Computing intersections

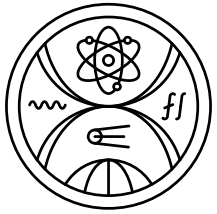
Róbert Bohdal

robert.bohdal@fmph.uniba.sk

https://flurry.dg.fmph.uniba.sk/webog/en/bohdal-teaching

Department of Algebra and Geometry
Faculty of Mathematics, Physics and Informatics
Comenius University Bratislava

Computer Graphics (1)
Lecture 7



Contents

1 Computing intersections

2 Segment–segment intersection in 2D

3 Point location relative to a polygon in 2D

4 Segment–polygon intersection in 2D

5 Segment–curve intersection in 2D

6 Curve–curve intersection

7 Segment–plane intersection in 3D

8 Segment–polygon intersection in 3D

9 Segment–polygon mesh intersection

10 Segment–implicit surface intersection



Computing intersections

- Computing object intersections, especially ray–polygon intersections in 3D, is a very common operation in computer graphics
- When rendering a 3D scene using the ray-tracing method, calculating the intersection takes up a significant amount of computating time
- To accelerate tests for intersection between two objects, a *bounding volume* is assigned to each object or part of an object
- Computing the intersection of an object with a simple bounding volume of another object is faster than computing its intersection with the original object
- Common bounding volumes include axis-aligned boxes, spheres (or ellipsoids), cylinders and capsules (a cylinder with hemispherical ends), or combinations of these when individual parts are bounded separately (e.g. a 3D character model)
- To find the intersection of two objects, we first test their bounding volumes for intersection
- If the bounding volumes are disjoint, the objects do not intersect
- Often (e.g. when objects are represented by polygon meshes), a complex object intersection problem can be reduced to finding the intersection of a line segment (or a ray) with a polygon

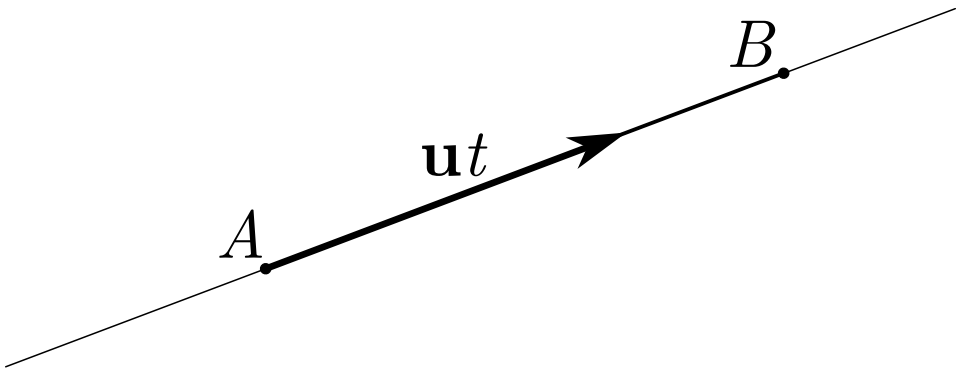


Parametric definition of a segment, line and ray in the plane

- A line segment with endpoints $A = (a_x, a_y)$, $B = (b_x, b_y)$, where $A \neq B$, is parametrised as $A + \mathbf{u} \cdot t$, where $\mathbf{u} = B - A$ and $t \in [0, 1]$:

$$x = a_x + u_x t$$
$$y = a_y + u_y t$$

- For a ray, $t \in [0, \infty)$; for a line, $t \in (-\infty, \infty)$





Segment–segment intersection in 2D – parallelism test

Consider the segment $A + \mathbf{u}s$ defined by the point $A = (a_x, a_y)$ and vector $\mathbf{u} = (u_x, u_y)$, and the segment $B + \mathbf{v}t$ defined by the point $B = (b_x, b_y)$ and vector $\mathbf{v} = (v_x, v_y)$:

$$\begin{aligned} x &= a_x + u_x s & x &= b_x + v_x t \\ y &= a_y + u_y s, \ s \in [0,1] & y &= b_y + v_y t, \ t \in [0,1] \end{aligned}$$

Denote their other endpoints by $A' = A + \mathbf{u}$ and $B' = B + \mathbf{v}$. Let $\mathbf{w}^\perp$ denote the vector perpendicular to $\mathbf{w} = (w_x, w_y)$ obtained by rotating $\mathbf{w}$ through 90°. For an anticlockwise rotation, $\mathbf{w}^\perp$ has coordinates $(-w_y, w_x)$

- The segments $A + \mathbf{u}s$ and $B + \mathbf{v}t$ are parallel $\Leftrightarrow$ $\mathbf{u}$ and $\mathbf{v}$ are parallel $\Leftrightarrow$ $\mathbf{u}$ and $\mathbf{v}^\perp$ are perpendicular $\Leftrightarrow$ $\mathbf{u} \cdot \mathbf{v}^\perp = 0$
  - the segments lie on the same line $\Leftrightarrow$ $B - A \parallel \mathbf{v} \Leftrightarrow (B - A) \cdot \mathbf{v}^\perp = 0$,
  - the segments lie on distinct lines $\Leftrightarrow$ $B - A \nparallel \mathbf{v} \Leftrightarrow (B - A) \cdot \mathbf{v}^\perp \neq 0$

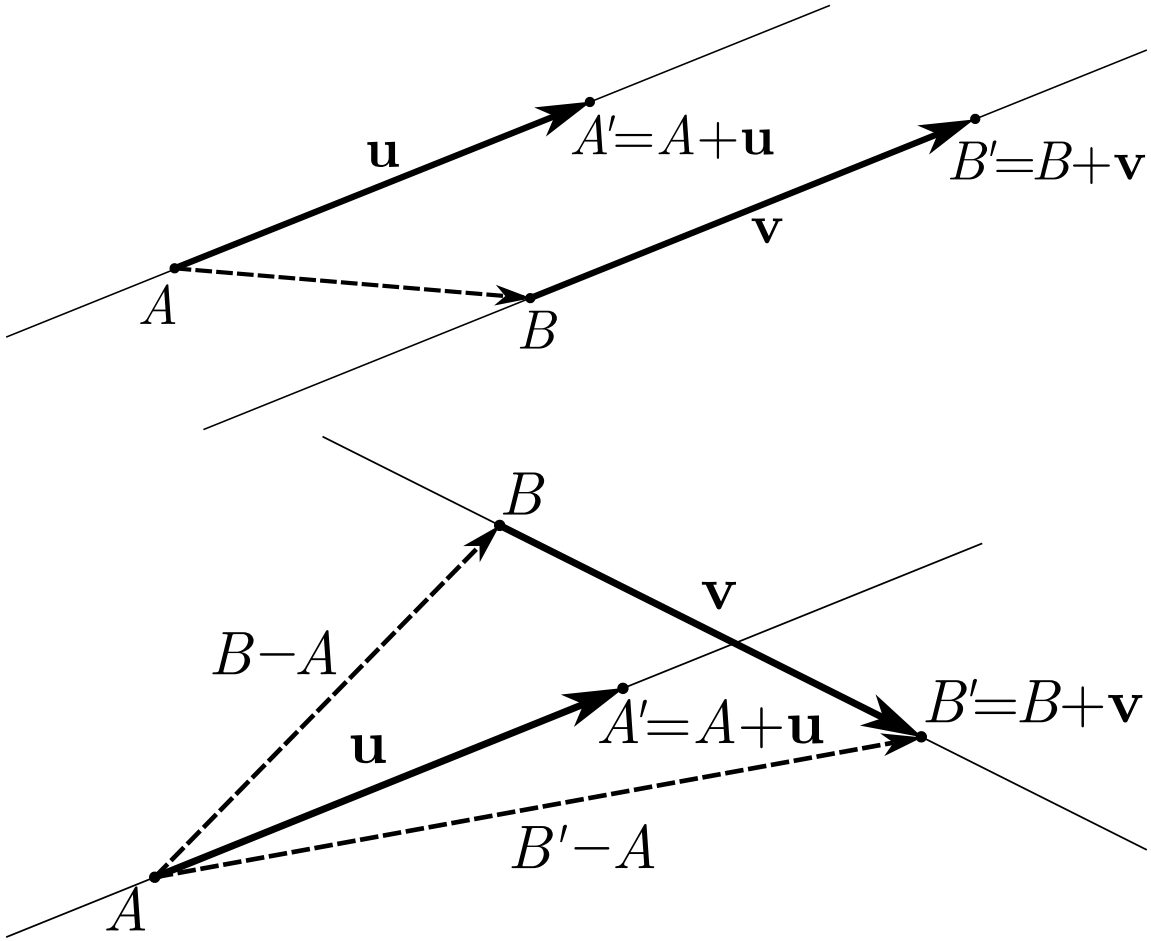
  If the segments lie on the same line, sort all four endpoints by the **appropriate** coordinate (x if $|u_x| \geq |u_y|$, otherwise y); their order determines whether the segments intersect. Segments on distinct parallel lines do not intersect

Note: The following tests for proper segment intersection exclude degenerate segments of zero length, collinear overlap, and cases where an endpoint of one segment lies on the other segment. These cases are handled separately



Segment–segment intersection in 2D

- The segments $A + \mathbf{u}s$ and $B + \mathbf{v}t$ are non-parallel $\Leftrightarrow \mathbf{u} \cdot \mathbf{v}^{\perp} \neq 0$
  - the segments **intersect** if the endpoints of each segment lie in **opposite** half-planes bounded by the line containing the other segment,
  - the segments **do not intersect** if the endpoints of one segment lie in **the same** half-plane bounded by the line containing the other segment



Non-parallel segment intersection in 2D – basis orientation test

- The points B and B' lie in opposite half-planes bounded by the line through $A + \mathbf{u}s$ if the bases $(\mathbf{u}, B - A)$ and $(\mathbf{u}, B' - A)$ (the order of the vectors matters!) have opposite orientations (a signed angle, clockwise or anticlockwise)

The following lemma determines the orientation of a basis:

Lemma

For any linearly independent vectors $\mathbf{u} = (u_x, u_y)$ *and* $\mathbf{v} = (v_x, v_y)$*, the basis* $(\mathbf{u}, \mathbf{v})$ *is positively (negatively) oriented if and only if:*

$$\begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix} > 0 \ (< 0)$$

A positively oriented basis has the same orientation as the basis formed by the standard unit vectors $e_1 = (1, 0)$, $e_2 = (0, 1)$

The proof of the lemma uses the cross product $\mathbf{u} \times \mathbf{v} = \left(0,\ 0,\ \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix}\right)$, identifying the planar vectors with the 3D vectors $(u_x, u_y, 0)$, $(v_x, v_y, 0)$



Non-parallel segment intersection in 2D – computation

If the bases $(\mathbf{u}, B - A)$ and $(\mathbf{u}, B' - A)$ have opposite orientations, and so do the bases $(\mathbf{v}, A - B)$ and $(\mathbf{v}, A' - B)$, then the segments $A + \mathbf{u}s$ and $B + \mathbf{v}t$ intersect

- To compute the intersection, it suffices to determine the unknown parameter s **or** t:

$$\begin{aligned} A + \mathbf{u}s &= B + \mathbf{v}t & \qquad A + \mathbf{u}s &= B + \mathbf{v}t \\ \mathbf{u}s - \mathbf{v}t &= B - A \quad | \cdot \mathbf{v}^\perp & \qquad \mathbf{u}s - \mathbf{v}t &= B - A \quad | \cdot \mathbf{u}^\perp \\ \mathbf{u} \cdot \mathbf{v}^\perp s &= (B - A) \cdot \mathbf{v}^\perp & \qquad -\mathbf{v} \cdot \mathbf{u}^\perp t &= (B - A) \cdot \mathbf{u}^\perp \\ s &= \frac{(B - A) \cdot \mathbf{v}^\perp}{\mathbf{u} \cdot \mathbf{v}^\perp} & \qquad t &= \frac{(B - A) \cdot \mathbf{u}^\perp}{-\mathbf{v} \cdot \mathbf{u}^\perp} = \frac{(B - A) \cdot \mathbf{u}^\perp}{\mathbf{u} \cdot \mathbf{v}^\perp} \end{aligned}$$

- The intersection point is obtained by substituting the computed parameter s or t into the corresponding segment expression $A + \mathbf{u}s$ or $B + \mathbf{v}t$



Segment–segment intersection in 2D – implicit equation test

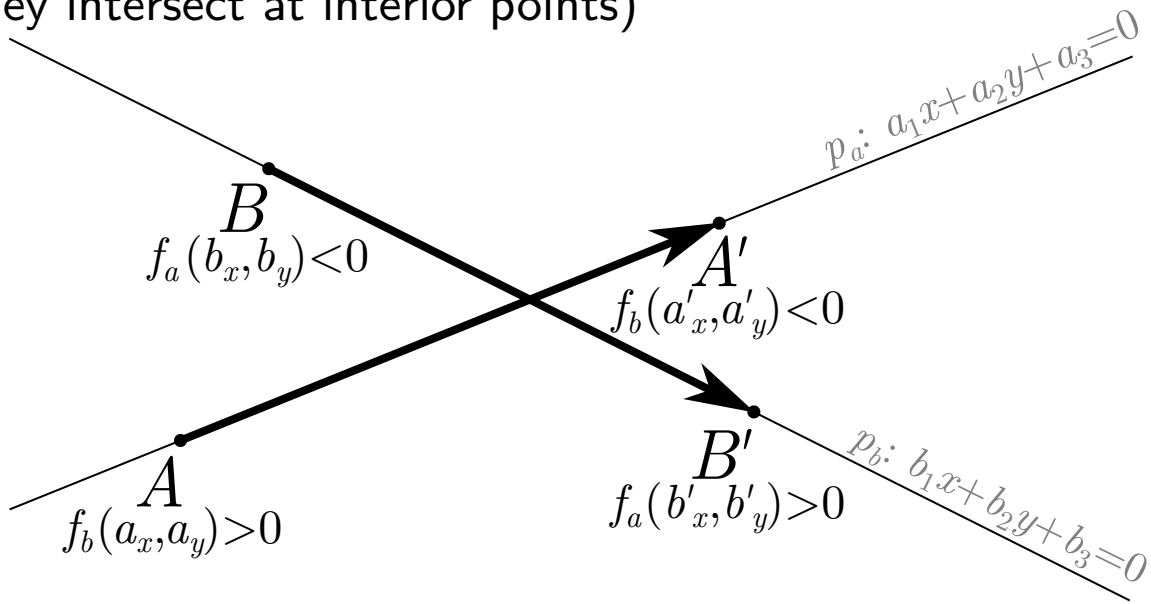
Let the line p_a containing the segment AA' have equation:

$$f_a(x, y) = 0, \text{ where } f_a(x, y) = a_1 x + a_2 y + a_3$$

and let the line p_b containing the segment BB' have equation:

$$f_b(x, y) = 0, \text{ where } f_b(x, y) = b_1 x + b_2 y + b_3$$

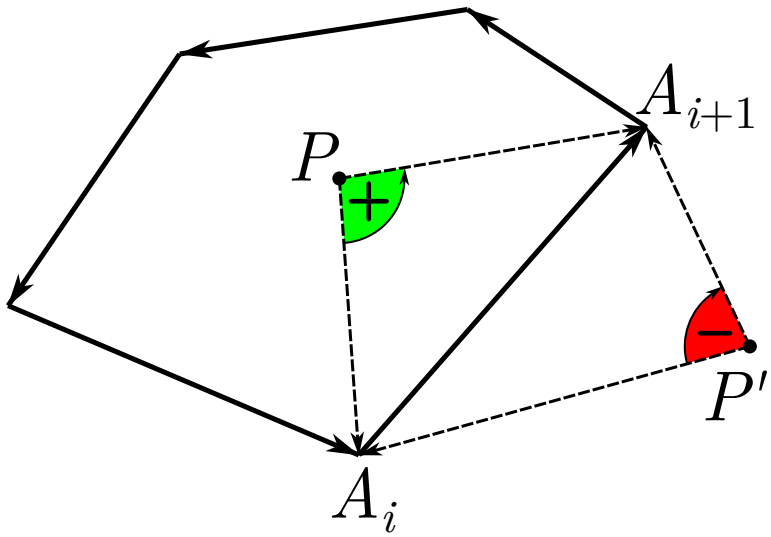
- The points B and B' lie in opposite half-planes bounded by the line containing AA' if $f_a(b_x, b_y) \cdot f_a(b'_x, b'_y) < 0$
- The points A and A' lie in opposite half-planes bounded by the line containing BB' if $f_b(a_x, a_y) \cdot f_b(a'_x, a'_y) < 0$
- If both inequalities **hold** simultaneously, the segments AA' and BB' **have** an inner intersection (they intersect at interior points)





Point location relative to a polygon in 2D

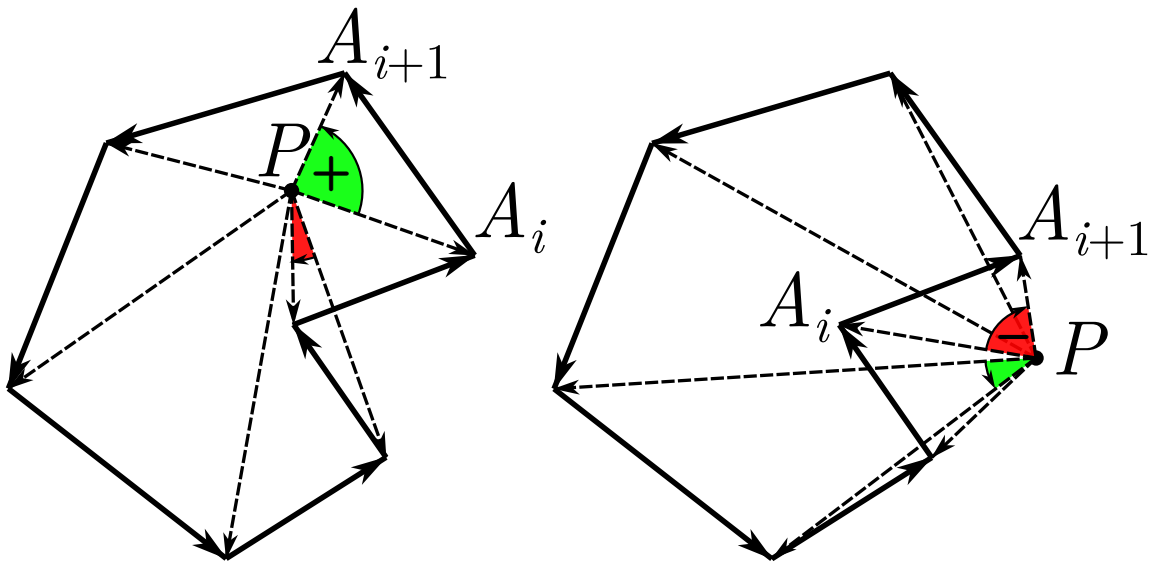
- Given a simple polygon $A_1 \ldots A_n$, determine whether a point P lies inside, outside or on its boundary. First check whether P lies on an edge of the polygon. Then, to decide whether P lies inside, use one of the following tests:
  - the test for a convex polygon,
  - the angle-sum test,
  - the ray-crossing parity test
- If the polygon $A_1 \ldots A_n$ is **convex** and **positively** oriented, it suffices to check that, for every edge A_iA_{i+1} (where $A_{n+1} \equiv A_1$), the basis $(A_i - P, A_{i+1} - P)$ is **positively** oriented. We can use the preceding lemma on the orientation of a vector basis





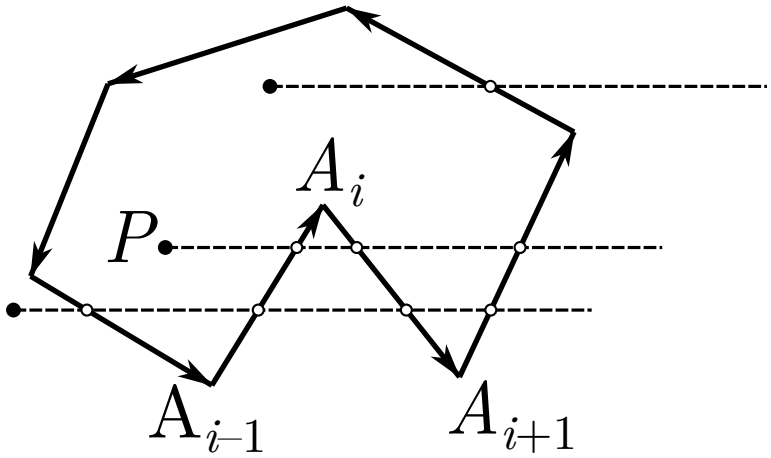
Angle-sum test

- Consider the sum of **signed** angles $\sum_{i=1}^{n} \angle A_i P A_{i+1}$ (where $A_{n+1} \equiv A_1$):
  - if the sum is **0°**, P lies **outside** the polygon,
  - if the sum is $\pm$**360°** ($\pm 2\pi$), P lies **inside** the polygon.
- The signed angle between vectors $\overrightarrow{PA_i}$ and $\overrightarrow{PA_{i+1}}$ is **positive** if we can move from the first vector to the second by rotating **anti-clockwise** through an angle $\in (0, \pi)$; it is **negative** if this rotation is **clockwise**. If both vectors have the same direction, the signed angle is zero
- Both the computation of angles (e.g. using `atan2(y,x)`) and their summation have limited numerical precision, so the resulting sum need not be exactly 0° or $\pm 360^\circ$
- In practice, this criterion is rarely used because of its high computational cost



Ray-crossing parity test

- Cast an arbitrary ray from P (usually horizontal or vertical) that intersects the polygon boundary at **finitely** many points:
  - if the number of intersections is **even** (including 0), P lies **outside** the polygon,
  - if the number of intersections is **odd**, P lies **inside** the polygon.

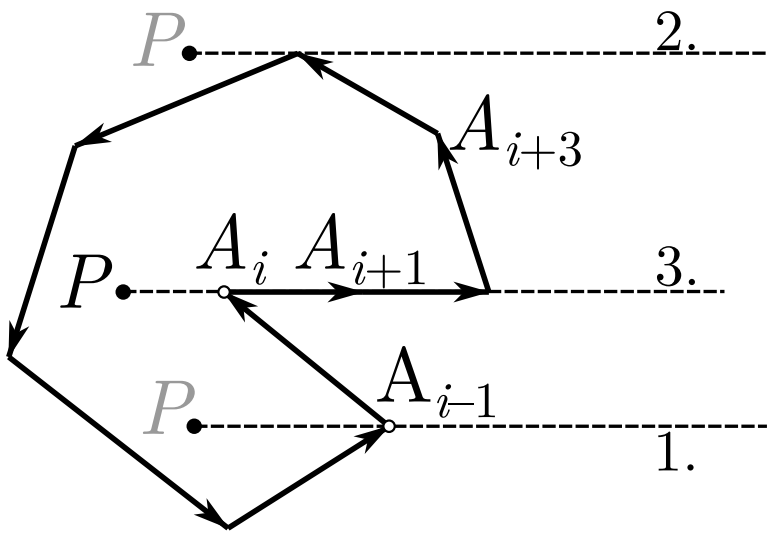


Note: Special cases where the ray passes through a vertex or along an edge of the polygon are addressed on the next slide



Ray-crossing parity test – special cases

- Cases where the ray passes through a polygon vertex A_i require special handling:
  1. If the vertices A_{i-1} (previous) and A_{i+1} (next) lie in **opposite** half-planes bounded by the line PA_i, count the intersection A_i **once**
  2. If the vertices A_{i-1} and A_{i+1} lie in **the same** half-plane bounded by the line PA_i, **do not count** A_i as an intersection
  3. If A_{i-1} or A_{i+1} also lies on the ray (i.e. a polygon **edge lies** on the ray), find the nearest preceding and following vertices that do not lie on the line.
     Use these two vertices to decide whether to count A_i as an intersection (we will use an analogy with the previous two rules).
     Do not repeat the test for the other vertices in this sequence that lie on the ray





Segment–polygon intersection in 2D

- To find only the intersection of a segment with the polygon boundary (a closed poly-line), repeatedly solve the segment–segment intersection problem for the given segment and each edge of the polygon
- To find the parts of the segment that lie inside the polygon, additional steps are needed: sort the intersections by the parameter t and alternate between entering and leaving. We must also determine the **position** (outside/inside/on the boundary) of the segment's **initial** point relative to the polygon

Note: Toggle the "inside/outside" state only when crossing the polygon boundary, not when merely touching it



Segment–**convex polygon** intersection in 2D

- A convex polygon $A_1 \dots A_n$ is the intersection of the closed half-planes defined by its edges. We therefore intersect the segment AA' with each defining **half-plane** of the polygon
- Relative to the **directed** line through the directed segment AA', classify each edge A_iA_{i+1} that is not parallel to this line as:
  - **entering**, if crossing its line **enters** the corresponding half-plane of the polygon
  - **leaving**, if crossing its line **leaves** the corresponding half-plane of the polygon
- For each line through an edge A_iA_{i+1} that is not parallel to AA', compute the parameter t by substituting the parametric expression $A + (A' - A)t$ (the line through AA') into the equation of the edge's line
- If the line through an edge A_iA_{i+1} is parallel to AA', then:
  - the segment does not intersect the polygon if it lies entirely outside the half-plane bounded by the line A_iA_{i+1}
  - do not compute t (the edge does not clip the segment) if the segment lies entirely in the closed half-plane bounded by the line A_iA_{i+1}
- Select the **largest** intersection parameter for the lines of **entering** edges (last entry), and the **smallest** for the lines of **leaving** edges (first exit)
- For the segment AA', set t_{in} to the maximum of 0 and the last-entry parameter, and t_{out} to the minimum of 1 and the first-exit parameter
- If $t_{\text{in}} > t_{\text{out}}$, the segment does not intersect the polygon. Otherwise, the intersection is the part of the segment with parameters $t \in [t_{\text{in}}, t_{\text{out}}]$



Segment–convex polygon intersection algorithm in 2D

$t_{\text{in}} := 0, \quad t_{\text{out}} := 1$ // initialise for a segment

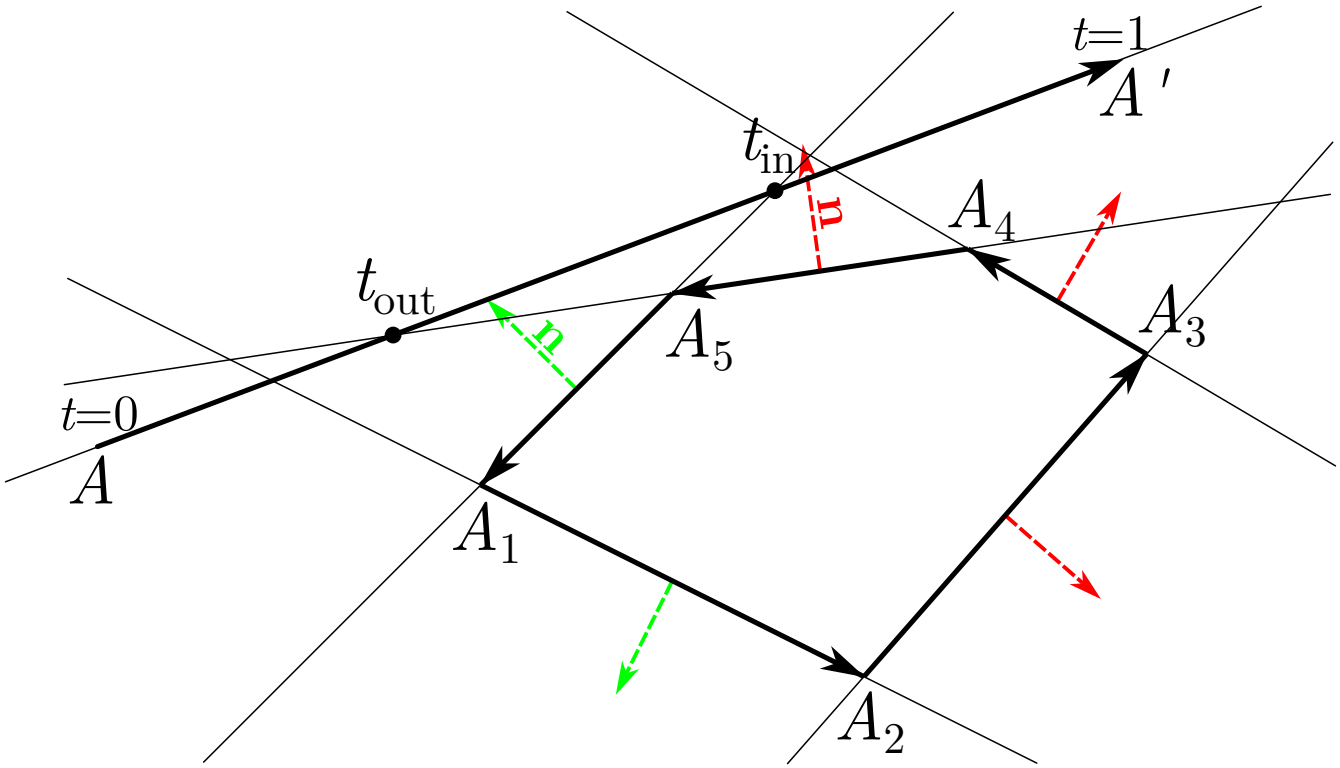
for each edge A_iA_{i+1} of the polygon:
  construct the line p through the edge A_iA_{i+1}
  if p is parallel to the line AA':
    if AA' lies outside the corresponding closed half-plane:
      the intersection is **empty**; terminate the algorithm
    continue with the next edge // this edge does not clip
  compute the parameter t of the intersection of the lines AA' and p
  if the edge is entering:
    $t_{\text{in}} := \max\{t_{\text{in}}, t\}$ // last entry
  else:
    $t_{\text{out}} := \min\{t_{\text{out}}, t\}$ // first exit

if $t_{\text{in}} > t_{\text{out}}$:
  the intersection is empty
else:
  $B := A + (A' - A)t_{\text{in}}$
  $B' := A + (A' - A)t_{\text{out}}$
  the intersection is the segment BB'

Note: An analogous procedure applies to a convex polyhedron in 3D



Segment–convex polygon intersection in 2D – example



The segment AA' enters the polygon's defining half-planes across the lines containing the edges A_1A_2 and A_5A_1. The intersection with the line through A_5A_1 has the larger parameter t, giving t_{in}. The other three lines are leaving lines: AA' exits their corresponding half-planes. The intersection with the line containing A_4A_5 has the smallest parameter $t = t_{\text{out}}$. Since $t_{\text{in}} > t_{\text{out}}$, the segment does not intersect the polygon.



Segment–curve intersection in 2D

- We want to find the intersection of the parametrically defined segment $A + \mathbf{u}t = (a_x + u_x t, a_y + u_y t)$, $t \in [0, 1]$ and the curve $\mathcal{K}$ given implicitly by $f(x, y) = 0$
- Substitute the parametric expression for the segment into the curve equation in place of x and y to obtain an equation for t:

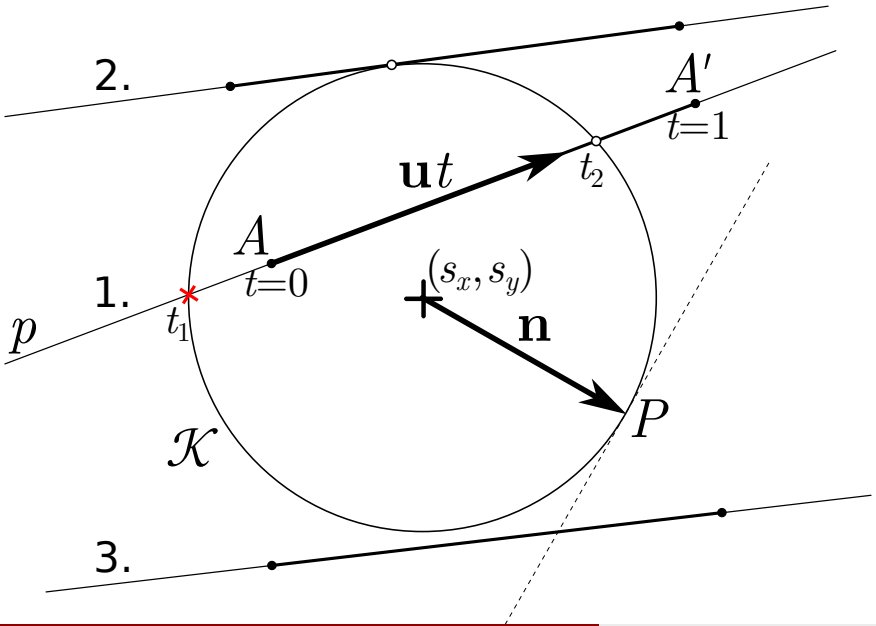
$$g(t) = f(a_x + u_x t, a_y + u_y t) = 0$$

  where $t \in [0, 1]$ is required; otherwise, the intersection point does not lie on the segment
- The choice of method for finding roots of $g(t) = 0$ depends on the type of function $f(x, y)$ defining the curve



Segment–conic intersection

- Let a **conic** $\mathcal{K}$ be given implicitly by $f(x, y) = 0$. Substituting the parametric expression for the segment into the equation of $\mathcal{K}$ yields a polynomial equation of degree at most two. If the coefficient of t^2 is non-zero, solve the quadratic equation $g(t) = at^2 + bt + c = 0$. Its roots are $t_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$, where $D = b^2 - 4ac$
- For $a \neq 0$, the discriminant D determines whether the **line** p through the segment $A + \mathbf{u}t$ intersects the conic $\mathcal{K}$:
  1. if $D > 0$, the line intersects the curve at two points, $t_1 \neq t_2$
  2. if $D = 0$, the line touches the curve at one point, $t_1 = t_2$
  3. if $D < 0$, the line does not intersect the curve, $\nexists\, t_1, t_2$
- After computing t_1,t_2 from $g(t) = 0$, test which values lie in $[0, 1]$ to determine the number of intersections of the segment with $\mathcal{K}$



Note:
If $a = 0$, solve the linear equation $bt + c = 0$
If also $b = 0$, there is no intersection, since for a **non-degenerate** conic we have $c \neq 0$ in this case



Segment–conic intersection – example

Consider a circle of radius r with centre (s_x, s_y), given by the implicit equation:

$$(x - s_x)^2 + (y - s_y)^2 - r^2 = 0$$

To find the intersection, substitute the parametric expression for the segment into the circle equation:

$$((a_x + u_x t) - s_x)^2 + ((a_y + u_y t) - s_y)^2 - r^2 = 0$$

Expanding gives a quadratic equation in t with discriminant D. There are three cases:

- $D < 0$: the line containing the segment has no point in common with the circle
- $D = 0$: the line containing the segment is tangent to the circle. The equation for t has a single solution t_0. If $t_0 \in [0, 1]$, the point of tangency $A + \mathbf{u}t_0$ is the only point common to the circle and the segment
- $D > 0$: the line containing the segment is a secant of the circle. The equation for t has two real solutions, t_1 and t_2. For each t_i satisfying $t_i \in [0, 1]$, the point $A + \mathbf{u}t_i$ lies on both the circle and the segment



Segment–conic intersection – normal at the intersection point

- In addition to finding the intersection point, we often need the direction of the normal vector $\mathbf{n}$ to the curve $f(x, y) = 0$ at that point, i.e. a vector perpendicular to the tangent at the intersection
- The normal to the circle $(x - s_x)^2 + (y - s_y)^2 - r^2 = 0$ at $P = (p_x, p_y)$ passes through its centre $\Rightarrow \mathbf{n} = (p_x - s_x, p_y - s_y)$. For a general implicit curve, use the following lemma:

Lemma

Let a curve be defined by $f(x, y) = 0$ and let $P = (p_x, p_y)$ lie on it, i.e. $f(p_x, p_y) = 0$. Suppose f is continuously differentiable in a neighbourhood of P. If $\frac{\partial f}{\partial x}(p_x, p_y) \neq 0$ or $\frac{\partial f}{\partial y}(p_x, p_y) \neq 0$, then a normal vector to the curve at P is

$$\mathbf{n} = \left(\frac{\partial f}{\partial x}(p_x, p_y), \frac{\partial f}{\partial y}(p_x, p_y) \right)$$



Segment–differentiable curve intersection – Newton's method

- If the curve $\mathcal{K}$ is **not** defined by a **polynomial** equation $f(x, y) = 0$, then substituting the parametric expression for the segment generally yields a non-polynomial function $g(t) = f(a_x + u_x t, a_y + u_y t)$
- If $g(t)$ is differentiable and its derivative is non-zero at every iterate t_i, we can use *Newton's method* to find a root of $g(t) = 0$:
  1. choose an initial estimate of a root of $g(t) = 0$ and denote it by t_0,
  2. if t_i approximates the root after iteration i, compute the next approximation using:

$$t_{i+1} = t_i - \frac{g(t_i)}{g'(t_i)},$$

  3. stop when $|g(t_{i+1})| < \varepsilon_f$, or when the maximum number of iterations is reached
- This stopping criterion is often combined with a test of whether the step size $|t_{i+1} - t_i|$ is smaller than a prescribed tolerance ε_t. The solution corresponds to an intersection with the segment only if $t \in [0, 1]$

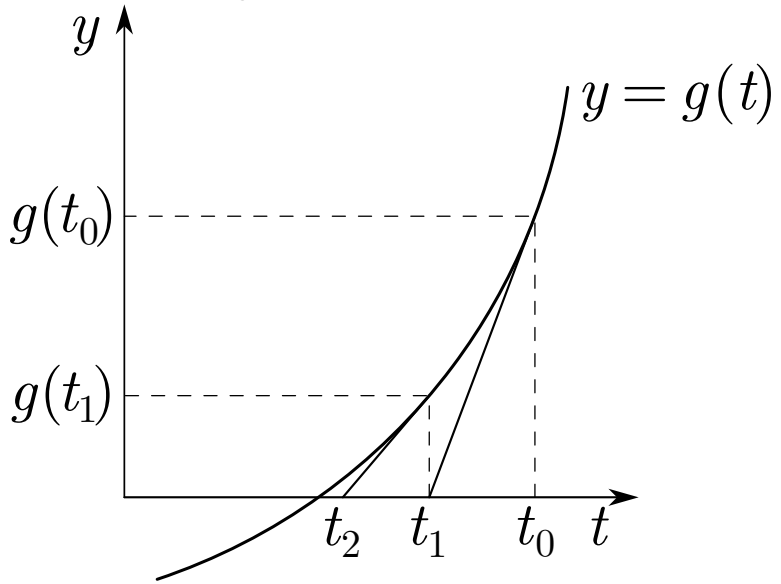


Segment–differentiable curve intersection – Newton's method

- From the approximation t_i, compute the next approximation t_{i+1} as the coordinate where the **tangent** to the graph of $g(t)$ at $(t_i, g(t_i))$ meets the t-axis. The tangent has slope $g'(t_i)$ and passes through $(t_i, g(t_i))$. Near t_i, we can approximate $g(t)$ by $g(t) \approx g(t_i) + g'(t_i)(t - t_i)$.
  The intersection coordinate t_{i+1} is then obtained from:

$$0 = g(t_i) + g'(t_i)(t_{i+1} - t_i) \Rightarrow t_{i+1} = t_i - \frac{g(t_i)}{g'(t_i)}$$

- A drawback is sensitivity to the initial estimate; a poor choice may cause the method to fail to converge
- Finding all roots is another difficulty



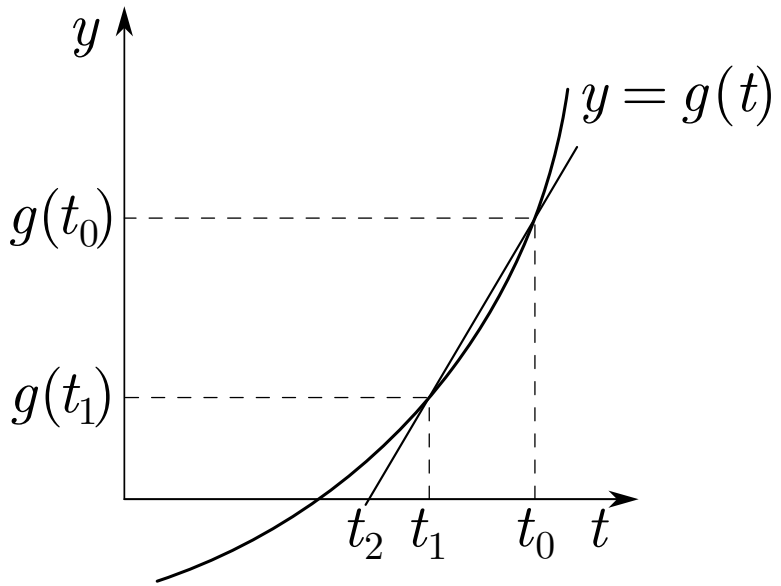


Segment–curve intersection – secant method

- From the approximations t_{i-1} and t_i, compute t_{i+1} as the coordinate where the **secant** through $(t_i, g(t_i))$ and $(t_{i-1}, g(t_{i-1}))$ meets the t-axis. Its slope is $m_i = \frac{g(t_i)-g(t_{i-1})}{t_i-t_{i-1}}$, and its equation is $y = g(t_i) + m_i(t - t_i)$.
  The intersection coordinate t_{i+1} is then obtained from:

$$0 = g(t_i) + m_i(t_{i+1} - t_i) \Rightarrow t_{i+1} = t_i - \frac{g(t_i)}{m_i} = t_i - g(t_i)\frac{t_i - t_{i-1}}{g(t_i) - g(t_{i-1})}$$

- The method requires two initial estimates and no derivative evaluations. We assume $t_i \neq t_{i-1}$ and $g(t_i) \neq g(t_{i-1})$



Segment–polynomial curve intersection – Bézier clipping

- If the curve $\mathcal{K}$ **is** defined by a **polynomial** equation $f(x, y) = 0$, substituting a general point of the segment yields a polynomial function $g(t) = f(a_x + u_x t, a_y + u_y t)$
- If the degree of $g(t)$ exceeds 2, numerical methods are used to find the roots of $g(t) = 0$
- A fast, numerically stable method called *Bézier clipping* finds all roots of the polynomial equation $g(t) = 0$ on $[0, 1]$ by expressing the graph of $g(t)$ in Bézier form



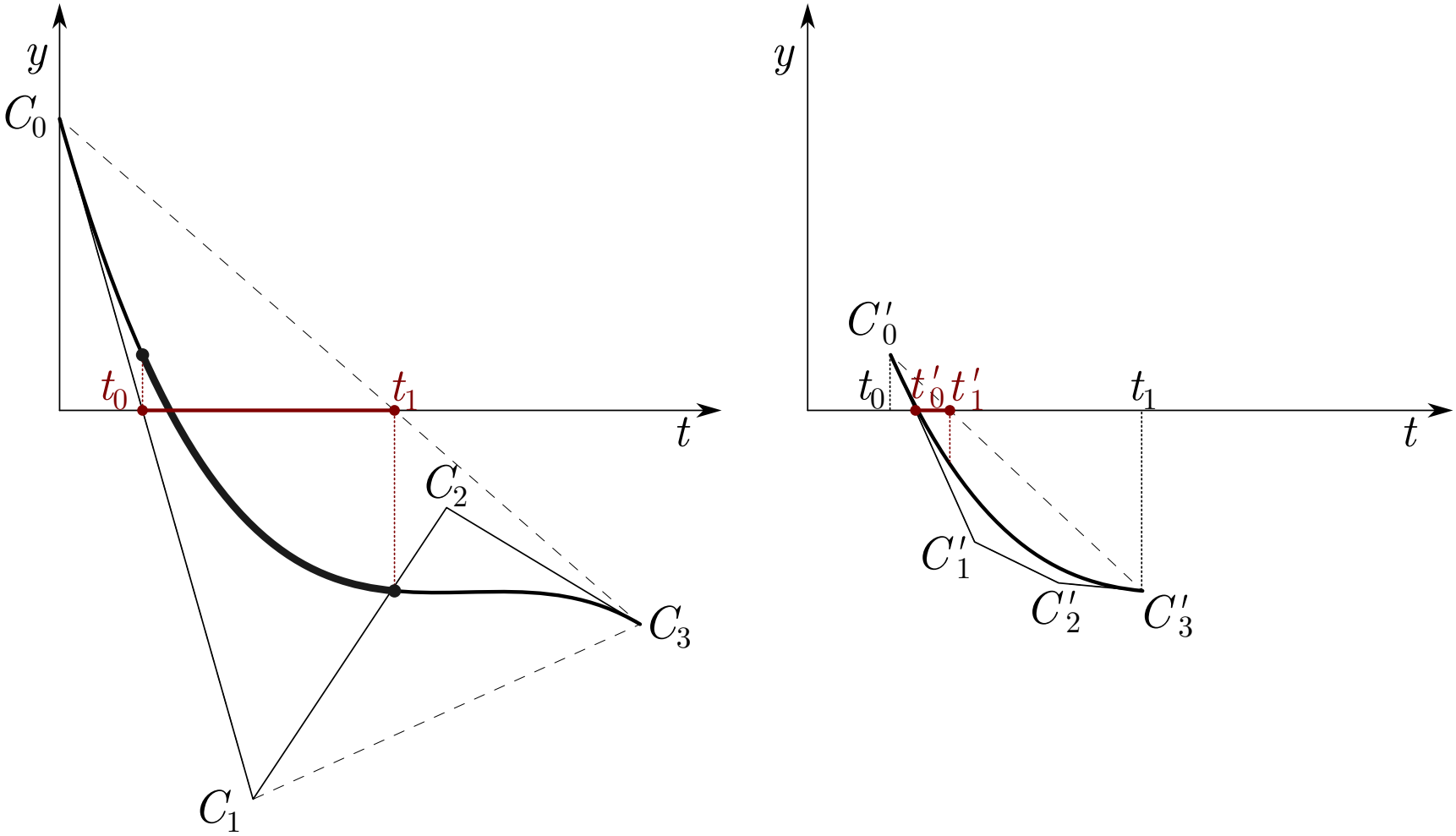
Bézier clipping algorithm

1. Express the graph of $g(t)$, i.e. the points $(t, g(t))$ for $t \in [0, 1]$, as a Bézier curve of degree $d \geq 1$ (= the degree of $g(t)$), e.g. using the polar form or a change-of-basis algorithm. The x-coordinates of its control points are $0, 1/d, 2/d, \ldots, 1$
2. A Bézier curve lies in the convex hull of its control polygon. The intersection of this convex hull with the x-axis is an interval $[t_0, t_1]$ containing all roots of $g(t) = 0$ in interval $[0, 1]$.
   If the convex hull does not intersect the x-axis, there is no root in the interval under consideration
3. Iteratively express the graph of $g(t)$ on the new (shorter) interval $[t_0, t_1]$ as a Bézier curve. Its control points now have x-coordinates t_0, $t_0 + (t_1 - t_0)/d$, $t_0 + 2(t_1 - t_0)/d$, $\ldots$, t_1. Intersect their convex hull with the x-axis again to obtain $[t'_0, t'_1]$
4. Continue until the length of $[t'_0, t'_1]$ is smaller than a prescribed tolerance, i.e. $|t'_1 - t'_0| < \varepsilon$



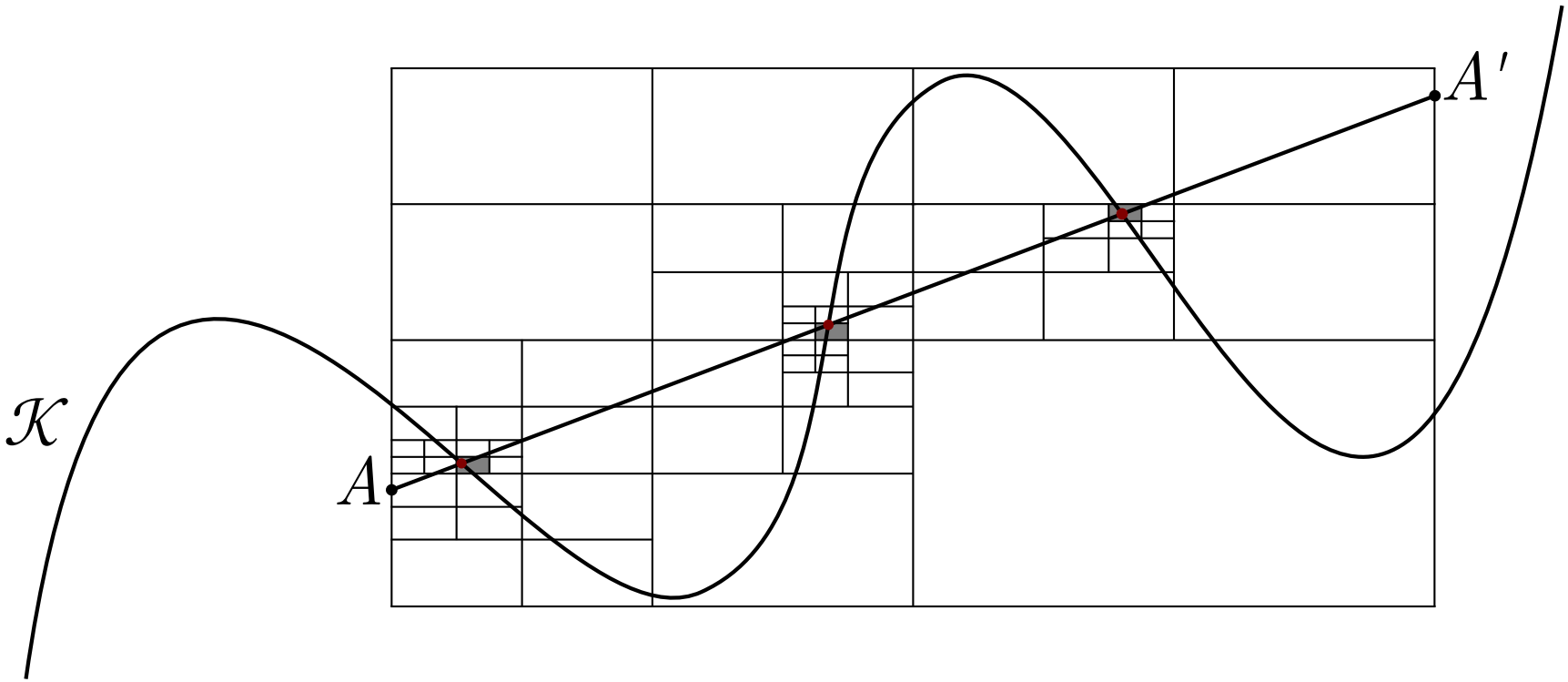
Bézier clipping algorithm

- If the interval $[t'_0, t'_1]$ does not shrink sufficiently after several iterations, for example when $g(t) = 0$ has several separated roots, bisect it into $[t'_0, \frac{t'_1+t'_0}{2}]$ and $[\frac{t'_1+t'_0}{2}, t'_1]$, and process each part separately



Other methods

- If none of the preceding methods applies, approximate the curve $f(x, y) = 0$ by a polyline and find the intersection of the segment with the polyline
- Another option is window subdivision. Divide a rectangular window horizontally and vertically into four smaller windows. Search only those subwindows that contain parts of both the segment and the curve. Continue subdividing until the window size is below the required accuracy, then **verify** the candidate **intersections**



Intersection of two conics $ax^2 + bxy + cy^2 + dx + ey + f = 0$

Given two conics $f_1(x, y) = 0$ and $f_2(x, y) = 0$ with no common component, compute their intersection as follows:

1a) If at least one equation has no y^2 term (without loss of generality, let it be $f_1(x, y) = 0$), rewrite it as $f_1(x, y) = y\,q(x) - p(x) = 0$, where $p(x)$ has degree at most 2 and $q(x)$ degree at most 1.

1b) If both equations contain y^2, write them as $c_1y^2 + \bar{f}_1(x, y) = 0$ and $c_2y^2 + \bar{f}_2(x, y) = 0$, where $\bar{f}_1(x, y)$ and $\bar{f}_2(x, y)$ contain no y^2 term. Multiply the first by c_2 and the second by c_1, then subtract to obtain $y \cdot q(x) - p(x) = 0$, the same form as in 1a)

- For $q(x) \neq 0$, write $y = p(x)/q(x)$ and proceed to step 2)
- If $q(x) \equiv 0$ (the y term vanishes identically), solve $p(x) = 0$ and determine y from $f_2(x, y) = 0$
- At values where $q(x) = 0$: if $p(x) \neq 0$, there is no solution; if also $p(x) = 0$, substitute x into the original equations and determine y

2) Substitute $y = p(x)/q(x)$, for $q(x) \neq 0$, into $f_2(x, y) = 0$ to obtain a rational equation in x:

$$\frac{r(x)}{s(x)} = 0,$$

hence $r(x) = 0$. Solve $r(x) = 0$, which in general has degree at most 4, and substitute each solution with $q(x) \neq 0$ into $y = p(x)/q(x)$ to determine y. Each resulting intersection point must satisfy both original equations



Intersection of two conics – examples

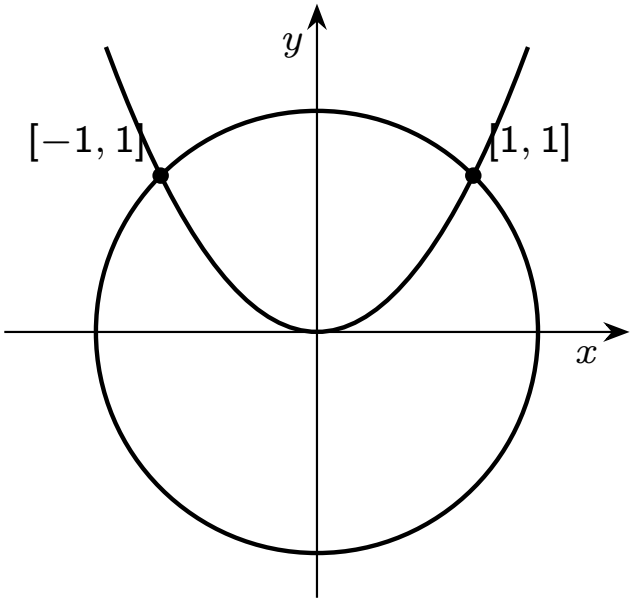
$q(x) \neq 0$ (case 1a)

Curves: $f_1(x, y) = y - x^2 = 0$ (parabola)
$f_2(x, y) = x^2 + y^2 - 2 = 0$ (circle)

The equation $y \cdot q(x) - p(x) = 0$ from $f_1 = 0$:
$y \cdot 1 - x^2 = 0 \Rightarrow q(x) = 1 \neq 0$

Write $y = x^2$ and substitute into $f_2(x, y) = 0$ to obtain $x^4 + x^2 - 2 = 0 \Rightarrow x = \pm 1$

Intersection points: $P_1 = [1, 1]$, $P_2 = [-1, 1]$



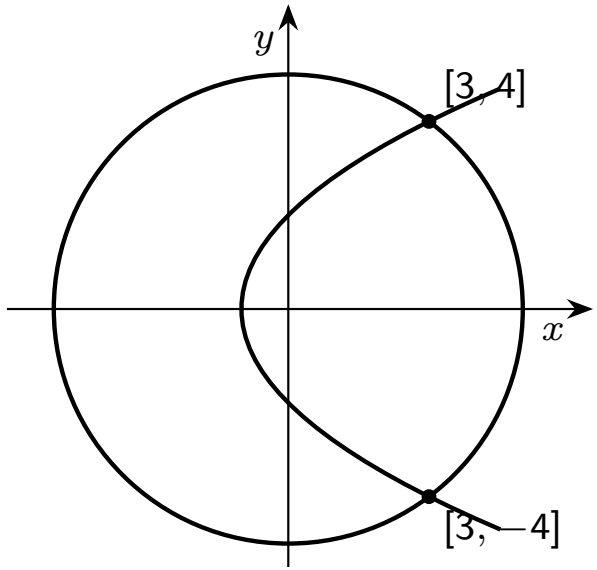
$q(x) \equiv 0$ (case 1b)

Curves: $f_1(x, y) = x^2 + y^2 - 25 = 0$ (circle)
$f_2(x, y) = y^2 - 4x - 4 = 0$ (parabola)

The equation $y \cdot q(x) - p(x) = 0$ from $f_1 - f_2 = 0$:
$y \cdot 0 - (-x^2 - 4x + 21) = 0 \Rightarrow q(x) \equiv 0$

The y term vanishes. From $p(x) = 0$, we obtain $x = 3$ (for $x = -7$, no real y exists). Substitution into $f_2(3, y) = 0$ gives $y = \pm 4$

Intersection points: $P_1 = [3, 4]$, $P_2 = [3, -4]$





Intersection of two conics – examples (cont.)

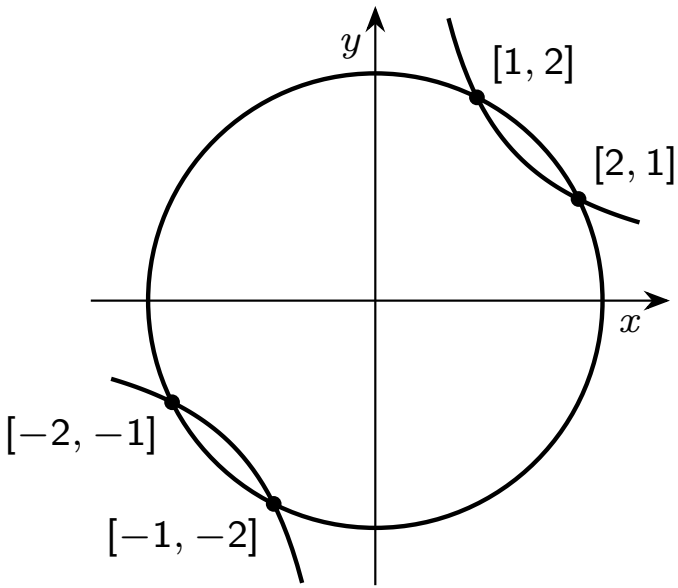
$q(x_0) = 0,\ p(x_0) \neq 0$ (case 1a)

Curves: $f_1(x, y) = xy - 2 = 0$ (hyperbola)
$f_2(x, y) = x^2 + y^2 - 5 = 0$ (circle)

The equation $y \cdot q(x) - p(x) = 0$ from $f_1 = 0$:
$y \cdot x - 2 = 0 \Rightarrow x_0 = 0 \quad (q(0) = 0,\ p(0) = 2)$

For $x_0 = 0$, the contradiction $0 \cdot y = 2 \Rightarrow$ no intersection on the y-axis. For $x \neq 0$, substitute $y = 2/x$ into $f_2(x, y) = 0$

Intersection points: $P_{1,2} = [\pm 1, \pm 2],\ P_{3,4} = [\pm 2, \pm 1]$



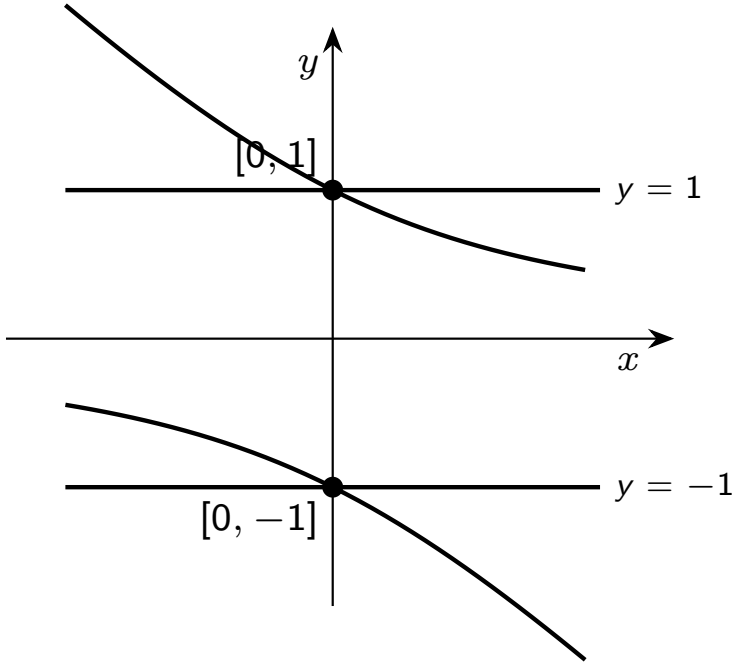
$q(x_0) = 0,\ p(x_0) = 0$ (case 1b)

Curves: $f_1(x, y) = y^2 + xy - 1 = 0$ (hyperbola)
$f_2(x, y) = y^2 - 1 = 0$ (two parallel lines)

The equation $y \cdot q(x) - p(x) = 0$ from $f_1 - f_2 = 0$:
$y \cdot x - 0 = 0 \Rightarrow x_0 = 0 \quad (q(0) = 0,\ p(0) = 0)$

For $x_0 = 0$, we have $0 \cdot y = 0$. Substituting $x_0 = 0$ into $f_2(0, y) = 0$ gives $y^2 - 1 = 0 \Rightarrow y = \pm 1$

Intersection points: $P_1 = [0, 1],\ P_2 = [0, -1]$



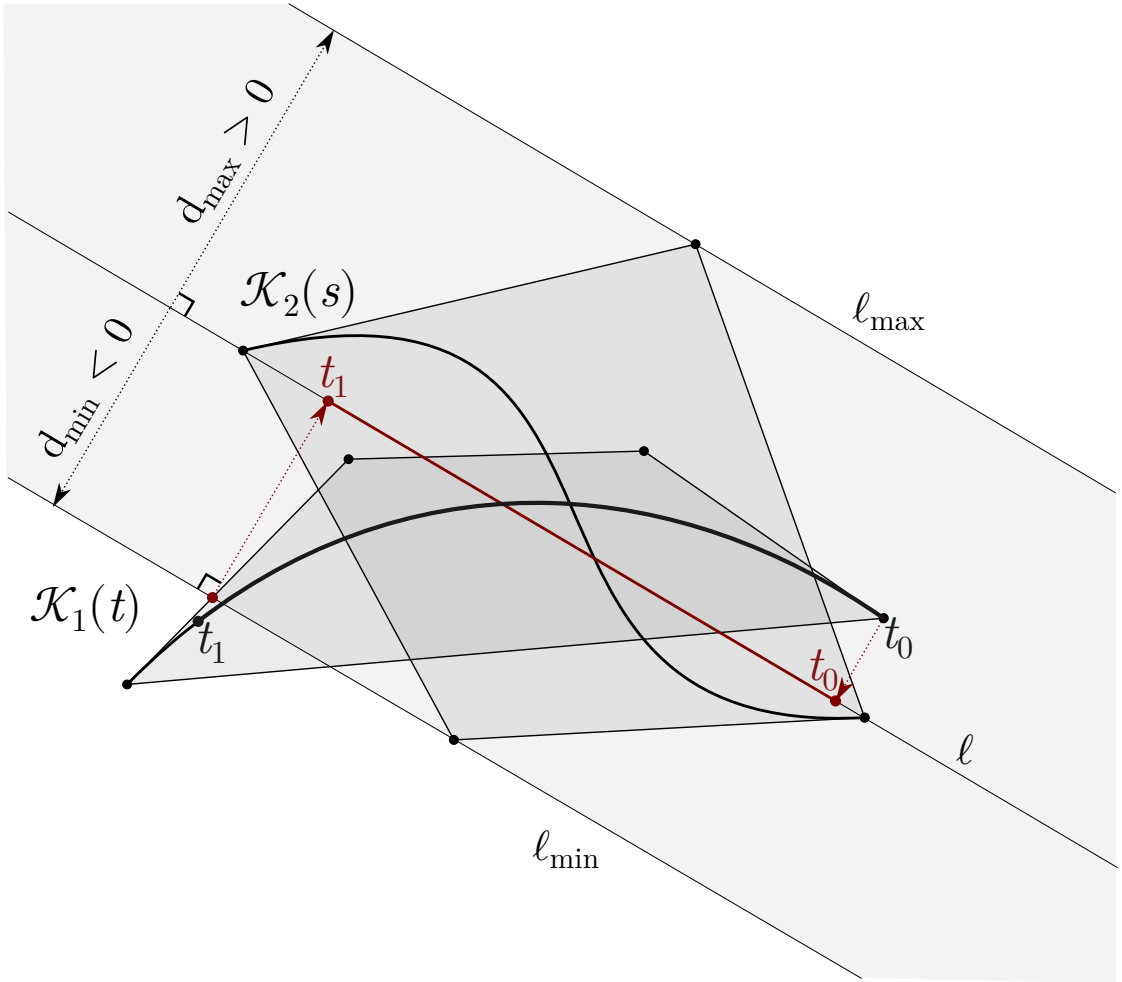


Intersection of two Bézier curves

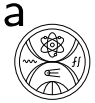
- For two Bézier curves $\mathcal{K}_1(t)$ and $\mathcal{K}_2(s)$, where $t, s \in [0, 1]$, we can again use Bézier clipping
- Any common point of the curves lies in the intersection of the convex hulls of their control polygons
- If the convex hulls are disjoint, the curves have no common point
- Procedure for finding the intersections (see the next slide):
  1. Choose the line ℓ through the endpoints of $\mathcal{K}_2$ and compute the signed distances of its control points from this line. Their minimum $d_{\min}$ and maximum $d_{\max}$ define a strip between two parallel lines containing the entire control polygon and hence $\mathcal{K}_2$.
  2. The signed distance of $\mathcal{K}_1(t)$ from ℓ is a polynomial $g(t)$. Express its graph in Bézier form, as in root finding. **Intersect the convex hull** of its control points **with the strip** $d_{\min} \leq y \leq d_{\max}$. **Project** this intersection onto the t-axis to obtain $[t_0, t_1]$. If the intersection is empty, the curves have no common point
  3. Discard the parts of $\mathcal{K}_1$ outside $[t_0, t_1]$, since they lie outside the strip containing $\mathcal{K}_2$. Use de Casteljau's algorithm to represent the remaining part by new control points.
  4. Swap the roles of the curves: construct a new strip from the remaining part of $\mathcal{K}_1$ and clip $\mathcal{K}_2$. Repeat clipping alternately for both curves
  5. If the intervals do not shrink sufficiently, split one subcurve into two and process both pairs separately. Discard pairs with disjoint convex hulls.
  6. Once the intervals are sufficiently small, choose their midpoints t_m and s_m and verify that $\|\mathcal{K}_1(t_m) - \mathcal{K}_2(s_m)\| < \varepsilon$. This gives an intersection within the required geometric tolerance



Other approaches



- If the curves are not polynomial, approximate them by polylines and find the intersections of the polylines
- As with curve–segment intersection, we can also use the algorithm that subdivides a window into four subwindows

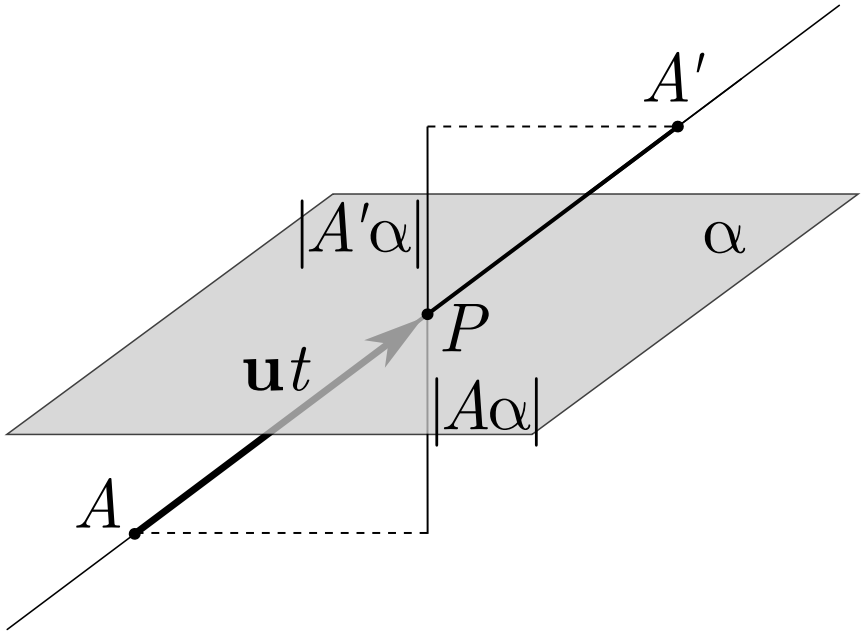


Segment–plane intersection in 3D

- As in the plane, use the parametric segment $AA' : A + \mathbf{u}t$, for $t \in [0, 1]$, where $\mathbf{u} = A' - A$:

$$x = a_x + u_x t$$
$$y = a_y + u_y t$$
$$z = a_z + u_z t$$

and the plane α given by the implicit linear equation $f(x, y, z) = 0$, i.e. $ax + by + cz + d = 0$



Segment–plane intersection in 3D

- First evaluate $f(A)$ and $f(A')$ by substituting each point into the implicit equation of the plane α
- If $f(A) = f(A') = 0$, the entire segment lies in the plane
- If $f(A) \cdot f(A') > 0$ (both points lie in the same half-space bounded by α), the segment does not intersect the plane
- Otherwise, the intersection $P = AA' \cap \alpha$ exists. Compute it using $P = A + (A' - A)t$, since it must lie on AA'
- Determine t from $t = \frac{|PA|}{|A'A|}$, using the distances of A, A' from α:

$$t = \frac{|PA|}{|A'A|} = \frac{|A\alpha|}{|A'\alpha| + |A\alpha|} = \frac{\frac{|f(A)|}{\sqrt{a^2+b^2+c^2}}}{\frac{|f(A')|}{\sqrt{a^2+b^2+c^2}} + \frac{|f(A)|}{\sqrt{a^2+b^2+c^2}}} = \frac{|f(A)|}{|f(A')| + |f(A)|}$$

- Thus, the intersection point P is given by:

$$P = A + \frac{|f(A)|}{|f(A')| + |f(A)|}(A' - A)$$



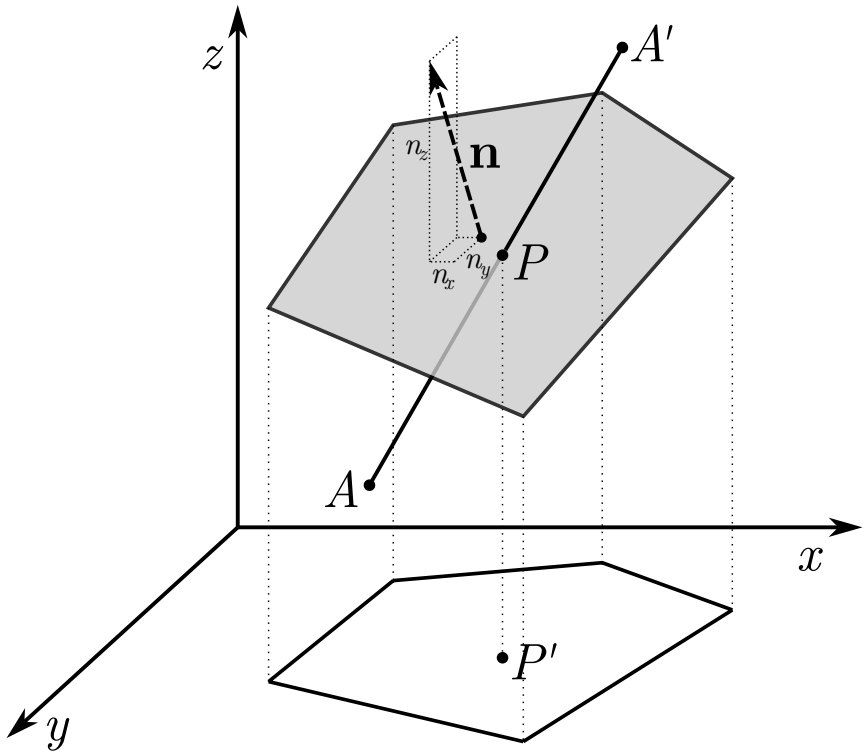
Segment–polygon intersection in 3D

- Segment–polygon intersection in 3D requires two tasks:
  1. find the intersection of the segment with the plane containing the polygon,
  2. determine whether the segment–plane intersection lies inside or on the boundary of the polygon
- The first task is solved by the method already described
- For the second task, project the scene from 3D to 2D using orthogonal projection. Choose the coordinate plane onto which the polygon has the largest projected area:
  1. compute a normal vector $\mathbf{n} = (n_x, n_y, n_z)$ to the plane $n_x x + n_y y + n_z z + d = 0$ containing the polygon
  2. find the largest absolute component of the plane's normal vector $\max\{|n_x|, |n_y|, |n_z|\}$
  3. project orthogonally onto the coordinate plane perpendicular to the axis corresponding to this component. For example, if $\max\{|n_x|, |n_y|, |n_z|\} = |n_z|$, project along the z-axis onto the xy-plane: $(x, y, z) \mapsto (x, y)$ (see the figure on the next slide)
- If the entire segment lies in the polygon's plane, solve the planar segment–polygon intersection problem after projection

For triangles and rectangles, simpler and more efficient methods exist for testing whether the intersection lies in the polygon (covered in CG(2))



Segment–polygon mesh intersection



- To find the common points of a segment and a polygon mesh, intersect the segment with every polygon in the mesh
- If the mesh forms the faces of a **convex** polyhedron, use a procedure **analogous** to planar segment–convex polygon clipping ($t_{\text{in}}, t_{\text{out}}$). Clipping against half-spaces gives the part of the segment inside the polyhedron. To find intersections with its surface, then determine which points of this part lie on the polyhedron's boundary



Segment–implicit surface intersection

- The surface is given implicitly by $f(x, y, z) = 0$ and consists of the points in space satisfying $f(x, y, z) = 0$
- Proceed as for the intersection of a segment with an implicit curve in the plane
- Substituting the parametric expression for the segment gives an equation $g(t) = 0$ for t; seek solutions in $[0, 1]$. If $g(t)$ is quadratic, its roots can be computed analytically; otherwise, use a numerical root-finding method
- Alternatively, **approximate** the implicit surface by a **polygon** mesh and use the segment–polygon mesh intersection procedure
- The surface normal direction at a point $P = (p_x, p_y, p_z)$ can be computed using the following lemma:

Lemma

Let a surface be defined by $f(x, y, z) = 0$ and let $P = (p_x, p_y, p_z)$ lie on it, i.e. $f(p_x, p_y, p_z) = 0$. Suppose f is continuously differentiable in a neighbourhood of P. If $\frac{\partial f}{\partial x}(p_x, p_y, p_z) \neq 0$ or $\frac{\partial f}{\partial y}(p_x, p_y, p_z) \neq 0$ or $\frac{\partial f}{\partial z}(p_x, p_y, p_z) \neq 0$, then a normal vector to the surface at P is

$$\mathbf{n} = \left(\frac{\partial f}{\partial x}(p_x, p_y, p_z), \frac{\partial f}{\partial y}(p_x, p_y, p_z), \frac{\partial f}{\partial z}(p_x, p_y, p_z) \right)$$

