Tutorial: Exact Numerical Computation in Algebra and Geometry

Chee K. Yap

Courant Institute of Mathematical Sciences
New York University

and

Korea Institute of Advanced Study (KIAS)
Seoul, Korea

34th ISSAC, July 28–31, 2009

Tutorial: Exact Numerical Computation in Algebra and Geometry

- Many problems in Computational Science & Engineering (CS&E) are defined on the continuum. Standard algorithms for these problems are numerical and approximate. Their computational techniques include iteration, subdivision, and approximation. Such techniques are rarely seen in exact or algebraic algorithms. In this tutorial, we discuss a mode of computation called Exact Numerical Computation (ENC) that achieves exactness through numerical approximation. Through ENC, we naturally incorporate iteration, subdivision and approximation into our design of algorithms for computer algebra and computational geometry. Such algorithms are both novel and practical. This tutorial on ENC is divided into three equal parts:
  (a) ENC and Zero Problems
  (b) Explicitization and Subdivision Algorithms
  (c) Complexity Analysis of Adaptivity

Overview of Tutorial

- Background is algebraic and geometric computation
- Motivation: much of computing world (CS&E) is continuous
- But Theoretical Computer Science has gone completely discrete
- The discrete view alone is inadequate for CS&E.
- What role for exact computation in the continuum?
- Geometric insights holds the key
- Exact Numerical Computation (ENC) is a proposed synthesis
- Lecture in 3 parts
  - (a) ENC and Zero Problems
  - (b) Explicitization and Subdivision Algorithms
  - (c) Complexity Analysis of Adaptivity

Overview of Tutorial

- Background is algebraic and geometric computation
- Motivation: much of computing world (CS&E) is continuous
- But Theoretical Computer Science has gone completely discrete
- The discrete view alone is inadequate for CS&E.
- What role for exact computation in the continuum?
- Geometric insights holds the key
- Exact Numerical Computation (ENC) is a proposed synthesis
- Lecture in 3 parts
  - (a) ENC and Zero Problems
  - (b) Explicitization and Subdivision Algorithms
  - (c) Complexity Analysis of Adaptivity

Overview of Tutorial

- Background is algebraic and geometric computation
- Motivation: much of computing world (CS&E) is continuous
- But Theoretical Computer Science has gone completely discrete
- The discrete view alone is inadequate for CS&E.
- What role for exact computation in the continuum?
- Geometric insights holds the key
- Exact Numerical Computation (ENC) is a proposed synthesis
- Lecture in 3 parts
  - (a) ENC and Zero Problems
  - (b) Explicitization and Subdivision Algorithms
  - (c) Complexity Analysis of Adaptivity

Overview of Tutorial

- Background is algebraic and geometric computation
- Motivation: much of computing world (CS&E) is continuous
- But Theoretical Computer Science has gone completely discrete
- The discrete view alone is inadequate for CS&E.
- What role for exact computation in the continuum?
- Geometric insights holds the key
- Exact Numerical Computation (ENC) is a proposed synthesis
- Lecture in 3 parts
  - (a) ENC and Zero Problems
  - (b) Explicitization and Subdivision Algorithms
  - (c) Complexity Analysis of Adaptivity

Overview of Tutorial

- Background is algebraic and geometric computation
- Motivation: much of computing world (CS&E) is continuous
- But Theoretical Computer Science has gone completely discrete
- The discrete view alone is inadequate for CS&E.
- What role for exact computation in the continuum?
- Geometric insights holds the key
- Exact Numerical Computation (ENC) is a proposed synthesis
- Lecture in 3 parts
  - (a) ENC and Zero Problems
  - (b) Explicitization and Subdivision Algorithms
  - (c) Complexity Analysis of Adaptivity

Overview of Tutorial

- Background is algebraic and geometric computation
- Motivation: much of computing world (CS&E) is continuous
- But Theoretical Computer Science has gone completely discrete
- The discrete view alone is inadequate for CS&E.
- What role for exact computation in the continuum?
- Geometric insights holds the key
- Exact Numerical Computation (ENC) is a proposed synthesis
- Lecture in 3 parts
  - (a) ENC and Zero Problems
  - (b) Explicitization and Subdivision Algorithms
  - (c) Complexity Analysis of Adaptivity

Overview of Tutorial

- Background is algebraic and geometric computation
- Motivation: much of computing world (CS&E) is continuous
- But Theoretical Computer Science has gone completely discrete
- The discrete view alone is inadequate for CS&E.
- What role for exact computation in the continuum?
- Geometric insights holds the key
- Exact Numerical Computation (ENC) is a proposed synthesis
- Lecture in 3 parts
  - (a) ENC and Zero Problems
  - (b) Explicitization and Subdivision Algorithms
  - (c) Complexity Analysis of Adaptivity

Overview of Tutorial

- Background is algebraic and geometric computation
- Motivation: much of computing world (CS&E) is continuous
- But Theoretical Computer Science has gone completely discrete
- The discrete view alone is inadequate for CS&E.
- What role for exact computation in the continuum?
- Geometric insights holds the key
- Exact Numerical Computation (ENC) is a proposed synthesis
- Lecture in 3 parts
  - (a) ENC and Zero Problems
  - (b) Explicitization and Subdivision Algorithms
  - (c) Complexity Analysis of Adaptivity

Overview of Tutorial

- Background is algebraic and geometric computation
- Motivation: much of computing world (CS&E) is continuous
- But Theoretical Computer Science has gone completely discrete
- The discrete view alone is inadequate for CS&E.
- What role for exact computation in the continuum?
- Geometric insights holds the key
- Exact Numerical Computation (ENC) is a proposed synthesis
- Lecture in 3 parts
  - (a) ENC and Zero Problems
  - (b) Explicitization and Subdivision Algorithms
  - (c) Complexity Analysis of Adaptivity

Overview of Tutorial

- Background is algebraic and geometric computation
- Motivation: much of computing world (CS&E) is continuous
- But Theoretical Computer Science has gone completely discrete
- The discrete view alone is inadequate for CS&E.
- What role for exact computation in the continuum?
- Geometric insights holds the key
- Exact Numerical Computation (ENC) is a proposed synthesis
- Lecture in 3 parts
  - (a) ENC and Zero Problems
  - (b) Explicitization and Subdivision Algorithms
  - (c) Complexity Analysis of Adaptivity

Overview of Tutorial

- Background is algebraic and geometric computation
- Motivation: much of computing world (CS&E) is continuous
- But Theoretical Computer Science has gone completely discrete
- The discrete view alone is inadequate for CS&E.
- What role for exact computation in the continuum?
- Geometric insights holds the key
- Exact Numerical Computation (ENC) is a proposed synthesis
- Lecture in 3 parts
  - (a) ENC and Zero Problems
  - (b) Explicitization and Subdivision Algorithms
  - (c) Complexity Analysis of Adaptivity

PART 1

Exact Numeric Computation and the Zero Problem

"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)

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

— JEAN LE ROND D'ALEMBERT (1717-83)

Coming Up Next

Introduction to Geometric Computation

- PUZZLE 1:
  Is Geometry discrete or continuous?
- PUZZLE 2:
  How come numbers do not arise in Computational Geometry?

Introduction to Geometric Computation

- PUZZLE 1:
  Is Geometry discrete or continuous?
- PUZZLE 2:
  How come numbers do not arise in Computational Geometry?

Introduction to Geometric Computation

- PUZZLE 1:
  Is Geometry discrete or continuous?
- PUZZLE 2:
  How come numbers do not arise in Computational Geometry?

Introduction to Geometric Computation

- PUZZLE 1:
  Is Geometry discrete or continuous?
- PUZZLE 2:
  How come numbers do not arise in Computational Geometry?

What is Computational Geometry?

Geometric Objects

- Prototype: Points, Lines, Circles (Euclidean Geometry)
- Arrangement of hyperplanes and hypersurfaces
- Zero sets and their Singularities
- Semi-algebraic sets
- Non-algebraic sets
- Geometric complexes

Geometric Problems

- Constructing geometric objects
- Searching in geometric complexes or structures

What is Computational Geometry?

Geometric Objects

- Prototype: Points, Lines, Circles (Euclidean Geometry)
- Arrangement of hyperplanes and hypersurfaces
- Zero sets and their Singularities
- Semi-algebraic sets
- Non-algebraic sets
- Geometric complexes

Geometric Problems

- Constructing geometric objects
- Searching in geometric complexes or structures

What is Computational Geometry?

Geometric Objects

- Prototype: Points, Lines, Circles (Euclidean Geometry)
- Arrangement of hyperplanes and hypersurfaces
- Zero sets and their Singularities
- Semi-algebraic sets
- Non-algebraic sets
- Geometric complexes

Geometric Problems

- Constructing geometric objects
- Searching in geometric complexes or structures

What is Computational Geometry?

Geometric Objects

- Prototype: Points, Lines, Circles (Euclidean Geometry)
- Arrangement of hyperplanes and hypersurfaces
- Zero sets and their Singularities
- Semi-algebraic sets
- Non-algebraic sets
- Geometric complexes

Geometric Problems

- Constructing geometric objects
- Searching in geometric complexes or structures

What is Computational Geometry?

Geometric Objects

- Prototype: Points, Lines, Circles (Euclidean Geometry)
- Arrangement of hyperplanes and hypersurfaces
- Zero sets and their Singularities
- Semi-algebraic sets
- Non-algebraic sets
- Geometric complexes

Geometric Problems

- Constructing geometric objects
- Searching in geometric complexes or structures

What is Computational Geometry?

Geometric Objects

- Prototype: Points, Lines, Circles (Euclidean Geometry)
- Arrangement of hyperplanes and hypersurfaces
- Zero sets and their Singularities
- Semi-algebraic sets
- Non-algebraic sets
- Geometric complexes

Geometric Problems

- Constructing geometric objects
- Searching in geometric complexes or structures

What is Computational Geometry?

Geometric Objects

- Prototype: Points, Lines, Circles (Euclidean Geometry)
- Arrangement of hyperplanes and hypersurfaces
- Zero sets and their Singularities
- Semi-algebraic sets
- Non-algebraic sets
- Geometric complexes

Geometric Problems

- Constructing geometric objects
- Searching in geometric complexes or structures

What is Computational Geometry?

Geometric Objects

- Prototype: Points, Lines, Circles (Euclidean Geometry)
- Arrangement of hyperplanes and hypersurfaces
- Zero sets and their Singularities
- Semi-algebraic sets
- Non-algebraic sets
- Geometric complexes

Geometric Problems

- Constructing geometric objects
- Searching in geometric complexes or structures

What is Computational Geometry?

Geometric Objects

- Prototype: Points, Lines, Circles (Euclidean Geometry)
- Arrangement of hyperplanes and hypersurfaces
- Zero sets and their Singularities
- Semi-algebraic sets
- Non-algebraic sets
- Geometric complexes

Geometric Problems

- Constructing geometric objects
- Searching in geometric complexes or structures

What is Computational Geometry?

Geometric Objects

- Prototype: Points, Lines, Circles (Euclidean Geometry)
- Arrangement of hyperplanes and hypersurfaces
- Zero sets and their Singularities
- Semi-algebraic sets
- Non-algebraic sets
- Geometric complexes

Geometric Problems

- Constructing geometric objects
- Searching in geometric complexes or structures

What is Computational Geometry?

Geometric Objects

- Prototype: Points, Lines, Circles (Euclidean Geometry)
- Arrangement of hyperplanes and hypersurfaces
- Zero sets and their Singularities
- Semi-algebraic sets
- Non-algebraic sets
- Geometric complexes

Geometric Problems

- Constructing geometric objects
- Searching in geometric complexes or structures

Dual Descriptions of Geometry

Where do Geometric Objects Live?

- As Points in Parametric Space $\mathcal{P}$
  - E.g., for lines given by $L(a,b,c) := aX + bY + c = 0$, the space is $\mathcal{P} := \{(a,b,c) : a^2 + b^2 > 0\} \subseteq \mathbb{R}^3$.
- As Loci in Ambient Space $\mathcal{A}$
  - E.g., Locus of the $Line(1,-2,0)$ is the set $\{(x,y) \in \mathbb{R}^2 : x - 2y = 0\} \subseteq \mathcal{A} = \mathbb{R}^2$.
- More involved example: Cell Complexes (in the sense of algebraic topology)

Dual Descriptions of Geometry

Where do Geometric Objects Live?

- As Points in Parametric Space $\mathcal{P}$
  - E.g., for lines given by $L(a,b,c) := aX + bY + c = 0$, the space is $\mathcal{P} := \{(a,b,c) : a^2 + b^2 > 0\} \subseteq \mathbb{R}^3$.
- As Loci in Ambient Space $\mathcal{A}$
  - E.g., Locus of the $Line(1,-2,0)$ is the set $\{(x,y) \in \mathbb{R}^2 : x - 2y = 0\} \subseteq \mathcal{A} = \mathbb{R}^2$.
- More involved example: Cell Complexes (in the sense of algebraic topology)

Dual Descriptions of Geometry

Where do Geometric Objects Live?

- As Points in Parametric Space $\mathcal{P}$
  - E.g., for lines given by $L(a,b,c) := aX + bY + c = 0$, the space is $\mathcal{P} := \{(a,b,c) : a^2 + b^2 > 0\} \subseteq \mathbb{R}^3$.
- As Loci in Ambient Space $\mathcal{A}$
  - E.g., Locus of the *Line*$(1,-2,0)$ is the set $\{(x,y) \in \mathbb{R}^2 : x - 2y = 0\} \subseteq \mathcal{A} = \mathbb{R}^2$.
- More involved example: Cell Complexes (in the sense of algebraic topology)

Dual Descriptions of Geometry

Where do Geometric Objects Live?

- As Points in Parametric Space $\mathcal{P}$
  - E.g., for lines given by $L(a,b,c) := aX + bY + c = 0$, the space is $\mathcal{P} := \{(a,b,c) : a^2 + b^2 > 0\} \subseteq \mathbb{R}^3$.
- As Loci in Ambient Space $\mathcal{A}$
  - E.g., Locus of the $Line(1,-2,0)$ is the set $\{(x,y) \in \mathbb{R}^2 : x - 2y = 0\} \subseteq \mathcal{A} = \mathbb{R}^2$.
- More involved example: Cell Complexes (in the sense of algebraic topology)

Dual Descriptions of Geometry

Where do Geometric Objects Live?

- As Points in Parametric Space $\mathcal{P}$
  - E.g., for lines given by $L(a,b,c) := aX + bY + c = 0$, the space is $\mathcal{P} := \{(a,b,c) : a^2 + b^2 > 0\} \subseteq \mathbb{R}^3$.
- As Loci in Ambient Space $\mathcal{A}$
  - E.g., Locus of the $Line(1,-2,0)$ is the set $\{(x,y) \in \mathbb{R}^2 : x - 2y = 0\} \subseteq \mathcal{A} = \mathbb{R}^2$.
- More involved example: Cell Complexes (in the sense of algebraic topology)

Dual Descriptions of Geometry

Where do Geometric Objects Live?

- As Points in Parametric Space $\mathcal{P}$
  - E.g., for lines given by $L(a,b,c) := aX + bY + c = 0$, the space is $\mathcal{P} := \{(a,b,c) : a^2 + b^2 > 0\} \subseteq \mathbb{R}^3$.
- As Loci in Ambient Space $\mathcal{A}$
  - E.g., Locus of the $Line(1,-2,0)$ is the set $\{(x,y) \in \mathbb{R}^2 : x - 2y = 0\} \subseteq \mathcal{A} = \mathbb{R}^2$.
- More involved example: Cell Complexes (in the sense of algebraic topology)

Dual Descriptions of Geometry

Where do Geometric Objects Live?

- As Points in Parametric Space $\mathcal{P}$
  - E.g., for lines given by $L(a,b,c) := aX + bY + c = 0$, the space is $\mathcal{P} := \{(a,b,c) : a^2 + b^2 > 0\} \subseteq \mathbb{R}^3$.
- As Loci in Ambient Space $\mathcal{A}$
  - E.g., Locus of the $Line(1,-2,0)$ is the set $\{(x,y) \in \mathbb{R}^2 : x - 2y = 0\} \subseteq \mathcal{A} = \mathbb{R}^2$.
- More involved example:
  Cell Complexes (in the sense of algebraic topology)

Computation: Geometric vs. Algebraic

Where is the Computation?

- **Algebraic Computation:** in parameter space $\mathcal{P}$
  - E.g., Gröbner bases
  - Polynomial manipulation, Expensive (double exponential time)
- **Geometric Computation:** in ambient space $\mathcal{A}$
  - E.g., Finding Zeros of Polynomials in $\mathbb{R}^n$
  - Numerical, Combinatorial, Adaptive (single exponential time)

Computation: Geometric vs. Algebraic

Where is the Computation?

- Algebraic Computation: in parameter space $\mathcal{P}$
  - E.g., Gröbner bases
  - Polynomial manipulation, Expensive (double exponential time)
- Geometric Computation: in ambient space $\mathcal{A}$
  - E.g., Finding Zeros of Polynomials in $\mathbb{R}^n$
  - Numerical, Combinatorial, Adaptive (single exponential time)

Computation: Geometric vs. Algebraic

Where is the Computation?

- Algebraic Computation: in parameter space $\mathcal{P}$
  - E.g., Gröbner bases
  - Polynomial manipulation, Expensive (double exponential time)
- Geometric Computation: in ambient space $\mathcal{A}$
  - E.g., Finding Zeros of Polynomials in $\mathbb{R}^n$
  - Numerical, Combinatorial, Adaptive (single exponential time)

Computation: Geometric vs. Algebraic

Where is the Computation?

- Algebraic Computation: in parameter space $\mathcal{P}$
  - E.g., Gröbner bases
  - Polynomial manipulation, Expensive (double exponential time)
- Geometric Computation: in ambient space $\mathcal{A}$
  - E.g., Finding Zeros of Polynomials in $\mathbb{R}^n$
  - Numerical, Combinatorial, Adaptive (single exponential time)

Computation: Geometric vs. Algebraic

Where is the Computation?

- Algebraic Computation: in parameter space $\mathcal{P}$
  - E.g., Gröbner bases
  - Polynomial manipulation, Expensive (double exponential time)
- Geometric Computation: in ambient space $\mathcal{A}$
  - E.g., Finding Zeros of Polynomials in $\mathbb{R}^n$
  - Numerical, Combinatorial, Adaptive (single exponential time)

Computation: Geometric vs. Algebraic

Where is the Computation?

- Algebraic Computation: in parameter space $\mathcal{P}$
  - E.g., Gröbner bases
  - Polynomial manipulation, Expensive (double exponential time)
- Geometric Computation: in ambient space $\mathcal{A}$
  - E.g., Finding Zeros of Polynomials in $\mathbb{R}^n$
  - Numerical, Combinatorial, Adaptive (single exponential time)

(Contd.) Computation: Geometric vs. Algebraic

Answer to PUZZLE 1: “BOTH”

- Geometry is discrete (in $\mathcal{P}$) (algebraic computation)
- Geometry is continuous (in $\mathcal{A}$) (analytic computation)

Actions in the Ambient Space

- Geometric Relationships on different Object types arise in $\mathcal{A}$
  - E.g., Point is ON/LEFT/RIGHT of a Line
- Analytic properties of Objects comes from their loci
  - E.g., Proximity, Approximations, Isotopy, etc

(Contd.) Computation: Geometric vs. Algebraic

Answer to PUZZLE 1: "BOTH"

- Geometry is discrete (in $\mathcal{P}$) (algebraic computation)
- Geometry is continuous (in $\mathcal{A}$) (analytic computation)

Actions in the Ambient Space

- Geometric Relationships on different Object types arise in $\mathcal{A}$
  - E.g., Point is ON/LEFT/RIGHT of a Line
- Analytic properties of Objects comes from their loci
  - E.g., Proximity, Approximations, Isotopy, etc

(Contd.) Computation: Geometric vs. Algebraic

Answer to PUZZLE 1: "BOTH"

- Geometry is discrete (in $\mathcal{P}$) (algebraic computation)
- Geometry is continuous (in $\mathcal{A}$) (analytic computation)

Actions in the Ambient Space

- Geometric Relationships on different Object types arise in $\mathcal{A}$
  - E.g., Point is ON/LEFT/RIGHT of a Line
- Analytic properties of Objects comes from their loci
  - E.g., Proximity, Approximations, Isotopy, etc

(Contd.) Computation: Geometric vs. Algebraic

Answer to PUZZLE 1: "BOTH"

- Geometry is discrete (in $\mathcal{P}$) (algebraic computation)
- Geometry is continuous (in $\mathcal{A}$) (analytic computation)

Actions in the Ambient Space

- Geometric Relationships on different Object types arise in $\mathcal{A}$
  - E.g., Point is ON/LEFT/RIGHT of a Line
- Analytic properties of Objects comes from their loci
  - E.g., Proximity, Approximations, Isotopy, etc

(Contd.) Computation: Geometric vs. Algebraic

Answer to PUZZLE 1: "BOTH"

- Geometry is discrete (in $\mathcal{P}$) (algebraic computation)
- Geometry is continuous (in $\mathcal{A}$) (analytic computation)

Actions in the Ambient Space

- Geometric Relationships on different Object types arise in $\mathcal{A}$
  - E.g., Point is ON/LEFT/RIGHT of a Line
- Analytic properties of Objects comes from their loci
  - E.g., Proximity, Approximations, Isotopy, etc

(Contd.) Computation: Geometric vs. Algebraic

Answer to PUZZLE 1: “BOTH”

- Geometry is discrete (in $\mathcal{P}$) (algebraic computation)
- Geometry is continuous (in $\mathcal{A}$) (analytic computation)

Actions in the Ambient Space

- Geometric Relationships on different Object types arise in $\mathcal{A}$
  - E.g., Point is ON/LEFT/RIGHT of a Line
- Analytic properties of Objects comes from their loci
  - E.g., Proximity, Approximations, Isotopy, etc

(Contd.) Computation: Geometric vs. Algebraic

Answer to PUZZLE 1: “BOTH”

- Geometry is discrete (in $\mathcal{P}$) (algebraic computation)
- Geometry is continuous (in $\mathcal{A}$) (analytic computation)

Actions in the Ambient Space

- Geometric Relationships on different Object types arise in $\mathcal{A}$
  - E.g., Point is ON/LEFT/RIGHT of a Line
- Analytic properties of Objects comes from their loci
  - E.g., Proximity, Approximations, Isotopy, etc

(Contd.) Computation: Geometric vs. Algebraic

Answer to PUZZLE 1: "BOTH"

- Geometry is discrete (in $\mathcal{P}$) (algebraic computation)
- Geometry is continuous (in $\mathcal{A}$) (analytic computation)

Actions in the Ambient Space

- Geometric Relationships on different Object types arise in $\mathcal{A}$
  - E.g., Point is ON/LEFT/RIGHT of a Line
- Analytic properties of Objects comes from their loci
  - E.g., Proximity, Approximations, Isotopy, etc

Mini Summary

- Geometry is discrete (algebraic view)
- Geometry is continuous (analytic view)
- Up Next : What do Computational Geometers think?

Mini Summary

- Geometry is discrete (algebraic view)
- Geometry is continuous (analytic view)
- Up Next : What do Computational Geometers think?

Mini Summary

- Geometry is discrete (algebraic view)
- Geometry is continuous (analytic view)
- Up Next: What do Computational Geometers think?

Mini Summary

- Geometry is discrete (algebraic view)
- Geometry is continuous (analytic view)
- Up Next : What do Computational Geometers think?

Mini Summary

- Geometry is discrete (algebraic view)
- Geometry is continuous (analytic view)
- Up Next: What do Computational Geometers think?

Coming Up Next

Five Examples of Geometric Computation

- (I) Convex Hulls
- (II) Euclidean Shortest Path
- (III) Disc Avoiding Shortest Path
- (IV) Mesh Generation
- (V) Discrete Morse Theory

Five Examples of Geometric Computation

- (I) Convex Hulls
- (II) Euclidean Shortest Path
- (III) Disc Avoiding Shortest Path
- (IV) Mesh Generation
- (V) Discrete Morse Theory

Five Examples of Geometric Computation

- (I) Convex Hulls
- (II) Euclidean Shortest Path
- (III) Disc Avoiding Shortest Path
- (IV) Mesh Generation
- (V) Discrete Morse Theory

Five Examples of Geometric Computation

- (I) Convex Hulls
- (II) Euclidean Shortest Path
- (III) Disc Avoiding Shortest Path
- (IV) Mesh Generation
- (V) Discrete Morse Theory

Five Examples of Geometric Computation

- (I) Convex Hulls
- (II) Euclidean Shortest Path
- (III) Disc Avoiding Shortest Path
- (IV) Mesh Generation
- (V) Discrete Morse Theory

Five Examples of Geometric Computation

- (I) Convex Hulls
- (II) Euclidean Shortest Path
- (III) Disc Avoiding Shortest Path
- (IV) Mesh Generation
- (V) Discrete Morse Theory

Five Examples of Geometric Computation

- (I) Convex Hulls
- (II) Euclidean Shortest Path
- (III) Disc Avoiding Shortest Path
- (IV) Mesh Generation
- (V) Discrete Morse Theory

(I) Convex Hulls

Convex Hull of Points in $\mathbb{R}^n$

- $n = 1$: finding max and min

(I) Convex Hulls

Convex Hull of Points in $\mathbb{R}^n$

- $n = 1$: finding max and min
- $n = 2, 3$: find a convex polygon or polytope

(I) Convex Hulls

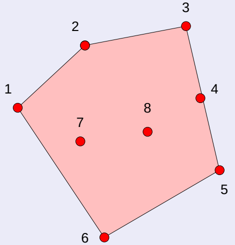
Convex Hull of Points in $\mathbb{R}^n$

- $n = 1$: finding max and min
- $n = 2, 3$: find a convex polygon or polytope

(I) Convex Hulls

Convex Hull of Points in $\mathbb{R}^n$

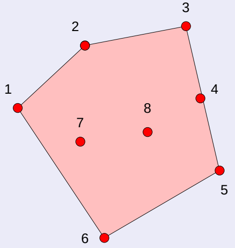
- $n = 1$: finding max and min
- $n = 2, 3$: find a convex polygon or polytope



(I) Convex Hulls

Convex Hull of Points in $\mathbb{R}^n$

- $n = 1$: finding max and min
- $n = 2, 3$: find a convex polygon or polytope

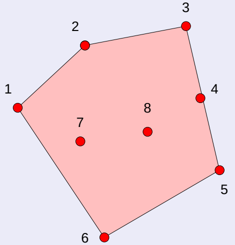


Can be reduced to a single predicate
Orientation$(P_0, P_1, \ldots, P_n)$

(I) Convex Hulls

Convex Hull of Points in $\mathbb{R}^n$

- $n = 1$: finding max and min
- $n = 2, 3$: find a convex polygon or polytope



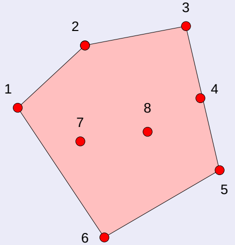
Can be reduced to a single predicate $Orientation(P_0, P_1, \ldots, P_n)$

- Main issue is combinatorial in nature

(I) Convex Hulls

Convex Hull of Points in $\mathbb{R}^n$

- $n = 1$: finding max and min
- $n = 2, 3$: find a convex polygon or polytope



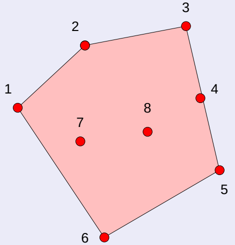
Can be reduced to a single predicate $\mathit{Orientation}(P_0, P_1, \ldots, P_n)$

- Main issue is combinatorial in nature

(I) Convex Hulls

Convex Hull of Points in $\mathbb{R}^n$

- $n = 1$: finding max and min
- $n = 2, 3$: find a convex polygon or polytope



Can be reduced to a single predicate $Orientation(P_0, P_1, \ldots, P_n)$

- Main issue is combinatorial in nature

(II) Euclidean Shortest Path (ESP)

Shortest Path amidst Polygonal Obstacles

- Shortest path from p to q avoiding A, B, C

(II) Euclidean Shortest Path (ESP)

Shortest Path amidst Polygonal Obstacles

- Shortest path from p to q avoiding A, B, C

(II) Euclidean Shortest Path (ESP)

Shortest Path amidst Polygonal Obstacles

- Shortest path from p to q avoiding A, B, C

(II) Euclidean Shortest Path (ESP)

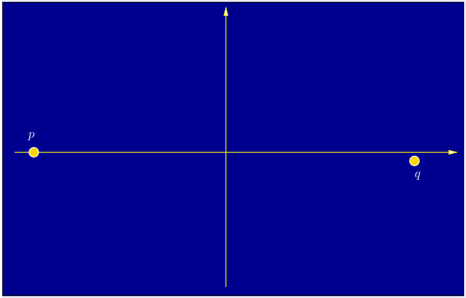
Shortest Path amidst Polygonal Obstacles

- Shortest path from p to q avoiding A, B, C

(II) Euclidean Shortest Path (ESP)

Shortest Path amidst Polygonal Obstacles

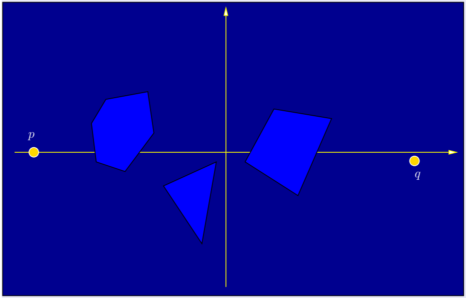
- Shortest path from p to q avoiding A, B, C



(II) Euclidean Shortest Path (ESP)

Shortest Path amidst Polygonal Obstacles

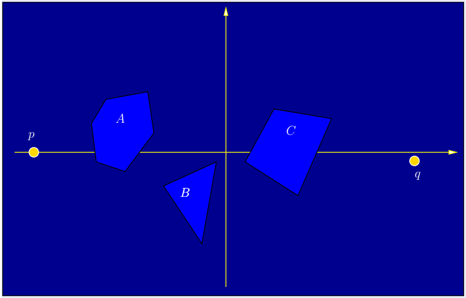
- Shortest path from p to q avoiding A, B, C



(II) Euclidean Shortest Path (ESP)

Shortest Path amidst Polygonal Obstacles

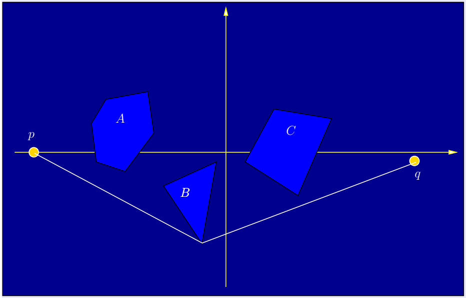
- Shortest path from p to q avoiding A, B, C



(II) Euclidean Shortest Path (ESP)

Shortest Path amidst Polygonal Obstacles

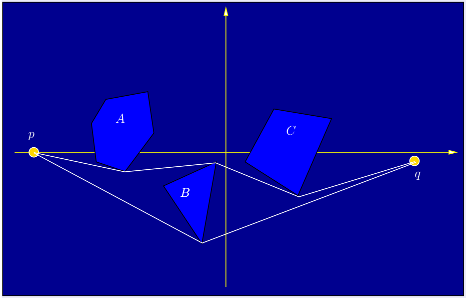
- Shortest path from p to q avoiding A, B, C



(II) Euclidean Shortest Path (ESP)

Shortest Path amidst Polygonal Obstacles

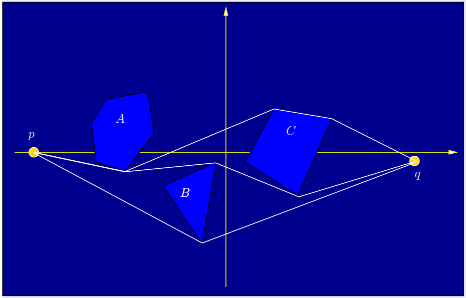
- Shortest path from p to q avoiding A, B, C



(II) Euclidean Shortest Path (ESP)

Shortest Path amidst Polygonal Obstacles

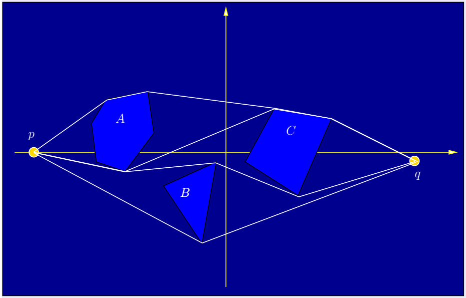
- Shortest path from p to q avoiding A, B, C



(II) Euclidean Shortest Path (ESP)

Shortest Path amidst Polygonal Obstacles

- Shortest path from p to q avoiding A, B, C

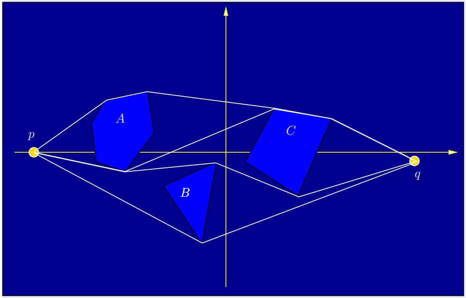


Segment length is a square-root

(II) Euclidean Shortest Path (ESP)

Shortest Path amidst Polygonal Obstacles

- Shortest path from p to q avoiding A, B, C

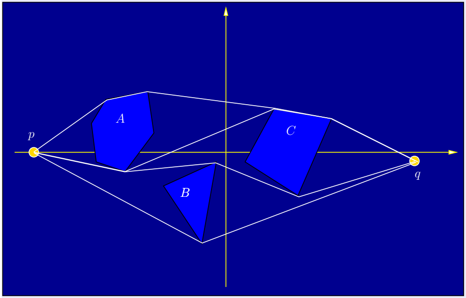


Segment length is a square-root

(II) Euclidean Shortest Path (ESP)

Shortest Path amidst Polygonal Obstacles

- Shortest path from p to q avoiding A, B, C



Segment length is a square-root

ESP, contd.

Reduction to Dijkstra's Algorithm

- Combinatorial complexity: $O(n^2 \log n)$ (negligible)
- Sum of Square-roots Problem: $\boxed{\text{Is } \sum_{i=1}^{m} a_i \sqrt{b_i} = 0?}$
- Not known to be polynomial-time!
- Algebraic Approach: Repeated Squaring Method (Nontrivial for Inequalites!)
  - $\Omega(2^m)$ (slow, unless you are lucky! (Illustrate))
- Numerical Approach: Zero Bound Method
  - $O(\log(1/|e|))$ (fast, unless you are unlucky! (Illustrate))
- Luck deals differently for the two approaches

ESP, contd.

Reduction to Dijkstra's Algorithm

- Combinatorial complexity: $O(n^2 \log n)$ (negligible)
- Sum of Square-roots Problem: $\boxed{\text{Is } \sum_{i=1}^{m} a_i \sqrt{b_i} = 0?}$
- Not known to be polynomial-time!
- Algebraic Approach: Repeated Squaring Method (Nontrivial for Inequalites!)
  - $\Omega(2^m)$ (slow, unless you are lucky! (Illustrate))
- Numerical Approach: Zero Bound Method
  - $O(\log(1/|e|))$ (fast, unless you are unlucky! (Illustrate))
- Luck deals differently for the two approaches

ESP, contd.

Reduction to Dijkstra's Algorithm

- Combinatorial complexity: $O(n^2 \log n)$ (negligible)
- Sum of Square-roots Problem: $\boxed{\text{Is } \sum_{i=1}^{m} a_i \sqrt{b_i} = 0?}$
- Not known to be polynomial-time!
- Algebraic Approach: Repeated Squaring Method (Nontrivial for Inequalites!)
  - $\Omega(2^m)$ (slow, unless you are lucky! (Illustrate))
- Numerical Approach: Zero Bound Method
  - $O(\log(1/|e|))$ (fast, unless you are unlucky! (Illustrate))
- Luck deals differently for the two approaches

ESP, contd.

Reduction to Dijkstra's Algorithm

- Combinatorial complexity: $O(n^2 \log n)$ (negligible)
- Sum of Square-roots Problem: $\boxed{\text{Is } \sum_{i=1}^{m} a_i \sqrt{b_i} = 0?}$
- Not known to be polynomial-time!
- Algebraic Approach: Repeated Squaring Method (Nontrivial for Inequalites!)
  - $\Omega(2^m)$ (slow, unless you are lucky! (Illustrate))
- Numerical Approach: Zero Bound Method
  - $O(\log(1/|e|))$ (fast, unless you are unlucky! (Illustrate))
- $\boxed{\text{Luck deals differently for the two approaches}}$

ESP, contd.

Reduction to Dijkstra's Algorithm

- Combinatorial complexity: $O(n^2 \log n)$ (negligible)
- Sum of Square-roots Problem: $\boxed{\text{Is } \sum_{i=1}^{m} a_i \sqrt{b_i} = 0?}$
- Not known to be polynomial-time!
- Algebraic Approach: Repeated Squaring Method (Nontrivial for Inequalites!)
  - $\Omega(2^m)$ (slow, unless you are lucky! (Illustrate))
- Numerical Approach: Zero Bound Method
  - $O(\log(1/|e|))$ (fast, unless you are unlucky! (Illustrate))
- Luck deals differently for the two approaches

ESP, contd.

Reduction to Dijkstra's Algorithm

- Combinatorial complexity: $O(n^2 \log n)$ (negligible)
- Sum of Square-roots Problem: Is $\sum_{i=1}^{m} a_i \sqrt{b_i} = 0$?
- Not known to be polynomial-time!
- Algebraic Approach: Repeated Squaring Method (Nontrivial for Inequalites!)
  - $\Omega(2^m)$ (slow, unless you are lucky! (Illustrate))
- Numerical Approach: Zero Bound Method
  - $O(\log(1/|e|))$ (fast, unless you are unlucky! (Illustrate))
- Luck deals differently for the two approaches

ESP, contd.

Reduction to Dijkstra's Algorithm

- Combinatorial complexity: $O(n^2 \log n)$ (negligible)
- Sum of Square-roots Problem: $\boxed{\text{Is } \sum_{i=1}^{m} a_i \sqrt{b_i} = 0?}$
- Not known to be polynomial-time!
- Algebraic Approach: Repeated Squaring Method (Nontrivial for Inequalites!)
  - $\Omega(2^m)$ (slow, unless you are lucky! (Illustrate))
- Numerical Approach: Zero Bound Method
  - $O(\log(1/|e|))$ (fast, unless you are unlucky! (Illustrate))
- Luck deals differently for the two approaches

ESP, contd.

Reduction to Dijkstra's Algorithm

- Combinatorial complexity: $O(n^2 \log n)$ (negligible)
- Sum of Square-roots Problem: $\boxed{\text{Is } \sum_{i=1}^{m} a_i \sqrt{b_i} = 0?}$
- Not known to be polynomial-time!
- Algebraic Approach: Repeated Squaring Method (Nontrivial for Inequalites!)
  - $\Omega(2^m)$ (slow, unless you are lucky! (Illustrate))
- Numerical Approach: Zero Bound Method
  - $O(\log(1/|e|))$ (fast, unless you are unlucky! (Illustrate))
- Luck deals differently for the two approaches

(III) Shortest Path Amidst Discs

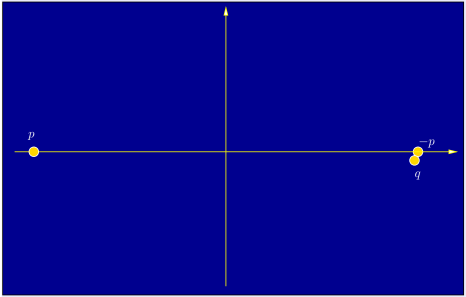
Shortest Path amidst Discs

- Shortest path from p to q avoiding discs A, B

(III) Shortest Path Amidst Discs

Shortest Path amidst Discs

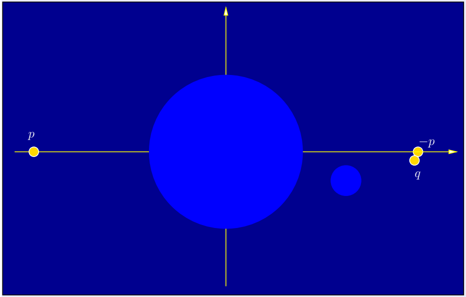
- Shortest path from p to q avoiding discs A, B



(III) Shortest Path Amidst Discs

Shortest Path amidst Discs

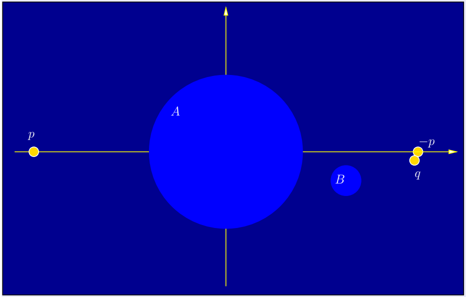
- Shortest path from p to q avoiding discs A, B



(III) Shortest Path Amidst Discs

Shortest Path amidst Discs

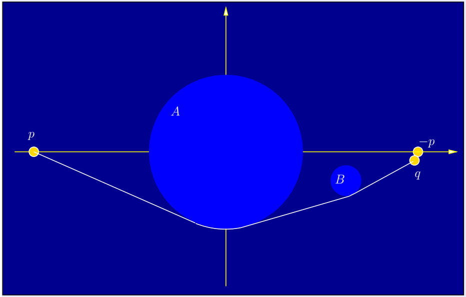
- Shortest path from p to q avoiding discs A, B



(III) Shortest Path Amidst Discs

Shortest Path amidst Discs

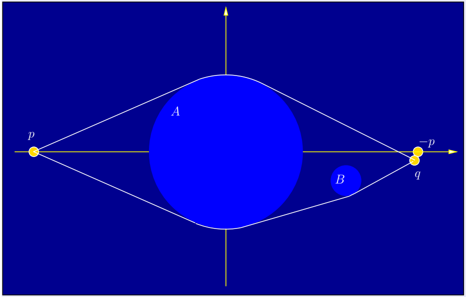
- Shortest path from p to q avoiding discs A, B



(III) Shortest Path Amidst Discs

Shortest Path amidst Discs

- Shortest path from p to q avoiding discs A, B

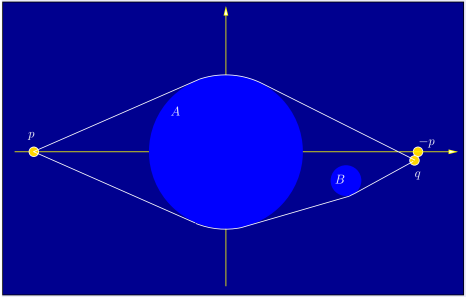


  - Segment length is a square-root of an algebraic number

(III) Shortest Path Amidst Discs

Shortest Path amidst Discs

- Shortest path from p to q avoiding discs A, B

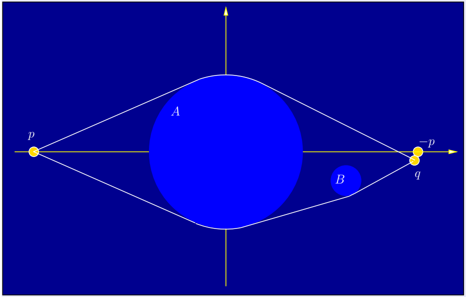


- Segment length is a square-root of an algebraic number
- Arc lengh is $r\,\theta$

(III) Shortest Path Amidst Discs

Shortest Path amidst Discs

- Shortest path from p to q avoiding discs A, B

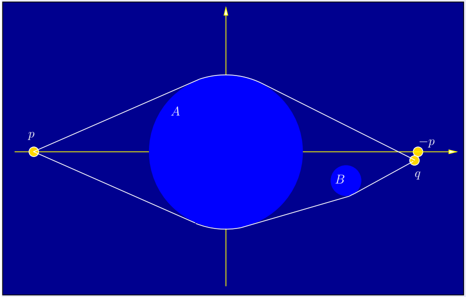


- Segment length is a square-root of an algebraic number
- Arc lengh is $r\theta$

(III) Shortest Path Amidst Discs

Shortest Path amidst Discs

- Shortest path from p to q avoiding discs A, B

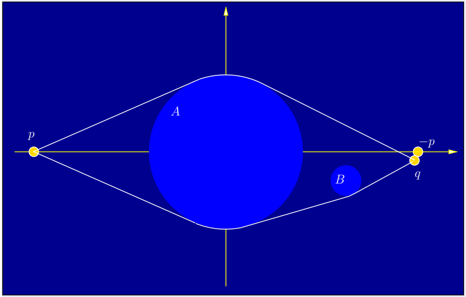


  - Segment length is a square-root of an algebraic number
  - Arc lengh is $r\,\theta$

(III) Shortest Path Amidst Discs

Shortest Path amidst Discs

- Shortest path from p to q avoiding discs A, B



- Segment length is a square-root of an algebraic number
- Arc lengh is $r\,\theta$

Disc Obstacles, contd.

Reduction to Dijkstra's Algorithm (Again?)

- Combinatorial complexity: $O(n^2 \log n)$ (negligible, exercise)
- Path length $= \gamma + \alpha$
  where γ is algebraic, but α is transcendental
- Not even clear that we can compute this!
  - Why? Numerical Halting Problem
- Analogue of the Turing Halting Problem
  - Also semi-decidable

Disc Obstacles, contd.

Reduction to Dijkstra's Algorithm (Again?)

- Combinatorial complexity: $O(n^2 \log n)$ (negligible, exercise)
- Path length $= \gamma + \alpha$
  where γ is algebraic, but α is transcendental
- Not even clear that we can compute this!
  - Why? Numerical Halting Problem
- Analogue of the Turing Halting Problem
  - Also semi-decidable

Disc Obstacles, contd.

Reduction to Dijkstra's Algorithm (Again?)

- Combinatorial complexity: $O(n^2 \log n)$ (negligible, exercise)
- Path length $= \gamma + \alpha$
  where γ is algebraic, but α is transcendental
- Not even clear that we can compute this!
  - Why? Numerical Halting Problem
- Analogue of the Turing Halting Problem
  - Also semi-decidable

Disc Obstacles, contd.

Reduction to Dijkstra's Algorithm (Again?)

- Combinatorial complexity: $O(n^2 \log n)$ (negligible, exercise)
- Path length $= \gamma + \alpha$
  where γ is algebraic, but α is transcendental
- Not even clear that we can compute this!
  - Why? $\boxed{\text{Numerical Halting Problem}}$
- Analogue of the $\boxed{\text{Turing Halting Problem}}$
  - Also semi-decidable

Disc Obstacles, contd.

Reduction to Dijkstra's Algorithm (Again?)

- Combinatorial complexity: $O(n^2 \log n)$ (negligible, exercise)
- Path length $= \gamma + \alpha$
  where γ is algebraic, but α is transcendental
- Not even clear that we can compute this!
  - Why? Numerical Halting Problem
- Analogue of the Turing Halting Problem
  - Also semi-decidable

Disc Obstacles, contd.

Reduction to Dijkstra's Algorithm (Again?)

- Combinatorial complexity: $O(n^2 \log n)$ (negligible, exercise)
- Path length $= \gamma + \alpha$
  where γ is algebraic, but α is transcendental
- Not even clear that we can compute this!
  - Why? $\boxed{\text{Numerical Halting Problem}}$
- Analogue of the $\boxed{\text{Turing Halting Problem}}$
  - Also semi-decidable

Disc Obstacles, contd.

Reduction to Dijkstra's Algorithm (Again?)

- Combinatorial complexity: $O(n^2 \log n)$ (negligible, exercise)
- Path length $= \gamma + \alpha$
  where γ is algebraic, but α is transcendental
- Not even clear that we can compute this!
  - Why? Numerical Halting Problem
- Analogue of the Turing Halting Problem
  - Also semi-decidable

Disc Obstacles, contd.

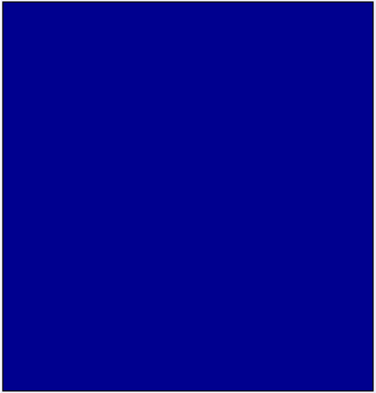
Reduction to Dijkstra's Algorithm (Again?)

- Combinatorial complexity: $O(n^2 \log n)$ (negligible, exercise)
- Path length $= \gamma + \alpha$
  where γ is algebraic, but α is transcendental
- Not even clear that we can compute this!
  - Why? Numerical Halting Problem
- Analogue of the Turing Halting Problem
  - Also semi-decidable

Addition/Subtraction of Arc Lengths

Simple Case: Unit Discs

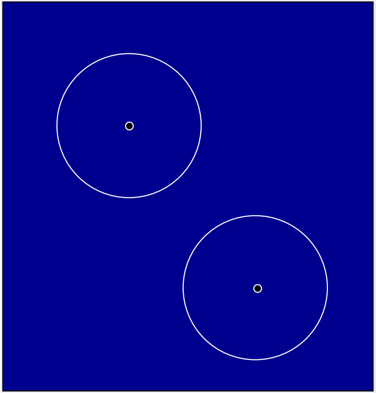
Let $A = [C, p, q, n]$ and $A' = [C', p', q', n']$ encode two arc lengths.



Addition/Subtraction of Arc Lengths

Simple Case: Unit Discs

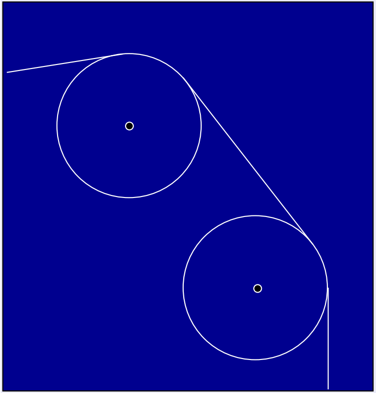
Let $A = [C, p, q, n]$ and $A' = [C', p', q', n']$ encode two arc lengths.



Addition/Subtraction of Arc Lengths

Simple Case: Unit Discs

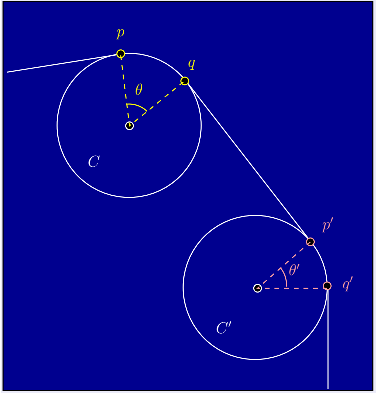
Let $A = [C, p, q, n]$ and $A' = [C', p', q', n']$ encode two arc lengths.



Addition/Subtraction of Arc Lengths

Simple Case: Unit Discs

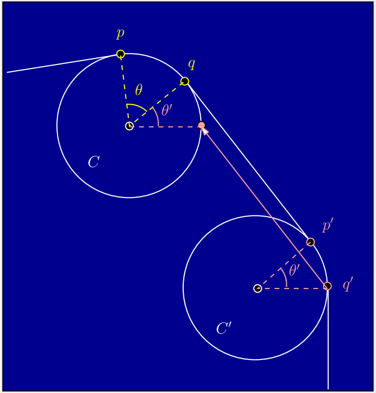
Let $A = [C, p, q, n]$ and $A' = [C', p', q', n']$ encode two arc lengths.



Addition/Subtraction of Arc Lengths

Simple Case: Unit Discs

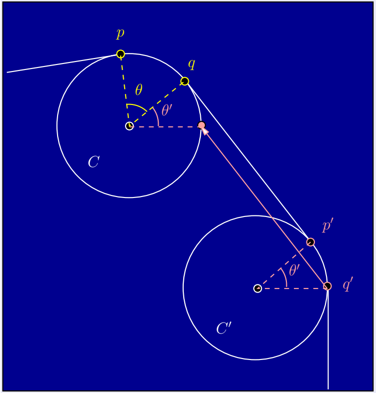
Let $A = [C, p, q, n]$ and $A' = [C', p', q', n']$ encode two arc lengths.

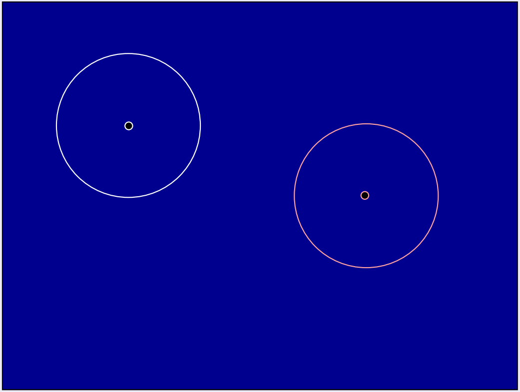


Addition/Subtraction of Arc Lengths

Simple Case: Unit Discs

Let $A = [C, p, q, n]$ and $A' = [C', p', q', n']$ encode two arc lengths.

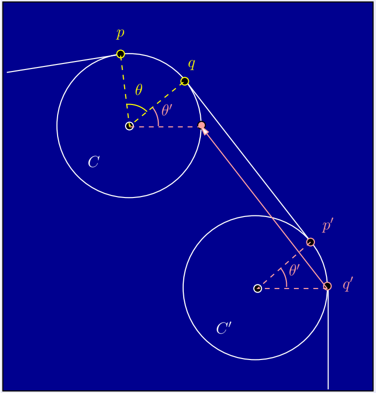


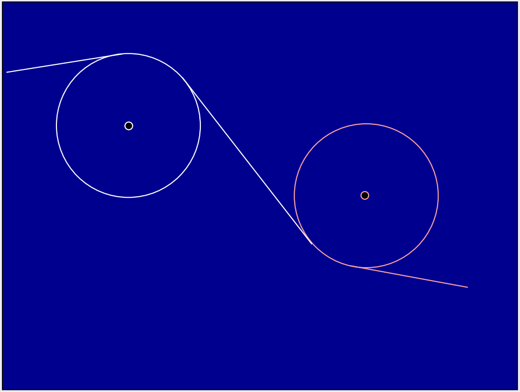


Addition/Subtraction of Arc Lengths

Simple Case: Unit Discs

Let $A = [C, p, q, n]$ and $A' = [C', p', q', n']$ encode two arc lengths.

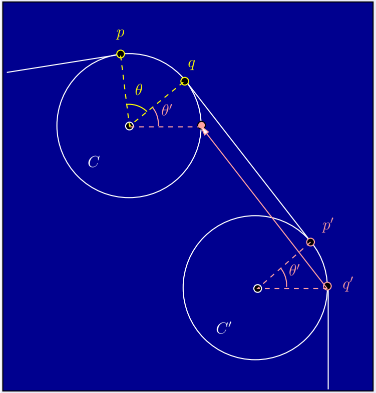


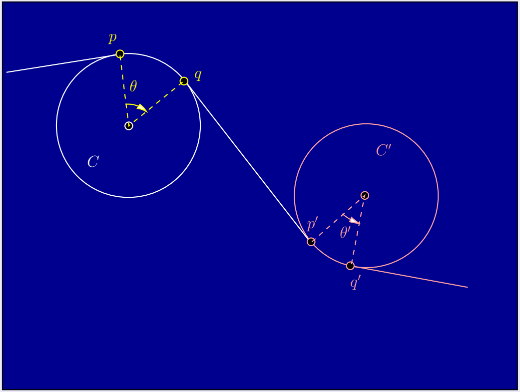


Addition/Subtraction of Arc Lengths

Simple Case: Unit Discs

Let $A = [C, p, q, n]$ and $A' = [C', p', q', n']$ encode two arc lengths.

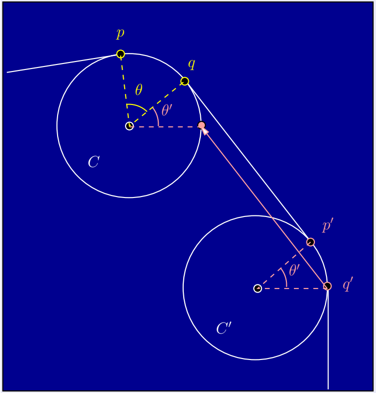


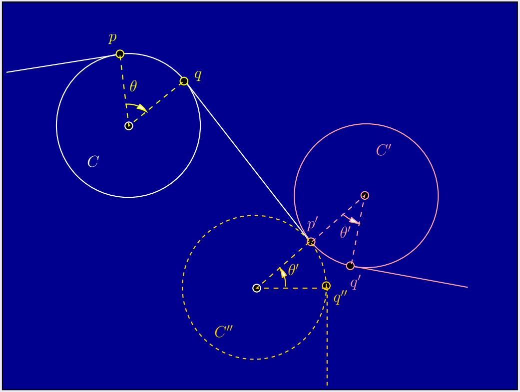


Addition/Subtraction of Arc Lengths

Simple Case: Unit Discs

Let $A = [C, p, q, n]$ and $A' = [C', p', q', n']$ encode two arc lengths.

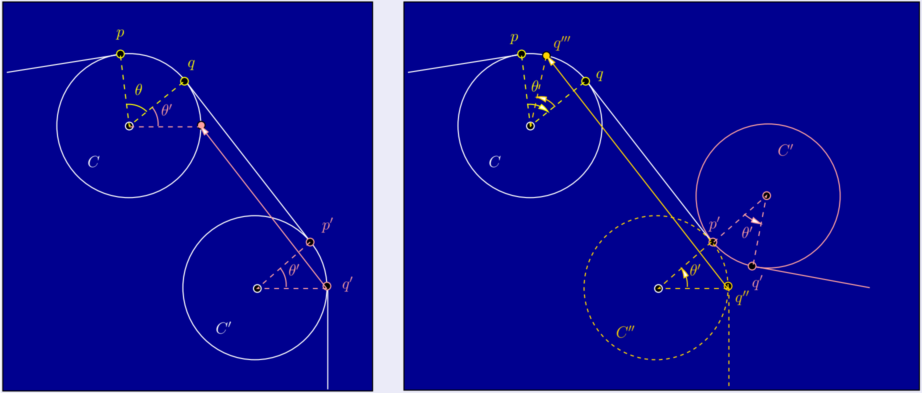




Addition/Subtraction of Arc Lengths

Simple Case: Unit Discs

Let $A = [C, p, q, n]$ and $A' = [C', p', q', n']$ encode two arc lengths.



Computability of Shortest Paths

[Chang/Choi/Kwon/Park/Y. (2005)]

Is it really Transcendental?

LEMMA: $\cos\theta_i$ is algebraic.
COROLLARY (Lindemann 1882): θ_i is transcendental.

Theorem (Unit Disc)

Shortest Path for unit disc obstacles is computable.

Rational Case

Much harder – use Chebyshev functions of first kind.
Main issue: how to transfer arc lengths to circles with different radii.

Theorem (Commensurable Radii)

Shortest Path for commensurable radii discs is computable.

Computability of Shortest Paths

[Chang/Choi/Kwon/Park/Y. (2005)]

Is it really Transcendental?

LEMMA: $\cos\theta_i$ is algebraic.
COROLLARY (Lindemann 1882): θ_i is transcendental.

Theorem (Unit Disc)

Shortest Path for unit disc obstacles is computable.

Rational Case

Much harder – use Chebyshev functions of first kind.
Main issue: how to transfer arc lengths to circles with different radii.

Theorem (Commensurable Radii)

Shortest Path for commensurable radii discs is computable.

Computability of Shortest Paths

[Chang/Choi/Kwon/Park/Y. (2005)]

Is it really Transcendental?

LEMMA: $\cos\theta_i$ is algebraic.
COROLLARY (Lindemann 1882): θ_i is transcendental.

Theorem (Unit Disc)

Shortest Path for unit disc obstacles is computable.

Rational Case

Much harder – use Chebyshev functions of first kind.
Main issue: how to transfer arc lengths to circles with different radii.

Theorem (Commensurable Radii)

Shortest Path for commensurable radii discs is computable.

Computability of Shortest Paths

[Chang/Choi/Kwon/Park/Y. (2005)]

Is it really Transcendental?

LEMMA: $\cos\theta_i$ is algebraic.

COROLLARY (Lindemann 1882): θ_i is transcendental.

Theorem (Unit Disc)

Shortest Path for unit disc obstacles is computable.

Rational Case

Much harder – use Chebyshev functions of first kind.

Main issue: how to transfer arc lengths to circles with different radii.

Theorem (Commensurable Radii)

Shortest Path for commensurable radii discs is computable.

Computability of Shortest Paths (contd.)

No complexity Bounds!

Elementary methods fail us.
Appeal to Baker's Linear Form in Logarithms: $|\alpha_0 + \sum_{i=1}^{n} \alpha_i \log \beta_i| > B$

Theorem (Commensurable Radii Complexity)

Shortest Paths for rational discs is in single-exponential time.

- Rare positive result from Transcendental Number Theory
- First transcendental geometric problem shown computable

Computability of Shortest Paths (contd.)

No complexity Bounds!

Elementary methods fail us.
Appeal to Baker's Linear Form in Logarithms: $|\alpha_0 + \sum_{i=1}^{n} \alpha_i \log \beta_i| > B$

Theorem (Commensurable Radii Complexity)

Shortest Paths for rational discs is in single-exponential time.

- Rare positive result from Transcendental Number Theory
- First transcendental geometric problem shown computable

Computability of Shortest Paths (contd.)

No complexity Bounds!

Elementary methods fail us.

Appeal to Baker's Linear Form in Logarithms: $\left|\alpha_0 + \sum_{i=1}^{n} \alpha_i \log \beta_i\right| > B$

Theorem (Commensurable Radii Complexity)

Shortest Paths for rational discs is in single-exponential time.

- Rare positive result from Transcendental Number Theory
- First transcendental geometric problem shown computable

Computability of Shortest Paths (contd.)

No complexity Bounds!

Elementary methods fail us.
Appeal to Baker's Linear Form in Logarithms: $\left|\alpha_0 + \sum_{i=1}^{n} \alpha_i \log \beta_i\right| > B$

Theorem (Commensurable Radii Complexity)

Shortest Paths for rational discs is in single-exponential time.

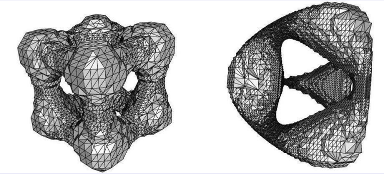
- Rare positive result from Transcendental Number Theory
- First transcendental geometric problem shown computable

(IV) Mesh Generation

Meshing of Surfaces

- Surface $S = f^{-1}(0)$ where $f : \mathbb{R}^n \to \mathbb{R}$ ($n = 1, 2, 3$)
- Wants a triangulated surface $\widetilde{S}$ that is isotopic to S

“Tangled Cube”



“Chair”

- Case $n = 1$ is root isolation!
- Return to meshing in Lecture 2

Applications

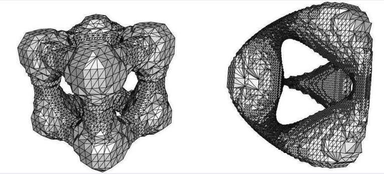
Visualization, Graphics, Simulation, Modeling: prerequisite

(IV) Mesh Generation

Meshing of Surfaces

- Surface $S = f^{-1}(0)$ where $f : \mathbb{R}^n \to \mathbb{R}$ ($n = 1, 2, 3$)
- Wants a triangulated surface $\widetilde{S}$ that is isotopic to S

"Tangled Cube"



"Chair"

- Case $n = 1$ is root isolation!
- Return to meshing in Lecture 2

Applications

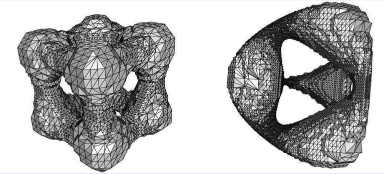
Visualization, Graphics, Simulation, Modeling: prerequisite

(IV) Mesh Generation

Meshing of Surfaces

- Surface $S = f^{-1}(0)$ where $f : \mathbb{R}^n \to \mathbb{R}$ ($n = 1, 2, 3$)
- Wants a triangulated surface $\widetilde{S}$ that is isotopic to S

"Tangled Cube"



"Chair"

- Case $n = 1$ is root isolation!
- Return to meshing in Lecture 2

Applications

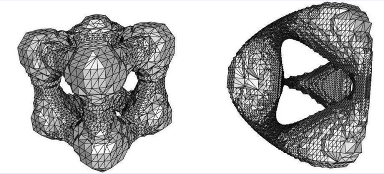
Visualization, Graphics, Simulation, Modeling: prerequisite

(IV) Mesh Generation

Meshing of Surfaces

- Surface $S = f^{-1}(0)$ where $f : \mathbb{R}^n \to \mathbb{R}$ ($n = 1, 2, 3$)
- Wants a triangulated surface $\widetilde{S}$ that is isotopic to S

"Tangled Cube"



"Chair"

- Case $n = 1$ is root isolation!
- Return to meshing in Lecture 2

Applications

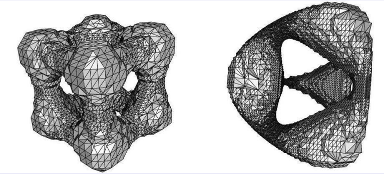
Visualization, Graphics, Simulation, Modeling: prerequisite

(IV) Mesh Generation

Meshing of Surfaces

- Surface $S = f^{-1}(0)$ where $f : \mathbb{R}^n \to \mathbb{R}$ ($n = 1, 2, 3$)
- Wants a triangulated surface $\widetilde{S}$ that is isotopic to S

"Tangled Cube"



"Chair"

- Case $n = 1$ is root isolation!
- Return to meshing in Lecture 2

Applications

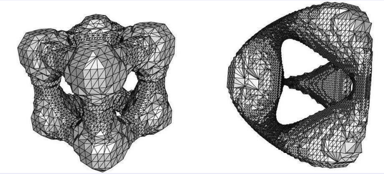
Visualization, Graphics, Simulation, Modeling: prerequisite

(IV) Mesh Generation

Meshing of Surfaces

- Surface $S = f^{-1}(0)$ where $f : \mathbb{R}^n \to \mathbb{R}$ ($n = 1, 2, 3$)
- Wants a triangulated surface $\widetilde{S}$ that is isotopic to S

"Tangled Cube"



"Chair"

- Case $n = 1$ is root isolation!
- Return to meshing in Lecture 2

Applications

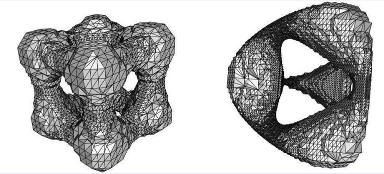
Visualization, Graphics, Simulation, Modeling: prerequisite

(IV) Mesh Generation

Meshing of Surfaces

- Surface $S = f^{-1}(0)$ where $f : \mathbb{R}^n \to \mathbb{R}$ ($n = 1, 2, 3$)
- Wants a triangulated surface $\widetilde{S}$ that is isotopic to S

"Tangled Cube"



"Chair"

- Case $n = 1$ is root isolation!
- Return to meshing in Lecture 2

Applications

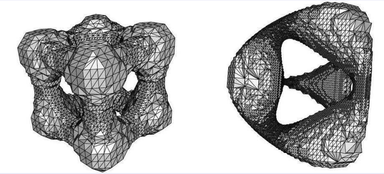
Visualization, Graphics, Simulation, Modeling: prerequisite

(IV) Mesh Generation

Meshing of Surfaces

- Surface $S = f^{-1}(0)$ where $f : \mathbb{R}^n \to \mathbb{R}$ $(n = 1, 2, 3)$
- Wants a triangulated surface $\widetilde{S}$ that is isotopic to S

"Tangled Cube"



"Chair"

- Case $n = 1$ is root isolation!
- Return to meshing in Lecture 2

Applications

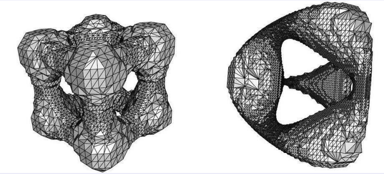
Visualization, Graphics, Simulation, Modeling: prerequisite

(IV) Mesh Generation

Meshing of Surfaces

- Surface $S = f^{-1}(0)$ where $f : \mathbb{R}^n \to \mathbb{R}$ ($n = 1, 2, 3$)
- Wants a triangulated surface $\widetilde{S}$ that is isotopic to S

“Tangled Cube”



“Chair”

- Case $n = 1$ is root isolation!
- Return to meshing in Lecture 2

Applications

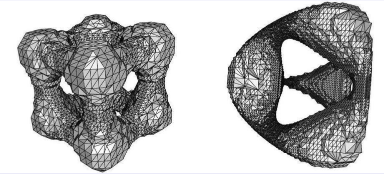
Visualization, Graphics, Simulation, Modeling: prerequisite

(IV) Mesh Generation

Meshing of Surfaces

- Surface $S = f^{-1}(0)$ where $f : \mathbb{R}^n \to \mathbb{R}$ ($n = 1, 2, 3$)
- Wants a triangulated surface $\widetilde{S}$ that is isotopic to S

"Tangled Cube"



"Chair"

- Case $n = 1$ is root isolation!
- Return to meshing in Lecture 2

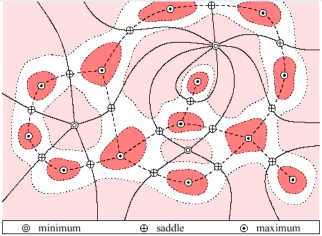
Applications

Visualization, Graphics, Simulation, Modeling: prerequisite

(V) Discrete Morse Theory

Edelsbrunner, Harer, Zomorodian (2003)

- Methodology: discrete analogues of continuous concepts
  - Differential geometry, Ricci flows, etc
- Morse-Smale Complex of a surface $S = f^{-1}(0)$:

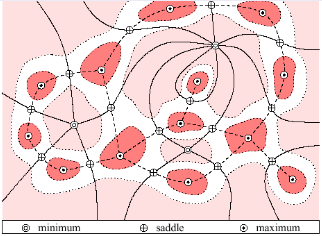


- Exactness Bottleneck: this "Continuous-to-Discrete" transformation

(V) Discrete Morse Theory

Edelsbrunner, Harer, Zomorodian (2003)

- Methodology: discrete analogues of continuous concepts
  - Differential geometry, Ricci flows, etc
- Morse-Smale Complex of a surface $S = f^{-1}(0)$:

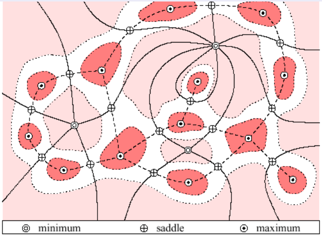


- Exactness Bottleneck: this “Continuous-to-Discrete” transformation

(V) Discrete Morse Theory

Edelsbrunner, Harer, Zomorodian (2003)

- Methodology: discrete analogues of continuous concepts
  - Differential geometry, Ricci flows, etc
- Morse-Smale Complex of a surface $S = f^{-1}(0)$:

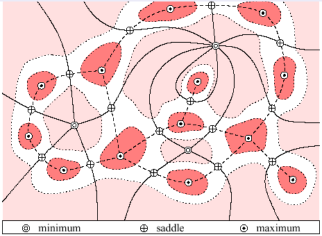


- Exactness Bottleneck: this “Continuous-to-Discrete” transformation

(V) Discrete Morse Theory

Edelsbrunner, Harer, Zomorodian (2003)

- Methodology: discrete analogues of continuous concepts
  - Differential geometry, Ricci flows, etc
- Morse-Smale Complex of a surface $S = f^{-1}(0)$:

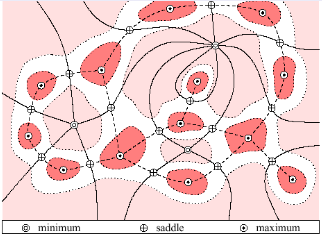


  - Critical Points (max/min/saddle)
  - Integral Lines
  - OPEN: How to connect saddle to its maximas
- Exactness Bottleneck: this “Continuous-to-Discrete” transformation

(V) Discrete Morse Theory

Edelsbrunner, Harer, Zomorodian (2003)

- Methodology: discrete analogues of continuous concepts
  - Differential geometry, Ricci flows, etc
- Morse-Smale Complex of a surface $S = f^{-1}(0)$:

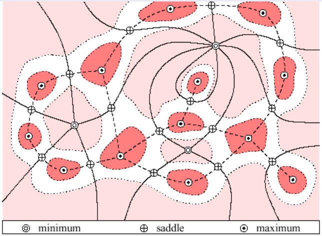


  - Critical Points (max/min/saddle)
  - Integral Lines
  - OPEN: How to connect saddle to its maximas
- Exactness Bottleneck: this “Continuous-to-Discrete” transformation

(V) Discrete Morse Theory

Edelsbrunner, Harer, Zomorodian (2003)

- Methodology: discrete analogues of continuous concepts
  - Differential geometry, Ricci flows, etc
- Morse-Smale Complex of a surface $S = f^{-1}(0)$:

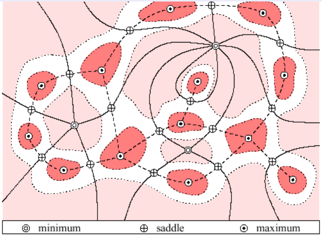


  - Critical Points (max/min/saddle)
  - Integral Lines
  - OPEN: How to connect saddle to its maximas
- Exactness Bottleneck: this "Continuous-to-Discrete" transformation

(V) Discrete Morse Theory

Edelsbrunner, Harer, Zomorodian (2003)

- Methodology: discrete analogues of continuous concepts
  - Differential geometry, Ricci flows, etc
- Morse-Smale Complex of a surface $S = f^{-1}(0)$:

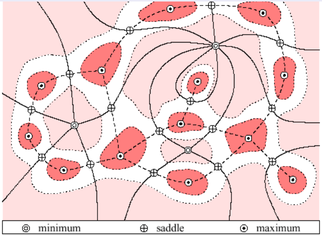


  - Critical Points (max/min/saddle)
  - Integral Lines
  - OPEN: How to connect saddle to its maximas
- Exactness Bottleneck: this “Continuous-to-Discrete” transformation

(V) Discrete Morse Theory

Edelsbrunner, Harer, Zomorodian (2003)

- Methodology: discrete analogues of continuous concepts
  - Differential geometry, Ricci flows, etc
- Morse-Smale Complex of a surface $S = f^{-1}(0)$:

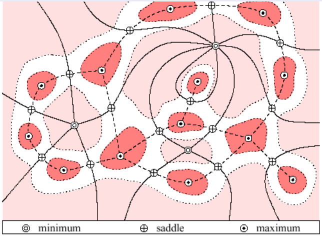


  - Critical Points (max/min/saddle)
  - Integral Lines
  - OPEN: How to connect saddle to its maximas
- Exactness Bottleneck: this "Continuous-to-Discrete" transformation

(V) Discrete Morse Theory

Edelsbrunner, Harer, Zomorodian (2003)

- Methodology: discrete analogues of continuous concepts
  - Differential geometry, Ricci flows, etc
- Morse-Smale Complex of a surface $S = f^{-1}(0)$:

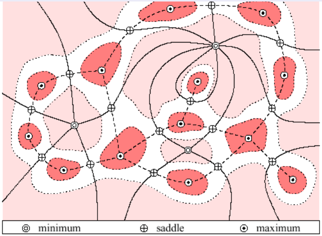


  - Critical Points (max/min/saddle)
  - Integral Lines
  - OPEN: How to connect saddle to its maximas
- Exactness Bottleneck: this “Continuous-to-Discrete” transformation

(V) Discrete Morse Theory

Edelsbrunner, Harer, Zomorodian (2003)

- Methodology: discrete analogues of continuous concepts
  - Differential geometry, Ricci flows, etc
- Morse-Smale Complex of a surface $S = f^{-1}(0)$:

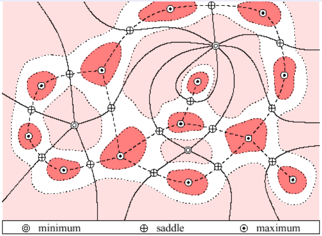


  - Critical Points (max/min/saddle)
  - Integral Lines
  - OPEN: How to connect saddle to its maximas
- Exactness Bottleneck: this “Continuous-to-Discrete” transformation

(V) Discrete Morse Theory

Edelsbrunner, Harer, Zomorodian (2003)

- Methodology: discrete analogues of continuous concepts
  - Differential geometry, Ricci flows, etc
- Morse-Smale Complex of a surface $S = f^{-1}(0)$:

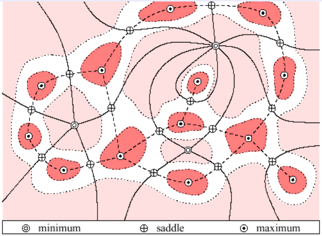


  - Critical Points (max/min/saddle)
  - Integral Lines
  - OPEN: How to connect saddle to its maximas
- Exactness Bottleneck: this “Continuous-to-Discrete” transformation

(V) Discrete Morse Theory

Edelsbrunner, Harer, Zomorodian (2003)

- Methodology: discrete analogues of continuous concepts
  - Differential geometry, Ricci flows, etc
- Morse-Smale Complex of a surface $S = f^{-1}(0)$:



  - Critical Points (max/min/saddle)
  - Integral Lines
  - OPEN: How to connect saddle to its maximas
- Exactness Bottleneck: this "Continuous-to-Discrete" transformation

Mini Summary

- We saw 5 Geometric Problems:
  I=classic, II=hard, III=very hard, IV=current, V=open
- Up Next : Let us examine their underlying computational models...

Mini Summary

- We saw 5 Geometric Problems:
  I=classic, II=hard, III=very hard, IV=current, V=open
- Up Next : Let us examine their underlying computational models...

Mini Summary

- We saw 5 Geometric Problems:
  I=classic, II=hard, III=very hard, IV=current, V=open
- Up Next : Let us examine their underlying computational models...

Mini Summary

- We saw 5 Geometric Problems:
  I=classic, II=hard, III=very hard, IV=current, V=open
- Up Next: Let us examine their underlying computational models...

Coming Up Next

1. Introduction: What is Geometric Computation?
2. Five Examples of Geometric Computation
3. Exact Numeric Computation – A Synthesis
4. Exact Geometric Computation
5. Constructive Zero Bounds

Two Worlds of Computing

- (EX) Discrete, Combinatorial, *Exact*.
  - Theoretical Computer Science, Computer Algebra
- (AP) Continuous, Numerical, *Approximate*.
  - Computational Science & Engineering (CS&E) or Physics
  - Problems too hard in exact framework (e.g., 3D Ising Model)
  - Even when exact solution is possible,...
- The 2 Worlds meet in Geometry
  - Solving Linear Systems (Gaussian vs. Gauss-Seidel)
  - Linear Programming (Simplex vs. Interior-Point)
  - Solving Numerical PDE (Symbolic vs. Numeric)

Two Worlds of Computing

- (EX) Discrete, Combinatorial, *Exact*.
  - Theoretical Computer Science, Computer Algebra
- (AP) Continuous, Numerical, *Approximate*.
  - Computational Science & Engineering (CS&E) or Physics
  - Problems too hard in exact framework (e.g., 3D Ising Model)
  - Even when exact solution is possible,...
- The 2 Worlds meet in Geometry
  - Solving Linear Systems (Gaussian vs. Gauss-Seidel)
  - Linear Programming (Simplex vs. Interior-Point)
  - Solving Numerical PDE (Symbolic vs. Numeric)

Two Worlds of Computing

- (EX) Discrete, Combinatorial, *Exact*.
  - Theoretical Computer Science, Computer Algebra
- (AP) Continuous, Numerical, *Approximate*.
  - Computational Science & Engineering (CS&E) or Physics
  - Problems too hard in exact framework (e.g., 3D Ising Model)
  - Even when exact solution is possible,...
- The 2 Worlds meet in Geometry
  - Solving Linear Systems (Gaussian vs. Gauss-Seidel)
  - Linear Programming (Simplex vs. Interior-Point)
  - Solving Numerical PDE (Symbolic vs. Numeric)

Two Worlds of Computing

- (EX) Discrete, Combinatorial, *Exact*.
  - Theoretical Computer Science, Computer Algebra
- (AP) Continuous, Numerical, *Approximate*.
  - Computational Science & Engineering (CS&E) or Physics
  - Problems too hard in exact framework (e.g., 3D Ising Model)
  - Even when exact solution is possible,...
- The 2 Worlds meet in Geometry
  - Solving Linear Systems (Gaussian vs. Gauss-Seidel)
  - Linear Programming (Simplex vs. Interior-Point)
  - Solving Numerical PDE (Symbolic vs. Numeric)

Two Worlds of Computing

- (EX) Discrete, Combinatorial, *Exact*.
  - Theoretical Computer Science, Computer Algebra
- (AP) Continuous, Numerical, *Approximate*.
  - Computational Science & Engineering (CS&E) or Physics
  - Problems too hard in exact framework (e.g., 3D Ising Model)
  - Even when exact solution is possible,...
- The 2 Worlds meet in Geometry
  - Solving Linear Systems (Gaussian vs. Gauss-Seidel)
  - Linear Programming (Simplex vs. Interior-Point)
  - Solving Numerical PDE (Symbolic vs. Numeric)

Two Worlds of Computing

- (EX) Discrete, Combinatorial, *Exact*.
  - Theoretical Computer Science, Computer Algebra
- (AP) Continuous, Numerical, *Approximate*.
  - Computational Science & Engineering (CS&E) or Physics
  - Problems too hard in exact framework (e.g., 3D Ising Model)
  - Even when exact solution is possible,...
- The 2 Worlds meet in Geometry
  - Solving Linear Systems (Gaussian vs. Gauss-Seidel)
  - Linear Programming (Simplex vs. Interior-Point)
  - Solving Numerical PDE (Symbolic vs. Numeric)

Two Worlds of Computing

- (EX) Discrete, Combinatorial, *Exact*.
  - Theoretical Computer Science, Computer Algebra
- (AP) Continuous, Numerical, *Approximate*.
  - Computational Science & Engineering (CS&E) or Physics
  - Problems too hard in exact framework (e.g., 3D Ising Model)
  - Even when exact solution is possible,...
- The 2 Worlds meet in Geometry
  - Solving Linear Systems (Gaussian vs. Gauss-Seidel)
  - Linear Programming (Simplex vs. Interior-Point)
  - Solving Numerical PDE (Symbolic vs. Numeric)

Two Worlds of Computing

- (EX) Discrete, Combinatorial, *Exact*.
  - Theoretical Computer Science, Computer Algebra
- (AP) Continuous, Numerical, *Approximate*.
  - Computational Science & Engineering (CS&E) or Physics
  - Problems too hard in exact framework (e.g., 3D Ising Model)
  - Even when exact solution is possible,...
- The 2 Worlds meet in Geometry
  - Solving Linear Systems (Gaussian vs. Gauss-Seidel)
  - Linear Programming (Simplex vs. Interior-Point)
  - Solving Numerical PDE (Symbolic vs. Numeric)

Two Worlds of Computing

- (EX) Discrete, Combinatorial, *Exact*.
  - Theoretical Computer Science, Computer Algebra
- (AP) Continuous, Numerical, *Approximate*.
  - Computational Science & Engineering (CS&E) or Physics
  - Problems too hard in exact framework (e.g., 3D Ising Model)
  - Even when exact solution is possible,...
- The 2 Worlds meet in Geometry
  - Solving Linear Systems (Gaussian vs. Gauss-Seidel)
  - Linear Programming (Simplex vs. Interior-Point)
  - Solving Numerical PDE (Symbolic vs. Numeric)

Two Worlds of Computing

- (EX) Discrete, Combinatorial, *Exact*.
  - Theoretical Computer Science, Computer Algebra
- (AP) Continuous, Numerical, *Approximate*.
  - Computational Science & Engineering (CS&E) or Physics
  - Problems too hard in exact framework (e.g., 3D Ising Model)
  - Even when exact solution is possible,...
- The 2 Worlds meet in Geometry
  - Solving Linear Systems (Gaussian vs. Gauss-Seidel)
  - Linear Programming (Simplex vs. Interior-Point)
  - Solving Numerical PDE (Symbolic vs. Numeric)

Two Worlds of Computing

- (EX) Discrete, Combinatorial, *Exact*.
  - Theoretical Computer Science, Computer Algebra
- (AP) Continuous, Numerical, *Approximate*.
  - Computational Science & Engineering (CS&E) or Physics
  - Problems too hard in exact framework (e.g., 3D Ising Model)
  - Even when exact solution is possible,...
- The 2 Worlds meet in Geometry
  - Solving Linear Systems (Gaussian vs. Gauss-Seidel)
  - Linear Programming (Simplex vs. Interior-Point)
  - Solving Numerical PDE (Symbolic vs. Numeric)

Two Worlds of Computing

- (EX) Discrete, Combinatorial, *Exact*.
  - Theoretical Computer Science, Computer Algebra
- (AP) Continuous, Numerical, *Approximate*.
  - Computational Science & Engineering (CS&E) or Physics
  - Problems too hard in exact framework (e.g., 3D Ising Model)
  - Even when exact solution is possible,...
- The 2 Worlds meet in Geometry
  - Solving Linear Systems (Gaussian vs. Gauss-Seidel)
  - Linear Programming (Simplex vs. Interior-Point)
  - Solving Numerical PDE (Symbolic vs. Numeric)

Again, What is Geometry?

Geometry is always about zeros

- Problem (I): Is a Point on a Hyperplane?
- Problems (II),(III): Are two path lengths are equal?
- Problems (IV),(V): Continuous-to-discrete transformations, defined by zero sets
- These zero decisions are captured by geometric predicates
- View developed by CG'ers in robust geometric computation

Again, What is Geometry?

Geometry is always about zeros

- Problem (I): Is a Point on a Hyperplane?
- Problems (II),(III): Are two path lengths are equal?
- Problems (IV),(V): Continuous-to-discrete transformations, defined by zero sets
- These zero decisions are captured by geometric predicates
- View developed by CG'ers in robust geometric computation

Again, What is Geometry?

Geometry is always about zeros

- Problem (I): Is a Point on a Hyperplane?
- Problems (II),(III): Are two path lengths are equal?
- Problems (IV),(V): Continuous-to-discrete transformations, defined by zero sets
- These zero decisions are captured by geometric predicates
- View developed by CG'ers in robust geometric computation

Again, What is Geometry?

Geometry is always about zeros

- Problem (I): Is a Point on a Hyperplane?
- Problems (II),(III): Are two path lengths are equal?
- Problems (IV),(V): Continuous-to-discrete transformations, defined by zero sets
- These zero decisions are captured by geometric predicates
- View developed by CG'ers in robust geometric computation

Again, What is Geometry?

Geometry is always about zeros

- Problem (I): Is a Point on a Hyperplane?
- Problems (II),(III): Are two path lengths are equal?
- Problems (IV),(V): Continuous-to-discrete transformations, defined by zero sets
- These zero decisions are captured by geometric predicates
- View developed by CG'ers in robust geometric computation

Again, What is Geometry?

Geometry is always about zeros

- Problem (I): Is a Point on a Hyperplane?
- Problems (II),(III): Are two path lengths are equal?
- Problems (IV),(V): Continuous-to-discrete transformations, defined by zero sets
- These zero decisions are captured by geometric predicates
- View developed by CG'ers in robust geometric computation

Again, What is Geometry?

Geometry is always about zeros

- Problem (I): Is a Point on a Hyperplane?
- Problems (II),(III): Are two path lengths are equal?
- Problems (IV),(V): Continuous-to-discrete transformations, defined by zero sets
- These zero decisions are captured by geometric predicates
- View developed by CG'ers in robust geometric computation

Four Computational Models for Geometry

How to compute in a Continuum ($\mathbb{R}^n$)?

- (EX) Algebraic Computational Model
  (e.g., Real RAM, Blum-Shub-Smale model, Disc Shortest Path)
  - PROBLEM: Zero is trivial
- (EX') Abstract Operational Models
  (e.g., CG, Traub, Orientation, Ray shooting, Giftwrap)
  - PROBLEM: Zero is hidden
- (AP) Analytic Computational Model (e.g., Ko, Weihrauch)
  - PROBLEM: Zero is undecidable
- (AP') Numerical Analysis Model (e.g., $x \oplus y = (x+y)(1+\varepsilon)$)
  - PROBLEM: Zero is abolished

Four Computational Models for Geometry

How to compute in a Continuum ($\mathbb{R}^n$)?

- (EX) Algebraic Computational Model
  (e.g., Real RAM, Blum-Shub-Smale model, Disc Shortest Path)
  - PROBLEM: Zero is trivial
- (EX') Abstract Operational Models
  (e.g., CG, Traub, Orientation, Ray shooting, Giftwrap)
  - PROBLEM: Zero is hidden
- (AP) Analytic Computational Model (e.g., Ko, Weihrauch)
  - PROBLEM: Zero is undecidable
- (AP') Numerical Analysis Model (e.g., $x \oplus y = (x+y)(1+\varepsilon)$)
  - PROBLEM: Zero is abolished

Four Computational Models for Geometry

How to compute in a Continuum ($\mathbb{R}^n$)?

- (EX) Algebraic Computational Model
  (e.g., Real RAM, Blum-Shub-Smale model, Disc Shortest Path)
  - PROBLEM: Zero is trivial
- (EX') Abstract Operational Models
  (e.g., CG, Traub, Orientation, Ray shooting, Giftwrap)
  - PROBLEM: Zero is hidden
- (AP) Analytic Computational Model (e.g., Ko, Weihrauch)
  - PROBLEM: Zero is undecidable
- (AP') Numerical Analysis Model (e.g., $x \oplus y = (x+y)(1+\varepsilon)$)
  - PROBLEM: Zero is abolished

Four Computational Models for Geometry

How to compute in a Continuum ($\mathbb{R}^n$)?

- (EX) Algebraic Computational Model
  (e.g., Real RAM, Blum-Shub-Smale model, Disc Shortest Path)
  - PROBLEM: Zero is trivial
- (EX') Abstract Operational Models
  (e.g., CG, Traub, Orientation, Ray shooting, Giftwrap)
  - PROBLEM: Zero is hidden
- (AP) Analytic Computational Model (e.g., Ko, Weihrauch)
  - PROBLEM: Zero is undecidable
- (AP') Numerical Analysis Model (e.g., $x \oplus y = (x + y)(1 + \varepsilon)$)
  - PROBLEM: Zero is abolished

Four Computational Models for Geometry

How to compute in a Continuum ($\mathbb{R}^n$)?

- (EX) Algebraic Computational Model
  (e.g., Real RAM, Blum-Shub-Smale model, Disc Shortest Path)
  - PROBLEM: Zero is trivial
- (EX') Abstract Operational Models
  (e.g., CG, Traub, Orientation, Ray shooting, Giftwrap)
  - PROBLEM: Zero is hidden
- (AP) Analytic Computational Model (e.g., Ko, Weihrauch)
  - PROBLEM: Zero is undecidable
- (AP') Numerical Analysis Model (e.g., $x \oplus y = (x+y)(1+\varepsilon)$)
  - PROBLEM: Zero is abolished

Four Computational Models for Geometry

How to compute in a Continuum ($\mathbb{R}^n$)?

- (EX) Algebraic Computational Model
  (e.g., Real RAM, Blum-Shub-Smale model, Disc Shortest Path)
  - PROBLEM: Zero is trivial
- (EX') Abstract Operational Models
  (e.g., CG, Traub, Orientation, Ray shooting, Giftwrap)
  - PROBLEM: Zero is hidden
- (AP) Analytic Computational Model (e.g., Ko, Weihrauch)
  - PROBLEM: Zero is undecidable
- (AP') Numerical Analysis Model (e.g., $x \oplus y = (x+y)(1+\varepsilon)$)
  - PROBLEM: Zero is abolished

Four Computational Models for Geometry

How to compute in a Continuum ($\mathbb{R}^n$)?

- (EX) Algebraic Computational Model
  (e.g., Real RAM, Blum-Shub-Smale model, Disc Shortest Path)
  - PROBLEM: Zero is trivial
- (EX') Abstract Operational Models
  (e.g., CG, Traub, Orientation, Ray shooting, Giftwrap)
  - PROBLEM: Zero is hidden
- (AP) Analytic Computational Model (e.g., Ko, Weihrauch)
  - PROBLEM: Zero is undecidable
- (AP') Numerical Analysis Model (e.g., $x \oplus y = (x+y)(1+\varepsilon)$)
  - PROBLEM: Zero is abolished

Four Computational Models for Geometry

How to compute in a Continuum ($\mathbb{R}^n$)?

- (EX) Algebraic Computational Model
  (e.g., Real RAM, Blum-Shub-Smale model, Disc Shortest Path)
  - PROBLEM: Zero is trivial
- (EX') Abstract Operational Models
  (e.g., CG, Traub, Orientation, Ray shooting, Giftwrap)
  - PROBLEM: Zero is hidden
- (AP) Analytic Computational Model (e.g., Ko, Weihrauch)
  - PROBLEM: Zero is undecidable
- (AP') Numerical Analysis Model (e.g., $x \oplus y = (x+y)(1+\varepsilon)$)
  - PROBLEM: Zero is abolished

Four Computational Models for Geometry

How to compute in a Continuum ($\mathbb{R}^n$)?

- (EX) Algebraic Computational Model
  (e.g., Real RAM, Blum-Shub-Smale model, Disc Shortest Path)
  - PROBLEM: Zero is trivial
- (EX') Abstract Operational Models
  (e.g., CG, Traub, Orientation, Ray shooting, Giftwrap)
  - PROBLEM: Zero is hidden
- (AP) Analytic Computational Model (e.g., Ko, Weihrauch)
  - PROBLEM: Zero is undecidable
- (AP') Numerical Analysis Model (e.g., $x \oplus y = (x+y)(1+\varepsilon)$)
  - PROBLEM: Zero is abolished

Four Computational Models for Geometry

How to compute in a Continuum ($\mathbb{R}^n$)?

- (EX) Algebraic Computational Model
  (e.g., Real RAM, Blum-Shub-Smale model, Disc Shortest Path)
  - PROBLEM: Zero is trivial
- (EX') Abstract Operational Models
  (e.g., CG, Traub, Orientation, Ray shooting, Giftwrap)
  - PROBLEM: Zero is hidden
- (AP) Analytic Computational Model (e.g., Ko, Weihrauch)
  - PROBLEM: Zero is undecidable
- (AP') Numerical Analysis Model (e.g., $x \oplus y = (x+y)(1+\varepsilon)$)
  - PROBLEM: Zero is abolished

Other Issues

You cannot avoid the Zero Problem

- (EX) How do you implement $\mathbb{R}$?
- (EX') We may abstract away too much
  - cf. Problems (II) and (III)
- (AP) Only continuous functions are computable
  - Geometry is a "discontinuous" phenomenon
- (AP') Approximate geometry maybe harder than exact geometry
  - Exercise: Program a geometric algorithm w/o equality test

Other Issues

You cannot avoid the Zero Problem

- (EX) How do you implement $\mathbb{R}$?
- (EX') We may abstract away too much
  - cf. Problems (II) and (III)
- (AP) Only continuous functions are computable
  - Geometry is a "discontinuous" phenomenon
- (AP') Approximate geometry maybe harder than exact geometry
  - Exercise: Program a geometric algorithm w/o equality test

Other Issues

You cannot avoid the Zero Problem

- (EX) How do you implement $\mathbb{R}$?
- (EX') We may abstract away too much
  - cf. Problems (II) and (III)
- (AP) Only continuous functions are computable
  - Geometry is a discontinuous phenomenon
- (AP') Approximate geometry maybe harder than exact geometry
  - Exercise: Program a geometric algorithm w/o equality test

Other Issues

You cannot avoid the Zero Problem

- (EX) How do you implement $\mathbb{R}$?
- (EX') We may abstract away too much
  - cf. Problems (II) and (III)
- (AP) Only continuous functions are computable
  - Geometry is a discontinuous phenomenon
- (AP') Approximate geometry maybe harder than exact geometry
  - Exercise: Program a geometric algorithm w/o equality test

Other Issues

You cannot avoid the Zero Problem

- (EX) How do you implement $\mathbb{R}$?
- (EX') We may abstract away too much
  - cf. Problems (II) and (III)
- (AP) Only continuous functions are computable
  - Geometry is a discontinuous phenomenon
- (AP') Approximate geometry maybe harder than exact geometry
  - Exercise: Program a geometric algorithm w/o equality test

Other Issues

You cannot avoid the Zero Problem

- (EX) How do you implement $\mathbb{R}$?
- (EX') We may abstract away too much
  - cf. Problems (II) and (III)
- (AP) Only continuous functions are computable
  - Geometry is a discontinuous phenomenon
- (AP') Approximate geometry maybe harder than exact geometry
  - Exercise: Program a geometric algorithm w/o equality test

Other Issues

You cannot avoid the Zero Problem

- (EX) How do you implement $\mathbb{R}$?
- (EX') We may abstract away too much
  - cf. Problems (II) and (III)
- (AP) Only continuous functions are computable
  - Geometry is a discontinuous phenomenon
- (AP') Approximate geometry maybe harder than exact geometry
  - Exercise: Program a geometric algorithm w/o equality test

Other Issues

You cannot avoid the Zero Problem

- (EX) How do you implement $\mathbb{R}$?
- (EX') We may abstract away too much
  - cf. Problems (II) and (III)
- (AP) Only continuous functions are computable
  - Geometry is a discontinuous phenomenon
- (AP') Approximate geometry maybe harder than exact geometry
  - Exercise: Program a geometric algorithm w/o equality test

Other Issues

You cannot avoid the Zero Problem

- (EX) How do you implement $\mathbb{R}$?
- (EX') We may abstract away too much
  - cf. Problems (II) and (III)
- (AP) Only continuous functions are computable
  - Geometry is a discontinuous phenomenon
- (AP') Approximate geometry maybe harder than exact geometry
  - Exercise: Program a geometric algorithm w/o equality test

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
$\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$=\sqrt{15-\sqrt{224}}$	≈ 0.0223

- $\sqrt{15-\sqrt{224}}$ is exact, but 0.0223 is more useful!
  - WHY? Want the locus of α in the continuum
  - JOKE: a physicist and an engineer were in a hot-air balloon...
- How to capture this Duality?
  - For exact computation, need algebraic representation
  - For analytic properties, need an approximation process
  - What about deciding zero? (Algebraic or Numeric)

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
$\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$= \sqrt{15-\sqrt{224}}$	≈ 0.0223

- $\sqrt{15-\sqrt{224}}$ is exact, but 0.0223 is more useful!

- How to capture this Duality?

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
$\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$=\sqrt{15-\sqrt{224}}$	≈ 0.0223

- $\sqrt{15-\sqrt{224}}$ is exact, but 0.0223 is more useful!

  WHY? Want the locus of α in the continuum

  JOKE: a physicist and an engineer were in a hot-air balloon...

- How to capture this Duality?

  For exact computation, need algebraic representation.

  For analytic properties, need an approximation process.

  What about deciding zero? (Algebraic or Numeric)

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
$\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$=\sqrt{15-\sqrt{224}}$	≈ 0.0223

- $\sqrt{15-\sqrt{224}}$ is exact, but 0.0223 is more useful!

  WHY? Want the locus of α in the continuum

  JOKE: a physicist and an engineer were in a hot-air balloon...

- How to capture this Duality?

  For exact computation, need algebraic representation.

  For analytic properties, need an approximation process

  What about deciding zero? (Algebraic or Numeric)

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
▸ $\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$= \sqrt{15-\sqrt{224}}$	≈ 0.0223

- $\sqrt{15-\sqrt{224}}$ is exact, but 0.0223 is more useful!

- How to capture this Duality?

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
▸ $\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$= \sqrt{15 - \sqrt{224}}$	≈ 0.0223

- $\sqrt{15-\sqrt{224}}$ is exact, but 0.0223 is more useful!

- How to capture this Duality?

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
▸ $\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$= \sqrt{15 - \sqrt{224}}$	≈ 0.0223

- $\sqrt{15 - \sqrt{224}}$ is exact, but 0.0223 is more useful!

- How to capture this Duality?

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
$\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$=\sqrt{15-\sqrt{224}}$	≈ 0.0223

- $\sqrt{15-\sqrt{224}}$ is exact, but 0.0223 is more useful!
  - WHY? Want the locus of α in the continuum
  - JOKE: a physicist and an engineer were in a hot-air balloon...
- How to capture this Duality?
  - For exact computation, need algebraic representation.
  - For analytic properties, need an approximation process
  - What about deciding zero? (Algebraic or Numeric)

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
$\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$=\sqrt{15-\sqrt{224}}$	≈ 0.0223

- $\sqrt{15-\sqrt{224}}$ is exact, but 0.0223 is more useful!
  - WHY? Want the locus of α in the continuum
  - JOKE: a physicist and an engineer were in a hot-air balloon...
- How to capture this Duality?
  - For exact computation, need algebraic representation.
  - For analytic properties, need an approximation process
  - What about deciding zero? (Algebraic or Numeric)

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
▸ $\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$= \sqrt{15 - \sqrt{224}}$	≈ 0.0223

- $\sqrt{15 - \sqrt{224}}$ is exact, but 0.0223 is more useful!
  - WHY? Want the locus of α in the continuum
  - JOKE: a physicist and an engineer were in a hot-air balloon...
- How to capture this Duality?
  - For exact computation, need algebraic representation.
  - For analytic properties, need an approximation process
  - What about deciding zero? (Algebraic or Numeric)

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
$\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$= \sqrt{15 - \sqrt{224}}$	≈ 0.0223

- $\sqrt{15 - \sqrt{224}}$ is exact, but 0.0223 is more useful!
  - WHY? Want the locus of α in the continuum
  - JOKE: a physicist and an engineer were in a hot-air balloon...
- How to capture this Duality?
  - For exact computation, need algebraic representation.
  - For analytic properties, need an approximation process
  - What about deciding zero? (Algebraic or Numeric)

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
▸ $\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$= \sqrt{15 - \sqrt{224}}$	≈ 0.0223

- $\sqrt{15 - \sqrt{224}}$ is exact, but 0.0223 is more useful!
  - WHY? Want the locus of α in the continuum
  - JOKE: a physicist and an engineer were in a hot-air balloon...
- How to capture this Duality?
  - For exact computation, need algebraic representation.
  - For analytic properties, need an approximation process
  - What about deciding zero? (Algebraic or Numeric)

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
$\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$=\sqrt{15-\sqrt{224}}$	≈ 0.0223

- $\sqrt{15-\sqrt{224}}$ is exact, but 0.0223 is more useful!
  - WHY? Want the locus of α in the continuum
  - JOKE: a physicist and an engineer were in a hot-air balloon...
- How to capture this Duality?
  - For exact computation, need algebraic representation.
  - For analytic properties, need an approximation process
  - What about deciding zero? (Algebraic or Numeric)

Duality in Numbers

- Physics Analogy:

	Discrete	Continuous
Light	particle	wave
▸ $\mathbb{R}$	field	metric space
Numbers	algebraic	analytic
α	$= \sqrt{15 - \sqrt{224}}$	≈ 0.0223

- $\sqrt{15 - \sqrt{224}}$ is exact, but 0.0223 is more useful!
  - WHY? Want the locus of α in the continuum
  - JOKE: a physicist and an engineer were in a hot-air balloon...
- How to capture this Duality?
  - For exact computation, need algebraic representation.
  - For analytic properties, need an approximation process
  - What about deciding zero? (Algebraic or Numeric)

Mini Summary

- Geometry is decided by Zeros
- Zero is a special number
- Numbers have a dual nature: need dual representation
- Up Next : A General Solution

Mini Summary

- Geometry is decided by Zeros
- Zero is a special number
- Numbers have a dual nature: need dual representation
- Up Next : A General Solution

Mini Summary

- Geometry is decided by Zeros
- Zero is a special number
- Numbers have a dual nature: need dual representation
- Up Next : A General Solution

Mini Summary

- Geometry is decided by Zeros
- Zero is a special number
- Numbers have a dual nature: need dual representation
- Up Next : A General Solution

Mini Summary

- Geometry is decided by Zeros
- Zero is a special number
- Numbers have a dual nature: need dual representation
- Up Next: A General Solution

Mini Summary

- Geometry is decided by Zeros
- Zero is a special number
- Numbers have a dual nature: need dual representation
- Up Next : A General Solution

Coming Up Next

The Universal Solution (EGC)

Key Principle of Exact Geometric Computation (EGC)

- Algorithm = Sequence of Steps
- Steps = Construction $\boxed{x := y + 2;}$ or Tests $\boxed{\text{if } x = 0 \text{ goto L}}$
- Geometric relations determined by Tests (Zero or Sign)
- THUS: if Tests are error free, the Geometry is exact
- Numerical robustness follows! Take-home message

The Universal Solution (EGC)

Key Principle of Exact Geometric Computation (EGC)

- Algorithm = Sequence of Steps
- Steps = Construction $\boxed{x := y + 2;}$ or Tests $\boxed{\text{if } x = 0 \text{ goto L}}$
- Geometric relations determined by Tests (Zero or Sign)
- THUS: if Tests are error free, the Geometry is exact
- Numerical robustness follows! Take-home message

The Universal Solution (EGC)

Key Principle of Exact Geometric Computation (EGC)

- Algorithm = Sequence of Steps
- Steps = Construction $\boxed{x := y + 2;}$ or Tests $\boxed{\text{if } x = 0 \text{ goto L}}$
- Geometric relations determined by Tests (Zero or Sign)
- THUS: if Tests are error free, the Geometry is exact
- Numerical robustness follows! Take-home message

The Universal Solution (EGC)

Key Principle of Exact Geometric Computation (EGC)

- Algorithm = Sequence of Steps
- Steps = Construction $\boxed{x := y + 2;}$ or Tests $\boxed{\text{if } x = 0 \text{ goto L}}$
- Geometric relations determined by Tests (Zero or Sign)
- THUS: if Tests are error free, the Geometry is exact
- Numerical robustness follows! Take-home message

The Universal Solution (EGC)

Key Principle of Exact Geometric Computation (EGC)

- Algorithm = Sequence of Steps
- Steps = Construction $\boxed{x := y + 2;}$ or Tests $\boxed{\text{if } x = 0 \text{ goto L}}$
- Geometric relations determined by Tests (Zero or Sign)
- THUS: if Tests are error free, the Geometry is exact
- Numerical robustness follows! Take-home message

The Universal Solution (EGC)

Key Principle of Exact Geometric Computation (EGC)

- Algorithm = Sequence of Steps
- Steps = Construction $\boxed{x := y + 2;}$ or Tests $\boxed{\text{if } x = 0 \text{ goto L}}$
- Geometric relations determined by Tests (Zero or Sign)
- THUS: if Tests are error free, the Geometry is exact
- Numerical robustness follows! Take-home message

The Universal Solution (EGC)

Key Principle of Exact Geometric Computation (EGC)

- Algorithm = Sequence of Steps
- Steps = Construction $\boxed{x := y + 2;}$ or Tests $\boxed{\text{if } x = 0 \text{ goto L}}$
- Geometric relations determined by Tests (Zero or Sign)
- THUS: if Tests are error free, the Geometry is exact
- Numerical robustness follows! Take-home message

Implementing the Universal Solution (Core Library)

Any programmer can access this capability

```
#define Core_Level 3
#include "CORE.h"
.... Standard C++ Program ....
```

Numerical Accuracy API

- Level 1: Machine Accuracy (int, long, float, double)
- Level 2: Arbitrary Accuracy (BigInt, BigRat, BigFloat)
- Level 3: Guaranteed Accuracy (Expr)
- Program should compile at every Accuracy Level

Implementing the Universal Solution (Core Library)

Any programmer can access this capability

```
#define Core_Level 3
#include "CORE.h"
.... Standard C++ Program ....
```

Numerical Accuracy API

- Level 1: Machine Accuracy (int, long, float, double)
- Level 2: Arbitrary Accuracy (BigInt, BigRat, BigFloat)
- Level 3: Guaranteed Accuracy (Expr)
- Program should compile at every Accuracy Level

Implementing the Universal Solution (Core Library)

Any programmer can access this capability

```
#define Core_Level 3
#include "CORE.h"
.... Standard C++ Program ....
```

Numerical Accuracy API

- Level 1: Machine Accuracy (int, long, float, double)
- Level 2: Arbitrary Accuracy (BigInt, BigRat, BigFloat)
- Level 3: Guaranteed Accuracy (Expr)
- Program should compile at every Accuracy Level

Implementing the Universal Solution (Core Library)

Any programmer can access this capability

```
#define Core_Level 3
#include "CORE.h"
.... Standard C++ Program ....
```

Numerical Accuracy API

- Level 1: Machine Accuracy (int, long, float, double)
- Level 2: Arbitrary Accuracy (BigInt, BigRat, BigFloat)
- Level 3: Guaranteed Accuracy (Expr)
- Program should compile at every Accuracy Level

Implementing the Universal Solution (Core Library)

Any programmer can access this capability

```
#define Core_Level 3
#include "CORE.h"
.... Standard C++ Program ....
```

Numerical Accuracy API

- Level 1: Machine Accuracy (int, long, float, double)
- Level 2: Arbitrary Accuracy (BigInt, BigRat, BigFloat)
- Level 3: Guaranteed Accuracy (Expr)
- Program should compile at every Accuracy Level

Implementing the Universal Solution (Core Library)

Any programmer can access this capability

```
#define Core_Level 3
#include "CORE.h"
.... Standard C++ Program ....
```

Numerical Accuracy API

- Level 1: Machine Accuracy (int, long, float, double)
- Level 2: Arbitrary Accuracy (BigInt, BigRat, BigFloat)
- Level 3: Guaranteed Accuracy (Expr)
- Program should compile at every Accuracy Level

Implementing the Universal Solution (Core Library)

Any programmer can access this capability

```
#define Core_Level 3
#include "CORE.h"
.... Standard C++ Program ....
```

Numerical Accuracy API

- Level 1: Machine Accuracy (int, long, float, double)
- Level 2: Arbitrary Accuracy (BigInt, BigRat, BigFloat)
- Level 3: Guaranteed Accuracy (Expr)
- Program should compile at every Accuracy Level

Implementing the Universal Solution (Core Library)

Any programmer can access this capability

```
#define Core_Level 3
#include "CORE.h"
.... Standard C++ Program ....
```

Numerical Accuracy API

- Level 1: Machine Accuracy (int, long, float, double)
- Level 2: Arbitrary Accuracy (BigInt, BigRat, BigFloat)
- Level 3: Guaranteed Accuracy (Expr)
- Program should compile at every Accuracy Level

What is Achieved?

Features

- Removed numerical non-robustness from geometry (!)
- Algorithm-independent solution to non-robustness
- Standard (Euclidean) geometry (why important?)
- Exactness in geometry (can use approximate numbers !)
- Implemented in LEDA , CGAL , Core Library

Other Implications

- A new approach to do algebraic number computation
- Euclidean Shortest Path need signs of expressions like $\sum_{i=1}^{100} a_i \sqrt{b_i}$.

  Standard algebraic approach is doomed

What is Achieved?

Features

- Removed numerical non-robustness from geometry (!)
- Algorithm-independent solution to non-robustness
- Standard (Euclidean) geometry (why important?)
- Exactness in geometry (can use approximate numbers !)
- Implemented in LEDA, CGAL, Core Library

Other Implications

- A new approach to do algebraic number computation
- Euclidean Shortest Path need signs of expressions like $\sum_{i=1}^{100} a_i \sqrt{b_i}$.

  Standard algebraic approach is doomed

What is Achieved?

Features

- Removed numerical non-robustness from geometry (!)
- Algorithm-independent solution to non-robustness
- Standard (Euclidean) geometry (why important?)
- Exactness in geometry (can use approximate numbers !)
- Implemented in LEDA , CGAL , Core Library

Other Implications

- A new approach to do algebraic number computation
- Euclidean Shortest Path need signs of expressions like $\sum_{i=1}^{100} a_i \sqrt{b_i}$.

  Standard algebraic approach is doomed

What is Achieved?

Features

- Removed numerical non-robustness from geometry (!)
- Algorithm-independent solution to non-robustness
- Standard (Euclidean) geometry (why important?)
- Exactness in geometry (can use approximate numbers !)
- Implemented in LEDA, CGAL, Core Library

Other Implications

- A new approach to do algebraic number computation
- Euclidean Shortest Path need signs of expressions like $\sum_{i=1}^{100} a_i \sqrt{b_i}$.

  Standard algebraic approach is doomed

What is Achieved?

Features

- Removed numerical non-robustness from geometry (!)
- Algorithm-independent solution to non-robustness
- Standard (Euclidean) geometry (why important?)
- Exactness in geometry (can use approximate numbers !)
- Implemented in LEDA , CGAL , Core Library

Other Implications

- A new approach to do algebraic number computation
- Euclidean Shortest Path need signs of expressions like $\sum_{i=1}^{100} a_i \sqrt{b_i}$.

  Standard algebraic approach is doomed

What is Achieved?

Features

- Removed numerical non-robustness from geometry (!)
- Algorithm-independent solution to non-robustness
- Standard (Euclidean) geometry (why important?)
- Exactness in geometry (can use approximate numbers !)
- Implemented in LEDA , CGAL , Core Library

Other Implications

- A new approach to do algebraic number computation
- Euclidean Shortest Path need signs of expressions like $\sum_{i=1}^{100} a_i \sqrt{b_i}$.

  Standard algebraic approach is doomed

What is Achieved?

Features

- Removed numerical non-robustness from geometry (!)
- Algorithm-independent solution to non-robustness
- Standard (Euclidean) geometry (why important?)
- Exactness in geometry (can use approximate numbers !)
- Implemented in LEDA , CGAL , Core Library

Other Implications

- A new approach to do algebraic number computation
- Euclidean Shortest Path need signs of expressions like $\sum_{i=1}^{100} a_i \sqrt{b_i}$.

  Standard algebraic approach is doomed

What is Achieved?

Features

- Removed numerical non-robustness from geometry (!)
- Algorithm-independent solution to non-robustness
- Standard (Euclidean) geometry (why important?)
- Exactness in geometry (can use approximate numbers !)
- Implemented in LEDA , CGAL , Core Library

Other Implications

- A new approach to do algebraic number computation
- Euclidean Shortest Path need signs of expressions like $\sum_{i=1}^{100} a_i \sqrt{b_i}$.

  Standard algebraic approach is doomed

What is Achieved?

Features

- Removed numerical non-robustness from geometry (!)
- Algorithm-independent solution to non-robustness
- Standard (Euclidean) geometry (why important?)
- Exactness in geometry (can use approximate numbers !)
- Implemented in LEDA , CGAL , Core Library

Other Implications

- A new approach to do algebraic number computation
- Euclidean Shortest Path need signs of expressions like $\sum_{i=1}^{100} a_i \sqrt{b_i}$.

  Standard algebraic approach is doomed

Coming Up Next

Adaptive Zero Determination

Core of Core Library

- Must use numerical method based on Zero Bounds
  - Must NOT use algebraic methods!
- Let $\Omega = \{+, -, \times, \ldots\} \cup \mathbb{Z}$ be a class of operators
  - $ZERO(\Omega)$ is the corresponding zero problem
- Zero Bound for Ω is a function $B : Expr(\Omega) \to \mathbb{R}_{\geq 0}$ such that $e \in Expr(\Omega)$ is non-zero implies

$$|e| > B(e)$$

- How to use zero bounds? Combine with approximation.
- Zero Bound is the bottleneck only in case of zero.

Adaptive Zero Determination

Core of Core Library

- Must use numerical method based on Zero Bounds
  - Must NOT use algebraic methods!
- Let $\Omega = \{+, -, \times, \ldots\} \cup \mathbb{Z}$ be a class of operators
  - $ZERO(\Omega)$ is the corresponding zero problem
- Zero Bound for Ω is a function $B : Expr(\Omega) \to \mathbb{R}_{\geq 0}$ such that $e \in Expr(\Omega)$ is non-zero implies

$$|e| > B(e)$$

- How to use zero bounds? Combine with approximation.
- Zero Bound is the bottleneck only in case of zero.

Adaptive Zero Determination

Core of Core Library

- Must use numerical method based on Zero Bounds
  - Must NOT use algebraic methods!
- Let $\Omega = \{+, -, \times, \ldots\} \cup \mathbb{Z}$ be a class of operators
  - $ZERO(\Omega)$ is the corresponding zero problem
- Zero Bound for Ω is a function $B : Expr(\Omega) \to \mathbb{R}_{\geq 0}$ such that $e \in Expr(\Omega)$ is non-zero implies

$$|e| > B(e)$$

- How to use zero bounds? Combine with approximation.
- Zero Bound is the bottleneck only in case of zero.

Adaptive Zero Determination

Core of Core Library

- Must use numerical method based on Zero Bounds
  - Must NOT use algebraic methods!
- Let $\Omega = \{+, -, \times, \ldots\} \cup \mathbb{Z}$ be a class of operators
  - $ZERO(\Omega)$ is the corresponding zero problem
- Zero Bound for Ω is a function $B : Expr(\Omega) \to \mathbb{R}_{\geq 0}$ such that $e \in Expr(\Omega)$ is non-zero implies

$$|e| > B(e)$$

- How to use zero bounds? Combine with approximation.
- Zero Bound is the bottleneck only in case of zero.

Adaptive Zero Determination

Core of Core Library

- Must use numerical method based on Zero Bounds
  - Must NOT use algebraic methods!
- Let $\Omega = \{+, -, \times, \ldots\} \cup \mathbb{Z}$ be a class of operators
  - $ZERO(\Omega)$ is the corresponding zero problem
- Zero Bound for Ω is a function $B : Expr(\Omega) \to \mathbb{R}_{\geq 0}$ such that $e \in Expr(\Omega)$ is non-zero implies

$$|e| > B(e)$$

- How to use zero bounds? Combine with approximation.
- Zero Bound is the bottleneck only in case of zero.

Adaptive Zero Determination

Core of Core Library

- Must use numerical method based on Zero Bounds
  - Must NOT use algebraic methods!
- Let $\Omega = \{+, -, \times, \ldots\} \cup \mathbb{Z}$ be a class of operators
  - $ZERO(\Omega)$ is the corresponding zero problem
- Zero Bound for Ω is a function $B : Expr(\Omega) \to \mathbb{R}_{\geq 0}$ such that $e \in Expr(\Omega)$ is non-zero implies

$$|e| > B(e)$$

- How to use zero bounds? Combine with approximation.
- Zero Bound is the bottleneck only in case of zero.

Adaptive Zero Determination

Core of Core Library

- Must use numerical method based on Zero Bounds
  - Must NOT use algebraic methods!
- Let $\Omega = \{+, -, \times, \ldots\} \cup \mathbb{Z}$ be a class of operators
  - $ZERO(\Omega)$ is the corresponding zero problem
- Zero Bound for Ω is a function $B : Expr(\Omega) \to \mathbb{R}_{\geq 0}$ such that $e \in Expr(\Omega)$ is non-zero implies

$$|e| > B(e)$$

- How to use zero bounds? Combine with approximation.
- Zero Bound is the bottleneck only in case of zero.

Some Constructive Bounds

- Degree-Measure Bounds [Mignotte (1982)], [Sekigawa (1997)]
- Degree-Height, Degree-Length [Yap-Dubé (1994)]
- BFMS Bound [Burnikel et al (1989)]
- Eigenvalue Bounds [Scheinerman (2000)]
- Conjugate Bounds [Li-Yap (2001)]
- BFMSS Bound [Burnikel et al (2001)]
    - One of the best bounds
- k-ary Method [Pion-Yap (2002)]
    - Idea: division is bad. k-ary numbers are good

An Example

- Consider the $e = \sqrt{x} + \sqrt{y} - \sqrt{x + y + 2\sqrt{xy}}$.
- Assume $x = a/b$ and $y = c/d$ where a, b, c, d are L-bit integers.
- Then Li-Yap Bound is $28L + 60$ bits, BFMSS is $96L + 30$ and Degree-Measure is $80L + 56$.
- Timing in seconds (Core 1.6):

L	50	100	500	5000
BFMS	0.637	9.12	101.9	202.9
Measure	0.063	0.07	1.93	15.26
BFMSS	0.073	0.61	1.95	15.41
Li-Yap	0.013	0.07	1.88	1.89

An Example

- Consider the $e = \sqrt{x} + \sqrt{y} - \sqrt{x + y + 2\sqrt{xy}}$.
- Assume $x = a/b$ and $y = c/d$ where a, b, c, d are L-bit integers.
- Then Li-Yap Bound is $28L + 60$ bits, BFMSS is $96L + 30$ and Degree-Measure is $80L + 56$.
- Timing in seconds (Core 1.6):

L	50	100	500	5000
BFMS	0.637	9.12	101.9	202.9
Measure	0.063	0.07	1.93	15.26
BFMSS	0.073	0.61	1.95	15.41
Li-Yap	0.013	0.07	1.88	1.89

An Example

- Consider the $e = \sqrt{x} + \sqrt{y} - \sqrt{x + y + 2\sqrt{xy}}$.
- Assume $x = a/b$ and $y = c/d$ where a, b, c, d are L-bit integers.
- Then Li-Yap Bound is $28L + 60$ bits, BFMSS is $96L + 30$ and Degree-Measure is $80L + 56$.
- Timing in seconds (Core 1.6):

L	50	100	500	5000
BFMS	0.637	9.12	101.9	202.9
Measure	0.063	0.07	1.93	15.26
BFMSS	0.073	0.61	1.95	15.41
Li-Yap	0.013	0.07	1.88	1.89

An Example

- Consider the $e = \sqrt{x} + \sqrt{y} - \sqrt{x + y + 2\sqrt{xy}}$.
- Assume $x = a/b$ and $y = c/d$ where a, b, c, d are L-bit integers.
- Then Li-Yap Bound is $28L + 60$ bits, BFMSS is $96L + 30$ and Degree-Measure is $80L + 56$.
- Timing in seconds (Core 1.6):

L	50	100	500	5000
BFMS	0.637	9.12	101.9	202.9
Measure	0.063	0.07	1.93	15.26
BFMSS	0.073	0.61	1.95	15.41
Li-Yap	0.013	0.07	1.88	1.89

An Example

- Consider the $e = \sqrt{x} + \sqrt{y} - \sqrt{x + y + 2\sqrt{xy}}$.
- Assume $x = a/b$ and $y = c/d$ where a, b, c, d are L-bit integers.
- Then Li-Yap Bound is $28L + 60$ bits, BFMSS is $96L + 30$ and Degree-Measure is $80L + 56$.
- Timing in seconds (Core 1.6):

L	50	100	500	5000
BFMS	0.637	9.12	101.9	202.9
Measure	0.063	0.07	1.93	15.26
BFMSS	0.073	0.61	1.95	15.41
Li-Yap	0.013	0.07	1.88	1.89

An Example

- Consider the $e = \sqrt{x} + \sqrt{y} - \sqrt{x + y + 2\sqrt{xy}}$.
- Assume $x = a/b$ and $y = c/d$ where a, b, c, d are L-bit integers.
- Then Li-Yap Bound is $28L + 60$ bits, BFMSS is $96L + 30$ and Degree-Measure is $80L + 56$.
- Timing in seconds (Core 1.6):

L	50	100	500	5000
BFMS	0.637	9.12	101.9	202.9
Measure	0.063	0.07	1.93	15.26
BFMSS	0.073	0.61	1.95	15.41
Li-Yap	0.013	0.07	1.88	1.89

Mini Summary

- There is a “Universal Solution” for synthesizing the Algebraic and the Geometric viewpoints
- Slogan: $\boxed{\text{Algebraic computation without Algebra}}$
  (Use approximations & zero bounds)
- PUZZLE 3: What was the answer to PUZZLE 2?

Mini Summary

- There is a "Universal Solution" for synthesizing the Algebraic and the Geometric viewpoints
- Slogan: $\boxed{\text{Algebraic computation without Algebra}}$
  (Use approximations & zero bounds)
- PUZZLE 3: What was the answer to PUZZLE 2?

Mini Summary

- There is a “Universal Solution” for synthesizing the Algebraic and the Geometric viewpoints
- Slogan: $\boxed{\text{Algebraic computation without Algebra}}$
  (Use approximations & zero bounds)
- PUZZLE 3: What was the answer to PUZZLE 2?

Mini Summary

- There is a “Universal Solution” for synthesizing the Algebraic and the Geometric viewpoints
- Slogan: $\boxed{\text{Algebraic computation without Algebra}}$
  (Use approximations & zero bounds)
- PUZZLE 3: What was the answer to PUZZLE 2?

Mini Summary

- There is a “Universal Solution” for synthesizing the Algebraic and the Geometric viewpoints
- Slogan: $\boxed{\text{Algebraic computation without Algebra}}$
  (Use approximations & zero bounds)
- PUZZLE 3: What was the answer to PUZZLE 2?

Summary of Lecture 1

- Nature of Geometric Computation:
  - Discrete as well as Continuous
  - Algebraic as well as Analytic
- It is possible to provide a fairly general solution (ENC) that combines the dual nature of numbers
- Up Next : Directly design ENC algorithms

Summary of Lecture 1

- Nature of Geometric Computation:
  - Discrete as well as Continuous
  - Algebraic as well as Analytic
- It is possible to provide a fairly general solution (ENC) that combines the dual nature of numbers
- Up Next : Directly design ENC algorithms

Summary of Lecture 1

- Nature of Geometric Computation:
  - Discrete as well as Continuous
  - Algebraic as well as Analytic
- It is possible to provide a fairly general solution (ENC) that combines the dual nature of numbers
- Up Next : Directly design ENC algorithms

Summary of Lecture 1

- Nature of Geometric Computation:
  - Discrete as well as Continuous
  - Algebraic as well as Analytic
- It is possible to provide a fairly general solution (ENC) that combines the dual nature of numbers
- Up Next : Directly design ENC algorithms

Summary of Lecture 1

- Nature of Geometric Computation:
  - Discrete as well as Continuous
  - Algebraic as well as Analytic
- It is possible to provide a fairly general solution (ENC) that combines the dual nature of numbers
- Up Next : Directly design ENC algorithms

Summary of Lecture 1

- Nature of Geometric Computation:
  - Discrete as well as Continuous
  - Algebraic as well as Analytic
- It is possible to provide a fairly general solution (ENC) that combines the dual nature of numbers
- Up Next : Directly design ENC algorithms

Summary of Lecture 1

- Nature of Geometric Computation:
  - Discrete as well as Continuous
  - Algebraic as well as Analytic
- It is possible to provide a fairly general solution (ENC) that combines the dual nature of numbers
- Up Next : Directly design ENC algorithms

PART 2

Explicitization and Subdivision

"It can be of no practical use to know that π is irrational, but if we can know, it surely would be intolerable not to know."

— E.C. Titchmarsh

Coming Up Next

Towards Exact Numerical Computation (ENC)

Beyond the Universal Solution

Design algorithms directly incorporating the principles of EGC

- What do we need? What are its features?
  - It must be numerical in nature
  - It must be arbitrary precision
  - It must respect zero
  - It must be adaptive
    - actively control precision
    - exploit filters

Towards Exact Numerical Computation (ENC)

Beyond the Universal Solution

Design algorithms directly incorporating the principles of EGC

- What do we need? What are its features?
  - It must be numerical in nature
  - It must be arbitrary precision
  - It must respect zero
  - It must be adaptive
    - actively control precision
    - exploit filters

Towards Exact Numerical Computation (ENC)

Beyond the Universal Solution

Design algorithms directly incorporating the principles of EGC

- What do we need? What are its features?
  - It must be numerical in nature
  - It must be arbitrary precision
  - It must respect zero
  - It must be adaptive
    - actively control precision
    - exploit filters

Towards Exact Numerical Computation (ENC)

Beyond the Universal Solution

Design algorithms directly incorporating the principles of EGC

- What do we need? What are its features?
  - It must be numerical in nature
  - It must be arbitrary precision
  - It must respect zero
  - It must be adaptive
    - actively control precision
    - exploit filters

Towards Exact Numerical Computation (ENC)

Beyond the Universal Solution

Design algorithms directly incorporating the principles of EGC

- What do we need? What are its features?
  - It must be numerical in nature
  - It must be arbitrary precision
  - It must respect zero
  - It must be adaptive
    - actively control precision
    - exploit filters

Towards Exact Numerical Computation (ENC)

Beyond the Universal Solution

Design algorithms directly incorporating the principles of EGC

- What do we need? What are its features?
  - It must be numerical in nature
  - It must be arbitrary precision
  - It must respect zero
  - It must be adaptive
    - actively control precision
    - exploit filters

Towards Exact Numerical Computation (ENC)

Beyond the Universal Solution

Design algorithms directly incorporating the principles of EGC

- What do we need? What are its features?
  - It must be numerical in nature
  - It must be arbitrary precision
  - It must respect zero
  - It must be adaptive
    - actively control precision
    - exploit filters

Towards Exact Numerical Computation (ENC)

Beyond the Universal Solution

Design algorithms directly incorporating the principles of EGC

- What do we need? What are its features?
  - It must be numerical in nature
  - It must be arbitrary precision
  - It must respect zero
  - It must be adaptive
    - actively control precision
    - exploit filters

Towards Exact Numerical Computation (ENC)

Beyond the Universal Solution

Design algorithms directly incorporating the principles of EGC

- What do we need? What are its features?
  - It must be numerical in nature
  - It must be arbitrary precision
  - It must respect zero
  - It must be adaptive
    - actively control precision
    - exploit filters

Computational Ring Approach

Computational Ring $(\mathbb{D}, 0, 1, +, -, \times, \div 2)$

- $\mathbb{D}$ is countable, dense subset of $\mathbb{R}$
- $\mathbb{D}$ is a ring extension of $\mathbb{Z}$
- Efficient representation $\rho : \{0,1\}^* \dashrightarrow \mathbb{D}$ for implementing ring operations, and exact comparison.

Examples of $\mathbb{D}$

- BigFloats or dyadic numbers:

$$\mathbb{F} := \{m2^n : m, n \in \mathbb{Z}\} = \mathbb{Z}[\frac{1}{2}]$$

- Rationals: $\mathbb{Q}$ (avoid, if possible)
- Real Algebraic Numbers: $\mathbb{A}$ (AVOID!)

Computational Ring Approach

Computational Ring $(\mathbb{D}, 0, 1, +, -, \times, \div 2)$

- $\mathbb{D}$ is countable, dense subset of $\mathbb{R}$
- $\mathbb{D}$ is a ring extension of $\mathbb{Z}$
- Efficient representation $\rho : \{0,1\}^* \dashrightarrow \mathbb{D}$ for implementing ring operations, and exact comparison.

Examples of $\mathbb{D}$

- BigFloats or dyadic numbers:

$$\mathbb{F} := \{m2^n : m, n \in \mathbb{Z}\} = \mathbb{Z}[\tfrac{1}{2}]$$

- Rationals: $\mathbb{Q}$ (avoid, if possible)
- Real Algebraic Numbers: $\mathbb{A}$ (AVOID!)

Computational Ring Approach

Computational Ring $(\mathbb{D}, 0, 1, +, -, \times, \div 2)$

- $\mathbb{D}$ is countable, dense subset of $\mathbb{R}$
- $\mathbb{D}$ is a ring extension of $\mathbb{Z}$
- Efficient representation $\rho : \{0,1\}^* \dashrightarrow \mathbb{D}$ for implementing ring operations, and exact comparison.

Examples of $\mathbb{D}$

- BigFloats or dyadic numbers:

$$\mathbb{F} := \{m2^n : m, n \in \mathbb{Z}\} = \mathbb{Z}[\frac{1}{2}]$$

- Rationals: $\mathbb{Q}$ (avoid, if possible)
- Real Algebraic Numbers: $\mathbb{A}$ (AVOID!)

Computational Ring Approach

Computational Ring $(\mathbb{D}, 0, 1, +, -, \times, \div 2)$

- $\mathbb{D}$ is countable, dense subset of $\mathbb{R}$
- $\mathbb{D}$ is a ring extension of $\mathbb{Z}$
- Efficient representation $\rho : \{0,1\}^* \dashrightarrow \mathbb{D}$ for implementing ring operations, and exact comparison.

Examples of $\mathbb{D}$

- BigFloats or dyadic numbers:

$$\mathbb{F} := \{m2^n : m, n \in \mathbb{Z}\} = \mathbb{Z}[\frac{1}{2}]$$

- Rationals: $\mathbb{Q}$ (avoid, if possible)
- Real Algebraic Numbers: $\mathbb{A}$ (AVOID!)

Computational Ring Approach

Computational Ring $(\mathbb{D}, 0, 1, +, -, \times, \div 2)$

- $\mathbb{D}$ is countable, dense subset of $\mathbb{R}$
- $\mathbb{D}$ is a ring extension of $\mathbb{Z}$
- Efficient representation $\rho : \{0,1\}^* \dashrightarrow \mathbb{D}$ for implementing ring operations, and exact comparison.

Examples of $\mathbb{D}$

- BigFloats or dyadic numbers:

$$\mathbb{F} := \{m2^n : m, n \in \mathbb{Z}\} = \mathbb{Z}[\frac{1}{2}]$$

- Rationals: $\mathbb{Q}$ (avoid, if possible)
- Real Algebraic Numbers: $\mathbb{A}$ (AVOID!)

Computational Ring Approach

Computational Ring $(\mathbb{D}, 0, 1, +, -, \times, \div 2)$

- $\mathbb{D}$ is countable, dense subset of $\mathbb{R}$
- $\mathbb{D}$ is a ring extension of $\mathbb{Z}$
- Efficient representation $\rho : \{0,1\}^* \dashrightarrow \mathbb{D}$ for implementing ring operations, and exact comparison.

Examples of $\mathbb{D}$

- BigFloats or dyadic numbers:

$$\mathbb{F} := \{m2^n : m, n \in \mathbb{Z}\} = \mathbb{Z}[\frac{1}{2}]$$

- Rationals: $\mathbb{Q}$ (avoid, if possible)
- Real Algebraic Numbers: $\mathbb{A}$ (AVOID!)

Computational Ring Approach

Computational Ring $(\mathbb{D}, 0, 1, +, -, \times, \div 2)$

- $\mathbb{D}$ is countable, dense subset of $\mathbb{R}$
- $\mathbb{D}$ is a ring extension of $\mathbb{Z}$
- Efficient representation $\rho : \{0,1\}^* \dashrightarrow \mathbb{D}$ for implementing ring operations, and exact comparison.

Examples of $\mathbb{D}$

- BigFloats or dyadic numbers:

$$\mathbb{F} := \{m2^n : m, n \in \mathbb{Z}\} = \mathbb{Z}[\frac{1}{2}]$$

- Rationals: $\mathbb{Q}$ (avoid, if possible)
- Real Algebraic Numbers: $\mathbb{A}$ (AVOID!)

Computational Ring Approach

Computational Ring $(\mathbb{D}, 0, 1, +, -, \times, \div 2)$

- $\mathbb{D}$ is countable, dense subset of $\mathbb{R}$
- $\mathbb{D}$ is a ring extension of $\mathbb{Z}$
- Efficient representation $\rho : \{0,1\}^* \dashrightarrow \mathbb{D}$ for implementing ring operations, and exact comparison.

Examples of $\mathbb{D}$

- BigFloats or dyadic numbers:

$$\mathbb{F} := \{m2^n : m, n \in \mathbb{Z}\} = \mathbb{Z}[\frac{1}{2}]$$

- Rationals: $\mathbb{Q}$ (avoid, if possible)
- Real Algebraic Numbers: $\mathbb{A}$ (AVOID!)

What else is needed in ENC Algorithms?

Intervals

- $\square\mathbb{D}$: set of dyadic intervals
- $\square\mathbb{D}^n$: set of n-boxes

Box Functions

Box function

What else is needed in ENC Algorithms?

Intervals

- $\square\mathbb{D}$: set of dyadic intervals
- $\square\mathbb{D}^n$: set of n-boxes

Box Functions

- Let $f : \mathbb{D}^m \longrightarrow \mathbb{D}$.
- Box function $\square f : \square^m(\mathbb{D}) \longrightarrow \square(\mathbb{D})$
  - (1) Inclusion
  - (2) Convergence

What else is needed in ENC Algorithms?

Intervals

- $\square\mathbb{D}$: set of dyadic intervals
- $\square\mathbb{D}^n$: set of n-boxes

Box Functions

- Let $f : \mathbb{D}^m \to \mathbb{D}$.
- Box function $\square f : \square^m(\mathbb{D}) \to \square(\mathbb{D})$

  (1) Inclusion

  (2) Convergence

What else is needed in ENC Algorithms?

Intervals

- $\square \mathbb{D}$: set of dyadic intervals
- $\square \mathbb{D}^n$: set of n-boxes

Box Functions

- Let $f : \mathbb{D}^m \to \mathbb{D}$.
- Box function $\square f : \square^m(\mathbb{D}) \to \square(\mathbb{D})$
  - (1) Inclusion: $f(B) \subseteq \square f(B)$.
  - (2) Convergence: $\lim_{i\to\infty} \square f(B_i) = f(\lim_{i\to\infty} \square B_i)$.

What else is needed in ENC Algorithms?

Intervals

- $\square\mathbb{D}$: set of dyadic intervals
- $\square\mathbb{D}^n$: set of n-boxes

Box Functions

- Let $f : \mathbb{D}^m \longrightarrow \mathbb{D}$.
- Box function $\square f : \square^m(\mathbb{D}) \longrightarrow \square(\mathbb{D})$
  - (1) Inclusion: $f(B) \subseteq \square f(B)$.
  - (2) Convergence: $\lim_{i\to\infty} \square f(B_i) = f(\lim_{i\to\infty} \square B_i)$.

What else is needed in ENC Algorithms?

Intervals

- $\square\mathbb{D}$: set of dyadic intervals
- $\square\mathbb{D}^n$: set of n-boxes

Box Functions

- Let $f : \mathbb{D}^m \longrightarrow \mathbb{D}$.
- Box function $\square f : \square^m(\mathbb{D}) \longrightarrow \square(\mathbb{D})$
  - (1) Inclusion: $f(B) \subseteq \square f(B)$.
  - (2) Convergence: $\lim_{i\to\infty} \square f(B_i) = f(\lim_{i\to\infty} \square B_i)$.

What else is needed in ENC Algorithms?

Intervals

- $\square\mathbb{D}$: set of dyadic intervals
- $\square\mathbb{D}^n$: set of n-boxes

Box Functions

- Let $f : \mathbb{D}^m \longrightarrow \mathbb{D}$.
- Box function $\square f : \square^m(\mathbb{D}) \longrightarrow \square(\mathbb{D})$
  - (1) Inclusion: $f(B) \subseteq \square f(B)$.
  - (2) Convergence: $\lim_{i\to\infty} \square f(B_i) = f(\lim_{i\to\infty} \square B_i)$.

What else is needed in ENC Algorithms?

Intervals

- $\square\mathbb{D}$: set of dyadic intervals
- $\square\mathbb{D}^n$: set of n-boxes

Box Functions

- Let $f : \mathbb{D}^m \longrightarrow \mathbb{D}$.
- Box function $\square f : \square^m(\mathbb{D}) \longrightarrow \square(\mathbb{D})$
  - (1) Inclusion: $f(B) \subseteq \square f(B)$.
  - (2) Convergence: $\lim_{i\to\infty} \square f(B_i) = f(\lim_{i\to\infty} \square B_i)$.

What else is needed in ENC Algorithms?

Intervals

- $\square\mathbb{D}$: set of dyadic intervals
- $\square\mathbb{D}^n$: set of n-boxes

Box Functions

- Let $f : \mathbb{D}^m \longrightarrow \mathbb{D}$.
- Box function $\square f : \square^m(\mathbb{D}) \longrightarrow \square(\mathbb{D})$
  - (1) Inclusion: $f(B) \subseteq \square f(B)$.
  - (2) Convergence: $\lim_{i\to\infty} \square f(B_i) = f(\lim_{i\to\infty} \square B_i)$.

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

- Why this class? Interface between Continuous and Discrete!
- ENC Algorithms is ideal for this class
- Interplay of Topological and Geometric requirements
- Domain subdivision as the general algorithmic paradigm

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

- Why this class? Interface between Continuous and Discrete!
- ENC Algorithms is ideal for this class
- Interplay of Topological and Geometric requirements
- Domain subdivision as the general algorithmic paradigm

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

- Why this class? Interface between Continuous and Discrete!
- ENC Algorithms is ideal for this class
- Interplay of Topological and Geometric requirements
- Domain subdivision as the general algorithmic paradigm

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

- Why this class? Interface between Continuous and Discrete!
- ENC Algorithms is ideal for this class
- Interplay of Topological and Geometric requirements
- Domain subdivision as the general algorithmic paradigm

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

- Why this class? Interface between Continuous and Discrete!
- ENC Algorithms is ideal for this class
- Interplay of Topological and Geometric requirements
- Domain subdivision as the general algorithmic paradigm

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

- Why this class? Interface between Continuous and Discrete!
- ENC Algorithms is ideal for this class
- Interplay of Topological and Geometric requirements
- Domain subdivision as the general algorithmic paradigm

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

- Why this class? Interface between Continuous and Discrete!
- ENC Algorithms is ideal for this class
- Interplay of Topological and Geometric requirements
- Domain subdivision as the general algorithmic paradigm

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

- Why this class? Interface between Continuous and Discrete!
- ENC Algorithms is ideal for this class
- Interplay of Topological and Geometric requirements
- Domain subdivision as the general algorithmic paradigm

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

- Why this class? Interface between Continuous and Discrete!
- ENC Algorithms is ideal for this class
- Interplay of Topological and Geometric requirements
- Domain subdivision as the general algorithmic paradigm

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

- Why this class? Interface between Continuous and Discrete!
- ENC Algorithms is ideal for this class
- Interplay of Topological and Geometric requirements
- Domain subdivision as the general algorithmic paradigm

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

- Why this class? Interface between Continuous and Discrete!
- ENC Algorithms is ideal for this class
- Interplay of Topological and Geometric requirements
- Domain subdivision as the general algorithmic paradigm

Our Target: Explicitization Problems

From Implicit to Explicit Representation

- Mesh generation [Problem (IV)]
- Discrete Morse-Smale complex [Problem (V)]
- Arrangement of hypersurfaces
- Voronoi diagram of a collection of objects
- Cell complex approximation of algebraic variety
- Representation of Flow fields

From Parameter Space to Ambient Space

- Why this class? Interface between Continuous and Discrete!
- ENC Algorithms is ideal for this class
- Interplay of Topological and Geometric requirements
- Domain subdivision as the general algorithmic paradigm

Three Approaches to Meshing, I:

1. Algebraic Approach

- Projection Based (Refinements of CAD)

  E.g., [Mourrain and Tecourt (2005); Cheng, Gao, and Li (2005)]
- Algebraic Subdivision Schemes

  E.g., [Wolpert and Seidel (2005)]
- Properties Exact; complete (usually); slow (in general); hard to implement

Three Approaches to Meshing, I:

1. Algebraic Approach

- **Projection Based** (Refinements of CAD)

  E.g., [Mourrain and Tecourt (2005); Cheng, Gao, and Li (2005)]

- **Algebraic Subdivision Schemes**

  E.g., [Wolpert and Seidel (2005)]

- **Properties** Exact; complete (usually); slow (in general); hard to implement

Three Approaches to Meshing, I:

1. Algebraic Approach

- Projection Based (Refinements of CAD)

  E.g., [Mourrain and Tecourt (2005); Cheng, Gao, and Li (2005)]
- Algebraic Subdivision Schemes

  E.g., [Wolpert and Seidel (2005)]
- Properties Exact; complete (usually); slow (in general); hard to implement

Three Approaches to Meshing, I:

1. Algebraic Approach

- Projection Based (Refinements of CAD)

  E.g., [Mourrain and Tecourt (2005); Cheng, Gao, and Li (2005)]

- Algebraic Subdivision Schemes

  E.g., [Wolpert and Seidel (2005)]

- Properties Exact; complete (usually); slow (in general); hard to implement

Three Approaches to Meshing, I:

1. Algebraic Approach

- Projection Based (Refinements of CAD)

  E.g., [Mourrain and Tecourt (2005); Cheng, Gao, and Li (2005)]

- Algebraic Subdivision Schemes

  E.g., [Wolpert and Seidel (2005)]

- Properties Exact; complete (usually); slow (in general); hard to implement

Three Approaches to Meshing, II:

2. Geometric Approach

- Sampling Approach (Ray Shooting)

  E.g., [Boissonnat & Oudot (2005); Cheng, Dey, Ramos and Ray (2004)]
- Morse theory

  E.g., [Stander & Hart (1997); Boissonnat, Cohen-Steiner & Vegter (2004)]
- Properties Implementation gaps; requires “niceness conditions” (Morseness, non-singularity, etc)

Three Approaches to Meshing, II:

2. Geometric Approach

- Sampling Approach (Ray Shooting)

  E.g., [Boissonnat & Oudot (2005); Cheng, Dey, Ramos and Ray (2004)]

- Morse theory

  E.g., [Stander & Hart (1997); Boissonnat, Cohen-Steiner & Vegter (2004)]

- Properties Implementation gaps; requires “niceness conditions” (Morseness, non-singularity, etc)

Three Approaches to Meshing, II:

2. Geometric Approach

- Sampling Approach (Ray Shooting)

  E.g., [Boissonnat & Oudot (2005); Cheng, Dey, Ramos and Ray (2004)]

- Morse theory

  E.g., [Stander & Hart (1997); Boissonnat, Cohen-Steiner & Vegter (2004)]

- Properties Implementation gaps; requires "niceness conditions" (Morseness, non-singularity, etc)

Three Approaches to Meshing, II:

2. Geometric Approach

- Sampling Approach (Ray Shooting)

  E.g., [Boissonnat & Oudot (2005); Cheng, Dey, Ramos and Ray (2004)]

- Morse theory

  E.g., [Stander & Hart (1997); Boissonnat, Cohen-Steiner & Vegter (2004)]

- Properties Implementation gaps; requires “niceness conditions” (Morseness, non-singularity, etc)

Three Approaches to Meshing, II:

2. Geometric Approach

- Sampling Approach (Ray Shooting)

  E.g., [Boissonnat & Oudot (2005); Cheng, Dey, Ramos and Ray (2004)]

- Morse theory

  E.g., [Stander & Hart (1997); Boissonnat, Cohen-Steiner & Vegter (2004)]

- Properties Implementation gaps; requires “niceness conditions” (Morseness, non-singularity, etc)

Three Approaches to Meshing, III:

3. Numeric Approach

- Curve Tracing Literature

  [Ratschek & Rokne (2005)]
- Subdivision Approach

  [Marching Cube (1987); Snyder (1992); Plantinga & Vegter (2004)]
- Properties Practical; easy to implement; adaptive; incomplete (until recently)
- This is our focus

Three Approaches to Meshing, III:

3. Numeric Approach

- Curve Tracing Literature

  [Ratschek & Rokne (2005)]

- Subdivision Approach

  [Marching Cube (1987); Snyder (1992); Plantinga & Vegter (2004)]

- Properties Practical; easy to implement; adaptive; incomplete (until recently)

- This is our focus

Three Approaches to Meshing, III:

3. Numeric Approach

- Curve Tracing Literature

  [Ratschek & Rokne (2005)]

- Subdivision Approach

  [Marching Cube (1987); Snyder (1992); Plantinga & Vegter (2004)]

- Properties Practical; easy to implement; adaptive; incomplete (until recently)

- This is our focus

Three Approaches to Meshing, III:

3. Numeric Approach

- Curve Tracing Literature

  [Ratschek & Rokne (2005)]

- Subdivision Approach

  [Marching Cube (1987); Snyder (1992); Plantinga & Vegter (2004)]

- Properties Practical; easy to implement; adaptive; incomplete (until recently)

- This is our focus

Three Approaches to Meshing, III:

3. Numeric Approach

- Curve Tracing Literature

  [Ratschek & Rokne (2005)]

- Subdivision Approach

  [Marching Cube (1987); Snyder (1992); Plantinga & Vegter (2004)]

- Properties Practical; easy to implement; adaptive; incomplete (until recently)

- This is our focus

Three Approaches to Meshing, III:

3. Numeric Approach

- Curve Tracing Literature

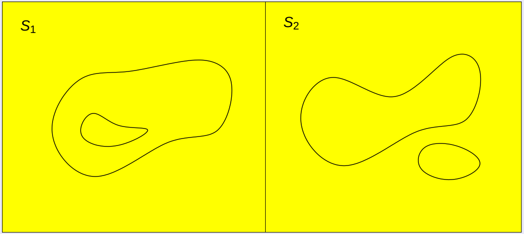
  [Ratschek & Rokne (2005)]
- Subdivision Approach

  [Marching Cube (1987); Snyder (1992); Plantinga & Vegter (2004)]
- Properties Practical; easy to implement; adaptive; incomplete (until recently)
- This is our focus

Two Criteria of Meshing

I. Topological Correctness

The approximation $\widetilde{S}$ is isotopic to the S.



- S_1 and S_2 are homeomorphic, but not isotopic
- Ambient space property!

(contd.) Two Criteria of Meshing

II. Geometrical Accuracy (ε-closeness)

For any given $\varepsilon > 0$, the Hausdorff distance $d(S, \widetilde{S})$ should not exceed ε.

- Set $\varepsilon = \infty$ to focus on isotopy.

Mini Summary

- Want ENC algorithms for Explicitization Problems
- Focus on (purely) Numerical Subdivision methods
- Algorithms for Meshing Curves (and Surfaces)
- What will be New?
  Numerical methods that are exact and can handle singularities

Mini Summary

- Want ENC algorithms for Explicitization Problems
- Focus on (purely) Numerical Subdivision methods
- Algorithms for Meshing Curves (and Surfaces)
- What will be New?

  Numerical methods that are exact and can handle singularities

Mini Summary

- Want ENC algorithms for Explicitization Problems
- Focus on (purely) Numerical Subdivision methods
- Algorithms for Meshing Curves (and Surfaces)
- What will be New?
  Numerical methods that are exact and can handle singularities

Mini Summary

- Want ENC algorithms for Explicitization Problems
- Focus on (purely) Numerical Subdivision methods
- Algorithms for Meshing Curves (and Surfaces)
- What will be New?
  Numerical methods that are exact and can handle singularities

Mini Summary

- Want ENC algorithms for Explicitization Problems
- Focus on (purely) Numerical Subdivision methods
- Algorithms for Meshing Curves (and Surfaces)
- What will be New?
  Numerical methods that are exact and can handle singularities

Mini Summary

- Want ENC algorithms for Explicitization Problems
- Focus on (purely) Numerical Subdivision methods
- Algorithms for Meshing Curves (and Surfaces)
- What will be New?
  Numerical methods that are exact and can handle singularities

Coming Up Next

Subdivision Algorithms

- Viewed as generalized binary search, organized as a quadtree.
- Here is a typical output:

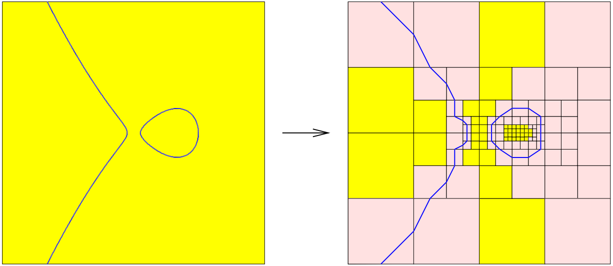


Figure: Approximation of the curve $f(X, Y) = Y^2 - X^2 + X^3 + 0.02 = 0$

The Generic Subdivision Algorithm

- INPUT: Curve $S = f^{-1}(0)$, box $B_0 \subseteq \mathbb{R}^2$, and $\varepsilon > 0$
- OUTPUT: Graph $G = (V, E)$, representing an isotopic ε-approximation of $S \cap B_0$.

1. Let $Q_{in} \leftarrow \{B_0\}$ be a queue of boxes
2. SUBDIVISION PHASE: $Q_{out} \leftarrow SUBDIVIDE(Q_{in})$
3. REFINEMENT PHASE: $Q_{ref} \leftarrow REFINE(Q_{out})$
4. CONSTRUCTION PHASE: $G \leftarrow CONSTRUCT(Q_{ref})$

The Generic Subdivision Algorithm

- INPUT: Curve $S = f^{-1}(0)$, box $B_0 \subseteq \mathbb{R}^2$, and $\varepsilon > 0$
- OUTPUT: Graph $G = (V, E)$,
  representing an isotopic ε-approximation of $S \cap B_0$.

1. Let $Q_{in} \leftarrow \{B_0\}$ be a queue of boxes
2. SUBDIVISION PHASE: $Q_{out} \leftarrow SUBDIVIDE(Q_{in})$
3. REFINEMENT PHASE: $Q_{ref} \leftarrow REFINE(Q_{out})$
4. CONSTRUCTION PHASE: $G \leftarrow CONSTRUCT(Q_{ref})$

The Generic Subdivision Algorithm

- INPUT: Curve $S = f^{-1}(0)$, box $B_0 \subseteq \mathbb{R}^2$, and $\varepsilon > 0$
- OUTPUT: Graph $G = (V, E)$,
  representing an isotopic ε-approximation of $S \cap B_0$.

1. Let $Q_{in} \leftarrow \{B_0\}$ be a queue of boxes
2. SUBDIVISION PHASE: $Q_{out} \leftarrow SUBDIVIDE(Q_{in})$
3. REFINEMENT PHASE: $Q_{ref} \leftarrow REFINE(Q_{out})$
4. CONSTRUCTION PHASE: $G \leftarrow CONSTRUCT(Q_{ref})$

The Generic Subdivision Algorithm

- INPUT: Curve $S = f^{-1}(0)$, box $B_0 \subseteq \mathbb{R}^2$, and $\varepsilon > 0$
- OUTPUT: Graph $G = (V, E)$,
  representing an isotopic ε-approximation of $S \cap B_0$.

1. Let $Q_{in} \leftarrow \{B_0\}$ be a queue of boxes
2. SUBDIVISION PHASE: $Q_{out} \leftarrow SUBDIVIDE(Q_{in})$
3. REFINEMENT PHASE: $Q_{ref} \leftarrow REFINE(Q_{out})$
4. CONSTRUCTION PHASE: $G \leftarrow CONSTRUCT(Q_{ref})$

The Generic Subdivision Algorithm

- INPUT: Curve $S = f^{-1}(0)$, box $B_0 \subseteq \mathbb{R}^2$, and $\varepsilon > 0$
- OUTPUT: Graph $G = (V, E)$,
  representing an isotopic ε-approximation of $S \cap B_0$.

1. Let $Q_{in} \leftarrow \{B_0\}$ be a queue of boxes
2. SUBDIVISION PHASE: $Q_{out} \leftarrow SUBDIVIDE(Q_{in})$
3. REFINEMENT PHASE: $Q_{ref} \leftarrow REFINE(Q_{out})$
4. CONSTRUCTION PHASE: $G \leftarrow CONSTRUCT(Q_{ref})$

The Generic Subdivision Algorithm

- INPUT: Curve $S = f^{-1}(0)$, box $B_0 \subseteq \mathbb{R}^2$, and $\varepsilon > 0$
- OUTPUT: Graph $G = (V, E)$,
  representing an isotopic ε-approximation of $S \cap B_0$.

1. Let $Q_{in} \leftarrow \{B_0\}$ be a queue of boxes
2. SUBDIVISION PHASE: $Q_{out} \leftarrow SUBDIVIDE(Q_{in})$
3. REFINEMENT PHASE: $Q_{ref} \leftarrow REFINE(Q_{out})$
4. CONSTRUCTION PHASE: $G \leftarrow CONSTRUCT(Q_{ref})$

The Generic Subdivision Algorithm

- INPUT: Curve $S = f^{-1}(0)$, box $B_0 \subseteq \mathbb{R}^2$, and $\varepsilon > 0$
- OUTPUT: Graph $G = (V, E)$,
  representing an isotopic ε-approximation of $S \cap B_0$.

1. Let $Q_{in} \leftarrow \{B_0\}$ be a queue of boxes
2. SUBDIVISION PHASE: $Q_{out} \leftarrow SUBDIVIDE(Q_{in})$
3. REFINEMENT PHASE: $Q_{ref} \leftarrow REFINE(Q_{out})$
4. CONSTRUCTION PHASE: $G \leftarrow CONSTRUCT(Q_{ref})$

The Generic Subdivision Algorithm

- INPUT: Curve $S = f^{-1}(0)$, box $B_0 \subseteq \mathbb{R}^2$, and $\varepsilon > 0$
- OUTPUT: Graph $G = (V, E)$,
  representing an isotopic ε-approximation of $S \cap B_0$.

1. Let $Q_{in} \leftarrow \{B_0\}$ be a queue of boxes
2. SUBDIVISION PHASE: $Q_{out} \leftarrow SUBDIVIDE(Q_{in})$
3. REFINEMENT PHASE: $Q_{ref} \leftarrow REFINE(Q_{out})$
4. CONSTRUCTION PHASE: $G \leftarrow CONSTRUCT(Q_{ref})$

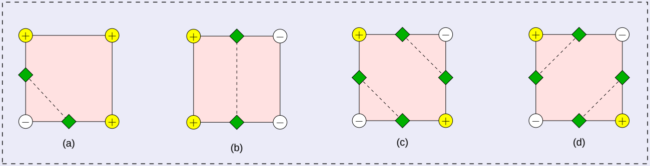
E.g., Marching Cube

Subdivision Phase

Subdivide until size of each box $\leq \varepsilon$.

Construction Phase

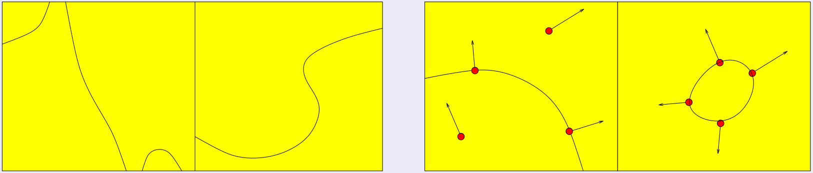
(1) Evaluate sign of f at grid points, (2) insert vertices, and (3) connect them in each box:



Cannot guarantee the topological correctness

Parametrizability and Normal Variation

Parametrizable in X-direction



(a) (b) (c) (d)

- (a) Parametrizable in X-direction
- (b) Non-parametrizable in X- or Y-direction
- (c) Small normal variation
- (d) Big normal variation

Box Predicates

Three Conditions (Predicates)

	C0	$0 \notin \square f(B)$	Exclusion
•	Cxy	$0 \notin \square f_x(B)$ or $0 \notin \square f_y(B)$	Parametrizability
	C1	$0 \notin \square f_x(B)^2 + \square f_y(B)^2$	Small Normal Variation

Implementation: e.g., $f(x,y) = x^2 - 2xy + 3y$

Box Predicates

Three Conditions (Predicates)

	C0	$0 \notin \Box f(B)$	Exclusion
•	Cxy	$0 \notin \Box f_x(B)$ or $0 \notin \Box f_y(B)$	Parametrizability
	C1	$0 \notin \Box f_x(B)^2 + \Box f_y(B)^2$	Small Normal Variation

Implementation: e.g., $f(x,y) = x^2 - 2xy + 3y$

- Interval Arithmetic (Box):
- Interval Taylor (Disc):

Box Predicates

Three Conditions (Predicates)

	C0	$0 \notin \square f(B)$	Exclusion
•	Cxy	$0 \notin \square f_x(B)$ or $0 \notin \square f_y(B)$	Parametrizability
	C1	$0 \notin \square f_x(B)^2 + \square f_y(B)^2$	Small Normal Variation

Implementation: e.g., $f(x,y) = x^2 - 2xy + 3y$

- Interval Arithmetic (Box):
- Interval Taylor (Disc):

Box Predicates

Three Conditions (Predicates)

	C0	$0 \notin \square f(B)$	Exclusion
•	Cxy	$0 \notin \square f_x(B)$ or $0 \notin \square f_y(B)$	Parametrizability
	C1	$0 \notin \square f_x(B)^2 + \square f_y(B)^2$	Small Normal Variation

Implementation: e.g., $f(x,y) = x^2 - 2xy + 3y$

- Interval Arithmetic (Box):
- Interval Taylor (Disc):

Box Predicates

Three Conditions (Predicates)

	C0	$0 \notin \square f(B)$	Exclusion
•	Cxy	$0 \notin \square f_x(B)$ or $0 \notin \square f_y(B)$	Parametrizability
	C1	$0 \notin \square f_x(B)^2 + \square f_y(B)^2$	Small Normal Variation

Implementation: e.g., $f(x,y) = x^2 - 2xy + 3y$

- Interval Arithmetic (Box):
- Interval Taylor (Disc):

Box Predicates

Three Conditions (Predicates)

C0	$0 \notin \square f(B)$	Exclusion
Cxy	$0 \notin \square f_x(B)$ or $0 \notin \square f_y(B)$	Parametrizability
C1	$0 \notin \square f_x(B)^2 + \square f_y(B)^2$	Small Normal Variation

Implementation: e.g., $f(x,y) = x^2 - 2xy + 3y$

- Interval Arithmetic (Box):
  - $\square f(I,J) = I^2 - 2IJ + 3J$
- Interval Taylor (Disc):
  - $\square f(x,y,r) = [f(x,y) \pm r(|2(x-y)| + |-2x+3| + 3r^2)]$

Box Predicates

Three Conditions (Predicates)

C0	$0 \notin \square f(B)$	Exclusion
Cxy	$0 \notin \square f_x(B)$ or $0 \notin \square f_y(B)$	Parametrizability
C1	$0 \notin \square f_x(B)^2 + \square f_y(B)^2$	Small Normal Variation

Implementation: e.g., $f(x,y) = x^2 - 2xy + 3y$

- Interval Arithmetic (Box):
  - $\square f(I,J) = I^2 - 2IJ + 3J$
- Interval Taylor (Disc):
  - $\square f(x,y,r) = [f(x,y) \pm r(|2(x-y)| + |-2x+3| + 3r^2)]$

Box Predicates

Three Conditions (Predicates)

C0	$0 \notin \square f(B)$	Exclusion
Cxy	$0 \notin \square f_x(B)$ or $0 \notin \square f_y(B)$	Parametrizability
C1	$0 \notin \square f_x(B)^2 + \square f_y(B)^2$	Small Normal Variation

Implementation: e.g., $f(x,y) = x^2 - 2xy + 3y$

- Interval Arithmetic (Box):
  - $\square f(I,J) = I^2 - 2IJ + 3J$
- Interval Taylor (Disc):
  - $\square f(x,y,r) = [f(x,y) \pm r(|2(x-y)| + |-2x+3| + 3r^2)]$

Box Predicates

Three Conditions (Predicates)

C0	$0 \notin \Box f(B)$	Exclusion
Cxy	$0 \notin \Box f_x(B)$ or $0 \notin \Box f_y(B)$	Parametrizability
C1	$0 \notin \Box f_x(B)^2 + \Box f_y(B)^2$	Small Normal Variation

Implementation: e.g., $f(x,y) = x^2 - 2xy + 3y$

- Interval Arithmetic (Box):
  - $\Box f(I,J) = I^2 - 2IJ + 3J$
- Interval Taylor (Disc):
  - $\Box f(x,y,r) = [f(x,y) \pm r(|2(x-y)| + |-2x+3| + 3r^2)]$

Box Predicates

Three Conditions (Predicates)

C0	$0 \notin \Box f(B)$	Exclusion
Cxy	$0 \notin \Box f_x(B)$ or $0 \notin \Box f_y(B)$	Parametrizability
C1	$0 \notin \Box f_x(B)^2 + \Box f_y(B)^2$	Small Normal Variation

Implementation: e.g., $f(x,y) = x^2 - 2xy + 3y$

- Interval Arithmetic (Box):
  - $\Box f(I,J) = I^2 - 2IJ + 3J$
- Interval Taylor (Disc):
  - $\Box f(x,y,r) = [f(x,y) \pm r(|2(x-y)| + |-2x+3| + 3r^2)]$

Box Predicates

Three Conditions (Predicates)

C0	$0 \notin \square f(B)$	Exclusion
Cxy	$0 \notin \square f_x(B)$ or $0 \notin \square f_y(B)$	Parametrizability
C1	$0 \notin \square f_x(B)^2 + \square f_y(B)^2$	Small Normal Variation

Implementation: e.g., $f(x,y) = x^2 - 2xy + 3y$

- Interval Arithmetic (Box):
  - $\square f(I,J) = I^2 - 2IJ + 3J$
- Interval Taylor (Disc):
  - $\square f(x,y,r) = [f(x,y) \pm r(|2(x-y)| + |-2x+3| + 3r^2)]$

Snyder's Algorithm

Subdivision Phase

For each box B:

- $C_0(B) \Rightarrow$ discard
- $\neg C_{xy}(B) \Rightarrow$ subdivide B

Construction Phase

- Determine intersections on boundary
- Connect the intersections
- (Non-trivial, unbounded complexity)

Boundary Analysis is not good (may not even terminate).

Snyder's Algorithm

Subdivision Phase

For each box B:

- $C_0(B) \Rightarrow$ discard
- $\neg C_{xy}(B) \Rightarrow$ subdivide B

Construction Phase

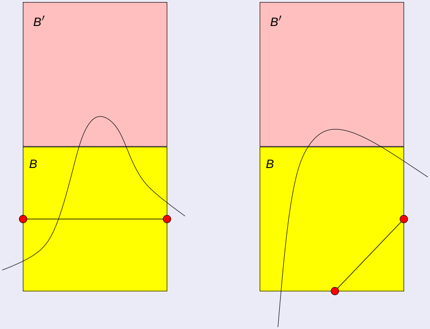
- Determine intersections on boundary
- Connect the intersections
- (Non-trivial, unbounded complexity)

Boundary Analysis is not good (may not even terminate).

Idea of Plantinga and Vegter

Introduce a strong predicate $C1$ predicate

- Allow local NON-isotopy
  Local incursion and excursions



- Locally, graph is not isotopic

- Simple box geometry
  (simpler than Snyder, less simple than Marching Cube)

Plantinga and Vegter's Algorithm

Exploit the global isotopy

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

(contd.) Plantinga and Vegter's Algorithm

Global, not local, isotopy

- **Construction Phase:**

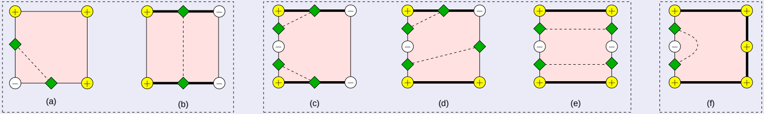


Figure: Extended Rules

- Local isotopy is NOT good !

Coming Up Next

Idea of Cxy Algorithm

Replace C1 by Cxy

- $C_1(B)$ implies $C_{xy}(B)$
- This would produce fewer boxes.

Exploit local non-isotopy

- Local isotopy is an artifact!
- This also avoid boundary analysis.

Obstructions to Cxy Idea

Replace C1 by Cxy

- Just run PV Algorithm but using C_{xy} instead:
- What can go wrong?

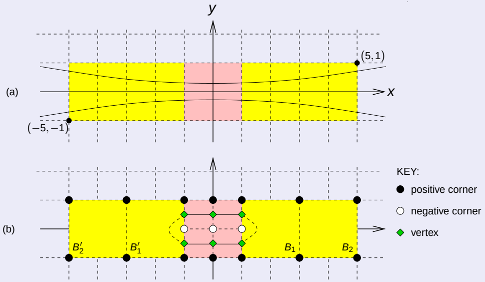


Figure: Elongated hyperbola

Cxy Algorithm

- Subdivision and Refinement Phases: As before
- Construction Phase:

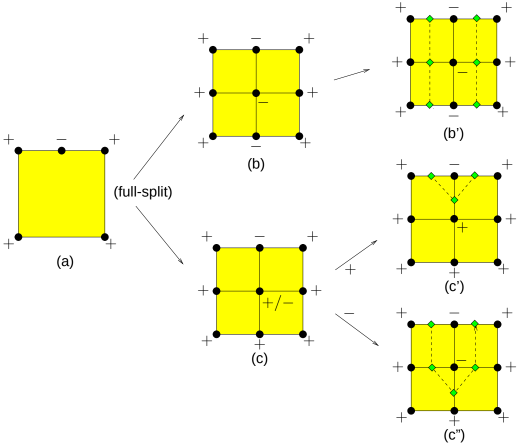


Figure: Resolution of Ambiguity

Mini Summary

- What has Cxy Algorithm done?
  - Exploit Parametrizability (like Snyder)
  - Rejected local isotopy (like PV)
- Up Next: More improvements

Mini Summary

- What has Cxy Algorithm done?
  - Exploit Parametrizability (like Snyder)
  - Rejected local isotopy (like PV)
- Up Next: More improvements

Mini Summary

- What has Cxy Algorithm done?
  - Exploit Parametrizability (like Snyder)
  - Rejected local isotopy (like PV)
- Up Next: More improvements

Mini Summary

- What has Cxy Algorithm done?
  - Exploit Parametrizability (like Snyder)
  - Rejected local isotopy (like PV)
- Up Next: More improvements

Mini Summary

- What has Cxy Algorithm done?
  - Exploit Parametrizability (like Snyder)
  - Rejected local isotopy (like PV)
- Up Next: More improvements

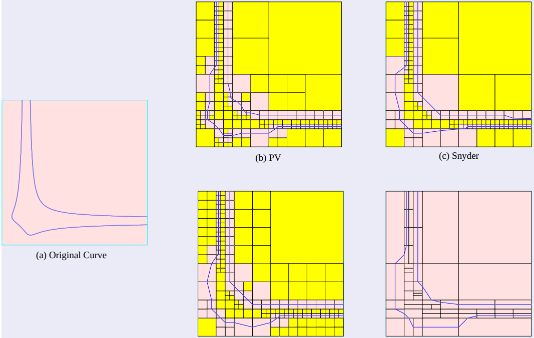
Mini Summary

- What has Cxy Algorithm done?
  - Exploit Parametrizability (like Snyder)
  - Rejected local isotopy (like PV)
- Up Next: More improvements

Coming Up Next

Idea of Rectangular Cxy Algorithm

Exploit Anisotropy



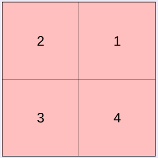
(a) Original Curve (b) PV (c) Snyder (d) Balanced Cxy (e) Rectangular Cxy

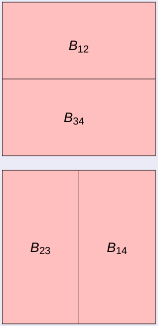
- "Heel Curve" $X^2Y^2 - X + Y - 1 = 0$ in box $B = [(-2,-10),(10,2)]$
- Comparing PV, Snyder, Cxy, Rect Cxy

Partial Splits for Rectangles

Splits

Full-splits:
$B \longrightarrow (B_1, B_2, B_3, B_4)$





- Horizontal Half-split:

  $B \longrightarrow (B_{12}, B_{34})$

- Vertical Half-split:

  $B \longrightarrow (B_{14}, B_{23})$

Rectangular Cxy Algorithm

What is needed

- Aspect Ratio Bound: $r > 1$ arbitrary but fixed.
- Splitting Procedure: do full-split if none of these hold

L_0 :	$C_0(B), C_{xy}(B)$	Terminate
L_{out} :	$C_0(B_{12}), C_0(B_{34}), C_0(B_{14}), C_0(B_{23})$	Half-split
L_{in} :	$C_{xy}(B_{12}), C_{xy}(B_{34}), C_{xy}(B_{14}), C_{xy}(B_{23})$	Half-split

- Axis-dependent balancing: each node has a X-depth and Y-depth.

Rectangular Cxy Algorithm

What is needed

- Aspect Ratio Bound: $r > 1$ arbitrary but fixed.
- Splitting Procedure: do full-split if none of these hold

L_0 :	$C_0(B), C_{xy}(B)$	Terminate
L_{out} :	$C_0(B_{12}), C_0(B_{34}), C_0(B_{14}), C_0(B_{23})$	Half-split
L_{in} :	$C_{xy}(B_{12}), C_{xy}(B_{34}), C_{xy}(B_{14}), C_{xy}(B_{23})$	Half-split

- Axis-dependent balancing: each node has a X-depth and Y-depth.

Rectangular Cxy Algorithm

What is needed

- Aspect Ratio Bound: $r > 1$ arbitrary but fixed.
- Splitting Procedure: do full-split if none of these hold

L_0 :	$C_0(B), C_{xy}(B)$	Terminate
L_{out} :	$C_0(B_{12}), C_0(B_{34}), C_0(B_{14}), C_0(B_{23})$	Half-split
L_{in} :	$C_{xy}(B_{12}), C_{xy}(B_{34}), C_{xy}(B_{14}), C_{xy}(B_{23})$	Half-split

- Axis-dependent balancing: each node has a X-depth and Y-depth.

Rectangular Cxy Algorithm

What is needed

- Aspect Ratio Bound: $r > 1$ arbitrary but fixed.
- Splitting Procedure: do full-split if none of these hold

L_0 :	$C_0(B), C_{xy}(B)$	Terminate
L_{out} :	$C_0(B_{12}), C_0(B_{34}), C_0(B_{14}), C_0(B_{23})$	Half-split
L_{in} :	$C_{xy}(B_{12}), C_{xy}(B_{34}), C_{xy}(B_{14}), C_{xy}(B_{23})$	Half-split

- Axis-dependent balancing: each node has a X-depth and Y-depth.

Rectangular Cxy Algorithm

What is needed

- Aspect Ratio Bound: $r > 1$ arbitrary but fixed.
- Splitting Procedure: do full-split if none of these hold

L_0 :	$C_0(B), C_{xy}(B)$	Terminate
L_{out} :	$C_0(B_{12}), C_0(B_{34}), C_0(B_{14}), C_0(B_{23})$	Half-split
L_{in} :	$C_{xy}(B_{12}), C_{xy}(B_{34}), C_{xy}(B_{14}), C_{xy}(B_{23})$	Half-split

- Axis-dependent balancing: each node has a X-depth and Y-depth.

Rectangular Cxy Algorithm

What is needed

- Aspect Ratio Bound: $r > 1$ arbitrary but fixed.
- Splitting Procedure: do full-split if none of these hold

L_0 :	$C_0(B), C_{xy}(B)$	Terminate
L_{out} :	$C_0(B_{12}), C_0(B_{34}), C_0(B_{14}), C_0(B_{23})$	Half-split
L_{in} :	$C_{xy}(B_{12}), C_{xy}(B_{34}), C_{xy}(B_{14}), C_{xy}(B_{23})$	Half-split

- Axis-dependent balancing: each node has a X-depth and Y-depth.

Rectangular Cxy Algorithm

What is needed

- Aspect Ratio Bound: $r > 1$ arbitrary but fixed.
- Splitting Procedure: do full-split if none of these hold

L_0 :	$C_0(B), C_{xy}(B)$	Terminate
L_{out} :	$C_0(B_{12}), C_0(B_{34}), C_0(B_{14}), C_0(B_{23})$	Half-split
L_{in} :	$C_{xy}(B_{12}), C_{xy}(B_{34}), C_{xy}(B_{14}), C_{xy}(B_{23})$	Half-split

- Axis-dependent balancing: each node has a X-depth and Y-depth.

Rectangular Cxy Algorithm

What is needed

- Aspect Ratio Bound: $r > 1$ arbitrary but fixed.
- Splitting Procedure: do full-split if none of these hold

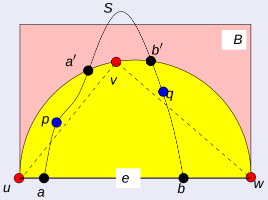
L_0 :	$C_0(B), C_{xy}(B)$	Terminate
L_{out} :	$C_0(B_{12}), C_0(B_{34}), C_0(B_{14}), C_0(B_{23})$	Half-split
L_{in} :	$C_{xy}(B_{12}), C_{xy}(B_{34}), C_{xy}(B_{14}), C_{xy}(B_{23})$	Half-split

- Axis-dependent balancing: each node has a X-depth and Y-depth.

Ensuring Geometric Accuracy

Buffer Property of C1 predicate

- Aspect Ratio ≤ 2:



Half-circle argument

- Generalize $C_1(B)$ to $C_1^*(B)$. for any box B

Comparisons

- Compare Rect Cxy to PV (note: Snyder has degeneracy).
  - Curve $X(XY-1)=0$, box $B_s := [(-s,-s),(s,s)]$, Aspect ratio bound $r=5$: (JSO=Java stack overflow)

#Boxes/Time(ms)	$s=15$	$s=60$	$s=100$
PV	5686/157	JSO	JSO
Cxy	2878/125	45790/2750	JSO
Rect	258/32	3847/766	11196/7781

- Increasing r can increase the performance of Rect Cxy.
  - $r=80, s=100 \Rightarrow Boxes/Time(ms) = 751/78$

Comparisons (2)

- Compare to Snyder's Algorithm.
  - Curve $X(XY-1)=0$, box $B_n := [(-14\times 10^n, -14\times 10^n), (15\times 10^n, 15\times 10^n)]$. Maximum aspect ratio $r = 257$.

#Boxes/Time(ms)	$n=-1$	$n=0$	$n=1$
Snyder	10/15	1306/125	JSO
Cxy	13/0	1510/62	JSO
Rect	6/0	13/0	256/47

Comparisons (2)

- Compare to Snyder's Algorithm.
  - Curve $X(XY-1)=0$, box $B_n := [(-14\times 10^n, -14\times 10^n), (15\times 10^n, 15\times 10^n)]$. Maximum aspect ratio $r = 257$.

#Boxes/Time(ms)	$n=-1$	$n=0$	$n=1$
Snyder	10/15	1306/125	JSO
Cxy	13/0	1510/62	JSO
Rect	6/0	13/0	256/47

Comparisons (2)

- Compare to Snyder's Algorithm.
  - Curve $X(XY-1)=0$, box $B_n := [(-14\times 10^n, -14\times 10^n), (15\times 10^n, 15\times 10^n)]$. Maximum aspect ratio $r = 257$.

#Boxes/Time(ms)	$n=-1$	$n=0$	$n=1$
Snyder	10/15	1306/125	JSO
Cxy	13/0	1510/62	JSO
Rect	6/0	13/0	256/47

Summary of Experimental Results

- Cxy combines the advantages of Snyder & PV Algorithms.
- Can be significantly faster than PV & Snyder's algorithm.
- Rectangular Cxy Algorithm can be significantly faster than Balanced Cxy algorithm.

Coming Up Next

Boundary (Summary)

- An Obvious Way and a Better Way
  - Exact Way : Recursively solve the problem on ∂B_0
  - Better Way : Exploit isotopy
- Price for Better Way: Weaker Correctness Statement

$$\text{For some } B_0 \subseteq B_0^+ \subseteq B_0 \oplus B(\varepsilon),$$
$$G \text{ is isotopic to } S \cap B_0^+ .$$

- APPLICATIONS:
  - Singularity (below)
  - Input region B_0 to have "any" geometry (even holes), provided it contains no singularities.

Boundary (Summary)

- An Obvious Way and a Better Way
  - Exact Way : Recursively solve the problem on ∂B_0
  - Better Way : Exploit isotopy
- Price for Better Way: Weaker Correctness Statement

$$\text{For some } B_0 \subseteq B_0^+ \subseteq B_0 \oplus B(\varepsilon),$$
$$G \text{ is isotopic to } S \cap B_0^+ .$$

- APPLICATIONS:
  - Singularity (below)
  - Input region B_0 to have "any" geometry, even holes, provided it contains no singularities

Boundary (Summary)

- An Obvious Way and a Better Way
  - Exact Way : Recursively solve the problem on ∂B_0
  - Better Way : Exploit isotopy

Boundary (Summary)

- An Obvious Way and a Better Way
  - Exact Way : Recursively solve the problem on ∂B_0
  - Better Way : Exploit isotopy
- Price for Better Way: Weaker Correctness Statement

  For some $B_0 \subseteq B_0^+ \subseteq B_0 \oplus B(\varepsilon)$,

  G is isotopic to $S \cap B_0^+$.
- APPLICATIONS:

  Singularity (below)

  Input region B_0 to have "any" geometry, even holes, provided it contains no singularities.

Boundary (Summary)

- An Obvious Way and a Better Way
  - Exact Way : Recursively solve the problem on ∂B_0
  - Better Way : Exploit isotopy
- Price for Better Way: Weaker Correctness Statement

  For some $B_0 \subseteq B_0^+ \subseteq B_0 \oplus B(\varepsilon)$,

  G is isotopic to $S \cap B_0^+$.
- APPLICATIONS:
  - Singularity (below)
  - Input region B_0 to have "any" geometry, even holes, provided it contains no singularities.

Boundary (Summary)

- An Obvious Way and a Better Way
  - Exact Way : Recursively solve the problem on ∂B_0
  - Better Way : Exploit isotopy
- Price for Better Way: Weaker Correctness Statement

  For some $B_0 \subseteq B_0^+ \subseteq B_0 \oplus B(\varepsilon)$,

  G is isotopic to $S \cap B_0^+$.
- APPLICATIONS:
  - Singularity (below)
  - Input region B_0 to have "any" geometry, even holes, provided it contains no singularities.

Boundary (Summary)

- An Obvious Way and a Better Way
  - Exact Way: Recursively solve the problem on ∂B_0
  - Better Way: Exploit isotopy
- Price for Better Way: Weaker Correctness Statement

  For some $B_0 \subseteq B_0^+ \subseteq B_0 \oplus B(\varepsilon)$,

  G is isotopic to $S \cap B_0^+$.
- APPLICATIONS:
  - Singularity (below)
  - Input region B_0 to have "any" geometry, even holes, provided it contains no singularities.

Boundary (Summary)

- An Obvious Way and a Better Way
  - Exact Way : Recursively solve the problem on ∂B_0
  - Better Way : Exploit isotopy
- Price for Better Way: Weaker Correctness Statement

  For some $B_0 \subseteq B_0^+ \subseteq B_0 \oplus B(\varepsilon)$,

  G is isotopic to $S \cap B_0^+$.
- APPLICATIONS:
  - Singularity (below)
  - Input region B_0 to have "any" geometry, even holes, provided it contains no singularities.

Boundary (Summary)

- An Obvious Way and a Better Way
  - Exact Way : Recursively solve the problem on ∂B_0
  - Better Way : Exploit isotopy
- Price for Better Way: Weaker Correctness Statement

  For some $B_0 \subseteq B_0^+ \subseteq B_0 \oplus B(\varepsilon)$,

  G is isotopic to $S \cap B_0^+$.
- APPLICATIONS:
  - Singularity (below)
  - Input region B_0 to have "any" geometry, even holes, provided it contains no singularities.

Boundary (Summary)

- An Obvious Way and a Better Way
  - Exact Way : Recursively solve the problem on ∂B_0
  - Better Way : Exploit isotopy
- Price for Better Way: Weaker Correctness Statement

  For some $B_0 \subseteq B_0^+ \subseteq B_0 \oplus B(\varepsilon)$,

  G is isotopic to $S \cap B_0^+$.
- APPLICATIONS:
  - Singularity (below)
  - Input region B_0 to have “any” geometry, even holes, provided it contains no singularities.

Coming Up Next

Singularity : Algebraic Preliminary

- Square-free part of $f(X_1, \ldots, X_n) \in \mathbb{Z}[X_1, \ldots, X_n]$:
  $$\frac{f}{\mathtt{GCD}(f, \partial_1 f, \ldots, \partial_n f)} = \frac{f}{\mathtt{GCD}(f, \nabla(f))}$$
- For $n = 1$: square-free implies no singularities
- Generally:
  Singular set $sing(f) := \mathtt{Zero}(f, \nabla(f))$ has co-dimension ≥ 2.
- For Curves:
  we now assume $f(X, Y) \in \mathbb{Z}[X, Y]$ has isolated singularities.

Singularity : Algebraic Preliminary

- Square-free part of $f(X_1, \ldots, X_n) \in \mathbb{Z}[X_1, \ldots, X_n]$:

$$\frac{f}{\mathtt{GCD}(f, \partial_1 f, \ldots, \partial_n f)} = \frac{f}{\mathtt{GCD}(f, \nabla(f))}$$

- For $n = 1$: square-free implies no singularities
- Generally:

  Singular set $sing(f) := \mathtt{Zero}(f, \nabla(f))$ has co-dimension ≥ 2.
- For Curves:

  we now assume $f(X, Y) \in \mathbb{Z}[X, Y]$ has isolated singularities.

Singularity : Algebraic Preliminary

- Square-free part of $f(X_1,\ldots,X_n) \in \mathbb{Z}[X_1,\ldots,X_n]$:
  $$\frac{f}{\mathtt{GCD}(f,\partial_1 f,\ldots,\partial_n f)} = \frac{f}{\mathtt{GCD}(f,\nabla(f))}$$
- For $n=1$: square-free implies no singularities
- Generally:
  Singular set $sing(f) := \mathtt{Zero}(f,\nabla(f))$ has co-dimension ≥ 2.
- For Curves:
  we now assume $f(X,Y) \in \mathbb{Z}[X,Y]$ has isolated singularities.

Singularity : Algebraic Preliminary

- Square-free part of $f(X_1,\ldots,X_n) \in \mathbb{Z}[X_1,\ldots,X_n]$:
  $$\frac{f}{\mathtt{GCD}(f,\partial_1 f,\ldots,\partial_n f)} = \frac{f}{\mathtt{GCD}(f,\nabla(f))}$$
- For $n=1$: square-free implies no singularities
- Generally:

  Singular set $sing(f) := \mathtt{Zero}(f,\nabla(f))$ has co-dimension ≥ 2.
- For Curves:

  we now assume $f(X,Y) \in \mathbb{Z}[X,Y]$ has isolated singularities.

Singularity : Algebraic Preliminary

- Square-free part of $f(X_1,\ldots,X_n) \in \mathbb{Z}[X_1,\ldots,X_n]$:
$$\frac{f}{\mathtt{GCD}(f,\partial_1 f,\ldots,\partial_n f)} = \frac{f}{\mathtt{GCD}(f,\nabla(f))}$$
- For $n = 1$: square-free implies no singularities
- Generally:
  Singular set $sing(f) := \mathtt{Zero}(f,\nabla(f))$ has co-dimension ≥ 2.
- For Curves:
  we now assume $f(X,Y) \in \mathbb{Z}[X,Y]$ has isolated singularities.

Singularity : Algebraic Preliminary

- Square-free part of $f(X_1, \ldots, X_n) \in \mathbb{Z}[X_1, \ldots, X_n]$:
$$\frac{f}{\mathtt{GCD}(f, \partial_1 f, \ldots, \partial_n f)} = \frac{f}{\mathtt{GCD}(f, \nabla(f))}$$
- For $n = 1$: square-free implies no singularities
- Generally:
  Singular set $sing(f) := \mathtt{Zero}(f, \nabla(f))$ has co-dimension ≥ 2.
- For Curves:
  we now assume $f(X, Y) \in \mathbb{Z}[X, Y]$ has isolated singularities.

Some Zero Bounds

Evaluation Bound Lemma

If $f(X,Y)$ has degree d and height L then

$$\boxed{-\log EV(f) = O(d^2(L + d\log d))}$$

where $EV(f) := \min\{|f(\alpha)| : \nabla(\alpha) = 0, f(\alpha) \neq 0\}$

Singularity Separation Bound [Y. (2006)]

Any two singularities of $f = 0$ are separated by

$$\boxed{\delta_3 \geq (16^{d+2} 256^L 81^{2d} d^5)^{-d}}$$

Closest Approach Bound

The "locally closest" approach of a curve $f = 0$ to its own singularities is

$$\boxed{\delta_4 \geq (6^2 e^7)^{-30D} (4^4 \cdot 5 \cdot 2^L)^{-5D^4}}$$

where $D = \max\{2, \deg f\}$

Some Zero Bounds

Evaluation Bound Lemma

If $f(X,Y)$ has degree d and height L then

$$\boxed{-\log EV(f) = O(d^2(L + d\log d))}$$

where $EV(f) := \min\{|f(\alpha)| : \nabla(\alpha) = 0, f(\alpha) \neq 0\}$

Singularity Separation Bound [Y. (2006)]

Any two singularities of $f = 0$ are separated by

$$\boxed{\delta_3 \geq (16^{d+2}256^L 81^{2d} d^5)^{-d}}$$

Closest Approach Bound

The "locally closest" approach of a curve $f = 0$ to its own singularities is

$$\boxed{\delta_4 \geq (6^2 e^7)^{-30D}(4^4 \cdot 5 \cdot 2^L)^{-5D^4}}$$

where $D = \max\{2, \deg f\}$

Some Zero Bounds

Evaluation Bound Lemma

If $f(X,Y)$ has degree d and height L then

$$\boxed{-\log EV(f) = O(d^2(L + d\log d))}$$

where $EV(f) := \min\{|f(\alpha)| : \nabla(\alpha) = 0, f(\alpha) \neq 0\}$

Singularity Separation Bound [Y. (2006)]

Any two singularities of $f = 0$ are separated by

$$\boxed{\delta_3 \geq (16^{d+2} 256^L 81^{2d} d^5)^{-d}}$$

Closest Approach Bound

The "locally closest" approach of a curve $f = 0$ to its own singularities is

$$\boxed{\delta_4 \geq (6^2 e^7)^{-30D} (4^4 \cdot 5 \cdot 2^L)^{-5D^4}}$$

where $D = \max\{2, \deg f\}$

Isolating Singularities

Mountain Pass Theorem

Consider $F := f^2 + f_X^2 + f_Y^2$.

Any 2 singularities in B_0 are connected by paths $\gamma : [0,1] \to \mathbb{R}^2$ satisfying

$$\min \gamma(F([0,1])) \geq \varepsilon_0$$

where

$$\varepsilon_0 := \min \{EV(f), \min F(\partial B_0)\}$$

Can provide a subdivision algorithm using F, ε_0 to isolate regions containing singularities.

Isolating Singularities

Mountain Pass Theorem

Consider $F := f^2 + f_X^2 + f_Y^2$.
Any 2 singularities in B_0 are connected by paths $\gamma : [0,1] \to \mathbb{R}^2$ satisfying

$$\min \gamma(F([0,1])) \geq \varepsilon_0$$

where

$$\varepsilon_0 := \min\{EV(f), \min F(\partial B_0)\}$$

Can provide a subdivision algorithm using F, ε_0 to isolate regions containing singularities.

Isolating Singularities

Mountain Pass Theorem

Consider $F := f^2 + f_X^2 + f_Y^2$.

Any 2 singularities in B_0 are connected by paths $\gamma : [0,1] \to \mathbb{R}^2$

satisfying

$$\min \gamma(F([0,1])) \geq \varepsilon_0$$

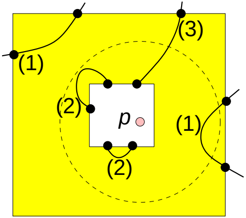
where

$$\varepsilon_0 := \min\{EV(f), \min F(\partial B_0)\}$$

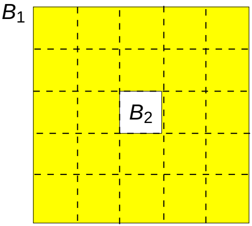
Can provide a subdivision algorithm using F, ε_0 to isolate regions containing singularities.

Degree of Singularites

- Degree of singularity $:=$ number of half-branches
- Use two concentric boxes $B_2 \subseteq B_1$:
  inner box has singularity, outer radius less than δ_3, δ_4



(a)



(b)

(a) Singularity p with 3 types of components

(b) Concentric boxes (B_1, B_2)

Mini Summary

- We have seen how to combine Snyder and PV, and make several practical improvements
- Future Work: Extend *3D* (and beyond?)
- Improve efficiency of refinement
- Improve efficiency of singularity
  - Nonsingular case is fast
  - Singular case is (currently) not fast

Mini Summary

- We have seen how to combine Snyder and PV, and make several practical improvements
- Future Work: Extend $3D$ (and beyond?)
- Improve efficiency of refinement
- Improve efficiency of singularity
  - Nonsingular case is fast
  - Singular case is (currently) not fast

Mini Summary

- We have seen how to combine Snyder and PV, and make several practical improvements
- Future Work: Extend $3D$ (and beyond?)
- Improve efficiency of refinement
- Improve efficiency of singularity
  - Nonsingular case is fast
  - Singular case is (currently) not fast

Mini Summary

- We have seen how to combine Snyder and PV, and make several practical improvements
- Future Work: Extend *3D* (and beyond?)
- Improve efficiency of refinement
- Improve efficiency of singularity
  - Nonsingular case is fast
  - Singular case is (currently) not fast

Mini Summary

- We have seen how to combine Snyder and PV, and make several practical improvements
- Future Work: Extend *3D* (and beyond?)
- Improve efficiency of refinement
- Improve efficiency of singularity
  - Nonsingular case is fast
  - Singular case is (currently) not fast

Mini Summary

- We have seen how to combine Snyder and PV, and make several practical improvements
- Future Work: Extend *3D* (and beyond?)
- Improve efficiency of refinement
- Improve efficiency of singularity
  - Nonsingular case is fast
  - Singular case is (currently) not fast

Mini Summary

- We have seen how to combine Snyder and PV, and make several practical improvements
- Future Work: Extend *3D* (and beyond?)
- Improve efficiency of refinement
- Improve efficiency of singularity
  - Nonsingular case is fast
  - Singular case is (currently) not fast

Mini Summary

- We have seen how to combine Snyder and PV, and make several practical improvements
- Future Work: Extend *3D* (and beyond?)
- Improve efficiency of refinement
- Improve efficiency of singularity
  - Nonsingular case is fast
  - Singular case is (currently) not fast

Mini Summary

- We have seen how to combine Snyder and PV, and make several practical improvements
- Future Work: Extend *3D* (and beyond?)
- Improve efficiency of refinement
- Improve efficiency of singularity
  - Nonsingular case is fast
  - Singular case is (currently) not fast

Summary of Lecture 2

- Problems at the interface of continuous and discrete: Explicitization Problems
- ENC Algorithms for them are novel
- Numerical Treatment of Singularity and Degeneracy
  - Possible in theory, but severe practical challenge

Summary of Lecture 2

- Problems at the interface of continuous and discrete: Explicitization Problems
- ENC Algorithms for them are novel
- Numerical Treatment of Singularity and Degeneracy
    - Possible in theory, but severe practical challenge

Summary of Lecture 2

- Problems at the interface of continuous and discrete: Explicitization Problems
- ENC Algorithms for them are novel
- Numerical Treatment of Singularity and Degeneracy
    - Possible in theory, but severe practical challenge

Summary of Lecture 2

- Problems at the interface of continuous and discrete: Explicitization Problems
- ENC Algorithms for them are novel
- Numerical Treatment of Singularity and Degeneracy
  - Possible in theory, but severe practical challenge

Summary of Lecture 2

- Problems at the interface of continuous and discrete: Explicitization Problems
- ENC Algorithms for them are novel
- Numerical Treatment of Singularity and Degeneracy
  - Possible in theory, but severe practical challenge

Summary of Lecture 2

- Problems at the interface of continuous and discrete: Explicitization Problems
- ENC Algorithms for them are novel
- Numerical Treatment of Singularity and Degeneracy
  - Possible in theory, but severe practical challenge

PART 3

Complexity Analysis of Adaptivity

"A rapacious monster lurks within every computer,
and it dines exclusively on accurate digits."

— B.D. McCullough (2000)

Coming Up Next

Towards Analysis of Adaptive Algorithms

- Major Challenge in Theoretical Computer Science
  - Analysis of discrete algorithms is highly developed
  - What about continuous, adaptive algorithms?
- Previous such analysis requires probabilistic assumptions.
  - Basically in Linear Programming: [Smale, Borgwardt, Teng-Spielman]
- We focus on the recursion tree size
  - Return to 1-D !
- Adaptive algorithms may have some deep paths, but overall size is only polynomial in depth.
  - Previous (trivial) result – size is exponential in depth [Kearfott (1987)]

Towards Analysis of Adaptive Algorithms

- Major Challenge in Theoretical Computer Science
  - Analysis of discrete algorithms is highly developed
  - What about continuous, adaptive algorithms?
- Previous such analysis requires probabilistic assumptions.
  - Basically in Linear Programming: [Smale, Borgwardt, Teng-Spielman]
- We focus on the recursion tree size
  - Return to 1-D !
- Adaptive algorithms may have some deep paths, but overall size is only polynomial in depth.
  - Previous (trivial) result – size is exponential in depth [Kearfott (1987)]

Towards Analysis of Adaptive Algorithms

- Major Challenge in Theoretical Computer Science
  - Analysis of discrete algorithms is highly developed
  - What about continuous, adaptive algorithms?
- Previous such analysis requires probabilistic assumptions.
  - Basically in Linear Programming: [Smale, Borgwardt, Teng-Spielman]
- We focus on the recursion tree size
  - Return to 1-D !
- Adaptive algorithms may have some deep paths, but overall size is only polynomial in depth.
  - Previous (trivial) result – size is exponential in depth [Kearfott (1987)]

Towards Analysis of Adaptive Algorithms

- Major Challenge in Theoretical Computer Science
  - Analysis of discrete algorithms is highly developed
  - What about continuous, adaptive algorithms?
- Previous such analysis requires probabilistic assumptions.
  - Basically in Linear Programming: [Smale, Borgwardt, Teng-Spielman]
- We focus on the recursion tree size
  - Return to 1-D !
- Adaptive algorithms may have some deep paths, but overall size is only polynomial in depth.
  - Previous (trivial) result – size is exponential in depth [Kearfott (1987)]

Towards Analysis of Adaptive Algorithms

- Major Challenge in Theoretical Computer Science
  - Analysis of discrete algorithms is highly developed
  - What about continuous, adaptive algorithms?
- Previous such analysis requires probabilistic assumptions.
  - Basically in Linear Programming: [Smale, Borgwardt, Teng-Spielman]
- We focus on the recursion tree size
  - Return to 1-D !
- Adaptive algorithms may have some deep paths, but overall size is only polynomial in depth.
  - Previous (trivial) result – size is exponential in depth [Kearfott (1987)]

Towards Analysis of Adaptive Algorithms

- Major Challenge in Theoretical Computer Science
  - Analysis of discrete algorithms is highly developed
  - What about continuous, adaptive algorithms?
- Previous such analysis requires probabilistic assumptions.
  - Basically in Linear Programming: [Smale, Borgwardt, Teng-Spielman]
- We focus on the recursion tree size
  - Return to 1-D !
- Adaptive algorithms may have some deep paths, but overall size is only polynomial in depth.
  - Previous (trivial) result – size is exponential in depth [Kearfott (1987)]

Towards Analysis of Adaptive Algorithms

- Major Challenge in Theoretical Computer Science
  - Analysis of discrete algorithms is highly developed
  - What about continuous, adaptive algorithms?
- Previous such analysis requires probabilistic assumptions.
  - Basically in Linear Programming: [Smale, Borgwardt, Teng-Spielman]
- We focus on the recursion tree size
  - Return to 1-D !
- Adaptive algorithms may have some deep paths, but overall size is only polynomial in depth.
  - Previous (trivial) result – size is exponential in depth [Kearfott (1987)]

Towards Analysis of Adaptive Algorithms

- Major Challenge in Theoretical Computer Science
  - Analysis of discrete algorithms is highly developed
  - What about continuous, adaptive algorithms?
- Previous such analysis requires probabilistic assumptions.
  - Basically in Linear Programming: [Smale, Borgwardt, Teng-Spielman]
- We focus on the recursion tree size
  - Return to 1-D !
- Adaptive algorithms may have some deep paths, but overall size is only polynomial in depth.
  - Previous (trivial) result – size is exponential in depth [Kearfott (1987)]

Towards Analysis of Adaptive Algorithms

- Major Challenge in Theoretical Computer Science
  - Analysis of discrete algorithms is highly developed
  - What about continuous, adaptive algorithms?
- Previous such analysis requires probabilistic assumptions.
  - Basically in Linear Programming: [Smale, Borgwardt, Teng-Spielman]
- We focus on the recursion tree size
  - Return to 1-D !
- Adaptive algorithms may have some deep paths, but overall size is only polynomial in depth.
  - Previous (trivial) result – size is exponential in depth [Kearfott (1987)]

Towards Analysis of Adaptive Algorithms

- Major Challenge in Theoretical Computer Science
  - Analysis of discrete algorithms is highly developed
  - What about continuous, adaptive algorithms?
- Previous such analysis requires probabilistic assumptions.
  - Basically in Linear Programming: [Smale, Borgwardt, Teng-Spielman]
- We focus on the recursion tree size
  - Return to 1-D !
- Adaptive algorithms may have some deep paths, but overall size is only polynomial in depth.
  - Previous (trivial) result – size is exponential in depth [Kearfott (1987)]

Towards Analysis of Adaptive Algorithms

- Major Challenge in Theoretical Computer Science
  - Analysis of discrete algorithms is highly developed
  - What about continuous, adaptive algorithms?
- Previous such analysis requires probabilistic assumptions.
  - Basically in Linear Programming: [Smale, Borgwardt, Teng-Spielman]
- We focus on the recursion tree size
  - Return to 1-D !
- Adaptive algorithms may have some deep paths, but overall size is only polynomial in depth.
  - Previous (trivial) result – size is exponential in depth [Kearfott (1987)]

Analytic Approach to Root Isolation

- Suppose you want to isolate real roots of $f(x)$ in $I = [a, b]$
- Midpoint $m(I) := (a+b)/2$, Width $w(I) := b - a$
- Exclusion Predicate: $C_0(I) : |f(m)| > \sum_{i \geq 1} \frac{|f^{(i)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Inclusion Predicate: $C_1(I) : |f'(m)| > \sum_{i \geq 1} \frac{|f^{(i+1)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Confirmation (Bolzano) Test: $f(a)f(b) < 0$
- Simple analytic method for root isolation!
- Simpler than algebraic subdivision methods:

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

- Suppose you want to isolate real roots of $f(x)$ in $I = [a, b]$
- Midpoint $m(I) := (a+b)/2$, Width $w(I) := b - a$
- Exclusion Predicate: $C_0(I) : |f(m)| > \sum_{i \geq 1} \frac{|f^{(i)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Inclusion Predicate: $C_1(I) : |f'(m)| > \sum_{i \geq 1} \frac{|f^{(i+1)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Confirmation (Bolzano) Test: $f(a)f(b) < 0$
- Simple analytic method for root isolation!
- Simpler than algebraic subdivision methods:

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

- Suppose you want to isolate real roots of $f(x)$ in $I = [a,b]$
- Midpoint $m(I) := (a+b)/2$,   Width $w(I) := b - a$
- Exclusion Predicate: $C_0(I) : |f(m)| > \sum_{i\geq 1} \frac{|f^{(i)}(m)|}{i!}\left(\frac{w(I)}{2}\right)^i$
- Inclusion Predicate: $C_1(I) : |f'(m)| > \sum_{i\geq 1} \frac{|f^{(i+1)}(m)|}{i!}\left(\frac{w(I)}{2}\right)^i$
- Confirmation (Bolzano) Test: $f(a)f(b) < 0$
- Simple analytic method for root isolation!
- Simpler than algebraic subdivision methods:

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

- Suppose you want to isolate real roots of $f(x)$ in $I = [a, b]$
- Midpoint $m(I) := (a+b)/2$, Width $w(I) := b - a$
- Exclusion Predicate: $C_0(I) : |f(m)| > \sum_{i\geq 1} \frac{|f^{(i)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Inclusion Predicate: $C_1(I) : |f'(m)| > \sum_{i\geq 1} \frac{|f^{(i+1)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Confirmation (Bolzano) Test: $f(a)f(b) < 0$
- Simple analytic method for root isolation!
- Simpler than algebraic subdivision methods:

$$\mathit{STURM} > \mathit{DESCARTES} > \mathit{BOLZANO}$$

Analytic Approach to Root Isolation

- Suppose you want to isolate real roots of $f(x)$ in $I = [a,b]$
- Midpoint $m(I) := (a+b)/2$, Width $w(I) := b - a$
- Exclusion Predicate: $C_0(I) : |f(m)| > \sum_{i\geq 1} \frac{|f^{(i)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Inclusion Predicate: $C_1(I) : |f'(m)| > \sum_{i\geq 1} \frac{|f^{(i+1)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Confirmation (Bolzano) Test: $f(a)f(b) < 0$
- Simple analytic method for root isolation!
- Simpler than algebraic subdivision methods:

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

- Suppose you want to isolate real roots of $f(x)$ in $I = [a, b]$
- Midpoint $m(I) := (a+b)/2$, Width $w(I) := b - a$
- Exclusion Predicate: $C_0(I) : |f(m)| > \sum_{i\geq 1} \frac{|f^{(i)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Inclusion Predicate: $C_1(I) : |f'(m)| > \sum_{i\geq 1} \frac{|f^{(i+1)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Confirmation (Bolzano) Test: $f(a)f(b) < 0$
- Simple analytic method for root isolation!
- Simpler than algebraic subdivision methods:

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

- Suppose you want to isolate real roots of $f(x)$ in $I = [a, b]$
- Midpoint $m(I) := (a+b)/2$, Width $w(I) := b - a$
- Exclusion Predicate: $C_0(I) : |f(m)| > \sum_{i\geq 1} \frac{|f^{(i)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Inclusion Predicate: $C_1(I) : |f'(m)| > \sum_{i\geq 1} \frac{|f^{(i+1)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Confirmation (Bolzano) Test: $f(a)f(b) < 0$
- Simple analytic method for root isolation!
- Simpler than algebraic subdivision methods:

$$\mathit{STURM} > \mathit{DESCARTES} > \mathit{BOLZANO}$$

Analytic Approach to Root Isolation

- Suppose you want to isolate real roots of $f(x)$ in $I = [a, b]$
- Midpoint $m(I) := (a+b)/2$,   Width $w(I) := b - a$
- Exclusion Predicate: $C_0(I) : |f(m)| > \sum_{i \geq 1} \frac{|f^{(i)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Inclusion Predicate: $C_1(I) : |f'(m)| > \sum_{i \geq 1} \frac{|f^{(i+1)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Confirmation (Bolzano) Test: $f(a)f(b) < 0$
- Simple analytic method for root isolation!
- Simpler than algebraic subdivision methods:

$$STURM > DESCARTES > BOLZANO$$

Analytic Approach to Root Isolation

- Suppose you want to isolate real roots of $f(x)$ in $I = [a, b]$
- Midpoint $m(I) := (a+b)/2$, Width $w(I) := b - a$
- Exclusion Predicate: $C_0(I) : |f(m)| > \sum_{i \geq 1} \frac{|f^{(i)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Inclusion Predicate: $C_1(I) : |f'(m)| > \sum_{i \geq 1} \frac{|f^{(i+1)}(m)|}{i!} \left(\frac{w(I)}{2}\right)^i$
- Confirmation (Bolzano) Test: $f(a)f(b) < 0$
- Simple analytic method for root isolation!
- Simpler than algebraic subdivision methods:

$$STURM > DESCARTES > BOLZANO$$

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE $(Q \neq \emptyset)$ ◁ *Subdivision Phase*
3. $\quad I \leftarrow Q.remove()$
4. $\quad$ IF $(C_0(I)$ holds), discard I
5. $\quad$ ELIF $(C_1(I)$ holds), output I
6. $\quad$ ELSE
7. $\quad\quad$ IF $(f(m(I)) = 0)$, output $[m(I), m(I)]$
8. $\quad\quad$ Split I into two and insert in Q
9. PROCESS output list ◁ *Consruction Phase*

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE ($Q \neq \emptyset$) ◁ *Subdivision Phase*
3. $\quad I \leftarrow Q.remove()$
4. $\quad$ IF ($C_0(I)$ holds), discard I
5. $\quad$ ELIF ($C_1(I)$ holds), output I
6. $\quad$ ELSE
7. $\quad\quad$ IF ($f(m(I)) = 0$), output $[m(I), m(I)]$
8. $\quad\quad$ Split I into two and insert in Q
9. PROCESS output list ◁ *Consruction Phase*

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE ($Q \neq \emptyset$) ◁ *Subdivision Phase*
3. $I \leftarrow Q.remove()$
4. IF ($C_0(I)$ holds), discard I
5. ELIF ($C_1(I)$ holds), output I
6. ELSE
7. IF ($f(m(I)) = 0$), output $[m(I), m(I)]$
8. Split I into two and insert in Q
9. PROCESS output list ◁ *Consruction Phase*

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE $(Q \neq \emptyset)$ $\triangleleft$ *Subdivision Phase*
3. $\quad I \leftarrow Q.remove()$
4. $\quad$ IF $(C_0(I)$ holds), discard I
5. $\quad$ ELIF $(C_1(I)$ holds), output I
6. $\quad$ ELSE
7. $\quad\quad$ IF $(f(m(I)) = 0)$, output $[m(I), m(I)]$
8. $\quad\quad$ Split I into two and insert in Q
9. PROCESS output list $\triangleleft$ *Consruction Phase*

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE ($Q \neq \emptyset$) ◁ *Subdivision Phase*
3. $I \leftarrow Q.remove()$
4. IF ($C_0(I)$ holds), discard I
5. ELIF ($C_1(I)$ holds), output I
6. ELSE
7. IF ($f(m(I)) = 0$), output $[m(I), m(I)]$
8. Split I into two and insert in Q
9. PROCESS output list ◁ *Consruction Phase*

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE ($Q \neq \emptyset$) ◁ *Subdivision Phase*
3. $\quad I \leftarrow Q.remove()$
4. $\quad$ IF ($C_0(I)$ holds), discard I
5. $\quad$ ELIF ($C_1(I)$ holds), output I
6. $\quad$ ELSE
7. $\quad\quad$ IF ($f(m(I)) = 0$), output $[m(I), m(I)]$
8. $\quad\quad$ Split I into two and insert in Q
9. PROCESS output list ◁ *Consruction Phase*

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE ($Q \neq \emptyset$) ◁ *Subdivision Phase*
3. $I \leftarrow Q.remove()$
4. IF ($C_0(I)$ holds), discard I
5. ELIF ($C_1(I)$ holds), output I
6. ELSE
7. IF ($f(m(I)) = 0$), output $[m(I), m(I)]$
8. Split I into two and insert in Q
9. PROCESS output list ◁ *Consruction Phase*

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE ($Q \neq \emptyset$) ◁ *Subdivision Phase*
3. $I \leftarrow Q.remove()$
4. IF ($C_0(I)$ holds), discard I
5. ELIF ($C_1(I)$ holds), output I
6. ELSE
7. IF ($f(m(I)) = 0$), output $[m(I), m(I)]$
8. Split I into two and insert in Q
9. PROCESS output list ◁ *Consruction Phase*

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE ($Q \neq \emptyset$) ◁ *Subdivision Phase*
3. $\quad I \leftarrow Q.remove()$
4. $\quad$ IF ($C_0(I)$ holds), discard I
5. $\quad$ ELIF ($C_1(I)$ holds), output I
6. $\quad$ ELSE
7. $\quad\quad$ IF ($f(m(I)) = 0$), output $[m(I), m(I)]$
8. $\quad\quad$ Split I into two and insert in Q
9. PROCESS output list ◁ *Consruction Phase*

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE ($Q \neq \emptyset$) $\triangleleft$ *Subdivision Phase*
3. $\quad$ $I \leftarrow Q.remove()$
4. $\quad$ IF ($C_0(I)$ holds), discard I
5. $\quad$ ELIF ($C_1(I)$ holds), output I
6. $\quad$ ELSE
7. $\quad\quad$ IF ($f(m(I)) = 0$), output $[m(I), m(I)]$
8. $\quad\quad$ Split I into two and insert in Q
9. PROCESS output list $\triangleleft$ *Consruction Phase*

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE ($Q \neq \emptyset$) ◁ *Subdivision Phase*
3. $\quad I \leftarrow Q.remove()$
4. $\quad$ IF ($C_0(I)$ holds), discard I
5. $\quad$ ELIF ($C_1(I)$ holds), output I
6. $\quad$ ELSE
7. $\quad\quad$ IF ($f(m(I)) = 0$), output $[m(I), m(I)]$
8. $\quad\quad$ Split I into two and insert in Q
9. PROCESS output list ◁ *Consruction Phase*

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE ($Q \neq \emptyset$) $\triangleleft$ *Subdivision Phase*
3. $I \leftarrow Q.remove()$
4. IF ($C_0(I)$ holds), discard I
5. ELIF ($C_1(I)$ holds), output I
6. ELSE
7. IF ($f(m(I)) = 0$), output $[m(I), m(I)]$
8. Split I into two and insert in Q
9. PROCESS output list $\triangleleft$ *Consruction Phase*

EVAL Algorithm

EVAL

- INPUT: Function f and interval $I_0 = [a, b]$
- OUTPUT: Isolation intervals of roots of f in I_0

1. Let $Q_{in} \leftarrow \{I_0\}$ be a queue
2. WHILE ($Q \neq \emptyset$) ◁ *Subdivision Phase*
3. $I \leftarrow Q.remove()$
4. IF ($C_0(I)$ holds), discard I
5. ELIF ($C_1(I)$ holds), output I
6. ELSE
7. IF ($f(m(I)) = 0$), output $[m(I), m(I)]$
8. Split I into two and insert in Q
9. PROCESS output list ◁ *Consruction Phase*

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

- **Problem:** isolate ALL (real) roots of square-free $f(X) \in \mathbb{Z}[X]$ of degree $\leq d$ and height $< 2^L$.
- Highly classical problem:
  - Bit complexity is $\widetilde{O}(d^3 L)$ [Schöhage 1982].
    - Improvement: [illegible] arithmetic complexity [Pan]
  - Sturm tree size is $O(d(L + \log d))$ [Davenport, 1985]
  - Descartes tree size is $\Theta(d(L + \log d))$ [Eigenwillig-Sharma-Y, 2006]
- **MAIN RESULT:** Bolzano tree size is $O(d^2(L + \log d))$
  - Sketch in this lecture. See [Burr-Krahmer-Y-Sagraloff, 2008-9]

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

- Problem: isolate ALL (real) roots of square-free $f(X) \in \mathbb{Z}[X]$ of degree $\leq d$ and height $< 2^L$.
- Highly classical problem:
  - Bit complexity is $\widetilde{O}(d^3 L)$ [Schöhage 1982].
    - Improvement: $\widetilde{O}(d^2 L)$ arithmetic complexity [Pan]
  - Sturm tree size is $O(d(L + \log d))$ [Davenport, 1985]
  - Descartes tree size is $\Theta(d(L + \log d))$ [Eigenwillig-Sharma-Y, 2006]
- MAIN RESULT: Bolzano tree size is $O(d^2(L + \log d))$
  - Sketch in this lecture. See [Burr-Krahmer-Y-Sagraloff, 2008-9]

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

- Problem: isolate ALL (real) roots of square-free $f(X) \in \mathbb{Z}[X]$ of degree $\leq d$ and height $< 2^L$.
- Highly classical problem:
  - Bit complexity is $\widetilde{O}(d^3 L)$ [Schöhage 1982].
    - Improvement: $\widetilde{O}(d^2 L)$ arithmetic complexity [Pan]
  - Sturm tree size is $O(d(L + \log d))$ [Davenport, 1985]
  - Descartes tree size is $\Theta(d(L + \log d))$ [Eigenwillig-Sharma-Y, 2006]
- MAIN RESULT: Bolzano tree size is $O(d^2(L + \log d))$
  - Sketch in this lecture. See [Burr-Krahmer-Y-Sagraloff, 2008-9]

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

- Problem: isolate ALL (real) roots of square-free $f(X) \in \mathbb{Z}[X]$ of degree $\leq d$ and height $< 2^L$.
- Highly classical problem:
  - Bit complexity is $\widetilde{O}(d^3 L)$ [Schöhage 1982].
    - Improvement: $\widetilde{O}(d^2 L)$ arithmetic complexity [Pan]
  - Sturm tree size is $O(d(L + \log d))$ [Davenport, 1985]
  - Descartes tree size is $\Theta(d(L + \log d))$ [Eigenwillig-Sharma-Y, 2006]
- MAIN RESULT: Bolzano tree size is $O(d^2(L + \log d))$
  - Sketch in this lecture. See [Burr-Krahmer-Y-Sagraloff, 2008-9]

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

- Problem: isolate ALL (real) roots of square-free $f(X) \in \mathbb{Z}[X]$ of degree $\leq d$ and height $< 2^L$.
- Highly classical problem:
  - Bit complexity is $\widetilde{O}(d^3 L)$ [Schöhage 1982].
    - Improvement: $\widetilde{O}(d^2 L)$ arithmetic complexity [Pan]
  - Sturm tree size is $O(d(L + \log d))$ [Davenport, 1985]
  - Descartes tree size is $\Theta(d(L + \log d))$ [Eigenwillig-Sharma-Y, 2006]
- MAIN RESULT: Bolzano tree size is $O(d^2(L + \log d))$
  - Sketch in this lecture. See [Burr-Krahmer-Y-Sagraloff, 2008-9]

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

- Problem: isolate ALL (real) roots of square-free $f(X) \in \mathbb{Z}[X]$ of degree $\leq d$ and height $< 2^L$.
- Highly classical problem:
  - Bit complexity is $\widetilde{O}(d^3 L)$ [Schöhage 1982].
    - Improvement: $\widetilde{O}(d^2 L)$ arithmetic complexity [Pan]
  - Sturm tree size is $O(d(L + \log d))$ [Davenport, 1985]
  - Descartes tree size is $\Theta(d(L + \log d))$ [Eigenwillig-Sharma-Y, 2006]
- MAIN RESULT: Bolzano tree size is $O(d^2(L + \log d))$
  - Sketch in this lecture. See [Burr-Krahmer-Y-Sagraloff, 2008-9]

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

- Problem: isolate ALL (real) roots of square-free $f(X) \in \mathbb{Z}[X]$ of degree $\leq d$ and height $< 2^L$.
- Highly classical problem:
  - Bit complexity is $\widetilde{O}(d^3L)$ [Schöhage 1982].
    - Improvement: $\widetilde{O}(d^2L)$ arithmetic complexity [Pan]
  - Sturm tree size is $O(d(L+\log d))$ [Davenport, 1985]
  - Descartes tree size is $\Theta(d(L+\log d))$ [Eigenwillig-Sharma-Y, 2006]
- MAIN RESULT: Bolzano tree size is $O(d^2(L+\log d))$
  - Sketch in this lecture. See [Burr-Krahmer-Y-Sagraloff, 2008-9]

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

- Problem: isolate ALL (real) roots of square-free $f(X) \in \mathbb{Z}[X]$ of degree $\leq d$ and height $< 2^L$.
- Highly classical problem:
  - Bit complexity is $\widetilde{O}(d^3 L)$ [Schöhage 1982].
    - Improvement: $\widetilde{O}(d^2 L)$ arithmetic complexity [Pan]
  - Sturm tree size is $O(d(L + \log d))$ [Davenport, 1985]
  - Descartes tree size is $\Theta(d(L + \log d))$ [Eigenwillig-Sharma-Y, 2006]
- MAIN RESULT: Bolzano tree size is $O(d^2(L + \log d))$
  - Sketch in this lecture. See [Burr-Krahmer-Y-Sagraloff, 2008-9]

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

- Problem: isolate ALL (real) roots of square-free $f(X) \in \mathbb{Z}[X]$ of degree $\leq d$ and height $< 2^L$.
- Highly classical problem:
  - Bit complexity is $\widetilde{O}(d^3 L)$ [Schöhage 1982].
    - Improvement: $\widetilde{O}(d^2 L)$ arithmetic complexity [Pan]
  - Sturm tree size is $O(d(L + \log d))$ [Davenport, 1985]
  - Descartes tree size is $\Theta(d(L + \log d))$ [Eigenwillig-Sharma-Y, 2006]
- MAIN RESULT: Bolzano tree size is $O(d^2(L + \log d))$
  - Sketch in this lecture. See [Burr-Krahmer-Y-Sagraloff, 2008-9]

Main Complexity Goal – Benchmark Problem

Benchmark Problem in Root Isolation

- Problem: isolate ALL (real) roots of square-free $f(X) \in \mathbb{Z}[X]$ of degree $\leq d$ and height $< 2^L$.
- Highly classical problem:
  - Bit complexity is $\widetilde{O}(d^3 L)$ [Schöhage 1982].
    - Improvement: $\widetilde{O}(d^2 L)$ arithmetic complexity [Pan]
  - Sturm tree size is $O(d(L + \log d))$ [Davenport, 1985]
  - Descartes tree size is $\Theta(d(L + \log d))$ [Eigenwillig-Sharma-Y, 2006]
- MAIN RESULT: Bolzano tree size is $O(d^2(L + \log d))$
  - Sketch in this lecture. See [Burr-Krahmer-Y-Sagraloff, 2008-9]

Warm Up Technique: Algebraic Amortization

Idea of Amortization [Davenport (1985), Du/Sharma/Y. (2005)]

- Let $A(X) \in \mathbb{Z}[X]$ have degree n and L-bit coefficients.
- Root separation bound: $-\log|\alpha - \beta| = O(n(L + \log n))$
- Amortized bound: $\boxed{-\prod_{(\alpha,\beta)\in E} |\beta - \alpha| = O(n(L + \log n))}$
- What are restrictions on set E?

Warm Up Technique: Algebraic Amortization

Idea of Amortization [Davenport (1985), Du/Sharma/Y. (2005)]

- Let $A(X) \in \mathbb{Z}[X]$ have degree n and L-bit coefficients.
- Root separation bound: $-\log|\alpha - \beta| = O(n(L + \log n))$
- Amortized bound: $\boxed{-\prod_{(\alpha,\beta)\in E} |\beta - \alpha| = O(n(L + \log n))}$
- What are restrictions on set E?

Warm Up Technique: Algebraic Amortization

Idea of Amortization [Davenport (1985), Du/Sharma/Y. (2005)]

- Let $A(X) \in \mathbb{Z}[X]$ have degree n and L-bit coefficients.
- Root separation bound: $-\log|\alpha - \beta| = O(n(L + \log n))$
- Amortized bound: $\boxed{-\prod_{(\alpha,\beta)\in E} |\beta - \alpha| = O(n(L + \log n))}$
- What are restrictions on set E?

Warm Up Technique: Algebraic Amortization

Idea of Amortization [Davenport (1985), Du/Sharma/Y. (2005)]

- Let $A(X) \in \mathbb{Z}[X]$ have degree n and L-bit coefficients.
- Root separation bound: $-\log|\alpha - \beta| = O(n(L + \log n))$
- Amortized bound: $\boxed{-\prod_{(\alpha,\beta)\in E} |\beta - \alpha| = O(n(L + \log n))}$
- What are restrictions on set E?

Warm Up Technique: Algebraic Amortization

Idea of Amortization [Davenport (1985), Du/Sharma/Y. (2005)]

- Let $A(X) \in \mathbb{Z}[X]$ have degree n and L-bit coefficients.
- Root separation bound: $-\log|\alpha - \beta| = O(n(L + \log n))$
- Amortized bound: $\boxed{-\prod_{(\alpha,\beta)\in E} |\beta - \alpha| = O(n(L + \log n))}$
- What are restrictions on set E?

Warm Up Technique: Algebraic Amortization

Idea of Amortization [Davenport (1985), Du/Sharma/Y. (2005)]

- Let $A(X) \in \mathbb{Z}[X]$ have degree n and L-bit coefficients.
- Root separation bound: $-\log|\alpha - \beta| = O(n(L + \log n))$
- Amortized bound: $\boxed{-\prod_{(\alpha,\beta)\in E} |\beta - \alpha| = O(n(L + \log n))}$
- What are restrictions on set E?

The Davenport–Mahler Bound

Theorem ([Davenport (1985), Johnson (1991/98), Du/Sharma/Y. (2005)])

Consider a polynomial $A(X) \in \mathbb{C}[X]$ *of degree n. Let* $G = (V, E)$ *be a digraph whose node set* V *consists of the roots* $\vartheta_1, \ldots, \vartheta_n$ *of* $A(X)$*. If*

(i) $(\alpha, \beta) \in E \implies |\alpha| \le |\beta|$,

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \le 1$, *and*

(iii) G *is acyclic,*

then

$$\prod_{(\alpha,\beta)\in E} |\beta - \alpha| \ge \frac{\sqrt{|\mathrm{discr}(A)|}}{\mathrm{M}(A)^{n-1}} \cdot 2^{-O(n \log n)},$$

where

$$\mathrm{discr}(A) := a_n^{2n-2} \prod_{i>j} (\vartheta_i - \vartheta_j)^2 \quad \textit{and} \quad \mathrm{M}(A) := |a_n| \prod_i \max\{1, |\vartheta_i|\}.$$

Mini Summary

- Adaptive analysis is important but virgin territory
- Subdivision of Analytic Algorithms in 1-D is current challenge
- Standard target is Benchmark Problem for root isolation
- Warm-Up Exercise: Use Mahler-Davenport bound for Descartes Method

Mini Summary

- Adaptive analysis is important but virgin territory
- Subdivision of Analytic Algorithms in 1-D is current challenge
- Standard target is Benchmark Problem for root isolation
- Warm-Up Exercise: Use Mahler-Davenport bound for Descartes Method

Mini Summary

- Adaptive analysis is important but virgin territory
- Subdivision of Analytic Algorithms in 1-D is current challenge
- Standard target is Benchmark Problem for root isolation
- Warm-Up Exercise: Use Mahler-Davenport bound for Descartes Method

Mini Summary

- Adaptive analysis is important but virgin territory
- Subdivision of Analytic Algorithms in 1-D is current challenge
- Standard target is Benchmark Problem for root isolation
- Warm-Up Exercise: Use Mahler-Davenport bound for Descartes Method

Mini Summary

- Adaptive analysis is important but virgin territory
- Subdivision of Analytic Algorithms in 1-D is current challenge
- Standard target is Benchmark Problem for root isolation
- Warm-Up Exercise: Use Mahler-Davenport bound for Descartes Method

Mini Summary

- Adaptive analysis is important but virgin territory
- Subdivision of Analytic Algorithms in 1-D is current challenge
- Standard target is Benchmark Problem for root isolation
- Warm-Up Exercise: Use Mahler-Davenport bound for Descartes Method

Coming Up Next

What is the Descartes Method?

Same framework as EVAL or Sturm

- To isolate roots of square-free $A(X)$ in interval I
- Routine $DescartesTest(A(X), I)$ gives an upper estimate on the number of real roots in I.
- If $DescartesTest(A(X), I) \in \{0, 1\}$ then estimate is exact.
- We keep splitting intervals until we get an exact estimate.

What is the Descartes Method?

Same framework as EVAL or Sturm

- To isolate roots of square-free $A(X)$ in interval I
- Routine $DescartesTest(A(X), I)$ gives an upper estimate on the number of real roots in I.
- If $DescartesTest(A(X), I) \in \{0, 1\}$ then estimate is exact.
- We keep splitting intervals until we get an exact estimate.

What is the Descartes Method?

Same framework as EVAL or Sturm

- To isolate roots of square-free $A(X)$ in interval I
- Routine $DescartesTest(A(X), I)$ gives an upper estimate on the number of real roots in I.
- If $DescartesTest(A(X), I) \in \{0, 1\}$ then estimate is exact.
- We keep splitting intervals until we get an exact estimate.

What is the Descartes Method?

Same framework as EVAL or Sturm

- To isolate roots of square-free $A(X)$ in interval I
- Routine $DescartesTest(A(X), I)$ gives an upper estimate on the number of real roots in I.
- If $DescartesTest(A(X), I) \in \{0, 1\}$ then estimate is exact.
- We keep splitting intervals until we get an exact estimate.

What is the Descartes Method?

Same framework as EVAL or Sturm

- To isolate roots of square-free $A(X)$ in interval I
- Routine $DescartesTest(A(X), I)$ gives an upper estimate on the number of real roots in I.
- If $DescartesTest(A(X), I) \in \{0, 1\}$ then estimate is exact.
- We keep splitting intervals until we get an exact estimate.

What is the Descartes Method?

Same framework as EVAL or Sturm

- To isolate roots of square-free $A(X)$ in interval I
- Routine $\mathit{DescartesTest}(A(X), I)$ gives an upper estimate on the number of real roots in I.
- If $\mathit{DescartesTest}(A(X), I) \in \{0, 1\}$ then estimate is exact.
- We keep splitting intervals until we get an exact estimate.

Analysis of Descartes Method

Two-circle Theorem

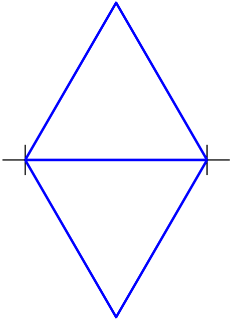
[Ostrowski (1950), Krandick/Mehlhorn (2006)]

If DescartesTest$(A(X),[c,d]) \geq 2$, then the two-circles figure in $\mathbb{C}$ around interval $[c,d]$ contains two roots α,β of $A(X)$.

Corollary

Can choose α,β to be complex conjugate or adjacent real roots.
Moreover, $|\beta-\alpha| < \sqrt{3}(d-c)$; i.e., $(d-c) > |\beta-\alpha|/\sqrt{3}$.

Analysis of Descartes Method



Two-circle Theorem

[Ostrowski (1950), Krandick/Mehlhorn (2006)]

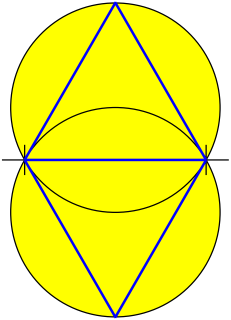
If DescartesTest$(A(X),[c,d]) \geq 2$, then the two-circles figure in $\mathbb{C}$ around interval $[c,d]$ contains two roots α, β of $A(X)$.

Corollary

Can choose α, β to be complex conjugate or adjacent real roots.

Moreover, $|\beta - \alpha| < \sqrt{3}(d-c)$; i.e., $(d-c) > |\beta - \alpha|/\sqrt{3}$.

Analysis of Descartes Method



Two-circle Theorem

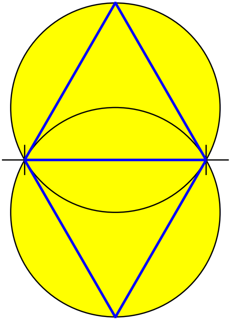
[Ostrowski (1950), Krandick/Mehlhorn (2006)]

If DescartesTest$(A(X),[c,d]) \geq 2$, then the two-circles figure in $\mathbb{C}$ around interval $[c,d]$ contains two roots α,β of $A(X)$.

Corollary

Can choose α,β to be complex conjugate or adjacent real roots.
Moreover, $|\beta-\alpha| < \sqrt{3}(d-c)$; i.e., $(d-c) > |\beta-\alpha|/\sqrt{3}$.

Analysis of Descartes Method



Two-circle Theorem

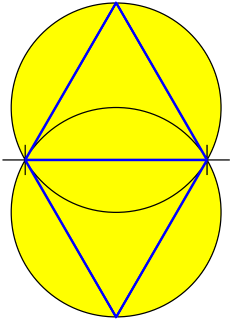
[Ostrowski (1950), Krandick/Mehlhorn (2006)]

If $DescartesTest(A(X),[c,d]) \geq 2$*, then the* *two-circles figure* *in* $\mathbb{C}$ *around interval* $[c,d]$ *contains two roots* α,β *of* $A(X)$*.*

Corollary

Can choose α,β *to be complex conjugate or adjacent real roots.*
Moreover, $|\beta-\alpha| < \sqrt{3}(d-c)$*; i.e.,* $(d-c) > |\beta-\alpha|/\sqrt{3}$*.*

Analysis of Descartes Method



Two-circle Theorem

[Ostrowski (1950), Krandick/Mehlhorn (2006)]

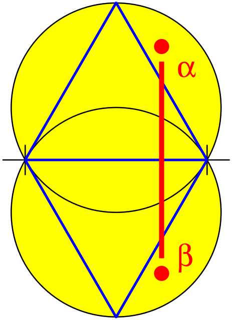
If DescartesTest$(A(X), [c,d]) \geq 2$, then the two-circles figure in $\mathbb{C}$ around interval $[c,d]$ contains two roots α, β of $A(X)$.

Corollary

Can choose α, β to be complex conjugate or adjacent real roots.

Moreover, $|\beta - \alpha| < \sqrt{3}(d-c)$; i.e., $(d-c) > |\beta - \alpha|/\sqrt{3}$.

Analysis of Descartes Method



Two-circle Theorem

[Ostrowski (1950), Krandick/Mehlhorn (2006)]

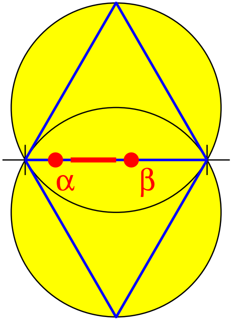
If DescartesTest$(A(X),[c,d]) \geq 2$, then the two-circles figure in $\mathbb{C}$ around interval $[c,d]$ contains two roots α, β of $A(X)$.

Corollary

Can choose α, β to be complex conjugate or adjacent real roots.

Moreover, $|\beta - \alpha| < \sqrt{3}(d-c)$; i.e., $(d-c) > |\beta - \alpha|/\sqrt{3}$.

Analysis of Descartes Method



Two-circle Theorem

[Ostrowski (1950), Krandick/Mehlhorn (2006)]

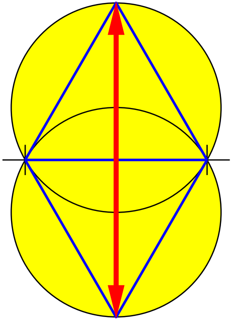
If DescartesTest$(A(X), [c, d]) \geq 2$*, then the two-circles figure in* $\mathbb{C}$ *around interval* $[c, d]$ *contains two roots* α, β *of* $A(X)$.

Corollary

Can choose α, β *to be complex conjugate or adjacent real roots.*

Moreover, $|\beta - \alpha| < \sqrt{3}(d - c)$*; i.e.,* $(d - c) > |\beta - \alpha| / \sqrt{3}$.

Analysis of Descartes Method



Two-circle Theorem

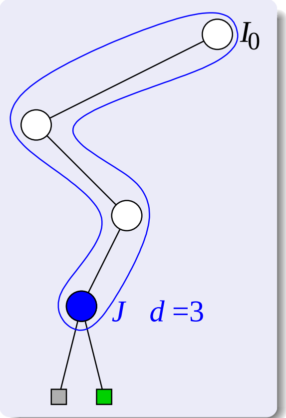
[Ostrowski (1950), Krandick/Mehlhorn (2006)]

If $DescartesTest(A(X), [c, d]) \geq 2$*, then the two-circles figure in* $\mathbb{C}$ *around interval* $[c, d]$ *contains two roots* α, β *of* $A(X)$*.*

Corollary

Can choose α, β *to be complex conjugate or adjacent real roots.*
Moreover, $|\beta - \alpha| < \sqrt{3}(d - c)$*; i.e.,* $(d - c) > |\beta - \alpha| / \sqrt{3}$*.*

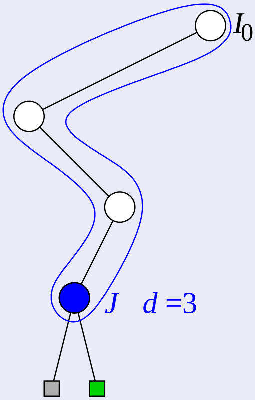
Tree Bound in terms of Roots (1)



A bound on path length

1. Consider any path in the recursion tree from I_0 to a parent J of two leaves.
2. At depth d, interval width is $2^{-d}|I_0|$. Hence depth of J is $d = \log|I_0|/|J|$.
3. The path consists of $d+1$ internal nodes.
4. There is a pair of roots (α_J, β_J) such that $|J| > |\beta_J - \alpha_J|/\sqrt{3}$; hence
$$\boxed{d + 1 < \log|I_0| - \log|\beta_J - \alpha_J| + 2.}$$

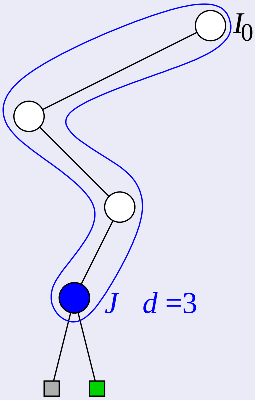
Tree Bound in terms of Roots (1)



A bound on path length

1. Consider any path in the recursion tree from I_0 to a parent J of two leaves.
2. At depth d, interval width is $2^{-d}|I_0|$. Hence depth of J is $d = \log|I_0|/|J|$.
3. The path consists of $d+1$ internal nodes.
4. There is a pair of roots (α_J, β_J) such that $|J| > |\beta_J - \alpha_J|/\sqrt{3}$; hence $\boxed{d+1 < \log|I_0| - \log|\beta_J - \alpha_J| + 2.}$

Tree Bound in terms of Roots (1)

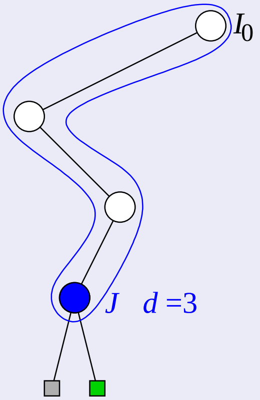


A bound on path length

1. Consider any path in the recursion tree from I_0 to a parent J of two leaves.
2. At depth d, interval width is $2^{-d}|I_0|$. Hence depth of J is $d = \log|I_0|/|J|$.
3. The path consists of $d+1$ internal nodes.
4. There is a pair of roots (α_J, β_J) such that $|J| > |\beta_J - \alpha_J|/\sqrt{3}$; hence

$$\boxed{d+1 < \log|I_0| - \log|\beta_J - \alpha_J| + 2.}$$

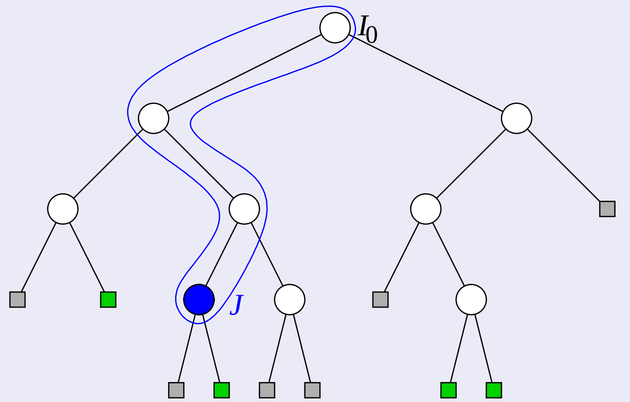
Tree Bound in terms of Roots (1)



A bound on path length

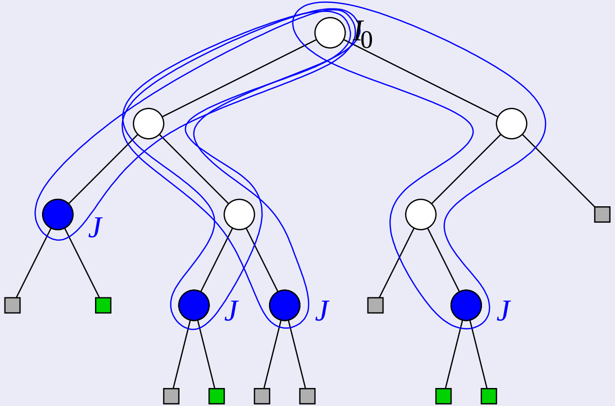
1. Consider any path in the recursion tree from I_0 to a parent J of two leaves.
2. At depth d, interval width is $2^{-d}|I_0|$. Hence depth of J is $d = \log|I_0|/|J|$.
3. The path consists of $d+1$ internal nodes.
4. There is a pair of roots (α_J, β_J) such that $|J| > |\beta_J - \alpha_J|/\sqrt{3}$; hence $\boxed{d+1 < \log|I_0| - \log|\beta_J - \alpha_J| + 2.}$

Tree Bound in terms of Roots (2)



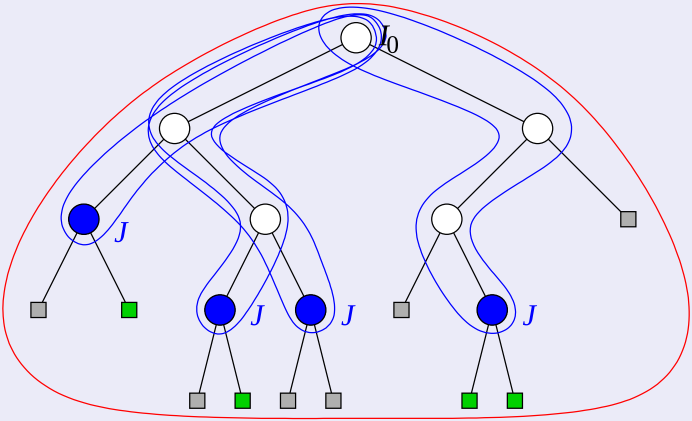
$$\begin{array}{lcr}
\#(\text{internal nodes on path}) & < & \log|I_0| - \log|\beta_J - \alpha_J| + 2 \\
\#(\text{internal nodes in tree}) & < & \sum_J (\log|I_0| - \log|\beta_J - \alpha_J| + 2) \\
\#(\text{all nodes in tree}) & < & 1 + 2 \cdot \sum_J (\log|I_0| - \log|\beta_J - \alpha_J| + 2)
\end{array}$$

Tree Bound in terms of Roots (2)



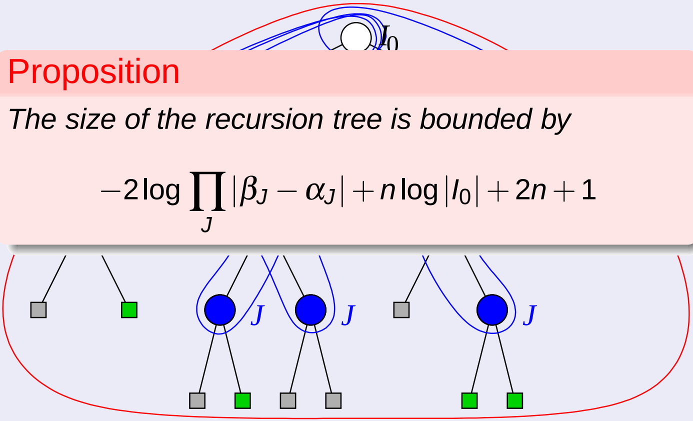
$$
\begin{array}{lcr}
\#(\text{internal nodes on path}) & < & \log|I_0| - \log|\beta_J - \alpha_J| + 2 \\
\#(\text{internal nodes in tree}) & < & \sum_J (\log|I_0| - \log|\beta_J - \alpha_J| + 2) \\
\#(\text{all nodes in tree}) & < & 1 + 2\cdot\sum_J (\log|I_0| - \log|\beta_J - \alpha_J| + 2)
\end{array}
$$

Tree Bound in terms of Roots (2)



$$
\begin{aligned}
\#(\text{internal nodes on path}) &< \log|I_0| - \log|\beta_J - \alpha_J| + 2 \\
\#(\text{internal nodes in tree}) &< \sum_J (\log|I_0| - \log|\beta_J - \alpha_J| + 2) \\
\#(\text{all nodes in tree}) &< 1 + 2 \cdot \sum_J (\log|I_0| - \log|\beta_J - \alpha_J| + 2)
\end{aligned}
$$

Tree Bound in terms of Roots (2)



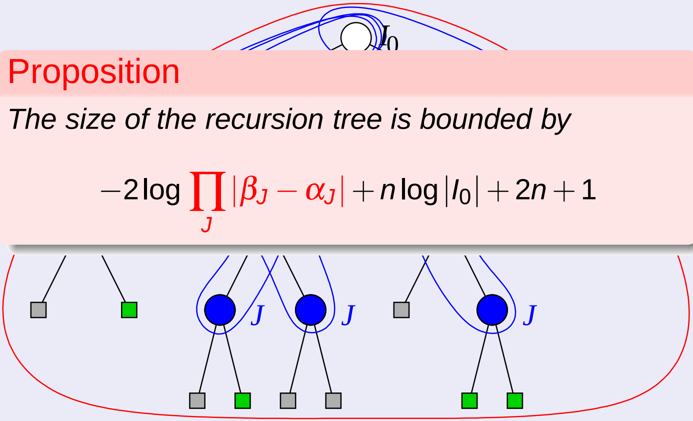
Proposition

The size of the recursion tree is bounded by

$$-2\log\prod_J|\beta_J-\alpha_J|+n\log|I_0|+2n+1$$

$$\begin{array}{lcr}
\#(\text{internal nodes on path}) & < & \log|I_0|-\log|\beta_J-\alpha_J|+2\\
\#(\text{internal nodes in tree}) & < & \sum_J(\log|I_0|-\log|\beta_J-\alpha_J|+2)\\
\#(\text{all nodes in tree}) & < & 1+2\cdot\sum_J(\log|I_0|-\log|\beta_J-\alpha_J|+2)
\end{array}$$

Tree Bound in terms of Roots (2)



Proposition

The size of the recursion tree is bounded by

$$-2\log\prod_J|\beta_J-\alpha_J|+n\log|I_0|+2n+1$$

$$\begin{array}{lcr}
\#(\text{internal nodes on path}) & < & \log|I_0|-\log|\beta_J-\alpha_J|+2 \\
\#(\text{internal nodes in tree}) & < & \sum_J(\log|I_0|-\log|\beta_J-\alpha_J|+2) \\
\#(\text{all nodes in tree}) & < & 1+2\cdot\sum_J(\log|I_0|-\log|\beta_J-\alpha_J|+2)
\end{array}$$

Turning our Product into an Admissible Graph

We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \quad \text{as} \quad \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)

Conditions on $G = (V, E)$

(i) $(\alpha, \beta) \in E \implies |\alpha| \leq |\beta|$ ✓

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓

Turning our Product into an Admissible Graph

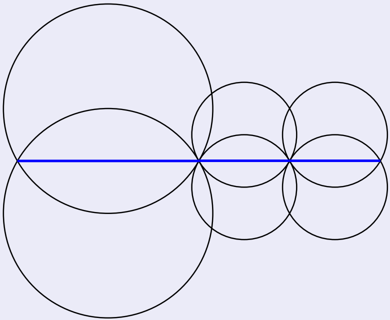
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \text{ as } \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)



Conditions on $G = (V, E)$

(i) $(\alpha,\beta) \in E \implies |\alpha| \leq |\beta|$ ✓

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓

Turning our Product into an Admissible Graph

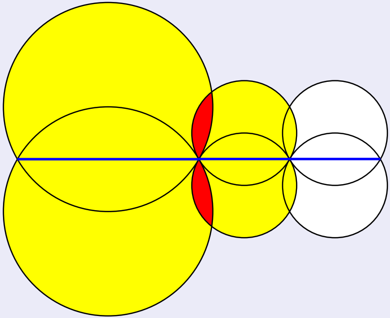
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \quad \text{as} \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)



Conditions on $G = (V, E)$

(i) $(\alpha, \beta) \in E \implies |\alpha| \leq |\beta|$ ✓

(ii) $\beta \in V \implies \text{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓

Turning our Product into an Admissible Graph

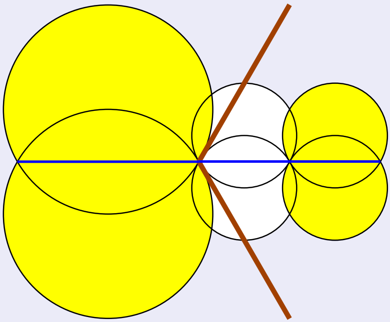
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \text{ as } \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)



Conditions on $G = (V, E)$

(i) $(\alpha, \beta) \in E \implies |\alpha| \leq |\beta|$ ✓

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓

Turning our Product into an Admissible Graph

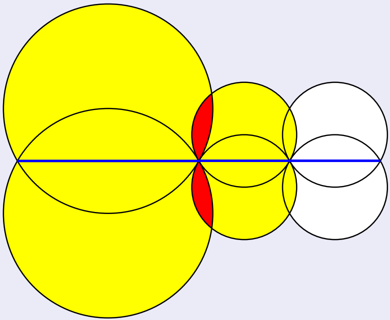
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \quad \text{as} \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate $\leq$ 2

We need two graphs. (Paper: just 1)



Conditions on $G = (V, E)$

(i) $(\alpha, \beta) \in E \implies |\alpha| \leq |\beta|$ ✓

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓

Turning our Product into an Admissible Graph

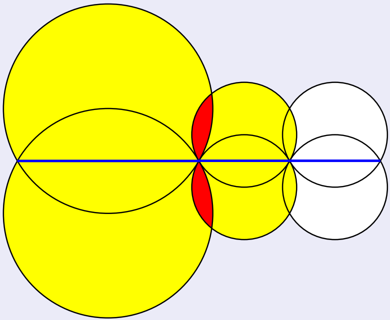
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \text{ as } \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate $\leq$ 2

We need two graphs. (Paper: just 1)

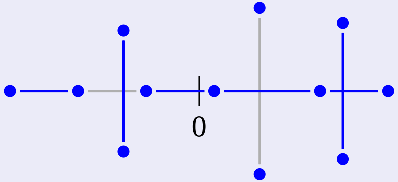


Conditions on $G = (V, E)$

(i) $(\alpha, \beta) \in E \implies |\alpha| \leq |\beta|$ ✓

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓



Turning our Product into an Admissible Graph

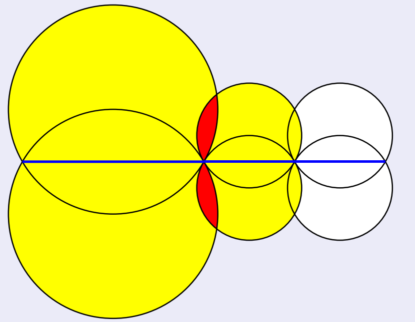
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \text{ as } \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)

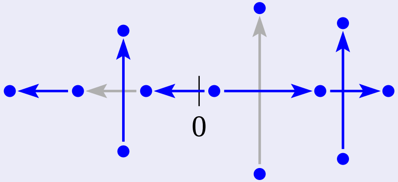


Conditions on $G = (V, E)$

(i) $(\alpha, \beta) \in E \implies |\alpha| \leq |\beta|$ ✓

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓



Turning our Product into an Admissible Graph

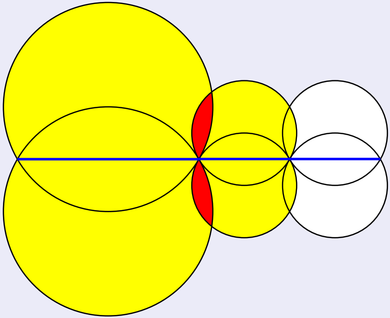
We want to rewrite

$$\prod_J |\beta_J - \alpha_J| \ \text{ as } \prod_{(\alpha,\beta)\in E} |\beta - \alpha|.$$

How often $|\beta_J - \alpha_J|$ appears?

- adjacent real: ≤ 1
- complex conjugate ≤ 2

We need two graphs. (Paper: just 1)

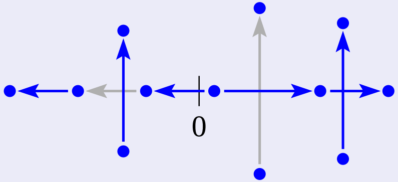


Conditions on $G = (V, E)$

(i) $(\alpha, \beta) \in E \implies |\alpha| \leq |\beta|$ ✓

(ii) $\beta \in V \implies \mathrm{indeg}(\beta) \leq 1$ ✓

(iii) G is acyclic ✓



Main Result on Descartes Analysis

Theorem (Eigenwillig/Sharma/Y. (2006))

On the Benchmark Problem, we obtain

$$|\mathcal{T}| = O(n(L + \log n)).$$

For $L \geq \log n$, this is optimal.

Argument of [Krandick/Mehlhorn, 2006]: $|\mathcal{T}| = O(n \log n (L + \log n))$.

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - "theory caught up with practice"
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - "theory caught up with practice"
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - “theory caught up with practice”
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - “theory caught up with practice”
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - "theory caught up with practice"
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - “theory caught up with practice”
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Mini Summary

- Almost Tight Bound on Descartes Method based on Algebraic Amortization
- Benchmark complexity of Sturm and Descartes are the same
  - "theory caught up with practice"
- What about EVAL?
  - New ideas needed – one is Amortized Evaluation Bounds

Coming Up Next

12 Analysis of Adaptive Complexity

13 Analysis of Descartes Method

14 Integral Bounds and Framework of Stopping Functions

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE $(Q \neq \emptyset)$
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) = C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$

1. WHILE ($Q \neq \emptyset$)
2. $I \leftarrow Q.remove()$
3. IF ($C(I)$ holds), output I
4. ELSE
5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$

1. WHILE ($Q \neq \emptyset$)
2. $\quad I \leftarrow Q.remove()$
3. $\quad$ IF ($C(I)$ holds), output I
4. $\quad$ ELSE
5. $\quad\quad$ Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$
  1. WHILE ($Q \neq \emptyset$)
  2. $I \leftarrow Q.remove()$
  3. IF ($C(I)$ holds), output I
  4. ELSE
  5. Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Subdivision Phase

Subdivision based on a Predicate $C(I)$

- Initialize a queue $Q \leftarrow \{I_0\}$

1. WHILE ($Q \neq \emptyset$)
2. $\quad I \leftarrow Q.remove()$
3. $\quad$ IF ($C(I)$ holds), output I
4. $\quad$ ELSE
5. $\quad\quad$ Split I and insert children into Q

Goal – Bound the size of recursion tree $T(I_0)$

- NOTE: $C(I) \equiv C_0(I) \vee C_1(I)$ in EVAL
- The leaves of $T(I_0)$ induces a partition $P(I)$ of I_0
- Suffices to upper bound $\#P(I_0)$

Framework of Stopping Functions

Stopping Function for $C(I)$ is $F : \mathbb{R} \to \mathbb{R}_{\geq 0}$

For all interval I:

$$\text{If } (\exists b \in I)[w(I) < F(b)],$$

$$\text{then } C(I) \text{ holds.}$$

How to use F? The Penultimate Property

- Similar to Descartes proof
- If $J \in P(I_0)$, its parent ("penultimate leaf") has width $2w(J)$.
- Conclude from definition of stopping function:

$$(\forall c \in J)\ [2w(J) \geq F(c)].$$

Framework of Stopping Functions

Stopping Function for $C(I)$ is $F : \mathbb{R} \to \mathbb{R}_{\geq 0}$

For all interval I:

If $(\exists b \in I)[w(I) < F(b)]$,

then $C(I)$ holds.

How to use F? The Penultimate Property

- Similar to Descartes proof
- If $J \in P(I_0)$, its parent ("penultimate leaf") has width $2w(J)$.
- Conclude from definition of stopping function:

$$(\forall c \in J)\ [2w(J) \geq F(c)].$$

Framework of Stopping Functions

Stopping Function for $C(I)$ is $F : \mathbb{R} \to \mathbb{R}_{\geq 0}$

For all interval I:

If $(\exists b \in I)[w(I) < F(b)]$,

then $C(I)$ holds.

How to use F? The Penultimate Property

- Similar to Descartes proof
- If $J \in P(I_0)$, its parent ("penultimate leaf") has width $2w(J)$.
- Conclude from definition of stopping function:

$$(\forall c \in J)\ [2w(J) \geq F(c)].$$

Framework of Stopping Functions

Stopping Function for $C(I)$ is $F : \mathbb{R} \to \mathbb{R}_{\geq 0}$

For all interval I:

$$\text{If } (\exists b \in I)[w(I) < F(b)],$$

$$\text{then } C(I) \text{ holds.}$$

How to use F? The Penultimate Property

- Similar to Descartes proof
- If $J \in P(I_0)$, its parent ("penultimate leaf") has width $2w(J)$.
- Conclude from definition of stopping function:

$$(\forall c \in J)\ [2w(J) \geq F(c)].$$

Framework of Stopping Functions

Stopping Function for $C(I)$ is $F : \mathbb{R} \to \mathbb{R}_{\geq 0}$

For all interval I:

$$\text{If } (\exists b \in I)[w(I) < F(b)], \quad \text{then } C(I) \text{ holds.}$$

How to use F? The Penultimate Property

- Similar to Descartes proof
- If $J \in P(I_0)$, its parent ("penultimate leaf") has width $2w(J)$.
- Conclude from definition of stopping function:

$$(\forall c \in J)\ [2w(J) \geq F(c)].$$

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \le \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

$$\begin{aligned}\int_J \frac{2dx}{F(x)} &\ge \int_J \frac{2dx}{F(c^*)} \\ &\ge \frac{2}{F(c^*)}\int_J dx \\ &= \frac{2w(J)}{F(c^*)} \\ &\ge 1 \ [PenultimateProp.]\end{aligned}$$

$$\begin{aligned}\int_{I_0} \frac{2dx}{F(x)} &= \sum_{J\in P(I_0)} \int_J \frac{2dx}{F(c^*)} \\ &\ge \sum_{J\in P(I_0)} 1 = \#P(I_0)\end{aligned}$$

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \le \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

$$\begin{aligned}\int_J \frac{2dx}{F(x)} &\ge \int_J \frac{2dx}{F(c^*)}\\ &\ge \frac{2}{F(c^*)}\int_J dx\\ &= \frac{2w(J)}{F(c^*)}\\ &\ge 1 \text{ [PenultimateProp.]}\end{aligned}$$

$$\begin{aligned}\int_{I_0} \frac{2dx}{F(x)} &= \sum_{J\in P(I_0)} \int_J \frac{2dx}{F(c^*)}\\ &\ge \sum_{J\in P(I_0)} 1 = \#P(I_0)\end{aligned}$$

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \le \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

$$\begin{aligned}\int_J \frac{2dx}{F(x)} &\ge \int_J \frac{2dx}{F(c^*)}\\ &\ge \frac{2}{F(c^*)}\int_J dx\\ &= \frac{2w(J)}{F(c^*)}\\ &\ge 1\ [PenultimateProp.]\end{aligned}$$

$$\begin{aligned}\int_{I_0} \frac{2dx}{F(x)} &= \sum_{J\in P(I_0)} \int_J \frac{2dx}{F(c^*)}\\ &\ge \sum_{J\in P(I_0)} 1 = \#P(I_0)\end{aligned}$$

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \le \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

$$\begin{aligned}
\int_J \frac{2dx}{F(x)} &\ge \int_J \frac{2dx}{F(c^*)} \\
&\ge \frac{2}{F(c^*)}\int_J dx \\
&= \frac{2w(J)}{F(c^*)} \\
&\ge 1 \text{ [PenultimateProp.]}
\end{aligned}$$

$$\begin{aligned}
\int_{I_0} \frac{2dx}{F(x)} &= \sum_{J\in P(I_0)} \int_J \frac{2dx}{F(c^*)} \\
&\ge \sum_{J\in P(I_0)} 1 = \#P(I_0)
\end{aligned}$$

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \leq \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

$$\begin{aligned}
\int_J \frac{2dx}{F(x)} &\geq \int_J \frac{2dx}{F(c^*)} \\
&\geq \frac{2}{F(c^*)} \int_J dx \\
&= \frac{2w(J)}{F(c^*)} \\
&\geq 1 \ [PenultimateProp.]
\end{aligned}$$

$$\begin{aligned}
\int_{I_0} \frac{2dx}{F(x)} &= \sum_{J \in P(I_0)} \int_J \frac{2dx}{F(c^*)} \\
&\geq \sum_{J \in P(I_0)} 1 = \#P(I_0)
\end{aligned}$$

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \leq \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

$$\begin{aligned}
\int_J \frac{2dx}{F(x)} &\geq \int_J \frac{2dx}{F(c^*)} \\
&\geq \frac{2}{F(c^*)} \int_J dx \\
&= \frac{2w(J)}{F(c^*)} \\
&\geq 1 \;[PenultimateProp.]
\end{aligned}$$

$$\begin{aligned}
\int_{I_0} \frac{2dx}{F(x)} &= \sum_{J \in P(I_0)} \int_J \frac{2dx}{F(c^*)} \\
&\geq \sum_{J \in P(I_0)} 1 = \#P(I_0)
\end{aligned}$$

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \leq \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

$$\begin{aligned}
\int_J \frac{2dx}{F(x)} &\geq \int_J \frac{2dx}{F(c^*)} \\
&\geq \frac{2}{F(c^*)} \int_J dx \\
&= \frac{2w(J)}{F(c^*)} \\
&\geq 1 \ [PenultimateProp.]
\end{aligned}$$

$$\begin{aligned}
\int_{I_0} \frac{2dx}{F(x)} &= \sum_{J \in P(I_0)} \int_J \frac{2dx}{F(c^*)} \\
&\geq \sum_{J \in P(I_0)} 1 = \#P(I_0)
\end{aligned}$$

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \leq \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

$$\begin{aligned}
\int_J \frac{2dx}{F(x)} &\geq \int_J \frac{2dx}{F(c^*)} \\
&\geq \frac{2}{F(c^*)} \int_J dx \\
&= \frac{2w(J)}{F(c^*)} \\
&\geq 1 \text{ [PenultimateProp.]}
\end{aligned}$$

$$\begin{aligned}
\int_{I_0} \frac{2dx}{F(x)} &= \sum_{J \in P(I_0)} \int_J \frac{2dx}{F(c^*)} \\
&\geq \sum_{J \in P(I_0)} 1 = \#P(I_0)
\end{aligned}$$

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \le \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

$$\begin{aligned}
\int_J \frac{2dx}{F(x)} &\ge \int_J \frac{2dx}{F(c^*)} \\
&\ge \frac{2}{F(c^*)} \int_J dx \\
&= \frac{2w(J)}{F(c^*)} \\
&\ge 1 \; [PenultimateProp.]
\end{aligned}$$

$$\begin{aligned}
\int_{I_0} \frac{2dx}{F(x)} &= \sum_{J \in P(I_0)} \int_J \frac{2dx}{F(c^*)} \\
&\ge \sum_{J \in P(I_0)} 1 = \#P(I_0)
\end{aligned}$$

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \leq \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

$$\begin{aligned}
\int_J \frac{2dx}{F(x)} &\geq \int_J \frac{2dx}{F(c^*)} \\
&\geq \frac{2}{F(c^*)} \int_J dx \\
&= \frac{2w(J)}{F(c^*)} \\
&\geq 1 \; [PenultimateProp.]
\end{aligned}$$

$$\begin{aligned}
\int_{I_0} \frac{2dx}{F(x)} &= \sum_{J \in P(I_0)} \int_J \frac{2dx}{F(c^*)} \\
&\geq \sum_{J \in P(I_0)} 1 = \#P(I_0)
\end{aligned}$$

An Integral Bound

Theorem (Integral Bound [Burr/Krahmer/Y.])

$$\#P(I_0) \le \max\left\{1, \int_{I_0} \frac{2dx}{F(x)}\right\}$$

Proof.

1. If $\#P(I_0) = 1$, result is true.
2. Else pick any $J \in P(I_0)$: it has the penultimate property.
3. Choosing $c^* \in J$ such that $F(c^*)$ is maximum

Pf (contd)

$$\begin{aligned}
\int_J \frac{2dx}{F(x)} &\ge \int_J \frac{2dx}{F(c^*)} \\
&\ge \frac{2}{F(c^*)} \int_J dx \\
&= \frac{2w(J)}{F(c^*)} \\
&\ge 1 \; [PenultimateProp.]
\end{aligned}$$

$$\begin{aligned}
\int_{I_0} \frac{2dx}{F(x)} &= \sum_{J \in P(I_0)} \int_J \frac{2dx}{F(c^*)} \\
&\ge \sum_{J \in P(I_0)} 1 = \#P(I_0)
\end{aligned}$$

Remarks on Integral Bound

- Too hard to directly bound the integral implied by $C_0(I) \vee C_1(I)$.
  - So we devise stopping functions $F(x)$ that can be analyzed.
- Technique of bounding $\int_I \phi(x)dx$ is Continuous Amortization where $\phi(x)$ is charge function.
  - In discrete "amortization arguments", we bound $\sum_{i=1}^{n} \phi(i)$ where $\phi(i)$ is "charge" for the ith operation.
- Ruppert (1995) introduced a similar integral for triangulation.
  - Unlike us, he does not evaluate his integral.

Remarks on Integral Bound

- Too hard to directly bound the integral implied by $C_0(I) \vee C_1(I)$.
  - So we devise stopping functions $F(x)$ that can be analyzed.
- Technique of bounding $\int_I \phi(x)dx$ is Continuous Amortization where $\phi(x)$ is charge function.
  - In discrete "amortization arguments", we bound $\sum_{i=1}^{n} \phi(i)$ where $\phi(i)$ is "charge" for the ith operation.
- Ruppert (1995) introduced a similar integral for triangulation.
  - Unlike us, he does not evaluate his integral.

Remarks on Integral Bound

- Too hard to directly bound the integral implied by $C_0(I) \vee C_1(I)$.
  - So we devise stopping functions $F(x)$ that can be analyzed.
- Technique of bounding $\int_I \phi(x)dx$ is Continuous Amortization where $\phi(x)$ is charge function.
  - In discrete "amortization arguments", we bound $\sum_{i=1}^{n} \phi(i)$ where $\phi(i)$ is "charge" for the ith operation.
- Ruppert (1995) introduced a similar integral for triangulation.
  - Unlike us, he does not evaluate his integral.

Remarks on Integral Bound

- Too hard to directly bound the integral implied by $C_0(I) \vee C_1(I)$.
  - So we devise stopping functions $F(x)$ that can be analyzed.
- Technique of bounding $\int_I \phi(x)dx$ is Continuous Amortization where $\phi(x)$ is charge function.
  - In discrete "amortization arguments", we bound $\sum_{i=1}^n \phi(i)$ where $\phi(i)$ is "charge" for the ith operation.
- Ruppert (1995) introduced a similar integral for triangulation.
  - Unlike us, he does not evaluate his integral.

Remarks on Integral Bound

- Too hard to directly bound the integral implied by $C_0(I) \vee C_1(I)$.
  - So we devise stopping functions $F(x)$ that can be analyzed.
- Technique of bounding $\int_I \phi(x)dx$ is Continuous Amortization where $\phi(x)$ is charge function.
  - In discrete "amortization arguments", we bound $\sum_{i=1}^{n} \phi(i)$ where $\phi(i)$ is "charge" for the ith operation.
- Ruppert (1995) introduced a similar integral for triangulation.
  - Unlike us, he does not evaluate his integral.

Remarks on Integral Bound

- Too hard to directly bound the integral implied by $C_0(I) \vee C_1(I)$.
  - So we devise stopping functions $F(x)$ that can be analyzed.
- Technique of bounding $\int_I \phi(x)dx$ is Continuous Amortization where $\phi(x)$ is charge function.
  - In discrete “amortization arguments”, we bound $\sum_{i=1}^{n} \phi(i)$ where $\phi(i)$ is “charge” for the ith operation.
- Ruppert (1995) introduced a similar integral for triangulation.
  - Unlike us, he does not evaluate his integral.

An Amortized Evaluation Bound

The Idea

- Want lower bounds on $|f(\alpha)|$
- Multivariate version used in [Cheng/Gao/Y. ISSAC'2007]
- Amortization: give lower bounds on $\prod_{i\in J}|f(\alpha_i)|$.

Theorem

Let $F, H \in \mathbb{Z}[X]$ be relatively prime such that $F = \phi\widetilde{\phi}$, $H = \eta\widetilde{\eta}$ where $\phi, \widetilde{\phi}, \eta, \widetilde{\eta} \in \mathbb{C}[X]$ have degrees $m, \widetilde{m}, n, \widetilde{n}$, respectively. If $\beta_1, \ldots, \beta_n$ are all the zeros of $\eta(X)$, then

$$\prod_{i=1}^{n}|\phi(\beta_i)| \geq \frac{1}{\mathrm{lc}(\eta)^m\left((m+1)\|\phi\|\right)^{\widetilde{n}} M(\widetilde{\eta})^m \left((\widetilde{m}+1)\|\widetilde{\phi}\|\right)^{n+\widetilde{n}} M(H)^{\widetilde{m}}}.$$

An Amortized Evaluation Bound

The Idea

- Want lower bounds on $|f(\alpha)|$
- Multivariate version used in [Cheng/Gao/Y. ISSAC'2007]
- Amortization: give lower bounds on $\prod_{i\in J}|f(\alpha_i)|$.

Theorem

Let $F,H \in \mathbb{Z}[X]$ be relatively prime such that $F = \phi\widetilde{\phi}$, $H = \eta\widetilde{\eta}$ where $\phi,\widetilde{\phi},\eta,\widetilde{\eta} \in \mathbb{C}[X]$ have degrees $m,\widetilde{m},n,\widetilde{n}$, respectively. If $\beta_1,\ldots,\beta_n$ are all the zeros of $\eta(X)$, then

$$\prod_{i=1}^{n}|\phi(\beta_i)| \geq \frac{1}{\mathrm{lc}(\eta)^m\left((m+1)\|\phi\|\right)^{\widetilde{n}} M(\widetilde{\eta})^m \left((\widetilde{m}+1)\|\widetilde{\phi}\|\right)^{n+\widetilde{n}} M(H)^{\widetilde{m}}}.$$

An Amortized Evaluation Bound

The Idea

- Want lower bounds on $|f(\alpha)|$
- Multivariate version used in [Cheng/Gao/Y. ISSAC'2007]
- Amortization: give lower bounds on $\prod_{i \in J} |f(\alpha_i)|$.

Theorem

Let $F, H \in \mathbb{Z}[X]$ be relatively prime such that $F = \phi\widetilde{\phi}$, $H = \eta\widetilde{\eta}$ where $\phi, \widetilde{\phi}, \eta, \widetilde{\eta} \in \mathbb{C}[X]$ have degrees $m, \widetilde{m}, n, \widetilde{n}$, respectively. If $\beta_1, \ldots, \beta_n$ are all the zeros of $\eta(X)$, then

$$\prod_{i=1}^{n} |\phi(\beta_i)| \geq \frac{1}{\mathrm{lc}(\eta)^m \left((m+1)\|\phi\|\right)^{\widetilde{n}} M(\widetilde{\eta})^m \left((\widetilde{m}+1)\|\widetilde{\phi}\|\right)^{n+\widetilde{n}} M(H)^{\widetilde{m}}}.$$

An Amortized Evaluation Bound

The Idea

- Want lower bounds on $|f(\alpha)|$
- Multivariate version used in [Cheng/Gao/Y. ISSAC'2007]
- Amortization: give lower bounds on $\prod_{i\in J}|f(\alpha_i)|$.

Theorem

Let $F,H\in\mathbb{Z}[X]$ be relatively prime such that $F=\phi\widetilde{\phi}$, $H=\eta\widetilde{\eta}$ where $\phi,\widetilde{\phi},\eta,\widetilde{\eta}\in\mathbb{C}[X]$ have degrees $m,\widetilde{m},n,\widetilde{n}$, respectively. If $\beta_1,\ldots,\beta_n$ are all the zeros of $\eta(X)$, then

$$\prod_{i=1}^{n}|\phi(\beta_i)|\geq\frac{1}{\mathrm{lc}(\eta)^m\left((m+1)\|\phi\|\right)^{\widetilde{n}}M(\widetilde{\eta})^m\left((\widetilde{m}+1)\|\widetilde{\phi}\|\right)^{n+\widetilde{n}}M(H)^{\widetilde{m}}}.$$

An Amortized Evaluation Bound

The Idea

- Want lower bounds on $|f(\alpha)|$
- Multivariate version used in [Cheng/Gao/Y. ISSAC'2007]
- Amortization: give lower bounds on $\prod_{i\in J}|f(\alpha_i)|$.

Theorem

Let $F, H \in \mathbb{Z}[X]$ *be relatively prime such that* $F = \phi\widetilde{\phi}$, $H = \eta\widetilde{\eta}$ *where* $\phi, \widetilde{\phi}, \eta, \widetilde{\eta} \in \mathbb{C}[X]$ *have degrees* $m, \widetilde{m}, n, \widetilde{n}$, *respectively. If* $\beta_1, \ldots, \beta_n$ *are all the zeros of* $\eta(X)$, *then*

$$\prod_{i=1}^{n}|\phi(\beta_i)| \geq \frac{1}{\mathrm{lc}(\eta)^m\left((m+1)\|\phi\|\right)^{\widetilde{n}} M(\widetilde{\eta})^m \left((\widetilde{m}+1)\|\widetilde{\phi}\|\right)^{n+\widetilde{n}} M(H)^{\widetilde{m}}}.$$

Complex Roots: Lesson from Meshing Curves

How to isolate complex roots?

- Previous subdivision methods:
  - Pan-Weyl Algorithm (Turan Test)
  - Root isolation on boundary of boxes (topological degree)
- Hints from Curve Meshing (Snyder/PV/Cxy) – not good idea

New Result (with Sagraloff)

There is an exact analog CEVAL for complex roots that is simple and easy to implement exactly.
It achieves the same bit complexity bound as in the real case.

Complex Roots: Lesson from Meshing Curves

How to isolate complex roots?

- Previous subdivision methods:
  - Pan-Weyl Algorithm (Turan Test)
  - Root isolation on boundary of boxes (topological degree)
- Hints from Curve Meshing (Snyder/PV/Cxy) – not good idea

New Result (with Sagraloff)

There is an exact analog CEVAL for complex roots that is simple and easy to implement exactly.
It achieves the same bit complexity bound as in the real case.

Complex Roots: Lesson from Meshing Curves

How to isolate complex roots?

- Previous subdivision methods:
  - Pan-Weyl Algorithm (Turan Test)
  - Root isolation on boundary of boxes (topological degree)
- Hints from Curve Meshing (Snyder/PV/Cxy) – not good idea

New Result (with Sagraloff)

There is an exact analog CEVAL for complex roots that is simple and easy to implement exactly.
It achieves the same bit complexity bound as in the real case.

Complex Roots: Lesson from Meshing Curves

How to isolate complex roots?

- Previous subdivision methods:
  - Pan-Weyl Algorithm (Turan Test)
  - Root isolation on boundary of boxes (topological degree)
- Hints from Curve Meshing (Snyder/PV/Cxy) – not good idea

New Result (with Sagraloff)

There is an exact analog CEVAL for complex roots that is simple and easy to implement exactly.
It achieves the same bit complexity bound as in the real case.

Mini Summary

- The Bolzano approach to Root Isolation is an Exact and Analytic approach to root isolation
- It seems to have complexity that matches Sturm and Descartes
- It is much easier to implement than either

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Lecture 3

- Complexity Analysis of Adaptivity at infancy
- Analysis Techniques we have seen so far:
  - Continuous amortization via integral bounds
  - Amortized root separation bounds
  - Amortized evaluation bounds
  - Cluster analysis
- Major Open Problems
  - How to characterize local complexity?
  - How to extend to higher dimensions

Summary of Tutorial

- MANY advantages in numerical/analytic approaches to algebraic and geometric problems
  - practical, adaptive, easy to implement
- New ingredient we seek: a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- MANY advantages in numerical/analytic approaches to algebraic and geometric problems
  - practical, adaptive, easy to implement
- New ingredient we seek: a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- MANY advantages in numerical/analytic approaches to algebraic and geometric problems
  - practical, adaptive, easy to implement
- New ingredient we seek: a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- MANY advantages in numerical/analytic approaches to algebraic and geometric problems
  - practical, adaptive, easy to implement
- New ingredient we seek: a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- MANY advantages in numerical/analytic approaches to algebraic and geometric problems
  - practical, adaptive, easy to implement
- New ingredient we seek: a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- MANY advantages in numerical/analytic approaches to algebraic and geometric problems
  - practical, adaptive, easy to implement
- New ingredient we seek: a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- MANY advantages in numerical/analytic approaches to algebraic and geometric problems
  - practical, adaptive, easy to implement
- New ingredient we seek: a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- MANY advantages in numerical/analytic approaches to algebraic and geometric problems
  - practical, adaptive, easy to implement
- New ingredient we seek: a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Summary of Tutorial

- MANY advantages in numerical/analytic approaches to algebraic and geometric problems
  - practical, adaptive, easy to implement
- New ingredient we seek: a priori guarantees and exactness
- Zero problems is the locus of our investigation
- Exact Numerical Computation (ENC) is a suitable computational model
- The explicitization problems are central for ENC
- Analysis of adaptive algorithms is wide open

Thank you!

Website http://cs.nyu.edu/exact/

- Download Papers
- Download Core Library

GENERAL REFERENCE

Chee K. Yap.

Theory of Real Computation according to EGC.

In P. Hertling, Ch.M. Hoffmann, W. Luther, and N.Revol, editors, *Reliable Implementation of Real Number Algorithms: Theory and Practice*, No. 5045 in LNCS, pp. 193–237. Springer, 2008.