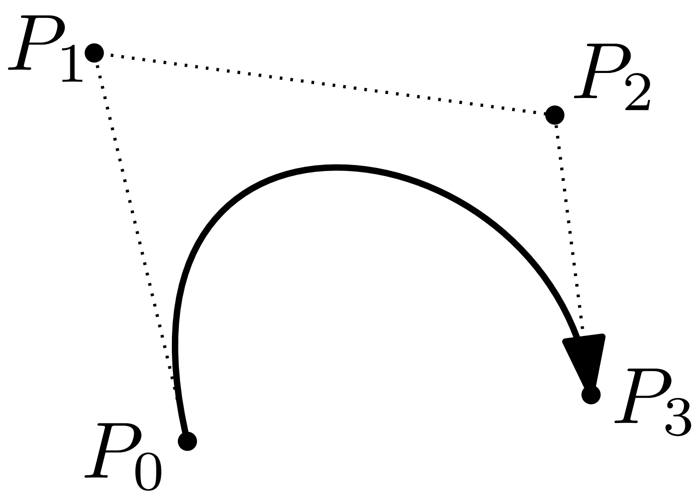
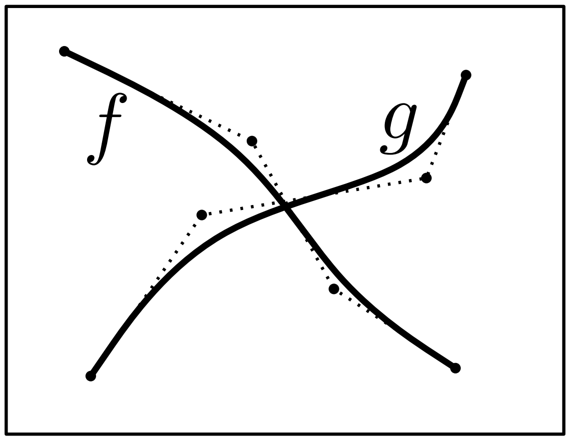
Adaptive Intersection of Bézier Splines by the SUPER-COMPOSITION Method

Günter Rote
Freie Universität Berlin, Institut für Informatik

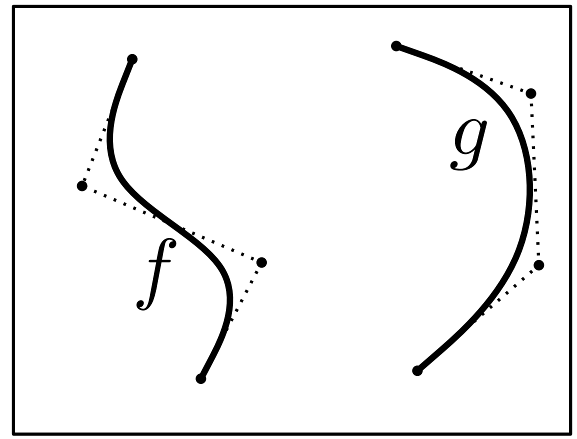


$$f(t) = \underbrace{\sum_{i=0}^{d} \binom{d}{i} t^i (1-t)^{d-i}}_{\text{Bernstein polynomials}} \cdot P_i$$

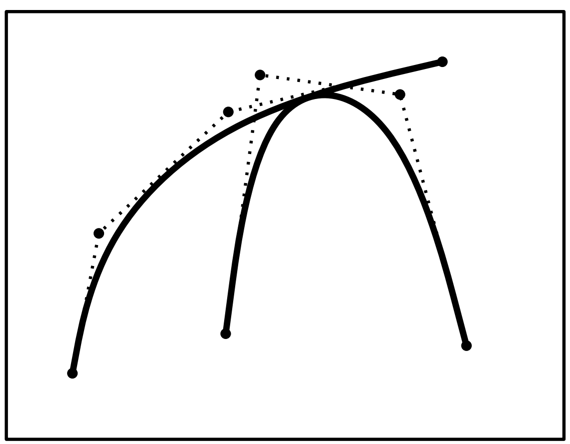
Intersecting two Bézier splines



easy

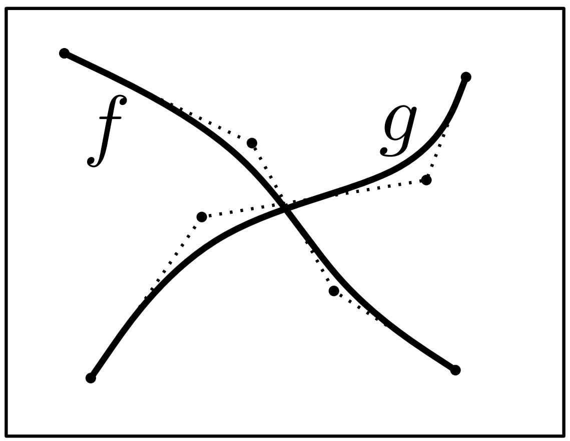


easy

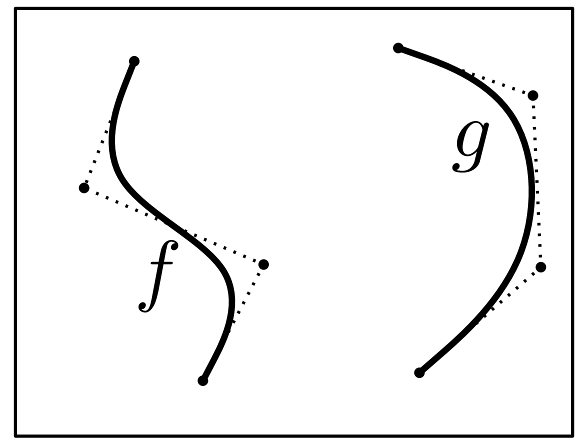


hard

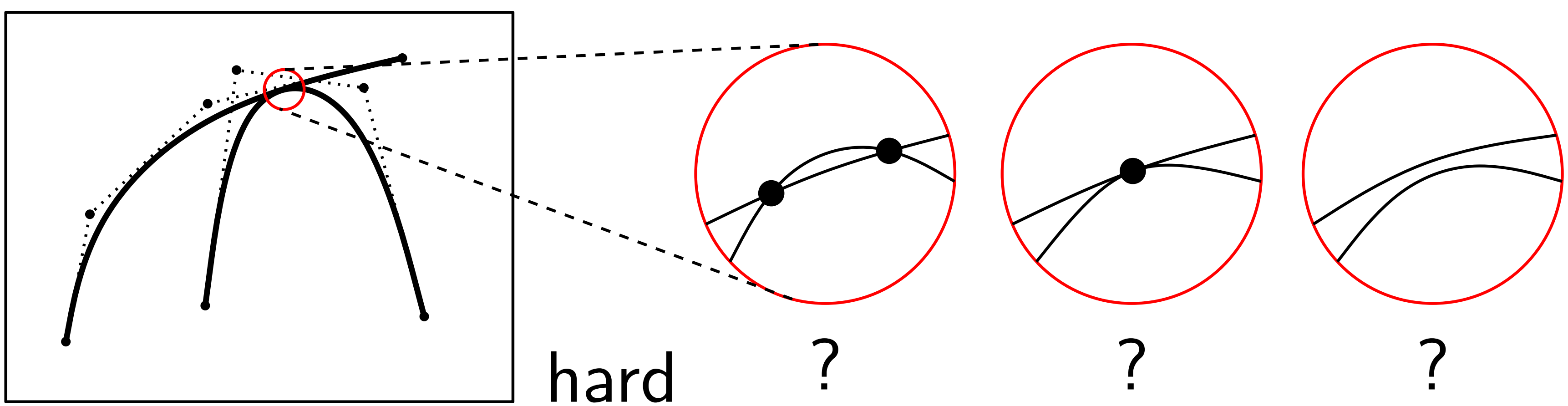
Intersecting two Bézier splines



easy



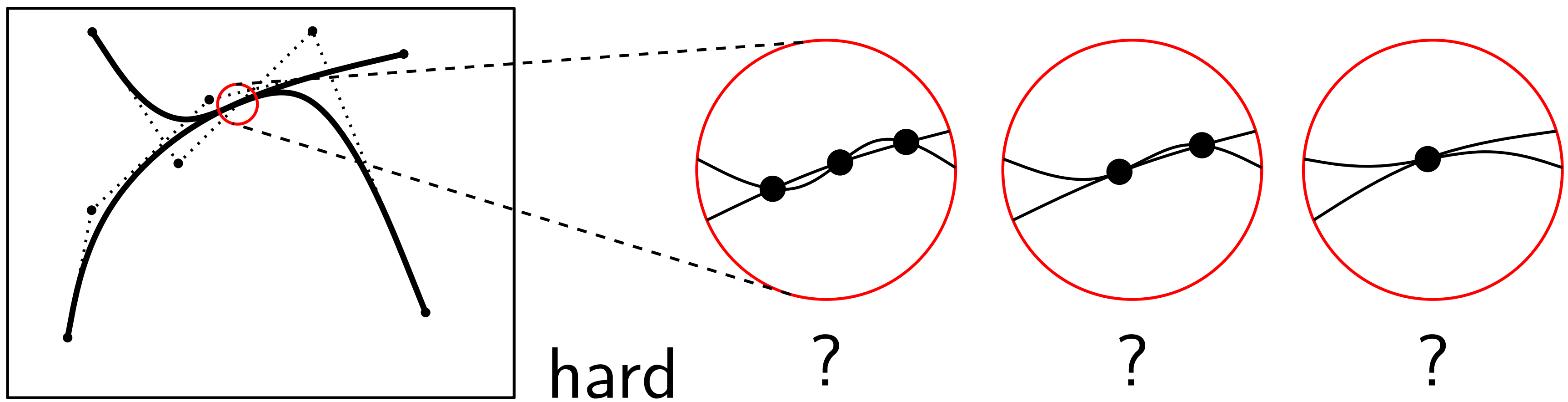
easy

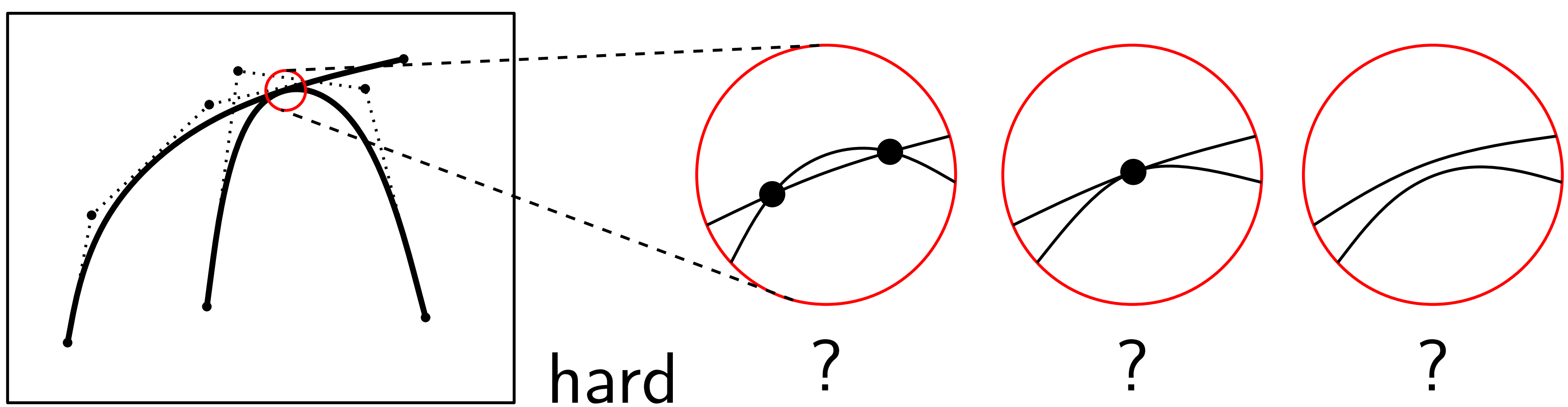


hard

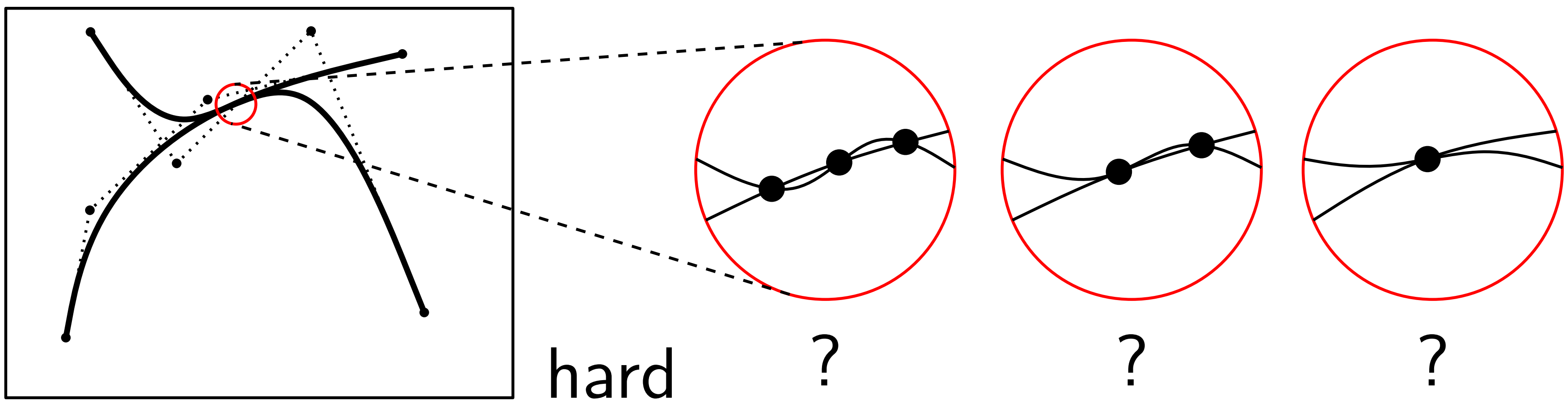
? ? ?

other hard cases

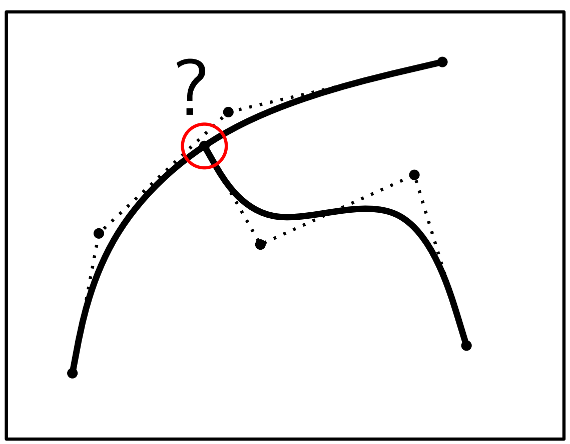




other hard cases



hard



hard

Intersecting two Bézier splines

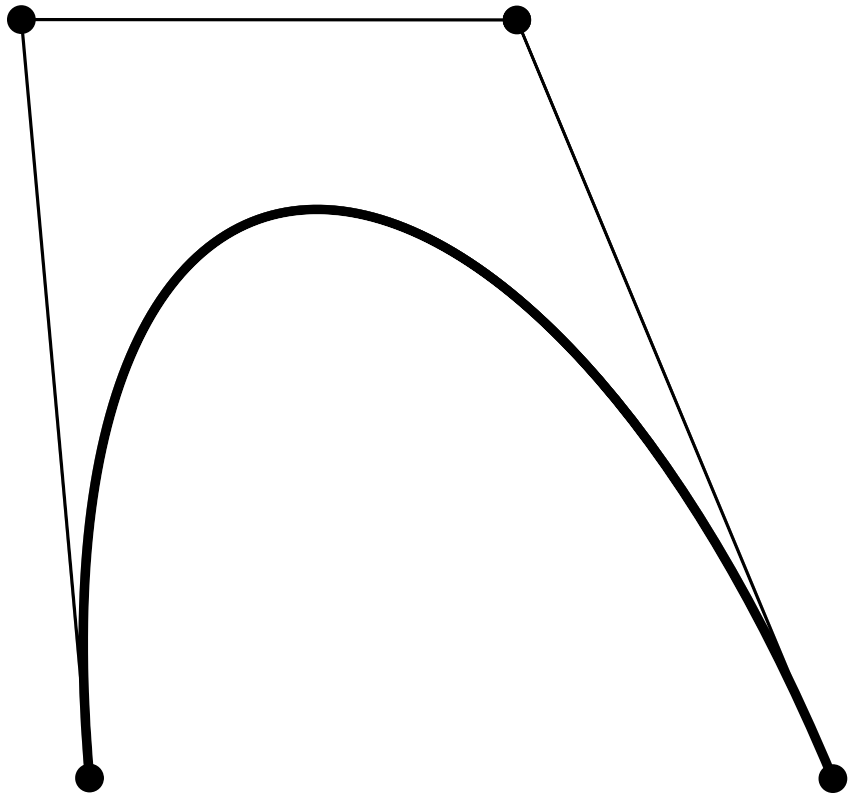
Easy cases:

- *transverse* intersection (large angle)
- *large* distance between curves

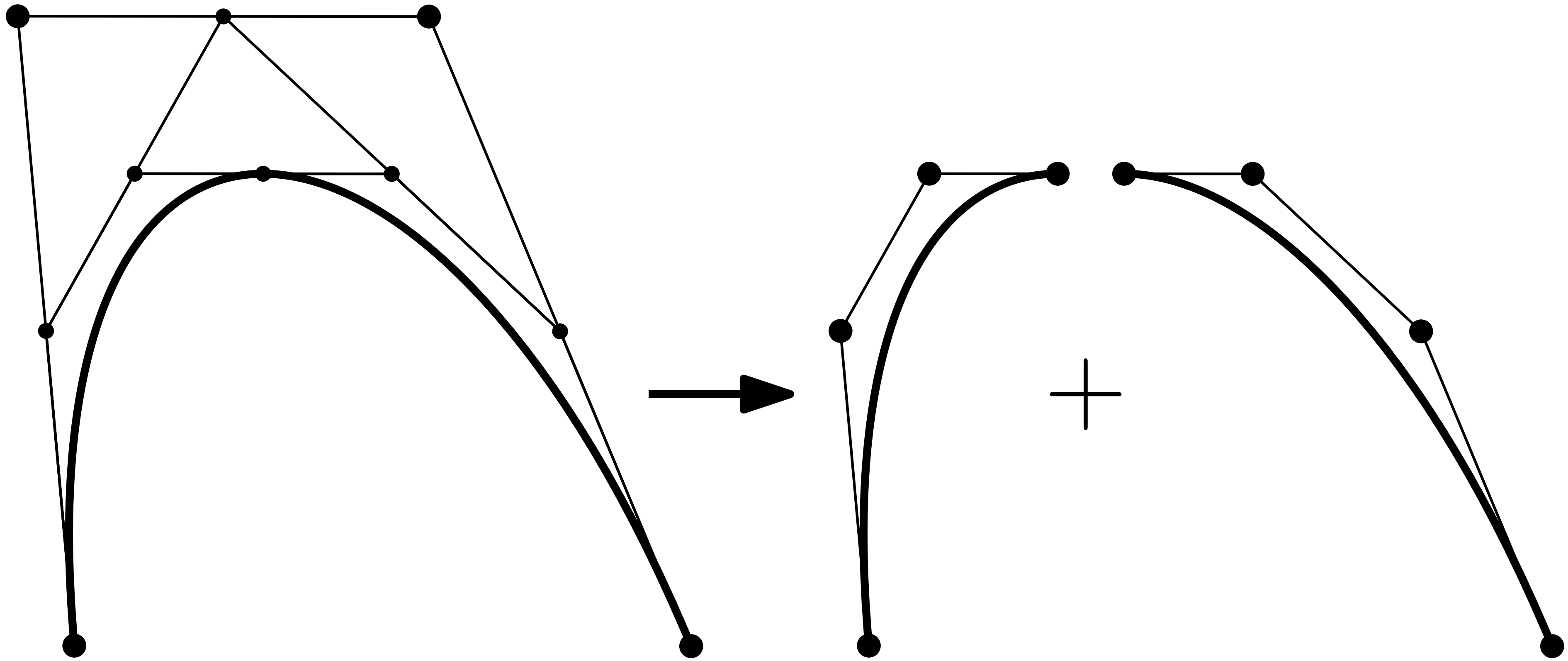
Hard cases:

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

Bézier curve subdivision



Bézier curve subdivision

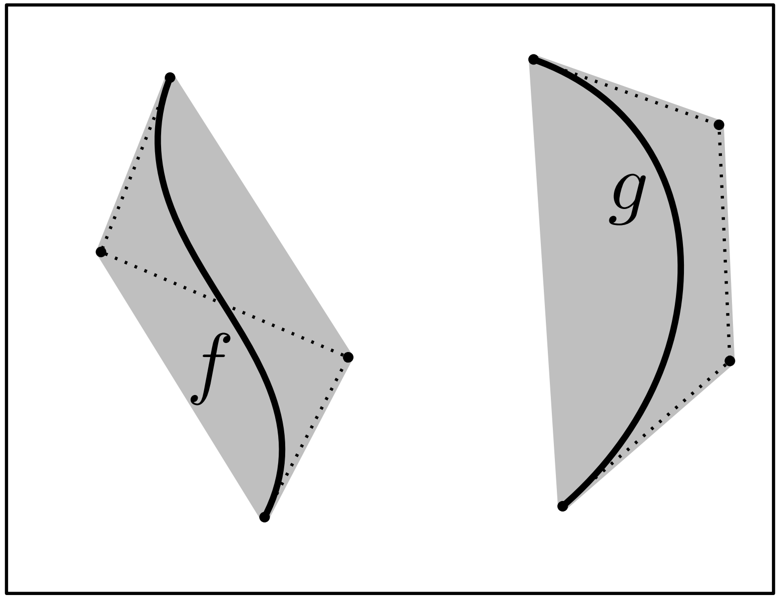


Intersection algorithm by subdivision

Freie Universität Berlin

PUSH (f, g) on stack
while stack is not empty:
  POP (f, g) from stack
  if f, g are guaranteed to have no intersection:
    discard f, g
  elsif f, g are guaranteed to have a unique intersection
   and the precision of intersection is good enough:
    report intersection
  else:
    subdivide the larger curve (say, f) into f_1 and f_2.
    PUSH (f_1, g) on stack
    PUSH (f_2, g) on stack

Sufficient condition for disjointness



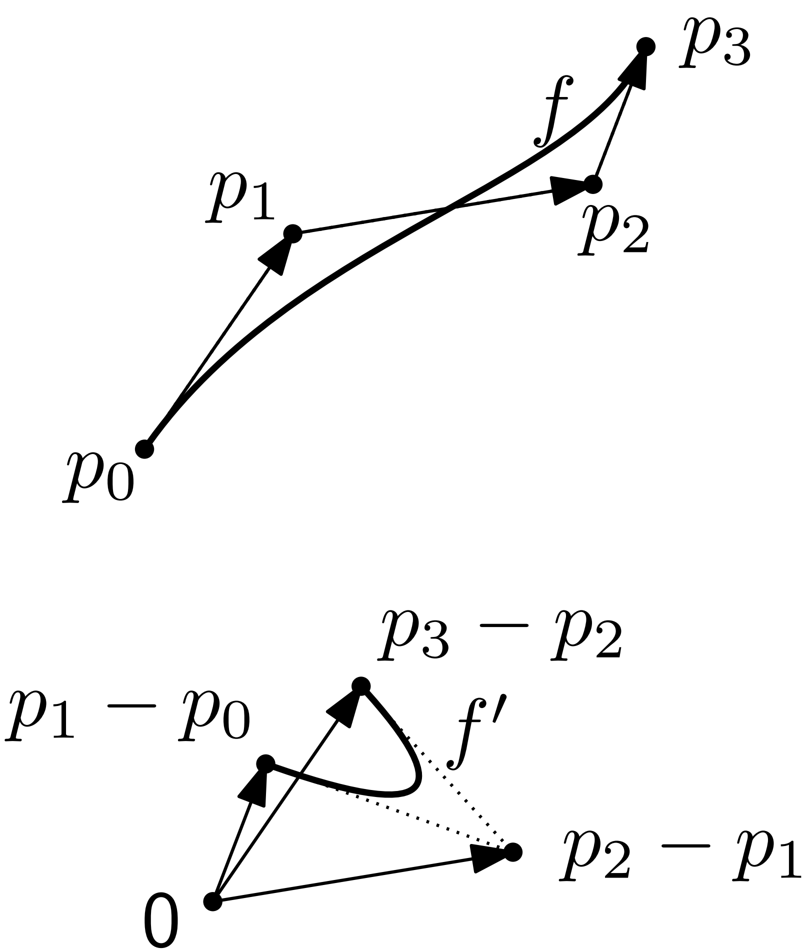
(easy)

The curve is inside the convex hull of the control polygon.

convex hulls disjoint $\Rightarrow$ no intersection.

Sufficient condition for unique crossing

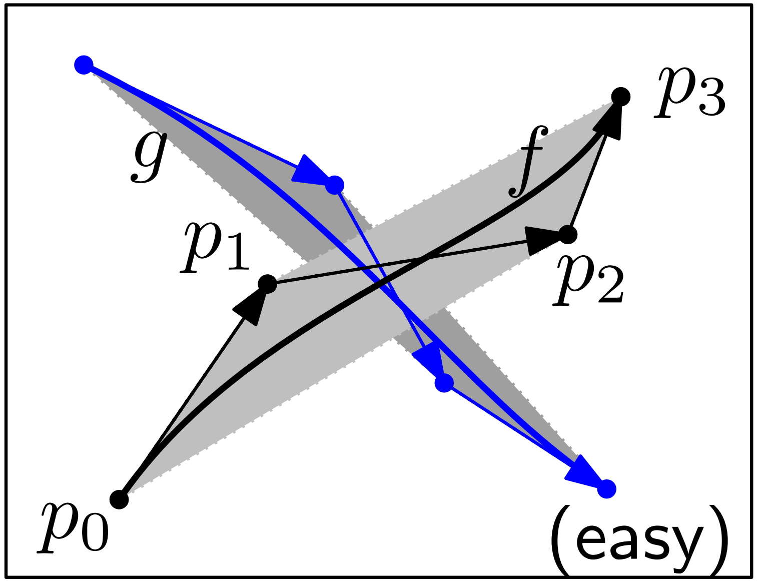
Freie Universität Berlin

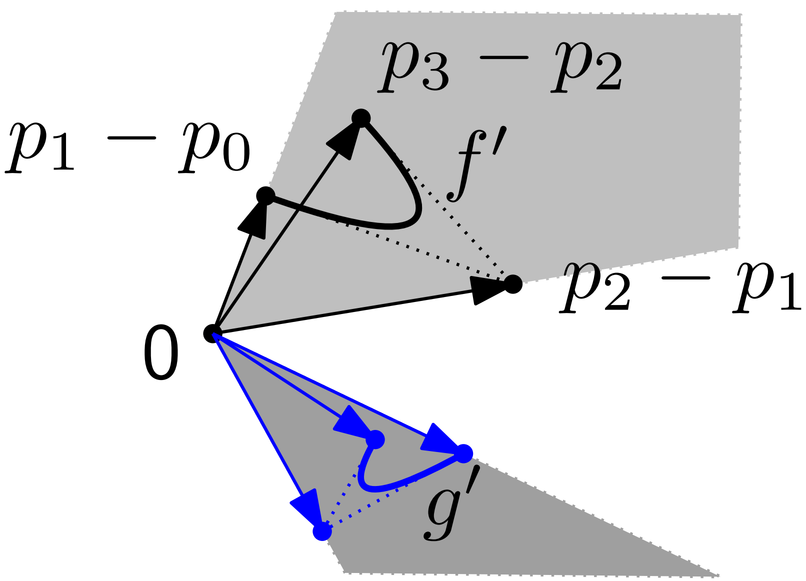


The derivative of a Bézier curve f is a Bézier curve f', of degree one less.

The control polygon of f' is formed from the differences $p_{i+1} - p_i$ of the original control polygon, times a constant factor.

Sufficient condition for unique crossing





If

- the convex hulls of f and g "cross" (the endpoints stick out),
- and the cones of directions of f' and g' are disjoint,

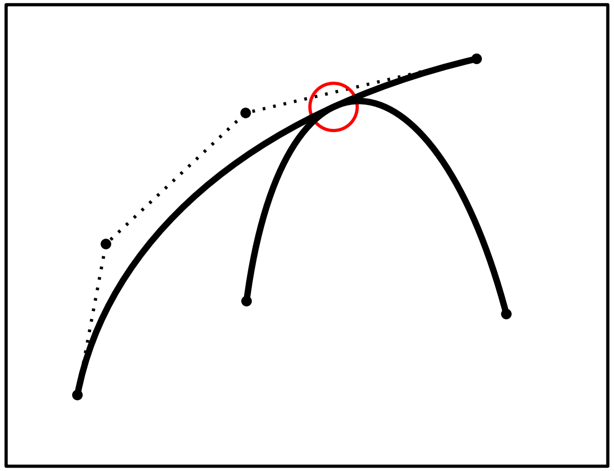
then f and g intersect in a single point.

Intersection algorithm by subdivision

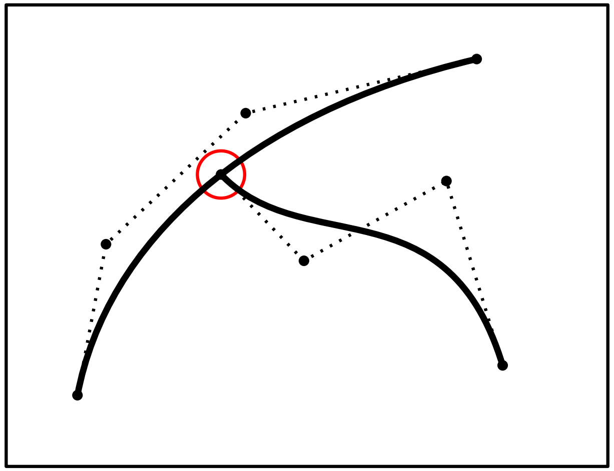
PUSH (f, g) on stack
while stack is not empty:
  POP (f, g) from stack
  if f, g are guaranteed to have no intersection:
    discard f, g
  elsif f, g are guaranteed to have a unique intersection
   and the precision of intersection is good enough:
    report intersection
  else:
    subdivide the larger curve (say, f) into f_1 and f_2.
    PUSH (f_1, g) on stack
    PUSH (f_2, g) on stack

Problems with termination

The subdivision algorithm
will *never* terminate



- when curves are *tangent*, or

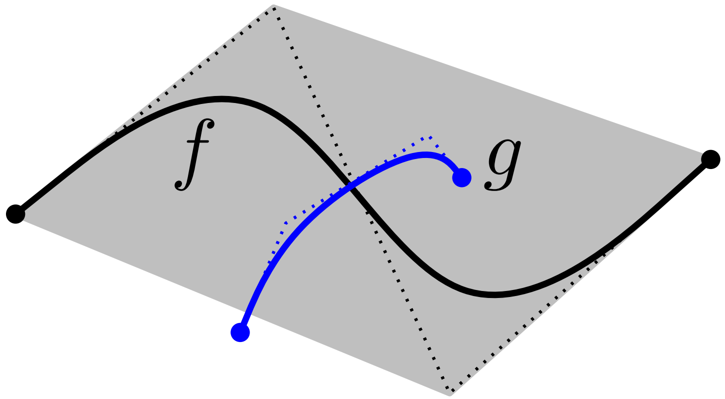


- when an endpoint lies *on* a curve.

(hard cases)

Problems with termination

The algorithm may also fail in easy cases:

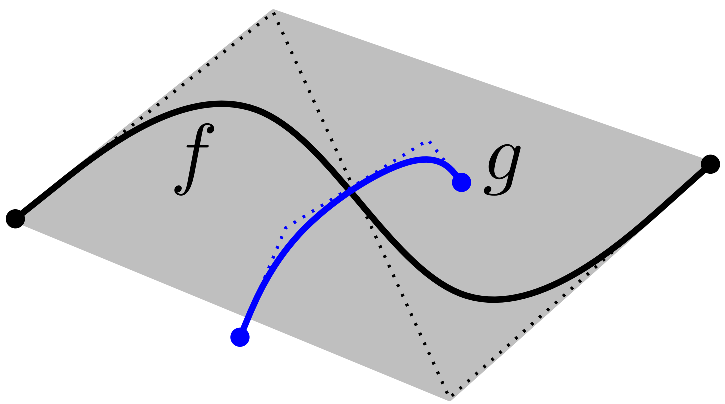


no decision.

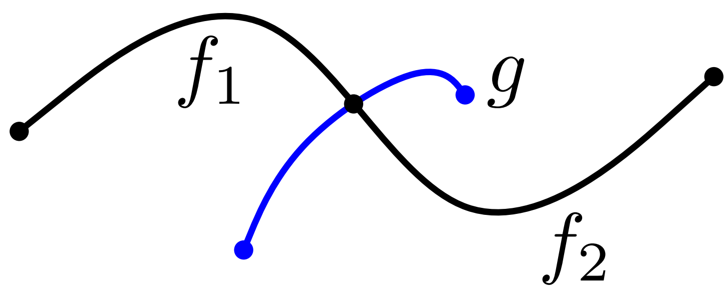
→ subdivide $f = f_1 + f_2$

Problems with termination

The algorithm may also fail in easy cases:



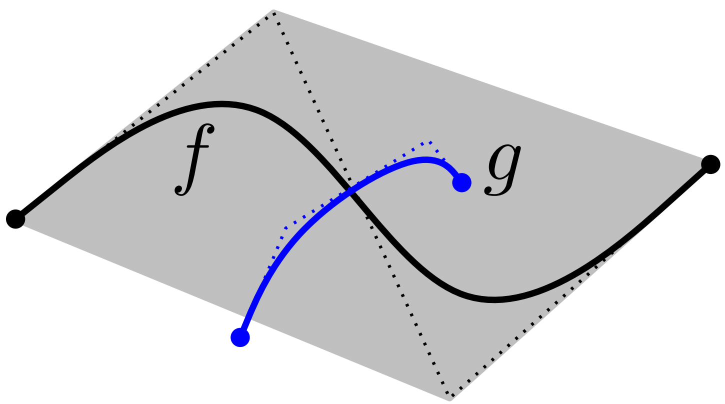
no decision.
$\rightarrow$ subdivide $f = f_1 + f_2$



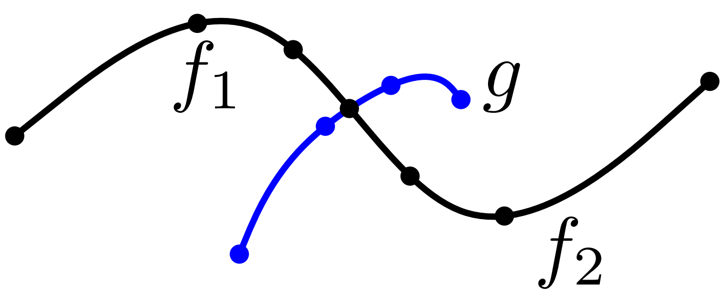
The subdivision point $f(1/2)$ happens to fall on g.
$\rightarrow$ infinite loop for (f_1, g) and for (f_2, g)

Problems with termination

The algorithm may also fail in easy cases:



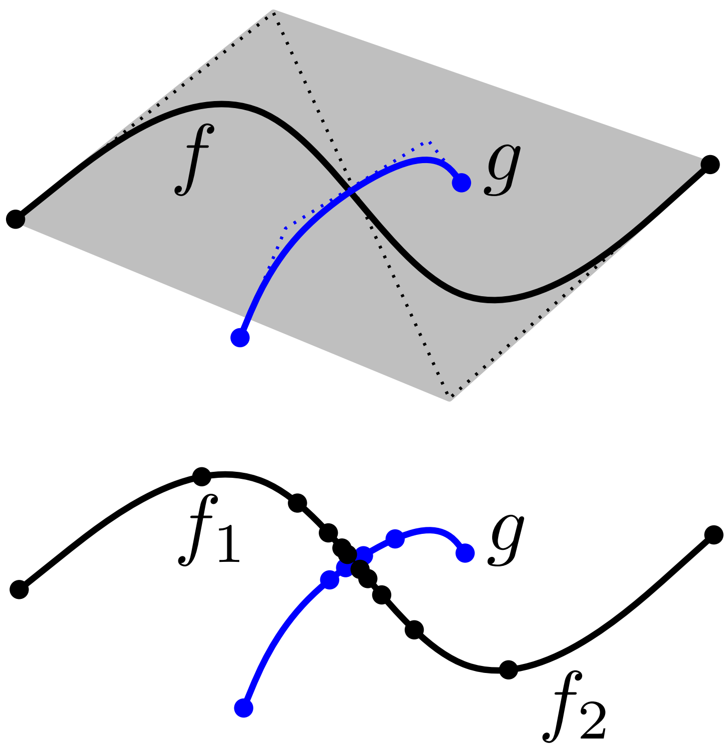
no decision.
$\rightarrow$ subdivide $f = f_1 + f_2$



The subdivision point $f(1/2)$ happens to fall on g.
$\rightarrow$ infinite loop for (f_1, g) and for (f_2, g)

Problems with termination

The algorithm may also fail in easy cases:



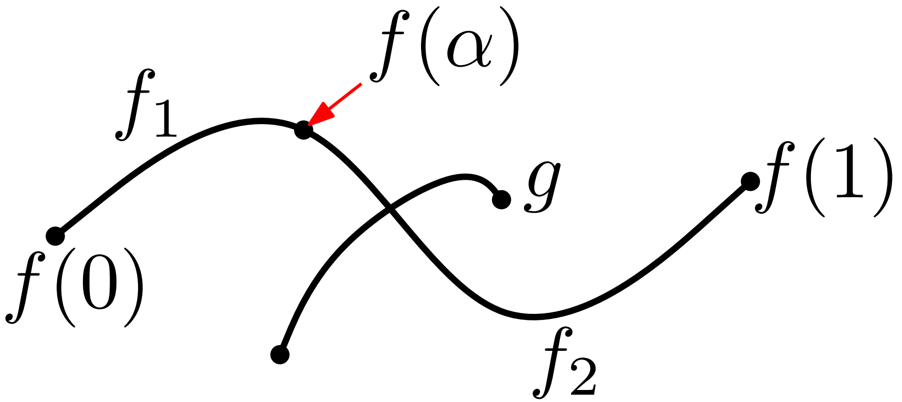
no decision.
$\rightarrow$ subdivide $f = f_1 + f_2$

The subdivision point $f(1/2)$ happens to fall on g.
$\rightarrow$ infinite loop for (f_1, g) and for (f_2, g)

Even it the algorithm terminates, it may make unnecessarily many subdivision steps.

Randomization

One possible solution:
Don't subdivide at $1/2$, but at a *random* point $0 < \alpha < 1$.



$\rightarrow$ The problematic case is avoided with high probability.

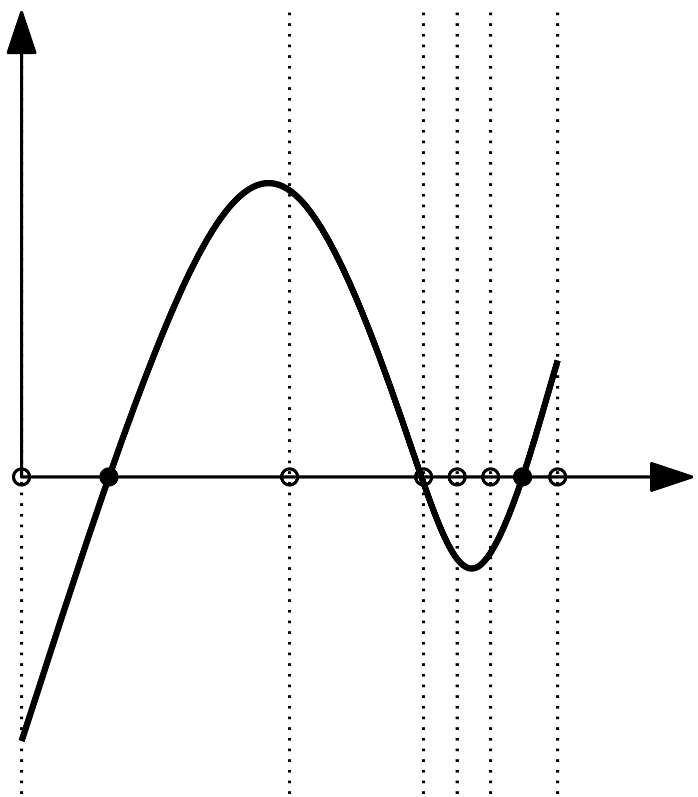
[A. Eigenwillig, L. Kettner, W. Krandick, K. Mehlhorn, S. Schmitt, N. Wolpert: A Descartes algorithm for polynomials with bit-stream coefficients (CASC 2005)]

Drawback:
Subdivision at points other than $1/2$ is costly in terms of bit complexity.

Zeros of a polynomial

Freie Universität Berlin

Eigenwillig, Kettner, Krandick, Mehlhorn, Schmitt, Wolpert (2005)
A Descartes algorithm for polynomials with bit-stream coefficients



Task: root isolation

Assumption: no multiple roots

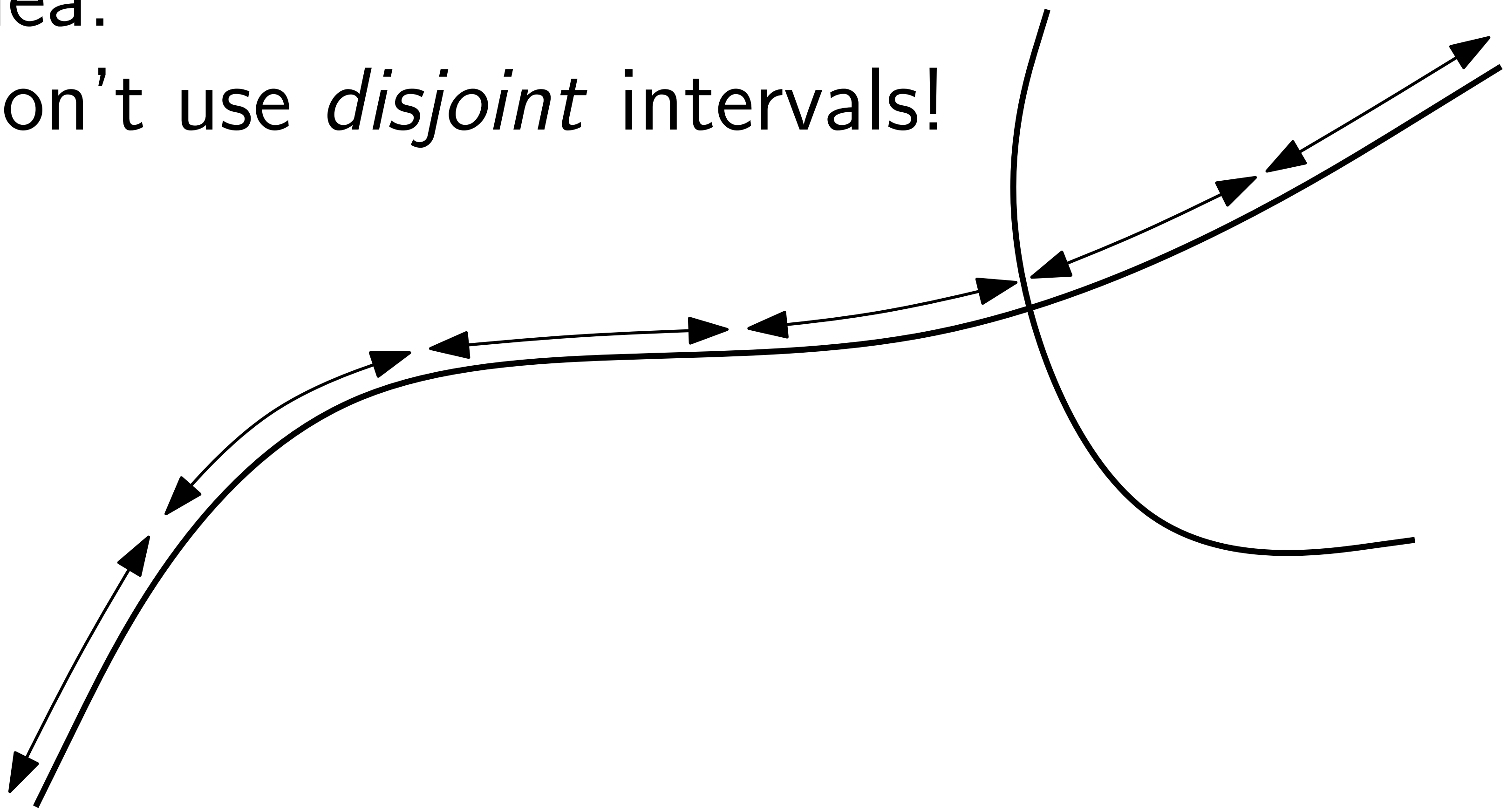
Descartes Rule of Signs may identify an interval as contain 0 roots or exactly 1 root.

If not, bisect and repeat.

SUPER-Composition

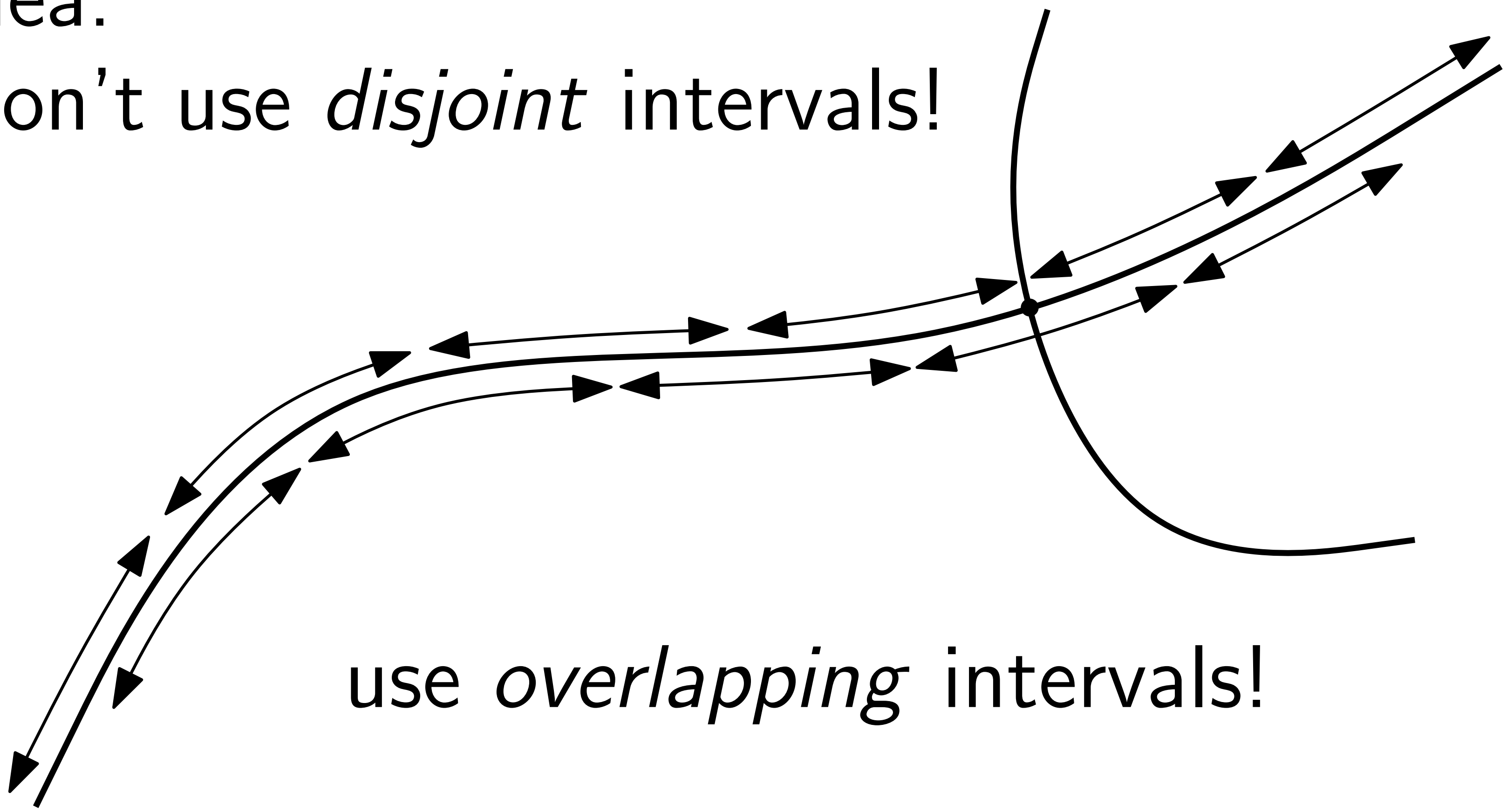
Idea:

Don't use *disjoint* intervals!



SUPER-Composition

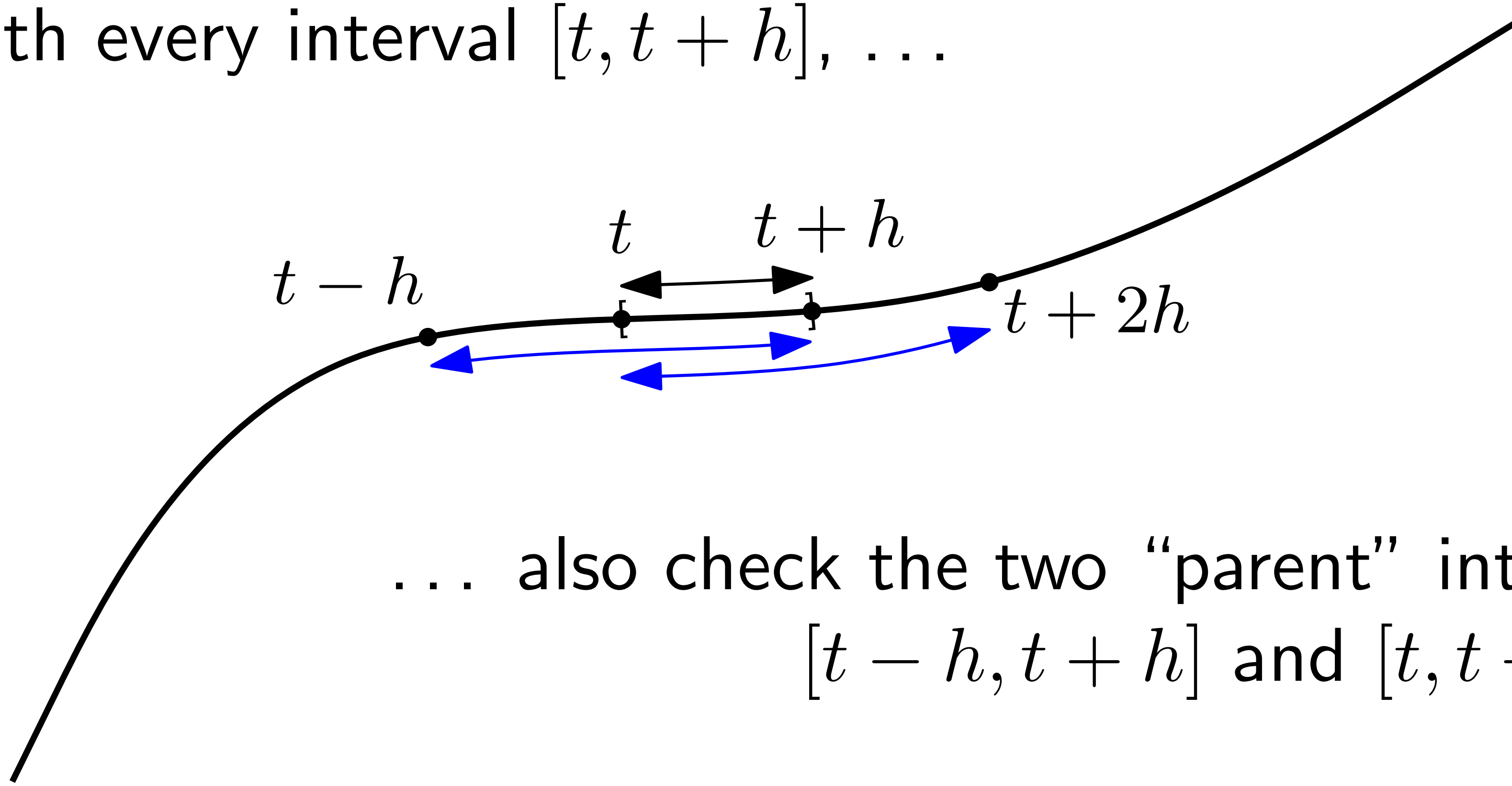
Idea:
Don't use *disjoint* intervals!



use *overlapping* intervals!

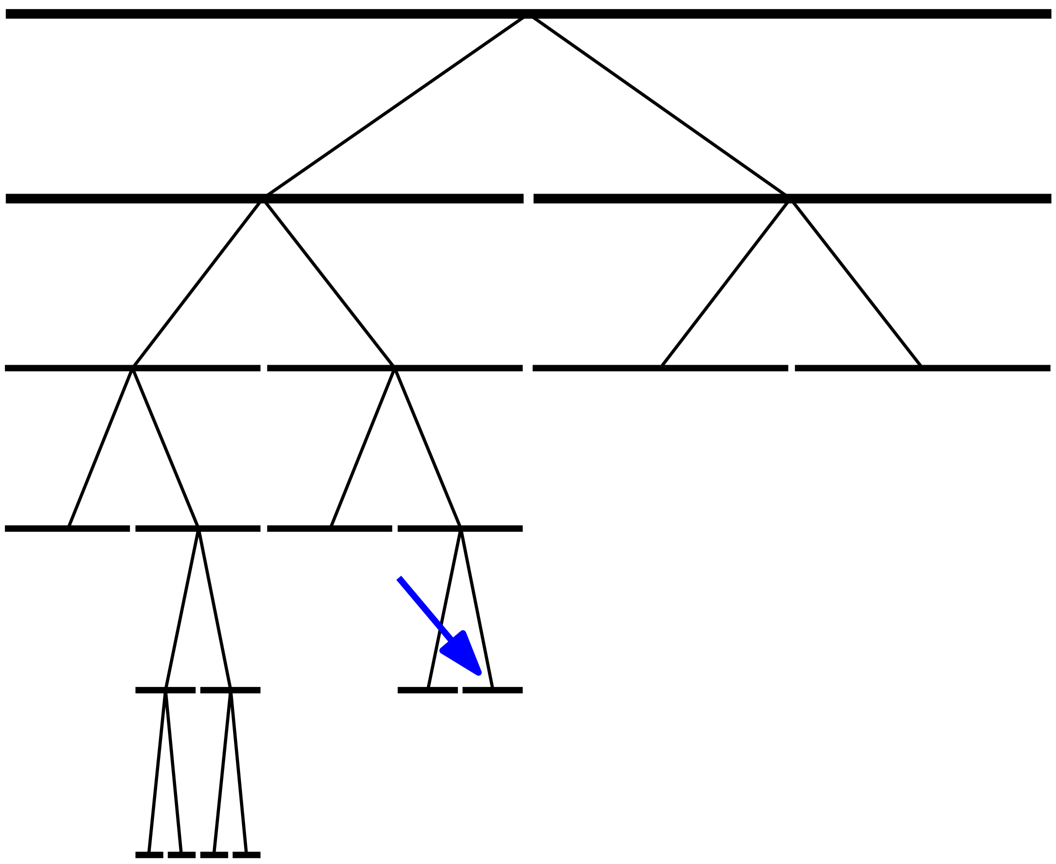
SUPER-Composition

With every interval $[t, t+h]$, . . .



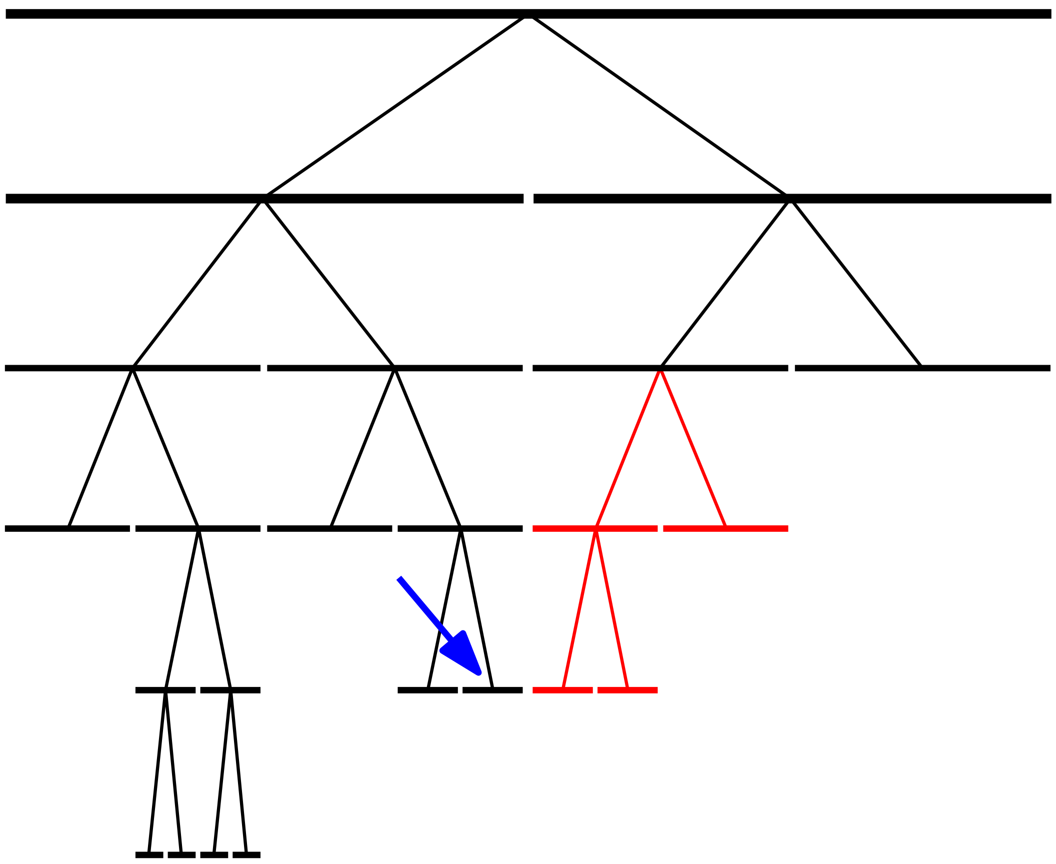
. . . also check the two “parent” intervals $[t-h, t+h]$ and $[t, t+2h]$.

SUPER-Composition



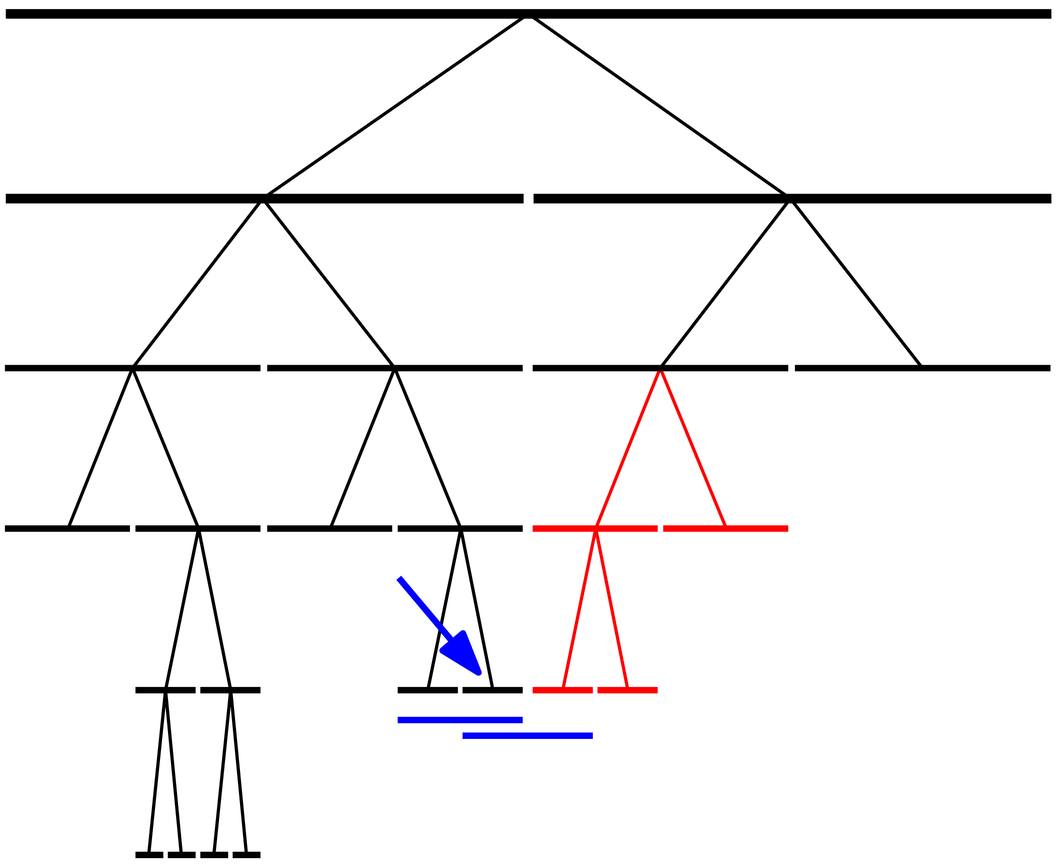
binary
decomposition tree

SUPER-Composition



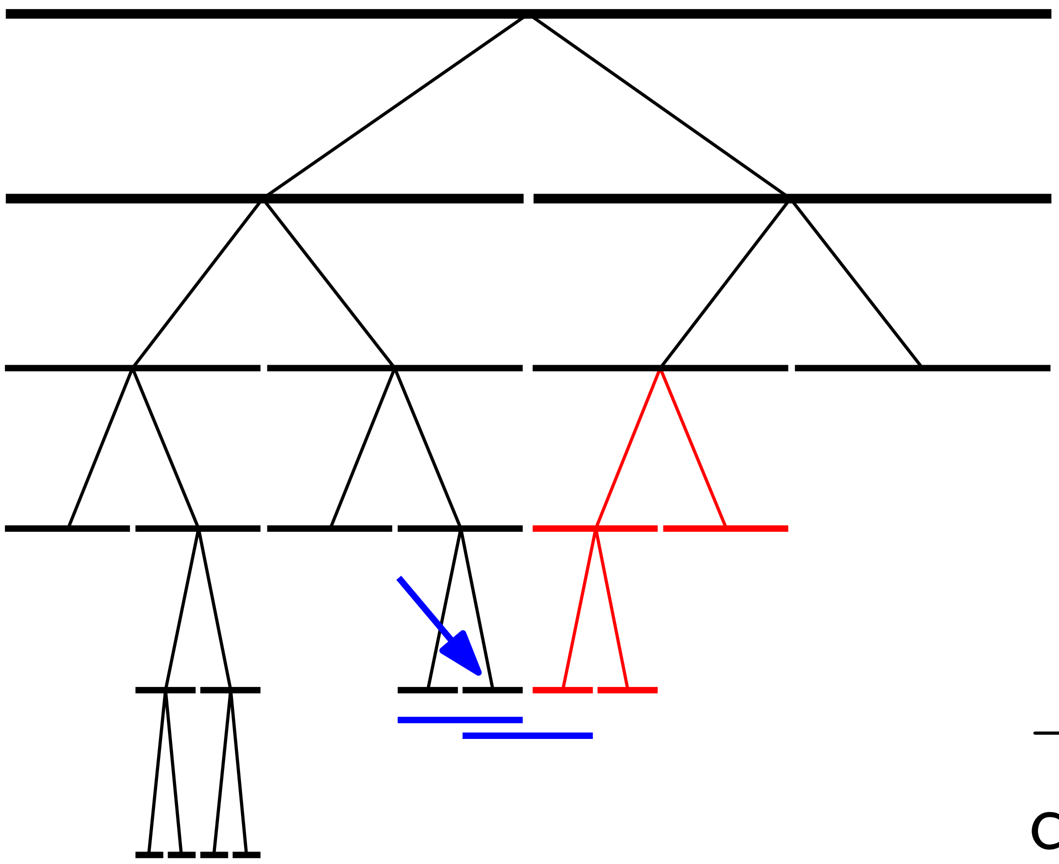
binary
decomposition tree

SUPER-Composition



binary decomposition tree

SUPER-Composition



binary decomposition tree

algorithm must cross subtree boundaries.

$\longrightarrow$

constant-factor overhead (cf. balanced quadtree)

Result

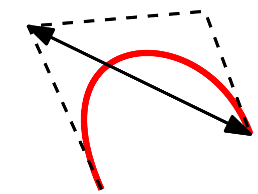
Theorem 1 *If*

- *the derivative of* f *and* g *is nowhere zero, and*
- *at every intersection point, the curves cross at a positive angle, and*
- *no endpoint of* f *or* g *lies on the other curve,*

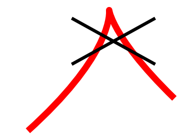
then the subdivision-supercomposition algorithm terminates.

Result

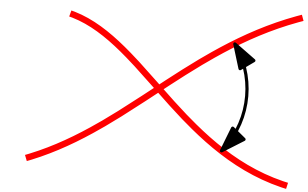
Theorem 2 • *If the diameter of the control polygon of f and g is at most D,*



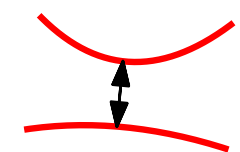
- $\|f'(t)\| \geq v_{\min}$ *and* $\|g'(t)\| \geq v_{\min}$ *everywhere,*



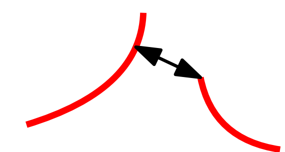
- *every intersection angle is at least α, and*



- *the distance between f and g is at least ε, at every local minimum and at every endpoint,*

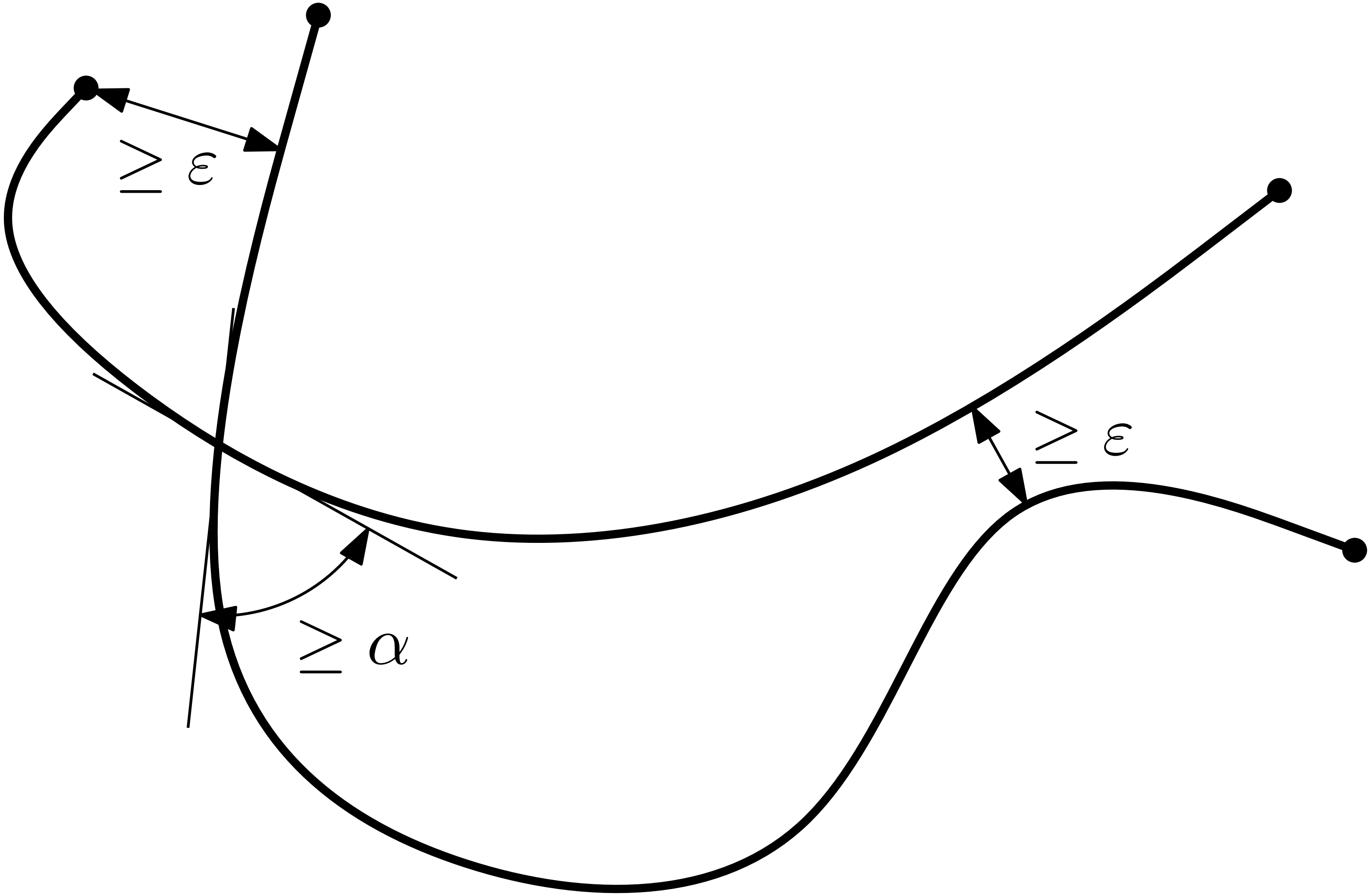


then the number of subdivision levels is at most



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

The conditions of the theorem



Proof of the theorem

Assumptions:
f and g are Bézier curves of degree d.
Initial parameter interval $= [0, 1]$.

diameter of control polygon at most D
$\Rightarrow$

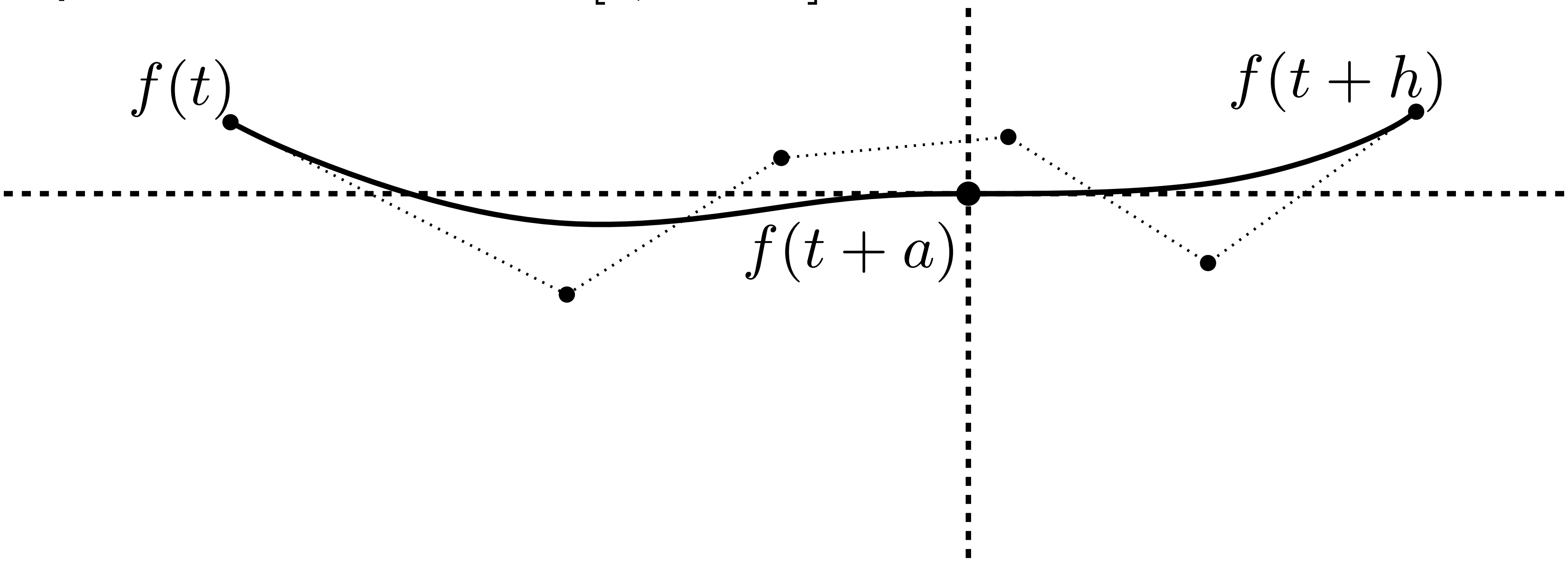
$$\|f'(t)\|, \|g'(t)\| \le 2dD$$

$$\|f''(t)\|, \|g''(t)\| \le S := 4d(d-1)D$$

$\|f'(t)\|, \|g'(t)\| \ge v_{\min}$
$\Rightarrow$ curvature of f and g is at least $S/v_{\min}$

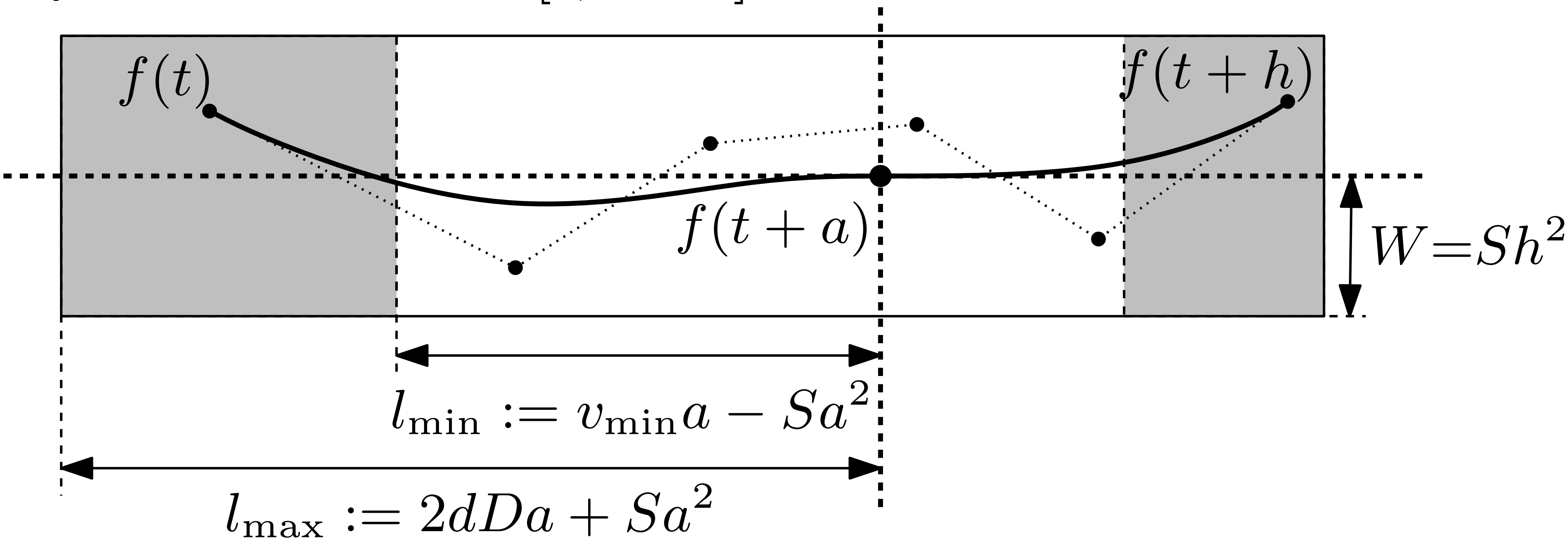
Proof of the theorem

parameter interval $[t, t+h]$



Proof of the theorem

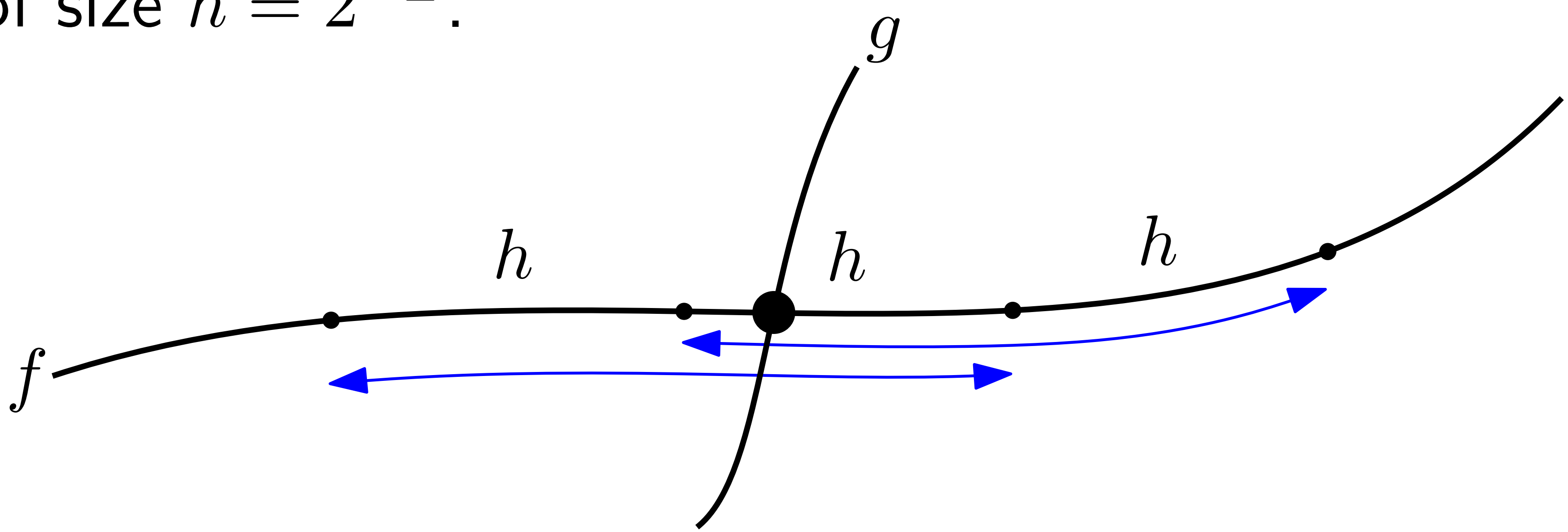
parameter interval $[t, t+h]$



The curve and the control polygon is contained in a rectangular strip of width $W = S \cdot h^2$ and length $\Theta(h)$.

Intersection points

Assume complete subdivision for L levels into intervals of size $h = 2^{-L}$.

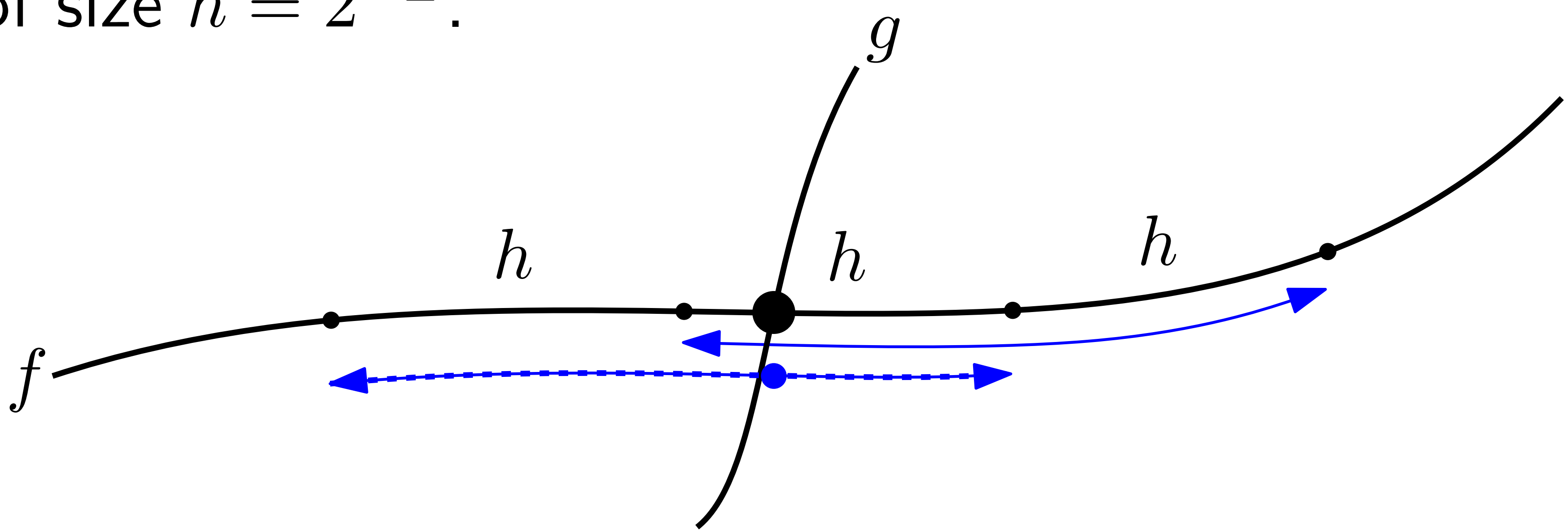


Then there is an interval where the intersection point is at least $h/2$ away from both ends.

Intersection points

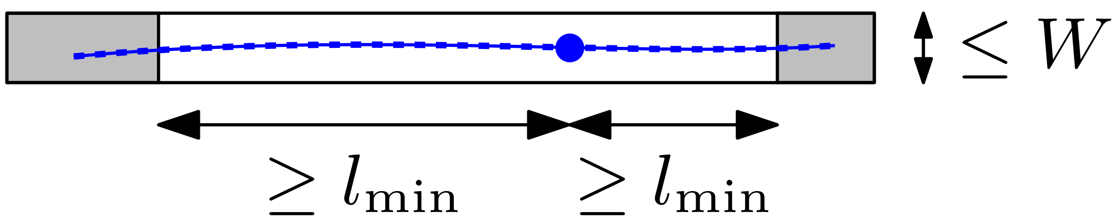
Freie Universität Berlin

Assume complete subdivision for L levels into intervals of size $h = 2^{-L}$.



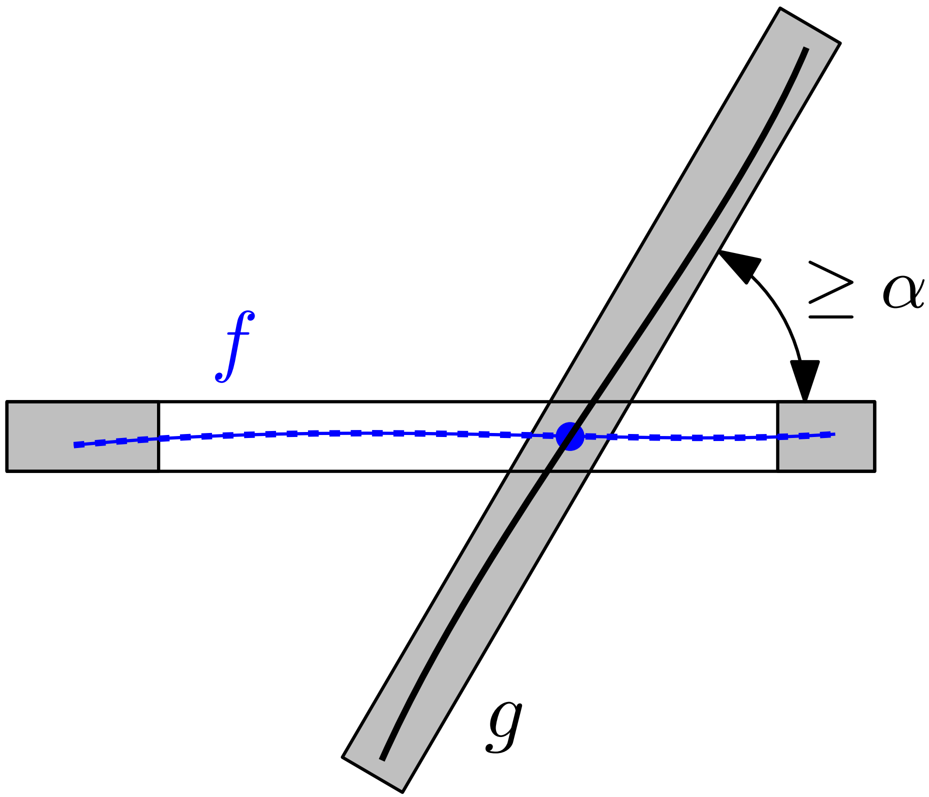
Then there is an interval where the intersection point is at least $h/2$ away from both ends.

Intersection points



Intersection points

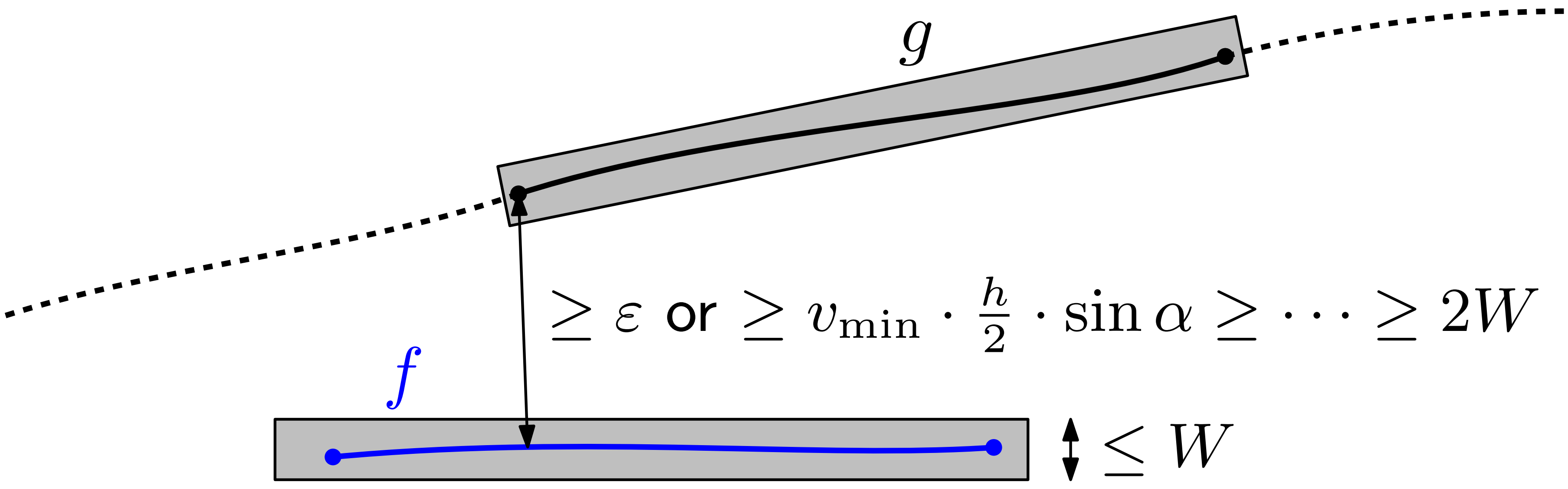
Freie Universität Berlin



$\Rightarrow$

The endpoints of f stick out of the control polygon of g.

No intersections



Runtime analysis

Freie Universität Berlin

Theorem 2 *The number of subdivision levels is at most*

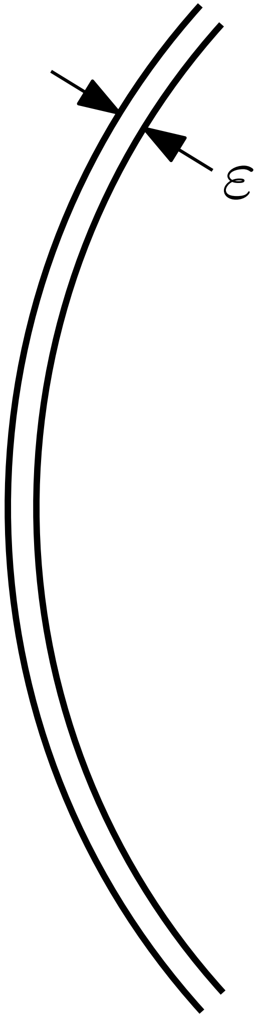
$$L := \max\left\{\log_2 \frac{D}{v_{\min}\cdot\alpha}, \frac{1}{2}\cdot\log_2\frac{D}{\varepsilon}\right\} + O(1).$$

Corollary 1 *The running time is at most*

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

Tightness of the bound

Freie Universität Berlin

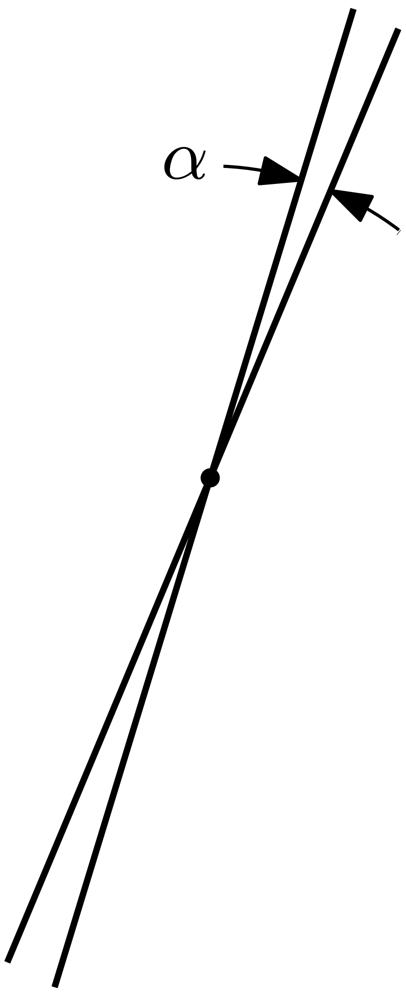


Each curve is subdivided into $O(\sqrt{1/\varepsilon})$ pieces.

Running time is $O(\sqrt{1/\varepsilon})$.

NOT $O(\sqrt{1/\varepsilon} \times \sqrt{1/\varepsilon})$, as given by the corollary!

Tightness of the bound



Each curve is subdivided into $O(\log \frac{1}{\alpha})$ pieces.

Running time is $O(\log \frac{1}{\alpha})^2$.

NOT $O(1/\alpha^2)$, as given by the corollary!

Future work — Open questions

- Better analysis of the runtime

Future work — Open questions

Freie Universität Berlin

- Better analysis of the runtime
- higher dimensions
  (e. g., intersecting a curve with a surface patch)

Future work — Open questions

Freie Universität Berlin

- Better analysis of the runtime
- higher dimensions
  (e. g., intersecting a curve with a surface patch)
- What is the right name
  for $\begin{cases} \text{Sturm-Habitch sequences?} \\ \text{Sturm-Habicht sequences?} \end{cases}$