LECTURE NOTES ON SPLINES

Michael S. Floater

Contents

Preface

This is a collection of working lecture notes about spline theory, with a view to data fitting and geometric modelling. We consider a spline to be a piecewise polynomial. In the first chapter we study Bernstein-Bézier (BB) polynomials and Bézier curves. In the second chapter we see how some important examples of splines can be contructed by representing each polynomial piece in BB form and enforcing conditions on the pieces that guarantee that they fit together continuously or smoothly.

Chapter 1

Bernstein-Bézier polynomials

In this chapter we study 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

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

The first few examples are

$$B_0^0(x) = 1,$$

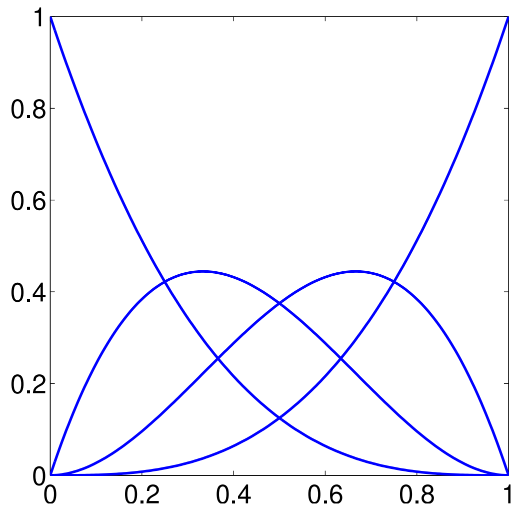

Figure 1.1: The cubic Bernstein basis polynomials

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

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

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

Let us now show that the Bernstein basis polynomials of degree d do indeed form a basis for π_d.

Theorem 1.1.1 *The set*

$$\mathcal{B}^d := \{B_i^d : i = 0, 1, \dots, d\}$$

of Bernstein basis polynomials is a basis for the space of polynomials π_d.

Since the number of elements of $\mathcal{B}^d$ 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 $\mathcal{B}^d$, 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 1.1.2 *For $j = 0, 1, \ldots, d$,*

$$x^j = \frac{1}{\binom{d}{j}} \sum_{i=j}^{d} \binom{i}{j} B_i^d(x). \tag{1.3}$$

Proof. We use the binomial theorem again:

$$\begin{aligned} x^j = x^j(x + (1-x))^{d-j} &= x^j \sum_{i=0}^{d-j} \binom{d-j}{i} x^i (1-x)^{d-j-i} \\ &= \sum_{i=j}^{d} \binom{d-j}{i-j} x^i (1-x)^{d-i} = \sum_{i=j}^{d} \frac{\binom{d-j}{i-j}}{\binom{d}{i}} B_i^d(x), \end{aligned} \tag{1.4}$$

which can be rewritten as (1.3). □

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 1.2.1 *For $i = 0, 1, \ldots, d$,*

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

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 same as (1.5). In the cases $i = 0$ and $i = d$ we have

$$\begin{aligned} B_0^d(x) &= (1-x)^d = (1-x)B_0^{d-1}(x), \\ B_d^d(x) &= x^d = xB_{d-1}^{d-1}(x), \end{aligned}$$

which agree with (1.5) 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

$$B_i^r = xB_{i-1}^{r-1} + (1-x)B_i^{r-1}, \quad i = 0, 1, \ldots, r. \tag{1.6}$$

The scheme is illustrated below. In each column, each value is computed from two values from the previous column.

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

1.3 Differentiation

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

Lemma 1.3.1 *For $d \geq 1$ and for $i = 0, 1, \ldots, d$,*

$$(B_i^d)'(x) = d\big(B_{i-1}^{d-1}(x) - B_i^{d-1}(x)\big). \tag{1.7}$$

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 (1.7). In the cases $i = 0$ and $i = d$,

$$\begin{aligned} (B_0^d)'(x) &= -d(1-x)^{d-1} = -dB_0^{d-1}(x), \\ (B_d^d)'(x) &= dx^{d-1} = dB_{d-1}^{d-1}(x), \end{aligned}$$

which agree with (1.7) 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 1.4.1 *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 (1.7),

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

for $i = 1, \ldots, d$. Integrating this equation with respect 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

Let $[a, b]$ be some real interval. Any point $x \in [a, b]$ can be expressed as a convex combination of the endpoints a and b,

$$x = (1 - \lambda)a + \lambda b, \tag{1.8}$$

where

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

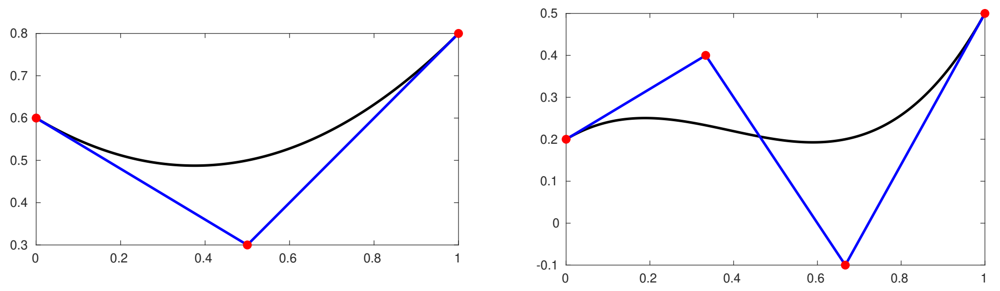

Figure 1.2: Quadratic and cubic BB polynomials on $[0, 1]$.

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

$$p(x) = \sum_{i=0}^{d} c_i B_i^d(\lambda), \tag{1.9}$$

a *Bernstein-Bézier polynomial* or *BB polynomial*. The coefficients $c_i \in \mathbb{R}$ are the *BB coefficients* of p with respect to $[a, b]$. In the special case that $[a, b] = [0, 1]$, a BB polynomial is simply

$$p(x) = \sum_{i=0}^{d} c_i B_i^d(x). \tag{1.10}$$

Although we are mainly interested in values of p for $x \in [a, b]$, we note that both λ and $p(x)$ are well defined for all $x \in \mathbb{R}$.

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

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

The shape of p relects, to a large extent, the shape of its control polygon. Figure 1.2 shows quadratic and cubic BB polynomials on $[0, 1]$ 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*,

$$p(a) = c_0, \qquad p(b) = c_d. \tag{1.11}$$

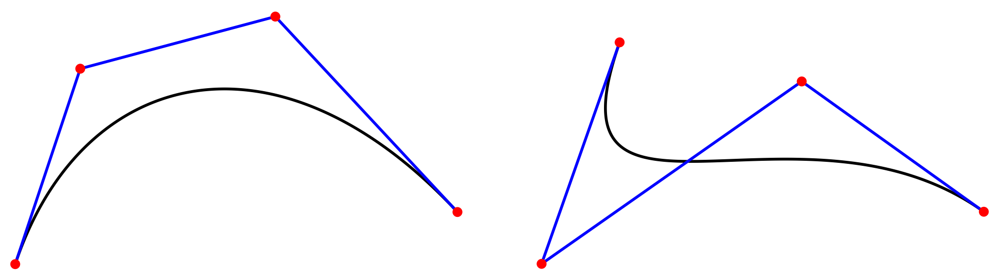

Figure 1.3: Cubic Bézier curves

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

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

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

$$\min_{0 \le i \le d} c_i \le p(x) \le \max_{0 \le i \le d} c_i, \qquad a \le x \le b. \tag{1.12}$$

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,

$$\mathbf{p}(x) = \sum_{i=0}^{d} \mathbf{c}_i B_i^d(\lambda), \tag{1.13}$$

where, as before, $\lambda = (x-a)/(b-a)$. This is a parametric polynomial curve, and if we restrict x to the interval $[a,b]$, then $[a,b]$ becomes the parameter domain of $\mathbf{p}$ and the curve $\mathbf{p}$ is a mapping, $\mathbf{p} : [a,b] \to \mathbb{R}^k$. 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. The shape of the curve $\mathbf{p}$ is independent of the choice of parameter domain $[a,b]$, since it is a reparameterization of the Bézier curve

$$\mathbf{p}(x) = \sum_{i=0}^{d} \mathbf{c}_i B_i^d(x). \tag{1.14}$$

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,

$$\mathbf{p}(a) = \mathbf{c}_0, \qquad \mathbf{p}(b) = \mathbf{c}_d.$$

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 $\mathbf{c}_0, \dots, \mathbf{c}_n$. 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

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

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

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

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

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

This same convex hull property also applies to each the d coordinates of $\mathbf{p}$ separately, and so $\mathbf{p}$ restricted to $[a,b]$ 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 $c_{i1}, \dots, c_{ik}$,

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

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 $p(x)$ in (1.9). First we compute $\lambda = (x-a)/(b-a)$. Then one way to proceed is to evaluate the Bernstein basis polynomials B_i^d at λ, by the recursion (1.6), and then apply the formula in (1.9). 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

$$c_i^r = (1-\lambda)c_i^{r-1} + \lambda c_{i+1}^{r-1}, \qquad i = 0, 1, \ldots, d-r. \tag{1.15}$$

Theorem 1.7.1 *The last value computed, c_0^d, is the value of $p(x)$ in (1.9).*

Proof. Consider the first step of the algorithm. By the recurrence (1.6),

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

where we used the convention that $B_{-1}^{d-1} = B_d^{d-1} = 0$, and hence, by (1.15),

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

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

$$p(x) = \sum_{i=0}^{d-r} c_i^r B_i^{d-r}(\lambda). \tag{1.16}$$

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

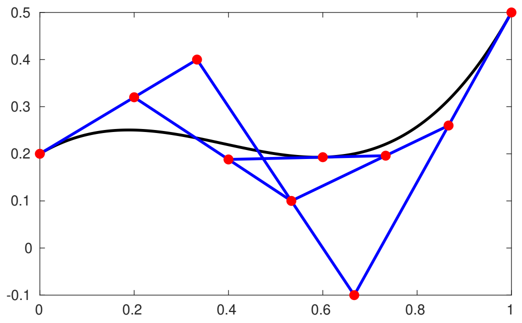

Figure 1.4: de Casteljau algorithm, $x = 0.6$

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 1.7.2 *The value c_i^r in the de Casteljau algorithm (1.15), viewed as a function of x, is the BB polynomial*

$$c_i^r = \sum_{j=0}^{r} c_{i+j} B_j^r(\lambda). \tag{1.17}$$

Proof. We proceed by induction on $r \geq 0$. The statement clearly holds when $r = 0$. For $r \geq 1$, we may suppose that (1.17) holds when r is replaced by $r-1$. Thus, from (1.15), we find

$$\begin{aligned}
c_i^r = (1-\lambda)c_i^{r-1} + \lambda c_{i+1}^{r-1} &= (1-\lambda)\sum_{j=0}^{r-1} c_{i+j} B_j^{r-1}(\lambda) + \lambda \sum_{j=0}^{r-1} c_{i+1+j} B_j^{r-1}(\lambda) \\
&= \sum_{j=0}^{r} c_{i+j}\left((1-\lambda)B_j^{r-1}(\lambda) + \lambda B_{j-1}^{r-1}(\lambda)\right) \\
&= \sum_{j=0}^{r} c_{i+j} B_j^r(\lambda),
\end{aligned}$$

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 (1.9) at a point x.

Theorem 1.8.1 *The first derivative of the BB polynomial p in (1.9) is*

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

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

Proof. By the chain rule,

$$\frac{d}{dx} B_i^d(\lambda) = (B_i^d)'(\lambda) \frac{d\lambda}{dx} = (B_i^d)'(\lambda)/h.$$

Then, using Lemma 1.3.1,

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

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} on $[a, b]$ whose coefficients are $(d/h)\Delta c_i$, $i = 0, 1, \ldots, d-1$. Thus by the endpoint property (1.11), it follows that

$$p'(a) = \frac{d}{h} \Delta c_0, \qquad p'(b) = \frac{d}{h} \Delta c_{d-1}. \tag{1.18}$$

By (1.12), we also have

$$\frac{d}{h} \min_{0 \le i \le d-1} \Delta c_i \le p'(x) \le \frac{d}{h} \max_{0 \le i \le d-1} \Delta c_i, \qquad a \le x \le b. \tag{1.19}$$

1.9 Higher order derivatives

We obtain higher order derivatives of p by applying Theorem 1.8.1 repeatedly, and by defining the r-th order forward difference of the coefficients by the recursion

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

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

$$\Delta^r c_0 = \sum_{i=0}^{r} \binom{r}{i} (-1)^{r-i} c_i, \tag{1.20}$$

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

Corollary 1.9.1 *The r-th derivative of the BB polynomial p in (1.9) is*

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

Thus the derivatives of p at the endpoints of $[a, b]$ have simple expressions in terms of the coefficients:

Corollary 1.9.2 *The r-th order endpoint derivatives of the BB polynomial p in (1.9) are*

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

1.10 Integration

There is a simple formula for the integral of a BB polynomial over its domain.

Theorem 1.10.1 *The integral of the BB polynomial p in (1.9) is the length h of the interval $[a, b]$ times the average of its coefficients,*

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

Proof. Integrating p in (1.9) gives

$$\int_a^b p(x)\,dx = \sum_{i=0}^{d} c_i \int_a^b B_i^d(\lambda)\,dx. \tag{1.21}$$

Then, by changing the variable of integration from x to λ,

$$\int_a^b B_i^d(\lambda)\,dx = h\int_0^1 B_i^d(\lambda)\,d\lambda,$$

and therefore, by Theorem 1.4.1,

$$\int_a^b B_i^d(\lambda)\,dx = \frac{h}{d+1}.$$

Substituting this into (1.21) gives the result. □

1.11 Exercises

1. It is sometimes necessary to convert a polynomial in BB form to monomial form. Consider a quadratic BB polynomial on the interval $[0,1]$, i.e., a polynomial
$$p(x) = c_0(1-x)^2 + 2c_1x(1-x) + c_2x^2.$$
Express p in the monomial form
$$p(x) = a_0 + a_1x + a_2x^2.$$

2. Consider a polynomial $p(x)$ of degree $\le d$, for arbitrary d. Show that if
$$p(x) = \sum_{j=0}^{d} a_j x^j = \sum_{i=0}^{d} c_i B_i^d(x),$$
then
$$a_j = \binom{d}{j}\Delta^j c_0.$$
Hint: use a Taylor approximation to p to show that $a_j = p^{(j)}(0)/j!$.

3. We might also want to convert a polynomial from monomial form to BB form. Using equation (1.4), show that in the notation of the previous question,
$$c_i = \frac{1}{\binom{d}{i}} \sum_{j=0}^{i} \binom{d-j}{i-j} a_j.$$

4. Show that the graph, $g(x) = (x, p(x))$ of the BB polynomial p in (1.9) is a Bézier curve in $\mathbb{R}^2$, with control points (ξ_i, c_i), $i = 0, 1, \dots, d$. Hint: express x as a linear combination of the $B_i^d(\lambda)$.

5. Show that the tangent vector $\mathbf{p}'(x)$ of the Bézier curve in (1.9) lies in the convex cone of the vectors $\Delta\mathbf{c}_i$, i.e., in
$$\operatorname{cone}(\Delta\mathbf{c}_0, \dots, \Delta\mathbf{c}_{d-1}) = \left\{ \sum_{i=0}^{d-1} \mu_i \Delta\mathbf{c}_i : \mu_1, \dots, \mu_{d-1} \geq 0 \right\}.$$

6. Show that the first derivative of p in (1.9) can be expressed (and computed) as
$$p'(x) = \frac{d}{h}(c_1^{d-1} - c_0^{d-1}),$$
where c_0^{d-1}, c_1^{d-1} are the points of order $d-1$ in de Casteljau's algorithm (1.15).

7. Show that the length of the Bézier curve $\mathbf{p}$ in (1.14) is bounded by the length of its control polygon,
$$\operatorname{length}(\mathbf{p}) \leq \sum_{i=0}^{d-1} \|\Delta c_i\|.$$

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 Linear spline

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

$$g(x_i) = y_i, \qquad i = 1, 2, \ldots, m. \tag{2.1}$$

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

$$g_i(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i} y_i + \frac{x - x_i}{x_{i+1} - x_i} y_{i+1}. \tag{2.2}$$

We then define g by $g(x) = g_i(x)$ for $x_i \leq x \leq x_{i+1}$. 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:

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

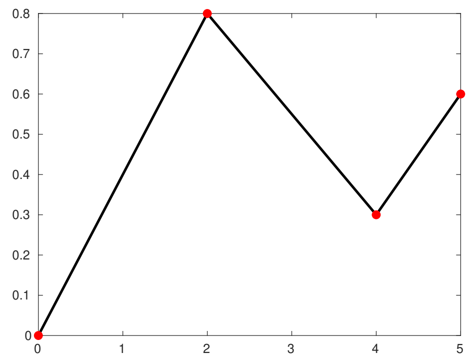

Figure 2.1: Piecewise linear interpolant.

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

$$\begin{aligned}(x_1, x_2, x_3, x_4) &= (0, 2, 4, 5), \\ (y_1, y_2, y_3, y_4) &= (0, 0.8, 0.3, 0.6). \qquad (2.3)\end{aligned}$$

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 2.1.1 *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,

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

and taking the maximum of $|e(x)|$ over $x \in [a, b]$ 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.

2.2 Cubic Hermite spline

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 (2.1), and first derivatives

$$g'(x_i) = s_i, \qquad i = 1, 2, \ldots, m. \tag{2.4}$$

Theorem 2.2.1 *There is a unique C^1 cubic spline g satisfying the interpolation conditions (2.1) and (2.4). We can express the i-th piece of g as*

$$g_i(x) = \sum_{j=0}^{3} c_j B_j^3(\lambda_i), \tag{2.5}$$

where

$$c_0 = y_i, \quad c_1 = y_i + \frac{h_i}{3}s_i, \quad c_2 = y_{i+1} - \frac{h_i}{3}s_{i+1}, \quad c_3 = y_{i+1}, \tag{2.6}$$

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

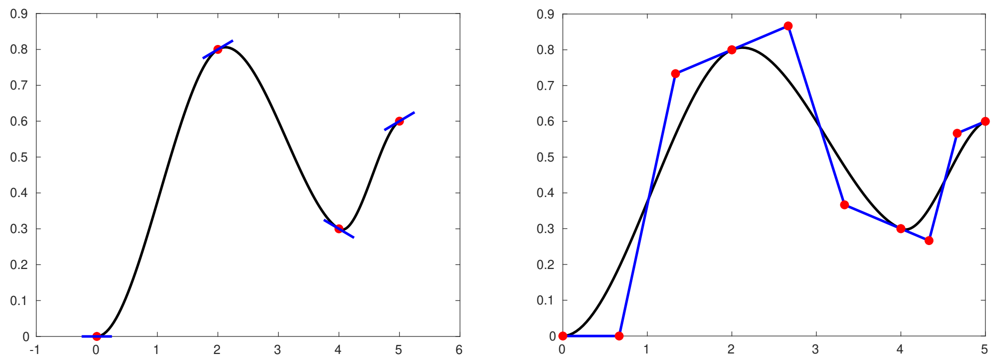

Figure 2.2: Cubic Hermite interpolant (left), BB control polygons (right).

Proof. Let g_i be a BB polynomial of the form (2.5). By Corollary 1.9.2,

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

Using the interpolation conditions (2.1) and (2.4), and solving for the coefficients c_0, c_1, c_2, c_3 gives (2.6).

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 (2.3) together with the derivative data

$$(y_1, y_2, y_3, y_4) = (0, 0.1, -0.1, 0.1). \tag{2.7}$$

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.

Theorem 2.2.2 *If $f \in C^4[a,b]$ then*

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

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,

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

□

2.3 C^2 cubic spline

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

$$g'(x_1) = s_1 \quad \text{and} \quad g'(x_m) = s_m. \tag{2.8}$$

Theorem 2.3.1 *There is a unique C^2 cubic spline g satisfying the interpolation conditions (2.1) and (2.8).*

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 1.9.2, in the notation of (2.5),

$$g_i''(x_i) = \frac{6}{h_i^2}(c_0 - 2c_1 + c_2) = \frac{6}{h_i}\left([x_i, x_{i+1}]f - \frac{2s_i + s_{i+1}}{3}\right), \tag{2.9}$$

where $[x_i, x_{i+1}]f$ is the divided difference,

$$[x_i, x_{i+1}]f = \frac{f(x_{i+1}) - f(x_i)}{h_i}.$$

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

$$g''_{i-1}(x_i) = \frac{6}{h_{i-1}^2}(\tilde{c}_1 - 2\tilde{c}_2 + \tilde{c}_3) = \frac{6}{h_{i-1}} \left(-[x_{i-1}, x_i]f + \frac{s_{i-1} + 2s_i}{3} \right). \tag{2.10}$$

Equating (2.9) and (2.10) and rearranging gives

$$\frac{h_i}{h_{i-1} + h_i} s_{i-1} + 2s_i + \frac{h_{i-1}}{h_{i-1} + h_i} s_{i+1} = 3 \left(\frac{h_i[x_{i-1}, x_i]f + h_{i-1}[x_i, x_{i+1}]f}{h_{i-1} + h_i} \right). \tag{2.11}$$

These equations for $i = 2, \ldots, m-1$ constitute a linear system of $m-2$ equations in the $m-2$ unknowns $s_2, \ldots, s_{m-1}$, which is strictly row diagonally dominant, and hence has a unique solution. □

2.4 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 2.4.1 *Let g be the C^2 cubic spline interpolant solving (2.1) and (2.8), and let h be any C^2 function satisfying the same conditions. Then*

$$\int_a^b (g''(x))^2 \, dx \leq \int_a^b (h''(x))^2 \, dx, \tag{2.12}$$

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

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

$$e(x_i) = 0, \quad i = 1, \ldots, m, \quad \text{and} \quad e'(x_1) = e'(x_m) = 0. \tag{2.13}$$

Then $h = g + e$ and so

$$\int (h'')^2 = \int (g'')^2 + 2 \int g''e'' + \int (e'')^2, \tag{2.14}$$

and therefore

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

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

$$\begin{aligned}
\int_a^b \phi e'' &= \sum_{i=1}^{m-1} \int_{x_i}^{x_{i+1}} \phi e'' \\
&= \sum_{i=1}^{m-1} \left\{ \Big[\phi e'\Big]_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} \phi' e' \right\} \\
&= \sum_{i=1}^{m-1} \left\{ \phi(x_{i+1}) e'(x_{i+1}) - \phi(x_i) e'(x_i) - \phi'|_{[x_i, x_{i+1}]} \int_{x_i}^{x_{i+1}} e' \right\} \\
&= \phi(x_m) e'(x_m) - \phi(x_1) e'(x_1) - \sum_{i=1}^{m-1} \left\{ \phi'|_{[x_i, x_{i+1}]} (e(x_{i+1}) - e(x_i)) \right\}. \qquad (2.15)
\end{aligned}$$

So, by (2.13), this integral is zero and this proves the inequality (2.12). To complete the proof, suppose that the integrals in (2.12) are equal. Then by (2.14), $\int (e'')^2 = 0$. 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 $h = g$. □

2.5 Natural end conditions

An alternative to imposing the Hermite end conditions (2.8) is to impose the so-called *natural end conditions*, which demand that the second derivative og g is zero at the endpoints, i.e.,

$$g''(x_1) = g''(x_m) = 0. \qquad (2.16)$$

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 (2.11). In addition, using equations (2.9) and (2.10), the end conditions (2.16) expand to

$$2s_1 + s_2 = 3[x_1, x_2]f,$$

and

$$s_{m-1} + 2s_m = 3[x_{m-1}, x_m]f.$$

We thus have m equations in the m unknowns $s_1, \ldots, s_m$, and this is again a strictly row diagonally dominant system of equations, and thus has a unique solution, which we call the *natural spline interpolant*.

The minimization property of Theorem 2.4 also holds for this natural spline. To see this, we just need to make a small change to the proof of Theorem 2.4. In that proof we have, as before, $e(x_i) = 0$ for all $i = 1, \ldots, m$ but now have $\phi(x_1) = \phi(x_m) = 0$. Thus, the integral $\int \phi e''$ in (2.15) is again zero.