Bézier- and B-spline techniques

Hartmut Prautzsch
Wolfgang Boehm
Marco Paluszny

March 26, 2002

To

Paul de Faget de Casteljau

Preface

Computer-aided modeling techniques have been developed since the advent of NC milling machines in the late 40's. Since the early 60's Bézier and B-spline representations evolved as the major tool to handle curves and surfaces. These representations are geometrically intuitive and meaningful and they lead to constructive numerically robust algorithms.

It is the purpose of this book to provide a solid and unified derivation of the various properties of Bézier and B-spline representations and to show the beauty of the underlying rich mathematical structure. The book focuses on the core concepts of Computer-aided Geometric Design (CAGD) with the intent to provide a clear and illustrative presentation of the basic principles as well as a treatment of advanced material, including multivariate splines, some subdivision techniques and constructions of arbitrarily smooth free-form surfaces.

In order to keep the book focused, many further CAGD methods are excluded. In particular, rational Bézier and B-spline techniques are not addressed since a rigorous treatment within the appropriate context of projective geometry would have been beyond the scope of this book.

The book grew out of several courses taught repeatedly at the graduate and intermediate under-graduate levels by the authors at the Rensselaer Polytechnic Institute, USA, the Universities of Braunschweig and Karlsruhe, Germany, and the Universidad Central de Venezuela. These courses were taught as part of the curricula in mathematics and computer sciences, and they were regularly attended also by students from electrical and mechanical engineering, geophysics and other sciences.

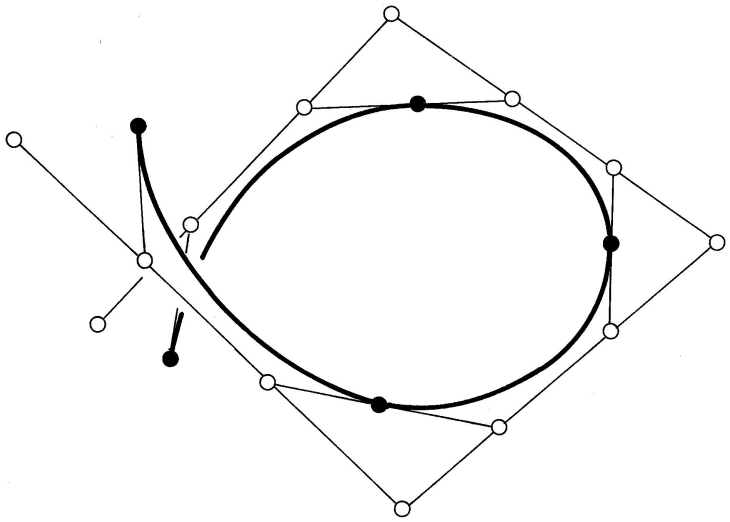
For the careful proofreading of parts of the manuscript, we like to thank Stefan Bischoff, Bernhard Garz, Georg Umlauf, Claudia Bangert and Norbert Luscher. Especially, we thank Christoph Pennekamp and Natalie Spinner for preparing the LaTeX file and Bernd Hamann for his critical, thorough and final proofreading.

Wolfenbüttel, Wolfgang Boehm
Caracas, Marco Paluszny
Karlsruhe, Hartmut Prautzsch

Contents

Part I

Curves



2 Bézier representation

2.1 Bernstein polynomials — 2.2 Bézier representation — 2.3 The de Casteljau algorithm — 2.4 Derivatives — 2.5 Singular parametrization — 2.6 A tetrahedral algorithm — 2.7 Integration — 2.8 Conversion to Bézier representation — 2.9 Conversion to monomial form — 2.10 Problems

Every polynomial curve segment can be represented by its so-called Bézier polygon. The curve and its Bézier polygon are closely related. They have common end points and end tangents, the curve segment lies in the convex hull of its Bézier polygon, etc. Furthermore, one of the fastest and numerically most stable algorithm used to render a polynomial curve is based on the Bézier representation.

2.1 Bernstein polynomials

Computing the binomial expansion

$$1 = (u + (1-u))^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i}$$

leads to the **Bernstein polynomials** of degree n,

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i} \ , \qquad i = 0, \ldots, n \ ,$$

which are illustrated in Figure 2.1 for $n = 4$.

The following properties of the Bernstein polynomials of degree n are important.

- *They are* **linearly independent**.

Namely, dividing $\sum_{i=0}^{n} b_i u^i (1-u)^{n-i} = 0$ by $(1-u)^n$, and setting $s =$

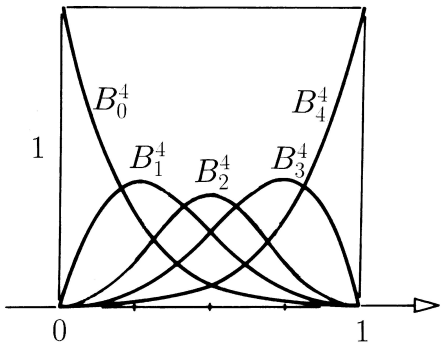


Figure 2.1: The Bernstein polynomials of degree 4 over $[0, 1]$.

$u/(1-u)$, gives $\sum_{i=0}^{n} b_i s^i = 0$, which implies $b_0 = \ldots = b_n = 0$.

- *They are* **symmetric**,

$$B_i^n(u) = B_{n-i}^n(1-u) \ .$$

- *They have* **roots** *at* 0 *and* 1 *only*,

$$B_i^n(0) = B_{n-i}^n(1) = \begin{cases} 1 & \text{for} \quad i = 0 \\ 0 & \text{for} \quad i > 0 \end{cases} \ .$$

- *They form a* **partition of unity**,

$$\sum_{i=0}^{n} B_i^n(u) = 1 \ , \quad \text{for all} \quad u \in \mathbb{R} \ .$$

- *They are* **positive** *in* $(0, 1)$,

$$B_i^n(u) > 0 \ , \quad \text{for} \quad u \in (0,1) \ .$$

- *They satisfy the* **recursion formula**

$$B_i^{n+1}(u) = uB_{i-1}^n(u) + (1-u)B_i^n(u) \ ,$$

where $B_{-1}^n = B_{n+1}^n = 0$ *and* $B_0^0 = 1$.

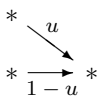
This recursion formula follows directly from the identity

$$\binom{n+1}{i} = \binom{n}{i-1} + \binom{n}{i} \ .$$

Remark 1: The computation of the Bernstein polynomials up to degree n can be arranged in a triangular scheme, as shown below, where the recursion is represented by the "key" on the right:

$$\begin{array}{lccccc} 1 = & B_0^0 & B_0^1 & B_0^2 & \cdots & B_0^n \\ & & B_1^1 & B_1^2 & \cdots & B_1^n \\ & & & B_2^2 & \cdots & B_2^n \\ & & & & \ddots & \vdots \\ & & & & & B_n^n \end{array}$$

key



2.2 Bézier representation

A dimension count shows that the $n+1$ (linearly independent) Bernstein polynomials B_i^n form a basis for all polynomials of degree $\leq n$. Hence, every polynomial curve $\mathbf{b}(u)$ of degree $\leq n$ has a unique nth degree **Bézier representation**

$$\mathbf{b}(u) = \sum_{i=0}^{n} \mathbf{c}_i B_i^n(u) \ .$$

Any affine parameter transformation

$$u = a(1-t) + bt \ , \qquad a \neq b \ ,$$

leaves the degree of the curve $\mathbf{b}$ unchanged. Consequently, $\mathbf{b}(u(t))$ has also an nth degree Bézier representation,

$$\mathbf{b}(u(t)) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t) \ .$$

The coefficients $\mathbf{b}_i$ are elements of $\mathbb{R}^d$ and are called **Bézier points**. They are the vertices of the **Bézier polygon** of $\mathbf{b}(u)$ over the interval $[a, b]$. The parameter t is called the **local** and u the **global parameter** of $\mathbf{b}$, see Figure 2.2.

The properties of Bernstein polynomials summarized in 2.1 are passed on to the Bézier representation of a curve.

- *The* **symmetry** *of the Bernstein polynomials implies that*

$$\mathbf{b}(u) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t) = \sum_{i=0}^{n} \mathbf{b}_{n-i} B_i^n(s) \ ,$$

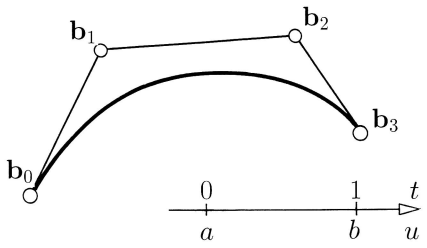


Figure 2.2: A cubic curve segment with its Bézier polygon over $[a, b]$.

where $u = a(1-t) + bt = b(1-s) + as$.

These two sums define the **Bézier representations of b** over $[a, b]$ and $[b, a]$, respectively. Thus, by using the oriented intervals $[a, b]$ and $[b, a]$, we can distinguish the two different parameter orientations of a polynomial curve.

- *For the* **end points** *of the curve segment* $\mathbf{b}[a, b]$*, one has*

$$\mathbf{b}(a) = \mathbf{b}_0 \qquad \textit{and} \qquad \mathbf{b}(b) = \mathbf{b}_n \ .$$

Since the Bernstein polynomials sum to one,

- *any point* $\mathbf{b}(u)$ *is an* **affine combination** *of the Bézier points.*

As a consequence,

- *the Bézier representation is* **affinely invariant***, i.e., given any affine map* Φ*, the image curve* $\Phi(\mathbf{b})$ *has the Bézier points* $\Phi(\mathbf{b}_i)$ *over* $[a, b]$*.*

Since the Bernstein polynomials are non-negative on $[0, 1]$,

- *one has for every* $u \in [a, b]$ *that* $\mathbf{b}(u)$ *is a* **convex combination** *of the* $\mathbf{b}_i$*. Hence, the curve segment* $\mathbf{b}[a, b]$ *lies in the convex hull of its Bézier points.*

This is illustrated in Figure 2.3.

Remark 2: Using the convex hull property separately for each coordinate function of the curve $\mathbf{b}(u)$, a **bounding box** is obtained for the curve segment $\mathbf{b}[a, b]$,

$$\mathbf{b}[a, b] \subset [\min_{i=0}^{n} \mathbf{b}_i, \ \max_{i=0}^{n} \mathbf{b}_i] \ , \qquad u \in [a, b] \ ,$$

as illustrated in Figure 2.4 for a planar curve.

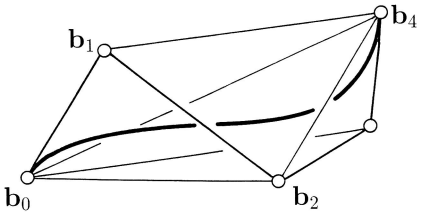


Figure 2.3: The convex hull of a Bézier polygon.

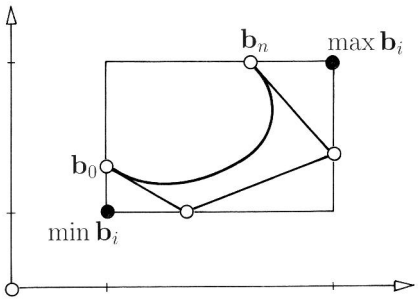


Figure 2.4: Bounding box.

2.3 The de Casteljau algorithm

A curve

$$\mathbf{b}(u) = \sum_{i=0}^{n} \mathbf{b}_i^0 B_i^n(t), \quad \text{with} \quad u = a(1-t) + bt,$$

can be evaluated easily by the so called **de Casteljau algorithm** [Casteljau '59]. Repeatedly using the recurrence relation of the Bernstein polynomials and collecting terms, one obtains

$$\mathbf{b}(u) = \sum_{i=0}^{n} \mathbf{b}_i^0 B_i^n(t) = \sum_{i=0}^{n-1} \mathbf{b}_i^1 B_i^{n-1}(t) = \cdots = \sum_{i=0}^{0} \mathbf{b}_i^n B_i^0(t) = \mathbf{b}_0^n \ ,$$

where

$$\mathbf{b}_i^{k+1} = (1-t)\mathbf{b}_i^k + t\,\mathbf{b}_{i+1}^k \ .$$

Two examples are shown in Figure 2.5, with $t = 0.4$ on the left and $t = 1.4$ on the right side.

The intermediate points $\mathbf{b}_i^k$ of the de Casteljau algorithm can be arranged in a triangular array, where each element is computed according to the "key"

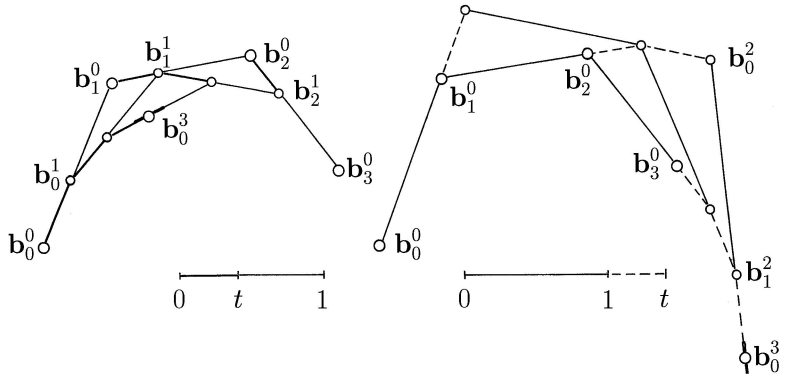
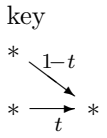


Figure 2.5: The de Casteljau construction.

on the right:

$$\begin{array}{llll} \mathbf{b}_0^0 & & & \\ \mathbf{b}_1^0 & \mathbf{b}_0^1 & & \\ \mathbf{b}_2^0 & \mathbf{b}_1^1 & \mathbf{b}_0^2 & \\ \vdots & & \ddots & \\ \mathbf{b}_n^0 & \mathbf{b}_{n-1}^1 & \mathbf{b}_{n-2}^2 \cdots & \mathbf{b}_0^n \end{array}$$



Remark 3: If t lies in $[0,1]$, then the de Casteljau algorithm consists only of convex combinations, which accounts for the numerical stability of this algorithm.

Remark 4: Horner's scheme is a very effective method to evaluate a polynomial in monomial form. It can also be used for a curve $\mathbf{b}(t) = \sum \mathbf{b}_i B_i^n(t)$ in Bézier form. After writing $\mathbf{b}(t)$ as

$$\mathbf{b}(t) = (1-t)^n \left(\sum_{i=0}^{n} \mathbf{b}_i \binom{n}{i} \left(\frac{t}{1-t} \right)^i \right),$$

one first evaluates the sum in parentheses by Horner's scheme for the value $t/(1-t)$ and then multiplies the result by $(1-t)^n$.

This method fails, if t is close to 1. In this, case one can use the relationship

$$\mathbf{b}(t) = t^n \left(\sum_{i=0}^{n} \mathbf{b}_{n-i} \binom{n}{i} \left(\frac{1-t}{t} \right)^i \right).$$

2.4 Derivatives

The derivative of a Bernstein polynomial of degree n is simple to compute. From the definition of the Bernstein polynomials one gets

$$\frac{d}{dt}B_i^n(t) = n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)) \qquad \text{for} \quad i = 0, \ldots, n \ ,$$

where $B_{-1}^{n-1} = B_n^{n-1} = 0$ as before. Thus, given a curve

$$\mathbf{b}(u) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t) \ , \qquad t = \frac{u-a}{b-a} \ ,$$

one obtains for its derivative $\mathbf{b}'(u)$

$$\frac{d}{du}\mathbf{b}(u) = \frac{n}{b-a}\sum_{i=0}^{n-1} \Delta\mathbf{b}_i B_i^{n-1}(t) \ ,$$

where $\Delta\mathbf{b}_i = \mathbf{b}_{i+1} - \mathbf{b}_i$ This is illustrated in Figure 2.6.

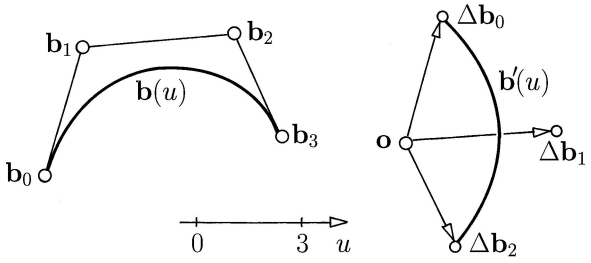


Figure 2.6: Bézier curve and its hodograph (1:3).

If the column $\mathbf{b}(u)$ is viewed as a point, then $\mathbf{b}'(u)$ is a vector. One obtains a point again if $\mathbf{b}'(u)$ is added to a point. In particular, $\mathbf{o} + \mathbf{b}'(u)$ is called the **(first) hodograph** of $\mathbf{b}$.

Applying the derivative formula above repeatedly, one obtains any rth derivative of $\mathbf{b}$,

$$\mathbf{b}^{(r)}(u) = \frac{n!}{(n-r)!(b-a)^r}\sum_{i=0}^{n-r} \Delta^r\mathbf{b}_i B_i^{n-r}(t) \ ,$$

where $\Delta^r\mathbf{b}_i = \Delta^{r-1}\mathbf{b}_{i+1} - \Delta^{r-1}\mathbf{b}_i$ denotes the **rth forward difference** of $\mathbf{b}_i$. As above one obtains the second and further hodographs.

Using the derivative formulas and the endpoint interpolation property of Bézier curves, we obtain a result that was fundamental for Bézier's first

development.

> *The derivatives of* $\mathbf{b}$ *at* $t = 0$ *(or* $t = 1$*) up to order* r *depend only on the first (or last)* $r + 1$ *Bézier points, and vice versa.*

Geometrically, this means that, in general, the **tangents** of $\mathbf{b}$ at $t = 0$ and 1 are spanned by $\mathbf{b}_0, \mathbf{b}_1$ and $\mathbf{b}_{n-1}$, $\mathbf{b}_n$, respectively, and that the **osculating planes** of $\mathbf{b}$ at $t = 0$ and 1 are spanned by $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ and $\mathbf{b}_{n-2}, \mathbf{b}_{n-1}, \mathbf{b}_n$, respectively, and so forth. Figure 2.7 gives an illustration.

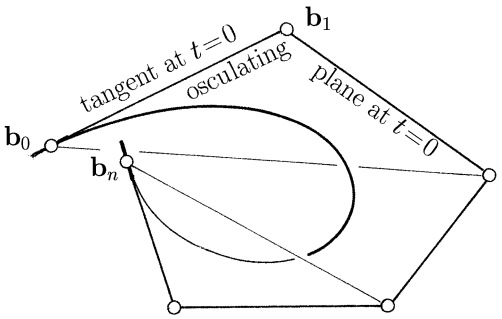


Figure 2.7: Tangents and osculating planes.

Remark 5: Viewing the Bézier polygon of a curve $\mathbf{b}(u) = \sum \mathbf{b}_i B_i^n(t)$, where $u = (1-t)a + t\,b$, as a piecewise linear function $\mathbf{p}(u)$ over $[a, b]$, one gets that

> *the derivative* $\mathbf{p}'(u)$ *of the Bézier polygon consists of the Bézier points of* $\mathbf{b}'(u)$.

This is illustrated in Figure 2.8 for a functional curve.

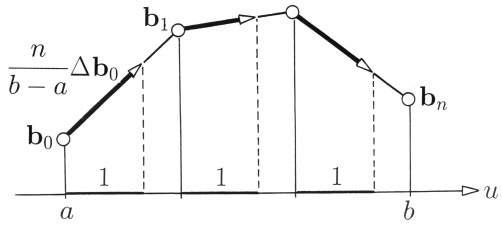


Figure 2.8: The derivative of a Bézier polygon.

2.5 Singular parametrization

Consider a polynomial curve

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

and its derivative

$$\dot{\mathbf{b}}(t) = n \sum_{i=0}^{n-1} \triangle \mathbf{b}_i B_i^{n-1}(t) \ ,$$

where the dot indicates differentiation with respect to the parameter t.

If $\triangle \mathbf{b}_0 = \mathbf{o}$, then $\dot{\mathbf{b}}(t)$ is zero at $t = 0$. However, with the singular reparametrization $t = \sqrt{s}$, one gets

$$\frac{d}{ds}\mathbf{b}(t(0)) = n \cdot \triangle \mathbf{b}_1 \ .$$

Thus, if $\triangle \mathbf{b}_0 = \mathbf{o}$ and $\triangle \mathbf{b}_1 \neq \mathbf{o}$, then the curve $\mathbf{b}(t)$ has a tangent at $t = 0$ that is directed towards $\mathbf{b}_2$, as illustrated in Figure 2.9.

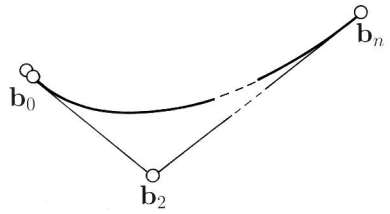


Figure 2.9: Singular parametrization.

Remark 6: If $\triangle \mathbf{b}_0 = \triangle \mathbf{b}_1 = \mathbf{o}$ and $\triangle \mathbf{b}_2 \neq \mathbf{o}$, then the tangent of $\mathbf{b}(t)$ at $t = 0$ is directed towards $\mathbf{b}_2$, etc.

2.6 A tetrahedral algorithm

Computing differences and the affine combinations of de Casteljau's algorithm can be combined. Namely, the rth derivative of a curve

$$\mathbf{b}(u) = \sum \mathbf{b}_i^0 B_i^n(t) \ , \quad t = \frac{u-a}{b-a} \ ,$$

at any u can be computed with de Casteljau's algorithm applied to multiples of the differences $\Delta^k \mathbf{b}_i$. Since the computation of affine combinations of affine combinations is commutative, i.e.,

$$\sum \alpha_i \sum \beta_j \, \mathbf{p}_{ij} = \sum \beta_j \sum \alpha_i \, \mathbf{p}_{ij} \ ,$$

the forward difference operator Δ commutes with the steps of de Casteljau's algorithm.

Hence, one can compute the rth derivative also by first computing $n - r$ steps of de Casteljau's algorithm, then r differencing steps and, finally, a multiplication by the factor $n \cdots (n - r + 1)/(b - a)^r$. Thus, it follows

$$\mathbf{b}^{(r)}(u) = \frac{n \cdots (n - r + 1)}{(b - a)^r} \quad \Delta^r \mathbf{b}_0^{n-r} \ ,$$

In particular, this formula says that the tangent and osculating plane of $\mathbf{b}$ at u are spanned by $\mathbf{b}_0^{n-1}, \mathbf{b}_1^{n-1}$ and $\mathbf{b}_0^{n-2}, \mathbf{b}_1^{n-2}, \mathbf{b}_2^{n-2}$, respectively, as illustrated in Figure 2.10 for a cubic.

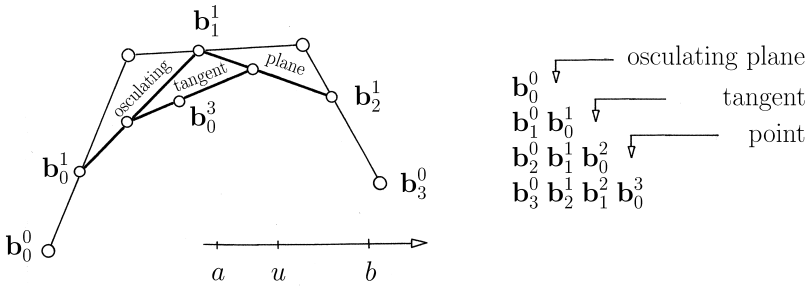


Figure 2.10: Osculating plane and tangent and the de Casteljau scheme.

Computing the points $\Delta^r \mathbf{b}_0^{n-k}$ for all k by successive de Casteljau and differencing steps, one also gets the intermediate points $\Delta^k \mathbf{b}_i^j$, $i + j + k \leq n$. All these points can be arranged conveniently in a tetrahedral array, as illustrated in Figure 2.11 for $n = 2$, where the key represents the recursion

$$\mathbf{c} = \mathbf{a}(1 - t) + \mathbf{b}t \quad \text{and} \quad \mathbf{d} = \mathbf{b} - \mathbf{a} \ .$$

This is not the only possible way to compute the tetrahedral array. Eliminating $\mathbf{a}$ or $\mathbf{b}$, one obtains

$$\mathbf{c} = \mathbf{b} + \mathbf{d}(t - 1) \quad \text{and} \quad \mathbf{c} = \mathbf{a} + \mathbf{d}t \ ,$$

respectively.

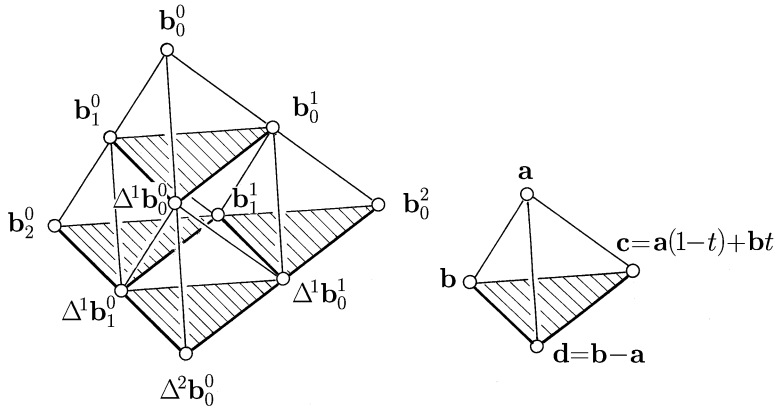


Figure 2.11: Tetrahedral algorithm.

When we use one of these rules, instead of the differencing step, it suffices to compute only the points of the two triangular schemes given by the points on the left and bottom (or right) side of the tetrahedron.

Remark 7: It should be noted that differencing is not a numerically stable process, in general. Consequently, computing derivatives is not a numerically stable process either.

2.7 Integration

The integral of a polynomial curve in Bézier representation

$$\mathbf{b}(u) \;=\; \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t), \quad t = \frac{u-a}{b-a} \;\;,$$

has the Bézier representation

$$\mathbf{c}(u) \;=\; \int \mathbf{b}(u)du \;=\; \sum_{i=0}^{n+1} \mathbf{c}_i B_i^{n+1}(t) \;\;,$$

where

$$\begin{aligned} \mathbf{c}_i &= \mathbf{c}_{i-1} + \frac{b-a}{n+1}\;\mathbf{b}_{i-1} \\ &= \mathbf{c}_0 + \frac{b-a}{n+1}\;(\mathbf{b}_0 + \cdots + \mathbf{b}_{i-1}) \;\;, \quad i = n+1, \ldots, 1 \;\;, \end{aligned}$$

and $\mathbf{c}_0$ is an arbitrary integration constant. This is verified by differentiating $\mathbf{c}(u)$.

As a consequence of the integration formula and the endpoint interpolation property of the Bézier representation, one obtains

$$\int_a^b \mathbf{b}(u)du \;=\; \frac{b-a}{n+1}(\mathbf{b}_0 + \cdots + \mathbf{b}_n)$$

and, in particular, independently of i,

$$\int_0^1 B_i^n(t)dt \;=\; \frac{1}{n+1} \;\;.$$

2.8 Conversion to Bézier representation

Some older CAD data formats represent curves by monomials. Thus, data conversion between different CAD systems is an application where it is necessary to convert the monomial to the Bézier representation and vice versa. Let

$$\mathbf{b}(t) = \sum_{i=0}^{n} \mathbf{a}_i \binom{n}{i} t^i$$

be a curve in monomial form with binomial factors as in the Bézier representation. Since

$$\begin{aligned}
\binom{n}{i} t^i (1-t+t)^{n-i} &= \sum_{k=0}^{n-i} \binom{n}{i} \binom{n-i}{n-i-k} t^{i+k} (1-t)^{n-i-k} \\
&= \sum_{k=0}^{n-i} \binom{i+k}{i} B_{i+k}^n \\
&= \sum_{j=0}^{n} \binom{j}{i} B_j^n \;\;,
\end{aligned}$$

one obtains the conversion formula

$$\mathbf{b}(t) = \sum_{j=0}^{n} \mathbf{b}_j B_j^n(t) \;\;,$$

where

$$\mathbf{b}_j = \sum_{i=0}^{n} \binom{j}{i} \mathbf{a}_i$$

and $\binom{j}{i} = 0$ for $j < i$.

The formula for the conversion to monomial form can be derived similarly by multiplying out the Bernstein polynomials, see Problem 4. In Section 2.9, we present a different derivation of it.

Remark 8: If $\mathbf{a}_2 = \cdots = \mathbf{a}_n = \mathbf{o}$ and $\mathbf{a}_1 \neq \mathbf{o}$, then $\mathbf{b}(\mathrm{t})$ is a linear polynomial represented over $[0, 1]$ by the Bézier points

$$\mathbf{b}_j = \mathbf{a}_0 + j\mathbf{a}_1 \ ,$$

as illustrated in Figure 2.12.

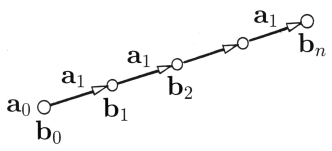


Figure 2.12: Equidistant Bézier points on a line.

Remark 9: Conversely, if the $n + 1$ Bézier points $\mathbf{b}_i$ lie equidistantly on a line, then $\mathbf{b}(\mathrm{t})$ is a linear polynomial, which can be written as

$$\mathbf{b}(t) = (1 - t)\mathbf{b}_0 + t\mathbf{b}_n \ .$$

This property is referred to as the **linear precision** of the Bézier representation.

Remark 10: As a consequence of Remark 9, the functional curve

$$\mathbf{b}(t) = \begin{bmatrix} t \\ b(t) \end{bmatrix}, \qquad b(t) = \sum b_i B_i^n(t) \ ,$$

has the Bézier points $[i/n \ \ b_i]^t$, as illustrated in Figure 2.13. The coefficients b_i are referred to as the **Bézier ordinates** of $b(t)$, and i/n as the **Bézier abscissae**.

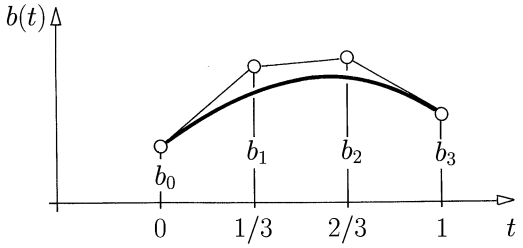


Figure 2.13: Bézier representation of a functional curve.

2.9 Conversion to monomial form

Given a polynomial curve in Bézier representation,

$$\mathbf{b}(u) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n \left(\frac{u-a}{b-a} \right) \ ,$$

one can obtain its monomial form simply by a Taylor expansion,

$$\begin{aligned} \mathbf{b}(u) &= \sum_{i=0}^{n} \mathbf{b}^{(i)}(a) \frac{(u-a)^i}{i!} \\ &= \sum_{i=0}^{n} \binom{n}{i} \Delta^i \mathbf{b}_0 \frac{(u-a)^i}{(b-a)^i} \ . \end{aligned}$$

Since $\Delta^i \mathbf{b}_0 = \sum_{k=0}^{i} \binom{i}{k} (-1)^{i-k} \mathbf{b}_k$, see Problem 3, one can write this explicitly as

$$\mathbf{b}(u) = \sum_{i=0}^{n} \sum_{k=0}^{i} (-1)^{i-k} \binom{n}{i} \binom{i}{k} \mathbf{b}_k \ t^i \ .$$

Remark 11: Using the tetrahedral algorithm in 2.6, one can compute the Taylor expansion at u,

$$\mathbf{b}(u+h) = \sum_{i=0}^{n} \frac{1}{(b-a)^i} \Delta^i \mathbf{b}_0^{n-i} \binom{n}{i} h^i \ .$$

2.10 Problems

1 Show that the Bernstein polynomial $B_i^n(t)$ has only one maximum in $[0,1]$, namely at $t = i/n$.

2 The **Bernstein operator** $\mathcal{B}$ assigns to a function f on $[0,1]$ the polynomial

$$\mathcal{B}[f] = \sum_{i=0}^{n} f(i/n) B_i^n(t) \ .$$

If f is a polynomial of degree $m \leq n$, then $\mathcal{B}[f]$ is also a polynomial of degree m, see also Problem 2 in 3.13. Show that this is true.

3 Show that

$$\Delta^i \mathbf{b}_0 = \sum_{k=0}^{i} \binom{i}{k} (-1)^{i-k} \mathbf{b}_k \ .$$

4 Derive the conversion formula in 2.9 to monomial form by elementary algebraic manipulations similar to what is done in 2.8.

5 Show that

$$n \ldots (n-k)\, t^{k+1} = \sum_{i=0}^{n} i \ldots (i-k)\, B_i^n(t) \ .$$

6 Show that a planar cubic $\mathbf{b}(t)$ has a **cusp** at $t = 0$, i.e., a point where it changes its direction, if $\dot{\mathbf{b}}(0) = \mathbf{o}$ and both coordinates of $\ddot{\mathbf{b}}(0)$ are non-zero. (The dots indicate differentiation with respect to t.)

7 Show that a planar cubic $\mathbf{b}(t) = \sum_{i=0}^{3} \mathbf{b}_i B_i^3(t)$ has a cusp if $\mathbf{b}_3$ lies on the parabola $\mathbf{p}(t) = (\mathbf{b}_0 + \mathbf{b}_1 - \mathbf{b}_2)B_0^2(t) + \mathbf{b}_1 B_1^2(t) + \mathbf{b}_2 B_2^2(t)$ [Pottmann & DeRose '91].

8 For which choices of $\mathbf{b}_3$ does the cubic $\mathbf{b}(t)$ have a loop?

9 Let $\sum_{i=0}^{n} \mathbf{a}_i \binom{n}{i} t^i = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t)$. Then, in matrix notation, one obtains

$$[\mathbf{a}_0 \ \ldots \ \mathbf{a}_n] = [\mathbf{b}_0 \ \ldots \ \mathbf{b}_n]\, \Delta \ ,$$

where $\Delta = \left[(-1)^{j-i}\binom{j}{i}\right]$ and $\Delta^{-1} = \left[\binom{i}{j}\right]$. The matrices Δ and Δ^{-1} are upper-triangular matrices.

5 B-spline representation

5.1 Splines — 5.2 B-splines — 5.3 A recursive definition of B-splines — 5.4 The de Boor algorithm — 5.5 The main theorem — 5.6 Derivatives — 5.7 B-spline properties — 5.8 Conversion to B-spline form — 5.9 The complete de Boor algorithm — 5.10 Conversions between Bézier and B-spline representations — 5.11 B-splines as divided differences — 5.12 Problems

Splines are piecewise polynomial curves that are differentiable up to a prescribed order. The simplest example is a piecewise linear C^0 spline, i.e., a polygonal curve. Other examples are the piecewise cubic C^1 splines, as constructed in 4.5.

The name spline is derived from elastic beams, so-called splines, used by draftsmen to lay out broad sweeping curves in ship design. Held in place by a number of heavy weights, these physical splines assume a shape that minimizes the strain energy. This property is approximately shared by the mathematical cubic C^2 splines.

5.1 Splines

A curve $\mathbf{s}(u)$ is called a **spline of degree** n with the **knots** $a_0, \ldots, a_m$, where $a_i \leq a_{i+1}$ and $a_i < a_{i+n+1}$ for all possible i, if

> $\mathbf{s}(u)$ *is* $n - r$ *times differentiable at any* r*-fold knot*[1]*, and* $\mathbf{s}(u)$ *is a polynomial of degree* $\leq n$ *over each knot interval* $[a_i, a_{i+1}]$*, for* $i = 0, \ldots, m-1$.

It is also common to refer to a spline of degree n as a **spline of order** $n+1$. Figures 5.1 and 5.2 show examples of splines with simple knots obtained by

[1] *A knot* a_{i+1} *is called* r*-fold if* $a_i < a_{i+1} = \cdots = a_{i+r} < a_{i+r+1}$.

Stärk's construction, see Figures 3.8 and 3.9. The inner and end Bézier points are marked by hollow and solid dots, respectively.

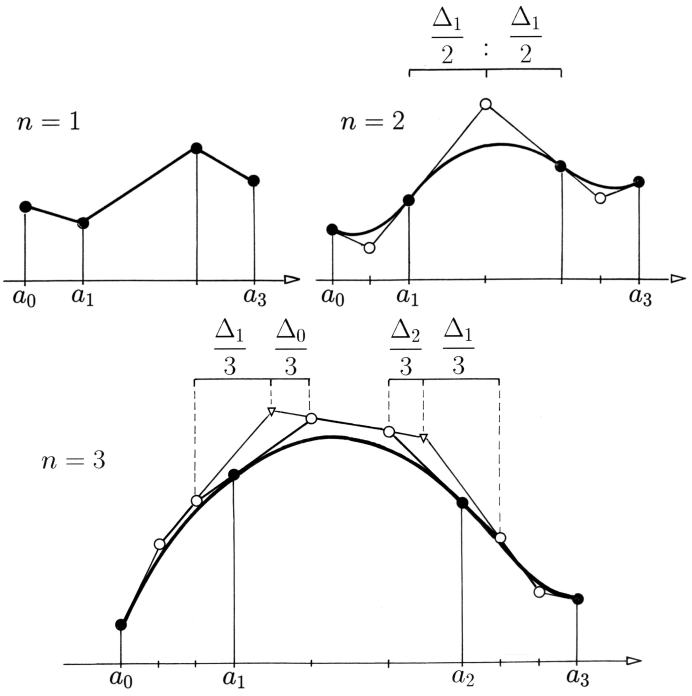


Figure 5.1: Spline functions of degree 1, 2 and 3.

5.2 B-splines

As with the Bézier representation of polynomial curves, it is desirable to write a spline $\mathbf{s}(u)$ as an affine combination of some control points $\mathbf{c}_i$, namely

$$\mathbf{s}(u) = \sum \mathbf{c}_i N_i^n(u) \ ,$$

where the $N_i^n(u)$ are basis spline functions with minimal support and certain continuity properties. Schoenberg introduced the name B-splines for these functions [Schoenberg '67]. Their Bézier polygons can be constructed by Stärk's theorem.

Figure 5.3 shows a piecewise cubic C^2 B-spline. Stärk's theorem is only needed for the Bézier ordinates, while the abscissae are given by Remark 8 in 2.3.

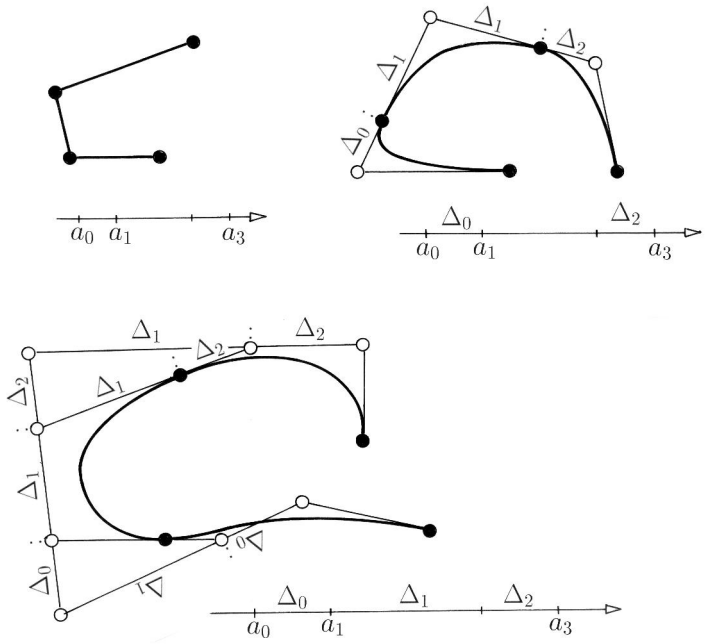


Figure 5.2: Parametric splines of degree 1, 2, and 3.

For higher degree this construction, albeit possible, becomes much less obvious and more complicated, see [Prautzsch '89]. Therefore, we use a recurrence relation, which was found independently by de Boor and Mansfield [de Boor '72] in 1970 and Cox [Cox '72] in 1971. We define B-splines from that relation and derive all important properties from that relation.

5.3 A recursive definition of B-splines

To define B-splines, let (a_i) be a, for simplicity, biinfinite and strictly increasing sequence of knots, which means $a_i < a_{i+1}$, for all i. We define the **B-splines** N_i^n with these knots by the recursion formula

$$N_i^0(u) = \begin{cases} 1 & \text{if} \quad u \in [a_i, a_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$

and

$$N_i^n(u) = \alpha_i^{n-1} N_i^{n-1}(u) + (1 - \alpha_{i+1}^{n-1}) N_{i+1}^{n-1}(u) \ ,$$

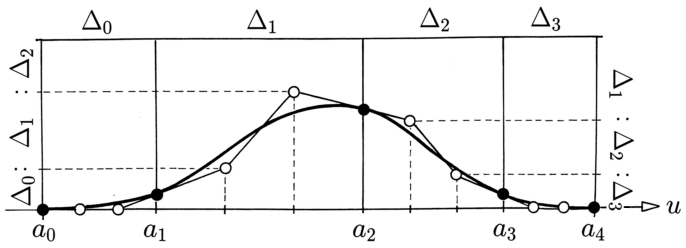


Figure 5.3: Bézier points of the B-spline $N_0^3(u)$.

where

$$\alpha_i^{n-1} = (u - a_i)/(a_{i+n} - a_i)$$

is the local parameter with respect to the support of N_i^{n-1}. Figure 5.4 shows B-splines of degree 0, 1 and 2.

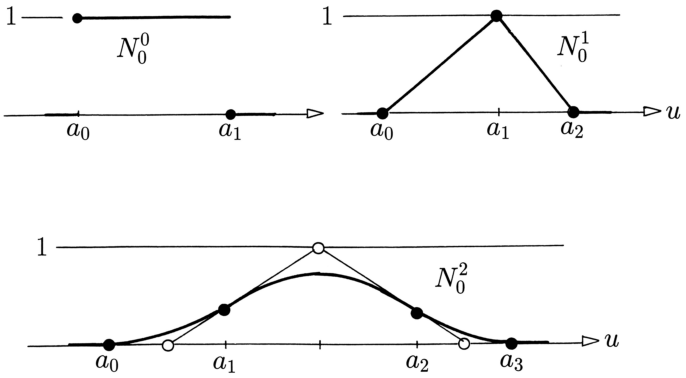


Figure 5.4: B-splines of degree 0, 1 and 2.

In case of multiple knots, the B-splines $N_i^n(u)$ are defined by the same recursion formula and the convention

$$N_i^{r-1} = N_i^{r-1}/(a_{i+r} - a_i) = 0 \quad \text{if} \quad a_i = a_{i+r} \ .$$

Figure 5.5 shows B-splines with multiple knots.

From the definition above, the following properties of B-splines are evident.

- $N_i^n(u)$ *is piecewise polynomial of degree* n,
- $N_i^n(u)$ *is positive in* (a_i, a_{i+n+1}),
- $N_i^n(u)$ *is zero outside of* $[a_i, a_{i+n+1}]$,

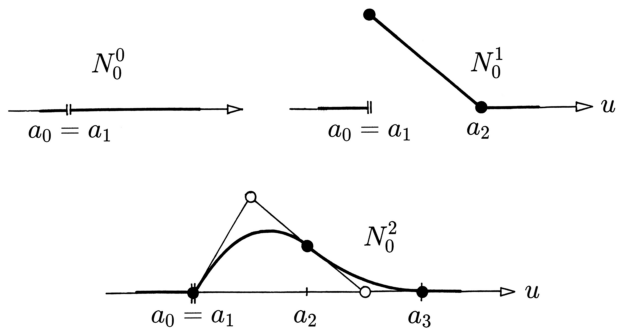


Figure 5.5: Some B-splines with multiple knots.

- $N_i^n(u)$ *is right side continuous.*

In Sections 5.5 and 5.6, we will see that the B-splines are $n - r$ times differentiable at r-fold knots, and that every spline is a unique combination of B-splines.

Remark 1: If, in particular, $a_1 = \ldots = a_n = 0$ and $a_{n+1} = \ldots = a_{2n} = 1$, then the above recursion formula for $N_0^n, \ldots, N_n^n$ and $u \in [0, 1)$ coincides with the recursion formula of the Bernstein polynomials. Hence, we have

$$N_i^n(u) = B_i^n(u) \quad \text{for} \quad i = 0, \ldots, n \quad \text{and} \quad u \in [0, 1) \ .$$

5.4 The de Boor algorithm

Consider a linear combination

$$\mathbf{s}(u) = \sum_i \mathbf{c}_i^0 N_i^n(u)$$

of the nth degree B-splines over some knot sequence (a_i). Since any finite sum can be converted to a formally biinfinite sum by adjunction of zero terms, we assume, without loss of generality, that the knot sequence and, hence, the sum above are biinfinite. Since the $N_i^n(u)$ have local supports, this sum is actually finite for any given u. In particular, let $u \in [a_n, a_{n+1})$, then $\mathbf{s}(u)$ can be written as

$$\mathbf{s}(u) = \sum_{i=0}^{n} \mathbf{c}_i^0 N_i^n(u) \ .$$

Using the B-spline recursion repeatedly and collecting terms, one obtains

$$\begin{aligned} \mathbf{s}(u) &= \sum_{i=1}^{n} \mathbf{c}_i^1 N_i^{n-1}(u) \\ &\vdots \\ &= \sum_{i=n}^{n} \mathbf{c}_i^n N_i^0(u) = \mathbf{c}_n^n \ , \end{aligned}$$

where the $\mathbf{c}_i^r$ are given by the affine combinations

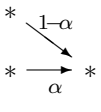
$$\mathbf{c}_i^r = (1-\alpha)\mathbf{c}_{i-1}^{r-1} + \alpha\,\mathbf{c}_i^{r-1} \ , \quad \alpha = \alpha_i^{n-r} = \frac{u - a_i}{a_{i+n+1-r} - a_i} \ .$$

Note that $\alpha \in [0,1]$ since $u \in [a_n, a_{n+1})$, i.e., the affine combinations are actually convex.

This algorithm was developed by de Boor in 1972 [de Boor '72]. The points $\mathbf{c}_i^r$ are conveniently arranged in a triangular array, as illustrated below, where the recursion formula is represented by the key on the right,

$$\begin{array}{llllll} \mathbf{c}_0^0 & & & & \\ \mathbf{c}_1^0 & \mathbf{c}_1^1 & & & \\ \mathbf{c}_2^0 & \mathbf{c}_2^1 & \mathbf{c}_2^2 & & \\ \vdots & & & \ddots & \\ \mathbf{c}_n^0 & \mathbf{c}_n^1 & \mathbf{c}_n^2 & \cdots & \mathbf{c}_n^n \end{array}$$

key



(α depending on key position)

An important consequence of de Boor's algorithm is that the spline $\mathbf{s}(u)$ over each knot interval is an affine and actually convex combination of $n+1$ consecutive coefficients $\mathbf{c}_i$. Hence, if the $\mathbf{c}_i$ represent points of some affine space, then $\mathbf{s}(u)$ is also a point. For this reason, we refer to the $\mathbf{c}_i$ as the **control points** of $\mathbf{s}(u)$.

Further, the spline lies in the affine hull of its control points, which implies that

$$\sum_{i=0}^{n} 1 \cdot N_i^n(u) = 1 \quad \text{for} \quad u \in [a_n, a_{n+1}) \ ,$$

i.e., the B-splines form a partition of unity. Figure 5.6 illustrates the geometric interpretation of de Boor's algorithm, which was first given in [Gordon & Riesenfeld '74].

Remark 2: For arbitrary $u \in \mathbb{R}$, de Boor's algorithm applied, as described above, to $\mathbf{c}_0^0, \ldots, \mathbf{c}_n^0$ does not compute $\mathbf{s}(u)$ in general, but the polynomial $\mathbf{s}_n(u)$ that agrees with $\mathbf{s}(u)$ over $[a_n, a_{n+1})$.

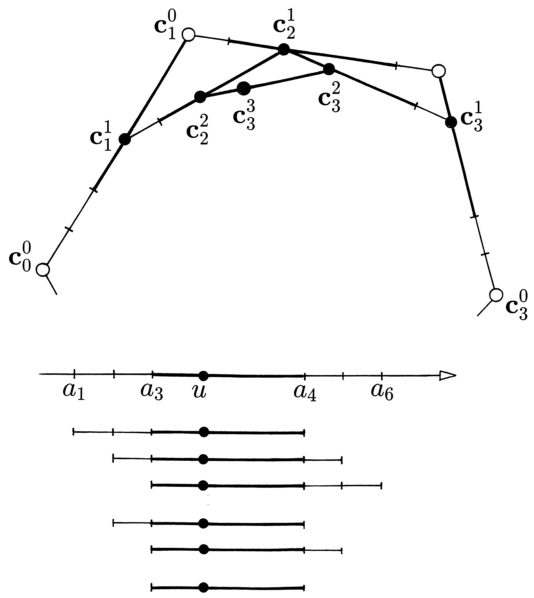


Figure 5.6: The convex combinations of de Boor's algorithm for $n = 3$.

5.5 The main theorem in its general form

Symmetric polynomials will help to see de Boor's algorithm in a wider context. As before, let

$$\mathbf{s}(u) = \sum_i \mathbf{c}_i N_i^n(u)$$

be an nth degree spline with knots a_i, and let $\mathbf{s}_i[u_1 \ldots u_n]$ be the polar form that agrees on its diagonal with $\mathbf{s}(u)$ over $[a_i, a_{i+1})$. Then we have the following general form of the **main theorem** 3.2.

As illustrated in Figure 5.7, the control points of $\mathbf{s}$ *are given by*

$$\mathbf{c}_i = \mathbf{s}_j[a_{i+1} \ldots a_{i+n}] \ , \qquad i = j-n, \ldots, j \ .$$

For a proof, let

$$\mathbf{p}_i^r = \mathbf{s}_j[a_{i+1} \ldots a_{i+n-r}\, u \,\overset{r}{\ldots}\, u]$$

and

$$u = (1-\alpha)a_i + \alpha a_{i+n-r+1} \ .$$

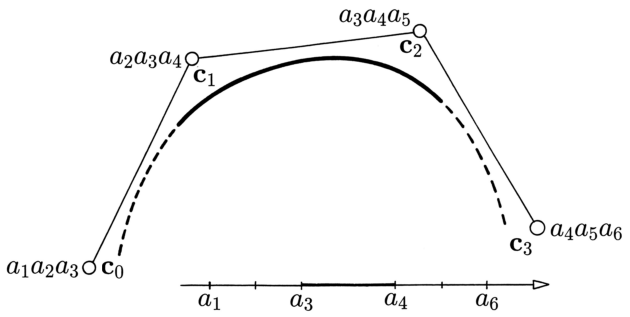


Figure 5.7: The main theorem for a cubic spline.

Then, since $\mathbf{s}_j$ is multiaffine and symmetric, it follows that

$$\mathbf{p}_i^r = (1-\alpha)\mathbf{p}_{i-1}^{r-1} + \alpha\,\mathbf{p}_i^{r-1} \quad , \quad \alpha = \alpha_i^{n-r} = \frac{u - a_i}{a_{i+n-r+1} - a_i} \quad ,$$

and, in particular,

$$\mathbf{p}_i^0 = \mathbf{s}_j[a_{i+1} \,\ldots\, a_{i+n}] \quad \text{and} \quad \mathbf{p}_j^n = \mathbf{s}_j(u) \quad .$$

For $u \in [a_j, a_{j+1})$, this construction agrees with de Boor's algorithm and can be used to compute any polynomial $\mathbf{s}_j[u \,\ldots\, u]$ of degree n. Hence, every polynomial of degree n can be written over $[a_j, a_{j+1})$ as a linear combination of the B-splines $N_{j-n}^n(u), \ldots, N_j^n(u)$. A dimension count shows that this linear combination is unique whence the assertion follows. $\diamond$

Remark 3: In the proof, we showed that, over $[a_n, a_{n+1}]$, the B-splines $N_0^n(u), \ldots, N_n^n(u)$ form a basis for the space of all polynomial up to degree n. This result is due to [Curry & Schoenberg '66].

Remark 4: The spline "segment" $\mathbf{s}_i$ determines the control points $\mathbf{c}_{i-n}, \ldots, \mathbf{c}_i$. Conversely, every point $\mathbf{c}_j$ is determined by any "segment" $\mathbf{s}_j, \ldots, \mathbf{s}_{j+n}$, i.e.,

$$\mathbf{c}_i = \mathbf{s}_i[a_{i+1} \,\ldots\, a_{i+n}] = \cdots = \mathbf{s}_{i+n}[a_{i+1} \,\ldots\, a_{i+n}] \quad .$$

Remark 5: The proof above shows that the symmetric polynomial $\mathbf{s}_n[u_1 \,\ldots\, u_n]$ can be computed by a generalization of de Boor's algorithm. All one has to do, is to replace $\alpha = \alpha(u)$ in the recursion formula by

$$\alpha(u_r) = \frac{u_r - a_i}{a_{i+n-r+1} - a_i} \quad .$$

If m of the n variables $u_1, \ldots, u_n$ are knots, then only $n-m$ recursion steps are needed to compute $\mathbf{s}_j[u_1 \ldots u_n]$. The computation can be arranged in a triangular array consisting of $1+2+\cdots+(n-m+1)$ points.

5.6 Derivatives and smoothness

Because of the basis property of B-splines, see Remark 3, the derivative of the polynomial spline segment $\mathbf{s}_n$ can be written as

$$\mathbf{s}_n'(u) = \sum_{i=1}^{n} \mathbf{d}_i N_i^{n-1}(u) \ , \quad u \in [a_n, a_{n+1}) \ ,$$

where the unknown vectors $\mathbf{d}_i$ can easily be expressed in terms of the $\mathbf{c}_i$. Let $\mathbf{s}_n'[u_2 \ \ldots \ u_n]$ be the symmetric polynomial of $\mathbf{s}_n'(u)$, and let the direction $\Delta = a_{i+n} - a_i$ be given by the support of the B-spline $N_i^{n-1}(u)$. Then, it follows from the main theorem and 3.9 that

$$\begin{aligned} \mathbf{d}_i &= \mathbf{s}_n'[a_{i+1} \ \ldots \ a_{i+n-1}] \\ &= \frac{n}{\Delta}\mathbf{s}_n[\Delta \ a_{i+1} \ \ldots \ a_{i+n-1}] \\ &= \frac{n}{a_{i+n}-a_i}(\mathbf{c}_i - \mathbf{c}_{i-1}) \ . \end{aligned}$$

Since the $\mathbf{d}_i$ do not depend on the knot interval $[a_n, a_{n+1})$, the derivative of the spline $\mathbf{s}$ can be written for all $u \in \mathbb{R}$ as

$$\mathbf{s}'(u) = \sum_i \frac{n}{a_{i+n}-a_i} \nabla \mathbf{c}_i N_i^{n-1}(u) \ , \tag{1}$$

where $\nabla \mathbf{c}_i = \mathbf{c}_i - \mathbf{c}_{i-1}$ denotes the **first backward difference**.

One can differentiate further so as to obtain the B-spline representation of higher derivatives. This is also useful in showing that the B-splines have the desired smoothness properties:

An nth degree spline $\mathbf{s}$ is continuous at any n-fold knot. Namely if $a_0 < a_1 = \cdots = a_n < a_{n+1}$, then it follows from Remark 4 in 5.5 that

$$\begin{aligned} \mathbf{s}_0(a_1) &= \mathbf{s}_0[a_1 \ \ldots \ a_n] = \mathbf{c}_0 \\ &= \mathbf{s}_n[a_1 \ \ldots \ a_n] \\ &= \mathbf{s}_n(a_n) \ . \end{aligned}$$

Thus, if a_i is an r-fold knot, then the $(n-r)$th derivative of $\mathbf{s}$ is continuous at a_i. In other words,

a B-spline satisfies the smoothness criteria of a spline given in 5.1.

5.7 B-spline properties

We summarize the basic properties of B-splines.

- The B-splines of degree n with a given knot sequence that do not vanish over some knot interval are **linearly independent** over this interval.
- A dimension count shows that the B-splines $N_0^n, \ldots, N_m^n$ with the knots $a_0, \ldots, a_{m+n+1}$ form a **basis** for all splines of degree n with support $[a_0, a_{m+n+1}]$ and the same knots.
- Similarly, the B-splines $N_0^n, \ldots, N_m^n$ over the knots $a_0, \ldots, a_{m+n+1}$ restricted to the interval $[a_n, a_{m+1})$ form a **basis** for all splines of degree n restricted to the same interval.
- The B-splines of degree n form a **partition of unity**, i.e.,

$$\sum_{i=0}^{m} N_i^n(u) = 1, \quad \text{for} \quad u \in [a_n, a_{m+1}) \ .$$

- A spline $\mathbf{s}[a_n, a_{m+1}]$ of degree n with n**-fold end knots**,

$$(a_0 =)a_1 = \ \ldots \ = a_n \quad \text{and} \quad a_{m+1} = \ \ldots \ = a_{m+n}(= a_{m+n+1})$$

  has the same end points and end tangents as its control polygon.
- The **end knots** a_0 and a_{m+n+1} **have no influence** on N_0^n and N_m^n over the interval $[a_n, a_{m+1}]$.
- The B-splines are **positive** over the interior of their support,

$$N_i^n(u) > 0 \quad \text{for} \quad u \in (a_i, a_{i+n+1}) \ .$$

- The B-splines have **compact support**,

$$\text{supp} N_i^n = [a_i, a_{i+n+1}] \ .$$

- The B-splines satisfy the **de Boor, Mansfield, Cox recursion formula**

$$N_i^n(u) = \alpha_i^{n-1} N_i^{n-1}(u) + (1 - \alpha_{i+1}^{n-1}) N_{i+1}^{n-1}(u) \ ,$$

  where $\alpha_i^{n-1} = (u - a_i)/(a_{i+n} - a_i)$ represents the local parameter over the support of N_i^{n-1}.
- The **derivative** of a single B-spline is given by

$$\frac{d}{du} N_i^n(u) = \frac{n}{u_{i+n} - u_i} N_i^{n-1}(u) - \frac{n}{u_{i+n+1} - u_{i+1}} N_{i+1}^{n-1}(u) \ .$$

- The B-spline representation of a spline curve is **invariant under affine maps**.
- Any segment $\mathbf{s}_j[a_j, a_{j+1})$ of an nth degree spline lies in the **convex hull** of its $n+1$ control points $\mathbf{c}_{j-n}, \ldots, \mathbf{c}_j$.
- A **degree elevation** formula is given in 6.5.

5.8 Conversion to B-spline form

Since any polynomial of degree n can be viewed as a spline of degree n or higher with an arbitrary sequence of knots, one can express the monomials as linear combinations of B-splines over any knot sequence (a_i). To do so, recall from 3.1 that the monomials $A_j^n(u) = \binom{n}{j} u^j$ have the polar forms

$$A_j^n[u_1 \ \ldots \ u_n] = \sum_{i< \ldots <k} u_i \ldots \overset{j}{\ldots} u_k \ .$$

Thus, it follows from the main theorem 5.5 that

$$A_j^n(u) = \sum_i \alpha_{ji} N_i^n(u) \ ,$$

where $\alpha_{ji} = A_j^n[a_{i+1} \ \ldots \ a_{i+n}]$, and, consequently,

$$\begin{aligned} \mathbf{a}(u) &= \mathbf{a}_0 A_0^n(u) + \cdots + \mathbf{a}_n A_n^n(u) \\ &= \sum_i (\mathbf{a}_0 \alpha_{0i} + \cdots \mathbf{a}_n \alpha_{ni}) N_i^n(u) \ , \end{aligned}$$

which generalizes Marsden's identity given in Problem 4 below. In particular, one obtains

$$\begin{aligned} u &= \frac{1}{n} A_1^n(u) \\ &= \sum_i \gamma_i N_i^n(u) \ , \end{aligned}$$

where $\gamma_i = \alpha_{1i}/n = (a_{i+1} + \cdots + a_{i+n})/n$. The γ_i are the so-called **Greville abscissae** [Greville '67].

Remark 6: The Greville abscissae show up naturally in the control points of the graph of a spline function

$$s(u) = \sum_i c_i N_i^n \ .$$

Namely, the graph $\mathbf{s}(u) = [u \ s(u)]^t$ has the control points $\mathbf{c}_i = [\gamma_i \ c_i]^t$. Fig-

ure 5.8 shows the example $s(u) = N_2^3(u)$. Other examples are shown in Figure 5.1.

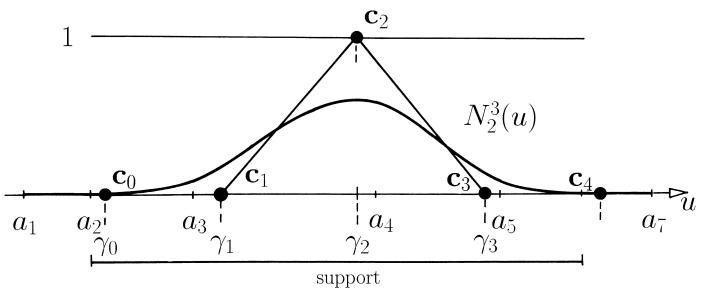


Figure 5.8: Control points of the cubic B-spline $N_2^3(u)$.

5.9 The complete de Boor algorithm

The Taylor expansion of a polynomial spline segment

$$\mathbf{s}_n(u) = \sum_{i=0}^{n} \mathbf{c}_i N_i^n(u) \ , \quad u \in [a_n, a_{n+1}) \ ,$$

can be computed at any $u \in \mathbb{R}$ following the ideas presented in 2.6 for Bézier curves.

Let $\mathbf{s}_n[u_1 \ \ldots \ u_n]$ be the polar form of $\mathbf{s}_n$ and consider the points and vectors

$$\mathbf{c}_{r,i,k} = \mathbf{s}_n[\varepsilon \ \overset{r}{\ldots} \ \varepsilon \, a_i \ \ldots \ a_{i+n-r-k} \, u \ \overset{k}{\ldots} \ u],$$

where ε denotes the direction $1 - 0$, for $i = r + k, \ldots, n$. It follows that

$$\frac{d^r}{du^r}\mathbf{s}_n(u) \quad = \quad \frac{n!}{(n-r)!}\mathbf{c}_{r,n,n-r} \ ,$$

and the Taylor expansion is given by

$$\mathbf{s}_n(u+h) = \sum_{r=0}^{n} \mathbf{c}_{r,n,n-r} \binom{n}{r} h^r \ .$$

The points and vectors $\mathbf{c}_{rik}$ can again be arranged conveniently in a tetrahedral array, see Figure 5.9, where $n = 2$ and $\varepsilon 4$ stands for $\mathbf{s}_n[\varepsilon \, a_4]$, etc.

This array, first considered by Sablonniere in 1978 [Sablonniere '78], is com-

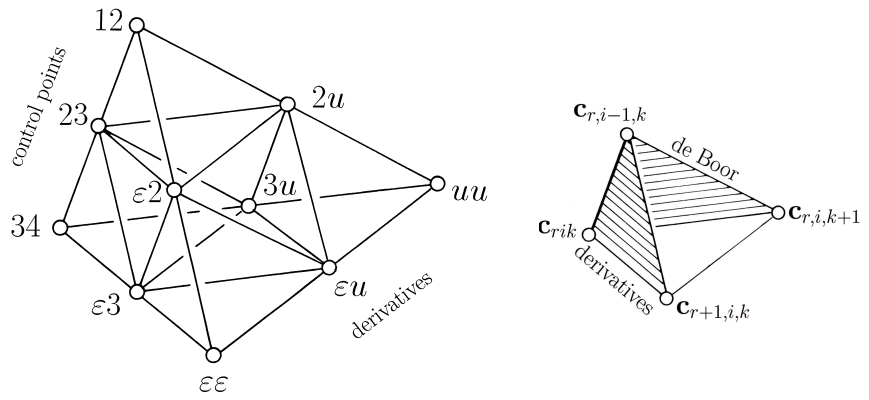


Figure 5.9: The complete de Boor algorithm.

posed of $\binom{n+2}{3}$ subtetrahedra and contains the given control points $\mathbf{c}_i = \mathbf{c}_{0,i,0}$ on the "left" edge and the multiples

$$\frac{(n-r)!}{n!}\,\mathbf{s}_n^{(r)}(u)$$

of the derivatives on the opposite edge.

Any two of the four points of a subtetrahedron can be computed from the two other points. The computation rules follow directly from the properties of multiaffine symmetric polynomials. For example, one has

$$\mathbf{c}_{r+1,i,k} = \frac{1}{a_{i+n-r-k} - a_i}(\mathbf{c}_{r,i,k} - \mathbf{c}_{r,i-1,k})$$

on the "left" face,

$$\mathbf{c}_{r,i,k+1} = (1-\alpha)\mathbf{c}_{r,i-1,k} + \alpha\,\mathbf{c}_{r,i,k}\ , \qquad \alpha = \frac{u - a_i}{a_{i+n-r-k} - a_i}\ ,$$

on the "rear" face,

$$\mathbf{c}_{r,i,k} = \mathbf{c}_{r,i,k+1} + (a_{i+n-r-k} - u)\mathbf{c}_{r+1,i,k}\ , \tag{2}$$

on the "bottom" face, and

$$\mathbf{c}_{r,i-1,k} = \mathbf{c}_{r,i,k+1} + (a_i - u)\mathbf{c}_{r+1,i,k} \tag{3}$$

on the "top" face.

Remark 7: One can use the above formulae to convert a B-spline representation to monomial representation and vice versa.

Remark 8: In order to obtain the derivatives from the control points or vice versa, it suffices to compute, for example, only the left and top faces of the tetrahedral array, see [Lee '82, Boehm '84b].

Remark 9: If one first computes the rear face and then the bottom (or top) face of the tetrahedral array, one needs to solve formula (2) above (or (3)) for $\mathbf{c}_{r+1,i,k}$ (or $c_{r,i-1,k}$). This is impossible if $u = a_{i+n-r-k}$ (or $u = a_i$). Hence, the derivatives of the polynomial $\mathbf{s}_n$ cannot be computed in this fashion for $u = a_{n+1}, \ldots, a_{2n}$ (or $u = a_0, \ldots, a_{n-1}$).

5.10 Conversions between Bézier and B-spline representations

There is also a tetrahedral algorithm to convert a B-spline representation into a Bézier representation and vice versa [Boehm '77, Sablonniere '78]. It can be derived similarly as the algorithm in 5.9. Let the notations be as in 5.9 and let

$$\mathbf{q}_{rik} = \mathbf{s}_n[a \overset{r}{\ldots} a\, a_{i+1} \ \ldots \ a_{i+n-r-k}\, b \overset{k}{\ldots} b]$$

for $i = r + k, \ldots, n$. Thus, the control points of the spline are given by

$$\mathbf{c}_i = \mathbf{q}_{0i0} \ ,$$

and the Bézier points of the polynomial $\mathbf{s}_n$ over $[a, b]$ are given by

$$\mathbf{b}_j = \mathbf{q}_{n-j,n,j} \ .$$

Again, the points $\mathbf{q}_{rik}$ are conveniently arranged in a tetrahedral array, as illustrated below in Figure 5.10 for $n = 2$, where $a3, ab$, etc. stand for $\mathbf{q}_{120}$, $\mathbf{q}_{101}$, etc.

The left face is computed according to the rule

$$\mathbf{q}_{r+1,i,k} = (1-\alpha)\mathbf{q}_{r,i-1,k} + \alpha\, \mathbf{q}_{r,i,k} \ , \qquad \alpha = \frac{a - a_i}{a_{i+n-r-k} - a_i} \ ,$$

and the bottom face according to the rule

$$\mathbf{q}_{r,i,k+1} = (1-\gamma)\mathbf{q}_{r+1,i,k} + \gamma\, \mathbf{q}_{r,i,k} \ , \qquad \gamma = \frac{b - a}{a_{i+n-r-k} - a} \ .$$

Conversely, one can compute the B-spline control points from the Bézier points. First, one solves the two formulae above for $\mathbf{q}_{r,i-1,k}$ and $\mathbf{q}_{rik}$. Second, one applies the formulae to compute the bottom and then the left face.

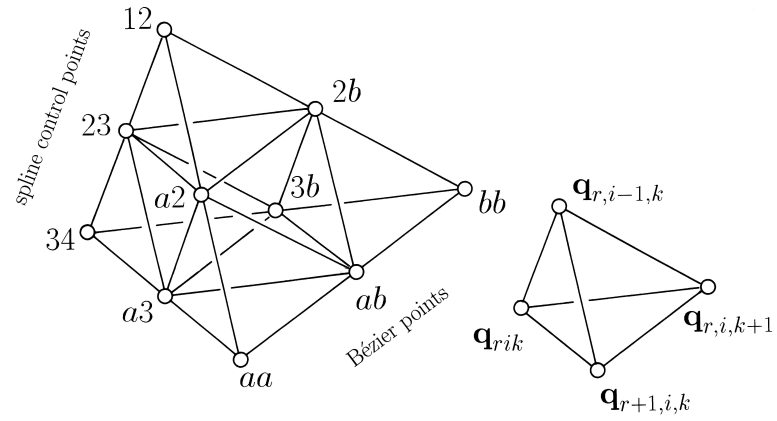


Figure 5.10: Conversion between Bézier and B-spline representations.

5.11 B-splines as divided differences

The standard definition of B-splines uses divided differences, and the calculus of divided differences has been heavily used in developing the univariate spline theory [de Boor '78]. In particular, divided differences were used by de Boor, Cox, and Mansfield to derive the recurrence relation.

Using the derivative formula (1) in 5.6, we show that B-splines are divided differences of the **truncated power function**

$$f(a) = (a-u)_+^n := \begin{cases} (a-u)^n & \text{if} \quad a > u \\ 0 & \text{otherwise} \end{cases} \quad ,$$

shown in Figure 5.11. Note that f is a function of a while u is some fixed parameter.

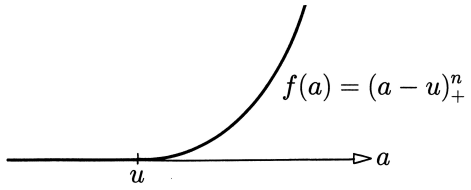


Figure 5.11: A truncated power function.

With the divided differences given in 4.3, one obtains that

the B-spline N_0^n with the knots $a_0, \ldots, a_{n+1}$ can be written as

$$N_0^n(u) = (a_{n+1} - a_0)[a_0 \ \ldots \ a_{n+1}](a-u)_+^n \ .$$

One can prove this fact by induction over n. For $n = 0$, the identity is easily checked. For the induction step from $n-1$ to n, we recall from 4.3 that $[a_0 \ \ldots \ a_{n+1}]f(a)$ is the leading coefficient of the $(n+1)$th degree polynomial interpolating $f(a)$ at $a_0, \ldots, a_{n+1}$. Hence, one can substitute $f(a)$ by the monomial $(a-u)^n$ of degree n in a if $u < a_0$ and by the zero function if $u \geq a_{n+1}$. This shows that the identity above holds for $u < a_0$ and $u \geq a_{n+1}$.

Thus, it suffices to show that the derivative of the claimed identity holds. Note that the divided difference is a linear combination of possibly differentiated power functions. Hence, the divided difference is differentiable in u, except at an $(n+1)$-fold knot. This causes no problem since there is at most one $(n+1)$-fold knot.

Using the recursive definition of divided differences, the induction hypothesis and the derivative formula for B-splines, we obtain

$$\begin{aligned}
&\frac{d}{du}(a_{n+1} - a_0)[a_0 \ \ldots \ a_{n+1}](a-u)_+^n \\
&\qquad = \ -n(a_{n+1} - a_0)[a_0 \ \ldots \ a_{n+1}](a-u)_+^{n-1} \\
&\qquad = \ n([a_0 \ \ldots \ a_n](a-u)_+^{n-1} - [a_1 \ \ldots \ a_{n+1}](a-u)_+^{n-1}) \\
&\qquad = \ \frac{n}{a_n - a_0} N_0^{n-1} - \frac{n}{a_{n+1} - a_1} N_1^{n-1} \\
&\qquad = \ \frac{d}{du} N_0^n(u) \ ,
\end{aligned}$$

which proves the assertion. $\diamond$

5.12 Problems

1 Consider a cubic C^2 spline $s(u)$ with the single knots $a_0, \ldots, a_m$. Show that every C^2 function $f(u) \neq s(u)$ which interpolates s at all knots and also the derivative of $s(u)$ at $u = a_0$ and a_m has greater strain energy than s, i.e.,

$$\int_{a_0}^{a_m} |f''(u)|^2 du > \int_{a_0}^{a_m} |s''(u)|^2 du \ .$$

For a solution, we refer to literature in numerical analysis, for example, [Boehm & Prautzsch '93, pp. 125 f].

2 Given a spline $s(u) = \sum_{i=0}^m c_i N_i^n(u)$ with the knots $a_0, \ldots, a_{m+n+1}$, show

that

$$\int_{a_0}^{a_{m+n+1}} s(u)du \;=\; \sum_{i=0}^{m} \frac{a_{i+n+1}-a_i}{n+1}\, c_i \;\;.$$

3 Sketch the cubic B-splines with the knots $0,0,0,0,1$; $0,0,0,1,2$; $0,0,1,2,3$ and $0,1,2,3,4$ with their Bézier polygons. Compute the values of their Bézier ordinates.

4 Use symmetric polynomials to prove **Marsden's identity**

$$(u-a)^n = \sum_i (a_{i+1}-a)\dots(a_{i+n}-a)N_i^n(u) \;\;.$$

5 Use the derivative formula of B-splines to derive the recursion formula of de Boor, Mansfield and Cox by induction.

6 Use Leibniz's identity for the product $f=gh$ of two functions, i.e.,

$$[a_i \;\dots\; a_{i+k}]f = \sum_{r=i}^{i+k} ([a_i \;\dots\; a_r]g)([a_r \;\dots\; a_{i+k}]h) \;\;,$$

to derive[de Boor '72] the recursion formula of de Boor, Mansfield and Cox, see also [de Boor '72].

7 Let $s(u)=\sum_{i=0}^{3} c_i N_i^3$ with the knots $0,1,2,\dots,7$ be given by $c_0,\dots,c_3 = 4,7,-2,1$.

a) Sketch $s[3,4]$ with its control polygon.

b) Compute s, s', s'' and s''' at $u=3$.

c) Compute the monomial representation of $s(u)$ over $[3,4]$.

d) Compute the Bézier representation of $s(u)$ over $[3,4]$.

8 Show that if a multiaffine symmetric polynomial can be computed from $n+1$ points $\mathbf{p}[a_{i,1} \;\dots\; a_{i,n}], i=0,\dots,n$, by affine combinations as in de Boor's algorithm, then there are real numbers $a_1,\dots,a_{2n}$ such that $a_{i,j}=a_{i+j}$.

13 Constructing smooth surfaces

13.1 The general C^1 joint — 13.2 The vertex enclosure problem — 13.3 The parity phenomenon — 13.4 Joining two triangular cubic patches — 13.5 Piper's G^1 interpolant — 13.6 Notes and Problems

The simple C^1 joint discussed in 11.7 is too restrictive for modelling smooth regular surfaces of arbitrary shape. Here, we present general C^1-conditions and a interpolation scheme that allows to design regular surfaces of arbitrary topology with triangular patches.

13.1 The general C^1 joint

Let $\mathbf{p}(x, y)$ and $\mathbf{q}(x, y)$ be two regular C^1 surface patches with a common boundary at $x = 0$, i.e.,

$$\mathbf{p}(0, y) = \mathbf{q}(0, y)$$

for all $y \in [0, 1]$. Figure 13.1 gives an illustration. The patches $\mathbf{p}$ and $\mathbf{q}$ neither need to be polynomial, nor three- or four-sided.

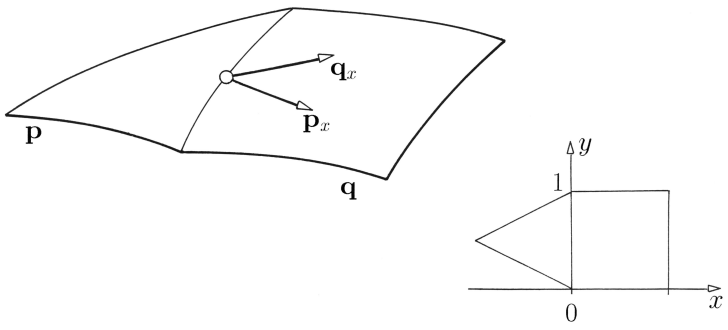


Figure 13.1: Two patches with a common boundary curve.

One says that $\mathbf{p}$ and $\mathbf{q}$ have a **general $\mathbf{C^1}$** or **geometric $\mathbf{C^1}$**- or, in short, **$\mathbf{G^1}$ joint** in $x = 0$ if they have equal normals along this parameter line, i.e., if

$$\frac{\mathbf{p}_x \times \mathbf{p}_y}{\|\mathbf{p}_x \times \mathbf{p}_y\|} = \frac{\mathbf{q}_x \times \mathbf{q}_y}{\|\mathbf{q}_x \times \mathbf{q}_y\|} \quad \text{for} \quad x = 0 \ .$$

Equivalently, one can characterize G^1-continuity by requiring that there are **connection functions** $\lambda(y), \mu(y)$ and $\nu(y)$ such that for $x = 0$ and all y

$$\lambda \mathbf{p}_x = \mu \mathbf{q}_x + \nu \mathbf{q}_y \quad \text{and} \quad \lambda\mu < 0 \ , \tag{1}$$

except for isolated zeros.

In particular, if $\mathbf{p}$ *and* $\mathbf{q}$ *have a* G^1 *joint and are polynomials, then the connection functions are also polynomials, and, up to a common factor, we have*

$$\begin{aligned} \text{degree } \lambda &\leq \text{degree } \mathbf{q}_x(0,y) + \text{degree } \mathbf{q}_y(0,y) \ , \\ \text{degree } \mu &\leq \text{degree } \mathbf{p}_x(0,y) + \text{degree } \mathbf{q}_y(0,y) \ , \\ \text{degree } \nu &\leq \text{degree } \mathbf{p}_x(0,y) + \text{degree } \mathbf{q}_x(0,y) \ . \end{aligned}$$

For a proof, we compute the vector product of equation (1) with $\mathbf{q}_x$ and $\mathbf{q}_y$. This gives

$$\begin{aligned} \lambda \mathbf{p}_x \times \mathbf{q}_x &= \nu \mathbf{q}_y \times \mathbf{q}_x \ , \quad \text{and} \\ \lambda \mathbf{p}_x \times \mathbf{q}_y &= \mu \mathbf{q}_x \times \mathbf{q}_y \ . \end{aligned}$$

Recall that $\mathbf{q}$ is regular. Hence, at least one coordinate, say the first of $[\mathbf{q}_x \times \mathbf{q}_y]$, denoted by $[\mathbf{q}_x \times \mathbf{q}_y]_1$, is non-zero. Since equation (1) can be multiplied by a factor, we may assume that

$$\lambda = [\mathbf{q}_x \times \mathbf{q}_y]_1 \ .$$

This implies

$$\mu = [\mathbf{p}_x \times \mathbf{q}_y]_1 \quad \text{and} \quad \nu = -[\mathbf{p}_x \times \mathbf{q}_x]_1 \ ,$$

which proves the assertion. $\diamond$

Remark 1: Often, one sets $\lambda = 1$. Then μ and ν are rational, in general.

Remark 2: The proof given above also holds for rational polynomials $\mathbf{p}$ and $\mathbf{q}$. Then, the functions λ, μ and ν are rational up to a common factor with the same degree estimates as above.

Remark 3: Any G^1 joint is a simple C^1 joint after a suitable parameter transformation. Namely, if $\mathbf{p}$ and $\mathbf{q}$ satisfy the G^1-condition (1), then $\mathbf{a}(x,y) = \mathbf{p}(\lambda x, y)$ and $\mathbf{b}(x,y) = \mathbf{q}(\mu x, \nu x + y)$ have a simple C^1 joint, see 9.7 and 11.7.

Remark 4: Whether two patches have a G^1 joint does not depend on their parametrization. However, the connection functions do depend on the parametrization. The maximum degree of the connection function is invariant under affine reparametrization.

13.2 Joining two triangular cubic patches

Consider two triangular cubic patches

$$\mathbf{p}(\mathbb{u}) = \sum \mathbf{p}_{\mathbb{i}} B^3_{\mathbb{i}}(\mathbb{u}) \quad \text{and} \quad \mathbf{q}(\mathbb{u}) = \sum \mathbf{q}_{\mathbb{i}} B^3_{\mathbb{i}}(\mathbb{u}) \ ,$$

where $\mathbb{0} \leq \mathbb{i} = (i,j,k)$ and $|\mathbb{i}| = i+j+k = 3$ and $\mathbf{p}_{\mathbb{i}} = \mathbf{q}_{\mathbb{i}}$ for $i = 0$ such that $\mathbf{p}$ and $\mathbf{q}$ join continuously at $n = 0$ and have common tangent planes at $\mathbb{e}_1 = (0,1,0)$ and $\mathbb{e}_2 = (0,0,1)$. This configuration is illustrated in Figure 13.2. The shaded quadrilaterals are planar but not necessarily affine.

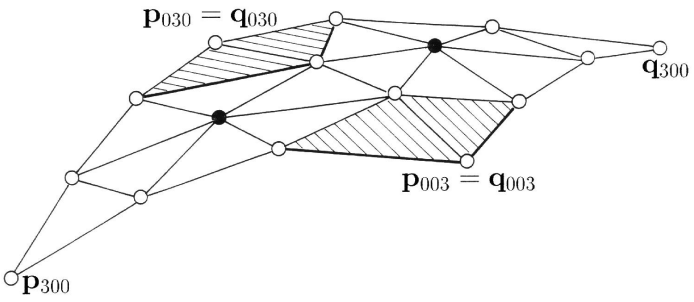


Figure 13.2: Moving interior Bézier points so as to achieve a G^1 joint.

In general, we can move both interior points $\mathbf{p}_{111}$ *and* $\mathbf{q}_{111}$ *so that* $\mathbf{p}$ *and* $\mathbf{q}$ *join* G^1*-continuously along* $u = 0$.

In particular, we show how to obtain such a smooth joint with linear connection functions $\lambda(v)$, $\mu(v)$ and $\nu(v)$. Then, the G^1-condition for $\mathbf{p}$ and $\mathbf{q}$ along $u = 0$ becomes a cubic equation in $w = 1 - v$. Denoting the partial derivatives with respect to the directions $\mathbb{e}_0 - \mathbb{e}_2$ and $\mathbb{e}_2 - \mathbb{e}_1$ by subindices 0 and 1, respectively, this cubic equation is

$$\lambda \mathbf{p}_0 + \mu \mathbf{q}_0 = \nu \mathbf{q}_1 \ , \quad \lambda\mu > 0 \ .$$

At $v = 0$, we know the derivatives $\mathbf{p}_0, \mathbf{q}_0$ and $\mathbf{q}_1$. Hence, this equation establishes a linear system for $\lambda_0 = \lambda(0)$, $\mu_0 = \mu(0)$ and $\nu_0 = \nu(0)$, with a

one parameter family of solutions. Similarly, there is a one parameter family of solutions λ_1, μ_1 and ν_1 at $v = 1$. We choose arbitrary solutions at $v = 0$ and $v = 1$, which determine the linear functions λ, μ and ν. Since a cubic is determined by its values and derivatives at two points, here $v = 0$ and $v = 1$, we are also interested in the derivative and, therefore, differentiate the G^1-condition along $u = 0$. Thus, we obtain

$$\lambda \mathbf{p}_{01} + \mu \mathbf{q}_{01} = \nu \mathbf{q}_{11} + \nu' \mathbf{q}_1 - \lambda' \mathbf{p}_0 + \mu' \mathbf{q}_0 \ .$$

Expressing $\mathbf{p}_{01}$, $\mathbf{q}_{01}$ etc. in terms of the Bézier points, we obtain at $\mathbb{e}_1$ the equations

$$\begin{aligned} \mathbf{p}_{01} &= 6(\mathbf{p}_{003} + \mathbf{p}_{111} - \mathbf{p}_{012} - \mathbf{p}_{102}) \ , \\ \mathbf{q}_{01} &= 6(\mathbf{q}_{021} + \mathbf{q}_{111} - \mathbf{q}_{012} - \mathbf{q}_{120}) \\ &\text{etc.} \end{aligned}$$

and similar expressions for $\mathbf{p}_{01}, \mathbf{q}_{01}$, etc. at $\mathbb{e}_2$. The points $\mathbf{q}_{11}, \mathbf{q}_1, \mathbf{q}_0$, and $\mathbf{p}_0$ do not depend on $\mathbf{p}_{111}$ and $\mathbf{q}_{111}$ for $v = 0$ and $v = 1$. Substituting these expressions into the differentiated G^1-condition leads to a linear system for $\mathbf{p}_{111}$ and $\mathbf{q}_{111}$ given by

$$[\mathbf{p}_{111}\mathbf{q}_{111}] \begin{bmatrix} \lambda_0 & \lambda_1 \\ \mu_0 & \mu_1 \end{bmatrix} = [\mathbf{w}_0 \mathbf{w}_1] \ ,$$

where $\mathbf{w}_0$ and $\mathbf{w}_1$ are combinations of known Bézier points $\mathbf{p}_{ijk}$ and $\mathbf{q}_{ijk}$, except $\mathbf{p}_{111}$ and $\mathbf{q}_{111}$. This system has a solution if the matrix

$$\begin{bmatrix} \lambda_0 & \lambda_1 \\ \mu_0 & \mu_1 \end{bmatrix}$$

is invertible. Hence, a solution exists, unless $\lambda(y) : \mu(y) = \textit{constant}$. Only for the configuration illustrated in Figure 13.3, a solution might not exist. $\diamond$

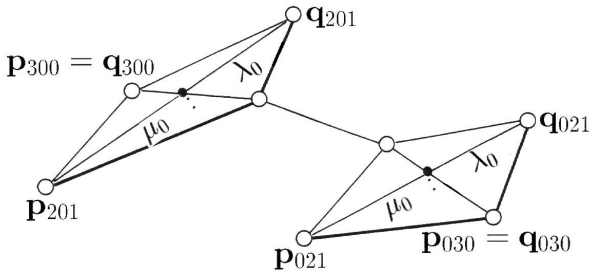


Figure 13.3: Critical configuration.

In fact, there is no solution if $\lambda(y) : \mu(y)$ is constant, if both quadrilaterals are not affine and if the common boundary $\mathbf{p}(0, y) = \mathbf{q}(0, y)$ is a regular cubic, i.e., if $\mathbf{q}_y(0, y)$ is a quadratic not passing through the origin.

Namely, rewriting the G^1-condition as

$$\mathbf{p}_x - \frac{\mu}{\lambda}\mathbf{q}_x = \frac{\nu}{\lambda}\mathbf{q}_y$$

results in a quadratic on the left. Since $\mathbf{q}_y$ is also quadratic without real root, it follows that ν/λ must be constant. This, finally, contradicts the assumption that the two quadrilaterals shown in Figure 13.3 are not affine.

If $\mathbf{q}(0, y)$ is quadratic or non-regular, a solution exists with linear functions λ, μ and ν, see Problem 3.

13.3 Piper's G^1 interpolant

In 1985, Bruce Piper [Piper '87] presented a scheme to construct a piecewise quartic G^1 surface interpolating a triangular network of cubic curves, as illustrated in Figure 13.4. We review the basic construction, but rule out critical situations so that cubic patches suffice.

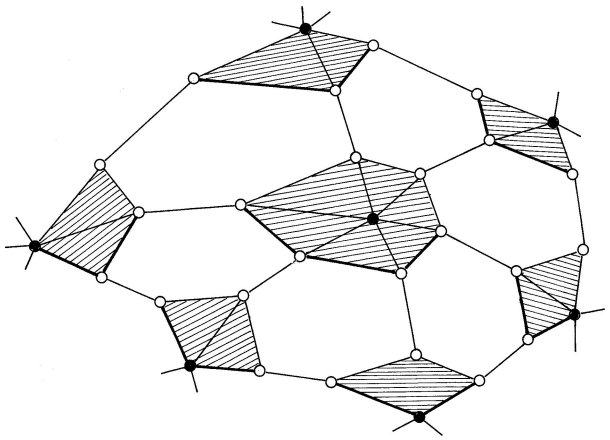


Figure 13.4: A triangular G^1-net of cubic curves.

Adjacent "triangles" of a cubic net exhibit the configurations discussed in 13.2. For simplicity, we assume that there are no critical configurations as in Figure 13.3. Then, any "triangle" can be interpolated by a macro patch consisting of three cubic patches, as described below. Figure 13.5 shows the Bézier points of such a macro patch schematically.

The Bézier points ○ on the boundary are given by the cubic net. The Bézier points ● are the centroids of their three neighbors ○ with which they form a planar quadrilateral which is shaded in the Figure.

The Bézier points ◓ are computed as in 13.2 such that adjacent macro patches have a G^1 joint.

The Bézier points □ are the centroids of the three neighbors ◓ ● ◓ with which they form a planar quadrilateral (shaded in the Figure).

The Bézier point ■ is the centroid of the shaded triangle □ □ □ .

Hence, adjacent patches of the macro patch have simple C^1 joints in analogy to the Clough-Tocher element, see 12.2.

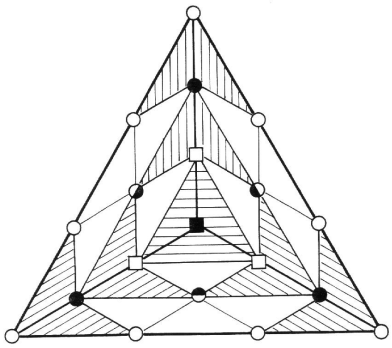


Figure 13.5: Piper's macro patch.

13.4 The vertex enclosure problem

Piper implicitly solves with his construction a special G^1-problem. The G^1-conditions for adjacent patches sharing a common vertex form a cyclic system. Whether this problem is solvable or not, is referred to as the **vertex enclosure problem**. To discuss this problem, we consider n trilateral or quadrilateral patches $\mathbf{p}^i(x,y),\ i=1,\ldots,n$, such that

$$\mathbf{p}^i(0,z) = \mathbf{p}^{i+1}(z,0) \ ,$$

where $\mathbf{p}^{n+1} = \mathbf{p}^1$, as illustrated in Figure 13.6, and

$$\lambda_i \mathbf{p}_x^i(0,z) = \mu_i \mathbf{p}_y^{i+1}(z,0) + \nu_i \mathbf{p}_x^{i+1}(z,0) \ , \tag{2}$$

with any $3n$ connection functions $\lambda_i(z), \mu_i(z)$ and $\nu_i(z)$.

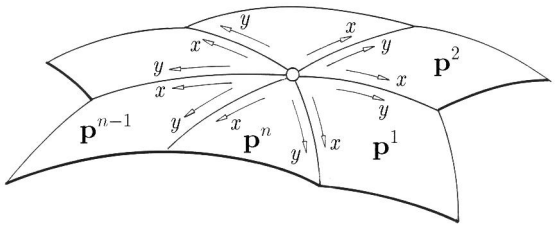


Figure 13.6: A vertex enclosed by n patches, where $n = 6$.

For $z = 0$, these equations form constraints on the derivatives $\mathbf{p}_x^i$ and the connection functions, which can easily be satisfied. More difficult are the **twist constraints**, which we obtain by differentiating the G^1-conditions (2),

$$\lambda_i' \mathbf{p}_x^i + \lambda_i \mathbf{p}_{xy}^i = \mu_i' \mathbf{p}_y^{i+1} + \mu_i \mathbf{p}_{xy}^{i+1} + \nu_i' \mathbf{p}_x^{i+1} + \nu_i \mathbf{p}_{xx}^{i+1} \ .$$

For $z = 0$, these equations form the cyclic linear system

$$[\mathbf{p}_{xy}^1 \cdots \mathbf{p}_{xy}^n] \begin{bmatrix} \lambda_1 & & & -\mu_1 \\ -\mu_2 & \lambda_2 & & \\ & \ddots & & \\ & & -\mu_n & \lambda_n \end{bmatrix} = [\mathbf{r}_1 \ldots \mathbf{r}_n] \ ,$$

where

$$\mathbf{r}_i = -\lambda_i' \mathbf{p}_x^i + \mu_i' \mathbf{p}_y^i + \nu_i' \mathbf{p}_x^{i+1} + \nu_i \mathbf{p}_{xx}^{i+1} \ .$$

We abbreviate this system by $TA = R$.

13.5 The parity phenomenon

The cyclic matrix A of the twist constraints exhibits the following phenomenon. The rank of A is n if n is odd, and it is $n - 1$ if n is even. Thus, A is non-singular only for odd n. Consequently, the twist constraints are solvable if the number n of patches is odd, while, in general, there is no solution if the number is even. In order to verify this surprising fact, we observe that

$$\begin{aligned} \det A &= \lambda_1 \ldots \lambda_n - \mu_1 \ldots \mu_n \\ &= \Pi\lambda_i - \Pi\mu_i \ . \end{aligned}$$

Computing, the vector product of

$$\lambda_i \mathbf{p}_x^i = \mu_i \mathbf{p}_y^{i+1} + \nu_i \mathbf{p}_x^{i+1} \ ,$$

with $\mathbf{p}_x^{i+1} = \mathbf{p}_y^i$ gives

$$\lambda_i : \mu_i = [\mathbf{p}_x^{i+2} \times \mathbf{p}_x^{i+1}] : [\mathbf{p}_x^{i+1} \times \mathbf{p}_x^i] \ ,$$

which implies that

$$\Pi\lambda_i : \Pi\mu_i = (-1)^n$$

and, since all $\lambda_i \neq 0$,

$$\det A \begin{cases} = 0 & \text{if } n \text{ is even} \\ \neq 0 & \text{if } n \text{ is odd} \end{cases} \ .$$

Since the submatrix of A obtained by deleting the first row and column has full rank, A has at least rank $n-1$, which concludes the proof.

Remark 5: The twist constraints $AT = R$ are solvable if the data originates from patches $\mathbf{p}^1, \ldots, \mathbf{p}^n$ forming a G^1 surface. In particular, this is the case when the $\mathbf{p}^i$ form a piecewise polynomial reparametrization of a polynomial patch.

Remark 6: If $n = 4$ and $\lambda_i'(0) = \mu_i'(0) = \nu_i'(0) = \nu_i(0) = 0$, for $i = 1, 2, 3, 4$, as illustrated in Figure 13.7, then the twist constraints are solvable, see 9.7.

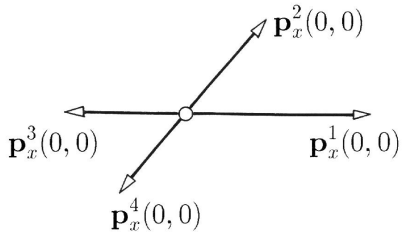


Figure 13.7: Equal opposite tangents.

Remark 7: One obtains always solvable twist constraints by splitting each patch $\mathbf{p}^i$ as in Piper's construction described in 13.3, see [Peters '91].

13.6 Problems

1 Show that the problem from Section 13.2 can always be solved if $\mathbf{p}$ and $\mathbf{q}$ are quartic patches.

2 Solve the problem from Section 13.2, where

$$\begin{array}{cccccc} & \mathbf{p}_{201} & \mathbf{p}_{111} & \mathbf{p}_{012} & & \\ & & & & \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} & \begin{bmatrix} \\ \mathbf{p}_{111} \\ \\ \end{bmatrix} & \begin{bmatrix} 6 \\ 1 \\ 1 \end{bmatrix} \\ \mathbf{p}_{300} & \mathbf{p}_{210} & \mathbf{p}_{120} & \mathbf{p}_{030} & & \\ \| & \| & \| & \| & = & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix} & \begin{bmatrix} 6 \\ 0 \\ 1 \end{bmatrix} \\ \mathbf{q}_{300} & \mathbf{q}_{210} & \mathbf{q}_{120} & \mathbf{q}_{030} & & \\ & \mathbf{q}_{201} & \mathbf{q}_{111} & \mathbf{q}_{012} & & \\ & & & & \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} & \begin{bmatrix} \\ \mathbf{q}_{111} \\ \\ \end{bmatrix} & \begin{bmatrix} 5 \\ -1/2 \\ 1 \end{bmatrix} \end{array} .$$

3 Show that the problem stated in 13.2, see Figure 13.3, has a solution with linear functions $\lambda = \mu$ and ν if $\mathbf{q}(0, v, 1-v)$ is quadratic or non-regular. Hint: If $\mathbf{q}(0, v, 1-v)$ is quadratic, choose $\lambda = \mu = 1$. If $\mathbf{q}_1(0, v, 1-v) = \mathbf{o}$ for $v = v_0$, set $\lambda = \mu = (v - v_0)c_1$ and $\nu = vc_2$, where the constants c_1 and c_2 are chosen so that the G^1-condition is satisfied at $v = 1$.

4 Show that the twist constraints in 13.4 have a solution if and only if

$$\mathbf{s} = \mathbf{t}_1 \left(1 - \frac{\mu_1 \cdots \mu_n}{\lambda_1 \cdots \lambda_n}\right)$$

can be solved for $\mathbf{t}_1$, where

$$\mathbf{s} = \frac{\mu_1 \cdots \mu_{n-1}}{\lambda_1 \cdots \lambda_n}\mathbf{r}_n + \cdots + \frac{\mu_1}{\lambda_1\lambda_2}\mathbf{r}_2 + \frac{1}{\lambda_1}\mathbf{r}_1 \ .$$

5 The equation $\mathbf{s} = \mathbf{o}$ can be viewed as a linear system for $\nu_1'(0), \nu_2'(0)$ and $\nu_1(0)$. It has a (unique) solution unless $\mathbf{p}_{xx}^2 = \mathbf{o}$.

6 Consider two biquartic patches

$$\begin{aligned} \mathbf{p}(x, y) &= \sum \mathbf{p}_{ij} B_{ij}^{44}(x, y) \quad \text{and} \\ \mathbf{q}(x, y) &= \sum \mathbf{q}_{ij} B_{ij}^{44}(x, y) \end{aligned}$$

such that the boundary curves $\mathbf{p}(x, 0)$ and $\mathbf{q}(x, 0)$ agree and are cubic, see Figure 13.8.

Let λ, μ and ν be any connection functions of degree $1, 1$ and 3, respectively, such that the G^1-condition

$$\lambda \mathbf{p}_y(x, 0) = \mu \mathbf{q}_y(x, 0) + \nu \mathbf{q}_x(x, 0)$$

and its derivative with respect to x hold at $x = 0$ and $x = 1$. Show that

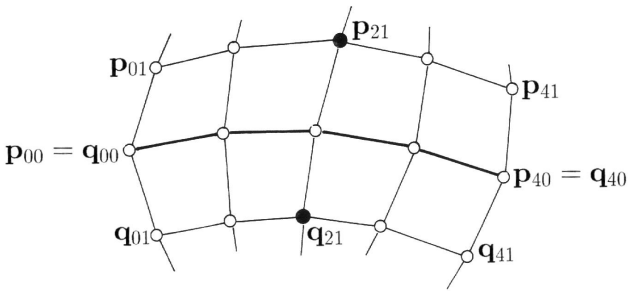


Figure 13.8: Moving two Bézier points so as to obtain a G^1 joint.

one can change $\mathbf{p}_{21}$ and $\mathbf{q}_{21}$, in general, so that the G^1-condition holds for all x.