Chapter 7

Planar Curve Intersection

Curve intersection involves finding the points at which two planar curves intersect. If the two curves are parametric, the solution also identifies the parameter values of the intersection points.

7.1 Bezout's Theorem

Bezout's theorem states that a planar algebraic curve of degree n and a second planar algebraic curve of degree m intersect in exactly mn points, if we properly count complex intersections, intersections at infinity, and possible multiple intersections. If they intersect in more than that number of points, they intersect in infinitely many points. This happens if the two curves are identical, for example.

The correct accounting of the multiplicity of intersections is essential to properly observe Bezout's theorem. If two curves intersect and they are not tangent at the point of intersection (and they do not self-intersect at that point), the intersection is said to have a multiplicity of one. Such an intersection is also called a transversal intersection, or a simple intersection. When two curves are tangent, they intersect twice at the point of tangency. They are also said to intersect with a multiplicity of two. If, in addition, the two curves have the same curvature, they are said to intersect with a multiplicity three. If two curves intersect with a multiplicity of n and there are no self-intersections at that point, the two curves are said to be G^{n-1} continuous at that point.

Bezout's theorem also tells us that two surfaces of degree m and n respectively intersect in an algebraic space curve of degree mn. Also, a space curve of degree m intersects a surface of degree n in mn points. In fact, this provides us with a useful definition for the degree of a surface or space curve. The degree of a surface is the number of times it is intersected by a general line, and the degree of a space curve is the number of times it intersects a general plane. The phrase "number of intersections" must be modified by the phrase "properly counted", which means that complex, infinite, and multiple intersections require special consideration.

7.1.1 Homogeneous coordinates

Proper counting of intersections at infinity is facilitated through the use of homogeneous coordinates. A planar algebraic curve of degree n

$$f(x,y) = \sum_{i+j \leq n} a_{ij} x^i y^j \;=\; 0$$

can be expressed in homogeneous form by introducing a homogenizing variable w as follows:

$$f(X,Y,W) = \sum_{i+j+k=n} a_{ij} X^i Y^j W^k \ = \ 0.$$

Note that, in the homogeneous equation, the degree of every term is n.

Any homogeneous triple (X,Y,W) corresponds to a point whose Cartesian coordinates (x,y) are $(\frac{X}{W}, \frac{Y}{W})$.

7.1.2 Circular Points at Infinity

A circle whose equation is $(x - x_c)^2 \ + \ (y \ - \ y_c)^2 \ - \ r^2 \ = \ 0$ has the homogeneous representation

$$(X - x_c W)^2 \ + \ (Y \ - \ y_c W)^2 \ - \ r^2 W^2 \ = \ 0. \tag{7.1}$$

A point whose homogeneous coordinate $W = 0$ corresponds to a point whose Cartesian coordinates are at infinity.

An example of the practical use of homogeneous coordinates is that it allows us to observe the interesting fact that every circle passes through the two points $(1, i, 0)$ and $(1, -i, 0)$. You can easily verify this by substituting these points into (7.1). The points $(1, i, 0)$ and $(1, -i, 0)$, known as the circular points at infinity, are both infinite and complex.

Circles are degree two curves, and thus Bezout s theorem requires two distinct circles to intersect in exactly four points. However, our practical experience with circles suggests that they intersect in at most two points. Of course, two distinct circles *can* only intersect in at most two **real** points. However, since all circles pass through the two circular points at infinity, we thus account for the two missing intersection points.

7.1.3 Homogeneous parameters

A unit circle at the origin can be expressed parametrically as

$$x \ = \ \frac{-t^2 \ + \ 1}{t^2 \ + \ 1} \qquad y \ = \ \frac{2t}{t^2 \ + \ 1}. \tag{7.2}$$

A circle can also be expressed in terms of homogeneous parameters (T,U) where $t \ = \ \frac{T}{U}$:

$$x \ = \ \frac{-T^2 \ + \ U^2}{T^2 \ + \ U^2} \qquad y \ = \ \frac{2TU}{T^2 \ + \ U^2}. \tag{7.3}$$

To plot the entire circle using equation (7.2), t would have to range from negative to positive infinity. An advantage to the homogeneous parameters is that the entire circle can be swept out with finite parameter values. This is done as follows:

- First Quadrant: $0 \leq T \leq 1$, $U = 1$
- Second Quadrant: $T = 1$, $1 \geq U \geq 0$
- Third Quadrant: $T = -1$, $0 \leq U \leq 1$
- Fourth Quadrant: $-1 \leq T \leq 0$, $U = 1$

7.1.4 The Fundamental Theorem of Algebra

The **fundamental theorem of algebra** can be thought of as a special case of Bezout's theorem. It states that a univariate polynomial of degree n has exactly n roots. That is, for $f(t) = a_0 + a_1 t + \cdots + a_n t^n$ there exist exactly n values of t for which $f(t) = 0$, if we count complex roots and possible multiple roots. If the a_i are all real, then any complex roots must occur in conjugate pairs. In other words, if the complex number $b + ci$ is a root of the real polynomial $f(t)$, then so also is $b - ci$.

7.2 The Intersection of Two Lines

A line is a degree one curve, so according to Bezout's theorem, two lines generally intersect in one point; if they intersect in more than one point, the two lines are identical. Several solutions to the problem of finding the point at which two lines intersect are presented in junior high school algebra courses. We present here an elegant solution involving homogeneous representation that is not found in junior high school algebra courses, and that properly accounts for intersections at infinity.

7.2.1 Homogeneous Points and Lines

We will denote by

$$\mathbf{P}(X, Y, W)$$

the point whose homogeneous coordinates are (X, Y, W) and whose Cartesian coordinates are $(x, y) = (X/W, Y/W)$. Likewise, we denote by

$$\mathbf{L}(a, b, c)$$

the line whose equation is $ax + by + c = aX + bY + cW = 0$.

The projection operator converts a point from homogeneous to Cartesian coordinates: $\Pi(\mathbf{P}(X, Y, W)) = (\frac{X}{W}, \frac{Y}{W})$.

The point $\mathbf{P}(X, Y, 0)$ lies at infinity, in the direction (X, Y). The line $\mathbf{L}(0, 0, 1)$ is called the line at infinite, and it includes all points at infinity.

Since points and lines can both be represented by triples of numbers, we now ask what role is played by the conventional operations of cross product

$$(a, b, c) \times (d, e, f) = (bf - ec, dc - af, ae - db)$$

and dot product

$$(a, b, c) \cdot (d, e, f) = ad + be + cf$$

are defined. The dot product determines incidence: point $\mathbf{P}(X, Y, W)$ lies on a line $\mathbf{L}(a, b, c)$ if and only if

$$\mathbf{P} \cdot \mathbf{L} = aX + bY + cW = 0.$$

The cross product has two applications: The line $\mathbf{L}$ containing two points $\mathbf{P}_1$ and $\mathbf{P}_2$ is given by $\mathbf{L} = \mathbf{P}_1 \times \mathbf{P}_2$ and the point $\mathbf{P}$ at which two lines $\mathbf{L}_1$ and $\mathbf{L}_2$ intersect is given by $\mathbf{P} = \mathbf{L}_1 \times \mathbf{L}_2$. This is an example of the principle of *duality* which, loosely speaking, means that general statements involving points and lines can be expressed in a reciprocal way. For example, "A unique line passes through two distinct points" has a dual expression, "A unique point lies at the intersection of two distinct lines".

Example The points $\mathbf{P}(2, 3, 1)$ and $\mathbf{P}(3, 1, 1)$ define a line $(2, 3, 1) \times (3, 1, 1) = \mathbf{L}(2, 1, -7)$. The points $\mathbf{P}(2, 3, 1)$ and $\mathbf{P}(1, 4, 1)$ define a line $(2, 3, 1) \times (1, 4, 1) = \mathbf{L}(-1, -1, 5)$. The lines $\mathbf{L}(2, 1, -7)$

and $\mathbf{L}(-1,-1,5)$ intersect in a point $(2,1,-7)\times(-1,-1,5) = \mathbf{P}(-2,-3,-1)$. Note that $\Pi(2,3,1) = \Pi(-2,-3,-1) = (2,3)$.

We noted that Bezout's theorem requires that we properly account for complex, infinite, and multiple intersections, and that homogeneous equations allow us to deal with points at infinity. We now illustrate. Students in junior high school algebra courses are taught that there is no solution for two equations representing the intersection of parallel lines, such as

$$3x + 4y + 2 = 0, \qquad 3x + 4y + 3 = 0.$$

However, using homogeneous representation, we see that the intersection is given by

$$\mathbf{L}(3,4,2) \times \mathbf{L}(3,4,3) = \mathbf{P}(-4,3,0)$$

which is the point at infinity in the direction $(-4,3)$.

7.3 Intersection of a Parametric Curve and an Implicit Curve

The points at which a parametric curve $(x(t), y(t))$ intersects an implicit curve $f(x,y) = 0$ can be found as follows. Let $g(t) = f(x(t), y(t))$. Then the roots of $g(t) = 0$ are the parameter values at which the parametric curve intersects the implicit curve. For example, let

$$x(t) = 2t - 1, \qquad y(t) = 8t^2 - 9t + 1$$

and

$$f(x,y) = x^2 + y^2 - 1$$

Then

$$g(t) = f(x(t), y(t)) = 64t^4 - 144t^3 + 101t^2 - 22t + 1$$

The roots of $g(t) = 0$ are $t = 0.06118$, $t = 0.28147$, $t = 0.90735$, and $t = 1.0$. The Cartesian coordinates of the intersection points can then be found by evaluating the parametric curve at those values of t: $(-0.8776, 0.47932)$, $(-0.43706, -0.89943)$, $(0.814696, -0.5799)$, $(1,0)$.

If the parametric curve is rational,

$$x = \frac{a(t)}{c(t)}, \quad y = \frac{b(t)}{c(t)}$$

it is more simple to work in homogeneous form:

$$X = a(t), \quad Y = b(t), \quad W = c(t).$$

For example, to intersect the curves

$$x = \frac{t^2 + t}{t^2 + 1}, \quad y = \frac{2t}{t^2 + 1}$$

and

$$f(x,y) = x^2 + 2x + y + 1 = 0$$

we homogenize the implicit curve

$$f(X,Y,W) = X^2 + 2XW + YW + W^2 = 0$$

and the parametric curve

$$X(t) = t^2 + t, \quad Y(t) = 2t, \quad W(t) = t^2 + 1$$

and make the substitution

$$g(t) = f(X(t), Y(t), W(t)) = 4t^4 + 6t^3 + 5t^2 + 4t + 1$$

In this case, $g(t)$ has two real roots at $t = 1$ and $t = 0.3576$ and two complex roots.

This method is very efficient because it reduces the intersection problem to a polynomial root finding problem. If both curves are parametric, it is possible to convert one of them to implicit form using *implicitization*. The implicitization curve intersection algorithm is briefly sketched in section 16.8. A curve intersection algorithm based on implicitization is very fast, although it has some limitations. For example, if the degree of the two curves is large, the method is slow and experiences significant floating point error propagation. For numerical stability, if the computations are to be performed in floating points, one should implicitize in the Bernstein basis as discussed in section 16.6. But even this does not adequately reduce floating point error if two two curves are, say, degree ten.

7.3.1 Order of Contact

Given a degree m parametric curve

$$\mathbf{P}(t) = (X(t), Y(t), W(t))$$

and a degree n implicit curve

$$f(X, Y, W) = 0$$

the polynomial $g(t) = f(X(t), Y(t), W(t))$ is degree mn and, according to the fundamental theorem of algebra, it will have mn roots. This is a substantiation of Bezout's theorem.

A point on the curve $f(X, Y, W) = 0$ is a *singular* point if $f_X = f_Y = f_W = 0$. A point that is not singular is called a simple point.

If $g(\tau) = 0$, then $\mathbf{P}(\tau)$ is a point of intersection and $g(t)$ can be factored into $g(t) = (1 - \tau)^k \tilde{g}(t)$ where k is the *order of contact* between the two curves. If $\mathbf{P}(\tau)$ is a simple point on $f(X, Y, W) = 0$, then the two curves are G^{k-1} continuous at $\mathbf{P}(\tau)$.

Example The intersection between the line $X(t) = -t + 1$, $Y(t) = t$, $W(t) = 1$ and the curve $X^2 + Y^2 - W^2 = 0$ yields $g(t) = 2t^2 - 2t$, so there are two intersections, at $t = 0$ and $t = 1$. The Cartesian coordinates of the intersection points are $(1, 0)$ and $(0, 1)$.

Example The intersection between the line $X(t) = 1 + t^2$, $Y(t) = t$, $W(t) = 1$ and the curve $X^2 + Y^2 - W^2 = 0$ yields $g(t) = t^2$, so there is an intersection at $t = 0$ with order of contact of 2. These two curves are tangent at $(1, 0)$.

Example The intersection between the line $X(t) = 1 - t^2$, $Y(t) = t + t^2$, $W(t) = 1$ and the curve $X^2 + Y^2 - W^2 = 0$ yields $g(t) = 4t^3 + 3t^4$, so there is an intersection at $t = 0$ with order of contact of 3, plus an intersection at $t = -\frac{4}{3}$ with order of contact 1. These two curves are curvature continuous at $(1, 0)$.

The order-of-contact concept gives a powerful way to analyze curvature: The osculating circle is the unique circle that intersects $\mathbf{P}(t)$ with an order of contact of three.

Example $f(X, Y, W) = X^2 + Y^2 - 2\rho YW = 0$ is a circle of radius ρ with a center at $(0, \rho)$. What is the curvature of the curve $X(t) = x_1 t + x_2 t^2$, $Y(t) = y_2 t^2$, $W(t) = w_0 + w_1 t + w_2 t^2$? For these two curves,

$$g(t) = (x_1^2 - 2\rho w_0 y_2)t^2 + (2x_1 x_2 - 2\rho w_1 y_2)t^3 + (x_2^2 - 2\rho w_2 y_2 + y_2^2)t^4$$

For any value of ρ these two curves are tangent continuous. They are curvature continuous if $x_1^2 - 2\rho w_0 y_2 = 0$, which happens if

$$\rho = \frac{x_1^2}{2w_0 y_2}$$

Therefore, the curvature of $\mathbf{P}(t)$ at $t = 0$ is

$$\kappa = \frac{2w_0 y_2}{x_1^2}.$$

7.4 Computing the Intersection of Two Bézier Curves

Several algorithms address the problem of computing the points at which two Bézier curves intersect. Predominant approaches are the Bézier subdivision algorithm [LR80], the interval subdivision method adapted by Koparkar and Mudur [KM83], implicitization [SP86a], and Bézier clipping [SN90].

7.4.1 Timing Comparisons

A few sample timing comparisons for these four methods are presented in [SN90]. Comparative algorithm timings can of course change somewhat as the implementations are fine tuned, if tests are run on different computers, or even if different compilers are used. These timing tests were run on a Macintosh II using double precision arithmetic, computing the answers to eight decimal digits of accuracy.

The columns in Table 7.1 indicate the relative execution time for the algorithms *clip* = Bézier clipping algorithm, *Impl.* = implicitization, *Int* = Koparkar's interval algorithm and *Sub* = the conventional Bézier subdivision algorithm. In general, the implicitization intersection algorithm is

Example	Degree	Clip	Impl.	Int	Sub.
1	3	2.5	1	10	15
2	3	1.8	1	5	6
3	5	1	1.7	3	5
4	10	1	na	2	4

Table 7.1: Relative computation times

only reliable for curves of degree up to five using double precision arithmetic. For higher degrees, it is possible for the polynomial condition to degrade so that no significant digits are obtained in the answers. For curves of degree less than five, the implicitization algorithm is typically 1-3 times faster than the Bézier clip algorithm, which in turn is typically 2-10 times faster than the other two algorithms. For curves of degree higher than four, the Bézier clipping algorithm generally wins.

A brief discussion of these curve intersection methods follows.

7.5 Bézier subdivision

The Bézier subdivision curve intersection algorithm relies on the convex hull property and the de Casteljau algorithm. Though we overview it in terms of Bézier curves, it will work for any curve

which obeys the convex hull property. Figure 7.1 shows the convex hull of a single Bézier curve, and the convex hulls after subdividing into two and four pieces.

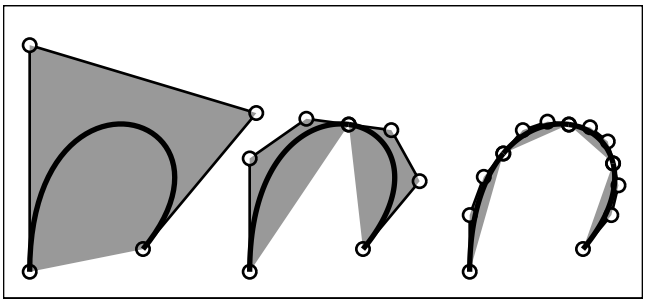


Figure 7.1: Convex Hulls

The intersection algorithm proceeds by comparing the convex hulls of the two curves. If they do not overlap, the curves do not intersect. If they do overlap, the curves are subdivided and the two halves of one curve are checked for overlap against the two halves of the other curve. As this procedure continues, each iteration rejects regions of curves which do not contain intersection points. Figure 7.2 shows three iterations.

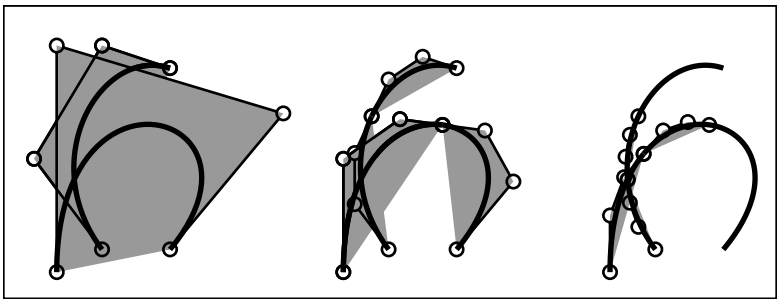


Figure 7.2: Three iterations of Bézier subdivision

Once a pair of curves has been subdivided enough that they can each be approximated by a line segment to within a tolerance ϵ (as given in equation 10.4), the intersection of the two approximating line segments is found.

Since convex hulls are rather expensive to compute and to determine overlap, in practice one normally just uses the $x - y$ bounding boxes.

7.6 Interval subdivision

The interval subdivision algorithm method is similar in spirit to the Bézier subdivision method. In this case, each curve is preprocessed to determine its vertical and horizontal tangents, and divided into "intervals" which have such tangents only at endpoints, if at all. Note that within any such interval, a rectangle whose diagonal is defined by any two points on the curve completely bounds the curve between those two endpoints. The power of this method lies in the fact that subdivision can now be performed by merely evaluating the curve (using Horner's method instead of the more

expensive de Casteljau algorithm) and that the bounding box is trivial to determine. Figure 7.3 illustrates this bounding box strategy.

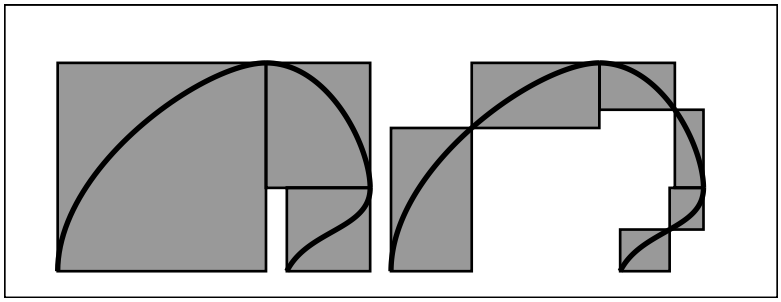


Figure 7.3: Interval preprocess and subdivision

7.7 Bézier Clipping method

The method of Bézier clipping has many applications, such as ray tracing trimmed rational surface patches [NSK90], algebraic curve intersection [Sed89], and tangent intersection computation [SN90]. It can be viewed as kind of an interval Newton's method, because it has quadratic convergence, but robustly finds all intersections. Since it is such a powerful tool, and since it is based on some ideas that haven't been discussed previously in these notes, the majority of this chapter is devoted to this method.

7.7.1 Fat Lines

Define a *fat line* as the region between two parallel lines. Our curve intersection algorithm begins by computing a fat line which bounds one of the two Bézier curves. Similar bounds have been suggested in references [Bal81, SWZ89].

Denote by $\bar{L}$ the line $\mathbf{P}_0 - \mathbf{P}_n$. We choose a fat line parallel to $\bar{L}$ as shown in Figure 7.4. If $\bar{L}$ is

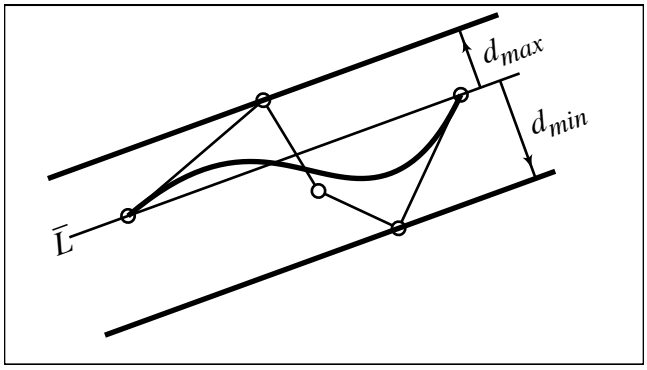


Figure 7.4: Fat line bounding a quartic curve

defined in its normalized implicit equation

$$ax + by + c = 0 \quad (a^2 + b^2 = 1) \tag{7.4}$$

then, the distance $d(x, y)$ from any point (x, y) to $\bar{L}$ is

$$d(x, y) = ax + by + c \tag{7.5}$$

Denote by $d_i = d(x_i, y_i)$ the signed distance from control point $\mathbf{P}_i = (x_i, y_i)$ to $\bar{L}$. By the convex hull property, a fat line bounding a given rational Bézier curve with non-negative weights can be defined as the fat line parallel to $\bar{L}$ which most tightly encloses the Bézier control points:

$$\{(x, y) | d_{min} \leq d(x, y) \leq d_{max}\} \tag{7.6}$$

where

$$d_{min} = \min\{d_0, \ldots, d_n\}, \quad d_{max} = \max\{d_0, \ldots, d_n\}. \tag{7.7}$$

7.7.2 Bézier Clipping

Figure 7.5 shows two polynomial cubic Bézier curves $\mathbf{P}(t)$ and $\mathbf{Q}(u)$, and a fat line $\mathbf{L}$ which bounds $\mathbf{Q}(u)$. In this section, we discuss how to identify intervals of t for which $\mathbf{P}(t)$ lies outside of $\mathbf{L}$, and hence for which $\mathbf{P}(t)$ does not intersect $\mathbf{Q}(u)$.

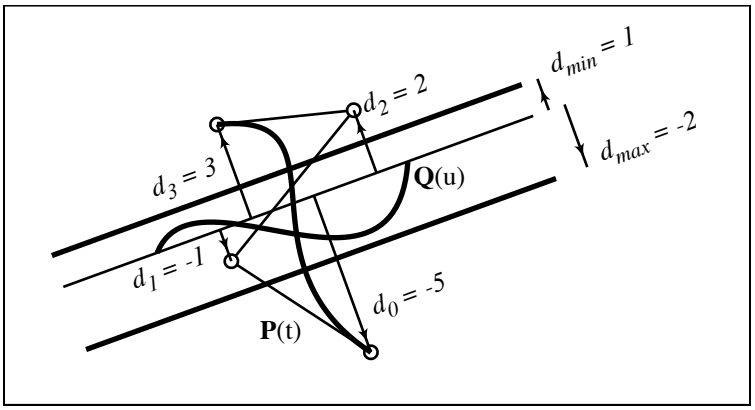


Figure 7.5: Bézier curve/fat line intersection

P is defined by its parametric equation

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{P}_i B_i^n(t) \tag{7.8}$$

where $\mathbf{P}_i = (x_i, y_i)$ are the Bézier control points, and $B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i$ denote the Bernstein basis functions. If the line $\bar{L}$ through $\mathbf{Q}_0 - \mathbf{Q}_n$ is defined by

$$ax + by + c = 0 \quad (a^2 + b^2 = 1), \tag{7.9}$$

then the distance $d(t)$ from any point $\mathbf{P}(t)$ to $\bar{L}$ can be found by substituting equation 7.8 into equation 7.9:

$$d(t) = \sum_{i=0}^{n} d_i B_i^n(t), \quad d_i = ax_i + by_i + c. \tag{7.10}$$

Note that $d(t) = 0$ for all values of t at which $\mathbf{P}$ intersects $\bar{L}$. Also, d_i is the distance from $\mathbf{P}_i$ to $\bar{L}$ (as shown in Figure 7.5).

The function $d(t)$ is a polynomial in Bernstein form, and can be represented as an explicit Bézier curve (Section 2.11) as follows:

$$\mathbf{D}(t) = (t, d(t)) = \sum_{i=0}^{n} \mathbf{D}_i B_i^n(t). \tag{7.11}$$

Figure 7.6 shows the curve $\mathbf{D}(t)$ which corresponds to Figure 7.5.

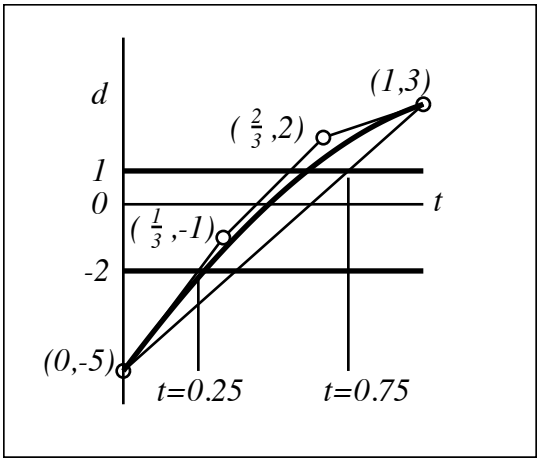


Figure 7.6: Explicit Bézier curve

Values of t for which $\mathbf{P}(t)$ lies outside of $\mathbf{L}$ correspond to values of t for which $\mathbf{D}(t)$ lies above $d = d_{max}$ or below $d = d_{min}$. We can identify parameter ranges of t for which $\mathbf{P}(t)$ is guaranteed to lie outside of $\mathbf{L}$ by identifying ranges of t for which the *convex hull* of $\mathbf{D}(t)$ lies above $d = d_{max}$ or below $d = d_{min}$. In this example, we are assured that $\mathbf{P}(t)$ lies outside of $\mathbf{L}$ for parameter values $t < 0.25$ and for $t > 0.75$.

Bézier clipping is completed by subdividing $\mathbf{P}$ twice using the de Casteljau algorithm, such that portions of $\mathbf{P}$ over parameter values $t < 0.25$ and $t > 0.75$ are removed.

Figure 7.6 shows how to clip against a fat line using a single explicit Bézier curve. This approach only works for polynomial Bézier curves. For rational Bézier curves, the explicit Bézier curves generated for each of the two lines are not simple translations of each other, so we must clip against each of the two lines separately. This is illustrated in Sections 7.7.7 and 7.7.8

7.7.3 Iterating

We have just discussed the notion of Bézier clipping in the context of curve intersection: regions of one curve which are guaranteed to not intersect a second curve can be identified and subdivided

away. Our Bézier clipping curve intersection algorithm proceeds by iteratively applying the Bézier clipping procedure.

Figure 7.7 shows curves $\mathbf{P}(t)$ and $\mathbf{Q}(u)$ from Figure 7.5 after the first Bézier clipping step in which regions $t < 0.25$ and $t > 0.75$ have been clipped away from $\mathbf{P}(t)$. The clipped portions of $\mathbf{P}(t)$ are shown in fine pen width, and a fat line is shown which bounds $\mathbf{P}(t)$, $0.25 \leq t \leq 0.75$. The next step in the curve intersection algorithm is to perform a Bézier clip of $\mathbf{Q}(u)$, clipping away regions of $\mathbf{Q}(u)$ which are guaranteed to lie outside the fat line bounding $\mathbf{P}(t)$. Proceeding as before, we

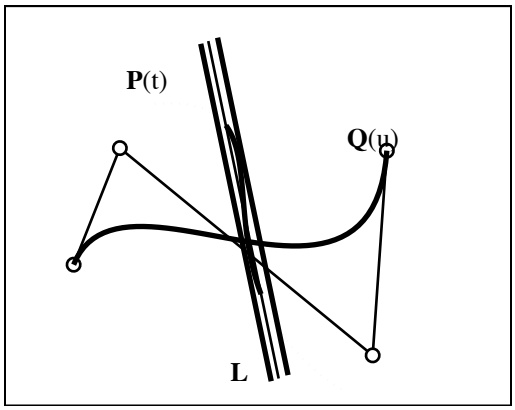


Figure 7.7: After first Bézier clip

define an explicit Bézier curve which expresses the distance from $\bar{L}$ in Figure 7.7 to the curve $\mathbf{Q}(u)$, from which we conclude that it is safe to clip off regions of $\mathbf{Q}(u)$ for which $u < .42$ and $u > .63$.

Next, $\mathbf{P}(t)$ is again clipped against $\mathbf{Q}(u)$, and so forth. After three Bézier clips on each curve, the intersection is computed to within six digits of accuracy (see Table 7.2).

Step	$\mathbf{t_{min}}$	$\mathbf{t_{max}}$	$\mathbf{u_{min}}$	$\mathbf{u_{max}}$
0	0	1	0	1
1	0.25	0.75	0.4188	0.6303
2	0.3747	0.4105	0.5121	0.5143
3	0.382079	0.382079	0.512967	0.512967

Table 7.2: Parameter ranges for $\mathbf{P}(t)$ and $\mathbf{Q}(u)$.

7.7.4 Clipping to other fat lines

The fat line defined in section 7.7.1 provides a nearly optimal Bézier clip in most cases, especially after a few iterations. However, it is clear that *any* pair of parallel lines which bound the curve can serve as a fatline. In some cases, a fat line perpendicular to line $\mathbf{P}_0 - \mathbf{P}_n$ provides a larger Bézier clip than the fat line parallel to line $\mathbf{P}_0 - \mathbf{P}_n$. We suggest that in general it works best to examine both fat lines to determine which one provides the largest clip. Experience has shown this extra overhead to reap a slightly lower average execution time.

7.7.5 Multiple Intersections

Figure 7.8 shows a case where two intersection points exist. In such cases, iterated Bézier clipping

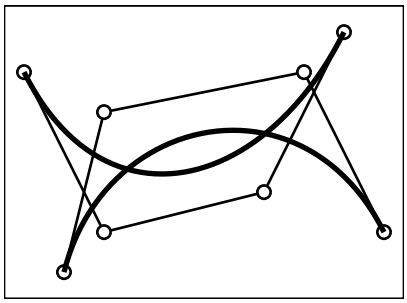


Figure 7.8: Two intersections

cannot converge to a single intersection point. The remedy is to split one of the curves in half and to compute the intersections of each half with the other curve, as suggested in Figure 7.9. A stack

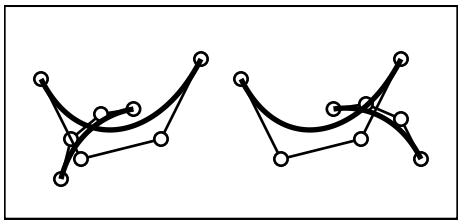


Figure 7.9: Two intersections, after a split

data structure is used to store pairs of curve segments, as in the conventional divide-and-conquer intersection algorithm [LR80].

Experimentation suggests the following heuristic. If a Bézier clip fails to reduce the parameter range of either curve by at least 20%, subdivide the "longest" curve (largest remaining parameter interval) and intersect the shorter curve respectively with the two halves of the longer curve. This heuristic, applied recursively if needed, allows computation of arbitrary numbers of intersections.

7.7.6 Rational Curves

If $\mathbf{P}$ is a *rational* Bézier curve

$$\mathbf{P}(t) = \frac{\sum_{i=0}^{n} w_i \mathbf{P}_i B_i^n(t)}{\sum_{i=0}^{n} w_i B_i^n(t)} \tag{7.12}$$

with control point coordinates $\mathbf{P}_i = (x_i, y_i)$ and corresponding non-negative weights w_i, the Bézier clip computation is modified as follows. Substituting equation 7.12 into equation 7.9 and clearing the denominator yields:

$$d(t) = \sum_{i=0}^{n} d_i B_i^n(t), \quad d_i = w_i(ax_i + by_i + c).$$

The equation $d(t) = 0$ expresses the intersection of $\mathbf{P}(t)$ with a line $ax + by + c = 0$. However, unlike the non-rational case, the intersection of $\mathbf{P}(t)$ with a fat line *cannot* be represented as $\{(x, y) = \mathbf{P}(t) | d_{min} \leq d(t) \leq d_{max}\}$. Instead, $\mathbf{P}$ must be clipped independently against each of the two lines bounding the fat line. Thus, we identify ranges of t for which

$$\sum_{i=0}^{n} w_i(ax_i + by_i + c - d_{max})B_i^n(t) > 0$$

or for which

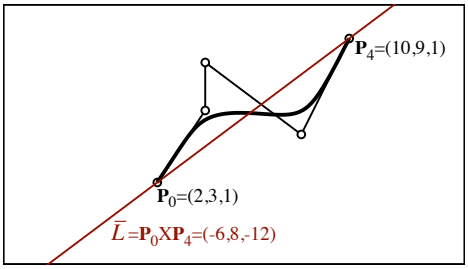
$$\sum_{i=0}^{n} w_i(ax_i + by_i + c + d_{min})B_i^n(t) < 0.$$

These ranges are identified using the Bézier clipping technique as previously outlined.

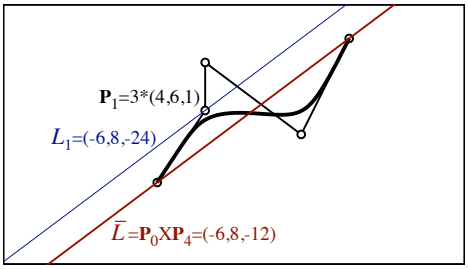
7.7.7 Example of Finding a Fat Line

We now run through an example of how to compute a fat line and perform a Bézier clip on a pair of rational Bézier curves. We use the notation for points and lines as triple of numbers, presented in Section 7.2. This leads to an elegant solution.

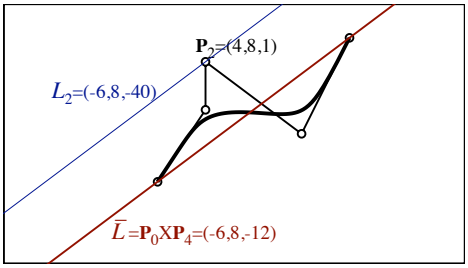
Figure 7.10.a shows a rational Bézier curve of degree $n = 4$. The control points are written in homogeneous form: $\mathbf{P}_i = w_i * (x_i, y_i, 1)$. For example, the notation $\mathbf{P}_1 = 3 * (4, 6, 1)$ means that the Cartesian coordinates are $(4, 6)$ and the weight is 3.



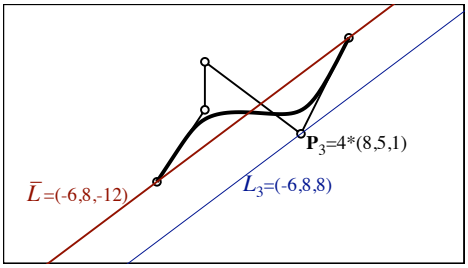
(a) Rational Bézier Curve. Line $\bar{L} = \mathbf{P}_0 \times \mathbf{P}_4$.



(b) Line L_1.



(c) Line L_1.



(d) Line L_1.

Figure 7.10: Example of how to find fat lines.

To compute a fat line that bounds this curve, begin by computing the base line $\bar{L} = \mathbf{P}_0 \times \mathbf{P}_n = (-6, 8, -12)$. Then, compute the lines L_i, $i = 1, \ldots, n-1$ that are parallel to $\bar{L}$ and that pass through $\mathbf{P}_i$. In general, if $\bar{L} = (a, b, c)$, then $L_i = (a, b, c_i)$. Thus, we must solve for c_i such that

$(a, b, c_i) \cdot \mathbf{P}_i = 0$. Thus,

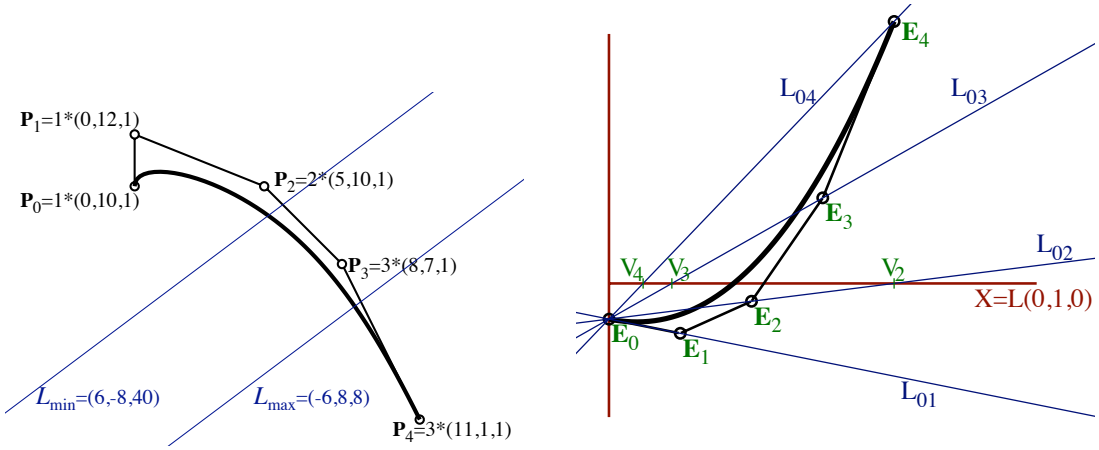
$$c_i = -ax_i - by_i.$$

Call the line with the smallest value of c_i L_{min} and the line with the largest value of c_i L_{max}. We want the curve to lie in the positive half space of each bounding line. It can be shown that this will happen if we scale L_{min} by -1.

Thus, in our example, we have

$$L_{min} = -L_2 = (6, -8, 40), \qquad L_{max} = L_3 = (-6, 8, 8). \tag{7.13}$$

7.7.8 Example of Clipping to a Fat Line



(a) Curve and fat line.

(b) Explicit Bézier Curve for Clipping against L_{min}.

Figure 7.11: Clipping to a fat line.

Figure 7.11.a shows a rational Bézier curve to be clipped against the fat line (7.13). Figure 7.11.b shows the resulting explicit Bézier curve where

$$E_i = (\frac{i}{n}, L_{min} \cdot \mathbf{P}_i, 1).$$

In this example,

$$E_0 = (0, -40, 1), \quad E_1 = (\frac{1}{4}, -56, 1), \quad E_2 = (\frac{1}{2}, -20, 1), \quad E_3 = (\frac{3}{4}, 96, 1), \quad E_4 = (1, 294, 1)$$

Note that we only want to clip away the portions of $\mathbf{P}(t)$ that lie in the negative half space of L_{min}. That half space corresponds to the values of t for which the explicit curve is < 0. Therefore, we can clip away the "left" portion of the curve if and only if $\mathbf{E}_0$ lies below the x-axis and we can clip away the "right" portion of the curve if and only if $\mathbf{E}_n$ lies below the x-axis. So in this example, we can compute a clip value at the "left" end of the curve, but not the "right" end.

To compute the clip value, we first compute the lines

$$L_{0,i} = E_0 \times E_i$$

and the points

$$L_{0,i} \times \mathbf{X} = V_i = (a_i, 0, c_i)$$

where $\mathbf{X} = (0, 1, 0)$ is the line corresponding to the x-axis ($y = 0$). In our example,

$$L_{01} = (16, \frac{1}{4}, 10), \quad L_{02} = (-20, \frac{1}{2}, 20), \quad L_{03} = (-136, \frac{3}{4}, 30), \quad L_{04} = (-334, 1, 40)$$

and

$$V_1 = (-10, 0, 16), \quad V_2 = (-20, 0, -20), \quad V_3 = (-30, 0, -136), \quad V_4 = (-40, 0, -334).$$

Our "clip" value will be the x-coordinate of the left-most V_i that is to the right of the origin. To determine this, we must project the homogeneous points V_i to their corresponding Cartesian points v_i which yields:

$$v_1 = (-\frac{5}{8}, 0), \quad v_2 = (1, 0), \quad v_3 = (\frac{30}{136}, 0) \approx (.2206, 0), \quad v_4 = (\frac{40}{334}, 0) \approx (.1198, 0).$$

So the desired t value at which to clip against L_{min} is .1198, and we can eliminate the domain $t \in [0, 0.1198)$.

We now clip against line L_{max}. The explicit Bézier curve is shown in Figure 7.12, for which

$$E_i = (\frac{i}{n}, L_{max} \cdot \mathbf{P}_i, 1).$$

In this example,

$$E_0 = (0, 88, 1), \quad E_1 = (\frac{1}{4}, 104, 1), \quad E_2 = (\frac{1}{2}, 116, 1), \quad E_3 = (\frac{3}{4}, 48, 1), \quad E_4 = (1, -150, 1)$$

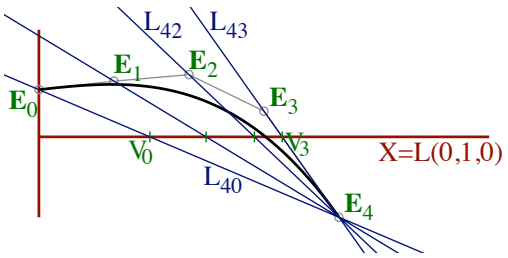


Figure 7.12: Clipping to L_{max}.

Since E_0 is above the x-axis and E_4 is below the x-axis, we can clip the right side of the curve but not the left. We have

$$L_{40} = (-238, -1, 88), \quad L_{41} = (-254, -\frac{3}{4}, 141.5), \quad L_{42} = (-266, -\frac{1}{2}, 191), \quad L_{43} = (-198, -\frac{1}{4}, 160.5)$$

and

$$V_0 = (-88, 0, -238), \quad V_1 = (-141.5, 0, -254), \quad V_2 = (-191, 0, -266), \quad V_3 = (-160.5, 0, -198).$$

Projecting the homogeneous points V_i to their corresponding Cartesian points v_i yields

$$v_0 \approx (.3697, 0), \quad v_1 \approx (.5571, 0), \quad v_2 \approx (.7180, 0), \quad v_3 \approx (.8106, 0).$$

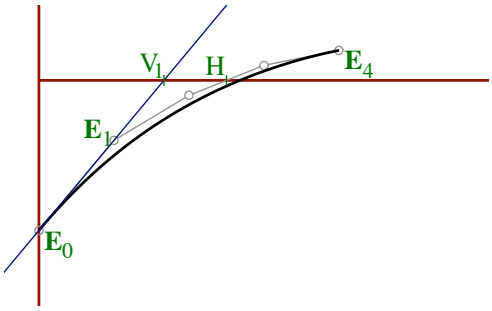
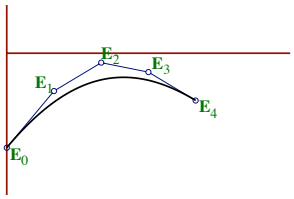


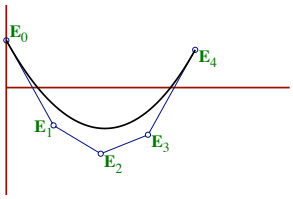
Figure 7.13: Clipping example.

When clipping away the right end of the curve, we need to identify the right-most V_i that is to the left of $(1, 0)$, which in this example is V_3. Thus, we can clip away $t \in (.8106, 1]$.

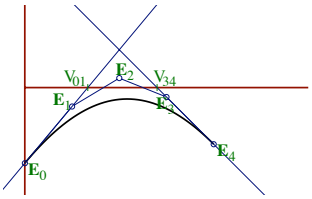
Note that we are not computing the exact intersection between the convex hull and the x-axis, as illustrated in Figure 7.13, where the algorithm described in this section would compute a clip value corresponding to V_1, whereas the convex hull crosses the x-axis at H. In our experiments, the method described here runs faster than the method where we compute the exact intersection with the convex hull because in most cases the two values are the same and the method described here is more simple.



(a) Clip away everything.



(b) Clip away nothing.



(c) Clip to the left of V_{01} and to the right of V_{34}.

Figure 7.14: Additional examples.

A few additional examples are illustrated in Figure 7.14. In Figure 7.14.a, all control points lie below the x-axis. In this case, there is no intersection. In Figure 7.14.b, E_0 and E_n lie above the x-axis. In this case, no clipping is performed. In Figure 7.14.c, clipping values are computed for both the left and right sides of the curve.