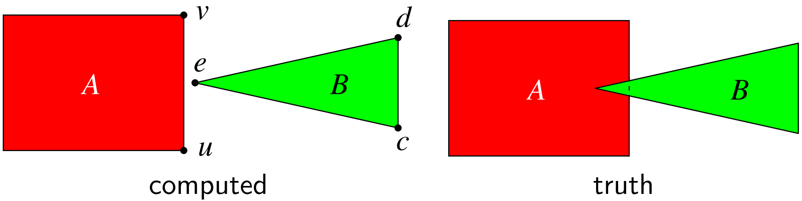
Robustness

Elisha Sacks

Robustness

- Robustness: accurate output for all input.
- Numerical error (rounding and truncation) is negligible per se.
- But numerical error can induce structural error.
- Controlling structural error is challenging.
- The problem is ubiquitous in computational geometry.

Structural Error

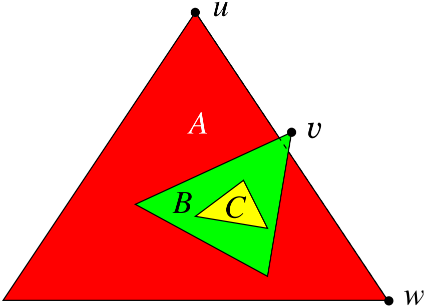


- Numerical error in line intersection causes structural error in the polygon intersection query (*no* instead of *yes*).
- Backward error metric: distance from input to alternate input for which computed result is the true result.
  - Norm in scientific computing.
  - Models numerical and structural error.
  - Distance from point a to line uv in our example.

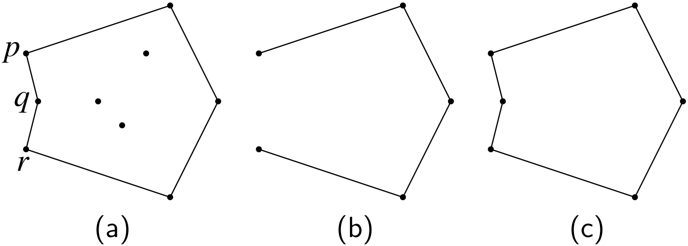
Inconsistency

An output is inconsistent if it is not true for any input, so the error is infinite and the output is nonsense.

- Polygon P is inside (outside) polygon Q if their edges do not intersect and any vertex a of P is inside (outside) Q.
- Numerical error: v is above uw.
- Structural error: B is outside A.
- Inconsistency: C is inside A and B.



Convex Hull



A naive convex hull algorithm tests if every pair of points forms a hull edge. A pair forms an edge if every other point is to its left.

(a) The points p, q, and r are almost collinear, so $LT(p, q, r) < 0$, $LT(p, r, q) < 0$, and $LT(q, r, p) < 0$ due to rounding error.

(b) This inconsistency causes a structural error: a gap in the hull.

(c) The textbook algorithm is robust: an incorrect LT predicate creates a small dent in the hull.

Robustness Problem 1: Large Errors

- CG algorithms branch on predicates: signs of expressions.
- A tiny numerical error can cause a sign error.
- The sign error can cause a control flow error.
- The control flow error can cause a structural error.
- The structural error can be arbitrarily large.
- The LeftTurn (LT) predicate caused the example inconsistency.

Robustness Problem 2: Degeneracy

- A predicate is degenerate if its value is zero.
  - LeftTurn is degenerate for collinear points.
  - PointInCircle is degenerate when the point is on the circle.
- Degeneracy arises from relations among geometric objects.
- Degeneracy is common in applications due to design constraints and symmetry.
- Degeneracy adds many special cases to algorithms.
- Challenges: efficient detection and correct handling.

Robustness Strategy 1: Workarounds

- Commercial software contains robustness workarounds.
- Each new problem requires lengthy analysis.
- New workarounds often invalidate old ones.
- Multi-step computations risk garbage-in-garbage-out.
- Robustness must be algorithmic.
- I learned the hard way.

Robustness Strategy 2: Inconsistency Sensitivity

- Evaluate predicates with floating point arithmetic.
- Extend algorithms to handle errors efficiently and accurately.
- No special treatment required for degeneracy.
- Milenkovic and I compute arrangements of algebraic plane curves with small errors [1].
- We could not extend the paradigm to 3D triangles!

[1] V. Milenkovic and E. Sacks, An approximate arrangement algorithm for semi-algebraic curves, *International Journal of Computational Geometry and Applications* 17(2), 175–198, 2007.

Robustness Strategy 3: Exact Computational Geometry

- Prevent structural error by computing predicates exactly [1].
- Handle easy cases with interval arithmetic.
- Handle hard cases with rational arithmetic.
- CGAL and LEDA libraries implement this approach.
- Adaptive precision evaluation is a fast alternative for polynomials in the input [2].
- Degeneracies must still be handled.

[1] C. Yap, Robust geometric computation, In *Handbook of discrete and computational geometry*, Second Edition, 2004.

[2] J. Shewchuk, Adaptive precision floating-point arithmetic and fast robust geometric predicates, *Discrete and Computational Geometry* 18, 305-363, 1997.

Interval Arithmetic

- Interval arithmetic takes an expression e and computes a floating point interval $[\underline{e}, \overline{e}]$ that contains its true value.
- If the interval excludes zero, the sign of e is determined.
- The interval of an input p is $[p, p]$.
- The arithmetic operators are extended to intervals.
- The output intervals are rounded outward.
- Interval arithmetic is about 50% slower than floating point.

Interval Arithmetic Operators

e	$\underline{e}$	$\overline{e}$	
a number p	p	p	
$a+b$	$\underline{a}+\underline{b}$	$\overline{a}+\overline{b}$	
$a-b$	$\underline{a}-\overline{b}$	$\overline{a}-\underline{b}$	
$a\times b$	$\underline{a}\times\underline{b}$	$\overline{a}\times\overline{b}$	for $\underline{a}\geq 0, \underline{b}\geq 0$
	$\overline{a}\times\underline{b}$	$\underline{a}\times\overline{b}$	for $\underline{a}\geq 0, \overline{b}< 0$
	$\overline{a}\times\underline{b}$	$\overline{a}\times\overline{b}$	for $\underline{a}\geq 0, \underline{b}< 0<\overline{b}$
	$\overline{a}\times\underline{b}$	$\underline{a}\times\underline{b}$	for $\overline{a}\leq 0, \overline{b}\leq 0$
a/b	$\underline{a}/\overline{b}$	$\overline{a}/\underline{b}$	for $\underline{a}\geq 0, \underline{b}> 0$
	$\underline{a}/\underline{b}$	$\overline{a}/\overline{b}$	for $\overline{a}\leq 0, \underline{b}> 0$
	$\underline{a}/\underline{b}$	$\overline{a}/\underline{b}$	for $\underline{a}< 0<\overline{a}, \underline{b}> 0$

- $\underline{e}$ is rounded down and $\overline{e}$ is rounded up.
- One can set the rounding mode to up and down.
- It is much faster to round up and compute $\underline{e} = -(\overline{-e})$.

Integer Arithmetic

- Integers are in base b with b a large integer, typically $b = 2^{64}$.
- Example: $(3, 2, 4)$ represents $3 + 2b + 4b^2$.
- Grade school addition and subtraction are $O(n)$ for n digits.
- Grade school multiplication is $O(n^2)$.
- State of the art is $O(n \log n)$.
- The GNU MP library is an excellent open source.

Rational Arithmetic

- Numbers are ratios of integers.
- Operators are grade school algebra.

$$
\begin{aligned}
\frac{a}{b} \pm \frac{c}{d} &= \frac{ad \pm bc}{bd} \\
\frac{a}{b} \times \frac{c}{d} &= \frac{ac}{bd} \\
\frac{a}{b} \div \frac{c}{d} &= \frac{ad}{bc}
\end{aligned}
$$

- The denominators grow rapidly.
- The growth is slowed by removing common factors.

Robustness Strategy 4: Controlled Perturbation

- Compute predicates exactly for a perturbed input [1].
- Compute predicates in floating point with a safety check.
- Determine an input perturbation size δ that makes predicates safe with high probability.
- Apply a δ-perturbation and execute the algorithm.
- If any check fails, restart with a different δ-perturbation.
- Alternately, start with a small δ and double it at each restart.
- The backward error equals the final δ.
- Fast: small overhead over pure double-float evaluation.
- No special treatment is required for degeneracy.
- Problem: rare bad cases force large δ.

[1] D. Halperin, Controlled Perturbation for Certified Geometric Computing with Fixed-Precision Arithmetic, ICMS, 92–95, 2010.

Adaptive Controlled Perturbation (ACP)

- User-specified error bound δ (typically 10^{-8}).
- Replace restarts with extended precision predicate evaluation.
  - Initial evaluation in floating point interval arithmetic.
  - Rare bad cases handled by repeatedly doubling the precision.
  - Extended precision arithmetic uses MPFR library.
  - Object hierarchies control memory cost.
- 10%-20% slower than floating point predicate evaluation.

[1] E. Sacks and V. Milenkovic, Robust cascading of operations on polyhedra, *Computer-Aided Design* 46, pp. 216–220, 2014.

[2] V. Milenkovic, E. Sacks, and S. Trac, Robust Free Space Computation for Curved Planar Bodies, *IEEE Transactions on Automation Science and Engineering* 10:4, pp. 875–883, 2013.

[3] M.H Kyung, E. Sacks, and V. Milenkovic, Robust polyhedral Minkowski sums with GPU implementation, *Computer-Aided Design* 67-–68, pp. 48–57, 2015.

Identities

- An identity is a predicate that is degenerate due to relations among its antecedents.
- Example: let segments ab and cd intersect at p. The predicate $\mathrm{LeftTurn}(a, b, p)$ is an identity.
- Identities persist under input perturbation.
- They must be detected and handled.
- Manual detection is practical for simple cases only.
- We have developed an efficient identity detection algorithm.

[4] V. Milenkovic and E. Sacks, Efficient predicate evaluation using probabilistic degeneracy detection, *International Journal of Computational Geometry and Applications*, 32:39-54, 2022.