Computer Aided Geometric Design 3 (1986) 83–127
North-Holland

Triangular Bernstein–Bézier patches

Gerald FARIN

Department of Mathematics, University of Utah, Salt Lake City, UT 84112, U.S.A.

Received 14 January 1986
Revised 12 May 1986

Contents

Keywords. Bernstein polynomials, Bézier methods, scattered data interpolation, triangular schemes.

Notation

$\mathcal{T}$, $\mathcal{S}$	domain triangles and simplices
T_i, P	points in the domain of a function
τ_i, σ_i	barycentric coordinates
α, β, γ	directions in the domain of a function
λ, μ	multi-indices
ϵ^i	barycentric coordinates of vertex T_i of domain triangle or simplex
$D_\alpha f$	directional derivative of f with respect to α
$B_\lambda^n(\tau)$	Bernstein polynomial
b^n	Bernstein–Bézier triangular patch
b_λ	Bézier ordinate corresponding to λ
λ^r	multi-index λ with first component $\lambda_1 = r$
τ^0	barycentric coordinates with $\tau_1 = 0$
$\mathcal{B}$	control net (a piecewise linear function)
$b_\lambda^r(\tau)$	intermediate de Casteljau ordinates
D_i	directional derivative with respect to edge direction no. i
$\boldsymbol{b}_i$, $\boldsymbol{b}_\lambda$	points in $\mathbb{R}^2$ or $\mathbb{R}^3$ (where $\mathbb{R}^2$ or $\mathbb{R}^3$ is the range of a curve or surface)

Introduction

The rapidly growing field of CAGD (Computer Aided Geometric Design) has been dominated by the theory of rectangular surface patches since its inception in the late sixties by S. Coons (from MIT) and P. Bézier (from Rénault). Historically, it is therefore very remarkable that triangular patches were already considered in the late fifties by P. de Casteljau (from Citroën). He had then realized that what is now known as Bézier curves admitted a symmetric formulation in terms of barycentric coordinates. This concept is immediately generalizable to surfaces – it yields triangular patches. So de Casteljau considered triangular patches before he defined tensor product 'Bézier' patches. This historical 'first' of triangular patches is reflected by the mathematical statement that they are a more 'natural' generalization of Bézier curves than are tensor product patches. We should note that, while de Casteljau's work was never published, that of Bézier was, which accounts for the fact that the corresponding field now bears Bézier's name. In this paper, we adopt this naming convention, although to the author's knowledge Bézier never considered triangular patches. For the placement of triangular Bernstein–Bézier surfaces in the field of CAGD, see [Barnhill '85].

While de Casteljau's work (laid down in two internal Citroën technical reports and not easily read) remained unknown until its discovery by W. Boehm about 1975, other researchers realized the need for triangular patches. M. Sabin [Sabin '77] worked on triangular patches in terms of Bernstein polynomials, unaware of the work of de Casteljau. Among possibly more people concerned with the development of triangular patches we name P.J. Davis [Davis '76], L. Frederickson [Frederickson '70, '71], G. Farin [Farin '79], and P. Sablonnière [Sablonnière '82]. All of the mentioned Bézier-type approaches relied on the fact that piecewise surfaces were defined over regular triangulations; arbitrary triangulations were considered by [Farin '80]. A Coons-type triangular patch (a so-called transfinite interpolant) was developed by [Barnhill, Birkhoff, Gordon '74].

The main attraction of Bernstein–Bézier patches is that they lend themselves easily to a geometric understanding of the mathematical concepts that are involved. The structure of this paper is itself an illustration of this fact: the first three chapters contain a mostly algebraic treatment of basic properties of Bernstein–Bézier patches. After these results have been established, it is basically sufficient to address geometric concepts (collinearity of points, area ratios of triangles, etc.) in order to formulate algorithms. These would otherwise require extensive algebraic treatment and have in fact often been derived in a much more complicated way than described here.

In summary, we feel that the theory of Bernstein–Bézier methods very much helps to give meaning to the 'G' in CAGD.

I. THEORY

1. Introductory concepts

1.1. Barycentric coordinates

Any point P in the plane can be expressed in terms of *barycentric coordinates* with respect to any nondegenerate triangle $\mathcal{T}$ in that plane having vertices T_1, T_2, T_3:

$$P = \sum_{i=1}^{3} \tau_i T_i. \tag{1.1}$$

The τ_i are usually normalized by the requirement

$$|\tau| := \sum_{i=1}^{3} \tau_i = 1. \tag{1.2}$$

Equations (1.1) and (1.2) form a 3×3 linear system (note that (1.1) is a vector equation) which has the unique solution

$$\tau_1 = \frac{\text{area}(P, T_2, T_3)}{\text{area}(T_1, T_2, T_3)}, \qquad \tau_2 = \frac{\text{area}(T_1, P, T_3)}{\text{area}(T_1, T_2, T_3)}, \qquad \tau_3 = \frac{\text{area}(T_1, T_2, P)}{\text{area}(T_1, T_2, T_3)}.$$

The areas arising in the definition of the τ_i are the determinants arising from the solution of the 3×3 system by Cramer's rule.

If P is inside $\mathcal{T}$, one gets

$$\tau_i \geqslant 0; \quad i = 1, 2, 3.$$

The triangle vertices T_i have barycentric coordinates ϵ^i, where $\epsilon^1 = (1, 0, 0)$, $\epsilon^2 = (0, 1, 0)$, $\epsilon^3 = (0, 0, 1)$. Figs. 1.1a and 1.1b illustrate more geometric properties of barycentric coordinates.

A remark on the origin of the work 'barycentric': 'barycenter' means center of gravity. If masses τ_i are attached to the points T_i, their center of gravity is located at P, with the accumulated mass $\tau_1 + \tau_2 + \tau_3 = 1$.

An important property of barycentric coordinates is their *affine invariance*: if the triangle $\mathcal{T}$ together with P is transformed by an affine transformation, the transformed point has unchanged barycentric coordinates with respect to the transformed triangle. To see this, let Φ be an affine transformation $\Phi P = \mathbf{A}P + Q$. Then

$$\Phi P = \mathbf{A}P + Q = \mathbf{A}\left(\sum \tau_i T_i\right) + Q = \sum \tau_i (\mathbf{A}T_i + Q) = \sum \tau_i \Phi T_i.$$

Barycentric coordinates are *symmetric*: each side of the triangle is treated the same way as the

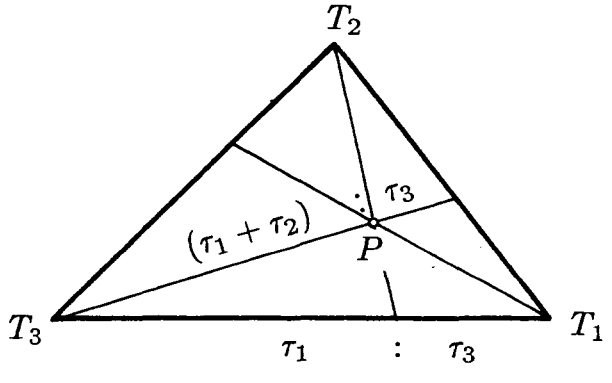


Fig. 1.1.a. The geometry of barycentric coordinates: the point P subdivides the drawn 'radial' lines in the indicated ratios; the endpoints of the 'radials' subdivide the edges in the indicated ratios. Only two of the six generated ratios are shown; the others follow by symmetry.

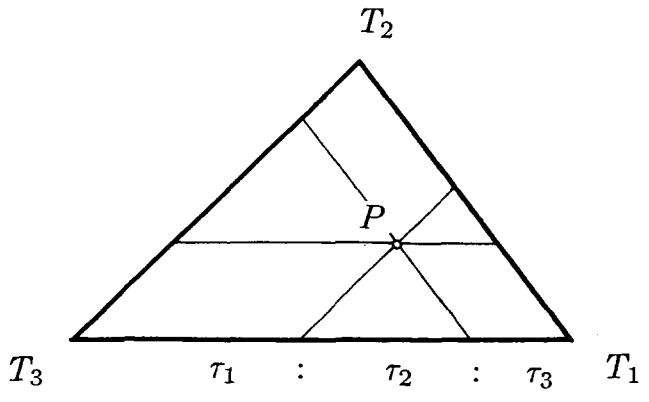


Fig. 1.1.b. The geometry of barycentric coordinates: the lines $\tau_i = const.$ are shown together with the ratios in which they subdivide the edges. Only one such ratio is shown.

other ones. The use of barycentric coordinates is thus easier and more elegant than the use of a standard triangle (defined by the three points (1, 0), (0, 1), (0, 0)), in which case distinctions are necessary for each side (see, e.g., [Barnhill, Birkhoff, Gordon '74]).

In the following we shall be concerned with bivariate functions that are defined over triangles. We shall write these as $f(\tau)$, i.e., as functions of three dependent variables. This raises the problem of how to handle differentiation: the terms $\partial f/\partial \tau_i$ do not have a geometric interpretation. Therefore, one resorts to *directional derivatives*: Let ρ, σ be (barycentric coordinates of) two arbitrary points. Their difference α (note that $|\alpha| := \sum \alpha_i = 0$) defines a direction with respect to which directional derivatives can be taken:

$$D_\alpha f(\tau) = \sum_{i=1}^{3} \frac{\partial f}{\partial \tau_i} \alpha_i. \tag{1.3}$$

Thus when dealing with functions in terms of barycentric coordinates, directional derivatives will be used instead of partial derivatives. The direction α need not be of unit length.

As an example for directions in barycentric notation, consider the direction α defined by $T_2 - T_1$. In terms of barycentric coordinates one has $\alpha = \epsilon^2 - \epsilon^1 = (-1, 1, 0)$. As a general rule, barycentric point coordinates sum to one while barycentric vector coordinates sum to zero.

1.2. Lagrange interpolation

One of the most important applications of univariate polynomials is in interpolation, and we shall now show how those concepts carry over to polynomial interpolation over a triangle. One difference to the univariate case is that the data points must be distributed in the triangle in a completely symmetric fashion, i.e., we can only generalize the case of uniform knot spacing in a straightforward way.

Let $\lambda = (\lambda_1, \lambda_2, \lambda_3)$, $|\lambda| = n$, $\lambda_i \in \{0, 1, \ldots, n\}$. Thus the points $\tau_\lambda := \lambda/n$ form an n-partition of $\mathscr{T}$, see Fig. 1.2 for the case $n = 4$. Suppose one is given point data p_λ at the locations τ_λ and one wants a polynomial of total degree n that interpolates to these data points (see also [Gregory '78]). The number of data equals $(n+1)(n+2)/2$, the $(n+1)$st so-called *triangle number*. The desired interpolating polynomial is given by

$$p(\tau) = \sum_{|\lambda|=n} p_\lambda L_\lambda^n(\tau), \tag{1.4}$$

where

$$L_\lambda^n(\tau) = \frac{n^n}{\lambda!} \prod_{r=0}^{\lambda_1 - 1} \prod_{s=0}^{\lambda_2 - 1} \prod_{t=0}^{\lambda_3 - 1} \left(\tau_1 - \frac{r}{n}\right)\left(\tau_2 - \frac{s}{n}\right)\left(\tau_3 - \frac{t}{n}\right). \tag{1.5}$$

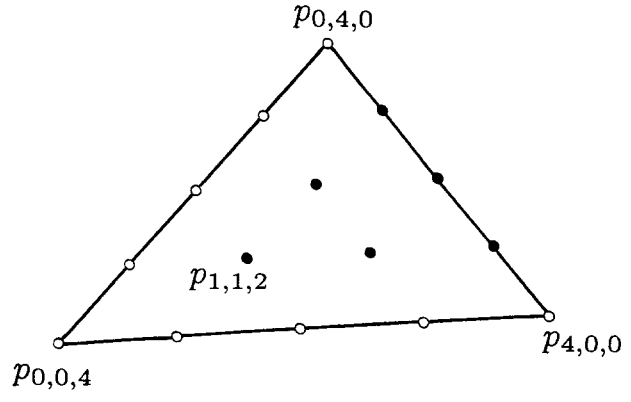


Fig. 1.2. Lagrange interpolation: the full circles indicate which data points influence the intermediate value $p^2_{1,1,0}$. In this example: $n = 4$, thus the data points are located over a four-partition of the triangle.

For a proof one simply checks that the L^n_λ are cardinal functions:

$$L^n_\lambda(\tau_\mu) = \delta_{\mu,\lambda},$$

where $\delta_{\mu,\lambda}$ denotes the Kronecker delta. (Note that λ and μ are triples.) This cardinal property also ensures that the L^n_λ form a basis for all nth degree polynomials defined over $\mathcal{T}$.

In univariate Lagrange interpolation, one often prefers a recursive evaluation scheme to the evaluation of the cardinal form. Such a scheme is also available for Lagrange interpolation over triangles:

Let p^r_λ, $|\lambda| = n - r$, interpolate to the points p_λ (see also Fig. 1.2). Then the recursion

$$p^r_\lambda(\tau) = \frac{n}{r} \sum_{k=1}^{3} \left(\tau_k - \frac{\lambda_k}{n} \right) p^{r-1}_{\lambda+\epsilon^k}(\tau) \tag{1.6}$$

with $p^0_\lambda = p_\lambda$ yields the result $p(\tau) = p^n_\phi(\tau)$ where $\phi = (0, 0, 0)$. Equation (1.6) is the analogue to Aitken's scheme for univariate polynomial interpolation, see also [Prenter '74].

Two special cases: along the boundaries of $\mathcal{T}$, the interpolant (1.4) reduces to the well-known univariate Lagrange interpolant to the data on that boundary. For the case $n = 1$, one has linear interpolation, i.e., the plane through the three data points.

1.3. Bernstein polynomials

Bernstein polynomials of degree n over a triangle are defined by

$$B^n_\lambda(\tau) = \frac{n!}{\lambda_1!\lambda_2!\lambda_3!} \tau_1^{\lambda_1}\tau_2^{\lambda_2}\tau_3^{\lambda_3}; \quad |\lambda| = n \tag{1.7}$$

and the convention $B^n_\lambda(\tau) = 0$ if $\lambda_k \notin [0, n]$ for some k. Bernstein polynomials are terms of the trinomial expansion of $1 = |\tau|^n$, thus

$$\sum_{|\lambda|=n} B^n_\lambda(\tau) \equiv 1. \tag{1.8}$$

Since for $\tau_k \geq 0$ we have

$$B^n_\lambda(\tau) \geq 0, \tag{1.9}$$

Bernstein polynomials form a *partition of unity*. Another property of Bernstein polynomials is that they have only one maximum over $\mathcal{T}$, namely B^n_λ assumes its maximum at $\tau = \lambda/n$. Fig. 1.3 shows two examples of cubic Bernstein polynomials.

Bernstein polynomials B^n_λ are the natural generalization of univariate Bernstein polynomials

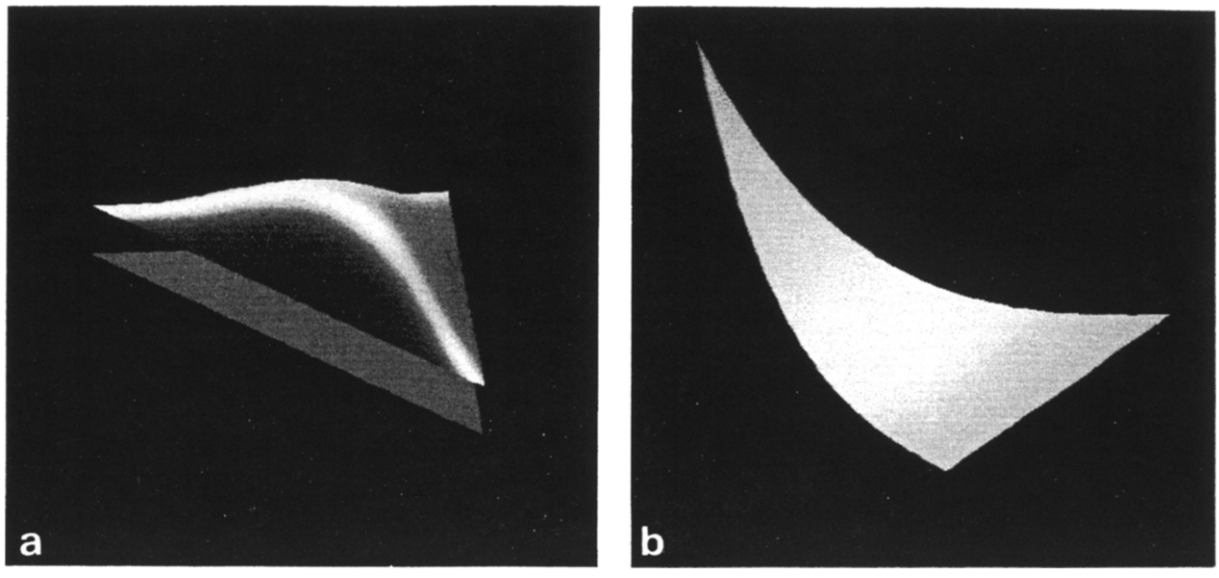


Fig. 1.3. Bernstein polynomials: the two cubic Bernstein polynomials $B^3_{1,1,1}$ and $B^3_{0,3,0}$. Both surfaces are shown with an offset over the domain triangle.

$B^n_i(t) = \binom{n}{i} t^i (1-t)^{n-i}$: the univariate B^n_i are defined over [0, 1], and with the definitions $\tau_1 = t$, $\tau_2 = 1 - t$, they can be written as

$$B^n_i(t) = B^n_\lambda(\tau) = \frac{n!}{\lambda_1!\lambda_2!}\tau_1^{\lambda_1}\tau_2^{\lambda_2}.$$

Here, τ_1 and τ_2 are barycentric coordinates on the interval [0, 1] (in fact, on any interval).

We shall be interested in polynomials b^n of the form

$$b^n(\tau) = \sum_{|\lambda|=n} b_\lambda B^n_\lambda(\tau). \tag{1.10}$$

The b_λ are called *Bézier ordinates* of b^n. The piecewise linear interpolant $\mathscr{B}$ to the points $(\lambda/n, b_\lambda)$ is called *Bézier net* or *control net* of b^n. As an example for a control net, consider the polynomial B^n_μ: its control net ordinates are defined by $b_\lambda = \delta_{\lambda,\mu}$. The control net is then given by $(\lambda/n, b_\lambda)$. Fig. 1.4 illustrates the example of a cubic patch with the corresponding control net. This control net uniquely defines the patch, a fact which is made use of in the so-called Bernstein–Bézier technique, where all information about the patch is extracted (very often in a geometric way) from this net.

It is important to mention that the representation (1.10) of b^n is unique, i.e., the B^n_λ form a *basis* for all polynomials of total degree n that are defined over $\mathscr{T}$, see, e.g., [de Boor '86a].

From equations (1.9) and (1.10) we immediately get the *convex hull property*: $b(\tau)$ lies in the convex hull of the defining ordinates, i.e., $\min\{b_\lambda\} \le b(\tau) \le \max\{b_\lambda\}$. This property is important for contouring and rendering of surfaces, see also Section 7.

We shall denote barycentric coordinates corresponding to $\tau_1 = 0$ by $\tau^0 = (0, \tau_2, \tau_3)$. Similarly, we shall reserve the notation λ^r for multiindices with $\lambda_1 = r$, in particular $\lambda^0 = (0, \lambda_2, \lambda_3)$. This notation is useful to describe phenomena pertaining to the edge $\tau_1 = 0$. Analogous notation could be developed for any edge, but because of the symmetry of the considered theory, it is sufficient to consider just one edge.

The boundary curves of the patch b^n are (univariate) Bézier polynomials (see [Boehm, Farin, Kahmann '84]). Their control polygons are the boundaries of the control net. For the example $\tau^0 = (0, \tau_2, \tau_3)$, i.e., for the triangle edge T_2, T_3, we obtain

$$b^n(\tau^0) = \sum_{|\lambda^0|=n} b_{\lambda^0} B^n_{\lambda^0}(\tau^0).$$

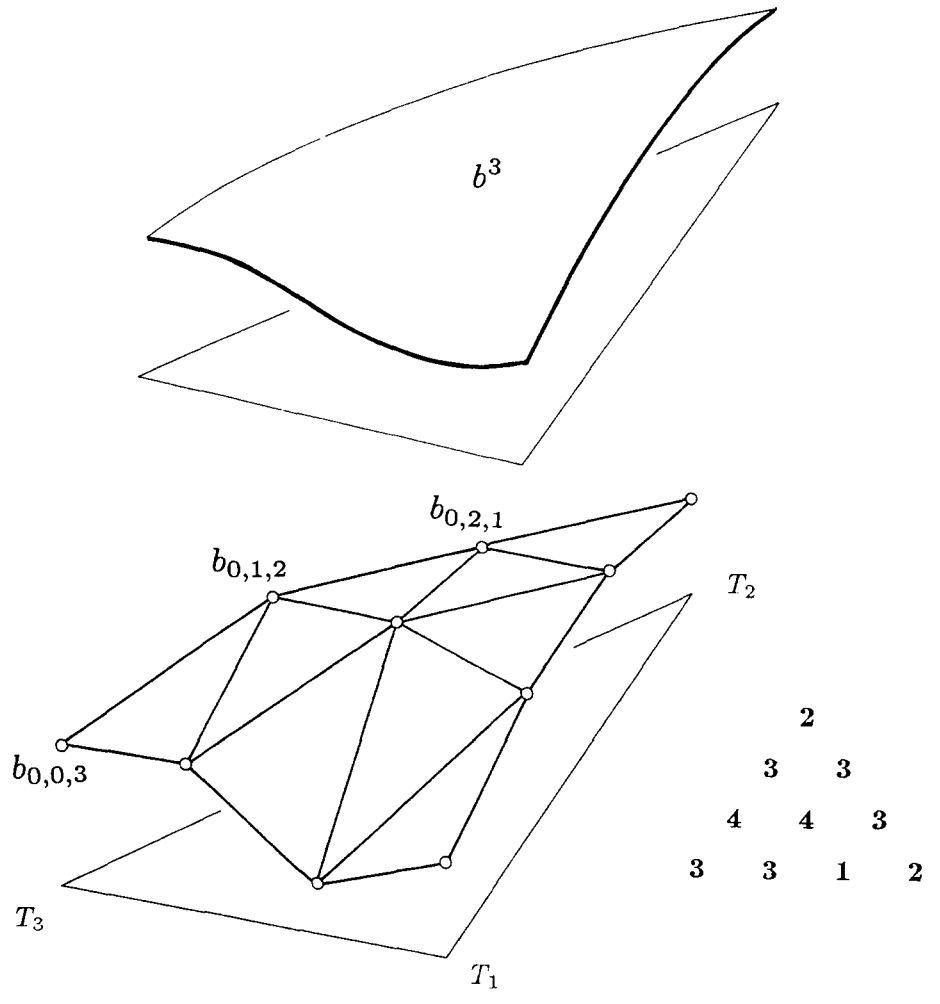


Fig. 1.4. Bernstein–Bézier patches: a cubic patch together with its control net.

(Note that this defines a univariate curve since $\tau_2 + \tau_3 = 1$.) As a consequence, the tangent plane of b^n at ϵ_3 is determined by $b_{0,0,n}, b_{1,0,n-1}, b_{0,1,n-1}$. A more general statement can be found in Theorem 2.4.

We finally note that for the case of the control net being planar, the generated patch is also planar. This is a property of Bernstein–Bézier patches that is referred to as *linear precision*.

1.4. Degree elevation

It is possible to write $b^n(\tau)$ as a Bézier polynomial of degree $n+1$, i.e.

$$\sum_{|\lambda|=n} b_\lambda B_\lambda^n(\tau) = \sum_{|\mu|=n+1} b_\mu^{(1)} B_\mu^{n+1}(\tau). \tag{1.11}$$

Lemma 1.1. *The $b_\mu^{(1)}$ are determined by*

$$b_\mu^{(1)} = \frac{1}{n+1} \sum_{k=1}^{3} \mu_k b_{\mu-\epsilon^k}; \quad |\mu| = n+1. \tag{1.12}$$

Proof. Simply multiply the left hand side of (1.11) by $1 = |\tau|$ and compare coefficients of like powers.

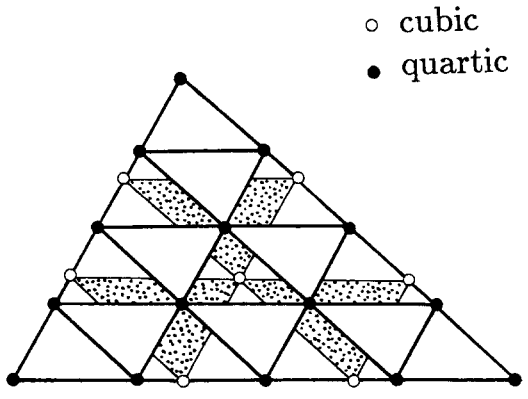


Fig. 1.5. Degree elevation: the control nets of a cubic patch together with that of the equivalent quartic patch. Note that the new control points are generated in 'upside down' subtriangles.

Equation (1.12) has a simple geometric interpretation: the ordinates $b_\mu^{(1)}$ are found by evaluating the (piecewise linear) control net $\mathscr{B}$ of b^n at the new abscissae $\mu/(n+1)$: $b_\mu^{(1)} = \mathscr{B}(\mu/(n+1))$; Fig. 1.5 illustrates this for the case $n=3$. As a consequence, the new control net $\mathscr{B}_1$ lies in the convex hull of the old net $\mathscr{B}$.

Let us now investigate what happens if we repeat the process of degree elevation. Let $\mathscr{B}_i$ denote the control net after the ith degree elevation. Also recall that $\mathscr{B}_i$ is a piecewise linear function, defined over the same triangle as b^n. The vertices of $\mathscr{B}_i$ are given by

$$\mathscr{B}_i\left(\frac{\mu}{n+i}\right) = b_\mu^{(i)}; \quad |\mu| = n+i.$$

After the ith degree elevation, we have

$$\sum_{|\lambda|=n} b_\lambda B_\lambda^n(\tau) = \sum_{|\mu|=n+i} b_\mu^{(i)} B_\mu^{n+i}(\tau). \tag{1.13}$$

The following Lemma is due to J. Zhou [Zhou '85]:

Lemma 1.2.

$$b_\mu^{(i)} = \sum_{|\lambda|=n} b_\lambda \binom{\mu_1}{\lambda_1}\binom{\mu_2}{\lambda_2}\binom{\mu_3}{\lambda_3} \Big/ \binom{n+i}{n}. \tag{1.14}$$

Proof. Multiply the left hand side of (1.13) by $1 = \sum_{|\nu|=i} B_\nu^i(\tau)$ and rearrange.

If one repeats the process of degree elevation, the control nets $\mathscr{B}_i$ will converge to the surface b^n which each of them defines [Farin '79]:

Theorem 1.3.

$$\lim_{i\to\infty} \mathscr{B}_i = b^n. \tag{1.15}$$

Proof. The first part of the proof given here is due to J. Zhou [Zhou '85]: consider a sequence $\mu = \mu(i)$ with $|\mu| = n+i$ and

$$\lim_{i\to\infty} \frac{\mu}{n+i} = \tau \tag{1.16}$$

uniformly for some τ inside the domain triangle. We shall first prove that

$$\lim_{i\to\infty} b_\mu^{(i)} = b^n(\tau). \tag{1.17}$$

The following inequalities are verified after some calculation:

$$\sum_{|\lambda|=n} b_\lambda B_\lambda^n\left(\frac{\mu-\lambda}{n+i}\right) \le b_\mu^{(i)} \frac{(n+i)!}{i!} \le \sum_{|\lambda|=n} b_\lambda B_\lambda^n\left(\frac{\mu}{i}\right). \tag{1.18}$$

Since $\mu/(n+i)$ tends to τ uniformly, both the left and the right hand side of (1.18) tend to $b^n(\tau)$, and (1.17) is proved.

For the second part of the proof, one observes that in the limit, the points $\mu/(n+i)$ are dense in the domain triangle. Thus for any τ, there exists a sequence (1.16) and thus a sequence of Bézier ordinates $b_\mu^{(i)}$ with $\lim_{i\to\infty} b_\mu^{(i)} = b^n(\tau)$. Hence for any τ, the sequence $(\mu/(n+i), \mathscr{B}_i(\mu/(n+i)))$ has a subsequence that converges to $(\tau, b^n(\tau))$, which completes the proof.

2. The de Casteljau algorithm and its applications

In the theory of Bézier curves, the algorithm for the recursive evaluation of the curve plays a fundamental role: it provides subdivision formulas, differentiability conditions, and more (see [Boehm, Farin, Kahmann '83]). This algorithm is due to P. de Casteljau [de Casteljau '63], and was developed together with the analogous algorithm for triangular patches, to be discussed below. It is interesting to note that the Casteljau developed the triangular algorithm before he studied the tensor product generalization of the curve algorithm, although it is a commonly held belief that tensor product patches are 'the' natural generalization of curve schemes.

2.1. The de Casteljau algorithm

The Bernstein polynomials B_λ^n satisfy a recurrence relation:

Lemma 2.1.

$$B_\lambda^n(\tau) = \sum_{i=1}^{3} \tau_i B_{\lambda-\epsilon^i}^{n-1}(\tau); \quad |\lambda| = n. \tag{2.1}$$

Proof. Use the definition of the B_λ^n and the identity

$$\frac{n!}{\lambda_1!\lambda_2!\lambda_3!} = \binom{n}{\lambda_1}\binom{n-\lambda_1}{\lambda_2},$$

and the recursion formula for binomial coefficients.

This lemma allows to expand b^n in terms of Bernstein polynomials of lower degree with (polynomial) coefficients $b_\lambda^r(\tau)$:

Theorem 2.2.

$$b^n(\tau) = \sum_{|\lambda|=n-r} b_\lambda^r(\tau) B_\lambda^{n-r}(\tau); \quad 0 \le r \le n, \tag{2.3}$$

where

$$b_\lambda^0(\tau) = b_\lambda,$$

$$b_\lambda^r(\tau) = \sum_{i=1}^{3} \tau_i b_{\lambda+\epsilon^i}^{r-1}(\tau); \quad |\lambda| = n-r. \tag{2.4}$$

It is interesting to note the similarity between the recursion (2.4) for the Bézier coefficients

with the recursion (1.6) for the Lagrange coefficients. This connection is considered in more detail by [Farin, Barry '86].

Proof. We use induction on r. The theorem is true for $r=0$ by definition. The induction proceeds as follows:

$$b^n(\tau) = \sum_{|\lambda|=n-r} b_\lambda^r(\tau) B_\lambda^{n-r}(\tau)$$

by the recurrence relation (2.1):

$$b^n(\tau) = \sum_{|\lambda|=n-r} b_\lambda^r(\tau) \sum_{i=1}^{3} \tau_i B_{\lambda-\epsilon^i}^{n-r-1}(\tau)$$

by the inductive hypothesis:

$$\begin{aligned} b^n(\tau) &= \sum_{|\lambda|=n-r-1} \sum_{i=1}^{3} \tau_i b_{\lambda+\epsilon^i}^r(\tau) B_\lambda^{n-r-1}(\tau) \\ &= \sum_{|\lambda|=n-r-1} b_\lambda^{r+1}(\tau) B_\lambda^{n-r-1}(\tau). \end{aligned}$$

Since $b^n(\tau) = b_\Phi^n(\tau)$, Theorem 2.2 is proved; it provides the so-called algorithm of de Casteljau for the recursive evaluation of $b^n(\tau)$, see also Fig. 2.1.

The recursion (2.4) has a simple geometric interpretation: The points $(\lambda/n,\ b_\lambda^r(\tau))$ are the image of the domain triangle $\mathcal{T}$ together with τ under affine maps onto the upright triangles of the net of $b_\mu^{r-1}(\tau)$. Consequently, Bernstein–Bézier patches are invariant under affine transformations: It does not matter if $b^n(\tau)$ is computed and then subjected to an affine transformation or if the net is subjected to that affine transformation and $\hat{b}^n(\tau)$ is computed by application of the de Casteljau algorithm to the transformed net.

It is of interest to note that in the special cases $b_\lambda = \lambda_i/n$; $i=1, 2, 3$, the de Casteljau algorithm computes τ_i, i.e.

$$\tau_i = \sum_{|\lambda|=n} \tau_i B_\lambda^n(\tau); \quad i=1, 2, 3.$$

As a consequence, Fig. 2.1 shows the 'top view' of the subnets generated by the de Casteljau algorithm.

The intermediate coefficients b_λ^r can also be written explicitly as

$$b_\lambda^r(\tau) = \sum_{|\mu|=r} b_{\lambda+\mu} B_\mu^r(\tau); \quad |\lambda| = n-r. \tag{2.5}$$

For a proof, one simply checks that (2.5) is consistent with the recursion (2.4).

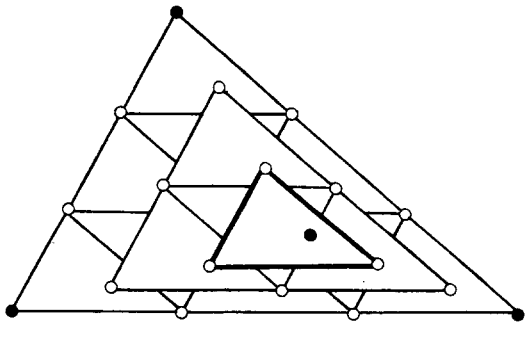


Fig. 2.1. The de Casteljau algorithm: the original cubic control net together with the quadratic and linear nets that determine the resulting point. All intermediate control points are generated in 'upright' subtriangles. The linear net (bold) of the b_λ^2, $|\lambda|=1$, determines the tangent plane at $b_\Phi^3(\tau)$; here, $\tau = (\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$.

Thus, the intermediate coefficients b_λ^r depend on those b_ν for which $\nu_i \geq \lambda_i$; $i = 1, 2, 3$, as is immediately verified by inspection of Fig. 2.1.

2.2. Derivatives

Let α be a direction with respect to which one wants to compute directional derivatives (see Section 1.1). Also let $\partial^\mu f(\tau)$ denote the partial derivative of a function f, i.e.

$$\partial^\mu f(\tau) = \frac{\partial^{|\mu|}}{\partial\tau_1^{\mu_1}\,\partial\tau_2^{\mu_2}\,\partial\tau_3^{\mu_3}} f(\tau).$$

The rth directional derivative of f with respect to α is given by a standard result from multivariate calculus:

$$D_\alpha^r f(\tau) = \sum_{|\mu|=r} B_\mu^r(\alpha)\,\partial^\mu f(\tau). \tag{2.6}$$

Note that the term $B_\mu^r(\alpha)$ is well-defined although we have $|\alpha| = 0$. It is important to emphasize the fact that Bernstein polynomials are the *natural* way to write down the directional derivative of any function. Hence it is not surprising that the derivatives of Bernstein–Bézier patches take on particularly simple forms.

A straightforward calculation verifies that in the case of $f = B_\lambda^n$

$$\partial^\mu B_\lambda^n(\tau) = \frac{n!}{(n-r)!} B_{\lambda-\mu}^{n-r}(\tau); \quad |\mu| = r.$$

This immediately yields

Theorem 2.3. *The rth directional derivative of a Bernstein polynomial B_λ^n is given by*

$$D_\alpha^r B_\lambda^n(\tau) = \frac{n!}{(n-r)!} \sum_{|\mu|=r} B_\mu^r(\alpha) B_{\lambda-\mu}^{n-r}(\tau). \tag{2.7}$$

We now have the tools necessary to prove

Theorem 2.4. *The rth directional derivative of b^n is given by*

$$D_\alpha^r b^n(\tau) = \frac{n!}{(n-r)!} \sum_{|\lambda|=r} b_\lambda^{n-r}(\tau) B_\lambda^r(\alpha). \tag{2.8}$$

Proof. Application of Theorem 2.3 to the definition of b^n yields

$$D_\alpha^r b^n(\tau) = \frac{n!}{(n-r)!} \sum_{|\mu|=n} \sum_{|\lambda|=r} b_\mu B_\lambda^r(\alpha) B_{\mu-\lambda}^{n-r}(\tau).$$

Rearranging:

$$D_\alpha^r b^n(\tau) = \frac{n!}{(n-r)!} \sum_{|\mu|=n-r} \sum_{|\lambda|=r} b_{\mu+\lambda} B_\lambda^r(\alpha) B_\mu^{n-r}(\tau) \tag{2.9}$$

$$= \frac{n!}{(n-r)!} \sum_{|\lambda|=r} B_\lambda^r(\alpha) \sum_{|\mu|=n-r} b_{\mu+\lambda} B_\mu^{n-r}(\tau).$$

Application of equation (2.5) gives the desired result.

A geometric interpretation of Theorem 2.4 is as follows: the three b_λ^{n-1}, $|\lambda| = 1$, determine the tangent plane at $b^n(\tau)$ etc. (see also Fig. 2.1).

A dual formula is provided by

Corollary 2.5.

$$D_\alpha^r b^n(\tau) = \frac{n!}{(n-r)!} \sum_{|\mu|=n-r} b_\mu^r(\alpha) B_\mu^{n-r}(\tau). \tag{2.10}$$

Proof. Replace τ by α in equation (2.5). Then (2.9) yields (2.10).

The terms $b_\mu^r(\alpha)$ in (2.10) have a simple geometric interpretation for the case $r=1$: $b_\mu^1(\alpha)$; $|\mu| = n-1$, denotes the slope (in direction α) of the plane spanned by the three $b_{\mu+\epsilon^i}$.

Theorem 2.4 and Corollary 2.5 have the following interpretation: If one computes D_α^r according to Theorem 2.4, one first performs $n-r$ de Casteljau steps with respect to τ (to obtain the $b_\lambda^{n-r}(\tau)$), then one performs r more de Casteljau steps with respect to α to obtain the result. Using Corollary 2.5, the situation is reversed: one first performs r de Casteljau steps with respect to α (to obtain the $b_\mu^r(\alpha)$), then $n-r$ steps with respect to τ to obtain the result. Moreover, the de Casteljau steps and the derivative steps commute [Farin '79], i.e., it is only important to perform r steps with respect to α and $n-r$ steps with respect to τ, but they may be carried out in any order.

We shall later need the special case of cross-boundary-derivatives: Let $\tau^0 = (0, \tau_2, \tau_3)$ be barycentric coordinates along the edge $\tau_1 = 0$, and α not parallel to that edge (for the notation, see end of Section 1.3). Then

$$D_\alpha^r b^n(\tau^0) = \frac{n!}{(n-r)!} \sum_{|\lambda^0|=n-r} b^r(\alpha) B_{\lambda^0}^{n-r}(\tau^0). \tag{2.11}$$

This expression depends only on the first $r+1$ rows of Bézier ordinates parallel to the edge under consideration. In that respect it is very similar to the univariate case, and it will be used to derive conditions for the differentiability between adjacent patches (see Section 2.3).

Mixed directional derivatives may be obtained from a generalization of Theorem 2.4: let α and β be two independent directions. Then

$$D_{\alpha,\beta}^{r+s} b^n(\tau) = \frac{n!}{(n-r-s)!} \sum_{|\mu|=r} \sum_{|\nu|=s} b_{\mu+\nu}^{n-r-s}(\tau) B_\mu^r(\alpha) B_\nu^s(\beta). \tag{2.12}$$

The corresponding generalization of Corollary 2.5 can be written as

$$D_{\alpha,\beta}^{r+s} b^n(\tau) = \frac{n!}{(n-r-s)} \sum_{|\lambda|=r} \sum_{|\mu|=s} b_{\lambda+\mu}^{r,s}(\alpha, \beta) B_{\lambda+\mu}^{n-r-s}(\tau). \tag{2.13}$$

The berm $b_{\lambda+\mu}^{r,s}(\alpha, \beta)$ in (2.13) indicates that we have to perform r de Casteljau steps with respect to α, then s steps with respect to β. Finally one has to perform $n-r-s$ steps with respect to τ. Of course all these steps commute, i.e., they may be carried out in any order.

An important special case is given by mixed derivatives at the vertices of $\mathcal{T}$. They depend only on the subnet of order $r+s$ located at that vertex.

Let us briefly consider the simple case $r=s=1$: the corresponding mixed derivatives are called *twists*. If we restrict α and β to be in the direction of the edges, we have *edge twists*. For the case $\alpha = \epsilon^1 - \epsilon^3$, $\beta = \epsilon^2 - \epsilon^3$, we obtain

$$b_\mu^{1,1}(\alpha, \beta) = \Delta^{1,1,0} b_\mu; \quad |\mu| = n-2, \tag{2.14}$$

where

$$\Delta^{1,0,0} b_\mu = b_{\mu+\epsilon^1} - b_{\mu+\epsilon^3} \quad \text{and} \quad \Delta^{0,1,0} b_\mu = b_{\mu+\epsilon^2} - b_{\mu+\epsilon^3}; \quad |\mu| = n-1.$$

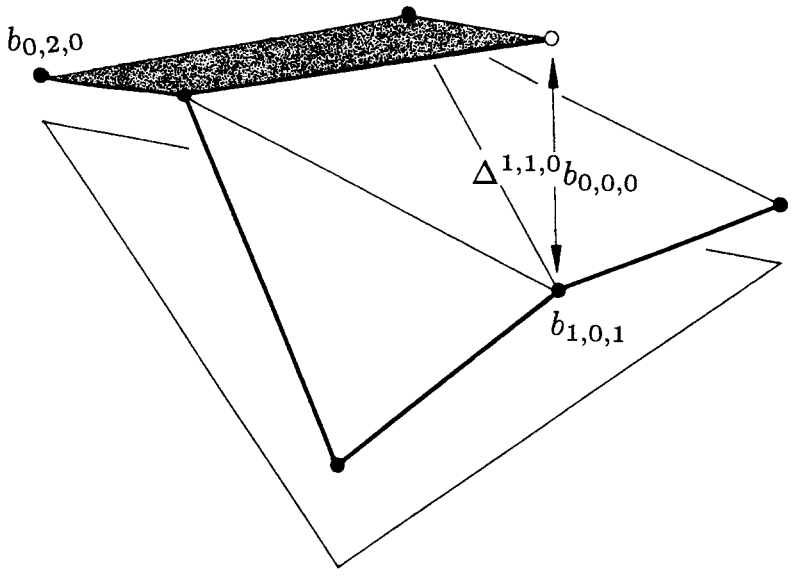


Fig. 2.2. Twists: the arrow (a constant multiple of the twist) indicates the local deviation of a quadratic control net from a plane (shaded).Note that two of the ordinates do not affect the twist at all.

Fig. 2.2 illustrates the geometric meaning of (2.14): each coefficient gives the deviation $\Delta^{1,1,0}b_\mu$ of the quadrilateral $(b_{\mu+\epsilon^1+\epsilon^2}, b_{\mu+\epsilon^2+\epsilon^3}, b_{\mu+\epsilon^1+\epsilon^3}, b_{\mu+2\epsilon^3})$ from a plane. Similar expressions hold for twists with respect to the other edges (see also Sections 2.3 and 3).

We conclude this section by giving a simple expression for partial derivatives:

Lemma 2.6.

$$\partial^\mu b^n(\tau) = \frac{n!}{(n-r)!} b_\mu^{n-r}(\tau); \quad |\mu| = r. \tag{2.15}$$

Proof.

$$\begin{aligned}\partial^\mu b^n(\tau) &= \frac{n!}{(n-r)!} \sum_{|\lambda|=n} b_\lambda B_{\lambda-\mu}^{n-r}(\tau) \\ &= \frac{n!}{(n-r)!} \sum_{|\lambda|=n-r} b_{\lambda+\mu} B_\lambda^{n-r}(\tau) \\ &= \frac{n!}{(n-r)!} b_\mu^{n-r}(\tau).\end{aligned}$$

As a simple consequence we get the following identity:

Corollary 2.7.

$$\partial^\mu b^n(\tau) = b_\mu; \quad |\mu| = n. \tag{2.16}$$

2.3. Domain transformation

Let $\mathcal{T}$ be a triangle with vertices T_1, T_2, T_3 and let $\hat{\mathcal{T}}$ be an adjacent triangle with vertices $\hat{T}_1$, T_2, T_3. We denote barycentric coordinates in $\mathcal{T}$ by τ, those in $\hat{\mathcal{T}}$ by $\hat{\tau}$. Triangle $\hat{\mathcal{T}}$ is not oriented counterclockwise, but for our purposes that is immaterial. The straight line T_2, T_3 is common to both triangles and corresponds to barycentric coordinates $\tau^0 = (0, \tau_2, \tau_3)$ and to

$\hat{\tau}^0 = (0, \hat{\tau}_2, \hat{\tau}_3)$. Let $\hat{T}_1$ have barycentric coordinates σ with respect to $\mathcal{T}$, thus $\tau_1 = \hat{\tau}_1\sigma_1$, $\tau_2 = \hat{\tau}_1\sigma_2 + \hat{\tau}_2$, $\tau_3 = \hat{\tau}_1\sigma_3 + \hat{\tau}_3$, where $\sigma_1 \neq 0$.

A polynomial $b^n(\tau)$ that is defined in terms of barycentric coordinates of $\mathcal{T}$ is of course defined over the whole plane, in particular also over $\hat{\mathcal{T}}$. We are interested in the Bézier ordinates that describe b^n as a Bernstein–Bézier polynomial over $\hat{\mathcal{T}}$. They are defined by $b^n(\tau) = \hat{b}^n(\hat{\tau})$, more explicitly:

$$\sum_{|\lambda|=n} b_\lambda B_\lambda^n(\tau) = \sum_{|\lambda|=n} \hat{b}_\lambda B_\lambda^n(\hat{\tau}). \tag{2.17}$$

Note that since T and $\hat{T}$ share a common edge, we immediately obtain

$$b_{\lambda^0} = \hat{b}_{\lambda^0}; \quad |\lambda^0| = n, \tag{2.18}$$

where $\lambda^0 = (0, \lambda_2, \lambda_3)$.

In order to determine the remaining $\hat{b}_\lambda$, let us consider a direction not parallel to T_2, T_3 and let us take directional derivatives up to order r with respect to it. This direction may be expressed as α with respect to triangle $\mathcal{T}$ and as $\hat{\alpha}$ with respect to triangle $\hat{\mathcal{T}}$. Hence (2.11) becomes

$$\sum_{|\lambda^0|=n-s} b_{\lambda^0}^s(\alpha) B_{\lambda^0}^{n-s}(\tau^0) = \sum_{|\lambda^0|=n-s} \hat{b}_{\lambda^0}^s(\hat{\alpha}) B_{\lambda^0}^{n-s}(\hat{\tau}^0); \quad s = 0, \ldots, r; \quad 0 \le r \le n. \tag{2.19}$$

For $r = 0$, (2.19) yields (2.18). Since $\tau^0 = \hat{\tau}^0$, comparing coefficients in (2.19) gives

$$b_{\lambda^0}^s(\alpha) = \hat{b}_{\lambda^0}^s(\hat{\alpha}); \quad s = 0, \ldots, r; \quad 0 \le r \le n. \tag{2.20}$$

Thus the two polynomials $b_{\lambda^0}^r$ and $\hat{b}_{\lambda^0}^r$ agree in all derivatives up to order r:

$$D_\alpha^s b_{\lambda^0}^s(\tau) = D_{\hat{\alpha}}^s \hat{b}_{\lambda^0}^s(\hat{\tau}); \quad s = 0, \ldots, r; \quad 0 \le r \le n,$$

and hence they are equal:

$$b_{\lambda^0}^r(\tau) = \hat{b}_{\lambda^0}^r(\hat{\tau}); \quad 0 \le r \le n.$$

The last equation must also hold for $\tau = \sigma$, i.e., for $\hat{\tau} = \epsilon^1$. This yields

Theorem 2.8 [Farin '80]. *Let b^n (with Bézier ordinates b_λ) be defined over $\mathcal{T} = \{T_1, T_2, T_3\}$ and let $\hat{b}^n$ (with Bézier ordinates $\hat{b}_\lambda$) be defined over $\hat{\mathcal{T}} = \{\hat{T}_1, T_2, T_3\}$.*

The two polynomials b^n and $\hat{b}^n$ are identical if and only if

$$\hat{b}_{\lambda^r} = b_{\lambda^0}^r(\sigma); \quad 0 \le r \le n, \tag{2.21}$$

where $\lambda^r = (r, \lambda_2, \lambda_3)$; $|\lambda^r| = n$, and σ are the barycentric coordinates of $\hat{T}_1$ with respect to $\mathcal{T}$.

Theorem 2.8 provides an algorithm to construct the desired $\hat{b}_\lambda$ from the given b_λ: Simply evaluate b^n at $\tau = \sigma$ by the de Casteljau algorithm and collect the 'rightmost' (referring to Fig. 2.7) intermediate coefficients from each step. Of course one can obtain the b_λ in the same way from the $\hat{b}_\lambda$.

A warning: Theorem 2.8 does not provide a *stable* algorithm: If σ is outside $\mathcal{T}$, then (2.21) does not use convex combinations any more, rather it is a formula using repeated extrapolation, suffering from the well-known numerical problems inherent in such schemes.

We have considered domain triangles $\mathcal{T}$ and $\hat{\mathcal{T}}$ that share a common edge. This restriction can be dropped, and $\hat{\mathcal{T}}$ may be completely independent of $\mathcal{T}$. If τ^i are the vertices of $\hat{\mathcal{T}}$ as expressed in terms of barycentric coordinates of $\mathcal{T}$, the $\hat{b}_\lambda$ are found by simultaneously performing the de Casteljau algorithms for the τ^i: one defines $b_\lambda^\kappa(\tau^1, \tau^2, \tau^3)$; $|\lambda| = n - |\kappa|$ in

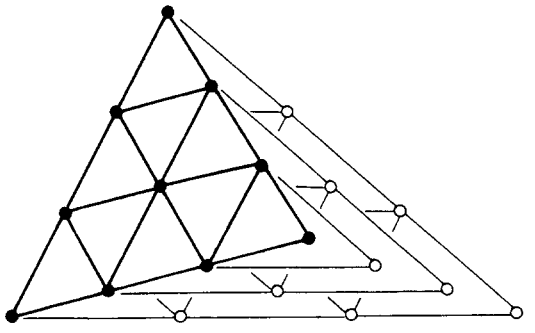


Fig. 2.3. Subdivision: The intermediate de Casteljau ordinates from Fig. 2.1 generate control nets for the subdivided patch. The ordinates for the subnet corresponding to the edge $\tau_1 = 0$ are shown as full circles.

complete analogy to the $b^{r,s}_{\mu+\lambda}$ from the definition of mixed partials (2.13). Thus b^{κ}_{λ} is obtained by applying κ_i de Casteljau steps with respect to τ^i. The desired coefficients $\hat{b}_\lambda$ are given by

$$\hat{b}_\kappa = b^\kappa_\phi(\tau^1, \tau^2, \tau^3); \quad |\kappa| = n.$$

It is again immaterial in which order the necessary de Casteljau steps are carried out. This approach is detailed in [Goldman '83] and illustrated in [Boehm, Farin '83]. An explicit formula for the $\hat{b}_\lambda$ is in [Chang, Davis '84].

Theorem 2.8 has two important applications:

Corollary 2.9 (subdivision). *The intermediate Bézier ordinates* b^r_λ, $|\lambda| = n - r$ *are the Bézier ordinates of the subpatches defined over the three subtriangles* $\{\hat{T}_1, T_3, T_1\}$, $\{\hat{T}_1, T_1, T_2\}$, $\{\hat{T}_1, T_2, T_3\}$.

Fig. 2.3 illustrates this subdivision procedure for the common case of $\hat{T}_1$ being inside the triangle.

A special case arises if $\hat{T}_1$ is on the edge (T_2, T_3). The straight line $(T_1, \hat{T}_1)$ is called a 'radial' line. The restriction of b^n to that line is a univariate Bézier polynomial of degree n, and its Bézier ordinates can be determined in a particularly simple way: Interpret all rows of Bézier ordinates parallel to (T_2, T_3) as Bézier ordinates of curves of degrees $0, 1, \ldots, n$. Evaluate each curve at $t = \sigma_3$. The values thus obtained are the desired Bézier ordinates; see Fig. 2.4 for an illustration.

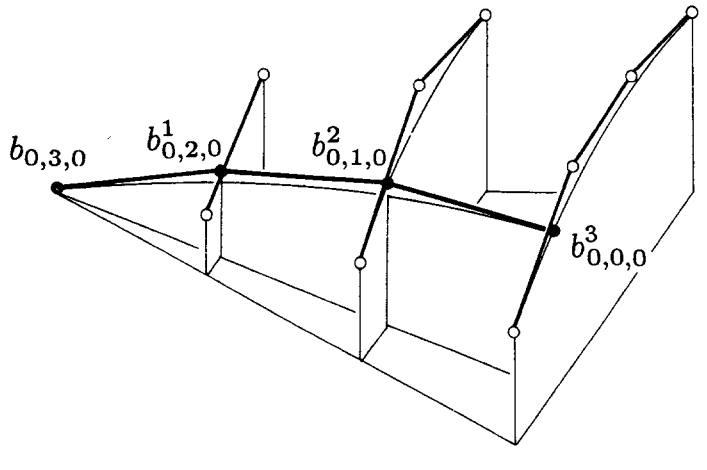


Fig. 2.4. Radial curves: the solid circles denote the Bézier points of a radial curve emanating from $b^3(\epsilon^2)$.

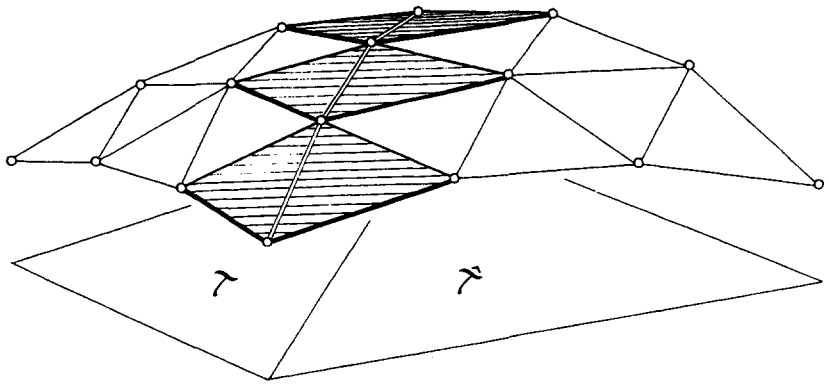


Fig. 2.5. C^1-conditions: the indicated pairs of subtriangles must be coplanar.

Corollary 2.10 (C^r conditions). *Let b^n be defined over T_1, T_2, T_3 and let c^n be defined over $\hat{T}_1$, T_2, T_3. Also let $\hat{T}_1$ have barycentric coordinates σ with respect to the first triangle. A necessary and sufficient condition for b^n and c^n to be C^r across the common boundary is*

$$c_{\lambda^s} = b^s_{\lambda^0}(\sigma); \quad s = 0, 1, \ldots, r. \tag{2.22}$$

This corollary is a generalization of the analogous result for Bézier curves, due to E. Stärk [Stärk '76].

For the case $r = 1$, Corollary 2.10 says that the bold pairs of triangles in Fig. 2.5 must be coplanar. For the case $r = 2$, we may rewrite (2.22) in a more symmetric way:

$$d_{\lambda^1} := c^1_{\lambda^1}(\rho) = b^1_{\lambda^1}(\sigma),$$

where ρ denotes the barycentric coordinates of T_1 with respect to the triangle $\{\hat{T}_1, T_2, T_3\}$. Thus the existence of the auxiliary points d_{λ^1} guarantees C^2 continuity across the common edge of two patches; see Fig. 2.6 for an illustration. The points d_{λ^1} bear a close resemblance to the auxiliary points that guarantee C^2 continuity between adjacent univariate Bézier curves (see [Boehm, Farin, Kahmann '83]. These auxiliary points can be interpreted as the B spline vertices of cubic C^2 spline curves, but it is not known if the d_{λ^1} could be utilized for the definition of C^2 splines over arbitrary triangulations.

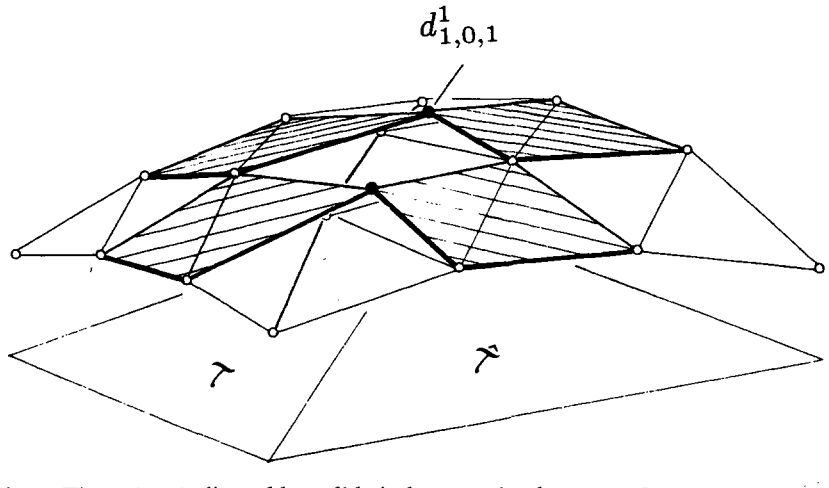


Fig. 2.6. C^2-conditions: The points indicated by solid circles must be the same when constructed from either control net. The indicated pairs of triangles are coplanar.

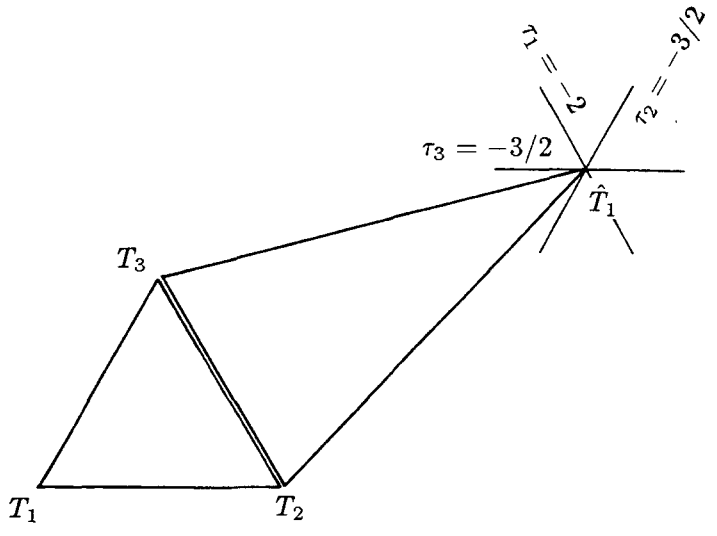


Fig. 2.7a. Domain transformation, example: the geometry of the domain triangles, characterized by $\sigma = (-2, \frac{3}{2}, \frac{3}{2})$.

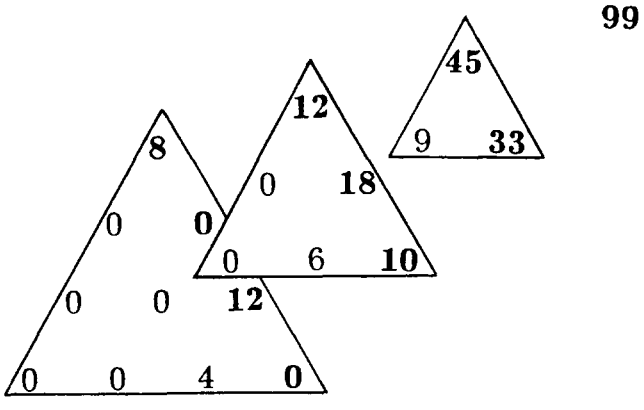


Fig. 2.7b. Domain transformation, example: the original ordinates together with the successively generated intermediate ordinates. The 'rightmost' ordinates $b'_{\lambda 0}$ are shown boldface.

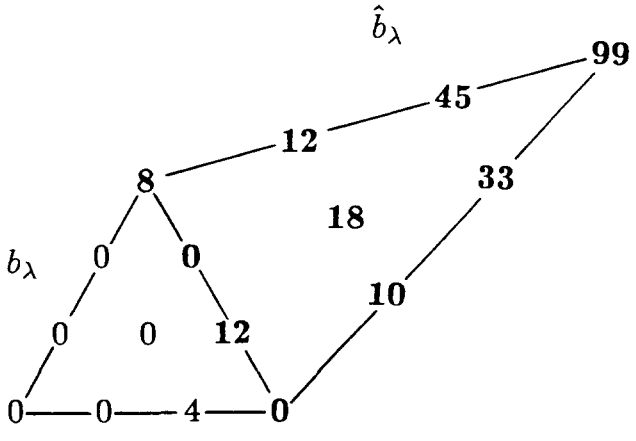


Fig. 2.7c. Domain transformation, example: the original ordinates b_λ (defined over $\mathscr{T}$) together with the ordinates $\hat{b}_\lambda$ defining the same surface over $\hat{\mathscr{T}}$.

We finish this section by giving an example of how to construct the Bézier ordinates of a patch that is C^3 with respect to a given patch: this is done in Fig. 2.7.

2.4. Alternative surface forms

It may sometimes be necessary to convert a patch in Bernstein–Bézier form to a different polynomial representation. This could arise if some of the subsequently developed interpolants need to be rewritten in order to fit into some specific software environment.

One of the standard forms for a bivariate polynomial is the expansion in terms of monomials $x^i y^j$:

$$f(x, y) = \sum_{i+j \le n} a_{i,j} x^i y^j, \tag{2.23}$$

which can be rewritten as a Taylor expansion:

$$f(x, y) = \sum_{i+j \le n} \frac{1}{i!j!} \frac{\partial^{i+j}}{\partial x^i \, \partial y^j} f(0, 0) x^i y^j. \tag{2.24}$$

Two remarks: (1) The partials appearing in (2.24) are directional derivatives with respect to the directions (1, 0) and (0, 1). (2) Eq. (2.24) is also valid if the x- and y-axes are not orthogonal.

It is now easy to write a Bernstein–Bézier polynomial $b^n(\tau)$ in monomial form. Define $\alpha = \epsilon^1 - \epsilon^3$, $\beta = \epsilon^2 - \epsilon^3$. Then by the above two remarks (2.24) may be applied to Bernstein–Bézier patches:

$$b^n(\tau) = \sum_{i+j \le n} \frac{1}{i!j!} D^{i+j}_{\alpha,\beta} b^n(\epsilon^3) \tau_1^i \tau_2^j. \tag{2.25}$$

The mixed derivatives appearing in (2.25) may be computed using the de Casteljau algorithm, as was done in (2.13):

$$D^{i+j}_{\alpha,\beta} b^n(\epsilon^3) = \frac{n!}{(n-i-j)!} \sum_{|\lambda|=i} \sum_{|\mu|=j} b^{i,j}_{\lambda+\mu}(\alpha, \beta) B^{n-i-j}_{\lambda+\mu}(\epsilon^3).$$

Due to the evaluation at ϵ^3, this expression may be simplified considerably, and one obtains

$$a_{i,j} = \frac{n!}{(n-i-j)!} b^{i,j}_{(0,0,n-i-j)}(\alpha, \beta).$$

Since and β are parallel to edges, each $b^{ikj}_{(0,0,n-i-j)}(\alpha, \beta)$ can be interpreted as an iterated mixed difference:

$$b^{i,j}_{(0,0,n-i-j)}(\alpha, \beta) = \Delta^{i,j,0} b_{0,0,n-i-j}, \tag{2.26}$$

where

$$\Delta^{1,0,0} b_\mu = b_{\mu+\epsilon^1} - b_{\mu+\epsilon^3} \quad \text{and} \quad \Delta^{0,1,0} b_\mu = b_{\mu+\epsilon^2} - b_{\mu+\epsilon^3},$$

see also Fig. 2.2.

Because of their relationship to intermediate de Casteljau points, the $a_{i,j}$ can conveniently be computed recursively: to compute $b^{i,j}_{(0,0,n-i-j)}(\alpha, \beta)$, apply the de Casteljau algorithm i times with respect to α and j times with respect to β. These steps can be carried out in any order. Each such de Casteljau step corresponds to a difference step; the possible combinations of difference steps are illustrated in Fig. 2.8.

The above monomial expansion amounts to the elimination of the third barycentric coordinate τ_3 from the standard expression for $b^n(\tau)$. Sometimes a different monomial expansion may be desired: the expression of $b^n(\tau)$ in terms of the global coordinates x, y, i.e., an expression of the form (2.23). In order to do that, one simply expresses the x- and y-directions in terms of the barycentric coordinates of the domain triangle $\mathcal{T}$: the x-direction corresponds to the difference ξ of the barycentric coordinates of (1, 0) and (0, 0), and the y-direction corresponds to the barycentric difference η of (0, 1) and (0, 0). With these definitions, the monomial expansion of b^n becomes:

$$b^n(\tau) = \sum_{i+j \le n} \frac{1}{i!j!} D^{i+j}_{\xi,\eta} b^n(\epsilon^3)(x - x_3)^i (y - y_3)^j, \tag{2.27}$$

where (x_3, y_3) are the (x, y)-coordinates of T_3, corresponding to ϵ^3.

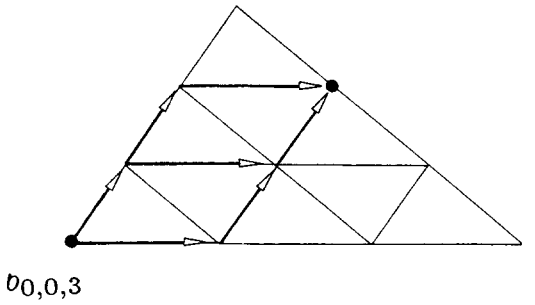


Fig. 2.8. Taylor expansion: the differences (indicated by arrows) that determine $a_{1,2} = \Delta^{1,2,0} b_{0,0,0}$. Three different 'paths' are possible for the evaluation of the iterated difference.

The derivatives appearing in (2.27) can again be computed using the de Casteljau algorithm; of course they cannot be computed using simple differencing any more.

The conversion to monomial form may be useful if a large number of points is to be evaluated on the patch: a monomial (2.23) can be evaluated by using nested multiplication, and the number of necessary operations is of order n^2. This compared favorably with the order n^3 computation count for the de Casteljau algorithm.

Some remarks may be useful concerning the choice of the evaluation algorithm: (1) The conversion from Bézier to monomial form makes use of nonconvex combinations, and is thus not very stable numerically. (2) The de Casteljau algorithm provides the tangent plane together with the point of evaluation, and this is very useful for algorithms such as shading or plane intersections.

Recently Schumaker and Volk [Schumaker, Volk '86] have addressed the problem of alternative representations of Bernstein–Bézier patches.

We finish this section by giving a special form for *quadratic* patches. A quadratic Bernstein–Bézier polynomial $b^2(\tau)$ can be written as a quadratic form:

$$b^2(\tau) = (\tau_1, \tau_2, \tau_3) \begin{pmatrix} b_{2,0,0} & b_{1,1,0} & b_{1,0,1} \\ b_{1,1,0} & b_{0,2,0} & b_{0,1,1} \\ b_{1,0,1} & b_{0,1,1} & b_{0,0,2} \end{pmatrix} \begin{pmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{pmatrix}. \tag{2.28}$$

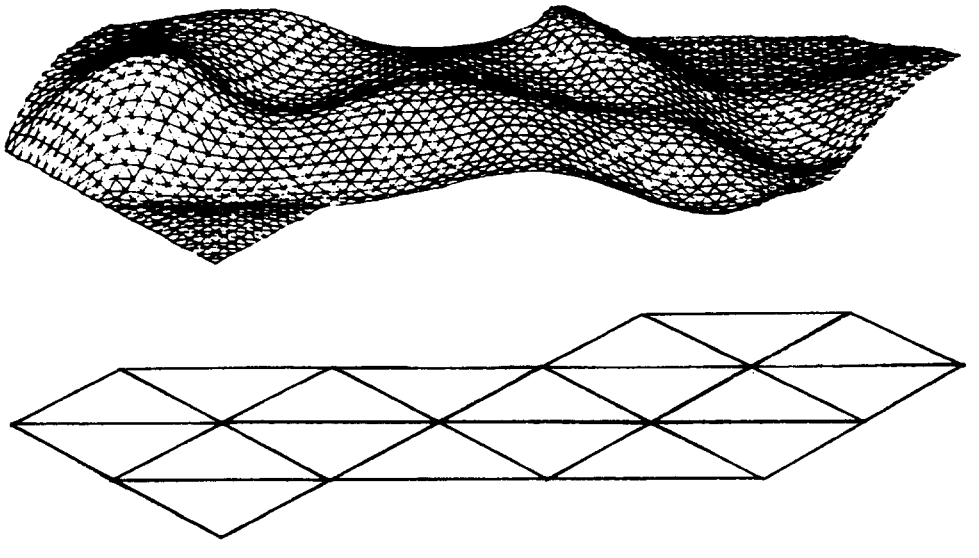


Fig. 2.9. C^1 piecewise cubic surfaces: an example surface. A regular triangulation was chosen in order to produce C^1 isoparametric curves.

Since the barycentric coordinates τ_i are related to the (x, y)-coordinates by a linear transformation, it is clear that (2.28) represents a quadric surface, more specifically, a paraboloid.

We conclude Section 2 with an illustration of a piecewise cubic C^1 surface, see Fig. 2.9.

3. Shape considerations

3.1. Convexity

A function f is called *convex* over the domain triangle $\mathcal{T}$ if at any point τ, every second directional derivative $D_\alpha^2 f(\tau)$ is nonnegative:

$$D_\alpha^2 f(\tau) \geq 0 \quad \text{all } \alpha, \quad \text{all } \tau. \tag{3.1a}$$

This definition is only meaningful for functions $f \in C^2(\mathcal{T})$; for functions that are only continuous, one defines

$$f\left(\tfrac{1}{2}\tau^1 + \tfrac{1}{2}\tau^2\right) \leq \tfrac{1}{2} f(\tau^1) + \tfrac{1}{2} f(\tau^2); \quad \text{all } \tau^1, \tau^2 \in \mathcal{T} \tag{3.1b}$$

as the condition defining convexity. For C^2 functions, both conditions are equivalent.

We shall rewrite (3.1a) in a more convenient form and to that end we introduce the notion of *edge derivatives* D_i. They are directional derivatives in the directions parallel to the edges δ^i, where $\delta^1 = \epsilon^3 - \epsilon^2$, $\delta^2 = \epsilon^1 - \epsilon^3$, $\delta^3 = \epsilon^2 - \epsilon^1$:

$$D_i := D_{\delta^i}.$$

Since one can write any direction α as a linear combination of the edge directions δ^i:

$$\alpha = \sum_{i=1}^{3} \beta_i \delta^i$$

where $\Sigma \beta_i = 0$ (e.g. $\beta_1 = (\alpha_3 - \alpha_2)/3$, $\beta_2 = (\alpha_1 - \alpha_3)/3$, $\beta_3 = (\alpha_2 - \alpha_1)/3$), we can write

$$D_\alpha = \sum_{i=1}^{3} \beta_i D_i.$$

Note that the representation of α in terms of the δ^i is not unique.

Edge twists $D_{i,j}^2 = D_{\delta^i, \delta^i}^2$ defined by (2.14), can be expressed as

$$D_{i,j}^2 := \begin{cases} D_i^2 & \text{if } i = j, \\ -D_i D_j & \text{if } i \neq j. \end{cases} \tag{3.2}$$

The minus sign in this definition selects the directions pointing from a vertex to the other vertices. The derivatives D_i and $D_{i,j}^2$ can be obtained from the Bézier ordinates by a differencing process, as outlined in Section 2.4 and illustrated in Figs. 2.2 and 2.8.

Now any second directional derivative with respect to a direction α may be written as

$$D_\alpha^2 = \sum_{i=1}^{3} \sum_{j=1}^{3} \beta_i \beta_j D_{i,j}^2. \tag{3.3}$$

The $D_{i,j}^2$ are the elements of the Hessian, which is symmetric: $D_{ij}^2 = D_{j,i}^2$. Just as $D_1 + D_2 + D_3 = 0$, the elements of the Hessian are not independent of each other:

Lemma 3.1. *For each row, the diagonal elements of the Hessian are the sum of the off-diagonal elements:*

$$D^2_{i,i} = D^2_{i,j} + D^2_{i,k}, \quad i,\ j,\ k \quad \textit{all different}. \tag{3.4}$$

Proof. We prove the case $i = 1$:

$$D^2_{1,3} = -D^2_{\delta^1,\delta^3} = -D^2_{\delta^1,(-\delta^1-\delta^2)} = D^2_{1,1} - D^2_{1,2}.$$

Lemma 3.1 has an important consequence:

Theorem 3.2. *For a C^2 function $f(\tau)$ to be convex over the domain triangle $\mathcal{T}$, it is sufficient that all edge twists be nonnegative, i.e.*

$$D^2_{i,j} f(\tau) \geq 0, \quad i \neq j,\ \tau \in \mathcal{T}. \tag{3.5}$$

Proof. Lemma 3.1 together with (3.5) implies weak diagonal dominance of the Hessian, thus the quadratic form (3.3) is nonnegative for any α, and convexity of f is assured.

We are interested in the special case where f is a Bernstein–Bézier polynomial b^n. Equation (2.14) provides a formula for edge twists. If all coefficients of the twist surface are nonnegative, then it is nonnegative itself:

Corollary 3.3. *A sufficient condition for b^n to be convex is*

$$\left.\begin{aligned} (b_{\mu+\epsilon^1+\epsilon^2} - b_{\mu+\epsilon^2+\epsilon^3}) - (b_{\mu+\epsilon^1+\epsilon^3} - b_{\mu+2\epsilon^3}) \geq 0 \\ (b_{\mu+\epsilon^2+\epsilon^3} - b_{\mu+\epsilon^1+\epsilon^3}) - (b_{\mu+\epsilon^1+\epsilon^2} - b_{\mu+2\epsilon^1}) \geq 0 \\ (b_{\mu+\epsilon^1+\epsilon^3} - b_{\mu+\epsilon^1+\epsilon^2}) - (b_{\mu+\epsilon^2+\epsilon^3} - b_{\mu+2\epsilon^2}) \geq 0 \end{aligned}\right\} \quad |\mu| = n-2. \tag{3.6}$$

This special case is due to [Chang, Davis '84]; the above proof, however, is more general. A geometric interpretation of Corollary 3.3 is that all subnets of order two of the control net form convex piecewise linear surfaces (with a suitable definition of convexity). A sharper condition is given in [Chang, Feng '85].

3.2. The variation diminishing property

Bézier curves possess the *variation diminishing property*: no straight line (or plane) has more intersections with the curve than with the defining polygon. For a proof, see [Lane, Riesenfeld '82]. A possible way to extend this definition to triangular patches is as follows: count the number of closed intersection curves of the control net with any plane and compare it to the number of closed intersection curves of the surface with the same plane. Under such a definition, Bernstein–Bézier triangular patches do not possess the variation diminishing property, as counterexamples by [Prautzsch '85] show. Another possibility would be to count the number of local extrema of both the net and the surface: again, under this definition triangular patches do not possess the variation diminishing property.

It is possible, however, to define the concept of variation for patches defined over triangles in terms of some set of suitable functionals. With these proper definitions, Bernstein–Bézier patches over triangles do possess a variation diminishing property [Chang, Hoschek '85], [Goodman '85], [Chang, Feng '85].

II. APPLICATIONS

4. Hermite interpolants

In this section we shall discuss interpolants that are defined over a triangle $\mathcal{T}$ and interpolate to position and derivative information at the vertices and across edges. This information could either come from some known primitive function $f(\tau)$, or it could be given in the form of discrete measurements (typically function and gradient values). Such interpolants are abundant in the finite element literature [Strang, Fix '73]. For the purposes of a finite element solver, no explicit form of the interpolant is necessary, however, and so some of the presented interpolants (e.g. Q_{21}, Q_{18}) are schemes known from finite elements and rewritten in a form suitable for CAGD (see, e.g., [Barnhill, Farin '81], [Grieger '85], [Sablonnière '85].)

4.1. The C^0 nine parameter interpolant

The nine given data are position and gradient at each triangle vertex, i.e.

$$D^{r,s}_{\alpha,\beta} f(\epsilon^1); \quad r+s \le 1$$

with $\alpha = \epsilon^2 - \epsilon^1$, $\beta = \epsilon^3 - \epsilon^1$ and analogous data for the remaining two vertices ϵ^2, ϵ^3. The computation of the nine boundary Bézier ordinates reduces to univariate cubic Hermite interpolation to each boundary edge. We obtain

$$b_{3,0,0} = f(\epsilon^1), \qquad b_{2,1,0} = \tfrac{1}{3} D_\alpha f(\epsilon^1) + b_{3,0,0}, \tag{4.1}$$

and the remaining Bézier ordinates by symmetry.

The only Bézier ordinate not found this way is $b_{1,1,1}$. It is independent of the prescribed data and can be assigned any arbitrary value. A reasonable choice is to select $b_{1,1,1}$ such that quadratic precision of the interpolant is maintained, i.e., if the nine prescribed data were read off from a quadratic, then the interpolant would reproduce this quadratic. If this quadratic was written as a cubic (by the process of degree elevation, see Section 1), the coefficient $b_{1,1,1}$ could be expressed as

$$\begin{aligned} b_{1,1,1} = {} & \tfrac{1}{4}(b_{2,0,1} + b_{1,0,2} + b_{0,2,1} + b_{0,1,2} + b_{2,1,0} + b_{1,2,0}) \\ & - \tfrac{1}{6}(b_{3,0,0} + b_{0,3,0} + b_{0,0,3}), \end{aligned} \tag{4.2}$$

thus choosing $b_{1,1,1}$ according to (4.2) ensures quadratic precision of the interpolant. P. Alfeld [Alfeld '85] has point out that other choices of $b_{1,1,1}$ also ensure quadratic precision; it is an open question if any of these choices can be sensibly labeled 'best'. In this context it is interesting to point out that experimental results obtained by B. Piper suggest that the concept of quadratic precision is totally inadequate when it comes to *parametric* interpolants, see Section 8 and [Piper '86].

In a situation where data are prescribed at several triangles in a triangulation, the above interpolant has a serious drawbacks: it requires C^1 data, but the produced overall surface is only C^0. This is easily seen from the fact that the value of $b_{1,1,1}$ depends on data prescribed around the whole triangle instead of data pertaining only to one edge.

4.2. C^1 quintic interpolants

In order to produce overall C^1 surfaces defined over a triangulation, one has to prescribe more data, namely up to second derivative data at the vertices and a cross-boundary derivative

at each edge midpoint. The resulting interpolant will now be at least quintic (see [Barnhill, Farin '82]). The prescribed vertex data are:

$$D^{r,s}_{\alpha,\beta} f(\epsilon^1); \quad r+s \le 2$$

with $\alpha = \epsilon^2 - \epsilon^1$, $\beta = \epsilon^3 - \epsilon^1$ and analogous data for the remaining two vertices. In order to make the interpolant unique, one has to prescribe three more pieces of information, and one standard choice is the prescription of cross-boundary derivatives D_γ at the edge midpoints:

$$D_\gamma f\left(\frac{\epsilon^2+\epsilon^3}{2}\right),$$

where γ is a direction not parallel to the edge $[\epsilon^2, \epsilon^3]$, and analogous data for the remaining two edges. Thus we have 21 pieces of information for the quintic interpolant, which is determined by 21 coefficients. The given data specify these coefficients uniquely, as follows from the subsequent construction of the interpolant, which is also known as Q_{21}. (See Fig. 4.1 for an illustration.)

The Bézier points of the quintic patch pertaining to data at vertex ϵ^1 are obtained from

$$\begin{aligned} b_{5,0,0} &= f(\epsilon^1), \\ b_{4,1,0} &= \tfrac{1}{5} D_\alpha f(\epsilon^1) + b_{5,0,0}, \\ b_{3,2,0} &= \tfrac{1}{20} D^2_\alpha f(\epsilon^1) + 2b_{4,1,0} - b_{5,0,0}, \\ b_{3,1,1} &= \tfrac{1}{20} D^2_{\alpha,\beta} f(\epsilon^1) + b_{4,1,0} + b_{4,0,1} - b_{5,0,0}. \end{aligned} \tag{4.3}$$

and analogously for the remaining two vertices.

The cross-boundary edge derivatives can be used to determine the remaining Bézier ordinates: for edge $[\epsilon^2, \epsilon^3]$, we obtain from (2.11)

$$D_\gamma f\left(\frac{\epsilon^2+\epsilon^3}{2}\right) = 5 \sum_{|\lambda^0|=4} b^1_{\lambda^0}(\gamma) B^4_{\lambda^0}\left(\frac{\epsilon^2+\epsilon^3}{2}\right).$$

This expression contains the unknown Bézier ordinate $b_{1,2,2}$, for which we can now solve:

$$b_{1,2,2} = \frac{1}{\gamma_1}\left[\frac{1}{5} D_\gamma f\left(\frac{\epsilon^2+\epsilon^3}{2}\right) - \sum_{\substack{|\lambda^0|=4 \\ \lambda^0 \neq (0,2,2)}} b^1_{\lambda^0}(\gamma) B^4_{\lambda^0}\left(\frac{\epsilon^2+\epsilon^3}{2}\right) - \gamma_2 b_{0,2,2} - \gamma_3 b_{0,2,3}\right]. \tag{4.4}$$

Eq. (4.4) is only meaningful if $\gamma_1 \neq 0$, which is equivalent to saying that γ is not parallel to edge $[\epsilon^2, \epsilon^3]$.

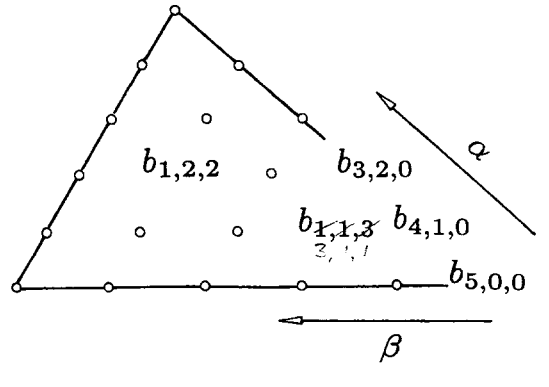


Fig. 4.1. The interpolant Q_{21}: the coefficients that are determined by equations (4.3) and (4.4). The remaining ones follow from symmetry.

The directions γ are usually chosen to be perpendicular to edges; thus for edge $[\epsilon^2, \epsilon^3]$ we obtain

$$\gamma = (1, \gamma_1, -1-\gamma_1),$$

where

$$\gamma_1 = \frac{(T_2 - T_3)^{\mathrm{T}}(T_1 - T_2)}{|T_2 - T_3|}.$$

The expressions for cross-boundary derivatives at the other two edges are completely analogous.

Often the cross-boundary derivative is not available in a specific problem. In this case one often makes use of *condensation of parameters*: The normal cross-boundary derivative of any edge, evaluated along that edge, is a univariate quartic polynomial. If we require all three of these polynomials to be cubic instead, we have three (linear) conditions from which we can compute the three Bézier ordinates not determined by the vertex data: the coefficients of the quartic cross-boundary derivative are the five $b^1_{\lambda 0}$. If a quartic is actually cubic, the fourth difference of its Bézier ordinates must vanish:

$$b^1_{0,0,4}(\gamma) - 4b^1_{0,1,3}(\gamma) + 6b^1_{0,2,2}(\gamma) - 4b^1_{0,3,2}(\gamma) + b^1_{0,4,0}(\gamma) = 0. \tag{4.5}$$

The term $b_{1,2,2}$ is defined as $b^1_{0,2,2} = \gamma_1 b_{1,2,2} + \gamma_2 b_{0,3,2} + \gamma_3 b_{0,2,3}$, and (4.5) can easily be solved for it.

If the same process is applied to all triangles in the triangulation, it is clear that all normal cross-boundary derivatives are continuous. Since only 18 pieces of information are prescribed per triangle this way, the resulting condensed interpolant is called Q_{18} in the literature (see [Barnhill, Farin '82]).

The interpolants consisting of piecewise quintic Q_{18} patches are local: if data are changed at only one data point, the interpolant is only affected over those triangles that have that data point as a vertex. In setting all data equal to zero at all but one data point, one may derive *cardinal functions* for the interpolation problem. These are identically zero except over the triangles pertaining to the data point under consideration. These cardinal functions are called *v-splines* (for vertex-splines) by [Chui '85].

4.3. The general case

In this section we shall discuss piecewise polynomial interpolants defined over a triangulation that generate C^r surfaces. Each interpolant is local, i.e., it is determined by data (linear or nonlinear functionals) that are defined over one triangle only. These local interpolants must be of a polynomial degree that depends on r:

Theorem 5.1. *A local piecewise polynomial interpolant that interpolates to all derivatives up to order r of some primitive function f at the vertices of a triangulation and that is globally r times differentiable must be of degree* $n \geq 4r + 1$.

Proof. We refer to Fig. 4.2, which illustrates the case $r = 3$. The interpolation requirement determines the degree r subnets at the triangle vertices. Sine we require C^r continuity across the boundaries, we have to specify all $r + 1$ rows of Bézier ordinates parallel to the edges. They are determined by cross-boundary derivatives that are also imposed on the neighboring triangles, thus enforcing C^r continuity. (These cross-boundary derivatives do not necessarily have to be those of f, since the interpolant only has to agree with f and its derivatives at the triangle vertices.) If no special care is taken, however, the ordinates in the shaded regions (Fig. 4.2) will be determined, namely those Bézier ordinates will be 'determined' by conflicting (since

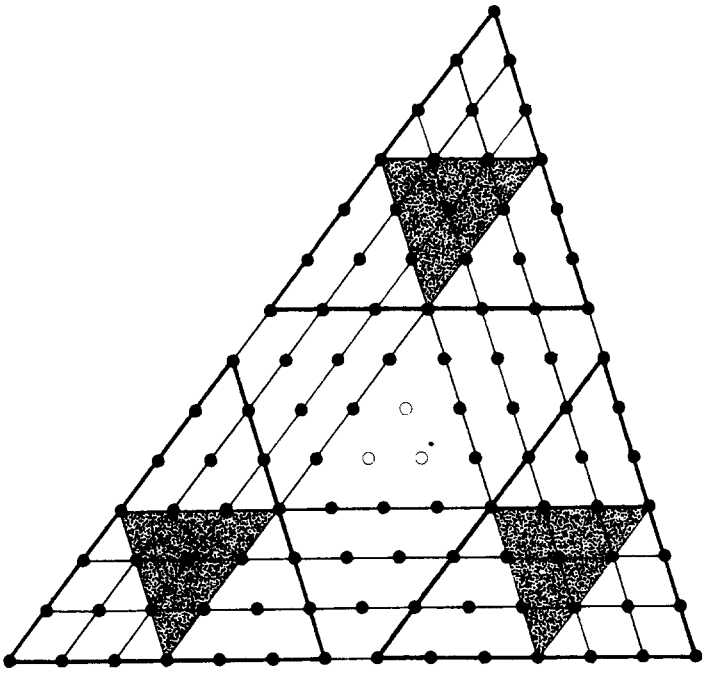


Fig. 4.2. Hermite interpolation: For the case $n = 13$, the full circles denote ordinates that can be directly inferred from the given data. In order for the ordinates in the shaded 'upside down' subnets not to be overdetermined, full C^5 data have to be prescribed at the vertices. The three center ordinates (marked by open circles) may be assigned arbitrarily.

independent) cross-boundary derivatives. In order to specify these Bézier ordinates unambiguously, all derivatives up to order $2r$ must be specified at each vertex. (They do not necessarily have to come from f, e.g., they could be set to zero.) In order for this information to be consistent (e.g. in order not to overdetermine the boundary curves), the polynomial degree of the interpolant must be at least $4r + 1$.

Hermite polynomials with $n = 4r + 1$ where first considered by [Zenisek '70]. We note that the derivative information specified at the vertices and across the edges is in general not sufficient to determine the interpolant uniquely: There are $r(r-1)/2$ Bézier ordinates left in the center region of the control net (see Fig. 4.2). These do not influence the smoothness of the global surface, but should be chosen carefully to yield 'good' local interpolants: if no additional information about the primitive function is available, the remaining Bézier ordinates could e.g. be determined such that the polynomial precision of the interpolant be as high as possible.

If not enough cross-boundary derivative information is available to specify the necessary $r + 1$ rows of Bézier ordinates, the process of condensation of parameters can be applied, this time not only to the first cross-boundary derivative, but to all up to order r.

A special case of the above interpolants is the quintic Q_{21} (resp. Q_{18}) from the preceding section. The case $r = 2$, yielding nonics, is discussed in [Whelan '85], [Rescorla '85], [Sablonnière '85].

5. Split triangle interpolants

The Hermite interpolants that were discussed in the previous section all share a serious flaw: in order to obtain a composite surface of some desired continuity class, the user has to provide derivative data of higher order than the desired order of continuity. Thus we have to provide all second derivatives at the vertices of a triangle to build the interpolants Q_{21} or Q_{18}, but we only achieve C^1 continuity of the overall surface. Obviously this situation is uneconomical.

The goal is thus to devise interpolants that require no derivative information of higher order than the order of continuity of the resulting interpolant. In the context of triangular interpolants, the key to the solution is to consider interpolants that are *piecewise polynomial* over

each triangle. Every triangle in the triangulation of the data points (in this context also called 'macro-triangle') is split into several 'mini-triangles'.

None of the following interpolants was originally developed in terms of Bernstein–Bézier patches. The use of this technique considerably simplifies the derivation of such interpolants. In fact, the easy construction of new interpolants is one of the main strengthes of this technique.

5.1. The C^1 Clough–Tocher interpolant

This interpolant is conceptually the simplest of all split-triangle interpolants, and it has been known in the finite element literature for some time [Strang, Fix '73], [Percell '76]. It is characterized by the 'simplest' symmetric split of a triangle: each vertex is joined to the centroid; thus a macro-triangle is split into three mini-triangles. (Any other interior point other than the centroid would do; the centroid is chosen for symmetry reasons.)

The first order data that this interpolant requires are position and gradient value at the vertices of the macro-triangle plus some cross-boundary derivative at the midpoint of each edge. (Note that this is a subset of the data for Q_{21} without the undesirable second order items.) The prescribed cross-boundary derivative could be in any direction not parallel to its edge; but since adjacent macro-triangles should share the same data along the common edge, it is most natural to choose the direction perpendicular to that edge. We then speak of a *cross boundary normal derivative*.

In summary, one has twelve data per macro-triangle. It is easily seen that interpolation to this data produces a globally C^1 surface if cubic polynomials are employed over each mini-triangle.

We shall now turn to the description of the actual interpolant; we refer to Fig. 5.1. The nine Bézier ordinates of the three boundary curves (marked by full circles) are found exactly as for the nine-parameter interpolant, see Section 5.1. The next 'layer' of ordinates, marked by full circles and triangles, is determined if we enforce interpolation to the cross-boundary derivatives. The cross-boundary derivative, evaluated along an edge, is a univariate quadratic polynomial. It can be written as a univariate Bézier polynomial with three coefficients, the first and last one being determined by the gradients at the vertices, the center one as well by the cross-boundary derivative at the midpoints of that edge. The formulas are completely analogous to (4.4) (see [Barnhill, Farin '81]).

We are still left with the task of specifying the four remaining ordinates. Since the interpolant must be C^1 over each macrotriangle, those ordinates must satisfy the C^1-conditions

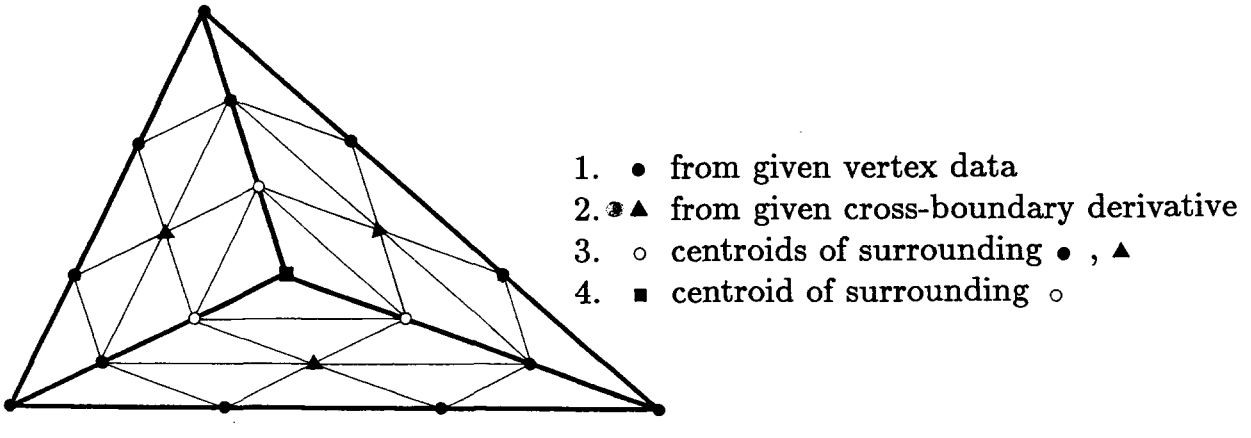


Fig. 5.1. The Clough–Tocher subnets: The macro-triangle is split into three mini-triangles and a cubic is built over each.

(2.22). Thus each of the three outer ordinates of the four ones under consideration must be the average of the adjacent three ordinates that have already been determined. Finally, the center ordinate must be the symmetric average of the three ones just found.

In many applications, we will not be given the required cross-boundary derivatives at the edge midpoints. The most obvious method to estimate this derivative is again *condensation of parameters*: For each edge of the macro-triangle, the cross-boundary derivatives can be computed at its two endpoints. The midpoint cross-boundary derivative is simply set to be the average of those values.

Some slightly more involved methods to estimate the midpoint cross-boundary derivative consider two adjacent macro-triangles at a time. We can construct the nine-parameter interpolant over each triangle and compute their respective midpoint cross-boundary derivatives. They will in general not agree; one way to specify a derivative that is common to both triangles is as follows: determine the common midpoint cross-boundary derivative such that it deviates as little as possible from those of the nine-parameter interpolants. This leads to a least squares problem that can be solved explicitly [Farin '83]. Similarly, one may determine the midpoint cross-boundary derivative from the requirement that the C^2 discontinuity between both patches be as small as possible [Farin '85]. Both methods yield surfaces that are smoother than those obtained from standard condensation of parameters. Also, these methods can easily be carried over to the parametric case, where the concept of normal cross-boundary derivatives becomes meaningless, see Section 8.

We conclude this section with a somewhat surprising result (recall that the Clough–Tocher interpolant is designed to be C^1):

Theorem 5.1. *The Clough–Tocher interpolant is C^2 at the centroid of the macro-triangle.*

For a proof of this statement, the following lemma is needed:

Lemma 5.2. *C^1 piecewise quadratics over a Clough–Tocher triangle split are also C^2, i.e., they are one global quadratic.*

Proof. Consider Fig. 5.2. Suppose the six solid Bézier ordinates were arbitrarily chosen. Then the Bézier ordinates marked by open circles are determined by the C^1 conditions between subpatches. But then the ordinate marked by a square is also determined by the C^1 conditions. Hence the three C^1 quadratics are uniquely determined by the six solid Bézier ordinates. Since any three C^1 quadratics can be constructed this way, it follows that the dimension of C^1

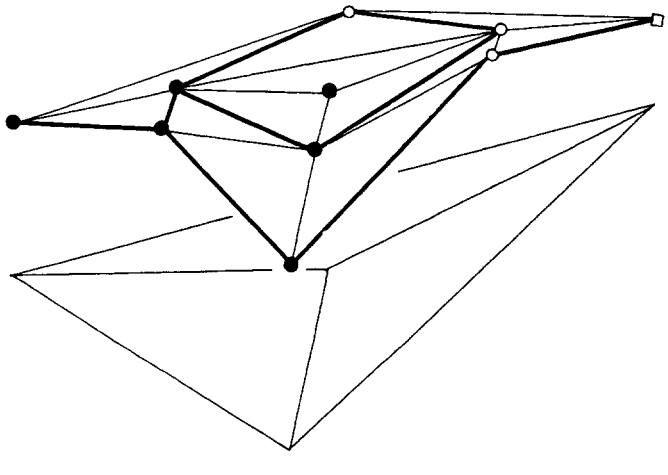


Fig. 5.2. C^1 piecewise quadratics: If the ordinates mrked by solid circles are arbitrarily specified, the remaining ones are determined by the C^1 condition: the ordinates marked by open circles lie in planes already determined by three solid ordinates. The ordinate marked by a square has to lie in the plane defined by the three open circles.

piecewise quadratics over the Clough–Tocher split is six, which equals the dimension of the space of global quadratics. Since C^1 piecewise quadratics are a superset of the global ones, the lemma is proved.

Returning to the *Proof of Theorem 5.1*, one observes that the Clough–Tocher control net (see Fig. 5.1) has a piecewise quadratic subnet obtained by 'stripping away' the control net edges. The remaining subnet is of the form of Fig. 5.2 and determines all derivatives at the centroid. Since this piecewise quadratic subnet is C^1 by construction of the interpolant, it represents a global quadratic according to the above lemma. Thus all derivatives of order two at the centroid are continuous, which completes the proof of Theorem 5.1.

5.2. Limitations of the Clough–Tocher split

The data stencil for the C^1 Clough–Tocher interpolant can be characterized as follows: (a) prescribe enough data at the (macro-) triangle vertices to ensure C^1 continuity there; (b) also prescribe enough cross-boundary derivative information to uniquely determine the (C^1) cross-boundary derivative.

A natural generalization would be to replace C^1 by C^2 in the above data stencil description and then to proceed to construct a C^2 interpolant over the Clough–Tocher split. It turns out however, that the resulting interpolant would be overdetermined [Farin '80].

In order to see this, we refer to Fig. 5.3, which shows part of the Bézier net of an alleged C^2 Clough–Tocher interpolant. The polynomials over each mini-triangle are at least quintic now. In order to satisfy the requirements of the generalized steps (a) and (b) from above, one has to generate enough C^2 information to determine all ordinates marked by full circles or squares. We note that the ones marked by squares are determined *independently* by C^2 cross-boundary derivatives that have to be prescribed at the two shown macro-triangle edges.

Now consider the two quadratic subnets that are shaded in Fig. 5.3. By C^2 continuity, they have to represent the same quadratic polynomial (see the proof of Theorem 2.8). A quadratic is

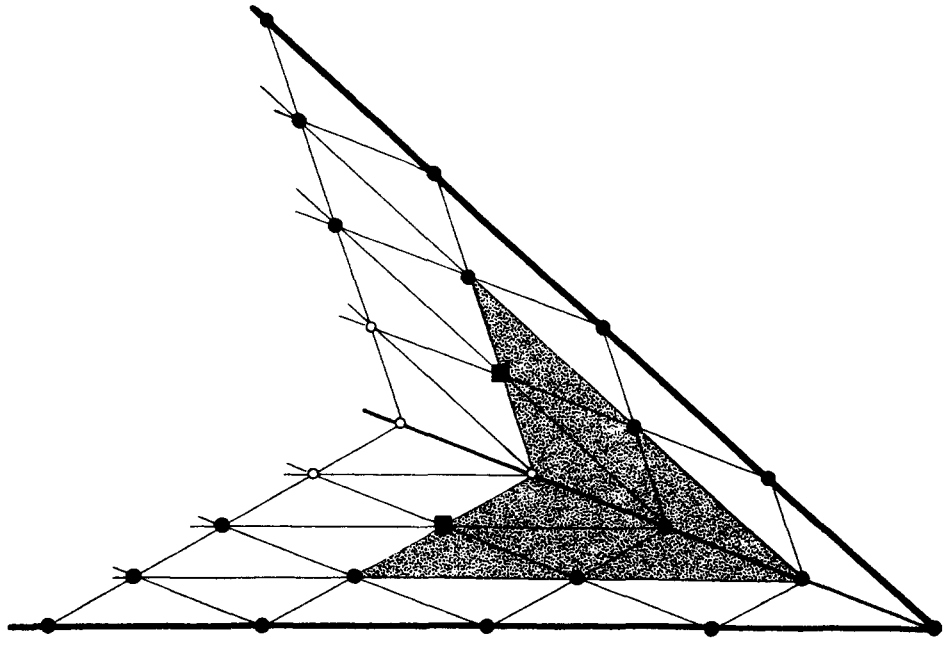


Fig. 5.3. An overdetermined C^2 interpolant: With only C^2 data prescribed at the vertices, the ordinates denoted by squares are overdetermined.

determined by six independent pieces of information. Counting the data that determine the two shaded quadratics, one finds five ordinates on the edges parallel to the macro-triangle edges plus the two ordinates marked by squares. Thus the proposed interpolation problem would result in finding a quadratic determined by seven independent pieces of information, which is clearly overdetermined.

In order to formulate split triangle C^2 interpolants, one must therefore (i) prescribe at least C^3 data at the vertices with a Clough–Tocher split or (ii) use splits with at least three mini-triangles meeting at the macro-triangle vertices.

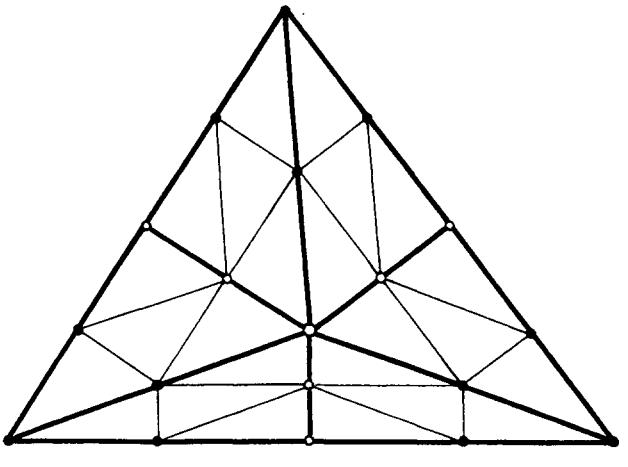
A scheme corresponding to (i) has been studied by [Sablonnière '85]: he considers C^2 piecewise seventh degree patches defined over the Clough–Tocher split with C^3 data at the macro-triangle vertices. A scheme corresponding to (ii) has been considered by [Alfeld '84b]: if the Clough–Tocher split is iterated, i.e., each mini-triangle resulting from the Clough–Tocher split is again subjected to a Clough–Tocher split, enough degrees of freedom exist to construct a C^2 piecewise quintic over this split. Although a solution to the given problem, the resulting surfaces often show unwanted oscillations.

5.3. *The C^1 Powell–Sabin interpolants*

These interpolants produce C^1 piecewise quadratic interpolants to C^1 data at the vertices of a triangulated data set [Powell, Sabin '77]. Each macro-triangle is split into several mini triangles; their number is determined by the macro-triangle geometry: if all angles are smaller than 75 degrees, a split into six mini-triangles is performed, see Fig. 5.4. Otherwise, twelve mini-triangles are generated, see Fig. 5.5. (The 75 degree case distinction is a purely heuristic one.)

Discussion of both interpolants:

(a) *The six-triangle split.* The macro-triangle is split by joining the center of the circumscribed circle to the edge midpoints, see Fig. 5.4. The Bézier ordinates of the quadratic mini-patches are determined in three steps as shown in Fig. 5.4. Note that as a consequence of the special geometry of this split we have that the normal cross-boundary derivatives, evaluated along the edges, are linear instead of piecewise linear. Thus any two adjacent macro-triangles of this type will be differentiable across their common edge.



1. • from given data
2. ○ on straight lines between •
3. ○ in plane of surrounding ○ ,

Fig. 5.4. The Powell–Sabin interpolant I: The macro-triangle is split into six mini-triangles and a quadratic is built over each.

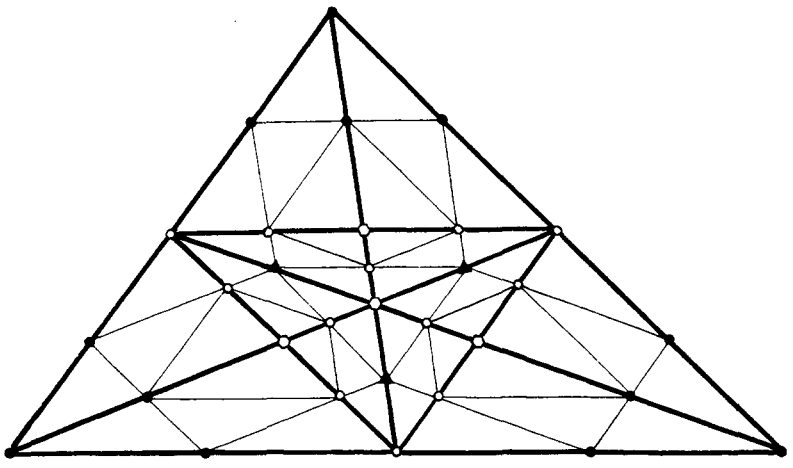


Fig. 5.5. The Powell–Sabin interpolant II: The macro-triangle is split into twelve mini-triangles and a quadratic is built over each.

This argument relies on the fact that the line joining two adjacent split-points intersects the common edge at the point that is used to subdivide that edge. (Thus incenters could be used as split-points as well.) For triangles with obtuse angles, the center of the circumscribed circle will not be within the macro-triangle or it may be so close to an edge that undesirably 'skinny' mini-triangles would result. In these cases, a different split is performed (although the use of incenters would make this unnecessary):

(b) *The twelve-triangle split.* The macro-triangle is split by joining its centroid to the edge midpoints and by joining the edge midpoints to one another, see Fig. 5.5. The Bézier ordinates of the quadratic mini-patches are determined in three steps as shown in Fig. 5.5. The ordinates mrked by triangles are determined such that the normal cross-boundary derivative, evaluated along an edge, is linear instead of piecewise linear. Thus any two adjacent patches of this type will be differentiable across their common boundary. By the same argument, these patches will form C^1 surfaces when joined to the six-triangle macro-patches from part (a).

One disadvantage of the original Powell–Sabin interpolants is that a case distinction is necessary regarding the number of mini-triangles per macro-triangle. It is important to note, however, that a valid triangulation is always guaranteed (i.e. the set of mini-triangles forms a valid triangulation).

Higher order Powell–Sabin elements are considered by [Sablonnière '85].

5.4. C^1 split square interpolants

The interpolants that were so far discussed in this section were all aiming at the solution of the scattered data interpolation problem. Triangular interpolants have also been used, however, to interpolate to data from a rectangular grid. Most of these interpolants are quadratic, for the reason of ease of contouring [Powell '74]. We shall discuss two interpolants due to [Sibson '80]. The reason for including a cubic interpolant here is mainly to illustrate the ease at which such interpolants can be devised using the Bernstein–Bézier technique.

We refer to Fig. 5.6 and 5.7 for the discussion of the interpolants. The given data for this problem are positional and gradient values at data points that are located on a rectangular grid.

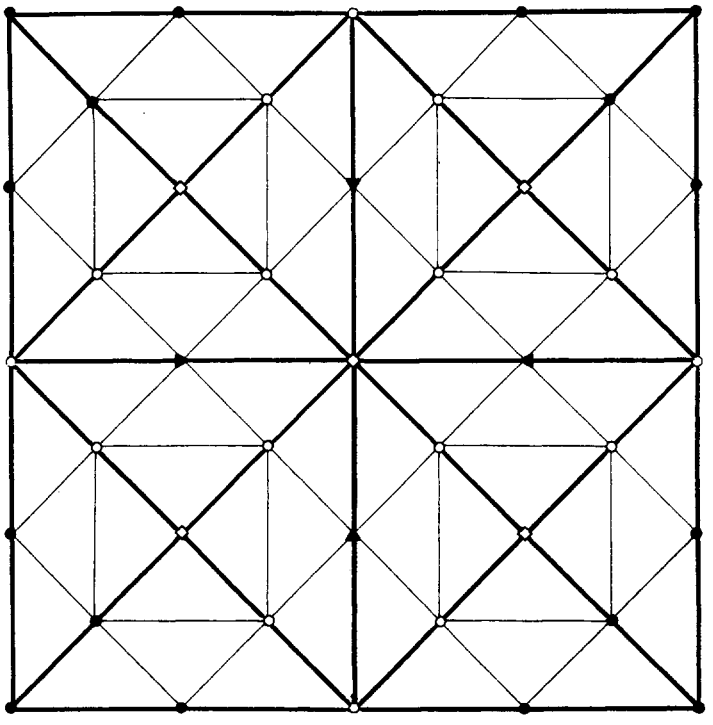


Fig. 5.6. Sibson's split-square interpolant I: The square is split into sixteen mini-triangles and a quadratic is built over each.

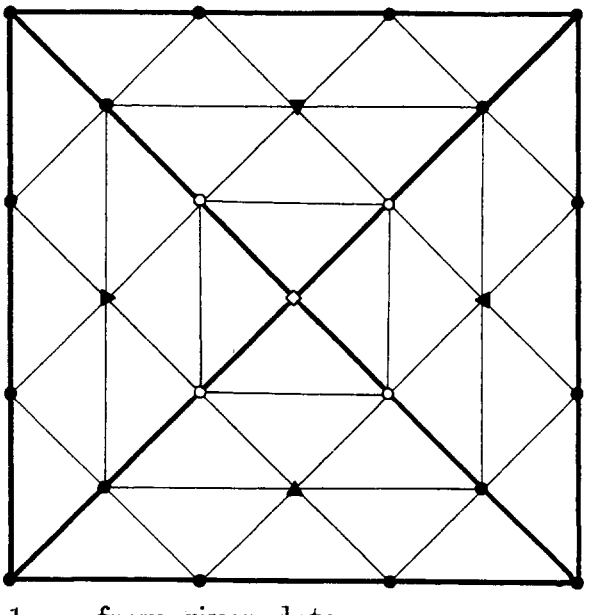


Fig. 5.7. Sibson's split-square interpolant II: The square is split into four mini-triangles and a cubic is built over each.

Each rectangle is subdivided into sixteen (Fig. 5.6) or four (Fig. 5.7) mini-triangles. While the subdivision into sixteen triangles yields a C^1 piecewise quadratic interpolant, that into four triangles yields C^1 piecewise cubic interpolant. The actual construction of the interpolants follows the principle already encountered for the Clough–Tocher and Powell–Sabin interpolants: translate the given data into Bézier ordinate values along the boundaries and then compute the remaining ordinates from the C^1-conditions.

The ordinates mrked by triangles in Fig. 5.6 are obtained from the requirement that the normal cross-boundary derivatives be linear instead of piecewise linear. Those marked by triangles in Fig. 5.7 are obtained by the similar requirement that the normal cross-boundary derivative be linear instead of quadratic. Both constructions thus ensure that the interpolants be C^1 globally.

6. The N-dimensional case

6.1. Basic results

While we have so far considered Bernstein–Bézier surfaces, mappings $\mathbb{R}^2 \to \mathbb{R}$, it is possible to generalize most of the previous concepts to mappings $\mathbb{R}^N \to \mathbb{R}$. The resulting 'surfaces' will be called *multivariate* Bernstein–Bézier polynomials. The underlying basic theory does not need any new derivation — it is sufficient to reread sections 1 and 2 and to simply replace the summation limit '3' by '$N+1$' (as in (1.1)). Greek letters will now denote multi-indices, e.g. $\mu = (\mu_1, \ldots, \mu_{N+1})$ etc. Thus barycentric coordinates τ are now defined by

$$P = \sum_{i=1}^{N+1} \tau_i T_i; \quad P, T_i \in \mathbb{R}^N, \tag{6.1}$$

where $|\tau| = 1$ and the T_i span a simplex $\mathcal{S}_N$ in $\mathbb{R}^N$. A Bernstein polynomial of degree n is defined by

$$B_\lambda^n(\tau) = \frac{n!}{\lambda!}\tau^\lambda; \quad |\lambda| = n \tag{6.2}$$

where $\lambda! = \lambda_1! \cdot \lambda_2! \cdots \lambda_{N+1}!$ and $\tau^\lambda = \tau_1^{\lambda_1} \cdots \tau_{N+1}^{\lambda_{N+1}}$. A Bernstein–Bézier polynomial $b^n(\tau)$ is defined by

$$b^n(\tau) = \sum_{|\lambda|=n} b_\lambda B_\lambda^n(\tau), \tag{6.3}$$

i.e., by a formula that is identical to (1.10) except that now λ and τ have $N+1$ components.

The de Casteljau algorithm becomes

$$b_\lambda^{r+1}(\tau) = \sum_{i=1}^{N+1} \tau_i b_{\lambda+\epsilon^i}^r(\tau); \quad |\lambda| = n-r; \quad r = 0, \ldots, n \tag{6.4a}$$

or, equivalently,

$$b_\lambda^{r+1}(\tau) = \sum_{|\epsilon|=1} B_\epsilon^1(\tau) b_{\lambda+\epsilon}^r(\tau); \quad |\lambda| = n-r, \epsilon \in \mathbb{Z}_+^{N+1}; \quad r = 0, \ldots, n. \tag{6.4b}$$

The intermediate coefficients $b_\lambda^r(\tau)$ provide the Bézier ordinates of the subdivided patch b^n, see also [Goldman '83].

In order to discuss C^r continuity between patches, whose graph is in $\mathbb{R}^{n+1}$, we have to consider two adjacent simplices in $\mathbb{R}^N$, say $\mathcal{S}_N^1$ and $\mathcal{S}_N^2$. In terms of barycentric coordinates of $\mathcal{S}_N^1$, its vertices are given by

$$\mathcal{S}_N^1 = \{\epsilon: |\epsilon| = 1, \epsilon \in \mathbb{Z}_+^{N+1}\}.$$

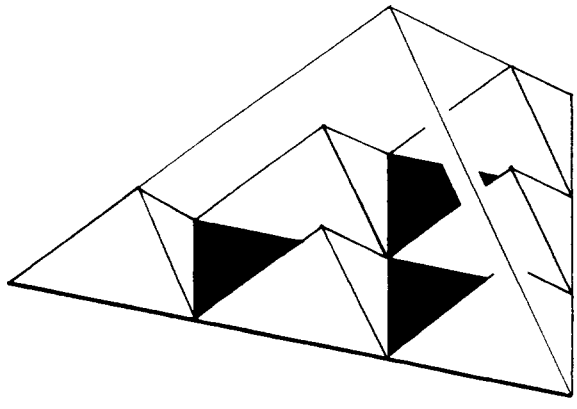


Fig. 6.1. Trivariate Bernstein–Bézier polynomials: The domain tetrahedron of a cubic Bernstein–Bézier polynomial is shown together with some of the subsimplices into which it is split. No interior Bézier ordinates exist for this case.

The vertices of $\mathcal{S}_N^2$ can also be expressed in terms of barycentric coordinates of $\mathcal{S}_N^1$:

$$\mathcal{S}_N^2 = \mathcal{S}_N^1 \setminus \{\epsilon^1\} \cup \{\sigma\}; \quad \sigma \in \mathbb{R}^{N+1}, \ |\sigma| = 1.$$

The only vertices at which $\mathcal{S}_N^1$ and $\mathcal{S}_N^2$ disagree are thus ϵ^1 and the arbitrary point σ.

Let a Bernstein–Bézier polynomial over $\mathcal{S}_N^1$ be defined by Bézier ordinates b_λ and one over $\mathcal{S}_N^2$ by $\hat{b}_\lambda$. Then the condition for C^r continuity between both patches is given by

$$\hat{b}_{\lambda^s} = b^s_{\lambda^0}(\sigma) \tag{6.5}$$

where λ^s is constrained by $\lambda_1 = s$.

We annotate a property of multivariate Bernstein–Bézier polynomials that becomes meaningful only in higher dimensions. Let us define as 'inner' Bézier ordinates all b_λ for which all $\lambda_i \neq 0$. The 'outer' ordinates thus correspond to points on the faces of the simplex $\mathcal{S}_N$. Then we have the following result: for a given N, all b^n with $n \leq N+1$ have no inner Bézier ordinates. This follows trivially since for $\lambda \in \mathbb{Z}_+^{N+1}$ the condition $|\lambda| = n$ implies $\lambda_i = 0$ for some i if $n \leq N+1$. Thus for example cubics have no inner Bézier ordinates for $N \geq 3$. This property is useful for the construction of multivariate interpolants. The case $N = 3$, $n = 3$ is illustrated in Fig. 6.1.

A much more detailed treatment of the N-dimensional case can be found in [de Boor '86a, '86b]. Results concerning convexity of multivariate Bernstein–Bézier patches can be found in [Dahmen, Micchelli '85].

6.2. The generalized nine parameter interpolant

While the purely theoretical results are carried over to the N-dimensional case almost instantaneously, the same is not true for the interpolants that were previously discussed. We shall start with a generalization of the simplest of those interpolants, the nine parameter interpolant from Section 4.1.

Suppose over a simplex $\mathcal{S}_N$, we are given position and gradient values at each vertex. We attempt to construct a cubic interpolant to this data and assume $N \geq 3$, so that by the above remark the interpolant will have no inner Bézier ordinates. The construction of the interpolant is recursive: suppose we can find generalized nine parameter interpolants to the $(N-1)$-dimensional data sets that are defined over the $N+1$ faces of $\mathcal{S}_N$. Then we have already found the Bézier ordinates of the final interpolant, since it has only outer ordinates. Continuing this reasoning, the first time we have to actually *compute* an interpolant is when the recursion reaches two-dimensional faces. Over these we apply the standard nine parameter interpolant from Section 4.1. Thus the generalized nine parameter interpolant is computed by building nine parameter interpolants over all two-dimensional subfaces of $\mathcal{S}_N$.

As in the case of the original nine parameter interpolant, it is clear for the higher dimensional cases that, although C^1 data are provided, the overall interpolant (defined over a tessellation into simplices of some given data set) will only be C^0. We feel that this may be sufficient for many applications, but if C^1 continuity is called for, one has to generalize the Clough–Tocher interpolant.

6.3. The generalized Clough–Tocher interpolant

We describe briefly a recent method to generalize the standard Clough–Tocher interpolant to higher dimensions [Worsey, Farin '86]. The idea is again an inductive approach: suppose one is able to construct Clough–Tocher interpolants to the restriction of the given data to the $N+1$ faces of $\mathcal{S}_N$, thereby fixing all outer Bézier ordinates of the interpolant. The remaining inner ordinates can easily be found from the C^1 conditions between adjacent mini-simplices. The key to the interpolant is thus how to split the macro-simplex $\mathcal{S}_N$ into mini-simplices.

Instead of describing the general case, we outline the three-dimensional case. The splitting process consists of three steps:

Step 1. Compute the incenter (the center of the inscribed sphere) of each tetrahedron.

Step 2. Join the incenters of any two adjacent tetrahedra and record the intersection of that straight line with the face common two both tetrahedra. For boundary faces, record the incenter.

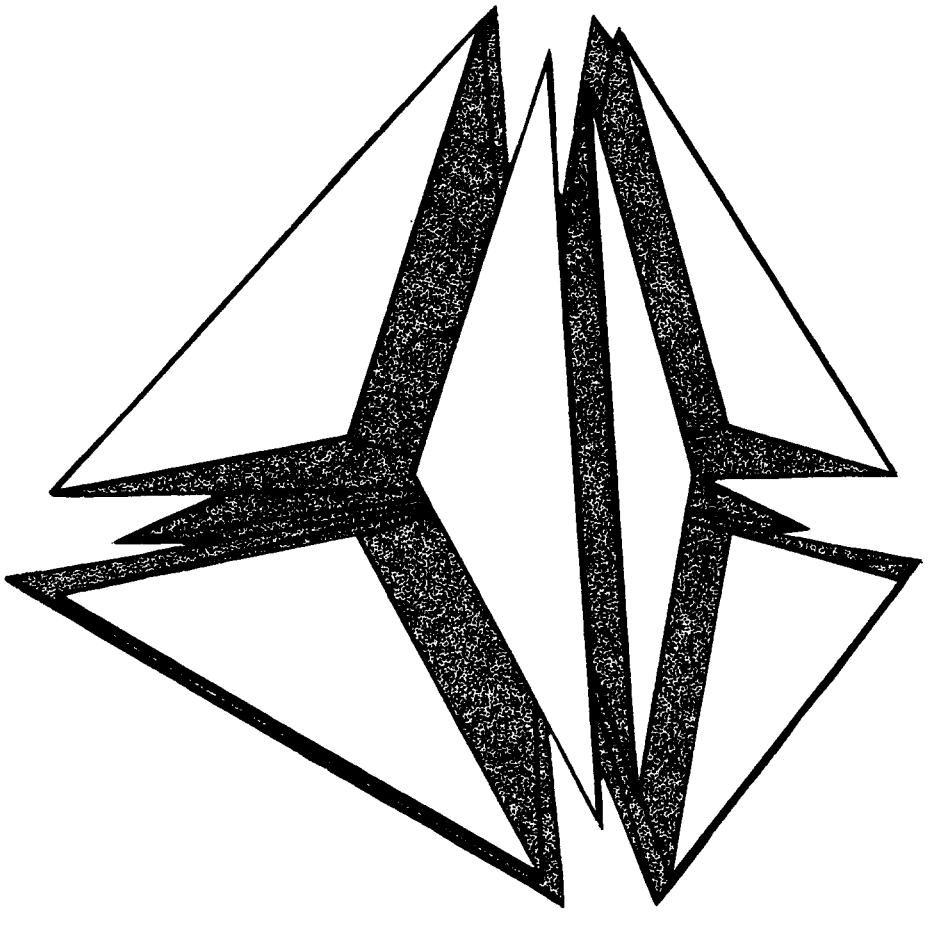


Fig. 6.2. Trivariate interpolation: A split of a tetrahedron into twelve subtetrahedra is shown as an explosion drawing. A cubic polynomial will be defined over each of them. Note that the two visible faces (white) are split according to the bivariate Clough–Tocher split.

Step 3. Split each tetrahedron into twelve mini-tetrahedra by joining its incenter to the four points recorded for it in Step 2 and to its four vertices (see Fig. 6.2).

These steps define the subdivision over which the piecewise cubic interpolant will be built. The choice of incenters in Step 1 is dictated by the differentiability requirements between adjacent tetrahedra.

The faces of each tetrahedron are split in a Clough–Tocher fashion, and the bivariate Clough–Tocher interpolant can be applied to generate the boundary Bézier ordinate of each original tetrahedron. The interior ordinates are then found from the C^1 conditions.

With this split and the accordingly defined interpolant, a globally C^1 surface is defined for a data set consisting of positional and gradient information at the vertices of a simplicial tessellation of data points.

Petersen, Piper, and Worsey [Petersen et al. '86] have developed a method to contour such surfaces, with a method analogous to the one described in Section 7.1. The first trivariate split-simplex interpolant was considered by [Alfeld '84a], who devised a C^1 piecewise quintic interpolant to C^2 data; each macro-simplex is split at the centroid into four mini-simplices.

7. Contouring

Once a surface has been built, for example by using some the previously discussed interpolants, the question arises as to how to render it. If simple line drawings are used, this should not be a problem: one can plot along constant barycentric lines or along lines parallel to the coordinate axes. One form of rendering is used frequently in scientific computing: plots of so-called *level curves* or *contour lines*. These are lines of constant z-value, an example being lines of constant pressure (isobars) obtained from fitting a surface to scattered air pressure measurements.

Although the following two sections restrict themselves to the computation of contour lines of functions (of the form $z = f(x, y) = f(\tau)$), the same principles can be used to solve the more general problem of intersection of a parametric surface (see Section 8) with a plane. A different generalization, contouring Bernstein–Bézier surfaces of several variables (see Section 6) was obtained by Petersen, Piper, and Worsey [Petersen et al. '86].

7.1. General contouring

The following algorithm was developed by F. Little and C. Petersen [Petersen '84]. It makes use of the convex hull property of Bézier patches, degree elevation/reduction, and the de Casteljau split of patches. To find the contour line for one patch of degree n, the following scheme is followed:

Step 0. Check if the minimal and maximal z-values of the Bézier net bracket the contour value. If not, no contour can be produced.

Step 1. Check if the patch can be approximated within a given tolerance by a patch of degree $n-1$. If so, replace the patch by this approximation (and n by $n-1$) and repeat. If the degree of the patch has become linear, contour. If an approximation fails, go to Step 2.

Step 2. Subdivide the patch at the midpoint of the longest edge. Also split the neighboring triangle there in order to maintain a valid triangulation. Perform Step 0 for both triangles that were generated by the subdivision.

Degree reduction, one of the main features of this algorithm, can simply be deduced from the degree elevation formula (1.12). Evidently, one could omit the degree reduction steps and consequently increase the number of subdivisions. It is interesting to note in this context that results obtained by [Sederberg '86] for the intersection of curves indicate that degree reduction

does not pay off in many cases. This question is so far unresolved for the surface case. Strategies for subdivisions other than splitting the longest edge are discussed in [Filip '86].

7.2. Contouring quadratics

The main incentive for the development of piecewise quadratic interpolants is the ease with which they can be contoured. Since a quadratic Bézier polynomial forms a paraboloid over its domain (see eq. (2.28)), any plane section, in particular also the contour line $z = const.$, is a conic section. For nondegenerate cases one obtains hyperbolae or ellipses as contour lines. If the plane $z = const.$ is moved in the z-direction, the resulting planar sections will be conics that are similar to the original contour line.

Any conic may be represented by a quadratic rational Bézier curve [Bohem, Farin, Kahmann '84] of the form

$$\boldsymbol{b}(t) = \frac{\boldsymbol{b}_0 B_0^2(t) + w\boldsymbol{b}_1 B_1^2(t) + \boldsymbol{b}_2 B_2^2(t)}{B_0^2(t) + wB_1^2(t) + B_2^2(t)}. \tag{7.1}$$

The $\boldsymbol{b}_i$ form the Bézier polygon of the curve, and w is the 'weight' that is assigned to $\boldsymbol{b}_1$ and that determines of which type the conic section is: for $w < 1$, it is an ellipse, for $w > 1$, it is a hyperbola. All $\boldsymbol{b}_i$ lie in the plane $z = const.$ Thus the contouring problem is solved by determining $\boldsymbol{b}_0$, $\boldsymbol{b}_1$, $\boldsymbol{b}_2$, and w.

Referring to Fig. 7.1, let $\boldsymbol{m}$ be the midpoint of $\boldsymbol{b}_0$ and $\boldsymbol{b}_2$, and let $\boldsymbol{s} = \boldsymbol{b}(\frac{1}{2})$. Then $\boldsymbol{m}$, $\boldsymbol{s}$, $\boldsymbol{b}_1$ lie on a straight line. Define r to be the ratio in which $\boldsymbol{s}$ divides $\boldsymbol{m}$ and $\boldsymbol{b}_1$, such that

$$\boldsymbol{s} = (1 - r)\boldsymbol{b}_1 + r\boldsymbol{m}. \tag{7.2}$$

This ratio and the weight w are related by [Boehm, Farin, Kahmann '84]

$$w = \frac{r}{1 - r}. \tag{7.3}$$

The following is an outline of an algorithm to find the contour line $z = const.$ of a quadratic Bernstein–Bézier patch:

Step 1. Intersect all three boundary curves of the quadratic with the plane. This requires the solution of three quadratic equations in one variable. Suppose we get exactly two intersection points; name them $\boldsymbol{b}_0$ and $\boldsymbol{b}_2$.

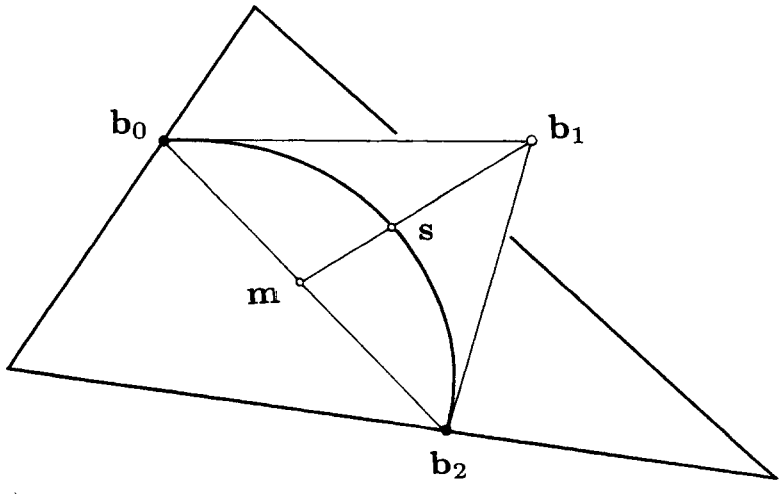


Fig. 7.1. Contouring a quadratic: The contour is defined by the Bézier polygon consisting of the $\boldsymbol{b}_i$ and a weight w attached to $\boldsymbol{b}_1$. The top view of the curve/surface configuration is shown.

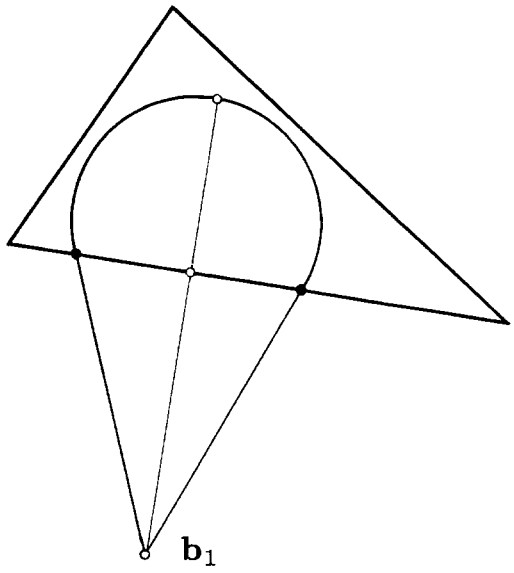


Fig. 7.2. Contouring quadratics: In the shown configuration, the intersection curve is defined over the biinfinite interval $(-\infty, +\infty)-(0, 1)$. In such cases further subdivision is advisable.

Step 2. Compute the tangent planes Π_0 and Π_2 at $\boldsymbol{b}_0$ and $\boldsymbol{b}_2$. Compute the intersection $\boldsymbol{b}_1$ of the three planes Π_0, Π_2, and $z = const$.

Step 3. In order to find the weight w, compute $\boldsymbol{m} = (\boldsymbol{b}_0 + \boldsymbol{b}_2)/2$. Intersect the straight line through $\boldsymbol{m}$ and $\boldsymbol{b}_1$ with the quadratic patch. This again requires the solution of a quadratic equation in one variable. Name the point of intersection $\boldsymbol{s}$. Determine r from (7.2). Determine w from (7.3).

Remarks. *Step 1* does not necessarily yield exactly two solutions $\boldsymbol{b}_0$ and $\boldsymbol{b}_2$, but possibly up to six different points. In such a case, the triangular patch has to be split into smaller patches (see Section 2) and the algorithm has to be applied to all resulting subpatches. Such a split is also advisable if no intersections are found with the boundary curves, but an interference between the control polygon and the plane is detected (possible closed intersection curve).

Step 2. The point $\boldsymbol{b}_1$ may be located at infinity or very far outside the patch. In that case: subdivide patch and repeat Step 1 for all subpieces.

Step 3. If the point $\boldsymbol{s}$ does not lie between $\boldsymbol{b}_1$ and $\boldsymbol{m}$, the patch should also be subdivided, see Fig. 7.2.

8. The parametric case

We have so far only considered maps $\mathcal{T} \to \mathbb{R}$, where $\mathcal{T}$ is the domain triangle. The surfaces that are generated that way are *explicit surfaces* of the form $z = f(x, y)$ or equivalently $z = f(\tau)$. We shall now consider a more general class of surfaces: *parametric surfaces* of the form

$$\boldsymbol{b}^n(\tau) = \sum \boldsymbol{b}_\lambda B_\lambda^n(\tau), \tag{8.1}$$

where boldface letters denote points in $\mathbb{R}^3$. A parametric Bernstein–Bézier patch is thus the graph of a map $\mathcal{T} \to \mathbb{R}^3$. Explicit triangular patches can be used successfully for scattered data interpolation, as outlined in Sections 4 and 5. Many problems in geometric modelling, however, require the additional flexibility offered by parametric patches.

8.1. Where to use parametric triangular patches

The theory of parametric tensor product polynomial patches is well established [Bohem, Farin, Kahmann '84], [Faux, Pratt '78], and such patches are successfully used by many commercial CAD systems to model complicated surfaces.

Many surfaces lend themselves naturally to a tensor product representation, but surfaces exist where this is not the case. The theory of tensor product patches requires that all data have a rectangular geometry and that parametrizations of opposite boundary curves be 'similar'. this is the case for some surfaces (e.g., the trunk lid of a car) but not for others (e.g., interior car body panels). For those more complicated surface, no rectangular geometry is apparent, and one has to invest considerable time and effort to impose a rectangular structure on such surfaces.

It is for these complex surfaces that the use of triangular patches is advantageous. The reason is simple: every surface can be covered with a triangular network, while this may be impossible with (nondegenerate) rectangular networks: a simple example is given by the sphere.

Systems that employ rectangular patches only must use degenerate patches to model regions where triangular patches are needed. Degenerate rectangular patches look like a triangular patch; they have an edge collapsed to a single point. They suffer seriously from the fact that the surface normal is not defined at that point. Degenerate patches frequently cause failures of standard algorithms, such as plane/surface intersections or ray tracing.

In other cases, one has a rectangular geometry, but the parametrizations of opposite boundary curves are too different to allow a reasonable tensor product surface to be fitted between the boundary curves (e.g. opposite boundary curves could consist of a different number of polynomial segments). A solution to one such problem using triangular patches is discussed in [Farin '86].

In summary: One should use rectangular patches whenever possible since they can be computed quickly and without the need for a complicated data structure. In those cases where rectangular patches are too tedious to use, triangular patches should be employed. They may be more CPU-intensive, but the expense incurred should weigh little compared to cost of (human) efforts to design complex surfaces with insufficient tools.

8.2. Parametric continuity

Let $\mathcal{T}$ and $\hat{\mathcal{T}}$ be two domain triangles that share a common edge. Let the pair $(\mathcal{T}, \hat{\mathcal{T}})$ be mapped to two triangular parametric patches $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$. If this map is continuous, $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$ share a common boundary curve: we say that the surface consisting of $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$ is C^0. If the map is C^1, we say that the surface consisting of $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$ is C^1, etc. A geometric interpretation of the two patches being C^r is as follows: Let l be a straight line crossing the common edge of $\mathcal{T}$ and $\hat{\mathcal{T}}$. Then the map $(\mathcal{T}, \hat{\mathcal{T}}) \to (\boldsymbol{b}, \hat{\boldsymbol{b}})$ takes l to a C^r curve (on the surface $(\boldsymbol{b}, \hat{\boldsymbol{b}})$) in $\mathbb{R}^3$.

The continuity conditions for parametric patches follow directly from those for nonparametric patches:

$$\hat{\boldsymbol{b}}_{\lambda^s} = \boldsymbol{b}^s_{\lambda^0}(\sigma); \quad s = 0, \ldots, r. \tag{8.2}$$

where the meaning of all symbols is the same as in (2.21), except that the $\boldsymbol{b}$'s are now points instead of scalars. Fig. 8.1 illustrates the geometry of the simple case of C^1 continuity. Fig. 8.2 demonstrates that simple coplanarity of adjacent subtriangles is not sufficient.

8.3. Visual continuity

Surfaces that are C^1 according to (8.2) turn out to be too 'rigid' for the successful modeling of complex surfaces. An example for the failure of such surfaces to cope even with simple

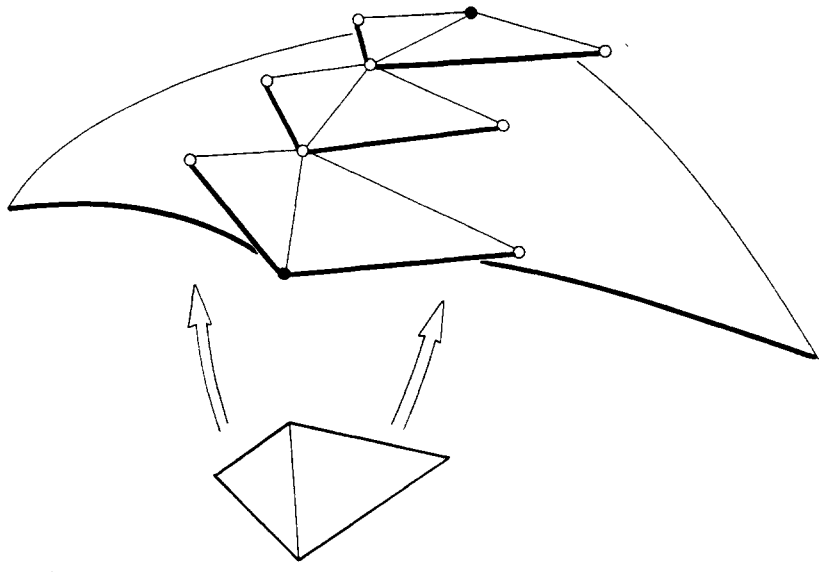


Fig. 8.1. Parametric C^1-continuity: The indicated pairs of subtriangles must be coplanar and also be affine images of the pair of domain triangles.

situations is given in Fig. 8.2: even if all boundary curves were elevated from quadratic to cubic, no interior Bézier points $\boldsymbol{b}_{1,1,1}$ could be found such that the two patches had a continuously varying tangent plane along their common boundary.

While that example exhibits a local problem that C^1 surfaces may encounter, another type of problem seems to be at least as severe: A piecewise triangular C^1 surface is the map of a triangulated domain. If such a surface is to be used for the representation of closed (e.g. sphere-like) surfaces, singularities must occur. A way to avoid these is the relaxation of the C^1 conditions. The requirement of tangent plane (or visually C^1, short: V^1) continuity between adjacent patches turns out to be general enough to overcome both the local and global problems mentioned above.

A parametric Bernstein–Bézier patch is *completely* defined by its Bézier net: no reference to a domain triangle is at all necessary. Therefore it should be possible to describe V^1 continuity between adjacent (adjacent simply means the existence of a common boundary curve) patches $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$ entirely in terms of their Bézier nets. An early approach towards this goal is in [Farin '82], but we follow here a generalization due to [Piper '86].

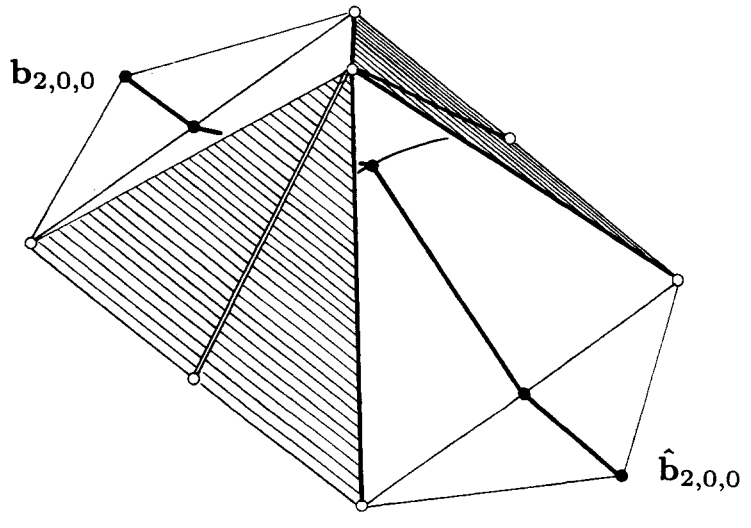


Fig. 8.2. Parametric C^1-continuity: Two adjacent quadratic patches are shown together with the visible part of their common boundary curve, the shaded adjacent subtriangles are coplanar. However, the radial lines from $b_{2,0,0}$ and $\hat{b}_{2,0,0}$ to the midpoint of the common boundary are not smooth, as their Bézier polygons indicate (see also Fig. 2.4). This illustrates that coplanarity of adjacent subtriangles is not sufficient as a C^1 condition.

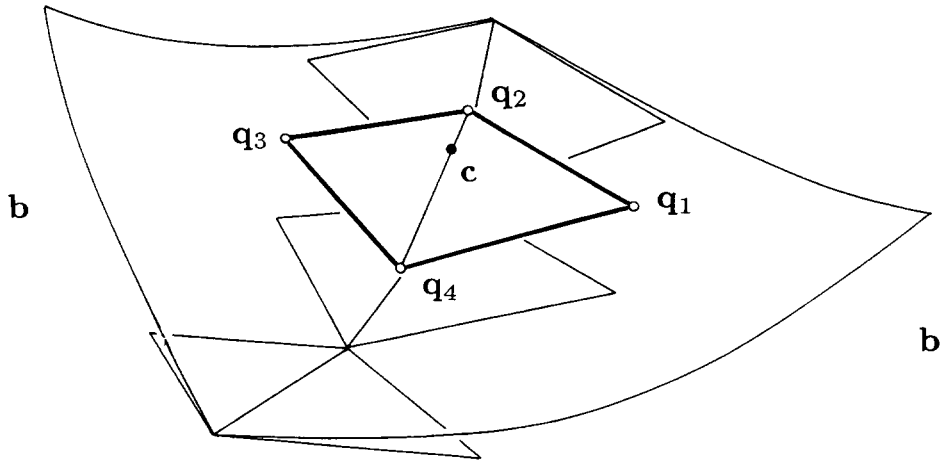


Fig. 8.3. V^1-continuity: The heavily delineated pair of triangles arises from the de Casteljau constructions (one for each patch) of the indicated point on the common boundary curve. This pair of triangles must be planar.

Let $\boldsymbol{c}$ be a point on the common boundary curve of $\boldsymbol{b}$ and $\hat{\boldsymbol{b}}$. The point $\boldsymbol{c}$ can be constructed by application of the de Casteljau algorithm to either patch. From the remarks after Theorem 2.4 it is clear that the intermediate points $\boldsymbol{b}_\lambda^{n-1}$ and $\hat{\boldsymbol{b}}_\lambda^{n-1}$ determine the tangent planes at $\boldsymbol{c}$. In order for both tangent planes to coincide, the points $\boldsymbol{b}_\lambda^{n-1}$ and $\hat{\boldsymbol{b}}_\lambda^{n-1}$ must be coplanar. There are four such points (taking into account that we have coincident points from the common boundary), and we denote them by $\boldsymbol{q}_i$, $i=1, 2, 3, 4$, see Fig. 8.3. The V^1-condition now becomes:

$$\det\begin{pmatrix} 1 & 1 & 1 & 1 \\ \boldsymbol{q}_1(t) & \boldsymbol{q}_2(t) & \boldsymbol{q}_3(t) & \boldsymbol{q}_4(t) \end{pmatrix} = 0. \tag{8.3}$$

We note again that (8.3) is purely a relationship between points in $\mathbb{R}^3$, without reference to any domains.

For univariate splines, an attempt to define V^1 curves is due to [Barsky '81]: he uses the knot spacing as a 'shape parameter', called β_1. This amounts to a harmonic spacing of the knots such that $\Delta t_i = \beta_1 \Delta t_{i+1}$. A generalization to triangular Bernstein–Bézier surfaces is due to [De Rose '85]: an equilateral triangulation is distorted in one direction such that a harmonic spacing of the parameter lines is achieved (see Fig. 8.4). The spacing factor, again called β_1, is called a shape parameter. This concept of V^1 continuity is a special case of C^1 continuity as

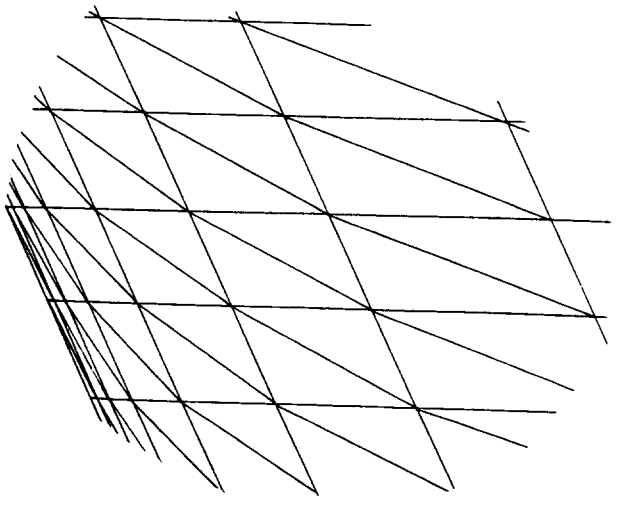


Fig. 8.4. V^1-continuity: The distorted equilateral triangulation shown serves as a domain for triangular surfaces that are a generalization of β-splines.

defined above. It should be mentioned, however, that the above V^1 surfaces are only a special case of a more general definition of V^1 continuity in terms of manifolds [De Rose '85].

Different approaches to the concept of V^1 (or G^1) continuity are in [Herron '85a, '85b], [Jensen '85], [Rescorla '85]. However, these authors do not base their arguments on the geometric properties of Bézier nets but rather on domain-dependent concepts.

The definition of higher order visual continuity has so far been a considerable problem. The only approach that (in some special cases) led to conditions for adjacent Bézier nets is the one by [Kahmann '83]: he defines two adjacent patches to be V^2 if their Dupin indicatrices are uniquely defined along the common boundary.

8.4. Parametric nine parameter interpolants

The nine parameter interpolant from Section 4.1 is not readily applicable in a parametric setting: instead of nine data items one is typically only given six, namely the vertices of the patch and the normals there. (Very often these normals have to be made up from point data as a preprocessing step.) The problem of finding the remaining six boundary Bézier points from these data is far from being trivial, and we refer the reader to the discussion in [Nielson '86].

Having determined the boundary curves of the patch, we are left with the task of determining $\boldsymbol{b}_{1,1,1}$. Equation (4.2) certainly solves this problem, but it turns out that in general the resulting patch tends to 'bulge out'. An alternative to (4.2) is given by

$$\boldsymbol{b}_{1,1,1} = \frac{\frac{1}{m_1}\boldsymbol{b}_1 + \frac{1}{m_2}\boldsymbol{b}_2 + \frac{1}{m_3}\boldsymbol{b}_3}{\frac{1}{m_1} + \frac{1}{m_2} + \frac{1}{m_3}},$$

where

$$\boldsymbol{b}_1 = \tfrac{1}{2}(\boldsymbol{b}_{1,2,0} + \boldsymbol{b}_{1,0,2}),$$
$$\boldsymbol{b}_2 = \tfrac{1}{2}(\boldsymbol{b}_{0,1,2} + \boldsymbol{b}_{2,1,0}),$$
$$\boldsymbol{b}_3 = \tfrac{1}{2}(\boldsymbol{b}_{0,2,1} + \boldsymbol{b}_{2,0,1}).$$

The quantity m_i is a measure of the deviation of edge i from a straight line ($m_i = 0$ labels edge i as linear). This formula uses inverse weighting and has some resemblance to Shepard's formula. Although somewhat ad hoc, its performance compares favorably to that of (4.2). Its main feature is that ruled surfaces are reproduced.

Of course this nine parameter interpolant will not produce globally smooth surfaces. Its importance stems from the fact that it can be used as a building block for more complicated smooth surfaces [Piper '86].

Conclusion

While we were able to cover some material related to the theory of Bernstein–Bézier patches defined over triangles, much had to be left undiscussed. We mention some of those topics and point out to relevant literature:

Box splines. A growing number of articles is concerned with a generalization of univariate B-splines to a multivariate setting. A special case of these multivariate splines are so-called *box splines*. They correspond (in the bivariate case) to surfaces defined over a regular tessellation of the plane; in those cases where the tessellation is composed of triangles, it is possible to represent the surface as a collection of Bernstein–Bézier patches, see [Dahmen, Micchelli '84], [Boehm '85], [Prautzsch '84].

Global interpolants. In Sections 4 and 5, we have discussed interpolants that were *local.* However, it is possible to consider interpolants that are defined by global requirements, e.g., to minimize a certain functional (such as strain energy) over the whole (triangulated) domain. This leads to the solution of (large) linear systems, see [Schmidt '82], [Bohem, Farin, Kahmann '84].

Connection to rectangular patches. If triangular patches are to be used for the representation of complex surfaces, it will become necessary to connect them smoothly to parts of the surface where rectangular (Bézier) patches are used. This can only be done in the sense of visual continuity, see [Farin '83], [Hosaka, Kimura '78]. Sometimes a conversion from triangular to rectangular patches will be necessary; an elegant algorithm for this is in [Brueckner '80] and an elementary development is in [Farin '79]. The 'inverse' problem, converting from rectangular to triangular form has been considered by [Goldman, Filip '86].

Dimension counts. Over a given triangulation, C^r piecewise polynomials form a linear space. Its dimension is unknown (except for some special cases), and so is the existence of (local support?) basis functions. Recent investigations have made successful use of the Bernstein–Bézier form: [Alfeld '86a, '86b], [Alfeld, Schumaker '86], [Alfeld, Piper, Schumaker '86].

Acknowledgement

This research was supported in part by Department of Energy contract DE-AC02-85ER12046 to the University of Utah. The author wishes to thank Peter Alfeld for pointing out an error in an earlier draft of this manuscript and also Wolfgang Boehm for many helpful comments.

Appendix: A data structure

For the computer representation of a surface that is comprised of many triangular patches (each being defined over some triangle of a triangulation of the given data points) it is extremely important to utilize an efficient data structure. We shall describe a structure that is due to F. Little and C. Petersen and that has been used successfully by the University of Utah Math CAGD group for several years. The programs were written in FORTRAN, and this is reflected by the fact that all data are contained in several independent arrays.

For the specification of these arrays, two integers are needed: the number MP of data sites and the number MTR of triangles in the triangulation. This last number should be the output of a routine that triangulates the given data sites. This routine determines the number MI of interior points and the number MB of boundary points of the triangulation. They are related by MTR = 2*MI + MB − 2.

The data structure consists of the following arrays:

P(3,MP) holds the x, y-location of the data sites as well as the given function values there.

ITR(9,MTR) contains the pointers that define the triangulation. The first three entries ITR(1,K), ITR(2,K), ITR(3,K) identify the vertices of triangle number K in the triangulation. They are P(*,ITR(1,K)), P(*,ITR(2,K)), P(*,ITR(3,K)). The next three entries, ITR(4,K), ITR(5,K), ITR(6,K) identify the triangles that are neighboring triangle number K, i.e., ITR(4,K) is the index (in ITR) of the triangle that shares edge one with triangle number K; ITR(5,K) corresponds to edge number two etc. If no such triangle exists, zeroes are entered. Finally, ITR(7,K), ITR(8,K), ITR(9,K) are the indices of the points in P that are opposite edge number 1,2, and 3, respectively (again zero entries if no such points exist).

PZ(10, MTR) contains the Bézier ordinates for each patch. They are ordered linearly, and a conversion from the triple subscripts used so far to this ordering is necessary, e.g., by an

implicit function definition. The '10' was chosen since no patches of degree greater than three were used; in general, one has to replace 10 by $(n+1)(n+2)/2$, if n is the anticipated maximum degree.

NDEG(MTR) contains the degrees of every patch.

This data structure is highly redundant (e.g., the boundary Bézier ordinates are stored twice in PZ), but this has proved to help speed up many subtasks like searches. The fact that each patch may be of different degree is useful for algorithms like the contouring algorithm due to [Petersen '84], see also Section 7.

References

Alfeld, P. (1984a), A trivariate Clough–Tocher scheme for tetrahedral data, Computer Aided Geometric Design 1, 169–181.

Alfeld, P. (1984b):, A bivariate C^2 Clough–Tocher scheme, Computer Aided Geometric Design 1, 257–267.

Alfeld, P. (1985), Private communication.

Alfeld, P. (1986a), On the dimension of piecewise polynomial functions, *Proc. Biennial Dundee Conference on Numerical Analysis*, June 25–28, 1985, Pitman, London.

Alfeld, P. (1986b), A case study of multivariate piecewise polynomials, to appear in: G. Farin, ed., *Geometric Modeling*, SIAM, Philadelphia, PA.

Alfeld, P., Piper, B. and Schumaker, L. (1986), Local bases for bivariate C^r piecewise polynomials of degree $d \geq 4r+1$, submitted for publication.

Alfeld, P. and Schumaker, L. (1986), The dimension of bivariate spline spaces of smoothness r for degree $4r+1$, submitted for publication.

Barnhill, R.E., Birkhoff, G. and Gordon, W. (1974), Smooth interpolation in triangles, J. Approx. Theory 8, 114–128.

Barnhill, R.E. (1977), Representation and approximation of surfaces, in: J. Rice, ed., *Mathematical Software III*, Academic Press, New York, 69–120.

Barnhill, R.E. (1985), Surfaces in computer aided geometric design: a survey with new results, Computer Aided Geometric Design 2, 1–17.

Barnhill, R.E. and Farin, G., (1981), C^1 quintic interpolation over triangles: two explicit representations, Int. J. Num. Methods in Engineering 17, 1763–1778.

Barsky, B. (1981), The β-spline: a local representation based on shape parameters and fundamental geometric measures, dissertation, Univ. of Utah.

Boehm, W. (1985), Multivariate spline algorithms, in: J. Gregory, ed., *Mathematics of Surfaces*, Oxford University Press, Oxford.

Boehm, W. and Farin, G. (1983), Letter to the Editor, CAD 15, 260–261.

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

de Boor, C. (1986a), B-net basics, to appear in: G. Farin, ed., *Geometric Modeling*, SIAM, Philadelphia, PA.

de Boor, C. (1986b), B-net basics, enlarged version of [de Boor '86a], in preparation.

Brueckner, I. (1980), Construction of Bézier points of quadrilaterals from those of triangles, CAD 12, 21–24.

de Casteljau, P. (1959), Outillage méthodes calcul, André Citroën Automobiles SA, Paris.

de Casteljau, P. (1963), Courbes et surfaces à poles, André Citroën Automobiles SA, Paris.

Chang, G. and Davis, P.J. (1984), A new proof for the convexity of the Bernstein–Bézier surfaces over triangles, J. Approx. Theory 40, 11–28.

Chang, G. and Feng, Y. (1985), An improved condition for the convexity of Bernstein–Bézier polynomials over triangles, Computer Aided Geometric Design 1, 279–283.

Chang, G. and Hoschek, J. (1985), Convexity and variation diminishing property of Bernstein polynomials over triangles, *Multivariate Approximation Theory III*, Birkhäuser, Basel.

Chang, G. and Feng, Y. (1985), A pair of compatible variations for Bernstein triangular polynomials, submitted for publication.

Chui, C. and Lai, M. (1985), On bivariate vertex splines, CAT report no. 74, Texas A&M Univ., College Station, TX.

Dahmen, W. and Micchelli, C. (1985), Convexity of multivariate Bernstein polynomials and box spline surfaces, Preprint.

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

Davis, P. (1976), Lecture notes on CAGD, held at the Univ. of Utah, Dept. of Mathematics.

Farin, G. (1979), Subsplines über Dreiecken, Dissertation, Braunschweig, FRG.

Farin, G. (1980), Bézier polynomials over triangles and the construction of piecewise C^r polynomials, TR/91, Dept. of Mathematics, Brunel University, Uxbridge, UK.

Farin, G., (1983), Smooth interpolation to scattered 3D data, in: R.E. Barnhill and W. Boehm, eds., *Surfaces in CAGD*, North-Holland, Amsterdam.

Farin, G. (1982), A construction for the visual C^1 continuity of polynomial surface patches, Computer Graphics and Image Processing 20, 272–282.

Farin, G. (1985a), A modified Clough–Tocher interpolant, Computer Aided Geometric Design 2, 19–27.

Farin, G. (1985b) Piecewise triangular C^1 surface strips, CAD 18, 45–47.

Farin, G. and Barry, P. (1986), A link between Lagrange and Bézier curve and surface schemes, submitted to CAD.

Faux, I. and Pratt, M. (1978), *Computational Geometry for Design and Manufacture*, Ellis Horwood, Chichester.

Filip, D. (1985), Adaptive subdivision algorithms for a set of Bézier triangles, submitted for publication.

Frederickson, (1970, 1971), Triangular spline interpolation/Generalized triangular splines, Math reports no. 6/70 and 7/71, Lakehead University, Canada.

Goldman, R. (1983), Subdivision algorithms for Bézier triangles, CAD 15, 159–166.

Goldman, R. and Filip, D. (1986), Conversion from Bézier rectangles to Bézier triangles, submitted for publication.

Goodman, T.N.T. (1985), Variation diminishing properties of Bernstein polynomials over triangles, to appear in J. Approx. Theory.

Gregory, J.A. (1978), Piecewise interpolation theory and finite element analysis, Dissertation, Brunel University, Uxbridge, UK.

Grieger, I. (1985), Geometry cells and surface definition by finite elements, Computer Aided Geometric Design 2, 213–222.

Herron, G. (1986), Techniques for Visual Continuity, to appear in G. Farin, ed., *Geometric Modeling*, SIAM, Philadelphia, PA.

Herron, G. (1985b), Smooth closed surfaces with discrete triangular interpolants, CAGD 2, 297–306.

Hosaka, M. and Kimura, F. (1978), Synthesis methods of curves and surfaces in interactive CAD, *Proceedings: Interactive Techniques in CAD*, Bologna, 151–156.

Jensen, T. (1985), Assembling triangular and rectangular patches and multivariate splines, to appear in: G. Farin, ed., *Geometric Modeling*, SIAM, Philadelphia, PA.

Kahmann, J. (1983), Continuity of curvature between adjacent Bézier patches, in: R.E. Barnhill and W. Boehm, *Surfaces in CAGD*, North-Holland, Amsterdam.

Kuang, Z. and Chang, G. (1985), Remarks on the convexity for parametric Bézier triangles, submitted for publication.

Lane, J. and Riesenfeld, R. (1983), A geometric proof of the variation diminishing property of B-spline approximation, J. Approx. Theory 37, 1–4.

Nielson, G. (1986), A visually continuous, transfinite triangular interpolant, to appear in: G. Farin, ed., *Geometric Modeling*, SIAM, Philadelphia, PA.

Percell, P. (1976), On cubic and quartic Clough–Tocher elements, SIAM J. Numerical Analysis 13, 100–103.

Petersen, C. (1984), Adaptive contouring of three-dimensional surfaces, Computer Aided Geometric Design 1, 61–74.

Petersen, C., Piper, B. and Worsey, A. (1986), Adaptive contouring of a trivariate interpolant, to appear in: G. Farin, ed., *Geometric Modeling*, SIAM, Philadelphia, PA.

Piper, B. (1986), Tangent plane continuity between triangular Bézier patches, to appear in: G. Farin, ed., *Geometric Modeling*, SIAM, Philadelphia, PA.

Powell, M.J. (1974), Piecewise quadratic surface fitting for contour plotting, in: D.J. Evans, ed., *Software for Numerical mathematics*, Academic Press, London.

Powell, M.J. and Sabin, M.A. (1977), Piecewise quadratic approximation on triangles, ACM Trans. Mathematical Software 3, 316–325.

Prautzsch, H. (1984), Unterteilungsalgorithmen für multivariate splines, Dissertation, Technical University Braunschweig, FRG.

Prautzsch, H. (1985), Simple concepts don't work as the definition of variation diminishing, Breakout session at the 1985 SIAM conference on Geometric Modeling and Robotics, Albany, NY.

Prenter, P. (1975), *Splines and Variational Methods*, Wiley, New York.

Rescorla, K. (1985), Multivariate interpolation, Dissertation, Univ. of Utah.

De Rose, T. (1985), Geometric continuity: A parametrization independent measure of continuity for CAGD, Report No. UCB/CSD 86/255, UC Berkeley.

Sabin, M.A. (1977), The use of piecewise forms for the numerical representation of shape, Dissertation, Budapest, Hungary.

Sablonnière, P. (1982), Bases de Bernstein et approximants splines, Dissertation, Lille, France.

Sblonnière P. (1985), Composite finite elements of class C^k, J. Comput. Appl. Math. 12&13, 541–550.

Schmidt, R. (1982), Eine Methode zur Konstruktion von C^1-Flächen zur Interpolation unregelmässig verteilter Daten, in: Schempp and Zeller, eds., *Multivariate Approximation II*, Birkhäuser, Basel, 343–361.

Schumaker, L. and Volk, W. (1986), Efficient evaluation of multivariate polynomials, submitted for publication.

Sederberg, T. and Parry, S. (1986), A comparison of three curve intersection algorithms, CAD 18, 58–64.
Sibson, R. (1980), A seamed quadratic element for contouring, Preprint, University of Bath, UK.
Stärk, E. (1976), Mehrfach differenzierbare Bézierkurven und Bézierflächen, Dissertation, Technical University Braunscheig, FRG.
Strang, G. and Fix, G. (1973), *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ.
Whelan, T. (1986), A representation of a C^2 interpolant over triangles, Computer Aideol Geometric Design 3, 53–66.
Worsey, A. and Farin, G. (1985):, An n-dimensional Clough–Tocher interpolant, being revised.
Zhou, J. (1985), The positivity and convexity for Bernstein polynomials over triangles, Dissertation (in Chinese), to appear in Math. Numer. Sinica.
Zenisek, A. (1970), Interpolation polynomials on the triangle, Numer. Math. 15, 283–296.