Robustness in Geometric Computation

Vikram Sharma[1]

[1] Institute of Mathematical Sciences, Chennai, India.

Mysore Park Workshop, August, 2013

Outline

1. Nonrobustness
   - In Computational Geometry: Example
   - The Exact Geometric Computation (EGC) Approach
   - Open Problems & Further Directions

2. Real Root Isolation
   - Two Algorithms & Their Analysis
   - Open Problems & Further Directions

What is Non-robustness?

Behaviour of a program

Inconsistent results, Infinite loops, “Crashes” (Segmentation Fault).

Implications

- Disasters caused by malfunctioning of software (e.g., Ariane crash in 1996).
- Reduces programmer’s effectiveness – time spent in debugging programs.

Computational Geometry Early Days – Theory

- People: Shamos, Preparata, Graham, Fortune ...
- Efficient algorithms for Convex Hulls, Voronoi Diagrams, Delaunay Triangulations ...

Computational Geometry Early Days – Theory

- People: Shamos, Preparata, Graham, Fortune ...
- Efficient algorithms for Convex Hulls, Voronoi Diagrams, Delaunay Triangulations ...

BCKO, Computational Geometry, Algo. and Appl.

"Robustness problems are often a cause of frustration when implementing geometric algorithms."

Computational Geometry Early Days – Theory

- People: Shamos, Preparata, Graham, Fortune ...
- Efficient algorithms for Convex Hulls, Voronoi Diagrams, Delaunay Triangulations ...

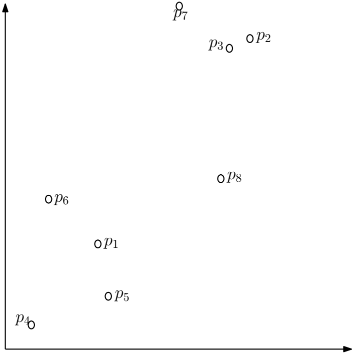
BCKO, Computational Geometry, Algo. and Appl.

> *"Robustness problems are often a cause of frustration when implementing geometric algorithms."*

But haven't the Numerical Analysts addressed nonrobustness?

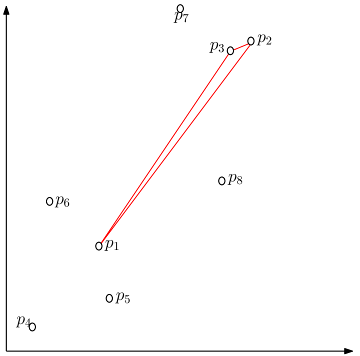
Backward stability, Forward-error analysis.

Computing Convex Hull – An Incremental Algorithm



$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

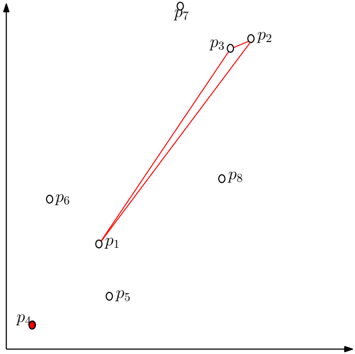
Computing Convex Hull – An Incremental Algorithm



$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

1. Three non-collinear points.

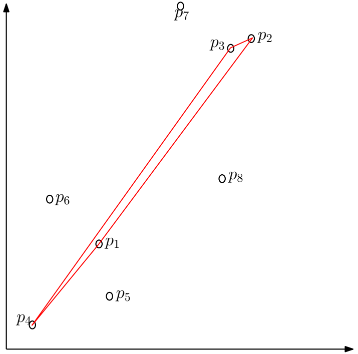
Computing Convex Hull – An Incremental Algorithm



$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

1. Three non-collinear points.
2. Is p_4 is in $\Delta(p_1, p_2, p_3)$?

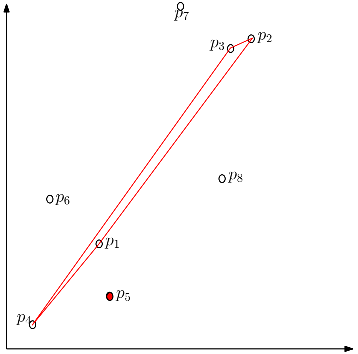
Computing Convex Hull – An Incremental Algorithm



$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

1. Three non-collinear points.
2. Is p_4 is in $\Delta(p_1, p_2, p_3)$?
3. Update convex hull to p_1, p_2, p_3, p_4.

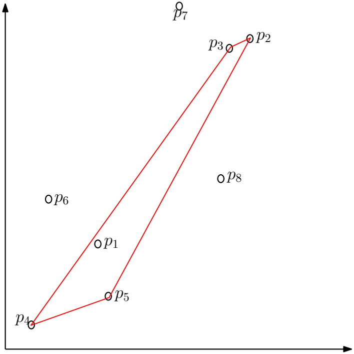
Computing Convex Hull – An Incremental Algorithm



$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

1. Three non-collinear points.
2. Is p_4 is in $\Delta(p_1, p_2, p_3)$?
3. Update convex hull to p_1, p_2, p_3, p_4.
4. Check if p_5 is in the convex hull.

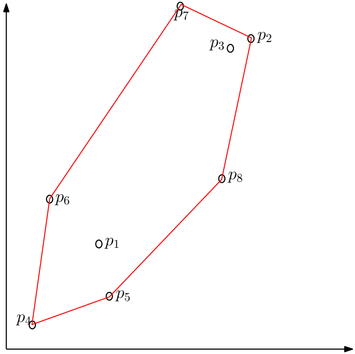
Computing Convex Hull – An Incremental Algorithm



$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

1. Three non-collinear points.
2. Is p_4 is in $\Delta(p_1, p_2, p_3)$?
3. Update convex hull to p_1, p_2, p_3, p_4.
4. Check if p_5 is in the convex hull.
5. Update, and continue...

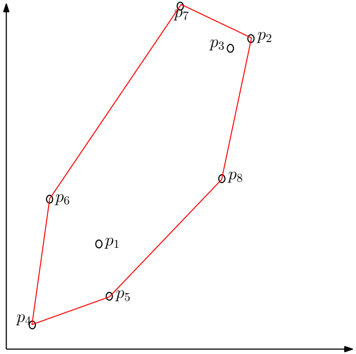
Computing Convex Hull – An Incremental Algorithm



$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

1. Three non-collinear points.
2. Is p_4 is in $\Delta(p_1, p_2, p_3)$?
3. Update convex hull to p_1, p_2, p_3, p_4.
4. Check if p_5 is in the convex hull.
5. Update, and continue...

Computing Convex Hull – An Incremental Algorithm

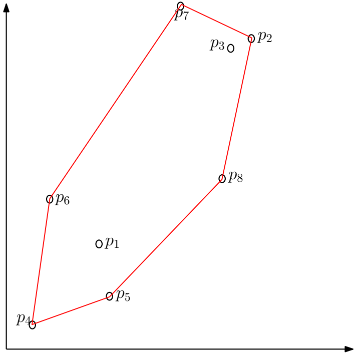


$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

1. Three non-collinear points.
2. Is p_4 is in $\Delta(p_1, p_2, p_3)$?
3. Update convex hull to p_1, p_2, p_3, p_4.
4. Check if p_5 is in the convex hull.
5. Update, and continue...

Orientation: $\text{orient}(p, q, r) \in \{L, R, S\}$.
$\text{orient}(p, q, r) = L$ iff (p, q, r) is left turn.

Computing Convex Hull – An Incremental Algorithm

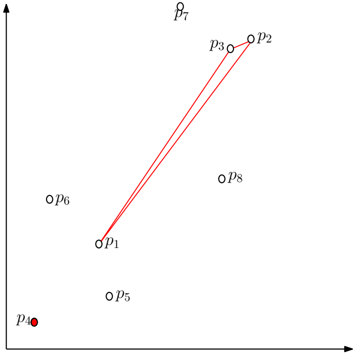


$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

1. Three non-collinear points.
2. Is p_4 is in $\Delta(p_1, p_2, p_3)$?
3. Update convex hull to p_1, p_2, p_3, p_4.
4. Check if p_5 is in the convex hull.
5. Update, and continue...

Orientation: $\text{orient}(p, q, r) \in \{L, R, S\}$.
$\text{orient}(p, q, r) = L$ iff (p, q, r) is left turn.

Given $CH(p_1, \ldots, p_k)$ and p. If $\forall i$, $\text{orient}(p_i, p_{i+1}, p) = L$ then p is in CH.

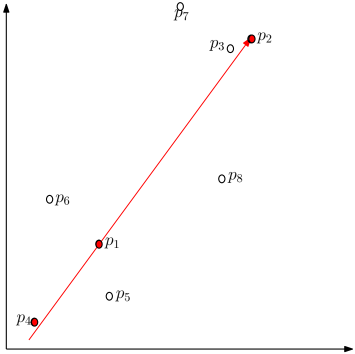


$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

Algebraic form of orient

Sign of
$((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$.
"L" is $+1$, "R" is -1, "S" is 0.
In practice, fl_orient(p, q, r).

1. Is p_4 in $\Delta(p_1, p_2, p_3)$?

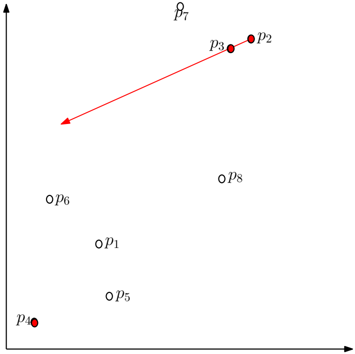


$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

Algebraic form of orient

Sign of
$((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$.
"L" is $+1$, "R" is -1, "S" is 0.
In practice, fl_orient(p, q, r).

1. Is p_4 in $\Delta(p_1, p_2, p_3)$?
2. Is fl_orient$(p_1, p_2, p_4) = L$? Yes.

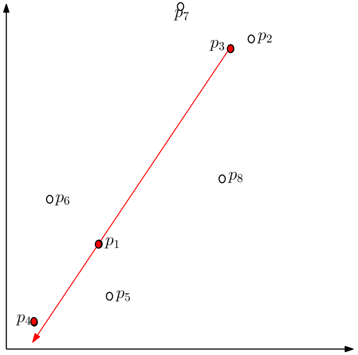


$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)\ \ p_6 = (4, 9)\ \ p_7 = (15, 27)\ \ p_8 = (19, 11).$

Algebraic form of orient

Sign of
$((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x)).$
"L" is $+1$, "R" is -1, "S" is 0.
In practice, fl_orient(p, q, r).

1. Is p_4 in $\Delta(p_1, p_2, p_3)$?
2. Is fl_orient$(p_1, p_2, p_4) = L$? Yes.
3. Is fl_orient$(p_2, p_3, p_4) = L$? Yes.

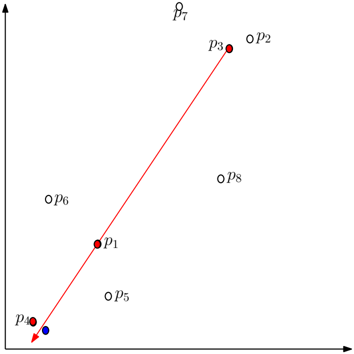


$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

Algebraic form of orient

Sign of
$((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$.
"L" is $+1$, "R" is -1, "S" is 0.
In practice, fl_orient(p, q, r).

1. Is p_4 in $\Delta(p_1, p_2, p_3)$?
2. Is fl_orient$(p_1, p_2, p_4) = L$? Yes.
3. Is fl_orient$(p_2, p_3, p_4) = L$? Yes.
4. Is fl_orient$(p_3, p_1, p_4) = L$?

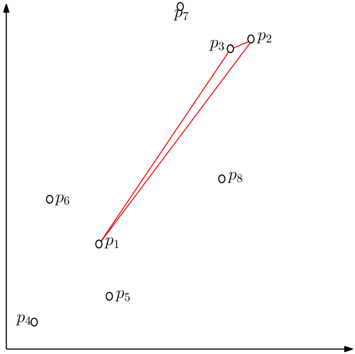


$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8,4)$ $p_6 = (4,9)$ $p_7 = (15,27)$ $p_8 = (19,11)$.

Algebraic form of orient

Sign of
$((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$.
"L" is $+1$, "R" is -1, "S" is 0.
In practice, fl_orient(p, q, r).

1. Is p_4 in $\Delta(p_1, p_2, p_3)$?
2. Is fl_orient$(p_1, p_2, p_4) = L$? Yes.
3. Is fl_orient$(p_2, p_3, p_4) = L$? Yes.
4. Is fl_orient$(p_3, p_1, p_4) = L$?
5. Yes.

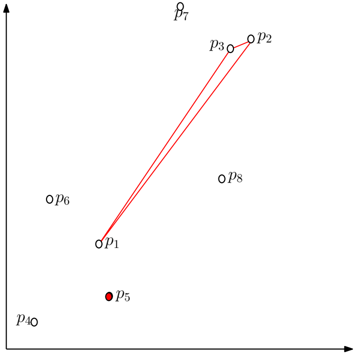


$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

Algebraic form of orient

Sign of
$((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$.
"L" is $+1$, "R" is -1, "S" is 0.
In practice, fl_orient(p, q, r).

1. Is p_4 in $\Delta(p_1, p_2, p_3)$?
2. Is fl_orient$(p_1, p_2, p_4) = L$? Yes.
3. Is fl_orient$(p_2, p_3, p_4) = L$? Yes.
4. Is fl_orient$(p_3, p_1, p_4) = L$?
5. Yes.

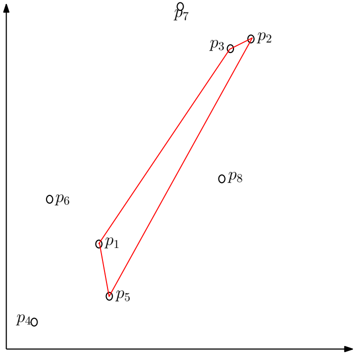


$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

Algebraic form of orient

Sign of
$((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$.
“L” is $+1$, “R” is -1, “S” is 0.
In practice, fl_orient(p, q, r).

1. Is p_4 in $\Delta(p_1, p_2, p_3)$?
2. Is fl_orient$(p_1, p_2, p_4) = L$? Yes.
3. Is fl_orient$(p_2, p_3, p_4) = L$? Yes.
4. Is fl_orient$(p_3, p_1, p_4) = L$?
5. Yes.
6. Is p_5 in $\Delta(p_1, p_2, p_3)$? No.

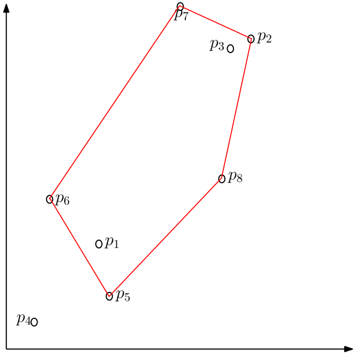


$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

Algebraic form of orient

Sign of
$((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$.
"L" is $+1$, "R" is -1, "S" is 0.
In practice, fl_orient(p, q, r).

1. Is p_4 in $\Delta(p_1, p_2, p_3)$?
2. Is fl_orient$(p_1, p_2, p_4) = L$? Yes.
3. Is fl_orient$(p_2, p_3, p_4) = L$? Yes.
4. Is fl_orient$(p_3, p_1, p_4) = L$?
5. Yes.
6. Is p_5 in $\Delta(p_1, p_2, p_3)$? No.
7. Update, and continue...

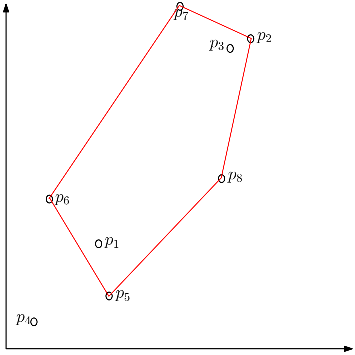


$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

Algebraic form of orient

Sign of
$((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$.
"L" is $+1$, "R" is -1, "S" is 0.
In practice, fl_orient(p, q, r).

1. Is p_4 in $\Delta(p_1, p_2, p_3)$?
2. Is fl_orient$(p_1, p_2, p_4) = L$? Yes.
3. Is fl_orient$(p_2, p_3, p_4) = L$? Yes.
4. Is fl_orient$(p_3, p_1, p_4) = L$?
5. Yes.
6. Is p_5 in $\Delta(p_1, p_2, p_3)$? No.
7. Update, and continue...



$p_1 = (7.3000000000000194, 7.3000000000000167)$
$p_2 = (24.000000000000068, 24.000000000000071)$
$p_3 = (24.00000000000005, 24.000000000000053)$
$p_4 = (0.50000000000001621, 0.50000000000001243)$
$p_5 = (8, 4)$ $p_6 = (4, 9)$ $p_7 = (15, 27)$ $p_8 = (19, 11)$.

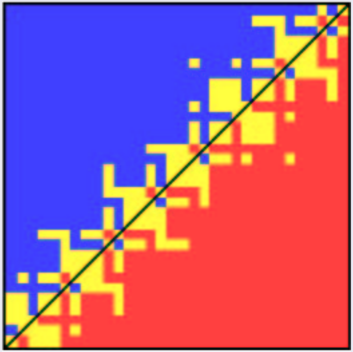
Algebraic form of orient

Sign of
$((q_x - p_x)(r_y - p_y) - (q_y - p_y)(r_x - p_x))$.
"L" is $+1$, "R" is -1, "S" is 0.
In practice, fl_orient(p, q, r).

1. Is p_4 in $\Delta(p_1, p_2, p_3)$?
2. Is fl_orient$(p_1, p_2, p_4) = L$? Yes.
3. Is fl_orient$(p_2, p_3, p_4) = L$? Yes.
4. Is fl_orient$(p_3, p_1, p_4) = L$?
5. Yes.
6. Is p_5 in $\Delta(p_1, p_2, p_3)$? No.
7. Update, and continue...

p_1, p_2, p_3, p_4 are almost collinear.

Geometry of fl_orient



- $p = (0.5, 0.5), q = (12, 12), r = (24, 24)$.
- $\text{fl\_orient}((p_x + iu, p_y + ju), q, r)$, $0 \leq i, j \leq 255$.
- $u = 2^{-53}$.

- Left turn
- Right turn
- Collinear

Kettner et al.'06

Classroom Examples of Robustness Problems in Geom. Comp.

The Exact Geometric Computation (EGC) Approach

The Exact Geometric Computation (EGC) Approach

Geometric Algorithms [Yap'94]

- Combinatorial structure representing discrete relations amongst geometric objects.
  E.g., a point is to the left, right or on a line.

The Exact Geometric Computation (EGC) Approach

Geometric Algorithms [Yap'94]

- Combinatorial structure representing discrete relations amongst geometric objects.
  E.g., a point is to the left, right or on a line.
- Numerical representation of the geometric objects.
  E.g., floating-point representation of coordinates.

The Exact Geometric Computation (EGC) Approach

Geometric Algorithms [Yap'94]

- Combinatorial structure representing discrete relations amongst geometric objects.
  E.g., a point is to the left, right or on a line.
- Numerical representation of the geometric objects.
  E.g., floating-point representation of coordinates.
- Characterize the combinatorial structure by verifying the discrete relations using numerical computations.
  E.g., using the orientation predicate.

The Exact Geometric Computation (EGC) Approach

Geometric Algorithms [Yap'94]

- Combinatorial structure representing discrete relations amongst geometric objects.
  E.g., a point is to the left, right or on a line.
- Numerical representation of the geometric objects.
  E.g., floating-point representation of coordinates.
- Characterize the combinatorial structure by verifying the discrete relations using numerical computations.
  E.g., using the orientation predicate.

Cause of Non-robustness

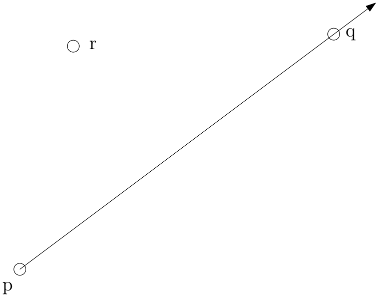
Numerical errors may give incorrect characterization.

The Exact Geometric Computation (EGC) Approach

The EGC solution

- Compute correct discrete relations between geometric objects.
- Correct sign evaluation of geometric predicates.
- What are geometric predicates?

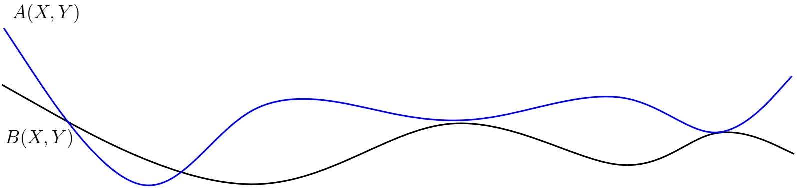
The Exact Geometric Computation (EGC) Approach



Orientation predicate

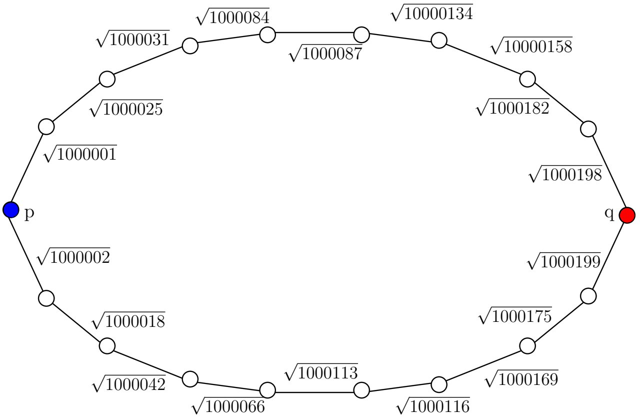
Whether r is to the left, right, or collinear with (p, q)?

The Exact Geometric Computation (EGC) Approach



Whether two algebraic curves intersect?

The Exact Geometric Computation (EGC) Approach



Which is the shorter path?

Almost equal – difference is smaller than 10^{-36}.

The Exact Geometric Computation (EGC) Approach

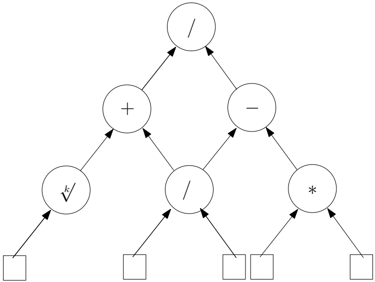
Geometric Predicates

Sign of (multivariate) polynomials evaluated at real algebraic numbers

Real Algebraic Numbers

Real roots of integer polynomials in one variable.
E.g. $\pm\sqrt{n}$, for positive integer n, $\frac{1+\sqrt{5}}{2}$, but not π, e.

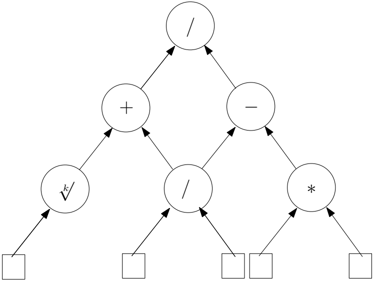
The Exact Geometric Computation (EGC) Approach



Numerical Representation of α

- Data Structure – Rooted DAG, G(α).

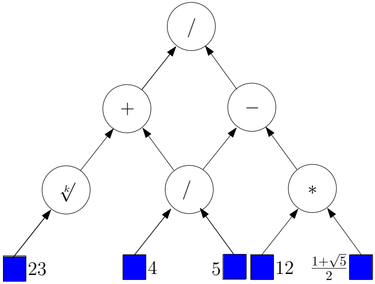
The Exact Geometric Computation (EGC) Approach



Numerical Representation of α

- Data Structure – Rooted DAG, G(α).
- Internal nodes – algebraic operations.

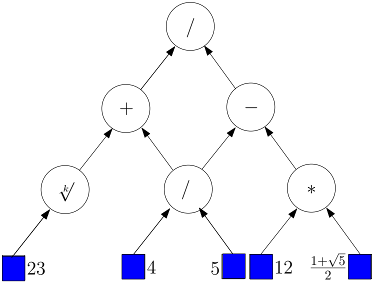
The Exact Geometric Computation (EGC) Approach



Numerical Representation of α

- Data Structure – Rooted DAG, G(α).
- Internal nodes – algebraic operations.
- Leaves – integers or real algebraic numbers.

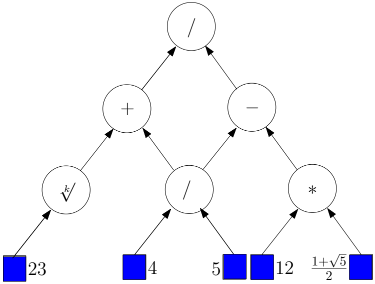
The Exact Geometric Computation (EGC) Approach



Numerical Representation of α

- Data Structure – Rooted DAG, G(α).
- Internal nodes – algebraic operations.
- Leaves – integers or real algebraic numbers.
- Isolating interval representation:
  1. $A(X) \in \mathbb{Z}[X]$, $A(\alpha) = 0$,
  2. Interval separating α from other roots (conjugates) of $A(X)$.

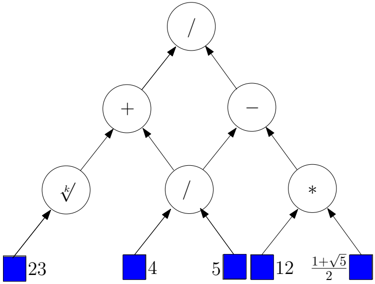
The Exact Geometric Computation (EGC) Approach



Numerical Representation of α

- Data Structure – Rooted DAG, G(α).
- Internal nodes – algebraic operations.
- Leaves – integers or real algebraic numbers.
- Isolating interval representation:
  1. $A(X) \in \mathbb{Z}[X]$, $A(\alpha) = 0$,
  2. Interval separating α from other roots (conjugates) of $A(X)$.
- Arithmetic is manipulating DAGs.

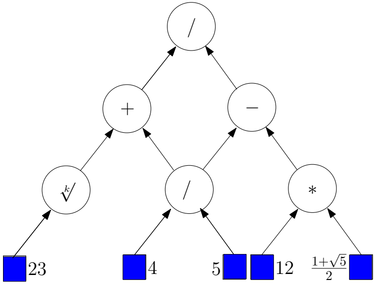
The Exact Geometric Computation (EGC) Approach



Comparing two numbers: $\alpha = \beta$?

- Input $G(\alpha)$ and $G(\beta)$.

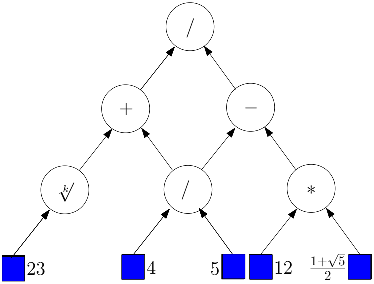
The Exact Geometric Computation (EGC) Approach



Comparing two numbers: $\alpha = \beta$?

- Input $G(\alpha)$ and $G(\beta)$.
- *Constructive Zero Bounds*: G(α), constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.

The Exact Geometric Computation (EGC) Approach



Recursive rules, e.g.

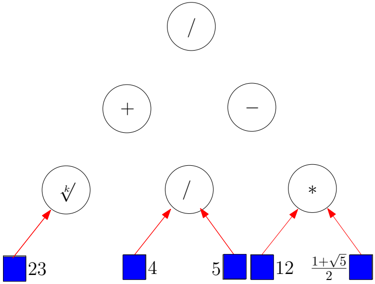
- $|a/b + c/d| \geq 1/|cd|$,
- if $a > 2^{-b}$ then $|\sqrt[k]{a}| > 2^{-bk}$.

Burnikel et al.'99, Li-Yap'00 etc.

Comparing two numbers: $\alpha = \beta$?

- Input $G(\alpha)$ and $G(\beta)$.
- *Constructive Zero Bounds*: $\mathrm{G}(\alpha)$, constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.

The Exact Geometric Computation (EGC) Approach



Recursive rules, e.g.

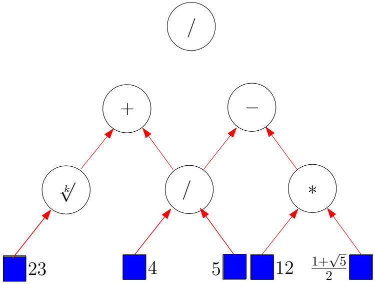
- $|a/b+c/d| \geq 1/|cd|$,
- if $a > 2^{-b}$ then $|\sqrt[k]{a}| > 2^{-bk}$.

Burnikel et al.'99, Li-Yap'00 etc.

Comparing two numbers: $\alpha = \beta$?

- Input $G(\alpha)$ and $G(\beta)$.
- *Constructive Zero Bounds*: $\mathrm{G}(\alpha)$, constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.

The Exact Geometric Computation (EGC) Approach



Recursive rules, e.g.

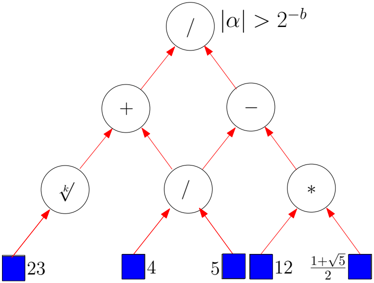
- $|a/b+c/d| \geq 1/|cd|$,
- if $a > 2^{-b}$ then $|\sqrt[k]{a}| > 2^{-bk}$.

Burnikel et al.'99, Li-Yap'00 etc.

Comparing two numbers: $\alpha = \beta$?

- Input $G(\alpha)$ and $G(\beta)$.
- *Constructive Zero Bounds*: $G(\alpha)$, constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.

The Exact Geometric Computation (EGC) Approach



Recursive rules, e.g.

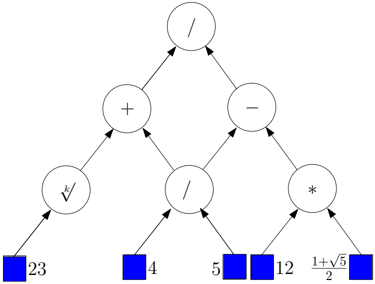
- $|a/b + c/d| \geq 1/|cd|$,
- if $a > 2^{-b}$ then $|\sqrt[k]{a}| > 2^{-bk}$.

Burnikel et al.'99, Li-Yap'00 etc.

Comparing two numbers: $\alpha = \beta$?

- Input $G(\alpha)$ and $G(\beta)$.
- *Constructive Zero Bounds*: G(α), constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.

The Exact Geometric Computation (EGC) Approach



Recursive rules, e.g.

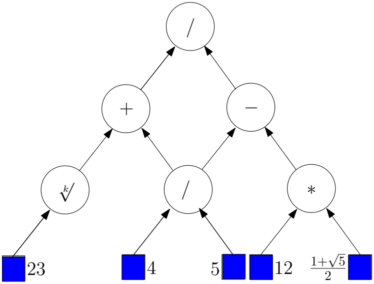
- $|a/b + c/d| \geq 1/|cd|$,
- if $a > 2^{-b}$ then $|\sqrt[k]{a}| > 2^{-bk}$.

Burnikel et al.'99, Li-Yap'00 etc.

Comparing two numbers: $\alpha = \beta$?

- Input $G(\alpha)$ and $G(\beta)$.
- *Constructive Zero Bounds*: $\mathrm{G}(\alpha)$, constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.
- Construct zero bound for $\mathrm{G}(\alpha) \ominus \mathrm{G}(\beta)$ to get b s.t. $|\alpha - \beta| > 2^{-b}$, if $\alpha \neq \beta$.

The Exact Geometric Computation (EGC) Approach



Recursive rules, e.g.

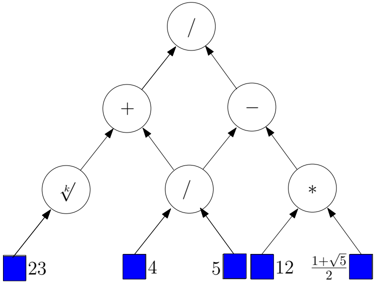
- $|a/b + c/d| \geq 1/|cd|$,
- if $a > 2^{-b}$ then $|\sqrt[k]{a}| > 2^{-bk}$.

Burnikel et al.'99, Li-Yap'00 etc.

Comparing two numbers: $\alpha = \beta$?

- Input $G(\alpha)$ and $G(\beta)$.
- *Constructive Zero Bounds*: G(α), constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.
- Construct zero bound for G(α) $\ominus$ G(β) to get b s.t. $|\alpha - \beta| > 2^{-b}$, if $\alpha \neq \beta$.
- $(b+1)$-bit approximations $\widetilde{\alpha}, \widetilde{\beta} \in \mathbb{Q}$.

The Exact Geometric Computation (EGC) Approach



Recursive rules, e.g.

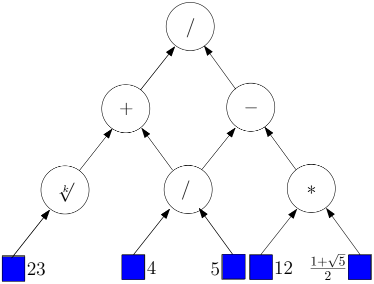
- $|a/b+c/d| \geq 1/|cd|$,
- if $a > 2^{-b}$ then $|\sqrt[k]{a}| > 2^{-bk}$.

Burnikel et al.'99, Li-Yap'00 etc.

Comparing two numbers: $\alpha = \beta$?

- Input $G(\alpha)$ and $G(\beta)$.
- *Constructive Zero Bounds*: G(α), constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.
- Construct zero bound for G(α) $\ominus$ G(β) to get b s.t. $|\alpha - \beta| > 2^{-b}$, if $\alpha \neq \beta$.
- $(b+1)$-bit approximations $\widetilde{\alpha}, \widetilde{\beta} \in \mathbb{Q}$.
- If $|\widetilde{\alpha} - \widetilde{\beta}| < 2^{-b-1}$ then $\alpha = \beta$.

The Exact Geometric Computation (EGC) Approach



Recursive rules, e.g.

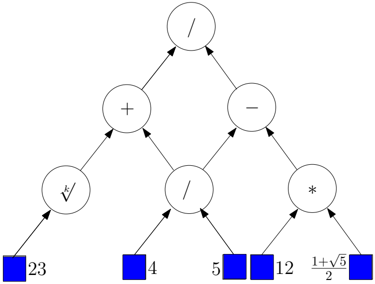
- $|a/b+c/d| \geq 1/|cd|$,
- if $a > 2^{-b}$ then $|\sqrt[k]{a}| > 2^{-bk}$.

Burnikel et al.'99, Li-Yap'00 etc.

Numerical Representation of α

- Data Structure – Rooted DAG, G(α).
  1. Internal nodes – algebraic operations.
  2. Leaves – integers or real algebraic numbers.
- Isolating interval representation:
  1. $A(X) \in \mathbb{Z}[X]$, $A(\alpha) = 0$,
  2. Interval separating α from other roots of $A(X)$.
- Arithmetic is manipulating DAGs.
- Constructive Zero Bounds: G(α), constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.

The Exact Geometric Computation (EGC) Approach



Recursive rules, e.g.

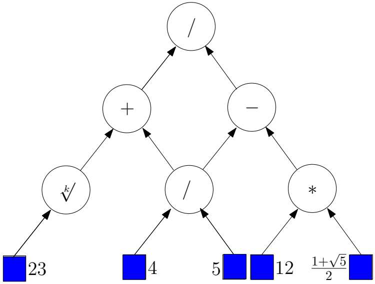
- $|a/b + c/d| \geq 1/|cd|$,
- if $a > 2^{-b}$ then $|\sqrt[k]{a}| > 2^{-bk}$.

Burnikel et al.'99, Li-Yap'00 etc.

Numerical Representation of α

- Data Structure – Rooted DAG, G(α).
  1. Internal nodes – algebraic operations.
  2. Leaves – integers or real algebraic numbers.
- Isolating interval representation:
  1. $A(X) \in \mathbb{Z}[X]$, $A(\alpha) = 0$,
  2. Interval separating α from other roots of $A(X)$.
- Arithmetic is manipulating DAGs.
- Constructive Zero Bounds: G(α), constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.

The Exact Geometric Computation (EGC) Approach



Recursive rules, e.g.

- $|a/b + c/d| \geq 1/|cd|$,
- if $a > 2^{-b}$ then $|\sqrt[k]{a}| > 2^{-bk}$.

Burnikel et al.'99, Li-Yap'00 etc.

Numerical Representation of α

- Data Structure – Rooted DAG, G(α).
  1. Internal nodes – algebraic operations.
  2. Leaves – integers or real algebraic numbers.
- Isolating interval representation:
  1. $A(X) \in \mathbb{Z}[X]$, $A(\alpha) = 0$,
  2. Interval separating α from other roots of $A(X)$.
- Arithmetic is manipulating DAGs.
- Constructive Zero Bounds: G(α), constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.

The Exact Geometric Computation Approach

Implementations

- Leda reals – `http://www.mpi-inf.mpg.de/LEDA/`
- Core Library – `http://www.cs.nyu.edu/exact/core`

Used in CGAL – `www.cgal.org`

Theoretical Foundations of EGC

Ideal World

- Algorithms in Real RAM model.
- Computes a function $f : \mathbb{R} \to \mathbb{R}$.

Theoretical Foundations of EGC

Ideal World

- Algorithms in Real RAM model.
- Computes a function $f : \mathbb{R} \to \mathbb{R}$.

The EGC World, [Yap, 2003]

$\widetilde{f}$ is f implemented in EGC model.

- Input to $\widetilde{f}$ is a dense subset (say $\mathbb{Q}$) of $\mathbb{R}$ and precision p.
- For $x \in \mathbb{Q}$, $\widetilde{f}$ computes a relative approx. to f i.e. $\widetilde{f}(x) = f(x)(1 \pm 2^{-p})$.

What class of functions are EGC-computable?

Theoretical Foundations of EGC

Ideal World

- Algorithms in Real RAM model.
- Computes a function $f : \mathbb{R} \to \mathbb{R}$.

The EGC World, [Yap, 2003]

$\widetilde{f}$ is f implemented in EGC model.

- Input to $\widetilde{f}$ is a dense subset (say $\mathbb{Q}$) of $\mathbb{R}$ and precision p.
- For $x \in \mathbb{Q}$, $\widetilde{f}$ computes a relative approx. to f
  i.e. $\widetilde{f}(x) = f(x)(1 \pm 2^{-p})$.

What class of functions are EGC-computable?

Fundamental Problem

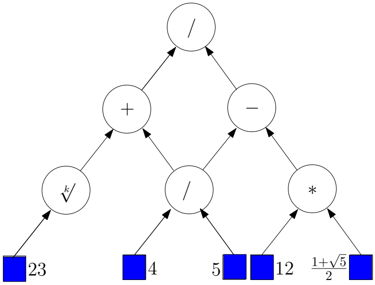
Zero Problem

Are two real numbers α, β equal?

Fundamental Problem

Zero Problem

Are two real numbers α, β equal?



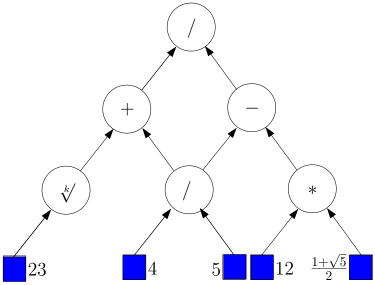
Numerical Representation of α

- Data Structure – Rooted DAG, G(α).
- Isolating interval representation:
  1. $A(X) \in \mathbb{Z}[X]$, $A(\alpha) = 0$,
  2. Interval separating α from other roots of $A(X)$.
- Constructive Zero Bounds: G(α), constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.

Fundamental Problem

Zero Problem

Are two real numbers α, β equal?



Numerical Representation of α

- Data Structure – Rooted DAG, G(α).
- Isolating interval representation:
  1. $A(X) \in \mathbb{Z}[X]$, $A(\alpha) = 0$,
  2. Interval separating α from other roots of $A(X)$.
- Constructive Zero Bounds: G(α), constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.

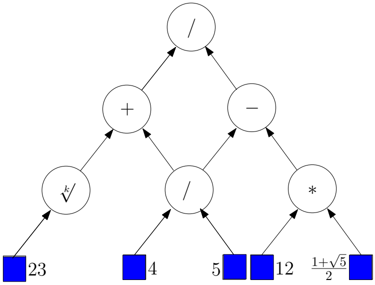
Transcendental Functions

Interior nodes were exp, log, sin, cos etc., or leaves e, π?

Fundamental Problem

Zero Problem

Are two real numbers α, β equal?



Numerical Representation of α

- Data Structure – Rooted DAG, G(α).
- Isolating interval representation:
  1. $A(X) \in \mathbb{Z}[X]$, $A(\alpha) = 0$,
  2. Interval separating α from other roots of $A(X)$.
- Constructive Zero Bounds: G(α), constructs b s.t. $|\alpha| > 2^{-b}$ if $\alpha \neq 0$.

Transcendental Functions

Interior nodes were exp, log, sin, cos etc., or leaves e, π?

Zero Problem for exp, log Expressions, Richardson'07

exp, log Expressions

Interior nodes are $\{+, -, *, /, \sqrt[k]{}\} \cup \{\exp, \log\}$

Zero Problem for exp, log Expressions, Richardson'07

exp, log Expressions

Interior nodes are $\{+, -, *, /, \sqrt[k]{}\} \cup \{\exp, \log\}$

Schanuel's Conjecture

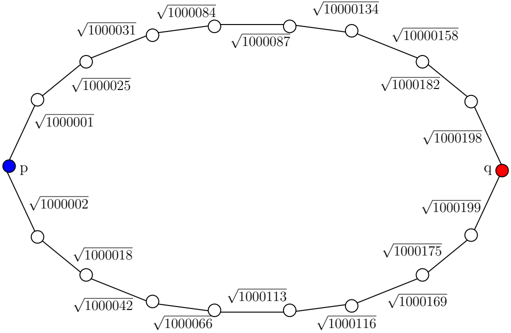
Given n complex numbers $z_1, \ldots, z_n$ linearly independent over $\mathbb{Q}$.
At least n transcendental numbers in $\{z_1, \ldots, z_n, e^{z_1}, \ldots, e^{z_n}\}$.

Generalizes Lindemann-Weierstrass Theorem

$z_1, \ldots, z_n$ are n linearly independent algebraic numbers.

Complexity of Algebraic Numbers

Which algebraic numbers can be relatively approximated in poly time?



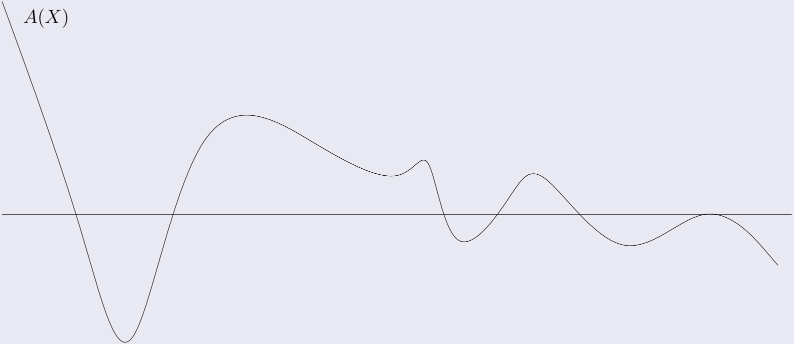
Sum of Square Roots

- $\sum_{i=1}^{k} \sqrt{a_i} - \sum_{i=1}^{k} \sqrt{b_i}$, $|a_i|, |b_i| \leq N$.
- $S(N,k)$, the minimum positive absolute value of the sum.
- Current bounds: $S(N,k) \gtrapprox N^{-2^k}$.
- Hope: $S(N,k) \gtrapprox N^{-k}$.

Real Root Isolation

The Problem

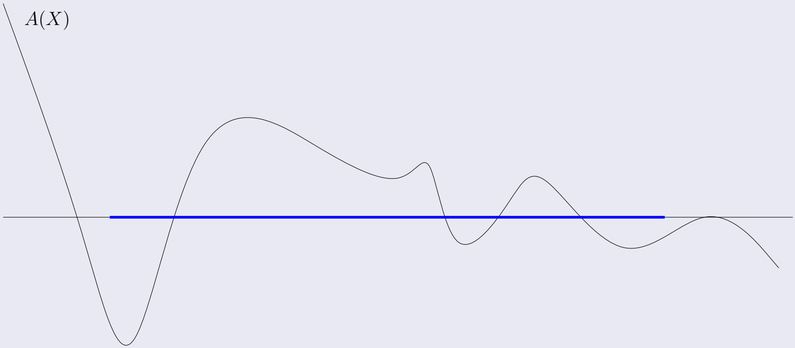
Given $A(X) \in \mathbb{Z}[X]$, degree d.



Real Root Isolation

The Problem

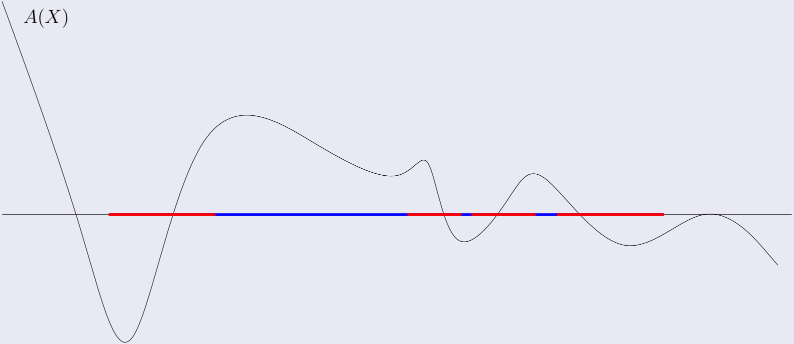
Given $A(X) \in \mathbb{Z}[X]$, degree d.



Real Root Isolation

The Problem

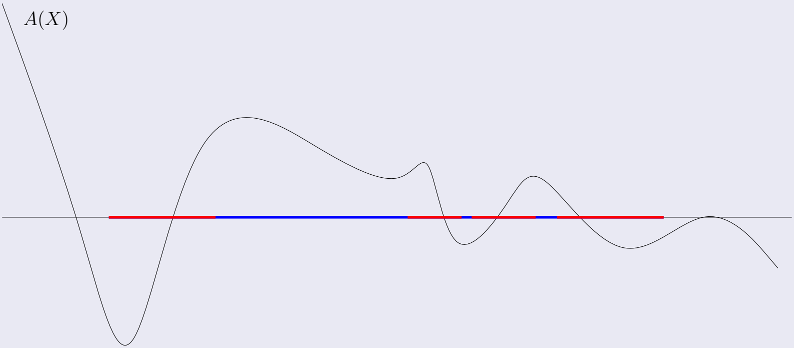
Given $A(X) \in \mathbb{Z}[X]$, degree d.



Real Root Isolation

The Problem

Given $A(X) \in \mathbb{Z}[X]$, degree d.



Assumption

All the roots are of multiplicity one, i.e. $\texttt{GCD}(A(X), A'(X)) = 1$.

Selective History

Classical Work

- Descartes, Newton, Fourier, Sturm, Vincent,
- Weierstrass, Obreshkoff, Ostrowski, Weyl, Henrici, ...

Modern Work – Complexity and Implementation

- Schönhage, Smale, Pan (optimal complexity results), ...
- Collins, Johnson, Krandick, Bini, Mehlhorn, Sagraloff, ...

Relevance

Fundamental problem in computational algebra, used in Cylindrical Algebraic Decomposition, in Ray Tracing, Computer Aided Design, for verifying conjectures, ...

A General Subdivision Algorithm

Input: $A(X) \in \mathbb{Z}[X]$ of degree d, and I_0.
Output: Isolating intervals for roots of $A(X)$ in I_0.

A General Subdivision Algorithm

Input: $A(X) \in \mathbb{Z}[X]$ of degree d, and I_0.
Output: Isolating intervals for roots of $A(X)$ in I_0.

The Real Root Counting function

$\mathrm{Count}(A, I) =$ number of real roots of $A(X)$ in I.

A General Subdivision Algorithm

Input: $A(X) \in \mathbb{Z}[X]$ of degree d, and I_0.
Output: Isolating intervals for roots of $A(X)$ in I_0.

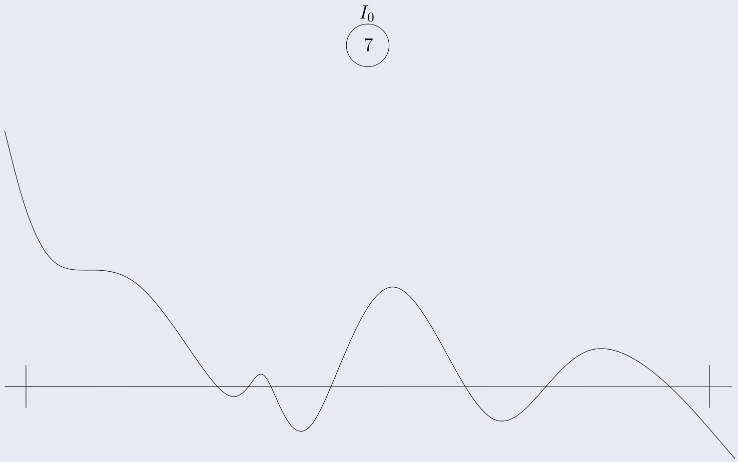
The Real Root Counting function

$\text{Count}(A, I) =$ number of real roots of $A(X)$ in I.

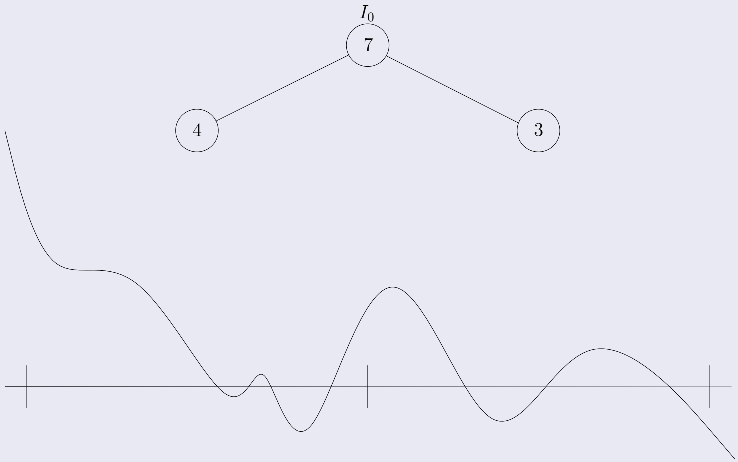
RootIsol($A(X), I$)

1. If $\text{Count}(A, I) = 0$ then return.
2. If $\text{Count}(A, I) = 1$ then output I and return.
3. Let m be the midpoint of $I = (I_l, I_r)$.
4. Recurse on $(A(X), (I_l, m))$ and $(A(X), (m, I_r))$.

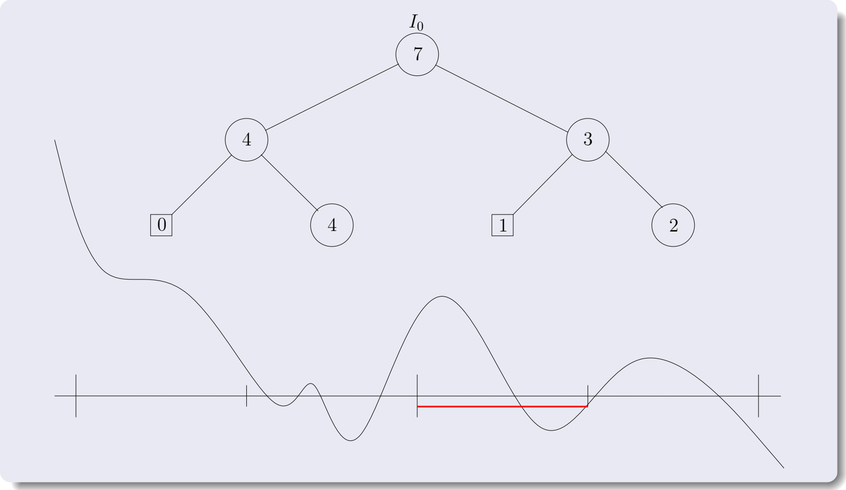
Illustration



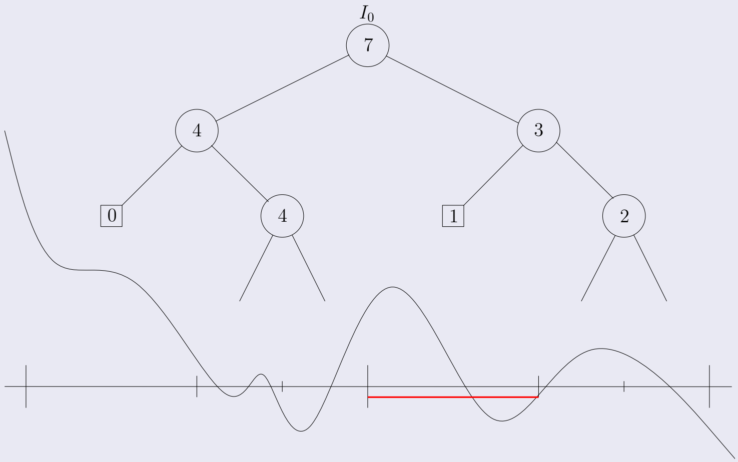
Illustration



Illustration



Illustration



Sturm Sequences

The Sequence

- $\overline{A} := (A_0 = A, A_1 = A', \ldots, A_k)$, $A_{i+1} := \quad \texttt{rem}(A_{i-1}, A_i)$.

Sturm Sequences

The Sequence

- $\overline{A} := (A_0 = A, A_1 = A', \ldots, A_k)$, $A_{i+1} := -\mathtt{rem}(A_{i-1}, A_i)$.

Sturm Sequences

The Sequence

- $\overline{A} := (A_0 = A, A_1 = A', \ldots, A_k)$, $A_{i+1} := -\mathtt{rem}(A_{i-1}, A_i)$.
- $A(x) = (x^2 - x - 1)$, $A'(x) = 2x - 1$
- $x^2 - x - 1 = (2x-1)\frac{(2x-1)}{4} - \frac{5}{4} = \frac{4x^2-4x+1}{4} - \frac{5}{4}$.
- $\overline{A} = (x^2 - x - 1, 2x - 1, -\frac{5}{4})$.

Sturm Sequences

The Sequence

- $\overline{A} := (A_0 = A, A_1 = A', \ldots, A_k)$, $A_{i+1} := -\mathtt{rem}(A_{i-1}, A_i)$.
- $A(x) = (x^2 - x - 1)$, $A'(x) = 2x - 1$
- $x^2 - x - 1 = (2x - 1)\frac{(2x-1)}{4} - \frac{5}{4} = \frac{4x^2-4x+1}{4} - \frac{5}{4}$.
- $\overline{A} = (x^2 - x - 1, 2x - 1, +\frac{5}{4})$.

Sturm Sequences

The Sequence

- $\overline{A} := (A_0 = A, A_1 = A', \ldots, A_k)$, $A_{i+1} := -\mathtt{rem}(A_{i-1}, A_i)$.
- $A(x) = (x^2 - x - 1)$, $A'(x) = 2x - 1$
- $x^2 - x - 1 = (2x - 1)\frac{(2x-1)}{4} - \frac{5}{4} = \frac{4x^2-4x+1}{4} - \frac{5}{4}$.
- $\overline{A} = (x^2 - x - 1, 2x - 1, +\frac{5}{4})$.

The Variation

- Given $c \in \mathbb{R}$, evaluate $\overline{A}$ at c, i.e., $(A_0(c), A_1(c), \ldots, A_k(c))$.
- Drop all the zeros from the sequence $(A_0(c), A_1(c), \ldots, A_k(c))$.
- $\mathrm{Var}(\overline{A}; c) :=$ no. of sign flips from $+$ to $-$ or vice versa.

Sturm Sequences

The Sequence

- $\overline{A} := (A_0 = A, A_1 = A', \ldots, A_k)$, $A_{i+1} := -\mathtt{rem}(A_{i-1}, A_i)$.
- $A(x) = (x^2 - x - 1)$, $A'(x) = 2x - 1$
- $x^2 - x - 1 = (2x-1)\frac{(2x-1)}{4} - \frac{5}{4} = \frac{4x^2-4x+1}{4} - \frac{5}{4}$.
- $\overline{A} = (x^2 - x - 1, 2x - 1, +\frac{5}{4})$.
- $\mathrm{Var}(\overline{A}; 1) = \#(-,+,+) = 1$, $\mathrm{Var}(\overline{A}; 2) = \#(+,+,+) = 0$.

The Variation

- Given $c \in \mathbb{R}$, evaluate $\overline{A}$ at c, i.e., $(A_0(c), A_1(c), \ldots, A_k(c))$.
- Drop all the zeros from the sequence $(A_0(c), A_1(c), \ldots, A_k(c))$.
- $\mathrm{Var}(\overline{A}; c) :=$ no. of sign flips from $+$ to $-$ or vice versa.

Sturm Sequences

The Sequence

- $\overline{A} := (A_0 = A, A_1 = A', \dots, A_k)$, $A_{i+1} := -\mathtt{rem}(A_{i-1}, A_i)$.
- $A(x) = (x^2 - x - 1)$, $A'(x) = 2x - 1$
- $x^2 - x - 1 = (2x-1)\frac{(2x-1)}{4} - \frac{5}{4} = \frac{4x^2-4x+1}{4} - \frac{5}{4}$.
- $\overline{A} = (x^2 - x - 1, 2x - 1, +\frac{5}{4})$.
- $\mathrm{Var}(\overline{A}; 1) = \#(-,+,+) = 1$, $\mathrm{Var}(\overline{A}; 2) = \#(+,+,+) = 0$.

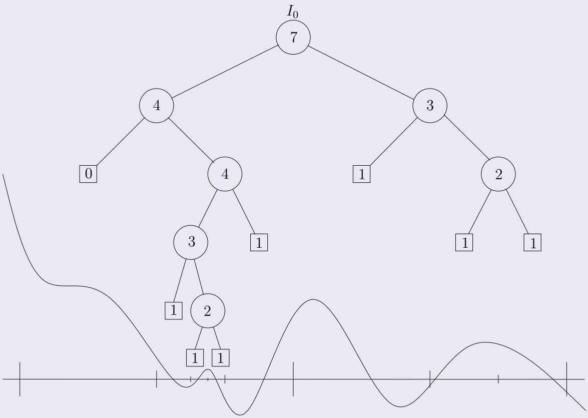
The Variation

- Given $c \in \mathbb{R}$, evaluate $\overline{A}$ at c, i.e., $(A_0(c), A_1(c), \dots, A_k(c))$.
- Drop all the zeros from the sequence $(A_0(c), A_1(c), \dots, A_k(c))$.
- $\mathrm{Var}(\overline{A}; c) :=$ no. of sign flips from $+$ to $-$ or vice versa.

Sturm's Theorem, 1829

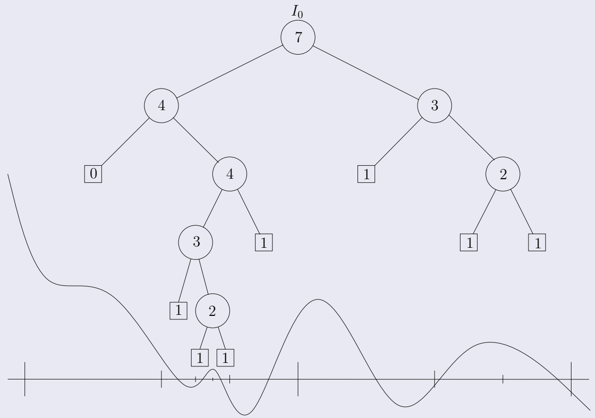
No. of real roots of $A(x)$ in (c, d) = $\mathrm{Var}(\overline{A}; c) - \mathrm{Var}(\overline{A}; d)$.

Complexity Analysis – Sturm's Algorithm, Davenport'85



1. Size of the subdivision tree, $|T|$.
2. Worst case complexity at every node.

Complexity Analysis – Sturm's Algorithm, Davenport'85



1. Size of the subdivision tree, $|T|$.
2. Worst case complexity at every node.

Complexity Analysis – Sturm's Algorithm

Measure of Complexity

Root Separation of A, $\mathrm{sep}(A) := \min\{|\alpha - \beta| : \alpha, \beta \in Z(A) \subseteq \mathbb{C}\}$.

Complexity Analysis – Sturm's Algorithm

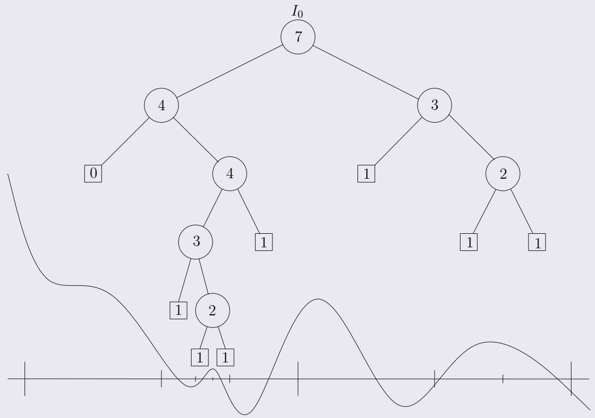
Measure of Complexity

Root Separation of A, $\mathrm{sep}(A) := \min\{|\alpha - \beta| : \alpha, \beta \in Z(A) \subseteq \mathbb{C}\}$.

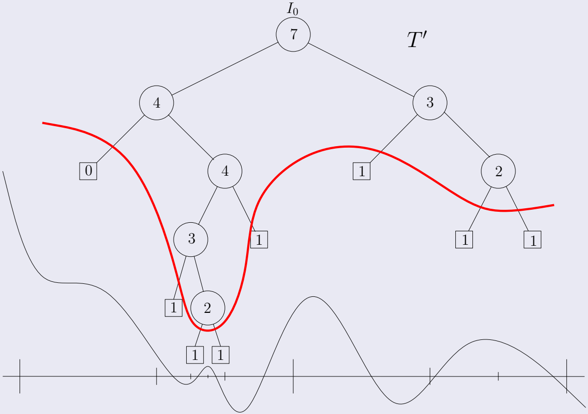
Bounds

- $A(x) = \sum_{i=0}^{d} a_i x^i \in \mathbb{Z}[x]$, degree d, $|a_i| \le 2^L$, $i = 0, \ldots, d$.
- $-\log \mathrm{sep}(A) = O(dL + d \log d)$.
- $w(I_0) < 2^L$.

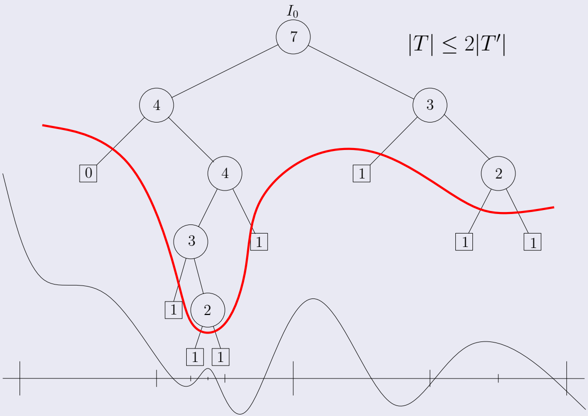
Complexity Analysis – Sturm's Algorithm



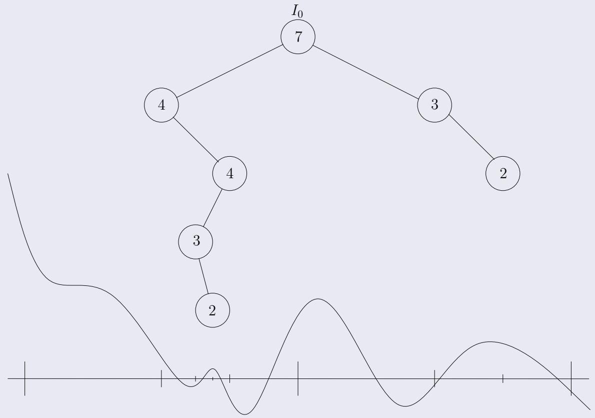
Complexity Analysis – Sturm's Algorithm



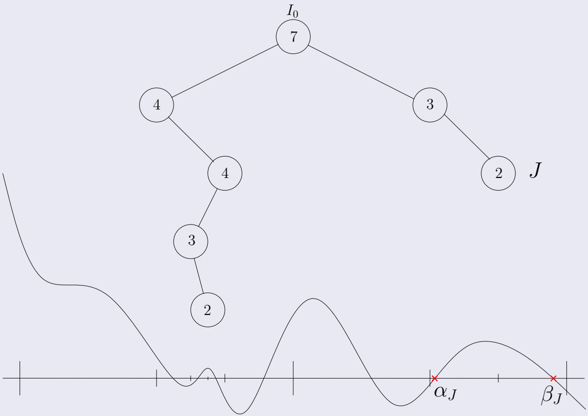
Complexity Analysis – Sturm's Algorithm



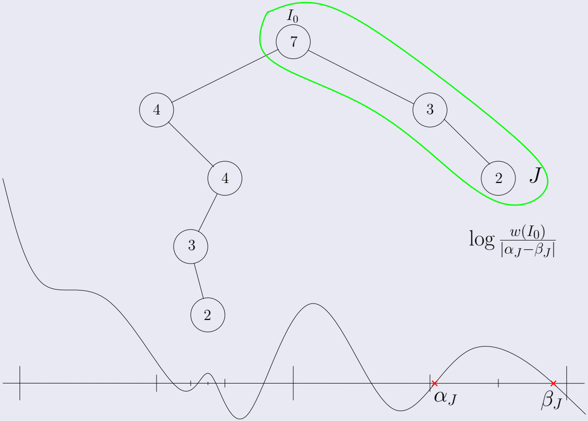
Complexity Analysis – Sturm's Algorithm



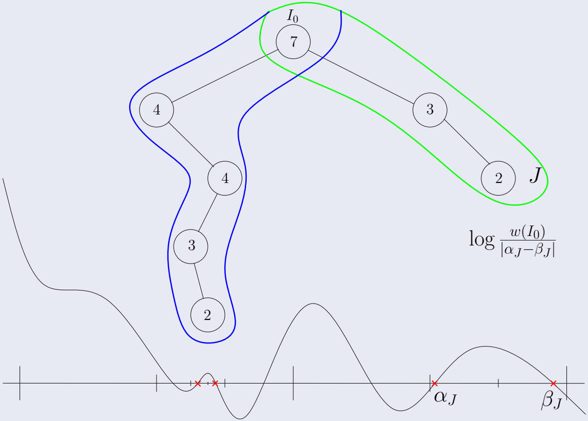
Complexity Analysis – Sturm's Algorithm



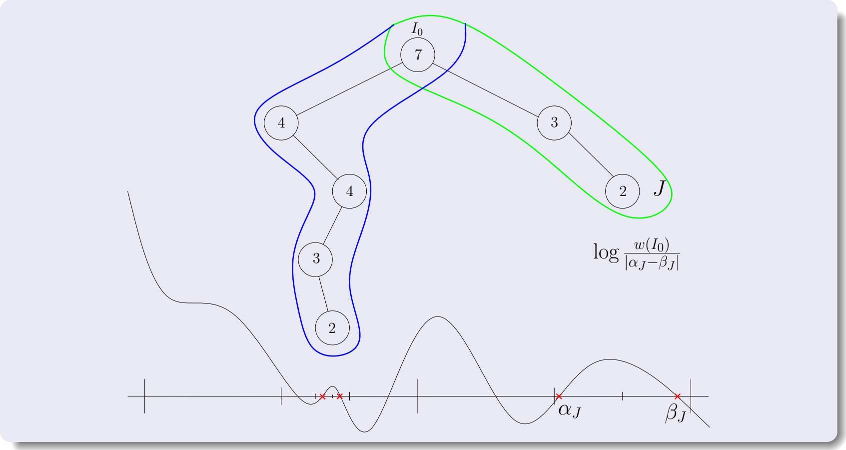
Complexity Analysis – Sturm's Algorithm



Complexity Analysis – Sturm's Algorithm

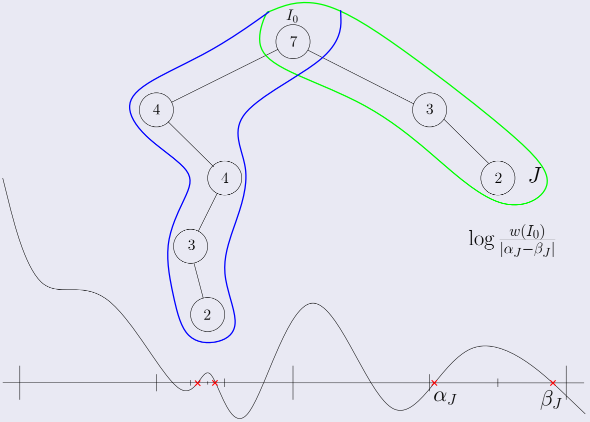


Complexity Analysis – Sturm's Algorithm



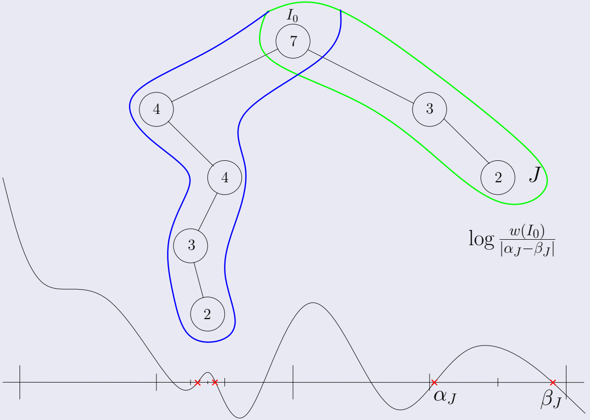
- $|T'| \leq \sum_J \log \frac{w(I_0)}{|\alpha_J - \beta_J|}$

Complexity Analysis – Sturm's Algorithm



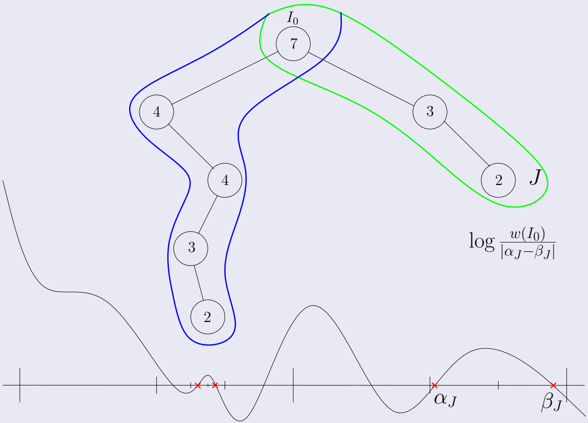
- $|T'| \leq \sum_J \log \frac{w(I_0)}{|\alpha_J - \beta_J|} = O(d \log w(I_0)) - \sum_J \log |\alpha_J - \beta_J|$

Complexity Analysis – Sturm's Algorithm



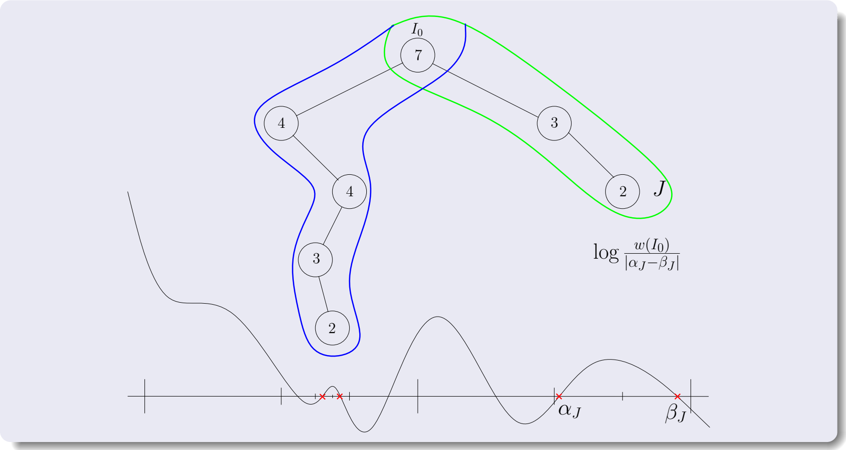
- $|T'| \leq \sum_J \log \frac{w(I_0)}{|\alpha_J - \beta_J|} = O(d \log w(I_0)) - \sum_J \log|\alpha_J - \beta_J|$
- $-\sum_J \log|\alpha_J - \beta_J|$

Complexity Analysis – Sturm's Algorithm



- $|T'| \le \sum_J \log \frac{w(I_0)}{|\alpha_J - \beta_J|} = O(d \log w(I_0)) - \sum_J \log|\alpha_J - \beta_J|$
- $-\sum_J \log|\alpha_J - \beta_J| = O(-d \log \operatorname{sep}(A)) = O(d(dL + d \log d))$

Complexity Analysis – Sturm's Algorithm



- $|T'| \leq \sum_J \log \frac{w(I_0)}{|\alpha_J - \beta_J|} = O(d \log w(I_0)) - \sum_J \log|\alpha_J - \beta_J|$
- $-\sum_J \log|\alpha_J - \beta_J| = O(-d \log \mathrm{sep}(A)) = O(d(dL + d\log d))$
- $-\sum_J \log|\alpha_J - \beta_J| = O(dL + d\log d)$, Davenport-Mahler.

Some Remarks on Sturm's Algorithm

- The subdivision tree size is optimal (Mignotte's polynomials).

Some Remarks on Sturm's Algorithm

- The subdivision tree size is optimal (Mignotte's polynomials).
- (Almost) Never used in practice nowadays.
- Prefer weaker estimates.
  - Estimate$(A; I) \geq$ number of real roots of A in I.
  - If Estimate$(A; I) \leq 1$ then exact number.

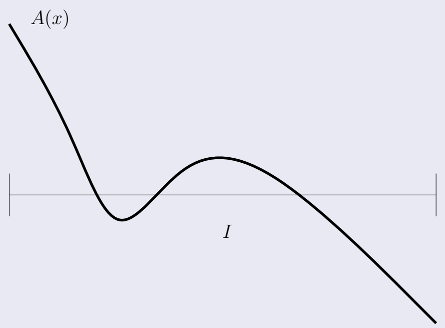
  E.g., Descartes's rule of signs.

Some Remarks on Sturm's Algorithm

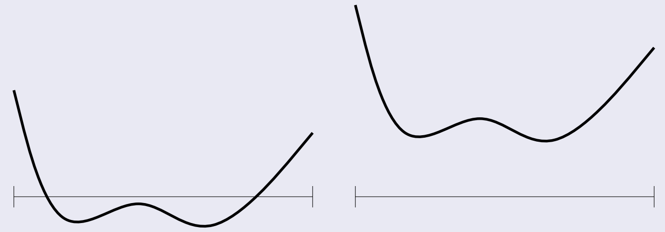
- The subdivision tree size is optimal (Mignotte's polynomials).
- (Almost) Never used in practice nowadays.
- Prefer weaker estimates.
    - Estimate$(A; I) \geq$ number of real roots of A in I.
    - If Estimate$(A; I) \leq 1$ then exact number.

  E.g., Descartes's rule of signs.
- It's not the first algorithm that comes to mind.

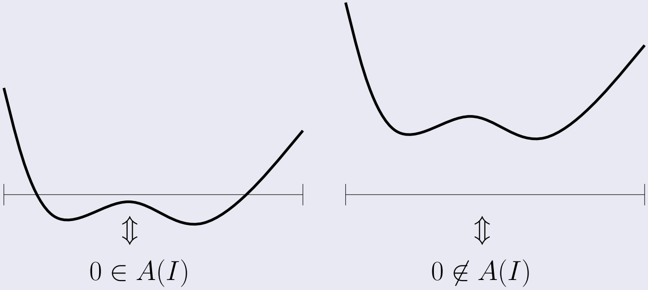
An Idea For Real Root Isolation



An Idea For Real Root Isolation

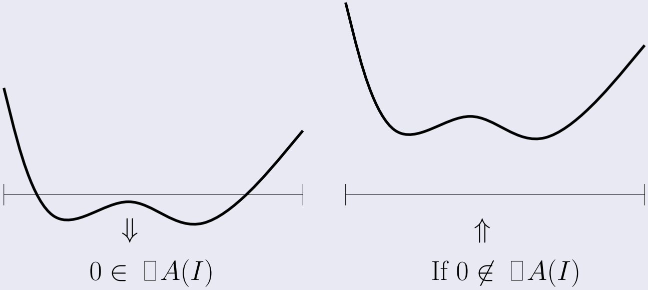


An Idea For Real Root Isolation



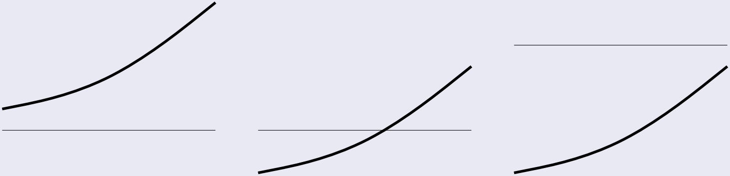
- $A(I) \subset \mathbb{R}$ range of values $A(x)$ takes on I.

An Idea For Real Root Isolation



- $A(I) \subset \mathbb{R}$ range of values $A(x)$ takes on I.
- Box-function: Given I, compute $\Box A(I)$ s.t. $A(I) \subseteq \Box A(I)$.

An Idea For Real Root Isolation



$\Uparrow$

If $0 \notin \square A'(I)$

- $A(I) \subset \mathbb{R}$ range of values $A(x)$ takes on I.
- Box-function: Given I, compute $\square A(I)$ s.t. $A(I) \subseteq \square A(I)$.

The Algorithm, Mitchell'90

Input: $A(X) \in \mathbb{R}[X]$ of degree d, and I_0.
Output: Isolating intervals for roots of $A(X)$ in I_0.

EVAL Algorithm

1. Initialize a queue $Q \leftarrow \{I_0\}$.
2. While Q is not empty do
3. Remove an interval I from Q.
4. If $0 \notin \square A(I)$ or $0 \notin \square A'(I)$ then stop.
5. Else
   Subdivide I into two halves and push them on Q.

Assumption

$A(x)$ is square-free, no multiple roots.

Implementing the Box-function

Let $A(x) = \sum_{i=0}^{n} a_i x^i$, $a_i \in \mathbb{R}$.

Interval Arithmetic

- $[a,b] + [c,d] := [a+c, b+d]$.
- $[a,b] * [c,d] := [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}]$.

Implementing the Box-function

Let $A(x) = \sum_{i=0}^{n} a_i x^i$, $a_i \in \mathbb{R}$.

Interval Arithmetic

- $[a,b] + [c,d] := [a+c, b+d]$.
- $[a,b] * [c,d] := [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}]$.

Implementing $\Box A(I)$

- Compute $\sum_{k=0}^{n} a_k I^k$.

Implementing the Box-function

Let $A(x) = \sum_{i=0}^{n} a_i x^i$, $a_i \in \mathbb{R}$.

Interval Arithmetic

- $[a, b] + [c, d] := [a + c, b + d]$.
- $[a, b] * [c, d] := [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}]$.

Implementing $\Box A(I)$

- Compute $\sum_{k=0}^{n} a_k I^k$.
- Horner's evaluation: $((a_n I + a_{n-1}) * I + \ldots + a_1) * I + a_0$.

Implementing the Box-function

Let $A(x) = \sum_{i=0}^{n} a_i x^i$, $a_i \in \mathbb{R}$.

Interval Arithmetic

- $[a,b] + [c,d] := [a+c, b+d]$.
- $[a,b] * [c,d] := [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}]$.

Implementing $\Box A(I)$

- Compute $\sum_{k=0}^{n} a_k I^k$.
- Horner's evaluation: $((a_n I + a_{n-1}) * I + \ldots + a_1) * I + a_0$.
- Centered Form: $\Box A(I) := \left[A(m(I)) \pm \sum_{k>0} \frac{|A^{(k)}(m)|}{k!} \left(\frac{w(I)}{2} \right)^k \right]$.

Implementing the Box-function

Let $A(x) = \sum_{i=0}^{n} a_i x^i$, $a_i \in \mathbb{R}$.

Interval Arithmetic

- $[a,b] + [c,d] := [a+c, b+d]$.
- $[a,b] * [c,d] := [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}]$.

Implementing $\square A(I)$

- Compute $\sum_{k=0}^{n} a_k I^k$.
- Horner's evaluation: $((a_n I + a_{n-1}) * I + \ldots + a_1) * I + a_0$.
- Centered Form: $\square A(I) := \left[A(m(I)) \pm \sum_{k>0} \frac{|A^{(k)}(m)|}{k!} \left(\frac{w(I)}{2} \right)^k \right]$.

Two Properties of Box-functions

- Conservative: $A(I) \subseteq \square A(I)$.
- Convergent: $I_1 \supset I_2 \supset I_3 \supset \cdots \supset \{x\}$ then $\square A(I_j) \to A(x)$.

EVAL: Bounds on Recursion Tree Size

Goal – Real Root Isolation

$A \in \mathbb{Z}[x]$ square-free, degree d, with L-bit coefficients.

$$\text{Aim: } O(d(L + \log d))$$

Similar bounds for Sturm's method.

EVAL: Bounds on Recursion Tree Size

Goal – Real Root Isolation

$A \in \mathbb{Z}[x]$ square-free, degree d, with L-bit coefficients.

$$\text{Aim: } O(d(L + \log d))$$

Similar bounds for Sturm's method.

Result

$O(d(L + r))$, where r is the number of real roots in input interval.

The EVAL Algorithm

Input: $A(X) \in \mathbb{R}[X]$ of degree d, and I_0.
Output: Isolating intervals for roots of $A(X)$ in I_0.

EVAL Algorithm: EVAL(A, I_0)

1. Initialize a queue $Q \leftarrow \{I_0\}$.
2. While Q is not empty do
3.   Remove an interval I from Q.
4.   If $0 \notin \square A(I)$ or $0 \notin \square A'(I)$ then stop.
5.   Else
        Subdivide I into two halves and push them on Q.

The EVAL Algorithm

Input: $A(X) \in \mathbb{R}[X]$ of degree d, and I_0.
Output: Isolating intervals for roots of $A(X)$ in I_0.

EVAL Algorithm: EVAL(A, I_0)

1. Initialize a queue $Q \leftarrow \{I_0\}$.
2. While Q is not empty do
3. Remove an interval I from Q.
4. If $0 \notin \square A(I)$ or $0 \notin \square A'(I)$ then stop.
5. Else
   Subdivide I into two halves and push them on Q.

Definition

$P(I_0)$ – partition of I_0 at the leaves of the subdivision tree EVAL(A, I_0).

EVAL: An Integral Bound on Tree Size

Stopping Function, Burr-Krahmer-Yap'09

- $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$.
- If $\exists x \in I$ such that $w(I)G(x) \leq 1$ then I is terminal.
- Terminal means either $0 \notin \square A(I)$ or $0 \notin \square A(I)$.

EVAL: An Integral Bound on Tree Size

Stopping Function, Burr-Krahmer-Yap'09

- $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$.
- If $\exists x \in I$ such that $w(I)G(x) \leq 1$ then I is terminal.
- Terminal means either $0 \notin \square A(I)$ or $0 \notin \square A(I)$.

Lemma

$$\#P(I_0) \leq 2 \int_{I_0} G(x)dx.$$

EVAL: An Integral Bound on Tree Size

Stopping Function, Burr-Krahmer-Yap'09

- $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$.
- If $\exists x \in I$ such that $w(I)G(x) \leq 1$ then I is terminal.
- Terminal means either $0 \notin \square A(I)$ or $0 \notin \square A(I)$.

Lemma

$$\#P(I_0) \leq 2 \int_{I_0} G(x) dx.$$

Proof

- Let $I \in P(I_0)$ and J be the interval associated with its parent.

EVAL: An Integral Bound on Tree Size

Stopping Function, Burr-Krahmer-Yap'09

- $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$.
- If $\exists x \in I$ such that $w(I)G(x) \leq 1$ then I is terminal.
- Terminal means either $0 \notin \square A(I)$ or $0 \notin \square A(I)$.

Lemma

$$\#P(I_0) \leq 2 \int_{I_0} G(x)dx.$$

Proof

- Let $I \in P(I_0)$ and J be the interval associated with its parent.
- $\forall x \in J,\ w(J)G(x) > 1$.

EVAL: An Integral Bound on Tree Size

Stopping Function, Burr-Krahmer-Yap'09

- $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$.
- If $\exists x \in I$ such that $w(I)G(x) \leq 1$ then I is terminal.
- Terminal means either $0 \notin \square A(I)$ or $0 \notin \square A(I)$.

Lemma

$$\#P(I_0) \leq 2 \int_{I_0} G(x)dx.$$

Proof

- Let $I \in P(I_0)$ and J be the interval associated with its parent.
- $\forall x \in J$, $w(J)G(x) > 1$.
- $\forall x \in I$, $2w(I)G(x) > 1$ (since $I \subseteq J$).

EVAL: An Integral Bound on Tree Size

Stopping Function, Burr-Krahmer-Yap'09

- $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$.
- If $\exists x \in I$ such that $w(I)G(x) \leq 1$ then I is terminal.
- Terminal means either $0 \notin \square A(I)$ or $0 \notin \square A(I)$.

Lemma

$$\#P(I_0) \leq 2 \int_{I_0} G(x)dx.$$

Proof

- Let $I \in P(I_0)$ and J be the interval associated with its parent.
- $\forall x \in J$, $w(J)G(x) > 1$.
- $\forall x \in I$, $2w(I)G(x) > 1$ (since $I \subseteq J$).
- $2 \int_{I_0} G(x)dx = \sum_{I \in P(I_0)} 2 \int_I G(x)dx \geq \sum_{I \in P(I_0)} 1 = \#P(I_0)$.

EVAL: What choice of Stopping Function?

Stopping Function $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$

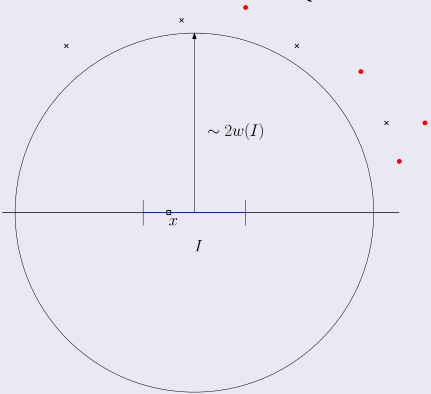
If $\exists x \in I$ such that $w(I)G(x) \leq 1$ then I is terminal.

EVAL: What choice of Stopping Function?

Stopping Function $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$

If $\exists x \in I$ such that $w(I)G(x) \leq 1$ then I is terminal.

Intuitively, we expect $G(x) \sim 2\min\left\{\frac{1}{|x-Z(A)|}, \frac{1}{|x-Z(A')|}\right\}$.



EVAL: What choice of Stopping Function?

Stopping Function $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$

If $\exists x \in I$ such that $w(I)G(x) \leq 1$ then I is terminal.

The Stopping Function (Burr-Krahmer, 2012)

- $S(x) := \sum_{\alpha \in Z(A)} \frac{1}{|x-\alpha|}$.
- $T(x) := \sum_{\alpha' \in Z(A')} \frac{1}{|x-\alpha'|}$.

$$G(x) := \min\{S(x), T(x)\}.$$

EVAL: What choice of Stopping Function?

Stopping Function $G:\mathbb{R}\to\mathbb{R}_{\geq 0}$

If $\exists x\in I$ such that $w(I)G(x)\leq 1$ then I is terminal.

The Stopping Function (Burr-Krahmer, 2012)

- $S(x) := \sum_{\alpha\in Z(A)} \frac{1}{|x-\alpha|}$.
- $T(x) := \sum_{\alpha'\in Z(A')} \frac{1}{|x-\alpha'|}$.

$$G(x) := \min\{S(x), T(x)\}.$$

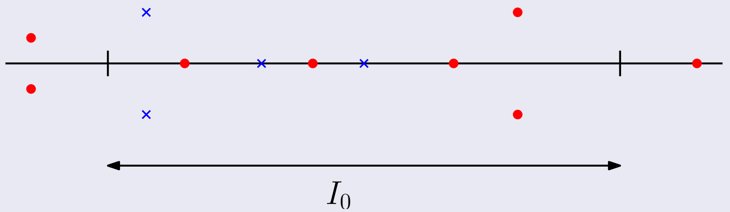
Key Property

- $\left|\frac{A'(x)}{A(x)}\right| \leq S(x), \left|\frac{A''(x)}{A(x)}\right| \leq S^2(x), \ldots, \left|\frac{A^{(k)}(x)}{A(x)}\right| \leq S^k(x), \ldots.$
- $\sum_{k>0}\left|\frac{A^{(k)}(x)}{k!A(x)}\right|\left(\frac{w(I)}{2}\right)^k \leq \sum_{k>0}\frac{1}{k!}\left(\frac{S(x)w(I)}{2}\right)^k < 1.$

EVAL: Bounding the Integral, S.-Yap'12

$$S(x) := \sum_{\alpha \in Z(A)} \frac{1}{|x-\alpha|} \qquad T(x) := \sum_{\alpha' \in Z(A')} \frac{1}{|x-\alpha'|}.$$

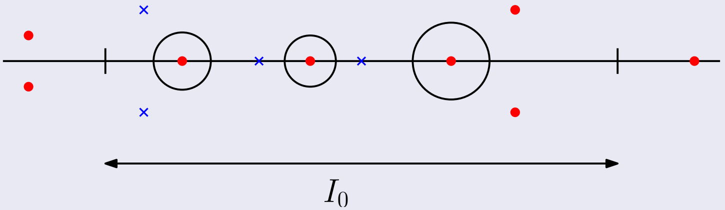
$$\#P(I_0) \leq \int_{I_0} \min\{S(x), T(x)\}\, dx$$



EVAL: Bounding the Integral, S.-Yap'12

$$S(x) := \sum_{\alpha \in Z(A)} \frac{1}{|x-\alpha|} \qquad T(x) := \sum_{\alpha' \in Z(A')} \frac{1}{|x-\alpha'|}.$$

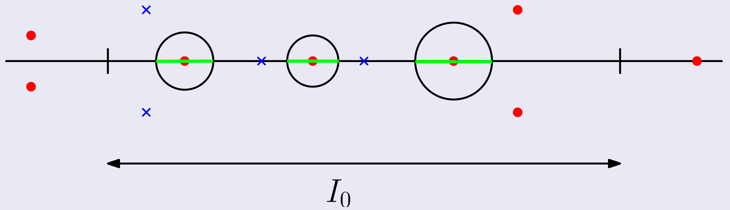
$$\#P(I_0) \leq \int_{I_0} \min\{S(x), T(x)\}\, dx$$



EVAL: Bounding the Integral, S.-Yap'12

$$S(x) := \sum_{\alpha \in Z(A)} \frac{1}{|x-\alpha|} \qquad T(x) := \sum_{\alpha' \in Z(A')} \frac{1}{|x-\alpha'|}.$$

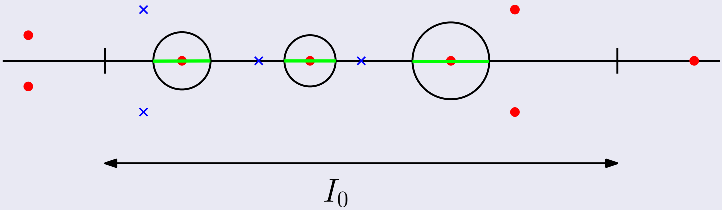
$$\#P(I_0) \leq \int_{I_0} \min\{S(x), T(x)\}\, dx$$



EVAL: Bounding the Integral, S.-Yap'12

$$S(x) := \sum_{\alpha \in Z(A)} \frac{1}{|x-\alpha|} \qquad T(x) := \sum_{\alpha' \in Z(A')} \frac{1}{|x-\alpha'|}.$$

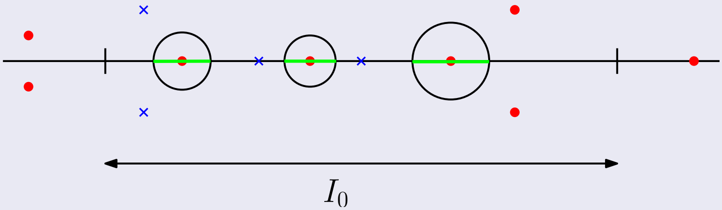
$$\#P(I_0) \leq \int_{I_0} \min\{S(x), T(x)\}\, dx \leq \int_{I_1} T(x)dx + \int_{I_0 \setminus I_1} S(x)dx$$



EVAL: Bounding the Integral, S.-Yap'12

$$S(x) := \sum_{\alpha \in Z(A)} \frac{1}{|x-\alpha|} \qquad T(x) := \sum_{\alpha' \in Z(A')} \frac{1}{|x-\alpha'|}.$$

$$\#P(I_0) \leq \int_{I_0} \min\{S(x), T(x)\}\, dx \leq \int_{I_1} T(x)dx + \int_{I_0 \setminus I_1} S(x)dx$$



$$\int_{I_1} T(x)dx = O(dr) \text{ and } \int_{I_0 \setminus I_1} S(x)dx = O(d(L + \log d)).$$

Real Root Isolation – Beyond Polynomials

Remarks on EVAL

- For differentiable functions f as long as $\square f(I), \square f'(I)$.

Real Root Isolation – Beyond Polynomials

Remarks on EVAL

- For differentiable functions f as long as $\square f(I), \square f'(I)$.
- Assume f has no multiple roots.

Real Root Isolation – Beyond Polynomials

Remarks on EVAL

- For differentiable functions f as long as $\square f(I), \square f'(I)$.
- Assume f has no multiple roots.
- To handle multiplicity we have to go to $\mathbb{C}$.

Real Root Isolation – Beyond Polynomials

Remarks on EVAL

- For differentiable functions f as long as $\square f(I), \square f'(I)$.
- Assume f has no multiple roots.
- To handle multiplicity we have to go to $\mathbb{C}$.

Root Clustering of Holomorphic Functions, Sagraloff-S.-Yap'13

- Pellet's Theorem: Disc $D(m,r) \subseteq \mathbb{C}$ and $|f_k(m)r^k| > \sum_{j \neq k} |f_j(m)| r^j$ then $D(m,r)$ contains k roots.
- Darboux's theorem.
- Soft-predicates: Given $\varepsilon \geq 0$, if $x \leq \varepsilon$ then treat x as 0.

The Subdivision Paradigm

In Other Contexts

- Root Isolation of Zero Dimensional Systems [Moore (1966), Kearfott (1987), Stahl (1995)].
- Isotopic meshing of curves and surfaces [Snyder (1992), Plantinga-Vegter (2004), Lin-Yap (2011)].
- Marching cube algorithms [Lorensen-Cline (1987)].
- Robot Motion Planning [Brooks and Lozano-Perez (1983), Zhu-Latombe (1991)].
- Voronoi Diagrams of Polytopes, Polyhedron. [Vleugel-Overmars (1995), Li-S.-Yap (2012)].

Thank You!