MAT4170

Exercises for Spline Methods

August Femtehjell

Spring 2025

Contents

1 Bernstein-Bézier polynomials

Exercise 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

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

Solution 1.1 Rather than using the explicit formula for conversion, we can just expand the coefficients and collect terms.

$$\begin{aligned}
p(x) &= c_0(1-x)^2 + 2c_1x(1-x) + c_2x^2 \\
&= c_0(1-2x+x^2) + 2c_1(x-x^2) + c_2x^2 \\
&= c_0 - 2c_0x + c_0x^2 + 2c_1x - 2c_1x^2 + c_2x^2 \\
&= c_0 + (-2c_0 + 2c_1)x + (c_0 - 2c_1 + c_2)x^2.
\end{aligned}$$

Exercise 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 $a_j = p^{(j)}(0)/j!$.

Solution 1.2 We have that

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

By the Taylor approximation, we have that

$$p(x) = p(x+0) = \sum_{j=0}^{d} \frac{p^{(j)}(0)}{j!} x^j.$$

We thus have that

$$a_j = \frac{p^{(j)}(0)}{j!}.$$

By properties of the Bézier curves, we have that

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

and specifically for $x = 0$,

$$p^{(j)}(0) = \frac{d!}{(d-j)!} \Delta^j c_0.$$

Combining these results, we have that

$$a_j = \frac{p^{(j)}(0)}{j!} = \frac{d!}{(d-j)!j!} \Delta^j c_0 = \binom{d}{j} \Delta^j c_0,$$

as we wanted to show.

Exercise 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,

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

Solution 1.3 Lemma 1.2 states that for $j = 0, 1, \ldots, d$,

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

We have that

$$\sum_{j=0}^{d} a_j x^j = \sum_{i=0}^{d} c_i B_i^d(x)$$
$$\sum_{j=0}^{d} a_j \left[\frac{(d-j)!}{d!} \sum_{i=j}^{d} \frac{i!}{(i-j)!} B_i^d(x) \right] = \sum_{i=0}^{d} c_i B_i^d(x)$$

As we have $i \geq j$, we can reorder the summation to the form $j \leq i$, by using

$$\sum_{j=0}^{d} \sum_{i=j}^{d} (\ldots) = \sum_{i=0}^{d} \sum_{j=0}^{i} (\ldots).$$

This gives us

$$\sum_{i=0}^{d} \left[\sum_{j=0}^{i} a_j \frac{(d-j)!}{d!} \frac{i!}{(i-j)!} \right] B_i^d(x) = \sum_{i=0}^{d} c_i B_i^d(x).$$

Which by isolating the coefficients, gives us

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

as we wanted to show.

Exercise 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 Bézier curve, like those in Figure 1.3.

Solution 1.4 The de Casteljau algorithm uses recursion to compute the value of a point along a Bézier curve by the following formula:

1. Initialize by setting $c_i^0 = c_i$ for $i = 0, 1, \ldots, d$.

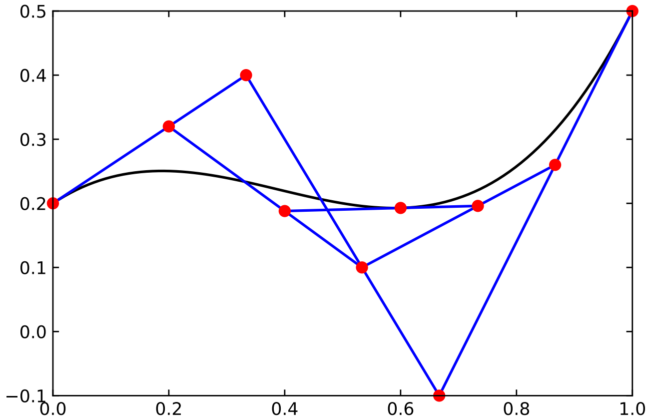

Figure 1: The de Casteljau algorithm applied to a cubic Bézier curve, with control points $(0.2, 0.4, -0.1, 0.5)$, illustrated at the point $x = 0.6$.

2. Then, for each $r = 1, 2, \ldots, d$, let

$$c_i^r = (1 - x)c_i^{r-1} + x c_{i+1}^{r-1}, \quad i = 0, 1, \ldots, d - r.$$

3. The last value c_0^d is the value of the Bézier curve at x.

This is implemented using Jax in Python in `de_casteljau.py`, and the result is shown in Figure 1, bearing striking resemblance to the figure in the book.