Two Strategies for Approximate Computational Geometry

Elisha Sacks, Purdue University

joint work with

Victor Milenkovic, University of Miami
Problem

- Algorithms are expressed in real RAM model.
- Input is assumed in general position.
- Implementations must use computer arithmetic.
- Implementations must handle degenerate input.
Geometric Predicates

- Main interface with real RAM model (also geometric constructions).
- Predicate $P(x)$ is true when polynomial $f(x)$ is positive.
- Unsafe predicate: $|f(x)|$ near the rounding unit.
- Degenerate predicate: $f(x) = 0$.
- Singular predicate: $f(x) = 0$ and $f'(x) = 0$. 
Exact Computational Geometry

- Implement predicates exactly using real algebraic geometry.
- Symbolic perturbation of degenerate predicates.
- Technical Problems
  - Running time grows rapidly with algebraic degree.
  - Bit complexity grows rapidly in iterated computation.
  - Large constant factors and programming overhead.
Conceptual Problem

- Scientific computing is approximate because exact solutions are impractical and unnecessary.
- That is why rounding and numerical analysis were invented.
- Why should computational geometry be exact?
Approximate Comp. Geometry

- Implement predicates approximately using floating point arithmetic and numerical solvers.
- Advantages:
  - Running time grows modestly with degree.
  - Constant bit complexity.
  - Small constant factors.
- Challenge: generate consistent output.
Consistency

• Error metric: distance from input to perturbed input for which the computed output is correct.
• Inconsistent output: no such perturbation exists.
• Example: plane curves in cyclic vertical order.
  • $a < b$ before $p_x$, $b < c$ before $r_x$, $c < a$ after $q_x$.
• Numerical error causes $q_x < p_x$.
• Inconsistency: $a < b < c < a$ on $(q_x, p_x)$.
Inconsistency Sensitive Strategy

- Adapt RAM algorithms to generate consistent output despite computation error.
- Bound output error and extra cost in terms of computation error and inconsistency count.
- Advantage: speed and accuracy.
- Disadvantage: lack of generality.
Arrangement Algorithm

- **Input:** $x$-monotonic semi-algebraic curves, crossing module.
- **Step 1:** Compute curve crossings and $y$-order.
- **Step 2:** Embed curve endpoints.
- **Output:** $O(\epsilon + kn\epsilon)$ accurate arrangement for $n$ curves and an $\epsilon$-accurate crossing module with $k$ inconsistencies.
- Proving consistency is much easier than proving an error bound!
Crossing Module

- Crossings computed with custom eigensolver.
- Accuracy, $\epsilon$, of 12–16 decimal digits.
- Running time $cd^4$ for degree $d$.
- $c = 6$ microseconds on 2.2 GHz processor.
Step 1: Curve $y$-order

- Crossing module defines curve $y$-order, $a <_x b$.
- $k$ inconsistencies: $a <_x b <_x c <_x a$ on maximal open interval.
- Bentley sweep with two modifications:
  1. Don’t swap non-adjacent curves.
  2. Immediately swap out-of-order curves.
- Sweep list defines output $y$-order, $a <'_x b$.
- Error analysis: bound distance between $a, b$ at $x$ where $a <'_x b$ and $b <_x a$.
- Key idea: there exists a sequence
  $a <_x s_1 <_x \cdots <_x s_p <_x b$ with $p \leq k$. 
Step 2: Endpoint Embedding

Inconsistency between endpoint and curve $y$-orders.

inconsistency

fix

Two Strategies for Approximate Computational Geometry – p. 12/26
Step 2: Endpoint Embedding

inconsistency

fix
Perturbation Methods

- Perturb input to avoid inconsistency and degeneracy.
- Minimize perturbation size relative to success probability.
- Advantage: general.
- Disadvantages of prior work
  - inaccurate, especially for near-singular input.
  - incompatible with equality constraints (implicit parameters).
Constrained Linear Perturb.

Strategy: assign signs to polynomials then compute minimal perturbation that realizes these signs.

- No error or cost for safe polynomials.
- Accurate perturbation of singular polynomials.
- Implicit parameters handled.
- Signs can be set to zero.
CLP Implementation Strategy

- Online algorithm: compute perturbation for both signs of polynomial subject to prior signs; select smaller perturbation.
- Linear programming implementation.
- Linear Taylor series for regular polynomials.
- Replace a near-singular polynomial with a regular proxy and constrain it to have the same sign.
CLP Definition

- CLP defined for polynomials $f_1, \ldots, f_m$ at $x = a$.
- $f_i$ safe: $|f_i(a)| > k_i \mu$ with $\mu$ the rounding unit.
- Perturbation: $p = a + \delta v$, $\delta \geq 0$, $||v|| = 1$.
- CLP: $p$ and signs $s_1, \ldots, s_m$ with $s_i = \pm 1$.
- If $f_i$ is safe, $s_i$ is the computed sign. If not, $s_i$ is the sign of the rate of change $\nabla f_i \cdot v$.
- $s_i f_i(p) \geq k_i \mu$ for $i = 1, \ldots, m$. 
Core Algorithm

- Extend CLP from $f_1, \ldots, f_{m-1}$ to $f_m$.
- If $f_m$ is safe, return the computed sign and the prior $p$.
- Else assign the sign and $v$ that maximize the minimum of the rates, $r_i = s_i \nabla f_i \cdot v / k_i$, at which the unsafe $f_i$ become safe.
- Maximize $r$ subject to $r_i \geq r$ and $s_m = \pm 1$; assign $s_m$ and $v$ from the larger $r$ value.
- Set $\delta = 2\mu / r$ to make $s_i$ correct for the linearized $f_i$ with margin $2k_i\mu$.
- Verify $s_i f_i(p) \geq k_i\mu$ for all unsafe $f_i$. 
Sorting Example

- Sort four equal numbers $x_i = 0$.
- Predicate polynomials are $x_i - x_j$ with $k_i = 1$.
- Six signs assigned during sorting.
- Perturbation direction constraints: $-1 \leq v_i \leq 1$.
- Sign 1: $x_2 - x_1$ with cases $v_2 - v_1 \geq r$ and $v_1 - v_2 \geq r$; maximum of $r = 2$ for both, so $s_1 = 1$ and $x_1 < x_2$. 

\[ \begin{array}{cccc}
    x_1 & x_4 & x_2 \\
    \hline
    -1 & 0 & 1 \\
\end{array} \]
Sorting Example

• Sign 2: \( x_3 - x_2 \)
  - \( s_2 = 1: v_3 - v_2 \geq r \) and \( v_2 - v_1 \geq r \) with maximum \( r = 1 \) at \( v = (-1, 0, 1, 0) \).
  - \( s_2 = -1: v_2 - v_3 \geq r \) and \( v_2 - v_1 \geq r \) with maximum \( r = 2 \) at \( v = (-1, 1, -1, 0) \).
  - Set \( s_2 = -1 \) and \( x_3 < x_2 \).

\[
\begin{array}{cccc}
  x_3 & x_1 & x_4 & x_2 \\
  \hline
  \text{\( -1 \)} & 0 & 1 \\
\end{array}
\]

• Sign 6: \( x_1 < x_3 < x_4 < x_2 \).

\[
\begin{array}{cccc}
  x_1 & x_3 & x_4 & x_2 \\
  \hline
  \text{\( -1 \)} & -1/3 & 1/3 & 1 \\
\end{array}
\]
Pappus Example

- Sort $x$ coordinates of the intersection points of 9 lines with 9 near-triple intersection points.
- First 8 triples permit all signs; 254 of these permit both signs for ninth triple.
Full CLP algorithm

- Proxies for near-singular polynomials.
  - Status: manual construction.
  - Research: automated construction for determinant polynomials.
- Implicit parameter definitions.
  - Status: regular definitions.
  - Research: singular definitions.
- Output simplification.
- Random perturbation direction.
CLP versus controlled pert.

- Convex hull of $n$ identical points: $\delta = 121\mu$ for $n = 100$, $\delta = 238\mu$ for $n = 200$, $\delta = 1619\mu$ for $n = 1000$.

- Controlled perturbations $2 \times 10^8$ times larger.

- Delaunay triangulation of $n$ identical points: $\delta = 399\mu$ for $n = 100$, $\delta = 1767\mu$ for $n = 200$, $\delta = 14959\mu$ for $n = 1000$.

- Controlled perturbations $10^{11}$ times larger.

- Delaunay triangulation of $n$ points on unit line segment: $\delta = 636\mu$ for $n = 100$, $\delta = 2933\mu$ for $n = 200$, $\delta = 8479\mu$ for $n = 1000$.

- Controlled perturbations $2 \times 10^6$, $7 \times 10^6$, $2 \times 10^9$ times larger.
CLP versus ECG

- Arrangement of 100 random degree-6 algebraic curves: 22 seconds with CLP; 220 seconds with ECG [Eigenwillig, 2008].
- Arrangement of 100 degenerate degree-6 semi-algebraic curves.
CLP versus ECG

- Arrangement contains 1330 vertices, including 43 clusters of nearly equal vertices with an average of 23 vertices per cluster and 55 vertices in the largest cluster.
- 1.5 seconds with CLP; estimated 30,000 seconds with ECG.
- Estimate based on measured root separation, $\rho$, and on published $\log^2 \rho$ running time.
Conclusion

- Approximate computational geometry is fast and accurate.
- Consistency is the challenge.
- Consistency sensitivity works case by case.
- CLP is algorithm-independent.
- We aim for a black box CLP library.