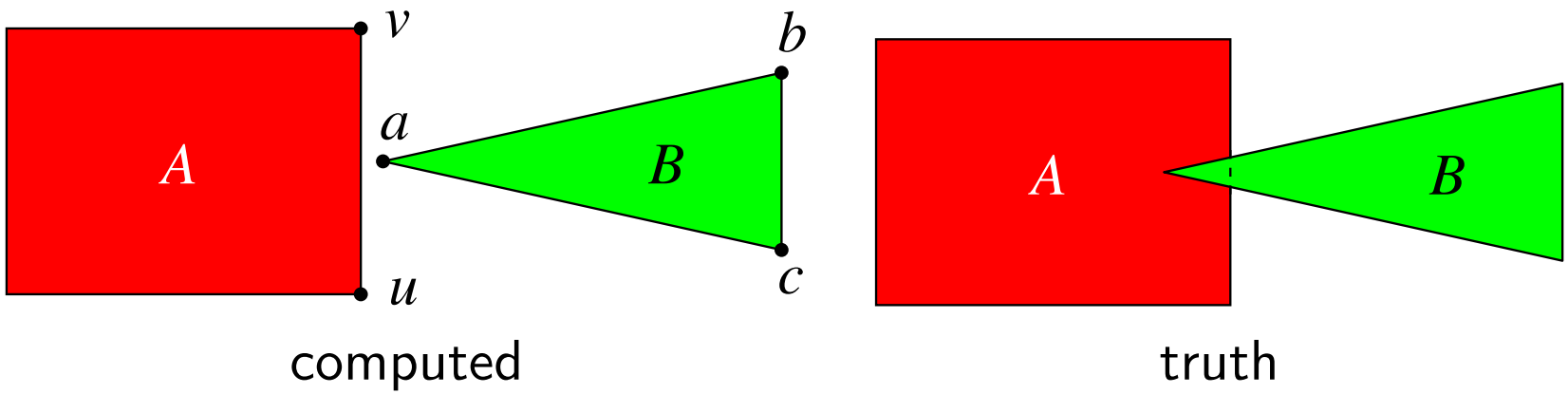
Robustness

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

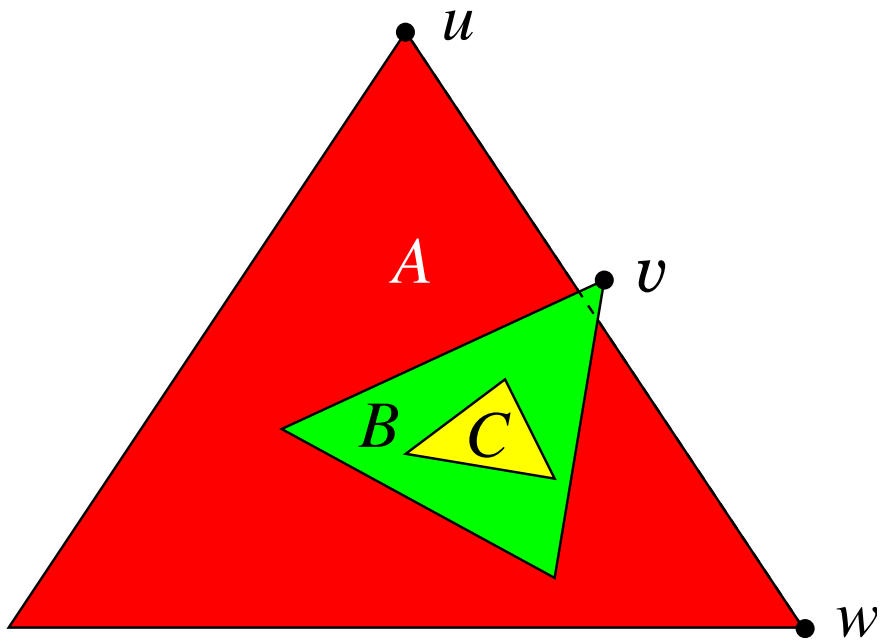


- Numerical error in line intersection causes structural error in the polygon intersection query (*no* instead of *yes*).
- Backward error metric: distance from input to alternate input for which computed result is the true result.
  - Norm in scientific computing.
  - Models numerical and structural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



Robustness Problem 1: Large Errors

- CG algorithms branch on the signs of predicates.
- 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.
- Errors in the circ predicate cause the example inconsistency.

Robustness Problem 2: Degeneracy

- A predicate is degenerate if its value is zero.
  - circ is degenerate for collinear points.
  - point-in-circle is degenerate when a, b, c are collinear.
- Degeneracy arises from relations among geometric primitives.
- 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 in machine double-float.
- Extend algorithms to handle errors efficiently and accurately.
- No special treatment required for degeneracy.
- Milenkovic and Sacks compute arrangements of algebraic plane curves with small errors.
- We could not extend the paradigm to 3D triangles!

Robustness Strategy 3: Exact Computational Geometry

Prevent structural error by computing predicates exactly.

- Yap 2004
  - Handle easy cases with double-float arithmetic.
  - Handle hard cases with root separation bounds.
  - LEDA and Core libraries implement this approach.
  - They are slow and memory bound.
- Shewchuk
  - Adaptive precision predicate evaluation.
  - Fast, but no defined parameters.
- Neither approach addresses degeneracy handling.

Robustness Strategy 4: Controlled Perturbation

Halperin 2010: compute predicates exactly for a perturbed input.

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

Adaptive Controlled Perturbation (ACP)

Sacks and Milenkovic 2014: error-bounded perturbation.

- User-specified error bound δ (typically 10^{-8}).
- Enforced by setting perturbation size to δ.
- Exact predicate evaluation for perturbed input.
  - Initial evaluation in floating point interval arithmetic.
  - Rare bad cases handled by repeatedly doubling the precision.
  - Extended precision arithmetic uses MPFR library.
  - Object-oriented geometric primitives control memory cost.
- 10%-20% slower than floating point predicate evaluation.
- Packing polyhedra, free spaces of curved planar bodies, Minkowski sums of polyhedra on the GPU.