Original Lecture #4: 10 October 1991

Topics: The Polar Forms of Polynomial Curves

Scribe: from your lecturers

These notes are about *blossoming*, which is the application of polar forms to spline methods in computer-aided geometric design (CAGD). In these notes, we consider one polynomial parametric curve in isolation and we study the properties of its polar form. Later notes will go on to consider the polar forms of spline curves and surfaces.

The polar approach to the theory of splines emerged in rather different guises in three independent research efforts: Paul de Faget de Casteljau called it 'shapes through poles' [7, 8]; Carl de Boor called it 'B-splines without divided differences' [5, 6, 12]; and I called it 'blossoming' [13, 14, 15]. More recently, it has been extended and applied by various researchers.

If one picture is worth a thousand words, that picture—in the case of blossoming—is Figure 1. The dots and lines show the de Casteljau Algorithm computing a point F(t) on a planar, cubic, parametric curve F, starting from the four Bézier points of the segment F([0..1]). That part of the picture is just like in any standard text on CAGD. What's new in blossoming is the labels on the points. In particular, the function f of three arguments that appears in those labels is the *polar form* or *blossom* of the curve F.

Before we say more about Figure 1 and polar forms, let's review some basics, to make sure that we all start out on the same wavelength.

1 Modeling curved shapes

Suppose that we want to build a mathematical model for a smooth curve or surface *S* that we are designing. That is, we want to find certain systems of equations that somehow determine which points lie on *S* and which points do not. There are various questions to address.

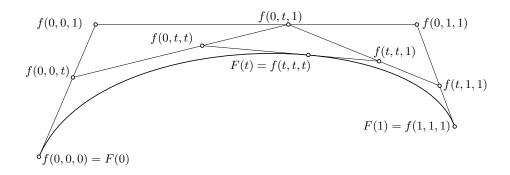


Figure 1: The de Casteljau Algorithm manipulating polar values

One piece or many? We might choose to model the entire shape S via a single system of equations. That works fine for very simple shapes. But designers of more complicated shapes want to be able to modify one region of their design without affecting the rest of it. The usual way to make this possible is to break up the shape S into pieces. We model each piece by its own system of equations, and we guarantee that the pieces join smoothly by enforcing certain constraints on the systems of equations that determine the joining pieces. The word 'spline', in its most general sense, means a piecewise model of a shape in which the pieces have been constrained to join with some level of smoothness.

Parametric or implicit? What a system of equations does, mathematically, is to specify a function. Given an input point in a space of some dimension, such a function determines an output point in a space of some—possibly different—dimension. There are two standard ways to use such functions to model shapes: *parametric models* and *implicit models*.

Our curved shape S lies in some larger, flat, ambient space, which we will call Q and refer to as the *object space*. Let s denote the dimension of the shape S and q denote the dimension of the object space Q. The two types of models are distinguished by whether the modeling function goes from some other space into Q (parametric) or from Q into some other space (implicit).

In parametric models, the shape S is defined as the range F(P) of a function $F: P \to Q$, where P is an auxiliary, s-dimensional space called parameter space. Thus, for a parametric model, the input to the modeling function is a point in parameter space—that is, values for the s parameters—and the output of the modeling function is a point on the shape S.

In *implicit models*, on the other hand, the shape S is defined by the formula $S := F^{-1}(\langle 0, ..., 0 \rangle)$, where $F : Q \to R$ is a function from the object space Q to an auxiliary space R of dimension q - s. The auxiliary space R doesn't have a standard name; *gauge space* might be good choice. For an implicit model, the input to the modeling function is a point in the object space, while the output of the modeling function is a point in gauge space—that is, values for the q - s gauges.

For example, consider the first parabola that everyone learns about: the graph of the function $y = x^2$. This shape is a smooth curve in the plane $Q = \mathbb{R}^2$, so s = 1 and q = 2.

The function $G: \mathbb{R} \to \mathbb{R}^2$ given by $G(t) := \langle t, t^2 \rangle$ is a parametric model for that parabola. The variable t here denotes a parameter value. The parameter space $P = \mathbb{R}$ of a curve is one-dimensional, so it is convenient to think of the parameter as time. The modeling function $G: P \to Q$ maps times to points on the curve S.

The function $H: \mathbb{R}^2 \to \mathbb{R}$ given by $H(\mathbf{z}) = H(\langle x, y \rangle) := y - x^2$ is an implicit model for that same parabola. The variable \mathbf{z} here denotes an arbitrary point in the object plane. The corresponding gauge value $H(\mathbf{z})$ is positive, zero, or negative according as the point \mathbf{z} lies above, on, or below the curve S.

Note that both parametric and implicit models of shapes encode some extra information, over and above the shape S itself. A parametric model, in addition to determining the

shape S, also provides a sort of roadmap to S—a way to name each point of S with a simple name. An implicit model, in addition to determining S, also associates gauge values with all of the other points in the object space. Both types of extra information sometimes come in handy.

What class of functions? A third decision that faces us is what class of functions to select from when choosing a function to model our shape S, either parametrically or implicitly.

The simplest functions are the *polynomial functions*, the functions F for which each (Cartesian) coordinate of the output point $F(\mathbf{p})$ can be written as a polynomial in the (Cartesian) coordinates of the input point \mathbf{p} . Both the parametric model $G(t) = \langle t, t^2 \rangle$ and the implicit model $H(\langle x, y \rangle) = y - x^2$ of the parabola above are polynomial functions. Note that polynomial functions can be built up using only addition, subtraction, and multiplication.

If we add division to the set of legal operations, we get the *rational functions*, where each coordinate of the output point can be written as the quotient of two polynomials in the coordinates of the input point. For example, the function

$$t \mapsto \left\langle \frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2} \right\rangle$$

is a rational function from the line to the plane. That particular function, in fact, happens to be a rational parametric model of the unit circle in the plane, as we can verify by noting that the two coordinates $x(t) = (1-t^2)/(1+t^2)$ and $y(t) = 2t/(1+t^2)$ satisfy the identity $x^2 + y^2 = 1$.

A note on nomenclature: Everyone agrees on the name 'rational function', but there is much less agreement about the name 'polynomial function'. Some people use the name 'integral function' instead, by analogy with the situation for numbers: Since a rational number with denominator 1 is an integer, they argue that a rational function whose denominators are all 1 should be called an 'integral function'. Other people use the name 'non-rational function'—but I find it awfully confusing to refer to a polynomial function as 'non-rational', given that the set of polynomial functions is a subset of the set of rational functions.

Polynomial functions and rational functions are the functions to which the technique of blossoming applies, so they are the two classes of functions that are most relevant to this course. But there are more complicated classes of functions, and they are worth mentioning briefly.

The next step up the ladder of complexity after allowing division is to allow the operation of taking square roots or, more generally, of solving polynomial equations of any degree. The resulting functions are called *algebraic*.

The next step after that is to allow the operation of summing an absolutely convergent, infinite series. The resulting functions are called *analytic* (or, to avoid confusion with

complex functions of a complex variable, *real analytic*). For example, the function $t \mapsto \langle \cos(t), \sin(t) \rangle$ is an analytic, parametric model for the unit circle in the plane.

The ladder goes even higher—there exist smooth (that is, C^{∞}) functions that are not analytic. But discussing such things would take us too far afield.

What degree? If we choose to use modeling functions that are either polynomial or rational, we can further control how complicated we allow them to be by putting bounds on the degrees of the polynomials involved.

In piecewise methods, there is a tradeoff between the number of pieces and the complexity of each piece. At one extreme are models that use a large number of simple pieces; for example, we might use a polyline with hundreds of segments to model the outline of a character in a printing font. At the other extreme are models that use only a few pieces, but in which the degree of each piece is relatively high.

What answers to those questions are relevant for these notes?

In current practice, models for all but the simplest shapes are piecewise in nature. In these notes, we will restrict ourselves to single pieces in isolation. Later, we will assemble those pieces into splines.

Curves are important in their own right, and they are mathematically simpler than surfaces. Furthermore, one of the most important classes of surfaces—the tensor-product surfaces—are best understood as curves of curves. Hence, we will be studying curves to start off with. Later, we will consider the generalizations to surfaces.

Curves in the plane can be conveniently modeled either parametrically or implicitly, and we will say a little bit about both types of models. But we will focus most of our attention on parametric models. While our examples will often be parametric curves in the plane, the methods apply equally well to parametric curves in object spaces of any dimension.

In these notes, we will restrict our attention to polynomial modeling functions. Another topic for later is the generalization from the polynomial to the rational case.

So, in these notes, we are going to be considering one-piece, polynomial, parametric models of curves in the plane.

2 Bézier points and the de Casteljau Algorithm

Suppose that we want to implement a computer package for drawing segments of planar, polynomial, parametric curves of degree at most *n*. What is a good design for the interface to our package? That is, in what format should we ask our clients to specify to us which segment of which curve they would like us to draw?

The follow-your-nose approach is as follows. Every planar, polynomial, parametric curve of degree at most n can be written in the form $F(t) = \langle x(t), y(t) \rangle$, where

$$x(t) := a_n t^n + \dots + a_1 t + a_0$$

 $y(t) := b_n t^n + \dots + b_1 t + b_0$

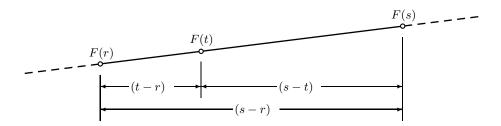


Figure 2: The de Casteljau Algorithm in the case n = 1

for some real coefficients a_0 through a_n and b_0 through b_n . We ask our clients to tell us those 2n+2 coefficients and to tell us also the end-points r and s of the interval [r..s] in parameter space corresponding to the segment of the bi-infinite curve F that they want drawn. Given this information, we can draw for them the segment F([r..s]).

A nit-picking note: We said 'of degree at most n' when defining F above because we want to include the special case where both of the high-order coefficients a_n and b_n happen to be zero, causing the degree of F (which is the maximum of the degrees of its two coordinate polynomials) to be strictly less than n. In what follows, we shall interpret the adjectives 'quadratic', 'cubic', and the like in that same inclusive fashion. For example, we interpret the phrase 'a cubic function' to mean a function whose degree is at most 3, as opposed to precisely 3.

2.1 The case n = 1

For n=1, the curve segments that we are volunteering to draw are line segments, so there is clearly a better interface to our drawing package than the one that comes from following your nose. Instead of having our clients tell us the six numbers a_0 , a_1 , b_0 , b_1 , r, and s, we have them tell us simply the s and s coordinates of the starting point s for s and s and the ending point s for s and s for s and s and the ending point s for s for s and s for s

This endpoint-based scheme is an improvement in many ways: It involves only four numbers, instead of six. It makes it easy to arrange that one line segment starts precisely where the previous one stopped; in fact, we can save two additional numbers for each such joint. Also, the numbers used in the endpoint-based scheme are the coordinates of graphically relevant points, so it is easy to see how precisely they need to be stated.

Let t be some parameter value in the interval [r..s]. Once we know the endpoints F(r) and F(s), we can locate F(t) using linear interpolation, as shown in Figure 2. Each coordinate of the point F(t) is (t-r)/(s-r) of the way from that coordinate of F(r) to that coordinate of F(s). Real simple.

But suppose that the degree n is greater than 1. Can we devise a scheme for the n-ic case that has the same simplicity and good features as specifying the endpoints has in the linear case?

2.2 The case n = 2

When n = 2, what curves are we volunteering to draw? Answer: Segments of parabolas. That is, if x(t) and y(t) are quadratic polynomials in the parameter t, the point $F(t) := \langle x(t), y(t) \rangle$ varies along a parabola.

Proving this in detail would take us too far afield, but here is a rough sketch: Eliminating t from the two quadratic equations $x = a_2t^2 + a_1t + a_0$ and $y = b_2t^2 + b_1t + b_0$ will give us a polynomial equation G(x,y) = 0 of total degree certainly no worse than 4. The function G models implicitly the same curve that the function F models parametrically, so the process of going from F to G is called *implicitization* [10]. In our current case, the total degree of G turns out to be only 2. One way to verify that is to calculate G in gory detail:

$$G(x,y) = b_2^2 x^2 - 2a_2b_2xy + a_2^2 y^2 + (b_1(a_1b_2 - a_2b_1) - 2b_2(a_0b_2 - a_2b_0))x - (a_1(a_1b_2 - a_2b_1) - 2a_2(a_0b_2 - a_2b_0))y + (a_0b_2 - a_2b_0)^2 - (a_0b_1 - a_1b_0)(a_1b_2 - a_2b_1).$$

A neater way is to observe that the curve modeled by F cannot intersect any line at more than two points, since, if the line is given by the implicit equation cx + dy + e = 0, the parameter value t of any intersection point must be a root of the quadratic equation cx(t) + dy(t) + e = 0. Since the curve intersects an arbitrary line at most twice, the degree of its implicit model G is at most 2, hence the curve is some kind of conic section. But F cannot model an ellipse, since the point F(t) goes off to infinity as t goes to infinity, while an ellipse is bounded. Furthermore, at t goes to infinity, the ratio of the x and y coordinates of the point F(t) approaches the single limiting ratio $a_2 : b_2$. Thus, F cannot model a hyperbola either, since hyperbolas go off to infinity with two different limiting ratios. So the curve that F models must be a parabola.

Our clients have to tell us which segment of which parabola they want drawn. The x and y coordinates of the two endpoints of the parabolic segment seem like good candidates for four of the six numbers that we need. But which other two numbers should we ask for? The winning idea is to ask for the x and y coordinates of the point where the starting and ending tangent lines to the parabolic segment intersect. The resulting three points—the starting point, the intersection of the starting and ending tangents, and the ending point—are called the $B\acute{e}zier$ points of the parabolic segment; Figure 3 shows an example.

Specifying a parabolic segment by giving the coordinates of its three Bézier points has lots of attractive properties. Six is the right number of numbers to be specifying. The six specified numbers all have geometric immediacy. It is easy to arrange that one parabolic segment will start precisely where the last one ended. It is even fairly easy to arrange that two joining segments will have the same tangent line at the joint: Make the last two Bézier points of the incoming segment collinear with the first two Bézier points of the outgoing segment.

There is another, less obvious advantage to using Bézier points. Suppose that we know the three Bézier points of a parabolic segment F([r..s]) and that we want to compute the location of the point F(t), for some t in the interval [r..s]. It turns out that we can locate F(t) using three linear interpolations, all with the same ratio (t-r)/(s-r) that appeared in the linear case.

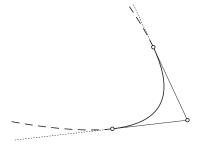


Figure 3: The three Bézier points of a parabolic segment

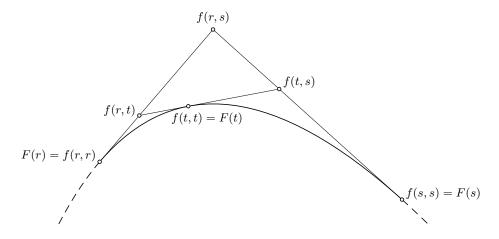


Figure 4: The de Casteljau Algorithm in the case n = 2

This process is called the *de Casteljau Algorithm*, and is illustrated in Figure 4. The points in Figure 4 are labeled as the values of a two-argument function f that we will get around to defining in just a moment.

The points f(r,r), f(r,s), and f(s,s) are the input data, the three Bézier points of the parabolic segment F([r..s]). In the first linear interpolation, we go (t-r)/(s-r) of the way from f(r,r) to f(r,s), and we call the resulting point f(r,t). In the second interpolation, we go (t-r)/(s-r) of the way from f(r,s) to f(s,s), and we call the resulting point f(t,s). In the final linear interpolation, we go (t-r)/(s-r) of the way from f(r,t) to f(t,s), and we claim that the resulting point f(t,t) is, in fact, F(t).

But what is the function f that appears in these labels? Here is a geometric definition: For any two distinct parameter values u and v, the point f(u,v) is the intersection of the tangents to the parabola F at the points F(u) and F(v). To handle the special case where the two arguments of f are equal, we define f(u,u) to be simply F(u). Since f(u,v) approaches F(u) as v approaches u, we have to define f(u,u) to be F(u) if we want f to be continuous. The resulting function f satisfies the identity f(u,v)=f(v,u), since intersection is a symmetric operation. For example, the middle Bézier point of the segment F([r..s]) can be equally well labeled f(r,s) or f(s,r). In Figure 4, the issue of whether to write f(u,v) or f(v,u) was resolved, in each case, by putting the two arguments in increasing numeric order, where

r < t < s.

The first linear interpolation in Figure 4 locates f(r,t) along the line joining f(r,r) to f(r,s). The first argument to the function f is staying fixed at r in this interpolation, while the second argument varies. Note that, as t varies, the point f(r,t) moves at a constant rate along the line joining f(r,r) to f(r,s). The second interpolation locates f(t,s) along the line joining f(r,s) to f(s,s). In this case, the second argument is staying fixed at s, while the first argument varies. The third interpolation locates f(t,t) along the line joining f(r,t) to f(t,s). Stated in this form, we can't think of the third interpolation as having one argument that stays fixed while the other varies. But remember that f is a symmetric function. Hence, we could equally well rewrite the left end-point f(r,t) as f(t,r), in which case the first argument would stay fixed at t.

Note that the left subsegment $F([r\mathinner{..}t])$ is a parabolic segment in its own right, as is the right subsegment $F([t\mathinner{..}s])$. From Figure 4, we can see that the Bézier points of the subsegment $F([r\mathinner{..}t])$ are f(r,r), f(r,t), and f(t,t). No surprise here: The subsegment $F([r\mathinner{..}t])$ has exactly the same relationship to its Bézier points as the original segment $F([r\mathinner{..}s])$ does. This means that, in the process of computing the point F(t), we have also computed all three Bézier points of both of the subsegments $F([r\mathinner{..}t])$ and $F([t\mathinner{..}s])$. Hence, the de Casteljau Algorithm can be used as the core of a divide-and-conquer rendering algorithm for segments of parabolas. When used in this way, it is common to choose t to be (r+s)/2, so that the only arithmetic operations necessary are addition and division by 2, the latter of which can be implemented with a right shift.

The heart of the de Casteljau Algorithm in the case n=2 is the fact that, as we vary t, the tangent line to a parabola F at F(t) intersects any fixed tangent line in a point that moves along that fixed tangent line at a constant rate of speed. We haven't proved that fact yet, but carpenters have exploited it for a long time, as shown in Figure 5. When cutting out a kitchen countertop, they round off the corners using the following rule: Choose a distance d. Mark the points at distance d, 2d, and 3d from the original, right-angled corner along each edge. Connect the two points at distance 2d to each other, and cross-connect the point at distance d along one edge to the point at distance 3d along the other edge and vice versa. Cut along the three resulting lines and file off the four remaining blunt corners. The resulting boundary curve is a parabolic segment that has the original corner as its middle Bézier point and the points at distance 4d from that corner along each edge as its first and last Bézier points. In Figure 5, the points are labeled assuming that this parabolic segment—call it F—has been parameterized from t=0 at the upper left to t=4 at the lower right; that is, the segment is F([0..4]). Note that each of the five tangent lines is divided into four segments of equal length. (Note also that rounding the corner off with a quarter-circle of radius 4d, as shown dotted in Figure 5, would result in a smaller countertop.)

The easiest way to prove that the de Casteljau Algorithm works as advertised is to write down an explicit, algebraic formula for the bivariate function f:

$$f(u,v) := \left\langle a_2 u v + a_1 \frac{u+v}{2} + a_0, \ b_2 u v + b_1 \frac{u+v}{2} + b_0 \right\rangle. \tag{1}$$

The function f thus defined is clearly symmetric; that is, f(u,v) = f(v,u). If we evaluate f(t,t),

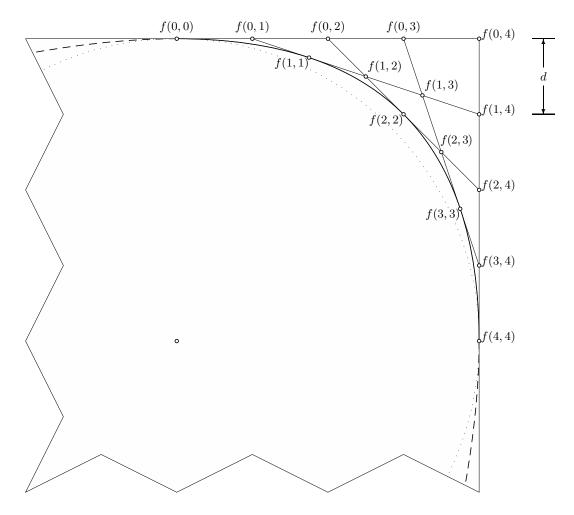


Figure 5: de Casteljau's kitchen counter

we get $f(t,t) = \langle a_2t^2 + a_1t + a_0, b_2t^2 + b_1t + b_0 \rangle$, so f satisfies the identity f(t,t) = F(t). And f has the property that, if we hold one argument fixed and vary the other, the resulting point moves along a straight line at a constant rate of speed. Those three properties are enough to justify everything that we did during the de Casteljau Algorithm. That is, assuming that f(r,r), f(r,s), and f(s,s) are the points so labeled in Figure 4, those three properties are enough to imply that F(t) = f(t,t) is the result of the three linear interpolations shown in that figure.

The one property of the function f that isn't obvious from the algebraic formula in equation (1) is the geometric fact by which we originally defined f: The point f(u,v) is the intersection of the tangent lines to the parabola F at the points F(u) and F(v). To verify that relationship between the geometry and the algebra, we would have to study the relationship between f and the derivatives of F, which we won't get to in these notes.

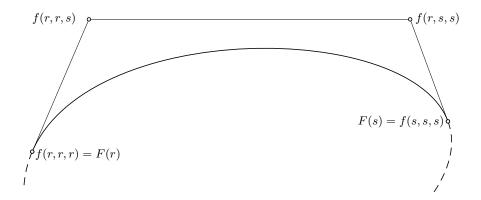


Figure 6: The four Bézier points of a cubic segment

2.3 The case n = 3

The same tricks that worked for n = 2 work, in pretty much the same way, for n = 3. One difference is that the polynomial parametric cubics are a less well-known class of curves than the parabolas. Another difference is that we need six linear interpolations in the de Casteljau Algorithm when n = 3.

Let F([r..s]) be a segment of a planar, cubic, polynomial, parametric curve. That is, the x and y coordinates of $F(t) = \langle x(t), y(t) \rangle$ are given by cubic polynomials in t:

$$x(t) := a_3t^3 + a_2t^2 + a_1t + a_0$$

 $y(t) := b_3t^3 + b_2t^2 + b_1t + b_0.$

The cubic segment F([r..s]) has four Bézier points, which—in the blossoming approach—are labeled f(r,r,r), f(r,r,s), f(r,s,s), and f(s,s,s), as shown in Figure 6. Note that the segment F([r..s]) starts out at F(r) = f(r,r,r), heading towards f(r,r,s). It ends at F(s) = f(s,s,s), coming from f(r,s,s).

The labeling function f has three arguments this time, because we are dealing with cubics. It is a symmetric function of its three arguments; it satisfies the identity f(t,t,t) = F(t); and it has the property that, if we hold two arguments fixed and vary the third, the resulting value moves along a straight line at a constant rate. Those three properties are easy to verify, working from the algebraic formula for f, which is as follows: $f(u,v,w) = \langle x(u,v,w), y(u,v,w) \rangle$, where

$$x(u,v,w) := a_3uvw + a_2\frac{uv + uw + vw}{3} + a_1\frac{u + v + w}{3} + a_0$$

$$y(u,v,w) := b_3uvw + b_2\frac{uv + uw + vw}{3} + b_1\frac{u + v + w}{3} + b_0.$$
(2)

Given the four Bézier points of the segment F([r..s]), we can compute the point F(t) = f(t,t,t) by performing six linear interpolations, as shown in Figure 7—the cubic case of the de Casteljau Algorithm. All six interpolations are controlled by the same ratio (t-r)/(s-r). The effect of the interpolations is to bring more and more copies of t, the desired parameter

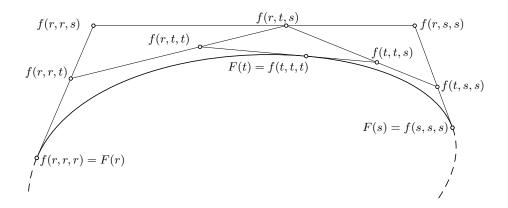


Figure 7: The de Casteliau Algorithm in the case n = 3

value, into the list of arguments to f. The following triangular array records the progress of the de Casteljau Algorithm symbolically:

$$f(r,r,r) \quad f(r,r,s) \quad f(r,s,s) \quad f(s,s,s)$$

$$f(r,r,t) \quad f(r,t,s) \quad f(t,s,s)$$

$$f(r,t,t) \quad f(t,t,s)$$

$$f(t,t,t)$$
(3)

The four points in the top row are the input. Each remaining point is computed by linearly interpolating between the two points diagonally above it.

In the process of computing the point F(t) = f(t,t,t), note that we also compute all four Bézier points of each of the two subsegments F([r..t]) and F([t..s]), just as happened in the quadratic case. The Bézier points of the left subsegment F([r..t]) are f(r,r,r), f(r,r,t), f(r,t,t), and f(t,t,t), which are the four points on the lower-left side of array (3), while the Bézier points of F([t..s]) form the lower-right side.

One thing that is missing in the cubic case is a geometric way of defining the labeling function f, that is, a rule analogous to the intersect-the-two-tangents rule in the quadratic case. Without such a rule, it isn't obvious—when given a cubic segment such as the one in Figure 6—how far out along the tangent line at F(r) the second Bézier point f(r,r,s) belongs. It turns out that there is such a geometric rule for twisted cubics, but not for planar cubics.

The four Bézier points f(r,r,r), f(r,r,s), f(r,s,s), and f(s,s,s) in Figure 6 happen to be coplanar. If we perturbed them to destroy this coincidence by moving them up out of the paper or down through it, the de Casteljau Algorithm would continue to work with no difficulty, but the resulting curve F would be a twisted cubic in 3-space. For a twisted cubic F, the point f(u,v,w) can be defined geometrically as the intersection of the osculating planes to F at the three points F(u), F(v), and F(w). (The osculating plane to a space curve at a point is the plane through that point which comes closest to containing the curve locally; it is spanned by the velocity and acceleration vectors.) Note that, by this rule, the second Bézier point f(r,r,s) in Figure 6 should be in the osculating plane to F at F(r) twice, which—it turns out—means

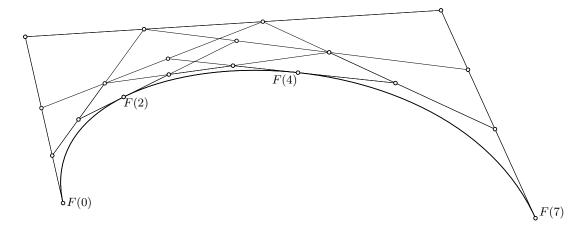


Figure 8: Computing F(2) and F(4) on the cubic segment F([0..7])

that f(r,r,s) should be on the tangent line to F at F(r). The point f(r,r,s) should also be in the osculating plane to F at F(s) once. Thus, the second Bézier point f(r,r,s) is the point where the tangent line at F(r) intersects the osculating plane at F(s).

Unfortunately, trying to apply this osculating-plane rule to a planar cubic leads to a degenerate situation, because all of the osculating planes of a planar cubic coincide. Fortunately, the algebraic formula for f in equation (2) and the algebraic properties of f that follow from that formula are all that we really need. A geometric definition of f in the planar case would be nice, but we can get along perfectly well without it.

Recall that the carpenter's technique for rounding off a kitchen countertop corresponds, in our context, to applying the de Casteljau Algorithm repeatedly to the same parabolic segment. Doing the analogous thing for cubic segments is a bit more complicated. Figure 8 shows an example, in which we compute both the points F(2) and F(4) on the cubic segment F([0..7]). As an exercise, label each of the twenty distinguished points in Figure 8 with a label of the form f(u,v,w), where u,v, and w are three numbers drawn from the set $\{0,2,4,7\}$. Since we consider two labels to be the same if they differ only in the order of the three arguments to f, there are precisely twenty such labels, which is just enough. When you are done, the four points along any one of the ten lines should have labels of the form f(0,u,v), f(2,u,v), f(4,u,v), and f(7,u,v), for some u and v drawn from the set $\{0,2,4,7\}$.

There are two intersections of line segments in Figure 8 that aren't distinguished by little dots. Those two intersections are artifacts of a degeneracy: namely, the planarity of F. If we perturbed the Bézier points in Figure 8 up or down, thus converting F into a twisted cubic, those two intersections would disappear—that is, the two lines in each of those two intersecting pairs would become skew. (If we redrew Figure 8 with full lines, rather than line segments, those two unstable intersections would be joined by 19 more. The points distinguished in Figure 8 are precisely the intersections that are stable under vertical perturbations of the Bézier points.)

3 The polar forms of univariate polynomials

It is time to back up all this geometry with some algebra.

In Section 2, we started with a polynomial, parametric curve F of degree n and found that, in labeling the resulting diagrams, it was helpful to consider a function f that takes n different parameter values as its arguments. The correspondence between F and f is one instance of the general, algebraic principle of polar forms: that is, f is the *polar form* of F. In essence, the principle of polar forms says that we can trade one parameter of degree n, such as the parameter t in the expression F(t), for n symmetric parameters, each of degree 1, such as the parameters u_1 through u_n in the expression $f(u_1, \ldots, u_n)$. Before we discuss how to perform this trade, we first pause to clarify our nomenclature for the concept 'of degree 1'; in particular, we must distinguish between the adjectives 'linear' and 'affine'.

The adjective 'linear' is used inconsistently in mathematics, in that it sometimes implies homogeneity and sometimes doesn't. A univariate polynomial F(x) is called linear if it has the form F(x) = ax + b, where $b \neq 0$ is usually allowed. If we reinterpret $F \colon \mathbb{R} \to \mathbb{R}$ as a transformation of the 1-dimensional vector space \mathbb{R} , however, we must have b = 0 in order for F to be called a linear transformation; that is, F must also be homogeneous. To avoid confusion in what follows, we make the convention that, for the rest of these notes, the word 'linear' will mean 'of degree 1 and homogeneous', while the word 'affine' will mean 'of degree 1, but not necessarily homogeneous'. For example, the polynomial F(x) = x + 1 is not linear, but it is affine. (Will this convention eventually take over all of mathematics? Note that, under this convention, the problems that the simplex algorithm solves are affine programming problems, not linear programming problems.)

A linear function is a function F that commutes with linear combinations; that is, $F(\sum_i \lambda_i x_i) = \sum_i \lambda_i F(x_i)$. An *affine function* can be defined in two ways: Either it is the sum of a linear function and a constant, or it is a function F that commutes with affine combinations, where an *affine combination* is a linear combination whose scalar coefficients λ_i sum to 1. That is, for an affine function F, we have $F(\sum_i \lambda_i x_i) = \sum_i \lambda_i F(x_i)$ whenever $\sum_i \lambda_i = 1$, but not necessarily otherwise. Note that the linear combinations that occur in the de Casteljau Algorithm are actually affine combinations. For example, the linear combination

$$F(t) = \frac{s-t}{s-r}F(r) + \frac{t-r}{s-r}F(s)$$

shown in Figure 2 is an affine combination, because (s-t)/(s-r)+(t-r)/(s-r)=1.

For future reference, an *affine space* (or *flat*) is a set of points that is closed under the operation of taking affine combinations. An *affine frame* for an affine space P is a set of points in P that are affinely independent and whose affine span is all of P; that is, an affine frame is the analog of a linear basis. Every affine frame for a d-dimensional space contains precisely d+1 points.

The word 'affine' is helpful in describing the algebraic properties of polar forms—that is, of the labeling functions f that we used in Section 2. For example, the polar form f of the parametric, cubic curve F in Figure 7 has the property that, if we hold u and v fixed and vary

w, the point f(u,v,w) moves along a straight line at a constant rate. We can restate that same property more concisely by saying that, for any fixed u and v, the point f(u,v,w) is an affine function of w. (In order for it to be a linear function of w, we would have to have f(u,v,0) = 0 as well—which isn't true, in general.)

Let's call a function f of n arguments multiaffine or n-affine if the value $f(u_1, \ldots, u_n)$ is an affine function of each argument u_i whenever the other arguments are held fixed at arbitrary values. If F(t) is a univariate polynomial of degree at most n, a polar form of F is an n-variate polynomial $f(u_1, \ldots, u_n)$ with the following three properties:

- f is n-affine; that is, $f(u_1, \ldots, u_n)$ is an affine function of each variable u_i when the others are treated as constants;
- f is symmetric; that is, $f(u_1, u_2, u_3, \dots, u_n) = f(u_2, u_1, u_3, \dots, u_n)$ and so forth for all permutations of the n variables u_1 through u_n ;
- f satisfies the correspondence identity f(t,...,t) = F(t).

Our next goal is to prove that a polar form always exist and that it is unique. For notational convenience, we will carry out that proof first for a particular, example—the cubic polynomial $G(t) := t^3 + 3t^2 - 6t - 8$. The proof generalizes without difficulty to arbitrary univariate polynomials F and to arbitrary values of the degree bound n.

From the definition above, a polar form g for G—if one exists—is a symmetric, triaffine polynomial g(u, v, w) that satisfies the identity g(t, t, t) = G(t). In order for a trivariate polynomial g to be triaffine, it must have the form

$$g(u,v,w) = c_1uvw + c_2uv + c_3uw + c_4vw + c_5u + c_6v + c_7w + c_8$$

for some real constants c_1 through c_8 . That is, no term in g can include any variable raised to any power higher than 1, or else g would not be an affine function of that variable. To make the correspondence identity g(t,t,t)=G(t) hold, we must arrange that $c_1=1$, $c_2+c_3+c_4=3$, $c_5+c_6+c_7=-6$, and $c_8=-8$. To make g(u,v,w) a symmetric function of its three arguments, we must have $c_2=c_3=c_4$ and $c_5=c_6=c_7$. We are left with the unique choice

$$g(u, v, w) = uvw + uv + uw + vw - 2u - 2v - 2w - 8.$$

Thus, our example cubic polynomial G has this trivariate polynomial g as its unique polar form. Going from G to g is the hard direction; note that it is easy to go the other way. If we are given the trivariate polynomial g(u,v,w) := uvw + uv + uw + vw - 2u - 2v - 2w - 8, the correspondence identity G(t) = g(t,t,t) trivially determines a unique cubic polynomial G: We simply substitute t for each of u, v, and w, getting $G(t) = t^3 + 3t^2 - 6t - 8$.

From these observations, we deduce that the two polynomials G(t) and g(u,v,w) are actually two different aspects of the same entity. That entity can be viewed either as a cubic function of the single parameter t or as a symmetric, triaffine function of the three parameters u, v, and w. The quantity G(t) = g(t,t,t) varies cubically as a function of t because, when t varies, all three of u, v, and w are varying in parallel.

The same arguments apply to an arbitrary, univariate polynomial F of degree at most n. We can always expand such a polynomial F(t) as a linear combination of powers of t: $F(t) = \sum_k a_k t^k$. The only way to get a polar form $f(u_1, \ldots, u_n)$ for F is to replace t^k in this linear combination, for each k from 0 to n, by the arithmetic average of all possible products of precisely k of the variables u_1 through u_n . For example, when n = 4, we must replace t^2 by

$$\frac{u_1u_2 + u_1u_3 + u_1u_4 + u_2u_3 + u_2u_4 + u_3u_4}{6}.$$

And, of course, going backward from f to F is easy.

We restate these results more formally in the following theorem, which is the nonhomogeneous, univariate version of the principle of polar forms.

Theorem 1. Univariate polynomials F(t) of degree at most n are equivalent to symmetric, n-affine polynomials $f(u_1,...,u_n)$ in the sense that, given a polynomial of either type, there exists a unique polynomial of the other type that satisfies the correspondence identity F(t) = f(t,...,t).

Definition 2. If F(t) is a polynomial of degree at most n, the polar form of F (or the n-polar form, if the intended degree bound n isn't obvious from the context) is the unique symmetric, n-affine polynomial $f(u_1, \ldots, u_n)$ that corresponds to F via the identity $F(t) = f(t, \ldots, t)$. A value $f(u_1, \ldots, u_n)$ of the polar form is a polar value of F, and each u_i that helps to determine such a value is a polar argument to F. In contrast, F itself is the diagonal form of F; a value F(t) is a diagonal value of F; and the t that determines such a value is a diagonal argument to F. Note that diagonal values are a special case of polar values, the case in which all n of the polar arguments are equal.

Multivariate polynomials F also have polar forms, but the concept of a degree bound is more complicated in the multivariate case. We can either bound the total degree in all of the variables or bound the degree in each variable separately. Those different ways of bounding the degree give rise to different polar forms, as we will see in later notes about surfaces.

In Theorem 1, neither F nor f is assumed homogeneous. When the diagonal form F is homogeneous, the corresponding polar form f turns out to be multilinear, as opposed to merely multiaffine. Conversely, when the polar form f is multilinear, the diagonal form is homogeneous. This homogeneous version of the principle of polar forms isn't very interesting, however, until we generalize to consider multivariate diagonal polynomials F, because a univariate polynomial F(t) must be a scalar multiple of t^n in order to be homogeneous of degree n. The n-polar form of the homogeneous n-ic $F(t) = t^n$ is, of course, the n-linear monomial $f(u_1, \ldots, u_n) = u_1 \cdots u_n$.

While polar forms are new to CAGD, their use has long been standard in other areas of mathematics. For example, consider quadratic forms and bilinear forms in linear algebra. It is well known that, for each quadratic form $F: V \to \mathbb{R}$ on a vector space V, there is a unique symmetric, bilinear form $f: V \times V \to \mathbb{R}$ that satisfies the identity $F(\mathbf{v}) = f(\mathbf{v}, \mathbf{v})$. That fact

is precisely the analog of Theorem 1 for multivariate polynomials F that are homogeneous of total degree 2.

Even though many areas of mathematics use polar forms, the terminology based on the word 'polar' has fallen out of favor in the last half-century. While algebra books up through van der Waerden [19] generally used the name 'polar form', more recent books often leave the correspondence of Theorem 1 nameless. Because I started out unaware of the name 'polar form', I proposed an alternative system of nomenclature in which f was called the 'blossom' of F [13, 14]. One advantage of that proposal was that the word 'blossoming' suggests the revealing of hidden structure, which describes rather well what happens when we convert from F to f. Before converting, all that we can do is to evaluate F(t) for various t, which corresponds to computing various diagonal values f(t,t,t). After converting, we are free to let the three polar arguments take on different values, thus computing arbitrary polar values f(u,v,w). We have thus revealed more of the structure that was hidden in F. The word 'polarization', on the other hand, suggests concentration into opposing extremes, which is not at all the right idea.

While 'blossoming' is a nicer name than 'polarization' for the process of converting from F to f, I like 'polar form' better than 'blossom' as a name for the function f itself, because it suggests a closer connection between F and f. 'The diagonal form F' and 'the polar form f' sound like two different aspects of the same underlying entity, which—in my opinion—is the proper point of view. 'A polynomial F' and 'its blossom f', on the other hand, sound like two different things. Note also that the phrase 'a polar value of F' is shorter and simpler than 'a value of the blossom of F'.

The words 'pole' and 'polar' have various uses in mathematics that are unrelated to Theorem 1: the poles of a sphere, polar coordinate systems, and the poles of a complex analytic function. 'Pole' and 'polar' are also used in projective geometry, when discussing conic sections and quadric surfaces. Those uses are related to Theorem 1; in particular, they arise from considering the polar forms of the implicit models of conics and quadrics, as we will discuss briefly in Section 4.2.

4 The polar forms of curves

4.1 The parametric case

If F is a polynomial parametric curve, each coordinate of the varying point F(t) is given by a polynomial in the parameter t. To compute the polar form f of the curve F, we compute the polar forms of each coordinate polynomial separately. The resulting function f is precisely the type of labeling function that we first met in Section 2.

Let's consider a simple example of this process in algebraic detail. As our example curve, we will take the standard parabola in the plane, modeled parametrically by $F(t) := \langle t, t^2 \rangle$. The polar form of this parabola is, by definition, the unique symmetric, biaffine function f(u,v) that satisfies f(t,t) = F(t). Computing as in the proof of Theorem 1, we find that the polar

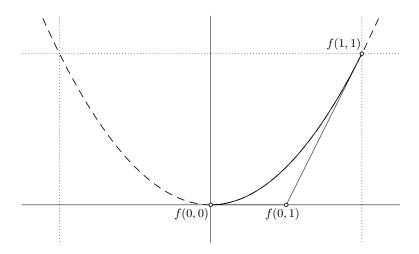


Figure 9: The segment F([0..1]) of the parabola $F(t) = \langle t, t^2 \rangle$

form f is given by

$$f(u,v) := \left\langle \frac{u+v}{2}, uv \right\rangle. \tag{4}$$

Given this formula for f, various questions become easy to answer.

For example, suppose that we were interested in the Bézier points of some segment of the parabola, say the segment F([0..1]). The three Bézier points of that segment are simply the following three polar values of $F: f(0,0) = \langle 0,0 \rangle$, $f(0,1) = \langle 1/2,0 \rangle$, and $f(1,1) = \langle 1,1 \rangle$, as shown in Figure 9.

The parabola F is a quadratic curve, but we can—if we choose—view it as a degenerate example of a cubic. The polar form f given in equation (4) is the 2-polar form of F, that is, the polar form of F interpreted as a quadratic. When we view F as a degenerate cubic, its polar form is the unique symmetric, triaffine function g that satisfies the identity g(t,t,t) = F(t). Computing once again as in Theorem 1, we find that the 3-polar form g of the parabola F is given by

$$g(u,v,w) := \left\langle \frac{u+v+w}{3}, \frac{uv+uw+vw}{3} \right\rangle. \tag{5}$$

Suppose that we wanted to know the Bézier points of the segment F([0..1]), viewed as a degenerate cubic segment. We can find them by plugging 0 and 1, in all possible ways, into the 3-polar form g, as follows:

$$g(0,0,0) = \langle 0,0 \rangle$$

 $g(0,0,1) = \langle \frac{1}{3},0 \rangle$
 $g(0,1,1) = \langle \frac{2}{3},\frac{1}{3} \rangle$
 $g(1,1,1) = \langle 1,1 \rangle$.

See Figure 10.

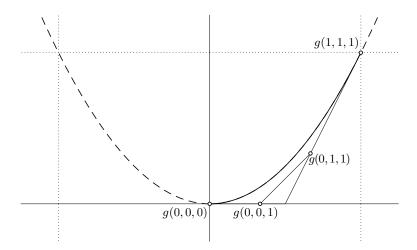


Figure 10: The same segment F([0..1]) viewed as a degenerate cubic

Viewing an n-ic curve as a degenerate example of a curve of degree n+1 in this way is called *degree raising*. To study degree raising via blossoming, we want to determine the relationship between the n-polar form and (n+1)-polar form of the same n-ic curve. For example, let F be a quadratic curve and let f be its 2-polar form. The 3-polar form g of F always satisfies

$$g(u, v, w) = \frac{f(u, v) + f(u, w) + f(v, w)}{3}.$$
 (6)

To verify this formula, it is enough to observe three things about the expression on the right-hand side: It is symmetric in the three variables u, v, and w; it is an affine function of each of those three variables when the other two are held fixed; and it simplifies to F(t) in the diagonal case u = v = w = t. Those three conditions mean that the expression on the right-hand side is a 3-polar form of F, and polar forms are unique.

From equation (6), we can easily derive the rules for degree raising in terms of Bézier points. For example, as shown in Figure 10 for the case r = 0 and s = 1, the second Bézier point g(r,r,s) of the (degenerate) cubic segment F([r..s]) is given by

$$g(r,r,s) = \frac{f(r,r) + 2f(r,s)}{3};$$

that is, g(r, r, s) is two-thirds of the way from f(r, r) to f(r, s).

4.2 The implicit case

Just as polar forms are helpful when analyzing polynomial parametric models of shapes, they are also helpful when analyzing polynomial implicit models. In the next few paragraphs, we will consider the implicit case briefly.

As our example, we will take the function $H(\mathbf{z}) = H(\langle x, y \rangle) := y - x^2$, which is an implicit model for the same standard parabola that the function $F(t) = \langle t, t^2 \rangle$ above models parametri-

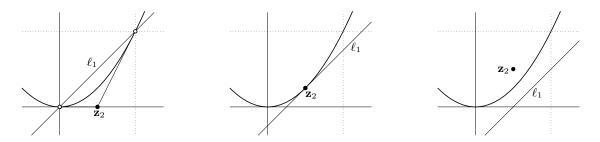


Figure 11: Pole points and polar lines of the parabola $H(x,y) = y - x^2$.

cally. Note that H is a multivariate polynomial of total degree 2. Theorem 1 doesn't cover the multivariate case, but the polar form h of H that corresponds to a bound of 2 on the total degree turns out to be given by

$$h(\mathbf{z}_1, \mathbf{z}_2) = h(\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle) := \frac{y_1 + y_2}{2} - x_1 x_2.$$
 (7)

The function h is a symmetric, biaffine function from the plane to the line that satisfies the correspondence identity $h(\mathbf{z}, \mathbf{z}) = H(\mathbf{z})$.

By the way, in order for $h(\mathbf{z}_1, \mathbf{z}_2)$ to be an affine function of the point \mathbf{z}_1 when \mathbf{z}_2 is held fixed, each term in h can contain either a factor of x_1 or a factor of y_1 or neither of them, but cannot contain both. In the case of H, there was no temptation to include both x_1 and y_1 in the same term of the polar form, since no term of H included both x and y. For a more instructive example on this point, consider the bivariate polynomial $K(\mathbf{z}) = K(\langle x, y \rangle) = xy - 1$, which implicitly models the standard rectangular hyperbola. The polar form k of K is given by

$$k(\mathbf{z}_1,\mathbf{z}_2) = k(\langle x_1,y_1\rangle,\langle x_2,y_2\rangle) := \frac{x_1y_2 + x_2y_1}{2} - 1.$$

Returning to the parabolic example of H and h, what characteristics of the functions H and h are we interested in? The implicit model H associates a gauge value $H(\mathbf{z})$ with each point \mathbf{z} in the plane. This gauge value varies quadratically as the point \mathbf{z} varies, and the parabola itself is the set of all points whose associated gauge value is zero. The polar form h of H associates gauge values with unordered pairs of points $(\mathbf{z}_1, \mathbf{z}_2)$. Which pairs $(\mathbf{z}_1, \mathbf{z}_2)$ have the property that $h(\mathbf{z}_1, \mathbf{z}_2) = 0$?

If we fix \mathbf{z}_2 , the equation $h(\mathbf{z}_1, \mathbf{z}_2) = 0$ constrains \mathbf{z}_1 to lie on some line, call it $\ell_1 := \{\mathbf{z}_1 \mid h(\mathbf{z}_1, \mathbf{z}_2) = 0\}$. Figure 11 shows three examples. If \mathbf{z}_2 is outside the parabola, as on the left, then ℓ_1 is the line joining the two points where lines through \mathbf{z}_2 are tangent to the parabola. Note the similarity between this diagram and Figure 9. If \mathbf{z}_2 lies on the parabola, as in the middle of Figure 11, then ℓ_1 is the tangent line through \mathbf{z}_2 . And if \mathbf{z}_2 is inside the parabola, as on the right, then ℓ_1 lies entirely outside it. (Actually, the right-hand case is just like the left-hand case, except that, on the right, the two tangents to the parabola from the point \mathbf{z}_2 touch the parabola at a pair of conjugate, complex points, which don't appear in our all-real diagram.) The correspondence between \mathbf{z}_2 and ℓ_1 is a duality between points and lines called

the *polarity* of the conic. Of the dual pair \mathbf{z}_2 and ℓ_1 , the point \mathbf{z}_2 is called the *pole* and the line ℓ_1 is called the *polar* [18].

Going up by 1 in dimension, an implicit model of a quadric surface in 3-space defines a duality between pole points and polar planes in a similar way [2].

Going up in degree instead of in dimension, suppose that $F: \mathbb{R}^2 \to \mathbb{R}$ is an implicit model of an algebraic plane curve C of degree n, for n > 2. The polar form f of F is a symmetric, n-affine map $F: (\mathbb{R}^2)^n \to \mathbb{R}$. If we fix k of the polar arguments of f at \mathbf{w}_1 through \mathbf{w}_k , the remaining function

$$(\mathbf{z}_{k+1},\ldots,\mathbf{z}_n)\mapsto f(\mathbf{w}_1,\ldots,\mathbf{w}_k,\mathbf{z}_{k+1},\ldots,\mathbf{z}_n)$$

is symmetric and (n-k)-affine, and it is hence the polar form of an implicit model of an algebraic plane curve D of degree n-k. When $\mathbf{w}_1 = \cdots = \mathbf{w}_k = \mathbf{w}$, the curve D is called the k^{th} polar curve of the pole point \mathbf{w} with respect to C; an $(n-2)^{\text{nd}}$ polar is a conic and an $(n-1)^{\text{st}}$ polar is a line [16]. (I mention this situation partially because I wanted these notes to include at least one reference to a work from the heyday of polar forms: the late nineteenth century. The date on reference [16] is 1879.)

One intriguing open question in the field of blossoming is to study the relationships between the polar forms of parametric and implicit models of the same shape. For an example of such a relationship, the identity h(f(u,u),f(u,v))=0 relates the polar forms f and h of the parametric and implicit models of the parabola, given in equations (4) and (7). Note that we must constrain the universe of possible shapes somewhat if we want each shape to have both a simple parametric model and a simple implicit model. For example, for curves: A curve must be rational, that is, of genus zero, in order to have a rational parametric model. It must be planar in order to be implicitly modeled by a single polynomial—that is, a single gauge value—rather than by some non-principal ideal.

5 Generalizing the de Casteljau Algorithm

We studied the de Casteljau Algorithm briefly back in Section 2. Given the n+1 Bézier points of a segment F([r..s]) of an n-ic, parametric curve F and given a desired parameter value t, the de Casteljau Algorithm computes the diagonal value F(t) by performing a total of $\binom{n}{2}$ affine interpolations, in n successive stages. All of the points that arise during this process are polar values of the curve F. For example, in the cubic case, as shown back in Figure 7, we compute the successive rows of the triangular array:

$$f(r,r,r) \quad f(r,r,s) \quad f(r,s,s) \quad f(s,s,s)$$

$$f(r,r,t) \quad f(r,s,t) \quad f(s,s,t)$$

$$f(r,t,t) \quad f(s,t,t)$$

$$f(t,t,t)$$
(8)

The only difference between this array and the similar array (3) in Section 2 is that, in this version, the polar arguments appear sorted in alphabetical order, rather than in the numerical order r < t < s.

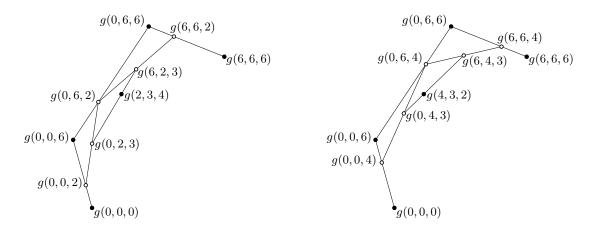


Figure 12: Computing the polar value g(2,3,4) in two different ways

The de Casteljau Algorithm generalizes in several important ways.

First, the new parameter value t can lie outside the interval [r..s], just as well as inside. When t lies outside [r..s], the affine interpolations in the de Casteljau Algorithm are actually affine extrapolations—that is, we must extend the segment joining the two input points, on one side or the other, in order to locate the output point. For an example, note that we can reinterpret Figure 7 to be the result of running the de Casteljau Algorithm to compute F(s), starting from the four Bézier points of the left-hand subsegment F([r..t]). In this new interpretation, precisely the same points and lines get drawn, but in a different order.

Second, we can use the de Casteljau Algorithm to compute polar values, just as easily as diagonal values. Let's consider the cubic case, for simplicity of notation. Suppose that we are given the four Bézier points of the cubic segment F([r..s]) and that we want to compute the polar value $f(t_1, t_2, t_3)$. We use each of the three polar arguments to control one stage of affine interpolations, as follows:

$$f(r,r,r) = f(r,r,s) = f(r,s,s) = f(s,s,s)$$

$$f(r,r,t_1) = f(r,s,t_1) = f(s,s,t_1)$$

$$f(r,t_1,t_2) = f(s,t_1,t_2)$$

$$f(t_1,t_2,t_3)$$
(9)

One new phenomenon here is that the various affine interpolations have different controlling ratios. In particular, the interpolations in the i^{th} stage are controlled by the position of t_i with respect to the interval [r..s].

Figure 12 shows two examples of computing polar values by using the generalized de Casteljau Algorithm given in array (9). On the left, we compute the polar value g(2,3,4) from the Bézier points of the cubic segment G([0..6]). On the right, starting from the same four Bézier points, we compute the polar value g(4,3,2). Of course, the two computations must arrive at the same result, because the polar form g of G is a symmetric function. But the intermediate points and lines differ.

Our third generalization of the de Casteljau Algorithm widens the class of possible input data, instead of widening the class of possible output values. In moving from array (8) to array (9) above, we replaced the single parameter value t by three independent parameter values t_1 , t_2 , and t_3 . The symmetry of the triangular array suggests that we similarly replace the single parameter values t_1 and t_2 by triples of independent parameter values t_3 and t_4 and t_5 by triples of independent parameter values t_4 and t_5 by triples of independent parameter values t_5 and t_5 and t_5 by triples of independent parameter values t_5 and t_5 and t_5 and t_5 are t_5 and t_5 and t_5 are t_5 and t_5 are t_5 and t_5 are t_5 are t_5 are t_5 and t_5 are t_5 are t_5 are t_5 are t_5 are t_5 and t_5 are t_5 are t_5 are t_5 are t_5 are t_5 are t_5 and t_5 are t_5 and t_5 are t_5 and t_5 are t_5 and t_5 are t_5

$$f(r_{1}, r_{2}, r_{3}) \quad f(r_{1}, r_{2}, s_{1}) \quad f(r_{1}, s_{1}, s_{2}) \quad f(s_{1}, s_{2}, s_{3})$$

$$f(r_{1}, r_{2}, t_{1}) \quad f(r_{1}, s_{1}, t_{1}) \quad f(s_{1}, s_{2}, t_{1})$$

$$f(r_{1}, t_{1}, t_{2}) \quad f(s_{1}, t_{1}, t_{2})$$

$$f(t_{1}, t_{2}, t_{3})$$

$$(10)$$

Note the structure of the polar arguments in this array. All rows below the first include a t_1 among their polar arguments; all rows below the second also include a t_2 ; and the bottom vertex also includes a t_3 . In a similar way, all positively sloped diagonals except for the longest include an t_1 ; all except for the two longest also include an t_2 ; and the upper-left vertex also includes an t_3 . And similarly for the negatively sloped diagonals and the t_3 .

The first step of the computation represented by array (10) computes $f(r_1, r_2, t_1)$ by interpolating between $f(r_1, r_2, r_3)$ and $f(r_1, r_2, s_1)$. The common polar arguments in this interpolation are r_1 and r_2 . As for the varying argument, the two input points correspond to the values r_3 and s_1 , while the output point corresponds to t_1 . Thus, the ratio of the interpolation will be determined by the position of t_1 with respect to the interval $[r_3..s_1]$. Note that we are in trouble if r_3 and s_1 are equal. In that case, we know only one value of the affine function $u \mapsto f(r_1, r_2, u)$, and that one value isn't enough for us to compute other values by interpolation. Worse yet, we have two different sources of information about that one value—the first two input points $f(r_1, r_2, r_3)$ and $f(r_1, r_2, s_1)$ —and they may not agree. We therefore rule out the case $r_3 = s_1$.

Each of the remaining five interpolations in array (10) also requires that we rule out some equality between an r_i and an s_j , for similar reasons. The following table shows the six necessary disequalities:

	s_1	s_2	<i>S</i> 3
r_1	\neq	\neq	\neq
r_2	\neq	\neq	
<i>r</i> ₃	\neq		

For general n, we must have $r_i \neq s_j$ whenever $i + j \leq n + 1$. In typical applications, we actually have $r_i < s_j$ for all i and j, which is more than enough.

The following theorem describes our fully generalized version of the de Casteljau Algorithm, the one in array (10).

Theorem 3. Let F be any n-ic parametric curve, let f be its polar form, and let r_1 through r_n and s_1 through s_n be any 2n numbers that satisfy the disequality constraints $r_i \neq s_j$ for $i+j \leq n+1$. If we are told the n+1 polar values $f(r_1, \ldots, r_{n-i}, s_1, \ldots, s_i)$ of F, for i in $[0 \ldots n]$, we can use the de Casteljau Algorithm to compute an arbitrary polar value $f(t_1, \ldots, t_n)$ of F with $\binom{n}{2}$ affine combinations.

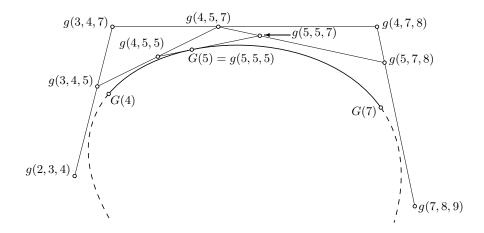


Figure 13: The de Boor case of the de Casteljau Algorithm

This generalized form of the de Casteljau Algorithm arises in the context of spline curves, where it is called the *de Boor Algorithm*. Although splines are properly the topic of later lectures, Figure 13 shows one example of the de Boor Algorithm in action.

Suppose that F is a cubic spline curve whose knot sequence runs, in part,

$$(\dots, 2, 3, 4, 7, 8, 9, \dots).$$
 (11)

Over each interval between pairs of adjacent knots, the spline F follows a segment of some cubic polynomial curve. Figure 13 shows the segment G([4..7]) that the spline F follows over the parameter interval [4..7] between the central pair of knots in (11). The dashed curves in Figure 13 show the continuations of the cubic polynomial curve G on either side of the interval [4..7]. For t < 4 and for t > 7, the spline F follows segments of other cubic curves, which are not shown.

Four adjacent control points of the spline curve F influence the segment F([4..7]) = G([4..7]), and it turns out that those four control points are all polar values of the influenced curve G, each with a consecutive triple of knots from the sequence (11) as its list of polar arguments. In particular, the four polar values g(2,3,4), g(3,4,7), g(4,7,8), and g(7,8,9) turn out to be four adjacent control points of the spline F. Suppose that we know those control points and that we want to compute F(5) = G(5). As shown in Figure 13, we are in a fine position to apply the generalized de Casteljau Algorithm of Theorem 3 with:

$$r_1 = 4$$
 $s_1 = 7$ $t_1 = 5$
 $r_2 = 3$ $s_2 = 8$ $t_2 = 5$
 $r_3 = 2$ $s_3 = 9$ $t_3 = 5$

In particular, the four input points required by the de Casteljau Algorithm with those parameters

are precisely the four control points that we know:

$$g(r_1, r_2, r_3) = g(4,3,2) = g(2,3,4)$$

 $g(r_1, r_2, s_1) = g(4,3,7) = g(3,4,7)$
 $g(r_1, s_1, s_2) = g(4,7,8) = g(4,7,8)$
 $g(s_1, s_2, s_3) = g(7,8,9) = g(7,8,9)$

Warning: Note that the parameters r_1 , r_2 , and r_3 must be indexed in reverse numerical order in order to make this work.

6 Polar interpolation

One unpleasant characteristic of Theorem 3 is that we have to start off knowing an n-ic curve F before we can apply the theorem. Suppose that P_0 through P_n are n+1 arbitrary points in our object space. What we would prefer to do is to use the constraints $f(r_1, \ldots, r_{n-i}, s_1, \ldots, s_i) = P_i$ for i in $[0 \ldots n]$ as a way of choosing which curve F we are interested in. In order to make that procedure valid, we have to prove that those constraints always determine a unique curve F of degree at most n. That is, we have to prove the following theorem—our last challenge for this first lecture.

Theorem 4. Let Q be an affine object space, let P_0 through P_n be any n+1 points in Q, and let r_1 through r_n and s_1 through s_n be any 2n numbers with $r_i \neq s_j$ for $i+j \leq n+1$. There exists a unique parametric curve F of degree at most n that satisfies the constraints $f(r_1, \ldots, r_{n-i}, s_1, \ldots, s_i) = P_i$ for i in $[0 \ldots n]$.

To understand what this theorem means, consider the special case where all of the r_i are equal, say equal to r, and all of the s_j are equal, say equal to s, with r < s. In that special case, the polar values that appear in the constraints are simply the Bézier points of the segment F([r..s]). Thus, the theorem says the following: Given any n+1 points and given r and s with r < s, there exists a unique n-ic curve segment F([r..s]) that has those n+1 points as its Bézier points. That is surely true, and, indeed, pre-blossoming approaches give various proofs. Our goal in this section is to give a proof based on blossoming.

For a different perspective on Theorem 4, recall the Lagrange Interpolation Formula. Given n+1 distinct numbers t_0 through t_n and n+1 arbitrary points P_0 through P_n , we can construct a parametric curve F of degree at most n that satisfies the interpolation constraints $F(t_i) = P_i$ for i in [0..n] by using the Lagrange Interpolation Formula on each coordinate polynomial separately. That is, we can specify an n-ic curve F by specifying n+1 of its diagonal values. Theorem 4 says something very similar, except that the specified values are polar values instead of diagonal values. Thus, Theorem 4 is actually a result about *polar interpolation*, which is the process of specifying a polynomial by giving certain polar values that it must—to stretch a term—pass through.

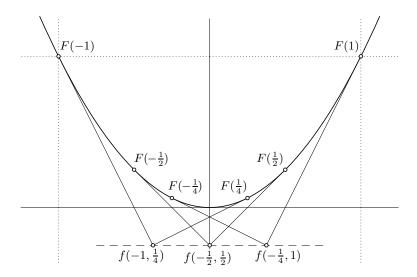


Figure 14: Three dependent polar values of the parabola $F(t) = \langle t, t^2 \rangle$

In the case of Lagrange-style, diagonal interpolation, there is a condition on the n+1 diagonal arguments t_0 through t_n : They must be distinct. In a similar way, in the polar case, we have to put some conditions on the n+1 lists of polar arguments in order to make it safe to specify the corresponding polar values independently. For example, when n=2, it wouldn't make sense to specify a quadratic curve by independently specifying the three polar values f(0,0), f(0,1), and f(0,2). Any two of those values determine the third, so the three are not independent. Theorem 4 gives a condition on n+1 lists of polar arguments that is sufficient to guarantee that the corresponding polar values are independent.

By the way, not all dependencies between polar values are as obvious as the dependency between f(0,0), f(0,1), and f(0,2). One example of a less obvious dependency is the following, again in the quadratic case: The polar value $f(-\frac{1}{2},\frac{1}{2})$ is always the midpoint of the line joining $f(-1,\frac{1}{4})$ to $f(-\frac{1}{4},1)$. Figure 14 illustrates this for the standard parabola $F(t)=\langle t,t^2\rangle$.

Theorem 4 claims that a unique interpolating n-ic curve F will exist. The uniqueness part of this claim is quite easy to prove. If F is any n-ic curve that satisfies $f(r_1, \ldots, r_{n-i}, s_1, \ldots, s_i) = P_i$ for i in $[0 \ldots n]$, we can use the generalized de Casteljau Algorithm of Theorem 3 to compute any polar value of F that we like by doing affine interpolations, starting with the P_i . Hence, all of the polar values of F and, in particular, all of the diagonal values of F are uniquely determined. That is more than enough to determine F uniquely.

The hard part of proving Theorem 4 is proving that some interpolant F exists. Most people's first instinct is to prove the existence of F by explicitly constructing F, using the de Casteljau Algorithm. That technique can be made to work, but things get pretty messy. Certainly, for any list of polar arguments (t_1, \ldots, t_n) , we can carry out the de Casteljau Algorithm and figure out what the polar value $f(t_1, \ldots, t_n)$ would have to be, if an interpolating curve F did exist. Let $c(t_1, \ldots, t_n)$ denote the value that comes out of the de Casteljau Algorithm. It's pretty easy to see that c will be a multiaffine function. But it takes a fair amount of work to show

that the function c will be symmetric, and hence the polar form of some n-ic curve C. And it takes more work to show that the curve C so constructed does interpolate, that is, that we have $c(r_1, \ldots, r_{n-i}, s_1, \ldots, s_i) = P_i$ for all i in $[0 \ldots n]$.

Fortunately, there is a slicker and easier way to prove the existence of an interpolant F: Count the degrees of freedom involved. As has been our custom in these notes, let's carry out that slicker proof in the case n = 3, for notational convenience; the same ideas work for any n.

Each coordinate of a parametric curve is independent of the other coordinates. Thus, in proving Theorem 4, it suffices to consider the case where the object space Q is simply \mathbb{R} . In the cubic case with $Q = \mathbb{R}$, we are given 4 arbitrary real numbers p_0 through p_3 , and we want to prove that there exists some cubic polynomial F whose polar form f satisfies the four polar interpolation constraints

$$f(r_1, r_2, r_3) = p_0$$

$$f(r_1, r_2, s_1) = p_1$$

$$f(r_1, s_1, s_2) = p_2$$

$$f(s_1, s_2, s_3) = p_3.$$
(12)

Using undetermined coefficients, the generic cubic polynomial can be written $F(t) = at^3 + 3bt^2 + 3ct + d$; we included extra factors of 3 in the middle two coefficients in order to avoid denominators in the polar form f of F, which is therefore given by

$$f(u, v, w) = auvw + b(uv + uw + vw) + c(u + v + w) + d.$$

Substituting this formula for f into any one of the four interpolation constraints in (12) gives an affine equation (what most people would call a non-homogeneous, linear equation) on the unknown coefficients a, b, c, and d. Overall, the polar interpolation constraints (12) turn into a system of equations of the form

$$\begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}.$$
(13)

From the de Casteljau Algorithm, we know that solutions to this system of equations are unique. In particular, if we set $p_0 = p_1 = p_2 = p_3 = 0$, the only corresponding solution is the trivial solution a = b = c = d = 0. From this, it follows that the columns of the coefficient matrix $M = (m_{ij})$ in equation (13) are linearly independent. But M is a square matrix: If its columns are linearly independent, then it is invertible. Thus, for any vector (p_0, p_1, p_2, p_3) of given data, a vector (a, b, c, d) of coefficients will exist that makes the polar form f of F satisfy the interpolation constraints (12). And that completes the slick proof of Theorem 4.

In the proof above, we didn't bother to express the matrix elements m_{ij} explicitly in terms of the parameters r_i and s_j . But doing so is not difficult. Here is the explicit form of the matrix M:

$$M = \begin{pmatrix} r_1 r_2 r_3 & r_1 r_2 + r_1 r_3 + r_2 r_3 & r_1 + r_2 + r_3 & 1 \\ r_1 r_2 s_1 & r_1 r_2 + r_1 s_1 + r_2 s_1 & r_1 + r_2 + s_1 & 1 \\ r_1 s_1 s_2 & r_1 s_1 + r_1 s_2 + s_1 s_2 & r_1 + s_1 + s_2 & 1 \\ s_1 s_2 s_3 & s_1 s_2 + s_1 s_3 + s_2 s_3 & s_1 + s_2 + s_3 & 1 \end{pmatrix}$$

From this, we can compute the determinant of M, which turns out to be

$$\det(M) = (r_1 - s_1)(r_1 - s_2)(r_1 - s_3)(r_2 - s_1)(r_2 - s_2)(r_3 - s_1).$$

Note that our assumption in Theorem 4 that $r_i \neq s_j$ for $i + j \leq n + 1$ is just enough to guarantee that det(M) is nonzero. That is, we didn't assume any more than we had to.

Computing the explicit form of the matrix M for general n and evaluating its determinant makes an interesting exercise in linear algebra. Furthermore, the solution to that exercise provides a proof of Theorem 4 that does not involve the de Casteljau Algorithm.

7 One last formula

I can't resist tacking on one last pretty formula. If F is a cubic curve and f is its polar form, de Casteljau discovered that

$$f(u,v,w) = \frac{(w-v)^2 F(u)}{3(w-u)(u-v)} + \frac{(w-u)^2 F(v)}{3(w-v)(v-u)} + \frac{(v-u)^2 F(w)}{3(v-w)(w-u)}.$$
 (14)

This equation expresses the generic polar value f(u, v, w) as an affine combination of the three diagonal values F(u), F(v), and F(w), with coefficients that are rational functions of u, v, and w. Among other things, this formula implies that the four points F(0), F(t), F(1), and F(0), and F(0), in Figure 1 must be coplanar, even when F(0) is a twisted cubic in space—something which is far from obvious.

Similar pretty formulas for computing polar values exist for each odd value of n; when n = 1, for example, we have the pretty but trivial formula f(t) = F(t). The corresponding formulas for even n aren't quite so pretty [2].

References

- [1] Richard H. Bartels, John C. Beatty, and Brian A. Barsky (1987), *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*, Morgan Kaufmann, 95 First Street, Los Altos, California 94022.
- [2] Maxime Bôcher (1907), Introduction to Higher Algebra, Macmillan, pages 114–128.
- [3] Wolfgang Boehm, Gerald Farin, and Jürgen Kahmann (1984), A survey of curve and surface methods in CAGD, Computer Aided Geometric Design 1, 1–60.
- [4] Carl de Boor (1978), A Practical Guide to Splines, Springer-Verlag.
- [5] Carl de Boor (1986), B(asic)-spline basics, first portion of course notes for course #5 at ACM SIGGRAPH '86.

[6] Carl de Boor and Klaus Höllig (1987), B-splines without divided differences, in: Gerald E. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, SIAM, pages 21–27.

- [7] Paul de Faget de Casteljau (1985), *Formes à Pôles*, volume 2 of the series Mathématiques et CAO, Hermes, 51 rue Rennequin, 75017 Paris. Also published in English translation (1986) as *Shape Mathematics and CAD*, Kogan Page Ltd., 120 Pentonville Road, London N1 9JN.
 - [1] Page 59 in the French, page 50 in the English.
- [8] Paul de Faget de Casteljau (1990), Le Lissage, Hermes.
- [9] Gerald Farin (1988), Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide, Academic Press.
- [10] R. N. Goldman, T. W. Sederberg, and D. C. Anderson (1984), Vector elimination: A technique for the implicitization, inversion, and intersection of planar parametric rational polynomial curves. Computer Aided Geometric Design 1, 327–356.
- [11] E. T. Y. Lee (1985), Some remarks concerning B-splines, Computer Aided Geometric Design **2**, 307–311.
- [12] E. T. Y. Lee (1989), A note on blossoming, Computer Aided Geometric Design **6**, 359–362.
- [13] Lyle Ramshaw (1987), Blossoming: a connect-the-dots approach to splines, Digital Equipment Corp. Systems Research Center technical report #19.
- [14] Lyle Ramshaw (1988), Béziers and B-splines as multiaffine maps, in: R. A. Earnshaw, editor, *Theoretical Foundations of Computer Graphics and CAD*, NATO ASI Series F, volume 40, Springer-Verlag, pages 757–776.
- [15] Lyle Ramshaw (1989), Blossoms are polar forms, Computer Aided Geometric Design **6**, 323–358.
 - [2] Section 8.6, page 338.
- [16] George Salmon (1879), A Treatise on the Higher Plane Curves, third edition, Chelsea, pages 49–52.
- [17] Hans-Peter Seidel (1989), A new multiaffine approach to B-splines, Computer Aided Geometric Design **6**, 23–32.
- [18] A. Seidenberg (1962), *Lectures in Projective Geometry*, Van Nostrand, pages 115-121 and 184–193.
- [19] B. L. van der Waerden (1970), *Algebra*, volume 2, translated from the German fifth edition, Frederick Ungar, New York, page 20.