Notes on Geometric Modeling

Igor Guskov[†]

[†] Computer Science 256-80, Caltech, Pasadena, CA 91125.
`ivguskov@cs.caltech.edu`.

CHAPTER 1

Blossoming for curves

1. Affine subspaces

The goal of geometric modeling is to create and process geometric objects such as curves and surfaces.

A curve is a map from the parametric interval $\mathcal{I} = (a, b) \subset \mathbf{R}$ to the d-dimensional Euclidean space, where $d = 2$ for a planar curve and $d = 3$ for the curve in space. In other words, we consider a function $c(t)$ that produces points in space as the real parameter t varies between a and b. We assume that a fixed coordinate system is specified so that each point is represented by its coordinates $p = (p^1, \dots, p^d)^T \in \mathbf{R}^d$.

Example 1. $c(t) = (x^1(t), x^2(t))^T$ where $x^1(t) = t^2, x^2(t) = 1 - t$ for $t \in \mathbf{R}$. See Figure 1

Example 2. $c(t) = (x^1(t), x^2(t), x^3(t))^T$ where $x^i(t) = A^i + B^i t + C^i t^2$ for $t \in \mathbf{R}$ and $i = 1, 2, 3$.

The concept of affine subspace of a vector space proves to be very useful for geometric modeling.

Definition 3. Let $\mathcal{V}$ be a vector space. A non-empty set $\mathcal{A} \subset \mathcal{V}$ is an *affine subspace* (of $\mathcal{V}$) if for any u and $v \in \mathcal{A}$, and any $\alpha \in \mathbf{R}$ the affine combination $\alpha u + (1 - \alpha) v \in \mathcal{A}$.

Needless to say, the whole space $\mathcal{V}$ is an affine subspace of itself. Other examples of affine subspaces of three-dimensional Euclidean space are planes, straight lines, and individual points.

In this section, $\mathcal{A}$ and $\mathcal{B}$ denote some fixed affine subspaces.

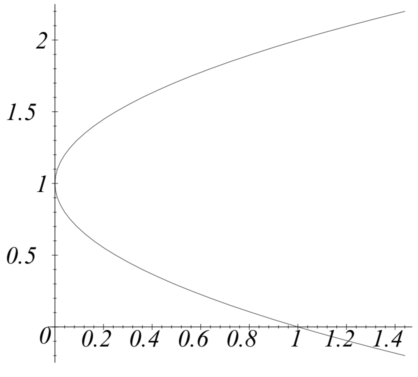

Figure 1. *The parametric curve* $c(t) = (t^2, 1 - t)$.

LEMMA 4. *If* $p_1, \dots, p_k \in \mathcal{A}$ *and* $\sum_i \alpha_i = 1$ *then* $\sum_{i=1}^k \alpha_i p_i \in \mathcal{A}$

We call $\sum_{i=1}^k \alpha_i p_i$ an *affine combination* of $p_1, \dots, p_k$ whenever $\sum_{i=1}^k \alpha_i = 1$.

We define the *affine span* of a finite set of points $P = \{p_1, \dots, p_k\}$ as the set of all the elements of $\mathcal{A}$ that can be represented as affine combinations of elements of P; that is, ${}^{\mathrm{a}}\mathrm{span}\,\{p_1, \dots, p_k\} := \{q = \sum_i \alpha_i p_i : \sum_i \alpha_i = 1\}$.

A collection $P = \{p_1, \dots, p_k\}$ of elements of $\mathcal{A}$ is *affinely independent* if for any $q \in P$ we have: $q \notin {}^{\mathrm{a}}\mathrm{span}\,(P \setminus \{q\})$, that is any point from P lies outside the affine span of the remaining points of P.

The affine frame of $\mathcal{A}$ is an affinely independent set P whose affine span is $\mathcal{A}$, that is ${}^{\mathrm{a}}\mathrm{span}\,P = \mathcal{A}$.

The following lemma characterizes all the possible affine subspaces of a vector space.

LEMMA 5. *Let* $\mathcal{V}$ *be a vector space.* $\mathcal{A}$ *is an affine subspace of* $\mathcal{V}$ *iff there exist a vector subspace* $\mathcal{T}$ *of* $\mathcal{V}$ *and a vector* $p \in \mathcal{V}$ *such that* $\mathcal{A} = p + \mathcal{T}$.

The vector subspace $\mathcal{T}$ *is uniquely determined by* $\mathcal{A}$. *Moreover, for any two points* a *and* b *of* $\mathcal{A}$ *the vector* $t = b - a$ *is in* $\mathcal{T}$.

2. Affine maps

A function $f : \mathcal{A} \to \mathcal{B}$ is affine if $f(\alpha p + (1-\alpha)q) = \alpha f(p) + (1-\alpha) f(q)$.

EXAMPLE 6. The function $f(t) = A + Bt$, $t \in \mathbf{R}$ is affine.

In fact, any affine function $f : \mathbf{R} \to \mathbf{R}$ is of the above form, since $f(t) = f((1-t)0 + t1) = f(0) + (f(1) - f(0))t$.

A function $g : \mathcal{A}^n \to \mathcal{B}$ is *multi-affine* (or n-affine) if it is affine in every argument when all other arguments are held fixed, that is

$$g(p_1, \dots, \alpha p_k + (1-\alpha)p'_k, \dots, p_n) = \alpha g(p_1, \dots, p_k, \dots, p_n) + (1-\alpha) g(p_1, \dots, p'_k, \dots, p_n).$$

It is easy to show that when $\sum_i \alpha_i = 1$ we also have

$$g(p_1, \dots, \sum_i \alpha_i p_{ik}, \dots, p_n) = \sum_i \alpha_i g(p_1, \dots, p_{ik}, \dots, p_n).$$

Let $\mathcal{A} = \mathcal{B} = \mathbf{R}$. Then any multi-affine function g can be expressed in the following way:

$$g(u_1, \dots, u_n) = \sum_{S \subset \{1, \dots, n\}} a_S \prod_{k \in S} u_k.$$

DEFINITION 7. A function $g : \mathcal{A}^n \to \mathcal{B}$ is symmetric if for any permutation $\sigma \in S_n$ we have $g(u_1, \dots, u_n) = g(u_{\sigma(1)}, \dots, u_{\sigma(n)})$.

THEOREM 8. *There exists a one-to-one correspondence between polynomials of degree* n *and symmetric multi-affine functions* (polar maps) *of* n *arguments. The polynomial* F *and polar map* f *are associated if the following relation holds:* $F(u) = f(u, \dots, u)$.

F *is called the* diagonal *of* f, *and* f *is called the* blossom *of* F.

EXAMPLE 9. Consider $F(u) = u^2$. The corresponding polar map f of two arguments is given as $f(u_1, u_2) = u_1 u_2$.

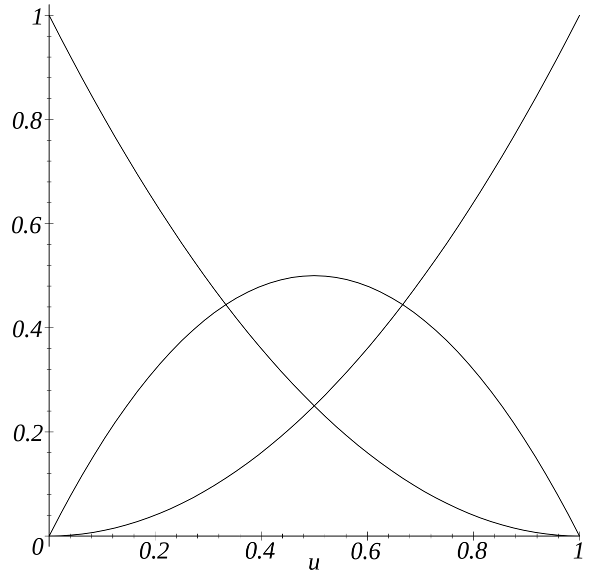

FIGURE 2. *The Bernstein polynomials of degree 2.*

Note that u^2 can be also considered as a "degenerate" polynomial of degree 3 — hence its blossom is a polar map with three arguments. The map $f(u_1, u_2, u_3) := (u_1u_2 + u_2u_3 + u_3u_1)/3$ is just what we need in this case.

Note: Polar forms can be added and multiplied by constants, so that they themselves form a vector space.

It is clear that the value of a polar form does not depend on the order of arguments; following Ramshaw, we shall call a collection of n arguments $u = (u_1, \dots, u_n)$ – a *bag* of polar arguments. Two bags u and v are equivalent if one is the permutation of the other one. For example, $(1, 2, 2, 3) \sim (2, 1, 3, 2)$. For details, see the Ramshaw's paper.

3. De Casteljau algorithm

Given values of some polar map on a finite number of bags, it is sometimes possible to find its value elsewhere.

For example, if $f(0, 1) = a_{01}$ and $f(1, 2) = a_{12}$ are given, then the value of f on any bag of the form $(1, u)$ is given by

$$f(1,u) = \frac{2-u}{2} f(0,1) + \frac{u}{2} f(2,1) = \frac{2-u}{2} a_{01} + \frac{u}{2} a_{12}.$$

This observation leads to the de Casteljau algorithm for evaluating Bezier curves.

Suppose that $f(0, 0) = a_0, f(0, 1) = a_1, f(1, 1) = a_2, a_i \in \mathcal{A}$ are given. Then we can find the values of the diagonal $F(u) = f(u, u, u)$ using the following procedure:

$$f(0,u) = (1-u)f(0,0) + uf(0,1),$$
$$f(1,u) = (1-u)f(0,1) + uf(1,1),$$

$$F(u) = f(u,u) = (1-u)f(0,u) + uf(1,u).$$

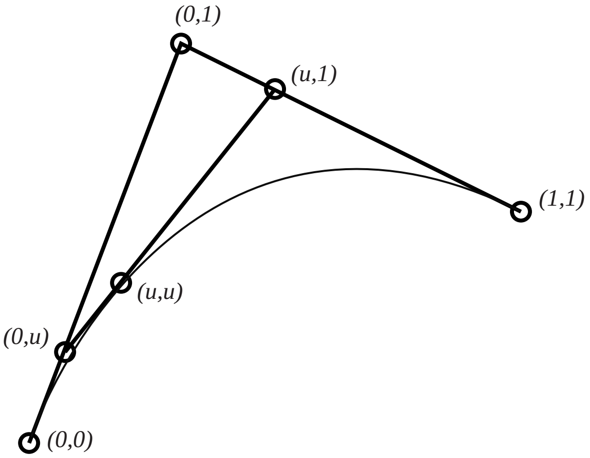

FIGURE 3. *A Bezier curve of degree 2.*

Collecting the terms we get the following expression for $F(u)$:

$$F(u) = a_0(1-u)^2 \, + \, a_1 2(1-u)u \, + \, a_2 u^2 = \sum_{i=0}^{2} a_i B_i^2(u);$$

where $B_i^2(u)$ are the Bernstein polynomials shown in Figure 3. Similarly, the Bezier form of a cubic curve is specified by four control points: $f(0,0,0), f(0,0,1), f(0,1,1), f(1,1,1)$. The evaluation of the diagonal cubic then proceeds as follows:

$$\begin{aligned} f(0,0,u) &= (1-u)f(0,0,0) + uf(0,0,1), \\ f(0,1,u) &= (1-u)f(0,0,1) + uf(0,1,1), \\ f(1,1,u) &= (1-u)f(0,1,1) + uf(1,1,1), \\[1em] f(0,u,u) &= (1-u)f(0,0,u) + uf(0,1,u), \\ f(1,u,u) &= (1-u)f(0,1,u) + uf(1,1,u), \\[1em] F(u) = f(u,u,u) &= (1-u)f(0,u,u) + uf(1,u,u). \end{aligned}$$

Generally, a polynomial curve G of arbitrary degree n is represented in the Bezier form as a linear combination of Bernstein polynomials with the coefficients (control points) given by the values of the corresponding polar form g. Namely,

$$F(u) = \sum_{i=0}^{n} a_i B_i^n(u),$$

where $a_i := g(\underbrace{0,\ldots,0}_{n-i}, \underbrace{1,\ldots,1}_{i})$ are the Bezier control points, and $B_i^n(u) := \binom{n}{i} u^i (1-u)^{n-i}$ are the Bernstein polynomials.

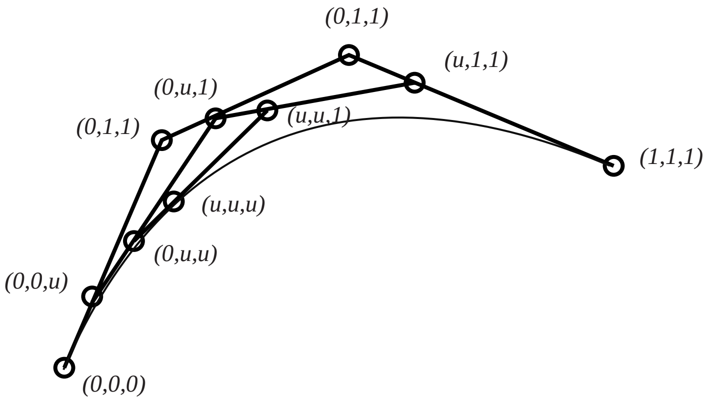

Figure 4. *A Bezier curve of degree 3.*

4. de Boor algorithm

Another convenient representation of a polynomial curve is given by B-spline curves. Later we shall return to define these curves in more detail. At this point in the discussion, we consider an example of the cubic B-spline evaluation from its control points with the following de Boor algorithm. Here we assume that the *four* control points are given as the values of the polar form $f(1,2,3), f(2,3,4), f(3,4,5), f(4,5,6)$. Then the computation of the diagonal can be performed as follows:

$$\begin{aligned} f(u,2,3) &= \frac{4-u}{3} f(1,2,3) + \frac{u-1}{3} f(2,3,4), \\ f(u,3,4) &= \frac{5-u}{3} f(2,3,4) + \frac{u-2}{3} f(3,4,5), \\ f(u,4,5) &= \frac{6-u}{3} f(3,4,5) + \frac{u-3}{3} f(4,5,6), \end{aligned}$$

$$\begin{aligned} f(u,u,3) &= \frac{4-u}{2} f(u,2,3) + \frac{u-2}{2} f(u,3,4), \\ f(u,u,4) &= \frac{5-u}{2} f(u,3,4) + \frac{u-3}{2} f(u,4,5), \end{aligned}$$

$$F(u) = f(u,u,u) = (4-u) f(u,u,3) + (u-3) f(u,u,4).$$

Note that if we have a similar *progressive* sequence that contains more than four control points then it is possible to produce a piecewise polynomial curve that is twice differentiable. Suppose that we have been given five control points $p_{-2}, p_{-1}, p_0, p_1, p_2$. Generally, these five points will not have a unique cubic passing through them — instead we shall be looking for *two* cubic curves F and G that match up at a certain point together with their derivatives. For that, we will again specify the values for the blossoms f and g of these two curves and run the de Boor

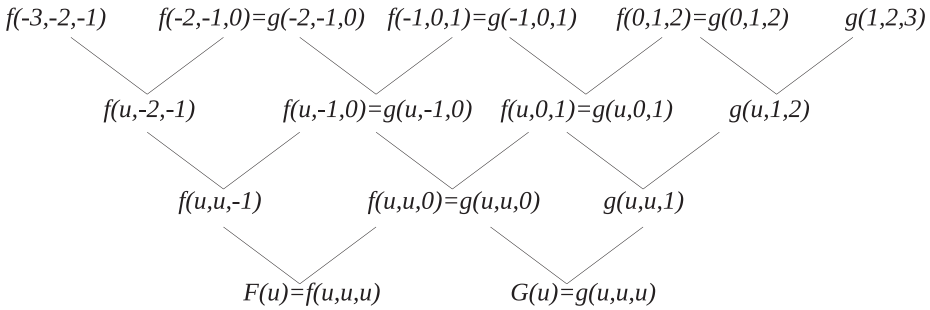

Figure 5. *Applying the de Boor algorithm for cubics on neighboring intervals.*

algorithm. We consider the polar forms f and g along with the following conditions:

$$\begin{aligned}
&f(-3,-2,-1) = p_{-2},\\
&f(-2,-1,0) = g(-2,-1,0) = p_{-1},\\
&f(-1,0,1) = g(-1,0,1) = p_0,\\
&f(0,1,2) = g(0,1,2) = p_1,\\
&g(1,2,3) = p_2.
\end{aligned}$$

Now we can apply the de Boor algorithm as shown in Figure 4. Now if we define the function

$$c(u) := \begin{cases} F(u) & \text{for } u \leq 0,\\ G(u) & \text{for } u > 0; \end{cases}$$

then $c(u)$ is polynomial on both sides of $u = 0$, and as we will see in the next section it is twice differentiable at the point $u = 0$ where two cubic curves meet.

5. Differentiating along the diagonal, differencing of the blossom.

We shall now find the derivative of the polynomial function that is the diagonal of a polar form. Let us consider a quadratic polynomial and the corresponding polar form with two arguments. Let $F(u) := f(u,u), u \in \mathbf{R}$. We will find the derivative of F starting from the "first principles". That is, by definition of derivative we have:

$$\frac{d}{du}F(u) := \lim_{\tau\to 0}\frac{F(u+\tau)-F(u)}{\tau} = lim_{\tau\to 0}\frac{f(u+\tau,u+\tau)-f(u,u)}{\tau} =$$
$$lim_{\tau\to 0}\frac{1}{\tau}\left(f(u+\tau,u+\tau)-f(u+\tau,u)+f(u+\tau,u)-f(u,u)\right).$$

Before proceeding any further, we introduce the "differenced" polar form, together with a new notation:

Definition 10. Given a polar form $f(u_1,\ldots,u_n)$, the "differenced" polar form $\Delta[\tau]f(u_1,\ldots,u_{n-1})$ is defined by the following formula

$$\Delta[\tau]f(u_1,\ldots,u_{n-1}) := f(u_1,\ldots,u_n+\tau) - f(u_1,\ldots,u_n).$$

The below lemma justifies the above definition.

Lemma 11. *Definition 10 is correct:*

1. *$f(u_1, \dots, u_n + \tau) - f(u_1, \dots, u_n)$ does not depend on u_n;*
2. *$\Delta[\tau]f(\cdot, \dots, \cdot)$ is (n-1)-affine symmetric form;*
3. *$\Delta[\tau]f(u_1, \dots, u_{n-1})$ is linear in τ.*
4. *Moreover, $\Delta[\tau]$ is linear that is*

$$\Delta[\tau](af + bg) = a\Delta[\tau]f + b\Delta[\tau]g.$$

PROOF. 1. Indeed, using the fact that $1 - 1 + 1 = 1$ we have

$$\begin{aligned} & f(u_1, \dots, u_n + \tau) = f(u_1, \dots, u_n - v + v + \tau) = \\ & \quad f(u_1, \dots, 1u_n + (-1)v + 1(v + \tau)) = \\ & \quad 1f(u_1, \dots, u_n) + (-1)f(u_1, \dots, v) + 1f(u_1, \dots, v + \tau) = \\ & \quad f(u_1, \dots, u_n) - f(u_1, \dots, v) + f(u_1, \dots, v + \tau). \end{aligned}$$

Hence, for any v we have

$$f(u_1, \dots, u_n + \tau) - f(u_1, \dots, u_n) = f(u_1, \dots, v + \tau) - f(u_1, \dots, v).$$

2. Exercise.
3. Here we prove linearity of $\Delta[\tau]f$ w.r.t. τ.
   - $\Delta[\tau + \tau']f = \Delta[\tau]f + \Delta[\tau']f$:

$$\begin{aligned} \Delta[\tau + \tau']f(\cdots) &= f(\cdots, u + \tau + \tau') - f(\cdots, u) = \\ & f(\cdots, u + \tau - u + u + \tau') - f(\cdots, u) = \\ & f(\cdots, u + \tau) - f(\cdots, u) + f(\cdots, u + \tau') - f(\cdots, u) = \\ & \Delta[\tau]f(\cdots) + \Delta[\tau']f(\cdots). \end{aligned}$$

   - $\Delta[a\tau]f = a\Delta[\tau]f$:

$$\begin{aligned} \Delta[a\tau]f(\cdots) &= f(\cdots, u + a\tau) - f(\cdots, u) = \\ & f(\cdots, a(u + \tau) + (1 - a)u) - f(\cdots, u) = \\ & af(\cdots, u + \tau) + (1 - a)f(\cdots, u) - f(\cdots, u) = \\ & af(\cdots, u + \tau) - af(\cdots, u) = a\Delta[\tau]f(\cdots). \end{aligned}$$

□

EXAMPLE 12. Let $f(u_1, u_2) = 1 + 3(u_1 + u_2) + 5u_1u_2$. It is easy to see that $\Delta[\tau]f(u_1) = \tau(3 + 5u_1)$.

Now we are ready to attack the derivative computation.

$$\begin{aligned} & \frac{d}{du}F(u) = \\ & \quad lim_{\tau \to 0}\frac{1}{\tau}\left(f(u + \tau, u + \tau) - f(u + \tau, u) + f(u + \tau, u) - f(u, u)\right) = \\ & \quad lim_{\tau \to 0}\frac{1}{\tau}\left(\Delta[\tau]f(u + \tau) + \Delta[\tau]f(u)\right) = \\ & \quad lim_{\tau \to 0}\left(\Delta[1]f(u + \tau) + \Delta[1]f(u)\right) = 2\Delta[1]f(u). \end{aligned}$$

In the general case, we have the following lemma

Lemma 13. *Let* $F(u) = f(\underbrace{u, \dots, u}_{n})$. *Then its derivative is given by*

$$\frac{d}{du}F(u) = n\Delta[1]f(\underbrace{u, \dots, u}_{n-1}).$$

In order to introduce higher-order derivatives we definitely need higher-order differencing of the blossom. Indeed, given a n-affine symmetric form $f(u_1, \dots, u_n)$, we recursively define

$$\begin{aligned}&\Delta[\tau_1, \dots, \tau_p]f(u_1, \dots, u_{n-p}) := \\ &\Delta[\tau_p]\Delta[\tau_1, \dots, \tau_{p-1}]f(u_1, \dots, u_{n-p}) = \\ &\Delta[\tau_1, \dots, \tau_{p-1}]f(u_1, \dots, u_{n-p}, u_{n-p+1} + \tau_{p-1}) - \\ &\Delta[\tau_1, \dots, \tau_{p-1}]f(u_1, \dots, u_{n-p}, u_{n-p+1}).\end{aligned}$$

It is possible to show that $\Delta[\tau_1, \dots, \tau_p]f$ does not depend on the order of τ's.

The following theorem then gives the p-th derivative of the diagonal in terms of its blossom's differences:

Theorem 14. *(Blossoming principle) Let* $F(u) = f(\underbrace{u, \dots, u}_{n})$. *Then its derivative is given by*

$$\frac{d^p}{du^p}F(u) = \frac{n!}{(n-p)!}\Delta[\underbrace{1, \dots, 1}_{p}]f(\underbrace{u, \dots, u}_{n-p}).$$

The following lemma relates the properties of the differenced polar form to the properties of the original form.

Lemma 15. *Let* f *be a polar n-form, and* $0 \leq p \leq n$. *Fix* $u \in \mathbf{R}$. *The following two statements are equivalent:*

1. *For all* $q = 0, \dots, p$ *and for all* $\tau_1, \dots \tau_p \in \mathbf{R}$ *we have*

$$\Delta[\tau_1, \dots, \tau_q]f(\underbrace{u, \dots, u}_{n-q}) = 0;$$

2. *For all* $v_1, \dots, v_p \in \mathbf{R}$ *we have*

$$f(\underbrace{u, \dots, u}_{n-p}, v_1, \dots, v_p) = 0.$$

Proof.

- 2 ⇒ 1 is obvious because the differenced forms are computed as linear combinations of the original forms with first $n - p$ arguments set to u so it can only be zero.
- 1 ⇒ 2 We can use the equation $f(\cdots, v) = f(\cdots, u) + \Delta[v - u]f(\cdots)$ together with the fact that $f(u, \dots, u) = 0$ to prove the result by induction.

□

Note also that the value of a differenced form is fully determined by its value when the τ's are all set to 1, indeed $\Delta[\tau_1, \dots, \tau_q]f = \tau_1 \cdots \tau_q \Delta[1, \dots, 1]f$ by linearity, therefore the following corollary holds:

Corollary 16. *Let* f *be a polar n-form, and* $0 \leq p \leq n$. *Fix* $u \in \mathbf{R}$. *The following two statements are equivalent:*

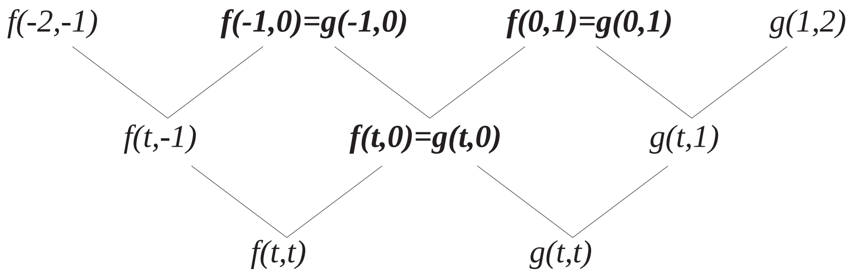

Figure 6. *Applying the de Boor algorithm to produce two neighboring quadratic B-spline segments.*

1. *For all* $q = 0, \dots, p$ *and for all* $\tau_1, \dots \tau_p \in \mathbf{R}$ *we have*

$$\Delta[\underbrace{1, \dots, 1}_{q}]f(\underbrace{u, \dots, u}_{n-q}) = 0;$$

2. *For all* $v_1, \dots, v_p \in \mathbf{R}$ *we have*

$$f(\underbrace{u, \dots, u}_{n-p}, v_1, \dots, v_p) = 0.$$

A similar characterization for the original polar form is somewhat harder. We shall return to it later. Let us now apply the above result to the smoothness of the piecewise polynomial functions. The combination of Theorem 14 and Corollary 16 produces the following lemma:

Lemma 17. *Let* f *be a polar form and* F *its diagonal. Fix some* $u \in \mathbf{R}$. *Then all the derivatives of* F *up to the order* p *at the point* u *are equal to zero iff* $f(\underbrace{u, \dots, u}_{n-p}, v_1, \dots, v_p)$ *is zero for all* $v_1, \dots, v_p \in \mathbf{R}$.

Applying this result to the difference of two polynomials G and H with blossoms g and h leads to the following theorem:

Theorem 18. *(C^p Condition) Fix* $u \in \mathbf{R}$. *Then*

$$\frac{d^q}{dt^q}G(u) = \frac{d^q}{dt^q}H(u) \, for \;\; q = 0, \dots, p$$

if and only if $g(\underbrace{u, \dots, u}_{n-p}, v_1, \dots, v_p)$ *and* $h(\underbrace{u, \dots, u}_{n-p}, v_1, \dots, v_p)$ *coincide for all* $v_1, \dots, v_p \in \mathbf{R}$.

Example 19. Let's look at the de Boor algorithm for quadratic curves.

$$f(0, u) = (1 - u)f(0, 0) + uf(0, 1),$$
$$f(1, u) = (1 - u)f(0, 1) + uf(1, 1),$$

$$F(u) = f(u, u) = (1 - u)f(0, u) + uf(1, u).$$

Figure 5 shows how the values of two neighboring polynomial pieces are computed. It is clear that Theorem 18 applies in this case with $u = 0, t = v_1, p = 1$ so that the two polynomial segments meet at the point 0 with continuous first derivative.

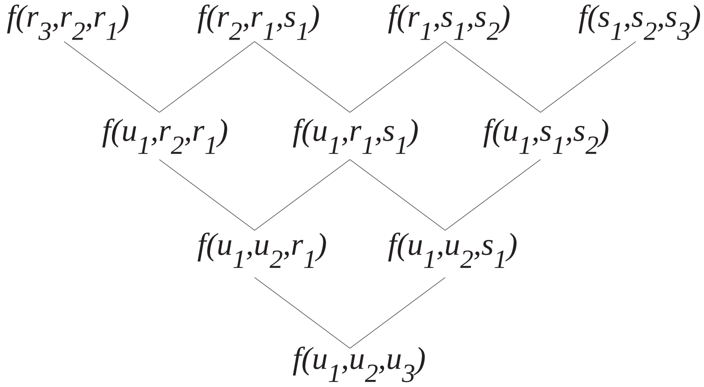

FIGURE 7. *The multi-affine version of the de Boor algorithm for cubics*

Let us now introduce a general case of the polynomial evaluation from a sequence of knots. Following Seidel, let

$$r_n \leq r_{n-1} \leq \ldots \leq r_1 < s_1 \leq \ldots \leq s_{n-1} \leq s_n,$$

be a sequence of real numbers. We claim that for any $2n$ points $p_1, \ldots, p_{2n} \in \mathcal{A}$ there exist a unique polar form $f : \mathbf{R}^n \to \mathcal{A}$ such that the following $2n$ conditions hold:

$$\begin{aligned} f(r_n, \ldots, r_1) &= p_1, \\ f(r_{n-1}, \ldots, r_1, s_1) &= p_2, \\ &\ldots \\ f(s_1, \ldots, s_n) &= p_{2n}. \end{aligned}$$

We shall use the *multi-affine* version of the de Boor algorithm to explicitly compute the value of the form f on an arbitrary bag of arguments $(u_1, \ldots, u_n)$. Indeed, the following recursive relations together with the above $2n$ conditions will produce the desired value. In particular, setting $u_1 = \ldots = u_n = u$ will evaluate the diagonal $F(u)$.

$$\begin{aligned} &f(u_1, \ldots, u_{k+1}, r_{n-m-k-1}, \ldots, r_1, s_1, \ldots, s_m) = \\ &\frac{s_{m+1} - u_{k+1}}{s_{m+1} - r_{n-m-k}} f(u_1, \ldots, u_k, r_{n-m-k}, \ldots, r_1, s_1, \ldots, s_m) + \\ &\frac{u_{k+1} - r_{n-m-k}}{s_{m+1} - r_{n-m-k}} f(u_1, \ldots, u_k, r_{n-m-k-1}, \ldots, r_1, s_1, \ldots, s_{m+1}) \end{aligned}$$

for $k = 0 \ldots n-1, m = 0 \ldots n-k-1$. Figure 5 illustrates the algorithm for $n = 3$.

Open Inventor spline curves. *Open Inventor* is a 3D object-oriented toolkit. We shall use it to illustrate a practical way to specify a spline curve. In order to create a spline curve one needs to specify two things: the control points sequence and the knot sequence. The order of the curve is implicitly determined as the difference between the number of knots and the number of control points (note that we've added dummy first and last knot.) Note also that in order to specify the curve of order M, you need at least M control points.

```
#Inventor V2.1 ascii
```

```
#file: spline.iv
Separator{
  # this is the control points sequence
  Coordinate3 {
    point [
      0 0 0,
      0 1 0,
      0 0 1,
      2 0 2,
      0 4 5,
      0 0 3
    ]
  }
  NurbsCurve {
    numControlPoints 6

    # NOTE: the values of the first and the
    # last knot do not matter in Inventor
    knotVector [
      -700, -3, -1.5, -1, 0, 1.8, 2.6, 3, 4.9, 500
    ]
  }
  # three segments of cubic spline curve
  # twice differentiable
  # order = (number of knots) - (number of control points)
  # order = 10 - 6 = 4
  # the polynomials with 4 coefficients A + B t + C t^2 + D t^3
  # cubics: degree = order - 1 = 3
}
```

Splines of degree d are C^{d-1}. Suppose that we have specified the knot sequence $t_1, \dots, t_{N+d-1}$ (plus two dummy knots t_0 and t_{N+d}) and N control points $p_1, \dots, p_N$, and that $N > d$. Also, at this point we assume that the knot sequence is strictly increasing, that is, $t_k < t_{k+1}$ for all k (later we will relax this assumption.) Then the resulting curve will consist of $N - d$ polynomial pieces segments of degree d with the parameter value u lying between t_d and t_N. For $u \in [t_k, t_{k+1}), (k = d, \dots, N-1)$ the polynomial curve $F^k$is computed by the de Boor algorithm starting with the following conditions on its blossom f^k:

$$f^k(t_{k-d+1}, \dots, t_k) = p_{k-d+1},$$

$$\dots$$

$$f^k(t_{k+1}, \dots, t_{k+d}) = p_{k+1}.$$

Note that writing out the multi-affine version of the de Boor algorithm for two neighboring segments F^{k-1} and F^k of the curve yields the following relation that holds for any values of $v_1, \dots, v_?$:

$$f^{k-1}(t_k, v_1, \dots, v_{d-1}) = f^k(t_k, v_1, \dots, v_{d-1}),$$

from which it immediately follows (see Theorem 18) that the curve is $d-1$ **times continuous differentiable.**

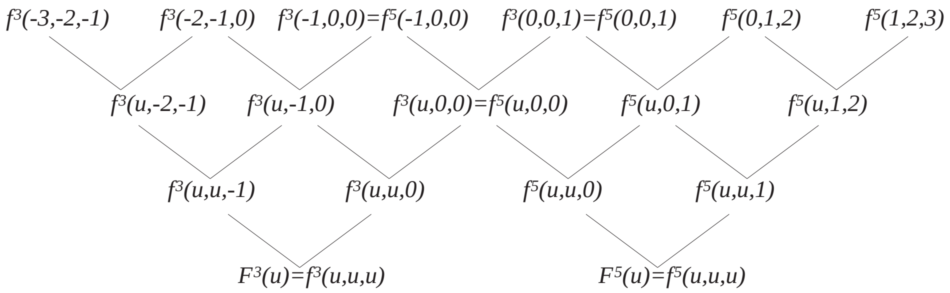

FIGURE 8. *The multi-affine version of the de Boor algorithm for cubics*

Multiple knots. Similar reasoning leads us to the conclusion that at the knot of multiplicity q we have $d-q$ continuous derivatives. Let us show that in a simple example:

EXAMPLE 20. Consider a knot sequence $t_0, t_1 = -3, t_2 = -2, t_3 = -1, t_4 = 0, t_5 = 0, t_6 = 1, t_7 = 2, t_8 = 3, t_9$, and six control points $p_1, \ldots, p_6$ that define a cubic spline curve. There is one knot of multiplicity 2, that is $t_4 = t_5 = 0$, therefore the parametric segment F^4 corresponding to the control points p_2, p_3, p_4, p_5 disappears, whereas the two curve segments F^3 and F^5 become neighbors. Figure 5 shows that $f^3(0,0,v_1) = f^5(0,0,v_1)$ for all v_1, from which it follows that the curve is continuously differentiable at the point 0 where the two segments meet.

6. References

- A paper by Lyle Ramshaw is available online at
  `http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-034.html`
- Papers by Hans-Peter Seidel.
- A book by Jean Gallier
  `Curves and Surfaces in Geometric Design : Theory and Algorithms.`