RICE UNIVERSITY

Three Approaches to Building Curves and Surfaces in Computer Aided Geometric Design

by

Ayman W. Habib

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Approved, Thesis Committee:

Ronald N. Goldman, Chairman
Professor of Computer Science

Joseph Warren
Associate Professor of Computer Science

John E. Akin
Professor of Mechanical Engineering and
Material Sciences

Houston, Texas

May, 1997

Abstract

Three Approaches to Building Curves and Surfaces in Computer Aided Geometric Design

by

Ayman W. Habib

Modeling free–form curves and surfaces is one of the fundamental problems in computer aided geometric design. To solve this problem, several modeling techniques have been proposed. Three of these techniques, are investigated. The unifying theme of these three techniques is the use and the control of geometric continuity.

The first technique deals with constructing parametric spline curves with controlled continuity between the spline segments at the knots. An axiomatic approach to geometric continuity for parametric representations is proposed. Based on this totally algebraic approach, many new flexible notions of continuity are developed. Corresponding to these notions, new spline curves are constructed in a way that gives the designer more control over the curve shape. Many examples are given.

When derivative information is available Hermite interpolation can be used to build high continuity surfaces. A dynamic programming algorithm that solves the problem of interpolating bivariate Hermite data where the interpolation positions are aligned on a triangular grid is develped and analyzed.

The third geometric continuity problem arises when modeling with subdivision surfaces, in reducing the continuity of these surfaces to allow for the insertion of

sharp edges/vertices on these surfaces. A new approach to solving this problem is introduced and analyzed with illustrative examples.

Acknowledgments

First and foremost, I would like to thank God for guiding me throughout this research and all my career up to this point. All that I have achieved is due to him.

My profound gratitude to Professor Ron Goldman, for his dedication and meticulous supervision. This work would not have been possible without his guidance and support. My deepest thanks to him.

I would also like to thank Professor Joe Warren for his support and encouragement, his sincere guidance and able advice. I cannot forget his enthusiastic teaching that got me interested in joining the Computer Graphics group in the Spring of 92. I am truly indebted to him.

My sincere gratitude to Professor Ed Akin for taking the time to serve on my thesis committee, and to Professor Tom Lyche who I enjoyed working with during his sabbatical in Rice.

Throughout my stay at Rice I was fortunate to interact with many professors and colleagues, I would like to thank them all for their continuous support. I cannot possibly forget Ivy Jorgenson the department's secretary and authority on all departmental and university regulations. I really appreciate her help and encouragement. More personally, I would like to acknowledge the various contributions of my friends and my family who helped me in various ways through the years that I spent in Rice.

Finally, this research would not have been possible without the generous support of Rice University, and the National Science Foundation.

Contents

Illustrations

Chapter 1

Introduction

Computer Graphics is one of the fast growing areas in computer science. The term "Computer Graphics" has been used to denote a broad spectrum of problems, including computer vision, robotics, virtual reality, computer animation, and scientific visualization. At the heart of all these disciplines are the common important issues of the representation and manipulation of shapes. The area of computer graphics that studies mathematical models to represent shapes and embody their geometry is known as **C**omputer **A**ided **G**eometric **D**esign (CAGD). Computer systems that aid in this process are usually referred to as geometric modelers.

Many industries depend on CAGD. The automotive, shipbuilding, and aerospace industries make extensive use of CAGD to build mathematical models for various components of their vehicles, and then utilize these models to analyze and refine their design before manufacturing. These models are much cheaper to construct and easier to modify than the physical models that were customarily built.

One particularly important application of CAGD is mesh generation for finite element analysis, which is used extensively in engineering applications to analyze the performance of various mechanical parts. Scientific visualization is another area that benefits from CAGD: visualizing pressure, heat distribution and fluid flow. It is now even possible to visualize functions in more than three variables. Yet another growing area that thrives on CAGD is the entertainment industry. One major part of the design of many movies, animated cartoons or commercials that make use of computer generated images, is building mathematical computer models. The movie "Toy

Story", completely generated by computer, has recently been released and considered a critical and financial success.

One of the main issues in designing a geometric modeler is selecting a mathematical representation for curves and surfaces. Selecting a particular representation is important because later manipulations and analyses depend greatly on the specific representation. Analysis of curves and surfaces entails studying their smoothness properties and devising algorithms for evaluation and rendering, differentiation and integration, trimming and computing intersections as well as conversion between different representations.

Now, given a real object – a table, a chair, or a computer – what would be the "best" representation? The best representation depends on the application, and the operations we are going to perform on this representation. For example, if all we need is a rendering of a chair from a fixed view under fixed lighting conditions and surroundings, it might suffice to scan a picture of the chair and perform some image processing algorithms on it, without building a three–dimensional mathematical model. However, for a furniture company that wants to manufacture the chair and analyze its endurance, or an interior designer who wants to experiment with how the chair looks in different colors, from different angles, under different lighting conditions or surroundings, a full three–dimensional mathematical model needs to be constructed.

Physical objects, as well as functions resulting from measuring physical phenomena, are usually smooth or composed of many smooth pieces. Consequently, it is important for the mathematical representations that model these shapes to have similar smoothness properties.

Analyzing mathematical models for continuity and smoothness is a hard problem that has been addressed by many researchers. The level of difficulty in analyzing the smoothness of a mathematical model depends on the specific representation used. In this thesis, three approaches to building curves and surfaces are investigated. The principle and common idea behind all three approaches is to use some form of geometric continuity as the driving force behind construction algorithms for curves and surfaces.

Using smoothness constraints to guide shape design is an important approach. When designing with curves and surfaces composed of pieces (also known as splines), it is important to provide the designer with the ability to control how the pieces join, or equivalently to control the geometric continuity between the pieces at their common boundaries.

Geometric continuity in this very broad sense is hard to define precisely in mathematical terms. Designers like to control what they see. However, what they see can be represented in many different ways, and depending on the representation the definition of smoothness may change. The main problem addressed in this thesis is how to use specific smoothness constraints to build smooth curves and surfaces. Our approach is constructive, so while we shall enrich the theory of geometric continuity by developing new definitions, our real objective is to provide algorithms and tools for designers to control the continuity and smoothness of the curves and surfaces they design.

1.1 Mathematical Representations

One major theme of this thesis is geometric continuity. What is geometric continuity? What would be a sensible definition of smoothness for curves and surfaces and how can we build curves and surfaces of controlled continuity through interpolation or subdivision? Depending on our mathematical representation, there are several ways to approach these problems.

Three classes of representations have been adopted in CAGD to model shapes. A short description of each of these classes is given below. In each case the formal definition is given first, followed by an example for illustration, and a brief discussion of when this representation is preferred. A comprehensive study of the properties of each of these representations is beyond the scope of this introduction. Many standard books on CAGD [6, 25] give broader expositions to this material.

- Parametric Representation: In this representation, a curve $f(t)$ in $\mathbb{R}^d$ is a map

  $$f(t) = \{f_1(t), f_2(t), \ldots, f_d(t)\} \quad f_i\colon I \mapsto \mathbb{R}^d; \quad I \subseteq \mathbb{R}^1$$

  The functions $f_i(t)$ are called coordinate functions, t is the parameter and I, the parameter domain, is an interval of $\mathbb{R}^1$. An example of a parametric curve in $\mathbb{R}^2$ is the segment of a parabola $\{t, t^2\}$, $t \in [-1, 1]$.

  For parametric surfaces, the parameter is $(u, v) \in \mathbb{R}^2$ and the parameter domain $I \subset \mathbb{R}^2$.

  It is easy to see the usefulness of this representation. A point on the curve or surface is generated by just substituting the corresponding parameter value into the formulas for the coordinate functions. This is one of the reasons why parametric representations are among the most commonly used representations in CAGD.

  Some geometric algorithms benefit from this representation by decoupling problems in higher dimensions into many problems in $\mathbb{R}^1$, treating each coordinate function independently. For example, to build a bounding box around a surface represented parametrically, the computations can be performed coordinatewise.

  The major drawback of parametric representations is the parameterization itself. Parameterization of curves introduces the concept of the speed by which a point moves along the curve. Intrinsically, as a point set, a curve does not have a speed. The only intrinsic parameterization is arc length, which is mainly a theoretical tool, generally too cumbersome for computation.

For any curve or surface there are many different parameterizations, and some parameterizations are preferable over others for theoretical or computational reasons. For example, polynomial and rational parameterizations are more efficient to compute with than trigonometric ones; however not every function has a polynomial parameterization, and rational parameterizations may introduce singularities.

- Implicit Representation: An implicit curve in $\mathbb{R}^2$ is the zero set of some bivariate function

$$f(x, y) = 0.$$

For example, a unit circle in the plane has the implicit equation $x^2 + y^2 - 1 = 0$. Implicit surfaces are zero sets of trivariate functions.

The main advantage of the implicit representation is that it splits the space into two regions, an inside and an outside depending on the sign of f. Testing if a point is inside or outside a surface is an important problem that needs to be solved over and over (for example to compute hidden and occluded surfaces in a complex scene). This problem is hard to solve using a parametric representation. On the other hand, there is no simple technique for producing points on an object represented implicitly; thus rendering implicit surfaces is not as easy as rendering parametric surfaces.

If we restrict our attention to polynomials, implicit representations are more general than parametric representations. The circle, which has a polynomial implicit equation, is known not to have a polynomial parameterization, although it does have a rational parametric representation.

- Procedural Representation: Curves and surfaces in this category are described by the process that generates them. Procedural representations encompass many different techniques: for example, defining a surface by the milling process that creates it. Offset and subdivision curves and surfaces are other instances of this type of representation. With the increased use of computers in geometric modeling, subdivision curves and surfaces have flourished because it seems appealing to represent a curve or surface by a simple procedure.

  To illustrate subdivision, consider the process of taking a polygon, inserting two new vertices at 1/4, 3/4 the distance between every pair of adjacent vertices, then connecting the new vertices, as shown in Figure 1.1. This very simple algorithm is known as Chaikin's algorithm, and the curve that results after repeating the process ad infinitum is a well known smooth curve (C^1–quadratic B–spline) [51].

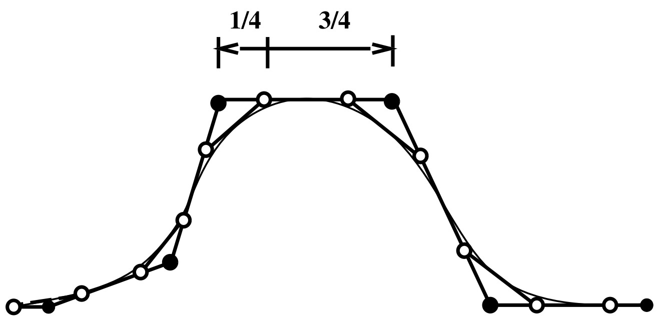


Figure 1.1: Chaikin's subdivision algorithm

As this example shows, the power of procedural representations comes from their ability to represent fairly complicated shapes using a simple process. Subdivision also provides a natural framework in which multi–resolution analysis can be performed since successive control polygons/polyhedra provide closer and closer approximations to the limit object. However, the analysis of the limit curves and surfaces is much harder than the analysis for the other two representations. Actually for weights other than (1/4, 3/4), Chaikin's algorithm never produces C^1 curves! For weights (1/2, 1/2), we get the control polygon back as the limit curve.

Analyzing the smoothness of subdivision schemes for curves and surfaces has received a lot of attention in recent years. Although there have been some results regarding the analysis of subdivision schemes for curves, subdivision schemes for surfaces and higher order data offer a wealth of open research problems.

1.2 Thesis Overview

The rest of this thesis is broken into three parts. Each part addresses one technique that has been used in CAGD to create curves and surfaces. The common unifying theme among the three parts is the use of a form of continuity constraint to construct curves and surfaces with controlled smoothness.

In Part I, the theory of geometric continuity for parametric curves and surfaces, one of the principle topics in CAGD, is addressed and a new axiomatic approach is proposed. The axioms are purely geometric conditions that make a notion of geometric continuity sensible. This axiomatization is then used to derive new notions of geometric continuity. Spline curves and surfaces that have the new types of continuity at the knots also contain more free parameters which can be used to control shape in geometrically meaningful way.

One approach that has been used to get higher order smoothness is to interpolate derivatives (also known as Hermite interpolation). In Part II we address a bivariate Hermite interpolation problem and present a novel interpolation algorithm based on dynamic programming that solves the bivariate interpolation problem when the interpolation nodes are arranged in a triangular grid.

A surface represented procedurally as a subdivision surface is presented in Part III, where a novel algorithm for building C^1–subdivision surfaces over polyhedra of arbitrary topological type is given. The construction algorithm produces provably tangent plane continuous surfaces. It also allows for modeling objects with boundaries, sharp edges and sharp vertices. Continuity conditions guide the construction algorithm over the smooth parts of the surface. Separability (that is, the ability to represent a smooth surface as multiple patches) is the main goal of the edge and vertex insertion algorithms that we propose. A formal formulation of this property, and a proof that our approach does indeed guarantee separability is also presented.

At the end of each Part of this thesis we discuss some implementation issues, summarize the contributions made in this particular Part, and outline a set of open research problems for further investigation.

Part I

Building Splines Using Geometric Continuity

Chapter 2

A General Theory of Contact for Parametric Curves

2.1 Introduction

The ability to design smooth curves and surfaces is fundamental to geometric modeling and CAGD applications. Several different definitions of continuity and smoothness between parametric curves are now available with different degrees of flexibility for designers. For each of these notions there exists a theory and a class of splines that have this type of continuity across the knots. The simple parametric continuity (C^n) requires the continuity of parametric derivatives up to order n. The main drawback of this notion of geometric continuity is its dependence on a specific parameterization. Reparameterization continuity (G^n) solves this problem by requiring the continuity of parametric derivatives after a suitable reparameterization [3, 4, 5, 6, 17, 30, 31, 35]. Unlike C^n, G^n depends only on the intrinsic geometry of the curve rather than on any particular parameterization. Other notions of contact, such as Frenet frame continuity (F^n) [20, 22, 33, 36] and continuity of osculating linear spaces (O^n) have also been investigated.

The scope of this study for parametric curves covers notions of geometric continuity that can be represented by connection matrices. All the continuity measures mentioned above fall into this category. In Section 2.2.2 we review the main properties of each of these standard notions.

Throughout the thesis the terms "notion", "measure", "form", "type" and "theory" are used as synonyms when they modify the phrase "geometric continuity". Thus, G^n and F^n are two different notions, measures, forms, types, or theories of geometric continuity. Also the phrases "geometric continuity" and "contact" are synonymous throughout the thesis; see Section 2.3.1 for further clarification.

The rest of this chapter is divided into two parts. Section 2.2 reviews some relevant literature and terminology, and gives a characterization of notions of geometric continuity embodied by connection matrices. The second part, Section 2.3, introduces a set of analytic and geometric properties that all reasonable notions of geometric continuity must satisfy. These properties are then interpreted into an equivalent collection of algebraic conditions that must be satisfied by any set of connection matrices representing some form of geometric continuity.

2.2 A Unified Theory of Contact for Parametric Curves

As mentioned in Section 2.1, the objective in this part is to study the design of spline curves with different types of geometric continuity across the knots. This study is conducted by analyzing smoothness at an individual knot/parameter value, rather than for a complete spline curve, with multiple knots and different connection matrices at the knots. Since there are readily available algorithms for the construction of spline curves that have continuity described by connection matrices at the knots [1, 20, 53], we shall restrict ourselves in this thesis to precisely these notions of geometric continuity that can be described by connection matrices. We begin with some terminology and notation.

2.2.1 Connection Matrices

Connection matrices were first introduced by Dyn and Micchelli [22] to study spline curves joined with different types of geometric continuity at the knots. Recently, the spaces of spline with continuity specified by connection matrices at the knots have been studied by several authors [1, 7, 20, 23, 53]. Typically these authors start with some fixed connection matrix at each knot and investigate the linear space of splines determined by a fixed sequence of knots and connection matrices. Our focus in this study will be different; we study the structure of the sets of connection matrices for a single knot, rather than the space of splines. Notice that if we allow a set of connection matrices at each knot rather than a single connection matrix then this set of splines is not a linear space, but this problem will be of no concern to us here.

We begin by introducing some notation that will be helpful throughout this part. Let $f : \mathbb{R}^1 \mapsto \mathbb{R}^d$ be an n–times differentiable function and let τ be a parameter value. We write

$$D_n(f)(\tau) = [f(\tau), f'(\tau), \ldots, f^{(n)}(\tau)]^T.$$

A piecewise n–times differentiable function $f : \mathbb{R}^1 \mapsto \mathbb{R}^d$ is said to have connection matrix $M = (M_{ij}), i, j = 0, 1, \ldots, n$, at parameter value τ if and only if

$$D_n(f^+)(\tau) = M \cdot D_n(f^-)(\tau) \tag{2.1}$$

where f^- and f^+ denote the function f to the left and to the right of τ respectively.

Notice that a curve $f(t) = \{f_1(t), f_2(t), \ldots, f_d(t)\}$ has the connection matrix M at τ iff

$$D_n(f_i^+)(\tau) = M \cdot D_n(f_i^-)(\tau) \quad i = 1, \ldots, d.$$

Since τ is fixed throughout, we shall often omit the parameter τ from equation (2.1) and simply write $D_n(f^+) = MD_n(f^-)$.

The same notation is used for pairs of curves f, g sharing parameter value τ. Thus, f and g have connection matrix M at parameter value τ iff

$$D_n(f)(\tau) = M \cdot D_n(g)(\tau). \tag{2.2}$$

The number n is called the order of contact.

In the following section we recall notions of contact representable by connection matrices that have been investigated in the literature.

2.2.2 Available Notions of Contact and Their Geometric Invariants

Many notions of contact, all representable by connection matrices, are readily available in literature. As we show in this section, different sets of connection matrices correspond to different notions of geometric continuity. However, each of these notions corresponds to some set of intrinsic geometric invariants that are preserved by the associated type of geometric continuity. These invariants provide the geometric intuition that justifies using these notions as measures of smoothness.

Parametric contact of order n (C^n): The first standard notion of contact, that is representable by connection matrices, is parametric contact of order n. For a pair of curves $f(t)$ and $g(t)$ to have this type of contact at a common parameter value τ, their first n parametric derivatives must match at τ. That is, the parametric derivatives must satisfy (2.2) with an $n+1 \times n+1$ connection matrix M equal to the identity matrix. The geometric invariants maintained by this notion of contact are, clearly, the parametric derivatives up to order n at $t = \tau$.

Parametric continuity is the most intuitive, and naive, measure of continuity. The major drawback of this notion is that it depends inherently on curve parameterizations. As explained in Section 1.1, parameterization of curves is not unique; thus parametric derivatives actually measure properties of the parameterization itself rather than the intrinsic geometry of curves. For example, parametric continuity of the first order does not only mean that the two curves meet with tangent direction smoothness, but, in addition, that the parameterizations of the two curves have equal speed at the join.

Geometric contact of order n (G^n): This notion is also referred to as reparameterization continuity of order n. It was studied first by Barsky and then by DeRose

[3, 4, 5, 17]. This type of contact holds between two parametric curves iff we can re-parameterize one of the curves to achieve parametric contact of order n. DeRose shows that G^n–continuity is equivalent to C^n–continuity if arc–length is used for the parameterization [17]; thus the geometric invariants for this notion are the parametric derivatives up to order n with respect to the arc–length parameterization. If we reparameterize a curve $g(t)$ at parameter value τ by a function h, the relation between the parametric derivatives of g at τ before and after reparameterization can be computed by repeated application of the chain rule. Moreover, $D_n(g \circ h)(\tau)$ depends linearly on $D_n(g)(\tau)$. Thus there is a matrix $R(h)(\tau)$ depending only on the derivatives of h at τ such that

$$D_n(f)(\tau) = D_n(g \circ h)(\tau) = R(h)(\tau) \cdot D_n(g)(\tau). \tag{2.3}$$

which is of the form (2.2) for $M = R(h)(\tau)$. Moreover,

$$R(h)(\tau) = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & \beta_1 & 0 & \cdots & \cdots & 0 \\ 0 & \beta_2 & \beta_1^2 & \cdots & \cdots & 0 \\ 0 & \beta_3 & 3\beta_1\beta_2 & \beta_1^3 & \cdots & 0 \\ 0 & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \beta_n & \cdots & \cdots & \cdots & \beta_1^n \end{bmatrix}, \tag{2.4}$$

with $\beta_i = h^{(i)}(\tau)$. Matrices of the form (2.4) are often referred to as β–matrices because spline curves built such that their derivatives satisfy (2.3) across the knots are called β-splines [5, 30, 37]. The β_i's are called shape parameters because they control the shape (e.g. β_1 controls the bias and β_2 controls the tension for cubic β–splines) of the spline curve.

Henceforth we shall refer to smooth functions $h : \mathbb{R}^1 \mapsto \mathbb{R}^1$ such that $h(\tau) = \tau$ and $\beta_1 = h'(\tau) > 0$ as reparameterization functions at τ. These conditions guarantee that h is regular and orientation preserving, that h has a local inverse and that h^{-1} is also a reparameterization function.

This definition of contact (G^n) has the advantage that it does not depend on a specific parameterization. However it also seems restrictive since it requires matching the first n arc–length parametric derivatives, even though the continuity of derivatives higher than the first order can not be detected by the human eye.

Another equivalent set of geometric invariants for G^n–continuity is the continuity of the curvature and its first $n-1$ arc length derivatives. This equivalence can be easily observed by expressing the curvature in terms of derivatives with respect to arc length parameterization, and differentiation.

Frenet Frame Contact of Order n (F^n): For a space curve in $\mathbb{R}^n$ with linearly independent parametric derivative vectors up to order n, we can build a Frenet frame and define $n-1$ curvatures (curvature and torsion for example when $n=3$) [20]. Frenet frame contact of order n (F^n) holds between two parametric curves $f(t)$ and $g(t)$ at $t=\tau$ iff the curvatures of f and g match at the common parameter value τ. Dyn, Edelman and Micchelli [20, 22] introduced this notion of contact and proved that two parametric curves have this type of contact iff their parametric derivatives satisfy (2.2) for a connection matrix M of the form

$$M = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \beta_1 & 0 & \cdots & 0 \\ 0 & * & \beta_1^2 & \cdots & 0 \\ 0 & \vdots & \vdots & \ddots & \vdots \\ 0 & * & * & \cdots & \beta_1^n \end{bmatrix}, \tag{2.5}$$

where the entries of M marked with $*$ are all free.

Goldman and Micchelli [31] prove that this definition of geometric contact is also invariant under reparameterization and thus it depends only on intrinsic geometric properties of the curves. This type of contact, however, introduces a huge number of shape parameters whose geometric effects are not fully understandable.

Osculating Linear Spaces Contact of Order n: This type of contact requires only that two curve segments share their osculating tangent spaces at the common parameter value. The corresponding set of connection matrices are lower triangular. This notion is the weakest notion that makes geometric sense.

Continuity of Tangent Surfaces: Pottmann [49] gives the only other example of notions of geometric continuity. His construction is based on building a tangent surface for the parametric curve and insuring some kind of continuity between the tangent surfaces of the two parametric curves at their common parameter value. The resulting set of connection matrices have some interesting structure. One of these examples appears in Section 3.3.2 of Chapter 3.

One of the reasons behind conducting this investigation is that each of the above definitions of geometric contact comes from a completely different geometric perspective, although all of them lead to definitions that are embodied by connection matrices. The goal here is to unify these theories within one framework. Another problem is that all the geometric intuition behind these definitions becomes extremely hard to understand in higher dimensions. We show later on in this chapter how to come to these definitions, and some more, from an entirely algebraic point of view, thus removing the intuitive difficulties arising from reasoning geometrically in higher dimensions.

2.2.3 The Characterization

A notion of contact such as G^n or F^n can be thought of as a binary relation on the set of curves defined in a neighborhood of the parameter value τ. If H^n is some form of contact, then a corresponding binary relation on parametric curves can be defined by:

$$(f, g) \in H^n \iff f \text{ and } g \text{ have } H^n \text{ contact at } \tau.$$

A notion of contact of order n, H^n, should specify the relation between the first n derivatives of f and g at the common parameter value τ. Thus there must exist some functions, R, relating these derivatives such that

$$(f, g) \in H^n \iff D_n(f)(\tau) = R\{D_n(g)(\tau)\}.$$

We shall use the notation $\Gamma(H^n)$ to denote the set of all such functions R. For example, $\Gamma(G^n)$ denotes the set of all reparameterization matrices, and $\Gamma(F^n)$ denotes the set of all Frenet frame connection matrices. We want to investigate the general conditions under which we can represent the functions in $\Gamma(H^n)$ by connection matrices.

Inspecting formula (2.1) which defines what it means for a curve $f : \mathbb{R}^1 \mapsto \mathbb{R}^d$ to have connection matrix M at parameter τ, we notice that each of the coordinate functions must have the *same* connection matrix M at τ. So, types of continuity that are represented by connection matrices are defined *coordinate-wise*. It follows that if $T \in \Gamma(H^n)$ is to end up being represented by a connection matrix, T must also be defined coordinate-wise. Intuitively, this means that T should perform identically on all coordinate functions. Thus, if $f(t) = \{f_1(t), f_2(t), \ldots, f_d(t)\}$, we shall insist that for all $T \in \Gamma(H^n)$

$$D_n(f^+) = T\{D_n(f^-)\} \iff D_n(f_i^+) = T\{D_n(f_i^-)\} \quad i = 1, \ldots, d. \tag{2.6}$$

We shall now determine conditions under which forms of geometric continuity defined coordinate-wise can be represented by connection matrices. For this, we introduce the notion of *linear invariance*.

Intuitively, a notion of geometric continuity H^n is *linearly invariant* if curves that are H^n–continuous preserve their H^n–continuity under all linear transformations. Formally, let $f : \mathbb{R}^1 \mapsto \mathbb{R}^d$, and let $L : \mathbb{R}^d \mapsto \mathbb{R}^d$ be a linear transformation. Then we say that H^n is linearly invariant if

$$D_n(f^+) = T\{D_n(f^-)\} \Rightarrow D_n(L(f^+)) = T\{D_n(L(f^-))\} \tag{2.7}$$

for all $T \in \Gamma(H^n)$ and all linear transformations L.

Any plausible notion of geometric continuity is expected to be linearly invariant because continuity should be a feature of the curves themselves rather than the coordinate system. Applying a nonsingular linear transformation (L) to a curve is equivalent to applying the inverse transformation (L^{-1}) to the coordinate system, an operation that must certainly preserve reasonable notions of continuity. If L is a singular linear transformation, then L represents some type of linear projection which should also maintain notions of continuity defined coordinate-wise. In fact, we shall insist later on (see Sections 2.3.3 and 2.3.4) that continuity should be preserved under all projective transformations.

We now show that, under the assumption that continuity is defined coordinate-wise, the set of geometric continuity measures that can be embodied by connection matrices are exactly those that are invariant under linear transformations.

Theorem 2.1

Let H^n be a notion of geometric continuity, then:

H^n can be represented by connection matrices $\iff$ H^n is defined coordinate-wise and is invariant under linear transformations.

Proof $\Rightarrow$:

Since H^n is a notion of geometric continuity that can be represented by connection matrices, it follows immediately from (2.1) that H^n is defined coordinate-wise. We need to show that H^n is invariant under linear transformations.

Suppose that $f : \mathbb{R}^1 \mapsto \mathbb{R}^d$ and that $(f^+, f^-) \in H^n$ at some parameter τ. Then, at τ, we have

$$D_n(f^+) = MD_n(f^-); \qquad M \text{ is an } H^n \text{ connection matrix.}$$

Now let $L : \mathbb{R}^d \mapsto \mathbb{R}^d$ be a linear transformation defined by $L(x) = xL$, where L is a $d \times d$ matrix. Then

$$D_n(f^+)L = MD_n(f^-)L.$$

So, by the linearity of differentiation

$$D_n(f^+L) \quad = \quad MD_n(f^-L).$$

Hence,

$$D_n(f^+) = MD_n(f^-) \Rightarrow D_n(L(f^+)) = MD_n(L(f^-)).$$

$\Leftarrow$:

Since H^n is defined coordinate-wise, it is enough to consider functions $f, g \colon \mathbb{R}^1 \mapsto \mathbb{R}^1$.

Now to show that $T \in \Gamma(H^n)$ can be represented by a connection matrix, we must show that T is linear. That is,

$$\begin{aligned} T\{cD_n(f)\} &= cT\{D_n(f)\} \\ T\{D_n(f) + D_n(g)\} &= T\{D_n(f)\} + T\{D_n(g)\}. \end{aligned}$$

Assume that

$$D_n(f^+) = T\{D_n(f^-)\}.$$

Since linear invariance implies scale invariance, scaling by a constant c, we have

$$D_n(cf^+) = T\{D_n(cf^-)\}.$$

Hence, by the linearity of differentiation

$$cT\{D_n(f^-)\} = T\{cD_n(f^-)\}. \tag{2.8}$$

Moreover, since H^n is defined coordinate-wise,

$$D_n(f^+) = T\{D_n(f^-)\}, D_n(g^+) = T\{D_n(g^-)\} \Rightarrow D_n(f^+, g^+) = T\{D_n(f^-, g^-)\}.$$

Applying the shearing transformation

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

to both sides of the last equation and employing linear invariance, we get

$$D_n((f^+ + g^+), g^+) = T\{D_n((f^- + g^-), g^-)\}.$$

It follows immediately from (2.6) that

$$D_n(f^+ + g^+) \quad = \quad T\{D_n(f^- + g^-)\}.$$

However, by the linearity of differentiation

$$D_n(f^+ + g^+) = D_n(f^+) + D_n(g^+) = T\{D_n(f^-)\} + T\{D_n(g^-)\}.$$

Hence

$$T\{D_n(f^- + g^-)\} = T\{D_n(f^-)\} + T\{D_n(g^-)\}. \tag{2.9}$$

From (2.8) and (2.9) it follows that T is linear, so T can be represented by a connection matrix. $\square$

2.3 Fundamental Properties of Geometric Continuity

From here on we shall deal only with notions of geometric continuity represented by connection matrices. Linear invariance implies that notions of geometric continuity represented by connection matrices are independent of both the scale and the orientation of the coordinate axes. However, arbitrary sets of connection matrices will not always correspond to plausible notions of geometric continuity. We expect notions of geometric continuity to satisfy some additional geometric criteria. Next we pick a generic notion of geometric continuity H^n and specify a minimal set of geometric properties that we require H^n to have. We shall soon see that these requirements impose some structure on the associated set of connection matrices $\Gamma(H^n)$.

Geometric Properties of H^n

1. H^n–continuity should induce an equivalence relation on the set of curves.

2. Reparameterizations should preserve H^n–continuity.

3. H^n–continuity should be invariant under affine transformations.

4. H^n–continuity should be invariant under perspective projection.

5. H^n-continuity should guarantee the continuity of the first n linear osculating spaces.

In the following subsections, we discuss each of the above properties, explain why it is natural, and derive its algebraic consequences for sets of connection matrices.

2.3.1 Equivalence Relation

A continuity measure H^n induces a binary relation on the set of curves defined in a neighborhood of some fixed parameter τ. We write $(f, g) \in H^n$ if and only if f and

g meet with H^n-continuity at τ. We claim that any reasonable notion of contact H^n must be an equivalence relation. That is, H^n should satisfy the three basic properties of equivalence relations:

1. Reflexivity: $(f, f) \in H^n$

   Consider a curve f that is C^n at τ. Whatever notion of continuity H^n represents, we expect f to be H^n–continuous at τ. Thus $(f, f) \in H^n$.
   Alternatively, we expect

   $$D_n(f) = D_n(g) \Rightarrow (f, g) \in H^n$$

   so certainly $(f, f) \in H^n$.

2. Symmetry : $(f, g) \in H^n \Rightarrow (g, f) \in H^n \cdot$

   For any reasonable notion of geometric continuity H^n, if two curves (f, g) are H^n–continuous at τ, then (g, f) should also be H^n–continuous at τ. That is, continuity should depend only on the curves, not their order.

3. Transitivity : $(f, g), (g, h) \in H^n \Rightarrow (f, h) \in H^n \cdot$

   So far we have been using the word *continuity* loosely to describe the property that parametric derivatives are related somehow at a point. This notion is better described as *contact* because we can always find curves whose derivatives satisfy some specified relation but are not, by our intuition, *smooth* (see Figure 2.1). We prefer to employ the term *continuity* here, however, because it is widely used and accepted in the CAGD literature for similar notions. Intuitively, contact is a transitive relation. All the standard notions of geometric continuity – G^n, F^n and O^n – induce transitive relations.

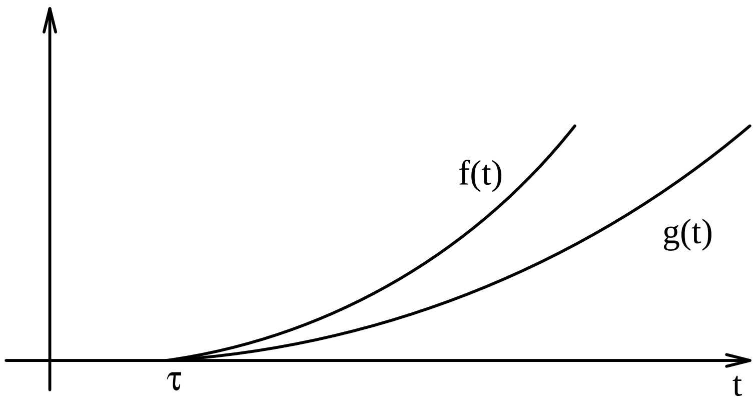


Figure 2.1: Two curves that have high order contact but their join is not smooth.

We shall show that together these requirements on H^n force the set of connection matrices $\Gamma(H^n)$ to form a group. We begin with a useful lemma.

Lemma 2.1 Let M be an arbitrary $(n+1)\times(n+1)$ matrix. Then there is a function $f\colon \mathbb{R}^1 \mapsto \mathbb{R}^{n+1}$ such that:

$$D_n(f)(\tau) = M.$$

Proof To construct $f(t) = \{f_0(t), f_1(t), \ldots, f_n(t)\}$ such that

$$D_n(f_i(t))(\tau) = \{M_{0i}, M_{1i}, \ldots, M_{ni}\}^T,$$

we treat the i^{th} column of M as the Taylor coefficients of $f_i(t)$ at $t = \tau$. That is, we pick

$$f_i(t) = \sum_{k=0}^{n} M_{ki} \cdot \frac{(t-\tau)^k}{k!}.$$

Now,

$$D_n(f)(\tau) = \{\{M_{00}, M_{10}, \ldots, M_{n0}\}^T, \ldots, \{M_{0n}, M_{1n}, \ldots, M_{nn}\}^T\} = M. \square$$

Proposition 2.1 H^n is an equivalence relation $\iff$ the corresponding set of connection matrices $\Gamma(H^n)$ form a group. In particular:

(i) H^n is reflexive $\iff$ $I \in \Gamma(H^n)$.

(ii) H^n is symmetric $\iff$ $(M \in \Gamma(H^n) \Rightarrow M^{-1} \in \Gamma(H^n))$.

(iii) H^n is transitive $\iff$ $(M, N \in \Gamma(H^n) \Rightarrow MN \in \Gamma(H^n))$.

Proof

- H^n is reflexive $\iff$ $I \in \Gamma(H^n)$.

  $\Rightarrow$:

  $$H^n \text{ is reflexive } \Rightarrow (f, f) \in H^n \qquad \text{for all } f.$$

  By the lemma we can choose $f^*(t)$ such that:

  $$D_n(f^*)(\tau) = I. \tag{2.10}$$

  Since $(f^*, f^*) \in H^n$, it follows that:

  $$D_n(f^*)(\tau) = M \cdot D_n(f^*)(\tau) \quad \text{for some } M \in \Gamma(H^n). \tag{2.11}$$

  Substituting (2.10) into (2.11), we get:

  $$I = M \in \Gamma(H^n).$$

  $\Leftarrow$:

  $$\begin{aligned} D_n(f) = I \cdot D_n(f) \text{ and } I \in \Gamma(H^n) \quad &\Rightarrow \quad (f, f) \in H^n \text{ for all } f \\ &\Rightarrow \quad H^n \text{ is reflexive.} \end{aligned}$$

- H^n is symmetric $\iff$ $M \in \Gamma(H^n) \Rightarrow M^{-1} \in \Gamma(H^n)$.

  $\Rightarrow$:

  Notice that here we also need to insure the existence of M^{-1}.

Suppose H^n is symmetric and let $M \in \Gamma(H^n)$. By the lemma we can pick f and g so that $D_n(f)(\tau) = I$ and $D_n(g)(\tau) = M$. Hence by construction,

$$D_n(g)(\tau) = M \cdot D_n(f)(\tau).$$

Since $M \in \Gamma(H^n)$ it follows that $(g, f) \in H^n$, so by symmetry $(f, g) \in H^n$. Hence,

$$D_n(f)(\tau) = N \cdot D_n(g)(\tau) \quad \text{for some } N \in \Gamma(H^n).$$

Substituting the definitions of $D_n(f)$ and $D_n(g)$, we obtain

$$I = N \cdot M;$$

so,

$$M^{-1} = N \in \Gamma(H^n).$$

$\Leftarrow$:

Suppose $M \in \Gamma(H^n) \Rightarrow M^{-1} \in \Gamma(H^n)$. If $(g, f) \in H^n$, then

$$D_n(g) = M \cdot D_n(f) \quad \text{for some } M \in \Gamma(H^n)$$
$$\Rightarrow D_n(f) = M^{-1} D_n(g) \Rightarrow (f, g) \in H^n.$$

So H^n is symmetric.

- H^n is transitive $\iff$ $M, N \in \Gamma(H^n) \Rightarrow M \cdot N \in \Gamma(H^n)$.

$\Rightarrow$:

Let H^n be transitive and pick arbitrary $M, N \in \Gamma(H^n)$. By the lemma, we can choose $f(t), g(t)$ and $h(t)$ so that $D_n(h)(\tau) = I$, $D_n(g)(\tau) = N$, $D_n(f)(\tau) = M \cdot N$. Then, by construction,

$$D_n(g)(\tau) = N \cdot D_n(h)(\tau)$$

$$D_n(f)(\tau) = M \cdot D_n(g)(\tau).$$

Since $M, N \in \Gamma(H^n)$, it follows that $(f, g), (g, h) \in H^n$, so by transitivity $(f, h) \in H^n$. Therefore,

$$D_n(f)(\tau) = P \cdot D_n(h)(\tau) \quad \text{for some } P \in \Gamma(H^n).$$

Substituting the definitions of $D_n(f)$ and $D_n(h)$, we obtain

$$M \cdot N = P \in \Gamma(H^n).$$

$\Leftarrow$:

For the converse, suppose $\Gamma(H^n)$ is closed under multiplication. If $(f, g), (g, h) \in H^n$, then:

$$D_n(f) = M \cdot D_n(g); \quad \text{for some } M \in \Gamma(H^n)$$

$$D_n(g) = N \cdot D_n(h); \quad \text{for some } N \in \Gamma(H^n).$$

Therefore,

$$D_n(f) = M \cdot N \cdot D_n(h)$$

Since $M \cdot N \in \Gamma(H^n)$, it follows that $(f, h) \in H^n$, hence H^n is transitive $\square$.

2.3.2 Reparameterization Invariance

It is desired that our generic notion of continuity (H^n) be an intrinsic geometric property, independent of the particular parameterization of the curves. Thus, reparameterizing either of two H^n–continuous curves f, g by a reparameterization function h (Section 2.3.2) we expect H^n–continuity at τ to be preserved because we did not change the curve but only its parameterization.

We say that H^n–continuity is invariant under reparameterization if

$$(f, g) \in H^n \Rightarrow (f, g \circ h) \in H^n$$

where h is a reparameterization function. If, as we generally assume, H^n is symmetric, we can reparameterize f as well. Notice that if H^n is reparameterization invariant and h is a reparameterization function, then

$$\begin{aligned}(f, g \circ h) \in H^n &\Rightarrow (f, (g \circ h) \circ h^{-1}) \in H^n \text{ by reparametrization invariance} \\ &\Rightarrow (f, g) \in H^n\end{aligned}$$

so,

$$(f, g) \in H^n \iff (f, g \circ h) \in H^n.$$

Given a notion of geometric continuity H^n, the following proposition gives the algebraic condition that must be satisfied by the set of connection matrices ($\Gamma(H^n)$) corresponding to the reparameterization invariance of H^n.

> **Proposition 2.2** Suppose H^n is an equivalence relation (i.e. $\Gamma(H^n)$ is a group). Then H^n is invariant under reparameterization $\iff \Gamma(G^n) \subseteq \Gamma(H^n)$.

Proof $\Rightarrow$:

Since all matrices in $\Gamma(G^n)$ are of the form $R(h)(\tau)$ for some reparameterization function h, to show that $\Gamma(G^n) \subseteq \Gamma(H^n)$ it suffices to show that

$$D_n(f)(\tau) = R(h)(\tau) \cdot D_n(g)(\tau) \Rightarrow (f, g) \in H^n.$$

But by (2.3)

$$\begin{aligned}D_n(f)(\tau) = R(h)(\tau) \cdot D_n(g)(\tau) &\Rightarrow D_n(f)(\tau) = D_n(g \circ h)(\tau) \\ &\Rightarrow (f, g \circ h) \in H^n \quad \text{since } I \in \Gamma(H^n) \\ &\Rightarrow (f, g) \in H^n \quad \text{by reparametrization invariance.}\end{aligned}$$

$\Leftarrow$:

This is straightforward since $\Gamma(H^n)$ is a group. $\square$

2.3.3 Invariance under Affine Transformations

We have shown that if H^n is a notion of geometric continuity represented by connection matrices then H^n is invariant under linear transformations. However, this is not enough. What we really want is that H^n should be invariant under all affine transformations. That is, not only do we expect H^n-continuity to be independent of the orientation of the coordinate axes, but we also expect H^n–continuity to be independent of the choice of the coordinate origin. Again, we demand this invariance because by moving the origin we are not changing the curves, but only their representation. The following proposition shows how invariance of H^n–continuity under affine transformations is related to the structure of the connection matrices in $\Gamma(H^n)$.

Proposition 2.3 The following 3 conditions are equivalent:

1. H^n–continuity is invariant under affine transformations.
2. $D_n(c) = M \cdot D_n(c)$ for all constant functions c and all $M \in \Gamma(H^n)$.
3. $M_{i0} = \delta_{i0}$ for all $M \in \Gamma(H^n)$.

Proof $1 \iff 2$:

Let $\mathcal{A}(x) = xA + c$ be an affine transformation and let $M \in \Gamma(H^n)$. Suppose that

$$D_n(f) = M \cdot D_n(g) \quad \text{for some } M.$$

Then by the linearity of differentiation,

$$D_n(fA) = M \cdot D_n(gA)$$

Hence,

$$D_n(fA + c) = M \cdot D_n(gA + c) \iff D_n(c) = M \cdot D_n(c)$$

$2 \iff 3$:

This follows immediately since

$$[c, 0, \ldots, 0]^T = M \cdot [c, 0, \ldots, 0]^T \iff M_{i0} = \delta_{i0}. \square$$

As a consequence of invariance under affine transformations, we get invariance under embeddings. We say that H^n–continuity is invariant under embeddings iff for all matrices $M \in \Gamma(H^n)$ and every constant c,

(f, g) have connection matrix M at $\tau \Rightarrow$

$$((f, c), (g, c)) \text{ have connection matrix } M \text{ at } \tau.$$

Invariance under embeddings is an essential property of contact because it asserts that continuity does not depend on the ambient space of the curves but only on their intrinsic geometry.

> **Corollary 2.1** H^n is invariant under arbitrary embeddings $\iff$ H^n is invariant under affine transformations.

Proof The proof follows directly from part 2 of Proposition 2.3. $\square$

This result comes as no surprise because we can think of each translation of $\mathbb{R}^d$ as an embedding of $\mathbb{R}^d$ into a translated copy of itself in $\mathbb{R}^{d+1}$.

2.3.4 Invariance Under Projective Transformations

We expect any reasonable continuity measure to be preserved under perspective projection because ultimately what we actually see is not the curve itself but rather a projection of the curve onto either a graphics screen or the retina. Projective invariance has recently received much attention; both G^n and F^n have been shown to be projectively invariant [31, 49].

A general projective transformation Λ is a map

$$\Lambda \colon \{x_1, x_2, \ldots x_n\} \mapsto \{x_1^*, x_2^*, \ldots x_n^*\}$$

such that:

$$x_k^* = \frac{a_{k1}x_1 + a_{k2}x_2 + \cdots + a_{kn}x_n + b_k}{d_1x_1 + d_2x_2 + \cdots + d_nx_n + c} \quad k = 1, \ldots, n.$$

Let:

$$\begin{aligned} x_k^{**} &= a_{k1}x_1 + a_{k2}x_2 + \cdots + a_{kn}x_n + b_k, \qquad k = 1, \ldots, n \\ x_{n+1}^{**} &= d_1x_1 + d_2x_2 + \cdots + d_nx_n + c. \end{aligned}$$

Then, $x_k^* = \dfrac{x_k^{**}}{x_{n+1}^{**}}$. Now we can form the transformation Λ by composing four transformations:

$$\Lambda \equiv \Pi_4 \circ \Pi_3 \circ \Pi_2 \circ \Pi_1$$

where

$$\begin{aligned} \Pi_1 &: \{x_1, x_2, \ldots, x_n\} \mapsto \{x_1, x_2, \ldots, x_n, 0\} \\ \Pi_2 &: \{x_1, x_2, \ldots, x_n, 0\} \mapsto \{x_1^{**}, x_2^{**}, \ldots, x_n^{**}, x_{n+1}^{**}\} \\ \Pi_3 &: \{x_1^{**}, x_2^{**}, \ldots, x_n^{**}, x_{n+1}^{**}\} \mapsto \{x_1^*, x_2^*, \ldots, x_n^*, 1\} \\ \Pi_4 &: \{x_1^*, x_2^*, \ldots, x_n^*, 1\} \mapsto \{x_1^*, x_2^*, \ldots, x_n^*\}. \end{aligned}$$

Notice that Π_1 is an embedding, Π_2 is an affine transformation, and Π_4 is an orthogonal projection. Each of these transformations preserves continuity under our previous criteria (orthogonal projection works because continuity is defined coordinate-wise). All that remains to guarantee invariance under projective transformations is invariance under transformations of the form

$$\Pi_3 \colon \{y_1, y_2, \ldots, y_n, y_{n+1}\} \mapsto \{y_1/y_{n+1}, y_2/y_{n+1}, \ldots, y_n/y_{n+1}, 1\}$$

which represent the perspective projection of the point $\{y_1, y_2, \ldots, y_n, y_{n+1}\}$ from the origin onto the plane $y_{n+1} = 1$.

Thus, to insure projective invariance, we need to insure that if $f^* = (f, w)$ and $g^* = (g, u)$ are two curves such that

$$D_n(f^*)(\tau) = M \cdot D_n(g^*)(\tau) \quad M \in \Gamma(H^n),$$

then their perspective projections $(f/w, g/u)$ are related by

$$D_n(f/w)(\tau) = N \cdot D_n(g/u)(\tau) \quad \text{for some } N \in \Gamma(H^n).$$

Notice that $N \neq M$. In general, the connection matrix changes under perspective projection because, unlike affine transformations such as translations or rotations which alter only the relationship between the curve and the coordinate system, perspective projection changes the actual shape of the curve. Thus, we expect the connection matrix to change under perspective projections. We insist only that the resulting connection matrix belongs to the same group $\Gamma(H^n)$ before and after projection. In fact, it is known that the only notion of geometric continuity that preserves shape parameters on perspective projection is G^n. Thus the only group for which $N = M$ is $\Gamma(G^n)$ [31].

Below we give the conditions under which a group of connection matrices represents a notion of continuity that is projectively invariant. We begin by recalling the definition of the Leibniz matrix $L(w)(\tau)$, the matrix that results from applying Leibniz's rule to the product $w \cdot f$, i.e.

$$D_n(w \cdot f)(\tau) = L(w)(\tau) \cdot D_n(f)(\tau). \tag{2.12}$$

By Leibniz's rule:

$$L_{ij}(w)(\tau) = \begin{cases} \binom{i}{j} w^{(i-j)}(\tau) & i \geq j \\ 0 & i < j \end{cases} \tag{2.13}$$

Thus the matrix $L(w)(\tau)$ is lower triangular and is given by

$$L(w)(\tau) = \begin{bmatrix} w(\tau) & 0 & 0 & \dots & 0 \\ w'(\tau) & w(\tau) & 0 & \dots & 0 \\ w''(\tau) & 2w'(\tau) & w(\tau) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w^{(n)}(\tau) & nw^{(n-1)}(\tau) & \binom{n}{2}w^{(n-2)}(\tau) & \dots & w(\tau) \end{bmatrix}.$$

Proposition 2.4 An affinely invariant notion of geometric continuity H^n is projectively invariant iff:

$$D_n(w) = M \cdot D_n(u), M \in \Gamma(H^n) \Rightarrow (L(1/w) \cdot M \cdot L(u)) \in \Gamma(H^n).$$

Proof Let $f^* = (f, w)$ and $g^* = (g, u)$ be two curves meeting at a parameter value τ with H^n–continuity represented by the connection matrix $M \in \Gamma(H^n)$. Then at τ

$$D_n(f) = M \cdot D_n(g), \qquad D_n(w) = M \cdot D_n(u).$$

Therefore by (2.12)

$$\begin{aligned} D_n(f/w) &= L(1/w) \cdot D_n(f) \\ &= L(1/w) \cdot M \cdot D_n(g) \\ &= L(1/w) \cdot M \cdot L(u) \cdot D_n(g/u) \end{aligned}$$

so $(f/w), (g/u)$ meet at τ with connection matrix $L(1/w) \cdot M \cdot L(u)$. Hence, H^n is projectively invariant iff:

$$D_n(w) = M \cdot D_n(u), \quad M \in \Gamma(H^n) \Rightarrow (L(1/w) \cdot M \cdot L(u)) \in \Gamma(H^n). \square$$

2.3.5 Continuity of Osculating Linear Spaces

Geometrically, the continuity of the first n osculating linear spaces (O^n–continuity) is the weakest notion of n^{th} order contact. If the osculating linear spaces fail to agree, then there can be no sensible geometric contact between the curves.

At parameter τ, O^n–continuous curves have common position (C^0), tangent line, and higher order osculating linear spaces. Thus, $\Gamma(O^n)$ is the group of $(n+1)\times(n+1)$ nonsingular lower triangular connection matrices with first column $[1,0,\ldots,0]^T$.

Because O^n–continuity is so weak, we expect that any plausible notion of contact H^n will imply O^n–continuity. That is, we expect that H^n–continuity $\Rightarrow$ O^n–continuity, or equivalently $H^n \subseteq O^n$. In terms of connection matrices,

$$H^n \subseteq O^n \iff \Gamma(H^n) \subseteq \Gamma(O^n).$$

That is, $H^n \subseteq O^n \iff$ all $M \in \Gamma(H^n)$ are $(n+1)\times(n+1)$ nonsingular lower triangular connection matrices with $M_{i0} = \delta_{i0}$.

2.4 Summary and Commentary

To summarize, the table below shows the five analytical properties we require on a sensible geometric continuity measure H^n, together with the corresponding algebraic conditions on the set of connection matrices $\Gamma(H^n)$. Underlying these properties, we proved in Section 2.2.1 that a notion of geometric continuity H^n can be represented by a set of connection matrices $\Gamma(H^n)$ iff H^n is defined coordinate-wise and is invariant under linear transformations.

Property of H^n–contact	**Property of $\Gamma(H^n)$**
1. Equivalence relation	$\Gamma(H^n)$ is a group
2. Invariant under reparameterization	$\Gamma(G^n) \subseteq \Gamma(H^n)$
3. Invariant under affine maps	For all $M \in \Gamma(H^n) : M_{i0} = \delta_{i0}$
4. Invariant under projective maps	$M \in \Gamma(H^n)$, $D_n(w) = MD_n(u)$
	$\Rightarrow (L(1/w)\cdot M \cdot L(u)) \in \Gamma(H^n)$
5. $H^n \Rightarrow O^n$	For all $M \in \Gamma(H^n), M_{ij} = 0$ $(j > i)$

Properties (1) and (5) establish the basic correspondence between our intuition and the meaning of geometric continuity. The first says that geometric continuity is some form of contact which is an equivalence relation. The last asserts that contact must at least insure the continuity of osculating linear spaces and so relates contact back to geometry. Properties (2) and (3) mean that H^n is an intrinsic property independent of both curve parameterization (2) and coordinate system (3). The fourth property relates geometric continuity to vision; it says that a mathematically smooth curve should look smooth.

The five properties listed above for a generic notion of continuity H^n are not all independent since both property (4) and (5) imply property (3). However, the five properties of $\Gamma(H^n)$ are independent and together they insure the equivalent set of properties of H^n. Moreover, these properties of $\Gamma(H^n)$ are cumulative; the first k properties in the list for $\Gamma(H^n)$ are equivalent to the k^{th} property in the list for H^n. We shall discuss these five properties further in the next section along with some examples illustrating their independence.

Although these five properties are the defining characteristics of geometric continuity, nevertheless it is sometimes useful during the investigation of geometric continuity to study sets of connection matrices that lack one or more of these properties. For example, in the study of rational curves, we may wish to investigate curves that are not smooth, but which become smooth under the canonical perspective projection $\Pi\colon \mathbb{R}^{d+1} \mapsto \mathbb{R}^d$ given by

$$\Pi(f_1(t), \ldots, f_d(t), f_{d+1}(t)) = \left(\tfrac{f_1(t)}{f_{d+1}(t)}, \ldots, \tfrac{f_d(t)}{f_{d+1}(t)}\right).$$

Let $\Gamma(L^n)$ denote the group of all $(n+1) \times (n+1)$ Leibniz matrices $L(w)(\tau)$, where $w(\tau) \neq 0$. Then both Hohmeyer and Barsky [37] and Goldman and Micchelli [31] show that the curves which become G^n under this canonical projection are precisely those curves with connection matrices in

$$\Gamma(G^n)\Gamma(L^n) = \{M | M = GL \text{ where } G \in \Gamma(G^n), L \in \Gamma(L^n)\}.$$

More generally, if H^n is a projectively invariant form of geometric continuity, then the curves that become H^n under this canonical projection are precisely those curves with connection matrices in $\Gamma(H^n)\Gamma(L^n)$. This result is proved for F^n in Theorem 11 of [31]; the general result for H^n follows in much the same manner.

Notice, however, that matrices in $\Gamma(H^n)\Gamma(L^n)$ do not satisfy property 3 because $\Gamma(L^n) \subset \Gamma(H^n)\Gamma(L^n)$ and matrices in $\Gamma(L^n)$ violate this constraint. Thus the set $\Gamma(H^n)\Gamma(L^n)$ is not affinely invariant. This makes sense because the canonical projection is actually the perspective projection from the origin onto the plane $x_{d+1} = 1$. Thus the canonical projection depends on the choice of the origin, and so the set $\Gamma(H^n)\Gamma(L^n)$ is not affinely invariant.

Notice too that if $\Gamma(H^n)$ satisfies the five characteristic properties of geometric continuity listed above, then the set $\Gamma(H^n)\Gamma(L^n)$ does indeed satisfy each of the other four characteristic properties of *sets of connection matrices*. Property 5 follows because both $\Gamma(H^n)$ and $\Gamma(L^n)$ contain only lower triangular matrices and property 2 holds because $\Gamma(G^n) \subseteq \Gamma(H^n) \subset \Gamma(H^n)\Gamma(L^n)$. The fourth property is more subtle, but can proved using the fact that $\Gamma(H^n)$ satisfies this constraint. Finally property 1 follows from the equality $\Gamma(H^n)\Gamma(L^n) = \Gamma(L^n)\Gamma(H^n)$ which, in turn, is a consequence of property 4. Thus the connection matrices in $\Gamma(H^n)\Gamma(L^n)$ define a weak form of geometric continuity that is not affinely invariant but does satisfy the four remaining constraints.

We close this chapter by observing that even though our entire analysis has been performed only for parametric curves, almost everything carries over to parametric surfaces as well, though sometimes conditions on entries of M must be replaced by conditions on sub-blocks of M.

We argued in this chapter that any set of matrices that have the structure and satisfy the algebraic conditions stated in Section 2.3 constitutes an eligible candidate for a notion of geometric continuity. In Chapter 3 we capitalize on this formulation by actually finding groups of matrices with the required structure, thus building novel notions of contact, and consequently new types of spline curves.

Chapter 3

New Theories of Contact

In this chapter, we make use of the results of the axiomatization proposed in Chapter 2 to derive new examples of notions of geometric continuity. Our derivation is completely algebraic, and it builds on the properties of connection matrices developed in Chapter 2. Using these properties and examples, we go on to show that there is no notion of geometric continuity between G^3 and F^3; that there are several between G^4 and F^4; and that the number of different notions of geometric continuity between G^n and F^n grows at least exponentially with n. We also exhibit, in Section 3.3, a continuum of a two–parameter family of notions of geometric continuity that extends between G^4 and F^4.

3.1 New Notions of Geometric Continuity from Groups of Connection Matrices

Up to this point, the only groups of connection matrices we have encountered are those representing the three standard notions of geometric continuity: G^n, F^n and O^n. Each of these groups satisfies all the properties listed in Section 2.3. This is to be expected because these notions of contact are based on intrinsic geometric characteristics of curves. The most subtle property to verify is projective invariance. Goldman and Micchelli [31] give an analytic proof for Frenet frame continuity (F^n); Pottmann [49] provides a more geometric argument which he then uses to introduce additional projectively invariant notions of geometric continuity.

In this chapter we present a general technique for generating groups of connection matrices representing new notions of geometric continuity. In Sections 3.2 and 3.3 we use this technique to confirm the existence of classes of geometric continuity strictly between the classical notions of G^n, F^n and O^n.

To simplify our notation from here on, we shall often omit the zeroth row and column of a connection matrix since by properties (3) and (5) of the Section 2.3, the zeroth row and column of any connection matrix M are fixed ($M_{i0} = \delta_{i0}, M_{0i} = \delta_{0i}$).

Before proceeding, recall that continuity under reparameterization, G^n, is represented by the group $\Gamma(G^n)$ of β–matrices. Leaving off the zeroth row and column, we have $G \in \Gamma(G^n) \iff G$ is represented by an $n \times n$ matrix

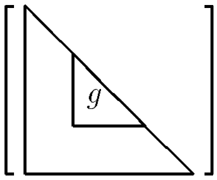
$$G = \begin{bmatrix} \beta_1 & 0 & \cdots & \cdots & 0 \\ \beta_2 & \beta_1^2 & \cdots & \cdots & 0 \\ \beta_3 & 3\beta_1\beta_2 & \beta_1^3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_n & \cdots & \cdots & \cdots & \beta_1^n \end{bmatrix},$$

where the entries of G are computed from the Chain Rule.

The matrices in $\Gamma(G^n)$ have a rich structure. Each $G \in \Gamma(G^n)$ is specified uniquely by the entries $G_{i1} = \beta_i, i = 1, \ldots, n$, and each entry $G_{ij}, i \geq j \geq 1$ depends only on $\beta_1, \beta_2, \ldots, \beta_{i-j+1}$ as can be verified by the Chain Rule. The matrices in $\Gamma(F^n)$ are less constrained; only the zeroth column and the diagonal entries are restricted to be of the same form as G, everything else – below the diagonal – is free.

To construct sets of connection matrices that have all the properties listed in Section 2.3, we find the hardest requirements to satisfy are the group property and the property corresponding to projective invariance. Shortly we shall describe a technique for building sets of connection matrices that satisfy both of these properties. We begin by constructing groups of matrices containing $\Gamma(G^n)$.

Let $G \in \Gamma(G^n)$ be an $n \times n$ matrix and consider matrices M of the form

$M =$ 

,

where the entries of M marked g match those in G, blank entries above the diagonal are zero, on the diagonal are free but nonzero, and below the diagonal are arbitrary. We call the seed matrix G a parent matrix for M, and refer to the lower triangular submatrix of M that matches that of the parent G as a *diagonal-block*. In fact, we shall allow arbitrarily many diagonal-blocks, of variable size, possibly overlapping, as long as each block comes from the same parent. Notice that parent matrices are not unique since each β_i in the first column of G affects only entries of distance $\geq (i-1)$ from the diagonal of G. Formally, we introduce:

Definition (S–construction): Fix $G \in \Gamma(G^n)$. Let $1 \leq r_1 < \cdots < r_p \leq n$, and $1 \leq s_1 < \cdots < s_p \leq n$ be two monotonic sequences of positive integers with $s_l \leq r_l$ for $l = 1, \ldots, p$. Suppose M is a nonsingular lower triangular matrix of the same size as G such that for $l = 1, \ldots, p$

$$M_{ij} = G_{ij} \quad s_l \leq j \leq i \leq r_l.$$

Suppose further that all the other entries of M below the diagonal (except for the fixed zeroth column) are free and the diagonal entries of M are non-zero. Then we call G a parent matrix of M, and denote the set of all such matrices M as the parent matrix G varies over $\Gamma(G^n)$ by $S_n[(r_1, s_1), (r_2, s_2), \ldots (r_p, s_p)]$. The parameters of $S_n[(r_1, s_1), (r_2, s_2), \ldots (r_p, s_p)]$ have the following interpretation:

- n denotes that matrices $M \in S_n$ are of size $n \times n$, or equivalently $(n+1) \times (n+1)$ when restoring the zeroth row and column.

- p denotes that there are p lower triangular submatrices of $M \in S_n$ matching those of the parent $G \in \Gamma(G^n)$ (p can be zero for none).

- (r_i, s_i) denote the indices of the vertex of the i^{th} diagonal-block of $M \in S_n$ that matches a diagonal-block in the parent $G \in \Gamma(G^n)$.

For example, omitting the zeroth row and column, we have

$$S_4[(4,3)] = \begin{bmatrix} * & & & \\ * & * & & \\ * & * & g & \\ * & * & g & g \end{bmatrix} = \begin{bmatrix} * & 0 & 0 & 0 \\ * & * & 0 & 0 \\ * & * & \beta_1^3 & 0 \\ * & * & 6\beta_1^2\beta_2 & \beta_1^4 \end{bmatrix}.$$

Notice too that by construction:

$$S_n[(r_1, s_1), \ldots, (r_p, s_p)] \subset \bigcap_{j=1}^{p} S[(r_j, s_j)].$$

Containment is strict because matrices in $\bigcap_{j=1}^{p} S[(r_j, s_j)]$ may have diagonal-blocks with different parents, while matrices in $S_n[(r_1, s_1), (r_2, s_2), \ldots, (r_p, s_p)]$ must have each diagonal-block come from the same parent.

Next we show that the sets of matrices built using the S–construction form groups.

Proposition 3.1 The set of matrices $S_n[(r_1, s_1), (r_2, s_2), \ldots (r_p, s_p)]$ forms a group. Moreover:

1. G is a parent of $M \Rightarrow G^{-1}$ is a parent of M^{-1}.
2. G is a parent of M, H is a parent of $N \Rightarrow G \cdot H$ is a parent of $M \cdot N$.

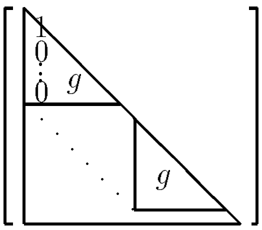
Proof We can ignore the zeroth row and column since these are fixed. Moreover it suffices to prove the proposition for individual diagonal-blocks because overlapping diagonal-blocks come from the same parent. The result is then straightforward because, by construction, these blocks are nonsingular, lower triangular, and lie along the diagonal. □

Not only do the sets of matrices generated using the S–construction form groups, these sets also satisfy most of the other properties listed in Section 2.3. Properties (3) and (5) are immediate from the construction. Property (2), $\Gamma(G^n) \subseteq S$, follows easily because we can instantiate free entries in $M \in S$ to the corresponding entries in its parent $G \in \Gamma(G^n)$, or in other words because G is a parent of G.

It remains to determine which sets built using the S–construction correspond to projectively invariant notions of continuity. For notational convenience we define:

$$S_{n,k}[(r_1, s_1), \ldots, (r_p, s_p)] = S_n[(k, 1), (r_1, s_1), \ldots, (r_p, s_p)]$$

with the restriction that for $i = 1, \ldots, p$ and $0 \leq (r_i - s_i) \leq k$. We shall show shortly that these sets correspond to projectively invariant notions of geometric continuity. Graphically this construction corresponds to sets of matrices of the form

$M =$ 

,

where entries marked g match those in G and for clarity we have restored the zeroth row and column of M. Again we may have arbitrarily many (possibly none or overlapping) diagonal-blocks but each diagonal-block must be no bigger than the one in the upper left corner which includes the zeroth column; everything else below the diagonal , except for the zeroth column, is free and diagonal entries are nonzero.

Notice that the three standard notions of geometric continuity are instances of the $S_{n,k}$–construction since:

- $\Gamma(G^n) = S_n[(n,1)] = S_{n,n}[()]$,
- $\Gamma(F^n) = S_n[(1,1),(2,2),\ldots,(n,n)] = S_{n,1}[(2,2),\ldots,(n,n)]$,
- $\Gamma(O^n) = S_n[()] = S_{n,1}[()]$.

Other examples will appear later in Section 3.3.2.

Next we show that groups of connection matrices that are of the form $S_{n,k}[(r_1,s_1),\ldots,(r_p,s_p)]$ correspond to projectively invariant notions of geometric continuity. We begin with a simple observation.

Let L, M and N be three lower triangular matrices, and let $P = L \cdot M \cdot N$. Then

$$P_{rs} = \sum_{i=s}^{r}\sum_{j=s}^{i} L_{r,i} M_{i,j} N_{j,s}. \tag{3.1}$$

This follows since for arbitrary matrices L, M, N

$$P_{rs} = \sum_{i=0}^{n}\sum_{j=0}^{n} L_{r,i} M_{i,j} N_{j,s},$$

and L, M, N are lower triangular. In particular, $P_{r,s}$ depends on $M_{r,s}$ and entries to the right and above but no other entries in M.

The next proposition uses the result by Goldman and Micchelli [31] which states that G^n–continuity is invariant under perspective projection and moreover the shape parameters do not change. In other words,

$$G \in \Gamma(G^n), D_n(w)(\tau) = G \cdot D_n(u)(\tau) \Rightarrow G = L(1/w) \cdot G \cdot L(u) \tag{3.2}$$

where the matrix $L(w)$ is the Leibniz matrix defined by equations (2.12), (2.13).

Proposition 3.2 Let $M \in S_{n,k}[(r_1,s_1),(r_2,s_2),\ldots,(r_p,s_p)]$ with parent matrix $G \in \Gamma(G^n)$ and let $D_n(w) = M \cdot D_n(u)$. Then

$$N = L(1/w) \cdot M \cdot L(u) \in S_{n,k}[(r_1,s_1),(r_2,s_2),\ldots,(r_p,s_p)].$$

Moreover, G is a parent of N.

Proof Construct w^* such that:

$$D_n(w^*) = G \cdot D_n(u).$$

Since the first $k+1$ rows of M and G are identical and

$$D_n(w) = M \cdot D_n(u),$$

it follows that

$$(w^*)^{(j)} = (w)^{(j)} \quad j = 0, \ldots, k.$$

Together with (2.13) and the fact that $(1/w)^{(p)}$ depends only on $w^{(j)}$ $j = 0, \ldots, p$, this implies that

$$L_{ij}(1/w^*) = L_{ij}(1/w) \quad 0 \leq (i-j) \leq k. \tag{3.3}$$

By (3.2) we know that $L(1/w^*) \cdot G \cdot L(u) = G$. Thus, by (3.1), we have

$$\begin{aligned} N_{rs} &= \sum_{i=s}^{r} \sum_{j=s}^{i} L_{r,i}(1/w) M_{i,j} L_{j,s}(u) \\ G_{rs} &= \sum_{i=s}^{r} \sum_{j=s}^{i} L_{r,i}(1/w^*) G_{i,j} L_{j,s}(u). \end{aligned}$$

Let $(r_0, s_0) = (k, 0)$ and consider the entries $N_{r,s}$, where $s_q \leq s \leq r \leq r_q$ and $q = 0, \ldots, p$. From (3.3) and the fact that $r_q - s_q \leq k$, it follows that

$$L_{r,i}(1/w) = L_{r,i}(1/w^*) \quad s_q \leq i \leq r \leq r_q.$$

Moreover by the construction of M

$$M_{ij} = G_{ij} \quad s_q \leq j \leq i \leq r_q.$$

Hence,

$$N_{rs} = G_{rs} \quad s_q \leq s \leq r \leq r_q \text{ and } q = 0, \ldots, p.$$

It follows that $N \in S_{n,k}[(r_1, s_1), \ldots, (r_p, s_p)]$, and that G is a parent of N. $\square$

Theorem 3.1 The set of matrices $S_{n,k}[(r_1, s_1), \ldots, (r_p, s_p)]$ represents a projectively invariant notion of geometric continuity.

Proof From the construction of $S_{n,k}$ and Propositions 3.1 and 3.2, the set $S_{n,k}$ satisfies all five properties listed in Section 2.3 and so represents a projectively invariant notion of geometric continuity. $\Box$

In fact, we can say more about matrices resulting from the projective invariance constraint. Let $M \in S_{n,k}[(r_1, s_1), \ldots, (r_p, s_p)]$ and construct M' so that the entries of M' are the same as the entries of M except at some of the *vertices* of diagonal-blocks. We claim that the *value* of M'_{r_k,s_k} is preserved under the construction corresponding to projection as shown by the proposition below.

Proposition 3.3 Let $M \in S_{n,k}[(r_1, s_1), (r_2, s_2), \ldots, (r_p, s_p)]$ and construct M' so that:

$$\begin{aligned} M'_{r_q,s_q} &\neq M_{r_q,s_q} && \text{for some values of } q \text{ in } \{1, \ldots, p\} \\ M'_{ij} &= M_{ij} && \text{otherwise.} \end{aligned}$$

If $D_n(w) = M' \cdot D_n(u)$ and $N' = L(1/w) \cdot M' \cdot L(u)$, then

$$N'_{r_q,s_q} = M'_{r_q,s_q} \quad q = 1, \ldots, p.$$

Moreover, $N'_{r,s} = M'_{r,s}$ whenever $M'_{r,s}$ is an entry in a diagonal-block of M'.

Proof By Proposition 3.2, we already know that whenever $M'_{r,s}$ is an entry in a diagonal-block of M' then $N'_{r,s} = M'_{r,s}$. It remains to show that $N'_{r_q,s_q} = M'_{r_q,s_q}$ when $M'_{r_q,s_q} \neq M_{r_q,s_q}$.

By (3.1)

$$N'_{r_q,s_q} = \sum_{i=s_q}^{r_q} \sum_{j=s_q}^{i} M'_{i,j} L_{j,s_q}(u) L_{r_q,i}(1/w). \tag{3.4}$$

Now let G be a parent matrix of M. Then, as in the proof of Proposition 3.2, we can build w^* such that

$$L_{ij}(1/w^*) \;=\; L_{ij}(1/w) \quad 0 \le (i-j) \le k, \tag{3.5}$$

and for all r, s,

$$G_{r,s} \;=\; \sum_{i=s}^{r}\sum_{j=s}^{i} G_{i,j}L_{j,s}(u)L_{r,i}(1/w^*). \tag{3.6}$$

By construction,

$$M'_{ij} = M_{ij} = G_{ij} \qquad s_q \le j \le i \le r_q - 1 \text{ and } s_q + 1 \le j \le i = r_q$$

Let $M'_{r_q,s_q} = G_{r_q,s_q} + \Delta_{r_q,s_q}$. Then by (3.4) and (3.5)

$$N'_{r_q,s_q} = \sum_{i=s_q}^{r_q}\sum_{j=s_q}^{i} G_{ij}L_{js_q}(u)L_{r_q,i}(1/w^*) + L_{r_q,r_q}(1/w)L_{s_q,s_q}(u)\Delta_{r_q,s_q}. \tag{3.7}$$

Now by (2.13) we have,

$$L_{r_q,r_q}(1/w) = 1/w \text{ and } L_{s_q,s_q}(u) = u.$$

Since $D_n(w) = M' \cdot D_n(u)$ and $M'_{00} = 1$, it follows that at the point of evaluation $w = u$; hence

$$L_{r_q,r_q}(1/w)L_{s_q,s_q}(u) = 1 \cdot$$

Thus by (3.6) and (3.7)

$$N'_{r_q,s_q} = G_{r_q,s_q} + \Delta_{r_q,s_q} = M'_{r_q,s_q}. \square$$

From the proof of Proposition 3.2 we conclude that entries in diagonal-blocks of matrices in $S_{n,k}[(r_1,s_1),\ldots,(r_p,s_p)]$ are invariant under projection. Proposition 3.3 says something more. Let $S^*_{n,k}[(r_1,s_1),\ldots,(r_p,s_p)]$ be a set of matrices defined the same way as $S_{n,k}[(r_1,s_1),\ldots,(r_p,s_p)]$ except that the constraints on the entries

(r_i, s_i), for some values of i in $\{1, \ldots, p\}$, are relaxed so that the (r_i, s_i)th entry is either free or depends only on entries in diagonal-blocks. This set need not form a group nor contain $\Gamma(G^n)$. But proposition 3.3 asserts that if such a set forms a group and contains $\Gamma(G^n)$, then it represents a projectively invariant notion of geometric continuity. We shall see later on (Examples I.1–I.4) that this result actually comes in handy for proving the projective invariance of notions of geometric continuity embodied by groups of connection matrices.

3.2 $F^n \subset H^n \subset O^n$

We start by showing that there are notions of geometric continuity H^n that lie strictly between F^n and O^n by constructing groups of connection matrices $\Gamma(H^n)$ such that $\Gamma(F^n) \subset \Gamma(H^n) \subset \Gamma(O^n)$.

Example I.1 Consider the geometric continuity measure H^n with connection matrices $\Gamma(H^n)$ of the form

$$M = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \beta_1 & 0 & \cdots & 0 \\ 0 & \beta_2 & \pm\beta_1^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \beta_n & \cdots & \cdots & \pm\beta_1^n \end{bmatrix}.$$

It is easy to check that $\Gamma(H^n)$ satisfies all our criteria. The only nontrivial property to verify is projective invariance which follows by Proposition 3.3.

The group $\Gamma(H^2)$ is represented by the set of connection matrices

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \beta_1 & 0 \\ 0 & \beta_2 & \pm\beta_1^2 \end{bmatrix}.$$

Let

$$D_n(f^+) = MD_n(f^-), M \in \Gamma(H^2).$$

Then

$$\begin{aligned} f^+ &= f^- \\ f^{+\prime} &= \beta_1 f^{-\prime} \\ f^{+\prime\prime} &= \beta_2 f^{-\prime} \pm \beta_1^2 f^{-\prime\prime}, \end{aligned}$$

so the curvature vectors κ are related by

$$\begin{aligned} \kappa^+ &= \frac{(f^+)' \times (f^+)''}{\|f^{+\prime}\|^3} \\ &= \frac{\beta_1 f^{-\prime} \times (\beta_2 f^{-\prime} \pm \beta_1^2 f^{-\prime\prime})}{\|\beta_1 f^{-\prime}\|^3} \\ &= \pm\kappa^-. \end{aligned}$$

Notice that F^2 continuity is stronger than H^2 because F^2 requires matching the curvatures exactly so that $\kappa^+ = \kappa^-$.

Other examples can easily be constructed that satisfy our algebraic requirements by freeing some entries along the diagonal. Using the S notation we have $F^n \subseteq S_n[(r_1, r_1), \ldots, (r_p, r_p)] \subseteq O^n$.

3.3 $G^n \subset H^n \subset F^n$

Now we turn our attention to continuity measures between G^n and F^n. We know that $\Gamma(G^2) = \Gamma(F^2)$, so there can be no groups in between. What groups of connection matrices lie between $\Gamma(G^n)$ and $\Gamma(F^n)$ for $n \geq 3$? We address this question first for continuity of order 3 and later on for order $n > 3$.

3.3.1 $G^3 \subset H^3 \subset F^3$

Here we prove that there is no notion of geometric continuity that satisfies our criteria and lies strictly between G^3 and F^3. From the point of view of geometric invariants like curvature and torsion, this result is important because most curves in CAGD lie in 3-space.

> **Theorem 3.2** There are no geometric continuity measures H^3 strictly between G^3 and F^3.

Proof The proof proceeds by showing that there is no group of 4×4 matrices $\Gamma(H^3)$ strictly between $\Gamma(G^3)$ and $\Gamma(F^3)$. We do this by adding one arbitrary non–β matrix, $A \in \Gamma(F^3) - \Gamma(G^3)$, to $\Gamma(G^3)$ and showing that we can then obtain any other matrix in $\Gamma(F^3)$ using only multiplication by matrices in $\Gamma(G^3)$.

For notational convenience, we shall again adopt the convention of omitting the zeroth row and column of the connection matrices since these are fixed.

Consider an arbitrary matrix $A \in \Gamma(F^3) - \Gamma(G^3)$; then

$$A = \begin{bmatrix} \beta_1 & 0 & 0 \\ \beta_2 & \beta_1^2 & 0 \\ \beta_3 & 3\beta_1\beta_2 + \delta & \beta_1^3 \end{bmatrix}, \qquad \beta_1 > 0, \delta \neq 0.$$

This matrix A can be factored as

$$A = \begin{bmatrix} \beta_1 & 0 & 0 \\ \beta_2 & \beta_1^2 & 0 \\ \beta_3 & 3\beta_1\beta_2 & \beta_1^3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{\delta}{\beta_1^3} & 1 \end{bmatrix}.$$

Call the second matrix on the right hand side D.

Now we show how to get any other matrix $C \in \Gamma(F^3) - \Gamma(G^3)$, from D and matrices in $\Gamma(G^n)$. For such matrix (C) we can write:

$$C = \begin{bmatrix} \gamma_1 & 0 & 0 \\ \gamma_2 & \gamma_1^2 & 0 \\ \gamma_3 & 3\gamma_1\gamma_2 + \sigma & \gamma_1^3 \end{bmatrix} = \begin{bmatrix} \gamma_1 & 0 & 0 \\ \gamma_2 & \gamma_1^2 & 0 \\ \gamma_3 & 3\gamma_1\gamma_2 & \gamma_1^3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{\sigma}{\gamma_1^3} & 1 \end{bmatrix}.$$

Since the first factor is in $\Gamma(G^3)$, it suffices to show how to construct the second factor from D and the β–matrices using matrix multiplication and inversion. We proceed in the following manner:

1. If $e_1 = \delta/\beta_1^3$ and $e_2 = \sigma/\gamma_1^3$ have the same signs, go to the next step. Otherwise, notice that

$$D^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{\delta}{\beta_1^3} & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{\delta}{\beta_1^3} & 1 \end{bmatrix}.$$

   But $D^{-1} \in \Gamma(H^3)$ since $\Gamma(H^3)$ is a group. Let $e_1 = -\delta/\beta_1^3$. Then e_1 has the same sign as $e_2 = \sigma/\gamma_1^3$.

2. Compute

$$\begin{aligned} P &= \begin{bmatrix} \alpha_1 & 0 & 0 \\ \alpha_2 & \alpha_1^2 & 0 \\ \alpha_3 & 3\alpha_1\alpha_2 & \alpha_1^3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & e_1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 & 0 & 0 \\ \alpha_2 & \alpha_1^2 & 0 \\ \alpha_3 & 3\alpha_1\alpha_2 & \alpha_1^3 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\alpha_2 e_1 & \alpha_1 e_1 & 1 \end{bmatrix}. \end{aligned}$$

Now $P \in \Gamma(H^3)$ since each of its factors is in $\Gamma(H^3)$. If we pick $\alpha_1 = e_2/e_1 > 0$ and $\alpha_2 = 0$, then we have

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{\sigma}{\gamma_1^3} & 1 \end{bmatrix},$$

which is all that is required to compute the matrix C. □

In fact Theorem 3.2 proves a stronger result than stated. The proof shows that there are no *groups* of 4×4 connection matrices strictly between $\Gamma(G^3)$ and $\Gamma(F^3)$. Thus even if we drop the requirement of projective invariance we still cannot find a notion of geometric continuity strictly between G^3 and F^3.

3.3.2 $G^4 \subset H^4 \subset F^4$

Although there are no groups of connection matrices between $\Gamma(G^3)$ and $\Gamma(F^3)$, the situation changes quickly for higher orders.

Example I.2 An interesting example appears in Pottmann [49] where he considers the tangent surface defined by the lines tangent to a given curve. Pottmann constructs connection matrices to describe the continuity of parametric curves by requiring that these tangent surfaces meet smoothly. His connection matrices for continuity of order 4 are given by

$$M = \begin{bmatrix} \beta_1 & 0 & 0 & 0 \\ \beta_2 & \beta_1^2 & 0 & 0 \\ \beta_3 & \beta_5 & \beta_1^3 & 0 \\ \beta_4 & \beta_6 & 3(\beta_1\beta_5 - \beta_1^2\beta_2) & \beta_1^4 \end{bmatrix}.$$

It is easy to verify that this set of matrices forms a group and that this group contains $\Gamma(G^4)$. The remaining properties of geometric continuity other than projective invariance are immediate from the construction. Projective invariance follows because by Proposition 3.3 the values directly below the main diagonal remain invariant under projection. This is an example of a set of connection matrices that satisfies all the properties listed in Section 2.3 but is not generated by the S-construction.

Example I.3 If in Example I.2 we pick $\beta_5 = 3\beta_1\beta_2$, we get another continuity measure, represented by the set of matrices of the form

$$M = \begin{bmatrix} \beta_1 & 0 & 0 & 0 \\ \beta_2 & \beta_1^2 & 0 & 0 \\ \beta_3 & 3\beta_1\beta_2 & \beta_1^3 & 0 \\ \beta_4 & \beta_5 & 6\beta_1^2\beta_2 & \beta_1^4 \end{bmatrix}.$$

This set is just $S_{4,3}[(4,3)]$, so by Theorem 3.1 it too satisfies all five properties listed in Section 2.3.

The S-construction provides us with two additional examples of projectively invariant notions of geometric continuity between G^4 and F^4: $S_{4,2}[(4,3)]$ and $S_{4,3}[(4,4)]$.

These four groups, however, do not exhaust all projectively invariant notions of geometric continuity between G^4 and F^4. Actually the next example demonstrates a continuum of a two–parameter family of notions of geometric continuity that includes both Example I.2 and Example I.3.

Example I.4 This family of notions of geometric continuity is represented by the set of connection matrices of the form:

$$M = \begin{bmatrix} \beta_1 & 0 & 0 & 0 \\ \beta_2 & \beta_1^2 & 0 & 0 \\ \beta_3 & \gamma & \beta_1^3 & 0 \\ \beta_4 & \beta_6 & C\beta_1\gamma + D\beta_1^2\beta_2 & \beta_1^4 \end{bmatrix}.$$

For arbitrary but fixed values of C, D, one can verify by matrix multiplication that the set of matrices of the form M forms a group. The rest of the algebraic conditions of Section 2.3 are easy to verify. Projective invariance follows from Proposition 3.3. By instantiating γ to $3\beta_1\beta_2$, C to 2, D to zero we get Example I.3. Instantiating γ to β_5, C to 3, and D to -3 results in Example I.2. Thus there is a two–parameter family of notions of geometric contact between G^4 and F^4.

3.3.3 Higher Order Continuity

We now show that the number of new notions of geometric continuity between G^n and F^n grows very rapidly with n.

Theorem 3.3 The number of new notions of geometric continuity between G^n and F^n increases at least exponentially with n (when $n \geq 5$).

Proof Let f_{n-1} denote the number of notions of continuity between G^{n-1} and F^{n-1}. Then we can build at least $2 * f_{n-1}$ new notions of continuity between G^n and F^n as follows: Let $\Gamma(G^{n-1}) \subset \Gamma(H^{n-1}) \subset \Gamma(F^{n-1})$ and define $\Gamma^*(H^{n-1})$ to be the collection of all $(n+1) \times (n+1)$ lower triangular matrices M^* such that:

$$\begin{aligned} M^*_{i,j} &= M_{i,j} \quad i,j = 0,\ldots,n-1 \quad \text{for some } M \in \Gamma(H^{n-1}) \\ M^*_{n,0} &= 0 \\ M^*_{n,n} &\neq 0 \\ M^*_{i,n} &= 0 \quad i = 0,\ldots,n-1. \end{aligned}$$

Then $\Gamma^*(H^{n-1})$ represents a projectively invariant notion of geometric continuity of order n. However $\Gamma^*(H^{n-1})$ is not contained in $\Gamma(F^n)$. To correct this defect, let

$$\begin{aligned} S^1(H^{n-1}) &= \Gamma^*(H^{n-1}) \cap S_{n,1}[(n,n)] \\ S^2(H^{n-1}) &= \Gamma^*(H^{n-1}) \cap S_{n,2}[(n,n-1)]. \end{aligned}$$

By construction both $S^1(H^{n-1}), S^2(H^{n-1}) \subset \Gamma(F^n)$. Moreover $S^1(H^{n-1})$ and $S^2(H^{n-1})$ are intersections of pairs of groups representing projectively invariant notions of geometric continuity of order n and hence they also represent projectively invariant notions of geometric continuity of order n. Notice too that if K^{n-1} and H^{n-1} are two distinct notions of geometric continuity between G^{n-1} and F^{n-1}, then $S^j(K^{n-1}) \neq S^j(H^{n-1})$ because the with $S_{n,j}$ affects only the n^{th} row for $j = 1, 2$; entries along the diagonal in the $(n-1)$st row are already constrained since, by assumption, $\Gamma(K^{n-1}), \Gamma(H^{-1}) \subset \Gamma(F^{n-1})$. Hence,

$$f_n \geq 2 * f_{n-1}.$$

Thus the number f_n of notions of geometric continuity between G^n and F^n is at least $O(2^n)$. $\square$

Up to this point all the new notions of geometric continuity that we developed were derived from an entirely algebraic approach. This was done by finding groups of connection matrices that satisfy the five algebraic conditions that guarantee, as we argued in Section 2.3, that the corresponding notion of geometric continuity is sensible. In general, a geometric interpretation in terms of geometric invariants preserved by each of these new notions is not clearly understood. In the next chapter we go back to geometry and associate a class of new notions of geometric continuity – built using our entirely algebraic approach – with a particular set of geometric invariants, thus adding geometric significance to the algebraic approach.

Chapter 4

Notions of Geometric Continuity Based on Geometric Invariants

In this chapter we develop a spectrum of new notions of geometric continuity that bridge the gap between the two previously isolated theories of G^n and F^n–continuity. Going back to the interpretation of notions of geometric continuity in terms of the geometric invariants they preserve, we recall from Section 2.2.2 that G^n continuity is equivalent to continuity of the curvature and its first $n-1$ arc length derivatives, while F^n continuity is equivalent to the continuity of the first $n-1$ higher order curvatures.

We study, in this chapter, notions of geometric continuity that lie between G^n and F^n and thus link these two isolated theories by considering the continuity of the arc length derivatives of the higher order curvatures. We begin by constructing the set of connection matrices that preserves the continuity of the higher order curvatures and their first order arc length derivatives (Section 4.2). Then, in Section 4.3, we extend this construction to the continuity of arbitrary higher order arc length derivatives of the curvatures. We shall show that the continuity of the first p arc length derivatives of the high order curvatures is preserved if and only if the corresponding set of lower triangular connection matrices agree with the β–matrices along their first $p+1$ main diagonals.

4.1 Connection Matrices and Higher Order Curvatures

We begin by introducing some notation and conventions. In this chapter, it will be more convenient to slightly modify the notation introduced in Section 2.2.1. In contrast to Chapters 2 and 3, we shall not deal with individual coordinate functions but with derivatives of a curve $\mathbf{x}(t)$. Thus, to keep in line with the standard conventions of using superscripts for derivatives, the $+/-$ sidedness symbol will be moved to the subscript position. Let $\mathbf{x}\colon \mathbb{R}^1 \mapsto \mathbb{R}^d$ be an n–times differentiable function. Then, for derivatives of $\mathbf{x}$ with respect to arc length, we shall write $\dot{\mathbf{x}}$ to denote first derivative, and $\mathbf{x}^{[k]}$ for higher order derivatives; we reserve $\mathbf{x}'$ and $\mathbf{x}^{(k)}$ for ordinary parametric derivatives. In this chapter we keep following the convention of omitting the zeroth row and column of a connection matrix. Thus, a continuous piecewise n–times differentiable function $\mathbf{x}\colon \mathbb{R}^1 \mapsto \mathbb{R}^d$ has connection matrix $M = (M_{ij}), i, j = 1, \ldots, n$, at parameter value τ if and only if

$$[\mathbf{x}_+^{(1)}(\tau), \ldots, \mathbf{x}_+^{(n)}(\tau)]^T = M \cdot [\mathbf{x}_-^{(1)}(\tau), \ldots, \mathbf{x}_-^{(n)}(\tau)]^T.$$

where $\mathbf{x}_-$ and $\mathbf{x}_+$ denote the curve $\mathbf{x}$ to the left and to the right of τ respectively. Unless otherwise stated, we shall henceforth write $\mathbf{x}$ to denote $\mathbf{x}(t)$ and $\mathbf{x}_\pm$ to denote $\mathbf{x}_\pm(\tau)$.

In this chapter we address the continuity of curvatures and their arc length derivatives. The first curvature measures the deviation of the curve from its tangent line; the second curvature – often called torsion – measures the deviation of the curve from its osculating plane. In general, the m^{th} curvature measures the deviation of the curve from its m^{th} osculating flat. All these higher order curvatures are geometric invariants; that is, they are all independent of the specific parameterization used to represent the curve. Since arc length is also a geometric invariant, all the arc length derivatives of the higher order curvatures are again geometric invariants. We are going to investigate the continuity of these geometric invariants.

There are many formulas for the curvatures in the literature; we shall use the definition given in [22, 54]. Let $\mathbf{x}(t) = \{x_1(t), x_2(t), \ldots, x_n(t)\}\colon \mathbb{R}^1 \mapsto \mathbb{R}^n$ be a parametric curve. If at some point $t = \tau$ the first n derivative vectors $\mathbf{x}^{(1)}(\tau), \mathbf{x}^{(2)}(\tau), \ldots, \mathbf{x}^{(n)}(\tau)$

are linearly independent, then we can define $n-1$ curvatures $\kappa_1, \kappa_2, \ldots, \kappa_{n-1}$ at τ by

$$\kappa_i(\tau) = \frac{\sqrt{g_{i-1}(\tau) g_{i+1}(\tau)}}{||\mathbf{x}^{(1)}(\tau)|| g_i(\tau)} \qquad i = 1, \ldots, n-1 \tag{4.1}$$

where

$$g_i(t) = \det(A); \quad A_{j,k} =< \mathbf{x}^{(j)}(t), \mathbf{x}^{(k)}(t) > \qquad j, k = 1, \ldots, i. \tag{4.2}$$

Here $< \bullet, \bullet >$ denotes the dot product on $\mathbb{R}^n$, and g_0 is defined to be the constant 1. Henceforth we shall always assume that the derivative vectors are linearly independent at τ and thus that we can compute the curvatures using (4.1). Notice that in $\mathbb{R}^3$, formula (4.1) yields the standard calculus definitions of κ_1 as curvature and κ_2 as torsion.

Formula (4.2) for computing $g_i(t)$ can be simplified if we notice that when $\mathbf{x}(t) \in \mathbb{R}^n$

$$g_n(t) = \det(\left[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(n)}\right]^T \cdot \left[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(n)}\right]).$$

Hence,

$$g_n(t) = \det{}^2 \left[\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(n)}\right]. \tag{4.3}$$

For $i < n$ we do not have (4.3), but a simple variation is given below in Lemma 4.1. First, however, we need to recall the formula of Cauchy-Binet [9].

Let A, B and C be matrices of size $m \times r$, $m \times n$ and $n \times r$ respectively such that $A = BC$, and let $A \begin{pmatrix} i_1, i_2, \ldots, i_p \\ j_1, j_2, \ldots, j_p \end{pmatrix}$ denote the determinant obtained from A by deleting all rows and columns except those labeled $i_1, i_2, \ldots, i_p$ and $j_1, j_2, \ldots, j_p$ respectively. Then the Cauchy-Binet formula states that

$$A \begin{pmatrix} i_1, i_2, \ldots, i_p \\ j_1, j_2, \ldots, j_p \end{pmatrix} = \sum_{\alpha_1 < \cdots < \alpha_p} B \begin{pmatrix} i_1, i_2, \ldots, i_p \\ \alpha_1, \alpha_2, \ldots, \alpha_p \end{pmatrix} C \begin{pmatrix} \alpha_1, \alpha_2, \ldots, \alpha_p \\ j_1, j_2, \ldots, j_p \end{pmatrix}$$

where the sum is over all subsequences of $1, \ldots, n$ of length p.

Lemma 4.1 Let $\mathbf{x}(t)$ be a parametric curve in $\mathbb{R}^n$ and let $g_m(t)$ $m \le n$ be given by (4.2). Then $g_m(t)$ can also be written as,

$$g_m(t) = \sum_\alpha g_{m,\alpha}(t) = \sum_\alpha \det{}^2 \left[\mathbf{x}_\alpha^{(1)}, \mathbf{x}_\alpha^{(2)}, \ldots, \mathbf{x}_\alpha^{(m)}\right] \qquad (4.4)$$

where the sum is over all subsequences $\alpha = [\alpha_1, \ldots, \alpha_m]$ of $1, \ldots, n$ of length m, and $g_{m,\alpha}(t)$ is the result of computing the expression (4.1) for $\mathbf{x}_\alpha(t) = \{x_{\alpha_1}(t), \ldots, x_{\alpha_m}(t)\}$.

Proof Let $\mathbf{x}(t) = \{x_1(t), x_2(t), \ldots, x_n(t)\}$, and let

$$A = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & & \vdots \\ x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix}.$$

Then $g_m = \det\,(A \cdot A^T)$. Applying the Cauchy-Binet formula gives the result. □

Notice that the functions $g_{m,\alpha}(t)$ that appear in (4.4) can be evaluated using (4.3), since the curves $\mathbf{x}_\alpha(t)$ lie in $\mathbb{R}^m$. Moreover, because each coordinate function of $\mathbf{x}_\alpha(t)$ is also a coordinate function of $\mathbf{x}(t)$, each curve $\mathbf{x}_\alpha(t)$ has the same connection matrix as $\mathbf{x}(t)$ at $t = \tau$.

4.2 First Derivatives of Curvatures

Here we show how to specify connection matrices so that the higher order curvatures and their first derivatives with respect to arc length are continuous. To proceed, we need a formula for the derivative of the m^{th} curvature, κ_m, with respect to arc length. The following proposition gives a simple formula to compute this derivative.

Proposition 4.1 Let $\mathbf{x}(t)$ be a curve in $\mathbb{R}^n$. Then,

$$\dot{\kappa_m} = \frac{\kappa_m}{||\mathbf{x}'||} \cdot \left[\frac{1}{2}\left(\frac{g'_{m-1}}{g_{m-1}} + \frac{g'_{m+1}}{g_{m+1}} - \frac{g'_1}{g_1}\right) - \frac{g'_m}{g_m}\right] \quad m = 1, \ldots, n-2 \quad (4.5)$$

where $\dot{\kappa_m}$ denotes derivative with respect to the arc length s.

Proof By the chain rule and equation (4.1),

$$
\begin{aligned}
\dot{\kappa_m} &= \frac{d\kappa_m}{dt} / \frac{ds}{dt} \\
&= \left(\frac{\sqrt{g_{m-1}g_{m+1}}}{||\mathbf{x}'||g_m} \right)' \cdot \frac{1}{||\mathbf{x}'||} \\
&= \frac{||\mathbf{x}'||g_m \dfrac{(g_{m-1}g_{m+1})'}{2\sqrt{g_{m-1}g_{m+1}}} - \sqrt{g_{m-1}g_{m+1}}(g_m \dfrac{\mathbf{x}' \cdot \mathbf{x}''}{||\mathbf{x}'||} + ||\mathbf{x}'||g_m')}{||\mathbf{x}'||^2 g_m^2} \cdot \frac{1}{||\mathbf{x}'||} \\
&= \frac{\kappa_m}{||\mathbf{x}'||} \cdot \left[\frac{(g_{m-1}g_{m+1})'}{2g_{m-1}g_{m+1}} - \frac{\mathbf{x}' \cdot \mathbf{x}''}{||\mathbf{x}'||^2} - \frac{g_m'}{g_m} \right] \\
&= \frac{\kappa_m}{||\mathbf{x}'||} \cdot \left[\frac{1}{2} \left(\frac{g_{m-1}'}{g_{m-1}} + \frac{g_{m+1}'}{g_{m+1}} \right) - \frac{\mathbf{x}' \cdot \mathbf{x}''}{||\mathbf{x}'||^2} - \frac{g_m'}{g_m} \right].
\end{aligned}
$$

Using formula (4.2), we have $g_1(t) = \mathbf{x}' \cdot \mathbf{x}'$. Thus,

$$
\frac{g_1'(t)}{g_1(t)} = \frac{2\mathbf{x}' \cdot \mathbf{x}''}{||\mathbf{x}'||^2}.
$$

Hence,

$$
\dot{\kappa_m} = \frac{\kappa_m}{||\mathbf{x}'||} \cdot \left[\frac{1}{2} \left(\frac{g_{m-1}'}{g_{m-1}} + \frac{g_{m+1}'}{g_{m+1}} - \frac{g_1'}{g_1} \right) - \frac{g_m'}{g_m} \right]. \square
$$

Now we are ready to prove the main theorem of this section, which gives the structure of the connection matrices that guarantee the continuity of the curvatures and their first arc length derivatives. Following the convention adopted in Chapters 2 and 3, we use $\Gamma(G^n)$ to refer to the set of connection matrices associated with G^n–continuity, i.e. the β–matrices. The theorem asserts that the continuity of the curvatures and their first derivatives with respect to arc length is guaranteed if and only if the entries of the connection matrix M along and immediately below the diagonal match those of some matrix G in $\Gamma(G^n)$. Explicit formulas for the entries of G in $\Gamma(G^n)$ are given in [31, 32]. In particular,

$$
G_{i,i} = \beta_1^i \qquad G_{i+1,i} = \binom{i+1}{2} \beta_1^{i-1} \beta_2 \tag{4.6}
$$

where $G_{1,1} = \beta_1$ and $G_{2,1} = \beta_2$.

Schematically the structure of the resulting connection matrices is,

$$\begin{bmatrix} g & 0 & 0 & \cdots & 0 \\ g & g & 0 & \cdots & 0 \\ * & g & g & 0 & 0 \\ * & * & \ddots & \ddots & 0 \\ * & * & * & g & g \end{bmatrix}$$

where entries marked g match those of G and entries marked $*$ are free.

To facilitate the proof of our theorem, we shall need the following lemma.

Lemma 4.2 Let $\mathbf{x}(t)$ be a curve in $\mathbb{R}^n$ with connection matrix M at $t = \tau$. If $M_{i,i} = \beta_1^i$, then

$$\frac{g'_{m+}(\tau)}{g_{m+}(\tau)} = \beta_1 \frac{g'_{m-}(\tau)}{g_{m-}(\tau)} + 2\,\frac{M_{m+1,m}}{\beta_1^m} \quad m = 1, \ldots, n-1.$$

Proof By Lemma 4.1 we can write g_m using equation (4.4) as

$$g_m(t) = \sum_\alpha \det{}^2 \left[\mathbf{x}_\alpha^{(1)}, \cdots, \mathbf{x}_\alpha^{(m)}\right] \quad m \in 1, \ldots, n-1$$

where the sum is over all subsequences α of $1, \ldots n$ of length m. Since $\mathbf{x}(t)$ has connection matrix M at τ and $M_{i,i} = \beta_1^i$, it follows that

$$\begin{aligned} g_{m+}(\tau) &= \sum_\alpha \det{}^2 \left[\mathbf{x}_{\alpha+}^{(1)}, \cdots, \mathbf{x}_{\alpha+}^{(m)}\right] \\ &= \sum_\alpha \det{}^2 \left[\beta_1 \mathbf{x}_{\alpha-}^{(1)}, \cdots, \beta_1^m \mathbf{x}_{\alpha-}^{(m)}\right] \\ &= \beta_1^{m(m+1)} g_{m-}(\tau). \end{aligned}$$

Moreover, differentiating equation (4.4) for $g_m(t)$, we obtain

$$g'_m(t) = 2 \sum_\alpha \det \left[\mathbf{x}_\alpha^{(1)}, \cdots, \mathbf{x}_\alpha^{(m)}\right] \cdot \det \left[\mathbf{x}_\alpha^{(1)}, \cdots, \mathbf{x}_\alpha^{(m-1)}, \mathbf{x}_\alpha^{(m+1)}\right].$$

The remaining terms vanish because the determinants have two identical columns. Now, substituting for $\mathbf{x}_{\alpha+}^{(k)}$ and $M_{i,i}$, we get

$$\begin{aligned} g'_{m+}(\tau) &= 2\sum_{\alpha}\det\left[\mathbf{x}_{\alpha+}^{(1)},\cdots,\mathbf{x}_{\alpha+}^{(m)}\right]\cdot\det\left[\mathbf{x}_{\alpha+}^{(1)},\cdots,\mathbf{x}_{\alpha+}^{(m-1)},\mathbf{x}_{\alpha+}^{(m+1)}\right]. \qquad (4.7)\\ &= 2\sum_{\alpha}\det\left[\beta_1\mathbf{x}_{\alpha-}^{(1)},\cdots,\beta_1^m\mathbf{x}_{\alpha-}^{(m)}\right]\cdot\\ &\qquad \det\left[\beta_1\mathbf{x}_{\alpha-}^{(1)},\cdots,\beta_1^{m-1}\mathbf{x}_{\alpha-}^{(m-1)},(M_{m+1,m}\mathbf{x}_{\alpha-}^{(m)}+\beta_1^{m+1}\mathbf{x}_{\alpha-}^{(m+1)})\right]\\ &= 2\,\beta_1^{m^2}M_{m+1,m}\,g_{m-}(\tau)+\beta_1^{m^2+m+1}\,g'_{m-}(\tau). \end{aligned}$$

Dividing both sides by $g_{m+}(\tau)=\beta_1^{m(m+1)}g_{m-}(\tau)$ yields the result. □

We shall refer to a formula that expresses the value of a term Y on one side of the curve Y_+ in terms of its value on the other side, Y_-, as the *difference–expression* for Y. So Lemma 4.2 gives the difference–expression for $g'_m(\tau)/g_m(\tau)$.

Theorem 4.1 The following two statements are equivalent:

1. M is a connection matrix with
   (a) $M_{i,i}=\beta_1^i \quad i=1,\ldots,n.$
   (b) $M_{i+1,i}=\binom{i+1}{2}\beta_1^{i-1}\beta_2 \quad i=1,\ldots,n-1.$
2. For any curve $\mathbf{x}(t)\in\mathbb{R}^n$ with connection matrix M at $t=\tau$, the following geometric invariants are continuous at τ:
   (a) Curvatures: $\kappa_{i+}=\kappa_{i-} \quad i=1,\ldots,n-1$
   (b) Derivatives of curvatures: $\dot{\kappa}_{i+}=\dot{\kappa}_{i-} \quad i=1,\ldots,n-2$ where derivatives are taken with respect to arc length.

Proof $1\Rightarrow 2$: Since by assumption $M_{i,i}=\beta_1^i$, $i=1,\ldots,n$, the continuity of the curvatures is immediate from [22], so

$$\kappa_{i+}=\kappa_{i-} \quad i=1,\ldots,n-1. \qquad (4.8)$$

Now consider $\dot{\kappa_m}$, $2 \leq m \leq n-2$. By Proposition 4.1 and Lemma 4.2, we have

$$\begin{aligned}
\dot{\kappa_{m+}} &= \frac{\kappa_{m+}}{||\mathbf{x}'_+||} \cdot \left[\frac{1}{2}\left(\frac{g'_{m-1+}}{g_{m-1+}} + \frac{g'_{m+1+}}{g_{m+1+}} - \frac{g'_{1+}}{g_{1+}}\right) - \frac{g'_{m+}}{g_{m+}}\right] \\
&= \frac{\kappa_{m-}}{\beta_1||\mathbf{x}'_-||} \cdot \left[\frac{1}{2}\left(\beta_1 \frac{g'_{m-1-}}{g_{m-1-}} + 2\,\frac{M_{m,m-1}}{\beta_1^{m-1}} + \beta_1 \frac{g'_{m+1-}}{g_{m+1-}} + 2\,\frac{M_{m+2,m+1}}{\beta_1^{m+1}}\right)\right. \\
&\qquad \left. - \frac{1}{2}\left(\beta_1 \frac{g'_{1-}}{g_{1-}} + 2\,\frac{M_{2,1}}{\beta_1}\right) - \left(\beta_1 \frac{g'_{m-}}{g_{m-}} + 2\,\frac{M_{m+1,m}}{\beta_1^m}\right)\right] \\
&= \frac{\kappa_{m-}}{||\mathbf{x}'_-||} \cdot \left[\frac{1}{2}\left(\frac{g'_{m-1-}}{g_{m-1-}} + \frac{g'_{m+1-}}{g_{m+1-}} - \frac{g'_{1-}}{g_{1-}}\right) - \frac{g'_{m-}}{g_{m-}}\right] \\
&\qquad + \frac{\kappa_{m-}}{||\mathbf{x}'_-||} \cdot \left[\frac{M_{m,m-1}}{\beta_1^m} + \frac{M_{m+2,m+1}}{\beta_1^{m+2}} - \frac{M_{2,1}}{\beta_1^2} - 2\,\frac{M_{m+1,m}}{\beta_1^{m+1}}\right] \qquad (4.9)
\end{aligned}$$

Terms on the first line of the last equation match literally those in formula (4.5) for $\dot{\kappa_m}$; thus their value is $\dot{\kappa_{m-}}$. Therefore, substituting for the entries of M on the right hand side, we get

$$\begin{aligned}
\dot{\kappa_{m+}} &= \dot{\kappa_{m-}} + \frac{\kappa_{m-}}{||\mathbf{x}'_-||} \cdot \frac{\beta_2}{\beta_1^2}\left[\binom{m}{2} + \binom{m+2}{2} - 2\binom{m+1}{2} - 1\right] \\
&= \dot{\kappa_{m-}}.
\end{aligned}$$

The proof carries over to deal with $\dot{\kappa_1}$ since $M_{1,0}$ is assumed to be zero.

$2 \Rightarrow 1$: Condition 2(a) implies 1(a) by [22]. We must show that the addition of condition 2(b) implies the addition of 1(b). To insure the continuity of $\dot{\kappa_m}$, we proceed as in the reverse direction to get (9). Since the first line of (9) is $\dot{\kappa_{m-}}$, it follows by 2(b) that the quantity in brackets on the second line of (9) must vanish. We have already seen that 1(b) gives a solution. Moreover, solving for $M_{m+2,m+1}$ we get,

$$M_{m+2,m+1} = \beta_1^{m+2}\left[\frac{M_{2,1}}{\beta_1^2} + 2\,\frac{M_{m+1,m}}{\beta_1^{m+1}} - \frac{M_{m,m-1}}{\beta_1^m}\right] \quad m = 1, \ldots, n-2$$

Thus starting with $M_{2,1} = \beta_2$, we can solve inductively for $M_{m+2,m+1}$. It follows that the solution given by 1(b) is unique. □

4.3 Higher Order Derivatives of Curvatures

Here we generalize the results of the previous section to deal with arc length derivatives of the curvatures up to some arbitrary order p. The computation of the curvature κ_m in (4.1) involves derivatives of $\mathbf{x}$ up to the $m+1^{st}$ order. So, the derivatives of κ_m that invoke derivatives of $\mathbf{x}$ only up to n^{th} order are $\kappa_m^{[q]}$ where $m+q \leq n-1$. Our objective is to find the structure of the connection matrices that guarantee the continuity of all the invariants $\kappa_m^{[q]}$ for which $q = 0, \ldots, p$ and $m+q \leq n-1$.

The main idea we are going to use is that G^n continuity preserves the continuity of all the curvatures κ_m and all their arc length derivatives up to the $n-m-1^{st}$ order. This result follows because G^n–continuity implies the continuity of the arc length parameterization of $\mathbf{x}$ up to order n and by (4.1), the invariants $\kappa_m^{[q]}$, $q+m \leq n-1$, can be expressed in terms of the first n derivatives of $\mathbf{x}$ with respect to arc length.

Now to find the connection matrices that guarantee the continuity of the derivatives of the curvatures up to some fixed order p, we need to express $\kappa_{m+}^{[q]}$, $q = 0, \ldots, p$, $m+q \leq n-1$ in terms of $\kappa_{m-}^{[j]}$, $j \leq q$, and entries of the connection matrix M. That is, we need to write the difference–expressions for the invariants $\kappa_m^{[q]}$. Only certain entries of M appear in these expressions. By setting these entries to match those of some matrix $G \in \Gamma(G^n)$, we are assured of the continuity of all the invariants $\kappa_m^{[q]}$.

For example, in the last section we were able to ensure continuity of the first arc length derivative of the curvatures by matching the entries of M that appeared in the difference–expression for $\dot{\kappa}_m$, (9), to those of some $G \in \Gamma(G^n)$. For the general case, the problem reduces to finding the entries of M used in the difference–expression for $\kappa_m^{[q]}$.

We begin by going back to inspect formula (4.5) for $\dot{\kappa}_m$. That formula contains only κ_m, $||\mathbf{x}'||$ and g_j'/g_j for some js in $1, \ldots, n-1$. We claim that higher order derivatives of κ_m can also be expressed in terms of κ_m, expressions involving quotients of g_j and derivatives of g_j, and $||\mathbf{x}'||$ as the lemmas below show.

Lemma 4.3 We can express $\left(\frac{g'_m}{g_m}\right)^{(j)}$ as,

$$\left(\frac{g'_m}{g_m}\right)^{(j)} = \frac{g_m^{(j+1)}}{g_m} + \text{polynomial in } \frac{g_m^{(k)}}{g_m}\ k \in 1, \ldots, j.$$

Proof The proof follows by induction on the number of derivatives j, employing

$$(\frac{g_m^{(k)}}{g_m})' = \frac{g_m^{(k+1)}}{g_m} - \frac{g_m^{(k)}}{g_m}\frac{g'_m}{g_m}.\square$$

Lemma 4.4 We can express $\kappa_m^{[q]}$ as a polynomial in κ_m, $1/||\mathbf{x}'||$ and $(g_h^{(r)}/g_h)$ for $r = 1, \ldots, q$ and $h = 1, m-1, m, m+1$. Moreover, this polynomial is linear in $g_{m+1}^{(q)}/g_{m+1}$.

Proof First, we make the observation that

$$\left(\frac{1}{||\mathbf{x}'||}\right)' = \frac{-\mathbf{x}' \cdot \mathbf{x}''}{||\mathbf{x}'||^3} = \frac{-1}{2}\frac{g'_1}{g_1}\frac{1}{||\mathbf{x}'||}.$$

Now differentiating formula (4.5) $q-1$ times with respect to arc length, we get

$$\begin{aligned} \kappa_m^{[q]} = & \quad \text{Polynomial in} \left(\tfrac{1}{||\mathbf{x}'||}\right), \kappa_m^{[j]} \text{ and } \left(\tfrac{g'_h}{g_h}\right)^{(j)}, \\ & \text{where } j = 0, \ldots, q-1 \quad h = 1, m-1, m, m+1. \end{aligned} \tag{4.10}$$

Recursively substituting for $\kappa_m^{[j]}$ $j = q-1, \ldots, 1$ using (10) gives the first result. Since the polynomial in (10) is linear in $(g'_{m+1}/g_{m+1})^{(q-1)}$, the second result follows by Lemma 4.3. $\square$

It follows from Lemma 4.4 that to find the entries of the connection matrix M that appear in the difference–expression for $\kappa_m^{[q]}$, we need to analyze the difference–expressions for κ_m, $1/||\mathbf{x}'||$ and $(g_h^{(r)}/g_h)$ for $r = 1, \ldots, q$ and $h = 1, m-1, m, m+1$.

Inspecting Lemma 4.2, we see that the entries of M that appear in the difference–expression for $g'_m(\tau)/g_m(\tau)$ are the entries $M_{i,i}$ $i = 1, \ldots, m+1$, and $M_{m+1,m}$. These entries are of distance no greater than 1 from the diagonal, and their row indices are no bigger than $m+1$. In general, we shall show that the entries of M that appear in the difference–expression for $g_m^{(q)}/g_m$ lie only in the first $m+q$ rows of M along the first $q+1$ main diagonals.

Proposition 4.2 Let $\mathbf{x}(t)$ be a curve in $\mathbb{R}^n$ with connection matrix, M, at $t = \tau$. Then the difference–expression for $\left(g_m^{(q)}/g_m\right)$ involves only the entries M_{ij} where $0 \le i - j \le q$ and $i \le m + q$. Moreover, this difference–expression is linear in $M_{m+q,m}$.

Proof Let α be a subsequence of $1, \ldots, n$ of length m and let

$$Y_\alpha(t) = \det\left[\mathbf{x}_\alpha^{(1)}, \cdots, \mathbf{x}_\alpha^{(m)}\right].$$

By Lemma 4.1, $g_m(t) = \sum_\alpha Y_\alpha^2(t)$, where the sum is over all subsequences of $1, \ldots, n$ of length m. Therefore

$$g_m^{(q)} = \sum_\alpha \sum_{r=0}^{q} \binom{q}{r} Y_\alpha^{(r)} Y_\alpha^{(q-r)}. \tag{4.11}$$

Hence, $g_m^{(q)}$ $q \ge 1$ is linear in $Y_\alpha^{(q)}$. We now find the entries of M that appear in the difference–expression for $Y_\alpha^{(q)}$. We can express $Y_\alpha^{(q)}$ as a sum of determinants; pick one of these determinants, y. Let $l(y)$ denote the smallest derivative of $\mathbf{x}_\alpha$ that does not appear in y, and let $u(y)$ denote the greatest derivative of $\mathbf{x}_\alpha$ that does appear in y. We claim that $u(y) - l(y) \le q$ and $u(y) \le m + q$. This can easily be seen by induction on the number of derivatives, q, since $l(Y'_\alpha) = m$ and $u(Y'_\alpha) = m+1$ and differentiating $Y_\alpha^{(q)}$ at most either increases $u(y)$ by 1 or decreases $l(y)$ by 1. Henceforth, we shall write l and u for $l(y)$ and $u(y)$ respectively, since y is fixed. Now we can write y_+ as,

$$\begin{aligned} y_+ &= \det[\mathbf{x}_{\alpha+}^{(1)}, \ldots, \mathbf{x}_{\alpha+}^{(l-1)}, \mathbf{x}_{\alpha+}^{(r)}, \ldots, \mathbf{x}_{\alpha+}^{(u)}] \qquad l+1 \le r \le u \\ &= \det\Big[M_{1,1}\mathbf{x}_{\alpha-}^{(1)}, \ldots, M_{l-1,l-1}\mathbf{x}_{\alpha-}^{(l-1)}, (M_{r,l}\mathbf{x}_{\alpha-}^{(l)} + \cdots + M_{r,r}\mathbf{x}_{\alpha-}^{(r)}), \\ &\qquad \ldots, (M_{u,l}\mathbf{x}_{\alpha-}^{(l)} + \cdots + M_{u,u}\mathbf{x}_{\alpha-}^{(u)})\Big] \end{aligned}$$

because M is lower triangular. Since $u - l \le q$ and $u \le m + q$, we conclude that the difference–expression for $Y_\alpha^{(q)}$ invokes only $M_{i,j}$ $0 \le i - j \le q$ and $i \le m + q$. Performing the same analysis on $Y_\alpha^{(k)}$ $k = 1, \ldots, q-1$, we can easily see that no new entries of M appear in their difference–expressions. Hence the difference–expression for $g_{m,\alpha}^{(q)}(\tau)$ involves only the entries $M_{i,j}$ where $0 \le i - j \le q$ and $i \le m + q$. Since $g_m(t) = \sum_\alpha Y_\alpha^2(t)$ and

$$Y_{\alpha+} = \det\left[\mathbf{x}_{\alpha+}^{(1)}, \ldots, \mathbf{x}_{\alpha+}^{(m)}\right] = \prod_{i=1}^{m} M_{i,i} \det\left[\mathbf{x}_{\alpha-}^{(1)}, \ldots, \mathbf{x}_{\alpha-}^{(m)}\right],$$

it follows that the difference–expression for $g_m^{(q)}/g_m$ also uses only the entries of M specified in the proposition.

Finally notice that y_+ is linear in $M_{u,l}$. Since $Y_\alpha^{(q)}$ is a sum of linear functions in $M_{i,j}$, $i - j = q$, and $g_m^{(q)}$ is linear in $Y_\alpha^{(q)}$, it follows that the difference–expression for $g_m^{(q)}/g_m$ is linear in $M_{m+q,m}$.$\square$

We conclude this section with our main theorem which generalizes the main result of the previous section. The theorem tells us the structure of the connection matrices that correspond to a notion of geometric continuity where the curvatures κ_m and their arc length derivatives up to some order p are continuous. Theorem 4.2 extends Theorem 4.1 in a nice way. To guarantee the continuity of the first p arc length derivatives of the curvatures, we need to set all entries of the connection matrix of distance less than or equal to p below the diagonal to be identical to those of some matrix G in $\Gamma(G^n)$. Schematically the connection matrices must have the following structure:

$$\begin{array}{c} p+1 \left\{ \begin{array}{c} \\ \\ \\ \end{array} \right. \\ \\ \\ \\ \\ \end{array} \begin{bmatrix} g & & & & & \\ \cdot & g & & & & \\ g & \cdot & g & & & \\ * & g & \cdot & g & & \\ \vdots & * & \ddots & \ddots & \ddots & \\ * & \cdots & * & g & \cdot & g \end{bmatrix}$$

where entries marked g match those of G and entries marked $*$ are free.

Theorem 4.2 The following two statements are equivalent:

1. M is a connection matrix such that: $M_{i,j} = G_{i,j} \quad 0 \leq i - j \leq p$ for some $G \in \Gamma(G^n)$,
2. For any curve $\mathbf{x}(t)$ in $\mathbb{R}^n$ with connection matrix, M, at $t = \tau$, the following geometric invariants are continuous at τ: $\kappa_{m+}^{[q]} = \kappa_{m-}^{[q]}, \quad q = 0, \ldots, p$ and $m + q \leq n - 1$.

Proof $1 \Rightarrow 2$: By Lemma 4.4, $\kappa_m^{[q]}$ depends only on $||\mathbf{x}'||$, κ_m and the quotients $(g_h^{(r)}/g_h)$ for $r = 1, \ldots, q$ and $h = 1, m-1, m, m+1$. Now $||\mathbf{x}'_+|| = M_{1,1}||\mathbf{x}'_-||$, and we have κ_m continuity by [22], so it remains to study the quotients $(g_h^{(r)}/g_h)$. By Proposition 4.2 the difference–expressions for these quotients depend only on $M_{i,j}$ $i - j \leq q$. But, by assumption, $M_{i,j} = G_{i,j}$ $i - j \leq q$ for some $G \in \Gamma(G^n)$. Moreover, we know that for the connection matrix G, condition (2) must hold since G^n–continuity implies the continuity of all the arc length derivatives of the higher order curvatures. Hence condition (2) must hold for M as well, since only the entries of M that match G appear in the difference–expression for $\kappa_m^{[q]}$.

$2 \Rightarrow 1$: The proof proceeds by induction on p the number of continuous derivatives of the curvatures.

Base Case: ($p = 0$) reduces to the continuity of the curvatures which implies part 1 by [22].

Induction step: Assume that $\kappa^{[r]}_{m+} = \kappa^{[r]}_{m-}$ $r = 0, \ldots, p-1$ only if $M_{i,j} = G_{i,j}$ $0 \leq i - j \leq p - 1$ for some $G \in \Gamma(G^n)$. Now, we want to show that $\kappa^{[p]}_m$ is continuous only if the entries M_{ij} $i - j = p$ agree with those of the same $G \in \Gamma(G^n)$. We use another induction on the row indices of the p^{th} diagonal of M. Consider the entry $M_{p+m+1,m+1}$. The case $m = 0$ is immediate since $G_{p+1,1}$ is free. For $m > 0$, by the second induction hypothesis we have $M_{p+r,r} = G_{p+r,r}$ $r = 1, \ldots, m$. Now, by Lemma 4.4 $\kappa^{[p]}_m$ is a polynomial in $g^{(r)}_h / g_h$ and this polynomial is linear in $g^{(p)}_{m+1}/g_{m+1}$. By Proposition 4.2 the difference–expression for $\kappa^{[p]}_m$ is linear in $M_{p+m+1,m+1}$ and all the other entries of M that appear in this difference–expression are already known by the induction hypotheses. Thus we can solve the equation $\kappa^{[p]}_{m+} = \kappa^{[p]}_{m-}$ for $M_{p+m+1,m+1}$. Since we already have a solution, $G_{p+m+1,m+1}$, this solution must be unique. Hence $M_{p+m+1,m+1} = G_{p+m+1,m+1}$, and the result follows by induction. □

4.4 Summary

In the previous sections of this chapter we have found the structure of the connection matrices that correspond to a definition of geometric continuity that preserves the continuity of the high order curvatures and their arc length derivatives. These new notions of continuity provide a natural progression from F^n to G^n. Although the properties of G^n and F^n continuity were used in the proofs, both arise as special cases of Theorem 4.2 by setting p to 0 and $n-1$ respectively.

Both G^n and F^n continuity are projectively invariant [31]. Projective invariance of all the new notions of continuity we have considered here can easily be verified using Theorem 3.1. There exist still other projectively invariant forms of geometric continuity. For example, we could define a notion of geometric continuity in $\mathbb{R}^4$ by preserving the continuity of the curvatures $\kappa_1, \kappa_2, \kappa_3$ and the arc length derivative of the torsion $\dot{\kappa}_2$ but not the continuity of the arc length derivative of the curvature $\dot{\kappa}_1$. This notion of geometric continuity is represented by the set of connection matrices of the form:

$$\begin{bmatrix} \beta_1 & 0 & 0 & 0 \\ \beta_2 & \beta_1^2 & 0 & 0 \\ \beta_3 & \alpha & \beta_1^3 & 0 \\ \beta_4 & \beta_5 & 2\,\beta_1\alpha & \beta_1^4 \end{bmatrix}$$

where $\beta_1 > 0$ and β_i, α are free $i = 2, \ldots, 5$. This definition of geometric continuity also turns out to be projectively invariant by Theorem 3.1.

Chapter 5

Implementation, Contributions and Future Work

This chapter is divided into three sections. In the first section, we discuss the implementation of an algorithm to evaluate spline curves with continuity described by connection matrices at the knots. In this section we also exhibit examples of spline curves with various, novel, types of geometric continuity at their knots. Section 5.2 summarizes the contributions made in the first part of this thesis both to the theory of geometric continuity and to the theory of splines. In the last section, several possible extensions to this work are proposed, and a list of open questions that deserve more investigation is provided.

5.1 Implementation and Examples

Knot insertion is the main tool used for evaluation and rendering of B–spline curves [1, 14, 53]. Various B–spline evaluation algorithms rely on the fact that a B–spline curve of polynomial degree n interpolates a knot whose multiplicity is n. Thus, the evaluation of a degree n B–spline at a parameter value $\hat{t}$ reduces to inserting a knot at $\hat{t}$ repeatedly until $\hat{t}$ has knot multiplicity n. The same observation applies to polynomial splines with continuity described by connection matrices at their knots [1, 53]. Thus, the fundamental algorithm in dealing with these splines is a knot insertion algorithm.

Polynomial splines with continuity described by connection matrices have been discussed in the literature [1, 5, 20, 22]. Both the β–splines and the Frenet frame continuous splines can be described by connection matrix continuity at the knots. These splines are generally referred to as geometrically continuous splines because they have different forms of geometric continuity at their knots as specified by the associated connection matrices.

Barry, Goldman and Micchelli [1] give an algorithm for knot insertion on geometrically continuous splines. The algorithm, as stated, requires the connection matrices to be lower triangular, and to have the first column $= \{1, 0, \ldots, 0\}^T$. Since we have shown in Section 2.3 that connection matrices corresponding to any sensible notion of geometric continuity must satisfy both of these conditions, these two constraints are satisfied by construction in all our examples.

Barry *et al.* [1] make the assumption that the connection matrices used by their knot insertion algorithm are also totally positive. This condition was first imposed by Dyn and Micchelli [20, 22] as a sufficient (though not necessary) condition to guarantee the existence of minimal support basis functions (B–splines).

The formulation proposed in Chapter 2 does not account for this algebraic condition of total positivity on connection matrices associated with notions of geometric continuity. The reason this condition is not accounted for in our formulation is that total positivity is an entirely algebraic condition, and we were not able to find a corresponding intuitive geometric condition. Another problem with total positivity is that the inverse of a totally positive matrix is not necessarily totally positive, thus if we are to include total positivity in our formulation the group property will be violated. It remains an interesting open question if there exists some geometric condition that two parametric curves must satisfy at their common parameter value that corresponds to the total positivity of their connection matrix at this value.

Although Barry *et al.* assume total positivity of connection matrices to prove the existence of basis functions, the algorithms provided in [1] can still be used for arbitrary connection matrices, except that singularities may occur for specific selections of shape parameters. Our implementation of their algorithms is written in Mathematica code, and it allows for specifying arbitrary connection matrices at the knots. In the next subsection we provide some examples.

5.1.1 Examples

Frenet frame continuity of order n, as well as the definition of the high order curvatures in (4.1) assumes the existence of n linearly independent, parametric derivative vectors. These vectors are then used to build the Frenet frame and later to compute the high order curvatures. When the control polygon of the spline curve lies in a lower dimensional space, some of the derivative vectors become linearly dependent. Thus, many higher order curvatures and their (arc length) derivatives vanish. For example, in $\mathbb{R}^2$ the torsion is identically zero, as well as all its arc length derivatives, so we can not hope to visualize the difference between various notions of geometric continuity, even though each one produces different curves.

Quartic Frenet frame continuous splines

These are piecewise quartic splines with connection matrices of the form:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \beta_1 & 0 & 0 \\ 0 & \beta_2 & \beta_1^2 & 0 \\ 0 & \beta_3 & \beta_4 & \beta_1^3 \end{bmatrix},$$

at the knots. By changing the values of β_i^s we get different spline curves with different properties. The sequence of figures below shows the effect of changing the shape parameters (β_i) on a piecewise quartic spline curve.

Barsky and Beatty [2] investigated the geometric meaning of the shape parameters (βs) in the case of cubic Beta–splines, with uniform knots and fixed connection matrix throughout the curve. Recall that for cubic beta–splines connection matrices are of

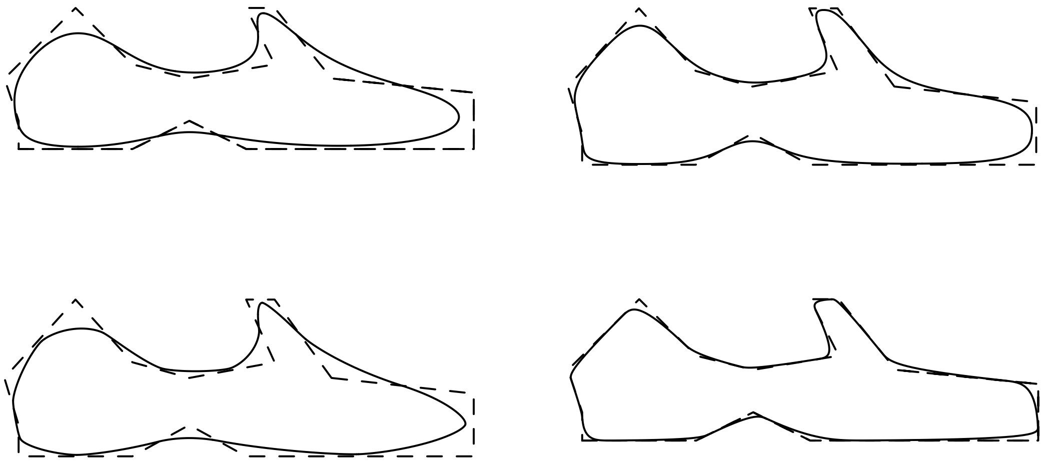


Figure 5.1: Changing the shape parameters of a quartic Frenet frame continuous spline curve. Values of shape parameters $(\beta_1, \beta_2, \beta_3, \beta_4)$ **from top left in clockwise order** $(1, 0, 0, 0), (1, 0, 0, 30), (10, 0, 0, 0), (1, 10, 0, 0)$.

the form,

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \beta_1 & 0 \\ 0 & \beta_2 & \beta_1^2 \end{bmatrix}.$$

Barsky and Beatty show that, for cubic splines with uniform knots, the parameter β_1 biases the spline curve to one side, while β_2 controls the tension of the spline curve.

For higher order polynomial splines, however, it is not known exactly how the entries of connection matrices affect the shape of the final spline curve. As Figure 5.1 demonstrates, the effects of changing β_1 and β_2 do not correspond to bias and tension in quartic splines. Also new shape parameters are introduced as the degree grows. More investigation is required to understand the effect of changing the shape parameters on the geometry of the final curve for arbitrary degrees.

Tensor–product bi–quartic Frenet frame continuous splines can be constructed on bidirectional parametric grids by applying the knot insertion algorithm of geometrically continuous splines in each direction independently. Figure 5.2 shows examples of tensor product bi–quartic, Frenet–frame continuous splines, and the effect of changing the shape parameters on them.

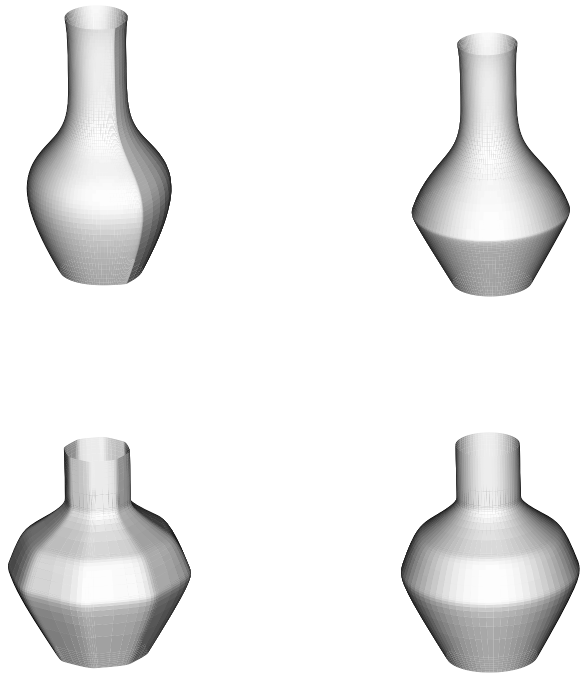


Figure 5.2: Changing the shape parameters in a tensor product quartic spline.

5.2 Summary of Contributions

In this part of the thesis we studied building spline curves utilizing geometric continuity between curve segments. In what follows we list the main new results we obtained.

- Axiomatization: The main contribution to the theory of geometric continuity is the axiomatization of continuity by a set of five simple geometric properties that must be satisfied by any sensible definition of a notion of geometric continuity.
- Algebraic Formulation: Another major contribution is to find the set of algebraic conditions on the structure of connection matrices corresponding to the five geometric properties of the axiomatization. These conditions make it easy, as well as a completely algebraic procedure, to verify if a proposed notion of geometric continuity is sensible.
- A Notion of Cardinality: Imposing a partial order on notions of geometric continuity representable by connection matrices, and using this order to answer geometric questions regarding the cardinality of certain notions geometric continuity through an entirely algebraic approach. In particular, using our formulation we show using the group structure that connection matrices must satisfy, that there can not be a sensible definition of geometric continuity strictly in between G^3 and F^3.
- Geometric Invariants: The development of many new notions of geometric continuity with various degrees of flexibility in curve design. One particular interesting family of these new notions bridges the gap between reparameterization continuity and Frenet frame continuity, two previously isolated theories.

5.3 Open Questions and Future Research

The following is a list of open questions that arose from the research conducted in this part of the thesis. Answering these questions would add significantly both to the theory of geometric continuity and to the theory of polynomial splines. It would also add substantially to our understanding of the free parameters involved in designing with geometrically continuous splines.

- Identify the geometric invariants that a notion of geometric continuity preserves given its associated set (group) of connection matrices. In particular solve this problem for those notions of continuity associated with the sets of connection matrices derived using the S–construction of Chapter 3.

- Find the most general form of geometric continuity and the associated group of connection matrices that preserves the continuity of a given collection of geometric invariants. This problem is the converse of the previous problem.

- Find the set of all transformations that preserve a given notion of geometric continuity. That is, for a notion of continuity H^n, characterize the set of transformations T such that $(f,g) \in H^n \Rightarrow (T \circ f, T \circ g) \in H^n$.

- Characterize all possible notions of geometric continuity that can be defined using the axiomatization given in Chapter 2. That is, categorize all possible groups of matrices that meet the specified algebraic properties of Section 2.3.

- Are there other axioms that can or should be added to our axiomatization of geometric continuity?. If so, what are the associated algebraic conditions on the corresponding sets of connection matrices.

  There is a theory for splines built with continuity described by connection matrices across the knots. However, developers of this theory generally assume totally positive connection matrices to push the proofs through and insure correct space dimensions [1, 20, 22, 53]. It is not well studied what happens if this total positivity condition is dropped. On the other hand, there is no clear geometric reason to insist on totally positive connection matrices. We would like to investigate this question further to study if such geometric property exists to add it to our axiomatization in Chapter 2.

- Study the properties of spline curves and surfaces that can be built using the various new notions of continuity across the knots. There are already some special cases that were considered in literature; most notably the β–splines and γ–splines.

- There are available algorithms [1, 53] for evaluation, blossoming and inserting knots on splines with connection matrices at the knots. However, these algorithms are highly complicated mainly because the connection matrices are involved in these computations. It is desirable to reduce the complexity of these algorithms to make these splines appealing for wider use in CAGD. Developing subdivision formulas for such splines is one possible approach that needs further investigation.

- Extend the general theory of geometric continuity to parametric surfaces. There exist formulations of geometric continuity for pairs of parametric surfaces meeting at a common parameter value in terms of connection matrices [43]. The main difference between curves and surfaces is that the connection matrices for surfaces are block structured.

  We can push the axiomatization and some subsequent results to parametric surfaces meeting at a point but the problem becomes more challenging when we start considering patches meeting across a curve. More research is needed to investigate how far can this extension be pursued. In particular, we would like to investigate :

  The cardinality of notions of geometric continuity of various orders that can be defined and satisfy our axioms.

The existence of notions of geometric continuity analog to those built using the S–construction of Chapter 3.

In Chapter 4 notions of continuity for pairs of parametric curves, based solely on the geometric invariants that they preserve, were presented. The definitions of these notions relied on the existence of high order curvatures, and an intrinsic parameterization for these curves (arc length parameterization). We would like to study the existence of corresponding notions of geometric continuity for parametric surfaces that can be defined in terms of the geometric invariants of parametric surfaces: Gaussian curvature, mean curvature, high order tensors and fundamental forms.

Part II

Building Surfaces Interpolating Hermite Data Using Dynamic Programming

Chapter 6

A Recursive Algorithm for Hermite Interpolation over a Triangular Grid

6.1 Introduction and motivation

Interpolation is one of the fundamental techniques of approximation theory that have been heavily used in CAGD to build curves and surfaces. Scientists, engineers and mathematicians model many physical problems as interpolation problems. One of the main issues in interpolation is the selection of the solution space. Different types of basis functions have been used, but polynomials are by far the most heavily studied basis for interpolation. For univariate polynomial interpolation, the answers to most questions can be found in any standard book on numerical analysis. Although the theory of univariate polynomial interpolation is almost complete, the multivariate analogue is not. Difficulties in the multivariate theory arise because, in general, there is no unique multivariate polynomial space that makes multivariate interpolation unique. When derivatives also need to be interpolated, the problem becomes even harder. In this part of the thesis we concentrate on bivariate Hermite interpolation.

For some special arrangements of interpolation positions in the plane there are techniques to solve Lagrange interpolation problems [11, 39, 46, 48]. One of these arrangements is the triangular grid which we consider in this part of the thesis. Hermite data at the nodes of a triangular grid can arise from sampling some non-polynomial function and its derivatives at these nodes. In this case the objective is to find a polynomial that approximates this non-polynomial function while matching the values and the derivatives at the nodes of the grid. The problem of finding a polynomial surface that fits some scientific data measurements on a triangular grid can also be formulated as a Hermite interpolation problem over the grid.

Interpolation using polynomials also has its problems. High degree polynomials tend to oscillate wildly away from the interpolation positions. This instability is the main reason why polynomials are usually used only for local approximation. But the global problem never ceases to be of interest as evidenced by some recent work on the problem by de Boor [16], Gasca [26, 27] , Martinez [28], Lorentz [42]. Other investigators approach this problem differently using splines [52], finite element methods [12, 44, 45, 55] or radial basis functions [47].

Neville's algorithm solves the one–dimensional Hermite interpolation problem recursively using dynamic programming. Lee and Phillips [39] introduced an extension to Neville's algorithm that interpolates Lagrange data over a triangular grid of points laid out as a geometric mesh, also referred to as principal lattice arrangement by Chung and Yao [11]. Our objective is to generalize this construction to interpolate derivative information at the grid points.

6.1.1 Problem definition

To specify a bivariate interpolation problem, we need to address the arrangement of the interpolation positions (nodes) in the plane. Interpolation problems have different degrees of difficulty depending on the locations of the nodes. Several special configurations have been studied in literature. The arrangement of the nodes we consider is usually referred to as a triangular grid. Definition 6.1 formally characterizes this configuration.

> **Definition 6.1 Triangular Grid:** A triangular grid of size n consists of $(n+1)(n+2)/2$ nodes. These nodes P_{ijk}, $i+j+k=n$, are arranged to lie at the intersection points of three sets of lines, R_i, S_i, T_i, $i=0,\dots,n$,

so that

$$\left.\begin{array}{rcll} P_{ijk} & = & R_i \cap S_j \cap T_k \quad i+j+k=n & \text{(dependence conditions)} \\ \phi & = & R_i \cap S_j \cap T_k \quad i+j+k<n & \text{(independence conditions)} \end{array}\right\} \quad (6.1)$$

Figure 6.1 shows an instance of a triangular grid of size 2. Usually the lines in each set are parallel and equidistant, but this need not be the case as long as the lines satisfy the (in)dependence conditions.

Our objective will be to compute a polynomial that interpolates function values, partial derivatives, and mixed partials up to order r_{ijk} at the nodes P_{ijk} of a triangular grid of size n. Formally we define our problem as:

> **Problem 6.1** Let P_{ijk}, $i+j+k=n$, be a triangular grid of size n in the s–t plane, and let $f(s,t)$ be a real–valued function at least r_{ijk} times differentiable at P_{ijk}. Find a polynomial $I(s,t)$ such that at each node P_{ijk}
>
> $$\frac{\partial^p I}{\partial s^q \partial t^{p-q}}(P_{ijk}) = \frac{\partial^p f}{\partial s^q \partial t^{p-q}}(P_{ijk}) \quad \text{for } q=0\ldots p,\ p=0\ldots r_{ijk}.$$

These numbers, r_{ijk} in Problem 6.1, denote the order of the highest derivatives that we want to interpolate at the points P_{ijk}. If all these numbers r_{ijk} are equal to some fixed nonnegative integer r, we call the problem *uniform* of order r. Thus Lagrange interpolation over a triangular grid is a uniform problem of order 0. If $r_{ijk} < 0$, then there are no interpolation conditions at the corresponding point P_{ijk}. Since our solution to Problem 6.1 depends critically on the structure of the triangular grid, we shall generally insist that $r_{ijk} \geq 0$ for all i, j, k; otherwise by choosing $r_{ijk} < 0$ at arbitrary points along the grid our problem could collapse to a scattered data problem and the grid would be of no help at all. The only exception we allow is when $r_{ijk} < 0$ for all the nodes along one of the boundaries of the grid. In this case we can simply remove this boundary line and replace the grid by one of smaller size.

The derivatives we are interpolating are directional derivatives along the two directions parallel to the s, t axes of the parameter domain. Any directional derivative of order less than or equal to r_{ijk} can be expressed in terms of these partial derivatives in the s, t directions.

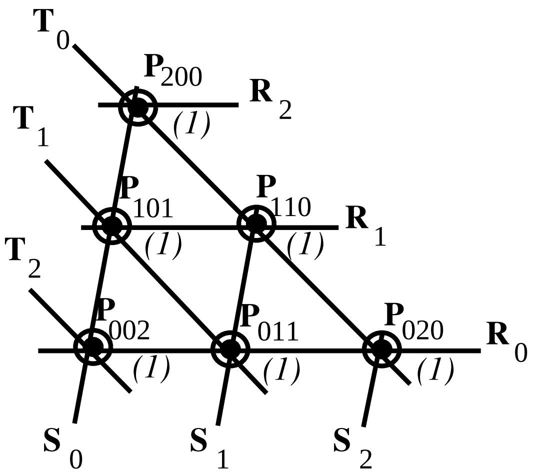


Figure 6.1: The parameter domain for a uniform problem with $n = 2$ and $r = 1$. Instead of using concentric circles to denote the order of the highest derivatives at a node, we shall write this number, in parenthesis, next to the corresponding node.

Figure 6.1 shows an instance of the parameter domain for a problem of size $n = 2$, and explains the notation that we are going to use throughout this part of the thesis.

Although the statement of Problem 6.1 assumes that the data comes from a real–valued function f, the extension to vector–valued functions is easy. We just treat each coordinate function independently. Thus, if we can solve the interpolation problem for real–valued functions, we can solve it for vector–valued functions.

The statement of Problem 6.1 assumes that the data we are interpolating comes from some smooth function $f(s, t)$. However we know nothing about this function, except for its differentiability at the nodes, and the function and derivative values there. In other words, we could just as well assume the data comes from scientific measurements at discrete points. The use of f here is just for notational convenience.

6.1.2 A recursive approach

For Lagrange interpolation over a triangular grid of nodes, there is a Neville–like recursive algorithm with a pyramidal structure due to Lee and Phillips [39], when the lines are parallel and equidistant. A generalization of this interpolation algorithm to more general setting is given in [40]. We would like our recursive scheme to reduce to this extended Neville–like algorithm, if no derivative information is available. Thus, we are going to develop an interpolation algorithm with a similar pyramidal structure.

The approach we are going to take to solve our Hermite interpolation problem is to break it into three smaller problems each with fewer interpolation conditions. Repeating this process over and over, throwing away boundaries with no interpolation conditions, we get smaller and smaller triangular grids until we get down to single triangles.

The problem of interpolation of point and derivative data over a single triangle will be addressed in Section 6.2. Our approach to Hermite interpolation over a single triangle will have the same recursive flavor. An interpolant of the data over one triangle is computed recursively from the solutions of simpler problems with fewer interpolation conditions at the vertices. The base cases for this recursion involve Hermite interpolation at a single point and Lagrange interpolation at the vertices of a triangle.

After building partial interpolants over individual triangles, we need to combine these into interpolants over bigger and bigger triangles and eventually into interpolants over all the data. This process is referred to as blending. To do this blending, we need to find a recurrence that describes how to blend partial interpolants into interpolants over bigger sets of data. We derive this recurrence in Section 6.3.

One of the major advantages of solving interpolation problems recursively is that we can reuse parts of the computations for one subinterpolant in computing another subinterpolant. Dynamic programming exploits these reusable computations. The interpolation algorithm we are going to present in Section 6.4 is a dynamic programming algorithm. In Section 6.5 we work an example to illustrate how the steps of the algorithm are implemented.

Two recent Lagrange interpolation algorithms lend themselves to the interpolation of Hermite data by coalescing points [16, 27]. In Section 6.6 we discuss the efficiency of our interpolation algorithm and study how it compares to these two algorithms.

6.2 Interpolation over individual triangles

We are going to solve our Hermite interpolation problem (Problem 6.1) recursively by solving smaller subproblems with fewer interpolation conditions. As the number of interpolation conditions decreases, we get smaller and smaller grids. Eventually we reduce the grid to one triangle (see Section 6.3). In this section we address the problem of Hermite interpolation at the vertices of an individual triangle. The configuration and indexing are shown in Figure 6.2.

6.2.1 Problem definition

The problem in this section is an instance of Problem 6.1 for the case $n = 1$. Formally,

Problem 6.2

Consider a triangle in the s–t plane, with vertices $P_{100}, P_{010}, P_{001}$ and let $f(s,t)$ be a real–valued function at least r_{ijk}-times differentiable at P_{ijk}.

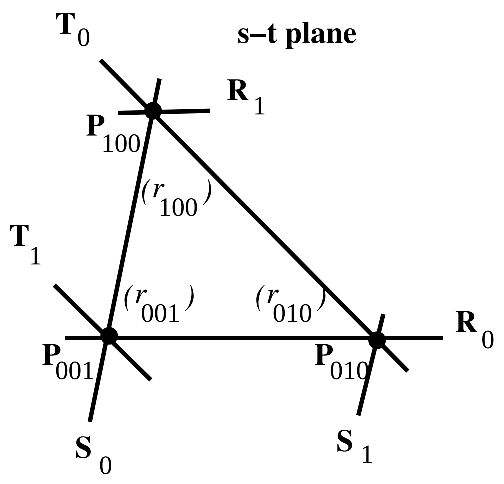


Figure 6.2: Indexing the nodes for interpolation over one triangle.

Find a polynomial $I(s,t)$ such that at each vertex P_{ijk}:

$$\frac{\partial^p I}{\partial s^q \partial t^{p-q}}(P_{ijk}) = \frac{\partial^p f}{\partial s^q \partial t^{p-q}}(P_{ijk}) \quad \text{for } q = 0 \ldots p,\ p = 0 \ldots r_{ijk}. \quad (6.2)$$

We shall assume that all the r_{ijk} are nonnegative. This condition corresponds to having interpolation data at all three vertices of the triangle. This problem will be reduced to the following base cases: Hermite interpolation at one vertex or Lagrange interpolation at the three vertices of the triangle. Taylor's expansion provides the solution for the interpolation problem at one vertex. The recurrences we derive avoid the situation where interpolation data is available only at two vertices of the triangle because two points do not form a triangular grid. A solution to the latter problem can still be constructed however, using a variant of the univariate Neville algorithm along the line joining the two points, but this interpolant has some undesirable properties (See Section 6.6). In Section 6.2 we address only the non-degenerate version of Problem 6.2.

Le Méhauté [44] gives a formula for Hermite interpolation at the vertices of a simplex, but the interpolant he computes is different from the interpolant we build here over one triangle. He also remarks that the interpolant is not unique since this interpolation problem is generally not unisolvant.

6.2.2 Notation

We shall adopt the notation $I_{u,v,w}$ to denote a polynomial in s, t that interpolates Lagrange data and partial derivatives up to order u at P_{100}, up to order v at P_{010}, and up to order w at P_{001}. Thus, the solution to our interpolation problem (Problem 6.2) is denoted by $I_{r_{100},r_{010},r_{001}}$. To be consistent we should use $r_{ijk} = -1$ when no interpolation conditions are specified at the node P_{ijk}. However for clarity of presentation, we shall use "*" rather than "-1" from here on. In particular, $I_{0,*,*}$ denotes the constant which interpolates Lagrange data at P_{100}, but interpolates no data at either P_{010} or P_{001}. In particular, $I_{0,-1,-1}$ denotes the constant which interpolates Lagrange data at P_{100}, but interpolates no data at either P_{010} or P_{001}, and $I_{u,1,0}$ denotes an interpolant of Hermite data up to order u at P_{100}, that interpolates function value and first derivatives at P_{010} and interpolates only function value at P_{001}.

We shall refer to the subscript vector of the interpolant I as the index vector. It will be helpful to define a measure of the size of an index vector. We define the partial ordering relation $<_v$ between pairs of index vectors $V_1 = (u_1, v_1, w_1)$ and $V_2 = (u_2, v_2, w_2)$ by

$$V_1 <_v V_2 \iff u_1 \leq u_2, v_1 \leq v_2,\ w_1 \leq w_2 \quad \text{and} \quad u_1 + v_1 + w_1 < u_2 + v_2 + w_2 \tag{6.3}$$

Since the solution we are going to construct invokes barycentric coordinates, we begin by recalling some of the properties of these barycentric coordinates. Given three non collinear points P_1, P_2, P_3 in the s–t plane, any point Q in this plane can be expressed as a unique affine combination of these three points. The coefficients used to express Q are called the barycentric coordinates of Q with respect to $\triangle P_1P_2P_3$. If we denote these coefficients by $\beta_1, \beta_2, \beta_3$, then the following properties are known to hold:

$$\beta_1 + \beta_2 + \beta_3 \equiv 1 \tag{6.4}$$

$$\beta_1, \beta_2, \beta_3 \quad \text{are linear functions in } \; s, t \tag{6.5}$$

$$\beta_i(P_j) = \delta_{ij} \quad i, j = 1, 2, 3. \tag{6.6}$$

From these basic properties other important properties of barycentric coordinates readily follow. For example, by differentiating (6.4) in any direction in the s–t plane, we get

$$\beta_1' + \beta_2' + \beta_3' \equiv 0. \tag{6.7}$$

It also follows from (6.5) and (6.6) that

$$\beta_i = 0 \quad \text{along the line } P_j P_k \quad i \neq j \neq k. \tag{6.8}$$

Barycentric coordinates will be used throughout as the blending functions for our interpolation scheme. In the next section we address the Hermite interpolation problem over one triangle (Problem 6.2). We solve the general Hermite interpolation problem over a triangular grid (Problem 6.1) in Section 6.3.

6.2.3 A Recurrence for Hermite interpolation over a triangle

In this section we present a recursive solution to the Hermite interpolation problem over a single triangle where all the entries in the index vector of the interpolant are nonnegative. Proposition 6.1 describes this recursive solution, but we begin with two helpful lemmas:

Lemma 6.1 Let $\beta_{100}, \beta_{010}, \beta_{001}$ be the barycentric coordinate functions with respect to the triangle P_{100}, P_{010}, P_{001}. Then an interpolant $I_{u,0,0}$

can be computed from the recurrence:

$$I_{u,0,0} = \beta_{100} I_{u,-1,-1} + (1-\beta_{100}) I_{u-1,0,0} \qquad u > 0 \tag{6.9}$$

$$I_{0,0,0} = \beta_{100} I_{0,-1,-1} + \beta_{010} I_{-1,0,-1} + \beta_{001} I_{-1,-1,0}. \tag{6.10}$$

Proof

- Lagrange interpolation at P_{010}, P_{001}:

  Since by (6.6)

  $$\beta_{100}(P_{001}) = \beta_{100}(P_{010}) = 0,$$

  it follows by (6.9) that

  $$I_{u,0,0}(P_{010}) = I_{u-1,0,0}(P_{010}) \quad \text{and} \quad I_{u,0,0}(P_{001}) = I_{u-1,0,0}(P_{001}) \quad u > 0.$$

  Therefore, by induction on u,

  $$I_{u,0,0}(P_{010}) = I_{0,0,0}(P_{010}) \quad \text{and} \quad I_{u,0,0}(P_{001}) = I_{0,0,0}(P_{001}).$$

  But by (6.10) $I_{0,0,0}$ is the standard linear interpolant over the triangle. Therefore $I_{u,0,0}$ interpolates the data at the nodes P_{010} and P_{001}.

- Interpolation of the j, k^{th} order derivative at P_{100}, $0 \le j+k \le u$:

  Differentiating (6.9) j times with respect to s and k times with respect to t, we get:

  $$\begin{aligned} I^{(j,k)}_{u,0,0} &= \beta_{100} I^{(j,k)}_{u,-1,-1} + (1-\beta_{100}) I^{(j,k)}_{u-1,0,0} + j\left(\beta^{(1,0)}_{100} I^{(j-1,k)}_{u,-1,-1} - \beta^{(1,0)}_{100} I^{(j-1,k)}_{u-1,0,0}\right) \\ &\quad + k\left(\beta^{(0,1)}_{100} I^{(j,k-1)}_{u,-1,-1} - \beta^{(0,1)}_{100} I^{(j,k-1)}_{u-1,0,0}\right) \end{aligned} \tag{6.11}$$

  The first term on the right hand side of (6.11) interpolates all derivatives up to order u at P_{100} by construction, and the second term vanishes because

$\beta_{100}(P_{100}) = 1$. The third and fourth terms sum to zero since both $I_{u,-1,-1}$ and $I_{u-1,0,0}$ interpolate the first $u-1$ derivatives at P_{100}. The same observation applies to the last two terms.

It follows that $I_{u,0,0}$ interpolates all derivatives up to order u at P_{100} as well as Lagrange data at P_{010}, P_{001}. The argument is symmetric for any reordering of the points. □

Lemma 6.2 Let $\beta_{100}, \beta_{010}, \beta_{001}$ be the barycentric coordinate functions with respect to the triangle P_{100}, P_{010}, P_{001}. Then an interpolant $I_{u,v,0}$ can be computed from the recurrences :

$$
\begin{aligned}
I_{u,v,0} &= \beta_{100} I_{u,v-1,0} + \beta_{010} I_{u-1,v,0} + \beta_{001} I_{u-1,v-1,0} && u, v > 0 \quad (6.12)\\
I_{u,0,0} &= \beta_{100} I_{u,-1,-1} + (1-\beta_{100}) I_{u-1,0,0} && u > 0\\
I_{0,0,0} &= \beta_{100} I_{0,-1,-1} + \beta_{010} I_{-1,0,-1} + \beta_{001} I_{-1,-1,0}
\end{aligned}
$$

Proof

- Lagrange interpolation at P_{001}:
  By (6.6) we have $\beta_{100}(P_{001}) = \beta_{010}(P_{001}) = 0$ and $\beta_{001}(P_{001}) = 1$, so by (6.12):
  $$I_{u,v,0}(P_{001}) = I_{u-1,v-1,0}(P_{001}) \quad u, v > 0.$$
  Eventually either u or v or both become zero and we can invoke Lemma 6.1 to compute $I_{u,v,0}(P_{001})$.

- Interpolation of the j, k^{th} order derivative at P_{100}, $0 \le j + k \le u$:
  Differentiating (6.12) j (k) times with respect to s (t), we get:
  $$
  \begin{aligned}
  I^{(j,k)}_{u,v,0} = {} & \beta_{100} I^{(j,k)}_{u,v-1,0} + \beta_{010} I^{(j,k)}_{u-1,v,0} + \beta_{001} I^{(j,k)}_{u-1,v-1,0}\\
  & + j\left(\beta^{(1,0)}_{100} I^{(j-1,k)}_{u,v-1,0} + \beta^{(1,0)}_{010} I^{(j-1,k)}_{u-1,v,0} + \beta^{(1,0)}_{001} I^{(j-1,k)}_{u-1,v-1,0}\right) \quad (6.13)\\
  & + k\left(\beta^{(0,1)}_{100} I^{(j,k-1)}_{u,v-1,0} + \beta^{(0,1)}_{010} I^{(j,k-1)}_{u-1,v,0} + \beta^{(0,1)}_{001} I^{(j,k-1)}_{u-1,v-1,0}\right)
  \end{aligned}
  $$

The first term on the right hand side of (6.13) interpolates all derivatives up to order u at P_{100} by construction. The second and third term vanish at P_{100} by (6.6). Terms on the second line of (6.13) add up to zero by (6.7) since by the inductive hypothesis the derivatives of all the interpolants on this second line interpolate $f^{(j-1,k)}$. By the same argument the terms on the third line sum to zero at P_{100}. Similar arguments show that $I_{u,v,0}$ interpolates derivatives up to order v at P_{010}. $\square$

Proposition 6.1 Let $\beta_{100}, \beta_{010}, \beta_{001}$ be the barycentric coordinate functions with respect to the triangle P_{100}, P_{010}, P_{001}. Then an interpolant $I_{u,v,w}$ can be computed from the following recurrences:

$$\left.\begin{array}{lll} I_{u,v,w} = \beta_{100} I_{u,v-1,w-1} + \beta_{010} I_{u-1,v,w-1} + \beta_{001} I_{u-1,v-1,w} & u,v,w>0 \\ I_{u,v,0} = \beta_{100} I_{u,v-1,0} + \beta_{010} I_{u-1,v,0} + \beta_{001} I_{u-1,v-1,0} & u,v>0 \\ I_{u,0,0} = \beta_{100} I_{u,-1,-1} + (1-\beta_{100}) I_{u-1,0,0} & u>0 \\ I_{0,0,0} = \beta_{100} I_{0,-1,-1} + \beta_{010} I_{-1,0,-1} + \beta_{001} I_{-1,-1,0} & \end{array}\right\} \quad (6.14)$$

Proof The last three recurrences have already been validated in the previous two lemmas. Therefore we need only verify the first recurrence. The proof proceeds by induction using the ordering relation $<_v$. Differentiating the first recurrence j times with respect to s and k times with respect to t $(0 \le j+k \le u)$ and using (6.5), we get

$$\begin{aligned} I^{(j,k)}_{u,v,w} \quad = \quad & \beta_{100} I^{(j,k)}_{u,v-1,w-1} + \beta_{010} I^{(j,k)}_{u-1,v,w-1} + \beta_{001} I^{(j,k)}_{u-1,v-1,w} + \\ & j[\beta^{(1,0)}_{100} I^{(j-1,k)}_{u,v-1,w-1} + \beta^{(1,0)}_{010} I^{(j-1,k)}_{u-1,v,w-1} + \beta^{(1,0)}_{001} I^{(j-1,k)}_{u-1,v-1,w}] + \qquad (6.15) \\ & k[\beta^{(0,1)}_{100} I^{(j,k-1)}_{u,v-1,w-1} + \beta^{(0,1)}_{010} I^{(j,k-1)}_{u-1,v,w-1} + \beta^{(0,1)}_{001} I^{(j,k-1)}_{u-1,v-1,w}]. \end{aligned}$$

Now the second and third terms of (6.15) vanish at P_{100} by (6.6), while the first term interpolates by the inductive hypothesis. Moreover, the interpolants on the second line all interpolate up to the $(u-1)$st derivatives at P_{100} by the inductive hypothesis, so they all interpolate $f^{(j-1,k)}$ at P_{100}. By (6.7) it follows that the terms on the second line of (6.15) sum to zero at P_{100}. A similar argument applies to the interpolants on the third line since they all interpolate $f^{(j,k-1)}$ at P_{100}, and hence the third line also sums to zero at P_{100}. The argument carries over by symmetry to P_{010} and P_{001}.

If j or k or both are zero, then the terms on the second or third or both of these terms on the right hand side vanish, but the proof is again unchanged. □

A dynamic programming algorithm for interpolation over one triangle based on the recurrence (6.14) has a pyramidal structure. It can be drawn schematically (for interpolation of first derivatives) as in Figure 6.3.

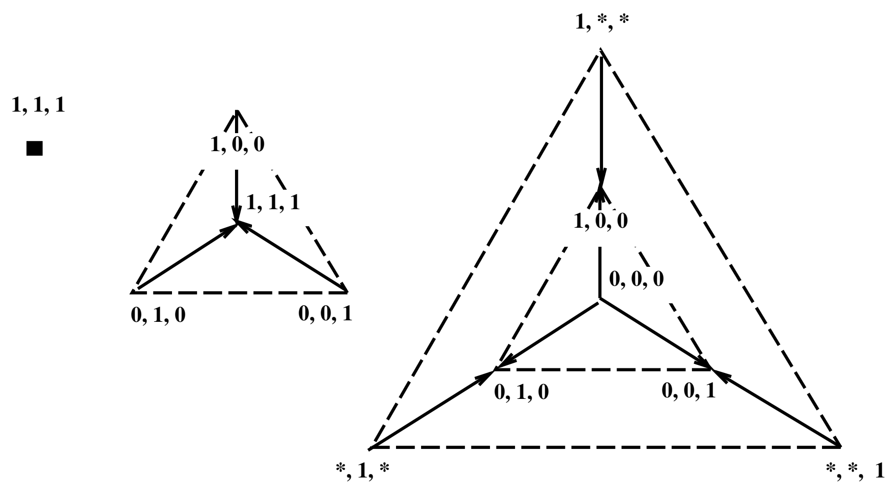


Figure 6.3: The pyramidal structure of the Hermite interpolation algorithm for the computation of $I_{1,1,1}$. Vertices denote partial interpolants. Dotted lines connect vertices on the same level of the pyramid and arrows pointing into a vertex indicate the subinterpolants used to compute the interpolant at this vertex.

Figure 6.3 shows an implementation of formula (6.14) for recursion over one triangle. In this figure, vertices denote partial interpolants. A vertex label of (α, β, γ)

denotes an interpolant $I_{\alpha,\beta,\gamma}$. An index with value -1 at some position denotes an interpolant that does not interpolate any data at the corresponding vertex of the triangle.

Figure 6.3 represents the Hasse diagram of the partial order relation $<_v$; thus there is a path from a vertex V_1 to another vertex V_2 if and only if $V_1 <_v V_2$.

Figure 6.3 also illustrates common computations between subproblems that can be used to make the overall computation more efficient. For example, the interpolant $(0,0,0)$ is used to compute the three interpolants $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$. This form of recursion utilizing shared computations is referred to as dynamic programming. We shall describe an implementation of all our recurrences in this dynamic programming fashion in Section 6.4. In the next section we derive a recurrence for computing an interpolant for Hermite data over triangular grids of arbitrary size.

6.3 Recursion over the grid

In this section we take on the general problem of Hermite interpolation at the vertices of a triangular grid (Problem 6.1). Our approach is a simple extension to the recursive procedure followed in the previous section for solving the problem over one triangle. We begin by describing the setting and notation.

Here we address the problem for a grid of size n bounded by the three lines R_0, S_0, T_0. As noted in Section 6.1.1 we shall require that each node has a nonempty set of conditions to be interpolated. This property needs to be consistently preserved so that the structure of the problem remains interpolation over a triangular grid, although the size of the grid can change.

Theorem 6.1 gives a recurrence relating the interpolant over the grid to the solutions of three simpler interpolation problems, each obtained by lowering the order of the derivatives that we are interpolating along one of the boundaries of the triangular grid while preserving the structure of the grid.

Now if r_{0jk} (r_{i0k}, r_{ij0}) are all zero, and if we decrease all these indices by one, then the line R_0 (S_0, T_0) will not have any interpolation data; thus it can be dropped leaving a smaller triangular grid. However, if lowering the number of derivatives alters the structure of the triangular grid by driving the number of derivatives only at some but not all nodes along the boundary to be negative, then keeping the triangular structure takes priority. Theorem 6.1 makes this point precise.

Theorem 6.1 A solution "Int" to the Hermite interpolation problem over a triangular grid (Problem 6.1) is given by the recurrence

$$\mathrm{Int} = \beta_R \mathrm{Int}_R + \beta_S \mathrm{Int}_S + \beta_T \mathrm{Int}_T \tag{6.16}$$

where $\beta_R, \beta_S, \beta_T$ are the barycentric coordinates with respect to the triangle bounded by R_0, S_0, T_0 and the base case is interpolation over a single triangle as described in Proposition 6.1. Here Int_R is the solution of an interpolation problem identical to Problem 6.1 except that:

- if all $r_{0jk} = 0$, then drop the line R_0 altogether.
- else $r_{0jk} = \max\{r_{0jk} - 1, 0\}$.

The interpolants $\mathrm{Int}_S, \mathrm{Int}_T$ are defined in an analogous manner.

Proof The decomposition of the problem is shown in Figure 6.4. We need to prove equation (6.16) for three distinct cases: corner nodes, edge nodes, and interior nodes. Without loss of generality (due to symmetry), we pick P_{n00}, P_{ij0} $i, j \neq 0$, and P_{ijk} $i, j, k \neq 0$ as representative points.

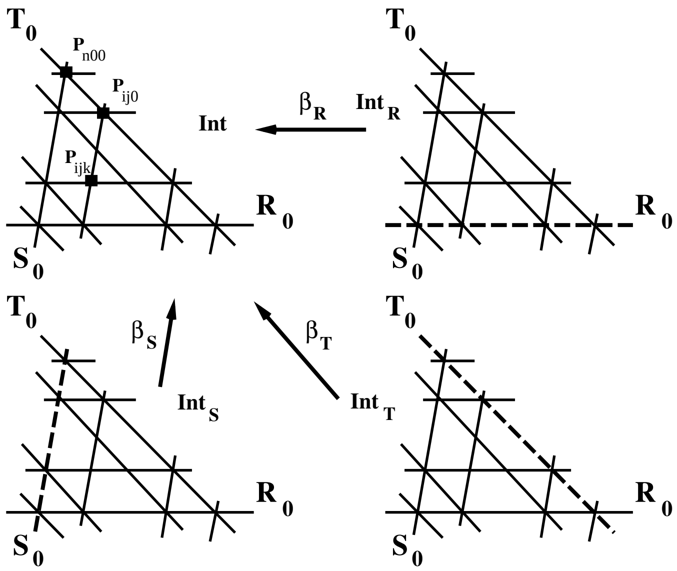


Figure 6.4: Problem decomposition. The order of the derivative data along the nodes on dotted lines is reduced by one, subject to the conditions in the statement of Theorem 6.1.

Differentiating recurrence (6.16) p times with respect to s, q times with respect to t and recalling (6.5), we get:

$$\begin{aligned} \mathrm{Int}^{(p,q)} &= \beta_R \mathrm{Int}_R^{(p,q)} + \beta_S \mathrm{Int}_S^{(p,q)} + \beta_T \mathrm{Int}_T^{(p,q)} \\ &\quad + p\left(\beta_R^{(1,0)} \mathrm{Int}_R^{(p-1,q)} + \beta_S^{(1,0)} \mathrm{Int}_S^{(p-1,q)} + \beta_T^{(1,0)} \mathrm{Int}_T^{(p-1,q)}\right) \\ &\quad + q\left(\beta_R^{(0,1)} \mathrm{Int}_R^{(p,q-1)} + \beta_S^{(0,1)} \mathrm{Int}_S^{(p,q-1)} + \beta_T^{(0,1)} \mathrm{Int}_T^{(p,q-1)}\right) \end{aligned} \tag{6.17}$$

Now we verify Lagrange and Hermite interpolation at the three representative nodes:

- P_{n00}, $0 \le p+q \le r_{n00}$:

  By property (6.6) of barycentric coordinates we have

  $$\beta_S(P_{n00}) = \beta_T(P_{n00}) = 0 \quad \beta_R(P_{n00}) = 1.$$

  Thus the second and third terms vanish and the first term on the right hand side of (6.17) interpolates the data by induction. For Lagrange interpolation case $p =$

$q = 0$; thus the proof is done. Otherwise, since $(p+q-1) \leq (r_{n00}-1)$, it follows that $\mathrm{Int}_R^{(p-1,q)}, \mathrm{Int}_S^{(p-1,q)}, \mathrm{Int}_T^{(p-1,q)}$ all interpolate $f^{(p-1,q)}$ at P_{n00}. Similarly the terms on the last line of (6.17) interpolate $f^{(p,q-1)}$ at P_{n00}.

Substituting in (6.17) and factoring, we get:

$$\begin{aligned}\mathrm{Int}^{(p,q)}(P_{n00}) &= \mathrm{Int}_R^{(p,q)}(P_{n00}) \\ &\quad + p f^{(p-1,q)}(P_{n00}) \left[\beta_R^{(1,0)}(P_{n00}) + \beta_S^{(1,0)}(P_{n00}) + \beta_T^{(1,0)}(P_{n00})\right] \\ &\quad + q f^{(p,q-1)}(P_{n00}) \left[\beta_R^{(0,1)}(P_{n00}) + \beta_S^{(0,1)}(P_{n00}) + \beta_T^{(0,1)}(P_{n00})\right]\end{aligned}$$

The terms in brackets vanish by (6.7). Therefore the result follows since by induction the first term interpolates the p, q^{th} derivative at P_{n00}.

- P_{ij0} $i, j \neq n, 0 \leq p+q \leq r_{ij0}$: Substituting into (6.17) and utilizing (6.8), we notice that:

$$\beta_T(P_{ij0}) = 0 \quad \text{and} \quad \beta_R(P_{ij0}) + \beta_S(P_{ij0}) = 1.$$

  Since by induction both $\mathrm{Int}_R^{(p,q)}$ and $\mathrm{Int}_S^{(p,q)}$ interpolate $f^{(p,q)}(P_{ij0})$, it follows that the sum of the first two terms on the right hand side of (6.17) interpolates $f^{(p,q)}(P_{ij0})$. The remaining terms vanish by the same argument as the previous case.

- P_{ijk} $i, j, k \neq 0, 0 \leq p+q \leq r_{ijk}$: By substituting into (6.17), we find that the sum of the first three term on the right hand side interpolates $f^{(p,q)}(P_{ijk})$, while the remaining terms vanish by the same argument as the previous cases.

The argument goes through essentially unchanged for derivatives with respect to s (t) only, by setting $q(p)$ to zero. $\square$

Notice that Proposition 6.1 is a special case of Theorem 6.1 when restricted to a triangular grid of size one. Thus, we do not really need a separate implementation of the recurrence (6.14).

6.4 The interpolation algorithm

The recurrences we established in previous sections give rise to a dynamic programming interpolation algorithm with a pyramidal structure as shown for the one triangle case in Figure 6.3. A dynamic programming algorithm can be thought of from two conceptually distinct yet practically equivalent points of view: bottom–up and top–down. A bottom–up implementation starts from the base of the pyramid and builds its way up until it computes the overall interpolant at the apex of the pyramid, while a top–down implementation starts from the apex and calls itself recursively, keeping track of partial interpolants to avoid unnecessary recomputations, until it gets down to the base cases. The implementation we give below is a bottom up implementation. This approach has the advantage that it does not require searching through subproblems to find common computations. This speed up is obtained by trading recursion for iteration.

The algorithm we describe in this section and the analysis that follows concentrates on uniform problems where the number of derivatives r_{ijk} – that is, the order – is identical at all nodes. This is done primarily because these problems are the simplest to describe and they seem to be the most important cases in practice.

For a uniform problem of order k, the algorithm assumes that there are $(k+1)(k+2)/2$ interpolation conditions (data) at each node. This number corresponds to the amount of Hermite data needed to specify function value and all partial and mixed partial derivatives up to order k at a point. For nonuniform cases, only the base cases and the way they are generated change; the rest of the algorithm remains the same.

We shall not discuss, in general, how these base cases are generated, but that should be clear from the construction.

Our interpolation algorithm has a pyramidal structure. The number of levels in the pyramid depends on the specific number of derivatives at the nodes and the size of the grid. For a uniform problem of order r over a triangular grid of size n, we shall show in Section 6.6 that the number of levels is $nr + n + r$.

The interpolation algorithm:

```
* Step 1: Generate base cases
* Nodes are stored in a triangular array with node (ijk) as Point[j,k]
* Only a sample is shown for illustration at pyramid level n(r+1)
* This is the topmost level at which base cases occur. The apex of the
* pyramid is at level 0.
* n : The size of the grid
* r : Order of the highest derivative to interpolate at each node
Level = n(r+1);
for row from 1 to n+1 do
   for col from 1 to row do
       SubInt[Level][(row-1)*(r+1)+1, (col-1)*(r+1)+1] = Interpolate r
                derivatives at the Point[row, co]);
   od;
od;
.. Computation of base cases in other levels of the pyramid.
* Step 2: Recur up the grid
for Level from (n+1)r+n downto 1 do
   for Row from 1 to Level+1 do
     for Column from 1 to Row do
```

```
            SubInt[Level][Row, Col] = Blend (SubInt[Level+1][Row, Col],
                                            SubInt[Level+1][Row+1, Col],
                                            SubInt[Level+1][Row+1, Col+1]);
          od;
   od;
od;
```

As the *level* loop in step 2 of the algorithm repeats, we compute partial interpolants of larger sets of data over bigger and bigger triangles in the grid. The final interpolant is obtained in *SubInt[1][1, 1]*.

The function **Blend()** implements recurrence (6.14) of Proposition 6.1 and its more general form in Theorem 6.1. The base cases at the bottom of the recursion correspond to Lagrange interpolation at the vertices of a single triangle or Taylor's expansion at a single point.

Blend() also computes the appropriate barycentric coordinates for blending. As we have shown in Proposition 6.1 and Theorem 6.1, the blending functions are the standard barycentric coordinates and thus are unique and can be computed easily.

In Section 6.6 we discuss the properties of the interpolant computed by the above interpolation algorithm including its degree and complexity. But first we work an example in detail to illustrate the steps of the proposed interpolation algorithm.

6.5 An example

Consider the problem shown in Figure 6.5. In this example, the grid size is two, and the grid lines are:

$$\begin{array}{lll} R_0 = t - 2 & S_0 = 3s - t - 4 & T_0 = 3s + 2t - 28 \\ R_1 = t - 5 & S_1 = 3s - t - 13 & T_1 = 3s + 2t - 19 \\ R_2 = t - 8 & S_2 = 3s - t - 22 & T_2 = 3s + 2t - 10. \end{array} \tag{6.18}$$

We shall investigate a problem of order one; thus only function values and partial derivatives of first order are available at the nodes P_{ijk}, $i+j+k=2$.

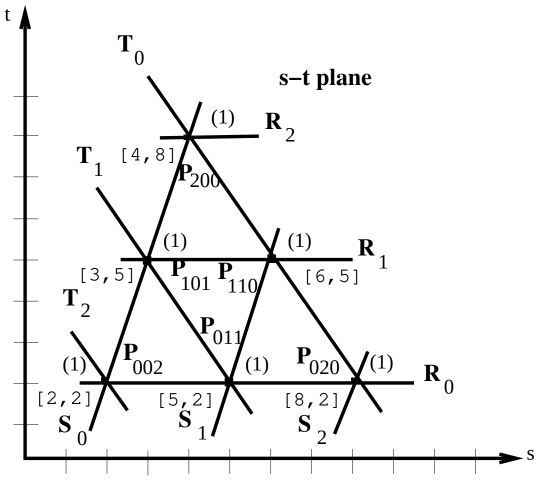


Figure 6.5: The example. Values in brackets represent the s–t coordinates of grid points.

The interpolation data for this example are

Point :	$f, \frac{\partial f}{\partial s}, \frac{\partial f}{\partial t}$	Point :	$f, \frac{\partial f}{\partial s}, \frac{\partial f}{\partial t}$
P_{200} :	$1, 2, -1$	P_{101} :	$4, 1, 1$
P_{110} :	$3, 3, 1$	P_{002} :	$2, 0, -2$
P_{011} :	$3, 1, 3$	P_{020} :	$3, 3, 6$

We now trace the steps of the algorithm described in Section 6.4. In step 1, we generate the base cases and write them in the two–dimensional array $SubInt[Level]$ where $Level = nr + n + r = 5$. The entries of this array are shown in the upright triangles in Figure 6.6. All the interpolation problems that appear in the upright triangles of Figure 6.6 are either Hermite interpolation problems at a point, in which case we can write the solution directly using Taylor's theorem, or Lagrange interpolation problems at the vertices of a triangle, which we can solve directly using the barycentric coordinates formula for $I_{0,0,0}$ in (6.14).

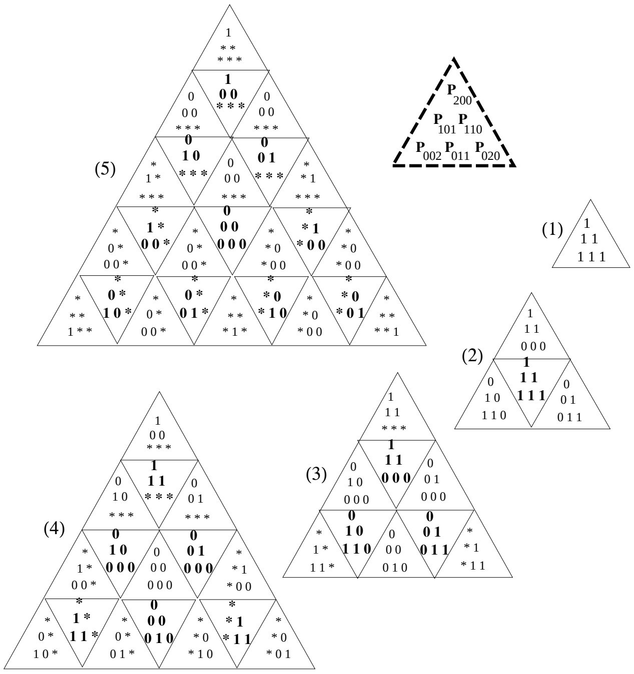


Figure 6.6: The base of the evaluation pyramid. Each of the small upright triangles represents one simple interpolation subproblem. Integers denote the number of derivatives we are interpolating at the corresponding nodes in the triangular array to the right. An "*", rather than -1, denotes no interpolation conditions at the corresponding node. The arrangement of the subproblems shows the sharing exploited by the interpolation algorithm. Subproblems written in upside-down triangles appear in the next level above it in the evaluation pyramid.

Subproblems in Figure 6.6 are encoded by the set of numbers denoting the orders of the highest derivatives that we want to interpolate at the corresponding nodes. For example, the interpolation problem at the top of Figure 6.6 ($SubInt[5][1,1]$) interpolates both function value and first derivatives at the point P_{200}. A solution to this problem, using Taylor's theorem, is given by $1 + 2(s-4) - (t-8) = 1 + 2s - t$. Similarly, using Taylor's theorem we compute:

$$\begin{aligned} SubInt[5][3,1] &= 4 + (s-3) + (t-5) = s + t - 4 \\ SubInt[5][3,3] &= 3 + 3(s-6) + (t-5) = 3s + t - 20 \end{aligned}$$

$$
\begin{aligned}
SubInt[5][5,1] &= 2-2(t-2) = -2t+6 \\
SubInt[5][5,3] &= 3+(s-5)+3(t-2) = s+3t-8 \\
SubInt[5][5,5] &= 3+3(s-8)+6(t-2) = 3s+6t-33
\end{aligned}
$$

The remaining base cases in Figure 6.6 are Lagrange interpolation problems at the vertices of single triangles. We show the computation of the entry $SubInt[5][2,1]$. First, the barycentric coordinates of $\triangle P_{200}P_{101}P_{110}$ are computed. We compute β_{200} by normalizing the equation of the line $P_{101}P_{110}$ to have the value 1 at P_{200}. Thus $\beta_{200} = (t-5)/3$. Similarly,

$$
\beta_{101} = -(3s+2t-28)/9 \quad \text{and} \quad \beta_{110} = (3s-t-4)/9.
$$

Using these barycentric coordinates we get:

$$
SubInt[5][2,1] = (t-5)/3 - 4(3s+2t-28)/9 + (3s-t-4)/3 = -s/3 - 8t/9 + 85/9.
$$

Similarly:

$$
\begin{aligned}
SubInt[5][4,1] &= 3(3s-t-4)/9 + 4(t-2)/3 + 2(-3s+19-2t)/9 \\
&= s/3 + 5t/9 + 2/9 \\
SubInt[5][4,3] &= 3(3s-t-13)/9 + 3(t-2)/3 + 3(-3s-2t+28)/9 = 3.
\end{aligned}
$$

To compute the entries at the next higher level (level 4), the function **Blend()** – which implements the recurrence in Theorem 6.1 – combines three neighboring subinterpolants in $SubInt[5]$ (Figure 6.6) into an interpolant for a larger set of data in $SubInt[4]$. For example, the entry $SubInt[4][1,1]$ is computed by blending the entries $SubInt[5][1,1]$, $SubInt[5][2,1]$, $SubInt[5][2,2]$. The blending functions are the barycentric coordinates with respect to $\triangle P_{200}P_{101}P_{110}$, which we already computed as:

$$
\beta_{200} = (t-5)/3 \qquad \beta_{101} = -(3s+2t-28)/9 \qquad \beta_{110} = (3s-t-4)/9.
$$

Thus,

$$SubInt[4][1,1] = \beta_{200} SubInt[5][1,1] + \beta_{101} SubInt[5][2,1] + \beta_{110} SubInt[5][2,2].$$

Notice that since $SubInt[5][2,1]$ and $SubInt[5][2,2]$ are identical, we can also write

$$SubInt[4][1,1] = \beta_{200} * SubInt[5][1,1] + (1-\beta_{200}) * SubInt[5][2,1].$$

which is exactly the formula we derived in Proposition 6.1. Repeating this process over and over going up the evaluation pyramid, and using the appropriate barycentric coordinates, we end up with the final interpolant:

$$\begin{aligned} SubInt[1][1,1] = \; & \frac{557\,t^5}{78732} + \frac{223\,t^4 s}{26244} + \frac{17\,t^3 s^2}{2916} + \frac{197\,t^2 s^3}{5832} + \frac{73\,t s^4}{1944} + \frac{s^5}{54} - \\ & \frac{8563\,t^4}{39366} - \frac{859\,t^3 s}{13122} - \frac{565\,t^2 s^2}{1944} - \frac{3929\,t s^3}{5832} - \frac{245\,s^4}{486} + \\ & \frac{157937\,t^3}{78732} - \frac{805\,t^2 s}{8748} + \frac{1001\,t s^2}{324} + \frac{14299\,s^3}{2916} - \\ & \frac{245165\,t^2}{39366} + \frac{12805\,t s}{6561} - \frac{57127\,s^2}{2916} - \frac{23339\,t}{19683} + \frac{150340\,s}{6561} - \frac{15409}{19683} \end{aligned}$$

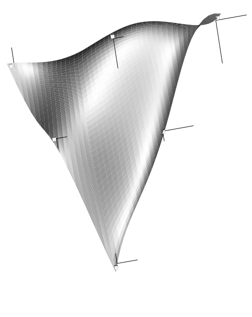


Figure 6.7: The Hermite interpolating surface for the worked example. Lines drawn at the interpolation positions represent first order derivative vectors.

6.6 Analysis of the algorithm and the interpolant

In this section we analyze the interpolation algorithm we proposed in Section 6.4 in terms of the degree of the resulting interpolant and the complexity of evaluation. Our analysis considers only uniform cases because only for these cases can we compute in a closed form the degree and complexity of the interpolant. We shall then use these measures to compare our interpolation algorithm to other interpolation schemes.

6.6.1 Degree of the interpolant

We begin by studying the case of interpolation over a single triangle; then we examine the general case of triangular grids of arbitrary size. For a single triangle, we go back to the notation of Section 6.2.3, using an index vector to denote the order of the derivatives we are interpolating at the vertices of the triangle. Our objective is to compute the degree of the interpolant $I_{r,r,r}$. We recall from Proposition 6.1 the recurrences used to compute this interpolant:

$$\begin{array}{rlll} I_{u,v,w} = & \beta_{100} I_{u,v-1,w-1} + \beta_{010} I_{u-1,v,w-1} + \beta_{001} I_{u-1,v-1,w} & u,v,w>0 & \textbf{(i)} \\ I_{u,v,0} = & \beta_{100} I_{u,v-1,0} + \beta_{010} I_{u-1,v,0} + \beta_{001} I_{u-1,v-1,0} & u,v>0 & \textbf{(ii)} \\ I_{u,0,0} = & \beta_{100} I_{u,-1,-1} + (1-\beta_{100}) I_{u-1,0,0} & u>0 & \textbf{(iii)} \\ I_{0,0,0} = & \beta_{100} I_{0,-1,-1} + \beta_{010} I_{-1,0,-1} + \beta_{001} I_{-1,-1,0} & & \textbf{(iv)} \end{array}$$

The next lemma gives a bound on the degree of the interpolant $I_{r,r,r}$ computed using the algorithm of Section 6.4.

> **Lemma 6.3** The total degree of the polynomial computed using the interpolation algorithm of Section 6.4 for a uniform problem of order r over a triangular grid of size 1 (one triangle) is less than or equal to $2r+1$.

Proof

Define $M(I_{u,v,w}) = \max(u+v, v+w, u+w)$. For the problem at hand, this maximum starts at $2r$. On application of each of the recurrences **(i)**–**(iv)** this number gets lowered by at least one per call. Eventually we must get down to one of the following two base cases:

1. Hermite interpolation at a single point $I_{u,-1,-1}$: We can obtain this problem only as a subproblem of $I_{u,0,0}$. Since $M(I_{u,0,0}) = u$, at most $(2r-u)$ recursive calls are needed to get $I_{u,0,0}$ from $I_{r,r,r}$. Now $I_{u,-1,-1}$ is obtained from $I_{u,0,0}$ by one recursive call, and by Taylor's theorem the degree of the interpolant $I_{u,-1,-1}$ is at most u. It follows that the degree of the final interpolant is bounded by $(2r-u)+1+u = 2r+1$.

2. Lagrange interpolation at the vertices of the triangle $I_{0,0,0}$: Since $M(I_{0,0,0}) = 0$, at most $2r$ recursive calls get us from $I_{r,r,r}$ to $I_{0,0,0}$. By **(iv)**, the interpolant $I_{0,0,0}$ is at most linear, so the degree of the final interpolant is bounded by $2r+1$.

   $\square$

Notice that $2r+1$ is the minimum possible degree bound for an interpolant of Hermite data of order r over a single triangle. This can be seen by restricting this problem to one of the boundaries of the triangle. The result is a univariate Hermite interpolation problem of order r at the two end points. By counting the number of interpolation conditions, we find that the solution can have a degree of up to $2r+1$.

To analyze grids of arbitrary size, notice that the order of the recursive calls in the R, S and T directions is irrelevant to the degree of the interpolant because each recursive call raises the degree by 1 and the subproblem we end up with does not depend on the order of the recursive calls. With this insight we are now ready to compute a bound on the degree of our interpolant; the following proposition will help in our analysis.

Proposition 6.2 For a uniform problem of order r on a triangular grid of size n, after $2r + m(r+1)$ recursive calls the size of the grid is at most $n - m$ whenever $m < n$.

Proof Every time we make $r+1$ recursive calls along any one direction, say R, we reduce the size of the grid by 1. Suppose then that after $2r + m(r+1)$ calls we have made:

- $\alpha(r+1) + a$ calls in the R–direction $\quad 0 \leq a \leq r$
- $\beta(r+1) + b$ calls in the S–direction $\quad 0 \leq b \leq r$
- $\gamma(r+1) + c$ calls in the T–direction $\quad 0 \leq c \leq r$

To prove our result, it is enough to show that $\alpha+\beta+\gamma \geq m$. Suppose to the contrary that $\alpha + \beta + \gamma < m$; then

$$\begin{aligned} 2r + m(r+1) &= (\alpha+\beta+\gamma)(r+1) + (a+b+c) \\ &\leq (m-1)(r+1) + 3r. \end{aligned}$$

Thus,

$$2r + m(r+1) \leq m(r+1) + 2r - 1.$$

But this is impossible. Hence $\alpha + \beta + \gamma \geq m$, so the grid size is at most $n - m$. □

Theorem 6.2 For a uniform problem of order r on a triangular grid of size n, the degree of the interpolant computed by the algorithm in Section 6.4 is at most $nr + n + r$.

Proof It follows from Proposition 6.2 that after $2r + (n-1)(r+1)$ recursive calls, the size of the grid is at most one, that is, a single triangle. To reduce the grid size to one, we must make

- $\alpha(r+1)$ calls in the R–direction.

- $\beta(r+1)$ calls in the S–direction.
- $\gamma(r+1)$ calls in the T–direction.

where $\alpha+\beta+\gamma=n-1$. Since for our analysis of the degree the order in which the calls are made in the R, S and T directions does not matter, we can reorder calls and make these $(n-1)(r+1)$ calls first. Now we are left with $2r$ recursive calls applied to a single triangle.

By Lemma 6.3 after these $2r$ recursive calls, the degree of the remaining interpolant is at most linear; thus the overall degree of the interpolant for a uniform problem of order r over a grid of size n is bounded by: $2r+(n-1)(r+1)+1=nr+n+r$.$\square$

Notice that this degree bound is again minimal as can be illustrated by restricting the original problem to one of the boundaries of the triangular grid and solving the resulting univariate Hermite interpolation problem.

6.6.2 Complexity analysis

What do we really mean by complexity analysis of an interpolation algorithm? Our intention is to measure the amount of work, in terms of number of *intermediate interpolants*, needed to evaluate the interpolant at an arbitrary point. Since we do not have a closed form solution for the interpolant, we must go through all the computations of the interpolation algorithm numerically. (If we are to evaluate the interpolant at many points, it might be beneficial to evaluate it symbolically once and for all using some symbolic computation package and then use any efficient bivariate evaluation algorithm.)

Evaluating the interpolant at one point involves the evaluation of each node (subproblem) of the evaluation pyramid. By the dynamic programming construction we are insuring that each node (subproblem) is evaluated only once. Since each subinterpolant corresponds to a node in the evaluation pyramid, determining the complexity of the algorithm reduces to counting these nodes. Our main objective is to show that this number is only polynomial, rather than exponential, in the size of the grid and the number of derivatives.

A pyramid with height N has

$$\sum_{k=1}^{N} \frac{k(k+1)}{2} = \frac{N(N+1)(2N+1)}{12} + \frac{N(N+1)}{4}$$

nodes which is $O(N^3)$. Since our algorithm has a pyramidal structure, its complexity is cubic in the height of the evaluation pyramid.

Consider the interpolation problem for a triangular grid of size n with derivatives up to order r at the nodes of the grid. By the analysis in the preceding section we know that the degree of the interpolant is at most $nr + n + r$. Since each level of the pyramid raises the degree by one, the height of the evaluation pyramid cannot exceed $nr + n + r$. Thus the number of nodes (subinterpolation problems) is $O(n^3 r^3)$.

6.6.3 Properties of the interpolant and comparison to other techniques

The general bivariate Hermite interpolation problem has been addressed by several authors, though usually as an extension of Lagrange interpolation. By extending Neville' s algorithm we have imposed very special conditions on the structure of our interpolation algorithm. The other methods we discuss below due to de Boor [16] and Gasca [27] are not constrained by such structure. The interpolation algorithm we described in Section 6.4 also makes use of the very special configuration of the nodes, while these other techniques handle more general arrangements. Both de Boor's and Gasca's techniques are essentially Lagrange interpolation algorithms; they lend themselves to Hermite interpolation by coalescing points in certain directions. The interpolation space and the interpolants built using de Boor and Ron's technique and those built using Gasca and Maeztu's method generally differ, and they are both different from the interpolation space and the interpolants computed using our dynamic programming algorithm.

De Boor and Ron [16] solve Lagrange, and hence also Hermite, interpolation problems by building a function space (from power series) with dimensions equal to the amount of data, then perform a Gram-Schmidt orthogonalization process to reduce the dimensions of the space. The process can also be interpreted as performing Gaussian elimination on the Vandermonde matrix associated with the interpolation problem. The resulting interpolants have many desirable properties including being of minimal degree.

Gasca [27] also addresses the bivariate Hermite interpolation problem as a special case of Lagrange interpolation. He arranges the interpolation nodes at the intersection points of two sets of lines, where repeated lines indicate derivatives. A basis for an interpolation space composed of products of some of these lines is then built (Newton form), and the interpolant coefficients are computed from a recursive procedure.

In this section we study some properties of the interpolants computed using our algorithm, and address how other interpolation algorithms fare regarding these properties. We shall consider five properties and compare and contrast our interpolants to those built by de Boor and Gasca.

1. **Low degree:** High degree polynomials tend to oscillate more than necessary and their evaluation is not as stable as low degree polynomials. Most algorithms that handle polynomial surfaces, such as evaluation and rendering, have complexity that depends on the degree of the polynomials. Thus it is desirable to have interpolants of as low degree as possible.

   De Boor and Ron solve Hermite interpolation problems by building a polynomial space with dimensions exactly equal to the number of interpolation conditions. A unique interpolant exists in that space by construction, and this interpolant has minimal degree. The interpolant built using Gasca's technique to solve a uniform Hermite interpolation problem of size n and order r has total degree bounded by $nr + n + r$.

As Theorem 6.2 shows, the interpolant computed using the dynamic programming approach described in Section 6.4 also has a degree bound of $nr + n + r$ for a uniform problem of order r on a grid of size n, and this degree bound is minimal. For nonuniform problems, however, our degree bound may be greater than the minimal bound obtained using either de Boor's or Gasca's technique because in our recursive approach we do not take advantage of missing data while the other two approaches do.

Although our degree bound is minimal, the degree of the interpolant itself may not be minimal. To demonstrate the difference, consider the problem of interpolating a function and its first derivatives at the vertices of a triangle. The minimum degree bound for the solution of this problem is three, as can be seen from the argument following Theorem 6.2. However, if the data is taken off a linear or quadratic polynomial, the minimal degree interpolant has degree less than three.

2. **Affine invariance:** Affine invariance means that the interpolants do not depend on the coordinate system. Equivalently, applying an affine transformation to the interpolant should produce the same interpolant as if we first transformed the data by the affine transformation and then applied the interpolation algorithm.

   While de Boor's interpolants have many desirable properties [16], his interpolants are not invariant under all affine transformations. The interpolants computed using de Boor's technique have coordinate system independence in the sense of invariance under rigid body transformations, as well as under scaling and some symmetries but they are not invariant under the full set of affine transformations. Gasca's interpolants do enjoy affine invariance because he is interpolating only directional derivatives, and his interpolants can be expressed as products of lines which are transformed by any affine transformation to a corresponding set of lines.

The interpolation recurrences of Proposition 6.1 and Theorem 6.1 use barycentric coordinates as blending functions, and barycentric coordinates are known to be invariant under all affine transformations. The only other thing that we need to check is the affine invariance of the base cases, but this is easy to verify since Taylor series expansions are affine invariant. Thus the interpolants we compute are also affine invariant.

3. **Polynomial precision:** Another property that we would like to have in an interpolation algorithm is high polynomial precision. By this property we mean that if the data we are interpolating is taken off a polynomial function then we get the same polynomial back from the interpolation algorithm. An interpolation algorithm has degree–m–precision if it reproduces polynomials up to degree m.

   De Boor's interpolants have polynomial precision of degree at least $m-1$ if the general interpolant is of degree m. In his construction [16], monomials of the highest degree are added one at a time until a space of the right dimensions is obtained. Thus by the uniqueness of his interpolant, he must reproduce all polynomials up to degree $m-1$. In Gasca's method, a Newton–like basis for the interpolation space is computed first; then the coefficients are calculated using a recursive procedure. If the polynomial lies in the interpolation space, then it is reproduced due to the uniqueness of the interpolant. However, the interpolation space itself is not unique, so we may be able to find a solution space that does not contain the original polynomial.

Our interpolant has only linear precision. We get linear precision because the base cases of our recursion (Hermite interpolation at a single point and Lagrange interpolation at the vertices of a triangle) both have linear precision, and our blending functions (barycentric coordinates) sum to one. Our interpolation algorithm also reproduces other polynomials. For example, if we pick Hermite data off an interpolant P computed using the proposed algorithm and run the algorithm on this data, the algorithm reproduces P as an interpolant. Section 6.6 elaborates further on this observation.

Even though we have insisted thus far on having some data at each node of the triangular grid, we could actually make do without this property, but then we might lose linear precision and possibly also affine invariance; thus it is better to avoid this situation. Indeed, along a line in the plane the formula for Lagrange interpolation is not unique because we can add to any Lagrange interpolant an arbitrary multiple α of the equation of the line connecting the two points as shown in Figure 6.8. Thus linear precision cannot be preserved, because it is lost for a base case. If at some node there is no data, we may recur until we get to a Hermite interpolation problem at the end points of a line. But along a line in the plane the formula for Lagrange interpolation is not unique because we can add to any Lagrange interpolant an arbitrary multiple α of the equation of the line connecting the two points as shown in Figure 6.8.

For two points we can still select a pair of barycentric coordinate functions that satisfies the analogues of conditions (6.4)–(6.6) by choosing a pair of (properly normalized) lines each passing through one of the two points (figure 6.8), but we lose linear precision since we lose linear precision on a base case. Moreover,

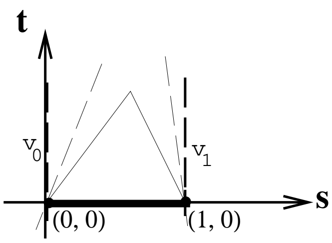


Figure 6.8: The two functions $s+\alpha t$ and $1-s-\alpha t$ are barycentric coordinates along the line $t=0$, for arbitrary choices of α.

for almost all selections of pairs of lines in Figure 6.8 we also lose affine invariance. For example, the selection of the two perpendicular lines gives us two barycentric coordinate functions (after normalization), but invariance under transformations that do not preserve right angles is lost. Since in this setting the end points of the line are embedded in a larger triangular grid, we can get a pair of barycentric coordinates along the line by restricting the triangular barycentric coordinates to this line. This selection preserves affine invariance even though linear precision is not preserved.

Now we investigate the possibility of keeping affine invariance. For any two points we can select a pair of barycentric coordinate functions that satisfies the analogues of conditions (4)–(6) by choosing a properly normalized pair of lines, each passing through one of the two points (Figure 6.8). If the two points are already embedded in a larger triangular grid, we can get a pair of barycentric coordinates along the line connecting them by restricting the triangular barycentric coordinates to this line. This selection preserves affine invariance since triangular barycentric coordinates are affine invariant. Other selections of barycentric coordinates destroy affine invariance. For example, the selection of the two vertical lines in Figure 6.8 gives us two barycentric coordinate functions (after normalization), but invariance under transformations that do not preserve right angles is lost.

4. **Symmetry:** For a triangular grid, we want an interpolant that is symmetric in the sense that it treats the R, S and T lines identically. Neither de Boor's nor Gasca's method have this symmetry property, while our interpolant does. This outcome is to be expected since we are taking advantage of the structure of the triangular grid, while the other two techniques ignore this special structure but handle more general configurations.

5. **Adding data**: Both de Boor and Gasca produce lower degree interpolants, but both are also global methods that do not take advantage of the local structure that the data may provide. Thus adding another set of data points will completely change their interpolants (de Boor addresses the continuity of his interpolants and describes the continuity of the interpolation space as a favorable condition). Since none of these techniques use dynamic programming, they do not build intermediate interpolants. On the other hand, if we keep the computations in the evaluation pyramid intact and then extend the grid, we can make use of the already computed entries of the evaluation pyramid and we do not have to begin again from scratch.

6.6.4 The interpolation space

Both de Boor and Gasca begin by defining a space of polynomials relative to a fixed interpolation problem and then show that the interpolation problem has a unique solution in the corresponding interpolation space. In contrast, rather than beginning with a polynomial space, we commence by constructing a specific algorithm for generating a polynomial interpolant. However for a fixed triangular grid, we too generate a polynomial space, namely the space of all polynomials produced by our algorithm.

The polynomials returned by our interpolation algorithm form a vector space because our base cases (Taylor expansion) are closed under addition and scalar multiplication, and our blending method (multiplying by barycentric coordinates) is linear. Moreover the interpolant we compute using our dynamic programming algorithm is the unique interpolant within this polynomial space. To prove this assertion, it suffices to show that the zero polynomial is the unique solution within our polynomial space to the homogeneous problem where all the data values are zero. This result, however, follows immediately from linear precision. For uniform problems of order r over a fixed grid of size n, the dimension of our interpolation space is $\binom{r+2}{2}\binom{n+2}{2}$ because there are $\binom{r+2}{2}$ independent interpolation conditions at each of the $\binom{n+2}{2}$ nodes of the grid.

It is easy to construct examples to demonstrate that the interpolation space we build is different from the spaces constructed by either de Boor or Gasca. What other properties our interpolation space may have and how these compare to the properties of the spaces constructed by either de Boor or Gasca remains an open question.

6.7 Conclusions and Contributions

We have proposed an interpolation algorithm based on dynamic programming to solve the bivariate Hermite interpolation problem for nodes arranged in a triangular grid. The interpolation algorithm has a pyramidal structure. The interpolant is computed recursively from three interpolants over smaller sets of data and then blended using barycentric coordinates. At the bottom of the recursion we have either Hermite interpolation at a single point or Lagrange interpolation at the vertices of a triangle. Taylor series expansion handles interpolation at individual points, while the standard linear interpolation formula (6.10) handles the Lagrange case.

Our interpolant has minimal degree bound for uniform problems and its complexity is at most cubic in the grid size and the order of derivatives. Our interpolation algorithm has linear precision and is invariant under any affine transformation. If data is added to the grid, we can make use of previous computations.

6.8 Future Work

Many questions remain unanswered and deserve further examination. The following is a short list of some of these open problems:

- Find explicit formulas for the basis functions for Hermite interpolation over a triangular grid. An explicit formula for the functions that multiply the data at the vertices of the grid would enable us to write a closed form solution for the overall interpolant directly.

- How do we characterize the space of the interpolating polynomials? In particular, how do the properties of these spaces compare to the properties of the polynomial spaces proposed by de Boor and Ron?

- Can we extend our construction to deal with scattered Hermite data instead of data over triangular grids? We need to study techniques for blending interpolants, built using our technique over individual triangles, into smooth interpolants over arbitrary triangulations. Such technique would be helpful both in finite–element analysis and in solving Hermite interpolation problems in scattered data settings.

Part III

Building Surfaces With Controlled Continuity Through Subdivision

Chapter 7

Building Smooth Subdivision Surfaces over Arbitrary Closed Polyhedra

7.1 Introduction

Building mathematical models for physical objects is one of the fundamental problems in computer aided geometric design. Manipulation of these modeled objects depends greatly on the properties of the associated mathematical models. In Chapter 1 we touched upon several classes of mathematical representations that have been proposed and used. In Part I of this thesis we considered the problem of modeling parametric curves using splines of controlled smoothness. Building surfaces that interpolate bivariate Hermite data through dynamic programming was investigated in Part II. In this part we consider yet another family of mathematical representations, and study how to build surfaces of controlled smoothness over polyhedra using subdivision.

Subdivision is one of the emerging techniques that has been receiving extensive attention in recent years [8, 19, 21, 38, 41, 50, 56]. Subdivision surfaces are obtained as the limit of a recursive refinement process applied to polyhedra. The power of this technique comes from its ability to model complicated shapes with much simpler polyhedra using a few subdivision rules. Subdivision surfaces avoid the need for trimming surface patches, that arises in boundary representations. During subdivision, each curved face of an object is represented by a portion of a polyhedron. The topology of the polyhedron automatically ensures correct connectivity of the final surface. Subdivision also automatically produces a hierarchy of polyhedra, that approximates the final limit surface. Multiresolution techniques, such as wavelets, can be defined using this hierarchy to obtain different level–of–details representations of these surfaces [18].

The quality of a subdivision surface depends on the properties of the subdivision scheme that builds it. Several subdivision schemes have been proposed in the literature [8, 19, 21, 41]. Box splines provide a natural framework for building subdivision surfaces. A well established theory of box splines, over regular parametric grids, has been developed (e.g. [15]). In this part of the thesis we adopt the four–directional, C^1 quadratic box spline, as an example subdivision scheme, though the treatment can be extended to deal with other subdivision schemes as well.

The above mentioned box spline builds C^1 piecewise quadratic surfaces only over polyhedra of special topology. In this chapter we extend the subdivision scheme associated with this box spline to build C^1 surfaces over polyhedra of arbitrary topology. In Chapter 8 we give an algorithm that introduces edges and vertices on this smooth surface.

The first section of this chapter introduces the notation and some background material on subdivision in general and the box spline we are going to use as a case study. Subdivision on the interior of a control polyhedron (interior subdivision) is addressed In the second section, where new subdivision rules that produce C^1 surfaces over arbitrary closed polyhedra are computed. In Section 7.3.3 a detailed proof of the tangent plane continuity and the regularity of the subdivision surfaces we build is given, followed by some examples.

7.2 Basic Concepts of Subdivision Schemes

To establish the notation, we start with a curve subdivision scheme. We recall one canonical example of curve subdivision schemes, Chaikin's algorithm [10], which computes quadratic B–splines, with uniform knots, recursively. One step of the algorithm is shown in Figure 7.1.

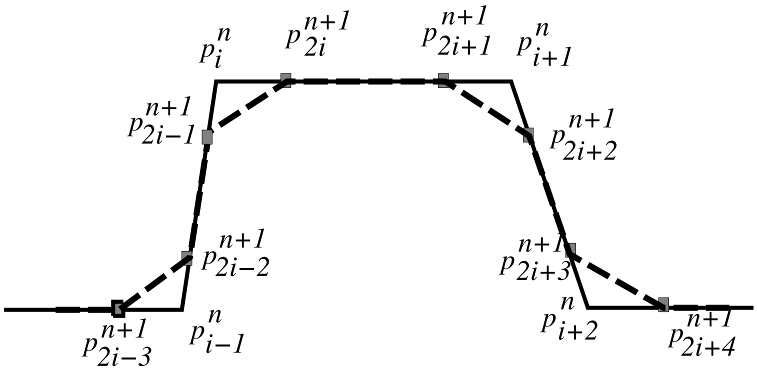


Figure 7.1: Chaikin's Algorithm. Solid (dashed) lines denote the control polygon before (after) subdivision.

The algorithm starts with a control polygon. This polygon is then transformed into a new polygon with twice as many vertices by inserting new control points 1/4 and 3/4 the way between old vertices along each segment of the control polygon. The process is then repeated over and over.

Formally, let $\mathbf{p}^0 = [p_0^0, p_1^0, \ldots]$ denote the vector of initial control points (for now we shall ignore boundaries and assume infinite length vectors of control points). Also let $\mathbf{p}^i$ denote the control polygon obtained by applying the subdivision process i times to $\mathbf{p}^0$. The subdivision process successively replaces the vector $\mathbf{p}^n$ by $\mathbf{p}^{n+1}$, where the control points in $\mathbf{p}^{n+1}$ are computed from:

$$\left.\begin{aligned} p_{2i}^{n+1} &= \tfrac{3}{4}p_i^n + \tfrac{1}{4}p_{i+1}^n \\ p_{2i+1}^{n+1} &= \tfrac{1}{4}p_i^n + \tfrac{3}{4}p_{i+1}^n \end{aligned}\right\} i \geq 0$$

Risenfield [51] proves that the curve obtained at the limit by repeating this subdivision process is the quadratic B–spline with control polygon $\mathbf{p}^0$ and uniform knots. Boldface $\mathbf{p}^i$ will generally denote the vector of control points obtained by subdividing $\mathbf{p}^0$, i

times. Individual control points (the j^{th} point) in $\mathbf{p}^i$ will be denoted p^i_j. One step of subdivision of the control polygon $\mathbf{p}^n$ is shown in Figure 7.1.

The subdivision rules for the four middle control points shown in Figure 7.1 can be written in matrix form as:

$$\begin{bmatrix} p^{n+1}_{2i-1} \\ p^{n+1}_{2i} \\ p^{n+1}_{2i+1} \\ p^{n+1}_{2i+2} \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 3 & 0 & 0 \\ 0 & 3 & 1 & 0 \\ 0 & 1 & 3 & 0 \\ 0 & 0 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} p^n_{i-1} \\ p^n_i \\ p^n_{i+1} \\ p^n_{i+2} \end{bmatrix} \tag{7.1}$$

The whole subdivision process can be described by the recurrence:

$$\mathbf{p}^{n+1} = S \cdot \mathbf{p}^n \tag{7.2}$$

The matrix S in (7.2) is usually referred to as a **subdivision matrix**. A subdivision scheme may have more than one subdivision matrix depending on the geometry of the control polygon, in the neighborhood at which subdivision is applied, or on the number of rounds of subdivision. The number of rows of the matrix S in (7.2) is double the number of columns (2:1 slant) and grows in size by 2 after each round of subdivision, thus it is eventually bi–infinite. However, when the every control point affects only a small neighborhood around it, the associated basis functions have finite supports, and only a few basis functions affect the continuity of the curve at one point; thus all the relevant properties of the limit curve resulting from using a subdivision matrix S are captured by only a finite portion of S (the part shown in (7.1) for Chaikin's algorithm). From here on, we shall refer to that finite portion as the subdivision matrix.

The values in the rows of S are referred to as the **subdivision mask**. The mask describes which control points in the vector $\mathbf{p}^i$ are used to compute the control point $\mathbf{p}^{i+1}_j$, and the corresponding weights. Usually only a few neighboring control points have non zero weights.

For surface subdivision, the process can also be represented by a recurrence of the form (7.2), but a more elaborate scheme for indexing control points is usually needed. Before introducing our case study box spline we give, in the next section, a classification of various surface subdivision schemes available in literature.

7.2.1 Available Surface Subdivision Schemes

Many surface subdivision schemes are already available in literature. In this section we give a broad classification of these schemes, to identify where does our case study box spline scheme fit in the whole picture of subdivision surfaces. In Section 7.2.2 we give more details on our specific scheme.

Based on different criteria there are many classifications of surface subdivision schemes:

Interpolatory vs. approximating schemes: As the name suggests we can classify subdivision schemes based on whether they approximate or interpolate their control polyhedra. Our box spline scheme is an approximating subdivision scheme.

Triangle vs. quadrangle based: A triangle based scheme starts with a triangulation of the control polyhedron. Every triangular face is then divided into four similar triangles by introducing new control points midway along each edge of the triangle (as shown at the top of Figure 7.2). The triangle based subdivision scheme perturbs the values of the new (and for approximating schemes also the old) control points on each round of subdivision to get the new control polyhedron. With this style of subdivision most new control points have exactly six neighbors. The remaining control points are called extraordinary vertices. These vertices require special treatment, and usually different subdivision rules need to be applied to get the desired properties of the limit surfaces.

In quadrangle based schemes, most faces of the control polyhedron are four–sided. Each four–sided face is split into four by subdivision. The remaining are isolated faces that become gradually covered with quadrangles as well; after more rounds of subdivision. Usually it is not hard to come up with subdivision rules that produce limit surfaces with the desired properties on a regular (all four–sided faces) polyhedron. Handling extraordinary vertices is much more challenging. In this chapter we extend a quadrangle based scheme to handle polyhedra where the faces are of arbitrary number of sides.

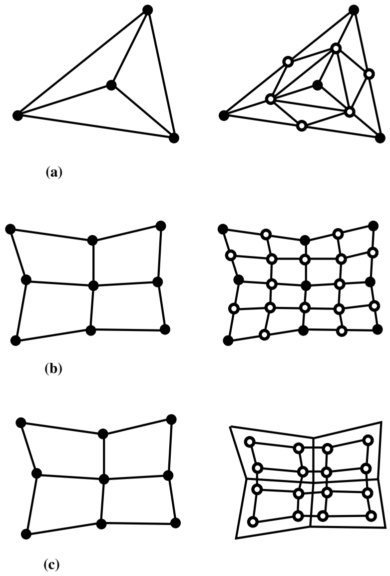


Figure 7.2: Different subdivision styles. Old control points are marked with • and new ones are marked with ◦. The top part shows a triangle based scheme, the middle shows a primal quadrangle based scheme while the bottom part shows an example of dual quadrangle based subdivision, similar to our case study box spline.

Primal vs. Dual Schemes: A subdivision surface is obtained as the limit of a refinement process performed on a grid. We can distinguish two different types of subdivision schemes, depending on how the underlying grid is refined, primal and dual. In a primal scheme, every square (triangle) in a quadrangle (triangle) based scheme, is divided into four squares (triangles), usually by halfing in each direction to get the finer grid. The old grid is a subgrid of the new one. All available triangle based schemes are primal.

Dual schemes also divide every square in the coarse grid into four, but then the polyhedral dual is taken to be the new grid. One simple example is tensor–product B–spline subdivision. Odd degree B–splines have primal subdivision schemes, since the basis functions of an odd degree B–spline are centered at the knots, while even degree B–splines have dual schemes since the basis functions of an even degree B–spline are centered between knot values. The scheme corresponding to the box spline that we are going to study is a dual subdivision scheme.

Figure 7.2 shows how different subdivision schemes perform on polyhedra.

For triangle based subdivision, all available schemes are primal. The schemes by Loop, and its extension by Hoppe *et al* are both approximating schemes. Dyn and Levin's butterfly scheme and L. Colbellt's are both interpolatory.

Doo–Sabin's [19] extension of the biquadratic tensor product biquadratic B–spline subdivision is an example of an approximating quadrangle based dual scheme, while the extension by Catmull-Clarck [8] of the tensor product bicubics B–spline is an approximating quadrangle based primal scheme.

Using the concepts and terminology established in this section we are ready to introduce our case study box spline and its associated subdivision scheme in the next section.

7.2.2 The Case Study Box Spline

Box splines are compactly supported smooth piecewise polynomial functions. They are instances of the more general polyhedral splines. In two variables, a degree n box spline is the projection of an $n+2$–dimensional hyper–cube to the plane. The reader is referred to the excellent book by de Boor et. al. [15] and the bibliography therein for a more elaborate presentation of box splines. For our purposes, it suffices to recall the relevant properties of the box spline at hand. It has the set of four direction vectors,

$$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & -1 \end{bmatrix} \tag{7.3}$$

The induced grid and the support of one basis function in the parameter plane are shown in Figure 7.3. Each subdivision step refines the grid by halfing in each direction. The basis functions are piecewise quadratic with C^1 continuity across grid lines. The associated subdivision scheme is an approximating, quadrangle based, dual scheme.

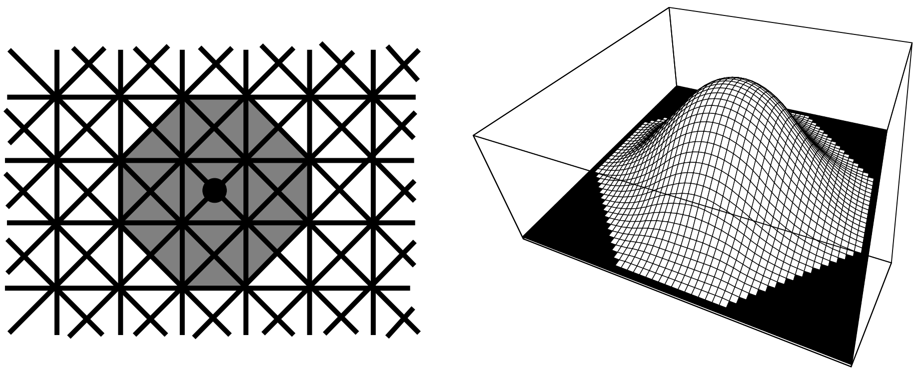


Figure 7.3: A four-directional grid, and a basis function. The dark region on the left indicates the support of the basis function centered at $\bullet$.

To specify the subdivision scheme for this box spline, we need to compute the associated subdivision mask. Generating functions can be used to compute the subdivision mask for arbitrary box splines (e.g. [56]). For this particular box spline, with direction vectors (7.3), the generating function for a basis function is

$$S(z) = \frac{1}{4}(1 + z_1)(1 + z_2)(1 + z_1 z_2)(1 + z_1/z_2).$$

To compute the subdivision formula (the expression of one basis function on the coarse grid in terms of basis functions on the fine grid), expand $S(z)$ and write the coefficient of $z_1^i z_2^j$ at position (i, j) of the z_1, z_2-plane as shown in Figure 7.4. Thus the subdivision mask on the shaded face in Figure 7.4 is given by $\{1/4, 1/2, 1/4, 0\}$ and the associated subdivision matrix is:

$$S = \frac{1}{4}\begin{bmatrix} 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix}. \tag{7.4}$$

Figure 7.4 also shows how control points subdivide under this scheme. If the control points lie in $\mathbb{R}^d$, the subdivision rules (7.4) are applied in each coordinate independently.

To use the subdivision rules shown in Figure 7.4, we must be able to place each vertex of the control polyhedron over the center of one square in the parameter plane, with the connectivity shown. This implies that every vertex of the polyhedron must have exactly four neighbors (*valence* 4) and that every face must be four–sided. These requirements restrict the set of polyhedra that the box spline can handle. In the next section we discuss these limitations and what more is needed to handle arbitrary polyhedra.

7.2.3 What is Needed to Handle Arbitrary Polyhedra?

The subdivision scheme associated with our box spline is capable of building only surfaces of genus one without boundaries. To see this, recall Euler's formula. For a

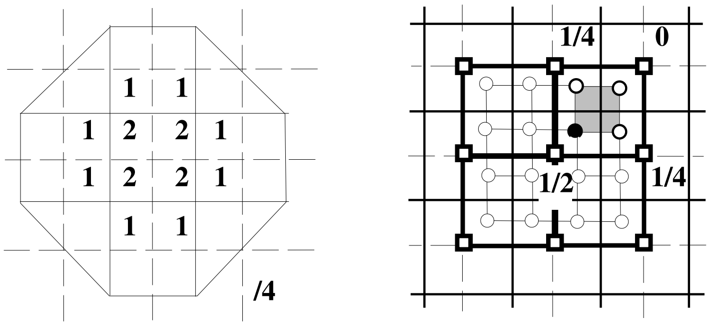


Figure 7.4: Subdivision scheme of the case–study box spline. The left figure shows the subdivision formula (the expression of a basis function on the coarse grid in terms of 12 basis functions on the fine grid). The right figure shows how control points subdivide; old control points are marked (□) and new control points are marked ∘. The subdivision mask used to compute the control point • is shown.

2-manifold, Euler's formula asserts that the number of vertices (v), edges (e), faces (f), and holes (h) are related to the genus (g) of the object by the formula:

$$v - e + f - h = 2(1 - g).$$

For a surface that can be tiled by a two–dimensional grid where each vertex has valence four and all faces are four–sided, with no holes, we have $e = 2 * v$ and $v = f$ and thus g must be one.

For an arbitrary closed polyhedron, the regular box spline subdivision rules (7.4) can be used only on four–sided faces (Figure 7.4). For other faces, however, we need to develop special subdivision rules such that the limit surface has tangent plane continuity everywhere. We shall compute these special rules in the next section. We call this process *interior subdivision* since all vertices of the control polyhedron are assumed to be interior vertices. That is, there are no boundary vertices. This will enable us to model closed surfaces of arbitrary genus.

In reality, surfaces may also have boundaries. We call the process that builds surfaces with boundaries *nonuniform subdivision*. In Chapter 8 we describe how to build special subdivision rules to be used on and next to nonuniformities.

7.3 Interior Subdivision

In this section we address the problem of building closed subdivision surfaces, that are C^1–continuous everywhere, over arbitrary closed polyhedra. We call this process interior subdivision because all the vertices of the control polyhedron are interior vertices. The next subsection explains how arbitrary polyhedra subdivide under our box spline subdivision, and specifies what we need to do to handle arbitrary closed polyhedra. Section 7.3.2 gives the actual construction. Details of the proof of tangent plane continuity of the limit surfaces are given in Section 7.3.3. We also show some example surfaces in Section 7.3.4.

7.3.1 Quadrangle Based Dual Subdivision Schemes

Using the terms introduced in Section 7.2.1, the subdivision scheme of our box spline is a quadrangle based dual scheme. Inspecting Figure 7.5 we notice that one round of subdivision of a dual scheme creates a new face for each face, edge or vertex of the old control polyhedron. In Figure 7.5, faces resulting from old r–sided faces are r–sided. Faces resulting from old vertices of valence m are also m–sided, while faces resulting from old edges are always four sided.

A key observation is that after one round of subdivision, although faces may have arbitrary number of sides, all new vertices are of valence four as shown in Figure 7.5. Thus, after the first round of subdivision, all new faces coming from old vertices as well as those coming from old edges have exactly 4 sides. Only faces coming from old faces that are not four–sided produce non four–sided faces. Since we know how to

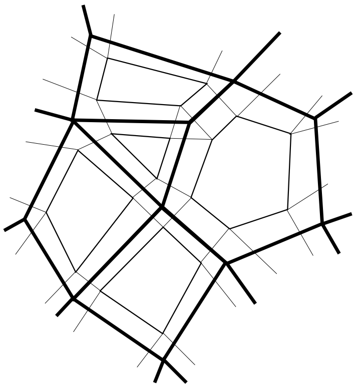


Figure 7.5: Dual subdivision for arbitrary valence polyhedra. Thick lines mark the edges of the original control polyhedron while thin lines mark the edges of the new polyhedron. After subdivision all control points have valence 4.

handle four–sided faces (box spline subdivision formula (7.4)), we need only analyze non four–sided faces. We call these faces *extraordinary faces*. The main issue in the rest of this chapter is the derivation of the subdivision rules for these extraordinary faces, and proving the smoothness of the limit surfaces that they produce.

7.3.2 Rules for Extraordinary Faces

According to the argument in the previous section, we can reformulate our subdivision process as a face subdivision process. The problem now becomes: for an n–sided polyhedral face, construct a set of subdivision rules that build a C^1 limit surface on this face. These new rules must also blend smoothly with subdivision rules applied to neighboring faces to ensure that no discontinuity is introduced between faces.

We begin by recalling the conditions stated by Reif [50] for tangent plane continuity and regularity of the limit surfaces of a subdivision process. Then we derive a set of subdivision rules for extraordinary faces. In Section 7.3.3 we prove that the limit surfaces built using the proposed rules have tangent plane continuity and regularity indeed.

Smoothness Conditions

It is well known that the tangent plane continuity of the limit surfaces of a subdivision scheme depends on the eigenstructure of the subdivision matrix [50, 56]. We shall use the analysis given by Reif [50] to study the smoothness of the limit surfaces of our subdivision scheme. First, we recall the set of conditions Reif gives for tangent plane continuity and regularity of the limit surfaces of a subdivision process:

Let M be an $r \times r$ subdivision matrix. Also let $\lambda_1,\ \lambda_2, \ldots, \lambda_r$ be the eigenvalues of M in descending moduli with corresponding eigenvectors $v_1, v_2, \ldots, v_r$ respectively. Then the following set of conditions are sufficient for M to produce tangent plane continuous and regular limit surfaces:

1. $1 = \lambda_1 > \lambda_2 = \lambda_3 > \lambda_i \quad i \geq 4$ where λ_2 has geometric and algebraic multiplicity 2.

2. The *characteristic map* of the eigenvectors v_2, v_3 of M and the basis functions is regular and injective (we shall explain this condition more, along with the proof of tangent plane continuity, and regularity in the next Section.)

Henceforth, we refer to the above two conditions as the *C–conditions* for short. For the rest of this part of the thesis we use the notation S_n to denote the $n \times n$ matrix containing the subdivision coefficients (masks) for an n–sided face. Our objective is to compute the values of the entries of S_n for arbitrary n. Our strategy will be to use the first C–condition to derive subdivision rules with correct eigenstructure, then verify that the regularity condition is satisfied as well.

The Construction

We would like our subdivision rulϒes to specialize to the regular box spline rules for four–sided faces. To this end we begin by studying the subdivision rules of four–sided faces. Faces with 4-sides subdivide according to the mask (7.4) as shown in Figure 7.4. We know that these rules applied to a polyhedron with only four-sided faces generate a limit surface that is piecewise quadratic, tangent plane continuous and regular everywhere. Studying the eigenstructure of S_4, the (4×4) matrix containing the subdivision rules (7.4), we notice:

1. The rows of S_4 are shifts of one another, that is S_4 is a *circulant* matrix.

2. The eigenvalues of S_4 in descending moduli are $1, 1/2, 1/2, 0$.

3. The corresponding matrix of eigenvectors,

$$V = \begin{bmatrix} 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & 1 \\ 1 & -1 & 0 & -1 \\ 1 & 0 & -1 & 1 \end{bmatrix}.$$

For the general case of an n–sided face, the matrix S_n must also be circulant. This property follows from index invariance, which means that the same subdivision rules are used irrespective of vertex indexing. For a circulant matrix S_n with first row $a_0, a_1, \ldots, a_{n-1}$, a well known result from linear algebra [13] asserts that the eigenvalues of S_n are the values of the polynomial

$$\sum_{k=0}^{n-1} a_k x^{kj} \qquad \text{evaluated at} \qquad x = \omega = e^{\frac{2\pi i}{n}} \quad j = 0, \ldots, n-1. \tag{7.5}$$

Now to compute the entries of S_n it suffices to specify its eigenvalues. These eigenvalues along with the (natural) assumption of index invariance specify the entries of S_n uniquely because if we pick the eigenvalues $\mu_0, \mu_1, \ldots, \mu_{n-1}$, and utilize (7.5), the problem of computing the entries of S_n reduces to solving the following system of linear equations in subdivision coefficients $a_i, i = 0, \ldots, n-1$:

$$\begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \ldots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \ldots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \ldots & \omega^{(n-1)^2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \\ \vdots \\ \mu_{n-1} \end{bmatrix}. \tag{7.6}$$

The matrix of coefficients on the left hand side of (7.6) is a well known matrix in linear algebra and Fourier analysis because it appears in Fast Fourier Transform (FFT) computation. From here on we shall refer to this matrix as an FFT–matrix.

By the first C–condition, the three leading eigenvalues must have the values $1, \lambda, \lambda$, where $0 < ||\lambda|| < 1$. Thus, for the three–sided–face subdivision matrix we need only specify the value of λ such that $||\lambda|| < 1$. For faces with more than four sides, there are many choices for the remaining eigenvalues, though their magnitudes must always be less than λ. In all cases we need to make sure that the second C–condition is satisfied as well.

One choice that guarantees that the mask reduces to the regular box spline mask when the number of sides is 4, is to pick the largest eigenvalue to be 1, the second and third largest to equal $1/2$ and the remaining eigenvalues to be zero. Since the eigenvalues in the first C–condition are ordered only by magnitude we must select three μ_i's to have the values $1, 1/2, 1/2$. We notice that the eigenvalues $1, 1/2, 1/2$ of S_4 correspond to $j = 0, 1, 3$ in (7.5). To generalize to S_n we pick

$$\mu_0 = 1 \quad \mu_1 = 1/2 \quad \mu_{n-1} = 1/2 \quad \text{and} \quad \mu_i = 0 \text{ for } i = 2, \ldots, n-2.$$

This selection also simplifies the computations in the general case.

Now utilizing the properties of the FFT matrix, the system of equations (7.6) solves to:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \dots & \omega^{-(n-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \dots & \omega^{-2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \dots & \omega^{-(n-1)^2} \end{bmatrix} \begin{bmatrix} 1 \\ 1/2 \\ 0 \\ \vdots \\ 1/2 \end{bmatrix}$$

$$= \frac{1}{n} \begin{bmatrix} 2 \\ 1+\cos(\theta) \\ 1+\cos(2\theta) \\ \vdots \\ 1+\cos((n-1)\theta) \end{bmatrix} \text{ where } \theta = \frac{2\pi}{\mathrm{n}}. \tag{7.7}$$

This solution enjoys the following properties:

1. The mask we obtain is symmetric since

$$\cos(r\theta) = \cos(r\frac{2\pi}{n}) = \cos((n-r)\frac{2\pi}{n}) = \cos((n-r)\theta).$$

2. All the coefficients, a_i, $i = 0, \dots, n-1$, are positive since $|\cos(\theta)| \leq 1$.

3. By construction, the new rules specialize to the box spline rules for $n = 4$.

4. The computation of the eigenvectors and thus the regularity analysis is simpler than for other selections.

7.3.3 Proof of Tangent Plane Continuity and Regularity

The first C–condition is built into the construction of the subdivision mask of the last section, thus we need only to verify the second C–condition.

In Reif's analysis subdivision is treated as a hole filling process. Each face is treated as a hole that gets filled gradually by subdivision. Tangent plane continuity of the limit surface on the inside of a face is proved by:

1. Computing the second and third eigenvectors of the subdivision matrix and pairing their entries to get a set of (two–dimensional) control points.
2. Subdividing the polyhedron obtained by connecting the resulting control points to get a surface.
3. An annular region of the resulting subdivision surface is called the characteristic map of the subdivision scheme. The width of this annular region depends on the specific subdivision scheme and the size of the support of its basis functions. Reif [50] proves that if this surface is regular and forms an injective map of the parameter space then the subdivision scheme produces tangent plane continuous and regular limit surfaces.

We begin the proof by recalling one standard result from linear algebra [13], which states that the eigenvectors of the $n \times n$ circulant matrix S_n, corresponding to the repeated second largest eigenvalue, can be written as:

$$v_2 = \mathcal{R}e\left[1, \omega, \omega^2, \ldots, \omega^{(n-1)}\right]^T, \quad v_3 = \mathcal{I}m\left[1, \omega, \omega^2, \ldots, \omega^{(n-1)}\right]^T, \tag{7.8}$$

where $\omega = e^{\frac{2\pi i}{n}}$.

Actually any pair of eigenvectors v_2, v_3 of S_n corresponding to eigenvalue $1/2$ are not unique since by scaling and taking linear combinations of any such pair we get other eigenvectors for S_n. However, the selection in (7.8) bears a special advantage because it leads to simple formulas for the eigenvector entries and exploits the symmetry, around the origin, of the eigenvector entries.

For the rest of the proof we shall use a single vector with complex entries (V) to represent both eigenvectors v_2, v_3. This vector V is defined by: $V = v_2 + iv_3$. Using this interpretation, plotting v_2 against v_3 corresponds to plotting the entries of V in the complex plane. For example, from (7.8) we have

$$V = \left[1, \omega, \omega^2, \ldots, \omega^{(n-1)}\right]^T \quad \text{where} \quad \omega = e^{\frac{2\pi j}{n}}. \tag{7.9}$$

Henceforth we shall fix ω to denote the n^{th} root of unity $e^{(2\pi i/n)}$, and θ to denote the angle $2\pi/n$, thus $\omega = e^{i\theta}$.

To compute the characteristic map for the subdivision scheme at hand we need to compute an extension $\overline{V}$ of the eigenvector V in (7.9), to a 3–ring of control points around an n–sided face. This requires building the subdivision matrix $\overline{S}_n$ over a 3–ring of control points. Figure 7.6 shows the control points in the support of $\overline{S}_n$.

From here on we shall use double indexing for the control points, as shown in Figure 7.6. Control points are arranged in blocks of 9–points each. The first index denotes the number of the control point within the block, while the second index denotes the block number.

The following lemma gives formulas for the entries of $\overline{V}$.

Lemma 7.1 Let $\overline{V}$ be an eigenvector of $\overline{S}_n$ (the extension of S_n to a 3–ring of control points) that extends the eigenvector V in (7.9). Then, using the indexing of control points in Figure 7.6, we have:

1. $\overline{V}_{1,j} = \omega^j$. $\quad$ (7.10)
2. $\overline{V}_{2,j} = (2 + \omega^{-1})\omega^j$. $\quad$ (7.11)
3. $\overline{V}_{3,j} = (3 + (\omega + \omega^{-1})/2)\omega^j$.
4. $\overline{V}_{4,j} = (2 + \omega)\omega^j$.
5. $\overline{V}_{5,j} = (3 + 2\omega^{-1})\omega^j$.

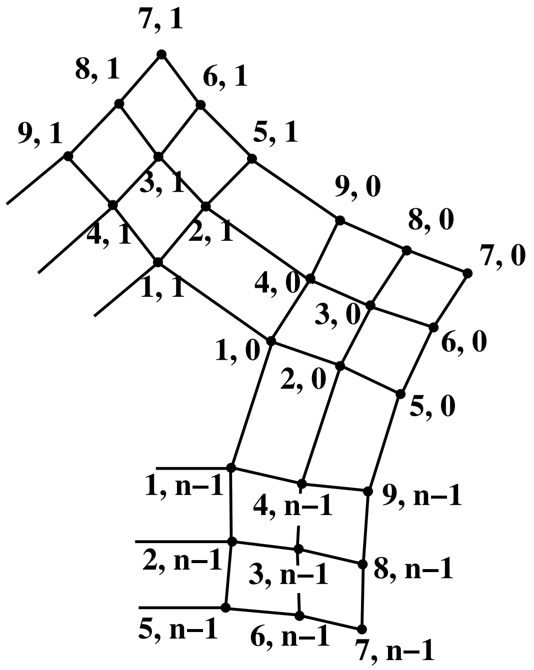


Figure 7.6: The support of the extended subdivision matrix $\overline{S}_n$ over a 3–ring.

6. $\overline{V}_{6,j} = (4 + \omega^{-1} + (\omega + \omega^{-1})/4)\omega^j$.
7. $\overline{V}_{7,j} = (5 + (\omega + \omega^{-1}))\omega^j$.
8. $\overline{V}_{8,j} = (4 + \omega + (\omega + \omega^{-1})/4)\omega^j$.
9. $\overline{V}_{9,j} = (3 + 2\omega)\omega^j$.

where $\omega = e^{2\pi i/n}$ and $j = 0, \ldots, n-1$

Proof:

Grouping the n control points with indices $\{1, j\}$, $j = 0, \ldots n-1$ together, and similarly for control points with indices $\{k, j\}$, $k = 2, \ldots 9$, we can write $\overline{S}_n$ in block form as:

$$\overline{S}_n = \begin{bmatrix} S_n & 0 & 0 & 0 & 0 & \dots & 0 \\ 1/2I + 1/4L & 1/4I & 0 & 0 & 0 & \dots & 0 \\ 1/2I & 1/4I & 0 & 1/4I & 0 & \dots & 0 \\ 1/2I + 1/4R & 0 & 0 & 1/4I & 0 & \dots & 0 \\ 1/4I & 1/2I & 0 & 1/4L & 0 & \dots & 0 \\ 1/4I & 1/2I & 1/4I & 0 & 0 & \dots & 0 \\ 0 & 1/4I & 1/2I & 1/4I & 0 & \dots & 0 \\ 1/4I & 0 & 1/4I & 1/2I & 0 & \dots & 0 \\ 1/4I & 1/4R & 0 & 1/2I & 0 & \dots & 0 \end{bmatrix}$$

where I is the $n \times n$ identity matrix, R, L are $n \times n$ permutation matrices specified by their first rows:

$$\begin{aligned} R[1,i] &= [0,1,0,\dots,0]\,. \\ L[1,i] &= [0,0,\dots,0,1]\,. \end{aligned}$$

Now we solve for the entries of $\overline{V}$ using the property that $\overline{V}$ is an eigenvector of $\overline{S}_n$ corresponding to eigenvalue 1/2. That is,

$$\overline{S}_n \cdot \overline{V} = \overline{V}/2, \tag{7.12}$$

and we notice:

1. Formula (7.10) is a restatement of (7.9).

2. Using the second block row of $\overline{S}_n$ in (7.12) we get one equation in one unknown $\overline{V}_{2,j}$:

   $$\omega^j/2 + \omega^{j-1}/4 + \overline{V}_{2,j}/4 = \overline{V}_{2,j}/2,$$

   which solves to (7.11).

   By similar computations, utilizing the block form of $\overline{S}_n$, we obtain the rest of the formulas in the lemma statement. □

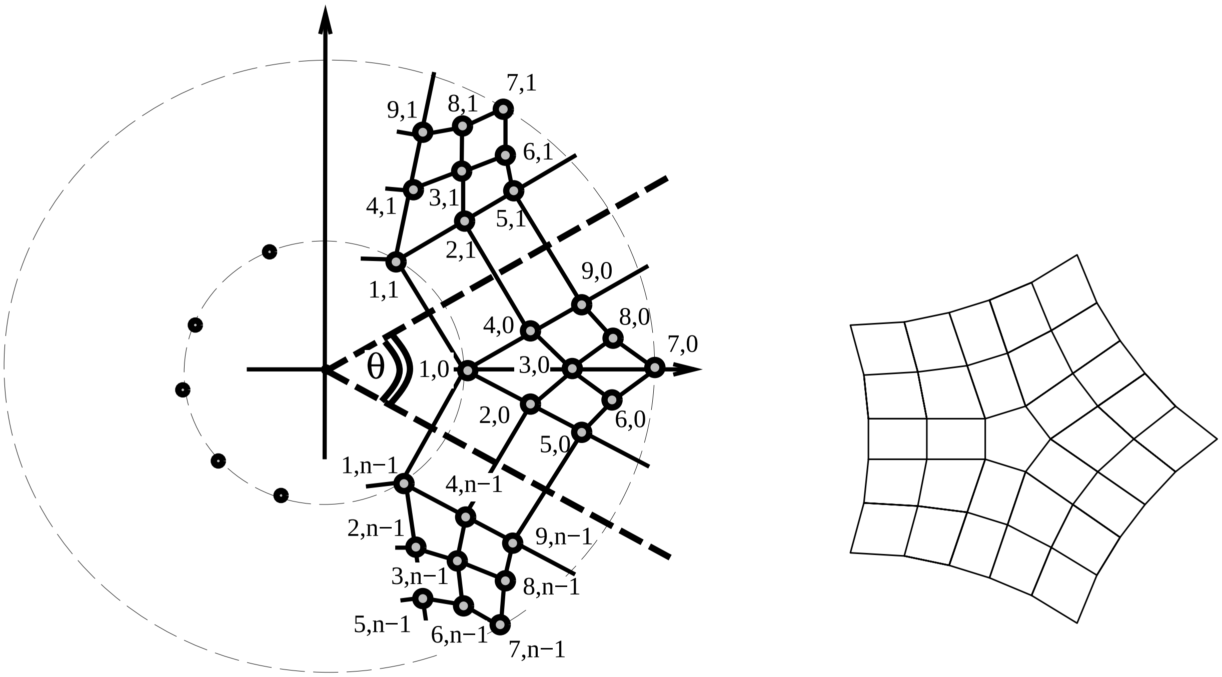


Figure 7.7: Plotting the entries of $\overline{V}$ in the complex plane. A valence 5 example is also shown to the right.

Figure 7.7 shows the layout of the entries of $\overline{V}$ in the complex plane, and a valence five example. Inspecting the geometry in Figure 7.7, and using the values of the entries of $\overline{V}$ computed in Lemma 7.1, we observe the following:

Lemma 7.2 In the plot of the eigenvector $\overline{V}$ computed in Lemma 7.1, (Figure 7.7), the following statements hold:

1. The plot of V is n-way symmetric around the origin.

2. The control points with index $\{k, j\}$, $k = 1, \ldots, 9$ for fixed $j \in \{0, \ldots, n-1\}$ lie in the sector specified by :

$$\left.\begin{aligned} \sin((j-1/2)\theta)x - \cos((j-1/2)\theta)y < 0 \\ \sin((j+1/2)\theta)x - \cos((j+1/2)\theta)y > 0 \end{aligned}\right\} \qquad (7.13)$$

where $\theta = 2\pi/n$.

Proof:

The proof of the first part follows directly from Lemma 7.1 by observing that multiplication by the n^{th} root of unity ω corresponds to a positive rotation by an angle $2\pi/n$ about the origin of the complex plane.

For the second part, substitute the entries of $\overline{V}$ computed in Lemma 7.1 in (7.13). For example, substituting for (x, y) by $(\mathcal{R}e[\overline{V}_{4,j}], \mathcal{I}m[\overline{V}_{4,j}])$, and using,

$$a = (j - 1/2)\theta, \quad b = j\theta, \quad c = (j + 1)\theta, \quad d = (j + 1/2)\theta$$

we get:

$$\begin{aligned}
\sin(a)x - \cos(a)y &= sin(a)(2\cos(b) + \cos(c)) - \cos(a)(2\sin(b) + \sin(c)) \\
&= 2\sin(a)\cos(b) + \sin(a)\cos(c) - 2\cos(a)\sin(b) - \cos(a)\sin(c) \\
&= 2\sin(a - b) + \sin(a - c) \\
&= 2\sin(-\theta/2) + \sin(-\theta/2) < 0 \qquad \text{for all } n \geq 3
\end{aligned}$$

Similarly,

$$\begin{aligned}
\sin(d)x - \cos(d)y &= \sin(d)(2\cos(b) + cos(c)) - \cos(c)(2\sin(b) + \sin(c)) \\
&= 2\sin(a)\cos(b) + \sin(a)\cos(c) - 2\cos(a)\sin(b) - \cos(a)\sin(c) \\
&= 2\sin(a - b) + \sin(a - c) \\
&= 2\sin(\theta/2) + \sin(-\theta/2) > 0 \qquad \text{for all } n \geq 3 \quad \Box
\end{aligned}$$

The last lemma asserts that the control points of the eigenvector $\overline{V}$ with indices $\{k, j\}$, $k = 1, \ldots, 9$ lie in the sector specified by the two lines (7.13). For example, the control points with indices $\{1, 0\}$–$\{9, 0\}$ lie in the sector of angle θ enclosed by the dotted lines in Figure 7.7.

It follows that it is sufficient to analyze the smoothness (injectivity and regularity) of the characteristic map only on one sector of the map shown in Figure 7.7. In

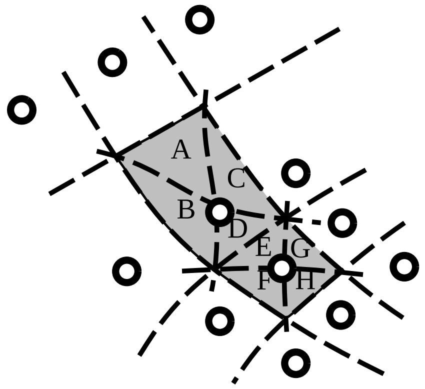


Figure 7.8: Dotted lines are isoparametric lines. We need to prove injectivity and regularity only over the shaded region.

particular, drawing the iso–parametric lines as shown in Figure 7.8, and employing symmetry, we need to prove the injectivity and regularity of the limit surface only on the shaded triangular regions marked (A)–(H).

To perform this analysis (of injectivity and regularity) we express the limit surface in terms of Bézier coefficients. This is done by expressing one basis function of the box spline at hand in Bézier form using the property that the basis function is C^1 piecewise–quadratic and utilizing the conditions given by Farin [24] for tangent plane continuity of a surface expressed in triangular Bernestein form across two Bézier triangles. The coefficients shown in Figure 7.9 are normalized by 16.

Now we (use Mathematica to) derive the Bézier coefficients of the limit surface over the shaded area in Figure 7.8. Injectivity of the limit surface is verified using the characterization by Goodman and Unsworth [34] of the conditions under which Bézier coefficients produce injective bivariate maps. For a quadratic, their condition amounts to verifying nine inequalities on the Bézier coefficients. Regularity is also proved using the Bézier representation, by computing the Jacobian over each of the shaded triangles and showing that it does not vanish. The Mathematica code used

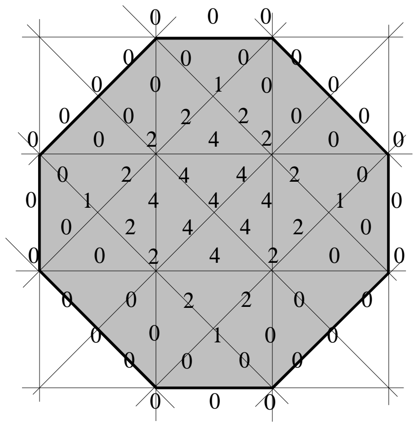


Figure 7.9: Bézier coefficients of a basis function of the box spline.

to compute the Bézier coefficients and verify both the injectivity and regularity of the characteristic map is available by anonymous ftp to "cs.rice.edu" under "/public/habib/edgeins/proof.txt". A listing is also included in Section 7.3.5

7.3.4 Examples

Figure 7.10 shows two examples of C^1 surfaces built using the subdivision rules computed in Section 7.3.2. Figure 7.10(a) shows an object with genus zero obtained by subdividing an octahedron. Notice the small triangles at the center of each original triangular face of the octahedron. Figure 7.10(b) shows a double torus.

7.3.5 Mathematica Tools

```
(* I'm implementing Complex Numbers myself (of course as pairs) *)
(* Complex Multiplication *)
MplyZ[Z1_, Z2_] := Simplify[{Z1[[1]]Z2[[1]]-Z1[[2]]Z2[[2]],
         Z1[[1]]Z2[[2]]+Z1[[2]]Z2[[1]]}];


(* "w" will denote "Omega = E^(2 Pi I/n)" the nth Root of 1 *)
```

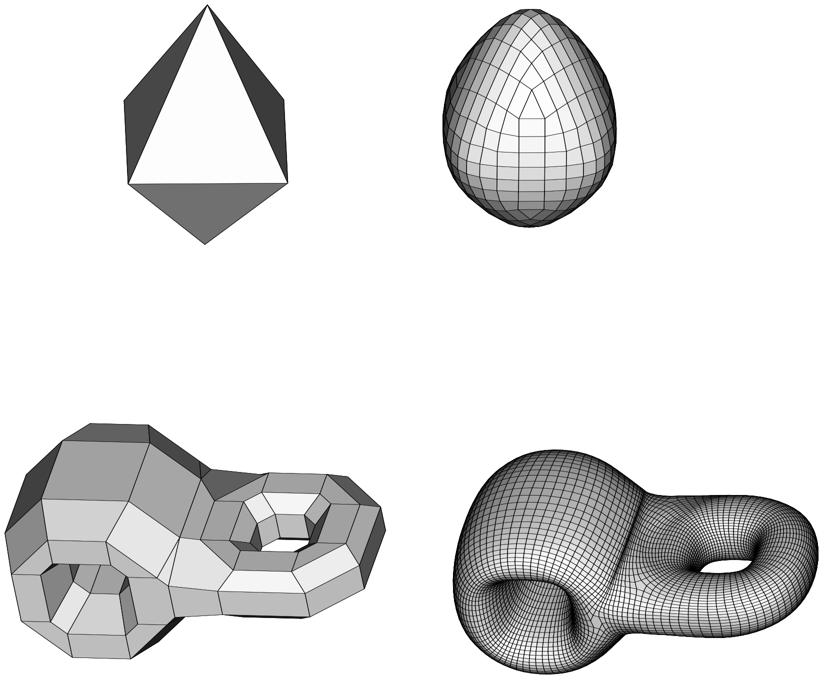


Figure 7.10: a. Subdividing an octahedron   b. A double torus

```
w = {Cos[2 Pi/n], Sin[2 Pi/n]};

(* wm1 = w^(-1) *)
wm1 = {Cos[2 Pi/n], -Sin[2 Pi/n]};

(* V is the eigenvector on a ring of size 3, after removing all
symmetries.  The formulas below are obtained from a diagram
that shows the subdivision of control points of one "sector"
of the eigenvector. Indexing of points 1--9 is the same as in
Figure 7.6 *)
```

```
V = Table[0, {i, 15}];
V[[1]]={1, 0};
V[[2]]={2+Cos[2 Pi/n], -Sin[2 Pi/n]};
V[[3]]={2+Cos[2 Pi/n], Sin[2 Pi/n]};
V[[4]]=V[[1]]+(V[[2]]+V[[3]])/2;
V[[7]]=V[[4]]+(V[[2]]+V[[3]])/2;
V[[5]]=V[[2]]+(V[[1]]+MplyZ[ V[[3]], wm1])/2;
V[[6]]=V[[2]]+(V[[1]]+V[[4]])/2;
V[[8]]=V[[3]]+(V[[1]]+V[[4]])/2;
V[[9]]=V[[3]]+(V[[1]]+MplyZ[w, V[[2]]])/2;
V[[10]]= MplyZ[ w, V[[3]]];
V[[11]]= MplyZ[ w, V[[4]]];
V[[12]]= MplyZ[ w, V[[6]]];
V[[13]]= MplyZ[ w, V[[1]]];
V[[14]]= MplyZ[ w, V[[2]]];
V[[15]]= MplyZ[ w, V[[5]]];

V = Simplify[Expand[V]];

(* ControlPoints is the array of control points of
 the characteristic map, again after removing all symmetries.
 The entries are organized in a two-dimensional array which
 represent their neighborhood relation correctly. Actually
 only a 3 by 4 array is needed as shown in Figure 7.7, namely
 the array obtained by ignoring the first row ControlPoints below.
*)
```

```
ControlPoints = Table[0, {i, 5}, {j, 3}];
ControlPoints[[1, 1]]= V[[10]];
ControlPoints[[1, 2]]= V[[11]];
ControlPoints[[1, 3]]= V[[12]];
ControlPoints[[2, 1]]= V[[13]];
ControlPoints[[2, 2]]= V[[14]];
ControlPoints[[2, 3]]= V[[15]];
ControlPoints[[3, 1]]= V[[1]];
ControlPoints[[3, 2]]= V[[3]];
ControlPoints[[3, 3]]= V[[9]];
ControlPoints[[4, 1]]= V[[2]];
ControlPoints[[4, 2]]= V[[4]];
ControlPoints[[4, 3]]= V[[8]];
ControlPoints[[5, 1]]= V[[5]];
ControlPoints[[5, 2]]= V[[6]];
ControlPoints[[5, 3]]= V[[7]];

(* A function to convert the coordinates of two points in s-t plane
   to the equation of the line going through them. *)
Pts2Line [s1_, t1_, s2_, t2_]:= If [t1 == t2, t - t1,
        Expand[ (s - s1) - (s2 - s1)*(t -t1)/(t2- t1)]];

(* Below are the Bezier coefficients of one basis function of the
   box spline under consideration. The entries are obtained by
   enforcing the C^1 condition across the boundaries and across
   grid lines, along with the fact that the basis function is
```

```
   piecewise quadratic. The coefficients are correct only up to
   a scaling factor (8) but this factor is irrelevant for our
   purposes since it multiplies all Bezier coefficients.
   The break up into Odd and Even coefficients seems to facilitate
   storage and indexing of Bezier coefficients. *)
BasisOddCoefficients =
         {{0, 0, 0, 0, 0, 0, 0},
          {0, 0, 0, 1, 0, 0, 0},
          {0, 0, 2, 4, 2, 0, 0},
          {0, 1, 4, 4, 4, 1, 0},
          {0, 0, 2, 4, 2, 0, 0},
          {0, 0, 0, 1, 0, 0, 0},
          {0, 0, 0, 0, 0, 0, 0}};

BasisEvenCoefficients = {{0, 0, 0, 0, 0, 0},
           {0, 0, 2, 2, 0, 0},
           {0, 2, 4, 4, 2, 0},
           {0, 2, 4, 4, 2, 0},
           {0, 0, 2, 2, 0, 0},
           {0, 0, 0, 0, 0, 0}};

GetBez[CtrlPts_]:=Module[{i, j, D1, D2},
D1 = Dimensions[CtrlPts][[1]];
D2 = Dimensions[CtrlPts][[2]];
(* OddCoefficients (EvenCoefficients) is a global table containing
 Odd (Even) Bezier Coefficients of the complete characteristic map
```

```
 (modulo symmetries). *)
OddCoeffs = Table[{0, 0}, {i, 2 D1 +1}, {j, 2 D2 + 1}];
EvenCoeffs = Table[{0, 0}, {i, 2 D1}, {j, 2 D2}];

(* Go thru ControlPoints and add the entries of the corresponding
   basis function to the two global (tallying) arrays. For point
   [Ro, Col], the basis function is centered at [2Ro, 2Col]; thus,
   it extends between (2Ro-3, 2Col-3) & (2Ro+3, 2Col+3) in the
   array of odd coefficients and the corresponding entries in the
   even entries array *)
For[Ro=1, Ro<=D1, Ro++,
   For[Col=1, Col<=D2, Col++,
        (* Processing CtrlPts[Ro, Col] *)
        (* Odd numbered coefficients first   *)
        For[i=1, i<=7, i++,
          For[j=1, j<=7, j++,
           If[(2Ro-4+i<=0)||(2Col-4+j<=0)||
              (2Ro-4+i>2D1+1)||(2Col-4+j>2D2+1),1,
                (OddCoeffs[[2Ro-4+i, 2Col-4+j]] +=
                  CtrlPts[[Ro, Col]]*BasisOddCoefficients[[i, j]];
        );];];];
        (* Now even numbered coefficients *)
        For[i=1, i<=6, i++,
          For[j=1, j<=6, j++,
               If[(2Ro-4+i<=0)||(2Col-4+j<=0)||
                  (2Ro-4+i>2D1)||(2Col-4+j>2D2),1,
```

```
                    (EvenCoeffs[[2Ro-4+i, 2Col-4+j]] +=
                     CtrlPts[[Ro, Col]]*BasisEvenCoefficients[[i, j]];
                  );];
        ];];
 ];];
];   (* End procedure GetBez *)

(* Compute Bezier Coefficients for "ControlPoints" *)
GetBez[ControlPoints];

(* Here we need to verify the Goodman condition on the Eight
  triangles given below, with the Bezier coefficients in canonical
  order. First three coefficients on one side, then two, then one
  at the opposite vertex  *)
MapTriangles= Table[Tri[j], {j, 8}];
Tri[1]= {OddCoeffs[[5, 3]], OddCoeffs[[5, 4]], OddCoeffs[[5, 5]],
        EvenCoeffs[[5, 3]], EvenCoeffs[[5, 4]], OddCoeffs[[6, 4]]};
Tri[2]= {OddCoeffs[[5, 3]],OddCoeffs[[6, 3]], OddCoeffs[[7, 3]],
        EvenCoeffs[[5, 3]], EvenCoeffs[[6, 3]], OddCoeffs[[6, 4]]};
Tri[3]= {OddCoeffs[[5, 5]], OddCoeffs[[6, 5]], OddCoeffs[[7, 5]],
        EvenCoeffs[[5, 4]], EvenCoeffs[[6, 4]], OddCoeffs[[6, 4]]};
Tri[4]= {OddCoeffs[[7, 3]], OddCoeffs[[7, 4]], OddCoeffs[[7, 5]],
        EvenCoeffs[[6, 3]], EvenCoeffs[[6, 4]], OddCoeffs[[6, 4]]};
Tri[5]={OddCoeffs[[7, 3]], OddCoeffs[[7, 4]], OddCoeffs[[7, 5]],
        EvenCoeffs[[7, 3]], EvenCoeffs[[7, 4]], OddCoeffs[[8, 4]]};
Tri[6]= {OddCoeffs[[7, 3]], OddCoeffs[[8, 3]], OddCoeffs[[9, 3]],
```

```
        EvenCoeffs[[7, 3]], EvenCoeffs[[8, 3]], OddCoeffs[[8, 4]]};
Tri[7]= {OddCoeffs[[7, 5]], OddCoeffs[[8, 5]], OddCoeffs[[9, 5]],
        EvenCoeffs[[7, 4]], EvenCoeffs[[8, 4]], OddCoeffs[[8, 4]]};
Tri[8]= {OddCoeffs[[9, 3]], OddCoeffs[[9, 4]], OddCoeffs[[9, 5]],
        EvenCoeffs[[8, 3]], EvenCoeffs[[8, 4]], OddCoeffs[[8, 4]]};

(* Now need to verify Goodman's condition on each of the eight
 * triangles. Goodman's condition can be reduced to :
 *
 *                       (6)
 *
 *                   (4)       (5)
 *
 *               (1)      (2)      (3)
 * 1: Computing the barycentric coordinates of (1), (3), (6) all
 *    with respect to the triangle (2)-(4)-(5).
 * 2: Checking that for each of the three points exactly one of
 * the barycentric coordinates is -ve.
 * Actually we can be more specific:
 * (1): beta4, beta2 >0 & beta5 <0
 * (3): beta2, beta5 >0 & beta4 <0
 * (6): beta4, beta5 >0 & beta2 <0 *)
(* Get Barycentric coordinates of a triangle given by its vertices
 * GetBary returns a matrix to be multiplied by the column:
 * Transpose[{x0, y0, 1}] in order to give the B-coordinates of
 * {x0, y0} with respect to the triangle P1-P2-P3 *)
```

```
GetBary[P1_, P2_, P3_]:= Inverse[Join[Transpose[{P1, P2, P3}],
        {{1, 1, 1}}]];

(* ChkInjective[Tri] Checks the injectivity of the surface specified
 *  by the six Bezier coefficients in Tri. It returns a matrix
 * of nine inequalities to be satisfied if the map is injective *)
ChkInjective[Tri_]:= Module[{Barys},
                (Barys = GetBary[Tri[[2]], Tri[[4]], Tri[[5]]];
                Betas = Barys .
                      Join[Transpose[{Tri[[1]], Tri[[3]],Tri[[6]]}],
                        {{1, 1, 1}}];
                (* Check signs of betas *)
                Cond1 = {Betas[[1, 1]]>0, Betas[[2, 1]]>0,
                                                Betas[[3, 1]]<0};
                Cond2 = {Betas[[1, 2]]>0, Betas[[3, 2]]>0,
                                                Betas[[2, 2]]<0};
                Cond3 = {Betas[[2, 3]]>0, Betas[[3, 3]]>0,
                                                Betas[[1, 3]]<0};
        Return[{Cond1, Cond2, Cond3}];
        )];

(* TestInj is a matrix of 8 3x3 logical expressions that must reduce
   to "True" if injectivity holds, a matrix per each triangle *)
TestInj = {}
For[i=1, i<=8, i++,
```

```
        TestInj = Join[TestInj, ChkInjective[Tri[i]]];];
Print["For Injectivity"];
Print["Need only verify that all inequalities below are valid"];
Print["which is immediate once we use the fact that Cos[2 Pi/n]"];
Print["has a magnitude less than 1 for all n>=3"];
Print[Flatten[Simplify[TestInj]]]
(* By inspection and using the fact that Cos[2 Pi/n] has a magnitude
   less than 1 for all n>=3 *)

(* Now for the proof of regularity. To do that we compute the
   determinant of the cross product of the partial derivatives over
    each triangle, then prove that it does not vanish on the
   interior of any triangle.
   Parameterization is irrelevant so I'll use the canonical Bezier
   parameterization. *)
ChkRegular[Tri_]:= Module[{Barys, Basis, i},
                (Basis = {(1-s-t)^2, 2t(1-s-t), t^2, 2s(1-s-t),
                                      2 s t, s^2};
                 Dss = Simplify[D[Basis, s]];
                 Dtt = Simplify[D[Basis, t]];
                 Return[Det[Jacob[Tri]]];)
        ];
Jacob[BezPoints_] := {{Transpose[BezPoints][[1]] . Dss,
                        Transpose[BezPoints][[2]] . Dss},
 {Transpose[BezPoints][[1]] . Dtt,
                        Transpose[BezPoints][[2]] . Dtt}}
```

```
TestReg = Table[ChkRegular[Tri[i]], {i, 8}];
Print["For Regularity"];
Print["Need only verify that all expressions below do not vanish on"];
Print["standard Bezier triangle. This follows from the fact that"];
Print["0 <= s, t, 1-s-t <=1"];
Print[Simplify[TestReg/(16 Sin[2 Pi/n])]];
(* By inspection and using the fact that Cos[2 Pi/n] has a magnitude
   less than 1 for all n>=3 and that s+t <=1 over the canonical
   triangle, the result follows. *)
```

Chapter 8

Edge and Vertex Insertion on a Class of C^1 Subdivision Surfaces

In this chapter we develop an algorithm that introduces (sharp) edges, or vertices, on smooth subdivision surfaces built using the techniques of Chapter 7. We also extend our subdivision scheme to handle open polyhedra so that surfaces with boundaries can also be modeled. By making sure that the spaces of polyhedra obtained after edge insertion contain those before the insertion we shall be able to express smooth C^1–limit surfaces as many patches that meet smoothly. The new subdivision rules we construct, in Section 8.1, handle boundaries such that boundary curves depend only on boundary control points. The benefit of this property is twofold. First, sharp edges can be introduced on a smooth surface by treating these edges as boundaries. Second, we obtain an alternative way to model surfaces of arbitrary genus, by decomposing these surfaces into patches that join smoothly.

This decomposition property distinguishes this work from the work of Hoppe et. al. [38] where a smooth surface must be perturbed to allow for its representation as multiple smooth patches.

The first section of this chapter addresses the problem of inserting edges, possibly sharp, on the smooth subdivision surfaces of Chapter 7. Then, in Section 8.2, we address vertex insertion, followed by some examples.

8.1 Edge Insertion

Using the technique of the last chapter we can build closed smooth surfaces that have tangent plane continuity. In reality however we would also like to model objects with boundaries, with creases or with sharp vertices. In Section 8.1.1 we discuss our overall strategy for edge insertion, and define precisely what we mean by edge insertion, then we give the actual construction in Section 8.1.2.

8.1.1 Strategy

To insert sharp edges and vertices on the limit surface, we take an approach similar to Hoppe et. al. [38], using a tagging scheme to mark vertices and edges of the initial control polyhedron that we want sharpened and then applying different subdivision rules depending on the tags. The essence of the solution is the computation of these different rules.

Hoppe *et. al.* also describe a scheme for building subdivision surfaces with creases and sharp vertices, in a primal triangular subdivision setting. Our approach differs from theirs in that edge insertion is treated as a decomposition problem. Thus our objective is not only to create surfaces with sharp edges, but also smooth surfaces that can split into patches that meet smoothly. For manufacturing purposes, this is an important property because a smooth surface may need to be built from many patches that are manufactured separately, then assembled to create the final surface. This assembly cannot be done using the scheme of Hoppe et. al. since their subdivision rules on the sides of an edge do not reproduce the exact subdivision rules used on the interior.

Our technique for inserting sharp edges on the limit surface is a two step process. In the first step, we insert the edge as a smooth edge, breaking the surface into two patches that meet smoothly, and introducing new control points that entirely control the edge curve. In the second step we move these new control points to make the edge into a sharp edge if desired, or just break the the surface across this edge into patches. Clearly we only need to show how to perform the first step that splits the surface since edge control points completely specify edge curve.

8.1.2 Rules for Edge Insertion

Starting with a subdivision surface built using the interior subdivision rules of Section 7.3, we now show how to insert a smooth edge on this surface. Initially we shall not worry about what happens at the end points of the inserted edge (this problem will be dealt with along with vertex insertion in Section 8.2); thus we only study the case of inserting an edge that goes across a four–sided face.

We formulate the problem as knot–line insertion in the parameter space (actually doubling a knot–line). For our specific box spline, it can be shown that curves on the limit surface that are maps of iso–parametric grid lines are quadratic B–splines with uniform knots. It is known, from the theory of B–splines, that doubling a knot on a quadratic B–spline with uniform knots corresponds to inserting a new control point half–way between two old control points as shown in Figure 8.1. After doubling the knot, the curve and its control polygon can both be broken into two pieces across this new control point. The new control point may also be moved to introduce a kink on the curve.

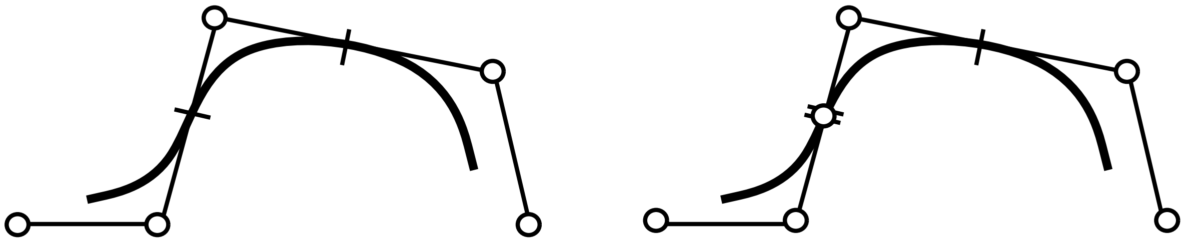


Figure 8.1: Doubling a knot on a quadratic B–splines with uniform knots. Marks on the curve denote knot values, and white circles denote control points.

Now we return to our box spline and study doubling the horizontal knot–line in the middle between the upper two and the lower two control points in Figure 8.2.a. Inspecting the support of a basis function of this box spline (shown in Figure 7.3), we notice that the only basis functions with support overlapping the knot–line being doubled are those corresponding to the four control points (1)–(4) in Figure 8.2.a. Since the images of the two vertical polyhedral edges shown are quadratic B–splines with uniform knots, doubling the middle knot–line creates two new control points (5), (6) half–way between the upper two and the lower two control points as shown in Figure 8.2.b.

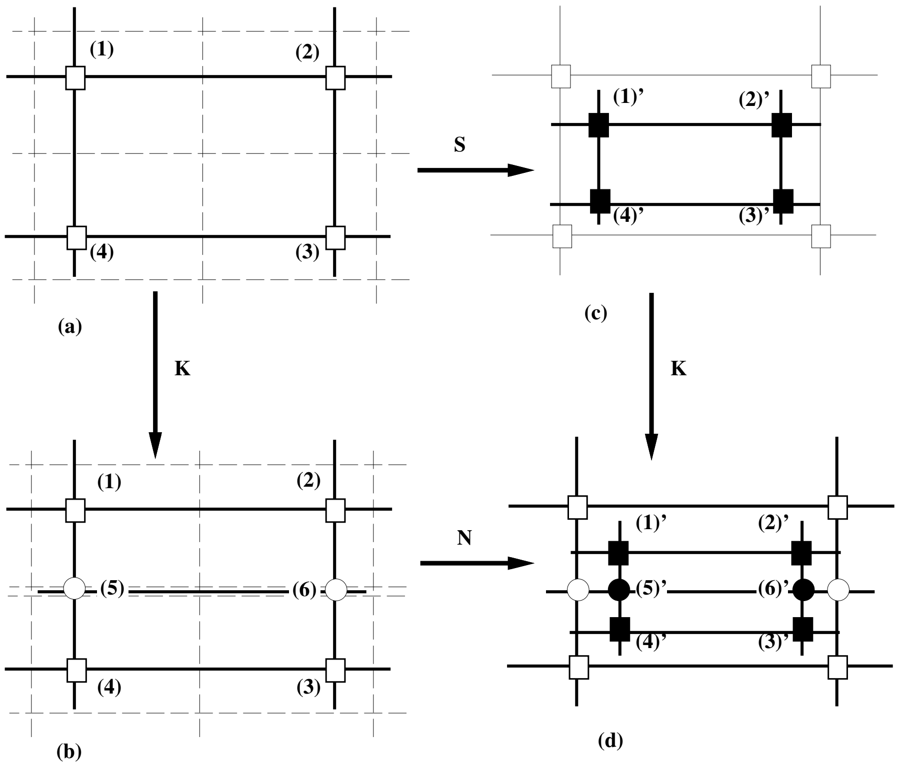


Figure 8.2: Commutativity of knot–line insertion and subdivision. Control points before (after) subdivision are shown in white (black). Solid lines denote polyhedral edges and dotted lines denote knot–lines.

Now, if we can perform knot–line insertion after any number of subdivision steps, we must be able to reproduce the limit surface as two patches, since we might as well perform the edge insertion on the limit surface itself. In other words, the commutativity of subdivision and knot–line insertion guarantees that the final surface will be reproducible from two pieces. This commutativity is the main tool that we use to derive the subdivision rules along and next to edges. Figure 8.2 demonstrates this commutativity diagram. Control points on the left (right) are before (after) subdivision. Those on the top (bottom) are before (after) knot–line doubling.

Formally, let K be the 6×4 knot–line insertion matrix that takes control points (1)–(4) of Figure 8.2.a to control points (1)–(6) in Figure 8.2.b. Then, using the numbering shown in figure for the control points we can write:

$$K = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1/2 & 0 & 0 & 1/2 \\ 0 & 1/2 & 1/2 & 0 \end{bmatrix}. \tag{8.1}$$

The same matrix is used to insert the knot–line after subdivision, and the regular box spline subdivision matrix S given in (7.4) takes control points (1)–(4) in Figure 8.2.a to the four new control points (1)'–(4)' in Figure 8.2.c.

What we are looking for is a 6×6 matrix N that takes the six control points (1)–(6) in Figure 8.2.b to the corresponding control points (1)'–(6)' in Figure 8.2.d. This matrix N describes how control points subdivide on and close to edges, that is how to perform subdivision in the presence of a non-uniformity.

By the commutativity of knot–line insertion and subdivision, the matrix N must satisfy:

$$N \cdot K = K \cdot S. \tag{8.2}$$

Assuming (the natural) symmetry between patches on the two sides of the inserted edge (in Figure 8.2), we can write N as a matrix of unknowns:

$$N = \begin{bmatrix} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ b_2 & b_1 & b_4 & b_3 & b_6 & b_5 \\ b_4 & b_3 & b_1 & b_2 & b_5 & b_6 \\ b_3 & b_4 & b_2 & b_1 & b_6 & b_5 \\ c_1 & c_2 & c_2 & c_1 & c_3 & c_4 \\ c_2 & c_1 & c_1 & c_2 & c_4 & c_3 \end{bmatrix}$$

for some $b_i, i = 1, \ldots, 6$, $c_j, j = 1, \ldots, 4$.

Now solving (8.2), we have many choices for the entries of N. This freedom is utilized to select values, for the entries, that satisfy some desirable properties:

1. Separability: Since we want to create a surface that can split across the inserted edge, new control points on one side of this edge must not depend on old control points on the other side of the edge. That is, the computation of (1)'–(2)' (in Figure 8.2) must not involve control points (3)–(4), similarly for (3)'–(4)' and (1)–(2).

   To satisfy this condition, we must pick $b_3 = b_4 = 0$. Now, substituting in (8.2) and solving we get:

   $$b_1 = 1/4, b_2 = 1/4, b_5 = 1/2, b_6 = 0.$$

2. Edge Control: As described in Section 8.1.1, our strategy is to decompose the edge insertion problem into two steps. First, a smooth edge is inserted, introducing new control points that we call edge control points. In the second step

control points are moved to sharpen the inserted edge. To decouple the two steps, the edge curve must depend entirely on edge control points. If this can be achieved, then creating C^0 surfaces will be easier since we need only move edge control points and we are guaranteed that no gaps are introduced across the inserted edge.

In terms of the entries of N this translates to requiring that new edge control points (5)'–(6)' depend only on old edge control points (5)–(6).

This implies that

$$c_1 = c_2 = 0, c_3 = 3/4, c_4 = 1/4.$$

Substituting the values of $b_i, i = 1, \ldots, 6$, and $c_j, j = 1, \ldots, 4$ we get:

$$N = \begin{bmatrix} 1/4 & 1/4 & 0 & 0 & 1/2 & 0 \\ 1/4 & 1/4 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 1/4 & 1/4 & 1/2 & 0 \\ 0 & 0 & 1/4 & 1/4 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 3/4 & 1/4 \\ 0 & 0 & 0 & 0 & 1/4 & 3/4 \end{bmatrix}. \tag{8.3}$$

Using N as a subdivision matrix creates new control points that control an edge on the surface. By moving these new control points we can make this edge into a sharp edge on the surface or split the surface across the edge. In the next subsection we discuss the properties of this edge insertion algorithm.

8.1.3 Properties of Edge Insertion Algorithm

The only other edge and vertex insertion algorithm on non-tensor-product subdivision surfaces is the algorithm given by Hoppe *et. al* [38]. One of the important properties of the algorithm presented in Section 8.1.2, that distinguishes it from Hoppe's algorithm, is that the algorithm we propose enjoys the *space containment* property, while Hoppe's does not.

In the following section we explain the meaning and implications of this property and prove that our edge insertion algorithm indeed achieves space containment. Other desirable properties and differences from Hoppe's scheme are also discussed.

Space Containment

One of the fundamental and appealing properties of B–splines is that of *space containment*. In the univariate case, a vector T of nondecreasing knot values defines a linear space of B–splines. The space containment property asserts that for any other vector T_1 of nondecreasing knot values such that $T \subseteq T_1$, any spline curve on the knot vector T can be represented exactly in the space of splines specified by the knot vector T_1. That is, the space of splines defined on the finer knot vector T_1 contains the space of all splines defined on T.

Space containment is an important property of B–splines because efficient algorithms are available for the manipulation of B–splines with special knot configurations. By knot insertion, B–spline curves can be converted into forms that are easier to deal with (for example Bézier form for rendering, or for splitting splines into pieces at the knots). In other contexts, this space containment property is used to perform multiresolution analysis by representing objects in different levels of details.

To prove space containment for our edge insertion scheme, we need to build a parameterization for each surface. Then, we need to show how to express (basis) functions on the coarse parameterization in terms of basis functions on the finer parameterization, after the discontinuity is introduced. In Theorem 8.1 below, we prove that if the edge insertion matrix (K), interior subdivision matrix (S), and the nonuniform subdivision matrix (N) are chosen to satisfy (8.2), in addition to some other simple intuitive conditions, then space containment is attained. The proof is outlined in a more general setting than our particular subdivision scheme. Our scheme will be used as an example for illustration.

Before going on to introduce the proof, and for notational convenience, we extend our definitions of the interior subdivision matrix S, the nonuniform subdivision matrix N and the edge insertion matrix, K, (as defined in (7.7), (8.3), and (8.1) respectively for our particular scheme) to operate on full polyhedra instead of individual (quadrilateral) faces. This is done by stringing the local matrices together, since all the basis functions we use have finite supports. Thus we shall use $\mathcal{S}$ to denote the global subdivision matrix (that employs the interior subdivision rules of Section 7.3 in our case). Also $\mathcal{N}$ extends N by behaving identically as $\mathcal{S}$ everywhere except on (quadrilateral) faces with and close to inserted edges, where it behaves as N.

The matrix $\mathcal{K}$ is a global edge insertion matrix. It leaves all the control points of the polyhedron unchanged, but adds new control points as specified locally by K along the inserted edge. We make the intuitive assumption that $\mathcal{K}$ computes new edge vertices by taking convex combinations of control points within a strip of width w of the control points around the inserted edge. In our particular example $w = 2$ since edge control points are computed by averaging two control points as given by (8.1).

From the above definitions along with (8.2) we get:

$$\mathcal{N} \cdot \mathcal{K} = \mathcal{K} \cdot \mathcal{S}.$$

Now, following an approach similar to [18], we can construct a parameterization for the surfaces built by $\mathcal{S}$–subdivision, over the initial polyhedron M^0. Employing this approach, we can parameterize the limit surfaces obtained by $\mathcal{S}$–subdivision using parametric basis functions, $\phi(x)$, as $\phi(x)\mathbf{p}^0$, where $x \in M^0$, and $\mathbf{p}^0$ is the vector of control points of the initial polyhedron.

Similarly we can express the limit surfaces obtained by nonuniform subdivision using $\mathcal{N}$ in terms of other parametric basis functions $\hat{\phi}(x)$, as $\hat{\phi}(x)\hat{\mathbf{p}}^0$, where $x \in M^0$ and $\hat{\mathbf{p}}^0$ is the vector of control points obtained by inserting an edge on M^0. Observe that, by construction,

$$\hat{\mathbf{p}}^0 = \mathcal{K}\mathbf{p}^0.$$

Now we can prove the main result of this section.

Theorem 8.1 Let $\mathcal{S}$ be the interior subdivision matrix for a subdivision scheme. Assume $\mathcal{K}$ is an edge insertion matrix for this scheme that computes new edge control points by taking convex combinations on a w–wide strip of control points. Assume further that $\mathcal{N}$ is a nonuniform subdivision matrix, chosen to satisfy: $\mathcal{N} \cdot \mathcal{K} = \mathcal{K} \cdot \mathcal{S}$. Then,

$$\hat{\phi}(x)\mathcal{K} = \phi(x) \quad x \in M^0.$$

Here $\phi(x)$, and $\hat{\phi}(x)$ are the vectors of parametric basis functions associated with the initial set of control points before, and after edge insertion respectively.

Proof: For a fixed initial vector of control points $\mathbf{p}^0$, we define a sequence of functions $H_i(x)$, associated with subdivision using $\mathcal{S}$ for i rounds. These functions can be expressed as

$$H_i(x) = \phi_i(x)\mathbf{p}^0 \qquad x \in M^0 \tag{8.4}$$

where $\phi_i(x)$ is the vector of basis functions associated with $\mathbf{p}^0$ after i rounds of subdivision using $\mathcal{S}$. Also define $H(x) = \lim_{i\mapsto\infty} H_i(x)$, and $\phi(x) = \lim_{i\mapsto\infty} \phi_i(x)$. The convergence of subdivision using $\mathcal{S}$ guarantees the existence of these limits. We define $\hat{H}_i(x), \hat{H}(x)$ and $\hat{\phi}(x)$ analogously for nonuniform subdivision using $\mathcal{N}$; thus

$$\hat{H}_i(x) = \hat{\phi}_i(x)\hat{\mathbf{p}}^0 \qquad x \in M^0. \tag{8.5}$$

These limits also exist by the convergence of $\mathcal{N}$–subdivision. Notice the correspondence, as well as the distinction, between vectors of control points (for example $\mathbf{p}^0, \mathcal{S}^j\mathbf{p}^0$) and the parametric functions defined on the polyhedra specified by these control points ($H_0(x), H_j(x)$ respectively).

Since $\mathcal{N} \cdot \mathcal{K} = \mathcal{K} \cdot \mathcal{S} \Rightarrow \mathcal{N}^i \cdot \mathcal{K} = \mathcal{K} \cdot \mathcal{S}^i$, we have:

$$\begin{aligned} \mathcal{K} \cdot \mathcal{S}^i\mathbf{p}^0 &= \mathcal{N}^i \cdot \mathcal{K}\mathbf{p}^0 \\ &= \mathcal{N}^i\hat{\mathbf{p}}^0 \end{aligned}$$

The set of control points on the right hand side correspond to $\hat{H}_i(x)$. On the left hand side we have $\mathcal{K} \cdot \mathcal{S}^i\mathbf{p}^0$. Since the new points introduced by $\mathcal{K}$ are convex combinations of control points within a strip of width w of control points in $\mathcal{S}^i\mathbf{p}^0$, and subdivision is a self–similar refinement process, this strip shrinks in width with i as shown in Figure 8.3. It follows that for points not on the inserted edge, though arbitrarily close, we can always find j such that $H_j(x) = \hat{H}_j(x)$. Agreement on the edge itself follows form the continuity of $H(x)$.

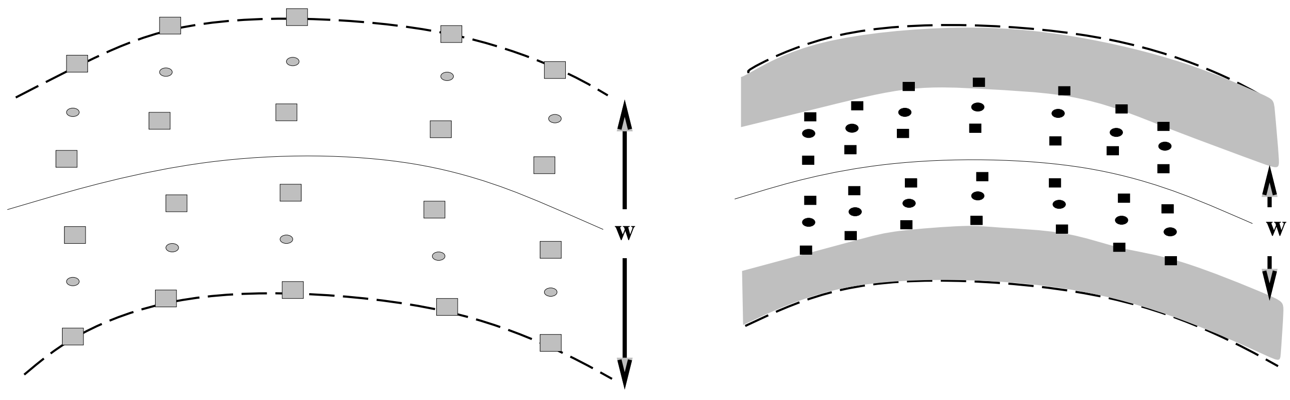


Figure 8.3: Control points used for edge insertion before (left) and after (right) one round of subdivision.

Thus we conclude that $\lim_{i\to\infty} |H_i(x) - \hat{H}_i(x)| \to 0$. It follows that $\lim_{i\to\infty} |H(x) - \hat{H}(x)| \to 0$, and thus $H(x) = \hat{H}(x)$.

By setting the vectors of initial control points equal to $\delta = \{\cdots, 0, 1, 0, \cdots\}$, we get a representation of the basis functions $\phi(x)$ in terms of $\hat{\phi}(x)$, and thus we get space containment. $\Box$

Other Desirable Properties

In addition to the space containment property of the last section, our edge insertion algorithm enjoys the following important properties:

Boundary management: By construction, (the separability condition of Section 8.1.2), the computations of new control points on one side of an inserted edge do not depend on the control points on the other side of this edge. Actually, nothing in the construction necessitates the existence of control points on both sides of the edge. Thus, the boundary of the control polyhedron can be treated exactly as an inserted edge. (Corner control points still need to be addressed. They will be discussed along with vertex insertion in Section 8.2).

Edge curve is a B–spline: The rules computed in Section 8.1.2 for subdivision along an inserted edge, as given by the last two rows of N in (8.3), are exactly the rules used by Chaikin's algorithm to compute a quadratic B–spline with uniform knots. This is another important property because there are available techniques (e.g. Blossoming [29]) for computing the control points of a B–spline given other representations of the curve. It also allows for the use of the full arsenal of B–spline design tools to design the shapes of edges and boundaries.

We remark that our subdivision scheme produces C^1 piecewise quadratic surfaces on regular connectivity regions of the polyhedron, compared to the piecewise C^2–quartic surfaces produced by Hoppe *et. al.* In fact, this is the main reason we have chosen our case study box spline subdivision scheme, since it produces surfaces with the lowest possible polynomial degree for a C^1 scheme.

8.1.4 Examples

Figure 8.4 shows some surfaces with sharp edges and with boundaries.

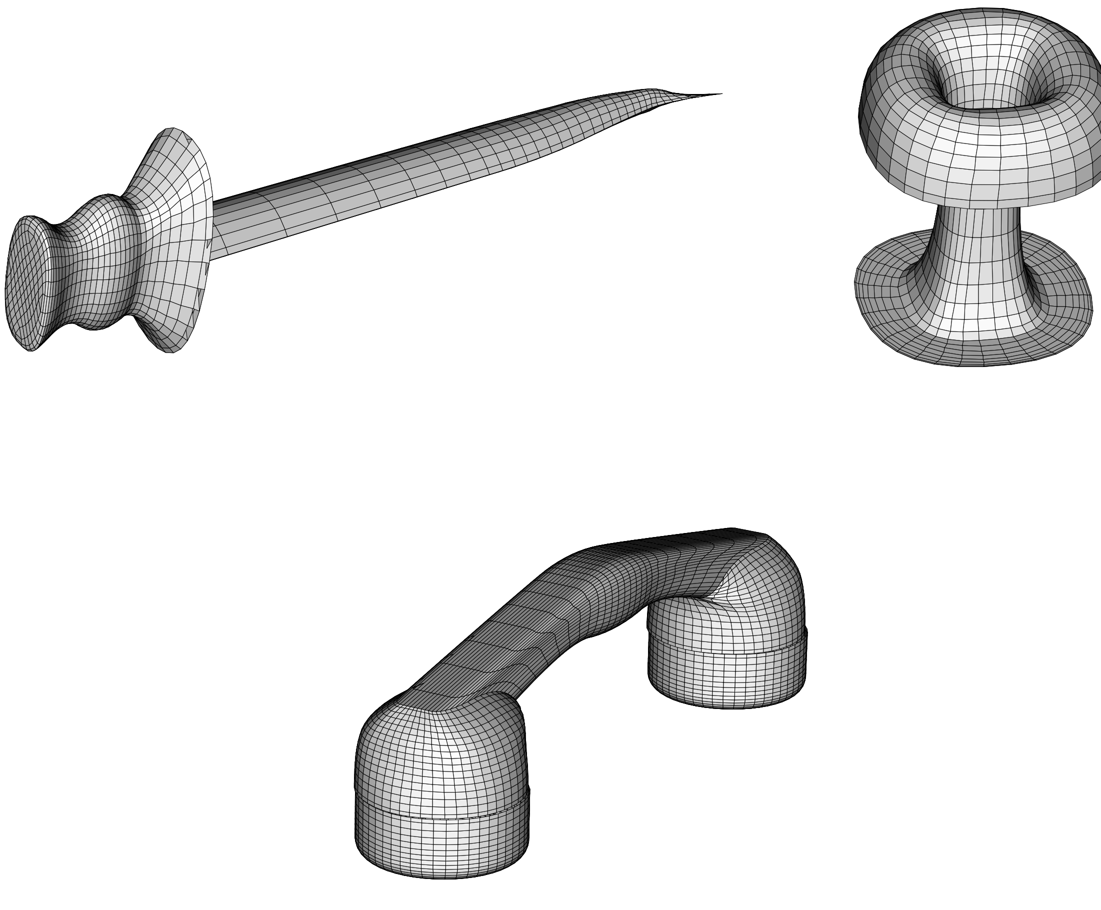


Figure 8.4: Surfaces with (sharp) edges and boundaries

8.2 Vertex Insertion

The techniques of Section 8.1 enable us to insert (possibly sharp) edges across four sided polyhedral faces. Realistically the requirement that faces be four sided is not a severe restriction to impose since, as shown in Figure 7.5, after one round of subdivision all faces become four–sided except for a few extraordinary faces. Also, after more and more subdivision, larger and larger regions of these extraordinary faces are covered with four–sided faces. Thus if we want to insert an edge that goes through but not to the centroid of an extraordinary face, we can use the already developed technique, though after some number of subdivision steps. The edges we can insert, however, are only maps of iso–parametric grid lines.

In this section we address the problem of how to insert a (possibly sharp) vertex at the centroid of an extraordinary face, and also the important question of what to do at the end points of an inserted edge, or if the edge terminates at the centroid of an extraordinary face.

Returning to our objective of building surfaces that can split/decompose, vertex insertion means:

1. Giving a scheme for breaking up a polyhedral surface, around the centroid of an arbitrary face, into pieces that can be subdivided separately then assembled to regenerate the surface obtained by subdividing the whole polyhedron using the interior subdivision rules of Chapter 7.

2. Specifying user control over the individual polyhedral pieces so that the inserted vertex can be made into a sharp vertex.

We begin by giving the construction for the insertion of a vertex of valence four at the centroid of a four–sided face in Section 8.2.1, then address general valence vertices in Section 8.2.2.

8.2.1 Vertices of Valence Four

As with edge insertion, the main idea here is the commutativity of vertex insertion and subdivision. Using the indexing of control points shown in Figure 8.5, we can write the vertex insertion matrix K_4 and the nonuniform subdivision matrix N_4 (assuming natural symmetry) as:

$$K_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 1/2 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{bmatrix}, \qquad N_4 = \begin{bmatrix} b_1 & b_2 & b_3 & b_2 & b_4 & b_5 & b_5 & b_4 & b_6 \\ b_2 & b_1 & b_2 & b_3 & b_4 & b_4 & b_5 & b_5 & b_6 \\ b_3 & b_2 & b_1 & b_2 & b_5 & b_4 & b_4 & b_5 & b_6 \\ b_2 & b_3 & b_2 & b_1 & b_5 & b_5 & b_4 & b_4 & b_6 \\ c_1 & c_1 & c_2 & c_2 & c_3 & c_4 & c_5 & c_4 & c_6 \\ c_2 & c_1 & c_1 & c_2 & c_4 & c_3 & c_4 & c_5 & c_6 \\ c_2 & c_2 & c_1 & c_1 & c_5 & c_4 & c_3 & c_4 & c_6 \\ c_1 & c_2 & c_2 & c_1 & c_4 & c_5 & c_4 & c_3 & c_6 \\ d_1 & d_1 & d_1 & d_1 & d_2 & d_2 & d_2 & d_2 & d_3 \end{bmatrix}.$$

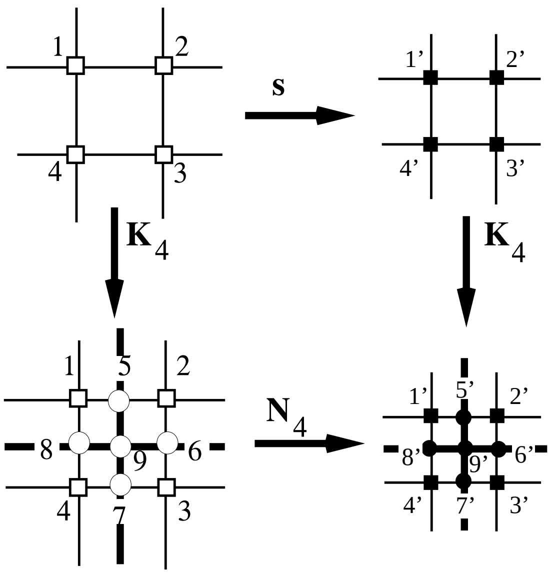


Figure 8.5: Vertex insertion, valence 4

The entries of N_4 are computed from the commutativity diagram (Figure 8.5) by solving the system of equations:

$$N_4 \cdot K_4 = K_4 \cdot S, \tag{8.6}$$

where S is the four–sided face subdivision matrix of the regular box spline (7.4). The solution must satisfy the following properties:

1. Separability: The surface can be broken into four patches around the inserted vertex. Thus, new control points in each patch must depend only on old control points in the same patch. In terms of the entries of N_4 this implies that:

$$b_2 = b_3 = b_5 = 0 \quad \text{and} \quad c_2 = c_5 = 0.$$

2. Interpolation of inserted vertex: Since the surface before vertex insertion does not have any holes, all four patches must interpolate the inserted vertex. One subdivision rule that guarantees this property is obtained by picking $d_1 = d_2 = 0$ and $d_3 = 1$.

3. Edge curves in each patch are controlled entirely by control points on the edge. This condition implies that $c_1 = 0$.

4. Now solving (8.6) for the remaining entries of N_4 we get:

$$b_1 = b_6 = 0, \quad b_4 = 1/2, \quad c_3 = c_6 = 1/2, \quad c_4 = 0.$$

Thus,

$$N_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8.7}$$

The matrix N_4 in (8.7) solves the first part of the vertex insertion problem: how to break the surface into four patches around a valence four vertex and how to subdivide these patches independently to get a smooth surface identical to the surface obtained by subdividing the whole control polyhedron using the box spline rules. The second part, concerning changing this inserted vertex into a sharp vertex, is easy since all four patches interpolate the inserted vertex. To maintain C^0 continuity of the composite surface, however, control points common between patches (e.g. control points (5)–(8) in Figure 8.5) need to be moved consistently in neighboring patches.

Two observations on the matrix N_4 in (8.7) are in order:

- Extending our approach in Section 8.1 of treating boundaries as inserted edges, it is easy to see that we can use the subdivision matrix N, for subdivision at and close to corners by treating a corner control point as an inserted vertex of valence four.

- As shown in Section 8.1, inserted edges are quadratic B–splines with uniform knots. Using the subdivision matrix N_4 in (8.7) at the end point of an edge corresponds to doubling the knot at the end of the corresponding B–spline curve.

8.2.2 Arbitrary Valence Vertices

Ideally we would like to address the vertex insertion problem uniformly, independent of the valence. To do that we need to draw a commutativity diagram similar to Figure 8.5 and use the corresponding system of equations to derive the special subdivision rules for vertex insertion. However, some problem arises in using this technique when the valence is not 4. The next subsection explains this problem, our approach to getting around it and the actual construction.

The Decomposition Problem

To illustrate the problem with extending the valence 4 approach to other valences, we examine the problem of inserting a vertex at the centroid of the pentagonal face shown in Figure 8.6.

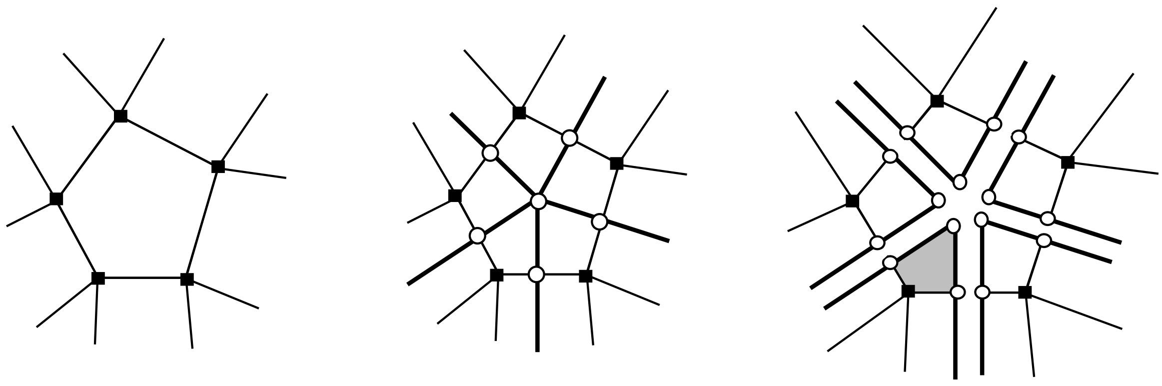


Figure 8.6: Vertex insertion, valence 5. Before insertion, points in the shaded part of the surface depend on 5 control points (n in general), while they depend on exactly 4 control points after vertex insertion.

Before inserting the vertex at the centroid of the pentagon, points in the shaded area (of Figure 8.6) are computed using the mask in (7.7), thus they belong to a five–dimensional space spanned by all five vertices of the pentagon (marked with black squares in the figure). To split this face into five patches, we must reproduce the same points using only the four control points enclosing the shaded patch. Since points computed from these four points span only a four–dimensional space, this reproduction can not generally be done.

For valence four vertices this problem does not arise because space dimensions match. For the general case, the reason for the above problem (we call it *decomposition problem* henceforth) is that after splitting the surface, the subdivision algorithm operating on individual patches does not have enough information to reproduce the corresponding part of the original surface.

A Solution to the Decomposition Problem

We solve the decomposition problem by passing to the subdivision process, *on each patch*, the data of all the vertices of the extraordinary face. Thus a vertex inserted at the centroid of an extraordinary face is treated as a special vertex. It carries the coordinates of all the vertices of the enclosing face. We call such vertex a *composite vertex.*

The decomposition of the surface for the example in Figure 8.6 can now be thought of as shown in Figure 8.7.

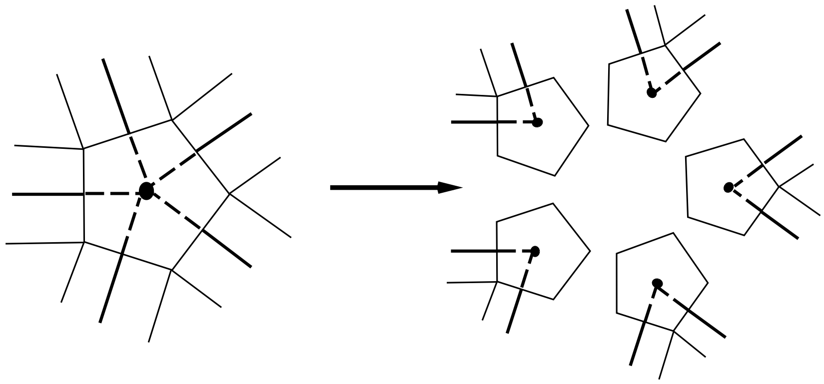


Figure 8.7: Patch decomposition of a valence five face

Subdividing Composite Vertices

If we decompose patches around extraordinary faces using composite vertices, we shall have enough information subdividing each patch to reproduce the corresponding part of the original surface. This solves the break–up part of the vertex insertion problem.

Now, we study how patches with composite vertices at their corners subdivide. We go back to our current example of a five sided face for illustration. One of the five patches that we get is shown in Figure 8.8.

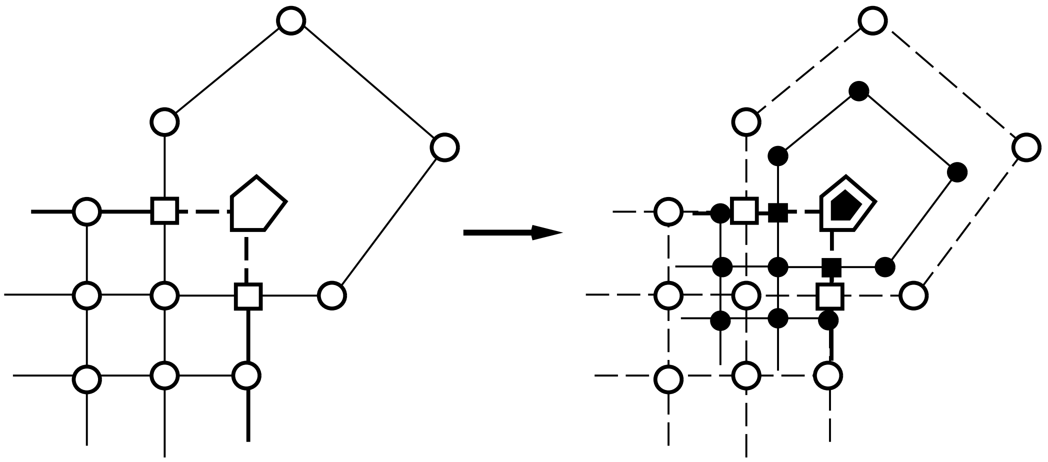


Figure 8.8: One of the five patches obtained by inserting a valence five vertex. The composite vertex at the centroid contains the information of all five vertices of the pentagon

In Figure 8.8, control points marked with circles control the shape of the patch while those marked with squares can be computed from the circled ones. Now the procedure to subdivide a patch with a composite vertex at the corner is as follows:

Algorithm 8.1 Subdivision with Composite Vertices:

1. Subdivide all interior faces using the interior rules in (7.7) and faces on the boundaries, other than those containing composite vertices, using the nonuniform rules in (8.3).
2. Perform interior subdivision, using the rules in (7.7), on the vertices of the composite vertex.
3. Compute two new mid–side control points (those marked with black squares for the valence five example of Figure 8.8) by averaging the control points of the corresponding two subdivided sides.
4. Save the subdivided polygon of the composite vertex as new composite vertex.

This subdivision algorithm (Algorithm 8.1) results in a decomposition of the smooth surface around the centroid of an arbitrary face. The following proposition shows that the surface resulting from the vertex insertion process has tangent plane continuity everywhere on the extraordinary face.

> **Proposition 8.1** Patches meeting with a common composite vertex form a tangent plane continuous limit surface over the common face.

Proof: The result follows directly from the construction since every point is computed using the exact same formulas before or after vertex insertion. □

For the second part of the vertex insertion problem, that is sharpening the inserted vertex, it suffices to move the vertices of the composite vertex differently between patches. This movement must be performed very carefully so that continuity of the surface is not destroyed.

Examples

Figure 8.9.a shows a patch with a valence three vertex inserted at its centroid, while Figure 8.9.b shows a valence five example.

8.3 Implementation

All the example surfaces shown in this part of the thesis were produced by a C–implementation of the subdivision scheme developed in Chapter 7, enhanced with the capability to model open polyhedra, sharp edges and vertices. The existence of a closed form formula for the subdivision mask, for arbitrary number of faces, makes the implementation uniform since the same template of rules is used everywhere on the interior of the polyhedra. This contrasts to tensor–produce based schemes by Catmull–Clark and Doo-Sabin [8, 19] where special treatment is given to quadrangular faces.

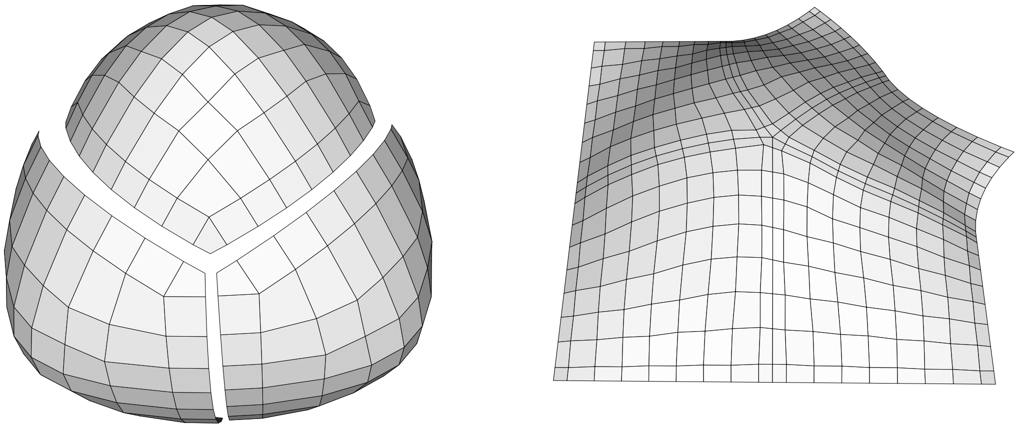


Figure 8.9: Vertex insertion examples.

Sharp edges in Figure 8.4 are produced by treating the control polyhedron as two separate polyhedra. The edge control property of the edge insertion algorithm insures that no gaps are introduced between the patches. The property that edge curves are quadratic B–splines with uniform knots comes in very handy in designing the examples.

Sharp vertices in Figure 8.4 are introduced by partitioning the surfaces into patches. The examples in Figure 8.9, however, are produced using the "composite vertices" idea of Section 8.2.2.

8.4 Summary of Contributions

In this part of the thesis we propose a technique for edge and vertex insertion on a class of C^1 subdivision surfaces. The approach uses the commutativity of knot insertion and subdivision to derive subdivision rules along and close to edges and vertices. The following is a list of the main results:

- The proof of tangent plane continuity of surfaces constructed using the interior subdivision rules of Chapter 7 is a significant contribution in itself. No such proof is available in literature or known to be available for a non–tensor–product quadrangle–based subdivision scheme, although the tools used for the proof are developed by other researchers [50].

- Using commutativity of edge insertion and subdivision to compute the non–uniform subdivision rules (on and next to inserted edges and vertices) is the main contribution of the edge insertion algorithm. The idea in itself can be applied to other subdivision schemes, although the details need to be considered on a case by case basis. In the next section we discuss the issues involved in such extension.

- Subspace Containment: The main advantage of using commutativity, to develop non–uniform subdivision rules, is that surfaces built using this approach can be split exactly into pieces across edges or centroids of arbitrary faces.

8.5 Future Work

Some questions are still left to be answered. Below, we give a list of these questions that we would like to investigate.

- The interior subdivision rules of Chapter 7 build provably tangent plane continuous surfaces over polyhedra of arbitrary topological type, however the large mask support causes the decomposition problem of Section 8.2.2. We would like to investigate if there exist other subdivision rules that also build tangent plane continuous limit surfaces but allow for the complete decomposition of the surface around extraordinary vertices without the need for the composite vertices machinery.

- Other subdivision schemes can make use of the methodology developed in this thesis for edge and vertex insertion. Here we list a set of factors that need to be considered when extending the approach to other subdivision schemes:
  - A major factor in deciding the difficulty of the extension is the (size of the) support of basis functions. For the case–study box spline only two layers of control points entirely control the surface midway between them, generally this may be bigger and more new subdivision rules may need to be developed.
  - The Edge control property followed from the fact that we were doubling a knot on a quadratic B–spline. For other subdivision schemes the inserted edges need not be quadratic B–splines, and the effect of the edge insertion in the parameter domain needs to be studied.

Bibliography

[1] P. J. Barry, R. N. Goldman, and C. A. Micchelli. Knot insertion algorithms for piecewise polynomial spaces determined by connection matrices. *Advances in Computational Mathematics*, 1:139–171, 1993.

[2] B. A. Barsky and J. C. Beatty. Varying the betas in beta–splines. Technical Report UCB/CSD 83/112, University of California, Berkeley, Department of Electrical Engineering and Computer Science, 1983.

[3] B. A. Barsky and T. D. Rose. Geometric continuity of parametric curves. Technical Report UCB/CSD 84/205, University of California, Berkeley, Department of Electrical Engineering and Computer Science, 1984.

[4] B. A. Barsky and T. D. Rose. Geometric continuity of parametric curves:Three equivalent characterizations. *IEEE Computer Graphics and Applications*, 10:60–68, 1989.

[5] B.A. Barsky. *The Beta-Spline: A Local Representation Based on Shape Parameters and Fundamental Geometric Measures.* PhD thesis, University of Utah, 1981.

[6] R. Bartels, J. Beatty, and B. A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann Publishers, Inc., 1987.

[7] W. Boehm. Curvature continuous curves and surfaces. *Computer Aided Geometric Design*, 2(4):313–323, 1985.

[8] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-aided Design*, 10:350–355, 1978.

[9] Cauchy. *Journal de l'Ec Polyt.*, x:29–112, 1812.

[10] G. Chaikin. An algorithm for high speed curve generation. *Computer Graphics and Image Processing*, 3:346–349, 1974.

[11] K. C. Chung and T. H. Yao. On lattices admitting unique Lagrange interpolations. *SIAM Journal of Numerical Analysis*, 14:735–743, 1977.

[12] P. G. Ciarlet and P. A. Raviart. General Lagrange and Hermite interpolation in R^n with applications to finite element methods. *Arch. Rational Mech. Anal.*, 46:177–199, 1972.

[13] P. Davis. *Circulant Matrices.* Wiley Interscience, New York, 1979.

[14] C. de Boor. On calculating with B–splines. *Journal of Approximation Theory*, 6:50–62, 1972.

[15] C. de Boor, K. Höllig, and S. Riemenschneider. *Box Splines.* Springer–Verlag, 1993.

[16] C. de Boor and A. Ron. Computational aspects of polynomial interpolation in several variables. *Mathematics of Computation*, 58:705–727, 1992.

[17] T. DeRose. *Geometric Continuity: A Parameterization Independent Measure of Continuity for Computer Aided Geometric Design.* PhD thesis, University of California at Berkeley, Department of Electrical Engineering and Computer Science, 1985.

[18] T. DeRose, M. Lounsbery, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. Technical Report 93-10-05, University of Washington, Department of Computer Science and Engineering, 1993.

[19] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-aided Design*, 10:356–360, 1978.

[20] N. Dyn, A. Edelmann, and C. A. Micchelli. On locally supported basis functions for the representation of geometrically continuous curves. *Analysis*, 7:313–341, 1987.

[21] N. Dyn and D. Levin. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9:160–169, 1990.

[22] N. Dyn and C. A. Micchelli. Piecewise polynomial spaces and geometric continuity of curves. *Numer. Math.*, 54:319–337, 1988.

[23] G. Farin. Some remarks on γ^2–splines. *Computer Aided Geometric Design*, 2(4):325–328, 1985.

[24] G. Farin. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design*, 3(2):83–128, 1986.

[25] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press Inc., New York, 1988.

[26] M. Gasca and E. Lebron. On Aitken–Neville fomulae for multivariate interpolation. In E. L. Ortiz, editor, *Numerical Approximation of Partial Deifferential Equations*, pages 133–140. Elsevier Science Publishers, 1987.

[27] M. Gasca and J. I. Maeztu. On Lagrange and Hermite interpolation in R^k. *Numer. Math.*, 39:1–14, 1982.

[28] M. Gasca and J. J. Martinez. On the solvability of bivariate Hermite–Birkhoff interpolation problems. *Journal of Computational and Applied Mathematics*, 32:77–82, 1990.

[29] R. Goldman. Blossoming and knot insertion algorithms for B–spline curves. *Computer Aided Geometric Design*, 7:69–81, 1990.

[30] R. N. Goldman and B. A. Barsky. Some basic results on beta–continuous functions and their applications to the construction of geometrically continuous curves and surfaces. In T. Lyche and L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 299–311. Academic Press Inc., 1989.

[31] R. N. Goldman and C. A. Micchelli. Algebraic aspects of geometric continuity. In T. Lyche and L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 313–332. Academic Press Inc., 1989.

[32] T. N. T. Goodman. Properties of β–splines. *Journal of Approximation Theory*, 44:132–153, 1985.

[33] T. N. T. Goodman. Constructing piecewise rational curves with frenet frame continuity. *Computer Aided Geometric Design*, 7:15–31, 1990.

[34] T.N.T. Goodman and K. Unsworth. Injective bivariate maps. Technical Report CS 94/02, University of Dundee, Mathematics and Computer Science, 1994.

[35] J. A. Gregory. Geometric continuity. In T. Lyche and L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 353–371. Academic Press Inc., 1989.

[36] H. Hagen. Geometric spline curves. *Computer Aided Geometric Design*, 2:223–227, 1985.

[37] M. E. Hohmeyer and B. A. Barsky. Rational continuity: Parametric and geometric continuity of rational polynomial curves. *ACM Transactions on Graphics, Special Issue on Computer Aided Geometric Design and Geometric Modeling*, pages 335–359, October 1989.

[38] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface construction. *Computer Graphics*, pages 295–302, 1994.

[39] S. L. Lee and G. M. Phillips. Interpolation on the triangle. *Comm. Appl. Numer. Methods*, 3:271–276, 1987.

[40] S. Lodha and R. Goldman. A unified approach to evaluation algorithms for multivariate polynomials. submitted to Mathematics of Computations, 1995.

[41] C. Loop. Smooth subdivision based on triangles. Master's thesis, University of Utah, 1987.

[42] R. A. Lorentz. *Multivariate Birkhoff Interpolation.* Springer Verlag, 1992.

[43] M. L. Mazure. Geometric contact for curves and surfaces. *Computer Aided Geometric Design*, 11:177–195, 1994.

[44] A. Le Mehaute. *Interpolation et approximation par des fonctions polynomiales par morceaux dans R^n.* PhD thesis, Universite de Rennes, 1984.

[45] A. Le Mehaute. Interpolation d'Hermite iteree. In P. Chenin, C. di Crescenzo, and F. Roberts, editors, *Computers and Computing, Informatique et Calcul.*, pages 77–81. Wiley-Masson, New York, 1986.

[46] G. Muhlbach and M. Gasca. Multivariate polynomial interpolation under projectivities I: Lagrange and Newton interpolation formulas. *Num. Algorithms*, 1:375–400, 1991.

[47] F. J. Narcowitch and J. D. Ward. Generalized Hermite interpolation via matrix–valued conditionally positive definite functions. *Mathematics of Computation*, 63:661–687, 1994.

[48] R. A. Nicolaides. On a class of finite elements generated by Lagrange interpolation. *SIAM Journal of Numerical Analysis*, 9:435–445, 1972.

[49] H. Pottmann. Projectively invariant classes of geometric continuity for CAGD. *Computer Aided Geometric Design*, 6(4):307–321, 1989.

[50] U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12:153–174, 1995.

[51] R.F. Riesenfeld. On Chaikin's algorithm. *Computer Graphics and Image Processing*, 4:304–310, 1975.

[52] L. Schumaker. *Spline Functions: Basic Theory.* Wiley Interscience, New York, 1981.

[53] H. P. Seidel. Polar forms for geometrically continuous spline curves of arbitrary degree. *ACM Transactions on Graphics*, 12:1–34, 1993.

[54] M. Spivak. *Differential Geometry.* Publish or Perish, 1975.

[55] A. Ženišek. A general theorem on triangular C^m elements. *RAIRO, Ser. Rouge, Anal. Numer.*, R–2:119–127, 1974.

[56] J. Warren. Subdivision methods for geometric design. Unpublished Monograph, 1995.