Project number IST-006413

ACS

Algorithms for Complex Shapes
with Certified Numerics and Topology

Extension of geometric filtering techniques to higher-degree parametric curves

Günter Rote

ACS Technical Report No.: ACS-TR-361503-01

Part of deliverable: WP-I/D5
Site: FUB
Month: 36

Project co-funded by the European Commission within FP6 (2002–2006)
under contract nr. IST-006413

Extension of Geometric Filtering Techniques to Higher-Degree Parametric Curves: Curve Intersection by the Subdivision-Supercomposition Method

Günter Rote

April 31, 2008

Abstract

We present a subdivision algorithm for computing the intersection of spline curves. The complexity depends on geometric quantities that represent the hardness of the computation in a natural way, like the angle of the intersection. The main idea is the application of the super-composition technique, which considers unions of adjacent parameter intervals that are not siblings in the subdivision tree. This approach addresses the common difficulty of non-termination of the classical subdivision approach when the intersection coincides with a subdivision point, but it avoids the numerical overhead associated to alternative methods like a random shift of the parameter.

1 Introduction

Easy and Hard Cases of Intersection. A basic task on geometric computing is intersection of spline curves. Depending on the position of the curves, this task may be have different levels of difficulty, at least from a visual point of view There are easy cases (Figure 1):

- The curves intersect *transversely*, forming a large angle.
- There is *large* distance between curves, and it is obvious that there is no intersection.

On the other hand, there are cases that appear hard (Figure 2):

- intersections with small angle or even *tangency*;
- curves come *close* without intersecting;
- the endpoint of one curve lies near the other curve.

These cases are intrinsically hard: Depending on small perturbations of the data, the number of intersection may change.

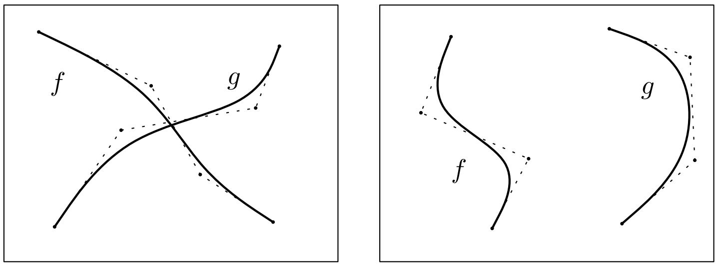

Figure 1: Easy cases for curve intersection

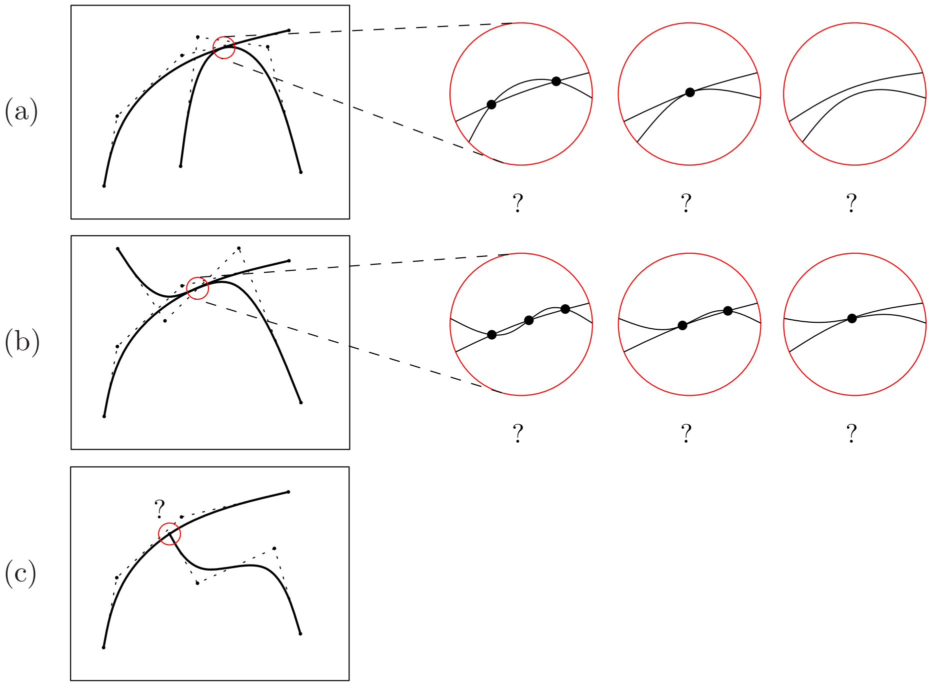

Figure 2: Inherently hard cases for curve intersection

We will present a simple intersection algorithm based on the subdivision paradigm, whose running time adapts to the intrinsic hardness. The hardness is expressed in terms of geometric quantities, like the angle of intersection, or the local minimum of the distance between the curves.

Bézier splines. In this paper we will mostly consider Bézier splines, for the sake of being explicit, but the method works also for other types splines. A Bézier spline of degree d is specified by a *control polygon* consisting of $d+1$ control points $P_0, P_1, \ldots, P_d$, and it is the parametric curve given by

$$f(t) = \sum_{i=0}^{d} \binom{d}{i} \frac{(t-t_1)^i (t_2-t)^{d-i}}{(t_2-t_1)^d} \cdot P_i,$$

for $t_1 \leq t \leq t_2$.

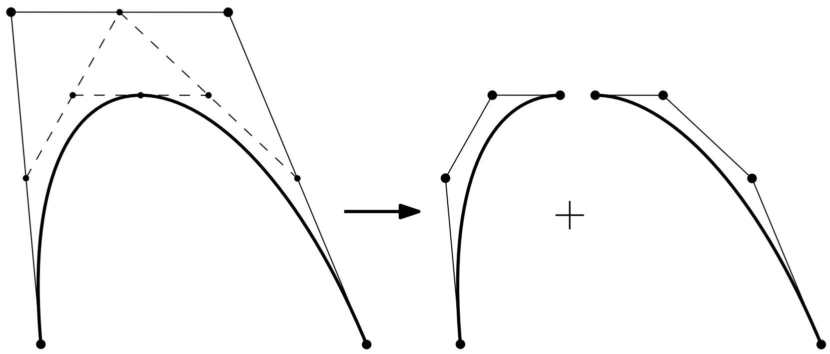

Figure 3: Bézier subdivision

```
Create a stack with (f, g) as the only element
while the stack is not empty:
  POP (f, g) from stack
  if f and g are guaranteed to have no intersection (A):
      discard f, g
  elsif f and g are guaranteed to have a unique intersection (B)
     and the precision of intersection is good enough:
      report intersection
  elsif f and g are smaller than a predefined error threshold (C):
      report potential intersection
  else:
      subdivide the larger curve (say, f) into f1 and f2.
      PUSH (f1, g) on the stack
      PUSH (f2, g) on the stack
```

Figure 4: The basic subdivision algorithm

A Bézier spline defined over the interval $[t_1, t_2]$ can be subdivided at any point t^* in this interval. The two parts over the intervals $[t_1, t^*]$ and $[t^*, t_2]$ are again Bézier splines (of the same degree), and their control polygons can be computed by a simple geometric scheme, de Casteljau's method (cf. [3]). The simplest case, when the parameter interval is subdivided into two equal parts is illustrated in Figure 3 for the case of cubic Bézier splines.

Intersection by Subdivision. The basic scheme for an intersection algorithm based on subdivision is shown in Figure 4. It requires some sufficient conditions (A) and (B) for concluding that there is no intersection (a rejection criterion) or a unique intersection (an acceptance criterion). When no conclusion can be reached by either criterion, the algorithms subdivides the curves and treats the pieces recursively. Moreover, there is an "emergency stop" (C) for terminating the recursion when the size of the curves diminishes beneath a specified error tolerance.

A sufficient condition for disjointness is based on the fact that the curve is

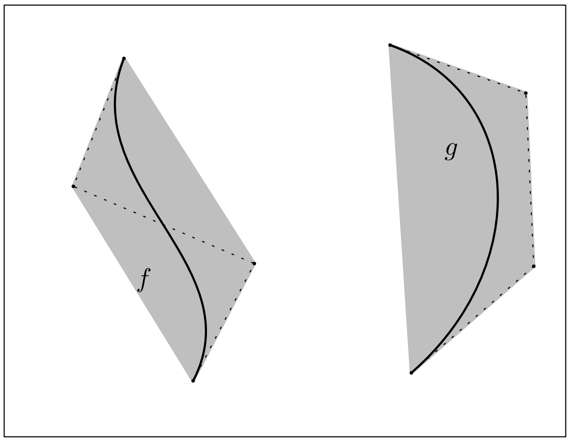

Figure 5: Disjoint convex hulls are a sufficient condition for disjointness.

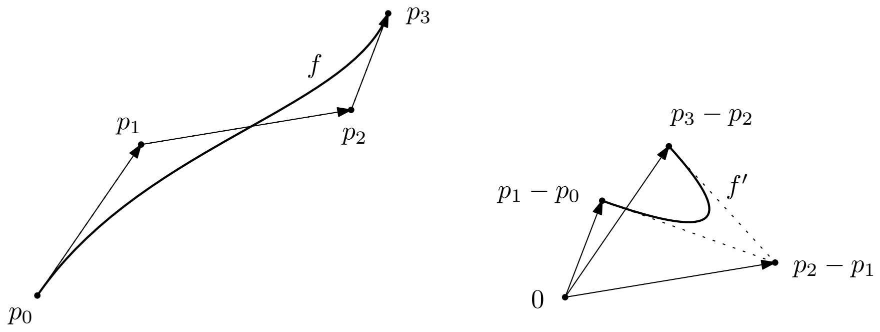

Figure 6: The hodograph gives the derivative of a Bézier curve.

inside the convex hull of the control polygon (Figure 5):

> Condition (A):
> If the convex hulls of the two control polygons are disjoint, the curves have no intersection.

A sufficient condition for unique crossing was given by Sederberg and Meyers [4]. It is based on the hodograph [1]: The derivative of a Bézier curve $f(t)$ is a Bézier curve $f'(t)$, of degree one less, called the *hodograph* of f, see Figure 6. If f has degree d and is defined on a parameter interval of length h, the control polygon of f' is formed from the differences $p_{i+1} - p_i$ of the original control polygon, times the constant factor d/h.

We have the following acceptance, see Figure 7.

Criterion (B).

- If the convex hulls of f and g "cross" (the endpoints of each of f and g stick out of the other curve's convex hull, and the endpoints of f alternate with the endpoints of g in the cyclic order),
- and the cones of directions of f' and g' are disjoint,

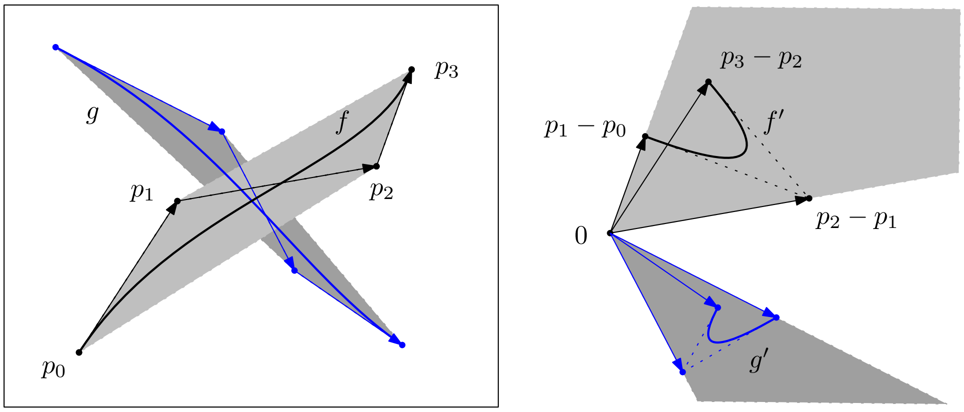

Figure 7: A sufficient condition for a unique crossing.

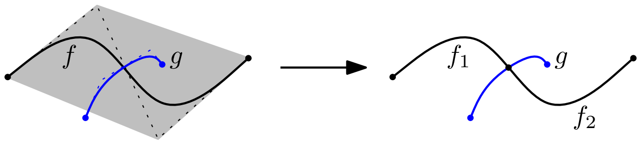

Figure 8: The subdivision algorithm may fail for easy cases.

then f and g intersect in a single point.

When no decision is reached, one of the curves is subdivided, and the test is repeated for the pieces. The simplest and most natural choice is to subdivide the parameter interval at the midpoint. This is also the most economical choice in terms of bit complexity: For Bézier curves of order d, every subdivision step increases the necessary number of bits for storing the exact coordinates of the control polygon by d. (This can be reduced to $d-1$ by separately storing the leading bits that are common to all control points.)

Possible Failure in Easy Cases. A good algorithm should be fast in identifying intersections or disjointness for easy cases, but probably one is ready to accept that it may take a long time to sort out hard cases.

However the basic subdivision algorithm of Figure 4 may have a problem to terminate even with easy cases. Consider the two curves f and g in the easy-looking example of Figure 8. Conditions (A) and (B) do not apply, and thus f is subdivided into f_1 and f_2. Suppose that the subdivision point happens to fall on g, or very close to g. Now the easy case has turned into a hard case like Figure 2c: the algorithm will continue to subdivide f and g further around this point, in an attempt to find out whether f_1 intersects g, and whether f_2 intersects g. The recursion might only be stopped by the emergency stop (C), without reaching a definite conclusion about the intersection. Even if the proper answer is eventually reached, it can take an unnecessarily long time.

Related Work. Eigenwillig et al. [2] have addressed precisely the same problem in a more special context: finding the root of a polynomial by subdivision.

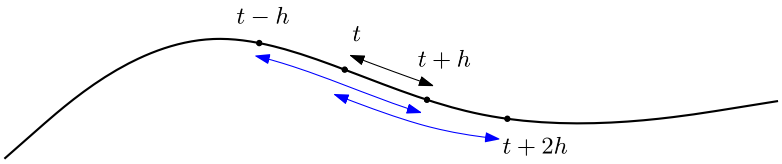

Figure 9: The main principle of the Subdivision-Supercomposition Method

(This can be viewed as a special case of curve intersecting by specializing one curve to be the x-axis. As acceptance and rejection conditions, they use a termination condition based on Descartes' Rule of Signs.) Eigenwillig et al. avoid the problem of slow termination by subdividing the curve not at the midpoint, but at a random point. The drawback of this solution is that it incurs a larger overhead in terms of bit complexity. The number of necessary bits for the random subdivision point must be balanced against the likelihood of hitting the intersection point (the zero of the polynomial) too closely. This would lead to a failure of the algorithm, making it necessary to restart the algorithm with a new random choice.

In contrast, our new Subdivision-Supercomposition method avoids the problem without any random choice and extra numerical overhead.

Yap [5] has given the first algorithm for intersection of Bézier splines that is *complete* in the sense that it produces the correct output for all inputs, even in cases with tangential intersections or stationary points on one of the curves. The algorithm is geometric, but unlike our approach, the termination criterion depends on separation bounds based on the algebraic representation of the curves and the size of the coordinates.

Our Subdivision-Supercomposition Algorithm can complement the "macro phase" of Yap's algorithm.

2 The Subdivision-Supercomposition Algorithm

The above discussion of the example in Figure 8 highlights why the subdivision algorithm runs into trouble. After subdividing f into f_1 and f_2, the algorithm goes to great lengths to find out *which* of f_1 or f_2 intersects g. However, this is a dichotomy that has only been created by the algorithm, and is of no intrinsic interest for the problem per se.

The solution is to avoid the strictly dichotomic subdivision procedure, which refines the parameter interval into disjoint parts, and consider a covering of the whole parameter interval by *overlapping* subintervals.

Our *Subdivision-Supercomposition Method* is based on binary subdivision, but transcends it by the following rule, see Figure 9:

> With every parameter interval $[t, t{+}h]$, we also check the two "parent intervals" $[t-h, t+h]$ and $[t, t+2h]$ (provided they are contained in the initial parameter interval of the whole curve).

In one of the parent intervals leads to a conclusion in one way or another by condition (A) or (B), then, of course, the original interval need not be considered any longer. Note that rule applies to both f and g simultaneously. Thus, when the original subdivision algorithm tests one pairs f, g, the above rule requires nine pairs to be tested. (We will see below that the effort is actually not as bad as this.)

This rule consider intervals that transcend the boundaries of the binary subdivision tree. Consider the interval marked by an arrow in the tree of Figure 10. When this interval is considered, the two "parent" intervals that are shown is also considered: The left one is actually the true parent interval in the tree, and has been considered already before reaching the current interval. The right one is not present in the tree.

There are two ways of creating the curve for this interval.

(i) One creates the "sibling" interval of the same length and combines the information from the two curve pieces. If the sibling interval is not yet present in the tree, the missing subtree must be created at this point (the dotted subtree in Figure 10).

This method requires some storage and book-keeping. On the other hand, some book-keeping is required anyway, to record which intervals of f have been compared to which intervals of g, and to prevent an intersection point from being reported several times; in addition it may save the effort of considering the same parent interval several times.

The number of extra subdivision steps is only a constant factor larger than the number of original subdivision steps. The analysis is the same as for balancing a quad-tree, where spatially adjacent leaves are allowed to differ by at most one in depth. (In the example, the parent of the dotted subtree must already have been created in a previous iteration.)

The information from the two adjacent subintervals can be combined in a straightforward way. For example, in the rejection criterion (A), one can simply take the convex hull of the union of the two control polygons. The acceptance criterion can be adapted similarly.

(ii) The de Casteljau subdivision method can also be applied to a "subdivision" point outside the given parameter interval. In this manner, it works as an extrapolation method. The missing parent interval can be created without extra storage and book-keeping.

The method is called the Subdivision-Supercomposition Method, because normal "sub-division" is complemented by the opposite operation of forming a larger interval from two adjacent subintervals: by "super-composing" them.

In some cases, method (i) is the only choice. For example, the Subdivision-Supercomposition Method readily extends to other types of spline curves, like B-splines. In this case, the sibling interval may not be part of the same initial control polygon; it is part of a separate polynomial curve.

3 Analysis of the Algorithm

We can now formulate the theorem about the running time of the Subdivision-Supercomposition Algorithm.

Theorem 1. *Suppose that the following conditions are satisfied:*

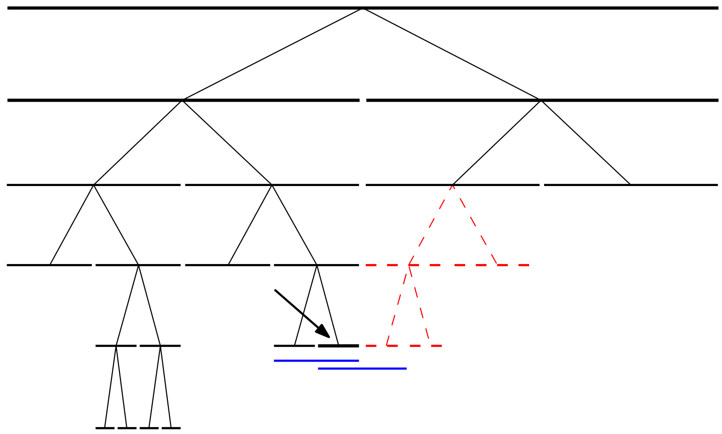

Figure 10: The binary subdivision tree

1. *f and g are Bézier curves of degree (at most) d, with initial parameter interval* $[0,1]$*,*

2. *the diameter of the control polygon of f and g is at most D,*

3. $\|f'(t)\| \geq v_{\min}$ *and* $\|g'(t)\| \geq v_{\min}$ *everywhere,*

4. *every intersection point, the curves cross at an angle at least α, and*

5. *the distance between f and g is at least ε, at every local minimum of the distance and at every endpoint of f or g,*

then the Subdivision-Supercomposition Algorithm terminates after at most

$$L := \max\left\{\log_2 \frac{D}{v_{\min} \cdot \alpha},\ \log_4 \frac{D}{\varepsilon}\right\} + 2\log_2 d + 4.$$

levels of subdivision.

The number of iterations is thus at most

$$2^L \times 2^L = O\left(\frac{D^2}{(v_{\min} \cdot \alpha)^2} + \frac{D}{\varepsilon}\right).$$

The assumptions of the theorem are illustrated in Figure 11. Conditions 4 and 5 forbid tangential crossings and tangential intersections. In addition, Condition 5 prevents one curve from terminating on the other curve. Condition 3 excludes stationary points where the derivative is zero.

Proof. From the assumptions, we conclude immediately

$$\|f'(t)\|, \|g'(t)\| \leq 2dD$$
$$\|f''(t)\|, \|g''(t)\| \leq S := 4d(d-1)D.$$

Together with the assumption that $\|f'(t)\|, \|g'(t)\| \geq v_{\min}$, we derive that the curvature of f and g is at most $S/v_{\min}$.

From these bounds, it is easy to derive the following lemma, which says that for small pieces ($h \to 0$) the curve, including its control polygon, runs essentially straight.

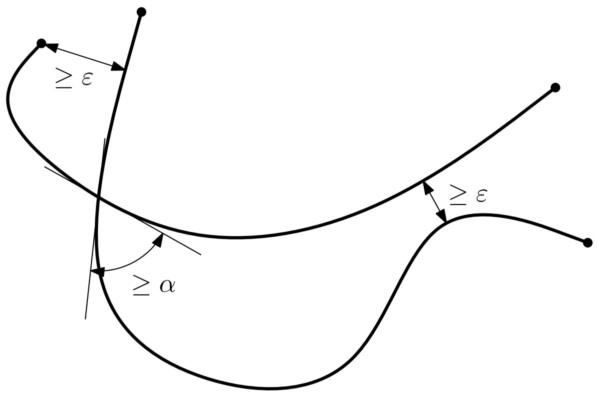

Figure 11: The assumptions of Theorem 1

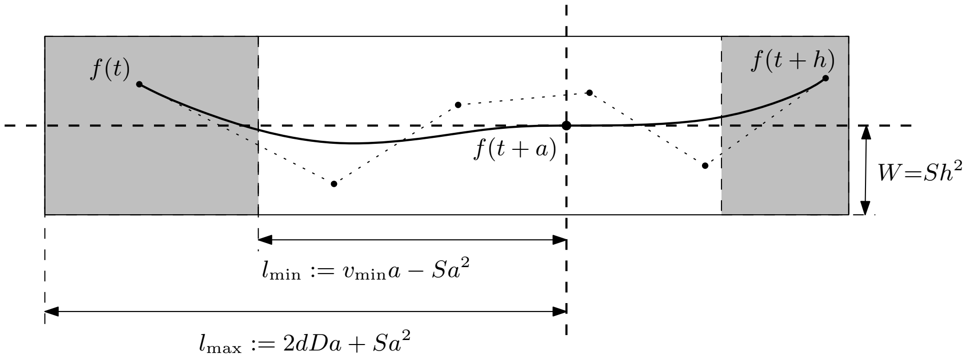

Figure 12: The sleeve lemma

Lemma 1 (The Sleeve Lemma). *Consider the curve f over the parameter interval $[t, t+h]$, and let $f(t+a)$ for $0 \leq a \leq h$ be a point in this interval. Assume without loss of generality that the tangent direction $f'(t+a)$ is horizontal, see Figure 12. Then the curve and the control polygon is contained in a horizontal rectangular strip of height (width) $W = S \cdot h^2$ and length $2dDh + Sh^2 = \Theta(h)$ whose horizontal axis goes through $f(t+a)$. The left endpoint has horizontal distance at least $l_{\min} := v_{\min}a - Sa^2$ from $f(t+a)$, and the right endpoint has horizontal distance at least $l'_{\min} := v_{\min}(h-a) - S(h-a)^2$ from $f(t+a)$.*

Proof. The proof is straightforward, considering that the tangent vector $f'(t+a)$ has the form $(\pm u, 0)$ with $v_{\min} \leq u \leq 2dD$, and the norm of the second derivative is bounded by $4d^2D$. □

To continue the proof of Theorem 1, let us now assume that the algorithm has subdivided the starting interval to depth L, creating an interval of size $h = 2^{-L}$. Consider first the case that the interval contains an intersection point. Then, by the main rule of the Subdivision-Supercomposition Algorithm, the algorithm must have checked an interval where the intersection point is at least $h/2$ away from both ends, see Figure 13.

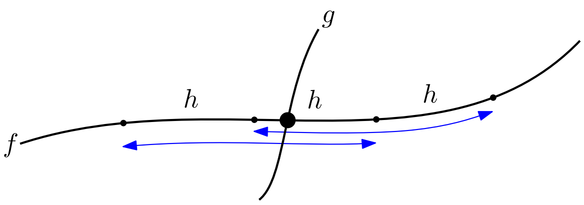

Figure 13: There is always an interval whose endpoint is far from the intersection point.

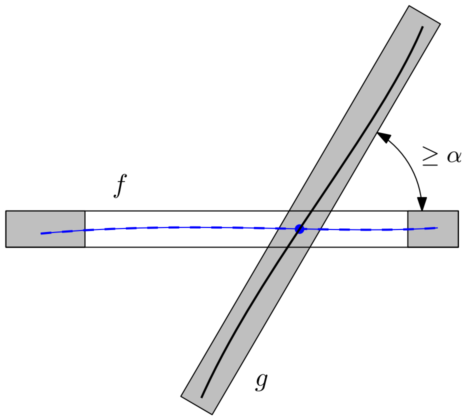

Figure 14: An intersection is detected by criterion (A).

Let f denote the corresponding piece. Since the intersection angle α is large, by assumption 4, and f is contained in a long and narrow strip, by the Sleeve Lemma, the endpoints of f must stick out of the convex hull of the control polygon of the other curve, by a straightforward geometric computation, see Figure 14. A symmetric argument applies to g, and thus criterion (A) applies. In addition, we conclude that the distance of the endpoints of g from f is at least

$$v_{\min} \cdot \frac{h}{2} \cdot \sin \alpha.$$

Let us now consider the case when f contains no intersection, see Figure 15.

Lemma 2. *Let g be a piece of a curve that has no intersection with f. Then the distance of its endpoints from f is at least*

$$\min\{\varepsilon, v_{\min} \cdot \frac{h}{2} \cdot \sin \alpha\}.$$

Proof. We start from an endpoint of g and move along the curve of which g is part, in a direction in which the distance from f decreases, monitoring this distance along the way. There are two possibilities:

- We reach a local minimum of the distance, or we reach an endpoint of the original curve. In either case, the distance from f is at least ε, by assumption 5. Since the distance was monotonically decreasing until we reached

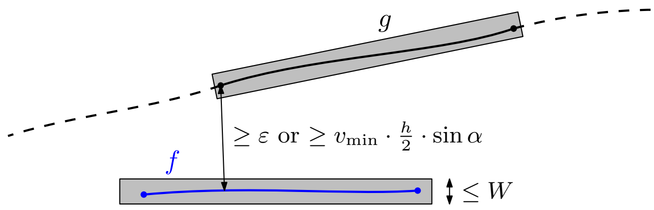

Figure 15: When the curves are disjoint, criterion (B) applies.

this point, the original distance from f is also at least ε. (The same argument holds if the endpoint from which we start is a local minimum, and we cannot move in either direction.)

- We reach an intersection. When we enter the subinterval of length $h = 2^{-L}$ containing this intersection, we know from the considerations above that the distance from f is at least $v_{\min} \cdot \frac{h}{2} \cdot \sin\alpha$. By the same monotonicity argument as before, this bounds holds also for the endpoint from which we started. □

A straightforward calculation shows that both ε and $v_{\min} \cdot \frac{h}{2} \cdot \sin\alpha$ are bigger than $2W$, where W is the width of the rectangle to which the control polygons of f and g are confined by the Sleeve Lemma. Thus, we can conclude that the control polygons have a positive distance from each other, and criterion (B) will terminate the recursion. □

Acknowledgement. I am grateful to the members of the ACS project team at the September 2005 kick-off meeting in Zurich for helpful discussions. In particular, I wish to thank Danny Halperin for suggestions in naming the algorithm of this paper.

References

[1] Apostulatos, T. A. Hodograph: A useful geometrical tool for solving some difficult problems in dynamics. *American Journal of Physics 71*, 3 (2003), 261–266.

[2] Eigenwillig, A., Kettner, L., Krandick, W., Mehlhorn, K., Schmitt, S., and Wolpert, N. A Descartes algorithm for polynomials with bit-stream coefficients. In *Computer Algebra in Scientific Computing, 8th International Workshop, CASC 2005, Kalamata, Greece, September 12-16, 2005, Proceedings* (2005), V. G. Ganzha, E. W. Mayr, and E. V. Vorozhtsov, Eds., vol. 3718 of *Lecture Notes in Computer Science*, Springer, pp. 138–149.

[3] Farin, G. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, 1993.

[4] SEDERBERG, T. W., AND MEYERS, R. J. Loop detection in surface patch intersections. *Comput. Aided Geom. Des. 5*, 2 (1988), 161–171.

[5] YAP, C.-K. Complete subdivision algorithms, I: intersection of Bezier curves. In *Proc. 22nd Annual Symposium on Computational Geometry, Sedona, Arizona, USA, June 5–7, 2006* (2006), N. Amenta and O. Cheong, Eds., ACM, pp. 217–226.