AN INTRODUCTION TO SPLINE THEORY

Michael S. Floater

Department of Mathematics

University of Oslo

Contents

Contents **i**

Preface **1**

1 Bernstein-Bézier polynomials **3**

1.1 Bernstein basis polynomials 3

1.2 Recursion 5

1.3 Differentiation 6

1.4 Integration 7

1.5 Bernstein-Bézier polynomials 7

1.6 Bézier curves 8

1.7 The de Casteljau algorithm 10

1.8 Derivatives 12

1.9 Higher order derivatives 13

1.10 Integration 14

1.11 Exercises 14

2 Splines in Bernstein-Bézier form **17**

2.1 BB polynomials on an arbitrary interval 17

2.1.1 The de Casteljau algorithm 18

2.1.2 Derivatives 18

2.2 Joining polynomial pieces together 19

2.3 Linear spline interpolation 21

2.4 Cubic Hermite spline interpolation 23
2.5 C^2 cubic spline interpolation 24
2.6 Minimization of second derivatives 26
2.7 Natural end conditions 27
2.8 Exercises 28

3 B-splines **31**
3.1 Divided differences 31
3.1.1 Distinct points 32
3.1.2 Arbitrary points 32
3.1.3 Leibniz rule 34
3.2 B-splines 34
3.3 Piecewise polynomials and smoothness 35
3.3.1 Distinct knots 35
3.3.2 Multiple knots 36
3.4 Recursion 37
3.5 Examples 38
3.5.1 Distinct knots 38
3.5.2 Uniform B-splines 39
3.5.3 Multiple knots 40
3.5.4 Bernstein polynomials 41
3.6 Value of a B-spline at a knot 41
3.7 Closed knot vector 42
3.8 Linear combinations of B-splines 42
3.8.1 Spline functions 43
3.8.2 Spline curves 44
3.9 Evaluation 44
3.9.1 Algorithm 1: B-spline recursion 45
3.9.2 Algorithm 2: de Boor algorithm 45
3.10 Exercises 46

4 Further properties of B-splines **49**

4.1 Marsden's identity 49

4.2 Linear independence of B-splines 50

4.2.1 Monomials as splines 51

4.3 B-splines as a basis for splines 52

4.4 Derivatives 55

4.5 Exercises 56

5 Knot insertion **57**

5.1 B-spline refinement 58

5.2 Spline refinement 59

5.3 Variation diminishing property 60

5.4 Exercises 63

6 B-spline interpolation and approximation **65**

6.1 Spline interpolation in B-spline form 65

6.1.1 Linear spline interpolation 65

6.1.2 Cubic Hermite spline interpolation 66

6.1.3 C^2 cubic spline interpolation 67

6.2 General spline interpolation 69

6.3 Least squares approximation 72

6.4 Variation-Dimishing Spline Approximation 74

6.5 Exercises 75

7 Surfaces **77**

7.1 Tensor-product BB polynomials 77

7.1.1 The de Casteljau algorithm 79

7.1.2 Joining polynomial pieces together 80

7.2 Tensor-product splines in B-spline form 82

7.2.1 Evaluation 82

7.2.2 Interpolation to gridded data 83

7.2.3 Least squares approximation 84
7.3 Triangular BB polynomials 86
7.3.1 The de Casteljau algorithm 90
7.3.2 Derivatives 92
7.3.3 Joining polynomial pieces together 94

8 Blossoming **97**
8.1 The three axioms 97
8.2 de Casteljau's algorithm 98
8.3 Joining BB polynomials together 100
8.4 Subdivision 101
8.5 Generalized de Casteljau algorithm 102
8.6 Blossom of a polynomial in B-spline form 103
8.7 Conversion of polynomial and spline representations 105
8.7.1 Converting a BB polynomial to B-spline form 105
8.7.2 Converting one B-spline form to another 106
8.8 Blossoms of bivariate polynomials 107

Preface

This is a collection of lecture notes about spline theory which are designed to accompany the course 'Spline methods' at the University of Oslo. The notes have developed over several years in conjuction with the two courses 'Topics in Geometric Modelling' and 'Spline Methods', taught previously by the Informatics department and more recently by the Maths department of the University of Oslo. We consider here a spline to be a piecewise polynomial, and we discuss both spline functions and spline curves and surfaces.

There are two common approaches to constructing a spline function or curve. One is to construct it one polynomial piece at a time and to enforce conditions on the pieces that guarantee that they fit together continuously or smoothly. In this approach, it is convenient to represent each piece in Bernstein-Bézier form. The other approach is to use the B-spline representation of the spline, in which the smoothness between the polynomial pieces is 'automatic'. The aim is to cover both approaches in these notes.

Much of the material presented here can be found in the literature. In particular, books that cover the theory of Bernstein-Bézier polynomials and Bézier curves are those of Farin [3], H. Prautzsch, W. Boehm and M. Paluszny [6], and Lai and Schumaker [5]. The reader is encouraged to look at these sources for further aspects of the theory. For the theory of B-splines, we refer the reader to the book by de Boor [2] and the lecture notes by Lyche and K. Mørken used in earlier versions of the Spline Methods course.

Chapter 1

Bernstein-Bézier polynomials

We start in this first chapter by studying Bernstein-Bézier polynomials and Bézier curves.

1.1 Bernstein basis polynomials

Recall that a real polynomial of a real variable $x \in \mathbb{R}$ of degree k is a function of the form

$$p(x) = a_0 + a_1 x + \cdots + a_k x^k = \sum_{i=0}^{k} a_i x^i,$$

where $a_i \in \mathbb{R}$, $i = 0, 1, \ldots, k$, and $a_k \neq 0$. We will denote by π_d the linear space of polynomials of degree at most d,

$$\pi_d = \left\{ \sum_{i=0}^{d} a_i x^i : a_i \in \mathbb{R}, i = 0, 1, \ldots, d \right\}.$$

The functions $1, x, \ldots, x^d$ form a basis for π_d, known as the *monomial basis*, and the dimension of the space π_d is therefore $d + 1$. The Bernstein basis polynomials provide an alternative basis for π_d and are defined as

$$B_i^d(x) = \binom{d}{i} x^i (1-x)^{d-i}, \qquad i = 0, 1, \ldots, d, \tag{1.1}$$

where

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

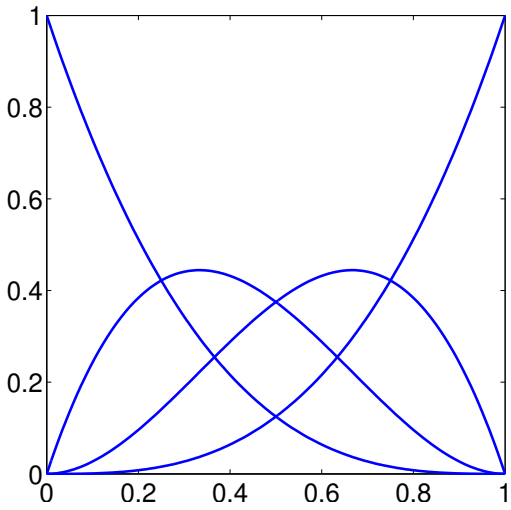

Figure 1.1: The cubic Bernstein basis polynomials

The first few examples are

$$B_0^0(x) = 1,$$
$$B_0^1(x) = 1 - x, \quad B_1^1(x) = x,$$
$$B_0^2(x) = (1-x)^2, \quad B_1^2(x) = 2x(1-x), \quad B_2^2(x) = x^2,$$
$$B_0^3(x) = (1-x)^3, \quad B_1^3(x) = 3x(1-x)^2, \quad B_2^3(x) = 3x^2(1-x), \quad B_3^3(x) = x^3.$$

The cubic ones are shown in Figure 1.1.

Let us consider some of their basic properties. For any $x \in \mathbb{R}$, they sum to 1 since, by the binomial theorem,

$$1 = 1^d = (x + (1-x))^d = \sum_{i=0}^{d} \binom{d}{i} x^i (1-x)^{d-i} = \sum_{i=0}^{d} B_i^d(x).$$

For x in the interval $[0,1]$ they are also non-negative. Thus, in the interval $[0,1]$, they form a so-called *partition of unity*. In fact, they are positive in the open interval $(0,1)$ and at the endpoints

$$B_i^d(0) = \begin{cases} 1, & i = 0; \\ 0, & i = 1, \ldots, d, \end{cases} \quad \text{and} \quad B_i^d(1) = \begin{cases} 0, & i = 0, \ldots, d-1; \\ 1, & i = d. \end{cases} \tag{1.2}$$

Let us now show that the Bernstein basis polynomials of degree d do indeed form a basis for the space of polynomials of degree $\leq d$.

Theorem 1.1 *The set* $\{B_i^d : i = 0, 1, \ldots, d\}$ *is a basis for* π_d.

Since the number of elements in the set is $d+1$, one way to prove the theorem is to show that any monomial x^j, $0 \le j \le d$, is in the span of the B_i^d, i.e., that x^j can be expressed as a linear combination of the B_i^d. This is demonstrated by the following lemma.

Lemma 1.2 *For $j = 0, 1, \ldots, d$,*

$$x^j = \frac{(d-j)!}{d!} \sum_{i=j}^{d} \frac{i!}{(i-j)!} B_i^d(x).$$

Proof. We use the binomial theorem again:

$$\begin{aligned} x^j = x^j(x + (1-x))^{d-j} &= x^j \sum_{i=0}^{d-j} \binom{d-j}{i} x^i (1-x)^{d-j-i} \\ &= \sum_{i=j}^{d} \binom{d-j}{i-j} x^i (1-x)^{d-i}, \end{aligned}$$

which, by the definition of $B_i^d(x)$ gives the result. □

1.2 Recursion

The Bernstein basis polynomials satisfy a recursion formula which provides an efficient way to compute them. Here and elsewhere we will make the convenient convention that $B_{-1}^d(x) = 0$ and $B_{d+1}^d(x) = 0$ for all $x \in \mathbb{R}$.

Lemma 1.3 *For $i = 0, 1, \ldots, d$,*

$$B_i^d(x) = xB_{i-1}^{d-1}(x) + (1-x)B_i^{d-1}(x). \tag{1.3}$$

Proof. Suppose $1 \le i \le d-1$. We will make use of the recursion formula for binomial coefficients,

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

Using this we can express $B_i^d(x)$ as

$$\begin{aligned} B_i^d(x) &= \left(\binom{d-1}{i-1} + \binom{d-1}{i} \right) x^i (1-x)^{d-i} \\ &= x\binom{d-1}{i-1} x^{i-1}(1-x)^{d-i} + (1-x)\binom{d-1}{i} x^i (1-x)^{d-i-1}, \end{aligned}$$

which is the right hand side of (1.3). In the cases $i = 0$ and $i = d$ we have

$$B_0^d(x) = (1-x)^d = (1-x)B_0^{d-1}(x),$$
$$B_d^d(x) = x^d = xB_{d-1}^{d-1}(x),$$

which agree with (1.3) due to the convention that $B_{-1}^{d-1} = B_d^{d-1} = 0$. □

Thus the Bernstein basis polynomials can be evaluated at a given $x \in [0,1]$ using a triangular scheme. For a fixed x we first set $B_0^0 = 1$. Then, for each $r = 1, 2, \ldots, d$, we let

$$B_i^r = xB_{i-1}^{r-1} + (1-x)B_i^{r-1}, \quad i = 0, 1, \ldots, r. \tag{1.4}$$

The scheme is illustrated below. In each column, each value is computed from two values from the previous column (except the first and last elements of the column, which are computed from just one value in the previous column).

$$\begin{array}{ccccccccc} 1 & = & B_0^0 & B_0^1 & B_0^2 & \cdots & B_0^d \\ & & & B_1^1 & B_1^2 & \cdots & B_1^d \\ & & & & B_2^2 & \cdots & B_2^d \\ & & & & & \ddots & \vdots \\ & & & & & & B_d^d \end{array}$$

1.3 Differentiation

We can also compute derivatives of Bernstein basis polynomials by a recursive formula.

Lemma 1.4 *For $d \geq 1$ and for $i = 0, 1, \ldots, d$,*

$$(B_i^d)'(x) = d\big(B_{i-1}^{d-1}(x) - B_i^{d-1}(x)\big). \tag{1.5}$$

Proof. Suppose $1 \leq i \leq d-1$. By the product rule for differentiation,

$$\begin{aligned}(B_i^d)'(x) &= \binom{d}{i}\left(ix^{i-1}(1-x)^{d-i} - (d-i)x^i(1-x)^{d-i-1}\right) \\ &= d\left(\binom{d-1}{i-1}x^{i-1}(1-x)^{d-i} - \binom{d-1}{i}x^i(1-x)^{d-i-1}\right),\end{aligned}$$

which is the right hand side of (1.5). In the cases $i = 0$ and $i = d$,

$$(B_0^d)'(x) = -d(1-x)^{d-1} = -dB_0^{d-1}(x),$$
$$(B_d^d)'(x) = dx^{d-1} = dB_{d-1}^{d-1}(x),$$

which agree with (1.5) by the convention that $B_{-1}^{d-1} = B_d^{d-1} = 0$. □

1.4 Integration

There is a simple formula for the integral of a Bernstein basis polynomial over $[0,1]$.

Theorem 1.5 *For $d \geq 0$ and for all $i = 0, 1, \ldots, d$,*

$$\int_0^1 B_i^d(x)\,dx = \frac{1}{d+1}.$$

Proof. By the derivative formula (1.5),

$$(B_i^{d+1})'(x) = (d+1)\big(B_{i-1}^d(x) - B_i^d(x)\big)$$

for $i = 1, \ldots, d$. Integrating this equation with respect to x in $[0,1]$ gives

$$B_i^{d+1}(1) - B_i^{d+1}(0) = (d+1)\left(\int_0^1 B_{i-1}^d(x)\,dx - \int_0^1 B_i^d(x)\,dx\right),$$

and since the left hand side is zero for $i = 1, \ldots, d$, we deduce that

$$\int_0^1 B_{i-1}^d(x)\,dx = \int_0^1 B_i^d(x)\,dx.$$

Thus the integral over $[0,1]$ of every Bernstein basis polynomial B_i^d, $i = 0, 1, \ldots, d$, is the same. Since the B_i^d sum to one, we also deduce that the sum of these integrals is 1:

$$\sum_{i=0}^{d} \int_0^1 B_i^d(x)\,dx = \int_0^1 \sum_{i=0}^{d} B_i^d(x)\,dx = \int_0^1 1\,dx = 1.$$

Hence, since there are $d+1$ of these integrals, each integral must have the value $1/(d+1)$. □

1.5 Bernstein-Bézier polynomials

We call a polynomial $p \in \pi_d$ written in the form

$$p(x) = \sum_{i=0}^{d} c_i B_i^d(x), \tag{1.6}$$

a *Bernstein-Bézier polynomial* or *BB polynomial*. The coefficients $c_i \in \mathbb{R}$ are the *BB coefficients* of p. Although we are mainly interested in values of p for $x \in [0,1]$, we note that $p(x)$ is well defined for all $x \in \mathbb{R}$.

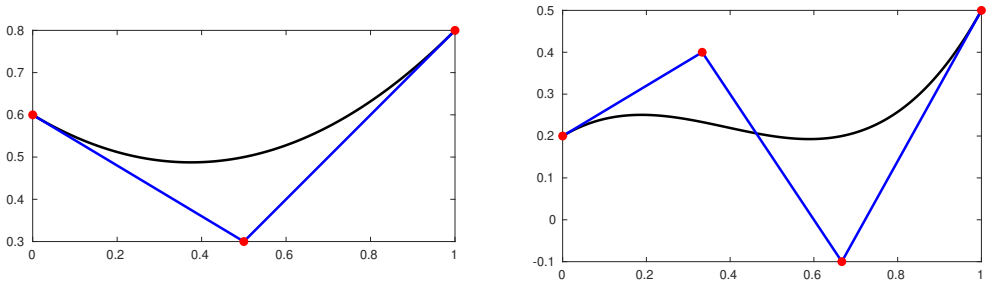

Figure 1.2: Quadratic and cubic BB polynomials.

From the BB coefficients we define the *control polygon* of p as the polygon passing through the points (ξ_i, c_i), $i = 0, 1, \ldots, d$, where the $d+1$ points ξ_i are the *domain points* of p with respect to $[a, b]$, defined as $\xi_i = i/d$. The shape of p relects, to a large extent, the shape of its control polygon. Figure 1.2 shows quadratic and cubic BB polynomials with their control polygons.

Various properties of BB polynomials follow from properties of the Bernstein basis polynomials. For example, from (1.2), we obtain the *endpoint property*,

$$p(0) = c_0, \qquad p(1) = c_d. \tag{1.7}$$

Since the values $B_i^d(x)$ sum to one for any $x \in \mathbb{R}$, any affine function Φ of p is the BB polynomial whose coefficients are $\Phi(c_i)$. This is because, if $\Phi(x) = \alpha x + \beta$, then

$$\Phi(p(x)) = \alpha \sum_{i=0}^{d} c_i B_i^d(x) + \beta = \sum_{i=0}^{d} (\alpha c_i + \beta) B_i^d(x) = \sum_{i=0}^{d} \Phi(c_i) B_i^d(x).$$

Since the B_i^d are, moreover, non-negative in $[0, 1]$, the value $p(x)$, $x \in [0, 1]$, is a convex combination of the coefficients $c_0, \ldots, c_d$ and so

$$\min_{0 \le i \le d} c_i \le p(x) \le \max_{0 \le i \le d} c_i, \qquad 0 \le x \le 1. \tag{1.8}$$

1.6 Bézier curves

By letting the coefficients of a BB polynomial be vector-valued, the polynomial becomes a parametric curve, which is commonly known as a *Bézier curve*. Thus, if we let $\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_d \in \mathbb{R}^k$ for some Euclidean space $\mathbb{R}^k$, $k \ge 2$, then we obtain a Bézier curve,

$$\mathbf{p}(x) = \sum_{i=0}^{d} \mathbf{c}_i B_i^d(x). \tag{1.9}$$

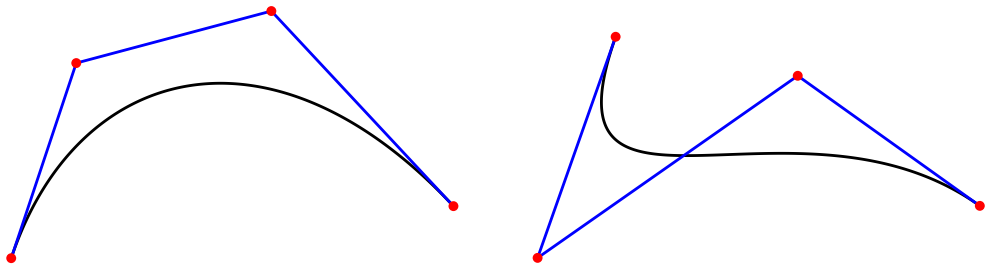

Figure 1.3: Cubic Bézier curves

This is a parametric polynomial curve, and if we restrict x to the interval $[0,1]$, then $[0,1]$ becomes the parameter domain of $\mathbf{p}$ and the curve is a mapping, $\mathbf{p} : [0,1] \to \mathbb{R}^k$. We call $\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_d$ the *control points* of $\mathbf{p}$, and we define the *control polygon* of $\mathbf{p}$ simply to be the polygon passing through the control points. Figure 1.3 shows two cubic Bézier curves in the plane with their control polygons. Similar to BB polynomials, a Bézier curve tends to mimic the shape of its control polygon, which is why it is a popular choice for designing geometry in an interactive graphical environment. As the user moves the control points interactively, the shape of the Bézier curve tends to change in an intuitive and predictable way.

Bézier curves have properties similar to BB polynomials. For example, from (1.2), we obtain the *endpoint property* of Bézier curves,

$$\mathbf{p}(0) = \mathbf{c}_0, \qquad \mathbf{p}(1) = \mathbf{c}_d.$$

Since the Bernstein basis polynomials sum to one for $x \in [0,1]$, every point $\mathbf{p}(x)$ is an affine combination of the control points $\mathbf{c}_0, \ldots, \mathbf{c}_n$. From this it follows that if Φ is an affine map $\mathbb{R}^k \to \mathbb{R}^k$, then the mapped curve $\Phi(\mathbf{p})$ has control points $\Phi(\mathbf{c}_i)$. So see this, recall that an affine map has the form

$$\Phi(\mathbf{x}) = A\mathbf{x} + \mathbf{b}, \qquad \mathbf{x} \in \mathbb{R}^k,$$

for some matrix A of dimension $k \times k$ and a vector $\mathbf{b}$ of length k. Then,

$$\Phi(\mathbf{p}(x)) = A \sum_{i=0}^{d} \mathbf{c}_i B_i^d(x) + \mathbf{b} = \sum_{i=0}^{d} (A\mathbf{c}_i + \mathbf{b}) B_i^d(x) = \sum_{i=0}^{d} \Phi(\mathbf{c}_i) B_i^d(x).$$

Since the Bernstein basis polynomials are non-negative in $[0,1]$, any point $\mathbf{p}(x)$, with $x \in [0,1]$, is a *convex combination* of the control points $\mathbf{c}_0, \ldots, \mathbf{c}_d$, and so $\mathbf{p}$, restricted to $[0,1]$, lies in the *convex hull* of its control points:

$$\operatorname{conv}\{\mathbf{c}_0, \ldots, \mathbf{c}_d\} = \left\{ \sum_{i=0}^{d} \mu_i \mathbf{c}_i : \mu_1, \ldots, \mu_d \geq 0, \sum_{i=0}^{d} \mu_i = 1 \right\}.$$

This same property also applies to each of the k coordinates of $\mathbf{p}$ separately, and so $\mathbf{p}$ restricted to $[0,1]$ also lies in the *bounding box*

$$[\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \cdots \times [\alpha_k, \beta_k],$$

where, if the point $\mathbf{c}_i$ has coordinates $c_{i1}, \ldots, c_{ik}$,

$$\alpha_j = \min_{0 \le i \le d} c_{ij} \quad \text{and} \quad \beta_j = \max_{0 \le i \le d} c_{ij}, \qquad j = 1, \ldots, k.$$

Bounding boxes are used in various algorithms, and are easier to compute than convex hulls.

1.7 The de Casteljau algorithm

Consider now how we evaluate a BB polynomial or Bézier curve, by which we mean how we compute the value of the polynomial or the point on the curve corresponding to a given input parameter x. The evaluation of a Bézier curve is similar to that of a BB polynomial, so it is sufficient to focus on the latter case.

Consider the computation of $p(x)$ in (1.6). One way to proceed is to evaluate the Bernstein basis polynomials B_i^d at x, by the recursion (1.4), and then apply the formula in (1.6). A more direct method is to use recursion on the coefficients, which is known as de Casteljau's algorithm. We initialize the algorithm by setting $c_i^0 = c_i$, $i = 0, 1, \ldots, d$. Then, for each $r = 1, \ldots, d$, let

$$c_i^r = (1-x)c_i^{r-1} + xc_{i+1}^{r-1}, \qquad i = 0, 1, \ldots, d-r. \tag{1.10}$$

Theorem 1.6 *The last value computed, c_0^d, is the value of $p(x)$ in (1.6).*

Proof. Consider the first step of the algorithm. By the recurrence (1.4),

$$\begin{aligned} p(x) &= \sum_{i=0}^{d} c_i^0 (xB_{i-1}^{d-1}(x) + (1-x)B_i^{d-1}(x)) \\ &= \sum_{i=0}^{d} c_i^0 xB_{i-1}^{d-1}(x) + \sum_{i=0}^{d} c_i^0 (1-x)B_i^{d-1}(x) \\ &= \sum_{i=0}^{d-1} c_{i+1}^0 xB_i^{d-1}(x) + \sum_{i=0}^{d-1} c_i^0 (1-x)B_i^{d-1}(x), \end{aligned}$$

where we used the convention that $B_{-1}^{d-1} = B_d^{d-1} = 0$, and hence, by (1.10),

$$p(x) = \sum_{i=0}^{d-1} c_i^1 B_i^{d-1}(x).$$

Continuing in this way we find that for any $r = 1, \ldots, d$,

$$p(x) = \sum_{i=0}^{d-r} c_i^r B_i^{d-r}(x). \tag{1.11}$$

The particular case $r = d$ gives us $p(x) = c_0^d$. □

This algorithm can be arranged in a triangular scheme as follows. In each column, each value is computed from two values from the previous column.

$$\begin{array}{ccccc} c_0^0 & c_0^1 & \cdots & c_0^{d-1} & c_0^d \\ c_1^0 & c_1^1 & \cdots & c_1^{d-1} & \\ \vdots & & \iddots & & \\ c_{d-1}^0 & c_{d-1}^1 & & & \\ c_d^0 & & & & \end{array}$$

Figure 1.4 illustrates the algorithm applied to the cubic BB polynomial of Figure 1.2

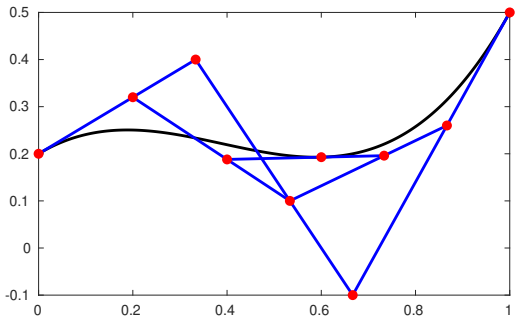

Figure 1.4: de Casteljau algorithm, $x = 0.6$

where $x = 0.6$.

It it sometimes useful to know that the points in the de Casteljau algorithm are themselves BB polynomials whose coefficients are subsets of $c_0, c_1, \ldots, c_d$.

Theorem 1.7 *The value c_i^r in the de Casteljau algorithm (1.10), viewed as a function of x, is the BB polynomial*

$$c_i^r = \sum_{j=0}^{r} c_{i+j} B_j^r(x). \tag{1.12}$$

Proof. We proceed by induction on $r \geq 0$. The statement clearly holds when $r = 0$. For $r \geq 1$, we may suppose that (1.12) holds when r is replaced by $r - 1$. Thus, from (1.10), we find

$$\begin{aligned} c_i^r = (1-x)c_i^{r-1} + xc_{i+1}^{r-1} &= (1-x)\sum_{j=0}^{r-1} c_{i+j} B_j^{r-1}(x) + x\sum_{j=0}^{r-1} c_{i+1+j} B_j^{r-1}(x) \\ &= \sum_{j=0}^{r} c_{i+j}((1-x)B_j^{r-1}(x) + xB_{j-1}^{r-1}(x)) \\ &= \sum_{j=0}^{r} c_{i+j} B_j^r(x), \end{aligned}$$

using again the convention that $B_{-1}^{r-1} = B_r^{r-1} = 0$.

□

1.8 Derivatives

We may also want to evaluate the derivatives of a BB polynomial or Bézier curve. Consider the computation of the first derivative of the BB polynomial p in (1.6) at a point x.

Theorem 1.8 *The first derivative of the BB polynomial p in (1.6) is*

$$p'(x) = d\sum_{i=0}^{d-1} \Delta c_i B_i^{d-1}(x),$$

where Δc_i denotes the forward difference, $\Delta c_i = c_{i+1} - c_i$.

Proof. Using Lemma 1.4,

$$\begin{aligned} p'(x) = \sum_{i=0}^{d} c_i (B_i^d)'(x) &= d\sum_{i=0}^{d} c_i (B_{i-1}^{d-1}(x) - B_i^{d-1}(x)) \\ &= d\left(\sum_{i=1}^{d} c_i B_{i-1}^{d-1}(x) - \sum_{i=0}^{d-1} c_i B_i^{d-1}(x)\right) \\ &= d\left(\sum_{i=0}^{d-1} c_{i+1} B_i^{d-1}(x) - \sum_{i=0}^{d-1} c_i B_i^{d-1}(x)\right), \end{aligned}$$

where we used our convention that $B_{-1}^{d-1} = B_d^{d-1} = 0$. Putting the two summations together and using the notation that $\Delta c_i = c_{i+1} - c_i$ completes the proof. □

This theorem implies that the derivative, $p'(x)$, is the BB polynomial in π_{d-1} whose coefficients are $d\Delta c_i$, $i = 0, 1, \ldots, d-1$. Thus by the endpoint property (1.7), it follows that

$$p'(0) = d\Delta c_0, \qquad p'(1) = d\Delta c_{d-1}, \tag{1.13}$$

from which we conclude that the gradient of p at $x = 0$ equals the gradient there of its control polygon, and likewise at $x = 1$. By (1.8), we also have

$$d \min_{0\le i\le d-1} \Delta c_i \le p'(x) \le d \max_{0\le i\le d-1} \Delta c_i, \qquad 0 \le x \le 1. \tag{1.14}$$

1.9 Higher order derivatives

We obtain higher order derivatives of p by applying Theorem 1.8 repeatedly, and by defining the r-th order forward difference of the coefficients by the recursion

$$\Delta^r c_i = \Delta^{r-1} c_{i+1} - \Delta^{r-1} c_i, \qquad r \ge 1,$$

where $\Delta^0 c_i = c_i$. One can show by induction that

$$\Delta^r c_0 = \sum_{i=0}^{r} \binom{r}{i} (-1)^{r-i} c_i, \tag{1.15}$$

so, for example,

$$\begin{aligned} \Delta^2 c_i &= c_i - 2c_{i+1} + c_{i+2}, \\ \Delta^3 c_i &= -c_i + 3c_{i+1} - 3c_{i+2} + c_{i+3}, \end{aligned}$$

and so on. Thus, one obtains

Theorem 1.9 *For $r = 0, 1, \ldots, d$, the r-th derivative of the BB polynomial p in (1.6) is*

$$p^{(r)}(x) = \frac{d^r}{dx^r} p(x) = \frac{d!}{(d-r)!} \sum_{i=0}^{d-r} \Delta^r c_i B_i^{d-r}(x).$$

The derivatives of p at the points $x = 0$ and $x = 1$ have simpler expressions:

Corollary 1.10 *For $r = 0, 1, \ldots, d$, the r-th order endpoint derivatives of the BB polynomial p in (1.6) are*

$$p^{(r)}(0) = \frac{d!}{(d-r)!} \Delta^r c_0, \qquad p^{(r)}(1) = \frac{d!}{(d-r)!} \Delta^r c_{d-r}.$$

1.10 Integration

There is a simple formula for the integral of a BB polynomial over $[0, 1]$.

Theorem 1.11 *The integral of p in (1.6) is the average of its coefficients,*

$$\int_0^1 p(x)\,dx = \frac{c_0 + c_1 + \cdots + c_d}{d+1}.$$

Proof. Integrating p in (1.6) gives

$$\int_0^1 p(x)\,dx = \sum_{i=0}^{d} c_i \int_0^1 B_i^d(x)\,dx. \tag{1.16}$$

By Theorem 1.5,

$$\int_0^1 B_i^d(x)\,dx = \frac{h}{d+1},$$

and substituting this into (1.16) gives the result. □

1.11 Exercises

1.1 It is sometimes necessary to convert a polynomial in BB form to monomial form. Consider a quadratic BB polynomial,

$$p(x) = c_0(1-x)^2 + 2c_1x(1-x) + c_2x^2.$$

Express p in the monomial form

$$p(x) = a_0 + a_1x + a_2x^2.$$

1.2 Consider a polynomial $p(x)$ of degree $\leq d$, for arbitrary d. Show that if

$$p(x) = \sum_{j=0}^{d} a_jx^j = \sum_{i=0}^{d} c_iB_i^d(x),$$

then

$$a_j = \binom{d}{j}\Delta^j c_0.$$

Hint: use a Taylor approximation to p to show that $a_j = p^{(j)}(0)/j!$.

1.3 We might also want to convert a polynomial from monomial form to BB form. Using Lemma 1.2, show that in the notation of the previous question,

$$c_i = \frac{i!}{d!} \sum_{j=0}^{i} \frac{(d-j)!}{(i-j)!} a_j.$$

1.4 Implement the de Casteljau algorithm for cubic Bézier curves in Matlab or Python (or some other programming language), taking repeated convex combinations. Choose a sequence of four control points and plot both the control polygon and the Bezier curve, like those in Figure 1.3.

1.5 Show that the graph, $g(x) = (x, p(x))$ of the BB polynomial p in (1.6) is a Bézier curve in $\mathbb{R}^2$, with control points (ξ_i, c_i), $i = 0, 1, \ldots, d$, where $\xi_i = i/d$. Hint: express x as a linear combination of $B_0^d(x), \ldots, B_d^d(x)$.

1.6 Show that the tangent vector $\mathbf{p}'(x)$ of the Bézier curve in (1.6) lies in the convex cone of the vectors $\Delta\mathbf{c}_i$, i.e., in

$$\text{cone}(\Delta\mathbf{c}_0, \ldots, \Delta\mathbf{c}_{d-1}) = \left\{ \sum_{i=0}^{d-1} \mu_i \Delta\mathbf{c}_i : \mu_1, \ldots, \mu_{d-1} \geq 0 \right\}.$$

1.7 Show that the first derivative of p in (1.6) can be expressed (and computed) as

$$p'(x) = d(c_1^{d-1} - c_0^{d-1}),$$

where c_0^{d-1}, c_1^{d-1} are the points of order $d-1$ in de Casteljau's algorithm (1.10).

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

1.9 Give a proof of the forward difference formula, (1.15).

1.10 The Bernstein approximation to a function $f : [0, 1] \to \mathbb{R}$ of order d is the polynomial $g : [0, 1] \to \mathbb{R}$ defined by

$$g(x) = \sum_{i=0}^{d} f\left(\frac{i}{d}\right) B_i^d(x).$$

Show that if f is a polynomial of degree $m \leq d$ then g has degree m.

1.11 Show that the length of the Bézier curve $\mathbf{p}$ in (1.9) is bounded by the length of its control polygon,

$$\text{length}(\mathbf{p}) \leq \sum_{i=0}^{d-1} \|\Delta\mathbf{c}_i\|.$$

Chapter 2

Splines in Bernstein-Bézier form

If we tried to model a complex function with a single polynomial, we would need a polynomial of very high degree. It is usually easier in practice to create a complex function by joining together several polynomials of low degree. The resulting function is a piecewise polynomial, or *spline*. To do this it helps to represent the polynomial pieces in BB form because the conditions for joining BB polynomials together with a certain order of smoothness are relatively simple.

2.1 BB polynomials on an arbitrary interval

In order to construct a spline whose polynomial pieces are defined on intervals of arbitrary lengths, we will represent each piece as a BB polynomial, and this requires defining a BB polynomial over an arbitrary interval $[a, b]$, rather than just over the 'canonical' interval $[0, 1]$. Any point $x \in [a, b]$ can be expressed as a convex combination of the endpoints a and b,

$$x = (1 - \lambda)a + \lambda b, \tag{2.1}$$

where

$$\lambda = \frac{x - a}{b - a}.$$

This is sometimes called the barycentric form of x with respect to $[a, b]$. The weights

$$1 - \lambda = \frac{b - x}{b - a} \qquad \text{and} \qquad \lambda = \frac{x - a}{b - a}$$

are the barycentric coordinates of x with respect to $[a, b]$.

We call a polynomial $p \in \pi_d$ written in the form

$$p(x) = \sum_{i=0}^{d} c_i B_i^d(\lambda), \tag{2.2}$$

a *BB polynomial with respect to the interval* $[a, b]$. The control polygon of p is now the polygon passing through the points (ξ_i, c_i), $i = 0, 1, \ldots, d$, where the *domain point* ξ_i is

$$\xi_i = \frac{d-i}{d}a + \frac{i}{d}b.$$

The endpoint property is now

$$p(a) = c_0, \qquad p(b) = c_d. \tag{2.3}$$

2.1.1 The de Casteljau algorithm

The de Castelaju algorithm of Chapter 1 generalizes in a simple way. We initialize the algorithm by setting $c_i^0 = c_i$, $i = 0, 1, \ldots, d$. Then, for each $r = 1, \ldots, d$, let

$$c_i^r = (1-\lambda)c_i^{r-1} + \lambda c_{i+1}^{r-1}, \qquad i = 0, 1, \ldots, d-r. \tag{2.4}$$

The last value computed, c_0^d, is the value of $p(x)$ in (2.2).

Analogous to Theorem 1.7, we can express the points in (2.4) as

$$c_i^r = \sum_{j=0}^{r} c_{i+j} B_j^r(\lambda). \tag{2.5}$$

2.1.2 Derivatives

The derivative formulas are similar to the $[0, 1]$ case, the only difference being that we must divide by the factor $b - a$ each time we differentiate, due to the fact that

$$\frac{d}{dx}\lambda = \frac{1}{b-a}.$$

Thus, defining $h = b - a$, we find by the chain rule that

$$p'(x) = \frac{1}{h}\frac{d}{d\lambda}p(x),$$

and more generally,

$$p^{(r)}(x) = \frac{1}{h^r}\frac{d^r}{d\lambda^r}p(x),$$

for $r = 0, 1, \ldots, d$. Therefore, applying the derivative formulas from the previous chapter we deduce

Theorem 2.1 *Let $h = b - a$. For $r = 0, 1, \ldots, d$, the r-th derivative of the BB polynomial p in (2.2) is*

$$p^{(r)}(x) = \frac{d!}{(d-r)!}\frac{1}{h^r}\sum_{i=0}^{d-r}\Delta^r c_i B_i^{d-r}(\lambda).$$

In particular, the endpoint derivatives are as follows.

Corollary 2.2 *The r-th endpoint derivatives of the BB polynomial p in (2.2) are*

$$p^{(r)}(a) = \frac{d!}{(d-r)!}\frac{1}{h^r}\Delta^r c_0, \qquad p^{(r)}(b) = \frac{d!}{(d-r)!}\frac{1}{h^r}\Delta^r c_{d-r}.$$

We can use these formulas to join BB polynomials together smoothly.

2.2 Joining polynomial pieces together

Suppose that we want to construct a spline s consisting of two polynomials pieces p and q of degree at most d, over two consecutive intervals $[a, b]$ and $[b, c]$, respectively. Thus we want to define

$$s(x) = \begin{cases} p(x), & a \le x < b; \\ q(x), & b \le x < c; \end{cases}$$

Then s is continuous at the breakpoint $x = b$ if $p(b) = q(b)$. We can represent both p and q as BB polynomials,

$$p(x) = \sum_{i=0}^{d} c_i B_i^d(\lambda), \qquad q(x) = \sum_{i=0}^{d} e_i B_i^d(\mu),$$

for coefficients $c_i, e_i \in \mathbb{R}$, and where $\lambda = (x-a)/(b-a)$ and $\mu = (x-b)/(c-b)$. Then, by the endpoint property of BB polynomials, the condition for the continuity of s is that the last coefficient of p equals the first coefficient of q, i.e.,

$$c_d = e_0. \tag{2.6}$$

More generally, if we want to ensure that s has C^r continuity, we require that the endpoint derivatives of p and q at $x = b$ of all orders up to r are equal, i.e, that

$$p^{(k)}(b) = q^{(k)}(b), \qquad k = 0, 1, \ldots, r.$$

We have already derived formulas for these quantities in Corollary 2.2 and so, if we agree beforehand that s will have certain derivatives up to some order r at $x = b$ it is

easy to find the coefficients of p and q needed to match these derivatives. This is the approach we will use for example to ensure C^1 continuity between cubic polynomial pieces in Section 2.4. However, there is an another approach to ensuring continuity between BB polynomials. To obtain C^r continuity between p and q above, it is sufficient to express the first $r+1$ coefficients of q directly in terms of the last $r+1$ coefficients of p. The starting point for this comes from Corollary 2.2:

Theorem 2.3 *The spline s has C^r continuity, $r = 0, 1, \ldots, d$, if and only if*

$$\frac{1}{(b-a)^k}\Delta^k c_{d-k} = \frac{1}{(c-b)^k}\Delta^k e_0, \qquad k = 0, 1, \ldots r.$$

Now it remains to express $e_0, e_1, \ldots, e_r$ as functions of $c_{d-r}, c_{d-r+1}, \ldots, c_d$. For example, the condition for C^1 continuity is (2.6) combined with the condition

$$\frac{c_d - c_{d-1}}{b-a} = \frac{e_1 - e_0}{c-b}.$$

This equation and (2.6) uniquely determine the two coefficients e_0 and e_1 from the coefficients c_d and c_{d-1},

$$e_0 = c_d, \qquad e_1 = (1-\alpha)c_{d-1} + \alpha c_d, \tag{2.7}$$

where

$$\alpha = \frac{c-a}{b-a}.$$

The coefficient α is easy to remember because it is the local coordinate of the point c with respect to the interval $[a, b]$, analogous to λ being the local coordinate of x. Notice, however, that while both λ and $1-\lambda$ are non-negative for $x \in [a, b]$, the coefficient $(1-\alpha)$ is negative because $c > b$.

Setting $r = 2$ in Theorem 2.3, a lengthier calculation shows that the condition for C^2 continuity of s can be expressed as the two equations in (2.7) plus the equation

$$e_2 = (1-\alpha)^2 c_{d-2} + 2(1-\alpha)\alpha c_{d-1} + \alpha^2 c_d. \tag{2.8}$$

These conditions generalize as follows:

Theorem 2.4 *The spline s has C^r continuity, $r = 0, 1, \ldots, d$, if and only if*

$$e_i = \sum_{j=0}^{i} c_{d-i+j} B_j^i(\alpha), \qquad i = 0, 1, \ldots, r.$$

We will postpone the proof of this till later (using blossoming).

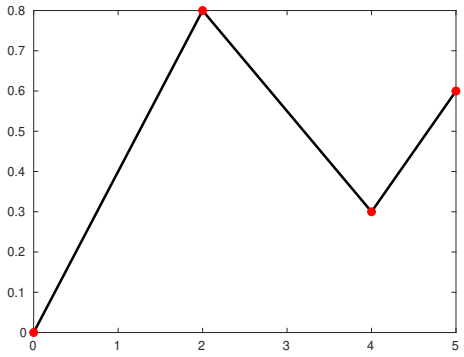

Figure 2.1: Piecewise linear interpolant.

2.3 Linear spline interpolation

As a first example of contructing a spline to match given data, let us consider piecewise linear interpolation. Suppose that $x_1, x_2, \ldots, x_m \in \mathbb{R}$ is an increasing sequence of points and that $y_1, y_2, \ldots, y_m \in \mathbb{R}$ are associated values. We let $a = x_1$ and $b = x_m$ and we would like to find a spline $g : [a, b] \to \mathbb{R}$ such that in each interval $[x_i, x_{i+1}]$, $i = 1, 2, \ldots, m-1$, g is a linear polynomial, and such that

$$g(x_i) = y_i, \qquad i = 1, 2, \ldots, m. \tag{2.9}$$

The solution to this is simple. For each $i = 1, 2, \ldots, m-1$, we let g_i be the linear polynomial

$$g_i(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i} y_i + \frac{x - x_i}{x_{i+1} - x_i} y_{i+1}. \tag{2.10}$$

We then define g by $g(x) = g_i(x)$ for $x_i \le x \le x_{i+1}$. So g_i is the i-th polynomial piece of g. Since g_{i-1} and g_i have the common value y_i at x_i for all $i = 2, \ldots, m-1$, it follows that g is continuous, and we write $g \in C[a, b]$.

We can alternatively express g_i using linear BB polynomials:

$$g_i(x) = y_i B_0^1(\lambda_i) + y_{i+1} B_1^1(\lambda_i),$$

where $\lambda_i = (x - x_i)/(x_{i+1} - x_i)$.

Figure 2.1 shows an example of piecewise linear interpolation, with three linear pieces. The data is

$$\begin{aligned} (x_1, x_2, x_3, x_4) &= (0, 2, 4, 5), \\ (y_1, y_2, y_3, y_4) &= (0, 0.8, 0.3, 0.6). \end{aligned} \tag{2.11}$$

In some applications the values y_i will be function values $y_i = f(x_i)$, $i = 1, \ldots, m$, for some function $f : [a, b] \to \mathbb{R}$. In this case, we can ask "what is the difference between f and g?" Equivalently, if we let $e = f - g$ be the error function on $[a, b]$, we can ask "what is the size of e in some sense?" Of course e is zero at the interpolation points x_i, but not in general in between. One way to measure the size of e is to use the *max* norm or *infinity* norm of e, defined as

$$\|e\| = \max_{a \le x \le b} |e(x)|.$$

Under the assumption that f is smooth enough, the max norm of the error is proportional to h^2 where h is the mesh size

$$h = \max_{i=1,\ldots,m-1} h_i,$$

and $h_i = x_{i+1} - x_i$.

Theorem 2.5 *If $f \in C^2[a, b]$ then*

$$\|f - g\| \le \frac{1}{8} h^2 \|f''\|.$$

Proof. Let $x \in [x_i, x_{i+1}]$. Since $g_i(x_i) = f(x_i)$ and $g_i(x_{i+1}) = f(x_{i+1})$, the Newton error formula for linear polynomial interpolation tells us that

$$f(x) - g_i(x) = (x - x_i)(x - x_{i+1}) \frac{f''(\xi_i)}{2!},$$

for some point $\xi_i \in (x_i, x_{i+1})$. The maximum of the function $(x - x_i)(x_{i+1} - x)$ over $x \in [x_i, x_{i+1}]$ is attained at $x = (x_i + x_{i+1})/2$, and has the value $h_i^2/4$ there. Therefore,

$$|f(x) - g(x)| = (x - x_i)(x_{i+1} - x) \frac{|f''(\xi_i)|}{2!} \le \frac{h_i^2}{8} \|f''\| \le \frac{h^2}{8} \|f''\|.$$

Taking the maximum of $|f(x) - g(x)|$ over x in all of $[a, b]$ gives the result. □

This theorem shows that if a and b and f are fixed, and if we increase the number of points x_i sampled from $[a, b]$ in such a way that $h \to 0$, then the associated splines g will converge to f, and at the rate of $O(h^2)$. For example, if we double the number of existing intervals $[x_i, x_{i+1}]$ by adding a new interpolation point at each midpoint $(x_i + x_{i+1})/2$ then the error will approximately go down by a factor of 4.

2.4 Cubic Hermite spline interpolation

We obtain a smoother spline by using polynomial pieces of higher degree. A popular choice of degree is three. We can build a cubic spline $g : [a, b] \to \mathbb{R}$ which is continuously differentiable by letting its pieces g_i be cubic polynomials and such that g matches both the values y_i in (2.9), and slopes, i.e., first derivatives,

$$g'(x_i) = s_i, \qquad i = 1, 2, \ldots, m. \tag{2.12}$$

Theorem 2.6 *There is a unique C^1 cubic spline g satisfying the interpolation conditions (2.9) and (2.12). We can express the i-th piece of g as*

$$g_i(x) = \sum_{j=0}^{3} c_j B_j^3(\lambda_i), \tag{2.13}$$

where

$$c_0 = y_i, \quad c_1 = y_i + \frac{h_i}{3} s_i, \quad c_2 = y_{i+1} - \frac{h_i}{3} s_{i+1}, \quad c_3 = y_{i+1}, \tag{2.14}$$

and $\lambda_i = (x - x_i)/h_i$ and $h_i = x_{i+1} - x_i$.

Proof. Let g_i be a BB polynomial of the form (2.13). By Corollary 2.2,

$$g_i(x_i) = c_0, \quad g_i'(x_i) = \frac{3}{h_i}(c_1 - c_0), \quad g_i(x_{i+1}) = c_3, \quad g_i'(x_{i+1}) = \frac{3}{h_i}(c_3 - c_2).$$

Using the interpolation conditions (2.9) and (2.12), and solving for the coefficients c_0, c_1, c_2, c_3 gives (2.14).

To show the uniqueness of the solution, suppose $\tilde{g}_i \in \pi_3$ also matches the four interpolation conditions at x_i and x_{i+1}. Then the difference $p = \tilde{g}_i - g_i$ is also in π_3 and

$$p(x_j) = p'(x_j) = 0, \qquad j = i, i + 1.$$

So p is a cubic with at least four roots counting multiplicities and the Fundamental Theorem of Algebra implies that $p = 0$. □

Figure 2.2 shows an example of cubic Hermite interpolation, with three cubic pieces. The data is as in (2.11) together with the derivative data

$$(s_1, s_2, s_3, s_4) = (0, 0.1, -0.1, 0.1). \tag{2.15}$$

As in the previous section, in some applications the values y_i will be function values $y_i = f(x_i)$, and the values s_i could be slopes, $s_i = f'(x_i)$. If f is smooth enough, the error is now proportional to h^4, which means that the convergence of these cubic splines to f will be faster than linear splines.

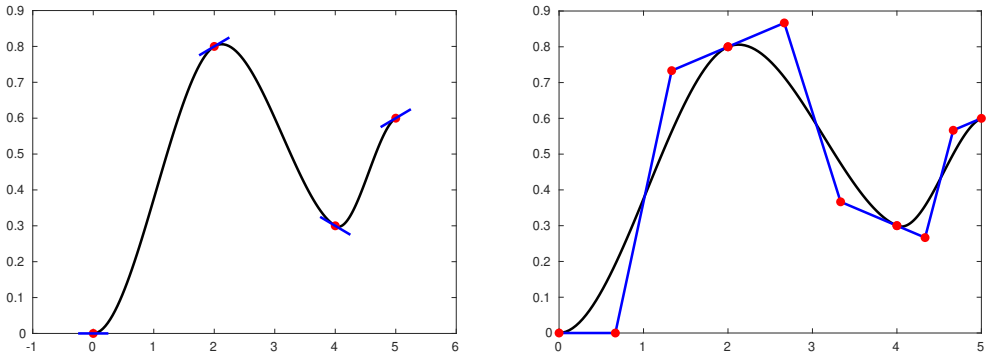

Figure 2.2: Cubic Hermite interpolant (left), BB control polygons (right).

Theorem 2.7 *If $f \in C^4[a,b]$ then*

$$\|f - g\| \leq \frac{1}{384} h^4 \|f^{(4)}\|.$$

Proof. Let $x \in [x_i, x_{i+1}]$. Since $g_i(x_j) = f(x_j)$ and $g_i'(x_j) = f'(x_j)$ for $j = i, i+1$, the Newton error formula for cubic Hermite polynomial interpolation implies that

$$f(x) - g_i(x) = (x - x_i)^2 (x - x_{i+1})^2 \frac{f^{(4)}(\xi_i)}{4!},$$

for some point $\xi_i \in (x_i, x_{i+1})$. Therefore,

$$|f(x) - g(x)| = ((x - x_i)(x_{i+1} - x))^2 \frac{|f^{(4)}(\xi_i)|}{4!} \leq \left(\frac{h_i^2}{4}\right)^2 \frac{\|f^{(4)}\|}{24} \leq \frac{h^4}{384} \|f^{(4)}\|.$$

□

2.5 C^2 cubic spline interpolation

Another approach to constructing a cubic spline g to fit data values $y_1, \ldots, y_m$ is to force g to have C^2 continuity at the interior points $x_2, \ldots, x_{m-1}$. If we then count degrees of freedom we find that we need to place two extra conditions on g to ensure the uniqueness of g. One way to do this is to fix the first derivative of g at the two endpoints x_1 and x_m. Thus we will construct a spline $g \in C^2[a,b]$ which, as before, has cubic pieces and interpolates the values y_i, but in addition satisfies the Hermite end conditions

$$g'(x_1) = s_1 \quad \text{and} \quad g'(x_m) = s_m. \tag{2.16}$$

Theorem 2.8 *There is a unique C^2 cubic spline g satisfying the interpolation conditions (2.9) and (2.16).*

Proof. One approach to solving this problem is to let g be the cubic Hermite spline of the previous section, and to compute the interior slopes $s_2, \ldots, s_{m-1}$ so as to ensure that g has C^2 continuity at the points $x_2, \ldots, x_{m-1}$. In this approach we have $m-2$ variables and $m-2$ conditions. To ensure the C^2 continuity of g, we require

$$g''_{i-1}(x_i) = g''_i(x_i), \qquad i = 2, \ldots, m-1.$$

From Corollary 2.2, in the notation of (2.13),

$$g''_i(x_i) = \frac{6}{h_i^2}(c_0 - 2c_1 + c_2) = \frac{6}{h_i}\left(\frac{y_{i+1} - y_i}{h_i} - \frac{2s_i + s_{i+1}}{3}\right). \tag{2.17}$$

Similarly, writing g_{i-1} as

$$g_{i-1}(x) = \sum_{j=0}^{3} \tilde{c}_j B_j^3(\lambda_{i-1}),$$

we find

$$g''_{i-1}(x_i) = \frac{6}{h_{i-1}^2}(\tilde{c}_1 - 2\tilde{c}_2 + \tilde{c}_3) = \frac{6}{h_{i-1}}\left(-\frac{y_i - y_{i-1}}{h_{i-1}} + \frac{s_{i-1} + 2s_i}{3}\right). \tag{2.18}$$

Equating (2.17) and (2.18) and rearranging gives

$$\alpha_i s_{i-1} + 2s_i + \beta_i s_{i+1} = b_i, \tag{2.19}$$

where

$$\alpha_i = \frac{h_i}{h_{i-1} + h_i}, \qquad \beta_i = \frac{h_{i-1}}{h_{i-1} + h_i},$$

and

$$b_i = \frac{3}{h_{i-1} + h_i}\left(h_i \frac{y_i - y_{i-1}}{h_{i-1}} + h_{i-1}\frac{y_{i+1} - y_i}{h_i}\right).$$

This gives us the linear system of equations

$$\begin{bmatrix} 2 & \beta_2 & & & \\ \alpha_3 & 2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_{m-2} & 2 & \beta_{m-2} \\ & & & \alpha_{m-1} & 2 \end{bmatrix} \begin{bmatrix} s_2 \\ s_3 \\ \vdots \\ s_{m-2} \\ s_{m-1} \end{bmatrix} = \begin{bmatrix} b_2 - \alpha_2 s_1 \\ b_3 \\ \vdots \\ b_{m-2} \\ b_{m-1} - \beta_{m-1} s_m \end{bmatrix},$$

The matrix is tridiagonal and strictly diagonally dominant (in its rows) and therefore has a unique solution. □

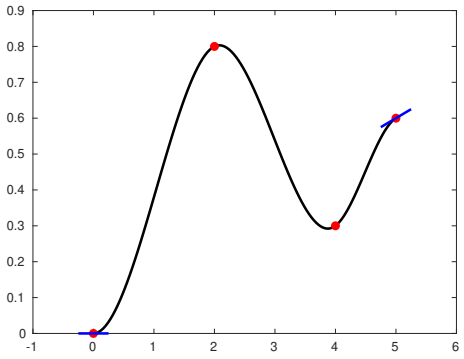

Figure 2.3: C^2 cubic spline interpolant.

Figure 2.3 shows an example of C^2 cubic interpolation, with three cubic pieces. The data is as in (2.11) together with the same end derivatives, $s_1 = 0$, $s_4 = 0.1$, as we used in the C^1 Hermite case (2.15).

2.6 Minimization of second derivatives

The C^2 cubic spline interpolant g has a remarkable property, that it has the least second derivative of all C^2 interpolants in the L_2 sense.

Theorem 2.9 *Let g be the C^2 cubic spline interpolant solving (2.9) and (2.16), and let h be any C^2 function satisfying the same interpolation conditions. Then*

$$\int_a^b (g''(x))^2\,dx \le \int_a^b (h''(x))^2\,dx, \tag{2.20}$$

with equality if and only if $h = g$.

Proof. Let $e = h - g$. Then $e \in C^2[a,b]$ and

$$e(x_i) = 0, \quad i = 1,\dots,m, \quad \text{and} \qquad e'(x_1) = e'(x_m) = 0. \tag{2.21}$$

Then $h = g + e$ and so

$$\int (h'')^2 = \int (g'')^2 + 2\int g''e'' + \int (e'')^2, \tag{2.22}$$

and therefore

$$\int (h'')^2 - \int (g'')^2 \ge 2\int \phi e'',$$

where $\phi = g''$, which is piecewise linear. We now compute the integral on the right:

$$\begin{aligned}\int_a^b \phi e'' &= \sum_{i=1}^{m-1} \int_{x_i}^{x_{i+1}} \phi e'' \\ &= \sum_{i=1}^{m-1} \left\{ \left[\phi e'\right]_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} \phi' e' \right\} \\ &= \sum_{i=1}^{m-1} \left\{ \phi(x_{i+1})e'(x_{i+1}) - \phi(x_i)e'(x_i) - \phi'|_{[x_i,x_{i+1}]} \int_{x_i}^{x_{i+1}} e' \right\} \\ &= \phi(x_m)e'(x_m) - \phi(x_1)e'(x_1) - \sum_{i=1}^{m-1} \left\{ \phi'|_{[x_i,x_{i+1}]}(e(x_{i+1}) - e(x_i)) \right\}. \end{aligned} \tag{2.23}$$

So, by (2.21), this integral is zero and this proves the inequality (2.20). To complete the proof, suppose that the integrals in (2.20) are equal. Then by (2.22), $\int (e'')^2 = 0$. This implies that e is a linear function, $e(x) = a_0 + a_1 x$. Since $e(x_1) = e(x_m) = 0$, this means that $e = 0$, and we conclude that $h = g$. □

2.7 Natural end conditions

An alternative to imposing the Hermite end conditions (2.16) is to impose the so-called *natural end conditions*, which demand that the second derivative og g is zero at the endpoints, i.e.,

$$g''(x_1) = g''(x_m) = 0. \tag{2.24}$$

There is again a unique solution. To see this, we now treat all the slopes $s_1, s_2, \ldots, s_m$ as unknowns. The requirement of C^2 continuity at the interior points $x_2, \ldots, x_{m-1}$ gives us again the $m-2$ equations (2.19). In addition, using equations (2.17) and (2.18), the end conditions (2.24) expand to

$$2s_1 + s_2 = b_1, \qquad s_{m-1} + 2s_m = b_m,$$

where

$$b_1 = 3\left(\frac{y_2 - y_1}{h_1}\right), \qquad b_m = 3\left(\frac{y_m - y_{m-1}}{h_{m-1}}\right).$$

We thus have m equations in the m unknowns $s_1, \ldots, s_m$,

$$\begin{bmatrix} 2 & 1 & & & & \\ \alpha_2 & 2 & \beta_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \alpha_{m-1} & 2 & \beta_{m-1} & \\ & & & 1 & 2 & \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{m-1} \\ s_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{m-1} \\ b_m \end{bmatrix}.$$

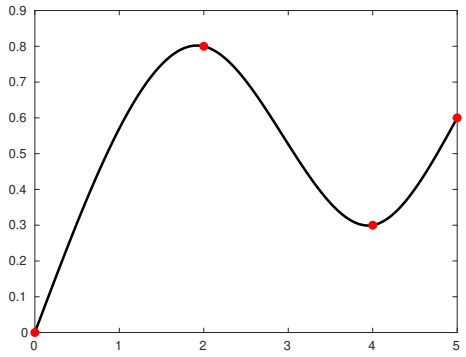

Figure 2.4: Cubic spline interpolant with natural end conditions.

This is again a strictly row diagonally dominant system of equations, and thus has a unique solution.

Figure 2.4 shows an example, with the data again as in (2.11).

The minimization property of Theorem 2.9 also holds for this natural spline: see Exercise 2.5.

2.8 Exercises

2.1 Prove equation (2.8).

2.2 Implement the de Casteljau algorithm for a planar Bézier curve of arbitrary degree d over a general interval $[a, b]$. Use the routine to make a program to plot the quadratic spline curve $\mathbf{s} : [0, 2] \to \mathbb{R}^2$, with pieces

$$\mathbf{p}(t) = \sum_{i=0}^{2} \mathbf{c}_i B_i^2(t), \qquad 0 \le t \le 1,$$

$$\mathbf{q}(t) = \sum_{i=0}^{2} \mathbf{d}_i B_i^2(t-1), \qquad 1 < t \le 2,$$

where $\mathbf{c}_0 = (-1, 1)$, $\mathbf{c}_1 = (-1, 0)$, $\mathbf{c}_2 = (0, 0)$, and $\mathbf{d}_0 = (0, 0)$, $\mathbf{d}_1 = (1, 0)$, $\mathbf{d}_2 = (2, 1)$.

2.3 What is the order of continuity of $\mathbf{s}$ in Exercise 2.2 at the breakpoint $t = 1$?

2.4 The curvature of a parametric curve $\mathbf{r}(t)$ in $\mathbb{R}^2$ can be expressed as

$$\kappa(t) = \frac{\mathbf{r}'(t) \times \mathbf{r}''(t)}{\|\mathbf{r}'(t)\|^3},$$

where $(a_1, a_2) \times (b_1, b_2) := a_1 b_2 - a_2 b_1$. What are the curvatures of $\mathbf{p}$ and $\mathbf{q}$ in Exercise 2.2 at the breakpoint $t = 1$? What can you say about the smoothness of $\mathbf{s}$?

2.5 Show that the minimization property of Theorem 2.9 also holds for the natural spline of Section 2.7.

Chapter 3

B-splines

In the previous chapter we saw several examples of spline functions and curves, where we represented each local polynomial piece as a BB polynomial. It turns out that there is an alternative way of representing a spline, as a linear combination of so-called B-splines. An advantage of this representation is that the smoothness of the spline is 'automatic' in the sense that it simply inherits the smoothness of the B-splines. B-splines have several properties in common with Bernstein basis polynomials, including non-negativity and the partition of unity property, and, like BB polynomials and Bézier curves, splines tend to mimic the shape of their B-spline control polygon, thus providing a convenient way to contruct and modify a spline.

3.1 Divided differences

We will define B-splines using divided differences, so we start by recalling some of their basic properties. More divided difference theory can be found in [4] and [1].

The divided difference of a function f at the points $x_0, x_1, \ldots, x_k$ is the coefficient of x^k of the unique polynomial $p(x)$ of degree at most k that interpolates f at these points. In other words, it is the coefficient c_k, the 'leading coefficient', in the representation

$$p(x) = \sum_{i=0}^{k} c_i x^i \tag{3.1}$$

of this interpolating polynomial. We denote it by $[x_0, x_1, \ldots, x_k]f$ and it is said to have k-th order.

We note that in the special case that f is itself a polynomial of degree less than k, then $p = f$ and $c_k = 0$ and so $[x_0, \ldots, x_k]f = 0$.

3.1.1 Distinct points

If the x_i are distinct, p is the Lagrange polynomial interpolant to f. We find $[x_0]f = f(x_0)$. For $k \geq 1$, we can express p in terms of two polynomials, p_0 and p_1, of degree at most $k-1$, as follows: we let p_0 be the interpolant to f at $x_0, x_1, \ldots, x_{k-1}$ and we let p_1 be the interpolant to f at $x_1, x_2, \ldots, x_k$. One can then easily check that

$$p(x) = \frac{x_k - x}{x_k - x_0} p_0(x) + \frac{x - x_0}{x_k - x_0} p_1(x), \tag{3.2}$$

and then, by considering the coefficient of x^k of the right hand side we deduce the recursion

$$[x_0, x_1, \ldots, x_k]f = \frac{[x_1, \ldots, x_k]f - [x_0, \ldots, x_{k-1}]f}{x_k - x_0}.$$

The first examples are therefore

$$[x_0]f = f(x_0), \qquad [x_0, x_1]f = \frac{f(x_1) - f(x_0)}{x_1 - x_0},$$

$$[x_0, x_1, x_2]f = \frac{[x_1, x_2]f - [x_0, x_1]f}{x_2 - x_0} = \frac{\frac{f(x_2)-f(x_1)}{x_2-x_1} - \frac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2 - x_0}.$$

We can also derive an explicit formula for the divided difference from the Lagrange form of p,

$$p(x) = \sum_{i=0}^{k} L_i(x) f(x_i), \qquad L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{k} \frac{x - x_j}{x_i - x_j}. \tag{3.3}$$

By considering the coefficient of x^k of p, we deduce the formula

$$[x_0, x_1, \ldots, x_k]f = \sum_{i=0}^{k} \frac{f(x_i)}{\prod_{\substack{j=0 \\ j \neq i}}^{k} (x_i - x_j)}. \tag{3.4}$$

So, for example, we can write a second order divided difference as

$$[x_0, x_1, x_2]f = \frac{f(x_0)}{(x_0 - x_1)(x_0 - x_2)} + \frac{f(x_1)}{(x_1 - x_0)(x_1 - x_2)} + \frac{f(x_2)}{(x_2 - x_0)(x_2 - x_1)}.$$

3.1.2 Arbitrary points

If any of the points $x_0, x_1, \ldots, x_k$ are equal we understand the interpolant p to be the Hermite interpolant to f. By this we mean that if x_i has multiplicity m, i.e., x_i appears m times in the sequence $x_0, x_1, \ldots, x_k$, then p and all its derivatives up

to order $m-1$ agree with f at this point x_i. We will always assume that f has sufficiently many derivatives at the points x_i for this to make sense. In the special case that all the points are equal, $x_0 = x_1 = \cdots = x_k$, the Hermite interpolant p is the Taylor approximation to f at x_0,

$$p(x) = \sum_{i=0}^{k} \frac{f^{(i)}(x_0)}{i!} x^i,$$

and so

$$[x_0, \ldots, x_k]f = [\underbrace{x_0, \ldots, x_0}_{k+1}]f = \frac{f^{(k)}(x_0)}{k!}. \tag{3.5}$$

If only some of the points are equal, then in analogy to the case of distinct points, the divided difference can be expressed recursively. One can show that as along as x_0 and x_k are distinct then equation (3.2) holds regardless of whether the remaining points are distinct or not. More generally, suppose that x_i and x_j are distinct and let p_0 be the interpolant of degree $\leq k-1$ to f at all points except x_j, and let p_1 be the interpolant of degree $\leq k-1$ to f at all points except x_i. Then one can show that

$$p(x) = \frac{x_j - x}{x_j - x_i} p_0(x) + \frac{x - x_i}{x_j - x_i} p_1(x). \tag{3.6}$$

Then, by taking the coefficient of x^k on both sides, we have

$$[x_0, \ldots, x_k]f = \frac{[x_0, \ldots, \hat{x}_i, \ldots, x_k]f - [x_0, \ldots, \hat{x}_j, \ldots, x_k]f}{x_j - x_i},$$

where $x_0, \ldots, \hat{x}_i, \ldots, x_k$ means the sequence $x_0, \ldots, x_k$ with the point x_i removed. For example, if $x_0 \neq x_1$,

$$[x_0, x_0, x_1]f = \frac{[x_0, x_1]f - [x_0, x_0]f}{x_1 - x_0} = \frac{\frac{f(x_1) - f(x_0)}{x_1 - x_0} - f'(x_0)}{x_1 - x_0},$$

which is a linear combination of $f(x_0), f'(x_0), f(x_1)$:

$$[x_0, x_0, x_1]f = c_{00} f(x_0) + c_{01} f'(x_0) + c_{10} f(x_1),$$

where

$$c_{00} = \frac{-1}{(x_1 - x_0)^2}, \quad c_{01} = \frac{-1}{x_1 - x_0}, \quad c_{10} = \frac{1}{(x_1 - x_0)^2}.$$

In general, it follows from the recursion that for a sequence of distinct points $x_0, x_1, \ldots, x_k$ with multiplicities $m_0, m_1, \ldots, m_k$, there are coefficients $c_{i,r}$, independent of f, such that

$$[\underbrace{x_0, \ldots, x_0}_{m_0}, \ldots, \underbrace{x_k, \ldots, x_k}_{m_k}]f = \sum_{i=0}^{k} \sum_{r=0}^{m_i - 1} c_{ir} f^{(r)}(x_i). \tag{3.7}$$

In other words, the divided difference is a linear combination of f and its derivatives at $x_0, x_1, \ldots, x_k$, where the highest order derivative at x_i is $m_i - 1$.

3.1.3 Leibniz rule

Later, we will make use of a convenient formula, called the Leibniz rule, for the divided difference of a product of two functions. For the product of functions f and g, the Leibniz rule is

$$[x_0, x_1, \ldots, x_k](fg) = \sum_{i=0}^{k} [x_0, \ldots, x_i] f \, [x_i, \ldots, x_k] g. \tag{3.8}$$

The name 'Leibniz rule' is due to the fact that in the case that $x_0 = \cdots = x_k$, equation (3.8) reduces to the well known Leibniz formula for the k-th derivative of fg (due to (3.5)).

3.2 B-splines

We define B-splines as follows. For any integers $d \geq 0$ and $n \geq 1$, we call a sequence $\mathbf{t} = (t_1, t_2, \ldots, t_{n+d+1})$, $t_i \in \mathbb{R}$, a *knot vector* if $t_i \leq t_{i+1}$, and $t_i < t_{i+d+1}$. For any real number x we write

$$x_+ = \begin{cases} x, & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$$

For $i = 1, 2 \ldots, n$, we define the i-th B-spline $B_{i,d,\mathbf{t}}$ by the formula

$$B_{i,d,\mathbf{t}}(x) = (t_{i+d+1} - t_i)[t_i, t_{i+1}, \ldots, t_{i+d+1}](\cdot - x)_+^d, \qquad x \in \mathbb{R}. \tag{3.9}$$

Here, the divided difference $[t_i, , \ldots, t_{i+d+1}]$ applies to the function

$$f(t) = \big((t - x)_+\big)^d, \tag{3.10}$$

with x fixed. We will sometimes use the alternative notation

$$B_{i,d,\mathbf{t}}(x) = B[t_i, t_{i+1}, \ldots, t_{i+d+1}](x),$$

and sometimes just write $B_{i,d}$ or $B_{i,\mathbf{t}}$ or even B_i if it is clear from the context what d and $\mathbf{t}$ are.

In the coming sections we will explore many properties of B-splines. One property that is simple to derive is the following.

Lemma 3.1 *If $x \notin [t_i, t_{i+d+1}]$ then $B_{i,d}(x) = 0$.*

Proof. If $x > t_{i+d+1}$, the function $f(t)$ in (3.10) is zero for all $t \in [t_i, t_{i+d+1}]$ and so $[t_i, \ldots, t_{i+d+1}]f = 0$. If $x < t_i$ then $f(t) = (t-x)^d$ for $t \in [t_i, t_{i+d+1}]$. Since $(t-x)^d$ is a polynomial in t of degree d, $[t_i, \ldots, t_{i+d+1}]f = 0$. □

It follows from this that in a non-empty knot interval $[t_\mu, t_{\mu+1}]$ there are only $d+1$ B-splines that can be non-zero, specifically, $B_{\mu-d,d}, \ldots, B_{\mu,d}$.

Another property that follows easily from the definition is that of *translation invariance*. For any $y \in \mathbb{R}$,

$$B[t_i + y, \ldots, t_{i+d+1} + y](x) = B[t_i, \ldots, t_{i+d+1}](x - y). \tag{3.11}$$

This is a simple consequence of (3.9) because with $f(t)$ as in (3.10),

$$[t_i + y, \ldots, t_{i+d+1} + y]f = [t_i, \ldots, t_{i+d+1}]g,$$

where $g(t) = f(t + y)$, which means that

$$g(t) = ((t + y) - x)_+^d = (t - (x - y))_+^d.$$

3.3 Piecewise polynomials and smoothness

We next observe that B-splines are piecewise polynomials (splines), and we determine their order of smoothness. We will do this in two parts, first considering distinct knots, then multiple knots.

3.3.1 Distinct knots

Suppose that the knots $t_i, \ldots, t_{i+d+1}$ are distinct. From the divided difference formula (3.4) we can write $B_{i,d}$ as

$$B_{i,d}(x) = \sum_{j=i}^{i+d+1} a_j (t_j - x)_+^d, \tag{3.12}$$

where $a_j = (t_{i+d+1} - t_i) / \prod_{\substack{k=i \\ k \neq j}}^{i+d+1} (t_j - t_k)$. Using this formula we deduce

Theorem 3.2 *Suppose $t_i, \ldots, t_{i+d+1}$ are distinct. Then $B_{i,d}$ is a piecewise polynomial of degree d with smoothness of order C^{d-1}.*

Proof. This follows from the fact that $(t_j - x)_+^d$, viewed as a function of x, is a polynomial of degree d for x in any knot interval $[t_k, t_{k+1}]$, and has order of continuity C^{d-1} at $x = t_j$. □

Consider now the simplest example, that $d = 0$. Then equation (3.12) implies that

$$B_{i,0}(x) = -(t_i - x)_+^0 + (t_{i+1} - x)_+^0,$$

and so $B_{i,0}(x) = 0$ for $x < t_i$ and $x > t_{i+1}$ and $B_{i,0}(x) = 1$ for $t_i < x < t_{i+1}$. So $B_{i,0}$ is discontinuous at $x = t_i$ and $x = t_{i+1}$, which confirms what the theorem claims in this case, that the smoothness is C^{-1}. In order to avoid ambiguity, we can simply make a convention that at each knot, the B-splines $B_{i,0}$ are either right continuous or left continuous. For example, if we agree that they are right continuous at every knot we have

$$B_{i,0}(x) = \begin{cases} 1, & \text{if } x \in [t_i, t_{i+1}); \\ 0, & \text{otherwise.} \end{cases}$$

For distinct knots, when $d \geq 1$, the B-splines $B_{i,d}$ are continuous. When $d = 1$, equation (3.12) implies that

$$B_{i,1}(x) = \frac{1}{h_i}(t_i - x)_+ - \left(\frac{1}{h_i} + \frac{1}{h_{i+1}}\right)(t_{i+1} - x)_+ + \frac{1}{h_{i+1}}(t_{i+2} - x)_+.$$

To write down simpler formulas for $B_{i,d}$ when $d = 1$, and even $d = 2$, it will be better to use the recursion formula which we will derive in Section 3.4.

3.3.2 Multiple knots

Suppose that the knots defining $B_{i,d}$ have the multiplicities

$$(t_i, t_{i+1}, \ldots, t_{i+d+1}) = (\underbrace{\tau_0, \ldots, \tau_0}_{m_0}, \ldots, \underbrace{\tau_k, \ldots, \tau_k}_{m_k}), \tag{3.13}$$

where $\tau_0 < \tau_1 < \cdots < \tau_k$. Since

$$\frac{d^r}{dt^r}(t - x)_+^d = \frac{d!}{(d-r)!}(t - x)_+^{d-r},$$

it follows from (3.7) that there are coefficients $a_{j,r}$, independent of x, such that

$$B_{i,d}(x) = \sum_{j=0}^{k} \sum_{r=0}^{m_j - 1} a_{j,r}(\tau_j - x)_+^{d-r}. \tag{3.14}$$

Theorem 3.3 *If $t_i, \ldots, t_{i+d+1}$ have the multiplicities in (3.13) then $B_{i,d}$ is a piecewise polynomial of degree d with smoothness of order C^{d-m_j} at τ_j, $j = 0, 1, \ldots, k$.*

Proof. This follows from the fact that $(\tau_j - x)_+^{d-r}$, as a function of x, is a polynomial of degree $d-r$ for x in any knot interval $[t_k, t_{k+1}]$, and has order of continuity C^{d-r-1} at $x = \tau_j$. □

The theorem implies that $B_{i,d}$ is a continuous function on $\mathbb{R}$ if the maximum multiplicity of its knots is d. This further implies that if the maximum multiplicity of its knots is d, then $B_{i,d}(t_i) = B_{i,d}(t_{i+d+1}) = 0$.

If $B_{i,d}$ has a knot of multiplicity $d+1$ then it is discontinuous there. Just as in the $d = 0$ case, we can avoid ambiguity by making the convention that all B-splines (of any degree) at any particular knot are right continuous, or that they are all left continuous.

3.4 Recursion

In order to derive simple formulas for B-splines of low degree, and in order to evaluate B-splines of arbitrary degree it is preferable to use the recursion formula, which we now derive. The recursion formula also shows that B-splines are *non-negative*.

Theorem 3.4 *For $d \geq 1$,*

$$B_{i,d}(x) = \frac{x - t_i}{t_{i+d} - t_i} B_{i,d-1}(x) + \frac{t_{i+d+1} - x}{t_{i+d+1} - t_{i+1}} B_{i+1,d-1}(x). \tag{3.15}$$

We will assume in the proof that $t_i < t_{i+d}$ and $t_{i+1} < t_{i+d+1}$, so that the division in (3.15) is well defined. We will discuss how to handle the cases $t_i = t_{i+d}$ and $t_{i+1} = t_{i+d+1}$ in Subsection 3.5.3.

Proof. Starting from the definition (3.9), we use the fact that $f(t) = (t - x)_+^d$ can be written as the product $f(t) = g(t)h(t)$, where $g(t) = t - x$ and $h(t) = (t - x)_+^{d-1}$. We then apply the divided difference $[t_i, t_{i+1}, \ldots, t_{i+d+1}]$ to this product, and use the Leibniz rule (3.8). Since

$$\begin{aligned} [t_i]g &= t_i - x, \\ [t_i, t_{i+1}]g &= 1, \\ [t_i, \ldots, t_j]g &= 0, \qquad j \geq i + 2, \end{aligned}$$

we find

$$[t_i, \ldots, t_{i+d+1}]f = (t_i - x)[t_i, \ldots, t_{i+d+1}]h + [t_{i+1}, \ldots, t_{i+d+1}]h. \tag{3.16}$$

Since

$$[t_i, \ldots, t_{i+d+1}]h = \frac{[t_{i+1}, \ldots, t_{i+d+1}]h - [t_i, \ldots, t_{i+d}]h}{t_{i+d+1} - t_i}, \tag{3.17}$$

multiplying both sides of (3.16) by $t_{i+d+1} - t_i$ gives

$$\begin{aligned} B_{i,d}(x) &= (t_i - x)([t_{i+1}, \ldots, t_{i+d+1}]h - [t_i, \ldots, t_{i+d}]h) \\ &\quad + (t_{i+d+1} - t_i)[t_{i+1}, \ldots, t_{i+d+1}]h \\ &= (x - t_i)[t_i, \ldots, t_{i+d}]h + (t_{i+d+1} - x)[t_{i+1}, \ldots, t_{i+d+1}]h. \end{aligned} \tag{3.18}$$

By the definition of $B_{i,d-1}$ and $B_{i+1,d-1}$, and assuming that $t_i < t_{i+d}$ and $t_{i+1} < t_{i+d+1}$, we can rewrite (3.18) in the form of (3.15). □

A simple but important consequence of the recursion formula is that, by induction on d, $B_{i,d}(x) > 0$ for all $x \in (t_i, t_{i+d+1})$. This implies that $B_{i,d}(x) \geq 0$ for $x \in \mathbb{R}$ and that the support of $B_{i,d}$ is $[t_i, t_{i+d+1}]$.

3.5 Examples

Let us now consider some examples of B-splines.

3.5.1 Distinct knots

A linear B-spline depends on three knots and has continuity C^0 when they are distinct. With $d = 1$, the recursion formula gives

$$B_{i,1}(x) = \begin{cases} (x - t_i)/(t_{i+1} - t_{i+1}), & \text{if } x \in [t_i, t_{i+1}); \\ (t_{i+2} - x)/(t_{i+2} - t_{i+1}), & \text{if } x \in [t_{i+1}, t_{i+2}). \end{cases}$$

A quadratic B-spline depends on four knots and has continuity C^1 when they are distinct. With $d = 2$, two steps of the recursion formula gives

$$B_{i,2}(x) = \begin{cases} \frac{(x-t_i)^2}{(t_{i+2}-t_i)(t_{i+1}-t_i)}, & \text{if } x \in [t_i, t_{i+1}); \\ \frac{(x-t_i)(t_{i+2}-x)}{(t_{i+2}-t_i)(t_{i+2}-t_{i+1})} + \frac{(t_{i+3}-x)(x-t_{i+1})}{(t_{i+3}-t_{i+1})(t_{i+2}-t_{i+1})} & \text{if } x \in [t_{i+1}, t_{i+2}); \\ \frac{(t_{i+3}-x)^2}{(t_{i+3}-t_{i+1})(t_{i+3}-t_{i+2})}, & \text{if } x \in [t_{i+2}, t_{i+3}). \end{cases}$$

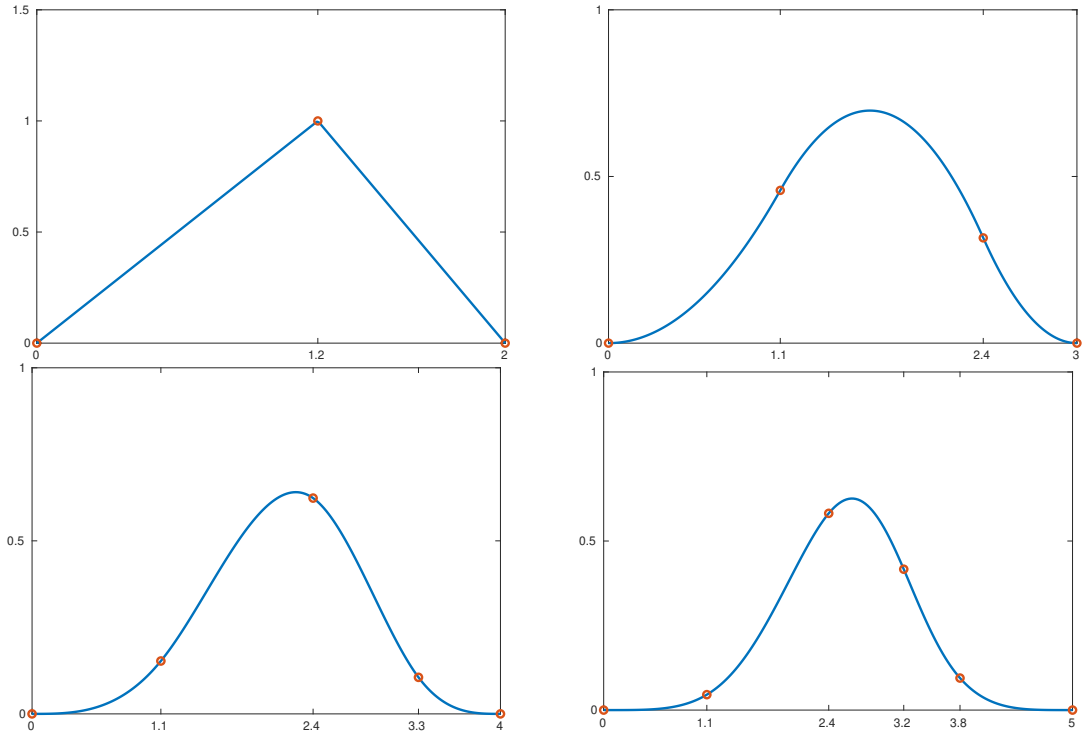

Figure 3.1: Linear, quadratic, cubic, and quartic B-splines.

Figure 3.1 shows linear, quadratic, cubic, and quartic B-splines with distinct knots. Figure 3.2 shows the five cubic B-splines defined on the knot interval

$$\mathbf{t} = (0.0, 1.1, 2.4, 3, 4, 5.2, 6.0, 7.2, 8).$$

Here, $d = 3$, $n = 5$, and $\mathbf{t}$ has length $n + d + 1 = 9$.

3.5.2 Uniform B-splines

The B-splines on a uniform knot vector are sometimes of special interest, and are simply shifts of a single B-spline. Let the knots be $t_i = i$ for all integers $i \in \mathbb{Z}$. We call the B-spline

$$M_d(x) = B_{0,d}(x) = B[0, 1, \ldots, d+1](x), \qquad x \in \mathbb{R},$$

the *cardinal B-spline* of degree d, which has continuity C^{d-1} at the knots. All the uniform B-splines are translates of M_d because, due to (3.11),

$$B_{i,d}(x) = B[i, i+1, \ldots, i+d+1](x) = B[0, 1, \ldots, d+1](x - i) = M_d(x - i).$$

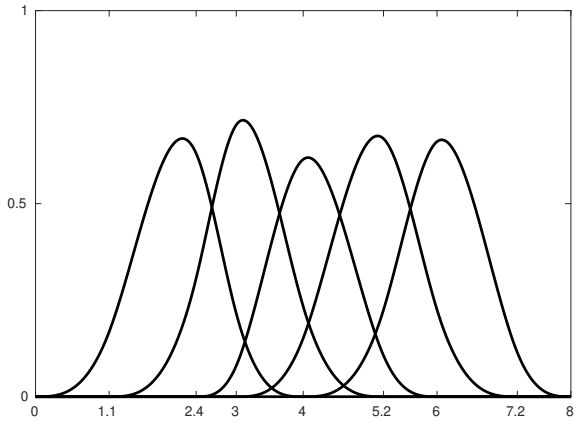

Figure 3.2: Cubic B-splines on distinct knots, $n = 5$.

In particular, for $d \geq 1$, $B_{1,d-1}(x) = B[1, \ldots, d+1](x) = M_{d-1}(x-1)$ and the recursion formula implies that

$$M_d(x) = \frac{x}{d} M_{d-1}(x) + \frac{d+1-x}{d} M_{d-1}(x-1).$$

3.5.3 Multiple knots

Let us next suppose that we have an arbitrary knot vector and consider what happens in the recursion formula if $t_i = t_{i+d}$ or $t_{i+1} = t_{i+d+1}$, or both. Suppose first that $t_i = t_{i+d}$. If $x \neq t_i$ then

$$[t_i, \ldots, t_{i+d}](\cdot - x)_+^{d-1} = 0, \tag{3.19}$$

and the first term on the right hand side of (3.18) disappears. Moreover, in the case that $x = t_i$ we can simply define this term to be zero. A similar consideration applies to the case that $t_{i+1} = t_{i+d+1}$. So a more precise version of the recurrence is as follows. We now allow the possibility that $t_i = t_{i+d+1}$ in $\mathbf{t}$. The only restriction on $\mathbf{t}$ is that it is non-decreasing. We define

$$s_1(x) = \frac{x - t_i}{t_{i+d} - t_i} B_{i,d-1}(x), \qquad \text{and} \qquad s_2(x) = \frac{t_{i+d+1} - x}{t_{i+d+1} - t_{i+1}} B_{i+1,d-1}(x).$$

Then

$$B_{i,d}(x) = \begin{cases} 0, & \text{if } t_i = t_{i+d+1}; \\ s_1(x), & \text{if } t_i < t_{i+d} \text{ and } t_{i+1} = t_{i+d+1}; \\ s_2(x), & \text{if } t_i = t_{i+d} \text{ and } t_{i+1} < t_{i+d+1}; \\ s_1(x) + s_2(x), & \text{otherwise.} \end{cases} \tag{3.20}$$

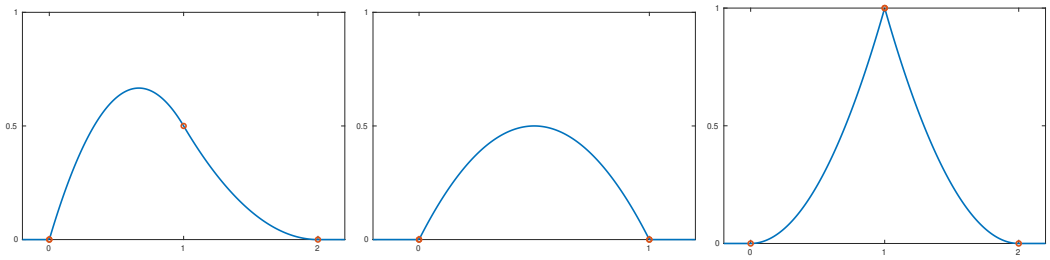

Figure 3.3: The B-splines $B[0,0,1,2]$, $B[0,0,1,1]$, and $B[0,1,1,2]$.

There still remains the possibility that $t_i < t_{i+d}$ and $t_{i+1} < t_{i+d+1}$, but that $B_{i,d-1}$ and $B_{i+1,d-1}$ are discontinuous at x. This happens if $t_i < t_{i+1} = \cdots = t_{i+d} < t_{i+d+1}$ and $x = t_{i+1}$. Fortunately, the formula (3.20) is still correct if we keep to the convention that all B-splines at a knot are right continuous (or left continuous).

Figure 3.3 shows some quadratic B-splines with double knots, and illustrates the fact that they are C^1 at simple knots and C^0 at double knots.

3.5.4 Bernstein polynomials

The Bernstein basis polynomials are examples of B-splines with multiple knots. In fact, for $i = 0, 1, \ldots, d$, the i-th Bernstein basis polynomial with respect to the interval $[0,1]$ can be expressed as the B-spline

$$B_i^d(x) = \binom{d}{i} x^i (1-x)^{d-i} = B[\underbrace{0,\ldots,0}_{d+1-i},\underbrace{1,\ldots,1}_{i+1}](x), \qquad x \in [0,1]. \tag{3.21}$$

To see this, observe that it is clearly true when $d = 0$. For $d \geq 1$, we can use induction on d. Assuming that (3.21) holds if d is replaced by $d-1$, the B-spline recursion formula implies that

$$\begin{aligned} B[\underbrace{0,\ldots,0}_{d+1-i},\underbrace{1,\ldots,1}_{i+1}](x) &= xB[\underbrace{0,\ldots,0}_{d+1-i},\underbrace{1,\ldots,1}_{i}](x) + (1-x)B[\underbrace{0,\ldots,0}_{d-i},\underbrace{1,\ldots,1}_{i+1}](x) \\ &= xB_{i-1}^{d-1}(x) + (1-x)B_i^{d-1}(x). \end{aligned}$$

But the right hand side is simply $B_i^d(x)$ by Lemma 1.3 (the recursion for Bernstein basis polynomials).

3.6 Value of a B-spline at a knot

A useful property of a B-spline is that its value at one of its knots equals the value there of the B-spline of lower degree resulting from removing that knot from the

knot vector.

Theorem 3.5 *For any $j = i, i+1, \ldots, i+d+1$,*

$$B[t_i, \ldots, t_{i+d+1}](t_j) = B[t_i, \ldots, t_{j-1}, t_{j+1}, \ldots, t_{i+d+1}](t_j).$$

Proof. See Exercise 3.9. □

A corollary of this is:

Corollary 3.6 *If $z = t_{i+1} = \cdots = t_{i+d} < t_{i+d+1}$ then $B_{i,d}(z) = 1$ and $B_{j,d}(z) = 0$ for all $j \neq i$.*

Proof. From Theorem 3.5,

$$B_{i,d}(z) = B[t_i, t_{i+d+1}](z) = 1,$$

and for $j \neq i$, one can similarly use the theorem to show that $B_{j,d}(z) = 0$. □

3.7 Closed knot vector

It is often useful in practice to use a knot vector $\mathbf{t}$ in which $t_1 = \cdots = t_{d+1}$ and $t_{n+1} = \cdots = t_{n+d+1}$, and all remaining knots have multiplicity at most d. In this case we will say that $\mathbf{t}$ is a *closed knot vector* with respect to the degree d. If we further make the convention that $B_{1,d}$ is right continuous at t_{d+1} and $B_{n,d}$ is left continuous at t_{n+1}, all the B-splines $B_{i,d}$ are continuous on the whole closed interval $I = [t_{d+1}, t_{n+1}]$. As we will see, the advantage of the closed knot vector is that $B_{1,d}(t_{d+1}) = 1$ and $B_{i,d}(t_{d+1}) = 0$ for all $i > 1$, and similarly, $B_{n,d}(t_{n+1}) = 1$ and $B_{i,d}(t_{n+1}) = 0$ for all $i < n$. Figure 3.4 illustrates this, showing the five cubic B-splines defined on the closed knot vector $\mathbf{t} = (3, 3, 3, 3, 4, 5.2, 5.2, 5.2, 5.2)$.

3.8 Linear combinations of B-splines

Having now defined B-splines, we can form spline functions and spline curves as linear combinations of B-splines.

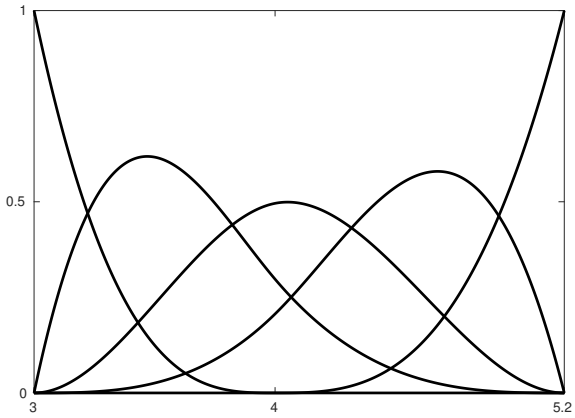

Figure 3.4: Cubic B-splines on a closed knot vector.

3.8.1 Spline functions

Consider again a knot vector $\mathbf{t} = (t_i)_{i=1}^{n+d+1}$ where $d \geq 0$ and $n \geq 1$. We can define n B-splines of degree d on $\mathbf{t}$, specifically $B_{1,d}, \ldots, B_{n,d}$. For any coefficients $c_i \in \mathbb{R}$, $i = 1, \ldots, n$, the linear combination

$$s(x) = \sum_{i=1}^{n} c_i B_{i,d}(x) \tag{3.22}$$

is a spline function, or simply a spline, $s : \mathbb{R} \to \mathbb{R}$, i.e., a piecewise polynomial whose pieces are of degree at most d. We will denote by $\mathcal{S}_{d,\mathbf{t}}$ the linear space of all such splines,

$$\mathcal{S}_{d,\mathbf{t}} = \operatorname{span}(B_{1,d}, \ldots, B_{n,d}) = \Big\{ \sum_{i=1}^{n} c_i B_{i,d} \,|\, c_i \in \mathbb{R}, i = 1, \ldots, n \Big\}.$$

We define the control polygon of s in (3.22) to be the piecewise linear function passing through the points (t_i^*, c_i), $i = 1, \ldots, n$, where t_i^* is the knot average

$$t_i^* = \frac{t_{i+1} + \cdots + t_{i+d}}{d}.$$

We will often restrict the spline function s to the interval $I := [t_{d+1}, t_{n+1}]$. The reason for this is that for $x \in I$, the B-splines sum to 1:

$$\sum_{i=1}^{n} B_{i,d}(x) = 1, \qquad x \in I,$$

and therefore form a partition of unity there. This will be shown in the next chapter. This means that

$$\min_i c_i \le s(x) \le \max_i c_i, \qquad x \in I.$$

In the special case that $\mathbf{t}$ is a closed knot vector, the spline s in (3.22) will have the endpoint property that

$$s(t_{d+1}) = c_1, \qquad \text{and} \qquad s(t_{n+1}) = c_n.$$

3.8.2 Spline curves

Spline curves are defined in a similar way to spline functions, except that the coefficients are replaced by points in $\mathbb{R}^k$ for some $k \ge 2$. For points $\mathbf{c}_i \in \mathbb{R}^k$, $i = 1, \ldots, n$, the linear combination

$$\mathbf{s}(x) = \sum_{i=1}^{n} \mathbf{c}_i B_{i,d}(x) \tag{3.23}$$

is a parametric spline curve, and we call the points $\mathbf{c}_i$ the control points of $\mathbf{s}$. The control polygon of $\mathbf{s}$ is simply the polygon in $\mathbb{R}^k$ passing through $\mathbf{c}_1, \ldots, \mathbf{c}_n$. The restriction of $\mathbf{s}$ to the parameter interval $I = [t_{d+1}, t_{n+1}]$ lies in the convex hull of the control polygon.

In the special case that $\mathbf{t}$ is a closed knot vector, the spline curve $\mathbf{s}$ will have the endpoint property that

$$\mathbf{s}(t_{d+1}) = \mathbf{c}_1, \qquad \text{and} \qquad \mathbf{s}(t_{n+1}) = \mathbf{c}_n.$$

3.9 Evaluation

Let us consider now how to compute the value $s(x)$ of the spline function in (3.22), given some $x \in I$. First we locate an index $\mu \in \{d+1, \ldots, n\}$ such that $[t_\mu, t_{\mu+1}]$ is non-empty and contains x. Then $s(x)$ is given by the local summation,

$$s(x) = \sum_{i=\mu-d}^{\mu} c_i B_{i,d}(x) \tag{3.24}$$

because all the B-splines other than $B_{\mu-d,d}, \ldots, B_{\mu,d}$ are zero at x. Then there are two ways of evaluating s at x, i.e, calculating the value $s(x)$.

3.9.1 Algorithm 1: B-spline recursion

The first algorithm is to use the B-spline recursion directly to compute the $d+1$ values $B_{i,d}(x)$, $i = \mu - d, \ldots, d$, and then to multiply them by the coefficients c_i, $i = \mu - d, \ldots, \mu$ and sum them up. The recursion formula (3.15) gives us a triangular scheme for computing the B-splines. We fix $x \in [t_\mu, t_{\mu+1}]$ and initialize the scheme by setting $B_{\mu,0} = 1$. Then, for $r = 1, 2, \ldots, d$, and $i = \mu - r, \ldots, \mu$, we set

$$B_{i,r} = \frac{x - t_i}{t_{i+r} - t_i} B_{i,r-1} + \frac{t_{i+r+1} - x}{t_{i+r+1} - t_{i+1}} B_{i+1,r-1}.$$

Here, we are using the fact that both $B_{\mu-r,r-1}$ and $B_{\mu+1,r-1}$ are zero at x. The flow of computations is as follows, where in each column, each value is computed from two values from the previous column.

$$\begin{array}{ccccc} B_{\mu,0} & B_{\mu-1,1} & B_{\mu-2,2} & \cdots & B_{\mu-d,d} \\ & B_{\mu,1} & B_{\mu-1,2} & \cdots & B_{\mu-d+1,d} \\ & & B_{\mu,2} & \cdots & B_{\mu-d+2,d} \\ & & & \ddots & \vdots \\ & & & & B_{\mu,d} \end{array}$$

3.9.2 Algorithm 2: de Boor algorithm

Alternatively, we can use recursion on the coefficients c_j in (3.24). This is de Boor's algorithm, and it generalizes de Casteljau's algorithm. We fix x and initialize the algorithm by setting $c_i^0 = c_i$, $i = \mu - d, \ldots, \mu$. Then for $r = 1, \ldots, d$, and $i = \mu - d + r, \ldots, \mu$, we set

$$c_i^r = \frac{t_{i+d-r+1} - x}{t_{i+d-r+1} - t_i} c_{i-1}^{r-1} + \frac{x - t_i}{t_{i+d-r+1} - t_i} c_i^{r-1}. \tag{3.25}$$

Theorem 3.7 *The last value computed, c_μ^d, is the value of s at x in (3.24).*

Proof. To prove this, consider the first step of the algorithm. By the B-spline recurrence for the $B_{i,d}$ we have

$$\begin{aligned} s(x) &= \sum_{i=\mu-d}^{\mu} c_i^0 \left(\frac{x - t_i}{t_{i+d} - t_i} B_{i,d-1}(x) + \frac{t_{i+d+1} - x}{t_{i+d+1} - t_{i+1}} B_{i+1,d-1}(x) \right) \\ &= \sum_{i=\mu-d+1}^{\mu} \left(\frac{t_{i+d} - x}{t_{i+d} - t_i} c_{i-1}^0 + \frac{x - t_i}{t_{i+d} - t_i} c_i^0 \right) B_{i,d-1}(x), \end{aligned}$$

where we have used the fact that both $B_{\mu-d,d-1}$ and $B_{\mu+1,d-1}$ are zero at x. Hence by the definition of c_i^1 in (3.25),

$$s(x) = \sum_{i=\mu-d+1}^{\mu} c_i^1 B_{i,d-1}(x).$$

Continuing in this way we find that for any $r = 1, \ldots, d$,

$$s(x) = \sum_{i=\mu-d+r}^{\mu} c_i^r B_{i,d-r}(x). \tag{3.26}$$

The case $r = d$ gives us $s(x) = c_\mu^d$. □

This algorithm can also be arranged in a triangular scheme, as follows. In each column, each value is computed from two values from the previous column.

$$\begin{array}{ccccc} c_{\mu-d}^0 & c_{\mu-d+1}^1 & \cdots & c_{\mu-1}^{d-1} & c_\mu^d \\ c_{\mu-d+1}^0 & c_{\mu-d+2}^1 & \cdots & c_\mu^{d-1} & \\ \vdots & & \iddots & & \\ c_{\mu-1}^0 & c_\mu^1 & & & \\ c_\mu^0 & & & & \end{array}$$

3.10 Exercises

3.1 Suppose that x_0, x_1, x_2 are distinct, and let $f_i = f(x_i)$, $i = 0, 1, 2$, for some function f. Show by direct calculation that the recursive formula

$$[x_0, x_1, x_2]f = \frac{\frac{f_2 - f_1}{x_2 - x_1} - \frac{f_1 - f_0}{x_1 - x_0}}{x_2 - x_0}$$

can be expressed as

$$[x_0, x_1, x_2]f = \sum_{i=0}^{2} \frac{f_i}{\prod_{j \neq i}(x_i - x_j)}.$$

3.2 Show that if $f(x) = 1/x$ and that $x_0, x_1, \ldots, x_k \neq 0$ then

$$[x_0, \ldots, x_k]f = \frac{(-1)^k}{x_0 x_1 \cdots x_k}.$$

3.3 Prove the Leibniz rule for divided differences:

$$[x_0, x_1, \ldots, x_k](fg) = \sum_{i=0}^{k} [x_0, \ldots, x_i]f \, [x_i, \ldots, x_k]g.$$

Hint: let p and q be the polynomials of degree $\leq k$ that interpolate f and g respectively at $x_0, x_1, \ldots, x_k$, and express p and q as

$$p(x) = \sum_{i=0}^{k} (x - x_0) \cdots (x - x_{i-1})[x_0, \ldots, x_i]f,$$

$$q(x) = \sum_{j=0}^{k} (x - x_{j+1}) \cdots (x - x_k)[x_j, \ldots, x_k]g.$$

Now consider the polynomial pq.

3.4 Use the recursion formula (Theorem 3.4) to show that

a) $B[0,0,0,1](x) = (1-x)^2 B[0,1](x)$,

b) $B[0,0,1,2](x) = x(2 - \frac{3}{2}x)B[0,1](x) + \frac{1}{2}(2-x)^2 B[1,2](x)$,

c) $B[0,1,1,2](x) = x^2 B[0,1](x) + (2-x)^2 B[1,2](x)$.

3.5 a) Prove Theorem 3.5, and b) use it to show that for distinct knots,

$$B_{i,2}(t_{i+1}) = \frac{t_{i+1} - t_i}{t_{i+2} - t_i}, \qquad B_{i,2}(t_{i+2}) = \frac{t_{i+3} - t_{i+2}}{t_{i+3} - t_{i+1}}.$$

3.6 Show that

$$B[\underbrace{a, \ldots, a}_{d}, b](x) = \left(\frac{b-x}{b-a}\right)^d B[a,b](x),$$

$$B[a, \underbrace{b, \ldots, b}_{d}](x) = \left(\frac{x-a}{b-a}\right)^d B[a,b](x).$$

Use this to show that

$$B[a, \underbrace{b, \ldots, b}_{d}, c](x) = \left(\frac{x-a}{b-a}\right)^d B[a,b](x) + \left(\frac{c-x}{c-b}\right)^d B[b,c](x).$$

Show that this B-spline is continuous on $\mathbb{R}$.

3.7 Show that

$$B_{i,d}(x) = (-1)^{d+1}(t_{i+d+1} - t_i)[t_i, t_{i+1}, \ldots, t_{i+d+1}](x - \cdot)_+^d, \qquad x \in \mathbb{R}.$$

Hint: use the fact that

$$(t-x)_+^d - (-1)^{d+1}(x-t)_+^d = (t-x)^{d-1}.$$

Use this to show that for distinct knots,

$$B_{i,d}(x) = (t_{i+d+1} - t_i) \sum_{j=i}^{i+d+1} \frac{(x-t_j)_+^d}{\prod_{\substack{k=i \\ k\neq j}}^{i+d+1}(t_k - t_j)}.$$

3.8 Use the recursion formula to show that (for $d \geq 1$)

$$\sum_{i=1}^{n} B_{i,d}(x) = 1, \qquad x \in [t_{d+1}, t_{n+1}].$$

3.9 Prove Theorem 3.5. Hint: show first that the case $j = i$ follows from (3.16). For any other j, derive an analogy of (3.16) by first expressing $[t_i, \ldots, t_{i+d+1}]$ in the form

$$[t_j, t_i, \ldots, t_{j-1}, t_{j+1}, \ldots, t_{i+d+1}].$$

3.10 Given a knot vector $\mathbf{t} = (t_i)_{i=1}^{n+d+1}$ and a real number x, with $x \in [t_{d+1}, t_{n+1}]$, write a routine to determine an index μ such that $x \in [t_\mu, t_{\mu+1}]$ with $[t_\mu, t_{\mu+1}]$ non-empty. A call to this routine is always needed before running Algorithms 1 and 2 of Section 3.9. By letting μ be an input parameter as well as an output parameter you can minimze the searching for example during plotting when the routine is called with many values of x in the same knot interval.

3.11 Implement Algorithm 1 of Section 3.9 in your favourite programming language.

3.12 Implement Algorithm 2 of Section 3.9 in your favourite programming language.

3.13 Suppose that $d = 3$ and that the knot vector is

$$\mathbf{t} = (0, 1, 2, 3, 4).$$

With this knot vector we can only associate one cubic B-spline, $B_{1,3}$. Therefore, if we want to compute $B_{1,3}(x)$ for some $x \in (0, 4)$, Algorithms 1 and 2 of Section 3.9 do not apply. Show, however, that by augmenting the knot vector to

$$\hat{\mathbf{t}} = (0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4),$$

we can compute $B_{1,3}$ in the interval $(0, 4)$ using Algorithm 2.

Chapter 4

Further properties of B-splines

In this chapter we derive further properties of B-splines and splines. In particular, we show that all piecewise polynomials can be represented as linear combinations of B-splines, and that the B-splines are linearly independent. The main tool for this is Marsden's identity. We also discuss the evaluation of derivatives.

4.1 Marsden's identity

As in the previous chapter, let $\mathbf{t} = (t_1, t_2, \ldots, t_{n+d+1})$ be a knot vector for some $d \geq 0$ and $n \geq 1$. For each $i = 1, \ldots, n$, let us define the so-called dual polynomial

$$\rho_{i,d}(y) = (y - t_{i+1})(y - t_{i+2}) \cdots (y - t_{i+d}).$$

Then Marsden's identity is as follows.

Theorem 4.1 *For any $x \in [t_{d+1}, t_{n+1}]$ and for any $y \in \mathbb{R}$,*

$$(y - x)^d = \sum_{i=1}^{n} \rho_{i,d}(y) B_{i,d}(x). \tag{4.1}$$

To prove this theorem, it is sufficient to derive the local form of the equation:

Theorem 4.2 *If x belongs to some non-empty interval $[t_\mu, t_{\mu+1}]$, for some $\mu \in \{d+1, \ldots, n\}$, then for any $y \in \mathbb{R}$,*

$$(y - x)^d = \sum_{i=\mu-d}^{\mu} \rho_{i,d}(y) B_{i,d}(x). \tag{4.2}$$

Proof. The proof uses Algorithm 2 of Section 3.9 applied to the initial data $c_i = \rho_{i,d}(y)$, $i = \mu - d, \ldots, \mu$. Consider the first step of the algorithm. With $r = 1$ in (3.25),

$$\begin{aligned} c_i^1 &= \frac{t_{i+d} - x}{t_{i+d} - t_i} c_{i-1}^0 + \frac{x - t_i}{t_{i+d} - t_i} c_i^0 \\ &= \frac{t_{i+d} - x}{t_{i+d} - t_i} \rho_{i-1,d}(y) + \frac{x - t_i}{t_{i+d} - t_i} \rho_{i,d}(y) \\ &= \left(\frac{t_{i+d} - x}{t_{i+d} - t_i}(y - t_i) + \frac{x - t_i}{t_{i+d} - t_i}(y - t_{i+d}) \right) \rho_{i,d-1}(y), \end{aligned}$$

and a simple calculation shows that

$$\frac{t_{i+d} - x}{t_{i+d} - t_i}(y - t_i) + \frac{x - t_i}{t_{i+d} - t_i}(y - t_{i+d}) = y - x. \tag{4.3}$$

This shows that

$$c_i^1 = (y - x)\rho_{i,d-1}(y).$$

In the next step of the algorithm, with $r = 2$ in (3.25), we find, similarly, that

$$c_i^2 = (y - x)^2 \rho_{i,d-2}(y).$$

Continuing in this way, we find that for all $r = 1, \ldots, d$,

$$c_i^r = (y - x)^r \rho_{i,d-r}(y). \tag{4.4}$$

The case $d = r$ gives $c_\mu^d = (y - x)^d$, which, by Theorem 3.7, proves (4.2). □

4.2 Linear independence of B-splines

We can use Marsden's identity to show that the B-splines $B_{1,d}, \ldots B_{n,d}$ are linearly independent with respect to the interval $[t_{d+1}, t_{n+1}]$. To this end, suppose that there are coefficients $c_1, \ldots, c_n$ such that

$$\sum_{i=1}^{n} c_i B_{i,d}(x) = 0, \qquad t_{d+1} \le x \le t_{n+1}. \tag{4.5}$$

The task is show that $c_1 = \cdots = c_n = 0$. From (4.5), for any non-empty $[t_\mu, t_{\mu+1}]$, $d + 1 \le \mu \le n$,

$$\sum_{i=\mu-d}^{\mu} c_i B_{i,d}(x) = 0, \qquad t_\mu \le x \le t_{\mu+1}.$$

We will have $c_{\mu-d} = \cdots = c_\mu = 0$ if the B-splines $B_{\mu-d,d}, \ldots, B_{\mu,d}$ are linearly independent. Since there are $d+1$ of these, it is sufficient to show that we can express any monomial x^r, $0 \le r \le d$, as a linear combination of them. To do this we use the local form (4.2) of Marsden's identity. First we differentiate (4.2) $d-r$ times with respect to y, giving

$$\frac{d!}{r!}(y-x)^r = \sum_{i=\mu-d}^{\mu} \rho_{i,d}^{(d-r)}(y) B_{i,d}(x),$$

and then we let $y = 0$, giving

$$\frac{d!}{r!}(-1)^r x^r = \sum_{i=\mu-d}^{\mu} \rho_{i,d}^{(d-r)}(0) B_{i,d}(x).$$

Rearranging this gives

$$x^r = \sum_{i=\mu-d}^{\mu} c_{i,r} B_{i,d}(x), \tag{4.6}$$

where

$$c_{i,r} = (-1)^r \frac{r!}{d!} \rho_{i,d}^{(d-r)}(0).$$

Thus we have shown that $B_{\mu-d,d}, \ldots, B_{\mu,d}$ are linearly independent on the knot interval $[t_\mu, t_{\mu+1}]$ and so $c_{\mu-d} = \cdots = c_\mu = 0$. By considering all μ, it follows that $c_1 = \cdots = c_n = 0$, as claimed, and so we conclude that $B_{1,d}, \ldots, B_{n,d}$ are indeed linearly independent on $[t_{d+1}, t_{n+1}]$.

4.2.1 Monomials as splines

We can obtain a more explicit formula for x^r in (4.6) by deriving a closed formula for the coefficients $c_{i,r}$. By the product rule for differentiating a product of d functions, we have

$$\rho_{i,d}^{(d-r)}(y) = (d-r)! \sum_{i+1 \le i_1 < \cdots < i_r \le i+d} (y - t_{i_1}) \cdots (y - t_{i_r}).$$

The sum is over all possible products of r of the d factors of $\rho_{i,d}(y)$. It follows that

$$\rho_{i,d}^{(d-r)}(0) = (-1)^r (d-r)! \sum_{i+1 \le i_1 < \cdots < i_r \le i+d} t_{i_1} \cdots t_{i_r},$$

and therefore,

$$c_{i,r} = \frac{1}{\binom{d}{r}} \sum_{i+1 \le i_1 < \cdots < i_r \le i+d} t_{i_1} \cdots t_{i_r}.$$

The sum here is over all possible products of r of the d interior knots $t_{i+1}, \ldots, t_{i+d}$ in the support of $B_{i,d}$. The binomial coefficient $\binom{d}{r}$ is the number of these products, and hence $c_{i,r}$ is simply the average of all these products.

From (4.6) we also have the global version,

$$x^r = \sum_{i=1}^{n} c_{i,r} B_{i,d}(x), \qquad t_{d+1} \le x \le t_{n+1}. \tag{4.7}$$

Some examples are

$$\begin{aligned} 1 &= \sum_{i=1}^{n} B_{i,d}(x), \\ x &= \sum_{i=1}^{n} t_i^* B_{i,d}(x), \\ x^2 &= \sum_{i=1}^{n} t_i^{**} B_{i,d}(x), \\ x^d &= \sum_{i=1}^{n} t_{i+1} \cdots t_{i+d} B_{i,d}(x), \end{aligned}$$

where

$$t_i^{**} = \frac{t_{i+1}t_{i+2} + t_{i+1}t_{i+3} + \cdots + t_{i+d-1}t_{i+d}}{\binom{d}{2}}.$$

The first example shows that the B-splines sum to one.

4.3 B-splines as a basis for splines

With a little more work, we can show that any spline s (a piecewise polynomial) can be represented by B-splines. Roughly speaking, it is just a matter of choosing the right knot vector: there should be a knot t_i at each break point of s and the multiplicity of t_i should reflect the order of continuity of s at t_i.

First, let's be precise about what we mean by a spline (piecewise polynomial). Let $[a, b]$ be a real interval and let $\Delta = (\xi_i)_{i=1}^N$ be a partition of $[a, b]$,

$$a = \xi_1 < \xi_2 < \cdots < \xi_N = b,$$

and let $\mathbf{r} = (r_i)_{i=2}^{N-1}$ be a sequence of integers with $r_i \geq -1$. For any $d \geq 0$ we define

$$\mathcal{S}_d^{\mathbf{r}}(\Delta) = \{s : [a,b] \to \mathbb{R} : s|_{(\xi_i,\xi_{i+1})} \in \pi_d, i = 1,\ldots,N-1, \\ s^{(k)} \text{ is continuous at } \xi_i,\ k = 0,1,\ldots,r_i,\ i = 2,\ldots,N-1\}.$$

Since any linear combination of functions in $\mathcal{S}_d^{\mathbf{r}}(\Delta)$ is also in $\mathcal{S}_d^{\mathbf{r}}(\Delta)$, we see that $\mathcal{S}_d^{\mathbf{r}}(\Delta)$ is a linear space. A basis for $\mathcal{S}_d^{\mathbf{r}}(\Delta)$ can be constructed using truncated powers.

Lemma 4.3 *Any $s \in \mathcal{S}_d^{\mathbf{r}}(\Delta)$ can be expressed in the truncated power form*

$$s(x) = \sum_{k=0}^{d} c_k x^k + \sum_{i=2}^{N-1} \sum_{k=r_i+1}^{d} c_{i,k}(x-\xi_i)_+^k. \tag{4.8}$$

Proof. Let p_i denote the polynomial in π_d representing s in (ξ_i, ξ_{i+1}), $i = 1,\ldots,N-1$. Then let

$$c_k = \frac{p_1^{(k)}(0)}{k!}, \quad k = 0,\ldots,d,$$
$$c_{i,k} = \frac{p_i^{(k)}(\xi_i) - p_{i-1}^{(k)}(\xi_i)}{k!}, \quad k = r_i+1,\ldots,d, \quad i = 2,\ldots,N-1.$$

We now claim that s is as in (4.8). Indeed, letting $x \in (\xi_1, \xi_2)$ in (4.8) implies

$$s(x) = \sum_{k=0}^{d} c_k x^k = p_1(x),$$

and letting $x \in (\xi_j, \xi_{j+1})$ in (4.8) for some $j = 2,\ldots,N-1$ gives

$$s(x) = \sum_{k=0}^{d} c_k x^k + \sum_{i=2}^{j} \sum_{k=r_i+1}^{d} c_{i,k}(x-\xi_i)^k = p_1(x) + \sum_{i=2}^{j}(p_i(x) - p_{i-1}(x)) = p_j(x),$$

since $p_i^{(k)}(\xi_i) = p_{i-1}^{(k)}(\xi_i)$ for $k = 0,1,\ldots,r$. □

Lemma 4.4 *The truncated power form of s in (4.8) is unique.*

Proof. It is sufficient to show that if

$$\tilde{s}(x) := \sum_{k=0}^{d} c_k x^k + \sum_{i=2}^{N-1} \sum_{k=r_i+1}^{d} c_{i,k}(x-\xi_i)_+^k = 0, \qquad x \in (a,b),$$

then the coefficients c_k and $c_{i,k}$ are zero. To see this, observe that for $x \in (\xi_1, \xi_2)$,

$$0 = \tilde{s}(x) = \sum_{k=0}^{d} c_k x^k,$$

which implies that $c_0 = \cdots = c_d = 0$. It then follows that for $x \in (\xi_2, \xi_3)$,

$$0 = \tilde{s}(x) = \sum_{k=r_2+1}^{d} c_{2,k}(x - \xi_2)^k,$$

which implies that $c_{2,r_2+1} = \cdots = c_{2,d} = 0$, and continuing in this way shows that all the $c_{i,k}$ in $\tilde{s}$ are zero. □

By Lemmas 4.3 and 4.4, we conclude that the functions x^k, $k = 0, \ldots, d$, and $(x - \xi_i)_+^k$, $k = r_i + 1, \ldots, d$, $i = 2, \ldots, N-1$, form a basis of $\mathcal{S}_d^{\mathbf{r}}(\Delta)$. Thus, the dimension of the space $\mathcal{S}_d^{\mathbf{r}}(\Delta)$ is

$$n := d + 1 + \sum_{i=2}^{N-1} (d - r_i). \tag{4.9}$$

We can now show that any spline in $\mathcal{S}_d^{\mathbf{r}}(\Delta)$ can be represented in terms of B-splines.

Theorem 4.5 (Curry-Schoenberg) *Let $\mathcal{S}_d^{\mathbf{r}}(\Delta)$ be a space of piecewise polynomials, let n be as in (4.9), and let $\mathbf{t} = (t_i)_{i=1}^{n+d+1}$ be any knot vector such that*

$$(t_{d+2}, \ldots, t_n) = (\underbrace{\xi_2, \ldots, \xi_2}_{d-r_2}, \underbrace{\xi_3, \ldots, \xi_3}_{d-r_3}, \ldots, \underbrace{\xi_{N-1}, \ldots, \xi_{N-1}}_{d-r_{N-1}}),$$

and such that $t_{d+1} \le a$ and $b \le t_{n+1}$. Then

$$\mathcal{S}_d^{\mathbf{r}}(\Delta) = \mathcal{S}_{d,\mathbf{t}}|_{[a,b]},$$

where $\mathcal{S}_{d,\mathbf{t}}|_{[a,b]}$ is the space obtained from $\mathcal{S}_{d,\mathbf{t}}$ by restricting its functions to $[a, b]$.

Proof. By the construction of the knot vector, the B-splines in $\mathcal{S}_{d,\mathbf{t}}$ satisfy the smoothness properties of $\mathcal{S}_d^{\mathbf{r}}(\Delta)$ and therefore $\mathcal{S}_{d,\mathbf{t}}|_{[a,b]} \subset \mathcal{S}_d^{\mathbf{r}}(\Delta)$. Moreover, since $t_{d+1} \le a < t_{d+2}$ and $t_n < b \le t_{n+1}$, the B-splines in $\mathcal{S}_{d,\mathbf{t}}$ are linearly independent in $[a, b]$, and therefore the space $\mathcal{S}_{d,\mathbf{t}}|_{[a,b]}$ has dimension n. But since n is also the dimension of $\mathcal{S}_d^{\mathbf{r}}(\Delta)$, we conclude that $\mathcal{S}_{d,\mathbf{t}}|_{[a,b]} = \mathcal{S}_d^{\mathbf{r}}(\Delta)$. □

4.4 Derivatives

If it is often important to be able to compute derivatives of B-splines and splines. The key to computing these is the following formula for the first derivative of $B_{i,d}$, which has many similarities to the recursion formula (3.15).

Theorem 4.6 *For $d \geq 1$,*

$$B'_{i,d}(x) = d\left(\frac{B_{i,d-1}(x)}{t_{i+d} - t_i} - \frac{B_{i+1,d-1}(x)}{t_{i+d+1} - t_{i+1}}\right). \tag{4.10}$$

Proof. Due to the recursion (3.17), we can express $B_{i,d}$ in (3.9) in the form

$$B_{i,d}(x) = [t_{i+1}, \ldots, t_{i+d+1}](\cdot - x)_+^d - [t_i, \ldots, t_{i+d}](\cdot - x)_+^d. \tag{4.11}$$

Differentiating this with respect to x gives

$$B'_{i,d}(x) = d(-[t_{i+1}, \ldots, t_{i+d+1}](\cdot - x)_+^{d-1} + [t_i, \ldots, t_{i+d}](\cdot - x)_+^{d-1}),$$

which, by the definition of $B_{i,d-1}$ and $B_{i+1,d-1}$, yields the result. □

We can apply this to find the first derivative of the spline

$$s(x) = \sum_{i=1}^{n} c_i B_{i,d}(x).$$

Differentiation gives

$$\begin{aligned} s'(x) &= \sum_{i=1}^{n} c_i B'_{i,d}(x) = d\sum_{i=1}^{\mu} c_i \left(\frac{B_{i,d-1}(x)}{t_{i+d} - t_i} - \frac{B_{i+1,d-1}(x)}{t_{i+d+1} - t_{i+1}}\right) \\ &= d\sum_{i=2}^{n} \left(\frac{c_i - c_{i-1}}{t_{i+d} - t_i}\right) B_{i,d-1}(x). \end{aligned}$$

Both Algorithms 1 and 2 of Section 3.9 can now easily be adapted to compute $s'(x)$ for a given x in a knot interval $[t_\mu, t_{\mu+1}]$.

Higher order derivatives are easily obtained by differentiating (4.10). For $d \geq 1$ and $r \geq 1$,

$$B^{(r)}_{i,d}(x) = d\left(\frac{B^{(r-1)}_{i,d-1}(x)}{t_{i+d} - t_i} - \frac{B^{(r-1)}_{i+1,d-1}(x)}{t_{i+d+1} - t_{i+1}}\right). \tag{4.12}$$

4.5 Exercises

4.1 Prove (4.3).

4.2 Derive the following alternative formula for the r-th derivative of the B-spline $B_{i,d}$ (where $d \geq 1$):

$$B_{i,d}^{(r)}(x) = \frac{d}{d-r}\left(\frac{x-t_i}{t_{i+d}-t_i}B_{i,d-1}^{(r)}(x) + \frac{t_{i+d+1}-x}{t_{i+d+1}-t_{i+1}}B_{i+1,d-1}^{(r)}(x)\right).$$

Hint: differentiate (3.15) r times and use (4.12).

Chapter 5

Knot insertion

As we have seen in numerial examples, the shape of a spline curve tends to mimic the shape of its control polygon. Examples also indicate that a spline curve is a kind of smoothed out version of its control polygon. This makes modelling with a spline curve easy: the designer can adjust an existing spline curve by moving the control points: the change in the curve will be reflected by the change in the polygon in an intuitive (and local) way.

Another useful aspect of spline curves in B-spline form is that there is a simple way to add degrees of freedom, i.e., add control points, to an existing spline curve without changing the curve: what we do is to add knots to the knot vector and calculate what the new control points should be so that the spline curve is unchanged. Once this refinement of the knot vector has been carried out, the designer will have more control points available to manipulate the spline curve.

In this chapter we study how to find the control points of the spline curve with respect to the refined knot vector from the previous control points. To understand this we focus on what happens to a single B-spline when we add one knot to the knot vector: we can can express the B-spline as a linear combination of two B-splines on the new knot vector.

As well as being of practical use in geometric modelling, knot insertion is also a useful tool for deriving further theoretical properties of splines. One such property is the variation diminishing property of splines in B-spline form: the number of sign changes in a spline function is bounded by the number of sign changes in its control polygon. For spline curves this translates to the number of times the spline curve crosses a straight line: it is bounded by the number of times its control polygon crosses the line. This variation dimishing property makes precise the intuitive idea that the spline curve is a smoothed out version of its control polygon.

5.1 B-spline refinement

Given a knot vector $\mathbf{t} = (t_1, t_2, \ldots, t_{n+d+1})$, for some $d \geq 0$ and $n \geq 1$, there are n B-splines of degree d, $B_{1,\mathbf{t}}, \ldots, B_{n,\mathbf{t}}$. Suppose we now add a new knot z to some non-empty knot interval $[t_\mu, t_{\mu+1}]$, where $1 \leq \mu \leq n+d$, to form the refined knot vector

$$\boldsymbol{\tau} = (\tau_1, \ldots, \tau_{n+d+2}) = (t_1, t_2, \ldots, t_\mu, z, t_{\mu+1}, \ldots, t_{n+d+1}). \tag{5.1}$$

We then ask, how can we represent the 'old' B-splines, $B_{1,\mathbf{t}}, \ldots, B_{n,\mathbf{t}}$ in terms of the 'new' B-splines, $B_{1,\boldsymbol{\tau}}, \ldots, B_{n+1,\boldsymbol{\tau}}$? Observe first that

$$B_{i,d,\mathbf{t}} = B_{i,d,\boldsymbol{\tau}}, \qquad i = 1, \ldots, \mu - d - 1, \tag{5.2}$$

since z is not among the knots that define these B-splines, and similarly,

$$B_{i,d,\mathbf{t}} = B_{i+1,d,\boldsymbol{\tau}}, \qquad i = \mu + 1, \ldots, n, \tag{5.3}$$

for the same reason. Thus, it remains to consider the B-splines $B_{i,\mathbf{t}}$ with $i = \mu - d, \ldots, \mu$. As we will see, in this case $B_{i,\mathbf{t}}$ can be expressed as a linear combination of $B_{i,\boldsymbol{\tau}}$ and $B_{i+1,\boldsymbol{\tau}}$. To this end, we will first derive a refinement formula for divided differences.

Lemma 5.1 *Consider a divided difference $[x_0, x_1, \ldots, x_k]f$ in which x_0 and x_k are distinct, and let $z \in \mathbb{R}$. Then*

$$[x_0, \ldots, x_k]f = \frac{z - x_0}{x_k - x_0}[x_0, \ldots, x_{k-1}, z]f + \frac{x_k - z}{x_k - x_0}[x_1, \ldots, x_k, z]f. \tag{5.4}$$

Proof. We have

$$\begin{aligned}
&[x_1, \ldots, x_k]f - [x_0, \ldots, x_{k-1}]f = \\
&([x_1, \ldots, x_{k-1}, z]f - [x_0, \ldots, x_{k-1}]f) + ([x_1, \ldots, x_k]f - [x_1, \ldots, x_{k-1}, z]f) \\
&= (z - x_0)[x_0, \ldots, x_{k-1}, z]f + (x_k - z)[x_1, \ldots, x_k, z]f,
\end{aligned}$$

and dividing by $x_k - x_0$ gives (5.4). □

From this lemma we obtain the refinement of the B-splines $B_{\mu-d,\mathbf{t}}, \ldots, B_{\mu,\mathbf{t}}$.

Theorem 5.2

$$\begin{aligned}
B_{\mu-d,\mathbf{t}} &= B_{\mu-d,\boldsymbol{\tau}} + \frac{t_{\mu+1} - z}{t_{\mu+1} - t_{\mu-d+1}} B_{\mu-d+1,\boldsymbol{\tau}}, \\
B_{i,\mathbf{t}} &= \frac{z - t_i}{t_{i+d} - t_i} B_{i,\boldsymbol{\tau}} + \frac{t_{i+d+1} - z}{t_{i+d+1} - t_{i+1}} B_{i+1,\boldsymbol{\tau}}, \qquad i = \mu - d + 1, \ldots, \mu - 1, \\
B_{\mu,\mathbf{t}} &= \frac{z - t_\mu}{t_{\mu+d} - t_\mu} B_{\mu,\boldsymbol{\tau}} + B_{\mu+1,\boldsymbol{\tau}}.
\end{aligned}$$

Proof. Let $i \in \{\mu - d, \ldots, \mu\}$. By Lemma 5.1, for any smooth enough function f,

$$(t_{i+d+1} - t_i)[t_i, \ldots, t_{i+d+1}]f = (z - t_i)[t_i, \ldots, t_{i+d}, z]f \\ + (t_{i+d+1} - z)[t_{i+1}, \ldots, t_{i+d+1}, z]f.$$

We now apply this equation to the function $f(t) = (t - x)_+^d$. Then the left hand side becomes $B_{i,\mathbf{t}}(x)$, and on the right hand side,

$$[t_i, \ldots, t_{i+d}, z]f = \begin{cases} \dfrac{B_{i,\boldsymbol{\tau}}(x)}{z - t_i}, & i = \mu - d; \\ \dfrac{B_{i,\boldsymbol{\tau}}(x)}{t_{i+d} - t_i}, & i > \mu - d, \end{cases}$$

and

$$[t_{i+1}, \ldots, t_{i+d+1}, z]f = \begin{cases} \dfrac{B_{i+1,\boldsymbol{\tau}}(x)}{t_{i+d+1} - t_{i+1}}, & i < \mu; \\ \dfrac{B_{i+1,\boldsymbol{\tau}}(x)}{t_{i+d} - z}, & i = \mu. \end{cases}$$

□

5.2 Spline refinement

Suppose now that s is a spline function

$$s(x) = \sum_{i=1}^{n} c_i B_{i,\mathbf{t}}(x), \qquad x \in I, \tag{5.5}$$

for some coefficients $c_1, \ldots, c_n \in \mathbb{R}$, where $I = [t_{d+1}, t_{n+1}]$. If we add a knot z to I to form the refined knot vector $\boldsymbol{\tau}$ in (5.1), we can represent s with respect to $\boldsymbol{\tau}$ due to Theorem 5.2. In other words, we can find coefficients $b_1, \ldots, b_{n+1}$ such that

$$s(x) = \sum_{i=1}^{n+1} b_i B_{i,\boldsymbol{\tau}}(x), \qquad x \in I. \tag{5.6}$$

The algorithm for computing the b_i is known as Boehm's algorithm.

Theorem 5.3 *The coefficients b_i in (5.6) are*

$$b_i = \begin{cases} c_i & \text{if } 1 \le i \le \mu - d; \\ \dfrac{t_{i+d} - z}{t_{i+d} - t_i} c_{i-1} + \dfrac{z - t_i}{t_{i+d} - t_i} c_i; & \text{if } \mu - d + 1 \le i \le \mu, \\ c_{i-1} & \text{if } \mu + 1 \le i \le n + 1. \end{cases} \tag{5.7}$$

Proof. We convert the sum in (5.5) into the form (5.6) by expressing each B-spline $B_{i,\mathbf{t}}$ as a linear combination of the B-splines $B_{i,\boldsymbol{\tau}}$. Consider the terms in (5.5) for $i = \mu - d, \ldots, \mu$. By Theorem 5.2,

$$\begin{aligned}\sum_{i=\mu-d}^{\mu} c_i B_{i,\mathbf{t}} &= c_{\mu-d}\left(B_{\mu-d,\boldsymbol{\tau}} + \frac{t_{\mu+1}-z}{t_{\mu+1}-t_{\mu-d+1}} B_{\mu-d+1,\boldsymbol{\tau}}\right) \\ &\quad + \sum_{i=\mu-d+1}^{\mu-1} c_i\left(\frac{z-t_i}{t_{i+d}-t_i} B_{i,\boldsymbol{\tau}} + \frac{t_{i+d+1}-z}{t_{i+d+1}-t_{i+1}} B_{i+1,\boldsymbol{\tau}}\right) \\ &\quad + c_\mu\left(\frac{z-t_\mu}{t_{\mu+d}-t_\mu} B_{\mu,\boldsymbol{\tau}} + B_{\mu+1,\boldsymbol{\tau}}\right) \\ &= \sum_{i=\mu-d}^{\mu+1} b_i B_{i,\boldsymbol{\tau}},\end{aligned}$$

with b_i defined in (5.7). Combining this with equations (5.2) and (5.3) gives the result. □

5.3 Variation diminishing property

In this section we use Boehm's algorithm to show that the number of sign changes in a spline function is bounded by the number of sign changes in its B-spline coefficient vector. Let us start by defining what we mean by sign changes.

Definition 5.4 *Let $\mathbf{c} = (c_i)_i^n$ be a vector of real numbers. The number of sign changes of $\mathbf{c}$ (zeros ignored) is denoted by $S^-(\mathbf{c})$. The number of sign changes of a function f on an interval I is denoted by $S_I^-(f)$, provided this number is finite. It is the largest integer r such that there is an increasing sequence of real numbers $x_1 < x_2 < \cdots < x_{r+1}$ in I such that $S^-(f(x_1), f(x_2), \ldots, f(x_{r+1})) = r$.*

For example,

$$\begin{aligned}S^-(1,-2) &= 1,\\ S^-(1,0,-1,3,2) &= 2,\\ S^-(1,-1,3,2) &= 2.\end{aligned}$$

We start by showing that the number of sign changes in the coefficients cannot increase when we insert one knot.

Lemma 5.5 *Let*

$$s(x) = \sum_{i=1}^{n} c_i B_{i,\mathbf{t}}(x) = \sum_{i=1}^{n+1} b_i B_{i,\boldsymbol{\tau}}(x), \qquad x \in I,$$

where $\boldsymbol{\tau}$ *is the knot vector formed from the knot vector* $\mathbf{t}$ *by adding one knot to* $\mathbf{t}$ *in* I*. Then* $\mathbf{b}$ *has no more sign changes than* $\mathbf{c}$*, i.e.,* $S^-(\mathbf{b}) \leq S^-(\mathbf{c})$*.*

Proof. By Boehm's algorithm, we can express the coefficients b_i in the form

$$b_i = (1 - \alpha_i)c_{i-1} + \alpha_i c_i, \qquad i = 1, 2, \ldots, n+1,$$

where $0 \leq \alpha_i \leq 1$, and where we have defined for convenience $c_0 = c_{n+1} = 0$. Since $1 - \alpha_i$ and α_i are both non-negative it follows that b_i is either zero or has the same sign as either c_{i-1} or c_i. Therefore,

$$S^-(\mathbf{c}) = S^-(b_1, c_1, b_2, c_2, b_3, \ldots, b_n, c_n, b_{n+1}) \geq S^-(\mathbf{b}).$$

The inequality on the right follows from the fact that when we remove an element from a vector, the number of sign changes cannot increase. □

By adding knots one by one we immediately obtain a corollary.

Corollary 5.6 *Let*

$$s(x) = \sum_{i=1}^{n} c_i B_{i,d,\mathbf{t}}(x) = \sum_{i=1}^{m} b_i B_{i,d,\boldsymbol{\tau}}(x), \qquad x \in I,$$

where $m > n$ *and* $\boldsymbol{\tau}$ *is the knot vector formed from the knot vector* $\mathbf{t}$ *by adding* $m - n$ *knots to* $\mathbf{t}$ *(so* $\mathbf{t} \subset \boldsymbol{\tau}$*) in* I*. Then* $\mathbf{b}$ *has no more sign changes than* $\mathbf{c}$*, i.e.,* $S^-(\mathbf{b}) \leq S^-(\mathbf{c})$*.*

With this corollary we can derive the variation diminishing property of spline functions.

Theorem 5.7 (Variation Diminishing Property) *Let*

$$s(x) = \sum_{i=1}^{n} c_i B_{i,\mathbf{t}}(x), \qquad x \in I,$$

be a spline. Then $S_I^-(s) \leq S^-(\mathbf{c})$*.*

Proof. Let $r = S_I^-(s)$ and let $(x_i)_{i=1}^{r+1}$ be an increasing sequence of points in I chosen such that

$$S^-(s(x_1), s(x_2), \ldots, s(x_{r+1})) = r.$$

Then let $\boldsymbol{\tau}$ be the knot vector formed from $\mathbf{t}$ by adding to $\mathbf{t}$ each point x_i, $i = 1, \ldots, r+1$, with multiplicity d (if any point x_i is already a knot in $\mathbf{t}$, we just add it $d - m_i$ times to $\mathbf{t}$, where m_i is is the multiplicity of x_i in $\mathbf{t}$). At each point x_i, only one B-spline $B_{j,\boldsymbol{\tau}}$ has the value 1, all others have value 0, and therefore $s(x_i) = b_j$. It follows that $S^-(\mathbf{b}) \geq r$, and hence, by Corollary 5.6,

$$S_I^-(s) = r \leq S^-(\mathbf{b}) \leq S^-(\mathbf{c}).$$

□

For spline curves in $\mathbb{R}^2$ the variation diminishing property translates into a property concerning the number of crossings of the curve with a straight line.

Theorem 5.8 (VDP for planar spline curves) *Let*

$$\mathbf{s}(x) = \sum_{i=1}^{n} \mathbf{c}_i B_{i,\mathbf{t}}(x), \qquad x \in I,$$

be a spline curve in $\mathbb{R}^2$, with $\mathbf{c}_i \in \mathbb{R}^2$, and let L be any straight line in $\mathbb{R}^2$. Then the number of times $\mathbf{s}$ crosses L is less than or equal to the number of times its control polygon crosses L.

Proof. Suppose the straight line is

$$L = \{\mathbf{p} \in \mathbb{R}^2 : \mathbf{p} \cdot \mathbf{n} - a = 0\},$$

for some unit vector $\mathbf{n} \in \mathbb{R}^2$ and constant $a \in \mathbb{R}$, and let

$$\sigma(x) = \mathbf{s}(x) \cdot \mathbf{n} - a, \qquad x \in I.$$

Then $S_I^-(\sigma)$ is the number of times $\mathbf{s}$ crosses L. Moreover, since $\sum_{i=1}^n B_{i,d}(x) = 1$,

$$\sigma(x) = \sum_{i=1}^{n} \gamma_i B_{i,\mathbf{t}}(x),$$

where $\gamma_i = \mathbf{c}_i \cdot \mathbf{n} - a$, and $S^-(\boldsymbol{\gamma})$ is the number of times the control polygon of $\mathbf{s}$ crosses L, where $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_n)$.

Since σ is a spline function with coefficients γ_i, the VDP of σ (Theorem 5.7) implies $S_I^-(\sigma) \leq S^-(\boldsymbol{\gamma})$. □

5.4 Exercises

5.1 Show that a spline curve $\mathbf{s}$ in $\mathbb{R}^3$ satisfies a variation diminishing property analogous to Theorem 5.8: the number of crossings of $\mathbf{s}$ through a plane P is not greater than the number of crossings of the control polygon of $\mathbf{s}$ through P.

Chapter 6

B-spline interpolation and approximation

6.1 Spline interpolation in B-spline form

We now return to the problem of interpolating data with a spline that we discussed in Chapter 2, but we can now use the alternative approach of representing the spline in B-spline form.

6.1.1 Linear spline interpolation

Consider again the example of contructing a piecewise linear spline to match given data. Suppose that $x_1, x_2, \ldots, x_m \in \mathbb{R}$ is an increasing sequence of points and that $y_1, y_2, \ldots, y_m \in \mathbb{R}$ are associated data values. We let $a = x_1$ and $b = x_m$ and we would like to find a spline $g : [a, b] \to \mathbb{R}$ such that in each interval $[x_i, x_{i+1}]$, $i = 1, 2, \ldots, m-1$, g is a linear polynomial, and such that

$$g(x_i) = y_i, \qquad i = 1, 2, \ldots, m. \tag{6.1}$$

In Chapter 2 we constructed g piecewise (in equation (2.10)). If instead we represent g using B-splines, the C^0 continuity of g is already guaranteed by the continuity of the B-splines. We set $d = 1$ and $n = m$, and we can choose $\mathbf{t}$ to be the closed knot vector

$$\mathbf{t} = (t_1, \ldots, t_{m+2}) = (x_1, x_1, x_2, x_3, \ldots, x_{m-2}, x_{m-1}, x_m, x_m).$$

Then

$$g(x) = \sum_{i=1}^{m+2} c_i B_{i,1,\mathbf{t}}(x), \qquad x \in [a, b],$$

where $c_i = y_i$, $i = 1, \ldots, n$.

6.1.2 Cubic Hermite spline interpolation

The cubic Hermite spline we constructed in Theorem 2.6 can also be represented using B-splines. We recall that this spline is the C^1 function $g : [a, b] \to \mathbb{R}$ whose pieces are cubic and such that

$$g(x_i) = y_i \quad \text{and} \quad g'(x_i) = s_i, \qquad i = 1, \ldots, m, \tag{6.2}$$

for given height values y_i and slope values s_i. To represent g in B-spline form, we must set $d = 3$ and $n = 2m$. As regards the knot vector $\mathbf{t}$, we can reflect the C^1 continuity of g in $\mathbf{t}$ by giving the interior knots multiplicity 2. Thus, choosing $\mathbf{t}$ to be closed, we obtain

$$\mathbf{t} = (t_1, \ldots, t_{2m+4}) = (x_1^{[4]}, x_2^{[2]}, x_3^{[2]}, \ldots, x_{m-1}^{[2]}, x_m^{[4]}),$$

where we have used the shorthand $x^{[k]} = \underbrace{x, \ldots, x}_{k}$.

Theorem 6.1 *The unique C^1 cubic spline satisfying the Hermite interpolation conditions (6.2) can be represented as*

$$g(x) = \sum_{i=1}^{2m} c_i B_{i,3,\mathbf{t}}(x), \qquad x \in [a, b], \tag{6.3}$$

where $c_1 = y_1$, $c_{2m} = y_m$, and

$$c_{2i} = y_i + \frac{h_i}{3} s_i, \quad c_{2i+1} = y_{i+1} - \frac{h_i}{3} s_{i+1}, \qquad i = 1, \ldots, m-1,$$

and $h_i = x_{i+1} - x_i$.

Proof. Consider g in (6.3). The $m - 2$ interior points $x_2, \ldots, x_{m-1}$ are knots in $\mathbf{t}$ of multiplicity 2. If we now insert these same $m - 2$ points into $\mathbf{t}$, we obtain a refined knot vector $\boldsymbol{\tau}$ in which each of these knots has multiplicity 3:

$$\boldsymbol{\tau} = (\tau_1, \ldots, \tau_{3m+2}) = (x_1^{[4]}, x_2^{[3]}, x_3^{[3]}, \ldots, , x_{m-1}^{[3]}, x_m^{[4]}).$$

By applying Boehm's algorithm repeatedly we obtain a new representation of g,

$$g(x) = \sum_{i=1}^{3m-2} b_i B_{i,3,\boldsymbol{\tau}}(x), \qquad x \in [a, b],$$

where $b_1 = c_1$, $b_{3m-2} = c_{2m}$, and

$$b_{3i-1} = c_{2i}, \quad b_{3i} = c_{2i+1}, \qquad i = 1, \ldots, m-1,$$

and

$$b_{3i-2} = \frac{x_{i+1} - x_i}{x_{i+1} - x_{i-1}} c_{2i-1} + \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}} c_{2i}, \qquad i = 2, \ldots, m-1.$$

The latter coefficients simplify:

$$b_{3i-2} = \frac{h_i}{h_{i-1} + h_i} \left(y_i - \frac{h_{i-1}}{3} s_i \right) + \frac{h_{i-1}}{h_{i-1} + h_i} \left(y_i + \frac{h_i}{3} s_i \right) = y_i.$$

Since the interior knots of $\boldsymbol{\tau}$ have multiplicity 3, and recalling Section 3.5.4, we see that this representaion of g is piecewise cubic BB form, and thus by checking its coefficients one can see that it is the same function as in Theorem 2.6. □

6.1.3 C^2 cubic spline interpolation

Consider now the C^2 cubic spline g that matches the values y_i at x_i and that satisfies the two Hermite end conditions,

$$g'(x_1) = s_1 \quad \text{and} \quad g'(x_m) = s_m. \tag{6.4}$$

We showed in Theorem 2.8 that g is uniquely determined and one way of computing g is to make the first derivatives of g at the interior points $x_2, \ldots, x_{m-1}$ be unknowns and to solve a linear system of $m-2$ equations to find these. We can instead represent g using cubic B-splines and solve for the coefficients, as follows. We let $d = 3$ and $n = m + 2$, and we can use the closed knot vector

$$\mathbf{t} = (t_1, \ldots, t_{m+6}) = (x_1^{[4]}, x_2, x_3, \ldots, x_{m-1}, x_m^{[4]}),$$

The simple interior knots reflect the C^2 continuity of g. We can now represent g in the form

$$g(x) = \sum_{i=1}^{m+2} c_i B_{i,3,\mathbf{t}}(x), \qquad x \in [a, b],$$

and it remains to compute the coefficients. To do this we simply use the $m + 2$ interpolation conditions to create $m + 2$ linear equations. We require

$$\sum_{j=1}^{m+2} c_j B_{j,3}(x_i) = y_i, \qquad i = 1, \ldots, m,$$

and

$$\sum_{j=1}^{m+2} c_j B'_{j,3}(x_i) = s_i, \qquad i = 1, m.$$

Since at most three cubic B-splines are non-zero at a knot and the same is true of their first derivatives, each equation involves at most three of the coefficients c_i. In fact, these equations form a tridiagonal system if we order the coefficients as $\mathbf{c} = [c_1, c_2, \ldots, c_{m+2}]^T$ and we order the data as

$$\mathbf{y} = [y_1, s_1, y_2, \ldots, y_{m-1}, y_m, s_m]^T.$$

The equation system is then $A\mathbf{c} = \mathbf{y}$, where A is the tridiagonal matrix

$$A = \begin{bmatrix} \alpha_1 & \gamma_1 & & & \\ \beta_2 & \alpha_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m+1} & \alpha_{m+1} & \gamma_{m+1} \\ & & & \beta_{m+2} & \alpha_{m+2} \end{bmatrix},$$

with

$$\begin{aligned} &\alpha_1 = 1, \quad \gamma_1 = 0, \\ &\beta_2 = B'_{1,3}(x_1), \quad \alpha_2 = B'_{2,3}(x_1), \quad \gamma_2 = 0, \\ &\beta_i = B_{i-1,3}(x_{i-1}), \quad \alpha_i = B_{i,3}(x_{i-1}), \quad \gamma_i = B_{i+1,3}(x_{i-1}), \qquad i = 3, \ldots, m, \\ &\beta_{m+1} = 0, \quad \alpha_{m+1} = 0, \quad \gamma_{m+1} = 1, \\ &\beta_{m+2} = 0, \quad \alpha_{m+2} = B'_{m+1,3}(x_m), \quad \gamma_{m+2} = B'_{m+2,3}(x_m). \end{aligned}$$

We know that the matrix A is non-singular because we have already shown that the spline g is unique.

If instead of the Hermite end conditions (6.4), we use the natural end conditions

$$g''(x_1) = 0 \quad \text{and} \quad g''(x_m) = 0, \tag{6.5}$$

then we can use the same knot vector and just two of the equations change. We know that these equations also have a unique solution because we have already show that the natural spline is unique.

A third alternative to ensuring there is a unique C^2 cubic spline interpolant to the data (x_i, y_i), $i = 1, \ldots, m$, is to use the so-called *free end condition*, or *not-a-knot condition*. In this approach, which is valid as long as $m \geq 4$, we force the spline g to be C^3 at the two points x_2 and x_{m-1}. This is equivalent to forcing g to be a single

cubic polynomial in the intervals $[x_1, x_3]$ and $[x_{m-2}, x_m]$. To do this we set $n = m$, and we let $\mathbf{t}$ be the knot vector

$$\mathbf{t} = (t_1, \ldots, t_{m+4}) = (x_1^{[4]}, x_3, \ldots, x_{m-2}, x_m^{[4]}),$$

and we look for a spline solution of the form

$$g(x) = \sum_{i=1}^{m} c_i B_{i,3,\mathbf{t}}(x), \qquad x \in [a, b].$$

We now simply obtain the $m \times m$ linear system $A\mathbf{c} = \mathbf{y}$ where $\mathbf{c} = [c_1, c_2, \ldots, c_m]^T$, $\mathbf{y} = [y_1, y_2, \ldots, y_m]^T$, and

$$A = [B_{j,3,\mathbf{t}}(x_i)]_{i,j=1,\ldots,m}.$$

We will show in the next section that this problem also has a unique solution by viewing it as a special case of more general spline interpolation, in which the degree is arbitrary and the so-called Schoenberg-Whitney conditions hold.

6.2 General spline interpolation

Let us now consider the more general problem of interpolation using splines of any degree. Suppose we are given the data (x_i, y_i), $i = 1, 2, \ldots, n$, where the x_i are increasing. To construct a spline g that passes through the data, we could consider choosing any degree d, and choose some knot vector $\mathbf{t} = (t_1, t_2, \ldots, t_{n+d+1})$, which defines n B-splines, the same number of B-splines as data. Then we can ask whether there exist unique coefficients $c_1, \ldots, c_n \in \mathbb{R}$ such that the spline

$$g(x) = \sum_{i=1}^{n} c_i B_{i,d,\mathbf{t}}(x)$$

interpolates the data, i.e., such that $g(x_i) = y_i$, $i = 1, \ldots, n$. Writing $B_i = B_{i,d,\mathbf{t}}$, this is equivalent to asking whether the matrix

$$A = [B_j(x_i)]_{i,j=1,\ldots,n} = \begin{bmatrix} B_1(x_1) & \cdots & B_n(x_1) \\ \vdots & & \vdots \\ B_1(x_n) & \cdots & B_n(x_n) \end{bmatrix}$$

is non-singular. It turns out that the answer is very simple: there is a simple set of conditions, known as the Schoenberg-Whitney conditions, that guarantee a unique solution.

Theorem 6.2 (Schoenberg-Whitney) *The matrix A is non-singular if and only if $B_i(x_i) > 0$ for all $i = 1, \ldots, n$.*

The conditions $B_i(x_i) > 0$ are of course equivalent to the single condition that the diagonal of A is positive. In the case that the maximum multiplicity of the knots in $\mathbf{t}$ is d, the conditions are also equivalent to

$$t_i < x_i < t_{i+d+1}, \qquad i = 1, \ldots, n.$$

In the case that the knot vector is closed, the condition that $B_1(x_1) > 0$ is equivalent to $t_1 \leq x_1 < t_{d+2}$, and the condition $B_n(x_n) > 0$ is equivalent to $t_n < x_n \leq t_{n+d+1}$. This allows the case that $t_1 = x_1$ and $t_{d+n+1} = x_n$. Therefore, the theorem can be applied to show that, for example, C^2 cubic spline interpolation using a closed knot vector and free end conditions, as in the previous section, is uniquely solvable.

We will give a proof of the Schoenberg-Whitney theorem by proving a more general result, as follows. It turns out that the matrix A is also *totally positive*, meaning that all its minors are non-negative (a minor of A is the determinant of a square submatrix of A). Not only that, a minor of A is positive if and only if its diagonal is positive. In summary, this more general property of B-splines can be stated as a follows.

Theorem 6.3 *Let $\mathbf{t} = (t_1, t_2, \ldots, t_{n+d+1})$ be a knot vector, and let $m \leq n$. For any increasing sequence of points $x_1, x_2, \ldots, x_m$ and any increasing sequence of indices $\mathbf{j} = (j_1, j_2, \ldots, j_m)$ with $1 \leq j_1 < j_m \leq n$, the matrix*

$$A(\mathbf{j}) = [B_{j_k}(x_i)]_{i,k=1,\ldots,m}$$

has non-negative determinant. The determinant is positive if and only if $B_{j_i}(x_i) > 0$ for all $i = 1, \ldots, m$.

The case $m = n$ of Theorem 6.3 implies the Schoenberg-Whitney theorem.

To prove Theorem 6.3, we will work directly with the determinant of $A(\mathbf{j})$, and apply knot insertion, and use the fact that the determinant is a linear function of its columns. Let us recall how the B-splines $B_1, \ldots, B_n$ are refined by adding a new knot z to some non-empty knot interval $[t_\mu, t_{\mu+1}]$. The refined knot vector is

$$\boldsymbol{\tau} = (t_1, t_2, \ldots, t_\mu, z, t_{\mu+1}, \ldots, t_{n+d+1}),$$

and, as observed in Chapter 5, if $\tilde{B}_1, \ldots, \tilde{B}_{n+1}$ are the B-splines on $\boldsymbol{\tau}$, then

$$B_j = \alpha_j \tilde{B}_j + \beta_j \tilde{B}_{j+1}, \qquad j = 1, \ldots, n, \tag{6.6}$$

where $\alpha_j, \beta_j \geq 0$ and

$$\begin{aligned} \alpha_j &> 0 \quad \text{if } t_j < z, \\ \beta_j &> 0 \quad \text{if } t_{j+d+1} > z. \end{aligned}$$

Proof of Theorem 6.3. Suppose first that for some p, $B_{j_p}(x_p) = 0$. Then there are two possible cases, either $x_p \leq t_{j_p}$ or $x_p \geq t_{j_p+d+1}$. If $x_p \leq t_{j_p}$ then $B_{j_k}(x_i) = 0$ for all $i \leq p$ and $k \geq p$. Then the first p rows of $A(\mathbf{j})$ are linearly dependent and $A(\mathbf{j})$ is singular. If on the other hand $x_p \geq t_{j_p+d+1}$ then $B_{j_k}(x_i) = 0$ for all $i \geq p$ and $k \leq p$. Then the first p columns of $A(\mathbf{j})$ are linearly dependent and $A(\mathbf{j})$ is again singular.

It remains to consider the case that $B_{j_i}(x_i) > 0$ for all $i = 1, \ldots, m$. Suppose that there are at least $d+1$ knots in $\mathbf{t}$ between each consecutive pair of interpolation points x_i and x_{i+1}. Since x_i belongs to $[t_{j_i}, t_{j_i+d+1}]$, any other point x_p, $p \neq i$, cannot belong to $[t_{j_i}, t_{j_i+d+1}]$. Thus $B_{j_i}(x_p) = 0$ if $p \neq i$, and $A(\mathbf{j})$ is a diagonal matrix and since its diagonal elements are positive, it is non-singular with positive determinant.

Otherwise, we use induction on the number of knots between pairs of points x_i and x_{i+1}. Suppose that between some pair of points x_p and x_{p+1}, there are less than $d+1$ knots of $\mathbf{t}$. We now form a new knot vector $\boldsymbol{\tau}$ by adding a new knot z to $\mathbf{t}$ between x_p and x_{p+1}. Then, recalling (6.6), we can replace each element $B_{j_k}(x_i)$ of $A(\mathbf{j})$ by

$$\alpha_{j_k} \tilde{B}_{j_k}(x_i) + \beta_{j_k} \tilde{B}_{j_k+1}(x_i).$$

Then, by the linearity of the determinant of $A(\mathbf{j})$ with respect to its columns,

$$\det A(\mathbf{j}) = \sum_{\boldsymbol{\epsilon} \in \{0,1\}^m} \gamma_{\boldsymbol{\epsilon}} \det \tilde{A}(\mathbf{j} + \boldsymbol{\epsilon}), \tag{6.7}$$

where $\boldsymbol{\epsilon} = (\epsilon_1, \epsilon_2, \ldots, \epsilon_m)$, and

$$\gamma_{\boldsymbol{\epsilon}} = \prod_{k=1}^{m} \left((1 - \epsilon_k) \alpha_{j_k} + \epsilon_k \beta_{j_k} \right) \geq 0,$$

and

$$\tilde{A}(\mathbf{j} + \boldsymbol{\epsilon}) = [\tilde{B}_{j_k+\epsilon_k}(x_i)]_{i,k=1,\ldots,m}.$$

Any $\mathbf{j} + \boldsymbol{\epsilon}$ in (6.7) is a non-decreasing subsequence of $(1, 2, \ldots, n+1)$. If any two consecutive elements of $\mathbf{j} + \boldsymbol{\epsilon}$ are equal then two of the columns of $\tilde{A}(\mathbf{j} + \boldsymbol{\epsilon})$ are equal and $\det \tilde{A}(\mathbf{j} + \boldsymbol{\epsilon}) = 0$. Therefore, we can remove such $\boldsymbol{\epsilon}$ from the sum in (6.7), and we have

$$\det A(\mathbf{j}) = \sum_{\substack{\boldsymbol{\epsilon} \in \{0,1\}^m \\ \mathbf{j}+\boldsymbol{\epsilon} \text{ increasing}}} \gamma_{\boldsymbol{\epsilon}} \det \tilde{A}(\mathbf{j} + \boldsymbol{\epsilon}). \tag{6.8}$$

By the induction hypothesis, all the determinants in the sum in (6.8) are non-negative, and this implies by induction that $\det A(\mathbf{j}) \geq 0$. To show that $\det A(\mathbf{j}) > 0$ we must show that there is at least one $\boldsymbol{\epsilon} \in \{0,1\}^m$ for which $\mathbf{j} + \boldsymbol{\epsilon}$ is increasing, $\gamma_{\boldsymbol{\epsilon}} > 0$, and $\det \tilde{A}(\mathbf{j} + \boldsymbol{\epsilon}) > 0$. By the induction hypothesis, $\det \tilde{A}(\mathbf{j} + \boldsymbol{\epsilon}) > 0$ if the diagonal of $\tilde{A}(\mathbf{j} + \boldsymbol{\epsilon})$ is positive. It turns out that this is true for $\boldsymbol{\epsilon}^* = (\epsilon_1^*, \ldots, \epsilon_m^*)$, where

$$\epsilon_1^* = \cdots = \epsilon_p^* = 0, \qquad \epsilon_{p+1}^* = \cdots = \epsilon_m^* = 1.$$

To see this observe that $\mathbf{j} + \boldsymbol{\epsilon}^*$ is clearly increasing, and it is sufficient to show that

$$\alpha_{j_k} > 0, \quad \tilde{B}_{j_k}(x_k) > 0, \qquad k = 1, \ldots, p, \tag{6.9}$$

$$\beta_{j_k} > 0, \quad \tilde{B}_{j_k+1}(x_k) > 0, \qquad k = p+1, \ldots, m. \tag{6.10}$$

Consider first (6.9), and let $k \in \{1, \ldots, p\}$. Since $t_{j_k} < x_k \leq x_p < z$, it follows that $\alpha_{j_k} > 0$. Further, let $\boldsymbol{\tau}$ be the knot vector obtained from inserting z in $\mathbf{t}$. Then $\tau_{j_k} = t_{j_k}$ and

$$\tau_{j_k+d+1} = \begin{cases} t_{j_k+d+1} & \text{if } z \geq t_k + d + 1, \\ \max(z, t_{j_k+d}) & \text{otherwise.} \end{cases}$$

Therefore, $\tau_{j_k} < x_k < \tau_{j_k+d+1}$, and so $\tilde{B}_{j_k}(x_k) > 0$, which establishes (6.9). To show that (6.10) also holds, let $k \in \{p+1, \ldots, m\}$. Since $t_{j_k+d+1} > x_k \geq x_{p+1} > z$ it follows that $\beta_{j_k} > 0$. Further, $\tau_{j_k+d+2} = t_{j_k+d+1}$ and

$$\tau_{j_k+1} = \begin{cases} t_{j_k} & \text{if } z \leq t_k, \\ \min(z, t_{j_k+1}) & \text{otherwise.} \end{cases}$$

Therefore, $\tau_{j_k+1} < x_k < \tau_{j_k+d+2}$, and so $\tilde{B}_{j_k+1}(x_k) > 0$. $\square$

6.3 Least squares approximation

While interpolation is suitable when the data set is small and relatively free of noise, a better solution otherwise is to make some kind of approximation. One of the easiest kinds of approximation is least squares approximation since it leads to a linear system of equations. So now let us suppose we have data (x_i, y_i), $i = 1, 2, \ldots, m$, where m could be large, and we would like to make an approxiamtion using a spline

$$g(x) = \sum_{i=1}^{n} c_j B_{j,d,\mathbf{t}}(x),$$

with $n \leq m$, using some suitable degree d and knot vector $\mathbf{t} = (t_1, t_2, \ldots, t_{n+d+1})$. Thus we want to find coefficients $c_1, \ldots, c_n$ so that $g(x_i) \approx y_i$, $i = 1, \ldots, m$. The

least squares solution to this is to minimize the sum of squares,

$$E = \sum_{i=1}^{m} (g(x_i) - y_i)^2, \tag{6.11}$$

which expands to

$$E = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} c_j B_j(x_i) - y_i \right)^2.$$

We can regard E as a function $E : \mathbb{R}^n \to \mathbb{R}$ in the unknowns $c_1, \ldots, c_n$. It is quadratic and at a minimum, all its partial derivatives are zero. The partial derivative with respect to the variable c_k is

$$\frac{\partial E}{\partial c_k} = 2 \sum_{i=1}^{m} B_k(x_i) \left(\sum_{j=1}^{n} c_j B_j(x_i) - y_i \right), \tag{6.12}$$

and so all the partial derivatives of E are zero if and only if

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} B_k(x_i) B_j(x_i) \right) c_j = \sum_{i=1}^{m} B_k(x_i) y_i, \qquad k = 1, \ldots, n. \tag{6.13}$$

This is a linear system of equations in the unknowns $c_1, \ldots, c_n$, often referred to as the *normal equations*. We can express this system in vector and matrix notation by defining A to be the (rectangular) matrix in $\mathbb{R}^{m,n}$,

$$A = [B_j(x_i)]_{i=1,\ldots,m, j=1,\ldots,n} = \begin{bmatrix} B_1(x_1) & \cdots & B_n(x_1) \\ \vdots & & \vdots \\ B_1(x_m) & \cdots & B_n(x_m) \end{bmatrix},$$

and letting $\mathbf{c} = [c_1, \ldots, c_n]^T \in \mathbb{R}^n$ and $\mathbf{y} = [y_1, \ldots, y_m]^T \in \mathbb{R}^m$. Then the system becomes

$$(A^T A)\mathbf{c} = A^T \mathbf{y}. \tag{6.14}$$

Thus there is a unique solution $\mathbf{c}$ if the $n \times n$ square matrix $A^T A$ is non-singular. When is the matrix $A^T A$ non-singular? Suppose that $A^T A\tilde{\mathbf{c}} = 0$ for some $\tilde{\mathbf{c}} \in \mathbb{R}^n$. Then $\tilde{\mathbf{c}}^T A^T A\tilde{\mathbf{c}} = 0$ and therefore $\|A\tilde{\mathbf{c}}\|^2 = 0$ and so $A\tilde{\mathbf{c}} = 0$. It follows that if A has full rank n, then $\tilde{\mathbf{c}} = 0$. Thus $A^T A$ is non-singular if A has full rank n.

Next, differentiating (6.12) with respect to c_l gives

$$\frac{\partial^2 E}{\partial c_k \partial c_l} = 2 \sum_{i=1}^{m} B_k(x_i) B_l(x_i),$$

and so the Hessian matrix of E is $2A^TA$. Therefore, if A^TA is positive definite, E is a strictly convex function (it is strictly convex along every straight line in $\mathbb{R}^n$), in which case the solution $\mathbf{c}$ to the normal equations is the unique minimum.

By the Schoenberg-Whitney conditions, A has full rank n if there exists a subsequence $(x_{k_1}, x_{k_2}, \ldots x_{k_n})$ of the points $(x_1, x_2, \ldots, x_m)$ such that $B_i(x_{k_i}) > 0$, $i = 1, \ldots, n$. We conclude as follows.

Theorem 6.4 *If there exists a subsequence $(x_{k_1}, x_{k_2}, \ldots x_{k_n})$ of $(x_1, x_2, \ldots, x_m)$ such that*

$$B_i(x_{k_i}) > 0, \quad i = 1, \ldots, n,$$

then there is a unique solution $\mathbf{c}$ to the minimization of E in (6.11) and it is the solution to the linear system (6.14).

6.4 Variation-Dimishing Spline Approximation

Schoenberg introduced the following spline approximation. Let $f : [a, b] \to \mathbb{R}$ be continuous, let $d \geq 1$, $n \geq 1$, and let $\mathbf{t} = (t_1, \ldots, t_{n+d+1})$ be a knot vector such that $t_{d+1} = a$ and $t_{n+1} = b$ and such that all knots $t_{d+2}, \ldots, t_n$ are simple. Then the spline

$$g(x) = \sum_{i=1}^{n} f(t_i^*) B_{i,d,\mathbf{t}}(x), \qquad x \in [a, b], \tag{6.15}$$

where t_i^* is the knot average $t_i^* = (t_{i+1} + \cdots + t_{i+d})/d$, is known as the *variation-diminishing spline approximation*. This is a very simple approximation, which we can think of as a generalization of piecewise-linear interpolation. It is in general smoother than piecewise-linear interpolation, but still has good shape-preserving properties, as we will see.

Theorem 6.5 *If f is non-negative then so is g, if f is monotonically non-decreasing then so is g, and if f is convex then so is g.*

Proof. We leave the cases $d = 1$ and $d = 2$ as Exercise 6.15. Otherwise, $d \geq 3$, in which case $g \in C^2[a, b]$. That g is non-negative when f is non-negative follows immediately from the fact, already established, that the B-splines $B_{i,d,\mathbf{t}}$ are non-negative.

Next, suppose that f is monotonically non-decreasing. By Theorem 4.6, the derivative of g for $x \in [a, b]$ is

$$g'(x) = d \sum_{i=1}^{n} f(t_i^*) \left(\frac{B_{i,d-1}(x)}{t_{i+d} - t_i} - \frac{B_{i+1,d-1}(x)}{t_{i+d+1} - t_{i+1}} \right) = d \sum_{i=2}^{n} \left(\frac{f(t_i^*) - f(t_{i-1}^*)}{t_{i+d} - t_i} \right) B_{i,d-1}(x),$$

which is non-negative since $f(t_1^*), \ldots, f(t_n^*)$ is a non-decreasing sequence.

Finally, suppose that f is convex. Observe that

$$t_i^* - t_{i-1}^* = \frac{t_{i+1} + \cdots + t_{i+d}}{d} - \frac{t_i + \cdots + t_{i+d-1}}{d} = \frac{t_{i+d} - t_i}{d},$$

and therefore

$$g'(x) = \sum_{i=2}^{n} c_i B_{i,d-1}(x),$$

where

$$c_i = \frac{f(t_i^*) - f(t_{i-1}^*)}{t_i^* - t_{i-1}^*}.$$

Differentiating g again and using Theorem 4.6 shows that

$$g''(x) = (d-1) \sum_{i=3}^{n} \left(\frac{c_i - c_{i-1}}{t_{i+d-1} - t_i} \right) B_{i,d-2}(x),$$

which is non-negative because the sequence $c_2, \ldots, c_n$ is non-decreasing by the convexity of f. □

6.5 Exercises

6.1 Check that Theorem 6.5 also holds in the cases $d = 1$ and $d = 2$.

Chapter 7

Surfaces

We will consider two ways of extending BB polynomials and Bézier curves to surfaces, one is to take tensor-products of BB basis polynomials, the other is to generalize BB polynomials to polynomials defined over triangles. The tensor-product approach is very general and also applies to splines in B-spline form. We consider both interpolation and least squares approximation by splines in tensor-product B-spline form.

7.1 Tensor-product BB polynomials

If we take the tensor-product of the two linear spaces of polynomials π_{d_1} and π_{d_2}, we obtain a new linear space $\pi_{d_1,d_2} = \pi_{d_1} \otimes \pi_{d_2}$, comprising all polynomials of two variables with degree $\leq d_1$ in the first variable and of degree $\leq d_2$ in the second variable. A basis can be constructed by taking products of bases for π_{d_1} and π_{d_2}. For example, the monomial bases for π_2 and π_1 are $\{1, x, x^2\}$ and $\{1, y\}$ respectively, so that a basis for $\pi_{2,1}$ is

$$\{1, x, x^2, y, xy, x^2y\}.$$

Thus a polynomial $p \in \pi_{d_1,d_2}$ can be expressed uniquely in the form

$$p(x, y) = \sum_{i=0}^{d_1} \sum_{j=0}^{d_2} a_{i,j} x^i y^j.$$

An alternative to the monomial basis is the BB polynomial basis, and thus we can represent a polynomial $p \in \pi_{d_1,d_2}$ as a *tensor-product BB polynomial*,

$$p(x, y) = \sum_{i=0}^{d_1} \sum_{j=0}^{d_2} c_{i,j} B_i^{d_1}(\lambda) B_j^{d_2}(\mu), \qquad c_{i,j} \in \mathbb{R}, \tag{7.1}$$

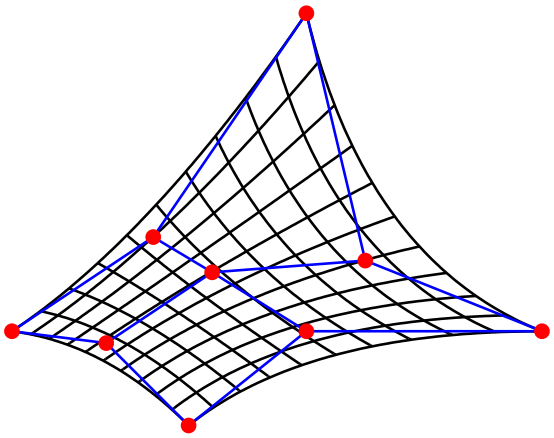

Figure 7.1: A biquadratic Bézier surface

with respect to some rectangular domain $[a_1, b_1] \times [a_2, b_2]$, where

$$\lambda = \frac{x - a_1}{b_1 - a_1}, \qquad \mu = \frac{y - a_2}{b_2 - a_2}. \tag{7.2}$$

Similarly, we define a tensor-product Bézier surface in $\mathbb{R}^k$ as a parametric polynomial

$$\mathbf{p}(x, y) = \sum_{i=0}^{d_1} \sum_{j=0}^{d_2} \mathbf{c}_{i,j} B_i^{d_1}(\lambda) B_j^{d_2}(\mu), \qquad \mathbf{c}_{i,j} \in \mathbb{R}^k. \tag{7.3}$$

We often restrict both p and $\mathbf{p}$ to the domain $[a_1, b_1] \times [a_2, b_2]$. In practice, the Euclidean space $\mathbb{R}^k$ will often be $\mathbb{R}^3$. The *control net* of $\mathbf{p}$ is the network of points and line segments consisting of the control points $\mathbf{c}_{i,j}$ and all line segments of the form $[\mathbf{c}_{i,j}, \mathbf{c}_{i+1,j}]$ and $[\mathbf{c}_{i,j}, \mathbf{c}_{i,j+1}]$. Figure 7.1 shows a biquadratic surface, where $m = n = 2$, with its control net. For the scalar-valued function p we take its control points to be $(i/d_1, j/d_2, c_{i,j})$, $i = 0, 1, \ldots, d_1$, $j = 0, 1, \ldots, d_2$.

Tensor-product BB polynomials inherit various properties from BB polynomials in one variable. On each of the four boundaries of the parameter domain $[a_1, b_1] \times [a_2, b_2]$ the function p is a univarite BB polynomial whose control polygon is one of the four boundaries of the control net of p. For example,

$$p(x, a_2) = \sum_{i=0}^{d_1} c_{i,0} B_i^{d_1}(\mu).$$

At the corners of the parameter domain, p equals one of the corner coefficients, for example

$$p(a_1, a_2) = c_{0,0}.$$

Since the tensor-product Bernstein polynomials

$$B_{i,j}^{d_1,d_2} = B_i^{d_1} \otimes B_j^{d_2},$$

sum to one:

$$\sum_{i=0}^{d_1}\sum_{j=0}^{d_1} B_{i,j}^{d_1,d_2}(\lambda,\mu) = \sum_{i=0}^{d_1}\sum_{j=0}^{d_2} B_i^{d_1}(\lambda)B_j^{d_2}(\mu) = \sum_{i=0}^{d_1} B_i^{d_1}(\lambda)\sum_{j=0}^{d_2} B_j^{d_2}(\mu) = 1,$$

every point $p(x,y)$ is a non-negative affine combination of the coefficents $c_{i,j}$. The surface $\mathbf{p}$ lies in the convex hull of its control points $\mathbf{c}_{i,j}$.

7.1.1 The de Casteljau algorithm

Given a parameter pair $(x,y) \in [a_1,b_1]\times[a_2,b_2]$, one way of computing the value $p(x,y)$ is to evaluate the basis polynomials $B_i^{d_1}$ and $B_j^{d_2}$ at x and y respectively and then apply the formula (7.1). Alternatively one can apply de Casteljau's algorithm to the rows of coefficients in each of the two directions. We apply d_1 steps of the algorithm with respect to x and d_2 steps with respect to y. The last point generated will be the point $\mathbf{p}(x,y)$, no matter how we order these d_1+d_2 steps. Consider an example. Let $d_1 = 2$, $d_2 = 3$, and

$$\begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 6 \\ 18 & 2 & 0 & 8 \\ 4 & 0 & 4 & 18 \end{bmatrix},$$

and suppose that $\lambda = 1/2$ and $\mu = 2/3$ in (7.2). Applying de Castlejau's algorithm first with respect to x gives

$$\begin{bmatrix} 0 & 0 & 0 & 6 \\ 18 & 2 & 0 & 8 \\ 4 & 0 & 4 & 18 \end{bmatrix} \to \begin{bmatrix} 9 & 1 & 0 & 7 \\ 11 & 1 & 2 & 13 \end{bmatrix} \to \begin{bmatrix} 10 & 1 & 1 & 10 \end{bmatrix}.$$

Then, applying the algorithm with respect to y gives

$$\begin{bmatrix} 10 & 1 & 1 & 10 \end{bmatrix} \to \begin{bmatrix} 4 & 1 & 7 \end{bmatrix} \to \begin{bmatrix} 2 & 5 \end{bmatrix} \to \begin{bmatrix} 4 \end{bmatrix},$$

so that $\mathbf{p}(x,y) = 4$. Alternatively, we could apply the algorithm first with respect to y, giving

$$\begin{bmatrix} 0 & 0 & 0 & 6 \\ 18 & 2 & 0 & 8 \\ 4 & 0 & 4 & 18 \end{bmatrix} \to \begin{bmatrix} 0 & 0 & 4 \\ 22/3 & 1/3 & 16/3 \\ 1/3 & 8/3 & 40/3 \end{bmatrix} \to \begin{bmatrix} 0 & 8/3 \\ 26/9 & 34/9 \\ 20/9 & 88/9 \end{bmatrix} \to \begin{bmatrix} 16/9 \\ 94/27 \\ 196/27 \end{bmatrix},$$

and then with respect to x,

$$\begin{bmatrix} 16/9 \\ 94/27 \\ 196/27 \end{bmatrix} \rightarrow \begin{bmatrix} 71/27 \\ 145/27 \end{bmatrix} \rightarrow \begin{bmatrix} 4 \end{bmatrix},$$

yielding the same answer.

7.1.2 Joining polynomial pieces together

Joining tensor-product BB polynomials is similar to joining univarite BB polynomials if we arrange their parameter domains in a rectangular grid. By Theorem 2.1, the partial derivative of p of order (r_1, r_2) is

$$D^{r_1,r_2}p(x,y) = \frac{d_1!}{(d_1-r_1)!}\frac{1}{h_1^{r_1}}\frac{d_2!}{(d_2-r_2)!}\frac{1}{h_2^{r_2}}\sum_{i=0}^{d_1-r_1}\sum_{j=0}^{d_2-r_2}\Delta^{r_1,r_2}c_{i,j}B_i^{d_1-r_1}(\lambda)B_j^{d_2-r_2}(\mu),$$

where $h_i = b_i - a_i$, $i = 1, 2$, and $\Delta^{r_1,r_2}c_{i,j}$ denotes the forward difference of $c_{i,j}$ formed by differencing r_1 times in i and r_2 times in j. So,

$$\begin{aligned}
\Delta^{0,0}c_{i,j} &= c_{i,j}, \\
\Delta^{1,0}c_{i,j} &= c_{i+1,j} - c_{i,j}, \\
\Delta^{0,1}c_{i,j} &= c_{i,j+1} - c_{i,j}, \\
\Delta^{1,1}c_{i,j} &= c_{i+1,j+1} - c_{i+1,j} - c_{i,j+1} + c_{i,j},
\end{aligned}$$

and so on.

If, for example, p is bicubic, i.e., $d_1 = d_2 = 3$ then it is uniquely determined by the sixteen Hermite corner data

$$D^{r_1,r_2}p(a_k, b_l), \qquad r_1, r_2 \in \{0,1\}, \quad k, l \in \{1,2\}.$$

By the formula for the partial derivatives, these Hermite data can be expressed in terms of differences of the BB coefficients. For example, at the corner point (a_1, a_2) we have

$$\begin{aligned}
p(a_1,a_2) &= c_{0,0} \\
D^{1,0}p(a_1,a_2) &= \frac{3}{h_1}(c_{1,0} - c_{0,0}), \\
D^{0,1}p(a_1,a_2) &= \frac{3}{h_2}(c_{0,1} - c_{0,0}), \\
D^{1,1}p(a_1,a_2) &= \frac{9}{h_1h_2}(c_{1,1} - c_{1,0} - c_{0,1} + c_{0,0}).
\end{aligned}$$

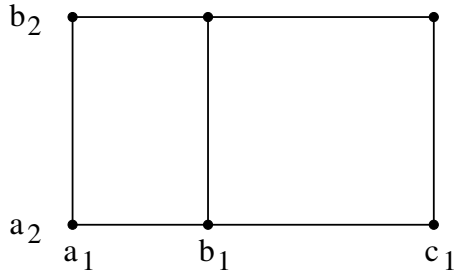

Figure 7.2: Parameter domains of p and q

Alternatively, one obtains the BB coefficients from the Hermite data.

In this way, in analogy to the method of Section 2.4, we could construct a bicubic Hermite spline interpolant to data given on a rectangular grid of points (x_k, y_l), for some $x_1 < x_2 < \cdots < x_{m_1}$ and for some $y_1 < y_2 < \cdots < y_{m_2}$. Given some Hermite data values $f_{k,l}^{r_1,r_2} \in \mathbb{R}$, $r_1, r_2 \in \{0, 1\}$, $k = 1, \ldots, m_1$, $l = 1, \ldots, m_2$, we could find a unique C^1 bicubic spline g that satisfies

$$D^{r_1,r_2} g(x_k, y_l) = f_{k,l}^{r_1,r_2}, \qquad r_1, r_2 \ \in \{0, 1\}, \quad k = 1, \ldots, m_1, \quad l = 1, \ldots, m_2.$$

An alternative approach to constructing a tensor-product spline is to derive the smoothness conditions required by the BB coefficients of adjacent polynomial pieces. Due to the tensor-product nature of the spline, these conditions just reduce to those for univariate polynomials, applied to each row of coefficients. For example, suppose that q is a second BB polynomial

$$q(x, y) = \sum_{i=0}^{d_1} \sum_{j=0}^{d_2} e_{i,j} B_i^{d_1}(\nu) B_j^{d_2}(\mu), \qquad e_{i,j} \in \mathbb{R}, \tag{7.4}$$

where

$$\nu = \frac{x - b_1}{c_1 - b_1},$$

on an adjacent parameter domain $[b_1, c_1] \times [a_2, b_2]$; see Figure 7.2. Then p and q join with continuity of order r at $x = b_1$, written C^r, if and only if

$$D^{k,0} p(b_1, y) = D^{k,0} q(b_1, y), \qquad k = 0, 1, \ldots, r, \quad y \in [a_2, b_2].$$

From the formula for partial derivatives, these conditions are equivalent to

$$\frac{1}{(b_1 - a_1)^k} \sum_{j=0}^{d_2} \Delta^{k,0} c_{d_1-k,j} B_j^{d_2}(\mu) = \frac{1}{(c_1 - b_1)^k} \sum_{j=0}^{d_2} \Delta^{k,0} e_{0,j} B_j^{d_2}(\mu),$$

for $k = 0, 1, \ldots, r$ and $\mu \in [0, 1]$. So by the linear independence of the basis polynomials $B_j^{d_2}$, the conditions are equivalent to

$$\frac{\Delta^{k,0} c_{d_1-k,j}}{(b_1 - a_1)^k} = \frac{\Delta^{k,0} e_{0,j}}{(c_1 - b_1)^k}, \qquad k = 0, 1, \ldots, r, \quad j = 0, 1, \ldots, d_2.$$

Similar to Theorem 2.4, these conditions can be expressed in the form

$$e_{i,j} = \sum_{k=0}^{i} c_{d-i+k,j} B_k^i(\alpha), \qquad i = 0, 1, \ldots, r, \quad j = 0, 1, \ldots, d_2,$$

where

$$\alpha = \frac{c_1 - a_1}{b_1 - a_1}.$$

7.2 Tensor-product splines in B-spline form

A tensor-product spline function has the form

$$s(x, y) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} c_{i,j} \phi_i(x) \psi_j(y), \tag{7.5}$$

where $\phi_i(x) = B_{i,d_1,\boldsymbol{\sigma}}(x)$ and $\psi_j(y) = B_{j,d_2,\boldsymbol{\tau}}(y)$ are the B-splines defined by the polynomial degrees d_1 and d_2 and the knot vectors

$$\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_{n_1+d_1+1}), \qquad \boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_{n_2+d_2+1}).$$

We can also write s as

$$s(x, y) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} c_{i,j} B_{i,j}(x, y),$$

a linear combination of the tensor-product B-splines $B_{i,j}(x, y) = \phi_i(x)\psi_j(y)$.

7.2.1 Evaluation

To evaluate s at a given point (x, y) we can first locate knot intervals $[\sigma_\mu, \sigma_{\mu+1}]$ and $[\tau_\nu, \tau_{\nu+1}]$ that contain x and y respectively, in which case

$$s(x, y) = \sum_{i=\mu-d_1}^{\mu} \sum_{j=\nu-d_2}^{\nu} c_{i,j} \phi_i(x) \psi_j(y).$$

We can now use Algorithm 1 of Subsection 3.9.1 to compute the $\phi_i(x)$ and $\psi_j(y)$ in the sum to compute $s(x, y)$. We could, for example, use the vector and matrix notation

$$\phi(x) = [\phi_{\mu-d_1}(x), \ldots, \phi_\mu(x)]^T, \qquad \psi(y) = [\psi_{\nu-d_2}(y), \ldots, \psi_\nu(y)]^T,$$

and

$$C = [c_{i,j}]_{i=\mu-d_1,\ldots,\mu, j=\nu-d_2,\ldots,\nu},$$

so that

$$s(x, y) = \phi(x)^T C \psi(y).$$

7.2.2 Interpolation to gridded data

Suppose next that we are given rectangular grid data $(x_i, y_j, f_{i,j})$, $i = 1, \ldots, n_1$, $j = 1, \ldots, n_2$, where the x_i are increasing and the y_j are increasing. We would like to find the coefficients $c_{i,j}$ of the spline in (7.5) such that $s(x_i, y_j) = f_{i,j}$ for all $i = 1, \ldots, n_1$, $j = 1, \ldots, n_2$. Thus we want to solve the equations

$$\sum_{p=1}^{n_1} \sum_{q=1}^{n_2} c_{p,q} \phi_p(x_i) \psi_q(y_j) = f_{i,j}, \qquad i = 1, \ldots, n_1, \quad j = 1, \ldots, n_2. \tag{7.6}$$

We can break this problem down into two simpler problems by defining

$$d_{p,j} = \sum_{q=1}^{n_2} c_{p,q} \psi_q(y_j), \qquad p = 1, \ldots, n_1, \quad j = 1, \ldots, n_2. \tag{7.7}$$

With this substitution we can rewrite the equations in (7.6) as

$$\sum_{p=1}^{n_1} d_{p,j} \phi_p(x_i) = f_{i,j}, \tag{7.8}$$

and we can now solve the problem in two steps. In the first step we solve the equations (7.8) for the unknowns $d_{p,j}$ and in the second step we solve (7.7) for the unknowns $c_{p,q}$. We can express both steps in vector and matrix notation, defining the matrices

$$A = \begin{bmatrix} \phi_1(x_1) & \cdots & \phi_{n_1}(x_1) \\ \vdots & & \vdots \\ \phi_1(x_{n_1}) & \cdots & \phi_{n_1}(x_{n_1}) \end{bmatrix}, \quad B = \begin{bmatrix} \psi_1(y_1) & \cdots & \psi_{n_2}(y_1) \\ \vdots & & \vdots \\ \psi_1(y_{n_2}) & \cdots & \psi_{n_2}(y_{n_2}) \end{bmatrix},$$

and

$$C = [c_{i,j}]_{i=1,\ldots,n_1,j=1,\ldots,n_2},$$
$$D = [d_{i,j}]_{i=1,\ldots,n_1,j=1,\ldots,n_2},$$
$$F = [f_{i,j}]_{i=1,\ldots,n_1,j=1,\ldots,n_2}.$$

Further, let $\mathbf{d}_1, \ldots, \mathbf{d}_{n_2}$ denote the columns of D and $\tilde{\mathbf{d}}_1, \ldots, \tilde{\mathbf{d}}_{n_1}$ the rows of D, and similarly for C and F. With this notation, the first step consists of solving

$$A\mathbf{d}_j = \mathbf{f}_j, \qquad j = 1, \ldots, n_2, \tag{7.9}$$

for the vectors $\mathbf{d}_j$, and the second step consists of solving

$$B\tilde{\mathbf{c}}_p = \tilde{\mathbf{d}}_p, \qquad p = 1, \ldots, n_1, \tag{7.10}$$

for the vectors $\tilde{\mathbf{c}}_p$. Thus, in the first step, for each j, we carry out interpolation of the j-th row of data $(x_i, f_{i,j})$. In the second step, for each p, we carry out interpolation of the p-th column of the coefficients computed in the first step, $(y_j, d_{p,j})$. We can also write the two steps more succinctly as

$$AD = F, \tag{7.11}$$
$$BC^T = D^T. \tag{7.12}$$

Thus, due to the tensor-product nature of the spline and the rectangular nature of the data, the interpolation problem (7.6) reduces to univariate interpolation in each variable independently, and has a unique solution if the matrices A and B are non-singular. Due to the Schoenberg-Whitney theorem (Theorem 6.2) we conclude as follows.

Theorem 7.1 *If $\phi_i(x_i) > 0$, $i = 1, \ldots, n_1$, and $\psi_j(y_j) > 0$, $j = 1, \ldots, n_2$, then the interpolation problem (7.6) has a unique solution whose coefficient matrix C can be found by solving the linear systems (7.11) and (7.12).*

Notice here that we are solving several linear systems using the same matrix A or B, only the right hand side changes. So we can reduce the amount of computation considerably if we use a technique like LU decomposition to factorize A and B in a preprocessing step, so that the individual linear systems can be solved efficiently.

7.2.3 Least squares approximation

In analogy to the interpolation problem, the least squares problem for a rectangular grid of data can be reduced to univariate approxmation. Suppose we are given

rectangular grid data $(x_i, y_j, f_{i,j})$, $i = 1, \ldots, m_1$, $j = 1, \ldots, m_2$, where the x_i are increasing and the y_j are increasing. Assuming that $n_1 \leq m_1$ and $n_2 \leq m_2$, we would like to find the coefficients $c_{i,j}$ of the spline in (7.5) such that $s(x_i, y_j) \approx f_{i,j}$ for all $i = 1, \ldots, m_1$, $j = 1, \ldots, m_2$. The least squares solution to this problem consists of minimizing the sum of squares

$$E = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} (s(x_i, y_j) - f_{i,j})^2.$$

Thus we want to find the $c_{p,q}$ that minimize

$$E = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \left(\sum_{p=1}^{n_1} \sum_{q=1}^{n_2} c_{p,q} \phi_p(x_i) \psi_q(y_j) - f_{i,j} \right)^2. \tag{7.13}$$

We can proceed as in the univariate case treated in Section 6.3. The function E, as a function of the $n_1 n_2$ variables $c_{p,q}$, is quadratic, and at a minimum its partial derivatives are zero. Its partial derivative with respect to $c_{k,l}$ is

$$\frac{\partial E}{\partial c_{k,l}} = 2 \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \phi_k(x_i) \psi_l(y_j) \left(\sum_{p=1}^{n_1} \sum_{q=1}^{n_2} c_{p,q} \phi_p(x_i) \psi_q(y_j) - f_{i,j} \right), \tag{7.14}$$

and so the partial derivatives of E are zero if and only if

$$\sum_{p=1}^{n_1} \sum_{q=1}^{n_2} \left(\sum_{i=1}^{m_1} \phi_k(x_i) \phi_p(x_i) \right) \left(\sum_{j=1}^{m_1} \psi_l(y_j) \psi_q(y_j) \right) c_{p,q} = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \phi_k(x_i) \psi_l(y_j) f_{i,j},$$

for $k = 1, \ldots, n_1$, $l = 1, \ldots, n_2$, These are the normal equations for the minimization problem and we can break them down into univariate linear systems. Let us define the matrices

$$A = \begin{bmatrix} \phi_1(x_1) & \cdots & \phi_{n_1}(x_1) \\ \vdots & & \vdots \\ \phi_1(x_{m_1}) & \cdots & \phi_{n_1}(x_{m_1}) \end{bmatrix}, \quad B = \begin{bmatrix} \psi_1(y_1) & \cdots & \psi_{n_2}(y_1) \\ \vdots & & \vdots \\ \psi_1(y_{m_2}) & \cdots & \psi_{n_2}(y_{m_2}) \end{bmatrix},$$

and

$$\begin{aligned} C &= [c_{i,j}]_{i=1,\ldots,n_1, j=1,\ldots,n_2}, \\ D &= [d_{i,j}]_{i=1,\ldots,n_1, j=1,\ldots,m_2}, \\ F &= [f_{i,j}]_{i=1,\ldots,m_1, j=1,\ldots,m_2}. \end{aligned}$$

Let $\mathbf{d}_1, \ldots, \mathbf{d}_{m_2}$ denote the columns of D and $\tilde{\mathbf{d}}_1, \ldots, \tilde{\mathbf{d}}_{n_1}$ the rows of D, and similarly for C and F. With this notation, we can solve the system in two steps. The first step consists of solving

$$(A^T A)\mathbf{d}_j = A^T \mathbf{f}_j, \qquad j = 1, \ldots, m_2,$$

for the vectors $\mathbf{d}_j$, and the second step consists of solving

$$(B^T B)\tilde{\mathbf{c}}_p = B^T \tilde{\mathbf{d}}_p, \qquad p = 1, \ldots, n_1,$$

for the vectors $\tilde{\mathbf{c}}_p$. We can write the two steps more succinctly as

$$(A^T A)D = A^T F, \tag{7.15}$$

$$(B^T B)C^T = B^T D^T. \tag{7.16}$$

Applying the Schoenberg-Whitney theorem again, we make the following conclusion.

Theorem 7.2 *Suppose that there exists a subsequence $(x_{k_1}, x_{k_2}, \ldots x_{k_{n_1}})$ such that $\phi_i(x_{k_i}) > 0$, $i = 1, \ldots, n_1$, and a subsequence $(y_{l_1}, y_{l_2}, \ldots y_{l_{n_2}})$ such that $\psi_j(x_{l_j}) > 0$, $j = 1, \ldots, n_2$. Then E in (7.13) has a unique minimizer whose coefficient matrix C can be found by solving the linear systems (7.15) and (7.16).*

7.3 Triangular BB polynomials

An alternative choice of bivariate polynomials is the linear space of polynomials of total degree at most d, for some $d \geq 0$, i.e, polynomials of the form

$$p(x, y) = \sum_{0 \leq i+j \leq d} a_{i,j} x^i y^j,$$

(where we understand $i, j \geq 0$ in the summation). The degree of p is the largest value of $i + j$ over all non-zero $a_{i,j}$ in the summation. As in the univariate case, we denote this space by π_d. The monomials $x^i y^j$ in the sum are linearly independent and form a basis for π_d. Since the number of such polynomials is

$$1 + 2 + \cdots + (d + 1) = \binom{d+2}{2},$$

this is also the dimension of π_d. For example, the monomial basis of π_2 is

$$\{1, x, y, x^2, xy, y^2\},$$

which has six elements.

If we use such polynomials to model surfaces it turns out that, similar to the univariate case, there is again an alternative basis that makes the modelling easier. This is the basis of triangular BB basis polynomials

$$B_{i,j}^d(x,y) = \frac{d!}{i!j!(d-i-j)!} x^i y^j (1-x-y)^{d-i-j}, \qquad 0 \le i+j \le d,$$

To show that these polynomials are linearly independent, it is sufficient to show that any monomial $x^k y^l$, $k+l \le d$, can be expressed as a linear combination of the $B_{i,j}^d$.

Lemma 7.3 *For $k, l \ge 0$ and $k+l \le d$,*

$$x^k y^l = \frac{(d-k-l)!}{d!} \sum_{\substack{i+j \le d \\ i \ge k,\, j \ge l}} \frac{i!j!}{(i-k)!(j-l)!} B_{i,j}^d(x,y).$$

Proof. We use the trinomial theorem:

$$\begin{aligned} x^k y^l &= x^k y^l (x + y + (1-x-y))^{d-k-l} \\ &= x^k y^l \sum_{\substack{i+j \le d-k-l \\ i \ge 0,\, j \ge 0}} \frac{(d-k-l)!}{i!j!(d-k-l-i-j)!} x^i y^j (1-x-y)^{d-k-l-i-j} \\ &= \sum_{\substack{i+j \le d \\ i \ge k,\, j \ge l}} \frac{(d-k-l)!}{(i-k)!(j-l)!(d-i-j)!} x^i y^j (1-x-y)^{d-i-j}, \end{aligned}$$

which by the definition of $B_{i,j}^d$ gives the result. □

We are often mainly interested in (x,y) in the triangular domain

$$D := \{(x,y) \in \mathbb{R}^2 : x, y \ge 0, x+y \le 1\}$$

because the polynomials $B_{i,j}^d$ are non-negative in D. By the trinomial theorem they also sum to one,

$$\sum_{0 \le i+j \le d} B_{i,j}^n(x,y) = \big(x + y + (1-x-y)\big)^d = 1.$$

The form of $B_{i,j}^d$ suggests a more symmetric way of viewing it, as a function of three variables. Let $\mathbf{i} = (i_1, i_2, i_3)$ for $i_k = 0, 1, 2, \ldots$, and let $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$, with $\lambda_k \in \mathbb{R}$. Then we define

$$B_{\mathbf{i}}^d(\boldsymbol{\lambda}) = \binom{d}{\mathbf{i}} \lambda_1^{i_1} \lambda_2^{i_2} \lambda_3^{i_3}, \tag{7.17}$$

where

$$\binom{d}{\mathbf{i}} = \frac{d!}{i_1!i_2!i_3!}.$$

Then we see that

$$B^d_{i,j}(x,y) = B^d_{i,j,d-i-j}(x,y,1-x-y).$$

For example, the cubic basis polynomials with the trivariate indexing are:

$$\begin{array}{ccccccc} & & & B^3_{003} & & & \\ & & B^3_{102} & & B^3_{012} & & \\ & B^3_{201} & & B^3_{111} & & B^3_{021} & \\ B^3_{300} & & B^3_{210} & & B^3_{120} & & B^3_{030} \end{array}$$

and are given by the formulas

$$\begin{array}{ccccccc} & & & \lambda_3^3 & & & \\ & & 3\lambda_1\lambda_3^2 & & 3\lambda_2\lambda_3^2 & & \\ & 3\lambda_1^2\lambda_3 & & 6\lambda_1\lambda_2\lambda_3 & & 3\lambda_2^2\lambda_3 & \\ \lambda_1^3 & & 3\lambda_1^2\lambda_2 & & 3\lambda_1\lambda_2^2 & & \lambda_2^3 \end{array}$$

We now define a triangular BB polynomial with respect to an arbitary triangle $T \in \mathbb{R}^2$ with vertices $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ as

$$p(\mathbf{x}) = p(x,y) = \sum_{|\mathbf{i}|=d} c_{\mathbf{i}} B^d_{\mathbf{i}}(\boldsymbol{\lambda}), \qquad c_{\mathbf{i}} \in \mathbb{R}, \tag{7.18}$$

where $|\mathbf{i}| = i_1 + i_2 + i_3$ and $\lambda_1, \lambda_2, \lambda_3$ are the *barycentric coordinates* of the point $\mathbf{x} = (x,y)$ with respect to the triangle T, i.e., the three values such that

$$\lambda_1 + \lambda_2 + \lambda_3 = 1, \tag{7.19}$$

$$\lambda_1\mathbf{v}_1 + \lambda_2\mathbf{v}_2 + \lambda_3\mathbf{v}_3 = \mathbf{x}. \tag{7.20}$$

Analogously, if we replace the coefficients $c_{\mathbf{i}}$ by points $\mathbf{c}_{\mathbf{i}}$ in some Euclidean space such as $\mathbb{R}^3$, we obtain a triangular Bézier surface

$$\mathbf{p}(\mathbf{x}) = \mathbf{p}(x,y) = \sum_{|\mathbf{i}|=d} \mathbf{c}_{\mathbf{i}} B^d_{\mathbf{i}}(\boldsymbol{\lambda}). \tag{7.21}$$

The points $\mathbf{c}_{\mathbf{i}}$ are the *control points* of $\mathbf{p}$, which, together with all line segments that connect neighbouring points, form the *control net* of $\mathbf{p}$. Similarly, the control points of p are defined to be $(\xi_{\mathbf{i}}, c_{\mathbf{i}}) \in \mathbb{R}^3$, $|\mathbf{i}| = d$, where $\xi_{\mathbf{i}}$ is the *domain point*

$$\xi_{\mathbf{i}} = \frac{i_1\mathbf{v}_1 + i_2\mathbf{v}_2 + i_3\mathbf{v}_3}{d}.$$

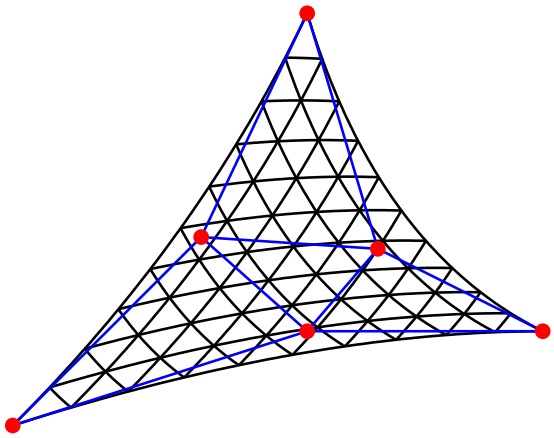

Figure 7.3: A quadratic triangular Bézier surface

Figure 7.3 shows a quadratic surface, where $d = 2$, with its control net.

How do we find $\lambda_1, \lambda_2, \lambda_3$? If $\mathbf{v}_k = (x_k, y_k)$, $k = 1, 2, 3$, then we can express (7.19–7.20) in matrix form as

$$\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ y \end{bmatrix},$$

and Cramer's rule gives the solution

$$\lambda_1 = \frac{D_1}{D}, \qquad \lambda_2 = \frac{D_2}{D}, \qquad \lambda_3 = \frac{D_3}{D},$$

where

$$D_1 = \begin{vmatrix} 1 & 1 & 1 \\ x & x_2 & x_3 \\ y & y_2 & y_3 \end{vmatrix}, \quad D_2 = \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x & x_3 \\ y_1 & y & y_3 \end{vmatrix}, \quad D_3 = \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x \\ y_1 & y_2 & y \end{vmatrix},$$

and

$$D = \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}.$$

Since the signed area $A(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ of the triangle $T = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ is $D/2$, and the signed area of, for example, the subtriangle $A(\mathbf{x}, \mathbf{v}_2, \mathbf{v}_3)$ is $D_1/2$, and so on, we can alternatively express the barycentric coordinates of $\mathbf{x}$ as ratios of triangle areas,

$$\lambda_1 = \frac{A(\mathbf{x}, \mathbf{v}_2, \mathbf{v}_3)}{A(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)}, \qquad \lambda_2 = \frac{A(\mathbf{v}_1, \mathbf{x}, \mathbf{v}_3)}{A(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)}, \qquad \lambda_3 = \frac{A(\mathbf{v}_1, \mathbf{v}_2, \mathbf{x})}{A(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)}.$$

If $\mathbf{x} \in T$ then these coordinates are non-negative.

Triangular BB polynomials have several properties analogous to tensor-product ones. For example, on each of the three edges of the triangle T, p is a univariate polynomial whose BB control polygon is part of the control net of p. For example, if $\mathbf{x}$ is a point on the edge $[\mathbf{v}_1, \mathbf{v}_2]$ then its barycentric coordinates are such that $\lambda_3 = 0$ and $\lambda_1 + \lambda_2 = 1$ and then

$$p(\mathbf{x}) = p(\lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2) = \sum_{i_1+i_2=d} c_{i_1,i_2,0} \frac{d!}{i_1! i_2!} \lambda_1^{i_1} \lambda_2^{i_2}.$$

At the corners of the triangle T, p equals the corresponding corner coefficient. For example if $\mathbf{x} = \mathbf{v}_1$ then $\lambda_2 = \lambda_3 = 0$ and $\lambda_1 = 1$ and so

$$p(\mathbf{x}) = p(\mathbf{v}_1) = c_{d,0,0}.$$

Like tensor-product Bézier surfaces, triangular Bézier surfaces are affinely invariant and have the convex hull and bounding box properties.

7.3.1 The de Casteljau algorithm

The basis polynomials satisfy a recursion formula analogous to the univariate case, c.f. Lemma 1.3. Here we make the convention that $B_{\mathbf{i}}^d(\mathbf{x}) = 0$ whenever $i_1 < 0$ or $i_2 < 0$ or $i_3 < 0$.

Lemma 7.4 *For $i_1, i_2, i_3 \geq 0$ and $|\mathbf{i}| = d$,*

$$B_{\mathbf{i}}^d(\boldsymbol{\lambda}) = \sum_{k=1}^{3} \lambda_k B_{\mathbf{i}-\mathbf{e}_k}^{d-1}(\boldsymbol{\lambda}), \tag{7.22}$$

where $\mathbf{e}_1 = (1,0,0)$, $\mathbf{e}_2 = (0,1,0)$, and $\mathbf{e}_3 = (0,0,1)$.

Proof. The proof is similar to that of Lemma 1.3. For $i_1, i_2, i_3 \geq 1$, one can easily check the following formula for trinomial coefficients,

$$\binom{d}{\mathbf{i}} = \binom{d-1}{\mathbf{i}-\mathbf{e}_1} + \binom{d-1}{\mathbf{i}-\mathbf{e}_2} + \binom{d-1}{\mathbf{i}-\mathbf{e}_3},$$

and substituting this into the definition (7.17) leads to (7.22). The cases when any of the indices i_1, i_2, i_3 are zero are easy to check. □

From this recursion we obtain a de Casteljau algorithm for evaluating p in (7.18) at $\mathbf{x} \in \mathbb{R}^2$. We initialize the algorithm by setting $c_{\mathbf{i}}^0 = c_{\mathbf{i}}$, $|\mathbf{i}| = d$. Then, for each $r = 1, \ldots, d$, let

$$c_{\mathbf{i}}^r = \lambda_1 c_{\mathbf{i}+\mathbf{e}_1}^{r-1} + \lambda_2 c_{\mathbf{i}+\mathbf{e}_2}^{r-1} + \lambda_3 c_{\mathbf{i}+\mathbf{e}_3}^{r-1}, \qquad |\mathbf{i}| = d - r. \tag{7.23}$$

Theorem 7.5 *The last value computed, $c_{0,0,0}^d$, is the value of $p(x)$ in (7.18).*

Proof. Similar to the proof of Theorem 1.6, one uses the recursion (7.22) to show, by induction on r, that

$$p(\mathbf{x}) = \sum_{|\mathbf{i}|=d-r} c_{\mathbf{i}}^r B_{\mathbf{i}}^{d-r}(\boldsymbol{\lambda}) \tag{7.24}$$

for all $r = 0, 1, \ldots, d$. For $r = 0$, (7.24) follows from the definition of $p(\mathbf{x})$. For $r \geq 1$, we may assume that (7.24) holds with r replaced by $r - 1$, and then

$$\begin{aligned}
p(\mathbf{x}) &= \sum_{|\mathbf{i}|=d-r+1} c_{\mathbf{i}}^{r-1} B_{\mathbf{i}}^{d-r+1}(\boldsymbol{\lambda}) \\
&= \sum_{|\mathbf{i}|=d-r+1} c_{\mathbf{i}}^{r-1} \sum_{k=1}^{3} \lambda_k B_{\mathbf{i}-\mathbf{e}_k}^{d-r}(\boldsymbol{\lambda}) \\
&= \sum_{k=1}^{3} \lambda_k \sum_{|\mathbf{i}|=d-r+1} c_{\mathbf{i}}^{r-1} B_{\mathbf{i}-\mathbf{e}_k}^{d-r}(\boldsymbol{\lambda}) \\
&= \sum_{k=1}^{3} \lambda_k \sum_{\substack{|\mathbf{i}|=d-r+1 \\ i_k \geq 1}} c_{\mathbf{i}}^{r-1} B_{\mathbf{i}-\mathbf{e}_k}^{d-r}(\boldsymbol{\lambda}) \\
&= \sum_{k=1}^{3} \lambda_k \sum_{|\mathbf{i}|=d-r} c_{\mathbf{i}+\mathbf{e}_k}^{r-1} B_{\mathbf{i}}^{d-r}(\boldsymbol{\lambda}),
\end{aligned}$$

which, by (7.23), reduces to (7.24). □

The de Casteljau algorithm can be viewed as a tetrahedral scheme. The flow of computations in the case $d = 2$ is as follows,

$$\begin{array}{ccccccccc}
 & & & c_{002}^0 & & & & c_{001}^1 & \\
 & c_{101}^0 & & & c_{011}^0 & & & & \\
c_{200}^0 & & & c_{110}^0 & & c_{020}^0 & \to & c_{100}^1 & & c_{010}^1 & \to & c_{000}^2
\end{array}$$

Analogous to Theorem 1.7, the de Casteljau coefficients are themselves triangular BB polynomials:

Theorem 7.6 *The coefficient $c_{\mathbf{i}}^r$, $|\mathbf{i}| = d - r$, in the de Casteljau algorithm (7.23), viewed as a function of $\mathbf{x}$, is the BB polynomial*

$$c_{\mathbf{i}}^r = \sum_{|\mathbf{j}|=r} c_{\mathbf{i}+\mathbf{j}} B_{\mathbf{j}}^r(\boldsymbol{\lambda}). \tag{7.25}$$

The proof is similar to that of Theorem 1.7, using induction on r but using instead the recursion (7.22).

7.3.2 Derivatives

Let us consider the directional derivative of p in some vector direction $\mathbf{u} \in \mathbb{R}^2$,

$$D_{\mathbf{u}} p(\mathbf{x}) = \mathbf{u} \cdot \nabla p(\mathbf{x}),$$

where $\cdot$ is the scalar product and $\nabla p(\mathbf{x})$ denotes the gradient of p at $\mathbf{x}$. In order to find a formula for $D_{\mathbf{u}} p(\mathbf{x})$ let us consider first how we take a directional derivative of a basis polynomial.

Lemma 7.7 *The directional derivative of the basis polynomial $B_{\mathbf{i}}^d$ in (7.18) in the vector direction $\mathbf{u} \in \mathbb{R}^2$ is*

$$D_{\mathbf{u}} B_{\mathbf{i}}^d(\boldsymbol{\lambda}) = d \sum_{k=1}^{3} \mu_k B_{\mathbf{i}-\mathbf{e}_k}^{d-1}(\boldsymbol{\lambda}), \tag{7.26}$$

where μ_1, μ_2, μ_3 are the unique solutions to the equations

$$\mu_1 + \mu_2 + \mu_3 = 0, \tag{7.27}$$

$$\mu_1 \mathbf{v}_1 + \mu_2 \mathbf{v}_2 + \mu_3 \mathbf{v}_3 = \mathbf{u}. \tag{7.28}$$

We might call μ_1, μ_2, μ_3 the 'directional coordinates' of $\mathbf{u}$ with respect to T. They are similar to barycentric coordinates, but they sum to zero instead of one. As an example, the vector $\mathbf{u} = \mathbf{v}_2 - \mathbf{v}_1$ has directional coordinates $\mu_1 = -1$, $\mu_2 = 1$, and $\mu_3 = 0$, and the vector $\mathbf{u} = \mathbf{v}_3 - (\mathbf{v}_1 + \mathbf{v}_2)/2$ has directional coordinates $\mu_1 = -1/2$, $\mu_2 = -1/2$, and $\mu_3 = 1$.

Proof. By the definition of $B_{\mathbf{i}}^d(\boldsymbol{\lambda})$ in (7.17), we see that

$$\frac{\partial}{\partial \lambda_k} B_{\mathbf{i}}^d(\boldsymbol{\lambda}) = d B_{\mathbf{i}-\mathbf{e}_k}^{d-1}(\boldsymbol{\lambda}), \quad k = 1, 2, 3,$$

and therefore, by the chain rule,

$$D_{\mathbf{u}} B_{\mathbf{i}}^d(\boldsymbol{\lambda}) = \sum_{k=1}^{3} \frac{\partial}{\partial \lambda_k} B_{\mathbf{i}}^d(\boldsymbol{\lambda}) D_{\mathbf{u}} \lambda_k = d \sum_{k=1}^{3} B_{\mathbf{i}-\mathbf{e}_k}^{d-1}(\boldsymbol{\lambda}) D_{\mathbf{u}} \lambda_k.$$

Taking the directional derivative of equations (7.19–7.20) in the direction $\mathbf{u}$ gives

$$\begin{aligned} D_{\mathbf{u}}\lambda_1 + D_{\mathbf{u}}\lambda_2 + D_{\mathbf{u}}\lambda_3 &= 0, \\ D_{\mathbf{u}}\lambda_1 \mathbf{v}_1 + D_{\mathbf{u}}\lambda_2 \mathbf{v}_2 + D_{\mathbf{u}}\lambda_3 \mathbf{v}_3 &= \mathbf{u}, \end{aligned}$$

and so $D_{\mathbf{u}}\lambda_k = \mu_k$, $k = 1, 2, 3$. □

In analogy to Theorem 1.8, we have

Theorem 7.8 *The directional derivative of the BB polynomial p in (7.18) in the vector direction $\mathbf{u} \in \mathbb{R}^2$ is*

$$D_{\mathbf{u}} p(\mathbf{x}) = d \sum_{|\mathbf{i}|=d-1} \Delta_{\mathbf{u}} c_{\mathbf{i}} B_{\mathbf{i}}^{d-1}(\boldsymbol{\lambda}), \tag{7.29}$$

where

$$\Delta_{\mathbf{u}} c_{\mathbf{i}} = \Delta_{\mathbf{u},T} c_{\mathbf{i}} = \sum_{k=1}^{3} \mu_k c_{\mathbf{i}+\mathbf{e}_k}.$$

Proof. By Lemma 7.7,

$$\begin{aligned} D_{\mathbf{u}} p(\mathbf{x}) &= \sum_{|\mathbf{i}|=d} c_{\mathbf{i}} D_{\mathbf{u}} B_{\mathbf{i}}^d(\boldsymbol{\lambda}) \\ &= d \sum_{|\mathbf{i}|=d} c_{\mathbf{i}} \sum_{k=1}^{3} \mu_k B_{\mathbf{i}-\mathbf{e}_k}^{d-1}(\boldsymbol{\lambda}) \\ &= d \sum_{k=1}^{3} \mu_k \sum_{|\mathbf{i}|=d} c_{\mathbf{i}} B_{\mathbf{i}-\mathbf{e}_k}^{d-1}(\boldsymbol{\lambda}) \\ &= d \sum_{k=1}^{3} \mu_k \sum_{\substack{|\mathbf{i}|=d \\ i_k \geq 1}} c_{\mathbf{i}} B_{\mathbf{i}-\mathbf{e}_k}^{d-1}(\boldsymbol{\lambda}) \\ &= d \sum_{k=1}^{3} \mu_k \sum_{|\mathbf{i}|=d-1} c_{\mathbf{i}+\mathbf{e}_k} B_{\mathbf{i}}^{d-1}(\boldsymbol{\lambda}). \end{aligned}$$

□

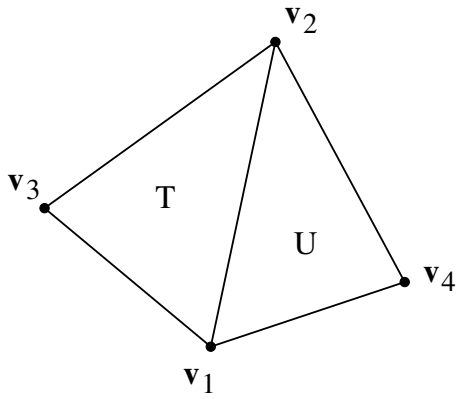

Figure 7.4: Triangles sharing a common edge.

Continuing to take directional derivatives in this way, we can also find higher order derivatives.

Theorem 7.9 *The r-th order derivative of the BB polynomial p in (7.18) in the vector directions* $\mathbf{u}_1, \ldots, \mathbf{u}_r \in \mathbb{R}^2$, $1 \leq r \leq d$, *is*

$$D_{\mathbf{u}_1} \cdots D_{\mathbf{u}_r} p(\mathbf{x}) = \frac{d!}{(d-r)!} \sum_{|\mathbf{i}|=d-r} \Delta_{\mathbf{u}_1} \cdots \Delta_{\mathbf{u}_r} c_{\mathbf{i}} B_{\mathbf{i}}^{d-r}(\boldsymbol{\lambda}).$$

7.3.3 Joining polynomial pieces together

Consider now how we might build a spline over a triangulation in the plane from triangular BB polynomials. The main issue is how to fit together two triangular BB polynomials whose triangular parameter domains share a common edge. Suppose then that p is again the BB polynomial in (7.18), of degree $\leq d$, whose parameter domain is the triangle $T = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ and that we again denote by $\lambda_1, \lambda_2, \lambda_3$ the barycentric coordinates of the point $\mathbf{x} \in \mathbb{R}^2$ with respect to T. Then, for any point $\mathbf{v}_4 \in \mathbb{R}^2$ on the side of the edge $[\mathbf{v}_1, \mathbf{v}_2]$ opposite to $\mathbf{v}_3$, let q be a BB polynomial of degree $\leq d$, defined on the triangle $U = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_4]$,

$$q(\mathbf{x}) = \sum_{|\mathbf{i}|=d} \tilde{c}_{\mathbf{i}} B_{\mathbf{i}}^d(\tilde{\boldsymbol{\lambda}}), \tag{7.30}$$

where $\tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\lambda}_3$ are the barycentric coordinates of $\mathbf{x}$ with respect to U; see Figure 7.4. Suppose that the point $\mathbf{x}$ belongs to the common edge $[\mathbf{v}_1, \mathbf{v}_2]$. Then $\lambda_1 = \tilde{\lambda}_1$,

$\lambda_2 = \tilde{\lambda}_2$, and $\lambda_3 = \tilde{\lambda}_3 = 0$, and

$$p(\mathbf{x}) = \sum_{i_1+i_2=d} c_{i_1,i_2,0} \frac{d!}{i_1! i_2!} \lambda_1^{i_1} \lambda_2^{i_2},$$
$$q(\mathbf{x}) = \sum_{i_1+i_2=d} \tilde{c}_{i_1,i_2,0} \frac{d!}{i_1! i_2!} \lambda_1^{i_1} \lambda_2^{i_2}.$$

It follows that p and q join continuously on the common edge $[\mathbf{v}_1, \mathbf{v}_2]$ if and only if

$$c_{i_1,i_2,0} = \tilde{c}_{i_1,i_2,0}, \qquad i_1 + i_2 = d. \tag{7.31}$$

This is equivalent to saying that the control nets of p and q have the same boundary polygon on $[\mathbf{v}_1, \mathbf{v}_2]$. Under this condition we can define the spline

$$s(\mathbf{x}) := \begin{cases} p(\mathbf{x}), & \mathbf{x} \in T; \\ q(\mathbf{x}), & \mathbf{x} \in U. \end{cases} \tag{7.32}$$

Under what condition is s also C^1? It is C^1 if its directional derivative $D_{\mathbf{u}} s$ is continuous on the edge $[\mathbf{v}_1, \mathbf{v}_2]$ for any vector $\mathbf{u}$ transversal to the vector $\mathbf{v}_2 - \mathbf{v}_1$, i.e., for any non-zero vector $\mathbf{u}$ that is not parallel to $\mathbf{v}_2 - \mathbf{v}_1$. For such a $\mathbf{u}$ we compare the derivative formulas

$$D_{\mathbf{u}} p(\mathbf{x}) = d \sum_{|\mathbf{i}|=d-1} \Delta_{\mathbf{u},T} c_{\mathbf{i}} B_{\mathbf{i}}^{d-1}(\boldsymbol{\lambda}),$$
$$D_{\mathbf{u}} q(\mathbf{x}) = d \sum_{|\mathbf{i}|=d-1} \Delta_{\mathbf{u},U} \tilde{c}_{\mathbf{i}} B_{\mathbf{i}}^{d-1}(\tilde{\boldsymbol{\lambda}}).$$

In the case that $\mathbf{x} \in [\mathbf{v}_1, \mathbf{v}_2]$ these reduce to

$$D_{\mathbf{u}} p(\mathbf{x}) = d \sum_{i_1+i_2=d-1} \Delta_{\mathbf{u},U} c_{i_1,i_2,0} \frac{(d-1)!}{i_1! i_2!} \lambda_1^{i_1} \lambda_2^{i_2},$$
$$D_{\mathbf{u}} q(\mathbf{x}) = d \sum_{i_1+i_2=d-1} \Delta_{\mathbf{u},U} \tilde{c}_{i_1,i_2,0} \frac{(d-1)!}{i_1! i_2!} \lambda_1^{i_1} \lambda_2^{i_2}.$$

The condition for C^1 continuity is therefore

$$\Delta_{\mathbf{u},T} c_{i_1,i_2,0} = \Delta_{\mathbf{u},U} \tilde{c}_{i_1,i_2,0}, \qquad i_1 + i_2 = d - 1, \tag{7.33}$$

in addition to (7.31). Condition (7.33) can be rewritten in the form

$$\tilde{c}_{i_1,i_2,1} = \sum_{k=1}^{3} \alpha_k \, c_{(i_1,i_2,0)+\mathbf{e}_k}, \qquad i_1 + i_2 = d - 1, \tag{7.34}$$

where $\alpha_1, \alpha_2, \alpha_3$ are the barycentric coordinates of $\mathbf{v}_4$ with respect to T. This follows from using (7.31) and taking the difference of (7.28) and the corresponding equation for U. The coordinates α_1 and α_2 are non-negative but α_3 is negative.

Chapter 8

Blossoming

The *blossom* or *polar form* of a polynomial provides a simple way to derive the continuity conditions for joining BB polynomials together (as in Theorem 2.4). It also shows how a BB polynomial is subdivided into two BB polynomial pieces through de Casteljau's algorithm.

8.1 The three axioms

Every polynomial of degree $\leq d$,

$$p(x) = \sum_{i=0}^{d} a_i x^i, \qquad a_i \in \mathbb{R}, \tag{8.1}$$

has a unique d-variate *blossom*, also called a *polar form*, P. The blossom is the unique polynomial $P(x_1, x_2, \ldots, x_d)$ that is

(i) symmetric,

(ii) multi-affine, and

(iii) agrees with p on its diagonal.

By symmetric we mean that P has the same value if we swap any two variables x_i and x_j:

$$P(\ldots, x_i, \ldots, x_j, \ldots) = P(\ldots, x_j, \ldots, x_i, \ldots).$$

By multi-affine we mean that P is affine with respect to each variable: if

$$x_1 = (1 - \lambda)x + \lambda y,$$

then

$$P(x_1, x_2, \ldots, x_d) = (1-\lambda)P(x, x_2, \ldots, x_d) + \lambda P(y, x_2, \ldots, x_d),$$

and similarly for the other variables. By the diagonal property we understand that

$$P(x^{[d]}) = P(\underbrace{x, x, \ldots, x}_{d}) = p(x).$$

We will see shortly that the blossom is unique, but to show that it exists, just replace each term x^i in (8.1) by the expression

$$\sum_{1 \le k_1 < k_2 < \cdots < k_i \le d} x_{k_1} x_{k_2} \cdots x_{k_i} \Big/ \binom{d}{i}. \tag{8.2}$$

For example, the quadratic

$$p(x) = a_0 + a_1 x + a_2 x^2$$

has the bivariate blossom

$$P(x_1, x_2) = a_0 + a_1 \frac{x_1 + x_2}{2} + a_2 x_1 x_2,$$

and the cubic

$$p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

has the trivariate blossom

$$P(x_1, x_2, x_3) = a_0 + a_1 \frac{x_1 + x_2 + x_3}{3} + a_2 \frac{x_1 x_2 + x_1 x_3 + x_2 x_3}{3} + a_3 x_1 x_2 x_3.$$

Using (8.2), the symmetry and multi-affine properties of P are easily verified. The diagonal property holds because $\binom{d}{i}$ is the number of terms in the sum in (8.2).

We will use the blossom to derive various properties of BB polynomials without needing the explicit formula (8.2): it is the defining properties (i), (ii), (iii) that are important.

8.2 de Casteljau's algorithm

It turns out that we can express both the coefficients of a BB polynomial p and the points in the de Casteljau algorithm very simply in terms of the blossom of p. Consider again the BB polynomial

$$p(x) = \sum_{i=0}^{d} c_i B_i^d(\lambda), \tag{8.3}$$

with coefficients $c_i \in \mathbb{R}$, and $\lambda = (x-a)/(b-a)$, and recall de Casteljau's algorithm: we let $c_i^0 = c_i$, $i = 0, 1, \ldots, d$, and for $r = 1, \ldots, d$, we let

$$c_i^r = (1-\lambda)c_i^{r-1} + \lambda c_{i+1}^{r-1}, \qquad i = 0, 1, \ldots, d-r. \tag{8.4}$$

Then $c_0^d = p(x)$.

Theorem 8.1 *The d-variate blossom P of p is unique and the points in de Casteljau's algorithm can be expressed as*

$$c_i^r = P(a^{[d-r-i]}, b^{[i]}, x^{[r]}), \quad r = 0, 1, \ldots, d, \quad i = 0, 1, \ldots, d-r.$$

In particular, since the coefficients c_i of p are the de Casteljau points $c_i = c_i^0$ (for any value of x), we also have

$$c_i = P(a^{[d-i]}, b^{[i]}), \qquad i = 0, 1, \ldots, d. \tag{8.5}$$

Proof. For each $r = 0, 1, \ldots, d$ and $i = 0, 1, \ldots, d-r$, let

$$\alpha_i^r = P(a^{[d-r-i]}, b^{[i]}, x^{[r]}).$$

By the symmetry and multi-affine properties of P and the fact that

$$x = (1-\lambda)a + \lambda b,$$

it follows that

$$\alpha_i^r = (1-\lambda)\alpha_i^{r-1} + \lambda\alpha_{i+1}^{r-1}, \qquad r \geq 1, \tag{8.6}$$

and therefore,

$$\alpha_0^d = \sum_{i=0}^{d} \alpha_i^0 B_i^d(\lambda).$$

So by the diagonal property of P,

$$p(x) = \sum_{i=0}^{d} \alpha_i^0 B_i^d(\lambda).$$

Since this holds for all x and the B_i^d are linearly independent, it follows that $\alpha_i^0 = c_i$. It then also follows that $\alpha_i^r = c_i^r$ for all r and i.

It remains to show that P is unique. To this end, suppose P and Q are d-variate blossoms of p, and let $x_1, \ldots, x_d \in \mathbb{R}$. We will show that

$$P(x_1, \ldots, x_d) = Q(x_1, \ldots, x_d). \tag{8.7}$$

For each $r = 0, 1, \ldots, d$ and $i = 0, 1, \ldots, d - r$, let

$$\alpha_i^r = P(a^{[d-r-i]}, b^{[i]}, x_1, \ldots, x_r),$$
$$\beta_i^r = Q(a^{[d-r-i]}, b^{[i]}, x_1, \ldots, x_r).$$

By the symmetry and multi-affine properties of P and Q and since

$$x_r = (1 - \lambda_r)a + \lambda_r b,$$

where $\lambda_r = (x_r - a)/(b - a)$, we have the recursions

$$\alpha_i^r = (1 - \lambda_r)\alpha_i^{r-1} + \lambda_r \alpha_{i+1}^{r-1},$$
$$\beta_i^r = (1 - \lambda_r)\beta_i^{r-1} + \lambda_r \beta_{i+1}^{r-1}.$$

By (8.5), $\alpha_i^0 = \beta_i^0 = c_i$, and so by induction on r, $\alpha_i^r = \beta_i^r$ for all r and i. Hence, $\alpha_0^d = \beta_0^d$, which gives (8.7). □

For example, the coefficients of de Casteljau's algorithm in the cubic case are:

$$\begin{array}{cccc} P(a,a,a) & P(a,a,x) & P(a,x,x) & P(x,x,x) \\ P(a,a,b) & P(a,b,x) & P(b,x,x) & \\ P(a,b,b) & P(b,b,x) & & \\ P(b,b,b) & & & \end{array}$$

8.3 Joining BB polynomials together

One application of the blossom is the derivation of the continuity conditions for joining BB polynomials together smoothly, i.e., a proof of Theorem 2.4. Consider again the spline

$$s(x) = \begin{cases} p(x), & a \le x < b; \\ q(x), & b \le x < c; \end{cases}$$

with p as in (8.3) and where

$$q(x) = \sum_{i=0}^{d} e_i B_i^d(\mu),$$

for coefficients $e_i \in \mathbb{R}$, with $\mu = (x - b)/(c - b)$.

Let us define

$$\nu = \frac{c - a}{b - a}.$$

To obtain the condition for C^r continuity of s at $x = b$, observe that the derivatives of q of order $\leq r$ at $x = b$ depend only $e_0, e_1, \ldots, e_r$. Thus we just need to express p as a BB polynomial with respect to the interval $[b, c]$ and equate the first $r + 1$ of its coefficients with $e_0, e_1, \ldots, e_r$. This motivates us to find the coefficients $\alpha_0, \ldots, \alpha_d$ in the representation

$$p(x) = \sum_{i=0}^{d} \alpha_i B_i^d(\mu),$$

with $\mu = (x - b)/(c - b)$. We can express these using the blossom of p:

$$\alpha_i = P(b^{[d-i]}, c^{[i]}).$$

On the other hand, from the de Casteljau algorithm applied to p on the interval $[a, b]$ with $x = c$ we also have

$$P(b^{[d-i]}, c^{[i]}) = c_{d-i}^i,$$

Moreover, by (2.5),

$$c_{d-i}^i = \sum_{j=0}^{i} c_{d-i+j} B_j^i(\nu),$$

and we conclude that

$$\alpha_i = \sum_{j=0}^{i} c_{d-i+j} B_j^i(\nu), \qquad i = 0, 1, \ldots, d.$$

Thus, the condition for C^r continuity of s is that

$$e_i = \sum_{j=0}^{i} c_{d-i+j} B_j^i(\nu), \qquad i = 0, 1, \ldots, r,$$

which proves Theorem 2.4.

8.4 Subdivision

Another application of the blossom is to subdivision. Consider again the BB polynomial $p(x)$ in (8.3), and consider the two outer diagonal rows of points in the de Casteljau algorithm (8.4), that is

$$c_0^0, c_0^1, \ldots, c_0^d, \qquad \text{and} \qquad c_0^d, c_1^{d-1}, \ldots, c_d^0.$$

By Theorem 8.1, these can be expressed in terms of P as

$$c_0^i = P(a^{[d-i]}, x^{[i]}), \qquad \text{and} \qquad c_i^{d-i} = P(b^{[i]}, x^{[d-i]}).$$

Therefore, again by Theorem 8.1, but applied to the sub-interval $[a, x]$, we see that the c_0^i are the coefficients of p when represented as a BB polynomial with respect to $[a, x]$, i.e.,

$$p(y) = \sum_{i=0}^{d} c_0^i B_i^d(\mu),$$

where $\mu = (y-a)/(x-a)$, and similarly, the c_i^{n-i} are the coefficients of p represented as a BB polynomials with respect to the sub-interval $[x, b]$,

$$p(y) = \sum_{i=0}^{d} c_i^{d-i} B_i^d(\nu),$$

where $\nu = (y-x)/(b-x)$.

On subdividing p repeatedly, one can obtain the BB polynomial coefficients of p on any number of adjacent sub-intervals $[a_0, a_1]$, $[a_1, a_2]$, ..., $[a_{k-1}, a_k]$, with $a_0 = a$ and $a_k = b$. Together, their individual control polygons form a composite control polygon for p with $kd + 1$ vertices. As the sub-intervals get smaller, the composite polygon tends to get closer to p. Thus subdivison offers an alternative way of plotting p. Instead of plotting p by sampling it at several parameter values $t_j \in [a, b]$ and plotting the polygon formed by the points $p(t_j)$, we can alternatively subdivide p a few times and plot its composite control polygon.

8.5 Generalized de Casteljau algorithm

We can also *compute* the blossom of a BB polynomial by generalizing de Casteljau's algorithm in a simple way; in fact we already did this in the latter part of the proof of Theorem 8.1. We will see later that computing a blossom is useful for converting a BB polynomial to B-spline form.

Suppose that $p(x)$ is the BB polynomial in (8.3). We can compute its blossom $P(x_1, \ldots, x_d)$ for any $x_1, \ldots, x_d \in \mathbb{R}$ using the the following *generalized de Casteljau algorithm*. We set $c_i^0 = c_i$, $i = 0, 1, \ldots, d$, and for $r = 1, \ldots, d$, we set

$$c_i^r = (1 - \lambda_r) c_i^{r-1} + \lambda_r c_{i+1}^{r-1}, \qquad i = 0, 1, \ldots, d - r. \tag{8.8}$$

where $\lambda_r = (x_r - a)/(b - a)$. As in the latter part of the proof of Theorem 8.1, one can show by induction on r that

$$c_i^r = P(a^{[d-r-i]}, b^{[i]}, x_1, \ldots, x_r),$$

and so, in particular, the last coefficient computed is $c_0^d = P(x_1, \ldots, x_d)$. For example, the coefficients of the algorithm in the cubic case are:

$$\begin{array}{llll} P(a,a,a) & P(a,a,x_1) & P(a,x_1,x_2) & P(x_1,x_2,x_3) \\ P(a,a,b) & P(a,b,x_1) & P(b,x_1,x_2) & \\ P(a,b,b) & P(b,b,x_1) & & \\ P(b,b,b) & & & \end{array}$$

By the symmetric property of the blossom P, this algorithm will yield the value $P(x_1, \ldots, x_d)$ regardless of the ordering of the points $x_1, \ldots, x_d$.

8.6 Blossom of a polynomial in B-spline form

We have seen how the coefficients of de Casteljau's algorithm can be expressed in terms of the blossom of the polynomial in question. This property generalizes to the coefficients in the de Boor algorithm (Algorithm 2 of Subsection 3.9.2). Consider the spline function

$$s(x) = \sum_{i=1}^{n} c_i B_{i,d,\mathbf{t}}(x),$$

for some knot vector $\mathbf{t} = (t_1, t_2, \ldots, t_{n+d+1})$, and suppose $[t_\mu, t_{\mu+1}]$ is a non-degenerate knot interval for some $\mu \in \{d+1, \ldots, n\}$. Then the polynomial piece $s_\mu = s|_{[t_\mu, t_{\mu+1}]}$ is given by

$$s_\mu(x) = \sum_{i=\mu-d}^{\mu} c_i B_{i,d,\mathbf{t}}(x), \qquad x \in [t_\mu, t_{\mu+1}]. \tag{8.9}$$

Let us recall de Boor's algorithm (see (3.25)): we set $c_i^0 = c_i$, $i = \mu - d, \ldots, \mu$, and for $r = 1, \ldots, d$ and $i = \mu - d + r, \ldots, \mu$, we set

$$c_i^r = (1 - \lambda_{i,r}) c_{i-1}^{r-1} + \lambda_{i,r} c_i^{r-1}, \tag{8.10}$$

where $\lambda_{i,r} = (x - t_i)/(t_{i+d-r+1} - t_i)$. Then $c_0^d = s_\mu(x)$.

Theorem 8.2 *The coefficients c_i^r in the de Boor algorithm can be expressed as*

$$c_i^r = P(t_{i+1}, \ldots, t_{i+d-r}, x^{[r]}), \quad r = 1, \ldots, d, \quad i = \mu - d + r, \ldots, \mu,$$

where P is the blossom of s_μ.

In particular, the coefficients of s_μ are

$$c_i = P(t_{i+1}, \ldots, t_{i+d}), \qquad i = \mu - d, \ldots, \mu. \tag{8.11}$$

Proof. The proof is similar to that of Theorem 8.1 except that we now define

$$\alpha_i^r = P(t_{i+1}, \ldots, t_{i+d-r}, x^{[r]}), \quad r = 1, \ldots, d, \quad i = \mu - d + r, \ldots, \mu.$$

By the symmetry and multi-affine properties of P and the fact that

$$x = (1 - \lambda_{i,r})t_i + \lambda_{i,r} t_{i+d-r+1},$$

it follows that

$$\alpha_i^r = (1 - \lambda_{i,r})\alpha_{i-1}^{r-1} + \lambda_{i,r}\alpha_i^{r-1}, \qquad r \geq 1,$$

and therefore,

$$\alpha_\mu^d = \sum_{i=d-\mu}^{\mu} \alpha_i^0 B_{i,d,\mathbf{t}}(x).$$

So by the diagonal property of P,

$$p(x) = \sum_{i=d-\mu}^{\mu} \alpha_i^0 B_{i,d,\mathbf{t}}(x).$$

By the linear independence of $B_{\mu-d,d,\mathbf{t}}, \ldots, B_{\mu,d,\mathbf{t}}$, it follows that $\alpha_i^0 = c_i$, and it then follows that $\alpha_i^r = c_i^r$ for all r and i. □

As an example, let $d = 3$, $\mu = 4$, and $t_4 < t_5$, and let P be the blossom of the cubic piece s_4. Then the coefficients in de Boor's algorithm for a point x are:

$$\begin{array}{llll} P(t_2,t_3,t_4) & P(t_3,t_4,x) & P(t_4,x,x) & P(x,x,x) \\ P(t_3,t_4,t_5) & P(t_4,t_5,x) & P(t_5,x,x) & \\ P(t_4,t_5,t_6) & P(t_5,t_6,x) & & \\ P(t_5,t_6,t_7) & & & \end{array}$$

Similar to the generalized de Casteljau algorithm, there is also a *generalized de Boor algorithm* which computes the blossom of s_μ at any points $x_1, \ldots, x_d$. We set $c_i^0 = c_i$, $i = \mu - d, \ldots, \mu$, and for $r = 1, \ldots, d$ and $i = \mu - d + r, \ldots, \mu$, we set

$$c_i^r = (1 - \lambda_{i,r})c_{i-1}^{r-1} + \lambda_{i,r}c_i^{r-1}, \tag{8.12}$$

where $\lambda_{i,r} = (x_r - t_i)/(t_{i+d-r+1} - t_i)$. One can show by induction on r that

$$c_i^r = P(t_{i+1}, \ldots, t_{i+d-r}, x_1, \ldots, x_r),$$

and so, in particular, the last coefficient computed is $c_0^d = P(x_1, \ldots, x_d)$. For example, the coefficients in this generalized algorithm for s_4 of the previous example are:

$$\begin{array}{llll} P(t_2,t_3,t_4) & P(t_3,t_4,x_1) & P(t_4,x_1,x_2) & P(x_1,x_2,x_3) \\ P(t_3,t_4,t_5) & P(t_4,t_5,x_1) & P(t_5,x_1,x_2) & \\ P(t_4,t_5,t_6) & P(t_5,t_6,x_1) & & \\ P(t_5,t_6,t_7) & & & \end{array}$$

8.7 Conversion of polynomial and spline representations

Suppose we are given the BB coefficients of a polynomial p with respect to an interval $[a, b]$. We have seen how we can use de Casteljau's algorithm to obtain the BB coefficients of p with respect to a sub-interval of the form $[a, c]$ or $[c, b]$. If we apply the algorithm a second time we will then obtain the BB coefficients of p with respect to any arbitrary interval $[c, d]$.

We might also need to convert a BB polynomial to B-spline form with respect to some chosen knot vector, or convert a spline in one B-spline representation into another. To do this we can make use of the blossoming theory we have developed.

8.7.1 Converting a BB polynomial to B-spline form

We saw earlier, in Subsection 4.2.1, how we can represent a polynomial p given in monomial form

$$p(x) = \sum_{i=0}^{d} a_i x^i,$$

into a spline over any knot vector, by using Marsden's identity. But suppose we are given p in the BB form

$$p(x) = \sum_{i=0}^{d} c_i B_i^d(\lambda),$$

with $\lambda = (x-a)/(b-a)$ for some interval $[a, b]$. How can we represent p as a spline in this case, i.e., how do we find the coefficients $\tilde{c}_1, \ldots, \tilde{c}_n$ in the representation

$$p(x) = \sum_{i=1}^{n} \tilde{c}_i B_{i,d,\mathbf{t}}(x), \qquad x \in [t_{d+1}, t_{n+1}],$$

with respect to some chosen knot vector $\mathbf{t} = (t_1, t_2, \ldots, t_{n+d+1})$? To do this we can use blossoming. We know that

$$\tilde{c}_i = P(t_{i+1}, \ldots, t_{i+d}), \qquad i = 1, \ldots, n,$$

where P is the d-variate blossom of p. Thus we can compute the coefficients $\tilde{c}_i$ using the generalized de Casteljau algorithm for p to compute $P(t_{i+1}, \ldots, t_{i+d})$ for each i. We just set $(x_1, \ldots, x_d) = (t_{i+1}, \ldots, t_{i+d})$ in (8.8).

8.7.2 Converting one B-spline form to another

Similarly, we can convert a polynomial given in B-spline form into some other B-spline form. Thus, consider the polynomial s_μ of (8.9), whose coefficients are $c_{\mu-d}, \ldots, c_\mu$. We might want to represent s_μ in the alternative form

$$s_\mu(x) = \sum_{i=\nu-d}^{\nu} \tilde{c}_i B_{i,d,\boldsymbol{\tau}}(x),$$

with respect to some new knot vector $\boldsymbol{\tau} = (\tau_1, \tau_2, \ldots, \tau_{m+d+1})$ for $\nu < \nu + 1$ and $\nu \in \{d+1, \ldots, n\}$. By the theory we have developed,

$$\tilde{c}_i = P(\tau_{i+1}, \ldots, \tau_{i+d}), \qquad i = \nu - d, \ldots, \nu,$$

where P is the d-variate blossom of s_μ, and moreover, we can compute the value $P(\tau_{i+1}, \ldots, \tau_{i+d})$ for each i from the generalized form of de Boor's algorithm, setting $(x_1, \ldots, x_d) = (\tau_{i+1}, \ldots, \tau_{i+d})$ in (8.12).

Consider now spline refinement. Suppose we want to represent a spline s over a given knot vector $\mathbf{t}$ as a spline over a refined knot vector $\boldsymbol{\tau}$, i.e., such that $\mathbf{t} \subset \boldsymbol{\tau}$. Thus we have

$$s = \sum_{i=1}^{n} c_i B_{i,d,\mathbf{t}} = \sum_{i=1}^{m} b_i B_{i,d,\boldsymbol{\tau}},$$

and we want to find the new coefficints b_i from the old ones, c_i. We saw in Chapter 5 how we can do this by adding one knot at a time, and using Boehm's algorithm to find the new coefficients at each step. Using blossoming, we now have an alternative method. We can express b_i as

$$b_i = P(\tau_{i+1}, \ldots, \tau_{i+d}),$$

where P is the blossom of any polynomial piece $p = s|_{[\tau_\nu, \tau_{\nu+1}]}$ such that $\tau_\nu < \tau_{\nu+1}$ and $\nu - d \le i \le \nu$. For any such choice of ν there is a unique μ such that

$$t_\mu \le \tau_\nu < \tau_{\nu+1} \le t_{\mu+1},$$

in which case

$$p(x) = \sum_{i=\mu-d}^{\mu} c_i B_{i,d,\mathbf{t}}(x),$$

and we can therefore compute $P(\tau_{i+1}, \ldots, \tau_{i+d})$ from the generalized de Boor algorithm, setting $(x_1, \ldots, x_d) = (\tau_{i+1}, \ldots, \tau_{i+d})$ in (8.12).

8.8 Blossoms of bivariate polynomials

As for polynomials of one variable, there is a blossom for polynomials of several variables. Consider the bivariate polynomial,

$$p(\mathbf{x}) = p(x, y) = \sum_{i+j \le d} a_{i,j} x^i y^j. \tag{8.13}$$

There is a unique d-variate blossom P for p in which each variable is a point in $\mathbb{R}^2$. The blossom, $P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d)$, $\mathbf{x}_k \in \mathbb{R}^2$, is, as before, uniquely defined by three properties: that it is (i) symmetric, (ii) multi-affine, and (iii) agrees with $p(\mathbf{x})$ on its diagonal. The blossom can be obtained from p by replacing the term $x^i y^j$ in (8.13) by the expression

$$\sum x_{k_1} x_{k_2} \cdots x_{k_i} y_{\ell_1} y_{\ell_2} \cdots y_{\ell_j} \Big/ \left(\frac{d!}{i!j!(d-i-j)!} \right). \tag{8.14}$$

Here the sum is over all pairs of sequences

$$1 \le k_1 < k_2 < \cdots < k_i \le d, \qquad 1 \le \ell_1 < \ell_2 < \cdots < \ell_j \le d,$$

whose elements are pairwise distinct, i.e., such that $\ell_\beta \ne k_\alpha$. For example, the quadratic

$$p(\mathbf{x}) = a_0 + a_{10} x + a_{01} y + a_{20} x^2 + a_{11} xy + a_{02} y^2$$

has the bivariate blossom

$$P(\mathbf{x}_1, \mathbf{x}_2) = a_0 + a_{10} \frac{x_1 + x_2}{2} + a_{01} \frac{y_1 + y_2}{2} + a_{20} x_1 x_2 + a_{11} \frac{x_1 y_2 + x_2 y_1}{2} + a_{02} y_1 y_2.$$

The monomials $p(\mathbf{x}) = xy$ and $p(\mathbf{x}) = x^2 y$ have the trivariate blossoms

$$P(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \frac{1}{6}(x_1 y_2 + x_1 y_3 + x_2 y_1 + x_2 y_3 + x_3 y_1 + x_3 y_2),$$

and

$$P(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = \frac{1}{3}(x_1 x_2 y_3 + x_1 x_3 y_2 + x_2 x_3 y_1),$$

respectively. Like in the univariate case, the symmetry and multi-affine properties of P are easily verified from (8.14), and the diagonal property holds because the number of terms in the sum is $d!/(i!j!(d-i-j)!)$.

Recall now the triangular BB polynomial p of (7.18) and the corresponding de Casteljau algorithm (7.23) for evaluating $p(\mathbf{x})$ for some point $\mathbf{x} \in \mathbb{R}^2$. In analogy to the proof of Theorem 8.1, one can show that the d-variate blossom P of p is unique and the coefficients in de Casteljau's algorithm can be expressed as

$$c_{\mathbf{i}}^r = P(\mathbf{v}_1^{[i_1]}, \mathbf{v}_2^{[i_2]}, \mathbf{v}_3^{[i_3]}, \mathbf{x}^{[r]}), \qquad r = 0, 1, \ldots, d, \quad |\mathbf{i}| = d - r, \tag{8.15}$$

and so, as a special case,

$$c_{\mathbf{i}} = P(\mathbf{v}_1^{[i_1]}, \mathbf{v}_2^{[i_2]}, \mathbf{v}_3^{[i_3]}), \qquad |\mathbf{i}| = d. \tag{8.16}$$

In analogy to the univariate case, since

$$c_{i_1,i_2,0}^{i_3} = P(\mathbf{v}_1^{[i_1]}, \mathbf{v}_2^{[i_2]}, \mathbf{x}^{[i_3]}), \qquad |\mathbf{i}| = d,$$

these are the BB coefficients of p with respect to the sub-triangle $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{x}]$, more precisely,

$$p(\mathbf{y}) = \sum_{|\mathbf{i}|=d} c_{i_1,i_2,0}^{i_3} B_{\mathbf{i}}^d(\tilde{\boldsymbol{\lambda}}),$$

where $\tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\lambda}_3$ are the barycentric coordinates of $\mathbf{y}$ with respect to the triangle $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{x}]$.

Next consider the conditions for two triangular BB polynomials to join with C^r continuity on a common edge. Recalling the spline s we defined in (7.32), in analogy to the univariate case, the condition for p and q to join with C^r continuity on the common edge $[\mathbf{v}_1, \mathbf{v}_2]$ is that

$$\tilde{c}_{\mathbf{i}} = \beta_{\mathbf{i}}, \qquad |\mathbf{i}| = d, \quad i_3 \le r$$

where the $\beta_{\mathbf{i}}$ are the BB coefficients of p with respect to the triangle U. These can be computed from the de Casteljau algorithm for p with respect to T with $\mathbf{x} = \mathbf{v}_4$, and the C^r condition thus reduces to

$$\tilde{c}_{\mathbf{i}} = \sum_{|\mathbf{j}|=i_3} c_{(i_1,i_2,0)+\mathbf{j}} B_{\mathbf{j}}^{i_3}(\boldsymbol{\alpha}), \qquad |\mathbf{i}| = d, \quad i_3 \le r,$$

where $\alpha_1, \alpha_2, \alpha_3$ are the barycentric coordinates of $\mathbf{v}_4$ with respect to T. This condition generalizes the C^1 condition of (7.31) and (7.34) in which $i_3 = 0$ and $i_3 = 1$ respectively.

Bibliography

[1] S. D. Conte and C. de Boor, *Elementary numerical analysis*, McGraw-Hill, 1980.

[2] C. de Boor, *A practical guide to splines*, Springer-Verlag, 1978.

[3] G. Farin, *Curves and surfaces for computer aided geometric design*, Academic Press, 1990.

[4] E. Isaacson and H. B. Keller, *Analysis of numerical methods*, Dover, 1994.

[5] M.-J. Lai and L. L. Schumaker, *Spline functions on triangulations*, Cambridge University Press, Cambridge, 2007.

[6] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-spline techniques*, Springer, 2002.