Overview

I. Introduction
II. Meshing
III. Roots
IV. Motion Planning
V. Voronoi Diagrams
VI. Conclusion
I. Introduction
Trouble with Computational Models

- Ancient Greek Geometry
Trouble with Computational Models

- Ancient Greek Geometry
  - Ruler and Compass Model

Impossibility of squaring a circle (Lindemann 1882)
Trouble with Computational Models

- Ancient Greek Geometry
  - Ruler and Compass Model
- General Models of Computation
Trouble with Computational Models

- Ancient Greek Geometry
  - Ruler and Compass Model

- General Models of Computation
  - Turing Machine Model
Trouble with Computational Models

- Ancient Greek Geometry
  - Ruler and Compass Model

- General Models of Computation
  - Turing Machine Model (Church’s Thesis)

- Models for Geometric Computing
Trouble with Computational Models

- Ancient Greek Geometry
  - Ruler and Compass Model

General Models of Computation

- Turing Machine Model (Church’s Thesis)

Models for Geometric Computing

- Real RAM model
Trouble with Computational Models

- Ancient Greek Geometry
  - Ruler and Compass Model

- General Models of Computation
  - Turing Machine Model (Church’s Thesis)

- Models for Geometric Computing
  - Real RAM model (not Church Equivalent!)

... the trouble begins
The Numerical Nonrobustness Phenomenon

- The trouble according to Numerical Analysts
The Numerical Nonrobustness Phenomenon

- The trouble according to Numerical Analysts
  - “Pitfalls of Numerical F.P. Computation”
The Numerical Nonrobustness Phenomenon

The trouble according to Numerical Analysts

The trouble according to Computational Geometers
The Numerical Nonrobustness Phenomenon

- The trouble according to Numerical Analysts
- The trouble according to Computational Geometers
  - (crash, loop, inconsistency, etc)
  - Geometric/topological errors
  - “Bugbear” [Sedgewick], “Unsolvable problem” [Forrest]
The Numerical Nonrobustness Phenomenon

- The trouble according to Numerical Analysts
- The trouble according to Computational Geometers
- Computational Geometry attacks (1980-2000)
The Numerical Nonrobustness Phenomenon

The trouble according to Numerical Analysts

The trouble according to Computational Geometers

Computational Geometry attacks (1980-2000)

- Many approaches: E.g., what is a Line?
  - finite precision geometry [Sugihara, Hobby, Yao]
  - interval geometry Segal-Sequin, Guibas
  - consistent geometry [Fortune, Hopcroft]
  - improved arithmetic packages [Ottmann, van Wyk]
The Numerical Nonrobustness Phenomenon

- The trouble according to Numerical Analysts
- The trouble according to Computational Geometers
- Computational Geometry attacks (1980-2000)
- ... but what about Exact Computation?
Exact Geometric Computation (EGC)

- The EGC prescription
Exact Geometric Computation (EGC)

- The EGC prescription

  - Ensure all branches are error-free \( R_x \)

  (The “Take Home Message”)

- Be exact, but only where needed!
Exact Geometric Computation (EGC)

The EGC prescription

- Ensure all branches are error-free $^R_x$

“Most general/successful solution”

- So numerical approximations are allowed
- suffices to have an EGC number type
- Libraries: LEDA, CGAL, Core Library
Exact Geometric Computation (EGC)

The EGC prescription

- Ensure all branches are error-free

“Most general/successful solution”

... therein lies the seed of our next challenge
Barriers to EGC

EGC algorithms may not be Turing computable
Barriers to EGC

EGC algorithms may not be Turing computable

- E.g., transcendental functions (log, sin, exp, ...)

- The Zero Problem (Numerical Halting Problem!)
Barriers to EGC

- EGC algorithms may not be Turing computable
- EGC may be too inefficient
Barriers to EGC

- EGC algorithms may not be Turing computable
- EGC may be too inefficient
  - E.g., Euclidean shortest paths
Barriers to EGC

- EGC algorithms may not be Turing computable
- EGC may be too inefficient
- EGC requires full degeneracy analysis
Barriers to EGC

- EGC algorithms may not be Turing computable
- EGC may be too inefficient
- EGC requires full degeneracy analysis
  - Vor diagram of polyhedral objects (Open!)
Barriers to EGC

- EGC algorithms may not be Turing computable
- EGC may be too inefficient
- EGC requires full degeneracy
- Exact computation is unnecessary/inappropriate
Barriers to EGC

- EGC algorithms may not be Turing computable
- EGC may be too inefficient
- EGC requires full degeneracy
- Exact computation is unnecessary/inappropriate
  - E.g., robot motion planning, wireless routing
  - No physical constant is accurate to > 8 digits!
Barriers to EGC

- EGC algorithms may not be Turing computable
- EGC may be too inefficient
- EGC requires full degeneracy
- Exact computation is unnecessary/inappropriate
- ...beyond EGC?
Towards an alternative Computational Model

...but which?
Towards an alternative Computational Model

...but which? – let us look at examples!
Towards an alternative Computational Model

...but which? – let us look at examples!

A. Algebraic Problems – \( f(x) \in \mathbb{C}[x] \)

B. Combinatorial Problems – polyhedronal set \( \Omega \subseteq \mathbb{R}^d \)
Towards an alternative Computational Model

...but which? – let us look at examples!

A. Algebraic Problems – $f(x) \in \mathbb{C}[x]$

B. Combinatorial Problems – polyhedral set $\Omega \subseteq \mathbb{R}^d$

A.1 Root isolation and clustering

- [ISSAC’06,’09,’11,’12,’16, SNC’11, CiE’13]

A.2 Isotopic approximation of surfaces

- [ISSAC’08,SoCG’09,’12, SPM’12, ICMS’14]
Towards an alternative Computational Model

...but which? – let us look at examples!

A. Algebraic Problems – $f(x) \in \mathbb{C}[x]$

B. Combinatorial Problems – polyhedral set $\Omega \subset \mathbb{R}^d$

A.1 Root isolation and clustering
   – [ISSAC’06,’09,’11,’12,’16, SNC’11, CiE’13]

A.2 Isotopic approximation of surfaces
   – [ISSAC’08,SoCG’09,’12, SPM’12, ICMS’14]

B.1 Robot motion planning
   – [SoCG’13, WAFR’14, FAW’15]

B.2 Voronoi diagrams
   – [ISVD’13, SGP’16]
Towards an alternative Computational Model

...but which? – let us look at examples!

A. Algebraic Problems – \( f(x) \in \mathbb{C}[x] \)

B. Combinatorial Problems – polyhedral set \( \Omega \subseteq \mathbb{R}^d \)

A.1 Root isolation and clustering
  – [ISSAC'06,'09,'11,'12,'16, SNC'11, CiE'13]

A.2 Isotopic approximation of surfaces
  – [ISSAC'08, SoCG'09,'12, SPM'12, ICMS'14]

B.1 Robot motion planning
  – [SoCG'13, WAFR'14, FAW'15]

B.2 Voronoi diagrams
  – [ISVD'13, SGP'16]
Towards an alternative Computational Model

- All Subdivision Algorithms!
Towards an alternative Computational Model

All Subdivision Algorithms!

What is new?

– Introduce soft predicates
– Sidesteps the Zero Problem (and its complexity)
– Local and adaptive
– Implementable (usually has been implemented)
– Practical
Towards an alternative Computational Model

All Subdivision Algorithms!

What is new?

Joint work with:

– Roots & Clustering:
  V.Sharma, A.Eigenwillig, M.Sagraloff, R.Becker, J.Xu

– Curves & Surfaces:
  V.Sharma, G.Vegter, M.Burr, S.Choi, L.Lin

– Robot Motion Planning:

– Voronoi Diagrams:
  V.Sharma, J.-M.Lien, E.Papadopoulou, H.Bennett
II. Meshing

“The history of the zero recognition problem is somewhat confused by the fact that many people do not recognize it as a problem at all.”

— Daniel Richardson (1996)
What is Meshing?
Meshing Generation Problem

What is Meshing?

- Given a surface $S$, represented by $f(x, y, z) = 0$, to find a piecewise linear approximation $\tilde{S}$.
- first step in continuous $\rightarrow$ discrete conversion
What is Meshing?

Two Criteria of Meshing:

A. Topological Correctness
Meshing Generation Problem

What is Meshing?

Two Criteria of Meshing:

A. Topological Correctness

\[ \tilde{S} \text{ is isotopic to } S \]
Meshing Generation Problem

What is Meshing?

Two Criteria of Meshing:

A. Topological Correctness
   - \( \tilde{S} \) is isotopic to \( S \)

B. Geometrical Accuracy
   - \( d_H(S, \tilde{S}) < \varepsilon \) (for given \( \varepsilon > 0 \))
Subdivision Algorithms

What are subdivision algorithms?

– Generalized binary search, organized as a quadtree.
Subdivision Algorithms

What are subdivision algorithms?

– Generalized binary search, organized as a quadtree.

Figure: Mesh approximation of curve $f(X, Y) = Y^2 - X^2 + X^3 + 0.02 = 0$
Power of Subdivision

Easy to implement: only numbers, minimal algebra!
Power of Subdivision

- Easy to implement: only numbers, minimal algebra!
- Adaptivity: fast on easy instances
Power of Subdivision

- Easy to implement: only numbers, minimal algebra!
- Adaptivity: fast on easy instances
- Locality: can restrict to region of interest.
Power of Subdivision

Easy to implement: only numbers, minimal algebra!

Adaptivity: fast on easy instances

Locality: can restrict to region of interest.

Our Challenge: make them “resolution-exact” and efficient
Generic Subdivision Algorithm

- **INPUT:** \((f(x), B_0, \varepsilon)\)
- **OUTPUT:** Graph \(G = (V, E)\), \(\varepsilon\)-isotopic to \(f^{-1}(0) \cap B_0\).
  - Let \(Q_{in} \leftarrow \{B_0\}\) be a queue of boxes

(I) **SUBDIVISION PHASE:** \(Q_{sub} \leftarrow \text{SUBDIVIDE}(Q_{in})\)

(II) **REFINEMENT PHASE:** \(Q_{ref} \leftarrow \text{REFINE}(Q_{sub})\)

(III) **CONSTRUCTION PHASE:** \(G \leftarrow \text{CONSTRUCT}(Q_{ref})\)

Slightly modify this basic outline for many of our algorithms!
Generic Subdivision Algorithm

► **INPUT:** \((f(x), B_0, \varepsilon)\)
► **OUTPUT:** Graph \(G = (V, E), \varepsilon\)-isotopic to \(f^{-1}(0) \cap B_0\).
  ► Let \(Q_{in} \leftarrow \{B_0\}\) be a queue of boxes

(I) SUBDIVISION PHASE: \(Q_{sub} \leftarrow \text{SUBDIVIDE}(Q_{in})\)

(II) REFINEMENT PHASE: \(Q_{ref} \leftarrow \text{REFINE}(Q_{sub})\)

(III) CONSTRUCTION PHASE: \(G \leftarrow \text{CONSTRUCT}(Q_{ref})\)

► Slightly modify this basic outline for many of our algorithms!
Generic Subdivision Algorithm

**INPUT:** \((f(x), B_0, \varepsilon)\)

**OUTPUT:** Graph \(G = (V, E)\), \(\varepsilon\)-isotopic to \(f^{-1}(0) \cap B_0\).

- Let \(Q_{\text{in}} \leftarrow \{B_0\}\) be a queue of boxes

(I) **SUBDIVISION PHASE:** \(Q_{\text{sub}} \leftarrow \text{SUBDIVIDE}(Q_{\text{in}})\)

(II) **REFINEMENT PHASE:** \(Q_{\text{ref}} \leftarrow \text{REFINE}(Q_{\text{sub}})\)

(III) **CONSTRUCTION PHASE:** \(G \leftarrow \text{CONSTRUCT}(Q_{\text{ref}})\)

Slightly modify this basic outline for many of our algorithms!
Generic Subdivision Algorithm

- **INPUT:** \((f(x), B_0, \varepsilon)\)
- **OUTPUT:** Graph \(G = (V, E)\), \(\varepsilon\)-isotopic to \(f^{-1}(0) \cap B_0\).
  - Let \(Q_{in} \leftarrow \{B_0\}\) be a queue of boxes

1. **SUBDIVISION PHASE:** \(Q_{sub} \leftarrow \text{SUBDIVIDE}(Q_{in})\)
2. **REFINEMENT PHASE:** \(Q_{ref} \leftarrow \text{REFINE}(Q_{sub})\)
3. **CONSTRUCTION PHASE:** \(G \leftarrow \text{CONSTRUCT}(Q_{ref})\)

Slightly modify this basic outline for many of our algorithms!
Generic Subdivision Algorithm

- **INPUT:** \((f(x), B_0, \varepsilon)\)
- **OUTPUT:** Graph \(G = (V, E)\), \(\varepsilon\)-isotopic to \(f^{-1}(0) \cap B_0\).
  - Let \(Q_{in} \leftarrow \{B_0\}\) be a queue of boxes

(I) **SUBDIVISION PHASE:** \(Q_{sub} \leftarrow SUBDIVIDE(Q_{in})\)

(II) **REFINEMENT PHASE:** \(Q_{ref} \leftarrow REFINES(Q_{sub})\)

(III) **CONSTRUCTION PHASE:** \(G \leftarrow CONSTRUCT(Q_{ref})\)

- Slightly modify this basic outline for many of our algorithms!
Generic Subdivision Algorithm

- **INPUT:** \((f(x), B_0, \varepsilon)\)
- **OUTPUT:** Graph \(G = (V, E)\), \(\varepsilon\)-isotopic to \(f^{-1}(0) \cap B_0\).

  - Let \(Q_{in} \leftarrow \{B_0\}\) be a queue of boxes

(1) **SUBDIVISION PHASE:** \(Q_{sub} \leftarrow \text{SUBDIVIDE}(Q_{in})\)

(II) **REFINEMENT PHASE:** \(Q_{ref} \leftarrow \text{REFINE}(Q_{sub})\)

(III) **CONSTRUCTION PHASE:** \(G \leftarrow \text{CONSTRUCT}(Q_{ref})\)

- Slightly modify this basic outline for many of our algorithms!
E.g., Marching Cube (Lorenson & Cline)

(I) Subdivision Phase:
E.g., Marching Cube (Lorenson & Cline)

(i) Subdivision Phase:

– Subdivide until the width of each box is $\leq \epsilon$. 
E.g., Marching Cube (Lorenson & Cline)

(I) Subdivision Phase:

(II) Refinement Phase: NULL

(III) Construction Phase:
E.g., Marching Cube (Lorenson & Cline)

(I) Subdivision Phase:

(II) Refinement Phase: NULL

(III) Construction Phase:

– Evaluate sign of $f$ at corners of boxes.
– Insert vertices on segments that had different signs.
– Connect vertices in within each box.
E.g., Marching Cube (Lorenson & Cline)

(I) Subdivision Phase:

(II) Refinement Phase: NULL

(III) Construction Phase:

– Evaluate sign of \( f \) at corners of boxes.

– Insert vertices on segments that had different signs.

– Connect vertices in within each box.

Figure: Simple Connection Rule
E.g., Marching Cube (Lorenson & Cline)

(I) Subdivision Phase:

(II) Refinement Phase: NULL

(III) Construction Phase:

Cannot guarantee the topological correctness
E.g., Marching Cube (Lorenson & Cline)

(I) Subdivision Phase:

(II) Refinement Phase: NULL

(III) Construction Phase:

Cannot guarantee the topological correctness

Yet one of the most widely used algorithms!
Box Evaluation and Interval Arithmetic

**Goal**: make Marching Cube rigorous!

- BigFloats, Intervals, Boxes
Box Evaluation and Interval Arithmetic

Goal: make Marching Cube rigorous!

BigFloats, Intervals, Boxes

- Bigfloats or dyadic numbers:
  \[ \mathbb{Z}[1/2] := \{m2^n : m, n \in \mathbb{Z}\} \]

- n-Boxes:
  \[ \mathbb{R}^n \]
Goal: make Marching Cube rigorous!

BigFloats, Intervals, Boxes

Box functions $f$
Goal: make Marching Cube rigorous!

BigFloats, Intervals, Boxes

Box functions $\Box f$

- Given $f : \mathbb{R}^m \to \mathbb{R}$, we have

  $\Box f : \Box \mathbb{F}^n \to \Box \mathbb{F}$

such that

1. **Conservative**: $x \in B \Rightarrow f(x) \in \Box f(B)$.

2. **Convergence**: $\lim_{i \to \infty} B_i \to p$ implies

   $\lim_{i \to \infty} \Box f(B_i) \to f(p)$
Box Evaluation and Interval Arithmetic

Goal: make Marching Cube rigorous!

BigFloats, Intervals, Boxes

Box functions $f$

Box functions are easy to construct!

– good ones are non-trivial (e.g., centered form)
Some Box Predicates

Exclusion Predicate “C0”

- $C_0(B) : 0 \notin \square f(B)$
Some Box Predicates

**Exclusion Predicate “C0”**

- C0(B): \[ 0 \notin \Box f(B) \]

**Normal Variation Predicate “C1”**

- C1(B): \[ 0 \notin \Box f_x(B)^2 + \Box f_y(B)^2 \]
Some Box Predicates

- Exclusion Predicate “C0”
  - $\text{C0}(B): \quad 0 \not\in f(B)$

- Normal Variation Predicate “C1”
  - $\text{C1}(B): \quad 0 \not\in f_x(B)^2 + f_y(B)^2$

- Parametrizability Predicate “Cxy”
  - $\text{Cxy}(B): \quad 0 \not\in f_x(B)\quad\text{or}\quad 0 \not\in f_y(B)$
Softness Concept: Local Non-Isotopy

“Local isotopy” (i.e., correct within each box)

- Based on exact predicates
- E.g., Snyder, Collins-Krandick
Softness Concept: Local Non-Isotopy

“Local isotopy” (i.e., correct within each box)

- Based on exact predicates
- E.g., Snyder, Collins-Krandick

“Local Isotopy implies Global Isotopy”

- Widely exploited
- Why it is a poor algorithmic idea...
Softness Concept: Local Non-Isotopy

“Local isotopy” (i.e., correct within each box)
- Based on exact predicates
- E.g., Snyder, Collins-Krandick

Key to soft approach
- “do not take boxes too seriously”
- Allow isotopic curves that respect box corners!
- Incursions and excursions allowed
Softness Concept: Local Non-Isotopy

“Local isotopy” (i.e., correct within each box)
- Based on exact predicates
- E.g., Snyder, Collins-Krandick

Key to soft approach
- “do not take boxes too seriously”
- Allow isotopic curves that respect box corners!
- Incursions and excursions allowed
Plantinga and Vegter’s Algorithm

- **Subdivision Phase:** For each box $B$:
  - $C_0(B) \Rightarrow$ discard
  - $\neg C_1(B) \Rightarrow$ subdivide $B$

- **Refinement Phase:** Balance!

- **Construction Phase:**

![Figure: Extended Rules](image-url)
Generalizations and Improvements

Issues

– improved Cxy-predicate
– boundary issues
– singularities
– anisotropic subdivision
– Work of Michael Burr (amortized analysis)

––––––––––––

– co-dimension $d > 1$
– non-Euclidean space (e.g., $SO(3), \mathbb{P}^n(\mathbb{C})$
– soft labels
Generalizations and Improvements

- Just Replace C1-predicate by Cxy-predicate in PV
  - What can go wrong?

The Obstruction

Figure: Elongated ellipse
III. Roots

“Eventually, the topic [...of proving non-zeroness...] takes over the whole subject [...of Transcendental Number Theory...]”

— David Masser (2000)
Introduction

Fundamental Theorem of Algebra (FTA)

*Every complex polynomial of degree \( n \) has exactly \( n \) complex roots, counted with multiplicity*
Introduction

Fundamental Theorem of Algebra (FTA)

Every complex polynomial of degree \( n \) has exactly \( n \) complex roots, counted with multiplicity

Figure: Roots of \( F(z) = z^{12} - 1 \)
Introduction

Root Approximation:
- compute an $\varepsilon$-disc for each root

Root Isolation:
- compute disjoint discs each containing a unique root
- Focus on root isolation (harder problem)
Classical History of FTA

Root Finding

has roots (sic) in antiquity (Babylonians, 1900 B.C.)

Associated with the “pantheon” of mathematicians

- Descartes, Newton, d’Alembert, Euler, Lagrange,
- Laplace, Gauss, Fourier, Sturm, Weierstrass,
- Vincent, Obreshkorff, Ostrowski, Brouwer, Weyl, Henrici, ...
Classical History of FTA

Root Finding

has roots (sic) in antiquity (Babylonians, 1900 B.C.)

Associated with the “pantheon” of mathematicians

- Descartes, Newton, d’Alembert, Euler, Lagrange,
- Laplace, Gauss, Fourier, Sturm, Weierstrass,
- Vincent, Obreshkoff, Ostrowski, Brouwer, Weyl, Henrici, ... 

Hugh and diverse classical literature

- Weierstrass (1891) (basis of Durand-Kerner-Aberth)
- Weyl (1924) (subdivision approach)
Modern History of FTA

Modern FTA research may be precisely dated to 1981-2:
Modern History of FTA

Modern FTA research may be precisely dated to 1981-2:

* **Arnold Schönhage (1982):**
  
  *Fundamental Theorem of Algebra in Terms of Computational Complexity*
  
  – Unpublished!

* **Steve Smale (1981):**

  *Fundamental Theorem of Algebra and Complexity Theory*
  
  – Bulletin (N.S.) of the AMS.
Modern History of FTA

Modern FTA research may be precisely dated to 1981-2:
Each paper initiated a line of research, active to this day
Modern History of FTA

- Modern FTA research may be precisely dated to 1981-2:
- Each paper initiated a line of research, active to this day

Useful terms:

- **Benchmark Problem**: isolate all roots of an integer polynomial

- **Near-Optimal Bound**: [Pan-Schönhage 2002]

  **THEOREM**: Benchmark Problem has bit complexity $\tilde{O}(n^2 L)$

  (where $n=$degree, $L=$coefficient bitsize)

- **Global vs. Local root isolation**
Modern History of FTA

Modern FTA research may be precisely dated to 1981-2:
Each paper initiated a line of research, active to this day

Useful terms:
Near-Optimal Algorithm: not implemented in 20 years
  – WHY? It is described in Real RAM model
Modern History of FTA

Pan (2002) [p. 705]:

... since Schönhage (1982b) already has 72 pages, and Kirrinnis (1998) has 67 pages, this ruled out a self-contained presentation of our root-finding algorithm.

Again:

Our algorithms are quite involved, and their implementation would require a non-trivial work, incorporating numerous known implementation techniques and tricks (Bini and Fiorentino, 2000; Fortune, 2001; Bini and Pan, to appear). We do not touch this vast domain here and just briefly comment on the precision of computing.
Modern History of FTA

Modern FTA research may be precisely dated to 1981-2:
Each paper initiated a line of research, active to this day

Useful terms:

Near-Optimal Algorithm: not implemented in 20 years
What is implemented for exact root finders?
   – (in Computer Algebra Systems such as Maple, etc)
   – Subdivision algorithms!
      (Descartes methods) [Rouillier-Zimmerman (2004)]
Cf. MPSolve [Bini-Florentini]
Modern History of FTA

Can practice meet theory in Root Isolation?

– I.e., Can subdivision algorithms reach near-optimal bounds?
– Independent of Schönhage’s “Circle Splitting Method”
Modern History of FTA

Can practice meet theory in Root Isolation?

Last decade: lots of progress in Subdivision methods

– (esp. complexity analysis)

– Finally, for Real Benchmark, yes!

[Sagraloff (2012), Sagraloff-Mehlhorn (2015)]

[Kobel-Rouillier-Sagraloff, ISSAC’16]

– Last year, for Complex Benchmark, also yes!

[Becker-Sagraloff-Sharma-Y (JSC’17)],

[Becker-Sagraloff-Sharma-Xu-Y (ISSAC’16)]
Soft Ideas in Near-Optimal Complex Roots

- Pellet’s Test (1831) for polynomial \( f(z) \)
- Let \( \Delta = \Delta(m, r) \subseteq \mathbb{C} \) be a disc.
- **THEOREM:** If
  \[
  T_k(m, r, K) = T_k(\Delta, K) : \sum_{i \neq k} \left| \frac{f^{(i)}(m) r^{i-k} k!}{f^{(k)}(m) i!} \right| < \frac{1}{K}
  \]
  then
  \[
  #(\Delta(m, r)) = k.
  \]
- where \( K \geq 1 \) and \( 0 \leq k < n \) (\( = \deg(f) \))
- and \( \#\Delta(m, r) \) is number of roots in \( \Delta(m, r) \).
Soft Ideas in Near-Optimal Complex Roots

Pellet’s Test (1831) for polynomial $f(z)$

What if Pellet’s test fails? Need converse!

– Soft predicates are always one sided
Soft Ideas in Near-Optimal Complex Roots

Pellet’s Test (1831) for polynomial $f(z)$

What if Pellet’s test fails? Need converse!

THEOREM (Converse of Pellet):

If \[ \#(\frac{1}{11n}\Delta) = \#(64n^3\Delta) \]

then \[ T_k(\Delta, 3/2) \text{ succeeds.} \]
Soft Ideas in Near-Optimal Complex Roots

- Pellet’s Test (1831) for polynomial $f(z)$
- What if Pellet’s test fails? Need converse!
- Graeffe-Pellet Test $T_k^G(\Delta, 3/2)$
  
  \[ \text{Let } \rho_1 = \frac{2\sqrt{2}}{3} \simeq 0.94 \text{ and } \rho_2 = \frac{4}{3} < 1.4 \]

  If $\#(\rho_1 \Delta) = \#(\rho_2 \Delta)$

  then $T_k^G(\Delta, 3/2)$ succeeds.

- Why $\rho_1, \rho_2$?
Soft Ideas in Near-Optimal Complex Roots

- **Pellet’s Test (1831)** for polynomial $f(z)$

- What if Pellet’s test fails? Need converse!

- **Graeffe-Pellet Test** $T_k^G(\Delta, 3/2)$

- **Soft Comparison** $E_{\ell} : E_r$
  - Returns “less than” or “greater than” correctly
  - But if returns “unknown” then $\frac{2}{3} < \left| \frac{E_r}{E_{\ell}} \right| < \frac{3}{2}$
Soft Ideas in Near-Optimal Complex Roots

- **Pellet’s Test (1831)** for polynomial $f(z)$
- What if Pellet’s test fails? Need converse!
- **Graeffe-Pellet Test** $T^G_k(\Delta, 3/2)$
- **Soft Comparison** $E_L : E_r$
- **Soft Graeffe-Pellet Test** $\tilde{T}^G_k(\Delta)$
  - Evaluate $T^G_k(\Delta, 3/2)$ using soft comparison,
  - Treat “unknown” as failure.
Soft Ideas in Near-Optimal Complex Roots

- Pellet’s Test (1831) for polynomial $f(z)$
- What if Pellet’s test fails? Need converse!
- Graeffe-Pellet Test $T^G_k(\Delta, 3/2)$
- Soft Comparison $E_\ell : E_r$
- Soft Graeffe-Pellet Test $\tilde{T}^G_k(\Delta)$
  - Newton is conditional, bisection unconditional.
  - Maintain a state (speed of Newton acceleration)
  - If cluster of $k$ roots, apply order-$k$ Newton
  - If Newton fails, apply bisection
Soft Ideas in Near-Optimal Complex Roots

Putting it all together

- Apply $\tilde{T}_0^G(\Delta)$ to each box $B$
- Discard $B$ if test succeeds,
  else keep it ($2B$ has roots!)
- Maintain connected components of kept boxes
- When components becomes well-isolated, use $\tilde{T}_*^G(\Delta)$ to determine $k$ for Newton-Bisection
Beyond Root Isolation: Clustering

Most general form of root isolation:

- coefficients of $f(z)$ given to any desired approximation
- Typically, coefficients are algebraic numbers
- E.g., solving polynomial systems
- root isolation must be generalized! (why?)
Beyond Root Isolation: Clustering

Most general form of root isolation:

**Root Clustering Problem:**

- Output a collection of pairwise disjoint discs, each with the number of roots in the disc
Beyond Root Isolation: Clustering

- Most general form of root isolation:

- **Root Clustering Problem:**

- Question of Naturalness

Figure: Red cluster is not natural, Blue cluster is natural
Beyond Root Isolation: Clustering

- Most general form of root isolation:

- **Root Clustering Problem:**

- Question of Naturalness

- **Complexity Analysis**
  - traditional parameters are “synthetic”
  - we use “local geometric” parameters
  - a discriminant substitute $D^*(f)$
Beyond Root Isolation: Clustering

- Most general form of root isolation:

**Root Clustering Problem:**

- Question of Naturalness

- Complexity Analysis

- Future work
  - “Most general setting possible for roots”
  - Analytic Roots with complexity analysis
    (cf. [CiE’2013])
  - Polynomial systems
IV. Motion Planning
Motion Planning

Youtube video presented in SoCG’2016.
Motion Planning

- What is Resolution-Exactness?
- Soft Predicates
- Soft Subdivision Search Framework
V. Voronoi Diagrams
Why Can’t we Compute Voronoi Diagrams in 3D?

- Huge variety of Voronoi diagrams
Why Can’t we Compute Voronoi Diagrams in 3D?

- Huge variety of Voronoi diagrams
  - many lies outside the Exact Computational model
  - but there is one classic Voronoi diagram we want ...

Vor Diagram of two Lines in 3D [Hemmer-Setter-Halperin, 2010]
Why Can’t we Compute Voronoi Diagrams in 3D?

- Huge variety of Voronoi diagrams
- A 20-Year Old “Voronoi Quest”
Why Can’t we Compute Voronoi Diagrams in 3D?

- Huge variety of Voronoi diagrams
- A 20-Year Old “Voronoi Quest”
  - “Major milestone” [Hemmer, Setter, Halperin (2010)]
Why Can’t we Compute Voronoi Diagrams in 3D?

- Huge variety of Voronoi diagrams
- A 20-Year Old “Voronoi Quest”
- Theorem of Everett, Gillot, Lazard, Pougheet (2009)
Why Can’t we Compute Voronoi Diagrams in 3D?

- Huge variety of Voronoi diagrams
- A 20-Year Old “Voronoi Quest”
- Theorem of Everett, Gillot, Lazard, Pougeet (2009)
  * a non-singular quartic if ”general position”
  * a cubic and a line if on hyperboloid
  * a nodal quartic if all parallel to a plane
  * 1 or 2 parabolas/hyperbola if one pair coplanar
  * 0 to 4 lines if two pairs of coplanar lines

"General position": Pairwise skew, not parallel to a plane, nor on surface of hyperboloid
Why Can’t we Compute Voronoi Diagrams in 3D?

- Huge variety of Voronoi diagrams
- A 20-Year Old “Voronoi Quest”
- Theorem of Everett, Gillot, Lazard, Pougeet (2009)
- What this tells us
  - **Voronoi Quest:**
    - Good news: Hardest of 10 Cases solved
    - Bad news: 9 other cases
  - We must explore soft solutions in general
Why Can’t we Compute Voronoi Diagrams in 3D?

- Huge variety of Voronoi diagrams
- A 20-Year Old “Voronoi Quest”
- Theorem of Everett, Gillot, Lazard, Pougeet (2009)
- What this tells us
- SGP Work with Huck and Evanthia
  - Application to anisotropic Voronoi diagrams
  - General minimization diagrams
  - Some implementation
  - What is the notion of a Voronoi cluster?
  - How to compute such clusters?
VI. Conclusion
Conclusion

- General outlook
Conclusion

- General outlook
  - give up exact model (Real RAM, BSS Model)
  - need a numerical computation model
  - main algorithmic paradigm: subdivision/iteration
Conclusion

- General outlook
- Broad consequences
Conclusion

General outlook

Broad consequences

– scope of computational geometry vastly broadened
– non-linear geometry becomes accessible
– unsolvable problems becomes feasible
– algorithms are implementable as well as practical
Conclusion

General outlook

Broad consequences

Research is wide open

- develop new algorithms for old CG problems
- produce complexity analysis of such algorithms
- theory of real computation and continuous complexity
Thanks for Listening!

“Algebra is generous, she often gives more than is asked of her.”

— Jean Le Rond D’Alembert (1717–83)

“To Generalize is to be an Idiot. To Particularize is the Alone Distinction of Merit – General Knowledges are those Knowledges that Idiots possess.”

— William Blake (1757 – 1827)

Annotations to Sir Joshua Reynolds’s Discourses, pp. xvii – xcvi