XI.4

SOME PROPERTIES OF BÉZIER CURVES

Ronald Goldman
University of Waterloo
Waterloo, Ontario, Canada

Most of the properties of Bézier curves can be derived from the de Casteljau evaluation algorithm (see Fig. 1). Here we show how quickly to differentiate Bézier curves and how easily to convert between the Bézier and monomial form.

Notation

Let $B(t)$ be the Bézier curve with control points $P_0, \ldots, P_n$. Then by definition

$$B(t) = \sum_k \binom{n}{k} t^k (1-t)^{n-k} P_k.$$

Let $M(t)$ be the monomial curve with coefficients $C_0, \ldots, C_n$. Then we define

$$M(t) = \sum_k \binom{n}{k} C_k t^k.$$

Notice that to the standard monomial form, we have added the binomial coefficient $\binom{n}{k}$. This will make all our algorithms simpler later on. To convert from this version of the monomial form to the standard monomial form is very easy; simply multiply C_k by $\binom{n}{k}$.

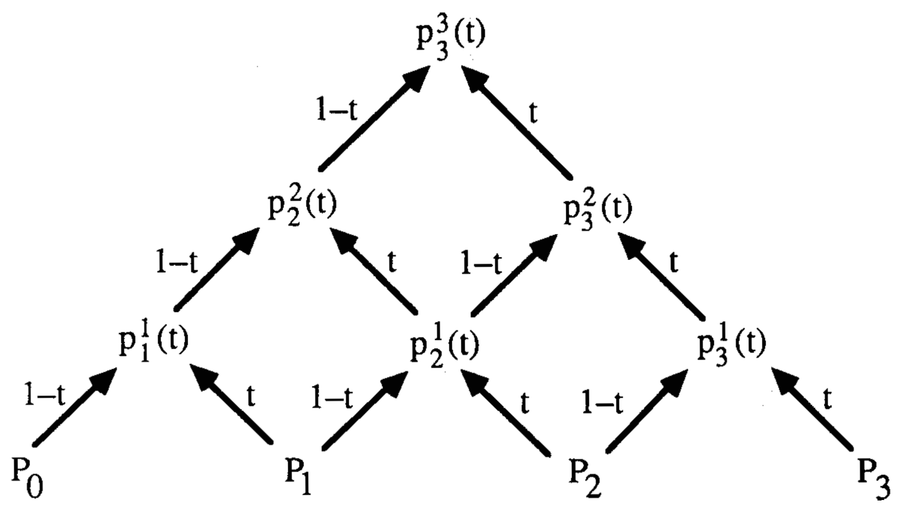

Figure 1. De Casteljau algorithm.

Remarks

We shall illustrate all of our algorithms in figures for cubic curves, but these algorithms work quite generally for polynomials of arbitrary degree *n*.

Much the same algorithms (evaluation, differentiation, and conversion) will work for *B*-spline curves if we replace the de Casteljau algorithm for Bézier curves by the de Boor algorithm for *B*-spline curves.

Bézier Curves

1. The de Casteljau Evaluation Algorithm for Bézier Curves (Fig. 1). Let

$$p_k^0(t) = P_k \quad k = 0, \ldots, n$$

$$p_k^r(t) = (1 - t)p_{k-1}^{r-1}(t) + tp_k^{r-1}(t) \quad k = r, \ldots, n$$

Then the Bézier curve with control points $P_0, \ldots, P_n$ is given by

$$B(t) = p_n^n(t).$$

We illustrate this algorithm for cubic Bézier curves in Fig. 1. Here the control points are placed at the base of a triangular array and the recursion is illustrated by labeling the arrows with the coefficients $(1 - t)$ and t. The final result, that is, a point on the curve at parameter value t, emerges at the apex of the triangle.

2. Differentiation of Bézier Curves (Fig. 2).
Let

$$p_k^0(t) = P_k \qquad k = 0, \ldots, n$$

$$p_k^r(t) = -p_{k-1}^{r-1}(t) + p_k^{r-1}(t) \qquad r = 1, \ldots, m$$

$$p_k^r(t) = (1 - t)p_{k-1}^{r-1}(t) + tp_k^{r-1}(t) \qquad r = m + 1, \ldots, n$$

Then

$$B^{(m)}(t) = \{n!/(n - m)!\}p_n^n(t)$$

Here the point is that to differentiate a Bézier curve, all we need to do is

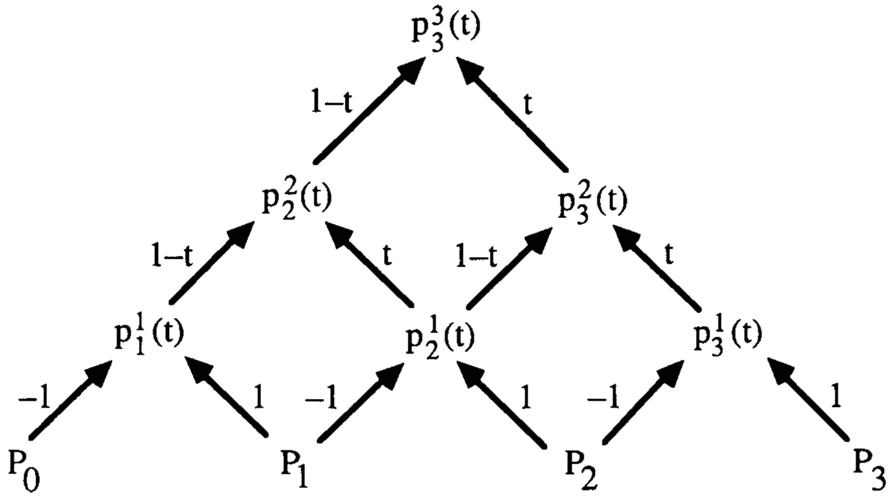

Figure 2. Differentiation algorithm.

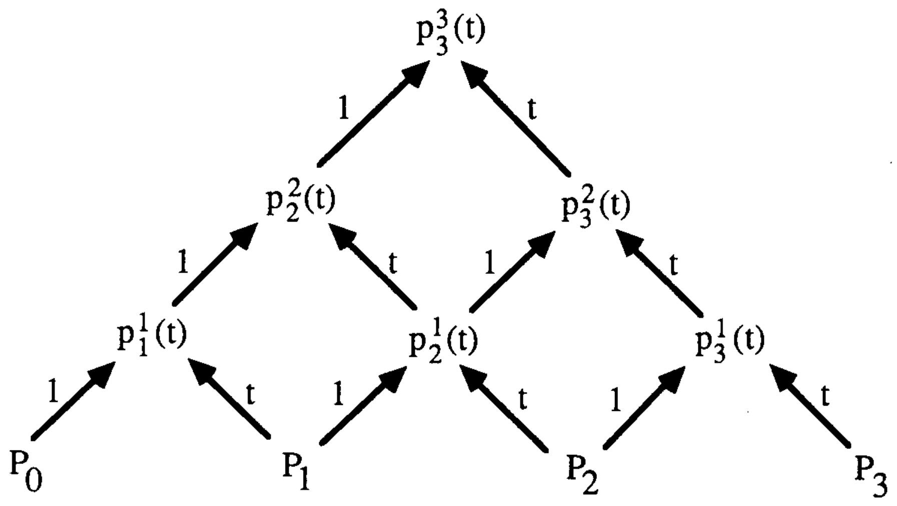

Figure 3. Monomial evaluation algorithm.

to differentiate the de Casteljau algorithm. But to differentiate the de Casteljau algorithm m times, we need only differentiate the coefficients t and $(1 - t)$ in m levels of the algorithm and multiply the final result by $n!/(n - m)!$; we need not differentiate the terms $p_k^r(t)$ at all! In fact, though we chose to differentiate the first m levels of the algorithm, we could actually differentiate any m levels and still get the correct answer. We illustrate this differentiation algorithm by finding the first derivative of a cubic Bézier curve in Fig. 2. (Remember that the result at the apex of the triangle must be multiplied by $n = 3$.) Notice that all we did is differentiate the labels on the first levels of the de Casteljau algorithm of Fig. 1.

Curves in Monomial Form

3. Evaluation Algorithm for Curves in Monomial Form (Fig. 3). Let

$$p_k^0(t) = P_k \qquad k = 0, \ldots, n$$

$$p_k^r(t) = p_{k-1}^{r-1}(t) + tp_k^{r-1}(t) \qquad k = r, \ldots, n$$

Then the monomial curve with coefficients $P_0, \ldots, P_n$ *is* given by

$$M(t) = p_n^n(t).$$

Notice that this algorithm is just the de Casteljau algorithm for Bézier curves with the coefficient $(1 - t)$ replaced by 1. This algorithm works because the monomials $\binom{n}{k}t^k$ are the same as the Bernstein basis functions $\binom{n}{k}t^k(1 - t)^{n-k}$, except that $(1 - t)$ is replaced by 1. This evaluation algorithm is much less efficient than evaluation by Horner's method. We illustrate it here only because our conversion algorithms are based on this evaluation technique.

4. Differentiation Algorithm for Curves in Monomial Form.
Let

$$p_k^0(t) = P_k \qquad k = 0, \ldots, n$$

$$p_k^r(t) = p_k^{r-1}(t) \qquad r = 1, \ldots, m$$

$$p_k^r(t) = p_{k-1}^{r-1}(t) + tp_k^{r-1}(t) \qquad r = m + 1, \ldots, n$$

Then

$$M^{(m)}(t) = \{n!/(n - m)!\}p_n^n(t).$$

This differentiation algorithm mimics the differentiation algorithm for Bézier curves. That is, only the coefficients 1 and t are differentiated. This technique works because

$$t^n = tt^{n-1} \text{ and } (t^n)' = nt^{n-1}.$$

That is, to differentiate t^n, we differentiate t and multiply the result by n.

Conversion between Bézier and Monomial Form

5. Conversion from Monomial to Bézier Form (Fig. 4).
Let the monomial coefficients be $P_0, \ldots, P_n$, and let

$$p_k^0(t) = P_k \qquad k = 0, \ldots, n$$

$$p_k^r(t) = p_{k-1}^{r-1}(t) + p_k^{r-1}(t) \qquad k = r, \ldots, n$$

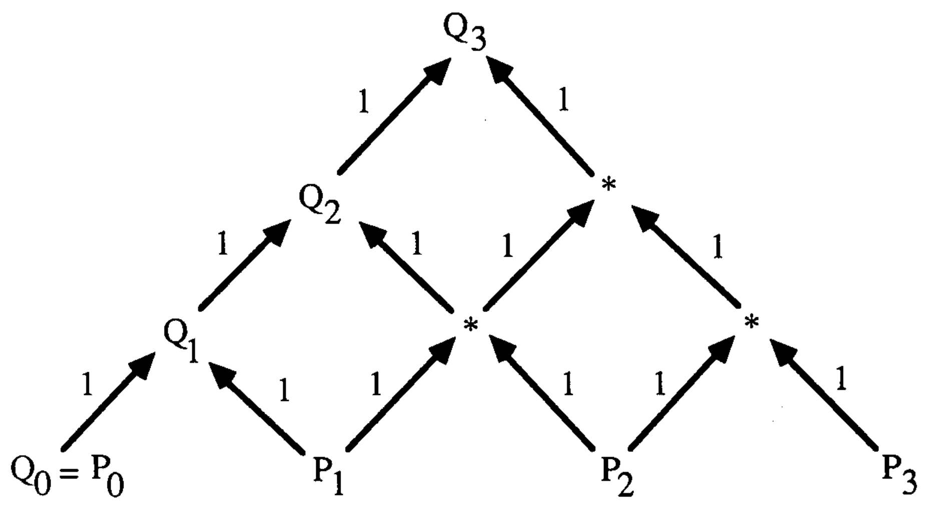

Figure 4. Conversion from monomial to Bézier form.

Then the Bézier control points $Q_0, \ldots, Q_n$ are given by

$$Q_k = p_k^k(t).$$

Notice that this algorithm is just the evaluation algorithm for monomial form with 1 substituted for t. Also the diagram is just Pascal's triangle in reverse (Fig. 4).

6. Conversion from Bézier to Monomial Form (Fig. 5).
Let the Bezier control points be $Q_0, \ldots, Q_n$, and let

$$p_k^0(t) = P_k \qquad k = 0, \ldots, n$$

$$p_k^r(t) = -p_{k-1}^{r-1}(t) + p_k^{r-1}(t) \qquad k = r, \ldots, n$$

Then the monomial coefficients $P_0, \ldots, P_n$ are given by

$$P_k = p_k^k(t).$$

Notice that this algorithm is just the differentiation algorithm for Bézier curves where we have differentiated every level of the recursion. Also the diagram is just forward differencing (Fig. 5).

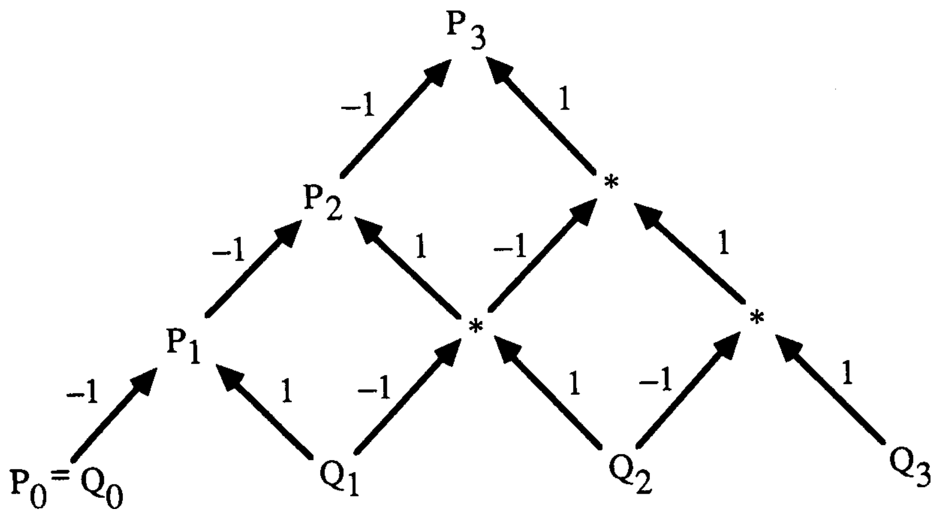

Figure 5. Conversion from Bézier to monomial form.

Observation

We can also convert from Bézier to monomial form by dividing a Bézier curve by either t^n or $(1 - t)^n$. That is, if we let $u = t/(1 - t)$, then

$$B(t)/(1 - t)^n = \sum_k \binom{n}{k} t^k (1-t)^{n-k} P_k / (1 - t)^n = \sum_k \binom{n}{k} u^k P_k.$$

Thus, we can evaluate $B(t)/(1 - t)^n$ by applying Horner's method to $\sum_k \binom{n}{k} u^k p_k$, and we can then retrieve $B(t)$ by multiplying the result by $(1 - t)^n$. This procedure is faster than converting directly to monomial form using the algorithm described above. Again this method illustrates the close connection between the Bézier and monomial form. This technique can be used to derive properties of the Bernstein polynomials and Bézier curves. For example, that the Bernstein polynomials $\binom{n}{k} tk(1-t)^{n-k}, k=0,\ldots,n$ form a basis and satisfy Descartes' Law of Signs follows easily from the corresponding facts about the monomial basis $\binom{n}{k} u^k, k=0,\ldots,n$, by applying this conversion procedure.

See also A Bézier Curve–Based Root-Finder (408)