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

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

$\boxed{\text{Design algorithms directly incorporating the principles of EGC}}$

- 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:

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

- 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!)

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

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

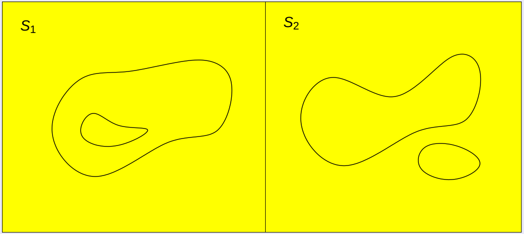
  [Ratschek & Rokne (2005)]
- Subdivision Approach

  [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:

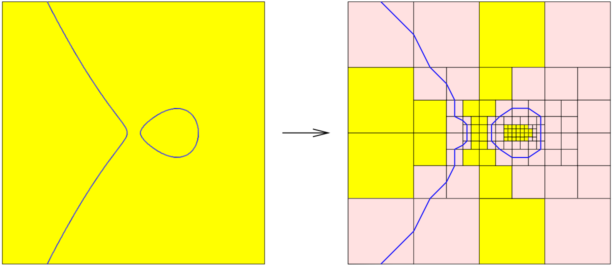


Figure: Approximation of the curve $f(X,Y) = Y^2 - X^2 + X^3 + 0.02 = 0$

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: $G \leftarrow CONSTRUCT(Q_{ref})$

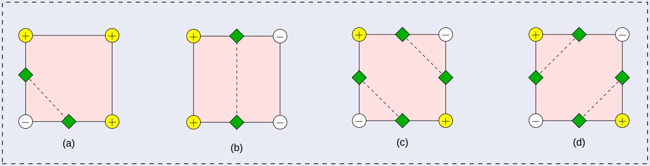
E.g., Marching Cube

Subdivision Phase

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

Construction Phase

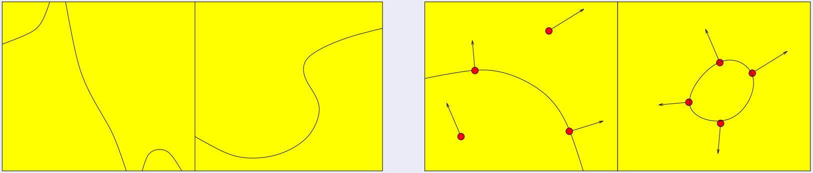
(1) Evaluate sign of f at grid points, (2) insert vertices, and (3) connect them in each box:



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	$0 \notin \square f(B)$	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	$0 \notin \Box f(B)$	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):
  - $\square f(I,J) = I^2 - 2IJ + 3J$
- 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	$0 \notin \square f(B)$	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	$0 \notin \square f(B)$	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

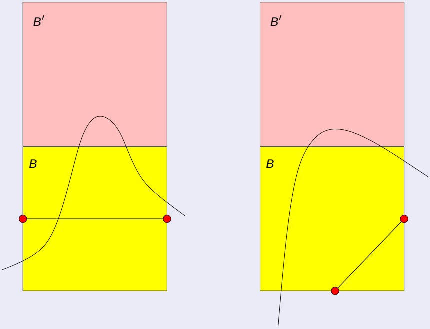
- Determine intersections on boundary
- Connect the intersections
- (Non-trivial, unbounded complexity)

$\boxed{\text{Boundary Analysis is not good (may not even terminate).}}$

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:

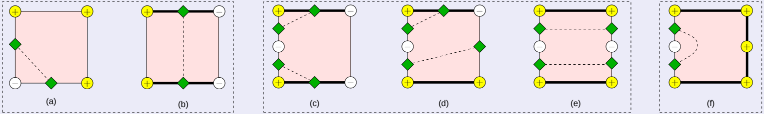


Figure: Extended Rules

- 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?

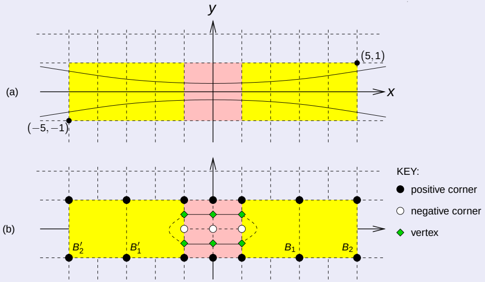


Figure: Elongated hyperbola

Cxy Algorithm

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

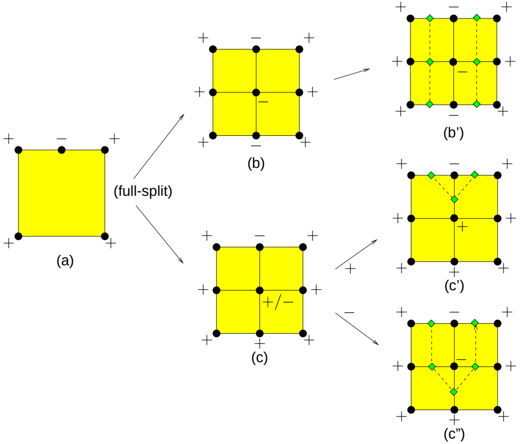


Figure: Resolution of Ambiguity

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

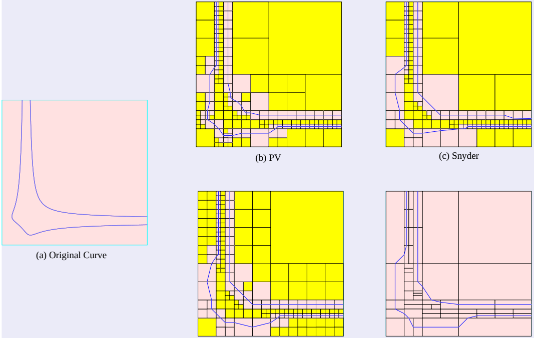
Mini Summary

- 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



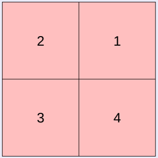
(a) Original Curve (b) PV (c) Snyder (d) Balanced Cxy (e) Rectangular Cxy

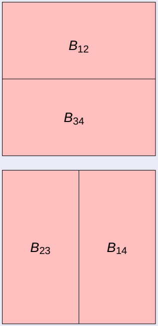
- "Heel Curve" $X^2Y^2 - X + Y - 1 = 0$ in box $B = [(-2, -10), (10, 2)]$
- Comparing PV, Snyder, Cxy, Rect Cxy

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

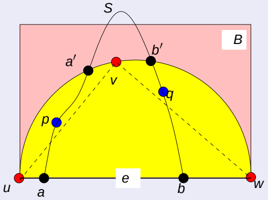
L_0 :	$C_0(B), C_{xy}(B)$	Terminate
L_{out} :	$C_0(B_{12}), C_0(B_{34}), C_0(B_{14}), C_0(B_{23})$	Half-split
L_{in} :	$C_{xy}(B_{12}), C_{xy}(B_{34}), C_{xy}(B_{14}), C_{xy}(B_{23})$	Half-split

- 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 $B_n := [(-14\times10^n, -14\times10^n), (15\times10^n, 15\times10^n)]$. 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

  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

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

- 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

  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

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

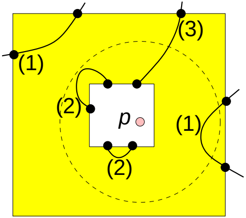
where

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

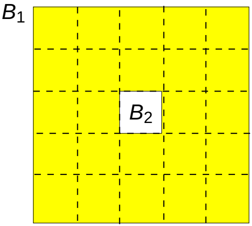
Can provide a subdivision algorithm using F, ε_0 to isolate regions containing singularities.

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