Curves for Computer Aided Geometric Design

Jens Gravesen
Department of Mathematics
Technical University of Denmark

March 3, 1999

1 Introduction

The acronym CAGD stands for *Computer Aided Geometric Design* and the field is concerned with specifying and analyzing classes of curves (and surfaces) which can be used to model free form shapes, e.g. in CAD-systems. In this note we will stick to curves, but most of what will be said about curves can be generalized to surfaces.

Suppose we are to define a class of curves to be used in CAGD. Which requirements would we want such a class to fulfill? An obvious list is the following:

- The curves should be able to produce any shape to any given precision.
- It should be easy to evaluate values, derivatives, etc. of the curves.
- The curves should be easy and intuitive to manipulate.

We have already stated that we want the curves to model any shape we like so the first point is obvious. We want computers to handle the curves and we often want the process to be interactive which means that all calculations have to be very fast, hence the second requirement is necessary. Finally if we have an interactive process the designer should not be required to know any mathematics (this should be handled by the computer) and the program should give the designer some intuitive "handles" which can be used to manipulate the curves.

If we think a little about the Taylor expansions of a curve we see that we can always approximate any (suitably differentiable[1]) curve locally by a polynomial,

[1]This is actually not required. Due to Weierstraß' approximation theorem, any continuous curve defined on a closed interval can be approximated by a polynomial curve

so the class of piecewise polynomial curves satisfies the first requirement. It is likewise easy to evaluate polynomials, so the second requirement is also fulfilled. In a short while we will see that the last requirement is satisfied as well, and this is the reason this class of curves is so popular. If a complicated shape is modeled by a single polynomial curve then the degree of this curve can be very high. In order to avoid that the curve is divided into small simple segments, and then each segment can be modeled by a polynomial curve of low degree. The industry standard is in fact piecewise rational curves[2] which offer a bit more flexibility, but more important have the possibility to represent e.g. circles exactly. As a rational curve can be handled as a polynomial curve in a space of one higher dimension the transition from polynomial to rational is not hard. So to keep things simple we will stick to polynomial curves. For further reading the book [1] by Gerald Farin is recommended.

2 Polynomial curves

Let us look at the following example of a polynomial curve of degree 3:

$$\mathbf{r}(t) = \left(3t + 6t^2 - 3t^3, 6t - 6t^2\right), \qquad t \in [0, 1],$$

In order to tell a computer about this curve we have to use some numbers which characterize the curve. The first thing which comes to mind is to give the coefficients with respect to the *power basis*, $\{1, t, t^2, \dots\}$. I.e., we write the curve as

$$\mathbf{r}(t) = 1 \cdot (0, 0) + t \cdot (3, 6) + t^2 \cdot (6, -6) + t^3(-3, 0),$$

and use the pairs $(0, 0)$, $(3, 6)$, $(6, -6)$, and $(-3, 0)$ as input to the computer. The geometric interpretation of these coefficients is the set of the derivatives of the curve up to order 3 at the parameter value $t = 0$, see Figure 1. These derivatives are obviously *not* good intuitive handles for a designer. They do provide control in one end of the curve, but only the first few derivatives gives immediately predictible control of the curve, and it is impossible to guess what happens at the other end. This is not because there is something wrong with polynomial curves, but because it is a bad idea to use the power basis to represent polynomial curves. We need to come up with an other basis for the polynomials (of degree 3 in this case). One other choice could be the so called *Hermite polynomials* $H_{kl}(t)$, $k, l = 0, 1$ which are uniquely defined by the following equations:

$$H_{kl}^{(i)}(t_j) = \delta_{ki}\delta_{lj} = \begin{cases} 1 & \text{if } k = i \text{ and } l = j, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

[2] Called NURBS for nonuniform rational B-splines

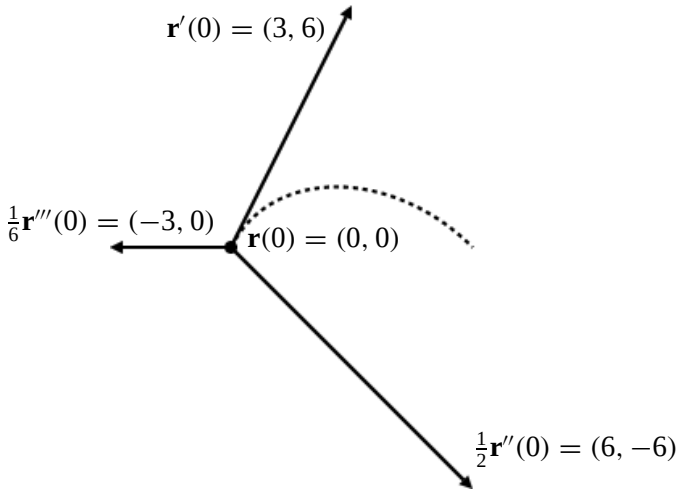

Figure 1: The data defining the curve $\mathbf{r}$, when we use the power basis. The curve is given by $\mathbf{r}(t) = \sum_k t^k \frac{1}{k!}\mathbf{r}^{(k)}(0)$, $t \in [0, 1]$.

where $i = 0, 1$ denotes the derivative, $t_0 = 0$ and $t_1 = 1$ are points of evaluation, and $k, l = 0, 1$. The Hermite polynomials are explicitly given by

$$\begin{aligned} H_{00}(t) &= 2t^3 - 3t^2 + 1, & H_{10}(t) &= t^3 - 2t^2 + t, \\ H_{01}(t) &= -2t^3 + 3t^2, & H_{11}(t) &= t^3 - t^2. \end{aligned}$$

So we write the curve $\mathbf{r}(t)$ as

$$\begin{aligned} \mathbf{r}(t) &= H_{00}(t) \cdot \mathbf{r}(0) + H_{10}(t) \cdot \mathbf{r}'(0) + H_{01}(t) \cdot \mathbf{r}(1) + H_{11}(t) \cdot \mathbf{r}'(1) \\ &= \left(2t^3 - 3t^2 + 1\right) \cdot (0, 0) + \left(t^3 - 2t^2 + t\right) \cdot (3, 6) \\ &\qquad + \left(-2t^3 + 3t^2\right) \cdot (6, 0) + \left(t^3 - t^2\right) \cdot (6, -3). \end{aligned}$$

The input to the computer would now be the coordinates with respect to the Hermite basis. The geometric interpretation of these coordinates are as the value and the first derivative at the two ends of the curve, see Figure 2. These coordinates are much more intuitive. We know the value and tangent at both ends so the shape of the curve is more easy to predict. This representation is indeed in use, and cubic polynomial curves in the Hermite representation was introduced by James Ferguson at Boeing, and goes under the name *Ferguson curve*. One drawback is that the generalization to curves of higher degree gives less intuitive control over the curve.

Finally we have the *Bernstein representation* of a polynomial curve. As the basis for the polynomials of degree 3 we use the *Bernstein polynomials* of degree

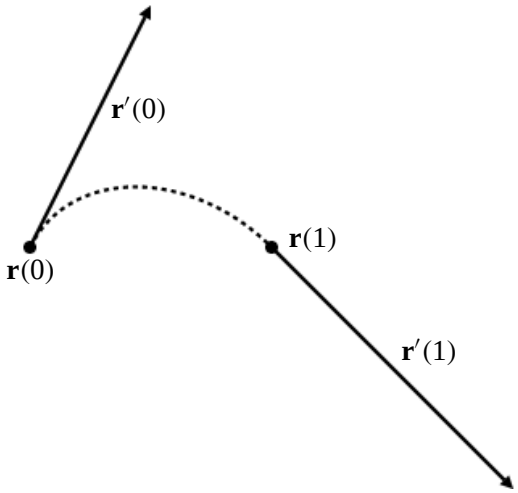

Figure 2: The data defining the curve $\mathbf{r}$, when we use the Hermite basis. The curve is called a *Ferguson curve* and is given by $\mathbf{r}(t) = \sum_{k,l} H_{kl}(t)\mathbf{r}^{(k)}(l)$, $t \in [0,1]$.

3, which are given by

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

We write the curve $\mathbf{r}(t)$ as

$$\begin{aligned}\mathbf{r}(t) &= B_0^3(t)\cdot P_0 + B_1^3(t)\cdot P_1 + B_2^3(t)\cdot P_2 + B_3^3(t)\cdot P_3 \\ &= (1-t)^3\cdot(0,0) + 3t(1-t)^2\cdot(1,2) + 3t^2(1-t)\cdot(4,1) + t^3\cdot(6,0).\end{aligned}$$

The geometric interpretation of the *control points* P_0, P_1, P_2, and P_3 can be seen in Figure 3. The control points form the *control polygon* and the curve is in some

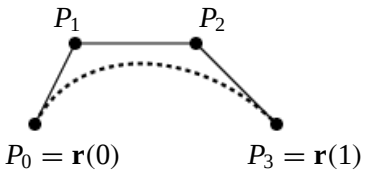

Figure 3: The data defining the curve $\mathbf{r}$, when we use the Bernstein basis. The curve is called a Bézier curve and is given by $\mathbf{r}(t) = \sum_{k=0}^{n} B_k^n(t)P_k$.

sense a "smoothened" version of the control polygon. Hence the control points provide good intuitive control over the curve.

Figure 4: The letter ‘B’is described by 8 line segments and 14 cubic Bézier curves. In the middle we have drawn the outline and marked the endpoints of all line segments and Bézier curves with massive circles. At the right we have drawn the control polygons for the Bézier curves and marked the middle control points with open circles.

Polynomial curves in the Bernstein representation was introduced by P. Bézier at Renault and are called *Bézier curves*. Bézier curves are widely used, e.g., most characters, including the ones you read right now, are described by Bézier curves, see Figure 4. Bézier curves are also a standard tool in many programs for drawing, see Figure 5.

3 Bézier curves in the Bernstein representation

Definition 1. The *Bernstein polynomials* of degree n are given by

$$B_k^n(t) = \binom{n}{k} t^k (1-t)^{n-k}, \qquad k = 0, \ldots, n.$$

where

$$\binom{n}{k} = \frac{n!}{(n-k)!\,k!}, \qquad k = 0, \ldots, n.$$

are the binomial coefficients, see Figure 6.

When the Bernstein polynomials are known, we can define a Bézier curve:

Definition 2. A *Bézier curve* of degree n with *control points* $P_0, \ldots, P_n$ is given by

$$\mathbf{r}(t) = \sum_{k=0}^{n} B_k^n(t)\, P_k, \qquad t \in [0, 1].$$

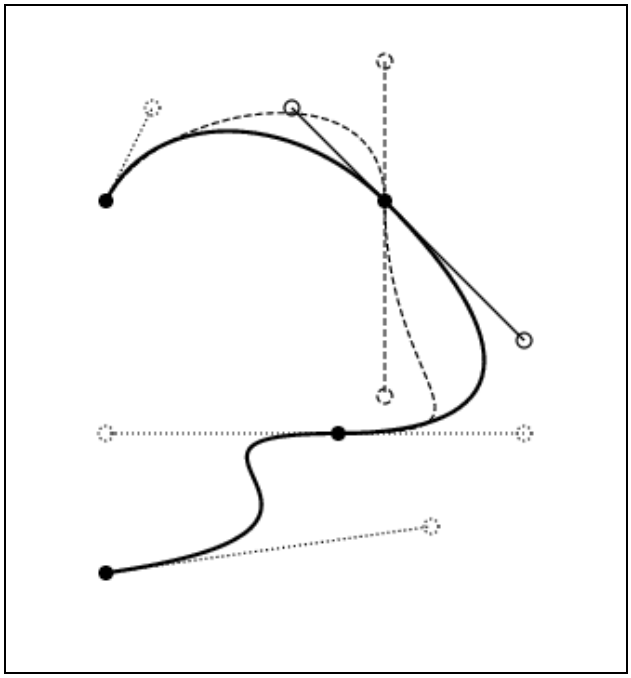

Figure 5: Control of a curve in a typical program for drawing. The segments between the solid circles are cubic Bézier curves and the two remaining control points are the open circles. By (9) the lines are the tangents to the curves at the endpoints. As can be seen, the program makes the three control points round an endpoint lie on a straight line. Hereby it is ensured that the two consecutive curves have a common tangent at the common endpoint.

In Figure 7 we have plotted Bézier curves of various degrees.

The properties of a Bézier curve can of course be derived from the properties of the Bernstein polynomials. It is not hard to show that:

$$B_k^n(0) = \begin{cases} 1 & \text{if } k = 0, \\ 0 & \text{otherwise,} \end{cases}$$

$$B_k^n(1) = \begin{cases} 1 & \text{if } k = n, \\ 0 & \text{otherwise,} \end{cases}$$

$$B_k^n(t) \leq B_k^n\left(\frac{k}{n}\right), \quad t \in [0,1],$$

$$B_l^n\left(\frac{k}{n}\right) < B_k^n\left(\frac{k}{n}\right), \quad l \neq k,$$

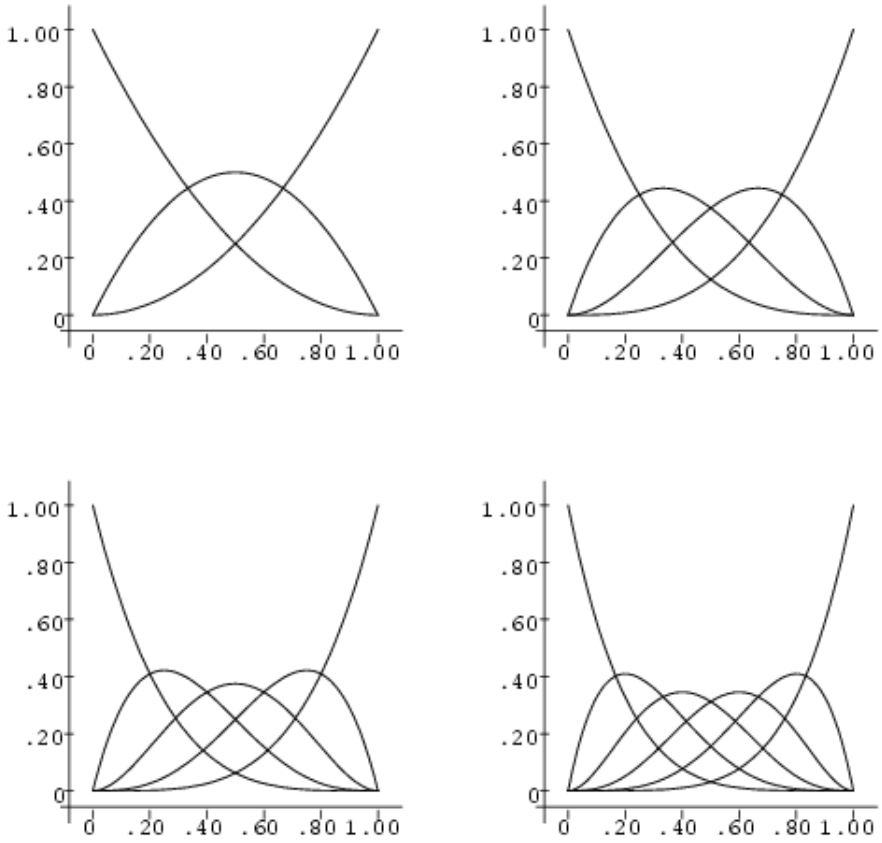

Figure 6: The Bernstein polynomials of degree 2, 3, 4, and 5.

$$\sum_{k=0}^{n} B_k^n(t) = 1,$$

see Figure 6. We will not prove these properties here. In the next section we will prove the corresponding properties for Bézier curves without the use of Bernstein polynomials. But given this information about Bernstein polynomials, we see that a Bézier curve has the following properties:

- The curve interpolates the first and the last control point.
- The curve is contained in the convex hull of the control points, see Figure 15.
- At the parameter value $\frac{k}{n}$ the control point P_k has the highest weight. That is the Bézier curve "tries to follow" the control polygon.

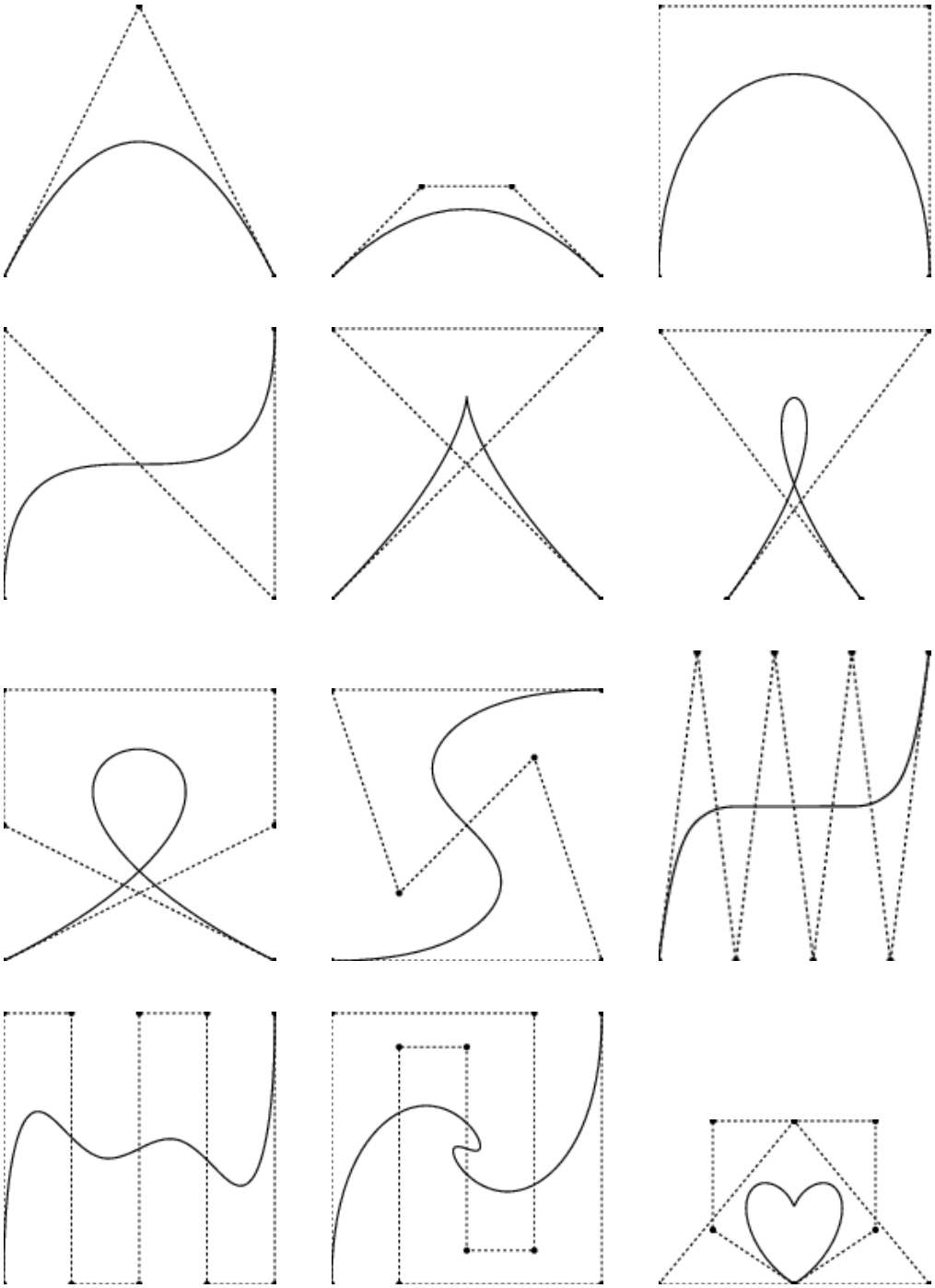

Figure 7: Bézier curves. I the first row the curves have degree 2, 3, and 3, in row number two all three curves have degree 3, in row number three the curves have degree 5, 5, and 7, and in the last row all three curves have degee 9.

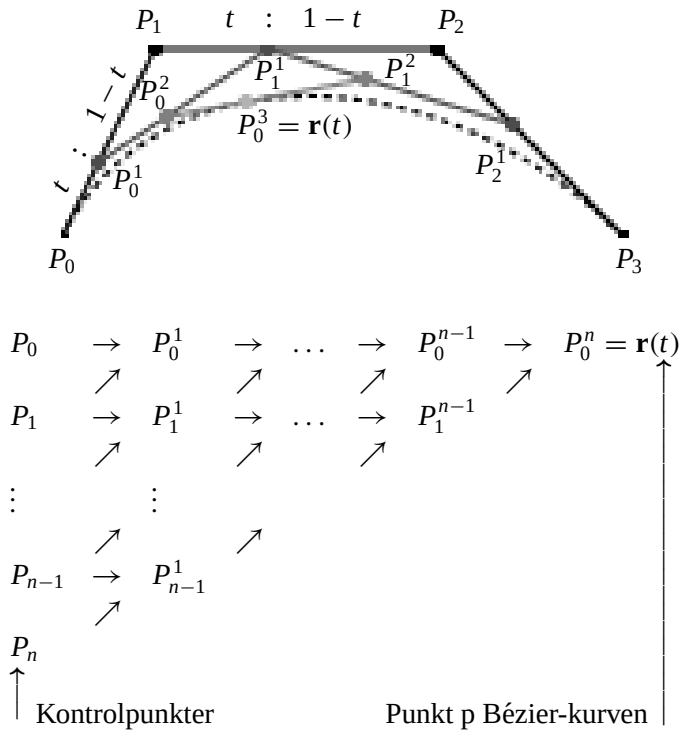

Figure 8: De Casteljau's algorithm starts with a polygon with $n+1$ points and in n it is reduced to one point, which is the desired point on the Bézier curve. In each step the points in the new polygon are obtained by dividing the legs of the previous polygon in the proportion $t : 1-t$. I the scheme this corresponds to multiply the number from the horizontal arrow by $1-t$ and multiply the point from the diagonal arrow by t. The two weighted points are then added and gives the new point.

Now we could go on and investigate the Bernstein polynomials in greater details and thus obtain information about Bézier curves. We will *not* do this, but instead give a new definition of a Bézier curve which are more geometric and in my opinion makes the analysis easier.

4 Bézier curves and de Casteljau's algorithm

At Citroen Paul de Casteljau introduced Bézier curves by repeated linear interpolation. As we shall see this approach is not only geometric in nature, but even offers simple proofs for the basic properties of Bézier curves. With few exceptions we follow the paper [3], the proof of Theorem 10 is taken from [4].

The de Casteljau algorithm is descriped in Figure 8. Formally it can be written

as

$$\begin{aligned} P_k^0(t) &= P_k, && k = 0, \ldots, n, \\ P_k^l(t) &= (1-t)P_k^{l-1}(t) + tP_{k+1}^{l-1}(t), && \begin{matrix} k = 0, \ldots, n-l, \\ l = 1, \ldots, n. \end{matrix} \end{aligned} \tag{2}$$

The de Casteljau algorithm (2) will be the basis for our development of the Bézier curve theory. As it stands, the algorithm is a bit hard to manipulate, so we will look a litle closer on the algorithm and we will introduce operators or matrices by which we can describe the algorithm. As we can see in the scheme in Figure 8 there are n steps in the algorithm and each step gives one point less. One step (going from one column to the next) is described by:

$$\begin{bmatrix} P_0^k \\ P_1^k \\ \vdots \\ P_{n-k-1}^k \\ P_{n-k}^k \end{bmatrix} \mapsto \begin{bmatrix} P_0^k \\ P_1^k \\ \vdots \\ P_{n-k-1}^k \end{bmatrix}, \begin{bmatrix} \\ P_1^k \\ \vdots \\ P_{n-k-1}^k \\ P_{n-k}^k \end{bmatrix} = \begin{bmatrix} P_0^k \\ P_1^k \\ \vdots \\ P_{n-k-1}^k \end{bmatrix}, \begin{bmatrix} P_1^k \\ P_2^k \\ \vdots \\ P_{n-k}^k \end{bmatrix}$$

$$\mapsto (1-t)\begin{bmatrix} P_0^k \\ P_1^k \\ \vdots \\ P_{n-k-1}^k \end{bmatrix} + t\begin{bmatrix} P_1^k \\ P_2^k \\ \vdots \\ P_{n-k}^k \end{bmatrix} = \begin{bmatrix} P_0^k \\ P_1^k \\ \vdots \\ P_{n-k-1}^k \end{bmatrix} + t\begin{bmatrix} P_1^k - P_0^k \\ P_2^k - P_1^k \\ \vdots \\ P_{n-k}^k - P_{n-k-1}^k \end{bmatrix}$$

Using matrix notation we can write one step in the algorithm as

$$\begin{bmatrix} P_0^{k+1} \\ P_1^{k+1} \\ \vdots \\ P_{n-k-1}^{k+1} \end{bmatrix} = \begin{bmatrix} 1-t & t & 0 & \ldots & 0 \\ 0 & 1-t & t & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & 1-t & t \end{bmatrix} \begin{bmatrix} P_0^k \\ P_1^k \\ \vdots \\ P_{n-k}^k \end{bmatrix}$$

$$= \left((1-t)\begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & 1 & 0 \end{bmatrix} + t\begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} P_0^k \\ P_1^k \\ \vdots \\ P_{n-k}^k \end{bmatrix}$$

$$= \left(\begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & 1 & 0 \end{bmatrix} + t\begin{bmatrix} -1 & 1 & 0 & \ldots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & -1 & 1 \end{bmatrix} \right) \begin{bmatrix} P_0^k \\ P_1^k \\ \vdots \\ P_{n-k}^k \end{bmatrix}.$$

We now give names to the two matrices in the middle line:

$$R = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix}, \qquad L = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 0 & 1 \end{bmatrix},$$

or equivalently we define two basic operators R and L, which act on a finite sequence of points producing a sequence with one point less. They simply remove the last, respectively the first point, from the sequence:

$$R\colon \big(P_0, P_1, \dots, P_k\big) \mapsto \big(P_0, \dots, P_{k-1}\big), \tag{3}$$
$$L\colon \big(P_0, P_1, \dots, P_k\big) \mapsto \big(P_1, \dots, P_k\big). \tag{4}$$

From R and L we define two new operators. The *forward difference* operator:

$$\Delta = L - R, \tag{5}$$

and for a $t \in \mathbb{R}$ the *de Casteljau* operator:

$$C(t) = (1-t)R + tL = R + t\Delta. \tag{6}$$

In matrix notation we have

$$\Delta = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -1 & 1 \end{bmatrix}, \qquad C(t) = \begin{bmatrix} 1-t & t & 0 & \dots & 0 \\ 0 & 1-t & t & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1-t & t \end{bmatrix}.$$

As $C(t)$ describes one step in de Casteljau's algorithm, and there are n steps all in all, we have the following definition:

Definition 3. A *Bézier curve* of degree n with *control points* $P_0, \dots, P_n$ is given by

$$\mathbf{r}(t) = C(t)^n\big(P_0, P_1, \dots, P_n\big), \qquad t \in [0,1].$$

We observe that a point on a Bézier curve is found by performing repeated interpolation between the control points so it is clear that such a point is a convex combination of the control points so we immediately have the *convex hull property*, see Figure 9.

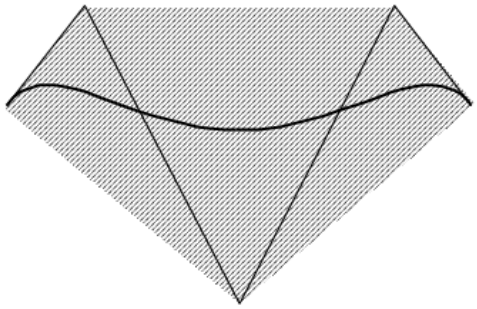

Figure 9: The convex hull property

Theorem 1. *If* $\mathbf{r}(t)$ *is a Bézier curve with control points* $P_0, \ldots, P_n$ *then*

$$\mathbf{r}(t) \in \textit{convex hull of } \{P_0, \ldots, P_n\}.$$

The following *fundamental property* is crucial for our analysis of Bézier curves.

Theorem 2. *The basic operators "commute"*:

$$LR = RL.$$

We immediately have the following

Corollary 3. *The operators* L, R, Δ, *and* $C(t)$ *"commute".*

We have put quotation marks around the word commute, because e.g. the equation in Theorem 2 should read

$$L_{k-1}R_k = R_{k-1}L_k, \tag{7}$$

where the subscript indicates the length of the sequences on which the operators acts, i.e., the R's on the two sides of the equation are different and similar with the L's. On the other hand, it will always be clear what the length of the sequence is. In order to keep the notation simple we won't decorate the operators and we will use the word commute without any further comments.

We better show that the two definitions of a Bézier curve agree:

Theorem 4. *If* $\mathbf{r}(t)$ *is a Bézier curve with control points* $P_0, \ldots, P_n$ *given by Definition 3, then*

$$\mathbf{r}(t) = \sum_{k=0}^{n} B_k^n(t) P_k,$$

in accordance with Definition 2.

Proof. The control point with index k can be found by

$$P_k = L^k R^{n-k}\big(P_0, \ldots, P_n\big).$$

As $RL = LR$ we can use the binomial formula and obtain

$$\begin{aligned} C(t)^n &= \big(tL + (1-t)R\big)^n \\ &= \sum_{k=0}^{n} \binom{n}{k} t^k (1-t)^{n-k} L^k R^{n-k}. \end{aligned}$$

Thus

$$\begin{aligned} \mathbf{r}(t) &= C(t)^n\big(P_0, P_1, \ldots, P_n\big) \\ &= \sum_{k=0}^{n} \binom{n}{k} t^k (1-t)^{n-k} L^k R^{n-k}\big(P_0, P_1, \ldots, P_n\big) \\ &= \sum_{k=0}^{n} B_k^n(t) P_k. \end{aligned}$$

□

Theorem 5. *The derivative of a Bézier curve* $\mathbf{r}(t)$ *of degree* n *with control points* $P_0, \ldots, P_n$ *can be written as*

$$\frac{1}{n}\mathbf{r}'(t) = P_1^{n-1}(t) - P_0^{n-1}(t),$$

where P_1^{n-1} *and* P_0^{n-1} *are the two second last intermediate points in de Casteljau's algorithm. Alternatively*: *the derivative* $\frac{1}{n}\mathbf{r}'(t)$ *is a Bézier curve of degree* $n-1$ *with control points*

$$\Delta\big(P_0, \ldots, P_n\big) = \big(P_1 - P_0, \ldots, P_n - P_{n-1}\big),$$

see Figure 10.

Proof. The de Casteljau operator (or matrix) is a function of t, $C(t) = R + t\Delta$, and the derivative is of course

$$\frac{d}{dt}C(t) = \Delta.$$

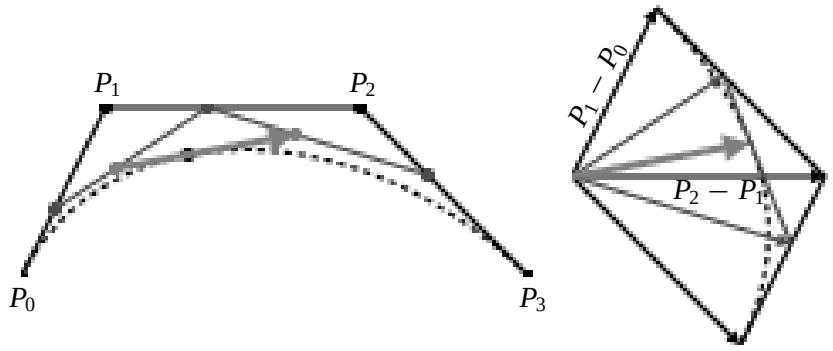

Figure 10: At the left the derivative as the difference of the penultimate points in de Casteljau's algorithm. at the right the derivative as a Bézier curve with control points equal to the differences of the originall control points.

Thus, the derivative of $C(t)^n$ is

$$\begin{aligned}\frac{d}{dt}\big(C(t)^n\big) &= \sum_{k=1}^{n} C(t)^{k-1}\left(\frac{d}{dt}C(t)\right)C(t)^{n-k}\\ &= \sum_{k=1}^{n} C(t)^{k-1}\Delta C(t)^{n-k}\\ &= nC(t)^{n-1}\Delta = n\Delta C(t)^{n-1}.\end{aligned} \tag{8}$$

If we apply the two expressions in the last line to the sequence $P_0, \ldots, P_n$ we obtain the two descriptions of the derivative. □

At the endpoints we have in particular that

$$\begin{aligned}\mathbf{r}'(0) &= n\big(P_1 - P_0\big),\\ \mathbf{r}'(1) &= n\big(P_n - P_{n-1}\big),\end{aligned} \tag{9}$$

i.e., the tangent at an endpoint is the line through the endpoint and the neighboring control point. It is equally easy to find the higher order derivatives:

Theorem 6. *The k'th derivative of a Bézier curve* $\mathbf{r}(t)$ *of degree n with control points* $P_0, \ldots, P_n$ *is given by*

$$\begin{aligned}\mathbf{r}^{(k)}(t) &= \frac{n!}{(n-k)!}\Delta^k C(t)^{n-k}\big(P_0, \ldots, P_n\big)\\ &= \frac{n!}{(n-k)!}C(t)^{n-k}\Delta^k\big(P_0, \ldots, P_n\big).\end{aligned}$$

The first expression says that the k'th derivative can be found by performing $n-k$ steps in de Casteljau's algorithm and then perform k times repeated differences. The second expression says that the k'th derivative is a Bézier curve of degree $n-k$ and its control points is found by performing k times repeated differences in the original control polygon.

If the points $P_1, \ldots, P_{n-1}$ all lie on the line segment P_0P_n, then the convex hull property implies that the image of the curve is the line segment, but the parametrization needs not be the usual. The curve might oscilate back and forth, see e.g., the y-coordinate of the first two curves in the last row in Figure 7. The next theorem tells that the control points should be equally spaced on the line segemnt in order to get the usual parametrization. This is called *linear precision*, see Figure 11.

Theorem 7. *Let $\mathbf{r}(t)$ be a Bézier curve with control points $P_0 \ldots, P_n$. Then*

$$\mathbf{r}(t) = (1-t)P_0 + tP_n,$$

if and only if

$$P_k = \frac{n-k}{n}P_0 + \frac{k}{n}P_n, \quad k = 1, \ldots, n-1.$$

Proof. As $\mathbf{r}(0) = P_0$ we have

$$\mathbf{r}(t) = (1-t)P_0 + tP_n \Longleftrightarrow \mathbf{r}'(t) = \overrightarrow{P_0P_n} \quad \text{for alle } t.$$

As $\mathbf{r}'(t)$ is a Bézier curve with control points $n\overrightarrow{P_0P_1}, \ldots, n\overrightarrow{P_{n-1}P_n}$, this only happens if and only if

$$n\overrightarrow{P_{k-1}P_k} = \overrightarrow{P_0P_n} \quad \text{for } k = 1, \ldots n.$$

Finally this is obviously equivalent with

$$P_k = \frac{n-k}{n}P_0 + \frac{k}{n}P_n, \quad k = 1, \ldots, n-1.$$

□

The above is an example of a Bézier curve (in this case of degree 1) which can be written as a curve of higher degree. The following *degree elevation theorem* tells how to raise the degree of an arbitrary Bézier curve, see Figure 11.

Theorem 8. *Let $\mathbf{r}(t)$ be a Bézier curve of degree n with control points $P_0 \ldots, P_n$. Considered as a curve of degree $n+1$, $\mathbf{r}(t)$ has control points $\widehat{P}_0 \ldots, \widehat{P}_{n+1}$, where*

$$\begin{aligned}
\widehat{P}_0 &= P_0, \\
\widehat{P}_k &= \frac{n+1-k}{n+1}P_k + \frac{k}{n+1}P_{k-1}, \quad k = 1, \ldots, n \\
\widehat{P}_{n+1} &= P_n.
\end{aligned}$$

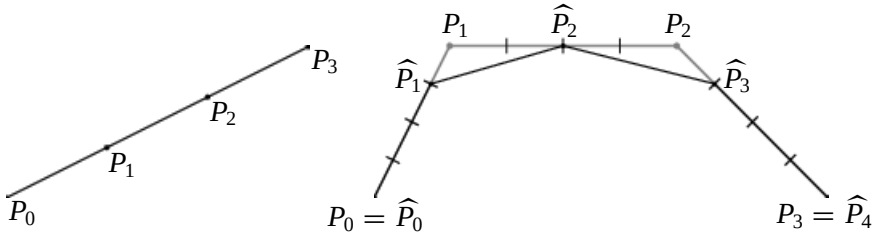

Figure 11: To the left linear precision: The control points should be equally spaced in order to get the usual parametrized line segment. To the right degree elevation: If the degree of the original curve is n then each leg of the control polygon is divided into n equally sized segments. Besides the first and the last control point, one picks one of the points on each leg of the polygon. The last point on the last leg, the penultimate point on the second leg, and so on, untiil the the first point on the last leg. This gives $n+1$ points all in all, Theese points are exactly the control points for the same curve considered as a Bézier curve of one degree more.

Proof. Even though it is possible to prove this theorem using operators, see [3], this is the one case where it is easier to use the Bernstein representation. As

$$\begin{aligned}(1-t)B_i^n(t) &= \frac{n!}{(n-i)!\,i!}(1-t)^{n+1-i}t^i \\ &= \frac{n+1-i}{n+1}\frac{(n+1)!}{(n+1-i)!\,i!}(1-t)^{n+1-i}t^i \\ &= \frac{n+1-i}{n+1}B_i^{n+1}(t),\end{aligned}$$

and

$$\begin{aligned}tB_i^n(t) &= \frac{n!}{(n-i)!\,i!}(1-t)^{n-i}t^{i+1} \\ &= \frac{i+1}{n+1}\frac{(n+1)!}{\big((n+1)-(i+1)\big)!\,(i+1)!}(1-t)^{(n+1)-(i+1)}t^{i+1} \\ &= \frac{i+1}{n+1}B_{i+1}^{n+1}(t),\end{aligned}$$

we get

$$\begin{aligned}\mathbf{r}(t) &= \big((1-t)+t\big)\mathbf{r}(t) = \sum_{i=0}^{n}\big((1-t)B_i^n(t)+tB_i^n(t)\big)P_i \\ &= \sum_{i=0}^{n}\left(\frac{n+1-i}{n+1}B_i^{n+1}(t)+\frac{i+1}{n+1}B_{i+1}^{n+1}(t)\right)P_i \\ &= B_0^{n+1}(t)P_0+\sum_{i=1}^{n}\frac{n+1-i}{n+1}B_i^{n+1}(t)P_i+\sum_{i=0}^{n-1}\frac{i+1}{n+1}B_{i+1}^{n+1}(t)P_i+B_{n+1}^{n+1}(t)P_n\end{aligned}$$

$$= B_0^{n+1}(t)P_0 + \sum_{i=1}^{n} B_i^{n+1}(t)\left(\frac{n+1-i}{n+1}P_i + \frac{i}{n+1}P_{i-1}\right) + B_{n+1}^{n+1}(t)P_n$$
$$= \sum_{i=0}^{n+1} B_i^{n+1}(t)\widehat{P}_i.$$

As claimed. □

For the proof of the next theorem we need the following lemma, which also have some interest in its own right.

Lemma 9. *De Casteljau's operator $C(t)$ is invariant under an affine change of parameter, that is for $a, b, t \in \mathbb{R}$:*

$$C\big((1-t)a+tb\big) = (1-t)C(a) + tC(b).$$

Proof. Using the definition of $C(t)$ we immediatly get:

$$\begin{aligned} C\big((1-t)a+tb\big) &= \big(1-\big((1-t)a+tb\big)\big)R + \big((1-t)a+tb\big)L \\ &= (1-a+ta-tb)R + (a-ta+tb)L, \end{aligned}$$

and

$$\begin{aligned} (1-t)C(a) + tC(b) &= (1-t)\big((1-a)R + aL\big) + t\big((1-b)R + bL\big) \\ &= (1-a-t+ta)R + (a-ta)L + (t-tb)R + tbL \\ &= (1-a+ta-tb)R + (a-ta+tb)L. \end{aligned}$$

The two expressions are equal and we have proven the lemma. □

If $\mathbf{r}(t)$ is a Bézier curve of degree n, then the curve $t \mapsto \mathbf{r}\big((1-t)a+tb\big)$, $t \in [0,1]$, is obviously also a polynomial curve of degree n, and as an abstract curve it is equal to the restriction of $\mathbf{r}$ to the interval $[a,b]$. As it is polynomial it can be considered as a Bézier curve and its control points can be found by de Casteljau's algorithm:

Theorem 10. *If $\mathbf{r}(t)$ is a Bézier curve with control points $P_0, \ldots, P_n$, then*

$$t \mapsto \mathbf{r}\big((1-t)a+tb\big)$$

is a Bézier curve with control points

$$\widehat{P}_k = C(a)^{n-k}C(b)^k\big(P_0, \ldots, P_n\big), \quad k = 0, \ldots, n.$$

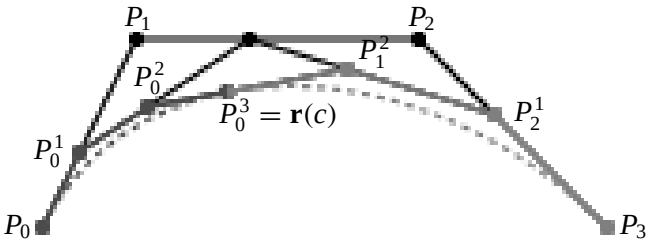

Figure 12: Subdivision of a Bézier curve at the parameter value c. The first part of the curve $t \mapsto \mathbf{r}(ct)$ has control points P_0, $P_0^1(c)$, $P_0^2(c)$, and $P_0^3(c)$, the last part of the curve $t \mapsto \mathbf{r}\big(c + (1-c)t\big)$ has control points $P_0^3(c)$, $P_1^2(c)$, $P_2^1(c)$, and P_3.

Proof. (From [4]). We have to prove that

$$C\big((1-t)a + tb\big)^n\big(P_0, \ldots, P_n\big) = C(t)^n\big(\widehat{P}_0, \ldots, \widehat{P}_n\big)$$

According to lemma 9 and the binomial formula we have

$$\begin{aligned} C\big((1-t)a + tb\big)^n\big(P_0, \ldots, P_n\big) &= \big((1-t)C(a) + tC(b)\big)^n\big(P_0, \ldots, P_n\big) \\ &= \sum_{i=0}^{n} \binom{n}{i} \big((1-t)C(a)\big)^{n-i}\big(tC(b)\big)^i\big(P_0, \ldots, P_n\big) \\ &= \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i C(a)^{n-i} C(b)^i\big(P_0, \ldots, P_n\big) \\ &= \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i\, \widehat{P}_i = C(t)^n\big(\widehat{P}_0, \ldots, \widehat{P}_n\big), \end{aligned}$$

and the proof is complete. □

This process is called *subdivision*, and the two cases $[a, b] = [0, c]$ and $[a, b] = [c, 1]$ are particular simple and are illustrated in Figure 12.

Subdivision forms the basis for a powerfull method by which we can treat Bézier curves, see [2]. Suppose we have some geometric quantity we want to determine, you may think of just the graph, i.e, the image of the curve, but it could be the length of the curve, the total curvature of the curve, the total curvature variation of the curve, etc. Suppose this quantity is easy to determine for the control polygon, then we can determine the quantity for the curve by the following principle:

- We determine the quantity for the control polygon.
- We estimate the error and if it is small, then we use this quantity.

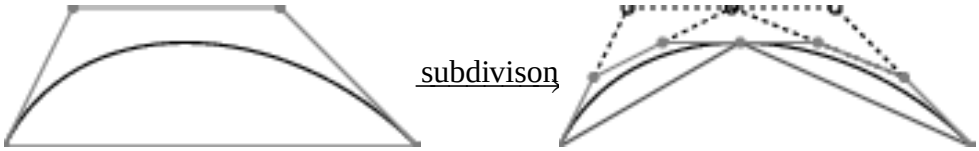

Figure 13: Drawing a Bézier curve by drawing the subdivided control polygons.

- Else we subdivide the curve and start over again with each half.

In Figure 13 this method is illustrated. The two control polygons on the right gives a much better approximation to the curve than the control polygon on the left.

This method works in a lot of instances and also imply the *variation diminishing property*, see Figure 15. Consider once more the scheme in Figure 8. If we keep the points in the top horizontal row and the points in the lower diagonal row, then a step in de Casteljau's algorithm increases the number of points by one. After k steps in the algorithm we have the polygon consisting of the points:

$$P_0, P_0^1, \ldots, P_0^k, \ldots, P_{n-k}^k, \ldots, P_{n-1}^1, P_n.$$

After n steps we have the control polygons for both halfs of the curve. With this interpretation of de Casteljau's algorithm we have the following lemma:

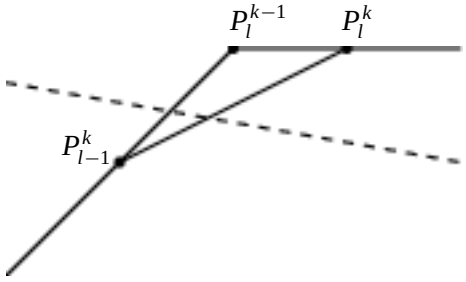

Figure 14: De Casteljau's algorithm can not increase the number of intersections.

Lemma 11. *A step in de Casteljau's algorithm can not increase the number of intersections with a hyper plane.*

Proof. Consider Figure 14, it is obvious that if the line segment $P_{l-1}^k P_l^k$ intersect a hyper plane, then the hyper plane must also intersect either the line segment $P_{l-1}^k P_l^{k-1}$ or the line segment $P_l^{k-1} P_l^k$. I.e., for each intersection in the new polygon exists a corresponding intersection in the old. □

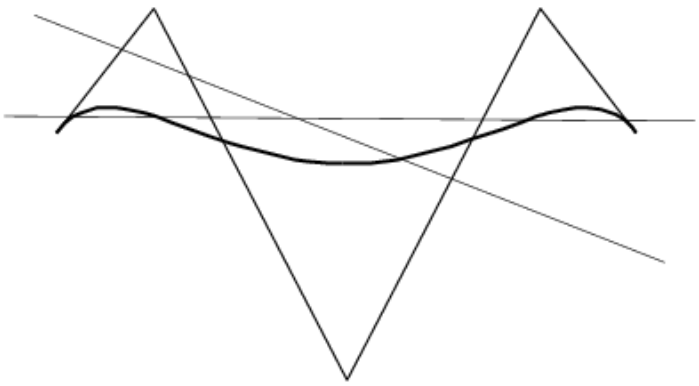

Figure 15: The variation diminishing property: the lines intersect the control polygon at least as many times as they intersect the Bézier curve.

Theorem 12. *Let* $\mathbf{r}(t)$ *be a Bézier curve with control polygon* $\mathbf{P}$*. Let furthermore* α *be a hyperplane, then*

$$\#\alpha \cap \mathbf{r} \leq \#\alpha \cap \mathbf{P},$$

i.e., the number of intersection between the curve and the hyper plane is less than the number of intersection between the control polygon and the hyper plane.

Proof. Let us consider a hyper plane which intersects a Bézier curve in a number of points. We now subdivide the curve in all those points, and hereby obtain a polygon containing all these points. The hyper plane then intersects this polygon at least as many times as it intersects the curve. According to Lemma 11 it must intersect the original control polygon at least that many times. □

References

[1] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design. A Practical Guide*. Academic Press, London, 1988.

[2] Jens Gravesen. Adaptive subdivision and the length and energy of Bézier curves. *Computational Geometry*, 8:13–31, 1997.

[3] Jens Gravesen. de Casteljau's algorithm revisited. In Morten Dæhlen, Tom Lyche, and Larry L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces II*, pages 221–228, Nashville & London, 1998. Vanderbilt University Press.

[4] Michael Ungstrup. Fairing in computer aided geometric design. Master's thesis, Department of Mathematics, Technical University of Denmark, 1998.