\ f_3^L := (f_0 + 3f_1 + 3f_2 + f_3)/8 \\ f_0^R := (f_0 + 3f_1 + 3f_2 + f_3)/8 \\ f_1^R := (f_0 + 2f_1 + f_3)/4 \\ f_2^R := (f_2 + f_3)/2 \\ f_3^R := f_3. \end{array} \right.$$

If we define:

(1.4) $P(t) := \sum_{i=0}^{3} f_i B_{i,3}(t)$, $P^L(t) := \sum_{i=0}^{3} f_i^L B_{i,3}(t)$ and $P^R(t) := \sum_{i=0}^{3} f_i^R B_{i,3}(t)$

then

(1.5) $$P(t) = \begin{cases} P^L(2t) & \text{for} \quad 0 \le t \le 1/2 \\ P^R(2t-1) & \text{for} \quad 1/2 \le t \le 1. \end{cases}$$

The importance of the algorithm lies in the fact that the produced curve has the *invariant* property, that is, the original curve can be split into two parts and each part can be represented in the same form as the original curve. It is just this property that leads to the recent intensive investigation of invariant curves and their corresponding subdivision algorithms [41,42,85] and [86,88,90...]. From this example, it can be easily shown, by some simple calculations, that the new, subdivided control polygons are much *smaller* (the convex hull) and *smoother* (less variation of the control polygons) than that of the original one. Another important idea in recursive subdivision algorithms, the so called *Bernstein Bezier Polygon (net) Iteration* technique, is initiated. These studies lead to the rapid development of the ***RSA***.

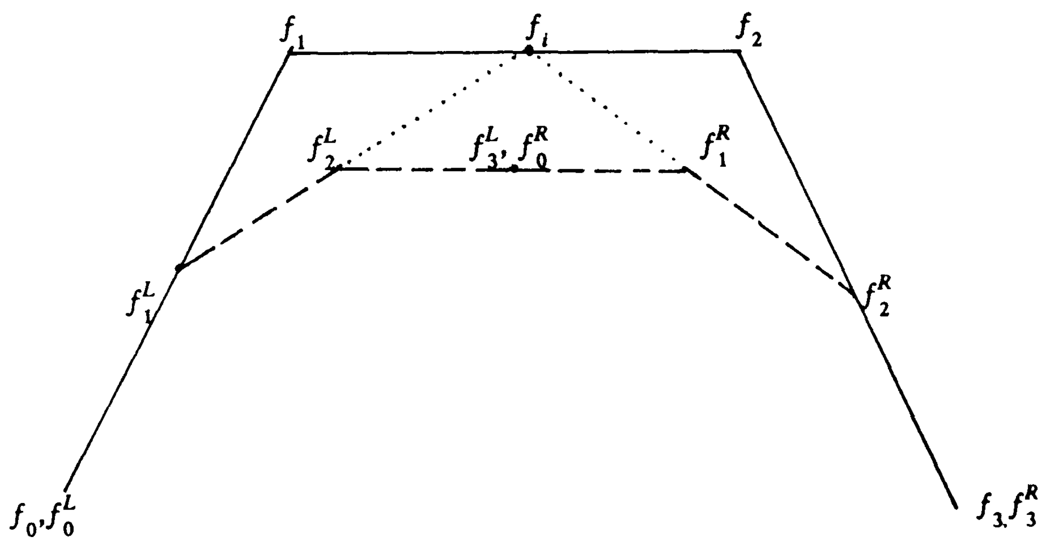


Figure 1.2.

Because of the stability, efficiency and simplicity of the algorithm, many computations concerning piecewise polynomial curves are firstly transformed into their equivalent *Bernstein-Bezier* forms and then manipulated [56,60].

Example 1.3. The Chaikin's Algorithm

In 1974, *G.M. Chaikin* derived a high speed curve generation algorithm from data points [30]. The algorithm is the very case *de Rahm* omitted to study [101,102]. The study of the algorithm uncovers an area of control point representation of curves and surfaces, sometimes referred to as *"discrete curves and surfaces"* [122].

The motivation of the algorithm is to generate smooth curves by a series of continuous piecewise linear segments, the *control polygon* series as it is called now, which can be computed sequentially by a local, simple and adjustable algorithm from the previous ones. The algorithm is as follows.

Given the data $\{f_i^0\}$, which is also called the initial control polygon (Figure 1.3), the algorithm generates a series of control polygons $\{f_i^k\}$, k=1, 2, ... by the following formulae (*mask*):

$$(1.6) \quad \left| \begin{array}{ll} f_{2i}^{k+1} & = (3f_i^k + f_{i+1}^k)/4 \\ f_{2i+1}^{k+1} & = (f_i^k + 3f_{i+1}^k)/4. \end{array} \right.$$

The curve of the algorithm is defined as the limit of the control polygon sequence. In 1975, *Riesenfeld* proved that the limit curve was just a uniform quadratic B-spline curve [103]. The result could also be obtained by comparing the algorithm with the *de Casteljau* algorithm for quadratics or by using *Boehm's* knot insertion algorithm for uniform quadratic B-spline curves [14].

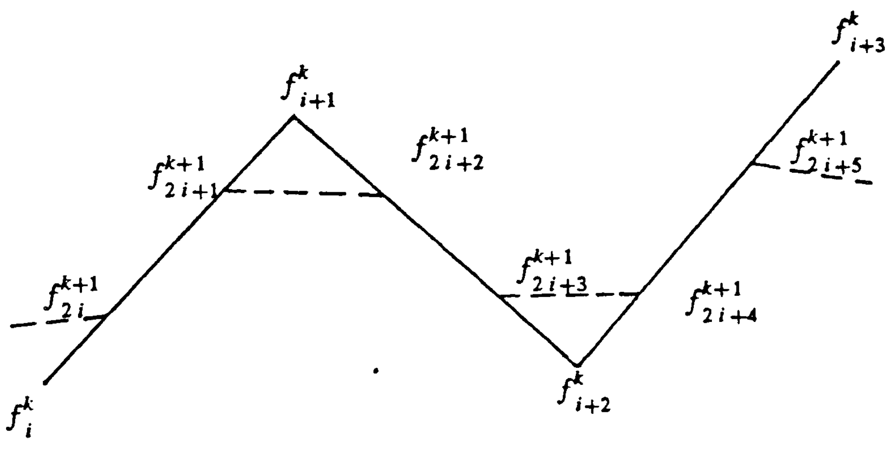


Figure 1.3.

Example 1.4. The Catmull-Clark Algorithm (for curves)

As the demands for smooth surface generation algorithms increased, in 1978, *Catmull* and *Clark* [27] and *Doo* and *Sabin* [45] formulated an algorithm for surfaces. Although the algorithm was derived for surfaces, it mainly came from the corresponding algorithm for the generation of uniform cubic B-spline curves since it is a tensor-product type algorithm.

In the curve case, the idea of the algorithm is the same as the *Chaikin's* algorithm. More explicitly, the algorithm produces cubic B-spline curves with uniform knots partition from given data points $\{f_i^0\}$. The subdivision equations:

$$
(1.7) \quad \left| \begin{array}{ll} f_{2i}^{k+1} & = (f_i^k + f_{i+1}^k)/2 \\ f_{2i+1}^{k+1} & = (f_i^k + 6f_{i+1}^k + f_{i+2}^k)/8 \end{array} \right.
$$

characterize the subdivision algorithm. Figure 1.4 shows the subdivision process.

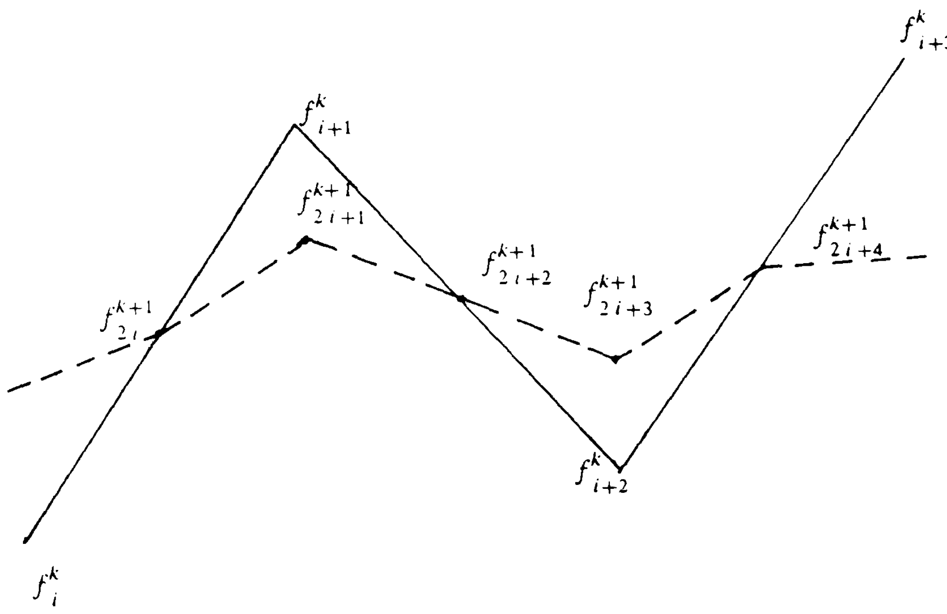


Figure 1.4.

Example 1.5. The Uniform Quartic B-spline Algorithm

This algorithm is a generalization of the *Chaikin's* and *Catmull-Clark's* algorithm for the generation of uniform quartic B-spline curves. It is shown in [50,76,85] that any uniform B-spline curve can be produced by such an algorithm (this can also be proved by *Boehm's* knot insertion algorithm). The process is similar to those described in example 1.3 and 1.4. Thus, it is suffices to just give the subdivision formulae of the algorithm:

$$(1.8)\quad \left| \begin{array}{ll} f_{2i}^{k+1} & = (5f_i^k + 10f_{i+1}^k + f_{i+2}^k)/16 \\ f_{2i+1}^{k+1} & = (f_i^k + 10f_{i+1}^k + 5f_{i+2}^k)/16. \end{array} \right.$$

Figure 1.5 shows the smoothing process.

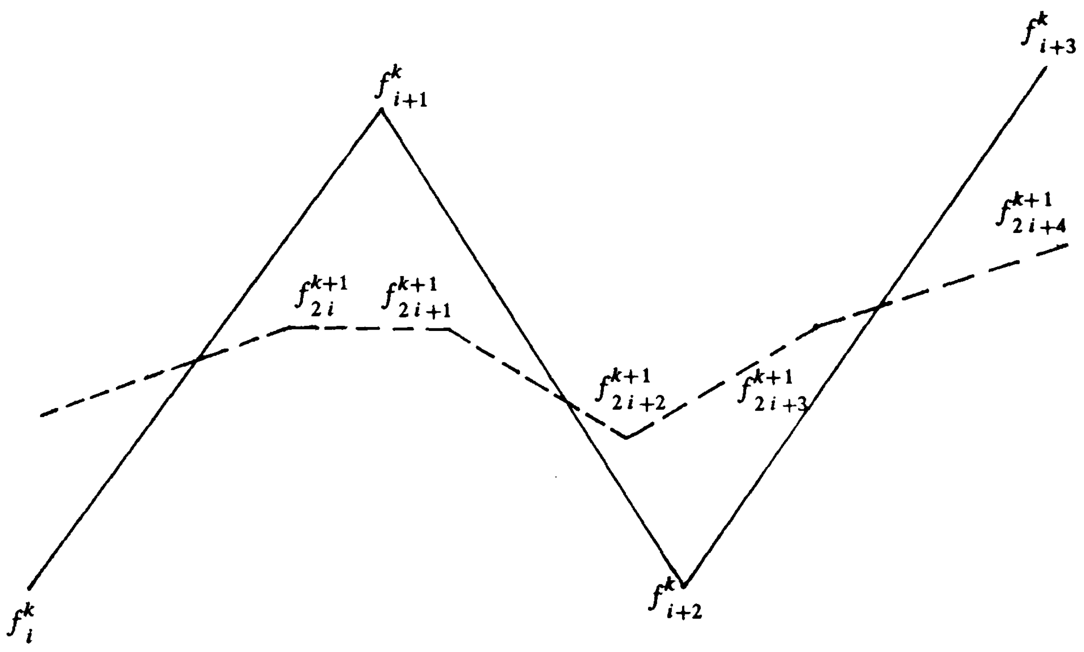


Figure 1.5.

Example 1.6. Non-uniform Corner Cutting Algorithm

This algorithm is just a complement of the *Chaikin's* algorithm for smooth curve generation. The motivation came from *de Boor's* corner cutting studies [21] and *de Rahm's* original works [101,102]. The algorithm seeks (sufficient) conditions for smooth corner cutting conditions (Chapter 3). It is proved that the scheme produces smooth curves if a proper parametrization (*adapted parametrization*) is used. However, it should be noted that the result thus obtained can not be proved by using the *"diadic parametrization"* technique as it is used in [50,51,85,86]. The scheme will be studied in detail in Chapter 3.

Example 1.7. The Subdivision Algorithm for Non-uniform B-spline Curves

Since both the *Chaikin's* algorithm and the *Catmull-Clark's* algorithm are for the generation of B-spline curves with uniform knot partition, we present here a generalization of these algorithms such that the B-spline curves with simple knots can be produced in a similar way. The scheme can be regarded as a generalisation of *Boehm's* knot insertion or a special result of the *Oslo* algorithm (Chapter 3). The difference is that the scheme uses a special simultaneous knot insertion technique (knot doubling process) so that the whole spline curve can be approximated.

In order to give an outline of the subdivision algorithm, we present only the schemes for cubic and quartic B-spline curves. Their corresponding subdivision formulae are given by equations (1.9) and (1.10) respectively. The subdivision processes are shown in Figure 1.7*a* and 1.7*b*.

$$(1.9)\qquad \left|\begin{array}{l} f_{2i}^{k+1} = (1-a_i^k)\, f_i^k + a_i^k\, f_{i+1}^k \\ f_{2i+1}^{k+1} = b_i^k\, f_i^k + (1-b_i^k-c_i^k) f_{i+1}^k + c_i^k\, f_{i+2}^k \end{array}\right.$$

and

$$(1.10)\qquad \left|\begin{array}{l} f_{2i}^{k+1} = u_i^k\, f_i^k + (1-u_i^k-v_i^k) f_{i+1}^k + v_i^k\, f_{i+2}^k \\ f_{2i+1}^{k+1} = x_i^k\, f_i^k + (1-x_i^k-y_i^k) f_{i+1}^k + y_i^k\, f_{i+2}^k, \end{array}\right.$$

where the weights $\{a_i^k, b_i^k, c_i^k, u_i^k, v_i^k, x_i^k, y_i^k\}$ are determined by the chosen knots $s_i^k < s_{i+1}^k$ and the chosen shape parameter t_i^k, $0 < t_i^k < 1$, in the following way:

$$(1.11)\quad \left|\begin{array}{ll} s_{2i}^{k+1} & = s_i^k \\ s_{2i+1}^{k+1} & = (1-t_i^k)\, s_i^k + t_i^k\, s_{i+1}^k \end{array}\right.$$

and

$$(1.12)\quad \left|\begin{array}{lll} a_i^k & = & \dfrac{(s_{i+1}^k - s_{2i-1}^{k+1})(s_{i+1}^k - s_{i-1}^k)}{(s_{i+1}^k - s_{i-1}^k)(s_{i+1}^k - s_{i-2}^k)} \\[2ex] b_i^k & = & \dfrac{(s_{i+1}^k - s_{2i+1}^{k+1})}{(s_{i+1}^k - s_{i-1}^k)}\, a_i^k \\[2ex] c_i^k & = & \dfrac{(s_{2i+1}^{k+1} - s_{i-1}^k)}{(s_{i+1}^k - s_{i-1}^k)}\, (1 - a_{i+1}^k) \end{array}\right.$$

and

$$(1.13)\quad \left|\begin{array}{lll} u_i^k & = & \dfrac{(s_{i+5}^k - s_{2i+5}^{k+1})\,(s_{i+5}^k - s_{2i+7}^{k+1})}{(s_{i+5}^k - s_{i+1}^k)(s_{i+5}^k - s_{i+2}^k)} \\[2ex] v_i^k & = & \dfrac{(s_{2i+5}^{k+1} - s_{i+4}^k)\,(s_{2i+7}^{k+1} - s_{i+4}^k)}{(s_{i+5}^k - s_{i+2}^k)(s_{i+6}^k - s_{i+2}^k)} \\[2ex] x_i^k & = & \dfrac{(s_{2i+9}^{k+1} - s_{2i+7}^{k+1})\,(s_{i+5}^k - s_{2i+7}^{k+1})}{(s_{i+5}^k - s_{i+1}^k)\,(s_{i+5}^k - s_{i+2}^k)} \\[2ex] y_i^k & = & \dfrac{(s_{2i+7}^{k+1} - s_{i+4}^k)\,(s_{2i+9}^{k+1} - s_{i+4}^k)}{(s_{i+5}^k - s_{i+2}^k)\,(s_{i+6}^k - s_{i+2}^k)}\,. \end{array}\right.$$

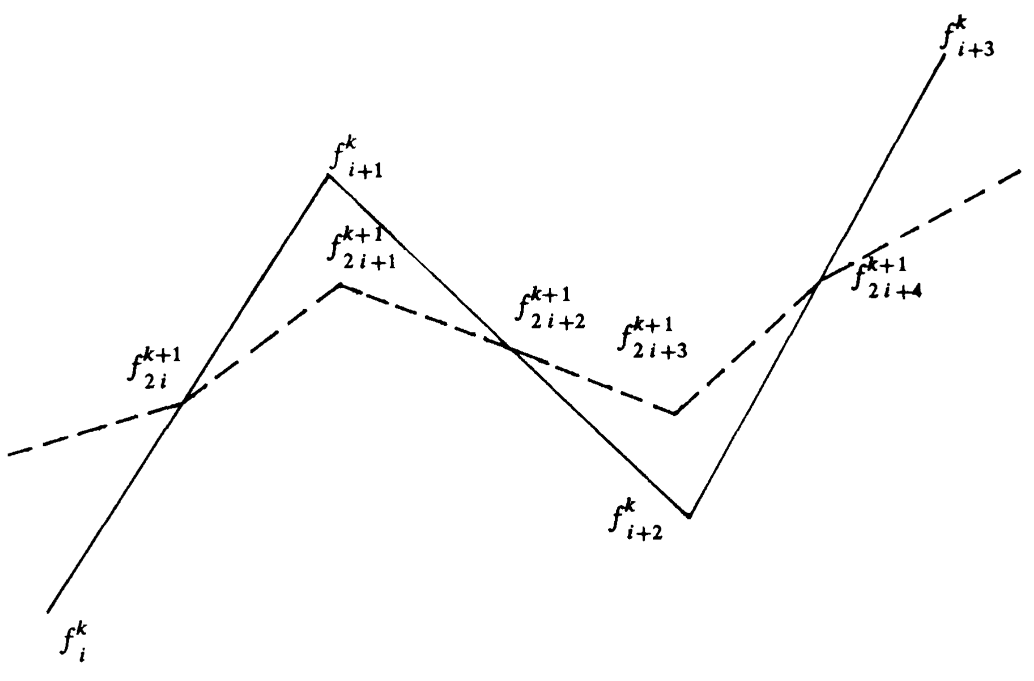


Figure 1.7*a*.

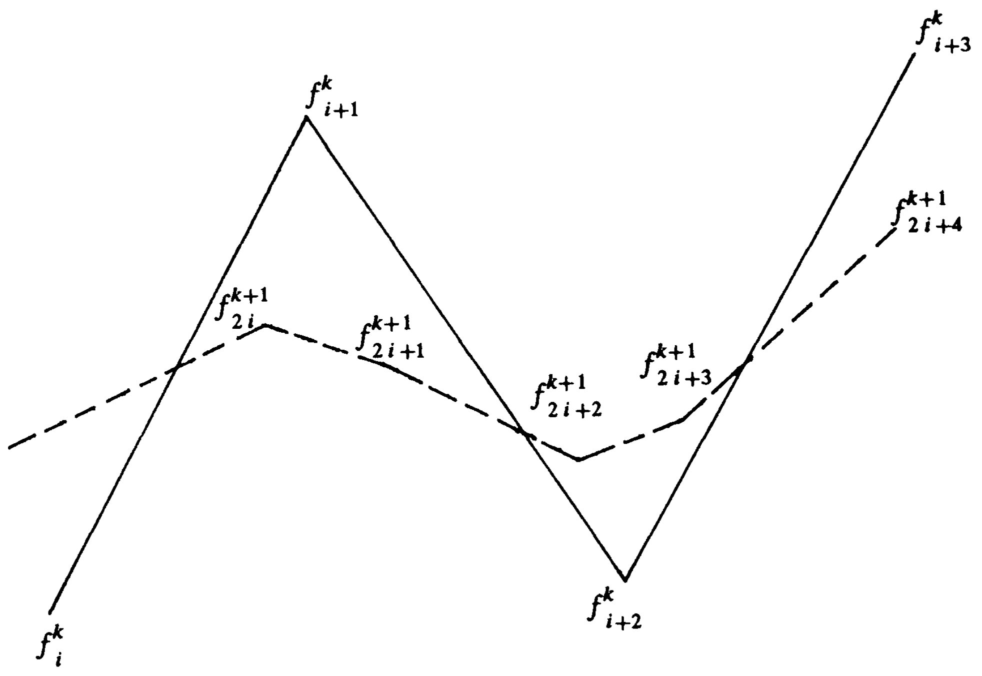


Figure 1.7*b*.

Example 1.8. The De Boor's Algorithm (for B-splines)

Since the *de Boor* algorithm for B-splines [24,37] plays a very important role in the construction and study of recursive subdivision algorithms, this part is devoted to a brief discussion on the *de Boor* algorithm.

Although the *de Boor* algorithm is not a ***RSA*** as we will define it in Chapter 2, many ***RSA*** concerning the generation, computation and approximation of spline curves can be derived from this algorithm. The main feature of the algorithm is the use of the *de Boor-Cox* recurrence relation of the normalized B-spline basis function $B_{i,n}$, the i-th normalized B-spline of order n with knots $x_i, x_{i+1}, \ldots, x_{i+n}$, where $x_i < x_{i+n}$ for all i. More concretely, the recurrence relation of $\{B_{i,n}\}$ is given by:

$$(1.14) \qquad B_{i,n+1} = \frac{x-x_i}{x_{i+n-1}-x_i} B_{i,n}(x) + \frac{x-x_{i+1}}{x_{i+n}-x_{i+1}} B_{i+1,n}(x)$$

where, x is the variable which is omitted in the basis functions.

From (1.14), the *de Boor* algorithm for B-spline curves can be easily formulated. Suppose the spline curve is given by

$$(1.15) \qquad P(x) := \sum_i P_i B_{i,n}(x)$$

then the value of the curve at $x = t$, $x_m \leq t < x_{m+1}$ is given by the *de Boor* algorithm:

$$(1.16)\qquad \begin{aligned} P(t) &= \textstyle\sum_{i=m-n+1}^{m} P_i\, B_{i,n}(t) \\ &= \textstyle\sum_{i=m-n+2}^{m} P_{i,1}(t)\, B_{i,n-1}(t) \\ &= \textstyle\sum_{i=m-n+3}^{m} P_{i,2}(t)\, B_{i,n-2}(t) \\ &= \ldots \\ &= \textstyle\sum_{i=m}^{m} P_{i,n}(t)\, B_{i,1}(t) \\ &= P_{m,n}(t) \end{aligned}$$

where, $P_{i,j}(t)$ is determined by the recursion:

$$(1.17)\qquad P_{i,j}(t) := \frac{t-x_i}{x_{i+n-1}-x_i} P_{i-1,j-1}(t) + \frac{x_{i+n-1}-t}{x_{i+n-1}-x_i} P_{i,j-1}(t)$$

for $j = 1, 2, \ldots, n-1$ and $i = m-n+j+1, \ldots, m$, and $P_{i,0}(t) := P_i$, $i = m-n+1, m-n+2, \ldots, m$.

The algorithm comes from the repeated applications of the B-spline recursion relations (1.14).

The algorithm has the following properties.

(i). Each step is a convex combination combination, so it is a stable algorithm.

(ii). The algorithm can be regarded as a corner cutting process (from the geometric construction).

(iii). The algorithm reflects the local property of the B-splines.

Figure 1.8 shows the geometric construction of the algorithm for a cubic B-spline curve.

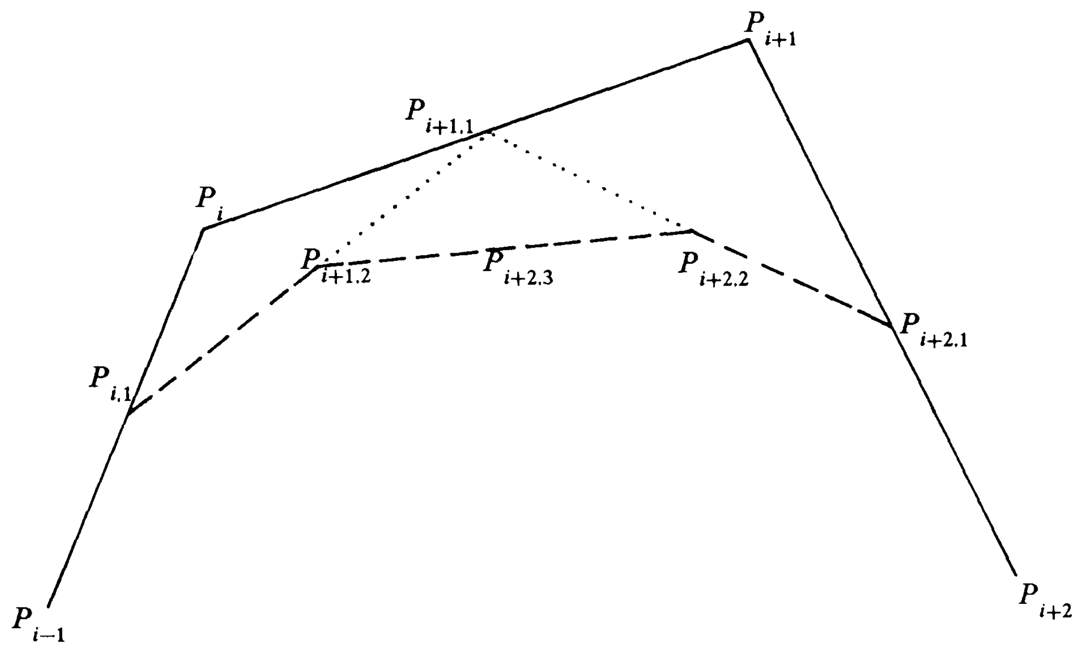


Figure 1.8.

Example 1.9. The Boehm's Knot Insertion Algorithm (for B-splines)

Although the *de Boor* algorithm is the best algorithm for calculating B-splines (*the algorithm dominates the point-evaluation of splines*), *W. Boehm* in 1980 provided yet another powerful alternative to the algorithm for manipulating B-spline curves [14]. The technique was introduced not from the viewpoint of calculating B-splines, but from the viewpoint of spline curve design, construction and subdivision.

The algorithm is even more important than the *de Boor* algorithm since the latter is just a special result of its inferences. More explicitly, the *de Boor* algorithm can be obtained by inserting the same knot repeatedly for a number of times. By inserting some proper new knots and adjusting their corresponding control points, one can obtain the desired curve very conveniently. Another advantage of the knot insertion algorithm is that it combines the calculation (exact value) of B-splines with the approximation (by control polygons) of B-splines by means of a recursive subdivision algorithm. It will be shown in the later Chapters that the recursive subdivision algorithm for non-uniform B-spline curves can be constructed by the knot insertion technique.

The algorithm is straight forward and simple. Since it deals with B-splines, the notations in *Example* 1.8 are used here.

Now, suppose a new knot, y, $x_l < y \leq x_{l+1}$ is be be inserted into the original knot sequence $\{x_i\}$, then the new knot sequence $\{y_i\}$ becomes:

$$(1.18) \qquad y_i := \left| \begin{array}{lll} x_i, & \text{for } i & \leq \; l \\ y, & \text{for } i & = \; l+1 \\ x_{i-1}, & \text{for } i & \geq \; l+2. \end{array} \right.$$

Let $\{M_{i,n}\}$ denote the normalized B-spline basis over the new knot sequence $\{y_i\}$, then the spline curve given by (1.15) can also be expressed in terms of the new basis $\{M_{i,n}\}$:

$$(1.19) \qquad P(x) := \sum_i P_i B_{i,n}(x) = \sum_i Q_i M_{i,n}(x)$$

The new control points $\{Q_i\}$ are given by the *Boehm's* knot insertion algorithm:

$$(1.20) \qquad Q_i := (1 - c_i)\, P_{i-1} + c_i\, P_i$$

where, c_i is given by

$$(1.21) \qquad c_i := \left| \begin{array}{ll} 1 & \text{for } i \le l-n+1 \\ \dfrac{y-y_i}{y_{i+n}-y_i} & \text{for } l-n+2 \le i \le l \\ 0 & \text{for } i \ge l+1. \end{array} \right.$$

Obviously, $0 \le c_i \le 1$.

Figure 1.9 shows the geometric structure of the algorithm for a cubic B-spline curve.

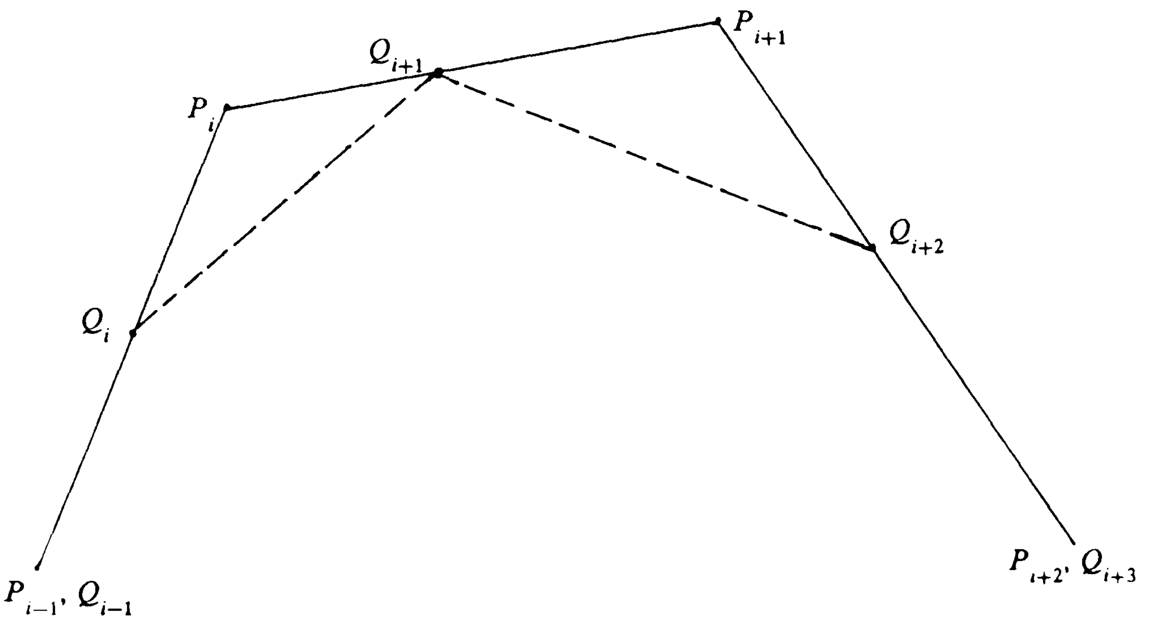


Figure 1.9.

The algorithm has wide applications. It can be used to transfer the control points of a B-spline curve into its corresponding piecewise *Bernstein-Bezier* form by some *corner cutting processes* [56,60]. This problem often arises in curve and surface design and computations. Another application is that it can be used to prove the *VD* (*Variation Diminishing*) property and the shape preserving property of B-spline curves and many other important properties of B-spline curves and surfaces [77].

Example 1.10. The Dyn-Gregory-Levin Scheme (for curves)

The *DGL* scheme is a class of uniform subdivision scheme, which is fully analysed in [50,51,85-90]. The scheme is a generalization of the *Chaikin's* and *Catmull-Clark's* algorithms. It can produce any uniform B-spline curves and some smooth interpolatory curves [48,121]. The main ideology of constructing this scheme is *weighted local averaging* or *weighted moving averaging.*

A special case of the scheme is the 4-point interpolatory subdivision scheme. The scheme is defined by:

$$(1.22)\quad \left| \begin{array}{ll} f_{2i}^{k+1} & = f_i^k \\ f_{2i+1}^{k+1} & = (1/2 + w)(f_i^k + f_{i+1}^k) - w(f_{i-1}^k + f_{i+2}^k) \end{array} \right.$$

where, w is the tension parameter.

It is proved that if $-1/2 < w < 1/2$, the limit curve is continuous. If $0 < w < (\sqrt{5} - 1)/8$, the scheme produces differentiable curves (with respect to the uniform parametrization). In addition, the scheme reproduces all parametric cubic polynomial curves when $w = 1/16$. For more details, the interested reader is referred to [48].

The subdivision process is shown in Figure 1.10.

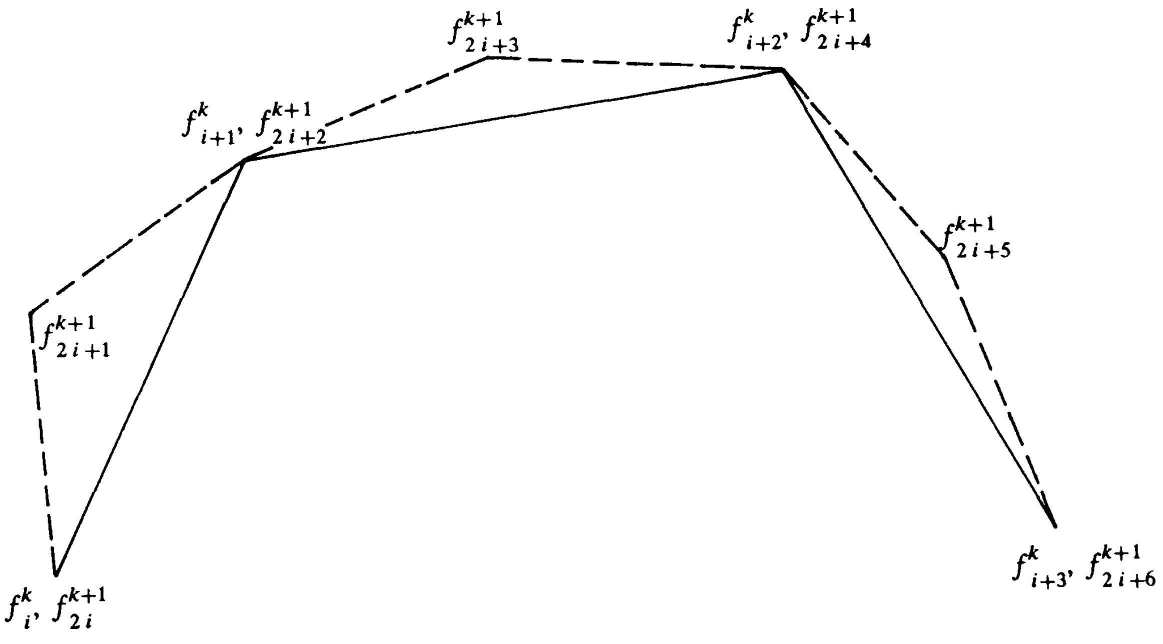

f^{k+1}_{2i+3}
$f^k_{i+2}, f^{k+1}_{2i+4}$
$f^k_{i+1}, f^{k+1}_{2i+2}$
f^{k+1}_{2i+5}
f^{k+1}_{2i+1}
$f^k_{i+3}, f^{k+1}_{2i+6}$
f^k_i, f^{k+1}_{2i}

Figure 1.10.

Example 1.11. The Doo-Sabin Algorithm

Inspired by the *Carpenter's Technique* and the *Chaikin's* algorithm, *Doo* and *Sabin* constructed a recursive subdivision algorithm for smoothing down a general polyhedron [44-46]. In fact, the algorithm is a tensor-product generalization of the *Chaikin's* algorithm with special treatments for the so-called *extraordinary points*. For tensor-product type data, the scheme produces uniform bi-quadratic B-spline surface patches. More about the algorithm will be given in Chapter 4.

Example 1.12. The Catmull-Clark's Algorithm (for Surfaces)

Aş observed in Example 1.4, *Catmull* and *Clark* [27] and *Doo* and *Sabin* [45] used the subdivision algorithm for bi-cubic B-spline patches. However, the main purpose of their papers was to consider generalizations to arbitrary networks. They observed that the algorithm could separate the extraordinary points by tensor-product type data on which smoother surfaces could be defined. Consequently, they modified the algorithm and hoped to get better results. The algorithm will be given in Chapter 4. For more details and mathematical analysis about the algorithm, see [1-3,27,127]

Example 1.13. The 10-Point Interpolatory Subdivision Algorithm

This scheme is an interpolatory ***RSA*** over arbitrary triangulations with three parameters. It is a generalization of the *DGL's 4-point interpolatory scheme* for surfaces [48]. In the uniform case, the three parameters work as three tension parameters along the three mesh directions. When they are all zeros, the refined control polygons are the same as the initial one; when they are chosen appropriately, the scheme reproduces any cubic parametric polynomial surfaces. In general, the scheme is a linear combination of the above two cases. It is proved that it produces smooth surfaces provided that the parameters are chosen to satisfy certain constraints. The scheme will be discussed in detal in Chapter 6.

Example 1.14. The Butterfly Scheme

From the *DGL's* 4-point interpolatory subdivision scheme (1.22), a corresponding scheme for interpolatory surface generation is constructed over arbitrary triangular networks. The *butterfly scheme*, which is firstly introduced in [52], is a special case of the 10-point interpolatory scheme, when the parameters satisfy certain conditions. The scheme will be discussed in detail in Chapter 6. The main advantages of the butterfly scheme are smooth interpolatory, simplicity and shape control. The tension parameter w, which has very clear geometric explanation as in the curve case can be used to manipulate the shape of the surfaces. In the special case of $w = -1/_{16}$, the scheme reproduces cubic parametric surfaces [52,69].

Example 1.15. The Dyn-Gregory-Levin Algorithm (for Surfaces)

As the *DGL* scheme is a very general subdivision scheme for curves, it is natural to derive its corresponding subdivision scheme for surface generation. The scheme has a direct tensor-product generalization for surfaces over uniform data. This will be described in Chapter 6.

Example 1.16. The Uniform Bi-quartic B-Spline Algorithm

The purpose of the construction of this scheme is to develop the *Doo-Sabin's* the *Catmull-Clark's* algorithms in order to generate smooth surfaces over arbitrary networks. The scheme produces uniform bi-quartic B-spline surfaces over uniform data. For arbitrary networks, it is proved that the scheme also produces smooth surfaces if the local shape parameters are chosen properly. The details will be given in Chapter 5.

<u>CHAPTER</u> <u>TWO</u>

SUBDIVISION ALGORITHMS FOR CURVES AND SURFACES-FORMULATIONS AND TECHNIQUES

In this Chapter, some recursive subdivision algorithms for curve and surface generation are introduced and their underlying mathematical techniques are described.

2.1. Introduction

Recursive Subdivision Algorithms (*RSA*) for curves and surfaces have received a good deal of research attention in recent years in the *CAGD* (*Computer Aided Geometric Design*) literature. As a result, many *RSA* have been developed for the generation of curves and surfaces. In some cases, due to different mathematical standpoints, several explanations may exist for the same algorithm. Consequently, different techniques are used to analyse them.

In the papers by *Dyn, Gregory* and *Levin* [50,51], the authors introduced, by using a constructive method, a general form of uniform subdivision algorithms (*DGL* scheme) which was called a *Uniform Recursive Subdivision Algorithm*. In their analysis, they used the *diadic parametrization* technique and the subdivision matrix (*generator matrix*) analysis to study the scheme, whereby the *difference scheme* and the *divided difference scheme* play a very important role. Recently, *Cavaretta, Dyn, Levin* and *Micchelli* used the so-called *generating polynomial* technique to analyse the uniform subdivision algorithms [28,29,54,89-91]. Although the ideas of the technique they used are the same as in the *DGL's* analysis, their notations are very nice which make the whole analysis of this type of algorithms very compact and neat.

Micchelli and *Prautzsch* [85,86,88-91] took also another (different) view on uniform subdivision schemes. They observed that these algorithms are just refinement algorithms and each iteration of any of the algorithms can be viewed as a representation of the limiting curve by a relatively refined basis. So they also introduced a general form of ***RSA*** and studied it together with the limiting curves in a systematic way.

Although uniform subdivision algorithms constitute most of the ***RSA***, there are still some non-uniform ones which are very important for curve and surface generation and also for shape design. Some examples of this class of ***RSA*** are the non-uniform corner cutting algorithms as described in [21,70,81] and some *geometry based* algorithms in [53,71].

Using the *adapted parametrization* technique, we constructed and analysed a smooth non-uniform corner cutting algorithm [70]. We also derived a recursive subdivision scheme for non-uniform B-spline curves with simple knots. The details of the algorithms will be given in Chapter 3.

Another type of non-uniform subdivision algorithms are studied by *N. Dyn, D. Levin* and *D. Liu* in [53] and *M. J. Hejna* in [71]. This class of algorithms consists of some *geometry based* algorithms which are complements of uniform subdivision algorithms. The authors analyse the algorithms together with limit curves by means of some special techniques. These techniques are different from any of the previous methods.

For the purpose of convergence analyses, some mathematical notations are needed. It is assumed that all the ***RSA*** are local algorithms. So, without loss of generality, we suppose that the control polygons and the limit curves are defined on a finite interval $[a, b]$, where, $a < b$. Thus, the uniform norm for curves and control polygons on this interval can be applied so that the norm

$$(2.0) \qquad \|f(t)\| := \max_{a \leq t \leq b} |f(t)|$$

is assumed throughout the curve generating scheme analyses. Furthermore, a ***RSA*** is called a uniform convergent scheme if for any initial control polygon its refined control polygon sequence converges uniformly to a curve which is continuous with respect to a regular parametrization.

2.2. Uniform Subdivision Algorithm Analyses

In this section, we present some methods used to analyse the uniform subdivision schemes. They are *Dyn-Gregory-Levin's* matrix analysis, *Micchelli-Prautzsch's* invariant analysis and the *Cavaretta-Dahmen-Micchelli's* regular subdivision analysis (*Dyn-Levin-Micchelli's* generating polynomial analysis). However, our method, the *Differences* and *Cross-Difference of Directional Divided Differences Analysis,* will be described in detail in Chapter 6.

2.2.1. Dyn-Gregory-Levin's Matrix Analysis

This uniform subdivision algorithm, or the binary subdivision scheme (*BSS*), was firstly introduced and analysed by *Dyn, Gregory* and *Levin* in [50,51]. The scheme is defined as follows.

Let $f_i^k \in \boldsymbol{R}^N$, $i \in \boldsymbol{Z}$, denote a sequence of control points in $\boldsymbol{R}^N$, $N \geq 2$, where k is a non-negative integer. Then the *DGL* scheme, or a *binary subdivision process*, $S(\boldsymbol{a}, \boldsymbol{b})$, is defined by the *mask*

$$(2.1)\quad \left| \begin{aligned} f_{2i}^{k+1} &= \sum_{j=0}^{m} a_i\, f_{i+j}^{k} \\ f_{2i+1}^{k+1} &= \sum_{j=0}^{m} b_i\, f_{i+j}^{k} \end{aligned} \right.$$

where, $\boldsymbol{a}$ and $\boldsymbol{b}$ are the coefficient vectors

(2.1*a*)
$$\left| \begin{array}{l} a := (a_0, a_1, \dots, a_m)^t \\ b := (b_0, b_1, \dots, b_m)^t \end{array} \right.$$

It is assumed that $m > 0$ and

(2.2) $|a_0| + |b_0| > 0$ and $|a_m| + |b_m| > 0$.

In particular, we assume $b_0 \neq 0$ since if $b_0 = 0$ the eqations defining the subdivision process can be interchanged.

Many uniform subdivision algorithms are encompassed by this scheme: the uniform B-spline scheme [51,76], *DGL's* 4-point interpolatory scheme [48], uniform corner cutting algorithms and many other uniform subdivision algorithms [76,78].

To analyse the limit curves, the authors used the *diadic parametrization*, that is, the control point f_i^k is associated with the *diadic parametric* point

(2.3) $t_i^k := i\, 2^{-k}$

for all integer i and k. By doing so, the control polygon $f^k(t)$, which is defined as the piecewise linear curve connecting the points $\{f_i^k\}$, can then be treated as the parametric piecewise linear interpolant satisfying $f^k(t_i^k) = f_i^k$. Hence, $f^k(t)$ is continuous.

Notice that in an interval $[t_i^k, t_{i+1}^k]$ at the k-th stage of the recursion, the limit curve is determined completely by the control point vector

(2.4) $f_{i,k} := [f_i^k, f_{i+1}^k, \dots, f_{i+n_1+1}^k]^t$

and that the control point vectors $f_{2i,k+1}$ and $f_{2i,k+1}$ at the k+1st stage are determined by two linear transformations on $f_{i,k}$, they introduced the square *Generator Matrix* A, of order M, $M = n_1 + 2$, where

$$(2.5) \qquad n_1 = \left| \begin{array}{lll} 2m-1, & for\ a_m & = 0 \\ 2m, & for\ a_m & \neq 0. \end{array} \right.$$

In the case of $a_m \neq 0$, A is defined by

$$(2.6)\quad A := \left| \begin{array}{cccccccc} a_0 & a_1 & a_2 & \dots a_m & 0 & \dots & 0 & 0 \\ b_0 & b_1 & b_2 & \dots b_m & 0 & \dots & 0 & 0 \\ 0 & a_0 & a_1 & \dots a_{m-1} & 0 & \dots & 0 & 0 \\ 0 & b_0 & b_1 & \dots b_{m-1} & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \\ 0 & 0 & 0 & \dots a_0 & a_1 & \dots & a_m & 0 \\ 0 & 0 & 0 & \dots b_0 & b_1 & \dots & b_m & 0 \end{array} \right| .$$

In the case of $a_m = 0$, then, $M = 2m + 1$, the generator matrix A is defined as the matrix (2.6) with the last row and column deleted.

From the generator matrix A, they also introduced the *Left* and *Right* transformation (square) matrices A_0 and A_1 , both of order $M-1$. A_0 is defined as the matrix A with the last row and column deleted, whilst A_1 is also defined as the matrix A with the first row and the last column deleted. Thus, from scheme (2.1), we can obtain

$$(2.7) \qquad f_{2i,k+1} = A_0 f_{i,k} \quad \text{and} \quad f_{2i+1,k+1} = A_1 f_{i,k}$$

Therefore, for any diadic point p defined by

(2.8) $p = \sum_{j=0}^{k} p_j 2^{-j}$

the history of the recursion is given by

(2.9) $f_{i,k} = A_{p_k} \dots A_{p_1} f_{p_0,0}.$

Furthermore, by studying the *differences* and the *divided differences*, they established the following analysis for the scheme together with its limit curves.

Let e_i^k denote the difference $f_{i+1}^k - f_i^k$, then, it can be shown, provided the necessary condition (2.12) below holds, that the differences satisfy the *Difference Scheme* (we assume that the scheme produces continuous curves):

(2.10)
$$\left|\begin{array}{ll} e_{2i}^{k+1} & = \sum_{j=0}^{m-1} c_i e_{i+j}^k \\ & \\ e_{2i+1}^{k+1} & = \sum_{j=0}^{m} d_i e_{i+j}^k \end{array}\right.$$

where, c_i and d_i are determined by $\{a_i\}$ and $\{b_i\}$. More explicitly,

(2.11) $c_i = \sum_{j=0}^{i} (a_j - b_j)$ and $d_i = a_i - c_i.$

Thus, the *divided differences* can be defined and its corresponding *divided difference scheme* can also be derived and studied. In much the same way, *higher order difference schemes* and *divided difference schemes* can also be defined, which leads to the higher order continuity analyses of the limit curves.

From the above discussion, it can be concluded that the limit curve of the scheme is characterized by the generator matrix A (A_0 and A_1). By analysing A, they obtained the properties of the scheme and its limit curves.

The following are the main results of their analyses. For more details, the interested reader is referred to [50]. We should also mention that throughout their analysis, the condition (2.12) is assumed.

<u>**Proposition**</u> 2.1. A necessary condition for the scheme (2.1) on the diadic points to converge to a continuous (non-degenerate) limit curve on some interval is:

(2.12) $\sum_{i=0}^{m} a_i = \sum_{i=0}^{m} b_i = 1.$

<u>**Proposition**</u> 2.2. The scheme (2.1) converges uniformly to a continuous curve (with respect to the *diadic parametrization*) if its difference scheme is a contraction scheme, that is,

(2.13) $Lim_{k\to\infty}\, max_i \{|e_i^k|\} = 0.$

<u>**Proposition**</u> 2.3. Scheme (2.1) converges uniformly to a differentiable curve $f(t)$ if its *divided difference polygon* converges uniformly to $d(t)$. Moreover $f'(t) = d(t)$.

Proposition 2.4 (*Higher order continuity*). The scheme (2.1) produces a C^n curve if its n-th order *Divided Difference scheme* is a C^0 scheme.

2.2.2. Cavaretta-Dahmen-Micchelli's Regular Subdivision Analysis

The second technique used to analyse the scheme (2.1) is the *Generation Polynomial Method,* introduced by *Cavaretta, Dahmen* and *Micchelli* [28,29]. The major advantage of this method is that it unifies the theories of all the uniform subdivision schemes (for both curves and surfaces) which can be written in a similar form as (2.1). This approach makes the *difference* and *divided difference analysis* much neater and simpler. The formulation of the method is described as follows.

Instead of using (2.1) to describe the scheme, they use a *Laurent Polynomial* $a(z)$

$$(2.14) \qquad a(z) := \textstyle\sum_i \; a_i \, z^i$$

to characterize the scheme. Thus, the scheme can be defined as a operator S_a in the following manner

$$(2.15) \qquad y_i := (S_a x)_i := \textstyle\sum_j \; a_{i-2j} \, x_j.$$

Explicitly, the method is just a change of notations when compared with (2.1) since (2.15) is equivalent to

$$(2.16)\quad \left| \begin{array}{l} y_{2i} = \dots + a_2 x_{i-1} + a_0 x_i + a_{-2} x_{i+1} + \dots \\ y_{2i+1} = \dots + a_3 x_{i-1} + a_1 x_i + a_{-1} x_{i+1} + \dots \end{array} \right.$$

Therefore, for any scheme of the form (2.1), there is a unique *Laurent Polynomial* $a(z)$ and vice versa. Further studies show that the properties of the scheme are fully characterized by the polynomial $a(z)$ [28].

The following are some results described in [28,29].

<u>**Proposition**</u> 2.5. Condition (2.12) is equivalent to

$$(2.17)\quad a(1) = 2 \text{ and } a(-1) = 0.$$

<u>**Proposition**</u> 2.6. If there exists some positive integers k, n and a *Laurent* polynomial $q(z)$ such that

$$(2.18)\quad a(z) = 2^{-k}(1 + z^{-1})^{k+1} q(z) \text{ and } q(1) = 1$$

and

$$(2.19)\quad \| S_q^n \| := max \{ \| S_q^n x \| : \|x\| \leq 1 \} < 1,$$

then the scheme is a C^k scheme.

<u>**Proposition**</u> 2.7 (*perturbation technique*). Let $a(z)$ satisfy (2.18), (2.19) and

(2.20) $a_w(z) = a(z) + 2^{-k}(1+z^{-1})^{k+1} wb(z),$

where, $b(z)$ is some *Laurent polynomial*. Then there exists a $w_0 > 0$ such that the subdivision scheme corresponding to $a_w(z)$ is a C^k scheme for $|w| < w_0$.

Proposition 2.8. Suppose

(2.21) $a_w(z) = 2^{-k}(1+z^{-1})^{k+1}(1 + wb(z)),$

where, $b(1) = 0$ and $b_0 \neq 0$. Then there exists a $w_0 > 0$ such that for $0 < -w\,sgn(b_0) < w_0$, the scheme defined by $a_w(z)$ is a C^k scheme.

Remark: The most important thing here is that the method can be applied to analyse uniform subdivision schemes for surfaces. The only difference is that the above notations should be replaced by *multiple notations*.

2.2.3. Micchelli-Prautzsch's Invariant Analysis

As early as in the forties, *de Rahm* studied a uniform subdivision scheme which he called a *trisection algorithm* [101,102]. His original motivation was to find some functions which were solutions of certain functional equations. The *Micchelli-Prautzsch* analysis of subdivision schemes is just a generalization of *de Rahm's* method. However, the results they achieved are much more than that of *de Rahm's*.

By studying the *de Casteljau's* and *Chaikin's* algorithms from a different point of view, *Micchelli* and *Prautzsch* introduced a general technique to analyse the uniform subdivision scheme [85,86,88-91]. Their motivation was to find all the curves which could be uniformly subdivided. Since *"Linear Subdivision is a Strictly Polynomial Phenomenon"* [61], the curves they sought could be called *Generalized Polynomial Curves.*

Their main observation is the following. Suppose a curve $P(t)$ is represented by some function basis $\{B_i(t)\}$, $i = 0, 1, ..., n$ and some corresponding *control points*, $\{P_i\}$, in a form

(2.22) $\quad P(t) := \sum_{i=0}^{n} P_i B_i(t) \quad$ for $0 \leq t \leq 1$

and that the curve has the properties that it can be split into two parts. Each of them can also be expressed in the same form as (2.22) but with a *refined basis*. Furthermore, it is assumed that the new control points can be obtained from the old ones by some local weighted averaging process. More explicitly, the curve has the properties that there exists two matrices A_0 and A_1 such that

(2.23) $\quad \boldsymbol{P}^R = A_0 \boldsymbol{P}$ and $\boldsymbol{P}^L = A_1 \boldsymbol{P}$,

where, $\boldsymbol{P}^R := (P^R_0, P^R_1, ..., P^R_n)^t$, $\boldsymbol{P}^L := (P^L_0, P^L_1, ..., P^L_n)^t$, $\boldsymbol{P} := (P_0, P_1, ..., P_n)^t$, and

$$(2.24)\quad P(t) = \begin{cases} \sum_{i=0}^{n} P_i^R\, B_i(2t) & \text{for} \quad 0 \le t \le 1/2 \\ \sum_{i=0}^{n} P_i^L\, B_i(2t-1) & \text{for} \quad 1/2 \le t \le 1. \end{cases}$$

By repeating this process, one can obtain

$$(2.25)\qquad \boldsymbol{Lim}_{k\to\infty} \boldsymbol{A}_{x_k} \dots \boldsymbol{A}_{x_1}\boldsymbol{P} = P(x)\, \boldsymbol{E},$$

where, $x := \sum_{i=1}^{\infty} x_i\, 2^{-i}$ and $\boldsymbol{E} := (1, 1, \dots, 1)^t$.

From the above discussion, it can be shown that the curve $P(t)$ is characterized by the functional equation

$$(2.26)\qquad Y(x,P) = \boldsymbol{Lim}_{k\to\infty}\ \boldsymbol{v}^t \boldsymbol{A}_{x_k} \dots \boldsymbol{A}_{x_1}\boldsymbol{P}, \quad \text{for} \quad 0 \le x \le 1,$$

where, $\{x_i\}$ are the binary expansion coefficients of x, and $\boldsymbol{v}$ is any vector of dimension n whose components sum to unity.

By studying the refinement equations (2.23), (2.26) and the subdivision matrices $\boldsymbol{A}_0$ and $\boldsymbol{A}_1$, the following results were obtained [85,86,88-91].

<u>Proposition</u> 2.9. Given two matrices $\boldsymbol{A}_0$ and $\boldsymbol{A}_1$, then the compatibility condition for the matrices to define a continuous curve in the form (2.22) is

$$(2.27)\qquad \boldsymbol{v}_0^t \boldsymbol{A}_1 = \boldsymbol{v}_1^t \boldsymbol{A}_0 ,$$

where, v_0^t and v_1^t are the eigenvectors of A_0 and A_1 corresponding to eigenvalue one. Furthermore, the matrices must be *regular matrices*.

Proposition 2.10. Any polynomial curve can be produced by properly choosing subdivision matrices A_0 and A_1.

Other properties of these curves such as continuity, differentiability and polynomial curve generations are fully studied in [85-91].

2.2.4. Cube-spline Algorithm Analysis

In order to develop the univariate B-spline theory to cope with multivariate problems, the *Cube-spline* (*Box-spline*) has been intensively studied [11-13,16,18,20,28,40,85-91]. The core of the Cube-splines is its line-averaging algorithm which is a generalization of the *Lane-Riesenfeld's* algorithm for multidimensional networks.

The Cube-splines are defined as follows. Let $X := \{x_1, x_2, ..., x_n\} \in Z^s \setminus \{0\}$ be a set of not necessarily distinct vectors, where $n \geq s$. For $i = 1, 2, ..., s$, we assume that $x_i = \varepsilon_i$, the i-th coordinate vector, and so $<X> := span\{X\} = R^s$. Let $X_k := \{x_1, x_2, ..., x_k\}$ for $k = s, s+1, ..., n$, so that $X_n = X$. Then, for $x \in R^s$, the Cube-spline is defined inductively by

$$(2.28) \quad M(x|X_s) = \begin{cases} 1, & x \in [0,1]^s, \text{ the unit cube in } R^s \\ 0, & \text{otherwise;} \end{cases}$$

and

(2.29) $M(x|X_k) = \int_0^1 M(x - tx_k|X_{k-1})\,d_t$ for $k = s+1, \ldots, n$.

From this definition, one can derive the distributional property of $M(x|X)$:

(2.30) $\int_{R^s} f(x)M(x|X)\,dx = \int_0^1 \ldots \int_0^1 f(<T,x>)\,d_T$

where, $T := (t_1, t_2, \ldots, t_n)^t$ and $f(t)$ is any continuous function. The notation $<T, x>$ is the scalar product of T and x. This can be proved inductively (induction on k).

From (2.30), we can obtain the *Fourier Transform* of the Cube-spline $M(x|X)$

(2.31) $\hat{M}(y|X) := \int_{R^s} e^{i<x,y>} M(x|X)\,dx = \prod_{j=1}^{n} (e^{i<x_j,y>} - 1)/(i<x_j, y>)$.

A straight forward calculation from this shows that

(2.32) $2^s \dfrac{\hat{M}(2y|X)}{\hat{M}(y|X)} = \prod_{j=1}^{n} (z^{x_j} + 1) = \sum_{\alpha \in Z^s} b_\alpha e^{i<y,\alpha>}$

where, $z := (z_1, z_2, \ldots, z_s)$ and $z := e^{iy}$, $j = 1, 2, \ldots, s$. It is assumed that z^{x_j} is defined by the multiple exponential role. The coefficient sequence $\{b_\alpha\}$, where $b_\alpha := b_\alpha(X)$, is called the *mask* of the subdivision algorithm. From (2.32), we have

(2.33) $$2^s \hat{M}(2y|X) = \sum_{\alpha \in Z^s} b_\alpha e^{i\langle y,\alpha\rangle} \hat{M}(y|X),$$

from which it follows that

(2.34) $$M(x/2|X) = \sum_{\alpha \in Z^s} b_\alpha M(x-\alpha|X).$$

From this refinement relation, we now can formulate the subdivision algorithm for Cube-spline curves and surfaces. Given a Cube-spline function (curve or surface):

(2.35) $$f(x) = \sum_{\alpha \in Z^s} c_\alpha M(x-\alpha|X),$$

from (2.34), we can represent the surface as

(2.36) $$f(x) = \sum_{\alpha \in Z^s} c_\alpha \sum_{\beta \in Z^s} b_\beta M(2x-2\alpha-\beta|X)$$

$$= \sum_{\beta \in Z^s} c^1_\beta M(2x-\beta|X),$$

where

(2.37) $$c^1_\beta = \sum_{\beta \in Z^s} c_\alpha b_{\beta-2\alpha}.$$

This equation defines a single iteration of the subdivision algorithm. It can also be shown that the subdivision algorithm is composed of certain line averages. Multiplying the equation

(2.38) $$\sum_{\alpha \in Z^s} b_\alpha(X_{k-1}) z^\alpha = 2^{-n+s+1} \prod_{j=1}^{k-1} (z^{x_j} + 1)$$

by $(z^{x_k} + 1)/2$, we obtain

(2.39) $$\sum_{\alpha \in Z^s} (b_\alpha(X_{k-1}) + b_{\alpha - x_k}(X_{k-1})\, z^\alpha = 2 \sum_{\alpha \in Z^s} b_\alpha(X)\, z^\alpha.$$

Hence we have

(2.40) $$b_\alpha(X_k) = [\, b_\alpha(X_{k-1}) + b_{\alpha - x_k}(X_{k-1})\,]/2.$$

Using this relation in (2.37), we obtain the recursion

(2.41) $$c^1_\beta(X_k) = [\, c^1_\beta(X_{k-1}) + c^1_{\beta - x_k}(X_{k-1})\,]/2.$$

To find the initial values of the recursion, we note in the special case $X_s = \{\epsilon_1, \epsilon_2, \ldots, \epsilon_s\}$ that

(2.42) $$\sum_{\alpha \in Z^s} b_\alpha(X_s)\, z^\alpha = \prod_{j=1}^{s} (z^{\epsilon_j} + 1).$$

Thus

(2.43) $$b_\alpha(X_s) = \left| \begin{array}{l} 1, \text{ for } \alpha \in \text{extremes } [\,0,1]^s, \\ 0, \text{ otherwise.} \end{array} \right.$$

Using (2.37) in this case, we find that

(2.44) $$c^1_\beta(X_s) = c_\alpha \quad \text{for } \beta = 2\alpha + \text{extreme}[\,0,1]^s.$$

This relation is called the expansion step (very similar to the *Lane-Riesenfeld* algorithm for univariate B-splines). The succeeding steps are the line averages given by (2.41).

The convergence and smoothness analysis of the algorithm comes from the refinement property (2.34) and studies about the Cube-splines. It is proved that the algorithm converges to a piecewise polynomial surface at the rate $O(2^{-k})$ (exponential convergence). The interested reader is referred to [28,35,38,48,58].

2.3. Analyses for Non-uniform Subdivision Algorithms

In this section we present some of the techniques used to analyse non-uniform subdivision schemes. Since non-uniform schemes are much more complicated than the uniform ones, successful analyses of them depend mainly on the special structures of the scheme. In fact, only a few non-uniform subdivision schemes have been studied so far.

2.3.1. Adapted Parametrization Technique

In order to complement *Chaikin's* algorithm and to investigate conditions on algorithms for the generation of smooth curves, especially non-uniform ***RSA*** for B-spline curves, some non-uniform corner cutting algorithms are introduced and analysed [21,70,71,80]. Unlike the diadic parametrization as used in [50,51,85-91], a natural *adapted parametrization* is used instead to seek better results. This turns out to be a successful choice. Moreover, this adapted parametrization approach could also be used to construct and to analyse some other non-uniform subdivision algorithms. An immediate example is the subdivision scheme for non-uniform B-spline curves with simple knots, which is studied in detail in Chapter 3.

The essence of this technique relies on the *difference* and *divided difference* analysis of the scheme (with respect to the *adapted parametrization*) as described in [50,51]. The only difference, however, is the parametrization of the processes and the limit curves. It is also hoped that better results can be obtained if an appropriate parametrization is used.

A more detailed description of this technique will be given in the next Chapter. However, in the remainder of this subsection, we shall describe briefly the non-uniform corner cutting scheme and some underlying ideas about the analysis.

The non-uniform corner cutting scheme is given by

$$(2.45) \quad \left| \begin{array}{ll} f^{k+1}_{2i} & = (1-a^k_i)\, f^k_i \;+\; a^k_i\, f^k_{i+1} \\ f^{k+1}_{2i+1} & = \;b^k_i\, f^k_i \;+ (1-b^k_i)\, f^k_{i+1}. \end{array} \right.$$

It is assumed that the initial control points $\{f^0_i\}$ and the parameter values $\{t^0_i\}$ are also given. By using (2.45), the control points $\{f^k_i\}$ can be determined recursively. The parameter values $\{t^k_i\}$ associated with $\{f^k_i\}$ are then defined such that $\{t_i^k\}$ satisfy the subdivision scheme (2.45). More explicitly, the parameter is taken as a component of the control points. Thus, the parameter t_i^k satisfies the same recursion:

$$(2.46)\quad \left| \begin{aligned} t_{2i}^{k+1} &= (1-a_i^k)\, t_i^k + a_i^k\, t_{i+1}^k \\ t_{2i+1}^{k+1} &= b_i^k\, t_i^k + (1-b_i^k)\, t_{i+1}^k . \end{aligned} \right.$$

We shall call this type of parametrization of the control polygons the *Adapted Parametrization.*

Now we can define the *divided difference scheme* corresponding to the scheme (2.45). The *divided differences* $\{d_i^k\}$ are defined as

$$(2.47)\quad d_i^k := \Delta f_i^k / \Delta t_i^k = (f_{i+1}^k - f_i^k)/(t_{i+1}^k - t_i^k).$$

After some simple calculations, it can be shown that $\{d_i^k\}$ satisfy the recursion

$$(2.48)\quad \left| \begin{aligned} d_{2i}^{k+1} &= d_i^k \\ d_{2i+1}^{k+1} &= (1-c_i^k)\, d_i^k + c_i^k\, d_{i+1}^k \end{aligned} \right.$$

where

$$(2.49)\quad c_i^k := \frac{a_{i+1}^k(t_{i+2}^k - t_{i+1}^k)}{b_i^k(t_{i+1}^k - t_i^k) + a_{i+1}^k(t_{i+2}^k - t_{i+1}^k)}.$$

The same parameter t is used to parametrize the control polygons, $d^k(t)$, of the divided differences $\{d_i^k\}$, that is

(2.50) $d^k(t_i^k) = d_i^k, \; i \in \mathbf{Z}, \; k \geq 0.$

From the above formulation, the following result is obtained in [70,81].

Proposition 2.11. If the *divided difference* sequence $\{d^k(t)\}$ converges to a continuous curve $d(t)$, then scheme (2.45) produces a continuously differentiable curve $f(t)$ and $f'(t) = d(t)$.

The scheme can also be generalized to produce any (non-uniform) B-spline curve with simple knots. This will be studied in detail in Chapter 3.

2.3.2. Geometry-Based Algorithm Analyses

Besides the above discussed uniform and non-uniform ***RSA*** for curve and surface generation, there are some other subdivision algorithms which can not be covered by the previous analyses. In [53], *N. Dyn, D. Levin* and *D. Liu* presented a geometrically constructed interpolatory non-linear subdivision algorithm for curve and surface generation. Another one is a complement of *Chaikin's* algorithm given by *M. J. Hejna* [71] in his dissertation in which he used a geometry based algorithm to generate smooth curves.

The main characteristics of this type of algorithms is that they depend on the local geometry, that is, the algorithms depend on the *tangent lines* and *tangent planes* and occasionally even the local convexity property of the control polygons. The analyses of such schemes are then based on these special geometrical properties of the algorithms. In some cases, monotonicity or convexity preserving feature of the schemes dominates their convergence and smoothness analyses. For example, the algorithm described in [53] is a typical local geometry based, non-uniform, non-linear interpolatory subdivision algorithm. However, we shall not discuss these schemes and their analyses here. For more details, the interested reader is referred to the papers [53,71].

CHAPTER THREE

A NON-UNIFORM CORNER CUTTING SCHEME AND THE SUBDIVISION ALGORITHM FOR B-SPLINE CURVES

The non-uniform corner cutting scheme, which is a generalization of *Chaikin's* and *Catmull-Clark's* algorithms to generate smooth curves, is discussed in detail in this Chapter. Furthermore, by using the *Adapted Parametrization* technique, a recursive subdivision algorithm for non-uniform B-spline curves of order k with simple knots is formulated which provides an alternative to the *de Boor* algorithm and the *Oslo* algorithm for the computation of B-splines.

3.1. A Non-uniform Corner Cutting Algorithm

The convergence of a non-uniform corner cutting process is studied in this section. Using a parametrization different from the *diadic parametrization* employed by *Dyn-Gregory-Levin* [50,51] and *Micchelli* and *Prautzsch* [83-91], it is shown that the process is smooth. That is, it produces differentiable curves provided the proportions of the corner cuts are kept within appropriate constraints. This work is to appear in the *CAGD Journal, Gregory-Qu* [70].

3.1.1. Background

The motivation of the investigation of the non-uniform corner cutting algorithm originally came from *de Boor's* paper [21]. This paper showed that *cutting corners* of a control polygon *always works*, in a sense that the limit curves are *Lipschitz* continuous. Although *Lipschiz* continuous curves are the best possible that one can achieve from *de Boor's* assumptions, from the application point of view they are not good enough for the design of smooth curves.

To develop corner cutting techniques for simple, smooth curve generations, we proposed the investigation of a simple non-uniform corner cutting process. Before we analyse the scheme mathematically, it should be noted that the process could produce any smooth quadratic B-spline curve by an appropriate choice of the parameters. This suggested that a more general choice of the corner cutting parameters might also produce C^1 curves. The aim then was to look for some *natural parametrization* which might be more appropriate than the uniform diadic parametrization as used in [50,51,83-91] whereby it is used to prove the smoothness properties of the limit curves.

In the following subsections, we develop the analysis of the non-uniform corner cutting process. And, as a consequence, a recursive subdivision algorithm for B-spline curves with simple knots will be derived in section 3.2.

3.1.2. The Non-uniform Corner Cutting Scheme

The scheme is defined as follows. Suppose $f_i^0 \in \mathbf{R}^2, i = 0, 1, ..., n+1$ are the initial control points $(n > 1)$, which are associated with the parameter values $t_0^0 < t_1^0 < t_2^0 < ... < t_{n+1}^0$. (In fact, the results are true for curves in any *Euclidean* space, although only the planar case is discussed here). The scheme is defined by the following *mask*: for $k = 0, 1, ...; i = 0, 1, ..., 2^k n$

$$(3.1) \quad \left| \begin{array}{ll} f_{2i}^{k+1} & = (1-a_i^k)\, f_i^k + a_i^k\, f_{i+1}^k \\ f_{2i+1}^{k+1} & = b_i^k\, f_i^k \quad + (1-b_i^k)\, f_{i+1}^k, \end{array} \right.$$

where, it is assumed that

$$(3.2) \quad a_i^k,\ b_i^k > 0, \text{ and } 1 - a_i^k - b_i^k > 0.$$

It is also assumed that the point sequence $\{f_i^k\}$, which represents the control polygon at level k, is associated with the parameter values $\{t_i^k\}$. The use of *adapted parametrization* means that parameter values satisfy the same corner cutting process

$$(3.3)\quad \left|\begin{array}{ll} t_{2i}^{k+1} & = (1-a_i^k)\, t_i^k + a_i^k\, t_{i+1}^k \\ t_{2i+1}^{k+1} & = b_i^k\, t_i^k \ + (1-b_i^k)\, t_{i+1}^k . \end{array}\right.$$

The condition (3.2) guarantees that the parametric points always form a strictly monotonic increasing set $t_0^k < t_1^k < t_2^k < \dots < t_{2^k n+1}^k$ since

$$(3.4)\quad t_i^k < t_{2i}^{k+1} < t_{2i+1}^{k+1} < t_{i+1}^k \quad \text{for all } i, k.$$

Denote by f^k the control polygon with vertex f_i^k, $i = 0, 1, \dots, 2^k n+1$. Then (3.1) is a process whereby f^{k+1} is created by corner cutting of the polygon f^k. In general, this process is non-uniform since the proportions a_i^k and b_i^k of the corner cuts depend both on i and k.

From the above discussion, the control polygon f^k can be identified unambiguously as the piecewise linear interpolant $f^k(t)$, where, for $t \in [t_i^k, t_{i+1}^k]$, $i = 0, 1, \dots, 2^k n$,

$$(3.5)\quad f^k(t) = \frac{t - t_i^k}{t_{i+1}^k - t_i^k} f_i^k + \frac{t_{i+1}^k - t}{t_{i+1}^k - t_i^k} f_{i+1}^k .$$

Since the corner cutting process is a geometric invariant process and the parametrization is regular (see subsection 3.1.5), it suffices from now on to consider the scalar case, that is, $\{f_i^k\}$ are scalars, see Figure 3.1.

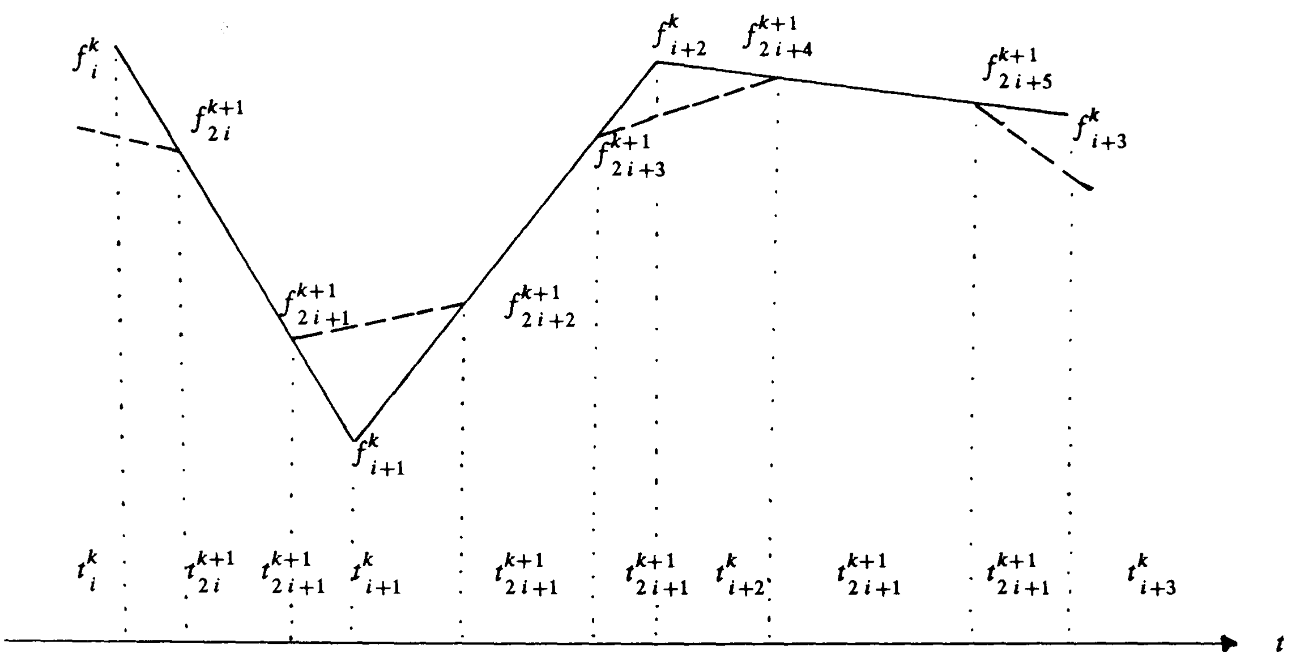


Figure 3.1.

It follows from (3.4) that $\{t_0^k\}$ and $\{t_{2^k n}^k\}$ form monotonic increasing and decreasing sequences bounded above and below by t_1^0 and t_n^0 respectively. Hence there exists

(3.6) $a := \mathit{Lim}_{k\to\infty} t_0^k \leq t_1^0$ and $b := \mathit{Lim}_{k\to\infty} t_{2^k n}^k \geq t_n^0$.

Then the uniform norm

(3.7) $\|f\| := \max_{a\leq t\leq b} |f(t)|$

is used on the interval $[a, b]$ throughout this Chapter.

3.1.3. Cutting Corner is C^0

Although our main purpose is to find conditions under which the corner cutting process has a smooth limit, the C^0 analysis given here lays the foundation of the smoothness analysis. The result is a very special case of *de Boor's* results in [21]. Here we present a different proof.

Firstly, we show that the sequence $\{f^k\}$ defines a *Cauchy* sequence in $C[a,b]$. In order to do this, the following lemma is required.

Lemma 3.1. For all $k, p \geq 0$,

(3.8) $\quad \|f^{k+p} - f^k\| \leq 2\, \max_i |\Delta f_i^k|,$

where, Δ is the *Forward Difference Operator:*

(3.9) $\quad \Delta f_i^k := f_{i+1}^k - f_i^k.$

Proof. Consider f^{k+p} and f^k on $[\, t_{2^p i}^{k+p}, t_{2^p(i+1)}^{k+p}]$. From (3.4) we have

(3.10) $\quad t_i^k < t_{2^p i}^{k+p} < t_{2^p(i+1)}^{k+p} < t_{i+2}^k$

and since the corner cutting process is a convex combination, we obtain

(3.11) $\quad m_i \leq f_j^{k+p} \leq M_i$ for all $2^p i \leq j \leq 2^p(i+1)$

where,

(3.12) $\quad m_i = \boldsymbol{min}_i \{ f_i^k, f_{i+1}^k, f_{i+2}^k \}$ and $M_i = \boldsymbol{max}_i \{ f_i^k, f_{i+1}^k, f_{i+2}^k \}$.

Hence $m_i \leq f^{k+p}(t), f^k(t) \leq M_i$,

which gives

(3.13) $\quad |f^{k+p}(t) - f^k(t)| \leq M_i - m_i \leq |\Delta f_i^k| + |\Delta f_{i+1}^k|$,

so the lemma follows.

This *Lemma* suggests an analysis of the *difference process* which is obtained from (3.1) as

$$(3.14)\quad \left| \begin{array}{ll} \Delta f_{2i}^{k+1} & = (1 - a_i^k - b_i^k)\, \Delta f_i^k \\ \Delta f_{2i+1}^{k+1} & = b_i^k\, \Delta f_i^k + a_{i+1}^k\, \Delta f_{i+1}^k. \end{array} \right.$$

Let

$$(3.15)\quad \left| \begin{array}{l} \overline{a} = \boldsymbol{Lim}_{k\to\infty} \boldsymbol{max}_i\, a_i^k, \quad \underline{a} = \boldsymbol{Lim}_{k\to\infty} \boldsymbol{min}_i\, a_i^k, \\ \overline{b} = \boldsymbol{Lim}_{k\to\infty} \boldsymbol{max}_i\, b_i^k, \quad \underline{b} = \boldsymbol{Lim}_{k\to\infty} \boldsymbol{min}_i\, b_i^k. \end{array} \right.$$

Then we have:

Theorem 3.2. The function sequence $\{f^k(t)\}$ converges uniformly to a continuous function $f(t)$ in $C[a, b]$ if

(3.16) $\underline{a}, \underline{b} > 0$ and $1 - \overline{a} - \overline{b} > 0$.

Proof. From the *difference process* (3.14), it follows that

(3.17) $max_i \, |\Delta f_i^{k+1}| \leq B^k \, max_i \, |\Delta f_i^k|$,

where

(3.18) $B^k := max_i \, \{1 - a_i^k - b_i^k, \; b_i^k + a_{i+1}^k\}$.

Moreover, it can be shown easily that

(3.19) $B^k \leq B < 1$

for some constant B, independent of k, if (3.16) holds. Hence the differences are contracting and from *Lemma* 3.1 it follows that $\{f^k(t)\}$ defines a *Cauchy* sequence in $C[a, b]$. That completes the proof.

Conditions (3.16) are sufficient for the generation of continuous curves by the scheme. They require that $(\underline{a}, \underline{b})$ and $(\overline{a}, \overline{b})$ lie strictly within the region Ω_0 depicted in Figure 3.2. In the following subsection, the conditions under which the scheme produces smooth curves will be discussed.

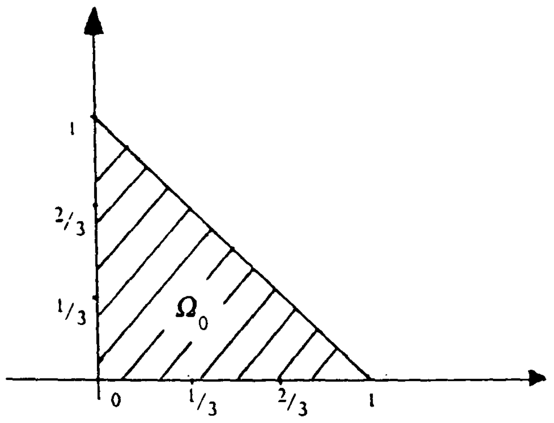


Figure 3.2.

3.1.4. Smooth Corner Cutting

The *divided differences* play an important role for the smoothness analysis of the scheme. We firstly give a result about the parametric points under the conditions derived in subsection 3.1.3.

<u>Lemma</u> 3.3. The parametric points $\{t_i^k\}$ becomes dense in $[a, b]$ when k goes to infinity.

<u>Proof</u>. Because the parametric points satisfy the same corner cutting process, it follows from the same arguments as in the proof of *Theorem* 3.2 that

$$(3.20) \qquad max_i\, |\Delta t_i^k| \;\leq\; B\; max_i\, |\Delta t_i^k|.$$

So we have

(3.21) $\quad \mathit{Lim}_{k\to\infty}\, \mathit{max}_i\, |\Delta t_i^k| = 0$

and this completes the proof.

Now we define the *divided difference process* of the scheme. The *divided difference* d_i^k is defined by

(3.22) $\quad d_i^k := \Delta f_i^k / \Delta t_i^k.$

From (3.3) and (3.14), it can be shown that the *divided differences* satisfy the following recursion (*the divided difference scheme*, or *DD scheme* for short)

(3.23) $$\left|\begin{array}{lll} d_{2i}^{k+1} & = d_i^k, & \text{at } t_i^k \\ d_{2i+1}^{k+1} & = (1-c_i^k)\, d_i^k + c_i^k\, d_{i+1}^k, & \text{at } t_{i+1}^k \end{array}\right.$$

where,

(3.24) $\quad c_i^k := a_{i+1}^k\, \Delta t_{i+1}^k / (b_i^k\, \Delta t_i^k + a_{i+1}^k\, \Delta t_{i+1}^k).$

(here the same parametrization is used).

The importance of analysing the *divided difference scheme* is given by the following theorem:

Theorem 3.4. If the *DDS* produces a continuous function $d(t)$ in $C[a, b]$ with respect to the parametric points $\{t_i^k\}$, then the corner cutting scheme produces a differentiable function $f(t)$. Moreover, $f'(t) = d(t)$.

Proof. Let $H_k(t)$ denote the piecewise cubic *Hermite* interpolant such that

(3.25) $$H_k(t_i^k) = f_i^k \quad \text{and} \quad H_k'(t_i^k) = d_i^k \quad \text{for all } i \text{ and } k.$$

Then for $t \in [t_i^k, t_{i+1}^k]$, with $x := (t-t_i^k)/\Delta t_i^k$, $x \in [0, 1]$, we can find explicitly:

(3.26) $$H_k(t) = (1-x)^2 (2x+1) f_i^k + x^2 (-2x+3) f_{i+1}^k$$

$$+ x(1-x)^2 \Delta t_i^k d_i^k + x^2(x-1)\Delta t_i^k d_{i+1}^k,$$

(3.27) $$H_k'(t) = (-3x^2 +2x+1) d_i^k + (3x^2 -2x) d_{i+1}^k \,.$$

Let $d^k(t)$ be the control polygon of the divided differences, that is, it is the piecewise linear interpolant to data (t_i^k, d_i^k). Then for $t \in [t_i^k, t_{i+1}^k]$, we have

(3.28) $$d^k(t) = (1-x)d_i^k + x\, d_{i+1}^k$$

where, by hypothesis $d^k(t) \longrightarrow d(t)$ uniformly on $[a, b]$ as k goes to infinity. Subtracting (3.27) from (3.28) leads to

(3.29) $\qquad \| d^k - H_k' \| \le (3/4)\, max_i \,| d^k_{i+1} - d^k_i |.$

Thus, by noticing that it is necessary that $max_i\, |d^k_{i+1} - d^k_i| \longrightarrow 0$ as $k \longrightarrow$ infinity (by hypothesis), we have

$$Lim_{k\to\infty} \| d - H_k' \| \le Lim_{k\to\infty} \|d - d^k\| + Lim_{k\to\infty} \|d^k - H_k'\| = 0,$$

that is, $H_k' \longrightarrow d$ uniformly on $[a, b]$.

Now we show that the sequence $\{H_k\}$ converges on $C^1[a,b]$. Assume, without loss of generality (because the scheme is a local scheme), that $f_0^0 = f_1^0 = f_2^0 = 0$. Then for all k, $f_0^k = f_1^k = f_2^k = 0$ and $d_0^k = d_1^k = 0$. So we have $f(a) = d(a) = 0$ and $H_k(a) = 0$. Now we define a differentiable function

(3.30) $\qquad F(t) := \int_a^t d(s)\, ds, \quad i.e., \quad F'(t) = d(t)$ and $F(a) = 0.$

Then it can be easily proved that

(3.31) $\qquad \|F - H_k\| \le (b-a)\, \| d - H_k' \|.$

Hence H_k converges uniformly to F in $C^1[a,b]$. Finally, we prove that $F(t)$ is just the limit of the sequence $\{f^k\}$. This is because

(3.32) $\qquad \|F - f^k\| \le \|F - H_k\| + \|H_k - f^k\|$

$$\leq \; \|F - H_k\| + (1/2) max_i \, (\Delta t_i^k)^2 \, \|H_k''\|$$

$$\longrightarrow 0 \quad \text{as } k \longrightarrow \text{infinity},$$

where, the *Cauchy* reminder for linear interpolation is used. The above relation means that the control polygon of the corner cutting scheme converges uniformly to a smooth function. This completes the proof.

This *Theorem* indicates that the C^1 convergence of the scheme can be proved if the C^0 convergence property of its *divided difference scheme*, with the same parametrization, can be proved. So we now investigate the *Divided Difference scheme*.

Using the same approach as in *Lemma* 3.1, the following lemma can also be proved.

<u>Lemma</u> 3.5. For all $k, p > 0$,

(3.33) $\quad \|d^{k+p} - d^k\| \;\leq\; 2 \, max_i \, |\Delta d_i^k|.$

To prove that the sequence $\{d^k\}$ is a *Cauchy* sequence in $C[a,b]$, the following lemma is required.

<u>Lemma</u> 3.6. Let

(3.34) $$r_i^k \;:=\; \Delta t_{i+1}^k / \Delta t_i^k$$

then there exist some constants r and R such that for all i and k,

(3.35) $$0 < r \leq r_i^k \leq R < \text{infinity}$$

if

(3.36) $$\underline{a}, \underline{b} > 0,\; 2\overline{a} + \overline{b} < 1 \quad \text{and} \quad \overline{a} + 2\overline{b} < 1.$$

Proof. From the recursion relations of the difference scheme (3.14), we can obtain the following non-linear recursion relations for r_i^k:

(3.37) $$\left|\begin{array}{ll} r_{2i}^{k+1} & = [b_i^k + a_{i+1}^k r_i^k]/[1-a_i^k-b_i^k] \\ r_{2i+1}^{k+1} & = [1-a_{i+1}^k - b_{i+1}^k] r_i^k / [b_i^k + a_{i+1}^k r_i^k] \end{array}\right.$$

Choosing any positive, finite numbers r and R such that (by hypothesis (3.36) this can be done):

(3.38) $$R \geq max_i \{ b_i^k/(1- a_i^k - b_i^k - a_{i+1}^k),\, (1- a_{i+1}^k - b_{i+1}^k - b_i^k)/a_{i+1}^k \},$$

(3.39) $$0 < r \leq min_i \{ b_i^k/(1- a_i^k - b_i^k - a_{i+1}^k),\, (1-a_{i+1}^k - b_{i+1}^k - b_i^k)/a_{i+1}^k \}$$

then it can be easily shown that condition (3.35) can be satisfied, which completes the proof.

From the above results, we can now prove the main result of the smooth corner cutting:

Theorem 3.7 (*C^1 convergence*). The divided difference scheme produces a C^0 limit if (3.36) holds. Hence, the corner cutting scheme produces a C^1 curve.

Proof. From the divided difference scheme (3.23) we obtain

$$(3.40) \quad \left| \begin{array}{ll} \Delta d^{k+1}_{2i} & = c^k_i \, \Delta d^k_i \\ \Delta d^{k+1}_{2i+1} & = (1-c^k_i) \, \Delta d^k_i, \end{array} \right.$$

where, $0 < c^k_i < 1$, and is given by (3.24). Thus

$$(3.41) \quad max_i \, |\Delta d^{k+1}_i| \le C^k \, max_i \, |\Delta d^k_i|,$$

where

$$(3.42) \quad C^k = max_i \, \{c^k_i, 1-c^k_i\}$$

and hence

$$(3.43) \quad 1/2 \le C^k < 1.$$

Condition (3.43) is not strong enough for our purposes (to prove $max_i\{|\Delta d^k_i|\} \longrightarrow 0$) and we wish to show a stronger contraction condition

(3.44) $0 < c^k \le c < 1$

for some constant c independent of k. From (3.24), we have

(3.45) $c_i^k = (1 + 1/s_i^k)^{-1}, \quad 1 - c_i^k = (1 + s_i^k)^{-1},$

where

(3.46) $s_i^k = a_{i+1}^k r_i^k / b_i^k.$

By *Lemma* 3.6 and hypothesis (3.36), it can be concluded that there exist two positive finite numbers $\underline{s}$ and $\overline{s}$ such that $0 < \underline{s} \le s_i^k \le \overline{s} <$ infinity for all i and k. For example, we can chose

(3.46*a*) $\underline{s} = \underline{a} r, \quad \overline{s} = R/\underline{b}.$

Hence (3.44) follows for some positive constant c, $c < 1$.

From *Lemma* 3.5, it now follows that the sequence $\{d^k(t)\}$ is a *Cauchy* sequence in $C[a, b]$, which completes the proof.

Note that condition (3.36) for C^1 convergence requires $(\underline{a}, \underline{b})$ and $(\overline{a}, \overline{b})$ to lie strictly within the region Ω_1 depicted in Figure 3.3.

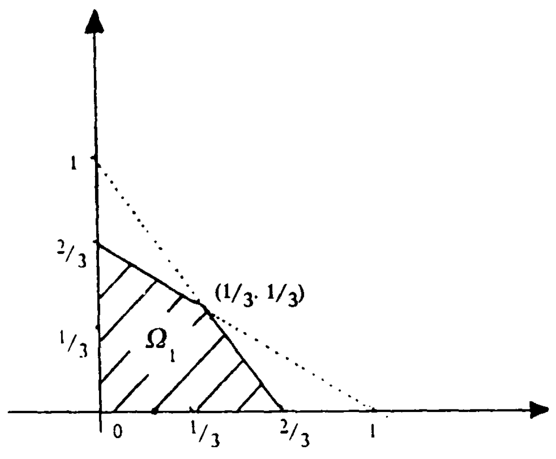


Figure 3.3.

In particular, $(0,0) < (\underline{a}, \underline{b}) < (\overline{a}, \overline{b}) < (1/3, 1/3)$ is a sufficient condition for a C^1 limit. If condition (3.36) is violated, then the convergence to a C^1 limit is no longer guaranteed. For example, with $a_i^k = a_0$, $b_i^k = b_0$, for all i and k, then it can be shown that

$$(3.47) \quad \Delta d_0^{k+1} = c_0^k \Delta d_0^{k-1} = c_0^k c_0^{k-1} c_0^{k-2} \cdots \Delta d_0^0$$

will not converge to zero for general data if $2a_0 + b_0 > 1$. This violates the necessary condition for C^1 convergence. Similarly, $a_0 + 2b_0 > 1$ is not allowable.

<u>Remark</u>: A similar result to *Theorem* 3.7 is also obtained in [81].

3.1.5. The Parametrization is Regular

It has been shown that the corner cutting process produces C^1 curves under condition (3.36), with respect to the *adapted parametrization* which itself is defined by the corner cutting process. It is necessary to prove that this kind of parametrization is regular so that the generated curves are non-singular curves. The following theorem shows that the parametrization is regular.

<u>**Theorem**</u> 3.8. In the case of corner cutting in $\mathbf{R}^N$, $N > 1$, the C^1 limit curve $f(t)$ in *Theorem* 3.7 is regular, that is, $f'(t) = d(t) \neq 0$ for all $t \in [a, b]$, except for the singular case, where, for some i, the initial control points satisfy

$(I)\quad f_i^0 = f_{i+1}^0$

or

$(II)\quad f_{i+1}^0 = (1-x)\, f_i^0 + x\, f_{i-1}^0$ for some $x > 0$.

<u>**Proof**</u>. Let

$$(3.48)\qquad J_k := \{\, d^k(t) \in \mathbf{R}^N\colon\ t \in [a, b] \,\}$$

be the image set of $d^k(t)$. Then from the previous arguments (because the process of the *DD* scheme is a convex combination), we have

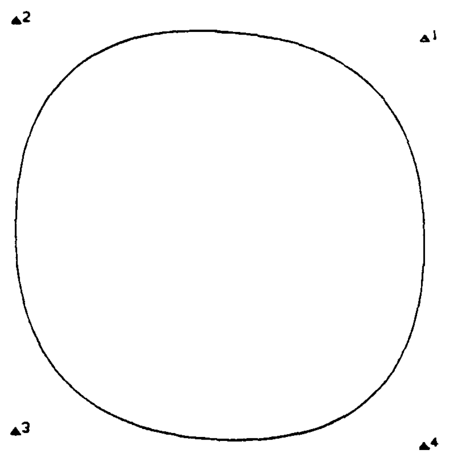
$$(3.49)\qquad J_{k+1} \in J_k \in \dots \in J_0 \text{ for all } k.$$

Thus $d(t) = 0$ implies that $d^0(t) = 0$ for some $t \in [a, b]$ and this can only occur if (i) $d_i^0 = 0$ or (ii) $d_i^0 = -x\, d_{i-1}^0$ for some i and $x > 0$. These are just the conditions given by (I) and (II).

Remark. It can be proved that the limit curve is regular and is at least C^n with respect to its *adapted parametrization* if it is C^n with respect to the diadic parametrization provided that the initial parametric values $\{t_i^0\}$ are chosen appropriately, that is, $t_i^0 < t_{i+1}^0$ for all i.

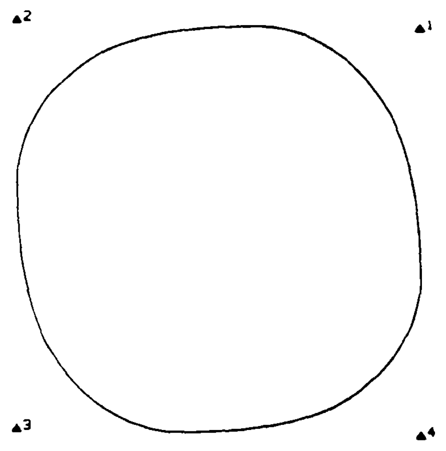
3.1.6. Graphic Examples

We present here four (closed) curves produced by the scheme from the same initial data indicated by $\triangle_i$ by using different parameters. Figure 3.4*a* is the *Chaikin's* algorithm, where $a_i^k = 1/4$, $b_i^k = 1/4$, which is just a uniform B-spline curve. Figure 3.4*b* is a smooth asymmetric curve, where $a_i^k = 1/9$, $b_i^k = 2/9$. The curve in Figure 3.4*c* is continuous but not smooth since the parameter values $a_i^k = 5/12$, $b_i^k = 5/12$ lie outside the C^1 convergence region and violate the necessary condition for C^1 curves. Figure 3.4*d* is a smooth curve produced by choosing the parameters randomly within $[1/9, 2/9]$, which is in the C^1 convergence range.



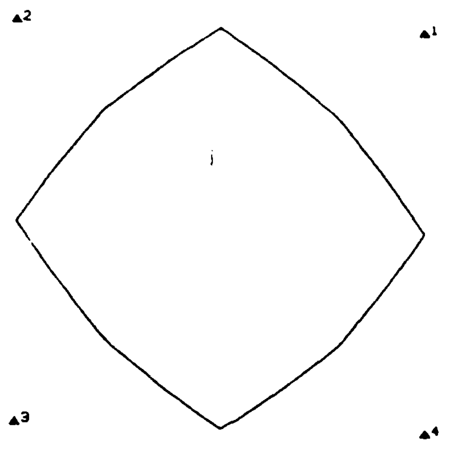
$a_i^k = 1/4,\ b_i^k = 1/4.$

Figure 3.4*a*.



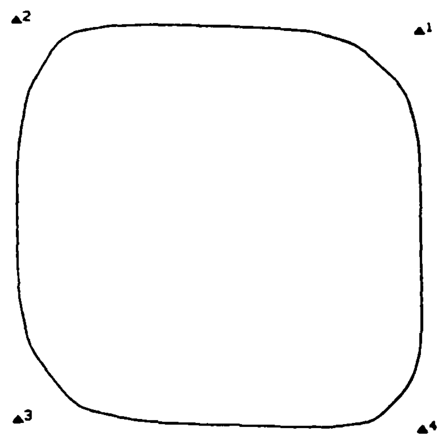
$a_i^k = 1/9,\ b_i^k = 2/9.$

Figure 3.4*b*.



$a_i^k = 5/12,\ b_i^k = 5/12.$

Figure 3.4*c*.



$1/9 \le a_i^k,\ b_i^k \le 2/9,$ *random.*

Figure 3.4*d*.

3.1.7. A Remark

Before going to the next section to discuss the subdivision algorithm for non-uniform B-spline curves, we first give an interesting observation from the geometric structure of non-uniform corner cutting scheme.

From the construction of algorithms for piecewise quadratic polynomial curves in [26,41,115], a question may arise: under what conditions does the above non-uniform corner cutting algorithm produce a (non-uniform) quadratic B-spline curve?

The question is answered as follows (a proof will be given in the next section).

Given any strictly increasing sequence $\{x_i^0\}$ (which is the knot sequence of the B-spline curve to be generated), where $x_i^0 < x_{i+1}^0$ for all i, and some parameters $\{s_i^k\}$ where $0 < \underline{s} \le s_i^k \le \overline{s} < 1$ for all i, k and some constants $\underline{s}$ and $\overline{s}$. Then scheme (3.1) produces a quadratic B-spline curve, with knots $\{x_i^0\}$ and control points $\{f_i^0\}$, if a_i^k and b_i^k are chosen to be

$$(3.50)\quad \left|\begin{array}{l} a_i^k = s_{i-1}^k(x_i^k - x_{i-1}^k)/(x_{i+1}^k - x_{i-1}^k) \\ b_i^k = (1-s_i^k)(x_{i+1}^k - x_i^k)/(x_{i+1}^k - x_{i-1}^k), \end{array}\right.$$

where

$$(3.51)\quad \left|\begin{array}{ll} x_{2i}^{k+1} & = x_i^k \\ x_{2i+1}^{k+1} & = (1-s_i^k)x_i^k + s_i^k\, x_{i+1}^k . \end{array}\right.$$

Remark: If we chose $x_i^0 = i$ and $x_i^k = 1/2$ for all i and k, then (3.50) gives $a_i^k = b_i^k = 1/4$ for all i and k. Thus scheme (3.1) becomes the *Chaikin's algorithm.*

3.2. The recursive subdivision algorithm for Non-uniform B-spline Curves

In this section we discuss a generalization of scheme (3.1) to find a recursive subdivision algorithm for non-uniform B-spline curves. Here, again, the parametrization of the control polygons is the crux of the analysis.

3.2.1. Motivation and Techniques

Although *Chaikin's* and *Catmull-Clark's* algorithms (for curves) have been used for a long time, it seems that no similar recursive subdivision algorithm for non-uniform B-spline curve^ has yet been developed, especially for cubic and quartic B-spline curves which are commonly used. From the above non-uniform corner cutting algorithm and the structures of parabolic (or piecewise parabolic) curves, we know that any quadratic B-spline curve can be generated by the non-uniform corner cutting scheme with parameters given by (3.50) and (3.51). Then, one may ask, can a cubic (or even any) B-spline curve be generated by a scheme similar to (3.1)?

By analysing the non-uniform corner cutting scheme, we derive the non-uniform subdivision algorithm for B-spline curves with simple knots.

The most important tool for the construction of the algorithm is still the *adapted parametrization* as used in our previous analysis. The ideas come from *integrating the non-uniform corner cutting algorithm* to obtain a smoother curve generating algorithm. Because this integrating technique works in the uniform subdivision case (see [51] for details), it is hoped that it should also work in the non-uniform case. Hence, the problem becomes how to integrate a non-uniform ***RSA***. The main difficulty here is how to establish the relations of the parametrization of the scheme and the parametrizations of its related schemes such as its *integrated* and *divided difference schemes.*

At first, one may think that this difficulty can be solved by using the same parametrization as in the case discussed in the previous section. However, it turned out to be too difficult to deal with for higher order schemes. Another choice, one may think, is to treat the scheme and its divided difference scheme separately as two independent schemes. In this case, it is hoped that the scheme might produce curvature continuous curves if, (i) the scheme with its adapted parametrization produces smooth curves and (ii) the divide difference scheme generates smooth curves with its own adapted parametrization. This seems at first quite tempting but, unfortunately, it can be proved not to work by a simple counter-example. This is because the *divided difference scheme* and the *second order divided difference scheme* are interrelated and they should not be treated separately. So some other relatively simple and effective techniques should be introduced.

The difficulty can be overcome when it is considered from another point of view. Our success of the construction of the algorithm is based on the well known result-the *Greville's* identity for B-splines. The process will be shown by an example discussed next.

3.2.2. A Corner Cutting Scheme for Quadratic B-spline curves

Now we construct the ***RSA*** for smooth quadratic B-spline curves. The subdivision scheme for higher order B-spline curves with simple knots can be constructed in the same way. Suppose the scheme (3.1) produces a smooth quadratic B-spline curve with knots $\{x_i^k\}$ and control points $\{f_i^k\}$ associated with the parametric points $\{t_i^k\}$ at level k, where the knot sequence and the parametric point sequence are strictly increasing and $\{x_i^k\} \in \{x_i^{k+1}\}$ for all k. Because the scheme is a continuous refinement of the control polygons, like the *Chaikin's* algorithm, this assumption is reasonable. Then, *Greville's* identity [19] suggests that the knots $\{x_i^k\}$ and the parametric points $\{t_i^k\}$ are interrelated by

$$(3.52) \qquad t_i^k = (x_{i-1}^k + x_i^k)/2 \quad \text{for all } i, k.$$

So, if a scheme in form (3.1) is constructed such that (3.52) is always satisfied, then the scheme should produce the quadratic B-spline curve with knots $\{x_i^0\}$ and *de Boor* points $\{f_i^0\}$ because of the uniqueness of B-spline curves.

Now we construct the required scheme. Firstly, by assumption, the scheme for the knots $\{x_i^k\}$ must be in the form

$$(3.53)\quad \left| \begin{array}{ll} x_{2i}^{k+1} & = x_i^k \\ x_{2i+1}^{k+1} & = (1 - s_i^k)x_i^k + s_i^k\, x_{i+1}^k \end{array} \right.$$

where

$$(3.54)\quad 0 < \underline{s} \le s_i^k \le \overline{s} < 1, \text{ for all } i, k.$$

From the relation (3.52) and (3.53), the recursive relations for the parametric points $\{t_i^k\}$ can easily be obtained as

$$(3.55)\quad \left| \begin{array}{ll} t_{2i}^{k+1} & = (1 - a_i^k)t_i^k + a_i^k\, t_{i+1}^k \\ t_{2i+1}^{k+1} & = b_i^k\, t_i^k + (1 - b_i^k)\, t_{i+1}^k \end{array} \right.$$

where,

$$(3.56)\quad \left| \begin{array}{ll} a_i^k & = s_{i-1}^k\,(x_i^k - x_{i-1}^k)/\,(x_{i+1}^k - x_{i-1}^k) \\ b_i^k & = (1 - s_i^k)\,(x_{i+1}^k - x_i^k)/(x_{i+1}^k - x_{i-1}^k). \end{array} \right.$$

Thus, from the above discussion, we can conclude that scheme (3.1) produces a quadratic B-spline curve with control points $\{f_i^0\}$ and knots $\{x_i^0\}$ if the shape parameters a_i^k and b_i^k are chosen to satisfy (3.53), (3.54) and (3.56) (because the *adapted parametrization* is used).

A simple mathematical explanation of this result is like this. From

Theorem 3.4, and the fact that the *divided difference scheme* (3.23) converges to the piecewise linear curve (with respect to the *knot* point $\{x_i^k\}$ parametrization) which is the initial *divided difference control polygon* with both ends being cut, the scheme converges to a piecewise quadratic curve with broken points $\{x_i^0\}$ since t_i^k and x_i^k are very close (3.52). That is, the limit curve is the required B-spline curve. A systematic proof of the result will be given in subsection 3.2.4.

For higher order B-spline curves, the subdivision scheme can be derived in a similar way which will be discussed in the next subsections. The idea is the same but the calculations of the corresponding coefficients are more complicated. We will also use the knot insertion technique to construct the subdivision algorithm for B-spline curves with simple knots.

3.2.3. Recursive Subdivision Algorithms for Cubic and Quartic B-spline Curves

From the above results and the *Catmull-Clark's* algorithm for uniform cubic B-spline curves, it is expected that a similar non-uniform recursive subdivision algorithm could generate non-uniform cubic B-spline curves. Now we construct the scheme.

Firstly, we assume that the scheme is in the form

$$(3.57)\quad \left| \begin{array}{ll} f_{2i}^{k+1} & = A_i^k\, f_i^k + (1-A_i^k)\, f_{i+1}^k \\ f_{2i+1}^{k+1} & = B_i^k\, f_i^k + (1-B_i^k - C_i^k) f_{i+1}^k + C_i^k\, f_{i+2}^k \end{array} \right.$$

and it produces a cubic B-spline curve with control points $\{f_i^k\}$ and knots $\{x_i^k\}$, where, $x_i^k < x_{i+1}^k$ for all i and k. Then, from the discussion in the previous section, the parametric points (*adapted parametrization*) $\{t_i^k\}$ should satisfy the *Greville's identity* at each level. That is,

(3.58) $$t_i^k = (x_{i-2}^k + x_{i-1}^k + x_i^k)/_3 \quad \text{for all } i, k$$

where, $\{x_i^k\}$ are determined by (5.53). Furthermore, the *divided difference scheme* of *scheme* (3.57) given by

(3.59) $$\left|\begin{array}{ll} d_{2i}^{k+1} & = (1 - a_i^k)d_i^k + a_i^k\, d_{i+1}^k \\ d_{2i+1}^{k+1} & = b_i^k\, d_i^k + (1 - b_i^k)\, d_{i+1}^k \end{array}\right.$$

where

(3.60) $$\left|\begin{array}{l} a_i^k = A_i^k\, \Delta t_{i+1}^k/[(A_i^k - B_i^k)\Delta t_i^k + C_i^k\, \Delta t_{i+1}^k] \\ b_i^k = B_i^k\, \Delta t_i^k\,/[B_i^k\, \Delta t_i^k + (1 - A_{i+1}^k - C_i^k)\Delta t_{i+1}^k] \end{array}\right.$$

should generate a quadratic B-spline with the same knots $\{x_i^k\}$. More precisely, the coefficients in (3.59) should satisfy the quadratic B-spline scheme constraints

(3.61) $$u_i^k = (x_{i-1}^k + x_i^k)/2 \quad \text{for all } i, k$$

where, $\{u_i^k\}$ are the *adapted parametric points* of scheme (3.59).

Not surprisingly, the under determined linear systems (two constraints

(3.58) and (3.61) and three variables, the coefficients: A_i^k, B_i^k and C_i^k) has a unique solution

$$\text{(3.62)} \quad \left| \begin{aligned} A_i^k &= (x_{i+1}^k - x_{2i-1}^{k+1})/(x_{i+1}^k - x_{i-2}^k) \\ B_i^k &= \{(x_{i+1}^k - x_{2i+1}^{k+1})/(x_{i+1}^k - x_{i-1}^k)\}\, A_i^k \\ C_i^k &= \{(x_{2i-1}^{k+1} - x_{i-1}^k)/(x_{i+1}^k - x_{i-1}^k)\}(1-A_{i+1}^k) \end{aligned} \right.$$

Thus, from the above discussion, we can conclude:

__Theorem__ 3.9. The non-uniform scheme (3.57) produces a cubic B-spline curve with given knots $\{x_i^0\}$ and control points $\{f_i^0\}$ if the coefficients A_i^k, B_i^k and C_i^k are chosen by (3.62) and the refined knot sequence $\{x_i^k\}$ is given by (3.53) satisfying (3.54).

__Remark__: If the initial knots are equally spaced and the subdivision parameter $s_i^k = 1/2$ for all i and k, then we have $A_i^k = 1/2$, and $B_i^k = C_i^k = 1/8$, and the scheme (3.57) becomes the *RSA* for uniform cubic B-spline curves (*Catmull-Clark's Algorithm*).

Similarly, we can obtain the ***RSA*** for quartic B-spline curves:

__Theorem__ 3.10. The non-uniform scheme

$$\text{(3.63)} \quad \left| \begin{aligned} f_{2i}^{k+1} &= A_i^k f_i^k + (1-A_i^k-B_i^k)f_{i+1}^k + B_i^k f_{i+2}^k \\ f_{2i+1}^{k+1} &= Y_i^k f_i^k + (1-Y_i^k-Z_i^k)f_{i+1}^k + Z_i^k f_{i+2}^k \end{aligned} \right.$$

produces a quartic B-spline curve with knots $\{x_i^0\}$ and control points $\{f_i^0\}$ if the coefficients A_i^k, B_i^k, Y_i^k and C_i^k satisfy

$$(3.64)\quad \left|\begin{aligned}
A_i^k &= \frac{(x_{i+2}^k - x_{2i-1}^{k+1})(x_{i+2}^k - x_{2i+1}^{k+1})}{(x_{i+2}^k - x_{i-2}^k)(x_{i+2}^k - x_{i-1}^k)} \\[2ex]
B_i^k &= \frac{(x_{2i-1}^{k+1} - x_{i-1}^k)(x_{2i+1}^{k+1} - x_{i-1}^k)}{(x_{i+2}^k - x_{i-1}^k)(x_{i+3}^k - x_{i-1}^k)} \\[2ex]
Y_i^k &= \frac{(x_{2i+3}^{k+1} - x_{2i+1}^{k+1})(x_{i+2}^k - x_{2i+1}^{k+1})}{(x_{i+2}^k - x_{i-2}^k)(x_{i+3}^k - x_{i-1}^k)} \\[2ex]
Z_i^k &= \frac{(x_{2i+1}^{k+1} - x_{i-1}^k)(x_{2i+3}^{k+1} - x_{i-1}^k)}{(x_{i+2}^k - x_{i-1}^k)(x_{i+3}^k - x_{i-1}^k)}.
\end{aligned}\right.$$

Here the refined knot sequence $\{x_i^k\}$ is given by (3.53) satisfying (3.54).

Remark: If the initial knots are equally spaced and the subdivision parameter $s_i^k = 1/2$ for all i, k, then we have $A_i^k = Z_i^k = 5/16$, and $B_i^k = Y_i^k = 1/16$, and the scheme (3.63) becomes the ***RSA*** for uniform quartic B-spline curves (*Example* 1.5 in section 1.2).

The above techniques can also be used to produce B-spline curves of any order with simple knots. The details will be given in subsection 3.2.5.

3.2.4. Theoretical Proof of these Results

In this subsection, we give the proofs of the above results, *Theorem* 3.9 and *Theorem* 3.10.

It should be pointed out that from the construction of B-spline curves and *Boehm's* knot insertion algorithm (or the *Oslo* algorithm) for such curves, there are non-uniform recursive subdivision algorithms, expressed in forms (3.1), (3.57) and (3.63), which are used in the generation of B-spline curves of any order [14,33,89]. Here the subdivision process is explained as an insertion of a new knot between every two adjacent knots. From the uniqueness of B-splines, such algorithms are unique in some sense.

The scheme for quadratic B-spline curves can be easily proved by many techniques. Consequently, only scheme (3.57), that is, the scheme for cubic B-spline curves, is proved here. Scheme (3.63) can be proved in much the same way.

Suppose $\{B^k_{i,4}(x)\}$ are the normalized cubic B-spline basis functions with knots $\{x^k_i\}$ for all k, where the knots are defined by (3.53) and (3.54). Here, the subscript 4 is the order of the spline curve. Then, by definition, we have

$$(3.65)\qquad B^k_{i,4}(x) := \langle x^k_i, x^k_{i+1}, \ldots, x^k_{i+4}\rangle_t \,(x-t)^3_+$$

$$= \sum\nolimits_{i=0}^{4} b^k_{i,j,4}\, B^{k+1}_{2i+j,4}(x)$$

$$= \sum_{i=0}^{4} b^k_{i,j,4} \, B^{k+1}_{2i+j,4}(x)$$

where, $<x^k_i, x^k_{i+1}, \ldots, x^k_{i+4}>_t$ is the *divided difference* operator with respect to variable t and $b^k_{i,j,4}$ depends only on the knots $\{x^{k+1}_i\}$. These $\{b^k_{i,j,4}\}$ will be given explicitly later. Hence, for any twice continuously differentiable cubic B-spline curve with knots $\{x^k_i\}$, we have

$$(3.66) \qquad P(x) := \sum_j f^k_i \, B^k_{i,4}(x)$$

$$= \sum_i f^k_i \sum_{i=0}^{4} b^k_{i,j,4} \, B^{k+1}_{2i+j,4}(x)$$

$$= \sum_i f^{k+1}_i \, B^{k+1}_{i,4}(x)$$

where, $\{f^{k+1}_i\}$ are given by the following formulae

$$(3.67) \qquad f^{k+1}_{2i} := f^k_{i-2} \, b^k_{i-2,4,4} + f^k_{i-1} \, b^k_{i-1,2,4} + f^k_i \, b^k_{i,0,4}$$

$$(3.68) \qquad f^{k+1}_{2i+1} := f^k_{i-1} \, b^k_{i-1,3,4} + f^k_i \, b^k_{i,1,4}.$$

Relations (3.67) and (3.68) define a linear non-uniform subdivision scheme since these coefficients $\{b^k_{i,j,4}\}$ (called *Discrete B-spline* in [33]) varies with i, j and k and are determined by the knot sequence. By construction we know that the scheme just refines the control points (*de Boor* points) of the B-spline curve with knots $\{x^0_i\}$ and control points $\{f^0_i\}$. Hence, in order to prove *Theorem* 3.9, we only need to prove that the scheme given by (3.67) and (3.68) is the same as that given by (3.57).

From the recurrence relations of B-splines, it can be shown that these coefficients $\{b^k_{i,j,4}\}$ are defined to be some ratios of the knot intervals (see subsection 3.2.5),

$$(3.69)\quad \left|\begin{array}{l} b^k_{i,3,4} = (\, x^k_{i+4} - x^{k+1}_{2i+5})/(x^k_{i+4} - x^k_i) \\ b^k_{i,4,4} = \{(x^k_{i+4} - x^{k+1}_{2i+7})/(x^k_{i+4} - x^k_{i+2})\}\, b^k_{i,3,4} \\ b^k_{i,0,4} = \{(x^{k+1}_{2i+5} - x^k_{i+2})/(x^k_{i+4} - x^k_{i+2})\}(1 - b^k_{i,3,4}). \end{array}\right.$$

Therefore, the two schemes are the same except that the notations are different. However, from the schematic point of view, the shift of indices is not important and it is obvious that the two schemes produce the same curve. This completes the proof of the theorem.

3.2.5. The Recursive Subdivision Algorithm for B-spline Curves of Order n with Simple Knots

In this subsection, we formulate the general subdivision scheme for B-spline curves of order n ($n > 1$) with simple knots.

Let $\{B^k_{i,n}(x)\}$ denote the normalized B-spline basis of order n with knots $\{x^k_i,\ x^k_i < x^k_{i+1}\}$ and $\{B^{k+1}_{i,n}(x)\}$ the normalized B-spline basis of order n with knots $\{x^{k+1}_i\!: x^{k+1}_i < x^{k+1}_{i+1}\}$, where

$$(3.70)\quad \left|\begin{array}{ll} x^{k+1}_{2i} & = x^k_i \\ x^{k+1}_{2i+1} & = (1 - s^k_i)x^k_i + s^k_i\, x^k_{i+1} \end{array}\right.$$

and

(3.71) $0 < \underline{s} \leq s_i^k \leq \overline{s} < 1$

for some constants $\underline{s}$ and $\overline{s}$ for all i, k. Then, from the definition of the B-spline basis [23], we have, as in (3.65), for all $n > 1$,

(3.72) $B_{i,n}^k(x) = \sum_{j=0}^n b_{i,j,n}^k B_{2i+j,n}^{k+1}(x),$

where, $\{b_{i,j,n}^k\}$ are determined by the knots $\{x_m^{k+1},\ m = 2i, 2i+1, ..., 2i+2n\}$ only.

Now, suppose a B-spline curve $P(x)$ is given by

(3.73) $P(x) := \sum_i P_i^k B_{i,n}^k(x).$

Then, by using the same arguments as in subsection 3.2.4, the following can be obtained:

(3.74) $P(x) = \sum_i P_i^k B_{i,n}^k(x)$

$= \sum_i P_i^k \sum_{j=0}^n b_{i,j,n}^k B_{2i+j,n}^{k+1}(x)$

$= \sum_i P_i^{k+1} B_{i,n}^{k+1}(x)$

where

$$(3.75)\quad \left|\begin{array}{lcl} P^{k+1}_{2i} & := & \sum_{j=0}^{[n/2]} b^k_{i-j,2j,n}\, P^k_{i-j} \\ \\ P^{k+1}_{2i+1} & := & \sum_{j=0}^{[(n-1)/2]} b^k_{i-j,2j+1,n}\, P^k_{i-j}. \end{array}\right.$$

It is obvious that the above relations defines a non-uniform subdivision scheme for B-spline curves with simple knots. Moreover, the B-spline relation coefficients $\{b^k_{i,j,n}\}$ are just the weights of the subdivision scheme.

Now, we derive the recurrence relations $\{b^k_{i,j,n}\}$ on n, the order of the splines. For simplicity, it is assumed that

$$(3.76)\quad b^k_{i,j,n} = 0 \text{ when } j < 0 \text{ or } j > n.$$

When $n = 1$ and 2, it can easily be shown that

$$(3.77)\quad b^k_{i,0,1} = b^k_{i,1,1} = 1$$

and

$$(3.78)\quad \left|\begin{array}{ll} b^k_{i,0,2} & = (x^{k+1}_{2i+1} - x^{k+1}_{2i})/(x^k_{i+1} - x^k_i) \\ b^k_{i,1,2} & = 1 \\ b^k_{i,2,2} & = (x^{k+1}_{2i+4} - x^{k+1}_{2i+3})/(x^k_{i+2} - x^k_{i+1}). \end{array}\right.$$

For $n \geq 3$, we have, from (3.72),

$$(3.79)\quad d(B^k_{i,n}(x))/dx = \sum_{j=0}^{n} b^k_{i,j,n}\; d\, B^{k+1}_{2i+j,n}(x)/dx.$$

Substituting the above derivative terms by the B-spline formula

$$(3.80)\qquad d(B^k_{i,n}(x))/dx = (n-1)\{ B^k_{i,n-1}(x)/(x^k_{i+n-1} - x^k_i) - B^k_{i+1,n-1}(x)/(x^k_{i+n} - x^k_{i+1})\}$$

and then replacing the terms $B^k_{i,n-1}(x)$ by (3.72) for $n-1$, by rearranging the appropriate terms, we can obtain

$$(3.81)\qquad \frac{\sum_{j=0}^{n} b^k_{i,j,n-1}\, B^{k+1}_{2i+j,n-1}(x)}{x^k_{i+n-1} - x^k_i} - \frac{\sum_{j=0}^{n} b^k_{i+1,j,n-1}\, B^{k+1}_{2i+j+2,n-1}(x)}{x^k_{i+n} - x^k_{i+1}}$$

$$= \sum_{j=0}^{n} b^k_{i,j,n}\left\{ \frac{B^{k+1}_{2i+j,n-1}(x)}{x^{k+1}_{2i+j+n-1} - x^{k+1}_{2i+j}} - \frac{B^{k+1}_{2i+j+1,n-1}(x)}{x^{k+1}_{2i+j+n} - x^{k+1}_{2i+j+1}} \right\}.$$

Because the basis $\{B^{k+1}_{i,n-1}(x)\}$ are linearly independent for all $x \in \boldsymbol{R}$, equality (3.81) can only be true if and only if the coefficients of $B^{k+1}_{i,n-1}(x)$, $i \in \mathbf{Z}$, are the same in both sides. This gives the following recurrence relations ($n \geq 3$):

$$(3.82)\qquad b^k_{i,j,n} = b^k_{i,j-1,n} + \frac{x^{k+1}_{2i+j+n-1} - x^{k+1}_{2i+j}}{x^{k+1}_{2i+2n-2} - x^{k+1}_{2i}}\, b^k_{i,j,n-1}$$

$$+ \frac{x^{k+1}_{2i+j+n-1} - x^{k+1}_{2i+j}}{x^{k+1}_{2i+2n} - x^{k+1}_{2i+2}}\, b^k_{i+1,j-2,n-1}.$$

From (3.77), (3.78), (3.82) and the assumption (3.76), all the weights $\{b^k_{i,j,n}\}$ can be obtained. A special case of (3.82) is the uniform case whereby all the knots are equally spaced. Then this recurrence relation becomes

$$(3.83)\qquad b^k_{i,j,n} = b^k_{i,j-1,n} + 1/2(b^k_{i,j,n-1} - b^k_{i+1,j-2,n-1}).$$

This linear difference equation, together with the initial condition (3.77), (3.78) and convention (3.76), has a unique solution:

(3.84) $$b^k_{i,j,n} = \frac{(n-1)!}{j!(n-j)!}$$ *(the binomial coefficients).*

This result is the same as the *line averaging algorithm for uniform B-splines* described by *Lane-Riesenfeld* in [76].

3.2.6. Remarks

1. The quadratic B-spline generating non-uniform corner cutting scheme can be proved by many methods. A simple geometric proof comes from the properties of parabolic curves.

2. The subdivision scheme for B-spline curves is just a refinement scheme. They can be regarded as a generalized *Boehm's* knot insertion algorithm (simultaneous knot insertion).

3. The C^1 condition (3.36), which is obtained in our analysis, is just a sufficient condition for the scheme to produce smooth curves. It can be shown that the necessary and sufficient condition for the scheme to produce smooth curves is

(3.85) $\quad Lim_{k\to\infty} max_i|\Delta d_i^k| = 0.$

4. From the perturbation point of view, more complicated smooth (non-polynomial spline) curves can be generated by these schemes when their coefficients are slightly perturbed [54].

5. The subdivision scheme can easily be generalized to surfaces. Also, any tensor-product B-spline (with simple knots) surface can be computed by a corresponding subdivision algorithm (uniform or non-uniform algorithm).

6. An important application of these algorithms is that, due to their flexibility, they are very useful for interactive design. For instance, for the same control polygon, different curves or surfaces can be produced if different knots are chosen. Also, by adjusting some appropriate control points, the curves and surfaces can be controlled easily .

7. The non-uniform scheme is a special case of the *Oslo* algorithm for B-splines [33]. Hence, if the initial knots are equally spaced and the new knots are spread uniformly, then the scheme degenerates to the *Lane-Riesenfeld's* line average algorithm [76].

3.3. Conclusions

A simple non-uniform corner cutting scheme is investigated and the sufficient conditions for smooth curve generating schemes are given. By using the *adapted parametrization,* rather than the uniform parametrization used in [50,85], the curves obtained are smoother than would be expected by the analysis in [21] in the case of simple corner cutting case.

Based on the analysis of the non-uniform corner cutting scheme (3.1) and the adapted parametrization, the non-uniform recursive subdivision algorithms for B-spline curves of any order are derived.

The key to the success of these studies is the *Greville's* identity for B-spline functions relating the non-uniform parametrization and the parametrization of their *divided differences.*

Other relatively simple proofs of the results can also be obtained by using either the curve refinement techniques or the *Boehm's* knot insertion ideas.

CHAPTER FOUR

RECURSIVE SUBDIVISION ALGORITHMS FOR SURFACES: AN INTRODUCTION

In this Chapter, some of the currently used recursive subdivision algorithms for the generation of surfaces are briefly described. They can broadly be classified into three types: (i) tensor-product type algorithms, which are generalizations of the curve genertating *DGL* schemes; (ii) subdivision algorithms based on uniform triangular control polyhedrons and (iii) non-uniform subdivision algorithms, for example, the constructive algorithm based on the local geometry of the control polyhedrons derived by *N. Dyn, D. Levin* and *D. Liu.*

4.0. Notation

In order to facilitate our description on the subdivision algorithms, the following notations are introduced. They will be used throughout the rest of the thesis.

$P^k_{i,j}$: control points at level k, and $i, j, k \geq 0$;

P^k: a vector whose components are part of $\{P^k_{i,j}\}$;

A: the local subdivision matrix at an extraordinary point;

N: integer, the indicator of the extraordinary point, $N \geq 3$, $N \neq 4$;

$\{a_i\}$: local shape control parameters;

$\{a_{m,n}, b_{m,n}, c_{m,n}, d_{m,n}\}$: weighting coefficients of algorithms;

LB_2, LB_3, LB_4: *Linear operators* for bi-quadratic, bi-cubic and bi-quartic B-spline algorithms respectively.

4.1. Recursive Subdivision Algorithms for Tensor-product B-spline Patches

Since any uniform B-spline curve can be generated by a uniform recursive subdivision algorithm [51,76,91], its tensor-product counterparts can also be generated. For example, *Doo-Sabin's* algorithm generates uniform bi-quadratic B-spline surfaces over uniform data, *Catmull-Clark's* algorithm generates uniform bi-cubic B-spline surfaces over uniform data. In this section, we present a summary of this type of algorithms.

4.1.1. The Doo-Sabin's Algorithm

The *Doo-Sabin's* algorithm over uniform data (*tensor-product-type data*) is the generalization of *Chaikin's algorithm* for surfaces [46]. The algorithm is defined by the following refinement equations

$$(4.1)\quad \left| \begin{array}{lcl} P^{k+1}_{2i,2j} & = & (9/16)P^k_{i,j} + (1/16)P^k_{i+1,j+1} + (3/16)P^k_{i,j+1} + (3/16)P^k_{i+1,j} \\ P^{k+1}_{2i,2j+1} & = & (3/16)P^k_{i,j} + (3/16)P^k_{i+1,j+1} + (9/16)P^k_{i,j+1} + (1/16)P^k_{i+1,j} \\ P^{k+1}_{2i+1,2j} & = & (3/16)P^k_{i,j} + (3/16)P^k_{i+1,j+1} + (1/16)P^k_{i,j+1} + (9/16)P^k_{i+1,j} \\ P^{k+1}_{2i+1,2j+1} & = & (1/16)P^k_{i,j} + (9/16)P^k_{i+1,j+1} + (3/16)P^k_{i,j+1} + (3/16)P^k_{i+1,j} \end{array} \right.$$

where, $\{P^k_{i,j}\}$ is the *control net* (*control polyhedron*) at the k-th lever and $\{P^{k+1}_{i,j}\}$ the control net at the k+1st level, that is, the refined *control net.*

It is obvious that the weights of the algorithm $\{9/16, 3/16, 3/16, 1/16\}$ are just the tensor products of the weightings of the *Chaikin's* algorithm $\{3/4, 1/4\}$. In fact, this is true for any uniform tensor-product type subdivision scheme.

The above algorithm is characterized by the linear operator LB_2: $R^{2,2} \to R$ defined by

$$(4.2)\quad y := LB_2(X) := (9/16)X_{1,1} + (1/16)X_{2,2} + (3/16)X_{1,2} + (3/16)X_{2,1}.$$

The most important feature of *Doo-Sabin's* algorithm is the technique used to treat non-uniform data. The technique divides the data into uniform data and non-uniform data and then isolates the non-uniform data by some locally uniform data on which the uniform scheme can be applied. For non-uniform data, some special local algorithms are introduced so that each group of the non-uniform data converges to the so-called *Extraordinary Point* (*E-point*). We should emphasize here that the number of *E-points* may be increased at the first subdivision but remains unchanged after that.

Since the uniform scheme is applied at ordinary points, the surface, in the limit, is a bi-quadratic B-spline patch. Thus, the surface is smooth at every point except for a fixed number of the so-called *extraordinary points*. To analyse the properties of the limit surfaces, it therefore suffices to analyse them only at these *extraordinary points*.

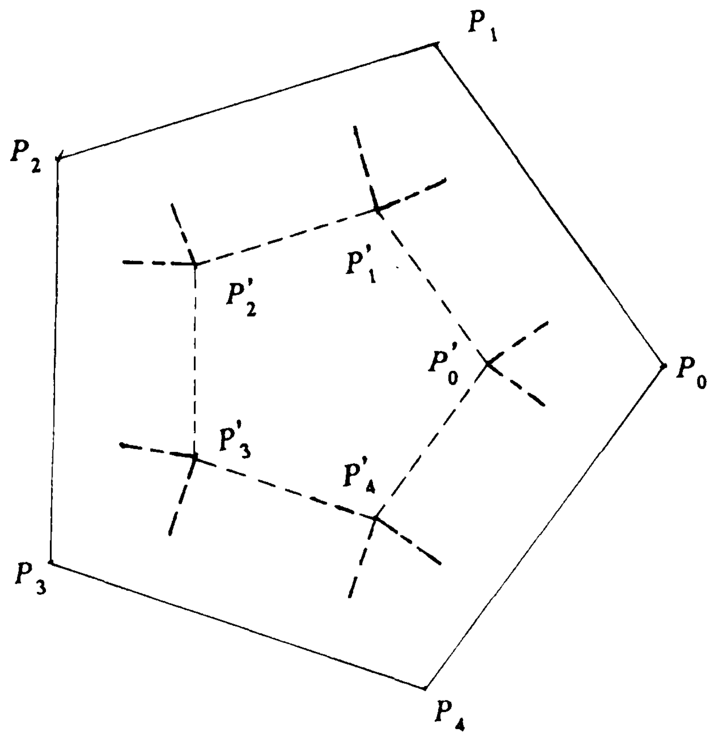


Figure 4.1.

Figure 4.1 shows the *Doo-Sabin's* algorithm near some non-uniform data (a 5-sided face) which is isolated by uniform data.

On applying the algorithm, a smaller N-sided face is produced. The new vertices of the N-sided face are obtained by the formula:

(4.3) $P'_0 := \sum_{i=0}^{N-1} a_i P_i$

where, the weghtings $\{a_i\}$ are given by

(4.3*a*) $a_i := [3 + 2\cos(2\pi i/_N)]/_{4N}$, $i = 0, 1, 2, \dots, N-1.$

Other points $\{P'_j\}$, $j = 1, 2, \dots, N-1$, can be calculated symmetrically. Some alternatives to this formula is also discussed in [46].

The following theorem states the main result about the *Doo-Sabin's* algorithm [33].

<u>Theorem</u> 4.1. The surface produced by the *Doo-Sabin's* algorithm has the following properties: (i) The surface is C^1 at any regular point and (ii) The surface has a unique tangent plane at any *E-point*. That is, for general data, the limit surfaces are smooth.

4.1.2. The Catmull-Clark's Algorithm

In 1978, *Catmull* and *Clark* developed a subdivision algorithm for the generation of uniform bi-cubic B-spline surfaces in [27]. Initially, they hoped that the algorithm may produce better results than that of the *Doo-Sabin's* algorithm. In fact, it turned out that the algorithm could not, in general, produce curvature continuous surfaces even when some optimized parameters are used [1,2,3,116]. However, the generated surfaces behave quite well since they are C^2 everywhere except the *E-points*.

The algorithm over uniform data is characterized by the following refinement equations

$$(4.4)\quad \left|\begin{array}{lcl}
P^{k+1}_{2i,2j} & = & (1/4)P^k_{i,j} + (1/4)P^k_{i+1,j+1} + (1/4)P^k_{i,j+1} + (1/4)P^k_{i+1,j} \\
P^{k+1}_{2i,2j+1} & = & (1/16)P^k_{i,j} + (6/16)P^k_{i+1,j+1} + (6/16)P^k_{i,j+1} + (1/16)P^k_{i+1,j} \\
 & & + (1/16)P^k_{i,j+2} + (1/16)P^k_{i+1,j+2} \\
P^{k+1}_{2i+1,2j} & = & (1/16)P^k_{i,j} + (6/16)P^k_{i+1,j+1} + (1/16)P^k_{i,j+1} + (6/16)P^k_{i+1,j} \\
 & & + (1/16)P^k_{i+2,j} + (1/16)P^k_{i+2,j+1} \\
P^{k+1}_{2i+1,2j+1} & = & (1/64)P^k_{i,j} + (36/64)P^k_{i+1,j+1} + (6/64)P^k_{i,j+1} + (6/64)P^k_{i+1,j} \\
 & & + (1/64)P^k_{i+2,j} + (6/64)P^k_{i+2,j+1} + (1/64)P^k_{i,j+2} \\
 & & + (6/64/)P^k_{i+1,j+2} + (1/64)P^k_{i+2,j+2}.
\end{array}\right.$$

The weightings of the algorithm are $1/4$, $1/16$, $6/16$, $1/64/$, $6/64$, and $36/64$. From (4.4), we observe that the algorithm is composed of three different formulae: the vertex point formula (the forth equation), the edge point formula (the second and the third equations) and the face point formula (the first (the second and the third equations) and the face point formula (the first equation).

For non-uniform data, the scheme is modified, in a similar way to the *Doo-Sabin's* algorithm, to separate the non-uniform data from the uniform data and then isolates each group of them. The technique guarantees that each group of the data, which is surrounded by locally uniform data, converges to an extraordinary point. Thus, the analysis of the algorithm becomes an extraordinary point analysis.

The algorithm at the near-extraordinary points is as follows. The extraordinary point in this case is an N-spoked vertex (When $N = 4$, the algorithm degenerates to the uniform algorithm). The formula for the new edge points and the new face points are the same as the corresponding formulae in (4.4). The formula for the new vertex V' is given by

$$(4.5) \quad V' := AV + BG + CQ,$$

where, V is the old vertex, G is the average of the new face points surrounding the N-node and Q is the average of the old vertex points connected to V by an edge; A, B and C are the free weights satisfying

$$(4.6) \quad 0 \leq A, B, C \quad \text{and} \quad A + B + C = 1.$$

By introducing the *subdivision matrix* at extraordinary point and

[46]) and using some results in differential geometry, the following result is obtained in [1,2,3].

Theorem 4.2. The algorithm produces a uniform bi-cubic B-spline patch over uniform data. At any extraordinary point, the limit surface has a unique tangent plane if the parameters A, B and C are chosen properly. Furthermore, the surface, in general, is not curvature continuous at the extraordinary point, although it is C^2 at other points.

Figure 4.2 shows the algorithm near an N-spoked vertex V (only when N=4, the vertex is called a regular vertex). Here, $N = 5$.

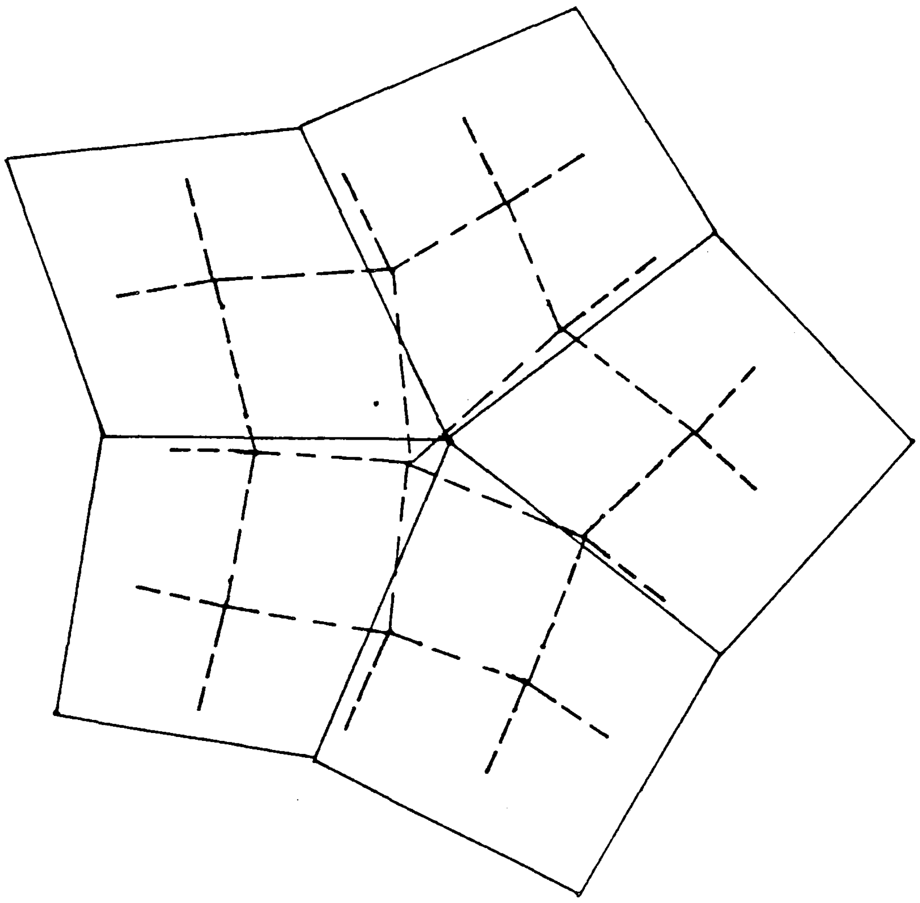


Figure 4.2.

4.1.3. The Uniform Bi-quartic B-spline Algorithm

This algorithm is the tensor-product generalization of the corresponding uniform scheme for the uniform quartic B-spline curves. It produces uniform bi-quartic B-spline surface patches over uniform data. For arbitrary data, the scheme separates the uniform data from the non-uniform data and then isolates the extraordinary points by using some local techniques. Hence, the generated surfaces are C^3 continuous except at the *extraordinary points*. At these *E-points*, special techniques are used to analyse the smoothness properties of the surfaces. The details about this algorithm will be given in Chapter 5.

4.2. Some Uniform Tensor-product Type Algorithms

In this section, we list some uniform subdivision algorithms generating tensor-product type surfaces.

4.2.1. The Tensor-Product of Dyn-Gregory-Levin's Algorithm

It can be shown that the *DGL's* subdivision scheme for curves [48] can be generalized to produce tensor product surfaces. For simplicity, however, only the *DGL's* 4-point interpolatory subdivision algorithm is given as an example. The scheme produces smooth interpolatory surfaces over uniform data. This scheme will be described in Chapter 6.

4.2.2. Other Tensor-product Type Algorithms

Besides the above discussed recursive subdivision algorithms for surfaces, there are many other ***RSA*** generating tensor product surfaces. One of them is the generalization of the non-uniform B-spline algorithms (discussed in Chapter 3) to produce tensor-product non-uniform B-spline surfaces. Another is the subdivision algorithm for the computation of tensor-product polynomial surfaces. One such example may be the line averaging algorithm for certain *Cube-spline* (*Box-spline*) surfaces [20,28,87].

These algorithms are direct generalization from the curve cases. Thus the (smoothness) properties of the schemes remain the same if the data is uniform. For non-uniform data, some special techniques are needed to treat the extraordinary points. However, the details will not be discussed in the thesis. The interested reader is referred to the papers [2,28,46,92].

4.3. Recursive Subdivision Algorithms Based on Triangulations

Another class of recursive subdivision algorithms is the *simplex-based* algorithms. They are constructed over triangulations in the 3-dimensional space. Both the *DGL* scheme and the *Micchelli-Prautzsch* (*MP*) scheme can be generalized to produce surfaces over uniform triangulations. In fact, we will show in Chapter 6 that the tensor-product type algorithms is just a special case of this class of algorithms.

Since the *DGL* approach and the *MP* scheme are equivalent in some sense, we discuss only the generalization of the *DGL* scheme over uniform triangulations.

4.3.1. A 10-point Interpolatory Recursive Subdivision Algorithm

This algorithm is a generalization of the *DGL's* 4-point interpolatory scheme [48] for surfaces over triangulations. The scheme has two shape control parameters and uses 10 local control points to refine the control nets. It produces smooth interpolatory surfaces over arbitrary triangulations.The details about this scheme will be given in Chapter 6.

4.3.2. A General Subdivision Scheme Defined over Uniform Triangulations

The *DGL* scheme for curves can be generalized to surfaces over uniform triangular control nets. The scheme is characterized by the following refinement equations

$$(4.7)\quad \left| \begin{array}{ll} P^{k+1}_{2i,2j} & = \sum_{m,n\in M_0} a_{m,n}\, P^k_{i+m,j+n} \\ P^{k+1}_{2i+1,2j} & = \sum_{m,n\in M_0} b_{m,n}\, P^k_{i+m,j+n} \\ P^{k+1}_{2i,2j+1} & = \sum_{m,n\in M_0} c_{m,n}\, P^k_{i+m,j+n} \\ P^{k+1}_{2i+1,2j+1} & = \sum_{m,n\in M_0} d_{m,n}\, P^k_{i+m,j+n} \end{array} \right.$$

where, the coefficients $\{a_{m,n}, b_{m,n}, c_{m,n}, d_{m,n}\}$ are constants and non-zeros in the set M_0. This set is a fixed, finite integer set (support set) describing the *local dependent* property of the algorithm.

It is obvious that both the 10-point interpolatory scheme, which will be studied in Chapter 6, and the tensor product type algorithms belong to this class of subdivision algorithms.

Using the *generating polynomial* technique, *Cavaretta, Dahmen, Micchelli, Dyn* et al [28,29,54,...] analysed this scheme and some necessary and sufficient conditions for generating smooth surfaces are studied. While our method, the *Matrix Analysis,* which is based upon the *Differences* and *Cross-Differences of Directional Divided Differences,* will be presented in Chapter 6.

Remark: The uniform algorithms can also be used to generate smooth surfaces over arbitrary triangular networks if one can construct some special local algorithms to cope with the *extraordinary points.*

4.4. Other Recursive Subdivision Algorithms for Surfaces

Just as in the case of curves, there are many recursive subdivision algorithms for surfaces. Some of them cannot be classified into the above two categories, for example, the non-linear or non-uniform algorithms [53,122] and the subdivision algorithms for polynomial surfaces [4,7], which will now be described briefly below.

4.4.1. The de Casteljau Algorithm for Bernstein-Bezier Polynomial Surfaces

Like most recursive subdivision algorithms for curve, the *de Casteljau* algorithm can be generalized to calculate the *Bernstein-Bezier* surfaces (*BB* surfaces) [19]. As in the curve case, the algorithm is based on the recurrence relation of the bi-variate *BB* function basis.

The algorithm for a cubic polynomial patch can be described as follows.

Firstly, the *BB* basis functions of degree n, $n \geq 0$, $\{B^n_{i,j,k}\}$, $i+j+k = n$ and i, j, $k \geq 0$, are defined by

$$(4.8) \quad B^n_{i,j,k} := \frac{n!}{i!j!k!} s^i r^j t^k ,$$

where, (s, r, t) is the *barycentric coordinates*, which are omitted in the basis function expressions. Similar to the univariate case, it can be shown the following recurrence relation of these basis functions

(4.9) $B^{n+1}_{i,j,k} = sB^n_{i+1,j,k} + rB^n_{i,j+1,k} + tB^n_{i,j,k+1}$, for $i+j+k = n+1$ and $s+r+t=1$.

The *BB* surface over a triangle is defined by (in terms of *barycentric coordinates*):

(4.10) $P^n(s,r,t) = \sum_{i+j+k=n} P_{i,j,k} B^n_{i,j,k}$

where, $\{P_{i,j,k}\}_{i+j+k=n}$ are the so-called *Bezier* points. By applying (4.10) repeatedly, the following subdivision process can be obtained

(4.11)
$$\begin{aligned} P(s,r,t) &= \sum_{i+j+k=n} P_{i,j,k}\, B^n_{i,j,k} \\ &= \sum_{i+j+k=n-1} P^1_{i,j,k}\, B^{n-1}_{i,j,k} \\ &= \sum_{i+j+k=n-2} P^2_{i,j,k}\, B^{n-2}_{i,j,k} \\ &= \dots \\ &= \sum_{i+j+k=0} P^n_{i,j,k}\, B^0_{i,j,k} \\ &= P^n_{0,0,0} \end{aligned}$$

where, the *Bezier* points $\{P^m_{i,j,k}\}$ are determined recursively by

(4.12) $P^{m+1}_{i,j,k} := s\,P^m_{i+1,j,k} + rP^m_{i,j+1,k} + t\,P^m_{i,j,k+1}$, for $0 \le m \le n$, $j+j+k = m+1$

and

(4.13) $P^0_{i,j,k} := P_{i,j,k}$, for all i, j and k.

The property of the subdivision algorithm for *BB* surfaces can be concluded by the following theorem:

<u>Theorem</u> 4.3. Using the above recursive algorithm, the *BB* surface $P(r,s,t)$ can be split into three sub-patches, P^0, P^1 and P^2 whereby the *Bezier* points (expressed in the *BB* form) are given by

(4.14) $\{P^m_{i,j,0},\ i+j = n-m,\ m = 0, 1, ..., n\}$,

(4.15) $\{ P^m_{0,j,k},\ j+k = n-m,\ m = 0, 1, ..., n\}$,

(4.16) $\{P^m_{i,0,k},\ \ i+k = n-m,\ m = 0, 1, ..., n\}$.

respectively.

Figure 4.3 shows the geometric structure of the algorithm for a cubic surface.

More about the algorithm is discussed in [19,61,113].

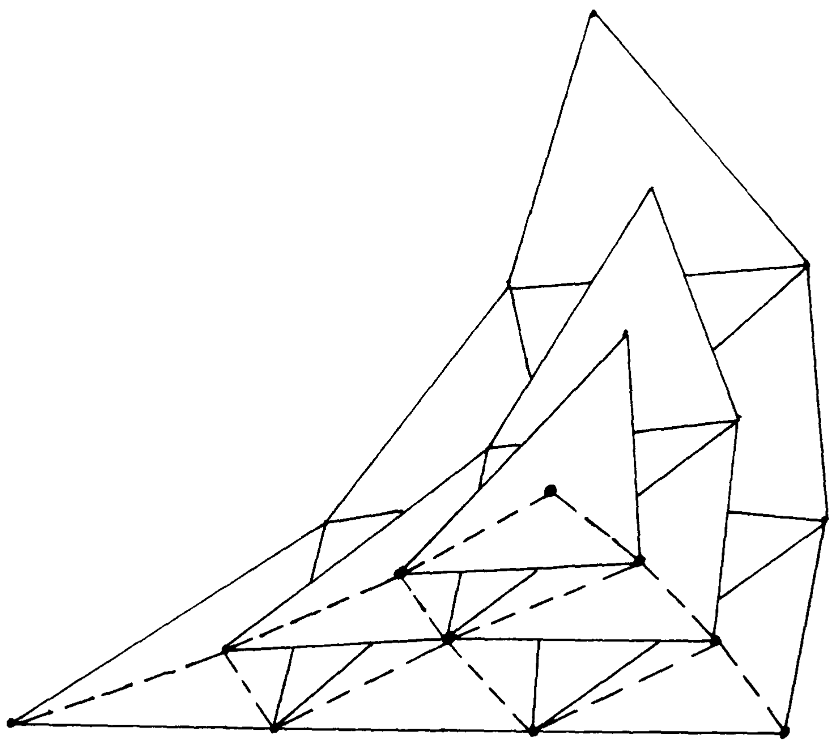


Figure 4.3.

4.4.2. Geometry Based Algorithms

This class of algorithms is introduced by *N. Dyn*, *D. Levin* and *D. Liu* [53] to construct convexity preserving interpolatory recursive subdivision algorithms for surfaces. The scheme is neither uniform nor linear. It is a local, geometry dependent algorithm.

The idea of the scheme is to refine the control nets under some convexity (shape) preserving constraints. The scheme is a generalization of the chape preserving subdivision algorithm used to produce interpolatory curves [47,48].

The main difference is that in the surface case the constructions and the constraints are much more complicated. For details, see [53].

4.4.3. The Cube-spline Algorithm

The algorithm for the computation of multivariate *Cube-splines* (*Box-splines*) [11,13,16,39,85-91] belongs to a very special class of recursive subdivision algorithms. This algorithm can be regarded as a generalization of the uniform *de Boor* algorithm, the *Lane-Riesenfeld* algorithm or the *de Casteljau* algorithm. The main feature of them is that they are *moving line averaging processes*.

The algorithm has been described in Chapter 2. For more details the reader is referred to [11,16,39,85-91].

CHAPTER FIVE

A SUBDIVISION ALGORITHM FOR UNIFORM BI-QUARTIC B-SPLINE SURFACES OVER ARBITRARY NETWORKS

A schematic analysis of the subdivision algorithm for uniform bi-quartic B-spline surfaces over arbitrary networks is presented in this Chapter. Our main result is the spectrum analysis of the subdivision matrix and the *Normal Curvature* property analysis of the limit surfaces at an extraordinary point. The *Block-Circulant Matrix* method is used to simplify our analysis.

5.1. Formulation of the Algorithm

Since any uniform B-spline curve can be produced by a recursive subdivision algorithm as discussed in the previous Chapters, the ***RSA*** for uniform bi-quartic B-spline surfaces can easily be derived. The scheme will be described in detail later in this section. From the construction we know that the algorithm produces a uniform bi-quartic B-spline surface if the initial data is uniform. However, since non-uniform topology often arises in practice. we will adapt this algorithm to non-uniform data.

5.1.1. The Ideas of the Analysis

In the papers by *Doo/Sabin* and *Ball/Storry* [1-3,44-46], the *Doo-Sabin's* and the *Catmull-Clark's* algorithms are thoroughly analysed. Their analyses are, in fact, an extraordinary point analysis. It will be shown later that the properties of the subdivision matrix A at the extraordinary point determine the behaviour of the limit surfaces at the extraordinary point.

Their analyses come from the following observations. Without loss of generality, we suppose that the initial control polyhedron has only one *N-extraordinary poin*t (or *facet*, where $N \neq 4$), which is surrounded by locally uniform data. Since the scheme does not introduce any more *E-point*s, at any subdivision stage the limit surface is well defined everywhere except an *N-sided hole* around the *E-point*. By repeating the subdivision process, the *N-sided hole* will be covered by smaller and smaller 4-sided, well defined polynomial patches apart from the central point and will finally converge to the *E-point* provided that the scheme produces a continuous surface (this is always assumed). It should be noted that for C^1 convergence, the tangent plane of the limit surface at the *E-point* is defined as the limit of the tangent planes of the well-defined surface patches defined by the *near-the-hole* points, see Figure 5.1 (it is assumed that the limit exists). This is reasonable since the tangent planes vary continuously on the well defined B-spline surface patches.

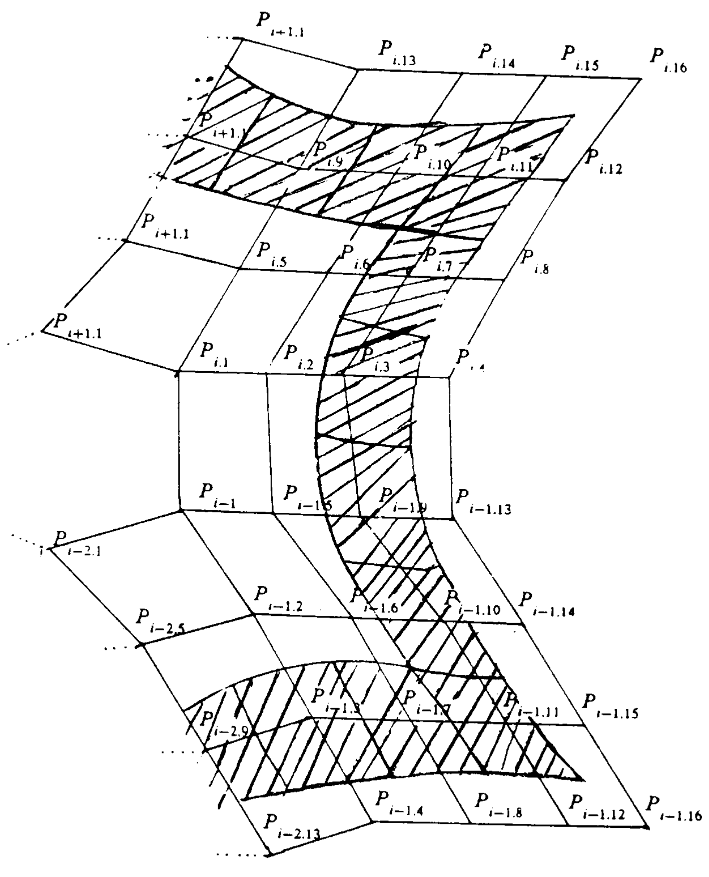


Figure 5.1.

The following result is obtained [3,46,116]:

<u>Proposition</u> 5.1. The algorithm produces a continuous surface if the hole converges to a point. Furthermore, the surface is C^1 if the well defined surface patches are C^1 and the tangent planes of them at the *near-the-hole* points converge to a plane, which is the tangent plane of the limit surface at the *E-point*.

For C^2 convergence, the analysis is more difficult and will be studied in section 5.3.

5.1.2. The Eigen-range of the Algorithm

In this subsection, we shall concentrate on the study of the N-sided hole problem. However, before we proceed our studies, two natural questions are raised: on which points the behaviour of the N-sided hole depend and what the control points are that determine the uniform tensor-product B-spline surface patches just around the hole.

In order to answer these questions, the *Eigen-range* of the algorithm should be introduced.

Definition: ***The eigen-range of the algorithm is defined as the number of control points and its topology near the N-sided hole that have effects on the behaviour of the limit surface at the E-point.***

For example, the *Eigen-range* of the *Doo-Sabin's* algorithm is the $4N$ control points around the hole (only two rings); the *Eigen-range* of the *Catmull-Clark's* algorithm is the $(6N+1)$ points around the hole (three rings) and the *Eigen-range* of the bi-quartic uniform B-spline algorithm is the $16N$ points around the hole (four rings). Generally, the *Eigen-range* of a scheme is m^2N (for even m) or $m(m-1)N+1$ (for odd m), where m is the degree of the B-spline patches produced by the algorithm over uniform data, $m > 1$). The algorithm is called an *even* (*odd*) algorithm if m is *even* (*odd*). Figure 5.2

shows the *Eigen-ranges* of the first three B-spline algorithms.

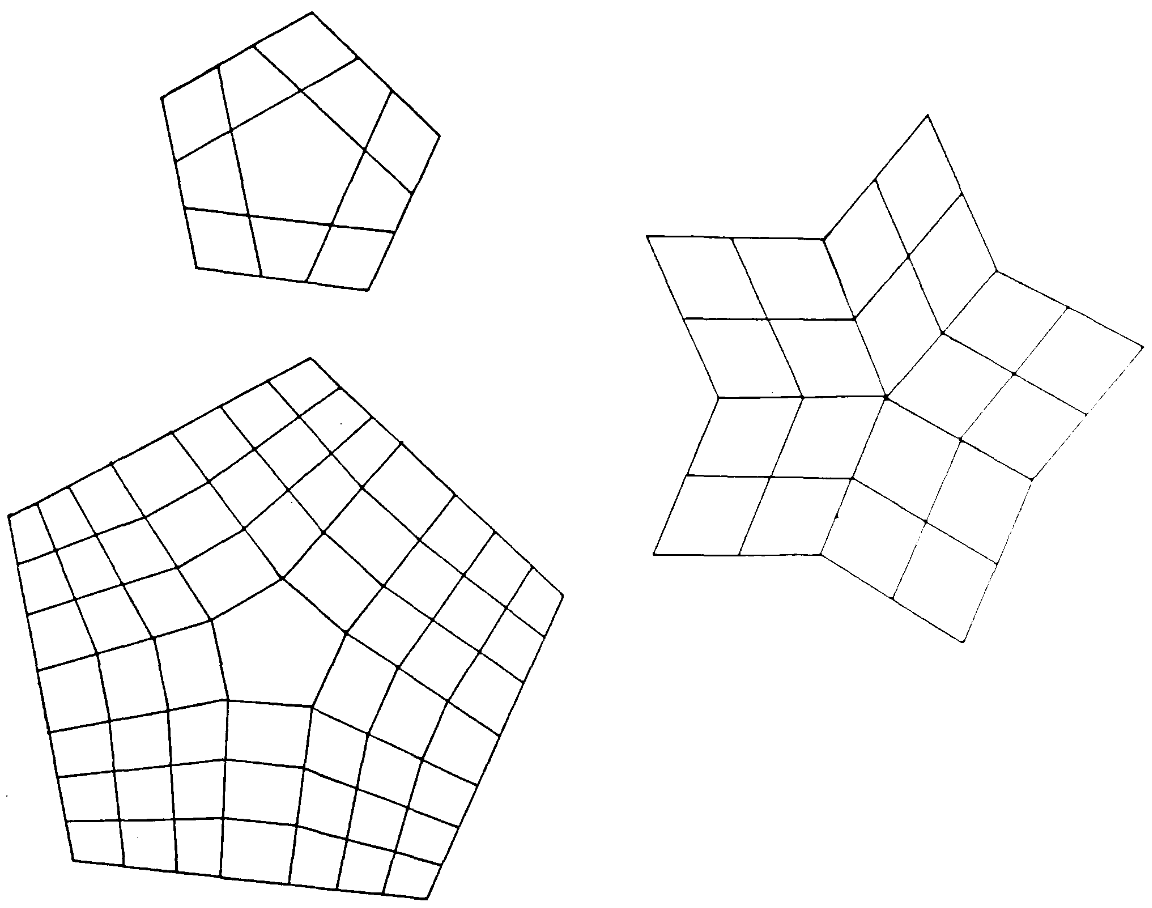


Figure 5.2.

It will be shown in the following subsections that the properties (the position and the partial derivatives) of the limit surface at the *E-point* depend only on the *Eigen-range* of the scheme.

5.1.3. The Local Subdivision Matrix A

Since the algorithm is a local, linear (in fact, convex combination) process, it can easily be shown that its *Eigen-range* at level $k+1$ can be obtained by a linear transformation (in fact, affine transformation) of the

Eigen-range at level k, where $k \geq 0$ is the iteration count. Using the *block circulant* ordering technique, this relation can be written in the form

(5.1) $\quad P^{k+1} = A P^k \quad$ for all $k \geq 0$,

where, A is the *Local Subdivision Matrix* and P^k and P^{k+1} are the *Eigen-range* vectors at level k and k+1 respectively and P^k is defined by

(5.2)
$$P^k := \begin{cases} (P^k_{0,1}, P^k_{0,2}, \ldots, P^k_{0,m}, P^k_{1,1}, \ldots, P^k_{N-1,m})^t \text{ (for even } m) \\ \\ (V^k, P^k_{0,1}, P^k_{0,2}, \ldots, P^k_{0,m-1}, P^k_{1,1}, \ldots, P^k_{N-1,m-1})^t \text{ (for odd } m). \end{cases}$$

and P^{k+1} is defined similarly. Figure 5.3 shows the details of the *Block-Circulant ordering technique* for the *Eigen-ranges.*

When m is even, the iteration matrix A is a ***Block-Circulant Matrix*** (*B–circ* matrix for short) of the form

(5.3)
$$A := B\text{–}circ(A_0, A_1, \ldots, A_{n-1})$$
$$:= \begin{vmatrix} A_0 & A_1 & \cdots & A_{n-1} \\ A_{n-1} & A_0 & \cdots & A_{n-2} \\ \cdots & \cdots & \cdots & \cdots \\ A_1 & A_2 & \cdots & A_0 \end{vmatrix},$$

where, A_i, $i = 0, \ldots, N-1$ is a square matrix of order m^2.

When m is odd, then the subdivision matrix A is in the form

(5.4)
$$A := \begin{vmatrix} a & V_1^t \\ V_2 & A' \end{vmatrix},$$

where, a is a (positive) number and A' is a *B-circ* matrix of order $m(m-1)N$. V_1 and V_2 are some specific (positive) vectors.

As an example, the subdivision matrix A for the cases $m = 4$ is constructed in subsection 5.1.5

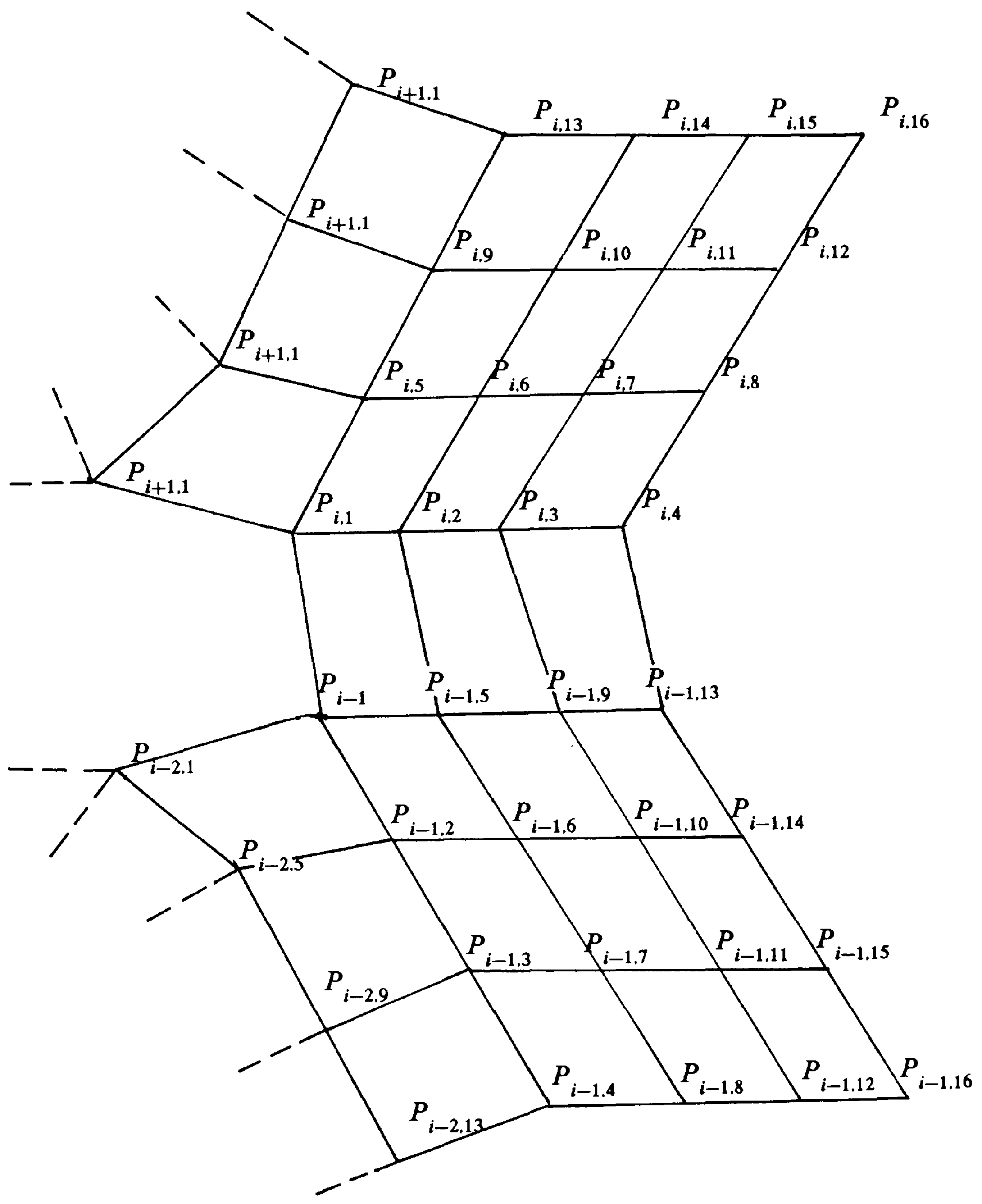


Figure 5.3.

5.1.4. The Role of the Eigen-range

In this subsection, we shall prove that the local properties of the limit surface at the *E-point* are completely determined by the *Eigen-range* of the algorithm at any level.

By definition, the tensor-product B-spline surface is given by

$$(5.5) \qquad P(u, v) := U^t M^t P M V,$$

where

$$(5.6) \quad \left| \begin{array}{l} U := (u^m, u^{m-1}, \ldots, u, 1)^t \\ V := (v^m, v^{m-1}, \ldots, v, 1)^t \\ M := \begin{vmatrix} a & \alpha^t \\ M' & 0 \end{vmatrix}, \text{ the subdivision matrix for B-spline curves,} \\ P := \{P_{i,j}, i,j = 0,\ldots,m\}, \text{ control points.} \end{array} \right.$$

Direct evaluation from (5.6) gives the following result:

Theorem 5.2. For all $0 \leq i, j < m$,

$$(5.7) \qquad \{ \left. \frac{\partial^{i+j} P(u,v)}{\partial u^i \, \partial v^j} \right|_{u=0} \} \text{ are determined by } \{ P_{l,n} |_{l,n < m} \}$$

On applying this theorem to the control points of the B-spline patches around the N-sided hole, we can conclude that all the values and the partial derivatives of the patches given by (5.7) which converge to the *E-point* are determined by the corresponding *Eigen-ranges.*

Since these well defined patches are connected very smoothly (C^3 continuous), the properties of the limit surface at the *E-point* depend on the properties of the patches near the hole. Consequently, we can conclude that the properties of the limit surface at the *E-point* depend on any of the *Eigen-ranges* (the *Eigen-range* at a lower level determines the *Eigen-range* at a higher level).

5.1.5. The Subdivision Scheme for Uniform Bi-quartic B-spline Surfaces over arbitrary networks

As an example, we give a brief description of the subdivision scheme for uniform bi-quartic B-spline patches. For uniform data, the scheme is characterized by the linear operator LB_4: $R^{3,3} \longrightarrow R$ defined by (see Figure 5.4):

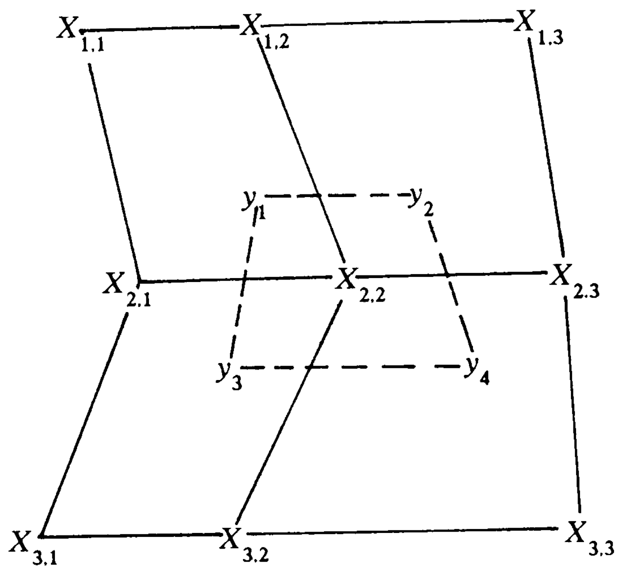


Figure 5.4.

(5.8) $$y_1 := LB_4(X) := \{ 25X_{1.1} + 50(X_{1.2} + X_{2.1}\} + 100\, X_{2.2} + 5\,(X_{1.3} + X_{3.1}\} + 10\,(X_{2.3} + X_{3.2}\} + X_{3.3}\}/256.$$

By symmetry, other refined points, such as y_2, y_3 and y_4 can also be computed.

For non-uniform data, the algorithm is like this. For any point away from the N-sided hole, the operator LB_4 is applied. For the *two layer* points around the hole, N *pseudo-points*, $\{Q_i\}$ are introduced so that the operator can also be applied. Figure 5.5 shows the construction of the algorithm over the hole.

These N $(N \neq 3)$ *pseudo*-points $\{Q_i\}$ are defined by the following *symmetric* formula:

(5.9) $$Q_i := a_0 P_{i.1} + \sum_{j=1}^{[(N/2]} a_j\,(P_{i+j.1} + P_{N+i-j.1}),\ i = 0, 1, ..., N-1$$

where, $\{a_i\}$ are some weightings satisfying

(5.10) $$a_0 + 2\sum_{i=1}^{[(N/2]} a_i = 1.$$

Thus, the refined points over the *hole* can then be defined: for $i = 0, 1, ..., N-1$,

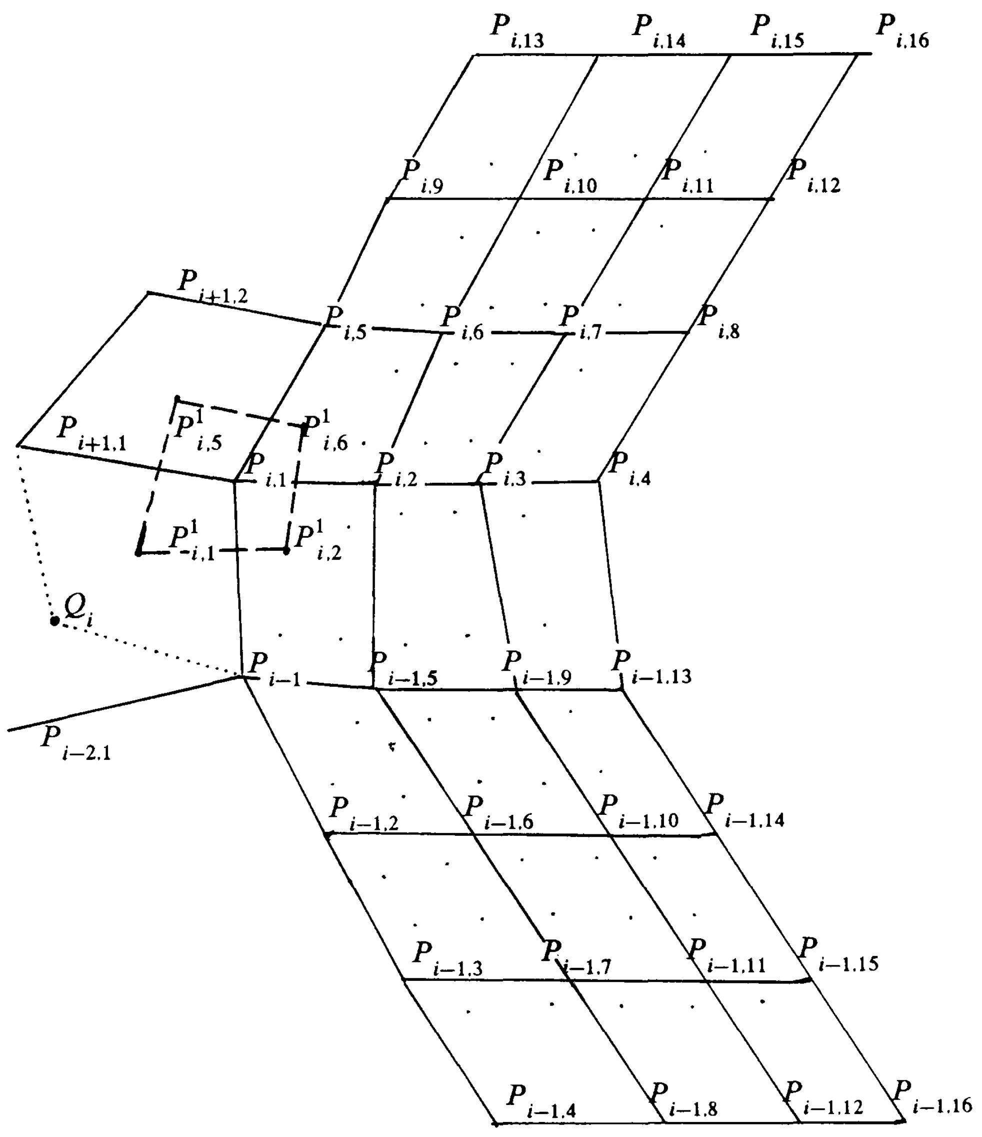


Figure 5.5.

$$(5.11)\quad \left|\begin{aligned}
P^1_{i,1} &= LB_4(Q_i, P_{i-1,1}, P_{i-1,5}, P_{i+1,1}, P_{i,1}, P_{i,2}, P_{i+1,2}, P_{i,5}, P_{i,6})\\
P^1_{i,2} &= LB_4(P_{i-1,5}, P_{i-1,1}, Q_i, P_{i,2}, P_{i,1}, P_{i+1,1}, P_{i,6}, P_{i,5}, P_{i+1,2})\\
P^1_{i,5} &= LB_4(P_{i+1,2}, P_{i+1,1}, Q_i, P_{i,5}, P_{i,1}, P_{i-1,1}, P_{i,6}, P_{i,2}, P_{i-1,5})\\
P^1_{i,6} &= LB_4(P_{i,6}, P_{i,2}, P_{i-1,5}, P_{i,5}, P_{i,1}, P_{i-1,1}, P_{i+1,2}, P_{i+1,1}, Q_i).
\end{aligned}\right.$$

Other refined points are determined by the bi-quartic B-spline subdivision process. This is also shown in Figure 5.5.

For simplicity, in our analysis, we assume that $a_i = 0$ for $i \geq 3$.

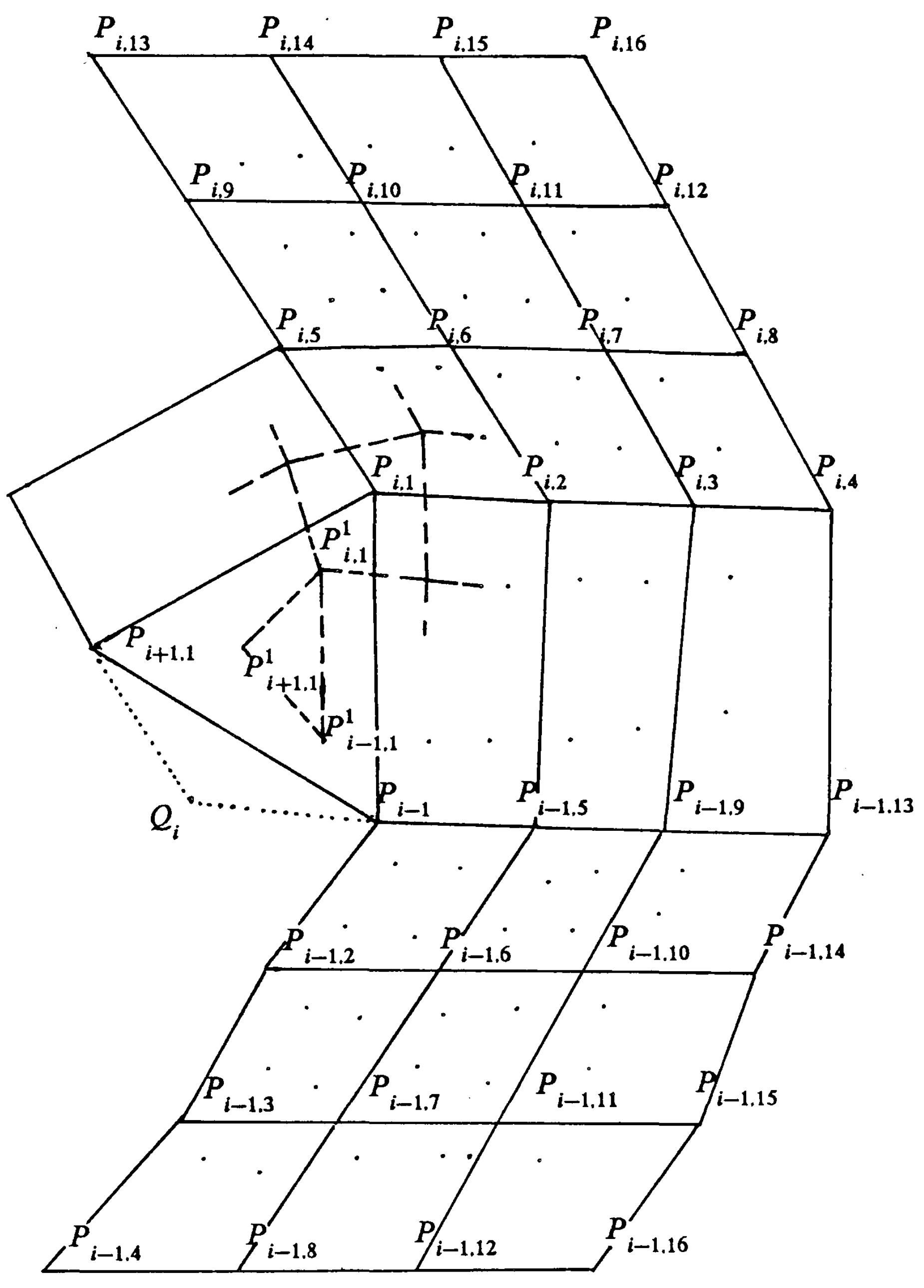


Figure 5.6.

In the case of $N = 3$, these *pseudo*-points can be introduced similarly. For example, Q_i can be defined as a symmetric affine combination of $P_{i,1}$, $P_{i+1,1}$, $P_{i-1,1}$, $P_{i,5}$, $P_{i+1,5}$, $P_{i-1,5}$, $P_{i,2}$, $P_{i+1,2}$ and $P_{i-1,2}$. One simple choice is

(5.12) $$Q_i := a_0 P_{i,1} + a_1 (P_{i+1,1} + P_{i-1,1}) + a_2 (P_{i+1,5} + P_{i-1,2})$$

where

(5.13) $$a_0 + 2a_1 + 2a_2 = 1.$$

The refined points near the hole $\{P^1_{i,j}|_{j=1,2,5,6}\}_{i=0}^{N-1}$ are given by (5.11). This process is demonstrated in Figure 5.6 where, the labels of the new control points are omitted.

From the above construction of the algorithm over the hole, we can see clearly that the process is linear and the subdivision matrix A, as introduced in (5.1) is a ***Block-Circulant Matrix.***

5.2. Convergence Analyses

In this section, we study the conditions under which the algorithm produces continuous and differentiable surfaces. The result obtained here is a generalization of the results reported in [1-3,45,116], whereby the quadratic and cubic B-spline algorithms are studied respectively.

5.2.1. The Extraordinary Point Analysis due to Ball and Storry

In a series of papers by *Ball/Storry* [1-3,116], they analysed the *Catmull-Clark's* algorithm in detail. Their analysis, much like the *Doo-Sabin's* analysis for the quadratic algorithm, is mainly composed of the extraordinary point analysis. Their idea for the high order continuity analysis of the limit surface at an extraordinary point is as follows.

For the C^1 convergence, they assert that the surface is C^1 at the *E-point* if the tangent planes of the well-defined B-spline patches near the *E-point* converge to a plane. This plane is then defined as the tangent plane of the limit surface at the *E-point*. Hence, their C^1 analysis is based on the study of the tangent plane series of the B-spline patches near the *E-point*. This investigation involves further studies on the eigen-properties of the local subdivision matrix. Similar techniques will be used in our analysis.

Although they have proved·that the *Catmull-Clark's* algorithm could not, in general, produce C^2 continuous surfaces over arbitrary topology, they still studied the curvature properties of the limit surface at the *E-point*. In their analysis it is implied that the limit surface may be C^2 if the following conditions are satisfied: (i) any aligned face or edge loci on the surface (passing through the *E-point*), which has tangent continuity at the *E-point*, has curvature continuous property and (ii) the normal curvatures of these loci at the *E-point* satisfy the *Gaussian* normal curvature condition (5.15), which will be studied in detail later. The second condition is very important since condition (i) is not sufficient to guarantee C^2 continuity. This can be shown by a simple counter-example. However, any one of the conditions is a necessary condition for the surface to be C^2. Hence, they only used condition (i) to prove the curvature discontinuous property of the limit surfaces.

In order to study the tangent planes and some appropriate loci on the surface, the eigenvalues and their corresponding eigenvectors of the local subdivision matrix should be analysed. In fact, it can be shown that given the initial data, the tangent planes and the curvatures of the loci can be expressed explicitly in terms of these eigenvalues and eigenvectors. To this end, they used both direct evaluation and the *Fourier Transformation* technique to find these values.

The mathematics behind this analysis is a combination of *differential geometry* and *mathematical analysis*. The whole theory lies on the technique of pointwise analysis, which is based on the local structure of surfaces.

Since our algorithm has a similar property, that is, the limit surface is curvature continuous (C^3 to be more precise) everywhere except at the *E-points*, the *Ball-Storry's* ideas for the smoothness analysis will be used to analyse the convergent property of the scheme at these *E-points*. Moreover, one major difference from the *Ball-Storry's* analysis is that their ideas will be developed to study the curvature property at the *E-point*.

Another difference of our analysis from theirs is that we use

Block-Circulant-Matrix technique (this is equivalent to ***(Block) Fourier Transformation Technique***) to analyse the subdivision matrix instead of the direct evaluation method. This difference in techniques is due to the fact that the subdivision matrix in our case is more complicated than that in *Ball-Storry's* analysis, and thereby making the process of direct evaluation of the eigenvalues and eigenvectors very dificult (but not impossible).

5.2.2. Continuity at the Extraordinary-point

As we have already known that the algorithm produces in general a C^3 surface over uniform data, our attention is therefore restricted to the analysis at the *E-points*. In order to prove the C^1 property of the surface, it is imperative that the C^0 property should be proved first. To this end, the *Adapted Parametrization* technique, that is, the parameter values satisfy the subdivision algorithm, is used to prove the C^0 continuity of the surface at the *E-point.*

Since the limit surface is smooth at any point except the *E-point*, the surface must have the same differentiability with respect to the *Adapted Parametrization* as with the diadic parametrization provided that the initial *(adapted) parametrization values* are chosen appropriately in the parameter plane. This is because the limit surface is locally a uniform bi-quartic B-spline patch except the *E-point.* In fact, a general result, as in the curve case, can be proved that the limit surface has at least the same differentiability with respect to the *adapted parametrization* as with the diadic parametrization. That is, *adapted parametrization* could be a better parametrization. Hence, we use the *adapted parametrization* to analyse the C^0 convergence of the algorithm at the *E-point*.

5.2.3. Spectrum Analysis

By choosing proper weightings (that is, the local shape control parameters $\{a_i\}$), the following properties of the subdivision matrix A can be obtained. It should be emphasized that these properties play a very important role in the convergence and smoothiness analysis of the limit surfaces.

Let $\{\lambda_i\}$, $|\lambda_i| \geq |\lambda_{i+1}|$, be the eigenvalues of A and $\{v_i\}$ be the corresponding (generalized) eigenvectors. Then we define the following eigen-properties.

$\boldsymbol{B}_0$. $\lambda_1 = 1$, $v_1 = (1, 1, ..., 1)^t$, $|\lambda_i| < 1$, for all $i > 1$.

$\boldsymbol{B}_1$. $\lambda_2 = \lambda_3 > 0$, $|\lambda_i| < \lambda_2$, for all $i > 3$

$\quad\ \ \textit{dim span}\{v_2, v_3\} = 2$.

$\boldsymbol{B}_2$. $\lambda_4 = \lambda_5 \geq \lambda_6 > 0$, $\lambda_4 = \lambda_2^2$

$\quad\ \ \lambda_5 > |\lambda_i|$ for all $i \geq 7$

$\quad\ \ \textit{dim span}\{v_4, v_5\} = 2$.

B_{20}. $\left|\begin{array}{l} \lambda_4 = \lambda_5 \geq \lambda_6 > 0,\ \lambda_4 < \lambda_2^2 \\ \\ \lambda_6 > |\lambda_i| \text{ for all } i > 6. \end{array}\right.$

Remark. It is required that v_i, $i = 1, 2, ..., 5, 6$, are eigenvectors of A.

In the cases of quadratic and cubic B-spline algorithms, that is, when $m = 2$ and $m = 3$, it is shown explicitly in [45,116] that properties B_0 and B_1 can be satisfied. However, in the case of cubic B-spline algorithm, property B_2 can not be obtained though property B_{20} be achieved. When $m = 4$, it can be shown that all the above properties can be obtained by an appropriate choice of the shape parameters $\{a_i\}$.

5.2.4. C^0 Convergence

Using the *adapted parametrization,* we can obtain the following result.

Theorem 5.3. If the local subdivision matrix A has property B_0, then, with respect to the *adapted parametrization,* the limit surface is uniformly continuous.

Proof. The condition B_0 guarantees that the *eigen-range* sequence of the algorithm converges uniformly to a point, which means that the limit surface is continuous at the *E-point.* It is obvious that the surface is uniformly continuous, respect to the *adapted parametrization,* at regular points. This completes the proof.

Remark. This *Theorem* can also be proved by using piecewise diadic parametrization method.

5.2.5. C^1 Convergence

For C^1 convergence, we need to prove that the tangent planes of the B-spline patches around the N-sided hole converges uniformly to a plane, the tangent plane of the limit surface at the *E-point*.

Theorem 5.4. If the subdivision matrix A has the properties B_0 and B_1, then the limit surface is C^1 at the *E-point*.

Proof. If properties B_0 and B_1 hold, then it can be shown, in the same way as in [2,45,116], that all the tangent planes of the well defined B-spline patches around the hole at level k have the form

$$(5.14) \qquad \Pi^k = span\{\alpha, \beta\} + R(k),$$

where, $\alpha, \beta \in R^3$ are constants depending on the initial data and the shape parameters and $R(k) = O(\lambda_4/\lambda_2)^k$. For general data, $<\alpha, \beta> \neq 0$. Property B_1 quarantees that $R(k)$ goes to zero uniformly when k goes to infinity. That is, all these tangent planes converge to the plane spanned by α and β. This completes the proof.

5.3. Normal Curvature Analysis

In this section, we investigate the curvature properties of the surface at the *E-point*. The main result is the *Normal Curvature* (*N-curvature*) analysis about the surface at the *E-point*. This result is also valid for higher (even) order schemes.

5.3.1. Formulation of the Curvature Continuous Problem

Since the B-spline patches around the N-sided hole are well defined, some simple calculations show that the *N-curvatures* of these patches can be represented by their corresponding control points. By definition, these *N-curvatures* can also be expressed in terms of the points in the *Eigen-range*. As a consequence, we have the following lemma:

<u>Lemma</u> 5.5. The *normal curvatures* of the well-defined patches adjacent to the N-sided hole at level $k+1$ can be represented by the *Eigen-range* at level k.

To analyse the curvature continuity of the limit surface at the *E-point*, we assume that the surface is at least C^1 at the *E-point* (tangent plane continuity).

We now define the *Normal-curvatures* of the limit surface at the *E-point* to be the corresponding limits (suppose the limits exist and are finite) of the *N-curvatures* of the well defined B-spline patches near the N-sided hole. It is obvious that the definition is compatible with the ordinary definition of C^2 surfaces if the limit surface is C^2 at the *E-point.*

From differential geometry, we know that the normal curvature K_n of a C^2 surface at point Q must satisfy the *Gaussian Curvature Condition* (*G-condition*):

(5.15) $$K_n = cos^2(\alpha)\, K_{max} + sin^2(\alpha)\, K_{min}$$

where, K_{max} and K_{min} are respectively the maximum and minimum *N-curvatures* of the surface at Q and α is the angle $<T_n, T_{max}>$. Here, T_n, T_{min} and T_{max} are the unit tangent vectors of the corresponding *N-curvatures*.

It should be pointed out that, in our isolated *E-point* case, the C^2 conditions and the *G-condition* at the *E-point* are equivalent. Furthermore, since the surface is a sufficiently smooth parametric surface, it can be regarded as a function (locally) at the *E-point* [79]. Hence, the *Gaussian* condition is a necessary and sufficient condition for the surface to be a C^2 surface. Consequently, if we can prove that the *Gaussian* condition is satisfied at the *E-point,* we can conclude that the limit surface is curvature continuous at that point.

5.3.2. C^2 Conditions of a Parametric Surface

In this subsection, some conditions for a parametric surface, $G(u,v)$, to be C^2 continuous at a point Q on the surface are presented. To this end, we first introduce the definition of the *normal curvature* of a regular smooth curve $r(t)$ on $G(u,v)$ passing through Q.

Let the unit normal direction of the surface at Q be denoted by N, and the derivative of $r(t)$ at Q with respect to t be denoted by r'. Then, from [79,116], the curvature of $r(t)$ at Q is defined as

$$(5.16) \quad K_Q := (<r', r'>r'' - <r'', r'>r')/<r', r'>^2$$

Thus the *N-curvature* of $r(t)$ at Q is given by

$$(5.17) \quad K_N(Q) := <K_Q, N> = <r'', N>/<r', r'>.$$

It can be shown that $K_N(Q)$ depends only on the surface $G(u,v)$ and the unit tangent vector $r'(t)/||r'(t)||$ at Q ($r'(t)/||r'(t)||$ and $K_N(Q)$ are *invariants*). Thus, $K_N(Q)$ is defined as the *N-curvature* of the surface $G(u,v)$ at Q along the direction $r'(t)/||r'(t)||$.

The following proposition describes the *G-condition* for a parametric surface.

<u>Proposition</u> 5.6 [79,116]. Suppose K, K_j (j=1, 2, 3) are four normal curvatures of a C^2 parametric surface $G(u,v)$ at Q along tangent direction T, T_j

(j=1, 2, 3) respectively, where T, T_j are the corresponding tangent vectors of some C^2 curves passing through Q on the surface. If $\{T_j\}$ satisfy

(5.18) $\quad T_j \times T_i \neq 0, \quad i, j = 1, 2, 3, \quad i \neq j,$

then we have the following *Guassian Normal Curvature Condition*

(5.19) $$K = \langle T,T\rangle^{-1} \sum_{j=1}^{3} \frac{\langle T_j,T_j\rangle\langle (T_{j-1} \times T), (T \times T_{j-1})\rangle}{\langle (T_{j-1} \times T_j), (T_j \times T_{j+1})\rangle} K_j$$

where, $\langle X, Y\rangle$ is the scalar product of X and Y, and $X \times Y$ is the cross product of X and Y. The index j is cyclic within the range (1, 2, 3).

Remark. If we define $T_4 := T$, $K_4 := K$ in *Proposition* 5.6 and suppose that

(5.20) $\quad$ angle $\langle T_i, T_{i+1}\rangle = \alpha =$ constant, $i = 1, 2, 3,$

then the *G-condition* (5.19) becomes

(5.21) $\quad K_4 + K_3 = K_1 + K_2 + 4\cos^2(\alpha)\,(K_3 - K_2).$

This condition is very useful to prove the curvature properties of the surface. In particular, if we solve (5.21) as a difference equation (of order three), then we can obtain the the following real basic solutions:

(5.22) $\quad 1,\ \cos(2i\alpha)$ and $\sin(2i\alpha)$.

Thus, the general solution of (5.21) is given by

(5.23) $\quad K_i = c_1 + c_2\cos(2i\alpha) + c_3\sin(2i\alpha) = c_1 + c_2\cos(2i\alpha + c_3),$

(5.23) $K_i = c_1 + c_2 cos(2i\alpha) + c_3 sin(2i\alpha) = c_1 + c_2 cos(2i\alpha + c_3)$,

where, c_1, c_2 and c_3 are some general constants.

Conditions (5.21) and (5.23) will be used later in our analysis.

5.3.3. Special Loci on the Surface Incident to the Extraordinary Point

In order to study the *N-curvature*s of the surface at the *E-point*, we need to introduce some special loci on the limit surface incident to the *E-point*. We then investigate the *N-curvature* properties of the surface on these loci and establish the curvature analysis about the limit surface at the *E-point*.

As in *Ball* and *Storry's* analysis, some special loci on the surface passing through the *E-point* can be introduced. For example, the edge loci and the diagonal loci on the well-defined B-spline patches around the hole can be defined. Other loci can also be studied in a similar way. Figure 5.7 shows these loci and their corresponding B-spline patches.

Mathematically, the i-th diagonal locus can be defined like this. At level k, those newly produced patches $\{B^k_{i,n}(u,v)|_{n=1,2,\dots,12}\}_{i=0,1,\dots,N-1}$, $0 \leq u, v \leq 1$, which are defined by *near-the-hole* control points, are well defined (Figure 5.7). Then, the i-th diagonal locus $D_i(\cdot)$ is defined by

(5.24) $D_i := \bigcup_k D^k_i$

where,

(5.24a) $\quad D_i^k \; := \; \{[B_{i,3}^k(t, t)\colon\ 0 \le t \le 1] \bigcup [B_{i,9}^k(t, t)\colon\ 0 \le t \le 1]\}.$

Other loci can be defined in the same way. For example, the i-th edge locus $E_i(\cdot)$ is given by

(5.25) $\quad E_i \; := \; \bigcup_k E_i^k$

where,

(5.25a) $\quad E_i^k \; := \; \{[B_{i,1}^k(0, t)\colon\ 0 \le t \le 1] \bigcup [B_{i,12}^k(0, t)\colon\ 0 \le t \le 1]\}.$

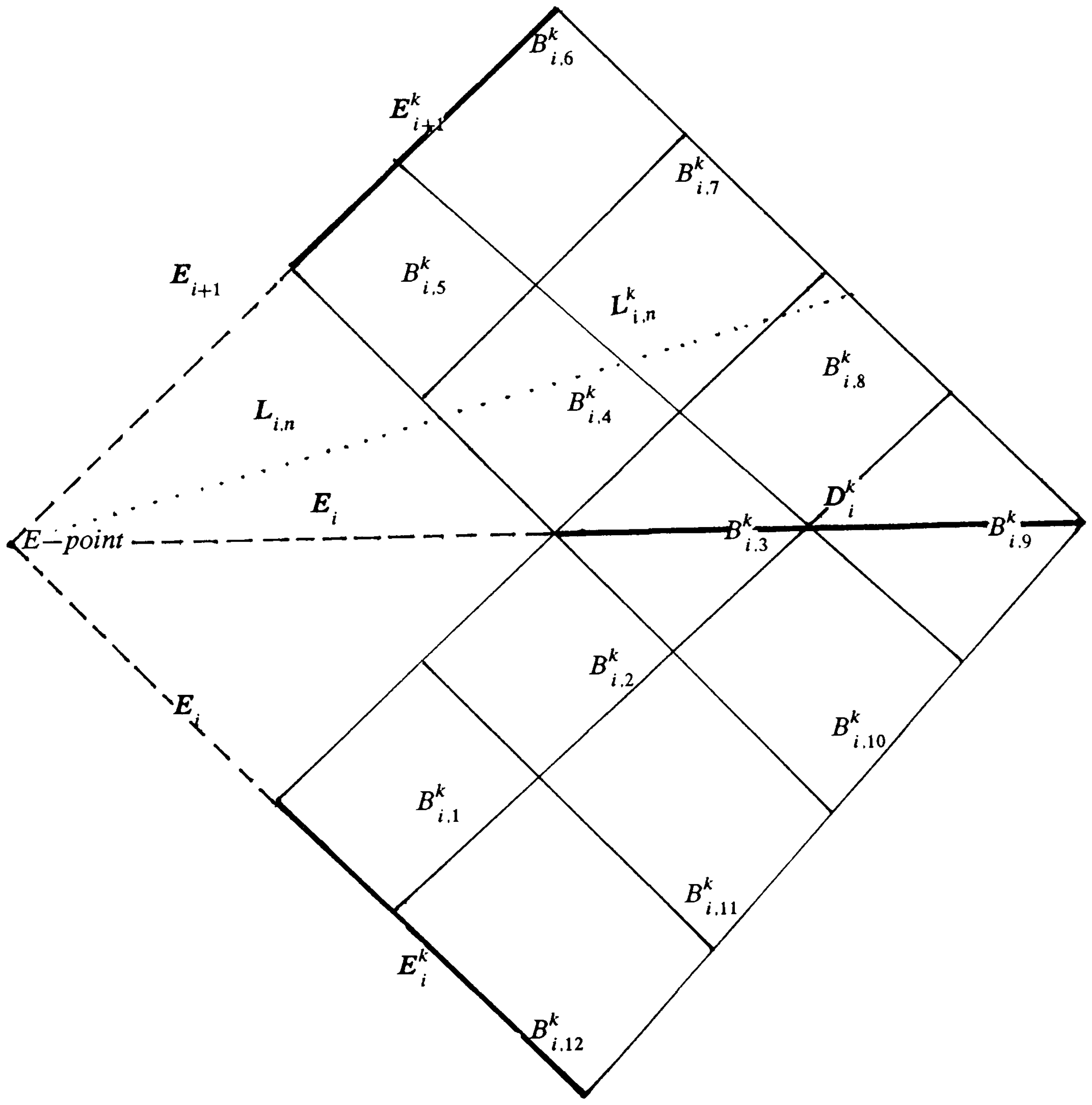


Figure 5.7.

The general loci, $L_{i,n}$, given by

(5.26) $\quad L_{i,n} := \bigcup_k L^k_{i,n}$

where,

(5.26*a*) $\quad L^k_{i,n} := \bigcup_j \, [B^k_{i,n_j}(u_{n_j}(t), v_{n_j}(t)) \colon 0 \le t \le 1].$

where, $u_{n_j}(t)$ and $v_{n_j}(t)$ are some special linear functions of t, can also be studied in the same way. Obviously, D_i and E_i are special cases of $L_{i,n}$.

Since the patches are uniform bi-quartic patches, we can obtain the following result:

__Lemma__ 5.7. Both the diagonal loci $\{D_i\}$ and the edge loci $\{E_i\}$ are C^3 curves. Furthermore $\{L_{i,n}\}$ are also C^3 if $\{u_{n_j}(t), v_{n_j}(t)\}$ and n_j are chosen appropriately.

In the next subsection, the *N-curvature*s of the diagonal loci and the surface at the *E-point* will be fully studied. It should be pointed out that the *N-curvature*s of these loci at the *E-point* are defined as the limits (suppose the limits exist, and they may be zeros) of the *N-curvature*s of the corresponding loci when k goes to infinity.

5.3.4. The N-curvatures of the Surface at the E-point

Since these loci defined above are C^3 except at the *E-point,* the N-curvature of these loci at this point can also be defined. For example, they are defined as the limits of the corresponding *N-curvature*s of $\{D_i^k\}$ as k goes to infinity. In general, the *N-curvature* of the surface at the *E-point* is defined as the limit of any C^2 continuous locus on the surface passing through the *E-point* which has a proper tangent direction at the *E-point*. It will be shown that the limits exist and are finite if the parameters $\{a_i\}$ are chosen appropriately.

In order to study these N-curvatures, more about the well defined patches around the N-sided hole should be investigated and some special techniques are also required to analyse them.

5.3.5. Results About Block-Circulant Matrices

The eigenvalues and their corresponding eigenvectors of the subdivision matrix play a very important role in the investigation of the curvature properties of the surface. In this subsection, some results about the subdivision matrix are presented. The proofs of these results can be found in or derived from [43].

Let A be a square *Block-Circulant-Matrix* of order mN:

(5.27) $A := B\text{–}circ(A_0, A_1, ..., A_{N-1})$

$$:= \left| \begin{matrix} A_0 & A_1 & A_2 & \cdots & A_{N-1} \\ A_{N-1} & A_0 & A_1 & \cdots & A_{N-2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ A_1 & A_2 & A_3 & \cdots & A_0 \end{matrix} \right|$$

where, m and N are some non-negative integers and define,

(5.28) $D_j := \sum_{n=0}^{N-1} w^{nj} A_n, \; j = 0, 1, ..., N-1,$

and

(5.29) $D := diag(D_0, D_1, D_2, ..., D_{N-1}),$

where, w is the i-th root (complex) of unity:

(5.30) $w := e^{2\pi i/N} := cos(2\pi/_N) + i\,sin(2\pi/_N) = cos(\beta) + i\,sin(\beta),$

and $\beta := 2\pi/_N$. Then we have the following results:

<u>Proposition</u> 5.8 [43]. The matrix A is unitarily similar to D.

<u>Corollary</u> 5.9. The spectrum of A is given by:

(5.31) $\{\lambda_{l,j}\}$ $j = 0, 1, ..., N-1, l = 1,2, ..., m,$

where, $\{\lambda_{l,j}\}$ $l = 1, 2, ..., m$ is the spectrum of D_j, $j = 0, 1, ..., N-1.$

<u>Proposition</u> 5.10 [43]. The eigenvectors of A, $\{v_i\}$, have a very special

form:

(5.32) $\quad v_i =: (v_{i,0}, v_{i,1}, ..., v_{i,N-1})^t$

where, $v_{i,j} \in R^m$ and

(5.33) $\quad v_{i,j} = w^{jM_i} v_{i,0}$ for all i, j and some integer $0 \leq M_i \leq N-1$.

Applying this result to our subdivision matrix, we can obtain:

Theorem 5.11. The subdivision matrix A for bi-quartic B-spline surfaces has the following properties provided that the weights $\{a_i\}$ are chosen properly: the eigenvectors v_2, v_3, v_4, v_5 and v_6 can be chosen to be in the form:

(5.34)
$$\left|\begin{array}{l} v_2 := \textit{Real Part}\{u_1\},\ v_3 := \textit{Imaginary Part}\ \{u_1\}, \\ v_4 := \textit{Real Part}\{u_2\},\ v_5 := \textit{Imaginary Part}\ \{u_2\}, \\ v_6 := (v_{6,0}, v_{6,0}, v_{6,0}, v_{6,0}, ..., v_{6,0})^t_, \text{ if } \lambda_4 = \lambda_5 = \lambda_6, \end{array}\right.$$

where, the vectors u_1 and u_2 have the form:

(5.35)
$$\left|\begin{array}{l} u_i := (u_{i,0}, u_{i,1}, u_{i,2}, u_{i,3}, ..., u_{i,N-1})^t_, \ i = 1, 2, \\ u_{i,j} \text{ is a vector of length 16, } i = 1, 2 \text{ and } j = 0, 1, ..., N-1, \\ u_{i,j} = w^{ji}\, u_{i,0},\ j = 0, 1, ...N-1, i = 1, 2. \end{array}\right.$$

Proof. The proof comes from the fact that

(5.36) $\quad D_i = \hat{D}_{N-i}$, for $i = 1, 2, ..., N-2$,

where, $\hat{D}_i$ is the complex conjugate of D_i. Moreover, it can be proved that $\{\lambda_i\}_{i=1, 2, ..., 6}$ are all real and for $N \geq 5$, if $\lambda_4 = \lambda_5 = \lambda_6$, λ_2 and λ_4 are simple eigenvalues of D_1 and D_2 respectively. If $\lambda_4 = \lambda_5 = \lambda_6$, then it is shown that λ_6 is an simple eigenvector of D_0. In the case of $N = 3$, similar results hold. This completes the proof.

Remark 1. Any vector in the invariant space $\mathit{span}\{v_2, v_3\}$ is an eigenvector of A corresponding to eigenvalue λ_2. This happens to the eigenvalue λ_4 too. The above special choices of the eigenvectors will simplify our curvature analysis.

Remark 2. This *Theorem* can also be proved by using the ***Fourier Transform Technique***.

Remark 3. By some simple formulation, it can be shown that all the eigenvalues and eigenvectors of A can be obtained analytically. In fact, we have:

i. D_i, $i = 0, 1, 2, \ldots, N-1$, has eigenvalues $\lambda_{i.a}, \lambda_{i.b}, \lambda_{i.c}, \lambda_{i.d}, 1/16, 1/32, 1/64, 1/128, 1/256, 0, 0, 0, 0, 0, 0, 0.$

ii. $\lambda_{0.a} = 1, \lambda_{0.b} = 1/4, \lambda_{0.c} = 1/16, \lambda_{0.d} = 1/16.$

iii. When N is even, $\lambda_{N/2.a} = 1/4, \lambda_{N/2.b} = 1/4, \lambda_{N/2.c} = 1/16, \lambda_{N/2.d} = 1/64.$

iv. $\lambda_{i.a} = \lambda_{N-i.a}, \quad \lambda_{i.b} = \lambda_{N-i.b}, \quad \lambda_{i.c} = \lambda_{N-i.c}, \quad \lambda_{i.d} = \lambda_{N-i.d}, \quad i = 1, 2, \ldots, (N-1)/2.$

v. $\lambda_{i.a}$, $\lambda_{i.b}$, $\lambda_{i.c}$ and $\lambda_{i.d}$, $i \neq 0$, $i \neq N/2$, are four roots of a quartic polynomial $P_i(\lambda)$ which depends on the weights $\{a_i\}$ of the algorithm.

For the purpose of the curvature analysis, we state here a further result about the subdivision matrix. This result can be obtained by direct evaluation.

Theorem 5.12. The local subdivision matrix can have properties B_0, B_1, B_2 and B_{20} (see section 5.2.3) if the shape control parameters $\{a_i\}$ are chosen appropriately.

5.3.6. The Curvature Property Analysis of the Surface at the E-point

In this subsection, we shall prove our main result: the normal curvature properties of the surface at the *E-point*.

Theorem 5.13. The tangent vector T_i of locus D_i at the extraordinary point V has the following form:

$$T_i = e_{i,2} f_2 + e_{i,3} f_3 = (f_2, f_3)\ rot(-i\beta)\,(c_1, c_2)^t, \text{ for } i = 0, 1, 2, \ldots, N-1, \tag{5.37}$$

where, c_1 and c_2 are constants depending on N and the initial data and $rot(-i\beta)$:= the rotation matrix with angle $-i\beta$. The tangent T_i is defined as the limit of the *Tangent Vector* of the i-th diagonal locus segment D_i^k as k goes to infinity.

Proof. Since the surface is C^1, the results follow from the results in the previous subsection about the eigenvectors and the recurrence relation of the eigen-ranges:

$$(5.38) \qquad T_i := Lim_{k\to\infty} T_i^k$$

$$:= Lim_{k\to\infty} \Delta_t\, (f_1\, v_{1.i} + f_2\, v_{2.i} + f_3\, v_{3.i} + \dots)$$

$$= \Delta_t\, \{(f_2 Re(w^i u_{1,0}) + f_3 Im(w^i u_{1,0})\},$$

where, Δ_t is a proper *Tangent Operator.*

Let $T_0 := c_1 f_2 + c_2 f_3$, then, from the above relation and the properties of the eigenvectors, we obtain that $T_i = (f_2, f_3) rot(-i\beta)(c_1, c_2)^t$. This completes the proof.

Here the rotation matrix $rot(\alpha)$ is defined explicitly by:

$$(5.39) \qquad rot(\alpha) := \begin{vmatrix} cos(\alpha) & sin(\alpha) \\ -sin(\alpha) & cos(\alpha) \end{vmatrix}.$$

Similarly, we have

Corollary 5.14. The tangent vector $TL_{i.n}$ of the locus $L_{i.n}$ at the extraordinary point V has the form:

$$(5.40) \qquad TL_{i.n} = e'_{i.2} f_2 + e'_{i.3} f_3 = (f_2, f_3)\; rot(-i\beta)\, (c'_1, c'_2)^t.$$

Remark 1. It should be pointed out that the tangent vectors $\{TL_{i,n}\}$ given by (5.40) may be a multiple of the actual tangent vectors. However, this will not affect our analysis.

Remark 2. $\{TL_{i,n}\}$ have the following properties

$$(5.41)\quad \left|\begin{array}{ll} TL_{i,n} = -TL_{i,n+N/2} & \text{for even } N, \\ TL_{i,n} = -2^{-1}cos^{-1}(\beta/2)(TL_{i+(N-1)/2,n} + TL_{i+(N+1)/2,n}) & \text{for odd } N. \end{array}\right.$$

Remark 3. Due to the symmetry of the algorithm and (5.41), we can conclude:

i. For even N, the edge loci E_i is aligned with $E_{i+N/2}$ and the diagonal loci D_i is aligned with $D_{i+N/2}$, $i = 0, 1, 2, ..., N/2-1$.

ii. For odd N, the edge loci E_i is aligned with the diagonal locus $D_{i+N-1/2}$, for $i = 0, 1, 2, ..., N-1$.

These results are consistent with that obtained in [116].

Theorem 5.15. Let C_i^k denote the N-curvature of the diagonal locus segment D_i^k. If conditions B_0, B_1, and B_2 (or B_{20}) hold, then

$$(5.42)\quad Lim_{k\to\infty} C_i^k := C_i \text{ is finite for } i = 0, 1, ..., N-1.$$

Proof. As in [45,46], it can be proved that the curvature C_i^k has the form

(5.43) $\quad C_i^k = const_i\,(\lambda_4/\lambda_2^2)^k + O(\lambda_4/\lambda_2)^{2k},$

where, $const_i$ is a constant depending on both the initial data and the tangent direction. Hence the result follows.

<u>Corollary</u> 5.16. Let $CL_{i,n}^k$ denote the N-curvature of the locus $L_{i,n}^k$ on patch $B_{i,n}^k$ at level k. If conditions B_0, B_1, and B_2 (B_{20}) hold, then

(5.44) $\quad Lim_{k\to\infty} CL_{i,n}^k \;:=\; CL_{i,n}$ is finite for $i = 0, 1,...N-1.$

From this result, we can conclude that for general data, the limits are zeros if

(5.45) $\quad 0 < \lambda_4 < \lambda_2^2$

the limits are finite (zeros or non-zeros) if

(5.46) $\quad \lambda_4 = \lambda_2^2$

and the limits are infinities if

(5.47) $\quad \lambda_4 > \lambda_2^2.$

Using the above results, we can prove the following *N-curvature* property of the surface at the *E-point.*

Theorem 5.17. If conditions B_0, B_1 and B_2 (B_{20}) hold, then there are some constants g_1, g_2, g_3, g_4 and C, which depend only on the initial data, the weights $\{a_i\}$ and the valency N of the *E-point*, such that

$$(5.48) \quad C_i = \{ (g_1, g_2) rot(-2i\beta)(g_3, g_4)^t + C\}/\langle T_i, T_i\rangle, \; i = 0, 1, 2, ..., N-1.$$

Proof. From the analysis of the eigenvectors v_4, v_5, v_6 (*Theorem* 5.11), we have

$$(5.49) \quad C_i := Lim_{k\to\infty} C_i^k$$

$$:= Lim_{k\to\infty} \langle \Delta_c(f_1 v_1 + f_2 v_2 + f_3 v_3 + f_4 v_4 + f_5 v_5 + ...), N\rangle/\langle T_i, T_i\rangle$$

$$= \langle \Delta_c\{f_4\, Re(w^{2i} u_{2,0}) + f_5\, Im(w^{2i} u_{2,0}) + f_6 v_{6,0}\}, N\rangle/\langle T_i, T_i\rangle,$$

where, Δ_c is a proper curvature operator and N is the unit normal vector of the surface at the *E-point*.

Let $g_1 := \langle f_4, N\rangle$, $g_2 := \langle f_5, N\rangle$, $g_3 := \Delta_c(Re(u_{2,0}))$, $g_4 := \Delta_c(Im(u_{2,0}))$ and $C := \langle f_6 \Delta_c v_{6,0}, N\rangle$. Then, from *Theorem* 5.11, we have

$$(5.50) \quad \Delta_c(Re(w^{2i} u_{2,0})) = g_3 cos(-2i\beta) - g_4 sin(-2i\beta)$$

and

$$(5.51) \quad \Delta_c(Im(w^{2i} u_{2,0})) = g_3 sin(-2i\beta) + g_4 cos(-2i\beta).$$

From these results, (5.49) becomes

(5.52) $\quad c_i = \{ (g_1, g_2)rot(-2i\beta)(g_3, g_4)^t + C\}/<T_i, T_i>.$

This completes the proof of the lemma.

In the same way, we can prove

Corollary 5.18. If conditions B_0, B_1 and B_2 (B_{20}) hold, then there are some constants g'_i and C' which depend only on the initial data, the weights $\{a_i\}$ and N, such that

(5.53) $\quad CL_{i,n} = \{ (g_1, g_2)rot(-2i\beta)(g'_3, g'_4)^t + C'\}/<TL_{i,n}, TL_{i,n}>, i = 0, 1, 2, ..., N-1.$

Remark 1. The *N-curvatures* $\{CL_{i,n}\}$ satisfy

(5.54) $\quad CL_{i,n} = CL_{i+N/2,n}, \; i = 0, 1, ..., N/2-1$ for even N.

Remark 2. These normal curvatures $\{CL_{i,n}\}$ may be a constant multiple of the actual *N-curvatures*. However, this will not affect our results.

Now, we prove that the normal curvatures $\{c_i\}$ in (5.48) satisfy the G-condition. It is sufficient to show that in such a case, G-condition is an identity for all the constants (f_1, f_2), (g_1, g_2), (g_3, g_3) and C.

Theorem 5.19. Suppose $N \geq 5$, then, the N normal curvatures $\{c_i\}_{i=0}^{N-1}$ satisfy the G-condition.

Proof. The G-condition (5.19) can be written in the form

$$(5.55) \qquad K_c^{j+3} = M^j K_c^j + M^{j+1} K_c^{j+1} + M^{j+2} K_c^{j+2}$$

where

$$(5.56) \qquad \left| \begin{array}{l} M^i := \langle T_{i-1} \times T_{j+3}, T_{j+3} \times T_{i+1} \rangle / \langle T_{i-1} \times T_i, T_i \times T_{i+1} \rangle \\ K_c^i := C_i \langle T_i, T_i \rangle - C. \end{array} \right.$$

Here, the cyclic convention for i in the range $(j, j+1, j+2)$ is assumed.

From (5.37), we can obtain

$$(5.57) \qquad \langle T_i \times T_j \rangle = (c_1^2 + c_2^2)\, sin(-(j-i)\beta)\, (f_2 \times f_3)$$

and hence

$$(5.58) \qquad M^i = 1, \quad M^{i+1} = - M^{i+2} = -sin(-3\beta)/sin(-\beta) = 1 - 4con^2(-\beta).$$

Combining this result with (5.56), equation (3.55) becomes

$$(5.59) \qquad (g_1, g_2)\, \{rot(-2(j+3)\beta) - rot(-2j\beta)$$
$$- (4cos^2(-\beta)-1)(rot(-2(j+2)\beta) - rot(-2(j+1)\beta))\}\, (g_3, g_4)^t = 0.$$

This is an identity for all $\{g_i\}$, j, N and (f_1, f_2) since, for any β and integer j, we know from (5.21), (5.22) and (5.23), that

$$(5.60) \qquad rot(-2(j+3)\beta) - rot(-2j\beta)$$
$$\equiv (4cos^2(-\beta) - 1)[rot(-2(j+2)\beta) - rot(-2(j+1)\beta)].$$

This completes the proof.

Corollary 5.20. The normal curvatures $\{CL_{i,n}\}$, given by (5.53) also satisfy the G-condition.

This result strongly suggests that all the *N-curvatures* of the surface at the *E-point* could satisfy the *G-condition* (5.19). Unfortunately, we cannot provide a mathematical proof. However, we believe that the surface should be C^2.

5.3.7. Remarks

1. The reason we cannot conclude that the surface is C^2 at the *E-point* is that we cannot prove any four *N-curvatures* of the surface at the *E-point* satisfy the *G-condition* (5.19). Even though all the aligned loci passing through the *E-point* is C^2, we still can not say that the surface is C^2. A simple (counter) example is $F(r,\theta) := r^2 \cos^2(M\theta + \theta_0)$, $M \neq 1$, in polar coordinates.

2. From our analyses, the surface at the *E-point* has certain *symmetric properties*. For example, the *tangent vector* T_i and the *N-curvature* C_i are just some rotations of T_0 and C_0 after affine maps. This is shown clearly in (5.37) and (5.48) etc.

3. Computer experiments show that the surfaces behave very well at

E-points.

4. The limit surface has zero curvatures (*locally flat* [3]) at the *E-point* if (5.47) holds therefore it is C^2.

5. It is suggested by *M. Sabin* [109] that condition (5.46) should be sufficient for the surface to be curvature continuous at the *E-point*. However, as far as we know, no mathematical proof is available now.

5.4. Conclusions

In this Chapter, the subdivision algorithm for bi-quartic B-spline surfaces is generalized to arbitrary networks. The main result is that the scheme produces *almost* C^2 surfaces over arbitrary topology. Curvature properties of the limit surfaces at the *E-point* are studied in detail.

The *Ball-Storry's* curvature analysis at the *E-point* for the *Catmull-Clark's* algorithm is developed to cope with even order tensor-product B-spline surface algorithms. The *Block-Circulant-Matrix* method provides a very powerful tool to study the tangent plane and (*Normal*) curvatures of the limit surfaces at one point.

Some graphic examples are given to show the smoothing process of the scheme over both uniform and non-uniform data.

5.5. Graphic Examples

The graphics were produced by *Nichlet Drum Plotter at Brunel University, UK., 1988-1990.*

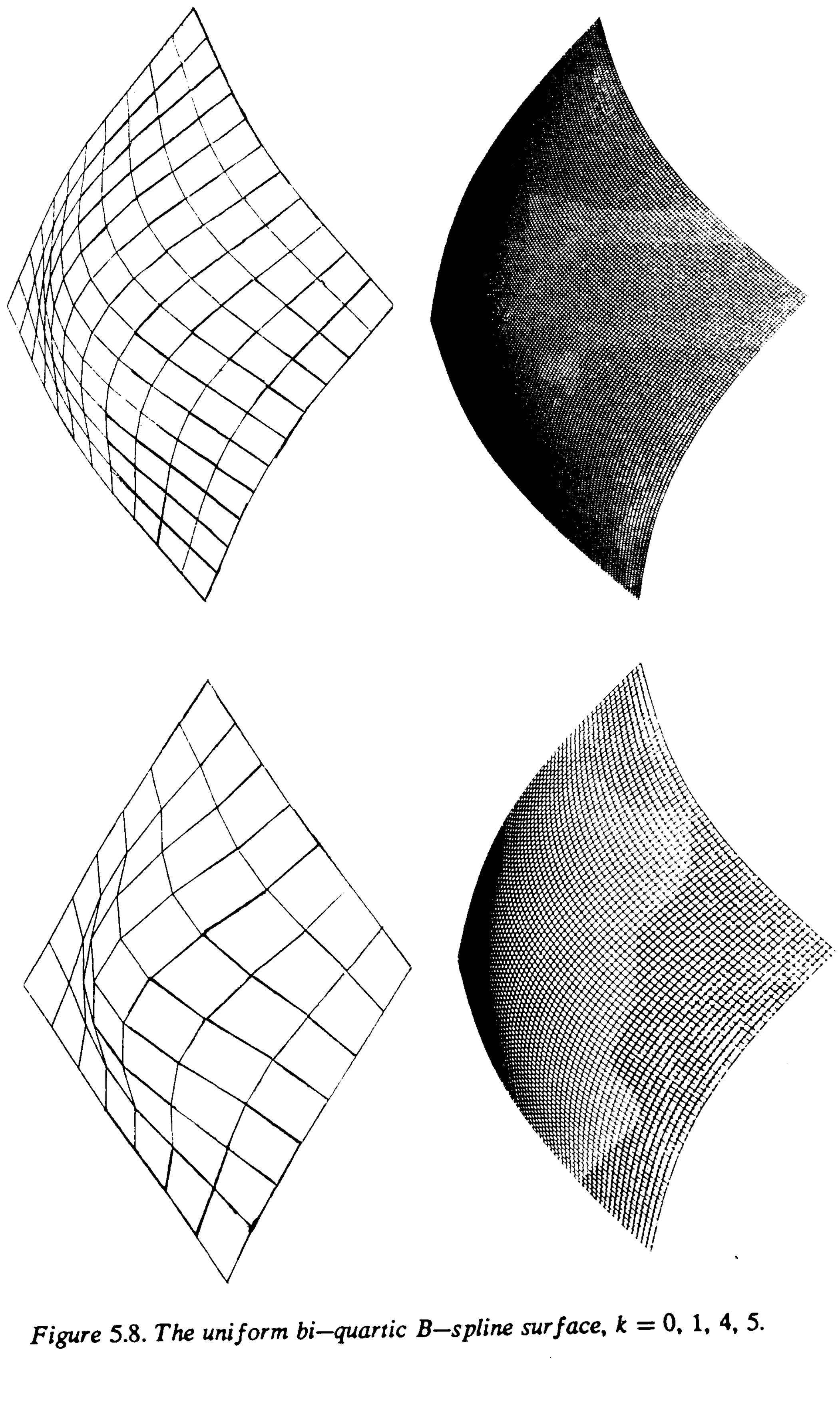


Figure 5.8. The uniform bi-quartic B-spline surface, $k = 0, 1, 4, 5$.

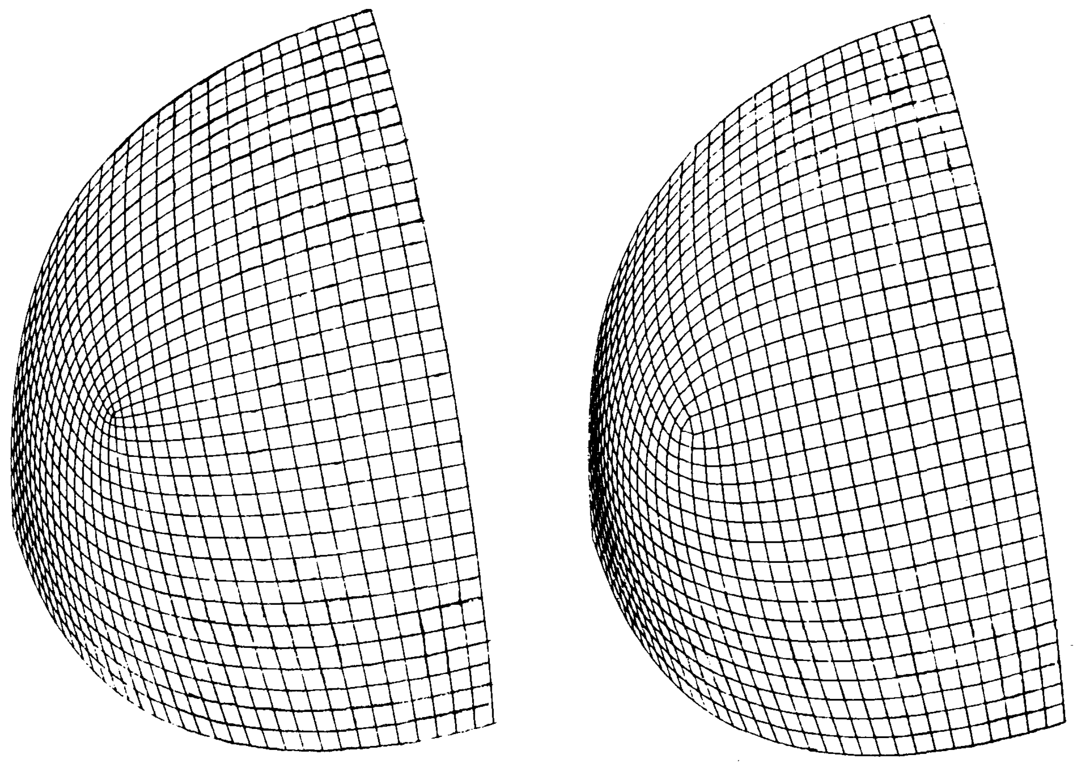


Figure 5.9. Smoothing down a 3–sided hole with different shape controls, k = 4.

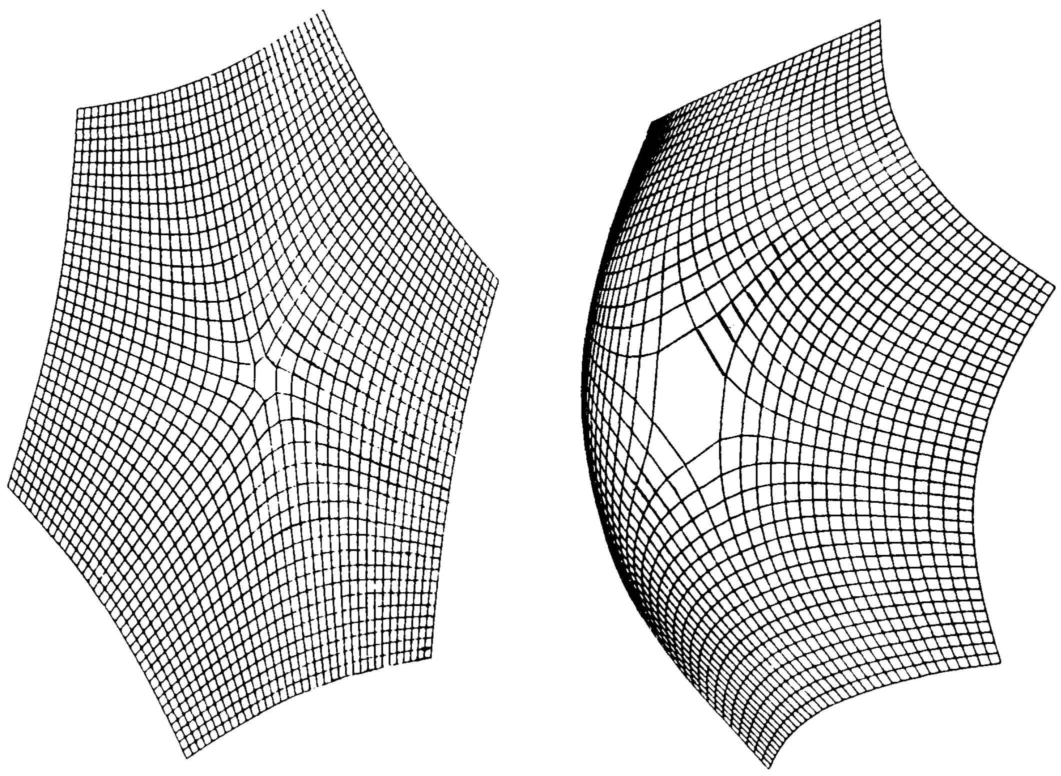


Figure 5.10. Smoothing down a 6–sided hole with different shape controls, k = 4.

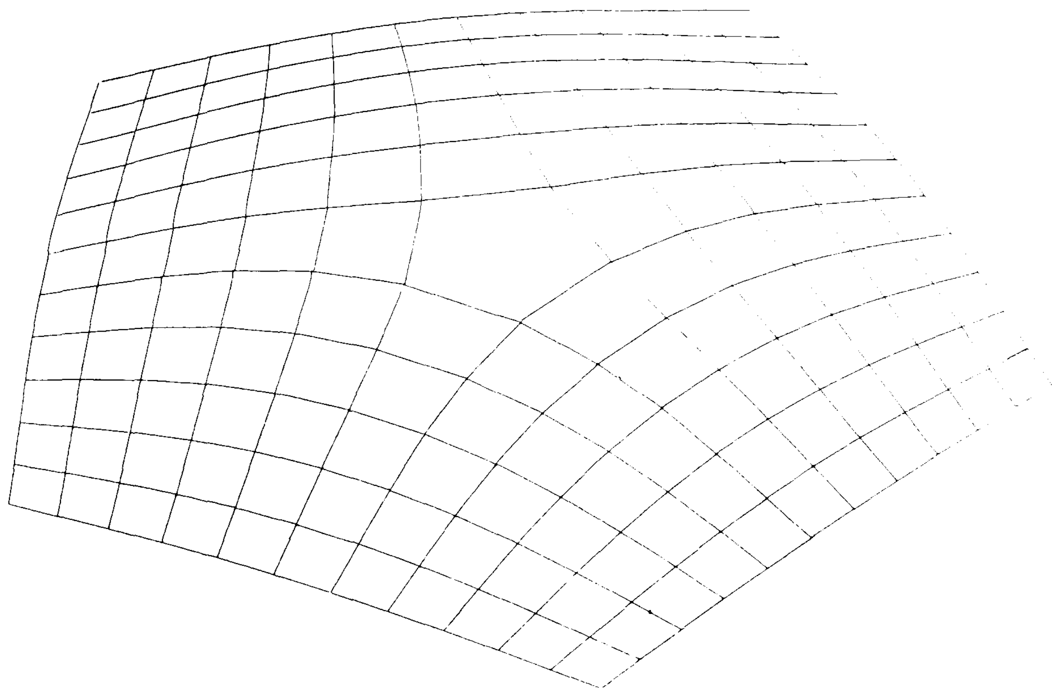


Figure 5.11. *The scheme at an E–point, $N = 5$, $k = 2$.*

CHAPTER SIX

SUBDIVISION ALGORITHMS BASED ON TRIANGULATIONS

In this Chapter, we shall restrict our attention to the study of recursive subdivision algorithms for the generation of surfaces over triangulations. A generalization of the *Dyn-Gregory-Levin* scheme defined over uniform triangulations will be discussed. By studying the *Cross-Differences of Directional Divided Differences,* a 10-point interpolatory subdivision scheme will then be constructed and studied. And, in particular, a special case of it, the so called *butterfly scheme,* will be analysed in detail. Finally, the 10-point interpolatory schemes will be generalized to non-uniform triangular networks. In both cases, the scheme generates smooth surfaces. Some graphic examples produced by the scheme over uniform data are also presented .

6.1. Introduction

In complete analogy to the generalization of the *Chaikin's* algorithm and the *DGL's* 4-point interpolatory algorithm to a general *Dyn-Gregory-Levin* scheme discussed in Chapter 1, a uniform subdivision algorithm for surfaces can be easily formulated. Similarly, uniform subdivision algorithms for surfaces in higher dimensional spaces can also be derived.

A scheme is said to be *uniform* if all its weightings are constants and non-uniform otherwise. Since the uniform parametrization, that is, the *diadic parametrization,* is used to analyse the convergent property of the scheme, uniform subdivision schemes are also called *Binary Subdivision Schemes* (*BSS*).

Before we consider the construction of the algorithm, two important features of the *Doo-Sabin's* and *Catmull-Clark's* algorithms should be noted. Firstly, both algorithms consist of repeated averages, which provides an easy means to increase the smoothness of the surfaces. Secondly, the underlying surface spaces are somehow nested which is reflected by the simple structural characterization of uniform tensor product splines as piecewise polynomial patches. Thus, for instance, *Catmull-Clark's* algorithm can be thought of a generalization of the *Doo-Sabin's* algorithm. Therefore, it is hoped that the scheme should also have similar properties.

It should be noted that the uniform or *diadic parametrization*, is assumed in the convergence analysis, unless stated otherwise.

The use of *diadic parametrization* means that the (uniform) *control nets*, say, $\{P^k_{i,j} \,/\, (i,j) \in Z^2\}$ are associated with the corresponding *diadic values* $\{2^{-k}(i,j) \,/ (i,j) \in Z^2\}$ in the u–v parameter plane. Hence the control net at level k, which is denoted by P^k, can then be represented unambiguously as the piecewise linear interpolant to the triangulated data $\{(2^{-k}(i,j), P^k_{i,j}) \,/ (i,j) \in Z^2\}$.

Using this type of parametrization, a generalized form of ***RSA*** for surface generation over uniform triangular control polyhedrons can be derived. For notational convenience, the notations introduced in Chapter 4 are used in this Chapter to formulate this algorithm.

A subdivision scheme, S, is said to be a convergent scheme if for every set of control points $P^k = \{P^k_{i,j} \,/ (i,j) \in Z^2\}$, there is a continuous function P on R^2 such that for all $(i,j) \in Z^2$,

(6.1) $$Lim_{k\to\infty} \,|\, (S^k P)_{(i,j)} - P(i2^{-k}, j2^{-k})| = 0.$$

We denote the above function P by $S^{\infty}P^0$, and call it the limit function of S on P^0. If P is a C^n surface, then the scheme is called a C^n scheme. We say that the convergence is uniform if for a given compact region Ω and an arbitrary real number, $\epsilon > 0$, there exists a positive integer $K(\epsilon, \Omega)$ such that for all $n > K(\epsilon, \Omega)$, $\alpha \in Z^2\}$, (6.2) $$\left| (S^n P)_\alpha - P(2^{-n}\alpha) \right| < \epsilon.$$

In order to define the uniform triangulation of the control net, we need to introduce the uniform triangulation of the parameter values, that is, the *diadic points*, in the parameter plane first. The triangulation of the control net is then defined according to the triangulation on the u–v plane. We assume that the uniform triangulation of the parameter plane is taken to be the standard uniform *3-D triangulation*, that is, it is the *uniform triangular mesh* produced by directions: $(0, 1)$, $(1, 0)$ and $(1, 1)$ as shown in Figure 6.1.

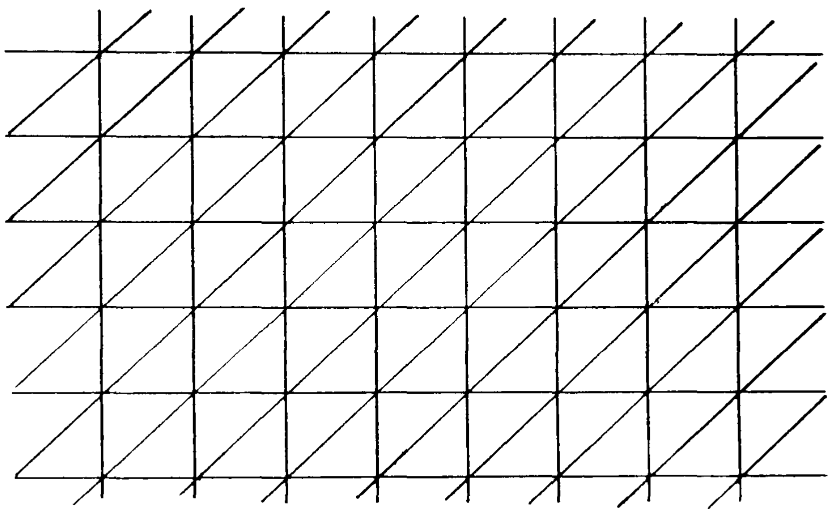


Figure 6.1.

6.2. A Uniform Subdivision Scheme Defined on Uniform Triangular Control Polyhedrons

In this section, we firstly describe a generalization of the *DGL* scheme for surfaces over a uniform triangulation of the control polyhedron and then study its basic properties. For the sake of clarity, several special cases of this scheme will be briefly introduced.

6.2.1. Mathematical Description of the Scheme

The *Dyn-Gregory-Levin* scheme for surfaces can be described as follows. Given initial data $\{P^0_{i,j}\}$, then the refined control points are given recursively by the following formulae (*masks*): for $k = 0, 1, 2, \ldots$

$$(6.3)\quad \left|\begin{array}{lcl} P^{k+1}_{2i,2j} & = & \sum_{m,n\in M_0} a_{m,n} P^k_{i+m,j+n} \\ P^{k+1}_{2i+1,2j} & = & \sum_{m,n\in M_0} b_{m,n} P^k_{i+m,j+n} \\ P^{k+1}_{2i,2j+1} & = & \sum_{m,n\in M_0} c_{m,n} P^k_{i+m,j+n} \\ P^{k+1}_{2i+1,2j+1} & = & \sum_{m,n\in M_0} d_{m,n} P^k_{i+m,j+n} \end{array}\right.$$

where, $\{a_{m,n}, b_{m,n}, c_{m,n}, d_{m,n}\}$ are constants and M_0 is a fixed finite integer set describing the local structure of the scheme. For convenience, the summation $\sum_{m,n\in M_0}$ will be replaced by $\sum_{m,n}$.

By the use of multiple notations, scheme (6.3) can be written in a more compact form:

$$(6.4)\quad P^{k+1}_{\alpha} = \sum_{\beta\in Z^2} a_{\alpha-2\beta} P_{\beta}, \quad \alpha\in Z^2.$$

An equivalent form of this formula is

$$(6.5)\quad P^{k+1}_{\gamma+2\alpha} = \sum_{\beta\in Z^2} a_{\gamma-2\beta} P_{\alpha+\beta}, \quad \alpha\in Z^2.$$

where, $\gamma := \{(\gamma_1, \gamma_2): \gamma_i = 0 \text{ or } 1,\ i = 1, 2\}$. Thus, the scheme is interpolatory if and only if

(6.6) $a_\alpha = \delta_{\alpha,0}$ for all $\alpha \in Z^2$.

Scheme (6.3) is a 4-step scheme, or a scheme with four simple *masks*. That is, there are four formulae to calculate the refined control points and each control point is calculated by using one of the *masks* according to its relative position and its local topology.

6.2.2. Basic Properties of the BSS

From (6.3), the following properties of the scheme can be easily obtained. They are very similar to the properties of the curve generating *DGL* scheme.

i. The algorithm is a weighted local averaging algorithm.

ii. The scheme is a linear scheme.

iii. The scheme is translation-and-rotation invariant (*coordinate-free*).

iv. The *Doo-Sabin's* algorithm and the *Catmull-Clark's* algorithm are special cases of the algorithm.

v. The algorithm can produce piecewise tensor-product polynomial surfaces.

vi. Any uniform tensor-product type *DGL* algorithm is a special case of this algorithm.

vii. Multivariate box (cube)-spline algorithms are encompassed by this scheme.

viii. A necessary condition for the scheme to produce continuous surfaces is that it reproduces any constant surface, that is,

$$(6.7) \qquad \sum_{m,n} a_{m,n} = \sum_{m,n} b_{m,n} = \sum_{m,n} c_{m,n} = \sum_{m,n} d_{m,n} = 1.$$

Further properties of the algorithm are also studied. For instance, the necessary and sufficient conditions for the scheme to produce C^n surfaces were derived by several methods. The *directional divided difference approach* using matrix analysis and the *generating polynomial method* are just two of them. The former method will be discussed in section 6.4 as a means to prove the C^1 property of the 10-point scheme and the latter one was used to obtain the same result by *Dyn, Levin* and *Micchelli* [54]. Other studies about this algorithm can also be found in [12,13,28,40,51,85,88].

It is interesting to point out that the *Interpolatory Subdivision Scheme* (*ISS*) also belongs to this category. Such schemes for curves and surfaces have been analysed by *Dubec* [47], *Dyn, Gregory* and *Levin* [48,49], *Dyn, Levin* and *Liu* [53], *Gregory* and *QU* [69], *Dyn and Levin* [49], *Dyn, Levin* and *Micchelli* [54], and *Weissman* [121].

In the case of ***ISS***, one of the *masks* is the identity *mask* which maps all the control points of level k into level $k+1$, that is, $P^k \in P^{k+1}$. This implies that the convergence of ***ISS*** is uniform.

In the paper by *Dyn and Levin* [53*a*], the following special properties of ***ISS*** are obtained.

Proposition 6.1. Let S be an ***ISS*** of the form (6.3). If S generates C^n limit functions, then S reproduces π_n (the space of all bivariate polynomials of degree $\leq n$).

Proposition 6.2. Let S be an ***ISS*** of the form (6.3). If S generates C^n limit functions, then there exists a (not unique) matrix ***BSS***, $S^{(m)}$ of order $m+1$ such that for $P^k =: S^k P^0$, $k > 0$,

(6.8) $$d^m P^{k+1} = S^{(m)}(d^m P^k), \quad \text{for all } 1 \leq m \leq n.$$

Moreover, for any initial data of the form $d^m P^0$, $S^{(m)}$ converges uniformly to C^{n-m} limit vector-valued functions.

Proposition 6.3. Let S be an ***ISS*** of the form (6.3) which reproduces π_n. Then the following conditions are equivalent:

(i). S converges uniformly to C^n functions.

(ii). The matrix scheme $S^{(m)}$, $m = 0, 1, 2, \ldots, n$, converges uniformly to C^{n-m}

vector-valued functions for any initial data of the form $d^m P^0$.

(iii). For any initial data, $\frac{1}{2}S^{(n+1)}$ converges uniformly to zero.

In the next subsection, we will study a special ***ISS*** for the generation of surfaces using only one mask and its *duals*.

6.2.3. The 10-point Interpolatory Scheme for Surfaces

Here, a special interpolatory scheme, the 10-point scheme, is described in this subsection. One of the advantages of this scheme is that it can be used on both uniform and non-uniform triangular networks. This will be discussed in detail in section 6.5. The main property of the scheme is that it reproduces cubic bivariate parametric polynomials when the parameters are chosen properly. In addition to this, there is a free parameter t, which can be used as a control to manipulate the shape of the surface.

The construction of the 10-point interpolatory scheme is, originally, motivated by the ideas described in [47] and [48]. The scheme is formulated in order to solve the problems of high accuracy surface fitting and the fast surface generation. Thus, our aim is to generalize the *4-point interpolatory subdivision scheme* as described in [47,48] to surfaces. The scheme is so constructed that it preserves all its advantages. The main properties of the scheme, in addition to the properties of ***BSS***, are interpolatory, shape control and reproductivity for cubic parametric polynomial surfaces when the parameters are well chosen.

The 10-point interpolatory scheme is defined by the following choice of the coefficients in (6.3):

$$
(6.9)\quad \left|\begin{array}{l}
a_{0,0} = 1 \\
b_{0,0} = b_{1,0} = 1/2 - 2w_1 - w_2 - w_3 \\
b_{-1,0} = b_{2,0} = w_3 \\
b_{0,-1} = b_{1,1} = w_2 \\
b_{-1,-1} = b_{1,-1} = b_{0,1} = b_{2,1} = w_1 \\
c_{0,0} = c_{0,1} = 1/2 - 2w_1 - w_2 - w_3 \\
c_{0,-1} = c_{0,2} = w_3 \\
c_{-1,0} = c_{1,1} = w_2 \\
c_{-1,-1} = c_{-1,1} = c_{1,0} = c_{1,2} = w_1 \\
d_{0,0} = d_{1,1} = 1/2 - 2w_1 - w_2 - w_3 \\
d_{-1,-1} = d_{2,2} = w_3 \\
d_{1,0} = d_{0,1} = w_2 \\
d_{0,-1} = d_{-1,0} = d_{1,2} = d_{2,1} = w_1
\end{array}\right.
$$

where, w_i, $i = 1, 2, 3$ are three shape control parameters.

Due to the 3-direction-symmetry property of the scheme, a simple way, which uses only a single formula (one mask and its *rotations*), to describe the algorithm is given below (Figure 6.2). Since the scheme is interpolatory, only the inserted values are to be evaluated. The formula for the inserted points is given by

$$(6.10)\quad P_o := 1/2\{P_e + P_f\} + w_1\{P_a + P_c + P_h + P_j - 2P_e - 2P_f\}$$
$$+ w_2\{P_b + P_i - P_e - P_f\} + w_3\{P_d + P_g - P_e - P_f\},$$

where, o is the *midpoint* of the edge joining the vertices e and f in the parameter plane and w_i, $i = 1, 2, 3$ are three shape controls.

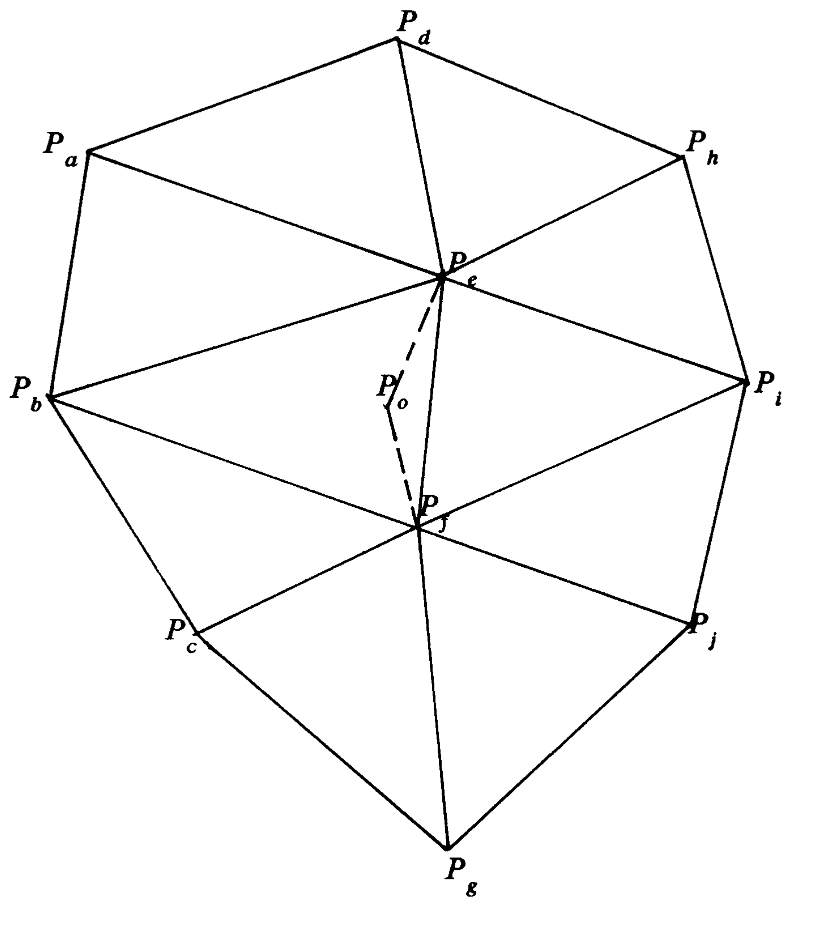


Figure 6.2. The 10-point scheme.

In this process, formula (6.10) is used to evaluate all the surface values at the *midpoints* in the u–v parameter plane to produce a refined uniform triangulation. The triangulation of the refined control polyhedron is formed accordingly by the refined uniform triangulation of the u–v plane. Repeated applications of this process will therefore result in finer and finer control polyhedrons which will finally converge to a smooth interpolatory surface provided that the parameters $\{w_i\}$ are chosen appropriately. This will be discussed in sections 6.4 and 6.5.

From formula (6.10), it can be easily shown that the scheme has the following properties.

i. The scheme is interpolatory.

ii. The tension parameters $\{w_i\}$ working along three mesh directions respectively.

iii. The scheme reproduces linear surfaces for all $\{w_i\}$. Furthermore, it reproduces any bivariate cubic polynomial surface if $\{w_i\}$ satisfies the conditions

$$(6.11)\quad \left|\begin{array}{lcl} w_1 & = & (16t-9)/16 \\ w_2 & = & -2(16t-9)/16 \\ w_3 & = & (8-16t)/16, \end{array}\right.$$

where, t is any real number.

iv. If the tension parameters are chosen to be

$$(6.12)\quad w_1 = w, \quad w_2 = -2w, \quad w_3 = 0,$$

then the scheme reduces to the *Butterfly Scheme*, which will be discussed in section 6.4.

v. The scheme has certain data-dependent shape preserving properties.

vi. Under certain conditions, the scheme produces C^1 surfaces. This will be shown in sections 6.4 and 6.5.

6.2.4. The Butterfly Scheme

The butterfly scheme is just a special case of the 10-point interpolatory subdivision scheme, where the parameters $\{w_i\}$ are given by (6.12). Thus, in the scheme, there is only one free parameter w. The parameter w has an obvious geometric meaning. When $w = 0$, the scheme has no effect on the control polyhedron (linear precision); when $w = -1/16$, the scheme reproduces cubic bivariate polynomials (cubic precision); when $-1/16 < w < 0$, the scheme is just a convex combination of the above linear precision and the cubic precision schemes.

The main advantage of the butterfly scheme over the 10-point *ISS* is that it uses only eight points instead of ten points and that the tension parameter w has an intuitive interpretation. Another important feature of the butterfly scheme is that if the initial data satisfy certain *convex conditions*, the scheme can produce smooth interpolatory convex surfaces. More about the scheme will be described in section 6.4.

6.2.5. The Tensor-product of DGL's Interpolatory Scheme

Another *ISS* for surfaces is the tensor-product *DGL's* 4-point interpolatory scheme. On uniform triangular networks, the scheme is given by the following choice of the coefficients in (6.3):

$$
(6.13)\quad \left|\begin{array}{lllll}
a_{0,0} & = 1 & & & \\
b_{0,0} & = b_{1,0} & = 1/2 + w_1 & & \\
b_{-1,0} & = b_{2,0} & = -w_1 & & \\
c_{0,0} & = c_{0,1} & = 1/2 + w_2 & & \\
c_{0,-1} & = c_{0,2} & = -w_2 & & \\
d_{0,0} & = d_{0,1} & = d_{1,0} & = d_{1,1} & = (1/2 + w_1)(1/2 + w_2) \\
d_{-1,-1} & = d_{2,2} & = d_{2,-1} & = d_{-1,2} & = w_1 w_2 \\
d_{-1,0} & = d_{-1,1} & = d_{2,0} & = d_{2,1} & = -w_1(1/2 + w_2) \\
d_{0,-1} & = d_{1,-1} & = d_{0,2} & = d_{1,2} & = -w_2(1/2 + w_1),
\end{array}\right.
$$

where, w_1 and w_2 are two tension parameters along two parametric directions respectively.

The tension parameters w_1 and w_2 have obvious geometric meanings. When $w_1 = 0$, the generated surfaces are piecewise linear surfaces along one parametric direction. Similar result holds for w_2. When $w_1 = w_2 = 1/16$, the scheme reproduces parametric bi-cubic polynomial surfaces. When $0 < w_1, w_2 < 1/16$, the scheme is a convex combination of the above two schemes.

As a consequence of the results obtained in [27,48], we can conclude the following.

Theorem 6.4. If the tension parameters $\{w_i\}$ are chosen such that

(6.14) $\quad 0 < w_1, w_2 < (\sqrt{5} - 1)/8$

then the scheme produces smooth interpolatory tensor-product type surfaces.

Proof. The proof comes from a simple observation that the fundamental function $\psi(u,v)$ produced by the scheme on the cardinal data $P^k_{i,j} = \delta_{i,0}\, \delta_{i,0}$ is just the product of the univariate fundamental functions $\phi(u)$ and $\phi(v)$. That is,

(6.15) $\quad \psi(u,v) = \phi(u)\, \phi(v)$ for all u and v.

In fact, by simple calculations, we can prove that the control points satisfy the condition

(6.16) $\quad P^k_{i,j} = F^k_i F^k_j$ for all i, j and k,

where, $\{F^k_i\}$ is the control polygon produced by *DGL*'s scheme on the cardinal data $\{\delta_{i,0}\}$. This completes the proof.

A simple way to describe the scheme is to write it in a more compact form. In fact, the algorithm can be described by the following recursive relation: for all $i, j \in Z$, $k \geq 0$,

$$(6.17)\quad \left| \begin{aligned} P^{k+1}_{2i,2j} &= P^{k}_{i,j} \\ P^{k+1}_{2i+1,2j} &= (1/2 + w_1)(P^{k}_{i,j} + P^{k}_{i+1,j}) - w_1(P^{k}_{i-1,j} + P^{k}_{i+2,j}) \\ P^{k+1}_{i,2j+1} &= (1/2 + w_2)(P^{k+1}_{i,2j} + P^{k+1}_{i,2j+2}) - w_2(P^{k+1}_{i,2j-2} + P^{k+1}_{i,2j+4}). \end{aligned} \right.$$

Other properties of the scheme, such as approximation property and shape preserving conditions, can also be investigated. However, we will not discuss these properties in this thesis. The interested reader is referred to the papers [48,53a].

6.3. Convergence Theories about the BSS

In this section, we discuss some of the methods and techniques used for the analysis of the uniform subdivision scheme. Although the results are quite similar to those obtained for curve generating schemes, the proofs in the surface case are much more difficult than that in the curve case [28,29,42,47,48,50,51,69,85,88,90,91,etc.].

6.3.1. Some General Results about Convergence

There are several ways to analyse the convergent property of the binary subdivision scheme. Among the techniques are the *generating function (polynomial) method* introduced by *Cavaretta, Dahmen* and *Micchelli* (also see *Dyn, Levin* and *Micchelli* in [54]) in [28,29], *difference and cross-difference of directional divided differences method* using matrix analysis used by

Gregory and *QU* [69] (which comes from the univariate matrix analysis [51]) and the *functional equation method* employed by *Micchelli* and *Prauzstch* etc. in [42,86,88,101,102]. Here, we state some of the convergent and smoothness results about the *BSS*.

__Proposition__ 6.5 [29,...]. A ***BSS*** is a C^0 scheme if it is contractive. More explicitly, there exists a constant $0 < M < 1$ and some positive integer p, such that

$$E^{k+p} \leq M\, E^k \quad \text{for all} \quad k \geq 0, \tag{6.18}$$

where

$$E^k := max_{\alpha,\gamma} \{|P^k_{\alpha+\gamma} - P^k_{\alpha}|\}, \; \alpha \in Z^2 \tag{6.19}$$

and

$$\gamma \in \{(0, 1), (1, 0), (1, 1)\}. \tag{6.20}$$

__Remark.__ (6.18) is only a sufficient condition, it is not necessary. It can be shown that a necessary condition for C^0 convergence is that $\{E^k\}$ converges to zero.

__Proposition__ 6.6 [28,53*a*,...]. The above scheme is a C^n scheme if all its corresponding *n-th order divided difference schemes* (there are a total of n+1 such schemes since there are n+1 possible combinations of the divided differences of order n along two different parameter directions) of the scheme are C^0 schemes.

Proposition 6.7 [28,54,...]. Let S_a, S_q be convergent *BSS* and their *Laurent* polynomials are denoted by $a(z)$ and $q(z)$ respectively. If for some $\alpha \in Z^2\backslash\{0\}$

(6.21) $a(z) = (1 + z^{-\alpha})\, q(z)/2$, for $z \in C^2$.

Then

(6.22) $\Delta_\alpha S_a(X) = S_q(\Delta_\alpha X)$.

Proposition 6.8 [28,29,...]. Suppose a *BSS* converges for all initial data and the limit surface is not always trivial. Then its mask $\{a_\alpha: \alpha \in Z^2\}$ determines a unique compactly supported continuous function f with the following properties.

(i) $f(X) = \sum_{\alpha \in Z^2} a_\alpha f(2X - \alpha)$.

(ii) $\sum_{\alpha \in Z^2} a_\alpha f(X - \alpha) = 1$.

Here, (i) is referred to as the functional equation associated with the *mask* $\{a_\alpha\}$.

Proposition 6.9 [53*a*,...]. Let $f \in C^n$ be a non-trivial solution to the above functional equation associated with the mask $\{a_\alpha: \alpha \in Z^2\}$. Then

(6.23) $\pi_n(R) \in \Phi$,

where

(6.24) $\Phi \; := \; span\{\, f(.\,-\beta) \colon \beta \in Z^2\}.$

In the following sections, we will study the 10-point scheme and the butterfly scheme in detail.

6.4. The Convergence Analysis of the Butterfly Scheme over Uniform Triangular Networks

In this section, by using the *matrix analysis* of the *differences* and the *cross-differences of the directional divided differences,* we analyse the C^0 and C^1 properties of the butterfly scheme over uniform triangular networks.

6.4.1. Introduction to the Butterfly Scheme

The *Butterfly Scheme,* as discussed in the previous sections, is an interpolatory subdivision scheme which is defined over arbitrary triangular networks. When the initial network is uniform, its smoothness properties have already been investigated by several authors in [52,54,69]. Different techniques are employed to analyse the convergence properties of the limit surfaces. One such method involves the analysis of the *generating polynomial* of the scheme as described in the paper by *Dyn, Levin* and *Micchelli* [54]. Equivalently, by using the matrix analysis, we study its corresponding *difference and directional divided difference schemes*. This approach is a generalization of the binary subdivision analysis for curves in [69]. Here, the main task is to show that all the *directional divided difference schemes* of the butterfly scheme are C^0 schemes in order to prove that the limit surfaces are C^1 continuous.

The analysis presented in this section is based on the uniform parametrization to the uniform triangular network. More explicitly, we assume that the initial surface triangulation consisting the given network is a uniform triangulation, that is, each face of the network is a triangular face and every vertex of it is of valency six. Thus, six and only six edges meet at a vertex. Under this assumption, the network is equivalent, topologically, to a uniform triangulation of a u–v parameter plane as shown by Figure 6.1. It is this property that makes the uniform (*diadic*) parametrization to be a proper parametrization. The use of uniform binary subdivision parametrization also simplifies the analysis. In order to prove that the surface is C^0 or C^1, we will show that any component of the limit surface is C^0 or C^1. Thus, throughout the rest of the Chapter, we shall restrict our attention to discuss the function surfaces instead of three-component parametric surfaces.

6.4.2. Mathematical Formulation of the Scheme

The butterfly scheme is an 8-point interpolatory recursive subdivision algorithm defined as follows.

Let $a, b, c, d, \ldots$ denote the vertices of a uniform triangulation of the $u\text{--}v$ parameter plane. Suppose the values of a function $F(u,v)$ are also given at these discrete points. These values are denoted by $F_a, F_b, F_c, F_d, \ldots$ respectively (Figure 6.3). The scheme uses a formula to estimate the value of $F(u,v)$ at the midpoint of any edge of the triangulation in the $u\text{--}v$ plane. The formula is given by

$$(6.25) \qquad F_o := 1/2\{F_d + F_e\} + w\{F_a + F_c + F_f + F_h\} - 2w\{F_b + F_g\},$$

where, o is the midpoint joining d and e, w is a real number called the tension parameter.

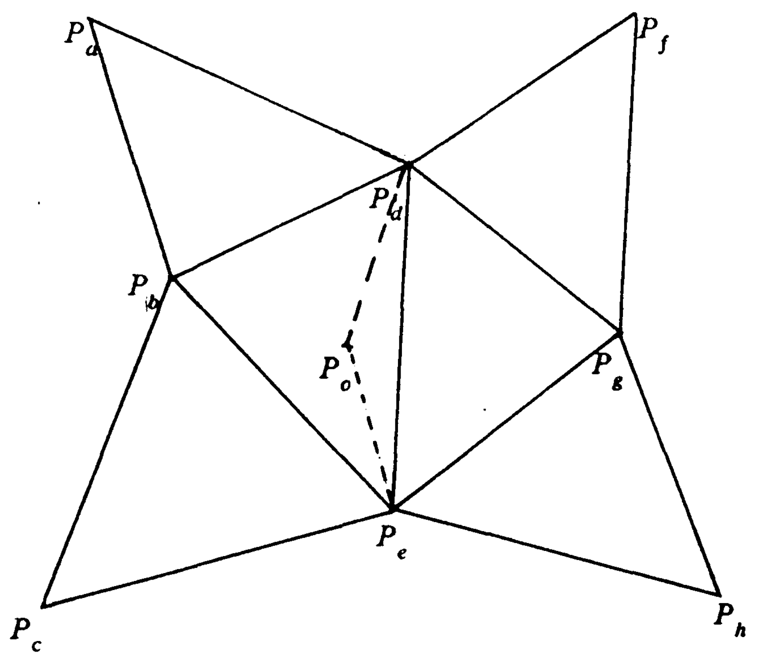


Figure 6.3. *The Butterfly Scheme.*

The formula (6.25) is used to compute all the function values at all the *midpoints* of the triangulation in the parameter plane. Thus, function values are now given on a refined triangulation which is formed by adding all the halving lines to the original triangulation. The use of midpoint subdivision means that the new triangulation vertices are created by a binary subdivision of the previous ones. This process defines one level of the recursion. In general, the recursion will define a new set of values at level $k+1$ from an old set of values at level k. The piecewise linear interpolant on the triangulation to the values at level k is called the control polyhedron (control net) F^k of the recursion. The process is interpolatory since the values at level k are included in those given at level $k+1$. The surface of the scheme is thus defined as the limit surface (if it exists) of the control net sequence $\{F^k\}$.

The scheme has the following properties.

i. The sum of the eight coefficients is unity.

ii. The scheme is exact for linear functions for all w.

iii. The scheme is exact for cubic polynomials if $w = -1/16$, that is, it has cubic precision.

iv. For general $-1/16 < w < 0$, the scheme is a convex combination of the linear precision scheme and the cubic precision scheme.

v. The scheme is both local and linear.

For the purpose of analysis, the following notations will be used. The initial function values of the control net F^0 are assumed to be given on the uniform integer grid $\{(i,j)/\ (i,j) \in \mathbf{Z}^2\}$. Then, at level k, the values

$$(6.26) \qquad F^k_{i,j} \ := \ F(i2^{-k}_{,}\ j2^{-k})$$

of the control net F^k are given on the refined grid $\{(i2^{-k}_{,}\ j2^{-k})/\ (i,j) \in \mathbf{Z}^2\}$.

The triangulation of the grid, which is used in the definition of the butterfly scheme, is taken along the directions $(1, 0)$, $(0, 1)$ and $(1, 1)$ as shown in Figure 6.1. Using this notations, the butterfly scheme is then defined by a binary subdivision on the uniform grid as

$$(6.27) \qquad \left| \begin{array}{lcl} F^{k+1}_{2i\,,2j} & = & F^k_{i,j} \\ F^{k+1}_{2i+1,2j} & = & 1/2\,\{F^k_{i,j} + F^k_{i+1,j}\} \ -2w\,\{F^k_{i,j-1} + F^k_{i+1,j+1}\} \\ & & +w\,\{F^k_{i-1,j-1} + F^k_{i+1,j-1} + F^k_{i,j+1} + F^k_{i+2,j+1}\} \end{array} \right.$$

with $F^{k+1}_{2i\,,2j+1}$ and $F^{k+1}_{2i+1,2j+1}$ being the duals (according to the local topology) of the second equation. Also, the *forward difference operators* $\{\Delta_i\}$ along the grid directions are defined by

$$(6.28) \qquad \left| \begin{array}{lcl} \Delta_1\, F^k_{i,j} & := & F^k_{i+1,j} \ - \ F^k_{i,j} \\ \Delta_2\, F^k_{i,j} & := & F^k_{i,j+1} \ - \ F^k_{i,j} \\ \Delta_3\, F^k_{i,j} & := & F^k_{i+1,j+1} \ - \ F^k_{i,j}. \end{array} \right.$$

6.4.3. C^0 Convergence Analysis-Difference Analysis

In this subsection, we discuss the C^0 property of the limit surface of the butterfly scheme. To this end, we will prove that the control net sequence $\{F^k\}$ is a *Cauchy* sequence for some properly chosen tension parameter w.

Theorem 6.10. The sequence $\{F^k\}$ is a *Cauchy* sequence in C^0 if

$$(6.29) \qquad -0.1215 \leq w \leq 0.0740.$$

Consequently, its limit $F(u,v)$ is continuous.

Proof. The proof that $\{F^k\}$ is a *Cauchy* sequence is a direct result of *Lemmas* 6.11 and 6.12.

Lemma 6.11. For $k = 0, 1, 2, 3, \dots,$

$$(6.30) \qquad \| F^{k+1} - F^k \| \leq 4|w|\, E^k,$$

where

$$(6.31) \qquad E^k := max_{(i,j) \in Z^2}\, max_m \{|\Delta_m F^k_{i,j}|\}.$$

Proof. Let $Err := |F^{k+1} - F^k|$. Then, by definition, Err is a piecewise linear function. Thus its extremes can only be achieved at the vertices of the k+1st triangulation. Hence

$$(6.32) \qquad max(Err) = |F_o^{k+1} - 1/2\,(F_d^k + F_e^k)|$$

$$\leq 4|w|\,E^k.$$

Here, o is some midpoint on the k-th triangulation on the parameter plane and F_o^{k+1} is given by (6.25). This completes the proof.

Lemma 6.12. For $k = 0, 1, 2, 3, ...,$

$$(6.33) \qquad E^{k+2} \leq C(w)\,E^k,$$

where, $0 < C(w) < 1$ if condition (6.29) holds.

Proof. By expressing the differences at level k+1 in terms of differences at level k gives, for $w < 0$,

$$(6.34) \qquad E^{k+1} \leq (1/2 - 6w)\,E^k,$$

and hence (6.33) holds for $-1/12 < w < 0$.

The proof of (6.33) under condition (6.29) can be obtained by expressing differences at level k+2 in terms of differences at level k and then use the triangle inequality to bound E^{k+2}. For example,

$$(6.35) \qquad F^{k+2}_{4i+1,4j} - F^{k+2}_{4i,4j}$$

$$= 1/2\{F^{k+1}_{2i,2j} + F^{k+1}_{2i+1,2j}\} - 2w\{F^{k+1}_{2i+1,2j+1} + F^{k+1}_{2i,2j-1}\}$$

$$+w\{F^{k+1}_{2i,2j+1} + F^{k+1}_{2i+2,2j+1} + F^{k+1}_{2i-1,2j-1} + F^{k+1}_{2i+1,2j-1}\} - F^{k+1}_{2i,2j}$$

$$= 1/2\{F^{k+1}_{2i+1,2j} - F^{k}_{i,j}\} + w\{F^{k+1}_{2i,2j+1} - 2F^{k+1}_{2i+1,2j+1} + F^{k+1}_{2i+2,2j+1}\}$$

$$+w\{F^{k+1}_{2i-1,2j-1} - 2F^{k+1}_{2i,2j-1} + F^{k+1}_{2i+1,2j-1}\}.$$

Writing the above control points at level $k+1$ in terms of the control points at level k and arranging them in a proper combination, we obtain

$$(6.36)\quad |F^{k+2}_{4i+1,4j} - F^{k+2}_{4i,4j}| \le \{2|w| + 17w^2 + |3w+6w^2| + |1/2 + w + 4w^2|\}\, E^k$$

$$= A_1(w)\, E^k.$$

It is obvious that $A_1(w) < 1$ if (6.29) holds. In a similar way, one can also prove the lemma.

In fact, the constant $C(w)$ is the infinity norm of a 14 x 14 w-matrix $A(w)$. This matrix is defined explicitly by

(6.37) $A(w) :=$

$$\left|\begin{array}{cccccccccccccc}
-b & b & 0 & a & 0 & 0 & -b & b & 0 & 0 & 0 & 0 & 0 & 0 \\
b & -b & 0 & a & 0 & 0 & b & -b & 0 & 0 & 0 & 0 & 0 & 0 \\
b & 0 & d & d & 0 & b & c & b & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & b & c & b & 0 & d & d & 0 & 0 & b & 0 & 0 & 0 \\
0 & b & 0 & d & d & 0 & b & c & b & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -b & b & 0 & a & 0 & 0 & -b & b & 0 & 0 & 0 & 0 \\
0 & 0 & b & -b & 0 & a & 0 & 0 & b & -b & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -b & b & 0 & 0 & c & 0 & 0 & -b & b & 0 & 0 \\
0 & 0 & 0 & b & -b & 0 & 0 & c & 0 & 0 & b & -b & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & b & c & b & 0 & d & d & 0 & b & 0 \\
0 & 0 & 0 & b & 0 & 0 & d & d & 0 & b & c & b & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & b & c & b & 0 & d & d & 0 & b \\
0 & 0 & 0 & 0 & b & 0 & -b & b & 0 & 0 & a & 0 & -b & b \\
0 & 0 & 0 & 0 & 0 & 0 & b & -b & 0 & 0 & a & 0 & b & -b
\end{array}\right|.$$

where, $a = 1/2$, $b = w$, $c = 1/2 + 3w$ and $d = -3w$.

Remark. Weaker C^0 conditions can also be obtained by applying the same technique for more subdivision levels. Since we are interested in the C^1 conditions, we will not discuss this further.

6.4.4. C¹ Analysis-the Directional Divided Difference Analysis

In order to prove the C^1 property of the limit surface F, the ***Cross Differences of the Directional Divided Differences*** (*CDD* for short) of the control net should be investigated. This process is similar to the ***Divided Difference*** analysis as described in [48,49,50].

First, we give the definition of the *CDD* at the k-th level along one direction (there are three such directions in all, see Figure 6.4). Since the three directions are mutually symmetric, only one of them is studied here. By symmetry, the results are also true for the other directions.

The *CDD* along direction 1 and 2 at level k is defined by

$$(6.38) \quad C^k_{i,j} := 2^k \Delta_1 \Delta_2 F^k_{i,j} \quad \text{for all } i, j, k.$$

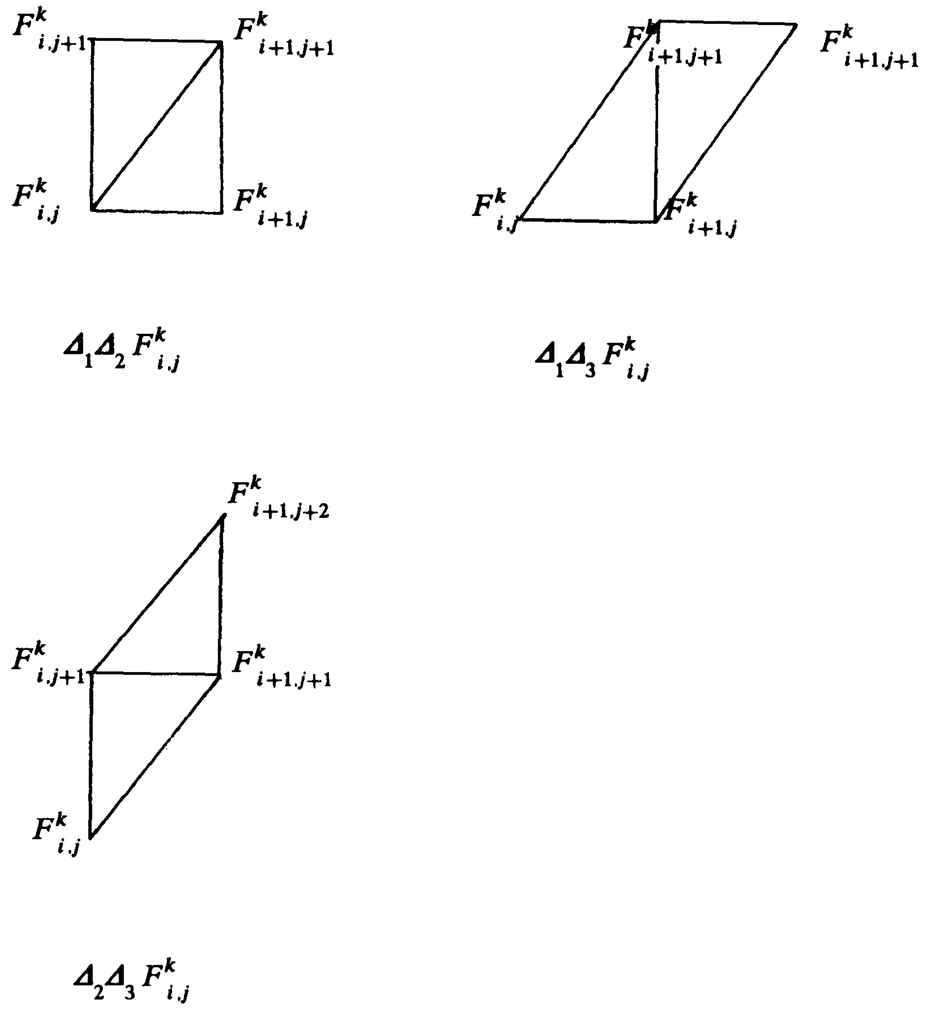


Figure 6.4. The *CDD*.

From (6.27), it can be shown that the *CDD* values satisfy the uniform subdivision scheme defined by the following refinement equations:

$$(6.39)\quad \left|\begin{aligned}
C^{k+1}_{2i,2j} &= 2wC^k_{i-1,j} - 4wC^k_{i-1,j-1} + (1+8w)C^k_{i,j} + 2wC^k_{i,j-1}\\
C^{k+1}_{2i+1,2j} &= 2wC^k_{i-1,j-1} - 8wC^k_{i,j} - 2wC^k_{i,j-1} + 2wC^k_{i+1,j+1} - 2wC^k_{i+1,j}\\
C^{k+1}_{2i,2j+1} &= -2wC^k_{i-1,j} + 2wC^k_{i-1,j-1} - 2wC^k_{i,j+1} - 8wC^k_{i,j} + 2wC^k_{i+1,j+1}\\
C^{k+1}_{2i+1,2j+1} &= 2wC^k_{i,j+1} + (1+8w)C^k_{i,j} - 4wC^k_{i+1,j+1} + 2wC^k_{i+1,j}.
\end{aligned}\right.$$

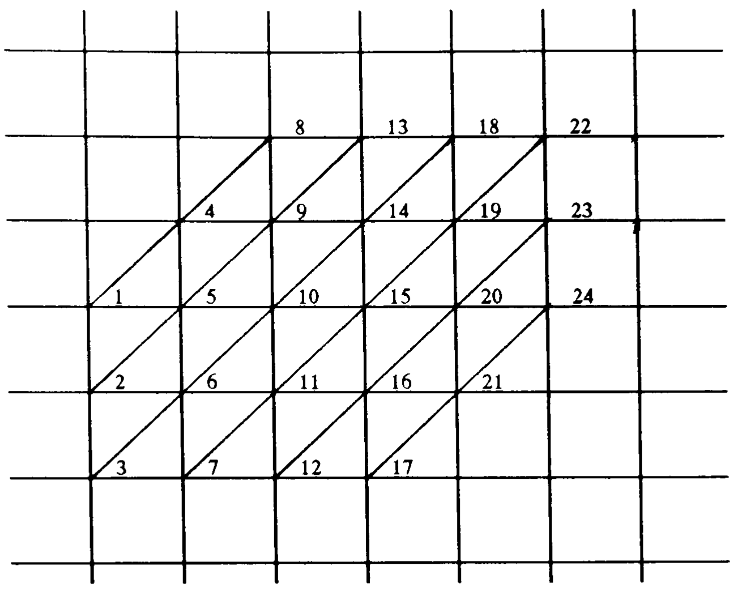


Figure 6.5. The *CDD* scheme.

Let $C^k := (C^k_1, C^k_2, ..., C^k_{24})^t$ denote a local labelling of the *CDD* values defined on the u–v plane centred on the point with index 15 as shown in Figure 6.5. Thus C^k_{15} denotes $C^k_{i,j}$ on the k-th level mesh and C^{k+1}_{15} denotes $C^{k+1}_{2i,2j}$ on the k+1st level mesh. Then, from (6.39), we obtain

(6.40) $C^{k+1} = B(w)\, C^k_,$

where, $B(w)$ is a w-matrix of order 24 x 24 defined by (6.41), where, $a = 2w$, $b = -2w$, $c = 4w$, $d = -4w$, $e = -8w$ and $f = 1+8w$ and the omitted elements in (6.41) are zeros.

(6.41) $B(w) :=$

$$\left|\begin{array}{ccccccccccccccccccccccc}
0&0&0&a&f&0&0&0&d&a&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&a&&&e&b&&&a&b&&&&&&&&&&&&&\\
0&&&&a&f&&&&d&a&&&&&&&&&&&&\\
0&&&a&d&&&&f&a&&&&&&&&&&&&&\\
0&&&&b&a&&&b&e&&&&&a&&&&&&&&\\
0&&&&a&d&&&&f&a&&&&&&&&&&&&\\
0&&&&&b&a&&&b&e&&&&&a&&&&&&&\\
0&&&&&&&a&f&&&&&d&a&&&&&&&&\\
0&&&&a&&&&e&b&&&&a&b&&&&&&&&\\
0&&&&&&&&a&f&&&&&d&a&&&&&&&\\
0&&&&&a&&&&e&b&&&&a&b&&&&&&&\\
0&&&&&&&&&a&f&&&&&d&a&&&&&&\\
0&&&&&&&a&d&&&&&f&a&&&&&&&&\\
0&&&&&&&&b&a&&&&b&e&&&&a&&&&\\
0&&&&&&&&a&d&&&&&f&a&&&&&&&\\
0&&&&&&&&&b&a&&&&b&e&&&&a&&&\\
0&&&&&&&&&a&d&&&&&f&a&&&&&&\\
0&&&&&&&&a&&&&&e&b&&&a&b&&&&\\
0&&&&&&&&&&&&&a&f&&&&d&a&&&\\
0&&&&&&&&&a&&&&&e&b&&&a&b&&&\\
0&&&&&&&&&&&&&&a&f&&&&d&a&&\\
0&&&&&&&&&&&&&a&d&&&&f&a&&&\\
0&&&&&&&&&&&&&&b&a&&&b&e&&&a\\
0&&&&&&&&&&&&&&a&d&&&&f&a&&
\end{array}\right|$$

We have chosen the order of C^k of sufficient order such that

(6.42) $C^{k+2} = B^2(w)\, C^k$

contains all possible types of *CDD* terms at level $k+2$ if the relation is applied at every vertex at level k. Therefore, for $k = 0, 1, 2, \ldots$, we can easily obtain

(6.43) $C_d^{k+2} \leq \|B^2(w)\|\, C_d^k,$

where

(6.44) $C_d^k := max_{m \neq n}\, max_{i,j} \{|2^k \Delta_m \Delta_n F_{i,j}^k|\}.$

It will be shown later that our C^1 analysis is mainly based on this relation about the *CDD*. Further studies of $B(w)$ leads to the following result.

Lemma 6.13. The iteration matrix $B(w)$ has the properties:

(6.45) $\quad \|B(w)\| \geq 1 \quad$ for all w,

and

(6.46) $\quad \|B^2(w)\| < 1 \quad$ for $\; -1/12 < w < 0.$

Proof. From (6.41), we have

(6.47) $$\| B(w)\| \geq |2w| + |1+8w| + |-4w| + |2w|$$

$$= |8w| + |1+8w| \geq 1.$$

Furthermore, multiplying $B(w)$ by itself, numerical results indicate that (6.46) holds. To show that $-1/2$ is the exact lower bound of w, the explicit form of $\|B^2(w)\|$ (which is a piecewise quadratic of w) is calculated for for w near $-1/12$. Thus, for w near $-1/12$,

(6.48) $$\|B^2(w)\| = |-20w^2| + |4w^2| + |48w^2| + |-2w-4w^2|$$

$$+ |-20w^2| + |-2w-4w^2| + |-8w^2| + |4w^2|$$

$$= -4w + 96w^2.$$

Hence, we have, when w is close to $-1/_{12}$,

(6.49) $$\|B^2(w)\| < 1$$

$$\Longleftrightarrow \quad -4w + 96w^2 < 1$$

$$\Longleftrightarrow \quad (-12w - 1)(-8w + 1) < 0$$

$$\Longleftrightarrow \quad -1/_{12} < w < 0.$$

This completes the proof.

Remark. The explicit form of $\|B^2(w)\|$ can be obtained since we know $B(w)$ is given by (6.41).

Now we study the ***Directional Divided Difference*** function of the control net. As mentioned previously, due to symmetry, we need only to investigate the *divided differences* in one direction.

Let d_1^k be the piecewise linear interpolant to the divided difference data $\{2^k \Delta_1 F_{i,j}^k\}$ at the k-th level diadic points. Then, the following *Lemma* 6.14 and *Theorem* 6.15 will show why the ***Cross Differences of the Directional Divided Differences*** are preferred to the ordinary *differences of divided differences* along one direction.

Lemma 6.14. Suppose that $-1/_{12} < w < 0$, then for $k = 0, 1, 2, 3, \ldots,$ we have

(6.50) $\|d_1^{k+1} - d_1^k\| < (1+16|w|)\, C_d^k.$

Proof. The result comes from the direct calculation of the butterfly scheme and a proper arrangement of the terms in order to obtain the desired result. The factor $1+16|w|$ is not important to the analysis which will be shown clearly in our analysis later. For example, by definition we can obtain the following estimates, where the parameters $(u_i^k, v_j^k) := (i2^{-k}, j2^{-k})$.

(6.51)
$$\begin{aligned} d_1^{k+1}(u_i^k, v_j^k) - d_1^k(u_i^k, v_j^k) &= \{F^{k+1}_{2i+1,2j} - F^{k+1}_{2i,2j}\}2^{k+1} - \{F^k_{i+1,j} - F^k_{i,j}\}2^k \\ &= 2^k\{2F^{k+1}_{2i+1,2j} - F^k_{i,j} - F^k_{i+1,j}\} \\ &= 2^k\,\{2wF^k_{i,j+1} - 4wF^k_{i+1,j+1} + 2wF^k_{i+2,j+1} \\ &\qquad + 2wF^k_{i-1,j-1} - 4wF^k_{i,j-1} + 2wF^k_{i+1,j-1}\}. \end{aligned}$$

By the definition of C_d^k, see (6.44), we have

(6.52) $|d_1^{k+1}(u_i^k, v_j^k) - d_1^k(u_i^k, v_j^k)|$

$$\begin{aligned} &\le |2w|\,\{|F^k_{i,j+1} - F^k_{i+1,j+1} - F^k_{i,j} + F^k_{i+1,j}| \\ &\quad + |F^k_{i,j} - F^k_{i+1,j} - F^k_{i+1,j+1} + F^k_{i+2,j+1}| \\ &\quad + |F^k_{i-1,j-1} - F^k_{i,j-1} - F^k_{i,j} + F^k_{i+1,j}| \\ &\quad + |F^k_{i,j} - F^k_{i+1,j} - F^k_{i,j-1} + F^k_{i+1,j-1}|\,\} \end{aligned}$$

$$\leq 8|w|C_d^k.$$

In the same way, we can obtain

$$(6.53) \qquad |d_1^{k+1}(u_{2i+1}^{k+1}, v_{2j}^{k+1}) - d_1^k(u_{2i+1}^{k+1}, v_{2j}^{k+1})| \leq (1+8|w|)C_d^k,$$

$$(6.54) \qquad |d_1^{k+1}(u_{2i}^{k+1}, v_{2j+1}^{k+1}) - d_1^k(u_{2i}^{k+1}, v_{2j+1}^{k+1})| \leq (1/2+16|w|)C_d^k,$$

$$(6.55) \qquad |d_1^{k+1}(u_{2i+1}^{k+1}, v_{2j+1}^{k+1}) - d_1^k(u_{2i+1}^{k+1}, v_{2j+1}^{k+1})| \leq (1/2+16|w|)C_d^k.$$

Therefore, we have (since d_1^{k+1} and d_1^k are piecewise linear functions):

$$(6.56) \qquad ||d_1^{k+1} - d_1^k|| \leq \max\{\, 8|w|, 1+8|w|, 1/2+16|w| \,\} C_d^k$$

$$< (1+16|w|)C_d^k.$$

This completes the proof.

From (6.43), *Lemma*s 6.13 and 6.14, we can conclude:

<u>**Theorem**</u> 6.15. The ***Directional Divided Difference*** function sequence $\{d_1^k\}$ is a *Cauchy* sequence in C^0, thus it converges uniformly to a continuous function $d_1(u,v)$ if

$$(6.57) \qquad -1/12 < w < 0.$$

This result will be used to prove the C^1 convergence property of the scheme.

6.4.5. C^1 Convergence of the Butterfly Scheme

We can now prove our main result about the smooth convergence of the butterfly scheme.

Theorem 6.16. The butterfly scheme produces C^1 surfaces if (6.57) holds, that is, if $-1/12 < w < 0$.

Proof. The proof of the theorem comes directly from *Theorem* 6.15, the symmetry of the butterfly scheme and the following lemma.

Lemma 6.17. If the *directional divided difference* function sequence $\{d_\alpha^k\}$ converges uniformly to a C^0 function d_α, then the butterfly scheme produces a C^1 function $F(u,v)$. Furthermore, we have

$$(6.58) \qquad D_\alpha F(u,v) = d_\alpha(u,v),$$

where, D_α is the *directional derivative operator* along the direction α, $\alpha \in \{(0,1), (1,0)\}$.

Proof. Without loss of generality, we assume that $\alpha = (1,0)$, the u-axis direction and that the initial data is the cardinal data, that is,

$$(6.59) \qquad F^0_{i,j} := \delta_{i,0}\,\delta_{j,0}.$$

Then, $\boldsymbol{D}_{\boldsymbol{\alpha}} = \boldsymbol{D}_{(1,0)}$ and the function $d_1(u,v)$ has a local support contained in $[-3,3]^2$. Moreover, for $k = 0, 1, 2, \ldots$, we define,

$$(6.60) \qquad G^k(u,v) := \int_a^u d_1^k(x,v)\, dx,$$

where, $a := -3$, which is a proper point outside the support of the function $d_1(u,v)$. Hence, from the definition of the uniform convergence and the local support properties of the scheme, we have

$$(6.61) \qquad \boldsymbol{Lim}_{k\to\infty} G^k(u,v) = \int_a^u \boldsymbol{Lim}_{k\to\infty} d_1^k(x,v)\, dx = \int_a^u d_1(x,v)\, dx$$

$$:= G(u,v).$$

Hence, $G(u,v)$ is differentiable with respect to u. We can also prove that $\{F^k(u,v)\}$ converges uniformly to $G(u,v)$. This means that $F(u,v)$, the limit of the control polyhedrons, is also differentiable with respect to u. The convergence can be shown by the following inequalities, where, $u_i^k := i2^{-k}$,

$$(6.62) \qquad \|F^k(u,v) - G(u,v)\| \le \| F^k(u,v) - G^k(u,v)\| + \| G^k(u,v) - G(u,v)\|$$

The right hand side can be estimated respectively by:

$$(6.63) \qquad \|F^k(u,v) - G^k(u,v)\| = \| \int_a^u \boldsymbol{D}_{\boldsymbol{\alpha}}(F^k(x,v)\, dx - \int_a^u d_1^k(x,v)\, dx)\|$$

$$\le (3-a)\, \boldsymbol{max}_i\, |d_1^k(u_i^k,v) - d_1^k(u_{i+1}^k,v)|$$

$$\longrightarrow 0 \text{ as } k \longrightarrow \text{infinity}.$$

By hypothesis,

(6.64) $\quad ||G^k(u,v) - G(u,v)|| \longrightarrow 0$ as k goes to infinity.

Thus, we complete the proof of the lemma.

Note that this lemma can easily be extended to any uniform subdivision scheme. In fact, in a similar way, we can prove the following theorem.

__Theorem__ 6.18. If two of the different *directional divided difference* function sequences $\{d^k_{\alpha_i}\}_{i=1,2}$, $\alpha_i \in \{(0,1), (1,0), (1,1)\}$, converge uniformly to C^0 functions $\{d_{\alpha_i}\}$, then the original scheme is a C^1 scheme. Furthermore, we have

(6.65) $\quad D_{\alpha_i}F(u,v) = d_{\alpha_i}(u,v),\ i = 1, 2.$

__Remark.__ Higher order continuity of the surfaces can also be analysed similarly by using ***Higher order Cross-Differences of Directional Divided Differences.***

6.5. Convergence of the 10-point Scheme

In this section, we generalize the convergence results about the butterfly scheme over uniform triangular polygons to the 10-point interpolatory subdivision scheme. For simplicity, the notations used in the previous section such as E^k (6.31) and C_d^k (6.44) etc. will also be used in this section.

6.5.1. C^0 Convergence of the scheme

Suppose, like in the analysis of the butterfly scheme, that the initial data (real numbers) is given on the uniform grid which are denoted by $\{F_{i,j}^k\}$ for all $(i,j) \in \mathbf{Z}^2$. Then, the 10-point subdivision scheme is defined by (6.3) with masks given by (6.9). It can also be described in a similar form as (6.27).

The ***Difference Operators*** $\{\Delta_i\}$ and the ***Cross Differences of Divided Differences*** $\{C_{i,j}^k\}$ will be introduced as in (6.28) and (6.38) respectively. Consequently, the iteration matrices $A(w_1,w_2,w_3)$ and $B(w_1,w_2,w_3)$, which are similar to the matrices $A(w)$ and $B(w)$ defined by (6.37) and (6.41), can also be introduced.

We now prove some of the C^0 convergent results about the 10-point *ISS*. In particular, conditions on the parameters of the scheme are given explicitly to quarantee the generation of smooth surfaces.

Theorem 6.19. The 10-point interpolatory scheme is a C^0 scheme if

$$(6.66)\quad \left|\begin{array}{l} |1/2 - 2w_1 - w_2| + 2|w_1| + 2|w_2| + 2|w_3| + |w_1 - w_3| < 1 \\ 4|w_1| + 2|w_2| + 2|w_3| < 1/2. \end{array}\right.$$

A sufficient symmetric solution to this condition is given by

(6.67) $$5|w_1| + 3|w_2| + 3|w_3| < 1/2.$$

A simple solution to (6.66) is provided by

(6.68) $$\left|\begin{array}{l} |w_1| < 1/30 \\ |w_2| < 1/18 \\ |w_3| < 1/18. \end{array}\right.$$

Proof. The proof comes from a direct estimate of the the differences of the control polyhedron at two adjacent levels. That is, the difference of P^{k+1} and P^k. It can be shown that the difference is bounded by the maximum difference, E^k, defined by (6.31). In fact, we have

(6.69) $$||F^{k+1} - F^k|| \leq \{4|w_1| + 2|w_2| + 2|w_3|\} E^k.$$

By expressing the *directional differences* defined by (6.28) recursively, we can show further that

(6.70) $$E^{k+1} \leq M(w_1,w_2,w_3) E^k, \quad \text{for all} \quad k = 0, 1, \dots$$

where

(6.71) $$M := \max \{|1/2 - 2w_1 - w_2| + 2|w_1| + 2|w_2| + 2|w_3| + |w_1 - w_3|,\ 1/2 + 4|w_1| + 2|w_2| + 2|w_3|\}.$$

Hence, the control polyhedron sequence $\{P^k\}$ is a *Cauchy* sequence and therefore converges to a continuous surface if (6.66) holds. This completes the proof.

Remark. The condition (6.66) is a very simple condition although better results can also be obtained by applying the same technique for more iteration levels.

If the parameters $\{w_i\}$ satisfy condition (6.11), then we can obtain the following C^0 convergent result about the cubic precision scheme:

Theorem 6.20. The cubic precision scheme is a C^0 scheme if:

(6.72) $\quad 1/2 < t < 37/64.$

Assuming $w_2 = -2w_1$ and by using the same techniques as in *Lemmas* 6.11 and 6.12, other sufficient C^0 conditions rather than condition (6.66) can be obtained. One of them is given by

(6.73) $\quad \| A^2(w_1, w_2, w_3) \| < 1,$

where, the iteration matrix $A(w_1,w_2,w_3)$ is defined by (6.74), where, $a := 1/2$, $b =: w_1$, $c =: 1/2+3w_1 - w_3$, $d =: -3w_1$, $e =: w_2 = -2w_1$, $f =: w_3$ and $g = w_1 - w_3 = b - f$.

(6.74) $A(w_1, w_2, w_3) :=$

$$\begin{vmatrix}
-b & b & -f & a & f & 0 & -b & b & 0 & 0 & 0 & 0 & 0 & 0 \\
b & -b & f & a & -f & 0 & b & -b & 0 & 0 & 0 & 0 & 0 & 0 \\
g & 0 & d & d & 0 & b & c & b & 0 & f & f & 0 & 0 & 0 \\
f & f & b & c & b & 0 & d & d & 0 & 0 & g & 0 & 0 & 0 \\
0 & g & 0 & d & d & 0 & b & c & b & 0 & f & f & 0 & 0 \\
0 & 0 & -b & b & -f & a & f & 0 & -b & b & 0 & 0 & 0 & 0 \\
0 & 0 & b & -b & f & a & -f & 0 & b & -b & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -b & b & 0 & -f & c & f & 0 & -b & b & 0 & 0 \\
0 & 0 & 0 & b & -b & 0 & f & c & -f & 0 & b & -b & 0 & 0 \\
0 & 0 & f & f & 0 & b & c & b & 0 & d & d & 0 & g & 0 \\
0 & 0 & 0 & g & 0 & 0 & d & d & 0 & b & c & b & f & f \\
0 & 0 & 0 & f & f & 0 & b & c & b & 0 & d & d & 0 & g \\
0 & 0 & 0 & 0 & b & f & -b & b & 0 & f & a & f & -b & b \\
0 & 0 & 0 & 0 & 0 & 0 & b & -b & 0 & f & a & -f & b & -b
\end{vmatrix}.$$

In next subsection, we will study the smooth convergence property of the scheme.

6.5.2. The C^1 Convergence of the Scheme

On applying the same techniques and analyses as used in section 6.4, we can obtain the following C^1 convergence results about the 10-point interpolatory subdivision scheme over uniform triangulations.

Lemma 6.21. Suppose

(6.75) $$w_2 = -2w_1.$$

Then, the *CDD* of the 10-point scheme satisfy the following recurrence relation (7.76), where, the parameters w and v are defined as

(6.76)
$$\left|\begin{array}{l}
C^{k+1}_{2i,2j} = 2wC^k_{i-1,j} -(4w-2v)C^k_{i-1,j-1} +2vC^k_{i,j+1} +(1+8w)C^k_{i,j} + 2wC^k_{i,j-1} \\
\qquad +2vC^k_{i+1,j+1} +2vC^k_{i+1,j} \\
C^{k+1}_{2i+1,2j} = (2w-2v)C^k_{i-1,j-1} - 8wC^k_{i,j} -(2w-2v)C^k_{i,j-1} \\
\qquad +(2w-2v)C^k_{i+1,j+1} -(2w-2v)C^k_{i+1,j} \\
C^{k+1}_{2i,2j+1} = -(2w+2v)C^k_{i-1,j} + (2w-2v)C^k_{i-1,j-1} - (2w+2v)C^k_{i,j+1} \\
\qquad - 8wC^k_{i,j} + (2w-2v)C^k_{i+1,j+1} \\
C^{k+1}_{2i+1,2j+1} = 2vC^k_{i-1,j} + 2v\, C^k_{i-1,j-1} + 2wC^k_{i,j+1} + (1+8w)C^k_{i,j} \\
\qquad + 2vC^k_{i,j+1} -(4w-2v)C^k_{i+1,j+1} +2wC^k_{i+1,j}
\end{array}\right.$$

(6.77) $w := w_1$ and $v := w_3$.

From this *Lemma*, we obtain

Lemma 6.22. Suppose (6.77) holds, then

(6.78) $C^{k+1} = B(w,v)\, C^k$,

where, $B(w,v)$ is a (w,v)-matrix of order 24 x 24 defined by (7.79) and the parameters are defined as: $a := 2w$, $b := -2w+2v$, $c := 2w-2v$, $d := -4w+2v$, $e := -8w$, $f := 1+8w$, $g := 2v$ and the omitted entries are zeros.

(6.79) $B(w,v)$:=

$$\left|\begin{array}{cccccccccccccccccccccccc}
g & g & 0 & a & f & g & 0 & 0 & d & a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & a & & & e & b & & & c & b & & & & & & & & & & & & & & \\
0 & g & g & & a & f & g & & & d & a & & & & & & & & & & & & & \\
0 & & & a & d & & & g & f & a & & & & g & g & & & & & & & & & \\
0 & & & & b & c & & & b & e & & & & & c & & & & & & & & & \\
0 & & & & a & d & & & g & f & a & & & & g & g & & & & & & & & \\
0 & & & & & b & c & & & b & e & & & & & c & & & & & & & & \\
0 & & & g & g & & & a & f & g & & & & d & a & & & & & & & & & \\
0 & & & & c & & & & e & b & & & & a & b & & & & & & & & & \\
0 & & & & g & g & & & a & f & g & & & & d & a & & & & & & & & \\
0 & & & & & c & & & & e & b & & & & c & b & & & & & & & & \\
0 & & & & & g & g & & & a & f & g & & & & d & a & & & & & & & \\
0 & & & & & & & a & d & & & & g & f & a & & & g & g & & & & & \\
0 & & & & & & & & b & c & & & & b & e & & & & c & & & & & \\
0 & & & & & & & & a & d & & & & g & f & a & & & g & g & & & & \\
0 & & & & & & & & & b & c & & & & b & e & & & & c & & & & \\
0 & & & & & & & & & a & d & & & & g & f & a & & & g & g & & & \\
0 & & & & & & & & c & & & & & e & b & & & c & b & & & & & \\
0 & & & & & & & & g & g & & & & a & f & g & & & d & a & & & & \\
0 & & & & & & & & & c & & & & & e & b & & & c & b & & & & \\
0 & & & & & & & & & g & g & & & & a & f & g & & & d & a & & & \\
0 & & & & & & & & & & & & & a & d & & & g & f & a & & g & g & \\
0 & & & & & & & & & & & & & & b & c & & & b & e & & & c & \\
0 & & & & & & & & & & & & & & a & d & & & g & f & a & & g & g
\end{array}\right|.$$

Here, as in section 6.4.5, we have chosen the order of C^k of sufficient order such that

$$(6.80) \qquad C^{k+2} = B^2(w,v)C^k$$

contains all possible types of *CDD* terms at level $k+2$ if the relation is applied at every vertex at level k. Therefore, for $k = 0, 1, 2, \ldots$, we can easily obtain

$$(6.81) \qquad C_d^{k+2} \leq \|B^2(w,v)\| C_d^k,$$

where, the C_d^k is defined by (6.44).

Lemma 6.23. The iteration matrix $B(w,v)$ has the following properties:

(6.82) $\quad \| B(w,v) \| \geq 1$ for all $(w,v) \in \mathbb{R}^2$

and

(6.83) $\quad \| B^2(w,v) \| < 1$, $w < 0$ and w and v are *sufficiently small.*

More explicitly, by using computer experiments, we obtain an explicit condition for (6.83):

(6.84) $\quad (w, v) \in \Omega_I.$

The region Ω_I is depicted in Figure 6.6.

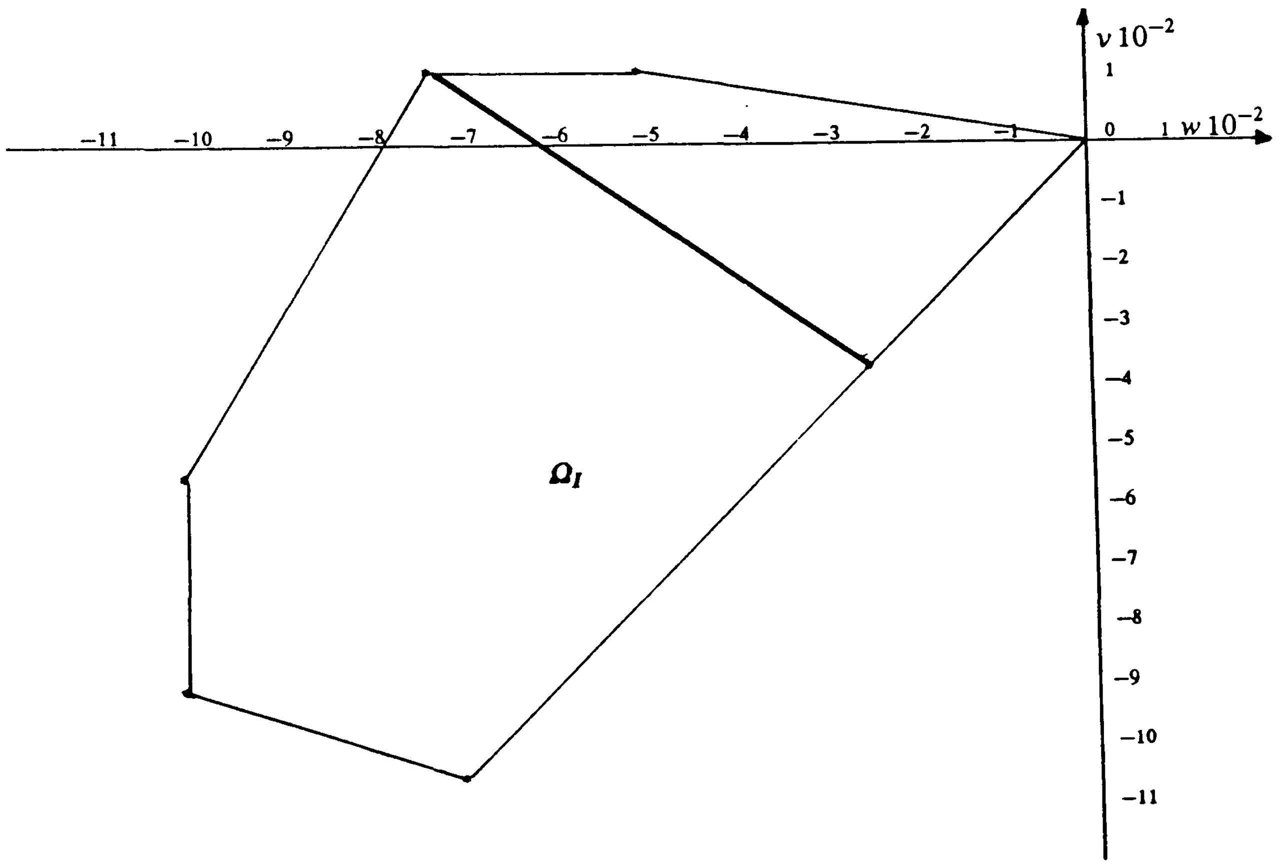


Figure 6.6. The C^1 convergence region Ω_I.

To prove the C^1 property of the scheme, the ***Directional Divided Differences*** of the scheme are to be investigated. As a result, the following lemma is needed. The proof of this lemma is the same as that of *Lemma* 6.14.

Lemma 6.24. Suppose that condition (6.75) holds, then, for $k = 0, 1, 2, 3, \ldots$, we have

$$|d_1^{k+1} - d_1^k| \leq (1+16|w| + 16|v|)\, C_d^k. \tag{6.85}$$

From *Lemmas* 6.22, 6.23 and 6.24 and *Theorem* 6.20, we can conclude our C^1 convergence about the 10-point scheme:

Theorem 6.25. The 10-point interpolatory scheme produces C^1 surfaces over uniform triangulations provided that conditions (6.75) and (6.84) hold.

Remark 1. This condition is only a sufficient condition for the scheme to produce smooth surfaces. Better conditions may be obtained by studying the recurrence relation (6.76) at more levels.

Remark 2. Condition (6.75) is used to quarantiee the existence of the recurrence relation of the ***CDD*** (6.76). Thus this condition is vital to our analysis.

To end this subsection, we give an explicit approximate solution to the sufficient C^1 condition (6.83).

An simple approximation to Ω_I is Ω'_1, which is small region within Ω_I bounded by four straight line segments $\{l_i\}$, $i = 1, 2, 3, 4$. These lines are given explicitly by:

$$(6.86) \quad \left|\begin{array}{ll} l_1: & w + 7v = 0 \\ l_2: & 8(w + 0.07) - 3(v - 0.01) = 0 \\ l_3: & (w + 0.10) + (v + 0.07) = 0 \\ l_4: & 10w - 7v = 0. \end{array}\right.$$

So, an easy solution to (6.83) is that w and v satisfy the following linear inequalities:

$$(6.87) \quad \left|\begin{array}{r} w \neq 0, \quad w + 7v \leq 0 \\ 8(w + 0.07) - 3(v - 0.01) \geq 0 \\ (w + 0.10) + (v + 0.07) \geq 0 \\ 10w - 7v \leq 0. \end{array}\right.$$

The region Ω'_1 is depicted in Figure 6.6*a*.

For the cubic precision scheme, where, $w = t - 9/16$ and $v = 1/2 - t$, condition (6.83) is satisfied if the shape control parameter t is chosen such that

$$(6.88) \quad 49/100 \leq t \leq 54/100.$$

This condition is just the thick lines within the region Ω_I and Ω'_1 depicted in Figure 6.6 and Figure 6.6*a* respectively.

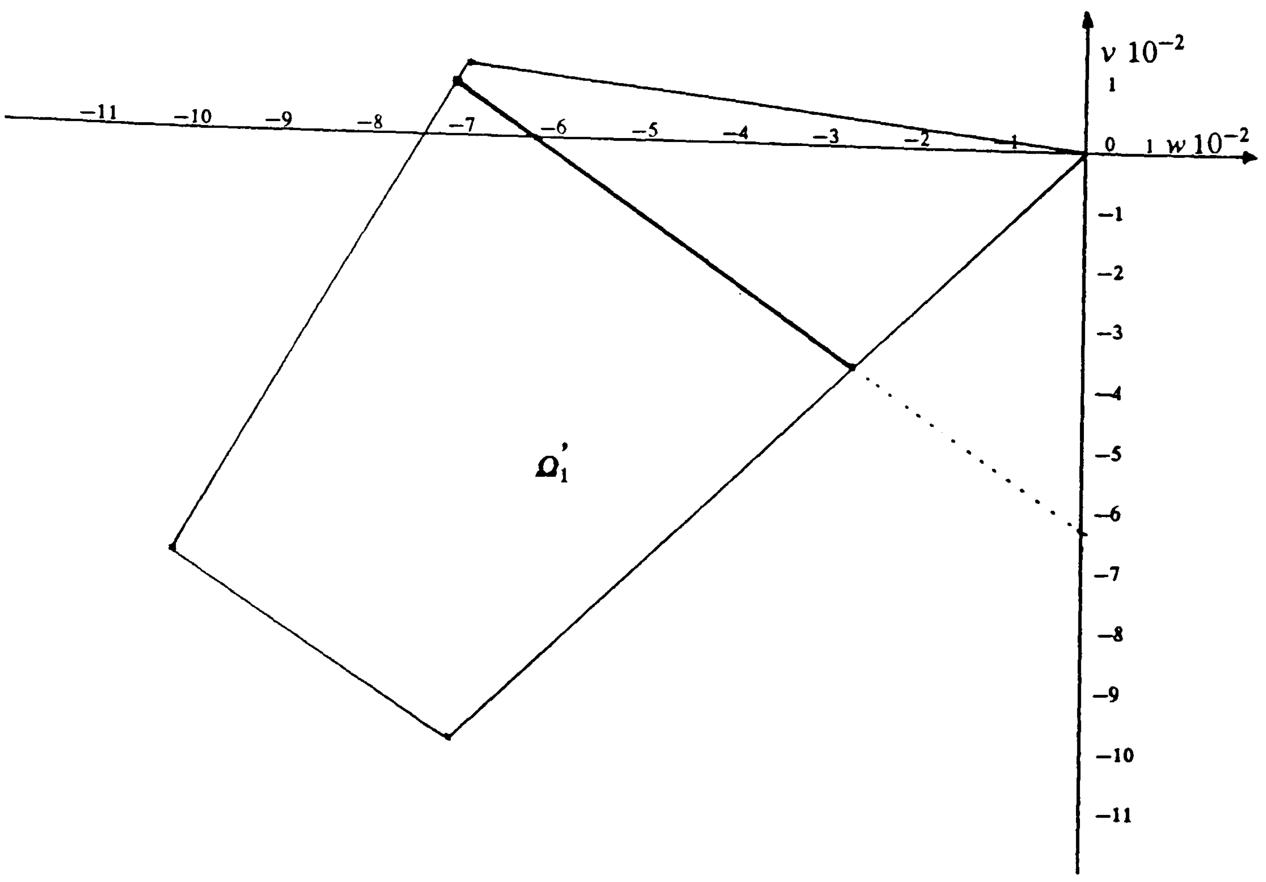


Figure 6.6*a*. The C^1 convergence region Ω_1'.

Some graphic examples produced by the scheme with different parameters will be given at the end of the this Chapter.

6.6. The 10-point Scheme over Non-uniform Triangulations

In this section, we study the 10-point scheme over non-uniform triangulations. Our main result is that the limit surface is smooth even at the extraordinary points provided that the scheme is modified properly at these points. In particular, these results are valid for the butterfly scheme.

The C^0 and C^1 analyses of the scheme here are different from the previous analyses of the scheme over uniform data. In fact, the analysis to be presented here is an extraordinary point analysis. The ***Block-Circulant Matrix*** theory is used here. This technique is quite suitable for the non-uniform analysis.

6.6.1. Generalization of the Scheme to Arbitrary Triangulations

In section 6.5, we studied the 10-point scheme over uniform triangulations in detail in which the uniform binary parametrization is used. However, since non-uniform triangular control polyhedrons often arise in practice, it is significant to investigate the behaviour of the scheme over non-uniform triangulations.

From its construction, we know that the scheme can be used to generate surfaces over arbitrary triangular networks. This can be done by introducing some local schemes only at the *Extraordinary points* (*E-point*) so that the *E-points* can be isolated by locally regular data. An *E-point* is a control point to which N edges ($N \neq 6$) of the control polyhedron incident. Otherwise, the vertex is called an ordinary point.

Depending on the local topology (more explicitly, the valances of the *E-points*), the modified 10-point scheme is defined as follows. At any ordinary point, the 10-point scheme is applied. However, at the near extraordinary points, some local schemes are used which are hoped to produce smooth surfaces. For each recursion, the scheme refines the control polyhedron but does not introduce any more *E-points*. Hence, the *E-points* can be isolated by locally regular data. Thus, the convergence analysis of the modified scheme becomes an extraordinary point analysis since we know that the scheme produces C^1 surfaces everywhere except at the *E-points*. Consequently, we will focus our analysis on these extraordinary points.

For simplicity, we assume that at regular points the parameter $\{w_i\}$ satisfy condition (6.12), that is, the scheme reduces to the *butterfly scheme*. However, the results are still true for the 10-point scheme with parameters $\{w_i\}$ satisfying (6.75) and (6.83).

The details of the scheme at an *E-point* will be given in the next subsection.

For the purpose of our analysis, the following notations are used throughout this section.

n: the indicator of the *E-point*, $(n+1)$ is the valency of the point, $n = 2, 3, \ldots$;

k: the subdivision level indicator, $k = 0, 1, 2 \ldots$;

i: a cyclic indicator, $i = 0, 1, 2, \ldots, n-1, n$;

$V, P_i, Q_i, R_i \ldots$: the control points near an extraordinary point, say, vertex V at level k, note, $V^{k+1} = V^k = V$ for all k;

p_i, q_i, r_i, ...: the refined control points near an extraordinary vertex V (at level $(k+1)$);

A: the local subdivision matrix (square) of order $3(n+1)+1$;

$\{\lambda_i, v_i\}$: the eigenvalues and their corresponding eigenvectors (generalized eigenvectors) of A;

A_i, $i = 0, 1, 2, \dots n$: matrices of order 3 x 3;

C_i, $i = 0, 1, 2, \dots (n-1), n$: basic subdivision matrices of order 3 x 3;

F^k: control point vector of length $3(n+1)+1$ at level k which will be defined explicitly in the context later;

w_i, w, t: (local) shape control parameters.

6.6.2. Formulation of the Scheme at an Extraordinary Point

The local scheme at an extraordinary point is constructed according to its valance. Before describing the modified scheme, we introduce some conventions. In the following formulae, the index i is a cyclic integer in the range $i = 0, 1, 2, \dots, n-1, n$, that is, (6.89) $p_i := p_j$, if and only if $i = j \,\mathbf{mod}(n+1)$, $j = 0, 1, 2, \ldots, n$.

It is also assumed in *case I* scheme ($n = 2$) that the cubic precision parameters are used. That is

(6.90)
$$\left|\begin{array}{ll} w_1 := t - {}^9/_{16} \\ w_2 := -2t + {}^9/_8 = -2w_1 \\ w_3 := {}^1/_2 - t \\ w_4 := {}^1/_2 - 2w_1 - w_2 - w_3 = t \\ t: \quad \text{shape parameter (local).} \end{array}\right.$$

Note that, for any real number t,

(6.91) $4w_1 + 2w_2 + 2w_3 + 2w_4 = 1.$

For simplicity, we assume also, without loss of generality, that the initial data is locally uniform except one extraordinary point V and that P_i, Q_i and R_i denote the control points at level k and p_i, q_i and r_i denote the corresponding refined control points. In fact, this situation can be achieved locally after the first subdivision.

Case I. $n = 2$, *valency* $= 3$.

In this case, there are several alternative schemes that can be used. One of them is described by the following (Figure 6.7). For $i = 0, 1, 2, \ldots, n-1, n$,

$$(6.92)\quad \begin{aligned} p_i &:= w_4 V + (w_1+w_4) P_i + (w_1+w_3) Q_i + w_2 R_i + w_1 P_{i+1} + w_3 R_{i+1} \\ &\quad + w_1 P_{i-1} + w_2 R_{i-1} \\ q_i &:= P_i \\ r_i &:= w_2 V + w_4 P_i + w_1 Q_i + w_2 R_i + w_4 P_{i+1} + w_1 Q_{i+1} + w_3 R_{i+1} \\ &\quad + 2w_1 P_{i-1} + w_3 R_{i-1}. \end{aligned}$$

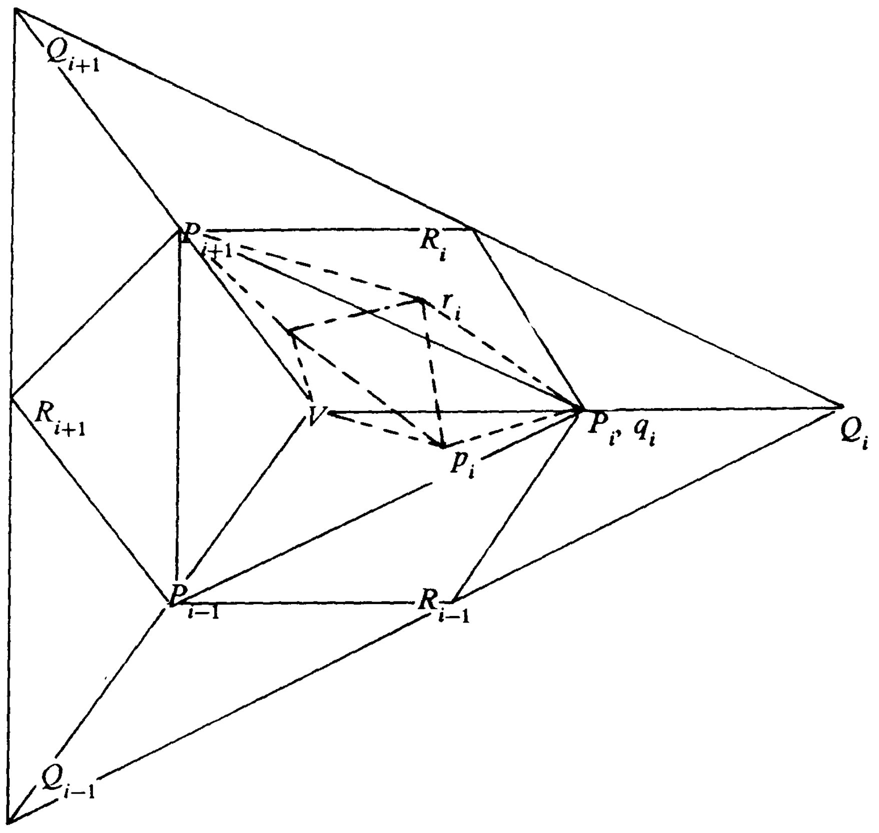


Figure 6.7. $n = 2$.

Case II. $n \geq 3$, *valency* ≥ 4.

In this case, the scheme is just the ***butterfly scheme***. That is, using the butterfly formula everywhere. Since in this case the scheme also produces C^1 surfaces (to be proved later), it is not necessary to construct more complicated schemes at the *E-points* although some other schemes may also be used. In fact, a cubic precision scheme can be constructed but the the coefficients of the formulae are quite complicated.

The scheme is like this. Applying the butterfly scheme near the extraordinary point V, we obtain the following subdivision formulae (Figure 6.8).

$$(6.93)\quad \left|\begin{aligned} p_i &= 1/2V + 1/2P_i + wR_i + w_2P_{i+1} + wP_{i+2} + w_2P_{i-1} + wR_{i-1} + w\,P_{i-2} \\ q_i &= P_i \\ r_i &= w_2V + 1/2P_i + wQ_i + w_2R_i + 1/2P_{i+1} + wQ_{i+1} + wP_{i+2} + wP_{i-1}, \end{aligned}\right.$$

where, w is the (local) tension parameter and

$$(6.94)\quad w_2 := -2w.$$

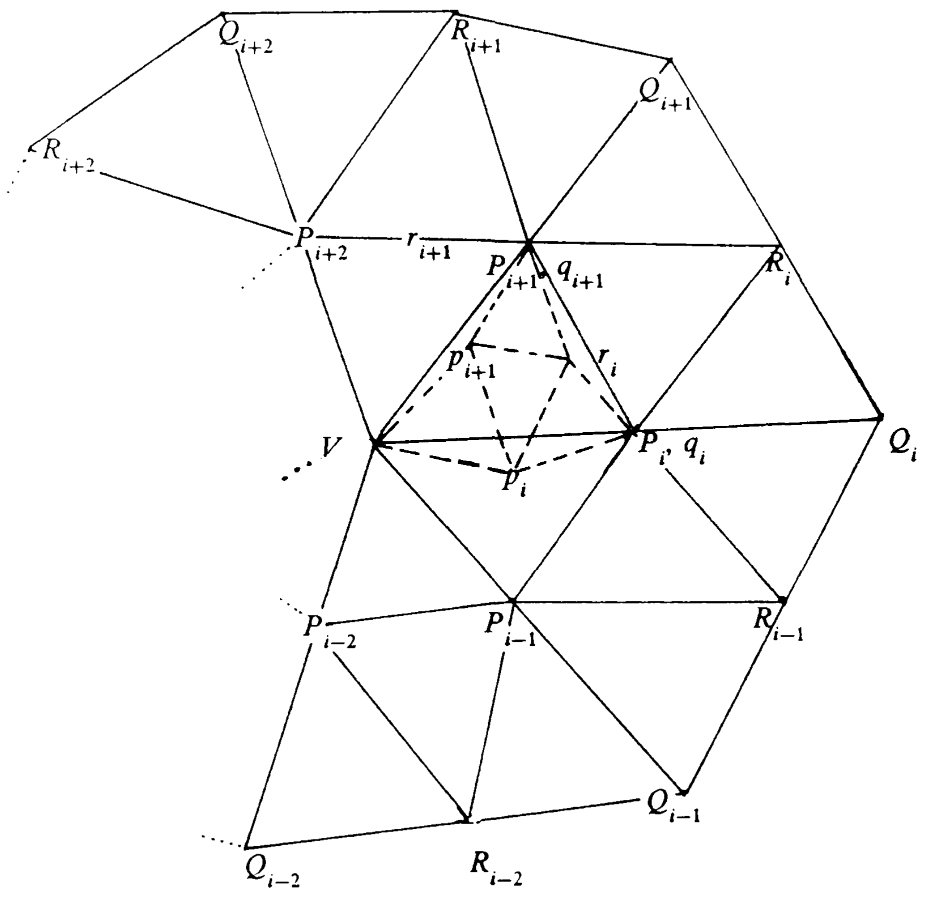


Figure 6.8. $n \geq 3$.

6.6.3. The Subdivision Matrix at the E-point

Writing (6.93) in a matrix form, we obtain:

$$(6.95)\quad \begin{vmatrix} p_i \\ q_i \\ r_i \end{vmatrix} = \begin{vmatrix} 1/2 & 0 & w \\ 1 & 0 & 0 \\ 1/2 & w & w_2 \end{vmatrix} \bullet \begin{vmatrix} P_i \\ Q_i \\ R_i \end{vmatrix} + \begin{vmatrix} w_2 & 0 & 0 \\ 0 & 0 & 0 \\ 1/2 & w & 0 \end{vmatrix} \bullet \begin{vmatrix} P_{i+1} \\ Q_{i+1} \\ R_{i+1} \end{vmatrix}$$

$$+ \begin{vmatrix} w & 0 & 0 \\ 0 & 0 & 0 \\ w & 0 & 0 \end{vmatrix} \bullet \begin{vmatrix} P_{i+2} \\ Q_{i+2} \\ R_{i+2} \end{vmatrix} + \begin{vmatrix} w_2 & 0 & w \\ 0 & 0 & 0 \\ w & 0 & 0 \end{vmatrix} \bullet \begin{vmatrix} P_{i-2} \\ Q_{i-2} \\ R_{i-2} \end{vmatrix}$$

$$+ \begin{vmatrix} w & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} \bullet \begin{vmatrix} P_{i-1} \\ Q_{i-1} \\ R_{i-1} \end{vmatrix} + \begin{vmatrix} 1/2 \\ 0.0 \\ w_2 \end{vmatrix} \bullet V$$

From this expression, we introduce the following basic matrices:

$$(6.96)\quad C_0 := \begin{vmatrix} 1/2 & 0 & w \\ 1 & 0 & 0 \\ 1/2 & w & w_2 \end{vmatrix}, \quad C_1 := \begin{vmatrix} w_2 & {}^{r} & 0 \\ 0 & 0 & 0 \\ 1/2 & w & 0 \end{vmatrix}, \quad C_2 = \begin{vmatrix} w & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix},$$

$$C_3 := \begin{vmatrix} w_2 & 0 & w \\ 0 & 0 & 0 \\ w & 0 & 0 \end{vmatrix}, \quad C_4 := \begin{vmatrix} w & 0 & 0 \\ 0 & 0 & 0 \\ w & 0 & 0 \end{vmatrix}$$

and the control point vectors:

(6.97) $\boldsymbol{F}^k := (V, P_0, Q_0, R_0, P_1, Q_1, R_1, , ..., P_n, Q_n, R_n)^t$

and

(6.98) $\boldsymbol{F}^{k+1} := (V, p_0, q_0, r_0, p_1, q_1, r_1, , ..., p_n, q_n, r_n)^t.$

Here, F^{k+1} and F^k are vectors of length $3(n+1)+1$. Thus, the subdivision process at the *E-point* can be written in a more compact form:

(6.99) $\boldsymbol{F}^{k+1} = \boldsymbol{A}\boldsymbol{F}^k,$

where, $\boldsymbol{A}$ is the *local subdivision matrix*. More explicitly, the matrix is given by

(6.100) $$\boldsymbol{A} := \begin{vmatrix} 1 & 0 \\ \boldsymbol{a} & \boldsymbol{A}' \end{vmatrix}$$

and $\boldsymbol{a}$ is a vector of length $3(n+1)$, and $\boldsymbol{A}'$ is a *block circulant matrix* defined by

(6.101) $\boldsymbol{A}' := \textit{B--circ}(\boldsymbol{A}_0, \boldsymbol{A}_1, \boldsymbol{A}_2, ..., \boldsymbol{A}_n)$

$$:= \begin{vmatrix} \boldsymbol{A}_0 & \boldsymbol{A}_1 & \boldsymbol{A}_2 & \boldsymbol{A}_3 & \cdots & \boldsymbol{A}_{n-1} & \boldsymbol{A}_n \\ \boldsymbol{A}_n & \boldsymbol{A}_0 & \boldsymbol{A}_1 & \boldsymbol{A}_2 & \cdots & \boldsymbol{A}_{n-2} & \boldsymbol{A}_{n-1} \\ \boldsymbol{A}_{n-1} & \boldsymbol{A}_n & \boldsymbol{A}_0 & \boldsymbol{A}_1 & \cdots & \boldsymbol{A}_{n-3} & \boldsymbol{A}_{n-2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{A}_2 & \boldsymbol{A}_3 & \boldsymbol{A}_4 & \boldsymbol{A}_5 & \cdots & \boldsymbol{A}_0 & \boldsymbol{A}_1 \\ \boldsymbol{A}_1 & \boldsymbol{A}_2 & \boldsymbol{A}_3 & \boldsymbol{A}_4 & \cdots & \boldsymbol{A}_n & \boldsymbol{A}_0 \end{vmatrix}$$

and $\{\boldsymbol{A}_i\}$ $i = 0, 1, 2, ..., n-1, n$ are some 3 by 3 matrices given explicitly below.

Similar results hold for $n = 2$ (*valency* $= 3$). In fact, in this case, we have

$$(6.102)\quad A_0 := \begin{vmatrix} w_1+w_4 & w_1+w_3 & w_2 \\ 1 & 0 & 0 \\ w_4 & w_1 & w_2 \end{vmatrix}, \qquad A_1 := \begin{vmatrix} w_1 & 0 & w_3 \\ 0 & 0 & 0 \\ w_4 & w_1 & w_3 \end{vmatrix},$$

$$A_2 := \begin{vmatrix} w_1 & 0 & w_2 \\ 0 & 0 & 0 \\ 2w_1 & 0 & w_3 \end{vmatrix}.$$

For $n = 3$, (*valency* $= 4$), $\{A_i\}$ are given by

$$(6.103)\quad \left|\begin{array}{lll} A_0 & := & C_0 \\ A_1 & := & C_1 \\ A_2 & := & C_2 + C_4 \\ A_3 & := & C_3. \end{array}\right.$$

For $n \geq 4$, $\{A_i\}$ are given by

$$(6.104)\quad \left|\begin{array}{llll} A_0 & := & C_0 & \\ A_1 & := & C_1 & \\ A_2 & := & C_2 & \\ A_i & := & 0, & \text{for} \quad i = 3, 4, 5, \ldots, n-2 \\ A_{n-1} & := & C_3 & \\ A_n & := & C_4. & \end{array}\right.$$

Now, we have constructed all the subdivision matrices $\{A_i\}$ upon which the properties of the limit surfaces depend. In the following subsections, we will study the convergent properties of the modified schemes at the *E-point*.

6.6.4. The Spectrum Analysis of the Subdivision Matrix

In order to study the C^0 and C^1 properties of the 10-point scheme over arbitrary triangulations, it is sufficient to prove that the limit surfaces of the scheme are C^0 or C^1 at the extraordinary points since the limit surfaces are C^1 everywhere else provided that the tension parameter w satisfies

(6.105) $-1/12 < w < 0.$

In much the same way as in Chapter 5, it can be shown that the eigen-properties of the subdivision matrix A play a very important role in the C^0 and C^1 analyses. Hence, we first study the eigen-properties of A. It should be stressed that its eigenvalues and their corresponding eigenvectors can be evaluated analytically since the matrix is a ***Block-Circulant-Matrix*** composed of 3 x 3 sub-matrices, therefore these eigenvalues are roots of cubic polynomials hence they can be obtained analytically.

Let the eigenvalues and their corresponding (generalized) eigenvectors of A be denoted by $\{\lambda_i, v_i\}$, where, $|\lambda_i| \geq |\lambda_{i+1}|$ for all i. Then, we can obtain the following result:

<u>Theorem</u> 6.26. The subdivision matrix A has the following properties:

(6.106) $\lambda_1 = 1, v_1 = (1,1,...,1)^t$ and

(6.107) $|\lambda_i| < 1, \quad \text{for all} \quad i = 2, 3, \ldots, 3n+3, 3n+4$

if

(6.108) $$\left| \begin{array}{lll} 0.3125 < t < 0.6000, & \text{for } n = 2 \\ -1/12 < w < 0, & \text{for } n \geq 3. \end{array} \right.$$

Furthermore, we have

(6.109) $0 < \lambda_2 = \lambda_3 < \lambda_1, \; |\lambda_i| < \lambda_2, \; i \geq 4$ and $\mathit{dim\ span}\{v_2, v_3\} = 2$

if

(6.110) $$\left| \begin{array}{lll} 0.5275 < t < 0.5500, & \text{for } n = 2 \\ -1/12 < w < 0, & \text{for } n \geq 3. \end{array} \right.$$

Proof. This theorem can be proved by direct evaluation.

From this theorem, we will establish our C^0 and C^1 convergence analyses in the next subsection.

Remark. The eigenvalue λ_2 is a double root of A and has two linearly independent eigenvectors v_2 and v_3. This can be shown clearly by using *Block-Circulant* matrix theory or *Fourier Transform* technique.

6.6.5. The Convergence Analysis

In this section, We will prove that the limit surface has tangent plane continuity at the *E-point*. Thus, the surface is smooth everywhere. Firstly, from *Theorem* 6.27, we can obtain a C^0 convergence result:

Theorem 6.27. The limit surface is C^0 if:

$$(6.111) \quad \left| \begin{array}{llll} 0.3125 < t < 0.6000, & \text{for } n = 2 \\ -1/12 < w < 0, & \text{for } n \geq 3. \end{array} \right.$$

Proof. The proof of this is almost the same as that of *Theorem* 5.4 in Chapter 5. The details are omitted here.

For the C^1 convergence, we have the following:

Theorem 6.28. The limit surface is C^1 if

$$(6.112) \quad \left| \begin{array}{llll} 0.5275 < t < 0.5500, & \text{for } n = 2 \\ -1/12 < w < 0, & \text{for } n \geq 3. \end{array} \right.$$

To prove this result, the following *Eigen-properties* of the subdivision matrix are needed:

Theorem 6.29. The limit surfaces of the interpolatory scheme is C^1 at an extraordinary point V, that is, it has a unique tangent plane at V, if the subdivision matrix $\boldsymbol{A}$ has properties:

$$(6.113)\quad \left|\begin{array}{ll} \text{(i).} & \lambda_1 = 1 \text{ is a simple eigenvalue and } \boldsymbol{v}_1 = (1, 1, 1, \dots, 1, 1)^t; \\ \text{(ii).} & 0 < \lambda_2 = \lambda_3 < 1, \quad \boldsymbol{dim\,span}\{\boldsymbol{v}_2, \boldsymbol{v}_3\} = 2; \\ \text{(iii).} & |\lambda_i| < \lambda_2, \; i = 4, 5, \dots, 3n+4. \end{array}\right.$$

It can also be shown that a necessary condition for the limit surface to have a unique tangent plane at the extraordinary point is:

$$(6.114)\quad \left|\begin{array}{ll} \text{(i).} & \lambda_1 = 1 \text{ is a simple eigenvalue and } v_1 = (1, 1, 1, \dots, 1, 1)^t; \\ \text{(ii).} & \text{there exists an integer } N_0 \geq 3 \text{, such that:} \\ & 0 < \lambda_2 = \lambda_3 = \dots = \lambda_{N0} < 1, \; \boldsymbol{dim\,span}(\boldsymbol{v}_2, \boldsymbol{v}_3, \dots, \boldsymbol{v}_{N0}) = 2; \\ \text{(iii).} & |\lambda_i| < \lambda_2, \text{ for } i > N_0. \end{array}\right.$$

<u>Proof.</u> We just prove the sufficient conditions. Suppose (113) is satisfied. So, from the subdivision relation (6.99), we have: for $k > 0$,

$$(6.115)\qquad \boldsymbol{F}^k = \boldsymbol{A}^k \boldsymbol{F}^0,$$

$$= V\boldsymbol{v}_1 + \lambda_2^k \alpha\, \boldsymbol{v}_2 + \lambda_2^k \beta\, \boldsymbol{v}_3 + O(\lambda_4/\lambda_2)^k,$$

where, for general data, $<\boldsymbol{\alpha}, \boldsymbol{\beta}> \neq 0$.

As in Chapter 5, we can prove that any well defined tangent plane of the limit surface near to V, say, at point Q_l, has the form

(6.116) $$\Pi^k(Q_i) = span\{\alpha, \beta\} + O(\lambda_4/\lambda_2)^k.$$

Hence, we have

(6.117) $$\Pi(V) := Lim_{k\to\infty} \Pi^k(Q_i) = span\{\alpha, \beta\}.$$

This means that all the tangent planes which are well defined near the *E-point* V converge to the unique plane $span\{\alpha, \beta\}$. That is, the limit surface has a tangent plane at V and the tangent plane varies continuously at the *E-point*. This completes the proof.

Remark 1. Since the scheme is interpolatory, *Theorem* 6.29 can also be proved by using *Directional Divided Difference* method or *adapted parametrization* technique.

Remark 2. The necessary conditions (6.114) can be easily proved. These necessary conditions are still true at regular points.

Remark 3. It can be shown that conditions $Lim_{k\to\infty} max_{i,j,m}\{|\Delta_m F^k_{i,j}|\} = 0$ and $Lim_{k\to\infty} max_{i,j,m\neq n}\{|2^k\Delta_m\Delta_n F^k_{i,j}|\} = 0$ are necessary conditions for the 10-point scheme to produce C^1 surfaces over uniform data. In fact, this is true for any uniform subdivision schemes.

6.7. Conclusions

In this Chapter, we have studied the subdivision algorithms based on triangulations. The following results are obtained.

1. The butterfly scheme produces C^1 surfaces over uniform triangular control nets provided that $-1/12 < w < 0$.

2. The 10-point interpolatory scheme is introduced and explicit sufficient conditions for it to produce C^0 and C^1 surfaces are also given. A simple sufficient C^1 condition is that the parameters w and v should lie in the polygonal region Ω'_1 depicted in Figure 6.6.

3. The cubic precision scheme produces smooth surfaces if the shape control parameter t satisfies $49/100 \leq t \leq 54/100$.

4. The cubic precision scheme is always recommended since its approximation order is four instead of two. Hence it might produce better results. Another reason for this is that it has the potential to produce even smoother surfaces [53*a*]. Our graphics also show that this scheme produces very nice surfaces.

5. The 10-point scheme over non-uniform data is investigated and it is proved that the limit surfaces are smooth everywhere provided that the parameters are chosen appropriately.

6. The method, ***Divided Difference and Cross Difference of Directional Divided Differences*** analyses, can also be used to study the higher order continuity of the surfaces generated by (uniform) subdivision algorithms. The only difference is that higher order ***Cross Differences of Directional Divided***

Differences should be studied and this process is much more complicated than that of our C^1 analysis.

7. The *Extraordinary Point* analysis is still valid for other subdivision algorithms.

6.8. Graphic Examples

Here, we present some graphic examples of the 10-point subdivision algorithm with different parameters. The surfaces are plotted by *Nichlet Drum Plotter at Brunel University, UK. 1988-1990.* The software used to produce surfaces on a rectangular grid and hence the triangulation along the (1,1) direction is unfortunately not displayed.

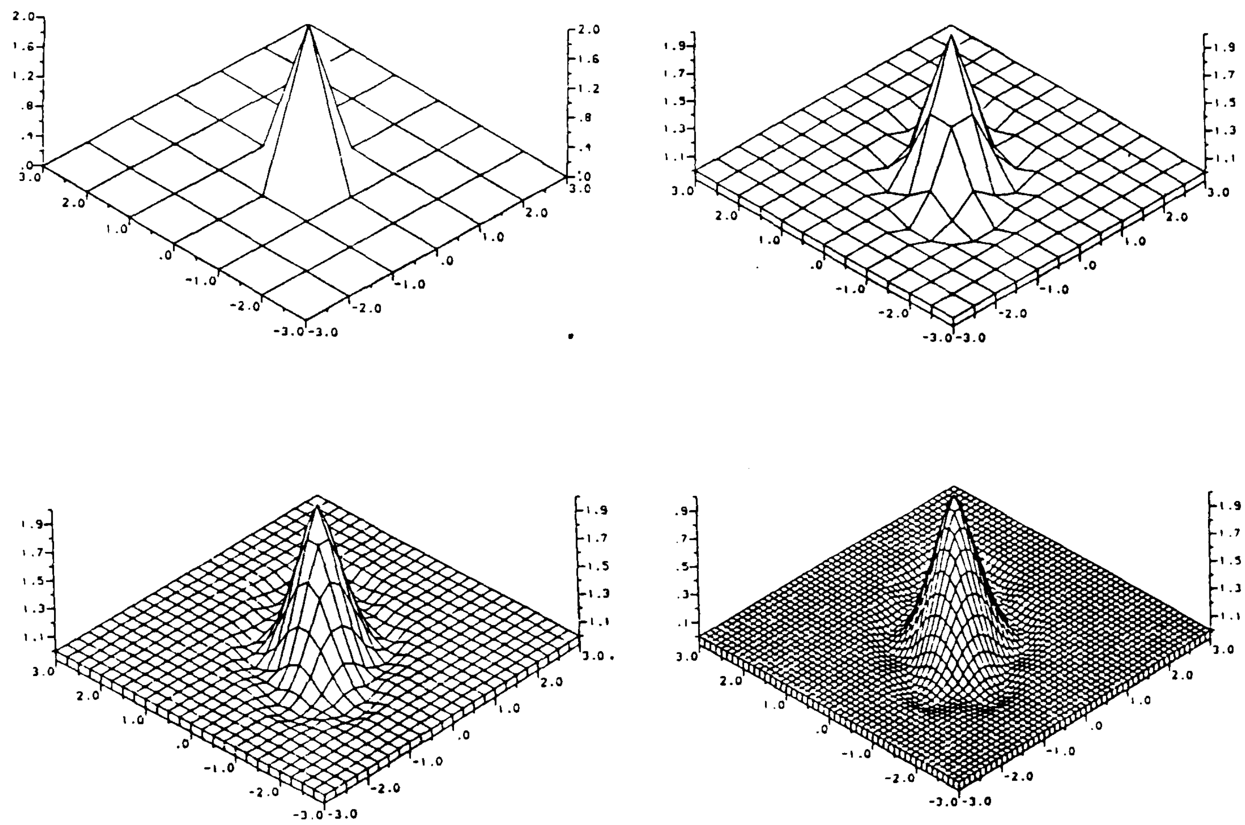


Figure 6.9. Butterfly scheme, the cardinal function, $w = -1/16$, $k = 0, 1, 2, 3$.

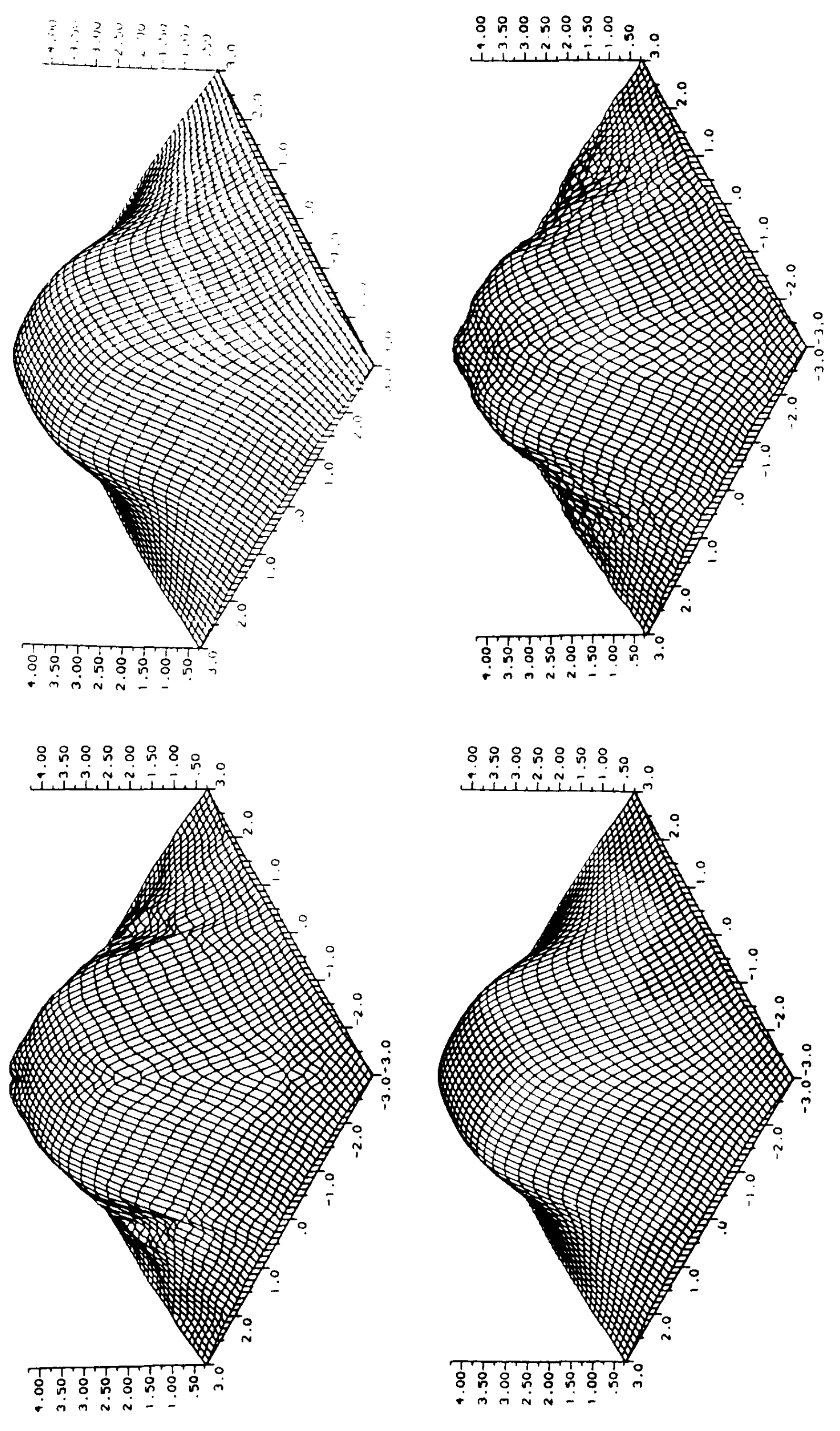


Figure 6.10. *Cubic precision scheme with* $t = 0.40, 0.50, 0.53, 0.62$, $k = 3$.

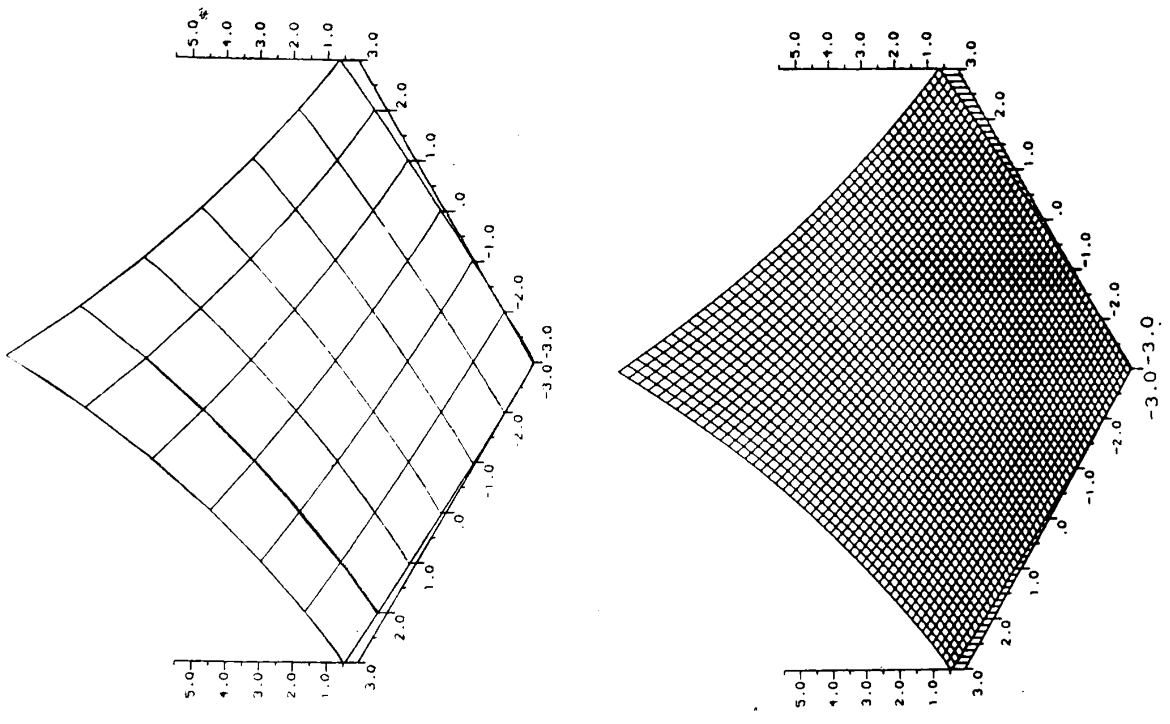


Figure 6.11. *Cubic precision scheme with* $t = 0.52$, $k = 3$.

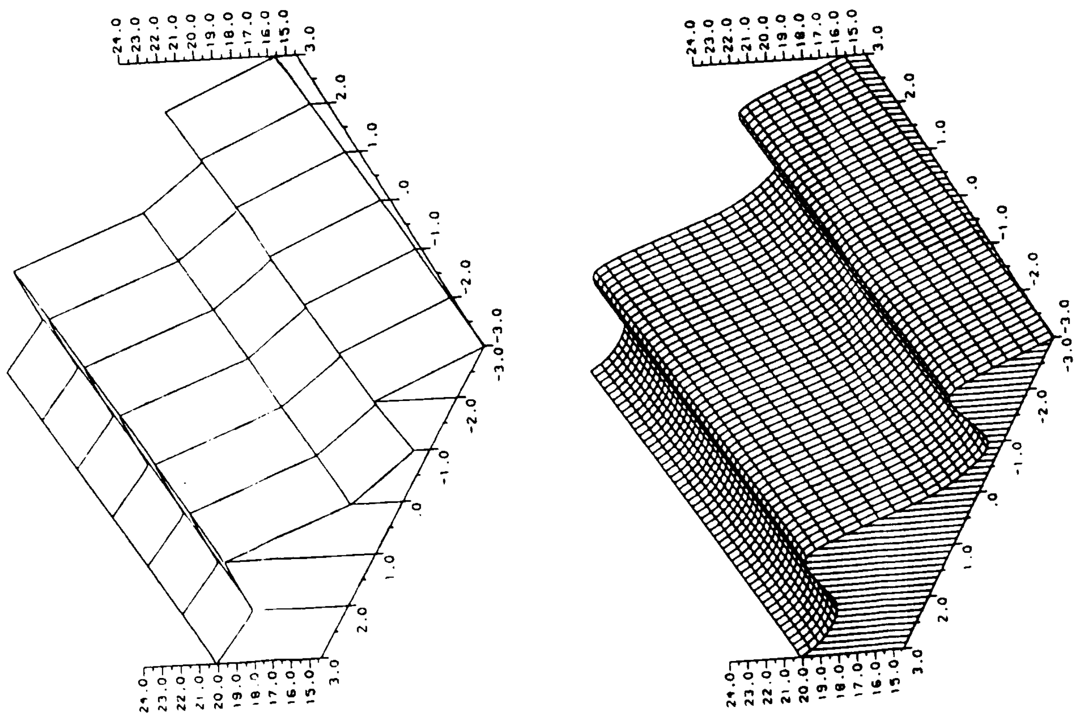


Figure 6.12. *Cubic precision scheme with* $t = 0.52$, $k = 3$.

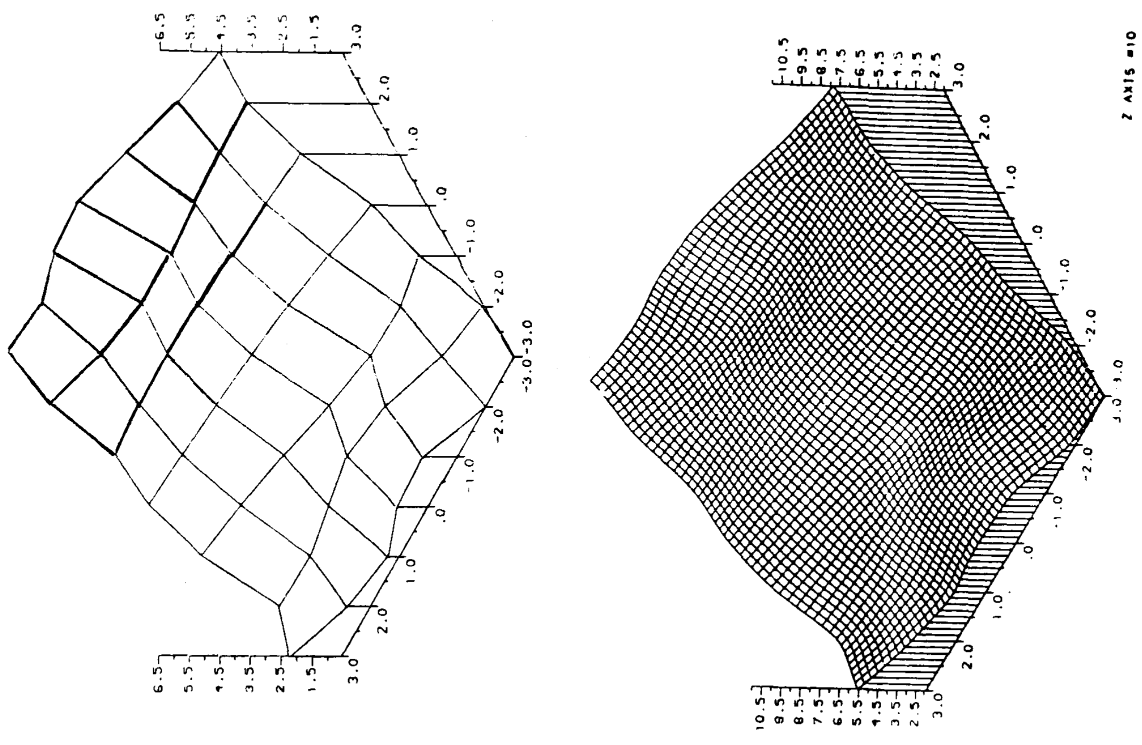


Figure 6.13. 10—*point scheme, with* $w = -0.06$, $v = -0.06$ $k = 3$.

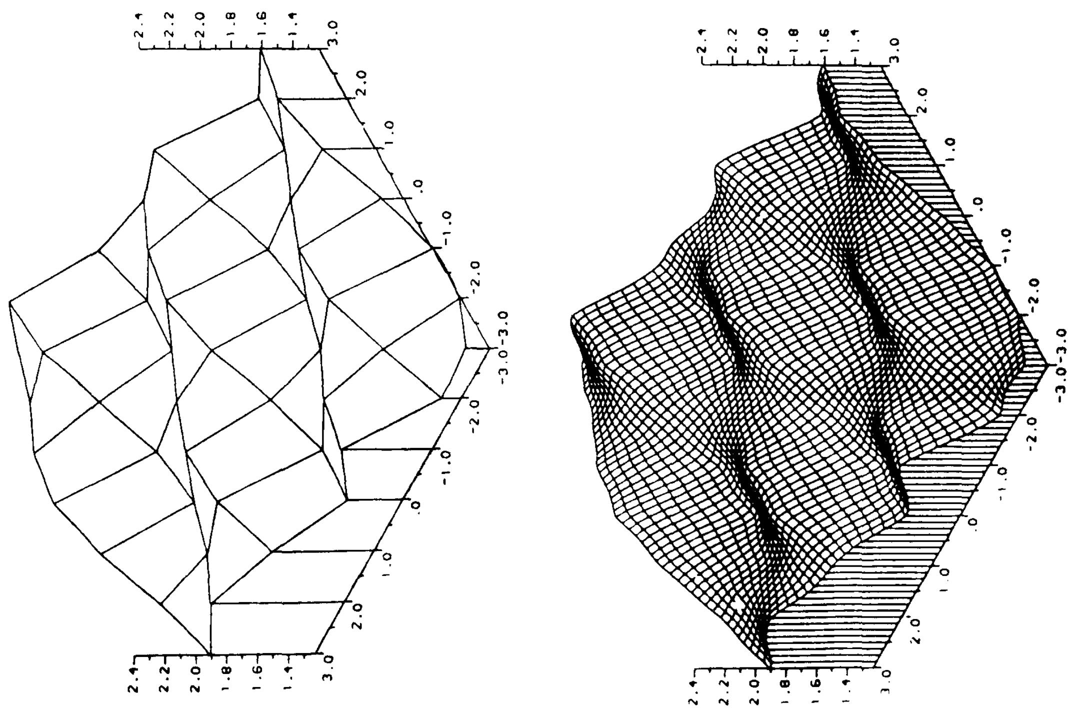


Figure 6.14. 10—*point scheme, with* $w = -0.08$, $v = -0.04$ $k = 3$.

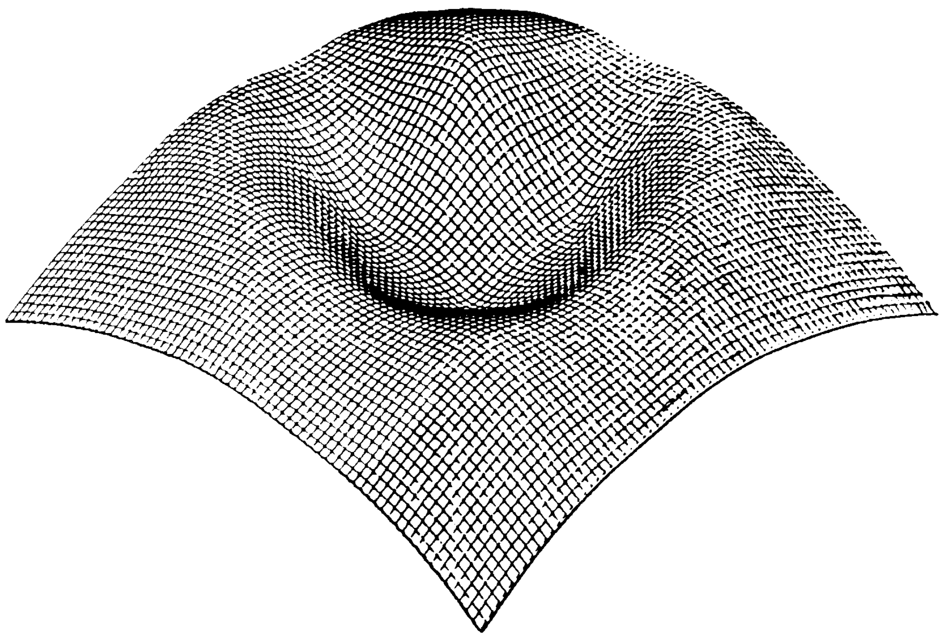


Figure 6.15. *Butterfly scheme*, $w = -0.08$, $k = 4$.

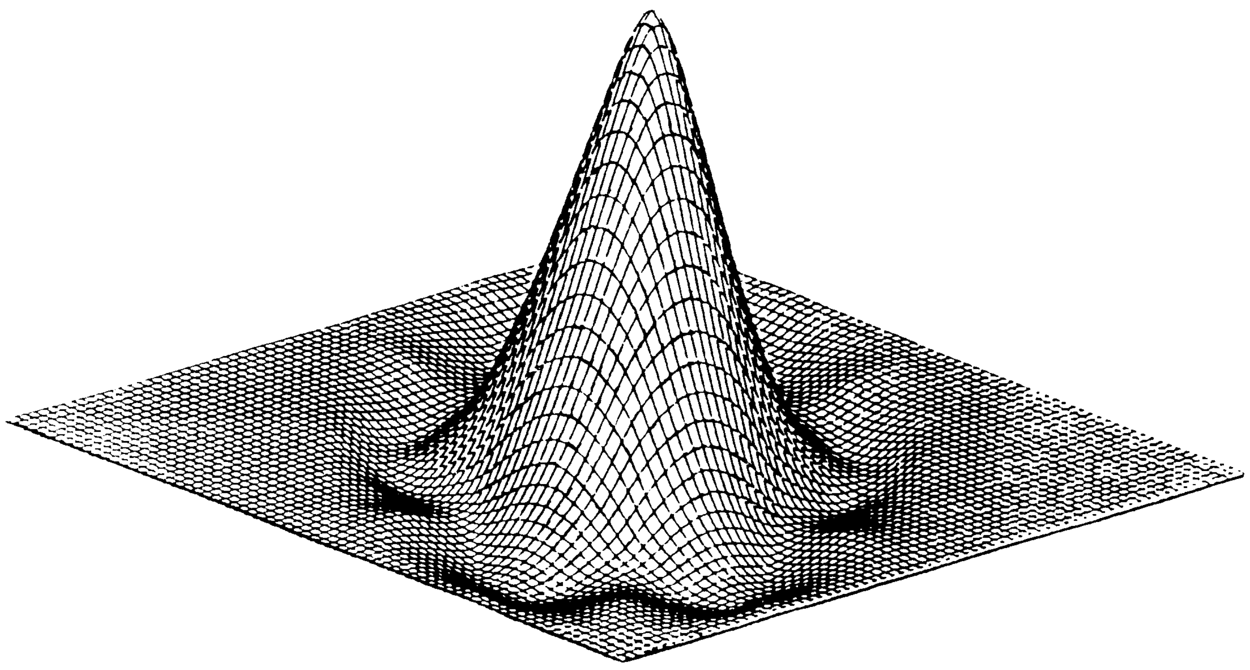


Figure 6.16. *Butterfly scheme, cardinal function*, $w = -1/16$, $k = 4$.

CHAPTER SEVEN

SUMMARY

In this thesis, we studied some subdivision algorithms for curves and surfaces. The thesis can be summarised briefly as follows.

- A brief review of recursive subdivision algorithms for curves and surfaces and a survey of the mathematical methods used to analyse them were presented.

- We studied a non-uniform subdivision scheme for smooth curve generation and derived the recursive subdivision algorithm for B-spline curves with simple knots. The *Adapted Parametrization* technique was introduced to analyse these non-uniform schemes.

- Necessary and sufficient conditions for the corner cutting schemes to produce smooth curves were studied and an explicit sufficient condition was given.

• The subdivision algorithm for uniform bi-quartic B-spline surfaces was formulated and generalized to arbitrary networks. The tangent plane and curvature properties of the limit surfaces at the *Extraordinary Points* were studied.

• The *Block Circulant Matrix* technique was used to simplify the *Extraordinary Point* analysis. This method could also be used to analyse higher even order uniform tensor-product B-spline algorithms.

• We constructed and studied the 10-point interpolatory subdivision scheme for surfaces over both uniform and non-uniform triangular control nets.

• The *Cross-Differences of Directional Divided Difference* approach for analysing uniform subdivision algorithms for surfaces was formulated. Using this method, the 10-point interpolatory subdivision scheme and the *butterfly scheme* were studied in detail. The necessary and sufficient condition for it to produce continuous and smooth surfaces were also discussed.

• The *butterfly scheme* produces smooth surfaces over uniform triangular networks if the tension parameter $-1/12 < w < 0$.

• The 10-point scheme produces smooth surfaces over uniform triangular networks if the shape parameters $\{w_i\}$ satisfy: $w := w_1$, $v := w_3$, $w_2 = -2w_1$

and $(w, v) \in \Omega_1$. The region Ω_1 is depicted in Figure 6.6.

• Sufficient conditions for the 10-point scheme to produce smooth surfaces over arbitrary triangular networks were also given.

• Most of the discussed algorithms are implemented in ***FORTRAN***. The non-uniform corner cutting algorithm, the uniform bi-quartic B-spline algorithm, the 10-point scheme for surfaces and the ***DGL*** scheme for curves as well are all programmed in ***FORTRAN*** routines which can be called to design curves and surfaces.

REFERENCES

[1]. Ball, A.A. and Storry, D.J.T.,
"Recursively generated B-spline surfaces in CAD", in *proceedings CAD,* 1984, Butterworths, p112-119.

[2]. Ball, A.A. and Storry, D.J.T.,
"Conditions for tangent plane continuity over recursively generated B-spline surfaces", ACM, Transactions on Graphics, Vol 7, 1988, p83-102.

[3]. Ball, A.A. and Storry, D.J.T.,
"A matrix approach to recursively generated B-spline surfaces", CAD, Vol 18, 1986, p437-442.

[4]. Barnhill, R.E. and Riesenfeld, R.F.,
"*Representation and Approximation of Surfaces', Mathematical Software III,* ed. by J. R. Rice, Academic Press, New York, 1977, p69-120.

[5]. Barnhill, R.E. and Riesenfeld, R.F.,
"*Computer Aided Geometric Design*", Academic Press, NY, 1974.

[6]. Barnhill, R.E. and Boehm, W.,
"*Surfaces in Computer Aided Geometric Design*", North Holland Publishing Company, Amerstdam, 1983.

[7]. Barnhill, R.E. and Boehm, W.,
"Surfaces in Computer Aided Geometric Design: A survey with new results", CAGD 2, 1985, p1-17.

[8]. Barry, P.J, and Goldman, K.N,
"A recursive proof of a B-spline identity for degree elevation", CAGD 5, 1988, p173-175.

[9]. Bezier, P.E. and Sioussiou, S.,
"Semiautomatic system for defining free-form curves and surfaces", CAD Vol. 15, 1983, p65-72.

[10]. Boehm, W.,
"On De Boor-like algorithms and blossoming", CAGD 5, 1988, p71-79.

[11]. Boehm, W.,
"Multivariate spline methods in CAGD", CAD Vol. 18, 1986, p103-106.

[12]. Boehm, W.,
"Triangular spline algorithms", CAGD 2, 1985, p61-68.

[13]. Boehm, W.,
"Multivariate spline algorithms", in ***Surfaces in Computer Aided Geometric Design***, 1983, edited by R.E. Barnhill and W. Boehm, North Holland Publishing Company, p197-216.

[14]. Boehm, W.,
"Inserting new knots into B-spline curves", CAD 12, 1980, p199-201.

[15]. Boehm, W.,
"Generating the Bezier points of B-spline curves and surfaces", CAD 13, 1981, p365-366.

[16]. Boehm, W.,
"Subdividing multivariate splines", CAD 15, 1983, p345-352.

[17]. Boehm, W.,
"The de Boor algorithm for triangular splines", in ***Surfaces in Computer Aided Geometric Design***, R.E. Barnhill and W. Boehm (eds), North Holland Publishing Company, Amerstdam, 1983.

[18]. Boehm, W.,
"Efficient evaluation of splines", Computing 30, 1984.

[19]. Boehm, W., Farin G. and Kahmann,
"A survey of curve and surface methods in CAGD", CAGD 1, 1984, p1-60.

[20]. Boehm, W., Farin G. and Kahmann,
"Calculation with Box-splines", CAGD 1, 1984, p149-162.

[21]. De Boor, C.,
"Corner cutting always works", CAGD 4, 1987, p125-131.

[22]. De Boor, C., Hollig, K. and Sabin, M.,
"High accuracy geometric Hermite interpolation", CAGD 4, 1987, p269-278.

[23]. De Boor, C.,
"A Practical Guide to Splines", Springer Verlag, New York, 1978.

[24]. De Boor, C.,
"On calculating with B-splines", J. Approximation Theory 6, 1972, p50-62.

[25]. Brunet, P.,
"Including shape handling in recursive subdivision surfaces", CAGD 5, 1988, p41-50.

[26]. De Casteljau, P.,
"Outlage Methode Calcul", Andre Citroen Automobiles, SA, Paris, 1959.

[27]. Catmull, E.E., and Clark, J.H.,
"Recursively generated B-spline surfaces on topological meshes", CAD 10, 1978, p350-355.

[28]. Cavaretta, A.S., Micchelli, C.A. and W. Dahmen,
"Regular Subdivision", preprint.

[29]. Cavaretta, A.S. and Micchelli, C.A.,
"The design of curves and surfaces by subdivision", in ***Mathematical Methods in Computer Aided Geometric Design***, Lyche, T. & Schumaker, L.L. eds., New York, 1989, p115-154.

[30]. Chaikin, G.M.,
"An algorithm for high speed curve generation", Computer Graphics and Image Processing 3, 1974, p346-349.

[31]. Charrot, P.

"The Use of Triangular and Pentagonal Patches in the Numerical Representation of Surfaces", Ph.D Thesis, 1980, Brunel University.

[32]. Cheng, Z.X.,
private communication.

[33]. Cohen, E., Lynche, T. and Riesenfeld, R.,
"Discrete B-splines and subdivision techniques in Computer aided Geometric Design and Computer Graphics", Computer Graphics and Image Processing 14, 1980, p87-111.

[34]. Cohen, E., Lynche, T. and Riesenfeld, R.,
"Discrete Box-splines and refinement algorithms", CAGD 1, 1984, p131-148.

[35]. Cohen, E. and Schumaker L.L.,
"Rates of convergence of control polygons", CAGD 2, 1985, p229-235.

[36]. Coons, S. A.,
"Surface patches and B-spline curves", in ***Computer Aided Geometric Design***, 1974, edited by R.E. Barnhill and R.F Riesenfeld, Academic Press, p1-16.

[37]. Cox, M.G.,
"The numerical evaluation of B-splines", Nat. Phys. Lab., Teddington, 1971, England.

[38]. Dahmen, W.,
"Subdivision algorithms converge quadratically", J. Comput. Appl.

Math. 16, 1986, p145-158.

[39]. Dahmen, W., N. Dyn and Levin, D.,
"On the convergence rates of subdivision algorithms for Box-splines", Constructive Approximation 1, 1985, p305-322.

[40]. Dahmen, W. and Micchelli, C.A.,
"Subdivision algorithms for the generation of box spline surfaces", CAGD 1, 1984, p115-129.

[41]. Dahmen, W. and Micchelli, C.A.,
"Line average algorithm: a method for the computer generation of smooth surfaces", CAGD 2, 1985, p77-85.

[42]. Daubechies, I. and Lagarias, J.,
"Two Scale Difference Equations I & II", preprint.

[43]. Davis, P.J.,
"Circulant Matrix", John Wiley & Sons, NY, 1979.

[44]. Doo, D.W.H.,
"A subdivision algorithm for smoothing down irregularly shaped polyhedrons", ***Proceeding Interactive Technique in Computer Aided Design, Bologna,*** Italy, 1978, p157-165.

[45]. Doo, D., and Sabin, M. A.,
"Behaviour of recursive subdivision of surfaces near extraordinary points", CAD, 10, 1978, p356-160.

[46]. Doo, D.,

"A recursive Subdivision Algorithm for Fitting Quadratic Surfaces to Irregular Polyhedrons", Ph.D Thesis, 1978, Department of Computer Science, Brunel University, Uxbridge, Middlesex, England.

[47]. Dubec, S.,

"Interpolation through an iterative scheme", J. Math. Anal. and Appl. 114, 1986, p185-204.

[48]. Dyn, N, Gregory, J.A. and Levin, D.,

"A 4-point interpolatory subdivision algorithm for curve design", CAGD 4, 1987, p257-268.

[49]. Dyn, N. and Levin, D.,

"Smooth interpolation by bisection algorithms", in *Approximation Theory* 5, 1986, Chui, Schumaker and Ward (eds)., p335-337.

[50]. Dyn, N, Gregory, J.A. and Levin, D.,

"Analysis of uniform binary subdivision scheme for curve design", to appear in Constructive Approximation.

[51]. Dyn, N, Gregory, J.A. and Levin, D.,

"Uniform subdivision algorithms for curves and surfaces design", 1988, preprint, [Shrivenham Conference].

[52]. Dyn, N, Levin, D. and Gregory, J.A.,

"A butterfly Subdivision scheme for surface interpolation with tension control", ACM Trans. on Graphics, to appear.

[53]. Dyn, N, Levin, D., and Liu, D.,
"A convexity preserving subdivision schemes for curves and surfaces", to appear.

[53*a*]. Dyn, N and Levin, D.,
"Interpolatory subdivision algorithms", preprint, presented at the Duisburg conference, 1989.

[54]. Dyn, N, Levin, D. and Micchelli, C.A.,
"Using parameters to increase smoothness of curves and surfaces generated by subdivision", to appear in CAGD.

[55]. Edwards, J.,
"***An Elementary Treatise on the Differential Calculus***", MacMillam and Company Limited, 1906.

[56]. Farouki, R.T. and Rajan, V.T,
"Algorithms for polynomials in Bernstein form", CAGD 5, 1988, p1-26.

[57]. Filip, D.J.,
"Adaptive subdivision algorithms for a set of Bezier triangles", CAD Vol. 18, 1986, p74-78.

[58]. Foley, T.A.,
"A shape preserving interpolant with tension control", CAGD 5,, 1988, p105-118.

[59]. Forrest, A.R.,
"Computational goemetry-achievements and problems", in ***Computer***

Aided Geometric Design, 1974, edited by R.E. Barnhill and R.F. Riesenfeld, Academic Press, p17-44.

[60]. Goodman, T.N.T, and Micchelli, C.A., "Corner cutting algorithms for the Bezier representation of free form curves", Rpt. No. 54611, 1986, IBM Research Centre, Yorktown Heights, New York.

[61]. Goldman, R.N. and Heath, D.C., "Linear subdivision is strictly a polynomial phenomenon", CAGD 1, 1984, p269-278.

[62]. Goldman, R.N., "Subdivision algorithms for Bezier triangles", CAD 15, 1983, p159-166.

[63]. Goldman, R.N. and DeRose, T.D., "Recursive subdivision without the convex hull property", CAGD 3, 1986, p247-265.

[64]. Gordon, W.J. and Riesenfeld, R.F., "B-spline curves and surfaces", in *Computer Aided Geometric Design*, 1974, edited by R.E. Barnhill and R.F. Riesenfeld, Academic Press, p95-126.

[65]. Gregory, J.A., "Smooth interpolation without twist constraints", *Computer Aided Geometric Design*, 1974, edited by R.E. Barnhill and R.F. Riesenfeld, Academic Press, p71-87.

[66]. Gregory, J.A.,
"N-sided surface patches", in ***Mathematics of Surfaces***, 1986, eds. by J.A. Gregory, Clarendon Press, Oxford, p217-232.

[67]. Gregory, J.A.,
"***Mathematics of Surfaces***", edited by J.A Gregory, Clarendon Press, Oxford, 1986.

[68]. Gregory, J.A. and Charrot, P
"A C^1 triangular interpolation patch for computer aided geometric design", Computer Graphics and Image Processing, Vol 13, 1980, p80-87.

[69]. Gregory, J.A. and QU Ruibin,
"An analysis of the butterfly scheme over uniform triangulations", preprint, presented at the Chamonix conference "Curves and Surfaces", June, 1990.

[70]. Gregory, J.A. and QU Ruibin,
"Non-uniform corner cutting", to appear in CAGD.

[71]. Hejna, M.J.,
"Curves constructed by geometrically based algorithms", dissertation, RPI, 1988.

[72]. Heyes, J.G.,
"***Numerical Approximation to Functions and Data***", The Athlone Press, 1970.

[73]. Hobson, E.W.,
"***Squaring the Circle, and other Monographs***", New York Chelsea Publish Co., 1953.

[74]. Koparkar, P.A. and Mudur, S.P.,
"A new class of algorithms for the processing of parametric curves", CAD Vol. 15, 1983, p41-45.

[75]. Kjellander, J.A.P.,
"Smoothing of bi-cubic parametric surfaces",CAD Vol. 15, 1983, p288-293.

[76]. Lane J.M and Riesenfeld, R.F.,
"An theoretical development for the computer generation and display of piecewise polynomial surfaces", IEEE Trans. Pattern. Anal. Math. Intel. PAM I-2, 1, 1980, p35-46.

[77]. Lane, J.M. and Riesenfeld, R.F.,
"A geometric proof of the variation diminishing property of B-spline approximation", J. Approximation Theory, 37, 1983, p1-4.

[78]. Lau, K.H.,
"***Geometrically Continuous Splines and Triangular Spline Surfaces in Computer Aided Geometric Design***", Ph.D Thesis, 1987, University of Dundee, Scotland.

[79]. Lipschitz, M.M.,
"***Differential Geometry***", McGraw-Hill, 1969.

[80] Loop, C.T.,
"Smooth Subdivision Surfaces Based on Triangles", Masters Thesis, University of Utah, 1987.

[81]. Lu Wei, Jin Tongguang and Liang Youdong,
"A new method for curve and surface modelling", to appear in CAGD.

[82]. Lyche, T. and Morken, K.,
"Knot removal for parametric B-spline curves and surfaces", Rpt., University of Oslo, 1987.

[83]. Lyche, T. and Morken, K.,
"Making the Oslo algorithm more efficient", SIAM J. of Numer. Anal. 23, 1986, p663-675.

[84]. Lyche, T. and Schumaker, L.L.,
"Mathematical Methods in Computer Aided Geometric Design", Academic Press, INC., London, 1989.

[85]. Micchelli, C.A. and Dyn, N,
"A uniform analysis for the binary subdivision algorithms", preprint.

[86]. Micchelli, C.A. and Prautzsch, H.,
"Uniform refinement of curves", IBM Research Report, 1987.

[87]. Micchelli, C.A. ,
"On a numerically efficient method for computing multivariate B-splines", in *Multivariate Approximation Theory*, Shempp, W. and Zeller, K eds, Birkhauseer, Basel, 1979, p211-248.

[88]. Micchelli, C.A. and Prautzsch, H.,
"Computing surfaces invariant under subdivision", IBM Research Report, 1987.

[89]. Micchelli, C.A. and Prautzsch, H.,
"Computing curves invariant under halving", CAGD 4, 1987, p136-140..

[90]. Micchelli, C.A. and Prautzsch, H.,
"Refinement and subdivision for spaces of integer translates of a compactly supported function", IBM Research Report, 1987.

[91]. Micchelli, C.A. ,
"Subdivision algorithms for curves and surfaces", Siggraph 86, Dallas, Texas.

[92]. Nasri, A.H.,
"Polyhedron subdivision methods for free-form surfaces", ACM Trans. on Graphics, Vol. 6, 1987, p29-73.

[93]. Piegl, L.,
"Curve fitting algorithm for rough cutting", CAD Vol 18, 1986, p79-82.

[94]. Piegl, L.,
"Recursive algorithms for the representation of parametric curves and surfaces", CAD Vol. 17, 1985, p225-229.

[95]. Piegl, L.,

"A generalization of the Bernstein-Bezier method", CAD Vol 16, 1984, p209-215.

[96]. Pratt, M.J.,
"Parametric curves and surfaces as used in computer aided design",in *Mathematics of Surfaces*, 1986, J.A. Gregory eds., Clarendon Press, Oxford, p17-45.

[97]. Prautzsch, H.,
"A short proof of the OSLO algorithm", CAGD 1, 1984, p95-96.

[98]. Prautzsch, H.,
"Generalized subdivision and convergence through quadrature", CAGD 2, 1985, p69-75.

[99]. Prautzsch, H.,
"*Unterteilungsalgorithmen fur Multivariate Splines-ein Geometrischer Zugang*", Diss. T.U. Braunschweig, 1983/1984.

[100]. Qi, D.X.,
"A class of local explicit many-knot spline interpolation schemes", Technical Summary Report #2238, Mathematics Research Centre, University of Wisconsin-Madison, US, 1981.

[101]. De Rham, G.,
"Un peu de mathematiques a propos d'une courbe plane", Elem. Math. 2, 1947, p73-76 & p89-97.

[102]. De Rham, G.,

"Sur une courbe plane", Journal de Mathematiques Pures et appliquees 39, 1956, p25-42.

[103]. Riesenfeld, R.F.,
"On Chaikin's Algorithm",Computer Graphics and Image Processing 4, 1975, p304-310.

[104]. Riesenfeld, R.F.,
"Applications of B-spline Approximation to Geometric Problems of Computer Aided Design", Ph.D Thesis, 1973, Syracuse University.

[105]. Sabin, M.A.,
"Recursive subdivision", Surfaces in CAGD , 1983, edited by R.E. Barnhill and W. Boehm, North Holland publishing Company, p269-274.

[106]. Sabin, M.A.,
"Some negative results in N-sided patches", CAD 18, 1986, p38-44.

[107]. Sabin, M.A.,
"Letter to the editor", CAGD 1, 1985, p289-290.

[108]. Sabin, M.A., and Martin, R.R.,
"Introduction to the basic mathematical tools", in *Surfaces in Computer Aided Geometric Design,* edited by R.E. Barnhill and W. Boehm, North Holland publishing Company, 1983.

[109]. Sabin, M.A.,
private communication.

[110]. Sapidis, N.S. and Kaklis, P.D.,
"An algorithm for constructing convexity and monotonicity-preserving splines in tension", CAGD 5, 1988, p127-137.

[111]. Schumaker, L.L.,
"Spline Functions: Basic Theory", John Willy & Sons, New York, 1981.

[112]. Seidel, H.P.,
"Knot insertion from a blossoming point of view", CAGD 5, 1988, p81-86.

[113]. Seidel, H.P.,
"A general subdivision theorem for Bezier triangles", in ***Mathematical Methods in Computer Aided Geometric Design***, T. Lynche and L.L Schumaker (eds), Academic Press, NY, 1989, p573-581.

[114]. Shi W.F.,
"An Interpolation method for recursively generated surfaces", to appear in CAGD.

[115]. Smith, C.,
"Conic Sections-Coordinate Geometry", MacMillan & Company Limited, 1904.

[116]. Storry, D.J.T.,
"B-spline Surfaces over an Irregular Topology by Recursive Subdivision", Ph.D thesis, Loughborough University of Technology, 1985.

[117]. Su Bu Chin et al,
"Proceedings of the second national conference on Computer Aided Geometric Design", July 1982, Qingdao, Shandong Province, P.R. of China.

[118]. Tan, S.T. and Chan, K.C.,
"A subdivision algorithm for axisymetric sections", CAD Vol. 16, 1984, p329-334.

[119]. Wang, C.Y.,
private communication.

[120]. Watson, G.A.,
"Approximation Theory and Numerical Methods", A Wiley Interscience Publication, London, 1980.

[121]. Weissman, A.,
"A 6-point Interpolatory Subdivision Algorithm for Curve Design", MSc. Thesis, 1989, Tel-Aviv University.

[122]. Ye, L.,
"Theory of Subdivision Surfaces and Its Applications", Ph.D Thesis, Fudan University, P. R. of China.