Chapter 4

Bézier Techniques

Dianne Hansford

This chapter introduces the fundamentals of Bézier techniques. As a core tool of 3D Modeling, Bézier techniques provide a geometric-based method for describing and manipulating polynomial curves and surfaces.

4.1. WHY BÉZIER TECHNIQUES?

Bézier techniques bring sophisticated mathematical concepts into a highly geometric and intuitive form. From a practical standpoint, this form facilitates the creative design process. Equally as important, Bézier techniques are an excellent choice in the context of numerical stability of floating point operations.[1] For these reasons, Bézier techniques are at the core of 3D Modeling or Computer Aided Geometric Design (CAGD).

This chapter provides a thorough review of fundamental Bézier techniques. The primary topics being curves, rectangular surfaces, and triangular surfaces. With this knowledge, the reader should be able to access research articles on these topics. Additionally, the study of Bézier techniques is greatly recommended before studying piecewise schemes, B-splines, or other advanced modeling applications.

The notation for this chapter was chosen to make this subject accessible to readers new to 3D modeling, and also to best serve those who use this handbook as a reference. Quite a few 3D Modeling texts (see e.g.,[43,56,91]) have adopted the *blossom*[2] notation, and although important, this notation has a tendency to abstract the geometric concepts, and is therefore not used here.

For in-depth information on Bézier techniques, a textbook is a good place to start. There are many to choose from: [17,43,45,69,80,81,86,91,102]. The origins of Bézier techniques may be found in the History chapter 1 of this handbook. Industrial uses of Bézier

[1] Surprisingly, this result from Farouki and Rajan [49] was not known until many years after Bézier techniques had become popular. See also [28].

[2] The blossom approach is a very powerful tool, theoretically and practically, and it was brought to light by de Casteljau [32] and Ramshaw [88,89]. For a tutorial, see Goldman and DeRose [36].

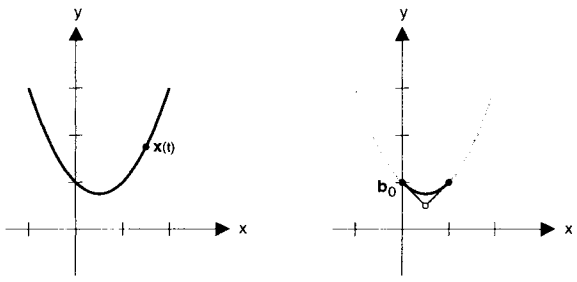

Figure 4.1. Left: The graph of a function. Right: The function as a Bézier curve.

techniques are described by Bézier [7] at Rénault, de Casteljau [33] at Citroën, Farin [40] at Daimler-Benz, and Hochfeld and Ahlers [65] at Volkswagen.

4.2. BÉZIER CURVES

In this section, we introduce the foundations of Bézier techniques via the Bézier curve. In particular, we present the properties and utility of Bézier curves, as well as an important evaluation algorithm: the de Casteljau algorithm. Also, we study the building block of Bézier techniques, the Bernstein polynomials. A thorough understanding of Bézier curves is a good place to start, since nearly all the principles of curves carry over to Bézier surface techniques.

4.2.1. Parametric curves

Curve modeling is primarily concerned with *parametric curves*. The simple quadratic function in the left of Figure 4.1, written as a 2D parametric curve, takes the form

$$\mathbf{x}(t) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} t \\ 1 - t + t^2 \end{bmatrix}, \qquad t \in I\!R. \tag{4.1}$$

Each coordinate is a function of the *parameter* t, and the real line is the *domain* of the curve. Only polynomial coordinate functions will be addressed here. A 3D parametric curve is formed by simply adding a $z(t)$-component.

The boldface notation for points and vectors[3] allows for a concise form for (4.1):

$$\mathbf{x}(t) = \mathbf{a}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2, \tag{4.2}$$

where

$$\mathbf{a}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \mathbf{a}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad \mathbf{a}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The $\mathbf{a}_i$ are called the *coefficients* of the curve and $1, t, t^2$ are the *quadratic monomial basis functions*.

[3]See the Geometric Fundamentals Chapter 2 for an introduction to these geometric entities.

There are many ways to represent a polynomial curve. The monomial form from above is one representation, however, it does not provide the most geometrically intuitive interpretation.[4] A better formulation comes with the Bernstein basis functions as the building block for Bézier curves. Figure 4.1, right, illustrates the curve in quadratic Bézier form. This *quadratic Bézier curve* takes the form

$$\mathbf{x}(t) = \mathbf{b}_0 B_0^2(t) + \mathbf{b}_1 B_1^2(t) + \mathbf{b}_2 B_2^2(t), \tag{4.3}$$

where

$$\mathbf{b}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{b}_1 = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{4.4}$$

are *Bézier control points* of the *Bézier polygon*, and

$$B_0^2(t) = (1-t)^2, \qquad B_1^2(t) = 2t(1-t), \qquad B_0^2(t) = t^2$$

are the *quadratic Bernstein polynomials* or basis functions. The standard procedure is to evaluate Bézier curves for $t \in [0,1]$, although since it is a polynomial, it is defined for all t over the reals. The reason for this will be apparent in Section 4.2.2.

A degree n Bézier curve takes the form

$$\mathbf{x}(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t) \qquad t \in [0,1], \tag{4.5}$$

where

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i, \tag{4.6}$$

are the *degree n Bernstein polynomials*, and the *binomial coefficients* are defined as

$$\binom{n}{i} = \frac{n!}{(n-i)!i!}.$$

Figure 4.2 illustrates several Bézier curves.

4.2.2. Properties of Bézier curves

The following list of properties characterizes Bézier curves. We'll revisit many of these properties in Sections 4.2.3 and 4.2.4. Many of these properties are apparent in Figure 4.2.

- *Endpoint interpolation*: The curve passes through the polygon endpoints: $\mathbf{x}(0) = \mathbf{b}_0$ and $\mathbf{x}(1) = \mathbf{b}_n$.
- *Symmetry*: The two polygons, $\mathbf{b}_0, \ldots, \mathbf{b}_n$ and $\mathbf{b}_n, \ldots, \mathbf{b}_0$, describe the same curve; the only thing that changes is the direction of traversal of the parameter.

[4] See Section 4.2.8 for the geometric interpretation of the monomial coefficients.

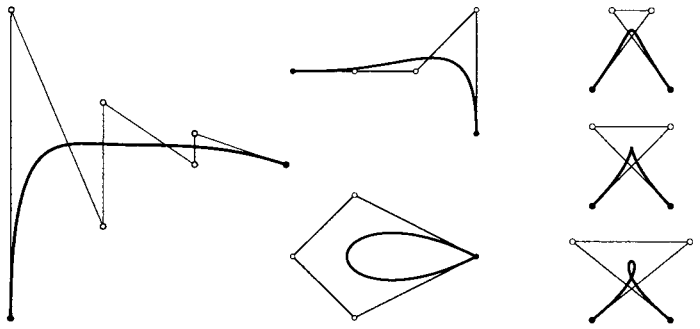

Figure 4.2. These Bézier curves reveal much about the relationship between the polygon and curve.

- *Affine invariance*: If an affine map Φ is applied to the control polygon, then the curve is mapped by the same map. More precisely,

$$\sum_{i=0}^{n} \Phi(\mathbf{b}_i) B_i^n(t) = \Phi(\sum_{i=0}^{n} \mathbf{b}_i B_i^n(t)) \tag{4.7}$$

- *Convex hull*: A point $\mathbf{x}(t)$ on the curve for $t \in [0,1]$ is in the convex hull of the control polygon. See Figure 4.3.

- *Variation diminishing*: If a straight line intersects a planar Bézier polygon m times, then the line can intersect the curve at most m times. In other words, the curve doesn't wiggle more that the polygon. This is evident in the left most curve of Figure 4.2.

- *Linear precision*: If the control points $\mathbf{b}_i$ for $i = 1, \ldots, n-1$ are evenly spaced on the straight line between $\mathbf{b}_0$ and $\mathbf{b}_n$, then the degree n Bézier curve is the linear interpolant between $\mathbf{b}_0$ and $\mathbf{b}_n$. See Figure 4.4.

- *Extrapolation*: For values of t outside $[0,1]$, the curve will in general not remain within the control polygon's convex hull. See Figure 4.5 for an illustration. Numerical stability issues [49,50] and unpredictable behavior make this an undependable tool in a practical setting.

- *Special geometry*: On the right of Figure 4.2, three cubic Bézier curves are illustrated; From top to bottom we have one with two inflection points, one with a cusp, and one with a loop and therefore self-intersects.

- *Functional curves*: The quadratic curve defined by (4.3) and (4.4) is a functional curve, and thus one dimension is a linear polynomial. The linear precision property

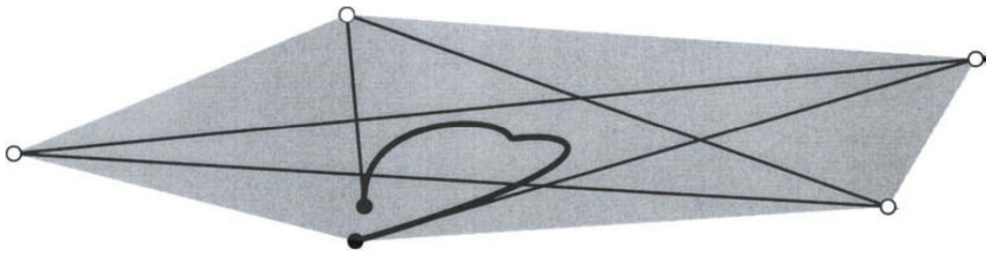

Figure 4.3. Convex hull property: The convex hull of the polygon is shaded. For $t \in [0,1]$, the point $\mathbf{x}(t)$ lies in the convex hull of the Bézier polygon.

of Bézier curves dictates that a functional curve defined for $t \in [0,1]$ takes the form

$$\mathbf{x}(t) = \sum_{i=0}^{n} \begin{bmatrix} i/n \\ b_i \end{bmatrix} B_i^n(t),$$

and an example is illustrated in the right of Figure 4.1; Also see Figure 4.4.

- *Pseudo-local control*: Suppose we move the i^{th} control point. The curve changes the most in the vicinity of $t = i/n$. In fact, all points on the curve move in a direction parallel to the vector formed by the difference of the old and new control point, as illustrated in Figure 4.6.

- *Invariance under affine parameter transformations*: Particularly in the context of piecewise curves, it might be necessary to associate a parameter interval $u \in [a,b]$ with a Bézier curve. The parameter interval does not effect the shape of the Bézier curve. It is common practice to associate the *global parameter* u with the *local parameter* $t \in [0,1]$ via the simple transformation $t = (u-a)/(b-a)$.

Cubic Bézier curves are perhaps utilized more than any other degree. This is primarily due to the fact that their shape is flexible "enough," while at the same time somewhat predictable because the degree is rather low.[5] Complex shapes are typically modeled with *piecewise polynomials* as discussed in the B-Spline Basics Chapter 6.

Other interesting properties of Bézier curves are explored in [4,11,26,34,44,55], effective algorithms are analyzed in [50,85], and applications which take advantage of the Bézier curve form are found in [14,23,68].

[5]There is a historical reason too: They are another representation of cubic Hermite curves, which have been used for many years. See Section 4.2.8 for more on the Hermite form.

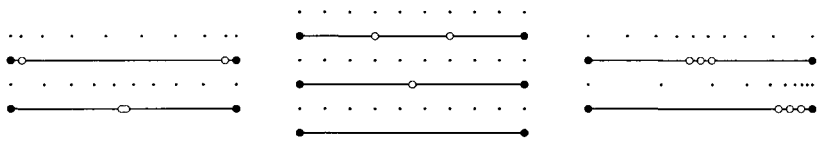

Figure 4.4. Linear precision property: For each curve, ten points with uniformly distributed parameters are plotted over the polygon. Middle column: Bézier curves with linear precision.

4.2.3. The de Casteljau algorithm for Bézier curves

The de Casteljau algorithm provides a means for evaluating Bézier curves, but it also provides for greater understanding of Bézier methods as a whole. Further insight into this important algorithm may be found in Boehm and Müller [16].

The de Casteljau algorithm for the evaluation of a degree n Bézier curve takes the following form.

de Casteljau Algorithm

Given: Bézier points $\mathbf{b}_i$ for $i = 0, \ldots, n$, and parameter $t \in [0, 1]$.
Find: The point $\mathbf{b}_0^n(t)$ on the curve.
Compute: Set $\mathbf{b}_i^0 = \mathbf{b}_i$ and compute the points

$$\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1} + t\mathbf{b}_{i+1}^{r-1} \qquad \begin{cases} r = 1, \ldots, n \\ i = 0, \ldots, n-r. \end{cases} \tag{4.8}$$

Figure 4.7 illustrates the algorithm for a cubic at $t = 1/4$. Notice that this recursive algorithm simply consists of repeated linear interpolation. Each step builds a new point from two other points in the *ratio* $t : (1-t)$.

A convenient schematic tool for describing the algorithm is to arrange the involved points in a triangular diagram. For example, evaluation of a cubic curve results in the following points.

$$\begin{array}{cccc} \mathbf{b}_0 & & & \\ \mathbf{b}_1 & \mathbf{b}_0^1 & & \\ \mathbf{b}_2 & \mathbf{b}_1^1 & \mathbf{b}_0^2 & \\ \mathbf{b}_3 & \mathbf{b}_2^1 & \mathbf{b}_1^2 & \mathbf{b}_0^3. \end{array} \tag{4.9}$$

An interesting point to note is that each of the intermediate points $\mathbf{b}_i^r(t)$ in the de Casteljau algorithm is actually a point on a degree r curve. In Figure 4.8, a degree five curve illustrates this point for which $\mathbf{b}_0^r(t)$ is plotted for $r = 2, 3, 4, 5$.

Examining Figure 4.7, observe two special polygons:

$$\mathbf{b}_0, \mathbf{b}_0^1, \mathbf{b}_0^2, \mathbf{b}_0^3 \quad \text{and} \quad \mathbf{b}_0^3, \mathbf{b}_1^2, \mathbf{b}_2^1, \mathbf{b}_3.$$

Figure 4.5. Extrapolation property: The curve is plotted for values of $t \in [-1.0, 2.3]$.

Each of these polygons defines the two segments of the curve corresponding to $[0, t]$ and $[t, 1]$ with respect to the original curve. Redefining a curve in this manner is called *subdivision.* In the schematic triangular diagram (4.9), the control points for these curves are along the diagonal and the base of the triangle, or more specifically the "left" and "right" control points are

$$\mathbf{l}_i = \mathbf{b}_0^i \quad \text{and} \quad \mathbf{r}_i = \mathbf{b}_i^{n-i}.$$

Subdivision must not necessarily take place within $[0, 1]$, although this is extrapolation. The foundations of subdivision are based on the work of de Casteljau [31] and Staerk [101]. Additional information may be found in [43,57,59,60]. Schwartz [95] develops formulas based on a *shift operator* technique.[6]

Subdivision may be repeated: Each of the two new control polygons may be subdivided, and so on. The resulting sequence of control polygons will ultimately converge to the curve. This result is explored by Cohen and Schumaker [24] and Dahmen [27]. Convergence is fast, and thus repeated subdivision could be used to render a curve. See Lane and Riesenfeld [75] and Bartels *et. al.* [6]. Another application of repeated subdivision is the intersection of a 2D Bézier curve with a line. Figure 4.9 illustrates the basic idea, which is to repeatedly subdivide at $t = 1/2$ and check the intersection of the control polygon's minmax box and the line. When the line intersects a minimax box whose dimension is less than some input tolerance, then an intersection is recorded at the midpoint of the minmax box.

In Section 4.2.5 a practical method of computing derivatives of a Bézier curve via the de Casteljau algorithm is discussed. Generally, Bézier curves are evaluated via the de Casteljau algorithm rather than computing the Bernstein polynomials directly.

4.2.4. Bernstein polynomials

Let us take a closer look at the *Bernstein polynomials* from (4.6). This study will give insight into the appealing properties of Bézier curves. Bernstein polynomials' relation to Bézier curves was discovered by Forrest [54], however also see Bézier [8] and de Casteljau [30].

A geometric approach to studying Bernstein polynomials is realized by formulating them as functional Bézier curves. To form the j^{th} basis function, recall from Section 4.2.2

[6]This technique, first introduced for Bézier techniques by Hosaka and Kimura [67] offers a concise notation. It is used to some extent in Hoschek and Lasser [69].

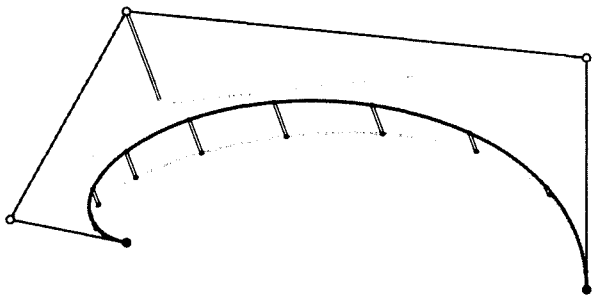

Figure 4.6. Pseudo-local control property: Moving the middle control point causes all points on the curve to move in the same direction. A double line indicates the change in a particular point.

that we assign

$$\mathbf{b}_i = \begin{bmatrix} i/n \\ b_i \end{bmatrix} \qquad \begin{cases} b_i = 1 \text{ if } i = j \\ b_i = 0 \text{ if } i \neq j. \end{cases}$$

A few sets of Bernstein basis functions are illustrated in Figure 4.10.

Let us look at some properties of the Bernstein polynomials.

- *Partition of unity*: For any particular value of t, the sum of the Bernstein polynomials is one:

  $$\sum_{i=0}^{n} B_i^n(t) = 1.$$

  This is necessary for (4.5) to be a barycentric combination.[7]

- *Non-negativity*: Each Bernstein polynomial is non-negative within the interval $[0, 1]$. This property, along with the fact that they sum to one, results in the convex hull property from Section 4.2.2.

- *Symmetry*: The relation $B_i^n(t) = B_{n-i}^n(1-t)$, follows directly from (4.6). This is reflected in the symmetry property of Bézier curves.

- *Recursion*: The degree n polynomials can be generated from the degree $n-1$ polynomials,

  $$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t).$$

  See [43] for a derivation of the de Casteljau algorithm using this property.

[7] See the Geometric Fundamentals Chapter 2.

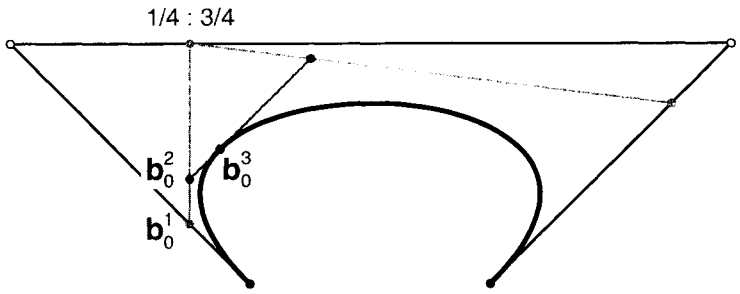

Figure 4.7. The de Casteljau algorithm applied to a cubic curve for $t = 1/4$.

- *Interval end conditions*:

$$B_i^n(0) = \delta_{i,0} \quad \text{and} \quad B_i^n(1) = \delta_{i,n} \quad \text{where} \quad \delta_{i,j} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$$

  As a result, Bézier curves have the endpoint interpolation property.

- *Linear precision*: The special linear combination

$$\sum_{i=0}^{n} \frac{i}{n} B_i^n(t) = t$$

  results in the linear precision property of Bézier curves.

- *Single maximum*: $B_i^n(t)$ has one maximum, and this maximum occurs at $t = i/n$. This allows Bézier curves pseudo-local control.

Goldman [58] takes an interesting look at Bernstein polynomials in relation to other blending functions.

4.2.5. Derivatives of Bézier curves

Differentiating a parametric curve simply involves differentiating each component. The resulting vector is the *tangent vector* of the curve. If we take a Bézier curve of the form (4.5), and differentiate with respect to the parameter t, we obtain

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \dot{\mathbf{x}}(t) = n \sum_{i=0}^{n-1} \Delta\mathbf{b}_i B_i^{n-1}(t), \tag{4.10}$$

where $\Delta\mathbf{b}_i = \mathbf{b}_{i+1} - \mathbf{b}_i$ is known as a *forward difference*. Notice that $\dot{\mathbf{x}}(t)$ is simply a degree $n-1$ Bézier curve with control vectors (rather than points). Figure 4.11 illustrates

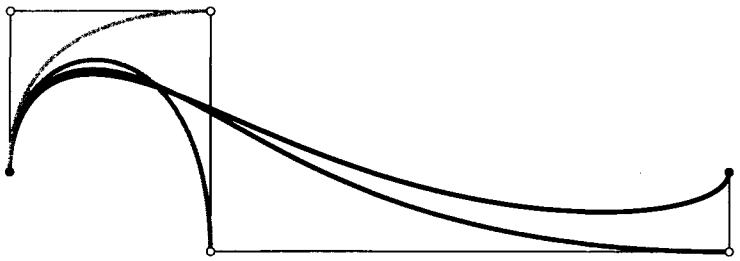

Figure 4.8. The intermediate points $\mathbf{b}_0^r(t)$ from the de Casteljau algorithm produce curves of degree r when plotted for all $t \in [0,1]$.

this formulation of the first derivative, which is also called a *hodograph*. At the endpoints, the derivative takes the simple form

$$\dot{\mathbf{x}}(0) = n\Delta\mathbf{b}_0 \quad \text{and} \quad \dot{\mathbf{x}}(1) = n\Delta\mathbf{b}_{n-1},$$

which implies that the tangents at the ends are parallel to the polygon legs there. More on hodographs may be found in Bézier [8], Forrest [54], Sederberg and Wang [97]; and see Nachman [82] for more on derivatives.

The first derivative (4.10) can be reformulated using the commutativity of the summation and difference operators, thus becoming

$$\dot{\mathbf{x}}(t) = n\Delta \sum_{i=0}^{n-1} \mathbf{b}_i B_i^{n-1}(t) = n\Delta\mathbf{b}_0^{n-1}. \tag{4.11}$$

Recall from the de Casteljau algorithm and (4.9) that $n\Delta\mathbf{b}_0^{n-1}$ is computed in the next to last step of the algorithm. Figure 4.12 illustrates.

The second derivative of a Bézier curve follows by differentiating (4.10) again, producing

$$\ddot{\mathbf{x}}(t) = n(n-1)\sum_{i=0}^{n-2} \Delta^2\mathbf{b}_i B_i^{n-2}(t), \tag{4.12}$$

where $\Delta^2\mathbf{b}_i = \Delta(\Delta\mathbf{b}_i) = \mathbf{b}_{i+2} - 2\mathbf{b}_{i+1} + \mathbf{b}_i$. As with the first derivative, the second derivative can also be reformulated in terms of intermediate points in the de Casteljau algorithm. Now, the points in the second-to-last column of (4.9) define the second derivative:

$$\ddot{\mathbf{x}}(t) = \Delta^2\mathbf{b}_0^{n-2}(t).$$

At the endpoints, the second derivative has a nice geometric interpretation which is illustrated in Figure 4.13. Higher derivatives follow similarly.

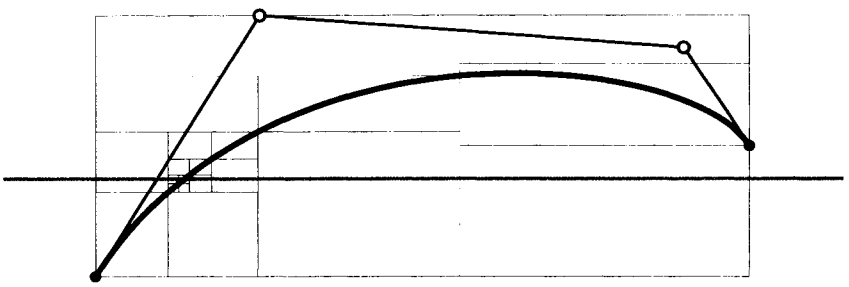

Figure 4.9. Finding an intersection of a cubic Bézier curve with the $x-$axis (gray) via repeated subdivision.

4.2.6. Degree elevation of Bézier curves

A degree n polynomial is also one of degree $n+1$. A curve in monomial form will simply have a zero leading coefficient. Similarly, we may write a degree n Bézier curve as one of degree $n+1$. We'll use a quadratic curve to demonstrate the principle. The trick is to multiply the quadratic expression by $[t+(1-t)]$. Reassembling common powers of t yields

$$\mathbf{x}(t) = \mathbf{b}_0 B_0^3 + [\frac{1}{3}\mathbf{b}_0 + \frac{2}{3}\mathbf{b}_1]B_1^3 + [\frac{2}{3}\mathbf{b}_1 + \frac{1}{3}\mathbf{b}_2]B_2^3 + \mathbf{b}_2 B_3^3.$$

This is the process of *degree elevation*. The trace of the curve written as a cubic is identical to that of the original quadratic.

Generalizing this process: Degree elevation of a degree n Bézier curve with control points $\mathbf{b}_j$ produces a curve of degree $n+1$ with control points $\mathbf{c}_i$, where

$$\mathbf{c}_i = \frac{i}{n+1}\mathbf{b}_{i-1} + (1 - \frac{i}{n+1})\mathbf{b}_i. \tag{4.13}$$

An example is illustrated in Figure 4.14.

Repeated degree elevation results in a polygon which converges to the curve, although Cohen and Schumaker [24] show this is not a practical method to render a Bézier curve. Trump and Prautzsch [103] examine arbitrarily high degree elevation. This convergence property follows from the *Weierstrass approximation theorem*, and more details are given by Farin [43]. Additionally, see Davis [29] and Korovkin [74].

A wealth of literature may be found on the reverse process: *degree reduction*. See [19,37,38,43,54,25,84,105] for a variety of methods. How can we tell if a Bézier curve is really a degree elevated curve? A degree $n-1$ curve will have an n^{th} derivative that is identically zero, or $\Delta^n \mathbf{b}_0 = \mathbf{0}$.

4.2.7. Interrogation techniques for Bézier curves

Determining if a curve meets certain design specifications calls for methods to measure the curve. The interrogation methods below, curvature and torsion, are the most basic

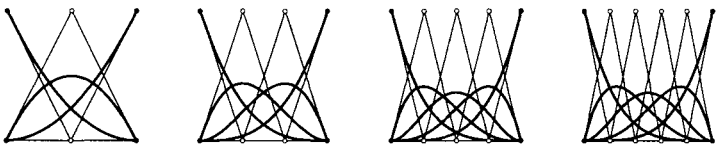

Figure 4.10. Quadratic, cubic, quartic, and quintic Bernstein basis functions plotted as functional Bézier curves.

measures. See the Geometric Fundamentals Chapter 2 for an introduction to these concepts. The discussion that follows focuses on these measures from the point of view of Bézier techniques.

The curvature of a curve is the most significant descriptor of its shape. Typically, curvature is visualized by graphing it as a function of t, which is called a *curvature plot.* An example is illustrated in Figure 4.15. For Bézier curves, curvature may be computed without having to compute derivatives explicitly. At $t = 0$, the curvature of a Bézier curve is given by

$$\kappa(0) = 2\frac{n-1}{n}\frac{\text{area}[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2]}{\|\mathbf{b}_1 - \mathbf{b}_0\|^3}. \tag{4.14}$$

We see that the curve has zero curvature at $t = 0$ if the three points $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ are collinear. A similar formula follows for $t = 1$. If the curvature is desired at parameter values other than 0 or 1, employ subdivision.

By definition, a 3D curve has nonnegative curvature. For 2D Bézier curves, signed curvature is easily introduced by defining the area in terms of a determinant. A signed curvature allows for identifying *inflection points*: These are points where the curvature changes sign. Roulier [92] examines Bézier curves with positive curvature.

The *torsion* measures the 3D twisting of a curve, or in other words, the change in the binormal vector. For Bézier curves, torsion takes on a simple form at the ends. At $t = 0$, we have

$$\tau(0) = \frac{3}{2}\frac{n-2}{n}\frac{\text{volume}[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]}{\text{area}[\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2]^2}.$$

The primary application of curvature and torsion has been in the context of piecewise curves, and some examples include [46,62,72,83,94].

4.2.8. Basis conversion

The Bernstein form has been shown to be numerically more stable than the monomial form by Farouki and Rajan [49]. Additionally, Farouki [47] showed that conversion between the monomial and Bernstein forms has the potential to be numerically unstable. A degree n curve in monomial form (4.2) is related to a curve in Bernstein form via

$$\mathbf{a}_i = \binom{n}{i}\Delta^i \mathbf{b}_0$$

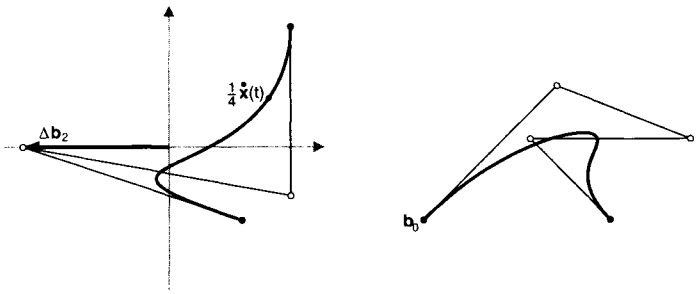

Figure 4.11. Right: A quartic Bézier curve. Left: The hodograph of the quartic, scaled by one-fourth.

for $i = 0, \ldots, n$. Thus, the first monomial coefficient is a point, namely $\mathbf{b}_0$, and the others are the scalings of the derivative vectors.

Cubic Hermite curves are another common curve form. They are specified by two points, $\mathbf{p}_0$ and $\mathbf{p}_1$, and tangent vectors $\mathbf{m}_0$ and $\mathbf{m}_1$ at the data points. More specifically, the Hermite form is defined as

$$\mathbf{x}(t) = \mathbf{p}_0 H_0^3(t) + \mathbf{m}_1 H_1^3(t) + \mathbf{m}_2 H_2^3(t) + \mathbf{p}_1 H_3^3(t),$$

where the Hermite basis functions H_i^3 are defined as

$$H_0^3(t) = B_0^3(t) + B_1^3(t), \qquad H_1^3(t) = \frac{1}{3} B_1^3(t),$$
$$H_3^3(t) = B_2^3(t) + B_3^3(t), \qquad H_2^3(t) = -\frac{1}{3} B_2^3(t).$$

This relation between the Bernstein and Hermite bases implies that

$$\mathbf{p}_0 = \mathbf{b}_0, \quad \mathbf{m}_0 = 3\Delta\mathbf{b}_0, \quad \mathbf{m}_1 = 3\Delta\mathbf{b}_2, \quad \mathbf{p}_1 = \mathbf{b}_3.$$

Li and Zhang [78] detail other basis conversions. Boehm [10,12] describes the conversion from B-spline to Bézier; See also the B-spline Basics Chapter 6.

4.2.9. Piecewise Bézier curves

Practical applications which take advantage of the geometric nature of Bézier curves typically depend on rather low degree curves. Complicated shapes are formed by piecing together curves. How to do so in a smooth manner is the focus of this section. A more rigorous examination of piecewise curves, called splines, may be found in the B-spline Basics Chapter 6. Issues of smoothness are investigated in the Geometric Continuity Chapter 8.

Suppose we begin with two degree n curves: $\mathbf{b}(u)$ with control points $\mathbf{b}_0, \ldots, \mathbf{b}_n$, defined over $[u_0, u_1]$, and $\mathbf{c}(u)$ with control points $\mathbf{b}_n, \ldots, \mathbf{b}_{2n}$, defined over $[u_1, u_2]$. Due to the

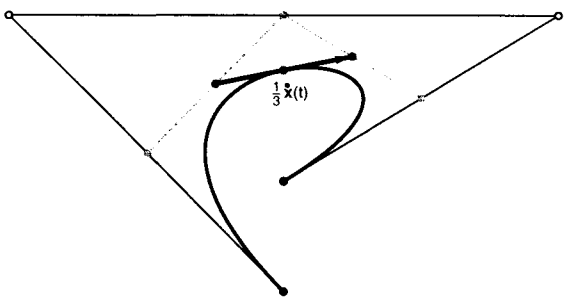

Figure 4.12. The first derivative of a Bézier curve is a scaling of the difference vector $(\mathbf{b}_1^{n-1} - \mathbf{b}_0^{n-1})$, and thus is a by-product of the de Casteljau algorithm.

endpoint interpolation property of Bézier curves, we know that these two curves meet, or are C^0, at $\mathbf{b}_n$.

What are the conditions on the control points for the curves to be differentiable at $\mathbf{b}_n$? The derivative of a Bézier curve defined over an arbitrary interval involves the chain rule, for example

$$\frac{d\mathbf{b}(u)}{du} = \frac{d\mathbf{b}}{dt}\frac{dt}{du} = \frac{n}{\Delta_0}\frac{d\mathbf{b}}{dt},$$

where $\Delta_0 = u_1 - u_0$. Drawing from Section 4.2.5, equate this expression for both curves at $u = u_1$, and we find

$$\frac{1}{\Delta_0}\Delta\mathbf{b}_{n-1} = \frac{1}{\Delta_1}\Delta\mathbf{b}_n$$

to be the condition for the curves to be C^1. More geometrically, this requires that the three points $\mathbf{b}_{n-1}, \mathbf{b}_n, \mathbf{b}_{n+1}$ be collinear and their spacing must be in the ratio $\Delta_0 : \Delta_1$ as illustrated in Figure 4.16, left.

C^2 constructions are also of practical importance. For the curves to be twice differentiable at the junction, the two quadratic polynomials defined by $\mathbf{b}_{n-2}, \mathbf{b}_{n-1}, \mathbf{b}_n$ and $\mathbf{b}_n, \mathbf{b}_{n+1}, \mathbf{b}_{n+2}$ must describe the same global quadratic polynomial. The curves must satisfy the C^1 conditions, and the additional geometric conditions are described in Figure 4.16, right. This necessitates the existence of an auxiliary point, shaded gray in the figure. Notice that the left figure is not C^2.

4.3. RECTANGULAR BÉZIER PATCHES

Let us extend Bézier techniques for curves to a surface form. A parametric surface is the result of a map of the real plane into 3-space. This plane, or *domain*, is defined by a (u, v)-coordinate system. A 3D surface point corresponding to a particular (u, v) is a

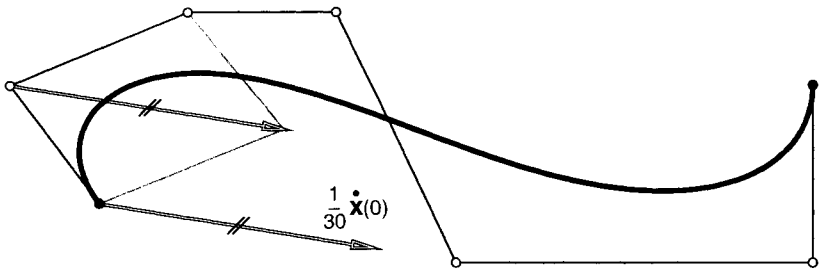

Figure 4.13. The second derivative of a degree six Bézier curve at $t = 0$ is a scaling of the diagonal of the parallelogram formed by the first three control points.

point:

$$\mathbf{x}(u,v) = \begin{bmatrix} f(u,v) \\ g(u,v) \\ h(u,v) \end{bmatrix}. \tag{4.15}$$

In this chapter, the functions f, g, h will be combinations of Bernstein polynomials. The surface $\mathbf{x}(u,v)$ is defined for all values of u and v, although we will primarily consider

$$\{(u,v) : 0 \leq u, v \leq 1\}. \tag{4.16}$$

As this indicates, the surface in this limited extent has a rectangular boundary. We will refer to this as a *patch*.

4.3.1. Bilinear patches

To begin our discussion of rectangular Bézier patches, let's start with the simplest form: bilinear patches. As the name suggests, the degree is linear in each parametric direction.

A bilinear Bézier patch $\mathbf{x}(u,v)$ is defined by four points $\mathbf{b}_{0,0}, \mathbf{b}_{0,1}, \mathbf{b}_{1,0}, \mathbf{b}_{1,1}$, and it takes the form

$$\mathbf{x}(u,v) = \begin{bmatrix} B_0^1(u) & B_1^1(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_{0,0} & \mathbf{b}_{0,1} \\ \mathbf{b}_{1,0} & \mathbf{b}_{1,1} \end{bmatrix} \begin{bmatrix} B_0^1(v) \\ B_1^1(v) \end{bmatrix}. \tag{4.17}$$

Figure 4.17, left, illustrates such a surface. The linear Bernstein polynomials, for example in u, are simply $B_0^1(u) = (1-u)$ and $B_1^1(u) = u$.

At first glance, (4.17) does not convey very much geometric information. By simply rewriting the bilinear patch as

$$\mathbf{x}(u,v) = (1-v)\mathbf{c}_0 + v\mathbf{c}_1 \tag{4.18}$$

where

$$\mathbf{c}_0 = (1-u)\mathbf{b}_{0,0} + u\mathbf{b}_{1,0} \quad \text{and} \quad \mathbf{c}_1 = (1-u)\mathbf{b}_{0,1} + u\mathbf{b}_{1,1},$$

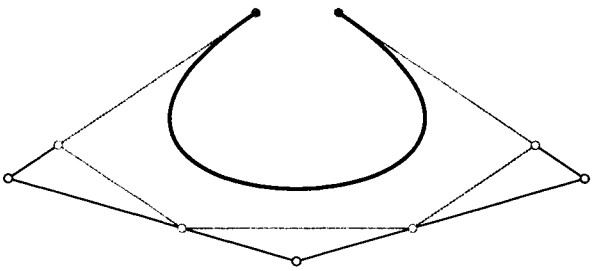

Figure 4.14. Degree elevation of a quartic Bézier curve. The quintic polygon is gray.

we can gather a much better feeling for the shape of the bilinear patch. Figure 4.17, right, illustrates this construction which builds intermediate points in the u-direction, and then builds the patch point by linear interpolation in the v-direction.

Alternatively, we could have built intermediate points in the v-direction:

$$\mathbf{d}_0 = (1-v)\mathbf{b}_{0,0} + v\mathbf{b}_{0,1} \quad \text{and} \quad \mathbf{d}_1 = (1-v)\mathbf{b}_{1,0} + v\mathbf{b}_{1,1},$$

and then the point on the patch is

$$\mathbf{x}(u,v) = (1-u)\mathbf{d}_0 + u\mathbf{d}_1. \tag{4.19}$$

The result (4.18) is the same as (4.19).

Another name for a bilinear patch is a *hyperbolic paraboloid.* It is covered by two families of straight lines, which is apparent when considering the curves defined by (4.18) and (4.19). These two sets of curves on the patch are called *isoparametric curves.* The four isoparametric curves (lines) corresponding to the edges, $(u,0)$,$(u,1)$,$(0,v)$, and $(1,v)$, are commonly referred to as the *boundary curves* of the patch.

A hyperbolic paraboloid also contains curves. For instance, consider the line $u = v$ in the domain. In parametric form, it may be written as $u(t) = t$, $v(t) = t$. This domain diagonal is mapped to the 3D curve $\mathbf{c}(t) = \mathbf{x}(t,t)$ on the surface. In more detail:

$$\mathbf{c}(t) = \begin{bmatrix} 1-t & t \end{bmatrix} \begin{bmatrix} \mathbf{b}_{0,0} & \mathbf{b}_{0,1} \\ \mathbf{b}_{1,0} & \mathbf{b}_{1,1} \end{bmatrix} \begin{bmatrix} 1-t \\ t \end{bmatrix},$$

and after collecting terms gives a quadratic Bézier curve

$$\mathbf{c}(t) = \mathbf{b}_{0,0}B_0^2(t) + [\frac{1}{2}\mathbf{b}_{0,1} + \frac{1}{2}\mathbf{b}_{1,0}]B_1^2(t) + \mathbf{b}_{1,1}B_2^2(t).$$

4.3.2. Bézier patches

Generalizing (4.17) to higher degrees, a degree (m,n) rectangular Bézier patch takes the form

$$\mathbf{x}(u,v) = \begin{bmatrix} B_0^m(u) & \dots & B_m^m(u) \end{bmatrix} \begin{bmatrix} \mathbf{b}_{0,0} & \dots & \mathbf{b}_{0,n} \\ \vdots & & \vdots \\ \mathbf{b}_{m,0} & \dots & \mathbf{b}_{m,n} \end{bmatrix} \begin{bmatrix} B_0^n(v) \\ \vdots \\ B_n^n(v) \end{bmatrix}. \tag{4.20}$$

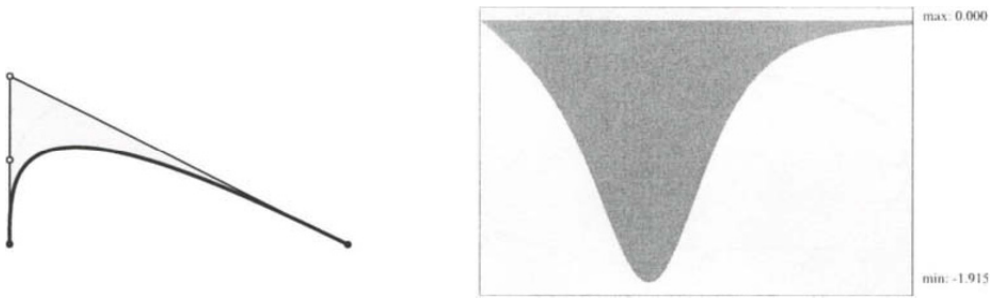

Figure 4.15. A curvature plot of a cubic Bézier curve.

Figure 4.18 illustrates a degree $(3,3)$ surface. The collection of control points is referred to as the *control net*. Equation (4.20) may be conveniently abbreviated as

$$\mathbf{x}(u,v) = M^{\mathrm{T}}\mathbf{B}N. \tag{4.21}$$

We may also write a Bézier patch as

$$\mathbf{x}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{b}_{i,j}B_i^m(u)B_j^n(v). \tag{4.22}$$

Bézier patches fall into the class of *tensor product surfaces*. The tensor product property is a very powerful conceptual tool for understanding Bézier patches. Figure 4.19 illustrates how the shape of a Bézier patch can be thought of as a record of the shape of a template moving and changing shape through space. Consider one value of v and the term $\mathbf{D} = \mathbf{B}N$ in (4.21). This defines a point $\mathbf{d}_i$ on each curve defined by the control polygon $\mathbf{b}_{i,j}, j = 0,\ldots,n$. The $\mathbf{d}_i$ are the control points for a curve in the u-direction, namely $M^{\mathrm{T}}\mathbf{D}$. As u varies, this expression defines the shape of the template for one particular v. In a chapter written by Bézier [43], this concept is discussed from a practitioner's point of view.

4.3.3. Properties of Bézier patches

Many of the properties of Bézier patches are direct generalizations of the curve ones.

- *Endpoint interpolation*: The patch passes through the four corner control points, that is

  $$\mathbf{x}(0,0) = \mathbf{b}_{0,0} \qquad \mathbf{x}(1,0) = \mathbf{b}_{m,0}$$
  $$\mathbf{x}(0,1) = \mathbf{b}_{0,n} \qquad \mathbf{x}(1,1) = \mathbf{b}_{m,n}.$$

  Also, the control net's boundary control points are the control points of the patch boundary curves. For example: the curve $\mathbf{x}(u,0)$ has the control polygon $\mathbf{b}_{i,0}$ for $i = 0,\ldots,m$.

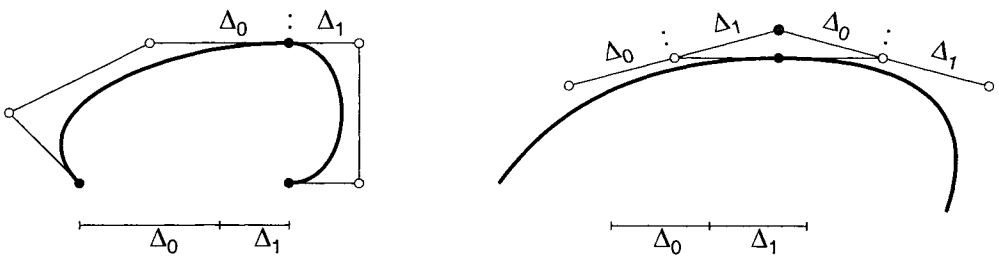

Figure 4.16. Bézier curve C^1 (left) and C^2 (right) conditions.

- *Symmetry*: We could re-index the control net so that any of the corners corresponds to $\mathbf{b}_{0,0}$, and evaluation would result in a patch with the same shape as the original one.

- *Affine invariance*: Apply an affine map to the control net, and then evaluate the patch. This surface will be identical to the surface created by applying the same affine map to the original patch. See (4.7) for the analogous property for curves.

- *Convex hull*: For $(u, v) \in [0, 1] \times [0, 1]$, the patch $\mathbf{x}(u, v)$ is in the convex hull of the the control net.

- *Bilinear precision*: A degree (m, n) patch is identical to the bilinear interpolant to the four corner control points if the control points satisfy the following conditions. The boundary curves are linearly precise, and the interior control points are uniformly-spaced on lines connecting corresponding boundary control points on adjacent edges

- *Tensor product*: Bézier patches are in the class of tensor product surfaces. This property allows Bézier patches to be dealt with in terms of isoparametric curves, which in turn simplifies evaluation and other operations. This also implies that the *total degree* of a (m, n) patch is $2mn$. The total degree is the maximum number of intersections of the patch with a straight line.

- *Functional patches*: A functional Bézier patch defined over $[0, 1] \times [0, 1]$ has Bézier control points

$$\mathbf{b}_{i,j} = \begin{bmatrix} i/m \\ j/n \\ b_{i,j} \end{bmatrix}.$$

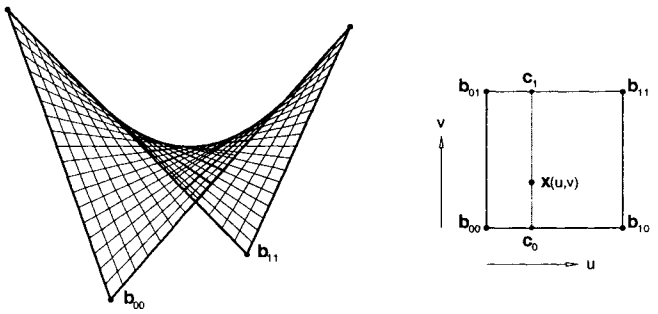

Figure 4.17. A bilinear Bézier patch illustrated in a 3D view on the left and the construction of $\mathbf{x}(1/3, 1/3)$ is on the right.

4.3.4. Evaluation of Bézier patches

Let us take advantage of the de Casteljau algorithm and the tensor product property as described in Section 4.3.2 to formulate a straightforward method to evaluate a Bézier patch, the *2-stage de Casteljau evaluation* method. First, consider $\mathbf{C} = M^{\mathrm{T}}\mathbf{B}$ in (4.21). The elements of $\mathbf{C}$ take the form

$$\mathbf{c}_j = \sum_{i=0}^{m} \mathbf{b}_{i,j} B_i^m(u) \tag{4.23}$$

for $j = 0, \ldots, n$. Simply evaluate each degree m Bézier curve using the de Casteljau algorithm. The second and final evaluation step, $\mathbf{x}(u, v) = \mathbf{C}N$ or

$$\mathbf{x}(u, v) = \sum_{j=0}^{n} \mathbf{c}_j B_j^n(v), \tag{4.24}$$

consists of evaluating a degree n Bézier curve via the de Casteljau algorithm. The roles of u and v can be switched: First compute $\mathbf{D} = \mathbf{B}N$, and then $\mathbf{x} = M^{\mathrm{T}}\mathbf{D}$.

Another evaluation method, the *3-stage de Casteljau evaluation* method, is quite useful if we want the first partial derivatives of the surface. It involves only a slight modification of the 2-stage method. Instead of computing the point on the curve in (4.23), stop the de Casteljau algorithm at the next to last step, saving the two points which span the tangent to the curve. As a result, we will have two "rows" of degree n curves. Next, evaluate these two curves at v, again stopping the de Casteljau algorithm at the next-to-last step, resulting in four points. These four points define a bilinear patch, and they span the tangent plane at (u, v). Evaluate this as we did in Section 4.3.1. How we use this algorithm in the calculation of derivatives is discussed in more detail in Section 4.3.5. Other evaluation methods and comparisons between them may be found in [43,45,79].

4.3.5. Derivatives of Bézier patches

A derivative of a surface is the tangent vector of a curve on the surface. There are two isoparametric curves through $\mathbf{x}(u, v)$. Let us focus on the u = *constant* curve, as in

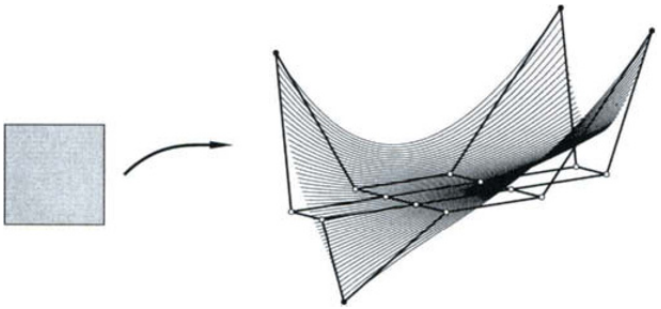

Figure 4.18. The control net and u = *constant* isoparametric curves of a degree $(3,3)$ Bézier patch are illustrated.

(4.24), and differentiate it with respect to v. The resulting tangent vector $\mathbf{x}_v$ is called the *v-partial derivative* or $v-$partial. More precisely, differentiating (4.22) with respect to v results in

$$\mathbf{x}_v(u,v) = \frac{\partial \mathbf{x}(u,v)}{\partial v} = n \sum_{i=0}^{m} \sum_{j=0}^{n-1} \Delta^{0,1}\mathbf{b}_{i,j} B_i^m(u) B_j^{n-1}(v), \tag{4.25}$$

where $\Delta^{0,1}\mathbf{b}_{i,j} = \mathbf{b}_{i,j+1} - \mathbf{b}_{i,j}$. The u-partial takes a very similar form,

$$\mathbf{x}_u(u,v) = m \sum_{i=0}^{m-1} \sum_{j=0}^{n} \Delta^{1,0}\mathbf{b}_{i,j} B_i^{m-1}(u) B_j^n(v), \tag{4.26}$$

where $\Delta^{1,0}\mathbf{b}_{i,j} = \mathbf{b}_{i+1,j} - \mathbf{b}_{i,j}$.

The second partials are also of practical use. For example, the second u-partial takes the form

$$\frac{\partial^2 \mathbf{x}(u,v)}{\partial u^2} = m(m-1) \sum_{i=0}^{m-2} \sum_{j=0}^{n} \Delta^{2,0}\mathbf{b}_{i,j} B_i^{m-2}(u) B_j^n(v),$$

where $\Delta^{2,0}\mathbf{b}_{i,j}$ is the second forward difference applied to the i indices. In other words, we simply take the derivative of the curve (4.26). The v-partial follows similarly, and formulae for higher order partials may be found in any of the texts cited in Section 4.1. More detail concerning computational efficiency may be found in Mann and DeRose [79] and Spitzmueller [100].

The mixed partial, or *twist vector*, is denoted by $\mathbf{x}_{u,v}(u,v)$ and is obtained in either of two ways:

$$\mathbf{x}_{u,v}(u,v) = \frac{\partial \mathbf{x}_u(u,v)}{\partial v} \quad \text{or} \quad \mathbf{x}_{u,v}(u,v) = \frac{\partial \mathbf{x}_v(u,v)}{\partial u}.$$

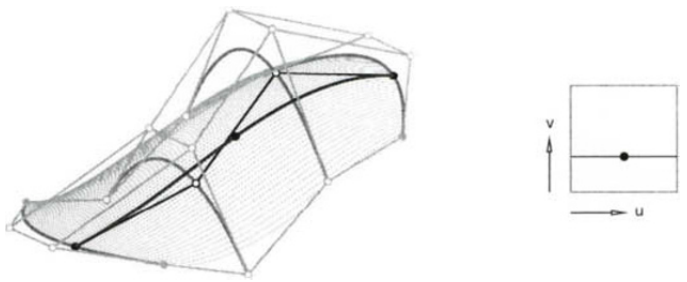

Figure 4.19. The shape of a Bézier patch may be described in terms of a moving template. The template for $v = 1/3$ is highlighted here.

Thus, by differentiating the v-partial (4.25) with respect to u,

$$\mathbf{x}_{u,v}(u,v) = mn \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \Delta^{1,1}\mathbf{b}_{i,j} B_i^{m-1}(u) B_j^{n-1}(v),$$

The mixed forward difference $\Delta^{1,1}\mathbf{b}_{i,j}$ is equivalent to $\Delta^{0,1}(\Delta^{1,0}\mathbf{b}_{i,j})$, that is

$$\Delta^{1,1}\mathbf{b}_{i,j} = \mathbf{b}_{i+1,j+1} - \mathbf{b}_{i+1,j} - \mathbf{b}_{i,j+1} + \mathbf{b}_{i,j}. \tag{4.27}$$

which measures the deviation of the quadrilateral defined by the four points in (4.27) from a parallelogram, and this is illustrated in Figure 4.20. Notice that the twist at the corners involves only the control points at the corners.

The *normal* is a fundamental geometric concept which is used throughout computer graphics and CAD/CAM; See the Direct Rendering of Freeform Surfaces Chapter 30. At a given point $\mathbf{x}(u,v)$ on a patch, the normal is perpendicular to the surface at $\mathbf{x}$, and the normal and $\mathbf{x}$ define the *tangent plane.* More precisely, the normal is defined by

$$\mathbf{n} = \frac{\mathbf{x}_u \wedge \mathbf{x}_v}{\|\mathbf{x}_u \wedge \mathbf{x}_v\|}, \tag{4.28}$$

and thus is a unit vector. Ideas for handling a denominator that is nearly zero are given attention in Farin [43].

Calculation of the normal requires knowledge of both the u- and v-partials. For this reason, the 3-stage evaluation method from Section 4.3.4 is more suitable in this situation than the 2-stage method.

4.3.6. Working with Bézier patches

Many of the operations applied to Bézier patches are direct generalizations of the techniques for curves. Again, this is due to the tensor product nature of Bézier patches, which allows for an algorithmic approach based on isoparametric curves.

Recall *degree elevation* for curves from Section 4.2.6; That same technique is used for an (m,n) Bézier patch in the following manner. Raising the degree from m to $m+1$ results

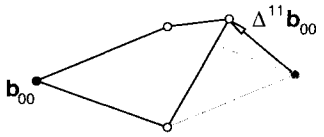

Figure 4.20. The mixed forward difference vector measures the deviation of the quadrangle from a parallelogram.

in a control net which has $n+1$ "columns" of control points, each column containing $m+2$ control points. These latter columns are simply obtained from the original columns by the process of degree elevation for curves.

Another curve operation from Section 4.2.3 is *subdivision*. Similarly, a patch may be subdivided into two patches. The $u-$parameter $\hat{u}$ splits the domain unit square into two rectangles as shown in Figure 4.21. The patch is split along this isoparametric curve into two patches, together identical to the original patch. The algorithm: Perform curve subdivision for each degree m "row" of the control net at parameter $\hat{u}$. Since each of the two new patches are Bézier patches, they are each defined over the domain (4.16). See Schwartz [95] for more on this topic.

The properties of Bézier patches are utilized to benefit applications in [2,64,77,90], convexity conditions are explored in [20], fitting and design issues are examined in [39,73, 99], and a hybrid Bézier patch is introduced in [52]. See the Geometric Fundamentals and Direct Rendering of Freeform Surfaces Chapters 2, 30 for information on interrogation techniques that may be applied to Bézier patches. Bézier patches can be extended to model volumes by generalizing the domain to a cube. For more information on this topic see [9,48,66,69,76,96].

4.3.7. C^1 Bézier patches

Following the discussion of piecewise curves in Section 4.2.9, here we examine the conditions under which two Bézier patches are differentiable. See the Geometric Continuity Chapter 8 for a more in-depth study.

We'll assume that the patches are the same degree (m, n) and C^0, that is, they share a common boundary curve. Additionally, define each Bézier patch over an arbitrary domain, thus we have $\mathbf{x}(u, v)$ defined over $[u_0, u_1] \times [v_0, v_1]$ and $\mathbf{y}(u, v)$ defined over $[u_1, u_2] \times [v_0, v_1]$. In order for $\mathbf{x}$ and $\mathbf{y}$ to be C^1 we require

$$\frac{\partial}{\partial u}\mathbf{x}(u, v) \mid_{u=u_1} = \frac{\partial}{\partial u}\mathbf{y}(u, v) \mid_{u=u_1} .$$

Drawing from the control point interpretation of the partials from Section 4.3.5 and applying the chain rule, this means that

$$\frac{1}{\Delta u_0} \sum_{j=0}^{n} \Delta^{1,0}\mathbf{b}_{m-1,j} B_j^n(v) = \frac{1}{\Delta u_1} \sum_{j=0}^{n} \Delta^{1,0}\mathbf{b}_{m,j} B_j^n(v),$$

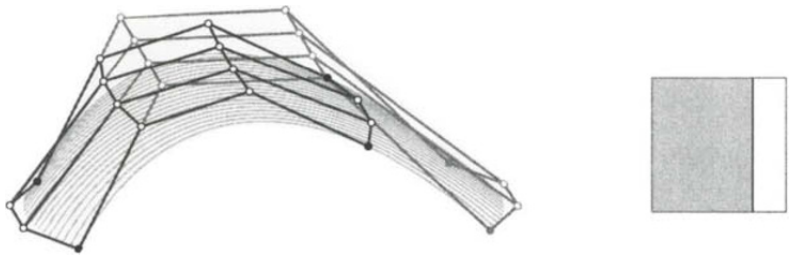

Figure 4.21. Subdivision of a Bézier patch at $\hat{u} = 0.75$. The original control net and the net for the patch over $[0, 0.75] \times [0, 1]$ are illustrated.

or in words, each row of control points must be collinear and reflect the corresponding ratio in the domain, as illustrated in Figure 4.22.

4.4. TRIANGULAR BÉZIER PATCHES

Triangular Bézier patches, or short *Bézier triangles*, are the true generalization of Bézier curves to a surface form.[8] In practice, the tensor product Bézier patch from Section 4.3 receives more attention due to their prevalent use in industry. Most notably, tensor product patches are an entity in data transfer formats such as IGES.

However, triangular patches have much to offer. One important offering is the ability to model objects with arbitrary topology. Think of a sphere-like object, for example. Modeling this entirely with rectangular patches would require a degenerate patch. Another offering of triangular patches is the ease of modeling quadric surfaces. Rational patches are necessary, however, and these are discussed in the Rational Techniques Chapter 5. See the History Chapter 1 for more uses of triangular methods.

4.4.1. Bézier triangles introduced

Bézier triangles are constructed from a triangular domain. Thus *barycentric coordinates* are fundamental to their construction by providing an elegant tool for defining points in a plane with respect to this triangular reference frame. An in-depth study of barycentric coordinates is provided in the Geometric Fundamentals Chapter 2; See also Boehm and Prautzsch [17].

Since the domain of a Bézier triangle is a triangle, the conceptual picture of the control points of the surface takes a triangular form as illustrated in Figure 4.23. Differing from their rectangular counterparts, Bézier triangles have a single degree associated with them.

[8] For a discourse on this topic, see Barry and Goldman [5].

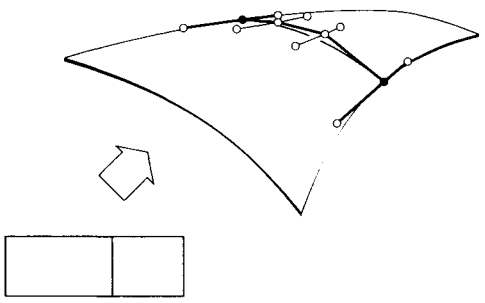

Figure 4.22. C^1 Bézier patches satisfy the criteria that the three control points in each row of along their common boundary curve are collinear, and the collinear points are positioned in the ratio dictated by the domain.

The control point indexing is described well by a quartic example.

$$\begin{array}{ccccccccc} & & & & \mathbf{b}_{004} & & & & \\ & & & \mathbf{b}_{103} & & \mathbf{b}_{013} & & & \\ & & \mathbf{b}_{202} & & \mathbf{b}_{112} & & \mathbf{b}_{022} & & \\ & \mathbf{b}_{301} & & \mathbf{b}_{211} & & \mathbf{b}_{121} & & \mathbf{b}_{031} & \\ \mathbf{b}_{400} & & \mathbf{b}_{310} & & \mathbf{b}_{220} & & \mathbf{b}_{130} & & \mathbf{b}_{040} \end{array}$$

Notice that the sum of the indices equals the degree. Often the abbreviated notation $\mathbf{b}_\mathbf{i}$ for $\mathbf{b}_{ijk}$ is used. There is a pattern to the notation: For example, the control points between $\mathbf{b}_{400}$ and $\mathbf{b}_{040}$ each take the form $\mathbf{b}_{\star\star 0}$. The barycentric coordinates $(1,0,0)$ are associated with $\mathbf{b}_{400}$, $(0,1,0)$ with $\mathbf{b}_{040}$, and $(0,0,1)$ with $\mathbf{b}_{004}$.

A degree n triangular Bézier patch is defined as

$$\mathbf{x}(\mathbf{u}) = \sum_{|\mathbf{i}|=n} \mathbf{b}_\mathbf{i} B_\mathbf{i}^n(\mathbf{u}) \tag{4.29}$$

where $\mathbf{u} = (u, v, w)$ are barycentric coordinates and $|\mathbf{i}| = n$ represents all (ijk) combinations which sum to n. The $B_\mathbf{i}^n(\mathbf{u})$ terms are the bivariate Bernstein polynomials

$$B_\mathbf{i}^n(\mathbf{u}) = \frac{n!}{i!j!k!} u^i v^j w^k. \tag{4.30}$$

They are bivariate since $w = 1 - u - v$. More on these in Section 4.4.4. A Bézier triangle consists of all points $\mathbf{x}(\mathbf{u})$ with barycentric coordinates $\mathbf{u}$ within the domain triangle, $0 \leq u, v, w \leq 1$. However, the surface is defined for $\mathbf{u}$ outside of the domain triangle also.

4.4.2. Properties of Bézier triangles

Many of the properties of Bézier triangles are direct generalizations of the curve ones.

Figure 4.23. Illustrated is a cubic triangular Bézier patch with its control net.

- *Endpoint interpolation*: The patch passes through the three corner control points, that is

  $$\mathbf{x}(1,0,0) = \mathbf{b}_{n,0,0} \quad \mathbf{x}(0,1,0) = \mathbf{b}_{0,n,0} \quad \mathbf{x}(0,0,1) = \mathbf{b}_{0,0,n}.$$

  Also, each control net boundary corresponds to the control polygon for the patch boundary curves. For example: the curve $\mathbf{x}(0,v,w)$ has the control polygon $\mathbf{b}_{0,j,k}$ for $k = 0,\ldots,n$ and $j + k = n$.

- *Symmetry*: We could re-index the control net so that any of the corners corresponds to $\mathbf{b}_{n,0,0}$, and evaluation would result in a patch with the same shape as the original one.

- *Affine invariance*: Apply an affine map to the control net, and then evaluate the patch. This surface will be identical to the surface created by applying the same affine map to the original patch.

- *Convex hull*: For $0 \leq u, v, w \leq 1$, the patch is in the convex hull of the control net.

- *Linear precision*: The three corner points define a plane. For a degree n patch, place the control points "uniformly" by constructing each $\mathbf{b}_{i,j,k}$ in this plane, and to have barycentric coordinates $(i/n, j/n, k/n)$. This Bézier triangle is identical to the linear Bézier triangle through the three corner points.

- *Total degree*: The total degree, or the maximum number of intersections of the patch with a straight line, of a degree n Bézier triangle is n^2.

- *Functional patches*: A functional triangular Bézier patch defined over $0 \leq u, v, w \leq 1$ has Bézier control points as follows. Construct the degree n function as $z = f(x,y)$. Suppose the three corner points are given. Then the other Bézier control points are constructed to have linear precision in the x- and y-coordinates. The z-coordinate is independent.

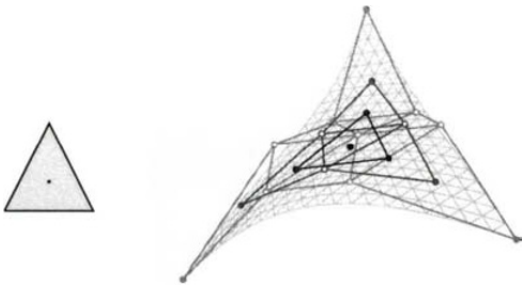

Figure 4.24. The triangular de Casteljau applied to a cubic patch for $\mathbf{u} = (1/3, 1/3, 1/3)$. The geometry of each step is differentiated by shades of gray.

4.4.3. The de Casteljau algorithm for Bézier triangles

Similar to the curve algorithm, the steps in the de Casteljau algorithm for Bézier triangles consists of repeated linear interpolation. Before describing the algorithm in more detail, we need to introduce the index notation $\mathbf{ei} = (100)$, $\mathbf{ej} = (010)$, and $\mathbf{ek} = (001)$.

Triangular de Casteljau Algorithm

Given: A degree n control net $\mathbf{b_i}$ with $|\mathbf{i}| = n$ and barycentric coordinates $\mathbf{u}$.

Find: A point $\mathbf{x}(\mathbf{u}) = \mathbf{b}^n_{0,0,0}(\mathbf{u})$ on the Bézier triangle.

Compute: Set $\mathbf{b}^0_\mathbf{i} = \mathbf{b_i}$, and compute the points:

$$\mathbf{b}^r_\mathbf{i}(\mathbf{u}) = u\mathbf{b}^{r-1}_{\mathbf{i}+\mathbf{ei}} + v\mathbf{b}^{r-1}_{\mathbf{i}+\mathbf{ej}} + w\mathbf{b}^{r-1}_{\mathbf{i}+\mathbf{ek}} \qquad \begin{cases} r = 1, \ldots, n & \text{and} \\ |\mathbf{i}| = n - r. \end{cases} \tag{4.31}$$

Figure 4.24 illustrates the steps of the algorithm applied to a cubic triangular patch. Linear interpolation is applied to the six "upright" triangles of the given control net. This produces a quadratic control net. Linear interpolation is applied to the three "upright" triangles, producing a linear control net. One linear interpolation step is applied, resulting in a point on the patch. Applying (4.31) to $\mathbf{u}$ on a domain boundary edge, for example $\mathbf{u} = (0, v, w)$, causes the triangular algorithm to take the form of the curve algorithm (4.8).

4.4.4. Bivariate Bernstein polynomials

Bivariate Bernstein polynomials were introduced in (4.30). Let us take a closer look at them with the purpose of gaining insight into the appealing geometric properties of Bézier triangles.

Just as with the univariate polynomials, a geometric approach to studying the bivariate Bernstein polynomials is achieved by formulating them as functional Bézier triangles. To plot the $\mathbf{i}^{\text{th}}$ basis function, recall the form the control points must take from Section 4.4.2. Giving one control point a $z = 1$ value, while all others have $z = 0$, will isolate the $\mathbf{i}^{\text{th}}$ basis function. Three cubic Bernstein basis functions are illustrated in Figure 4.25. A triangular diagram such as the following cubic one illustrates the correspondence between

Figure 4.25. Three cubic bivariate Bernstein basis functions plotted with their Bézier nets. The others follow from symmetry.

the control points and the basis functions.

$$\begin{array}{ccccccc} & & & w^3 & & & \\ & & uw^2 & & vw^2 & & \\ & u^2w & & uvw & & v^2w & \\ u^3 & & u^2v & & uv^2 & & v^3 \end{array} \tag{4.32}$$

Let us look at some properties of the Bernstein polynomials.

- *Partition of unity*: For any $\mathbf{u}$, the sum of the polynomials is one:

  $$\sum_{|\mathbf{i}|=n} B^n_{\mathbf{i}}(\mathbf{u}) = 1.$$

  This is necessary for (4.29) to be a barycentric combination.

- *Non-negativity*: Each polynomial is non-negative over $0 \leq u, v, w \leq 1$. This property, along with the fact that they sum to one, results in the *convex hull property*, as discussed in Section 4.4.2.

- *Symmetry*: The 3-fold symmetry they possess is apparent in (4.32).

- *Recursion*: The degree n polynomials can be generated from the degree $n-1$ polynomials as illustrated in the identity

  $$B^n_{\mathbf{i}}(t) = uB^{n-1}_{\mathbf{i}-\mathbf{ei}}(t) + vB^{n-1}_{\mathbf{i}-\mathbf{ej}}(t) + wB^{n-1}_{\mathbf{i}-\mathbf{ek}}(t).$$

- *Domain end conditions*:

  $$B^n_{\mathbf{i}}(1,0,0) = \delta_{\mathbf{i},(n,0,0)} \quad B^n_{\mathbf{i}}(0,1,0) = \delta_{\mathbf{i},(0,n,0)} \quad B^n_{\mathbf{i}}(0,0,1) = \delta_{\mathbf{i},(0,0,n)}$$

  where

  $$\delta_{\mathbf{i},(n,0,0)} = \begin{cases} 1 \text{ if } \mathbf{i} = (n,0,0) \\ 0 \text{ if } \mathbf{i} \neq (n,0,0) \end{cases} \qquad \delta_{\mathbf{i},(0,n,0)} = \begin{cases} 1 \text{ if } \mathbf{i} = (0,n,0) \\ 0 \text{ if } \mathbf{i} \neq (0,n,0) \end{cases} \qquad \delta_{\mathbf{i},(0,0,n)} = \begin{cases} 1 \text{ if } \mathbf{i} = (\\ 0 \text{ if } \mathbf{i} \neq (\end{cases}$$

  This results in corner point interpolation.

- *Linear precision:*

$$\sum_{|\mathbf{i}|=n} \frac{i}{n} B_{\mathbf{i}}^{n}(\mathbf{u}) = u,$$

and similarly for v and w. This identity results in the linear precision property of Bézier triangles.

See [20–22,61,104] for convexity analyses.

4.4.5. Derivatives of Bézier triangles

The nature of Bézier triangles calls for a more general derivative notation than was needed for rectangular patches. Here we need a *directional derivative*

$$D_{\mathbf{d}}\mathbf{b}^{n}(\mathbf{u}) = d\frac{\partial}{\partial u}\mathbf{b}^{n}(\mathbf{u}) + e\frac{\partial}{\partial v}\mathbf{b}^{n}(\mathbf{u}) + f\frac{\partial}{\partial w}\mathbf{b}^{n}(\mathbf{u}), \tag{4.33}$$

which is the derivative at $\mathbf{u}$ in the direction $\mathbf{d} = (d, e, f)$. The direction $\mathbf{d}$ is equivalent to the difference of two barycentric coordinates in the domain, thus $d + e + f = 0$.

The partials in (4.33) are defined by differentiating the bivariate Bernstein polynomials, for example

$$\frac{\partial}{\partial u}\mathbf{b}^{n}(\mathbf{u}) = n \sum_{|\mathbf{i}|=n-1} \mathbf{b}_{\mathbf{i}+\mathbf{ei}} B_{\mathbf{i}}^{n-1}(\mathbf{u}).$$

Combining these expression, we find

$$D_{\mathbf{d}}\mathbf{b}^{n}(\mathbf{u}) = n \sum_{|\mathbf{i}|=n-1} [d\mathbf{b}_{\mathbf{i}+\mathbf{ei}} + e\mathbf{b}_{\mathbf{i}+\mathbf{ej}} + f\mathbf{b}_{\mathbf{i}+\mathbf{ek}}] B_{\mathbf{i}}^{n-1}(\mathbf{u}). \tag{4.34}$$

Taking advantage of the notation from the triangular de Casteljau algorithm, we may rewrite (4.34) as

$$D_{\mathbf{d}}\mathbf{b}(\mathbf{u}) = n \sum_{|\mathbf{i}|=n-1} \mathbf{b}_{\mathbf{i}}^{1}(\mathbf{d}) B_{\mathbf{i}}^{n-1}(\mathbf{u}). \tag{4.35}$$

This may be interpreted as one step ($r = 1$) of the triangular de Casteljau algorithm with respect to $\mathbf{d}$, producing $\mathbf{b}_{\mathbf{i}}^{1}(\mathbf{d})$, and then $n - 1$ steps with respect to $\mathbf{u}$. Due to the linear nature of the de Casteljau algorithm, it is possible to rewrite (4.35) as

$$D_{\mathbf{d}}\mathbf{b}(\mathbf{u}) = n \sum_{|\mathbf{i}|=1} \mathbf{b}_{\mathbf{i}}^{n-1}(\mathbf{u}) B_{\mathbf{i}}^{1}(\mathbf{d}), \tag{4.36}$$

which should be interpreted as executing the de Casteljau algorithm on the original net with respect to $\mathbf{u}$ for $n-1$ steps, and then executing one step of the algorithm on the $\mathbf{b}_{\mathbf{i}}^{n-1}$ with respect to $\mathbf{d}$. This has a nice geometric interpretation: The three control points for the final step define the *tangent plane*, and thus the *normal* to the patch at $\mathbf{u}$.

The r^{th} directional derivative, mixed directional derivatives, and cross boundary directional derivatives are explored in detail by Farin [41,43].

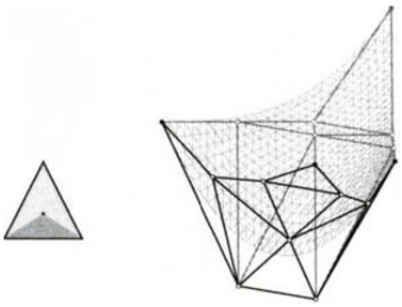

Figure 4.26. Subdivision of a Bézier triangle at $\hat{\mathbf{u}} = (1/3, 1/3, 1/3)$. One of the three resulting control nets is displayed in black.

4.4.6. Working with Bézier triangles

Just as for curves, *subdivision* of Bézier triangles is a by-product of the de Casteljau algorithm. Consider a point in the domain with barycentric coordinates $\hat{\mathbf{u}}$. The intermediate points from the de Casteljau algorithm, grouped appropriately, form three sub-patches as illustrated in Figure 4.26. More specifically, consider the patch across from $\mathbf{b}_{n00}$, it has control points

$$\hat{\mathbf{b}}_{r,j,k} = \mathbf{b}^r_{\mathbf{i}_0} \qquad \begin{cases} r = 0, \ldots, n \\ \mathbf{i}_0 = (0, j, k) \\ |\mathbf{i}_0| = n - r. \end{cases} \tag{4.37}$$

Repeated subdivision results in control nets which converge to the surface.

When $\hat{\mathbf{u}}$ is on a domain edge, for example $\hat{\mathbf{u}} = (0, v, w)$, then the control points from $\mathbf{b}_{n00}$ to $\mathbf{b}^n_{000}$ form the control polygon for the curve on the surface corresponding to the line in the domain from $(0, v, w)$ to $(1, 0, 0)$. This is called a *radial line.* When $\hat{\mathbf{u}}$ is outside the domain triangle, then continuity conditions for neighboring patches are revealed. See [13,15,51,57,69,98] for more detail and pointers to more literature.

Degree elevation for Bézier triangles allows a degree n patch to be written as a degree $n+1$ patch by defining $\mathbf{c}_\mathbf{j}$ such that

$$\sum_{|\mathbf{j}|=n+1} \mathbf{c}_\mathbf{j} B^{n+1}_\mathbf{j}(\mathbf{u}) = \sum_{|\mathbf{i}|=n} \mathbf{b}_\mathbf{i} B^n_\mathbf{i}(\mathbf{u}). \tag{4.38}$$

By multiplying the right-hand side of (4.38) by $(u+v+w)$, and gathering the appropriate terms, the new control points are found to be

$$\mathbf{c}_\mathbf{j} = \frac{1}{n+1}(i\mathbf{b}_{\mathbf{j}-\mathbf{ei}} + j\mathbf{b}_{\mathbf{j}-\mathbf{ej}} + k\mathbf{b}_{\mathbf{j}-\mathbf{ek}}), \qquad \begin{cases} \mathbf{j} = (i, j, k) \\ |\mathbf{j}| = n + 1. \end{cases}$$

Along a boundary curve, this expression reduces to that for curves. The degree elevated control net is within the convex hull of the original net. See Farin [41] for more details.

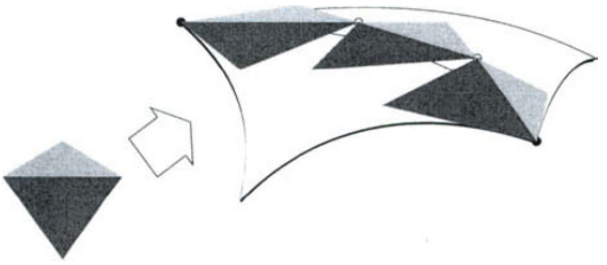

Figure 4.27. Two Bézier triangles are C^1 if adjacent triangles along the common boundary are coplanar and form affine pairs.

Utilizing the properties of Bézier triangles to benefit applications is explored in [3,42], triangular to rectangular patch conversion is studied in [18,70,71], and a classification of Bézier triangle may be found in [35].

To extend triangular patches to *volumes*, the domain becomes a tetrahedron. See [41,53,57,69,93] for more information.

4.4.7. C^1 Bézier triangles

The conditions under which two Bézier triangles are differentiable will draw from the directional derivative discussion of Section 4.4.5. We'll assume that the two patches are the same degree and C^0, that is, they share a common boundary curve.

Consider all cross-boundary directional derivatives at a point on the boundary of one triangle, for example along $u = 0$. As a result, $\mathbf{u} = (0, v, 1 - v)$ in (4.35), and the expression becomes univariate, thus these derivatives correspond to tangents to curves across the triangles. If all such derivates are equal for both patches, they are C^1.

However, a more constructive description is needed. Examining (4.36), we observe that the directional derivative at a boundary involves the first two rows of control points at that boundary. The subdivision formula (4.37) reveals a geometric description of the conditions on these control points. The second row of the Bézier triangle $\hat{\mathbf{b}}$ is described by

$$\hat{\mathbf{b}}_{1,j,k} = \mathbf{b}^1_{0,j,k} = u\mathbf{b}_{1,j,k} + v\mathbf{b}_{0,j+1,k} + w\mathbf{b}_{0,j,k+1}$$

for $j + k = n - 1$. This means that the triangle pairs along the boundary are coplanar and each of the triangle pairs can be described by the same affine map – they form affine pairs, as illustrated in Figure 4.27.

REFERENCES

1. G. Aumann. Interpolation with developable Bézier patches. *Computer Aided Geometric Design*, 8(5):409–420, 1991.

2. N. Aziz, R. Bataand, and S. Bhat. Bézier surface/surface intersection. *IEEE Computer Graphics and Applications*, 10(1):50–58, 1990.
3. R. Barnhill, B. Bloomquist, and A. Worsey. Adaptive contouring for triangular Bézier patches. In R. E. Barnhill, editor, *Geometry Processing*. SIAM, 1992.
4. P. Barry and R. Goldman. Three examples of dual properties of Bézier curves. In T. Lyche and L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 61–70. Academic Press, 1989.
5. P. Barry and R. Goldman. What is the natural generalization of a Bézier curve? In T. Lyche and L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 71–86. Academic Press, 1989.
6. R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1987.
7. P. Bézier. Example of an existing system in the motor industry: The Unisurf system. *Proc. Royal Soc. of Loindon*, A321:207–218, 1971.
8. P. Bézier. *Numerical Control: Mathematics and Applications*. Wiley, 1972. translated from the French by R. Forrest.
9. P. Bézier. General distortion of an ensemble of biparametric patches. *Computer-Aided Design*, 10(2):116–120, 1978.
10. W. Boehm. Generating the Bézier points of B-spline curves and surfaces. *Computer-Aided Design*, 13(6):365–366, 1981.
11. W. Boehm. On cubics: a survey. *Computer Graphics and Image Processing*, 19:201–226, 1982.
12. W. Boehm. Generating the Bézier points of triangular splines. In R. Barnhill and W. Boehm, editors, *Surfaces in Computer Aided Geometric Design*, pages 77–91. North-Holland, 1983.
13. W. Boehm. Subdividing multivariate splines. *Computer-Aided Design*, 15(6):345–352, 1983.
14. W. Boehm. Bézier presentations of airfoils. *Computer Aided Geometric Design*, 4(1-2):17–22, 1987.
15. W. Boehm and G. Farin. Letter to the editor. *Computer-Aided Design*, 15(5):260–261, 1983. Concerning subdivison of Bézier triangles.
16. W. Boehm and A. Müller. On de Casteljau's algorithm. *Computer Aided Geometric Design*, 16(7):583–586, 2000.
17. W. Boehm and H. Prautzsch. *Geometric Foundations of Geometric Design*. AK Peters, Boston, 1992.
18. I. Brueckner. Construction of Bézier points of quadrilaterals from those of triangles. *Computer-Aided Design*, 12(1):21–24, 1980.
19. G. Brunnett, T. Schreiber, and J. Braun. The geometry of optimal degree reduction of Bézier curves. *Computer Aided Geometric Design*, 13(8):773–788, 1996.
20. J Carnicer, M. Floater, and J. Peña. Linear convexity conditions for rectangular and triangular Bernstein–Bézier surfaces. *Computer Aided Geometric Design*, 15(1):27–38, 1997.
21. G. Chang and P. Davis. The convexity of Bernstein polynomials over triangles. *J Approx Theory*, 40:11–28, 1984.
22. G. Chang and Y. Feng. An improved condition for the convexity of Bernstein-Bézier

surfaces over triangles. *Computer Aided Geometric Design*, 1(3):279–283, 1985.
23. Y. Chua. Bézier brushstrokes. *Computer-Aided Design*, 22(9):550–555, 1990.
24. E. Cohen and L. Schumaker. Rates of convergence of control polygons. *Computer Aided Geometric Design*, 2(1-3):229–235, 1985.
25. J. Peters, D. Lutterkort, and U. Reif. Polynomial degree reduction in the L2-norm equals best Euclidean approximation of Bézier coefficients. *Computer Aided Geometric Design*, 16(7):607–612, 2000.
26. J. Peters, D. Nairn, and D. Lutterkort. Sharp, quantitative bounds on the distance between a polynomial piece and its Bézier control polygon. *Computer Aided Geometric Design*, 16(7):613–631, 2000.
27. W. Dahmen. Subdivision algorithms converge quadratically. *J. of Computational and Applied Mathematics*, 16:145–158, 1986.
28. M. Daniel and J. Daubisse. The numerical problem of using Bézier curves and surfaces in the power basis. *Computer Aided Geometric Design*, 6(2):121–128, 1989.
29. P. Davis. *Interpolation and Approximation*. Dover, New York, 1975. first edition 1963.
30. P. de Casteljau. Outillages méthodes calcul. Technical report, A. Citroën, Paris, 1959.
31. P. de Casteljau. Courbes et surfaces à pôles. Technical report, A. Citroën, Paris, 1963.
32. P. de Casteljau. *Shape Mathematics and CAD*. Kogan Page, London, 1986.
33. P. de Casteljau. de Casteljau's autobiography: My time at Citroën. *Computer Aided Geometric Design*, 16(7):583–586, 2000.
34. W. Degen. Some remarks on Bézier curves. *Computer Aided Geometric Design*, 5(3):259–268, 1988.
35. W. Degen. The types of triangular Bézier surfaces. In G. Mullineux, editor, *The Mathematics of Surfaces VI*, pages 153–170. Oxford University Press, 1996.
36. T. DeRose and R. Goldman. A tutorial introduction to blossoming. In H. Hagen and D. Roller, editors, *Geometric Modeling*. Springer, 1991.
37. M. Eck. Degree reduction of Bézier curves. *Computer Aided Geometric Design*, 10(3-4):237–252, 1993.
38. M. Eck. Least squares degree reduction of Bézier curves. *Computer-Aided Design*, 27(11):845–851, 1995.
39. E. Eisele. Best approximations of symmetric surfaces by biquadratic Bézier surfaces. *Computer Aided Geometric Design*, 11(3):331–343, 1994.
40. G. Farin. Some aspects of car body design at Daimler-Benz. In R. Barnhill and W. Boehm, editors, *Surfaces in Computer Aided Geometric Design*, pages 93–98. North-Holland, 1983.
41. G. Farin. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design*, 3(2):83–128, 1986.
42. G. Farin. The use of triangular patches in CAD. In M. Wozny, H. McLaughlin, and J. Encarnacao, editors, *Geometric Modeling for CAD Applications*, pages 191–194. North-Holland, 1988.
43. G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 1996. fourth edition.

44. G. Farin and P. Barry. A link between Lagrange and Bézier curve and surface schemes. *Computer-Aided Design*, 18:525–528, 1986.
45. G. Farin and D. Hansford. *The Essentials of CAGD*. AK Peters, 2000.
46. G. Farin and N. Sapidis. Curvature and the fairness of curves and surfaces. *IEEE Computer Graphics and Applications*, 9(2):52–57, 1989.
47. R. Farouki. On the stability of transformations between power and Bernstein polynomial forms. *Computer Aided Geometric Design*, 8(1):29–36, 1991.
48. R. Farouki and J. Hinds. A hierarchy of geometric forms. *IEEE Computer Graphics and Applications*, 5(5):51–78, 1985.
49. R. Farouki and V. Rajan. On the numerical condition of polynomials in Bernstein form. *Computer Aided Geometric Design*, 4(3):191–216, 1987.
50. R. Farouki and V. Rajan. Algorithms for polynomials in Bernstein form. *Computer Aided Geometric Design*, 5(1):1–26, 1988.
51. Y. Feng. Rates of convergence of Bézier nets over triangles. *Computer Aided Geometric Design*, 4(3):245–249, 1987.
52. T. Foley and K. Opitz. Hybrid cubic Bézier patches. In T. Lyche and L. Schumaker, editors, *Mathematical Methods in CAGD II*, pages 275–286. Academic Press, Boston, 1992.
53. T. Foley and H. Wolters. The hybrid quintic Bézier tetrahedron. *Computer Aided Geometric Design*, 14(7):603–618, 1997.
54. A. Forrest. Interactive interpolation and approximation by Bézier polynomials. *The Computer J*, 15(1):71–79, 1972. reprinted in CAD 22(9):527-537,1990.
55. A. Forrest. The twisted cubic curve: a computer-aided geometric design approach. *Computer-Aided Design*, 12(4):165–172, 1980.
56. J. Gallier. *Curves and Surfaces in Geometric Modeling: Theory and Algorithms*. Morgan-Kaufmann, 1998.
57. R. Goldman. Using degenerate Bézier triangles and tetrahedra to subdivide Bézier curves. *Computer-Aided Design*, 14(6):307–311, 1982.
58. R. Goldman. An urnful of blending functions. *IEEE Computer Graphics and Applications*, 3(7):49–54, 1983.
59. R. Goldman and T. DeRose. Recursive subdivision without the convex hull property. *Computer Aided Geometric Design*, 3(4):247–265, 1986.
60. R. Goldman and D. Heath. Linear subdivision is strictly a polynomial phenomenon. *Computer Aided Geometric Design*, 1(3):269–278, 1984.
61. J. Gregory and J. Zhou. Convexity of Bézier on sub-triangles. *Computer Aided Geometric Design*, 8(3):207–213, 1991.
62. H. Hagen. Bézier-curves with curvature and torsion continuity. *Rocky Mtn. J of Math.*, 16(3):629–638, 1986.
63. D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination*. Chelsea, New York, 1952.
64. H. Hochfeld. Surface description in the application at Volkswagen. In R. Barnhill and W. Boehm, editors, *Surfaces in Computer Aided Geometric Design*, pages 35–42. North-Holland, 1983.
65. H. Hochfeld and M. Ahlers. Role of Bézier curves and surfaces in the Volkswagen CAD approach from 1967 to today. *Computer-Aided Design*, 22(9):598–608, 1990.

66. D. Holliday and G. Farin. A geometric interpretation of the diagonal of a tensor-product Bézier volume. *Computer Aided Geometric Design*, 16(8):837–840, 1999.
67. M. Hosaka and F. Kimura. Synthesis methods of curves and surfaces in interactive CAD, Bologna. In *Proc. Interactive Techniques in CAD*, pages 151–156, 1978.
68. J. Hoschek. Offset curves in the plane. *Computer-Aided Design*, 17(2):77–82, 1985.
69. J. Hoschek and D. Lasser. *Grundlagen der Geometrischen Datenverarbeitung.* B.G. Teubner, Stuttgart, 1989. English translation: Fundamentals of Computer Aided Geometric Design, AK Peters, 1993.
70. S.-M. Hu. Conversion of a triangular Bézier patch into three rectangular bézier patches. *Computer Aided Geometric Design*, 13(3):219–226, 1996.
71. K. Iino and D. Wilde. Subdivision of triangular Bézier patches into rectangular Bézier patches. *Transactions of the ASME*, 1992.
72. A. Jones. Curvature integration through constrained optimization. In N. Sapidis, editor, *Designing Fair Curves and Surfaces*, pages 29–44. SIAM, Philadelphia, 1994.
73. L. Kocić. Modification of Bézier curves and surfaces by degree elevation technique. *Computer-Aided Design*, 23(10):692–699, 1991.
74. P. Korovkin. *Linear Operators and Approximation Theory.* Hindustan Publishing Co., Delhi, 1960.
75. J. Lane and R. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Trans. Pattern Analysis Machine Intell.*, 2(1):35–46, 1980.
76. D. Lasser. Bernstein-Bézier representation of volumes. *Computer Aided Geometric Design*, 2(1-3):145–150, 1985.
77. D. Lasser. Intersection of parametric surfaces in the Bernstein-Bézier representation. *Computer-Aided Design*, 18(4):186–192, 1986.
78. Y.-M. Li and X.-Y. Zhang. Basis conversion among Bézier, Tchebyshev and Legendre. *Computer Aided Geometric Design*, 15(6):637–642, 1998.
79. S. Mann and T. DeRose. Computing values and derivatives of Bézier and B-spline tensor products. *Computer Aided Geometric Design*, 12(1):107–110, 1995.
80. D. Marsh. *Applied Geometry for Computer Graphics and CAD.* Springer-Verlag, 1999.
81. M. Mortenson. *Geometric Modeling.* Wiley, 1985.
82. L. Nachman. A note on control polygons and derivatives. *Computer Aided Geometric Design*, 8(3):223–226, 1991.
83. A. Nutbourne, P. McLellan, and R. Kensit. Curvature profiles for plane curves. *Computer-Aided Design*, 4(4):176–184, 1972.
84. J. Peterson. Degree reduction of Bézier curves. *Computer-Aided Design*, 23(6):460–461, 1991. Letter to the Editor.
85. H. N. Phien and N. Dejdumrong. Efficient algorithms for Bézier curves. *Computer Aided Geometric Design*, 17(3):247–250, 2000.
86. L. Piegl and W. Tiller. *The NURBS Book.* Springer Verlag, 1997. second edition.
87. H. Pottmann and G. Farin. Developable rational Bézier and B-spline surfaces. *Computer Aided Geometric Design*, 12(5):513–531, 1995.
88. L. Ramshaw. Blossoming: a connect-the-dots approach to splines. Technical report, Digital Systems Research Center, Palo Alto, Ca, 1987.

89. L. Ramshaw. Béziers and B-splines as multiaffine maps. In R. Earnshaw, editor, *Theoretical Foundations of Computer Graphics and CAD*, pages 757–776. Springer Verlag, 1988.
90. T. Reuding. Bézier patches on cubic grid curves - an application to the preliminary design of a yacht hull surface. *Computer Aided Geometric Design*, 6(1):11–21, 1989.
91. A. Rockwood and P. Chambers. *Interactive Curves and Surfaces.* Morgan Kaufmann, 1996.
92. J. Roulier. Bézier curves of positive curvature. *Computer Aided Geometric Design*, 5(1):59–70, 1988.
93. M. Sabin. Trinomial basis functions for interpolation in triangular regions (Bézier triangles). Technical report, British Aircraft Corporation, 1971.
94. N. Sapidis and G. Koras. Visualization of curvature plots and evaluation of fairness: an analysis of the effect of scaling. *Computer Aided Geometric Design*, 14(4):299–311, 1997.
95. A. Schwartz. Subdividing Bézier curves and surfaces. In G. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 55–66. SIAM, Philadelphia, 1987.
96. T. Sederberg and S. Parry. Free-form deformation of solid geometric models. *Computer Graphics*, 20(4):151–160, 1986. SIGGRAPH proceedings.
97. T. Sederberg and X. Wang. Rational hodographs. *Computer Aided Geometric Design*, 4(4):333–335, 1987.
98. H.-P. Seidel. A general subdivision theorem for Bézier triangles. In T. Lyche and L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 573–582. Academic Press, 1989.
99. L. Shirman and C. Séquin. Local surface interpolation with Bézier patches: errata and improvements. *Computer Aided Geometric Design*, 8(3):217–222, 1991.
100. K. Spitzmueller. Partial derivatives of Bézier surfaces. *Computer-Aided Design*, 28(1), 1996.
101. E. Staerk. *Mehrfach differenzierbare Bézierkurven und Bézierflächen.* PhD thesis, T. U. Braunschweig, 1976.
102. B.-Q. Su and D.-Y. Liu. *Computational Geometry.* Academic Press, 1989.
103. W. Trump and H. Prautzsch. Arbitrarily high degree elevation of Bézier representations. *Computer Aided Geometric Design*, 13(5):387–398, 1996.
104. Z. Wang and Q. Liu. An improved condition for the convexity and positivity of Bernstein - Bézier surfaces over triangles. *Computer Aided Geometric Design*, 5(4):269–276, 1988.
105. M. Watkins and A. Worsey. Degree reduction for Bézier curves. *Computer-Aided Design*, 20(7):398–405, 1988.